```python
In [3]: import pandas as pd                 # It has functions for analyzing, cleaning, exploring, and manipulating data.
        import plotly.express as px         # it is advance library used for data visualization,it have dynamic feature.
        import plotly.graph_objects as go   # used for making advanced and customized graph.
        import plotly.io as pio             # used for customized graph template.
        import plotly.colors as colors
        pio.templates.default = "plotly_white"  # take theme white.
```

```python
In [4]: data = pd.read_csv("Sample - Superstore.csv", encoding = 'latin-1') # read data file.
```

```python
In [ ]: data    # call data file.
```

```python
In [6]: data.head()    # shows only top 5 rows.
```

Out[6]:

	Row ID	Order ID	Order Date	Ship Date	Ship Mode	Customer ID	Customer Name	Segment	Country	City	...	Postal Code	Region	Product ID	Category	Sub-Category	Product Name	Sales	Quantity	Discount	Profit
0	1	CA-2016-152156	11/8/2016	11/11/2016	Second Class	CG-12520	Claire Gute	Consumer	United States	Henderson	...	42420	South	FUR-BO-10001798	Furniture	Bookcases	Bush Somerset Collection Bookcase	261.9600	2	0.00	41.9136
1	2	CA-2016-152156	11/8/2016	11/11/2016	Second Class	CG-12520	Claire Gute	Consumer	United States	Henderson	...	42420	South	FUR-CH-10000454	Furniture	Chairs	Hon Deluxe Fabric Upholstered Stacking Chairs,...	731.9400	3	0.00	219.5820
2	3	CA-2016-138688	6/12/2016	6/16/2016	Second Class	DV-13045	Darrin Van Huff	Corporate	United States	Los Angeles	...	90036	West	OFF-LA-10000240	Office Supplies	Labels	Self-Adhesive Address Labels for Typewriters b...	14.6200	2	0.00	6.8714
3	4	US-2015-108966	10/11/2015	10/18/2015	Standard Class	SO-20335	Sean O'Donnell	Consumer	United States	Fort Lauderdale	...	33311	South	FUR-TA-10000577	Furniture	Tables	Bretford CR4500 Series Slim Rectangular Table	957.5775	5	0.45	-383.0310
4	5	US-2015-108966	10/11/2015	10/18/2015	Standard Class	SO-20335	Sean O'Donnell	Consumer	United States	Fort Lauderdale	...	33311	South	OFF-ST-10000760	Office Supplies	Storage	Eldon Fold 'N Roll Cart System	22.3680	2	0.20	2.5164

5 rows × 21 columns

```python
In [7]: data.describe()    # provides key metrics like mean,standard deviation, percentiles and more.
```

Out[7]:

	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

```python
In [8]: data.info()          #  The info() contains the number of columns, column labels, column data types, memory usage,
                             #...range index,and the number of cells in each column (non-null values).
```

```
<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
```

```python
In [13]: data['Order Date'] = pd.to_datetime(data['Order Date'])   # change data type => converting date columns.
         data['Ship Date'] = pd.to_datetime(data['Ship Date'])
```

```python
In [14]: 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
```

for analysis add three new columns in our dataset

```python
In [15]: 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
```

```python
In [16]: data.head()
```

