In [1]:
```
# importing necessary libraries
import pandas as pd
import plotly.express as px
import plotly.graph_objects as go
```

In [2]:
```
# Loading the dataset
data = pd.read_csv('Sample - Superstore.csv', encoding='latin-1')
data.head()
```

Out[2]:

	Row ID	Order ID	Order Date	Ship Date	Ship Mode	Customer ID	Customer Name	Segment	Country	City	...	Postal Code
0	1	CA-2016-152156	11/8/2016	11/11/2016	Second Class	CG-12520	Claire Gute	Consumer	United States	Henderson	...	42420
1	2	CA-2016-152156	11/8/2016	11/11/2016	Second Class	CG-12520	Claire Gute	Consumer	United States	Henderson	...	42420
2	3	CA-2016-138688	6/12/2016	6/16/2016	Second Class	DV-13045	Darrin Van Huff	Corporate	United States	Los Angeles	...	90036
3	4	US-2015-108966	10/11/2015	10/18/2015	Standard Class	SO-20335	Sean O'Donnell	Consumer	United States	Fort Lauderdale	...	33311
4	5	US-2015-108966	10/11/2015	10/18/2015	Standard Class	SO-20335	Sean O'Donnell	Consumer	United States	Fort Lauderdale	...	33311

5 rows × 21 columns

In [3]:
```
data.tail()
```

Out[3]:

	Row ID	Order ID	Order Date	Ship Date	Ship Mode	Customer ID	Customer Name	Segment	Country	City	...	P
9989	9990	CA-2014-110422	1/21/2014	1/23/2014	Second Class	TB-21400	Tom Boeckenhauer	Consumer	United States	Miami	...	3
9990	9991	CA-2017-121258	2/26/2017	3/3/2017	Standard Class	DB-13060	Dave Brooks	Consumer	United States	Costa Mesa	...	9
9991	9992	CA-2017-121258	2/26/2017	3/3/2017	Standard Class	DB-13060	Dave Brooks	Consumer	United States	Costa Mesa	...	9
9992	9993	CA-2017-121258	2/26/2017	3/3/2017	Standard Class	DB-13060	Dave Brooks	Consumer	United States	Costa Mesa	...	9
9993	9994	CA-2017-119914	5/4/2017	5/9/2017	Second Class	CC-12220	Chris Cortes	Consumer	United States	Westminster	...	9

5 rows × 21 columns

In [4]:
```
data.describe()
```

Out[4]:

	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 [5]:
```
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
```

In [6]:
```
data.isnull().sum()
```

Out[6]:
```
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
```

In [7]:
```
data.duplicated().sum()
```

Out[7]:
```
0
```

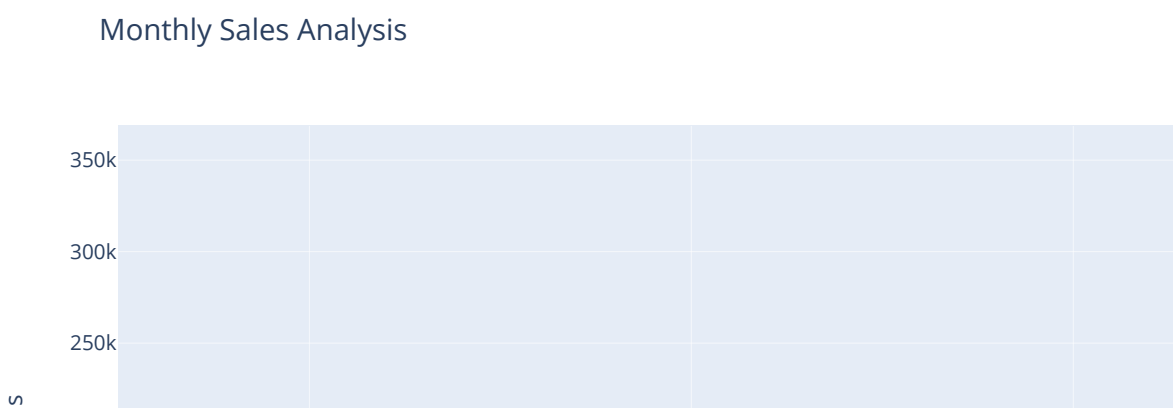
In [8]:
```
# Converting Dtype of 'Order Date' and 'Ship Date' from object to datetime
data['Order Date'] = pd.to_datetime(data['Order Date'])
data['Ship Date'] = pd.to_datetime(data['Ship Date'])
```

We can use order date column to create new columns like order month, order year, and order day, which will be very valuable for sales and profit analysis according to time periods.

