Importing Required Libraries & Dataset

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import ticker
import seaborn as sns
import plotly.express as px

path = 'GlobalSuperstore.csv'
data = pd.read_csv(path)

data.head()
```

```
   Row ID         Order ID  Order Date   Ship Date     Ship Mode
Customer ID  \
0   32298   CA-2012-124891   7/31/2012   7/31/2012      Same Day
RH-19495
1   26341    IN-2013-77878    2/5/2013    2/7/2013  Second Class
JR-16210
2   25330    IN-2013-71249  10/17/2013  10/18/2013   First Class
CR-12730
3   13524  ES-2013-1579342   1/28/2013   1/30/2013   First Class
KM-16375
4   47221     SG-2013-4320   11/5/2013   11/6/2013      Same Day
RH-9495

      Customer Name      Segment           City            State  ...
\
0       Rick Hansen     Consumer  New York City         New York  ...

1     Justin Ritter    Corporate     Wollongong  New South Wales  ...

2      Craig Reiter     Consumer       Brisbane       Queensland  ...

3  Katherine Murray  Home Office         Berlin           Berlin  ...

4       Rick Hansen     Consumer          Dakar            Dakar  ...


         Product ID    Category Sub-Category  \
0   TEC-AC-10003033  Technology  Accessories
1   FUR-CH-10003950   Furniture       Chairs
2   TEC-PH-10004664  Technology       Phones
3   TEC-PH-10004583  Technology       Phones
4  TEC-SHA-10000501  Technology      Copiers

                                        Product Name     Sales
Quantity  \
```

```
0  Plantronics CS510 - Over-the-Head monaural Wir...  2309.650
7
1          Novimex Executive Leather Armchair, Black  3709.395
9
2                  Nokia Smart Phone, with Caller ID  5175.171
9
3                     Motorola Smart Phone, Cordless  2892.510
5
4                     Sharp Wireless Fax, High-Speed  2832.960
8

  Discount    Profit  Shipping Cost  Order Priority
0      0.0  762.1845         933.57        Critical
1      0.1 -288.7650         923.63        Critical
2      0.1  919.9710         915.49          Medium
3      0.1  -96.5400         910.16          Medium
4      0.0  311.5200         903.04        Critical

[5 rows x 24 columns]
```

Assessing the Data

```
data.shape
```

```
(51290, 24)
```

```
data.columns
```

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

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 51290 entries, 0 to 51289
Data columns (total 24 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Row ID          51290 non-null  int64
 1   Order ID        51290 non-null  object
 2   Order Date      51290 non-null  object
 3   Ship Date       51290 non-null  object
 4   Ship Mode       51290 non-null  object
```

```
 5   Customer ID     51290 non-null  object
 6   Customer Name   51290 non-null  object
 7   Segment         51290 non-null  object
 8   City            51290 non-null  object
 9   State           51290 non-null  object
 10  Country         51290 non-null  object
 11  Postal Code     9994 non-null   float64
 12  Market          51290 non-null  object
 13  Region          51290 non-null  object
 14  Product ID      51290 non-null  object
 15  Category        51290 non-null  object
 16  Sub-Category    51290 non-null  object
 17  Product Name    51290 non-null  object
 18  Sales           51290 non-null  float64
 19  Quantity        51290 non-null  int64
 20  Discount        51290 non-null  float64
 21  Profit          51290 non-null  float64
 22  Shipping Cost   51290 non-null  float64
 23  Order Priority  51290 non-null  object
dtypes: float64(5), int64(2), object(17)
memory usage: 9.4+ MB
```

```
# Numerical Data Statistics
data.describe()
```

```
            Row ID   Postal Code         Sales      Quantity
Discount  \
count  51290.00000   9994.000000  51290.000000  51290.000000
51290.000000
mean   25645.50000  55190.379428    246.490581      3.476545
0.142908
std    14806.29199  32063.693350    487.565361      2.278766
0.212280
min        1.00000   1040.000000      0.444000      1.000000
0.000000
25%    12823.25000  23223.000000     30.758625      2.000000
0.000000
50%    25645.50000  56430.500000     85.053000      3.000000
0.000000
75%    38467.75000  90008.000000    251.053200      5.000000
0.200000
max    51290.00000  99301.000000  22638.480000     14.000000
0.850000

             Profit  Shipping Cost
count  51290.000000   51290.000000
mean      28.610982      26.375915
std      174.340972      57.296804
min    -6599.978000       0.000000
25%        0.000000       2.610000
```

```
50%        9.240000       7.790000
75%       36.810000      24.450000
max     8399.976000     933.570000

# Categorical Data Statistics
data.describe(include='object')

               Order ID Order Date   Ship Date       Ship Mode Customer
ID  \
count             51290      51290       51290           51290
51290
unique            25035       1430        1464               4
1590
top     CA-2014-100111  6/18/2014  11/22/2014  Standard Class    PO-
18850
freq                 14        135         130           30775
97

          Customer Name   Segment           City       State
Country  \
count              51290     51290          51290       51290
51290
unique               795         3           3636        1094
147
top     Muhammed Yedwab  Consumer  New York City  California  United
States
freq                 108     26518            915        2001
9994

       Market   Region       Product ID         Category Sub-Category
\
count   51290    51290            51290            51290        51290

unique      7       13            10292                3           17

top      APAC  Central  OFF-AR-10003651  Office Supplies      Binders

freq    11002    11117               35            31273         6152


       Product Name Order Priority
count         51290          51290
unique         3788              4
top         Staples         Medium
freq            227          29433

data.dtypes

Row ID              int64
Order ID           object
Order Date         object
```

```
Ship Date          object
Ship Mode          object
Customer ID        object
Customer Name      object
Segment            object
City               object
State              object
Country            object
Postal Code       float64
Market             object
Region             object
Product ID         object
Category           object
Sub-Category       object
Product Name       object
Sales             float64
Quantity            int64
Discount          float64
Profit            float64
Shipping Cost     float64
Order Priority     object
dtype: object
```

```
#Checking for Null Values
data.isnull().sum()
```

```
Row ID                0
Order ID              0
Order Date            0
Ship Date             0
Ship Mode             0
Customer ID           0
Customer Name         0
Segment               0
City                  0
State                 0
Country               0
Postal Code       41296
Market                0
Region                0
Product ID            0
Category              0
Sub-Category          0
Product Name          0
Sales                 0
Quantity              0
Discount              0
Profit                0
Shipping Cost         0
```

```
Order Priority         0
dtype: int64
```

```
print(f"Percentage of missing values in Postal Code column -
{np.round(data.isnull().sum()['Postal Code']/data.shape[0]*100,2)}%")
```

```
Percentage of missing values in Postal Code column - 80.51%
```

```
#Checking for Duplicated Data
data.duplicated().sum()
```

```
0
```