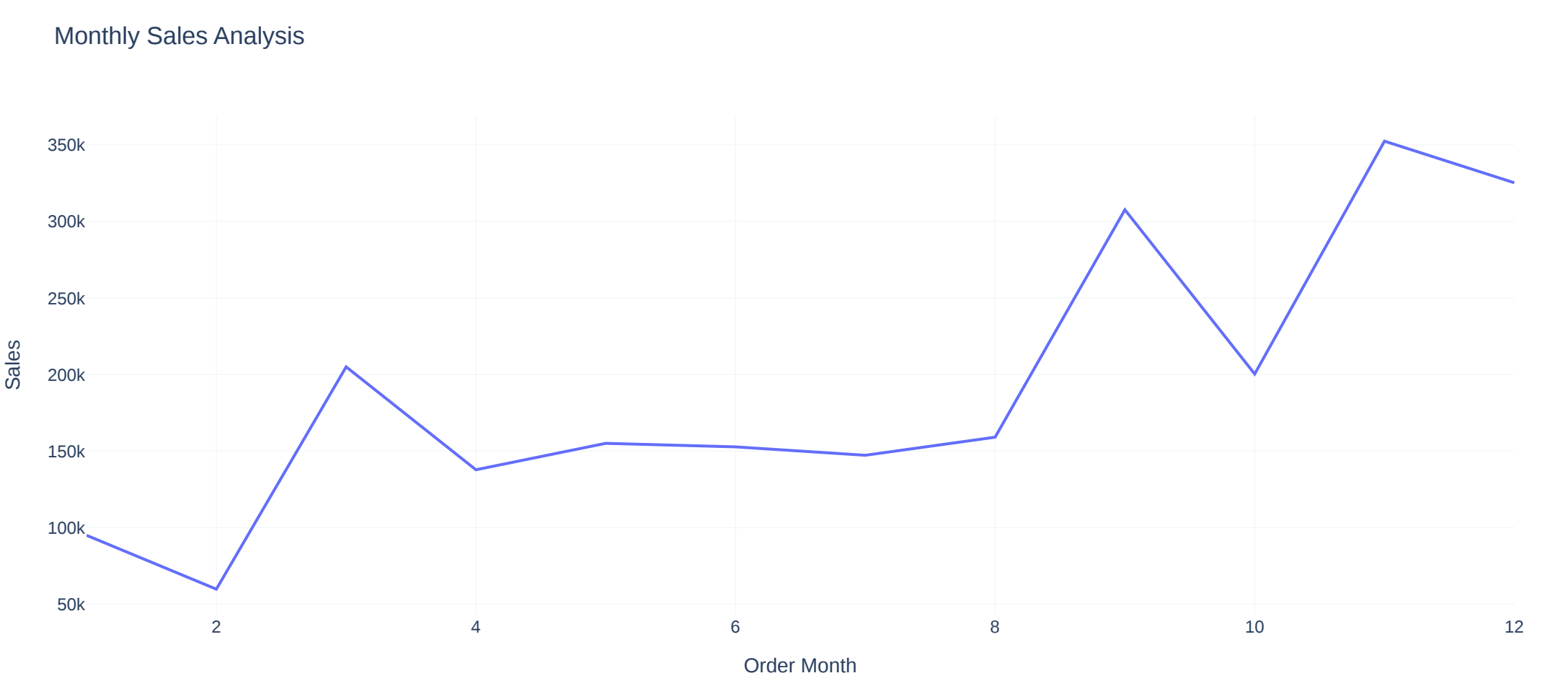
Out[16]:

	Row ID	Order ID	Order Date	Ship Date	Ship Mode	Customer ID	Customer Name	Segment	Country	City	...	Category	Sub-Category	Product Name	Sales	Quantity	Discount	Profit	Order Month	Order Year	Order Day of Week
0	1	CA-2016-152156	2016-11-08	2016-11-11	Second Class	CG-12520	Claire Gute	Consumer	United States	Henderson	...	Furniture	Bookcases	Bush Somerset Collection Bookcase	261.9600	2	0.00	41.9136	11	2016	1
1	2	CA-2016-152156	2016-11-08	2016-11-11	Second Class	CG-12520	Claire Gute	Consumer	United States	Henderson	...	Furniture	Chairs	Hon Deluxe Fabric Upholstered Stacking Chairs,...	731.9400	3	0.00	219.5820	11	2016	1
2	3	CA-2016-138688	2016-06-12	2016-06-16	Second Class	DV-13045	Darrin Van Huff	Corporate	United States	Los Angeles	...	Office Supplies	Labels	Self-Adhesive Address Labels for Typewriters b...	14.6200	2	0.00	6.8714	6	2016	6
3	4	US-2015-108966	2015-10-11	2015-10-18	Standard Class	SO-20335	Sean O'Donnell	Consumer	United States	Fort Lauderdale	...	Furniture	Tables	Bretford CR4500 Series Slim Rectangular Table	957.5775	5	0.45	-383.0310	10	2015	6
4	5	US-2015-108966	2015-10-11	2015-10-18	Standard Class	SO-20335	Sean O'Donnell	Consumer	United States	Fort Lauderdale	...	Office Supplies	Storage	Eldon Fold 'N Roll Cart System	22.3680	2	0.20	2.5164	10	2015	6