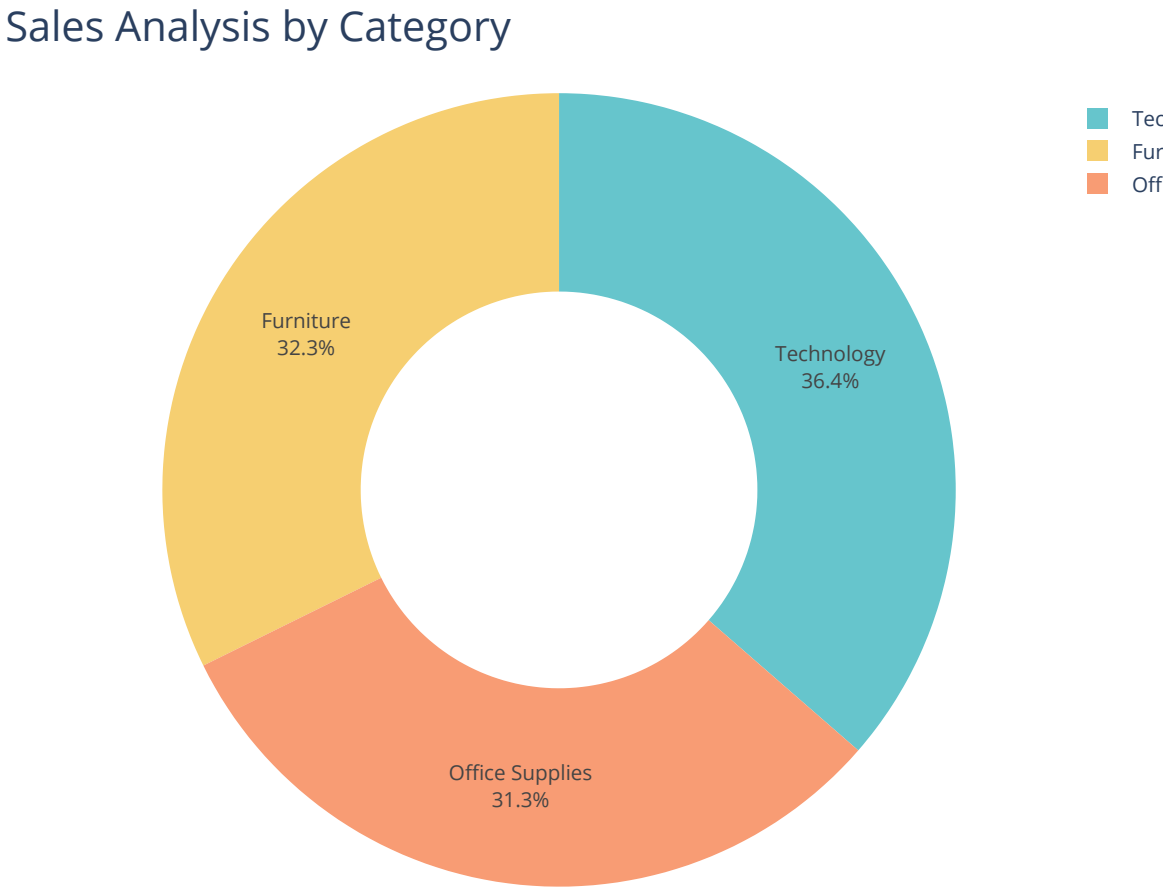
In [9]:
```
# Extracting additional time-related features: month, year, and day of the week
data['Order Month'] = data['Order Date'].dt.month
data['Order Year'] = data['Order Date'].dt.year
data['Order Day of Week'] = data['Order Date'].dt.dayofweek
```

In [10]:
```
#Checking Monthly sales
sales_by_month = data.groupby('Order Month')['Sales'].sum().reset_index()
fig1 = px.line(sales_by_month,
```