Assessment Summary The data had 24 attributes and 51920 rows.

> **Missing Data:** Postal Code column has almost 80% missing values. Therefore, we can ignore this for analysis and remove it from the dataset
>
> **Duplicates:** There are no duplicated rows.
>
> **Data Types:** Ship Date and Order Date are in String Format, needs to be changed

Cleaning Data

```
# Renaming Columns to Snake Case for standardization
data.columns = data.columns.str.replace(' ','_').str.lower()
data.columns
```

```
Index(['row_id', 'order_id', 'order_date', 'ship_date', 'ship_mode',
       'customer_id', 'customer_name', 'segment', 'city', 'state',
'country',
       'postal_code', 'market', 'region', 'product_id', 'category',
       'sub-category', 'product_name', 'sales', 'quantity',
'discount',
       'profit', 'shipping_cost', 'order_priority'],
      dtype='object')
```

```
# Converting the OrderDate and ShipDate to date type.
data['order_date'] = pd.to_datetime(data['order_date'])
data['ship_date'] = pd.to_datetime(data['ship_date'])
```

```
data[['order_date', 'ship_date']].dtypes
```

```
order_date    datetime64[ns]
ship_date     datetime64[ns]
dtype: object
```

```
print('The Date of First order in the dataset: ' +
str(data['order_date'].min()))
print('The Date of Last order in the dataset: ' +
str(data['order_date'].max()))
```

```
The Date of First order in the dataset: 2011-01-01 00:00:00
The Date of Last order in the dataset: 2014-12-31 00:00:00

# Creating new columns for Order Year and Order Month
data['order_year'] = pd.DatetimeIndex(data['order_date']).year
data['order_month'] = pd.DatetimeIndex(data['order_date']).month
data.head()

   row_id         order_id order_date  ship_date     ship_mode
customer_id  \
0   32298   CA-2012-124891 2012-07-31 2012-07-31      Same Day    RH-
19495
1   26341    IN-2013-77878 2013-02-05 2013-02-07  Second Class    JR-
16210
2   25330    IN-2013-71249 2013-10-17 2013-10-18   First Class    CR-
12730
3   13524  ES-2013-1579342 2013-01-28 2013-01-30   First Class    KM-
16375
4   47221     SG-2013-4320 2013-11-05 2013-11-06      Same Day     RH-
9495

      customer_name      segment           city            state  ...
\
0       Rick Hansen     Consumer  New York City         New York  ...

1     Justin Ritter    Corporate     Wollongong  New South Wales  ...

2      Craig Reiter     Consumer       Brisbane       Queensland  ...

3  Katherine Murray  Home Office         Berlin           Berlin  ...

4       Rick Hansen     Consumer          Dakar            Dakar  ...


  sub-category                                        product_name
sales  \
0  Accessories  Plantronics CS510 - Over-the-Head monaural Wir...
2309.650
1       Chairs          Novimex Executive Leather Armchair, Black
3709.395
2       Phones                   Nokia Smart Phone, with Caller ID
5175.171
3       Phones                      Motorola Smart Phone, Cordless
2892.510
4      Copiers                      Sharp Wireless Fax, High-Speed
2832.960

  quantity discount    profit shipping_cost order_priority  order_year
\
0        7      0.0  762.1845        933.57       Critical        2012
```

```
1         9      0.1 -288.7650        923.63       Critical         2013

2         9      0.1  919.9710        915.49         Medium         2013

3         5      0.1  -96.5400        910.16         Medium         2013

4         8      0.0  311.5200        903.04       Critical         2013
```

```
   order_month
0            7
1            2
2           10
3            1
4           11

[5 rows x 26 columns]
```

```
del data['postal_code']

#convert categorical columns data type from object to category
cat_cols = data.select_dtypes(include=['object']).columns
data[cat_cols] = data[cat_cols].astype('category')

data.dtypes
```

```
row_id                     int64
order_id                category
order_date        datetime64[ns]
ship_date         datetime64[ns]
ship_mode               category
customer_id             category
customer_name           category
segment                 category
city                    category
state                   category
country                 category
market                  category
region                  category
product_id              category
category                category
sub-category            category
product_name            category
sales                    float64
quantity                   int64
discount                 float64
profit                   float64
shipping_cost            float64
order_priority          category
order_year                 int32
```

```
order_month                int32
dtype: object
```

Exploratory Analysis and Visualization of Data

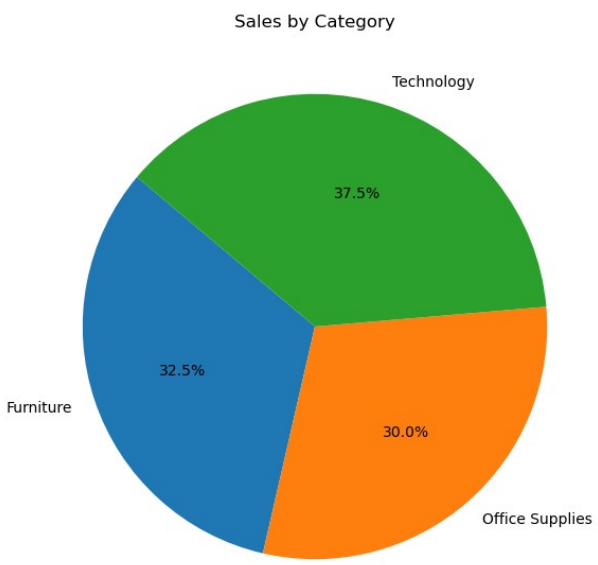
```
# Creating a new columns - Profit Margin, Shipping Time
data['profit_margin'] = data['profit'] / data['sales']
data['shipping_time'] = (data['ship_date'] -
data['order_date']).dt.days
data.head()
```