5 rows × 24 columns

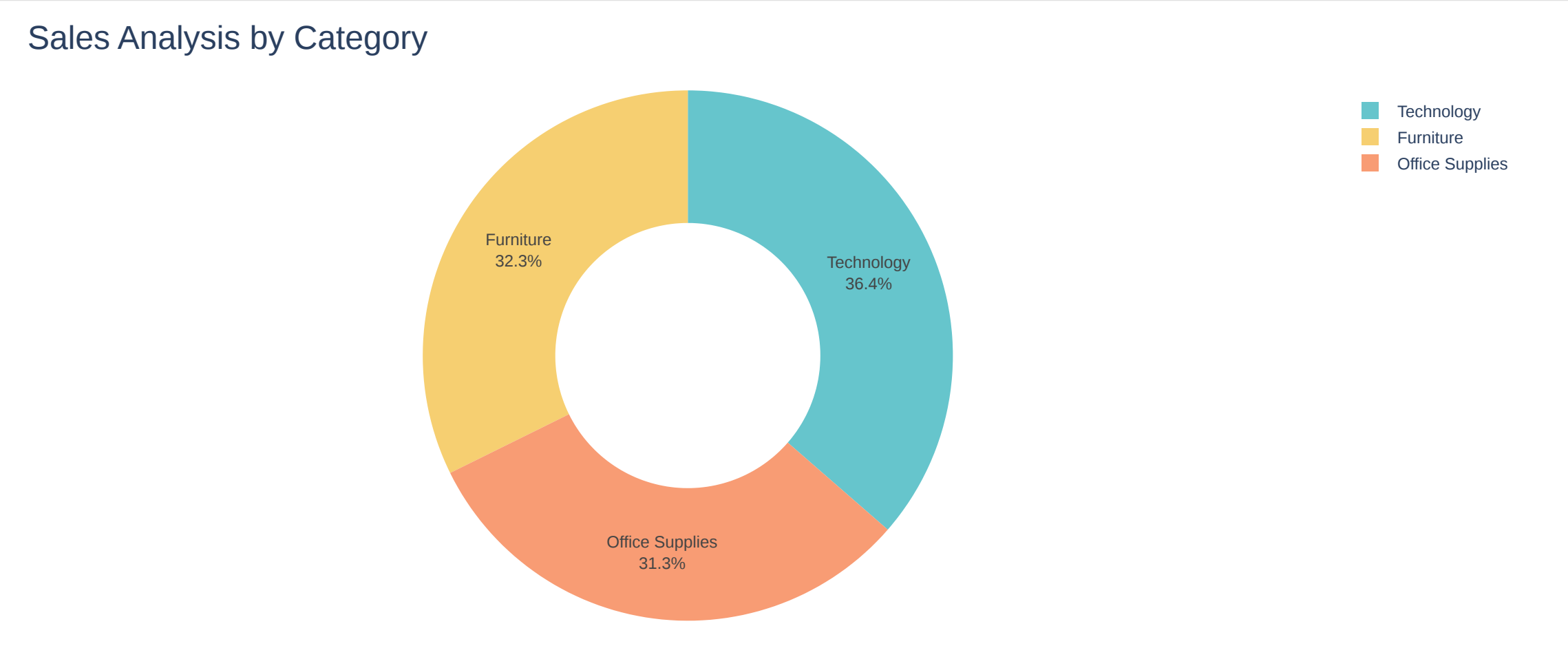
Analysis1: calculate the monthly sales of the store and identify which month had the highest sales and which month and which month had the lowest sales.

```python
In [17]: sales_by_month = data.groupby('Order Month')['Sales'].sum().reset_index()  # it is a Pandas method used to reset the index of a DataFrame — especially useful after operations like groupby()


         fig = px.line(sales_by_month,      # px.line used for making line chart refers to the line function within the plotly.express module.
                       x='Order Month',
                       y='Sales',
                       title='Monthly Sales Analysis')    # Graph Name
         fig.show()
```