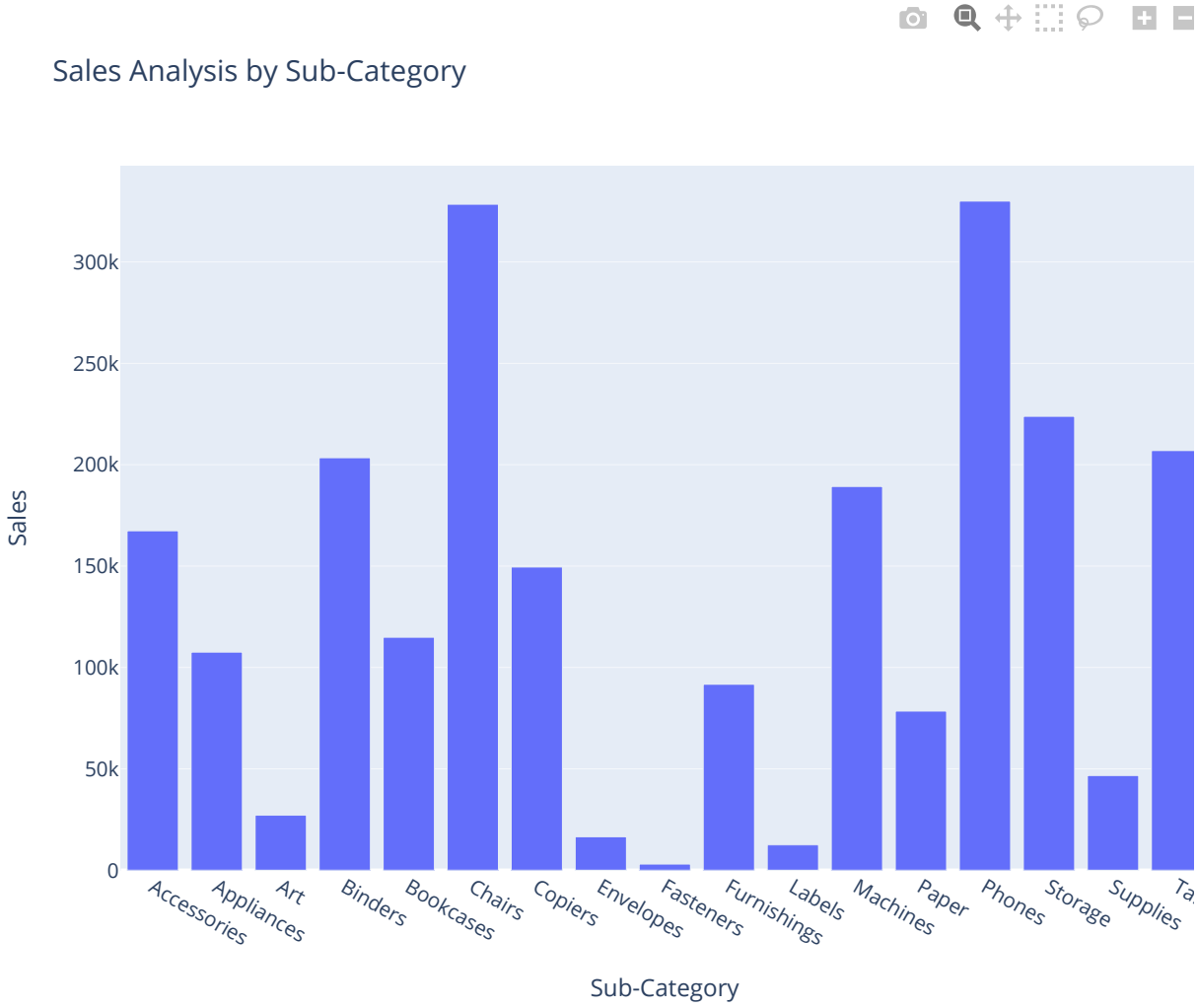
```
              x='Order Month',
              y='Sales',
              title='Monthly Sales Analysis')


fig1.show()
```