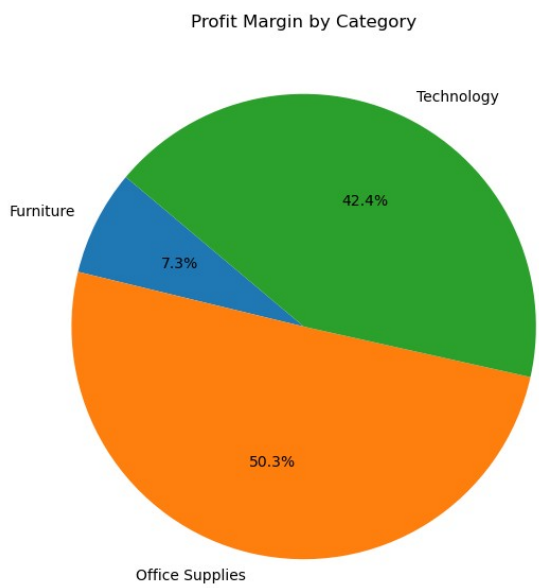
```
   row_id         order_id order_date  ship_date     ship_mode
customer_id  \
0   32298   CA-2012-124891 2012-07-31 2012-07-31      Same Day    RH-
19495
1   26341    IN-2013-77878 2013-02-05 2013-02-07  Second Class    JR-
16210
2   25330    IN-2013-71249 2013-10-17 2013-10-18   First Class    CR-
12730
3   13524  ES-2013-1579342 2013-01-28 2013-01-30   First Class    KM-
16375
4   47221     SG-2013-4320 2013-11-05 2013-11-06      Same Day     RH-
9495

      customer_name      segment           city            state  ...
\
0       Rick Hansen     Consumer  New York City         New York  ...

1     Justin Ritter    Corporate     Wollongong  New South Wales  ...

2      Craig Reiter     Consumer       Brisbane       Queensland  ...

3  Katherine Murray  Home Office         Berlin           Berlin  ...

4       Rick Hansen     Consumer          Dakar            Dakar  ...


      sales quantity discount    profit shipping_cost
order_priority  \
0  2309.650        7      0.0  762.1845        933.57       Critical

1  3709.395        9      0.1 -288.7650        923.63       Critical

2  5175.171        9      0.1  919.9710        915.49         Medium

3  2892.510        5      0.1  -96.5400        910.16         Medium

4  2832.960        8      0.0  311.5200        903.04       Critical
```

```
  order_year  order_month  profit_margin  shipping_time
0       2012            7       0.330000              0
1       2013            2      -0.077847              2
2       2013           10       0.177766              1
3       2013            1      -0.033376              2
4       2013           11       0.109963              1

[5 rows x 27 columns]
```

Sales and Profit Margin by Category

```
sales_by_category = data.groupby('category')['sales'].sum()
profit_margin_by_category = data.groupby('category')
['profit_margin'].mean()

fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(12, 6))

#Pie Chart for Sales
sales_by_category.plot(kind='pie', autopct='%1.1f%%', startangle=140,
ax=ax1)
ax1.set_title('Sales by Category')
ax1.set_ylabel('')

#Pie Chart for Profit Margin
profit_margin_by_category.plot(kind='pie', autopct='%1.1f%%',
startangle=140, ax=ax2)
ax2.set_title('Profit Margin by Category')
ax2.set_ylabel('')

plt.tight_layout()
plt.show()
```

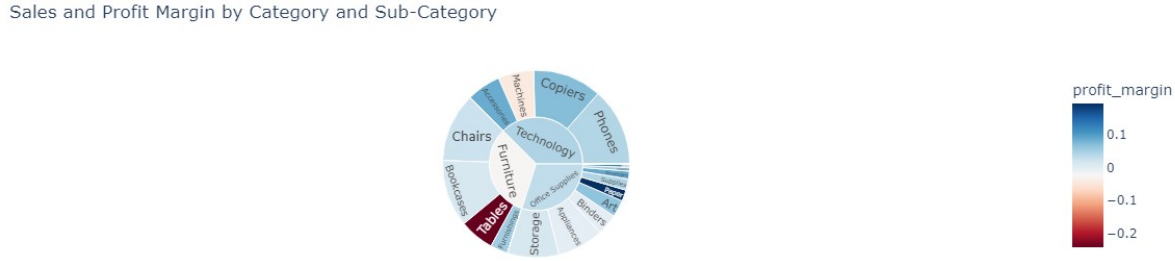




From the Pie Charts, we can observe that Technology(37.5%) has the highest Sales followed by Furniture(32.5%), but the Highest Profit Margin is gained from Office Supplies(50.3%) followed by Technology(42.4%) making Furniture a very low-profitable Category

Sales and Profit Margin by Category drill down with Sub-Category

```
sunburst_data = data.groupby(['category', 'sub-
category']).agg({'sales': 'sum', 'profit_margin':
'mean'}).reset_index()
sunburst_data = sunburst_data[sunburst_data['sales'] > 0]

fig = px.sunburst(
    sunburst_data,
    path=['category', 'sub-category'],
    values='sales',
    color='profit_margin',
    color_continuous_scale='RdBu',
    title='Sales and Profit Margin by Category and Sub-Category'
)
fig.show()
```