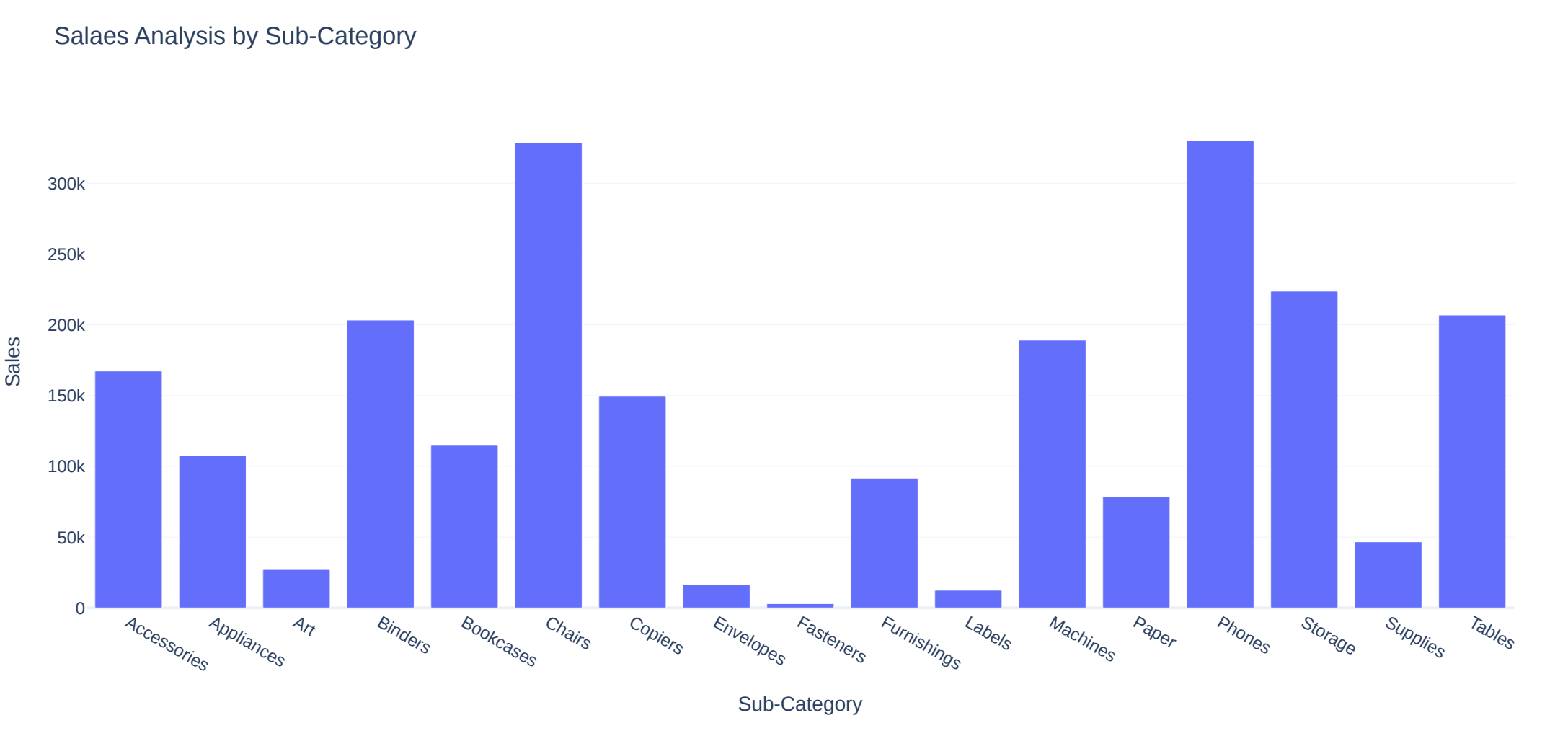
Analysis2: sales based on product categories and determine which category has the lowest sales and which category has the highest sales.

```python
In [23]: sales_by_category = data.groupby('Category')['Sales'].sum().reset_index()

         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 Analysis by Category',title_font=dict(size=24))
         fig.show()
```