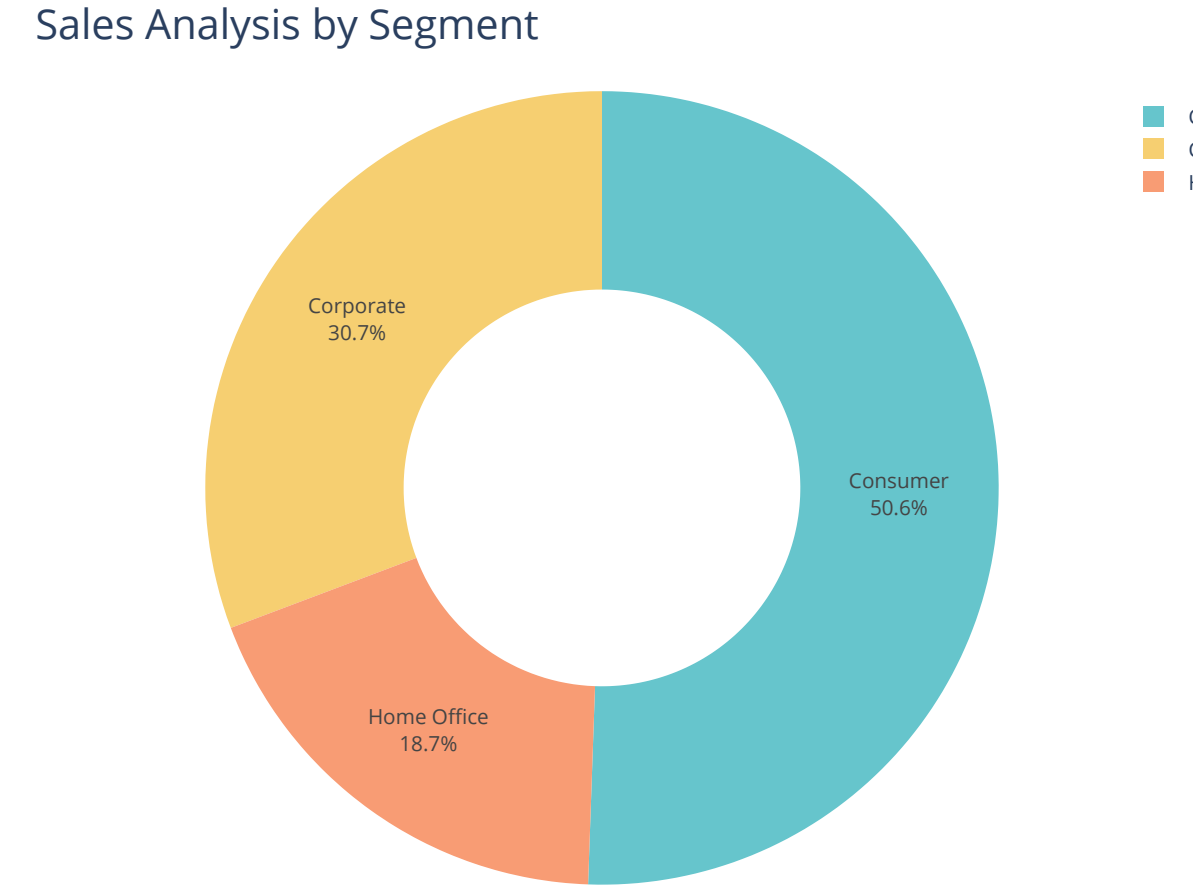
In [11]:
```
# sales by category
sales_by_category = data.groupby('Category')['Sales'].sum().reset_index()


fig2 = px.pie(sales_by_category,
              values='Sales',
              names='Category',
              hole=0.5,
              color_discrete_sequence=px.colors.qualitative.Pastel)

fig2.update_traces(textposition='inside', textinfo='percent+label')
fig2.update_layout(title_text='Sales Analysis by Category', title_font=dict(size=24),width=800, hei

fig2.show()
```