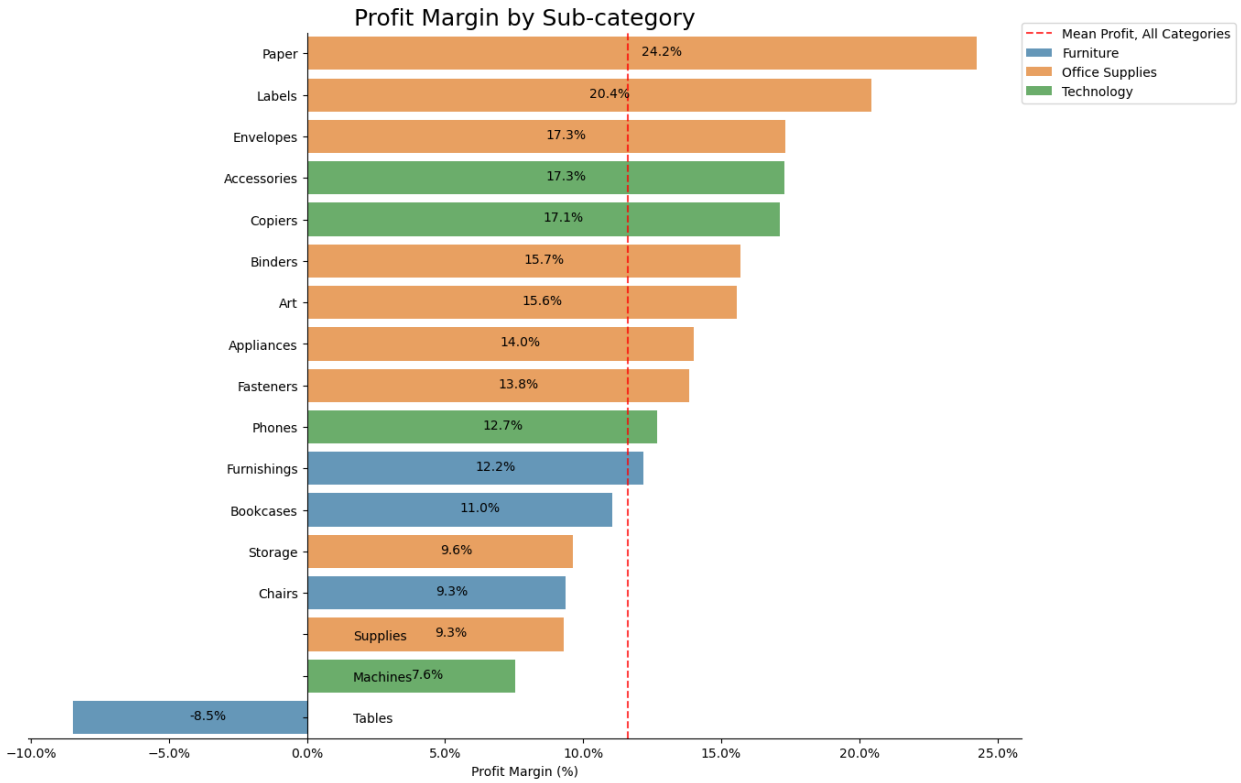


The above sunburst chart gives an idea of Sales(by size) and Profit Margin(by Color) for each Category drilling down through to subcategory

Profit Margin by Subcategory

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

# Grouping the data on category and its respective sub-categories and
calculating the profit margin
sales_per_subcategory = data.groupby(['category', 'sub-category'],
as_index=False)[['sales', 'profit']].sum()
sales_per_subcategory['profit_margin'] =
sales_per_subcategory['profit'] / sales_per_subcategory['sales']

# Sorting the dataframe based on profit margin
sales_per_subcategory =
sales_per_subcategory.sort_values(by='profit_margin', ascending=False)

# Plotting the profit margins sub-category bar chart
fig, ax = plt.subplots(figsize=(14,10))

# Plotting the profit margin per sub-category
sns.barplot(y=sales_per_subcategory['sub-category'],
x=sales_per_subcategory['profit_margin'],
hue=sales_per_subcategory['category'],

order=['Paper','Labels','Envelopes','Accessories','Copiers','Binders',
'Art','Appliances','Fasteners','Phones','Furnishings',

'Bookcases','Storage','Chairs','Supplies','Machines','Tables'],
alpha=0.75, dodge=False, ax=ax)

# Cleaning out bar junk
ax.spines['left'].set_position('zero')
ax.spines[['right', 'top']].set_visible(False)
ax.set(ylabel=None, xlabel='Profit Margin (%)')
```

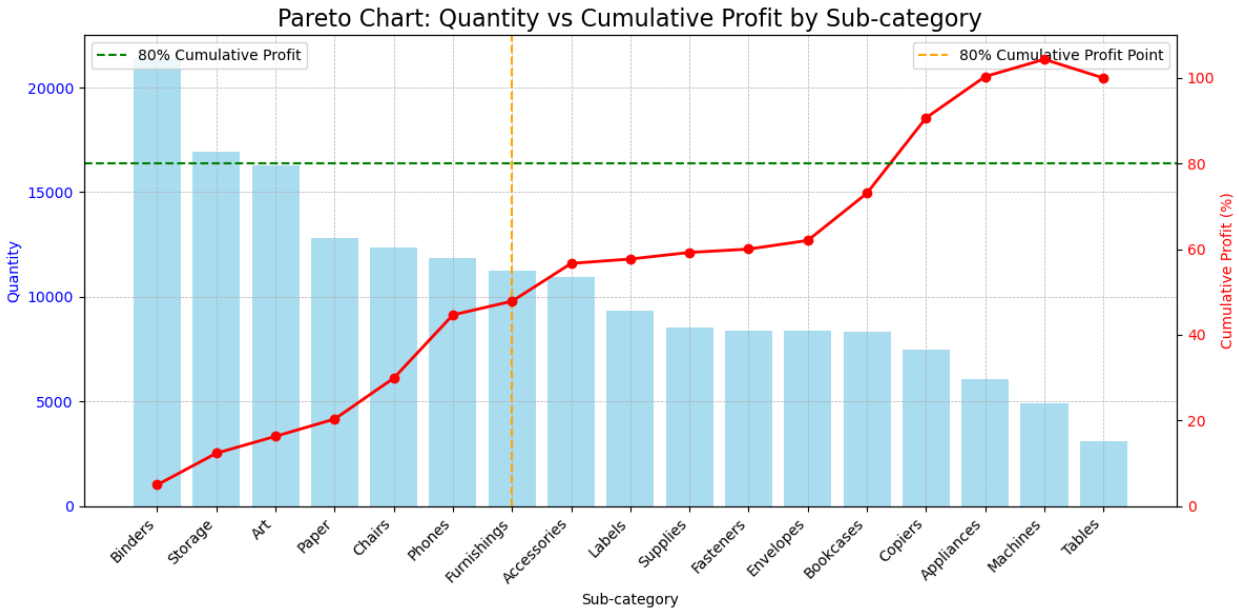
```
def move_ylabel_tick(index: list):
    """
    Moving the provided ylabel ticks
    """
    for tick in index:
        ax.get_yticklabels()[tick].set_x(0.02)
        ax.get_yticklabels()[tick].set_horizontalalignment('left')