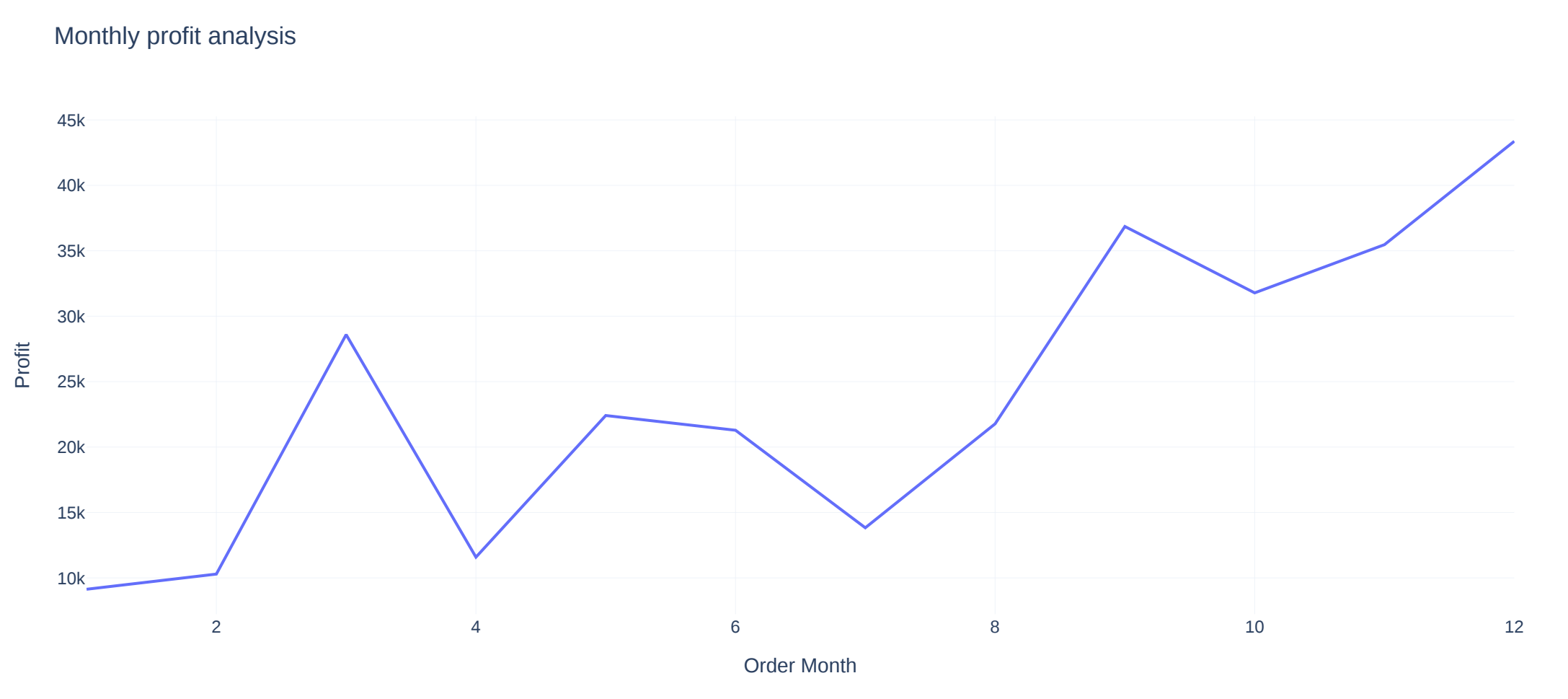
Analysis3: The sales analysis needs to be done based on sub_categories

```python
In [25]: sales_by_subcategory = data.groupby('Sub-Category')['Sales'].sum().reset_index()
         fig = px.bar(sales_by_subcategory,
                      x='Sub-Category',
                      y='Sales',
                      title='Salaes Analysis by Sub-Category')
         fig.show()
```