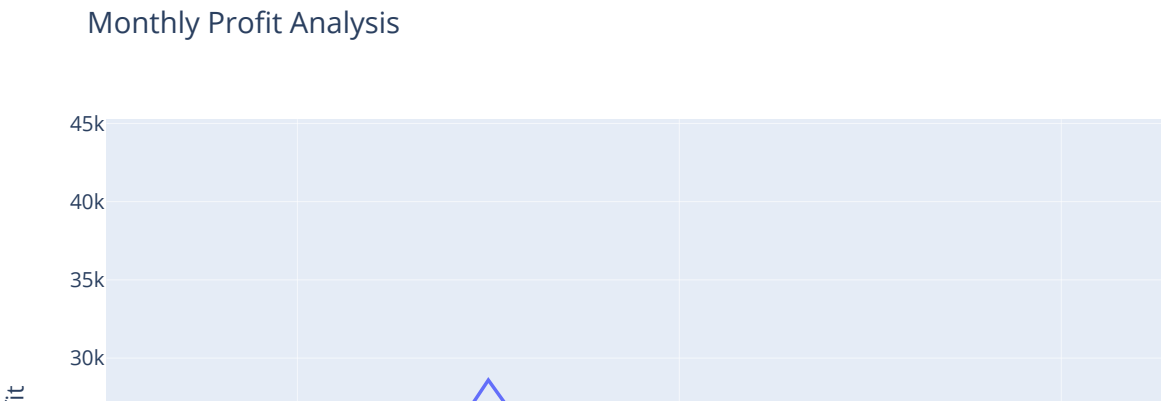
In [12]:
```
#Sales by Sub-Category
sales_by_subcategory = data.groupby('Sub-Category')['Sales'].sum().reset_index()
fig3 = px.bar(sales_by_subcategory,
              x='Sub-Category',
              y='Sales',
              title='Sales Analysis by Sub-Category')
fig3.update_layout(width=800, height=600)
fig3.show()
```



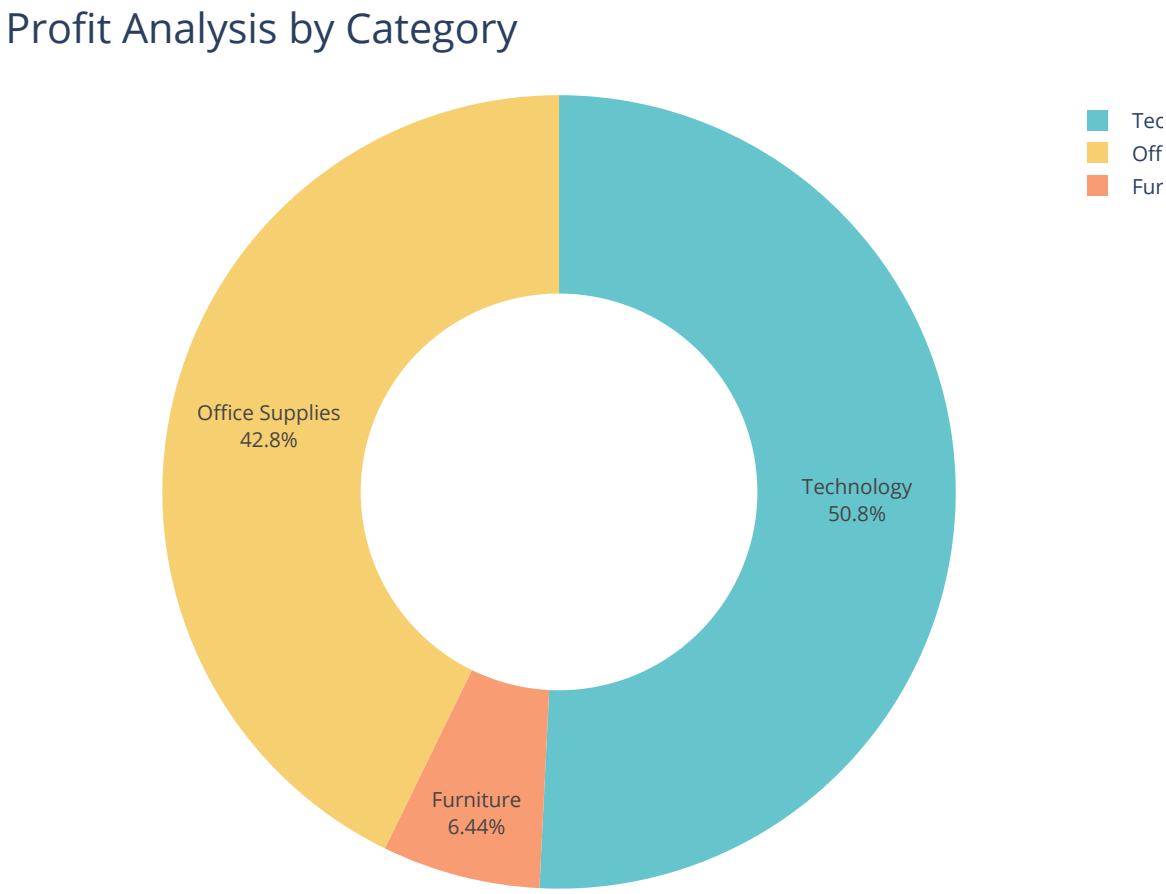
In [13]:
```
# sales by Segment
sales_by_segment = data.groupby('Segment')['Sales'].sum().reset_index()


fig4 = px.pie(sales_by_segment,
              values='Sales',
              names='Segment',
              hole=0.5,
              color_discrete_sequence=px.colors.qualitative.Pastel)

fig4.update_traces(textposition='inside', textinfo='percent+label')
fig4.update_layout(title_text='Sales Analysis by Segment', title_font=dict(size=24),width=800, heig

fig4.show()
```