# Moving the y-labels on sub-categories that are making a loss to
prevent collision of the bar and the text
move_ylabel_tick([-1, -2, -3])

# Annotating the profit margin amount for each bar
for p in ax.patches:
    _, y = p.get_xy()
    ax.annotate(f'{p.get_width() * 100 :.1f}%', (p.get_width() / 2, y
+ 0.45))

# Calculating the aggregate profit margin for comparison
mean_profit = sales_per_subcategory['profit'].sum() /
sales_per_subcategory['sales'].sum()

# Plotting a vertical line and annotating the aggregate profit margin
ax.axvline(mean_profit, color='red', label='Mean Profit, All
Categories', alpha=0.75, ls='--')

# Setting the title and legend
ax.set_title('Profit Margin by Sub-category',
fontdict={'fontsize':18})
ax.legend(loc=(1, 0.9))

# Formatting the x-axis as %
ax.xaxis.set_major_formatter(mtick.PercentFormatter(1.0))

plt.show()
```



Pareto Chart of Quantity vs. Profit for Sub-Categories

```
# Grouping data by sub-category to calculate total quantity and profit
pareto_data = data.groupby('sub-category').agg({'quantity': 'sum',
'profit': 'sum'}).reset_index()

# Sorting by quantity in descending order
pareto_data = pareto_data.sort_values(by='quantity', ascending=False)

# Calculating cumulative profit and cumulative percentage
pareto_data['cumulative_profit'] = pareto_data['profit'].cumsum()
pareto_data['cumulative_profit_pct'] =
pareto_data['cumulative_profit'] / pareto_data['profit'].sum() * 100

# Find the point where cumulative profit percentage is 80%
threshold = 80
threshold_row = pareto_data[pareto_data['cumulative_profit_pct'] >=
threshold].iloc[0]
threshold_subcategory = threshold_row['sub-category']
threshold_value = threshold_row['cumulative_profit_pct']

# Plotting the Pareto chart
fig, ax1 = plt.subplots(figsize=(12, 6))

# Bar chart for quantity (sorted in descending order)
ax1.bar(pareto_data['sub-category'], pareto_data['quantity'],
color='skyblue', alpha=0.7)
```

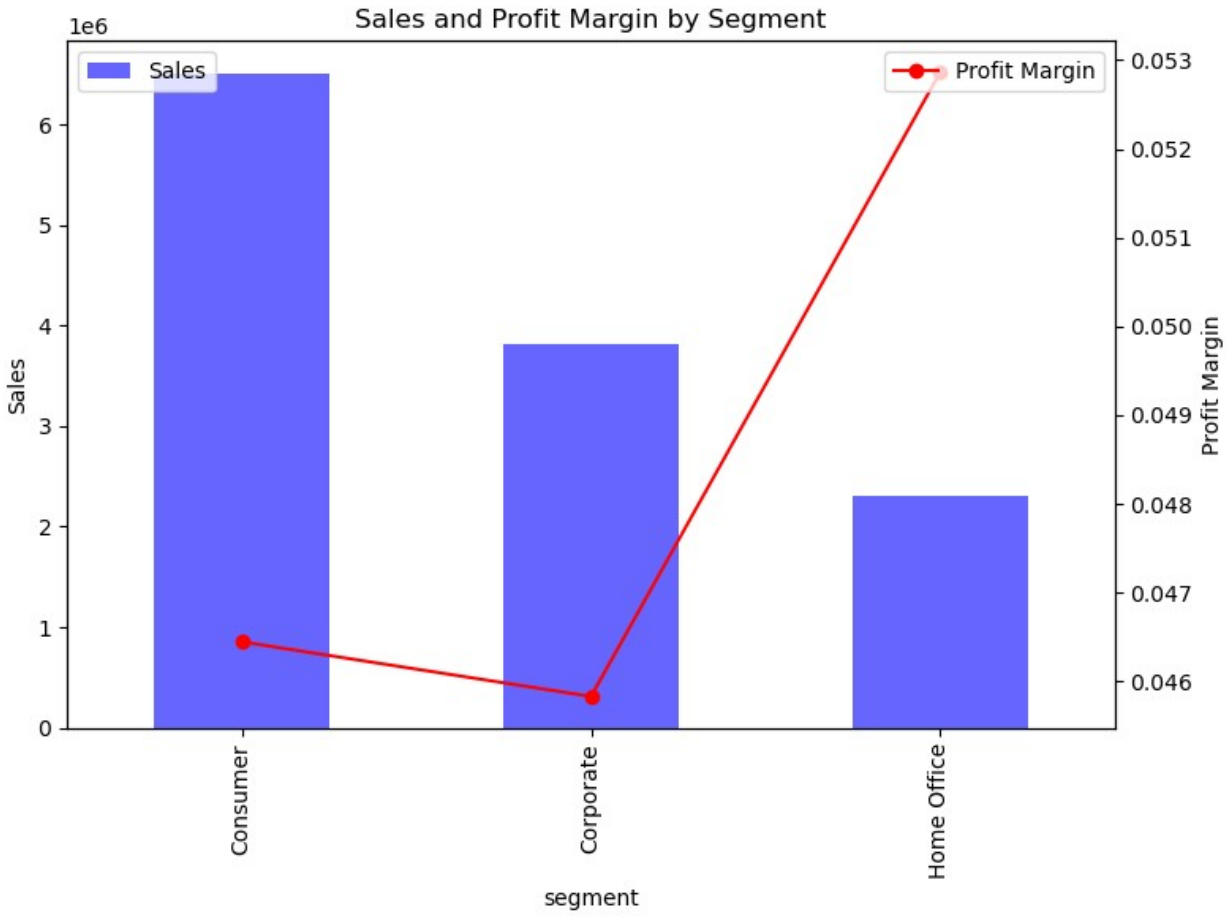
```
ax1.set_xlabel('Sub-category')
ax1.set_ylabel('Quantity', color='blue')
ax1.tick_params(axis='y', labelcolor='blue')
plt.xticks(rotation=45, ha='right')