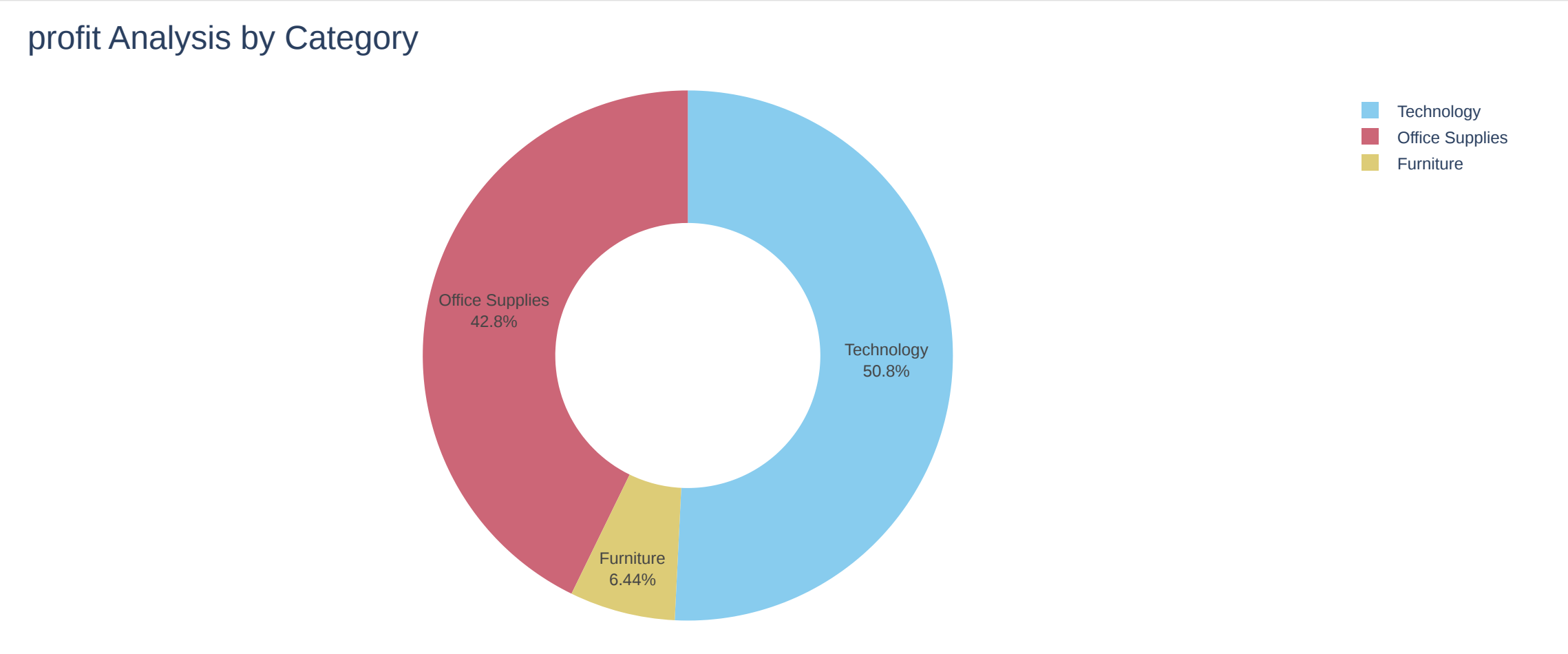
Analysis4: Analyze monthly profit from sales and determine which month had the highest profit.

```python
In [27]: profit_by_month = data.groupby('Order Month')['Profit'].sum().reset_index()
         fig = px.line(profit_by_month,
                       x='Order Month',
                       y='Profit',
                       title='Monthly profit analysis')
         fig.show()
```