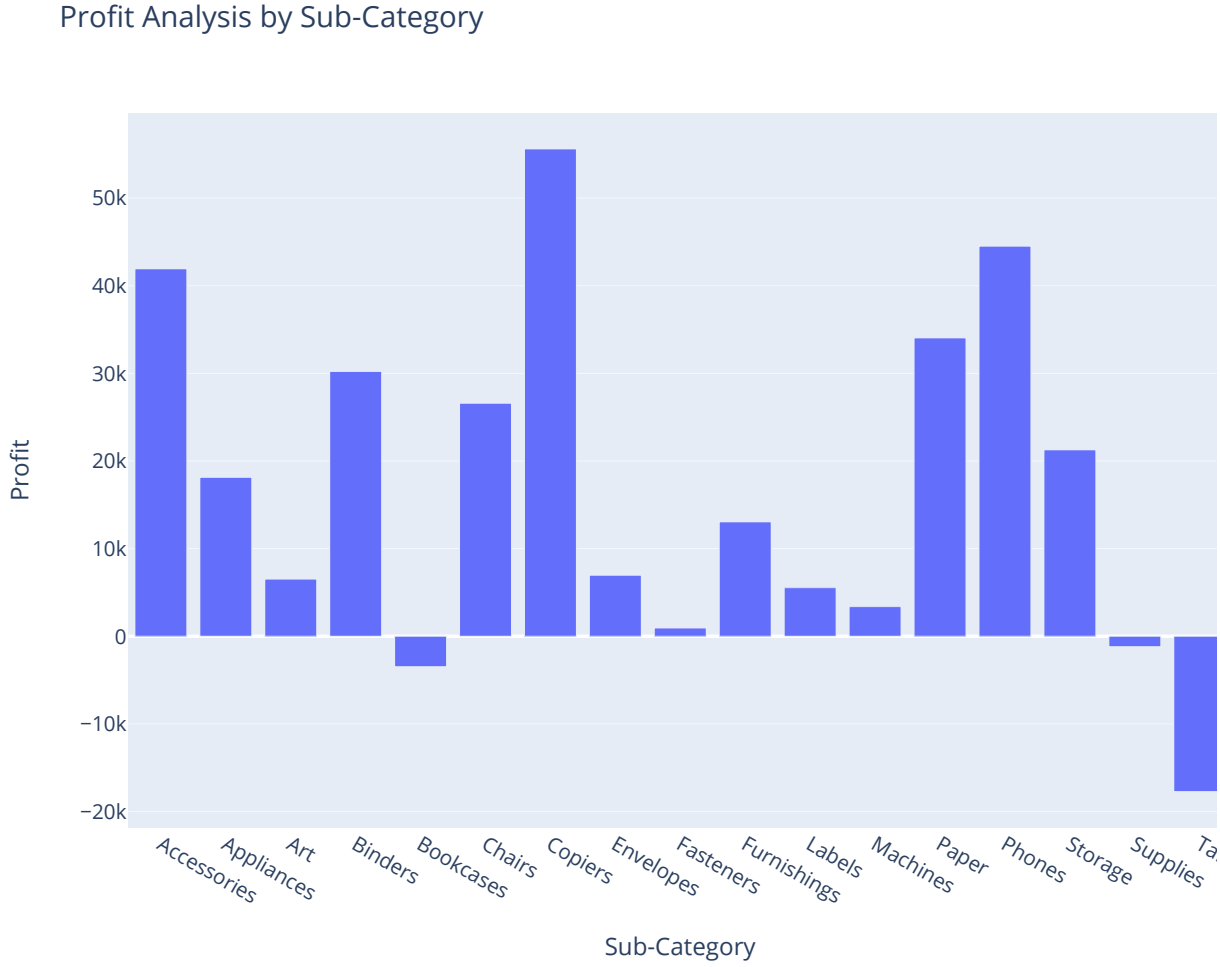
In [14]:
```
# Monthly Profits
profit_by_month = data.groupby('Order Month')['Profit'].sum().reset_index()
fig5 = px.line(profit_by_month,
               x='Order Month',
               y='Profit',
               title='Monthly Profit Analysis')
fig5.show()
```