# Line chart for cumulative profit percentage
ax2 = ax1.twinx()
ax2.plot(pareto_data['sub-category'],
pareto_data['cumulative_profit_pct'], color='red', marker='o',
linestyle='-', linewidth=2)
ax2.set_ylabel('Cumulative Profit (%)', color='red')
ax2.tick_params(axis='y', labelcolor='red')
ax2.set_ylim(0, 110)

# Adding vertical and horizontal lines at the 80% cumulative profit
point
ax2.axhline(threshold, color='green', linestyle='--', linewidth=1.5,
label='80% Cumulative Profit')
threshold_index = pareto_data[pareto_data['sub-category'] ==
threshold_subcategory].index[0]
ax1.axvline(threshold_index, color='orange', linestyle='--',
linewidth=1.5, label='80% Cumulative Profit Point')

# Adding title, labels, and legend
plt.title('Pareto Chart: Quantity vs Cumulative Profit by Sub-
category', fontsize=16)
ax1.set_xlabel('Sub-category')
ax1.set_ylabel('Quantity', color='blue')
ax2.set_ylabel('Cumulative Profit (%)', color='red')
ax1.grid(True, which='both', linestyle='--', linewidth=0.5)

# Adding legends
ax2.legend(loc='upper left')
ax1.legend(loc='upper right')

plt.tight_layout()
plt.show()
```



The Pareto Rule, 80% of Profit comes from 20% goods sold is not followed by this Superstore. So, The stroe needs to cut down on products with low quantity & profit for more efficient operations

Sales and Profit Margin by Segment

```
sales_profit_margin_segment = data.groupby('segment').agg({'sales':
'sum', 'profit_margin': 'mean'}).reset_index()

fig, ax1 = plt.subplots(figsize=(8, 6))
sales_profit_margin_segment.plot(kind='bar', x='segment', y='sales',
ax=ax1, color='blue', alpha=0.6, label='Sales')
ax2 = ax1.twinx()
sales_profit_margin_segment.plot(kind='line', x='segment',
y='profit_margin', ax=ax2, color='red', marker='o', label='Profit
Margin')

ax1.set_ylabel('Sales')
ax2.set_ylabel('Profit Margin')
plt.title('Sales and Profit Margin by Segment')
ax1.legend(loc='upper left')
ax2.legend(loc='upper right')
plt.tight_layout()
plt.show()
```



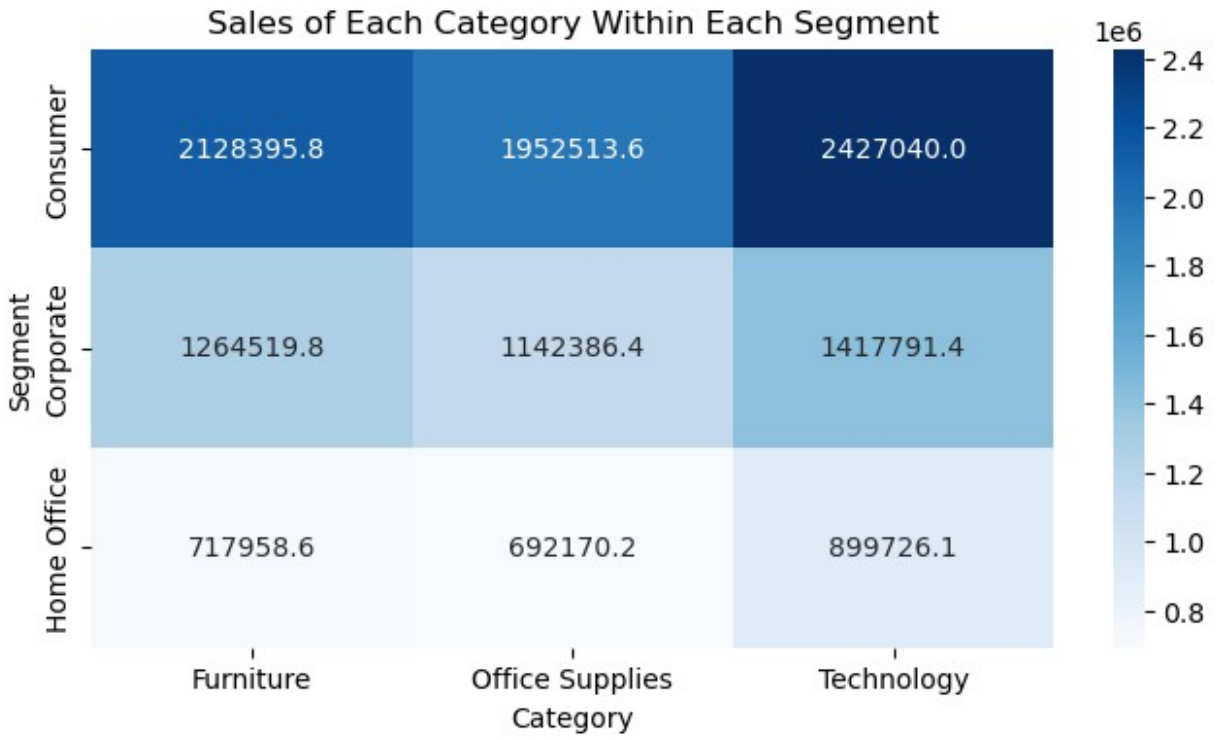
It is evident that Profit Margin of the Home Office(5.3%) Segment is the highest out of the 3 segments. It is to be noted that sales of Home Office is the lowest of all three.

Sales of Each Category Within Each Segment