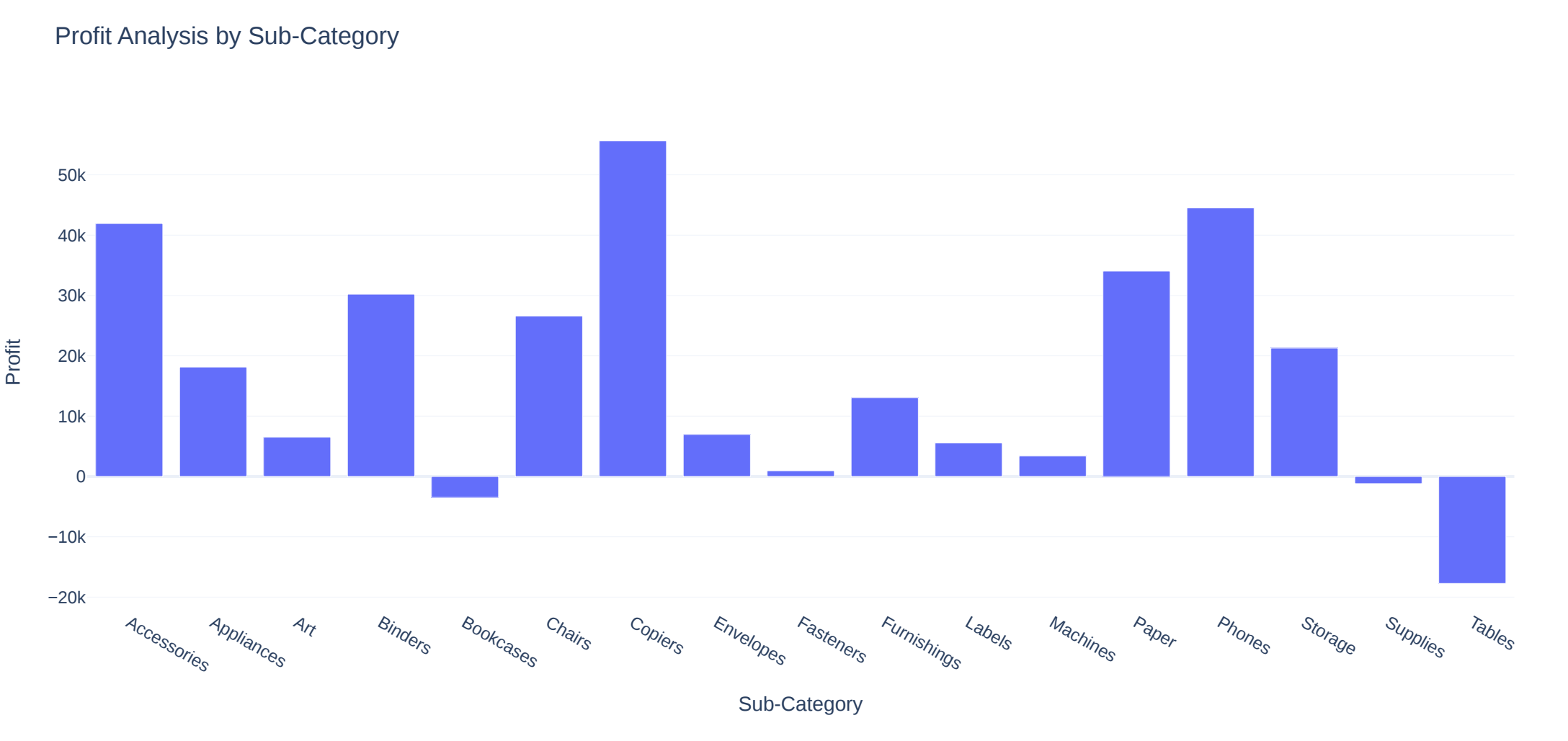


Analysis5: analyze the profit by category and sub-category.

```python
In [40]: profit_by_category = data.groupby('Category')['Profit'].sum().reset_index()
         fig =px.pie(profit_by_category,
                     values='Profit',
                     names='Category',
                     hole=0.5,
                     color_discrete_sequence=px.colors.qualitative.Safe)
         fig.update_traces(textposition='inside',textinfo='percent+label')
         fig.update_layout(title_text='profit Analysis by Category',title_font=dict(size=24))
         fig.show()
```