In [15]:
```
# Profit by Category
profit_by_category = data.groupby('Category')['Profit'].sum().reset_index()

fig6 = px.pie(profit_by_category,
              values='Profit',
              names='Category',
              hole=0.5,
              color_discrete_sequence=px.colors.qualitative.Pastel)

fig6.update_traces(textposition='inside', textinfo='percent+label')
fig6.update_layout(title_text='Profit Analysis by Category', title_font=dict(size=24),width=800, he

fig6.show()
```

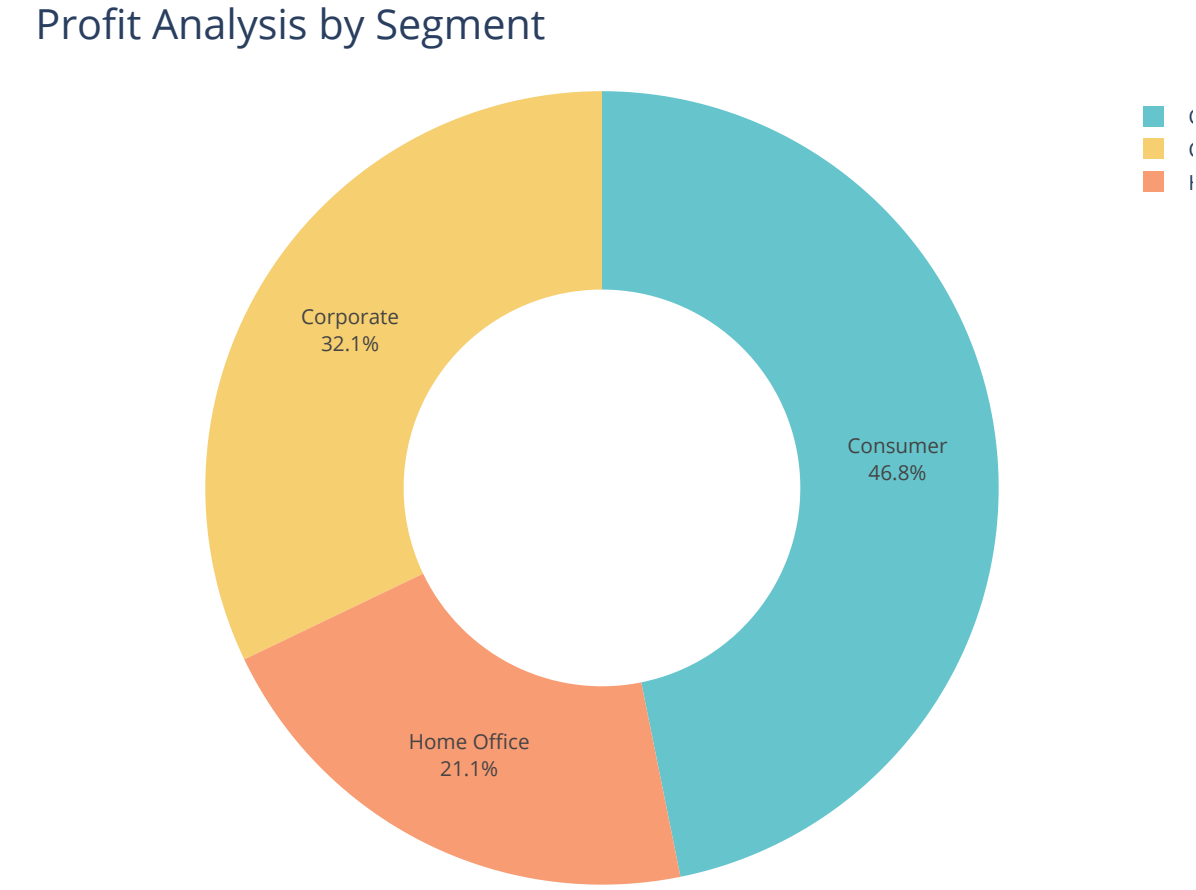


In [16]:
```
# Profit by Sub-Category
profit_by_subcategory = data.groupby('Sub-Category')['Profit'].sum().reset_index()
fig7 = px.bar(profit_by_subcategory, x='Sub-Category',
              y='Profit',
              title='Profit Analysis by Sub-Category')
fig7.update_layout(width=800, height=600)
fig7.show()
```

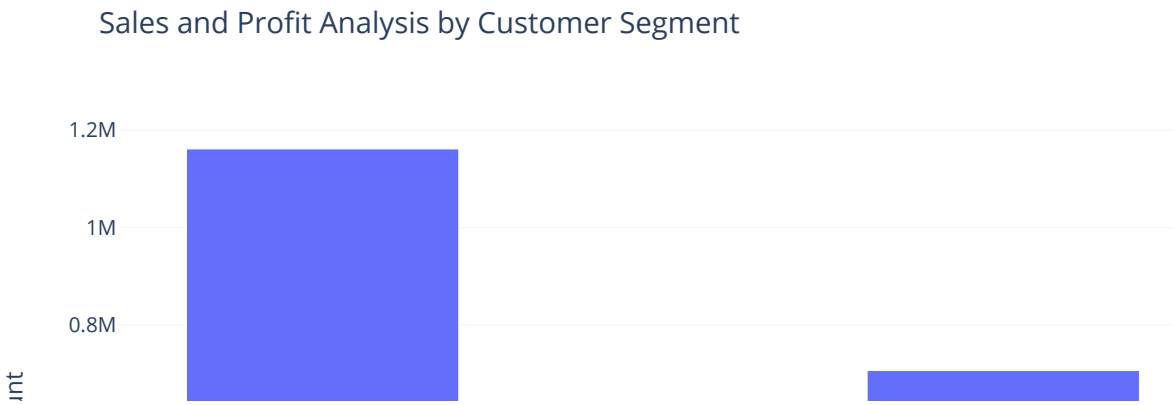


In [17]:

```
# Profit by Segment
profit_by_Segment = data.groupby('Segment')['Profit'].sum().reset_index()

fig8 = px.pie(profit_by_Segment,
             values='Profit',
             names='Segment',
             hole=0.5,
             color_discrete_sequence=px.colors.qualitative.Pastel)

fig8.update_traces(textposition='inside', textinfo='percent+label')
fig8.update_layout(title_text='Profit Analysis by Segment', title_font=dict(size=24),width=800, hei

fig8.show()
```



In [18]:
```
# Sales and profit analysis by customer segments
sales_profit_by_segment = data.groupby('Segment').agg({'Sales': 'sum', 'Profit': 'sum'}).reset_inde
fig9 = go.Figure()
fig9.add_trace(go.Bar(x=sales_profit_by_segment['Segment'], y=sales_profit_by_segment['Sales'], nar
fig9.add_trace(go.Bar(x=sales_profit_by_segment['Segment'], y=sales_profit_by_segment['Profit'], na
fig9.update_layout(title='Sales and Profit Analysis by Customer Segment', xaxis_title='Customer Seg
                   yaxis_title='Amount', barmode='group', template='plotly_white')
```



The store has higher profits from the product sales for consumers, but the profit from corporate product sales is better in the sales-to-profit ratio.

Summary

Store sales and profit analysis help businesses identify areas for improvement and make data-driven decisions to optimize their operations, pricing, marketing, and inventory management strategies to drive revenue and growth.*italicized text*