```
pivot_table = data.pivot_table(index='segment', columns='category',
values='sales', aggfunc='sum')
print(pivot_table)
```

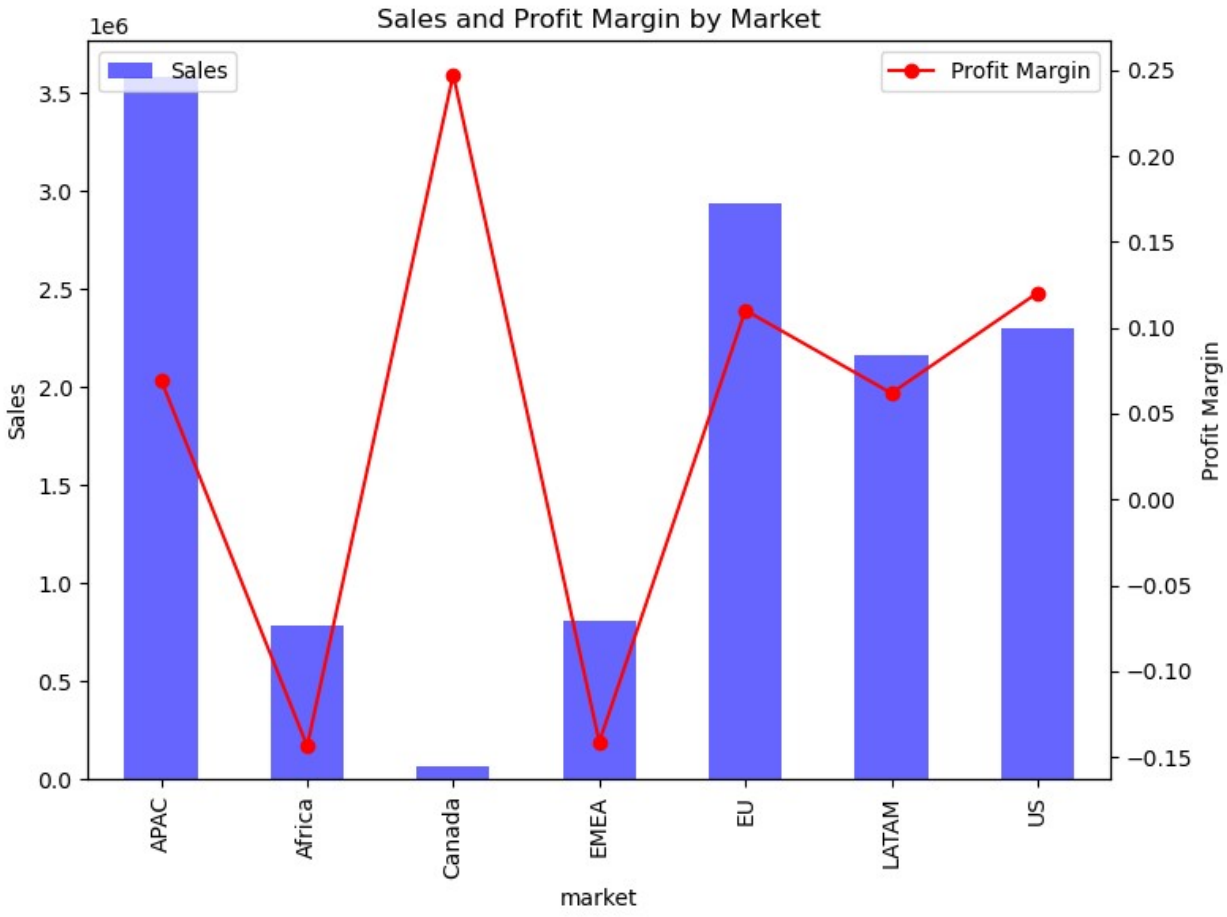
```
category        Furniture  Office Supplies    Technology
segment
Consumer     2.128396e+06     1.952514e+06  2.427040e+06
Corporate    1.264520e+06     1.142386e+06  1.417791e+06
Home Office  7.179586e+05     6.921702e+05  8.997261e+05
```