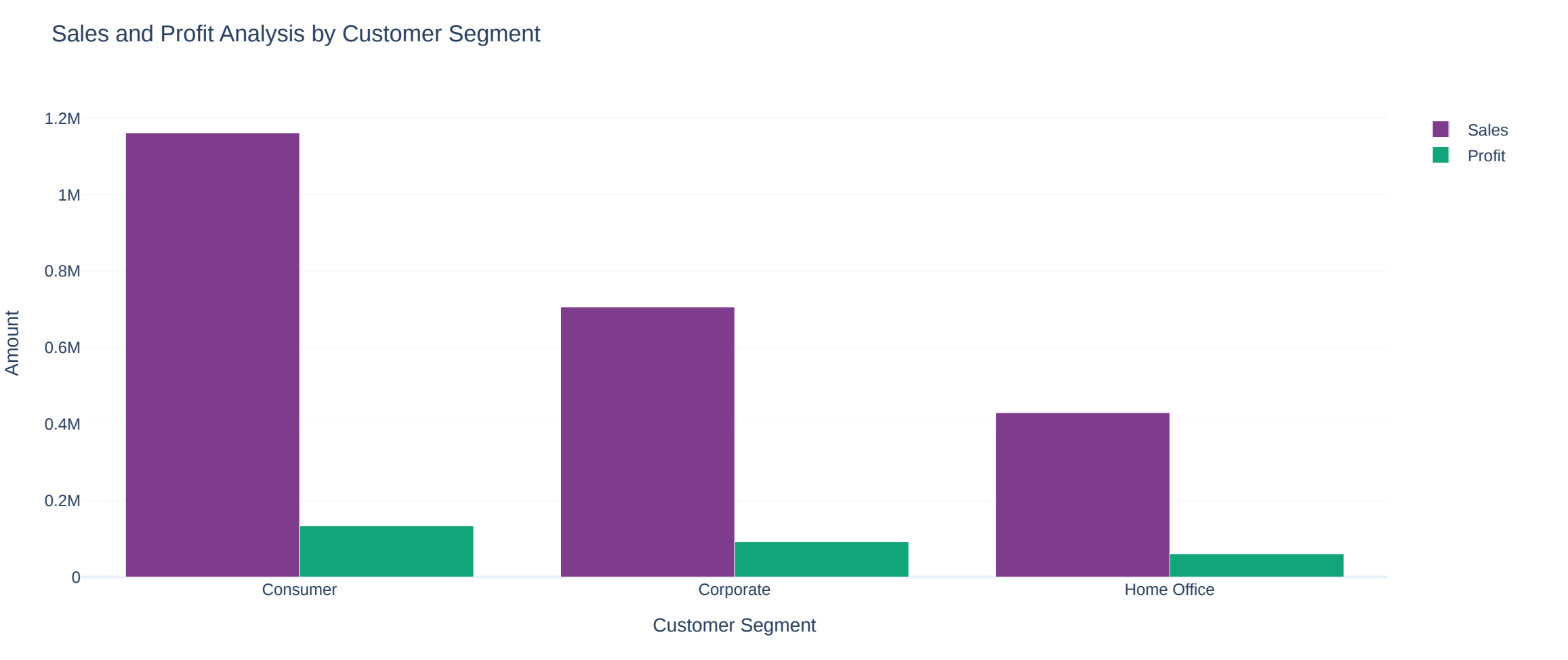



```python
In [41]: profit_by_subcategory = data.groupby('Sub-Category')['Profit'].sum().reset_index()
         fig =px.bar(profit_by_subcategory,
                     x='Sub-Category',
                     y='Profit',
                     title='Profit Analysis by Sub-Category')
         fig.show()
```



Analysis6: analyze the sales and profit by customer segment

```python
In [59]: sales_profit_by_segment=data.groupby('Segment').agg({'Sales':'sum','Profit':'sum'}).reset_index()

         color_palette = colors.qualitative.Bold

         fig = go.Figure()   # it is part of Plotly’s Graph Objects (go) interface, used for building charts in a more
                             # customizable and low-level way than plotly.express.

         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 and Profit Analysis by Customer Segment',
                           xaxis_title='Customer Segment',
                           yaxis_title='Amount')
         fig.show()
```



Analysis7: analyze the sales to profit ratio.

```python
In [62]: sales_profit_by_segment = data.groupby('Segment').agg({'Sales':'sum','Profit':'sum'}).reset_index()
         sales_profit_by_segment['Sales_to_Profit_Ratio']=sales_profit_by_segment['Sales']/sales_profit_by_segment['Profit']
         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
```