```
plt.figure(figsize=(8, 4))
sns.heatmap(pivot_table, annot=True, fmt='.1f', cmap='Blues')
plt.title('Sales of Each Category Within Each Segment')
plt.xlabel('Category')
plt.ylabel('Segment')
plt.show()
```



Sales and Profit Margin by Market

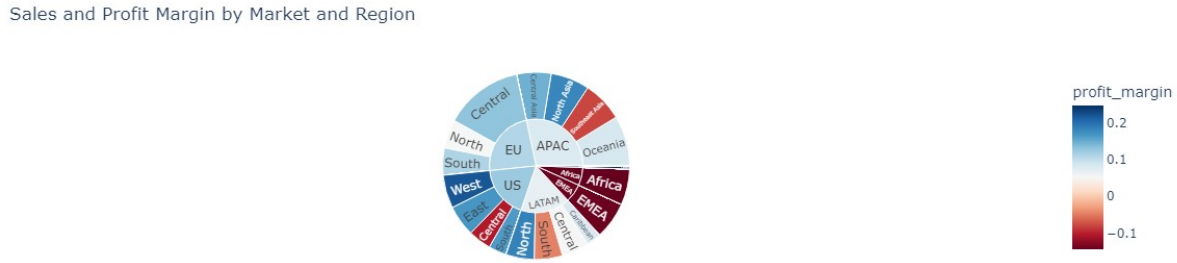
```
sales_profit_margin_market = data.groupby('market').agg({'sales':
'sum', 'profit_margin': 'mean'}).reset_index()

fig, ax1 = plt.subplots(figsize=(8, 6))
sales_profit_margin_market.plot(kind='bar', x='market', y='sales',
ax=ax1, color='blue', alpha=0.6, label='Sales')
ax2 = ax1.twinx()
sales_profit_margin_market.plot(kind='line', x='market',
y='profit_margin', ax=ax2, color='red', marker='o', label='Profit
Margin')

ax1.set_ylabel('Sales')
ax2.set_ylabel('Profit Margin')
plt.title('Sales and Profit Margin by Market')
ax1.legend(loc='upper left')
ax2.legend(loc='upper right')
plt.tight_layout()
plt.show()
```



The Canada Market's Sales is the lowest but has the highest Profit Margin(25%). Also we can see the Africa and EMEA Markets making negative Profit Margins.

Sales & Profit Margin Drill Down Through Region from Market

```
sunburst_market_region = data.groupby(['market',
'region']).agg({'sales': 'sum', 'profit_margin':
'mean'}).reset_index()
sunburst_market_region =
sunburst_market_region[sunburst_market_region['sales'] > 0]

fig = px.sunburst(
    sunburst_market_region,
    path=['market', 'region'],
    values='sales',
    color='profit_margin',
    color_continuous_scale='RdBu',
    title='Sales and Profit Margin by Market and Region'
)
fig.show()
```



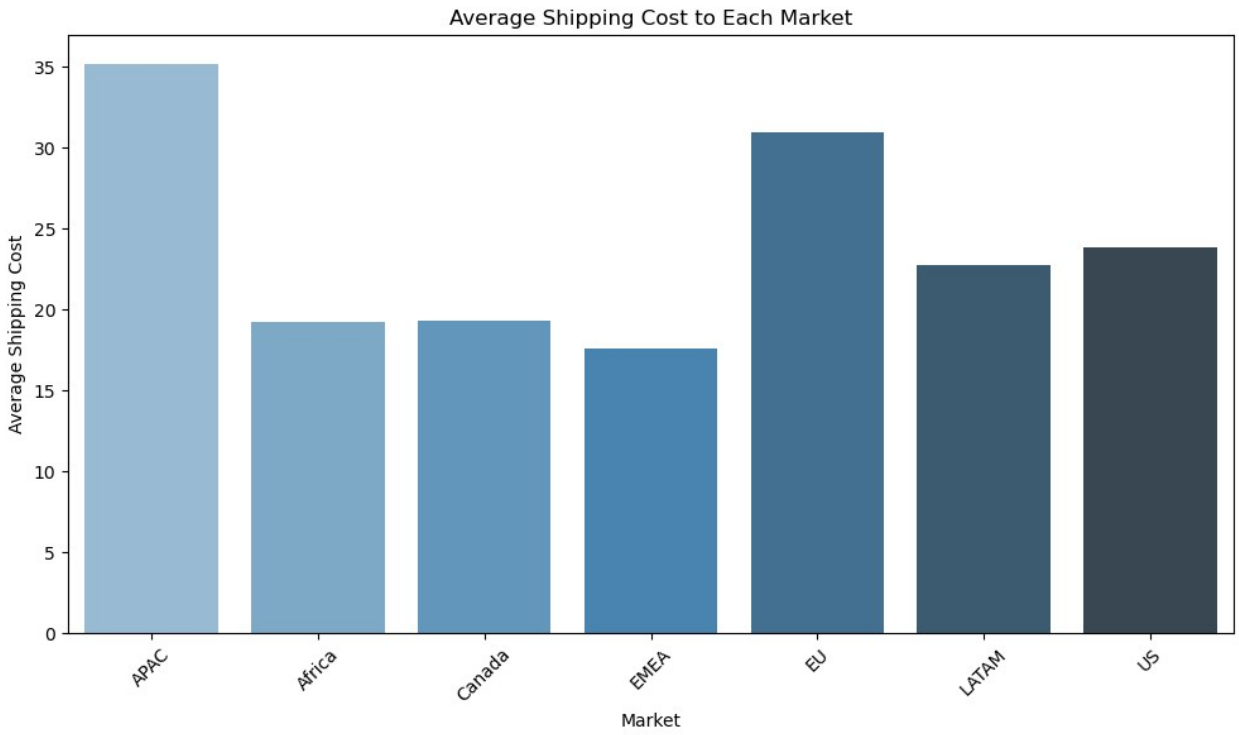
The above chart gives an idea of Sales(by size) and Profit Margin(by Color) for each Market drilling down through to Region

Average Shipping Cost to Each Market

```
avg_shipping_cost_market = data.groupby('market')
['shipping_cost'].mean().reset_index()
print(avg_shipping_cost_market)
```

```
   market  shipping_cost
0    APAC      35.190430
1  Africa      19.215058
2  Canada      19.285495
3    EMEA      17.573221
4      EU      30.942235
5   LATAM      22.745153
6      US      23.831678
```

```
plt.figure(figsize=(10, 6))
sns.barplot(data=avg_shipping_cost_market, x='market',
y='shipping_cost', palette='Blues_d')
plt.title('Average Shipping Cost to Each Market')
plt.xlabel('Market')
plt.ylabel('Average Shipping Cost')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



Average Shipping Time for Each Market

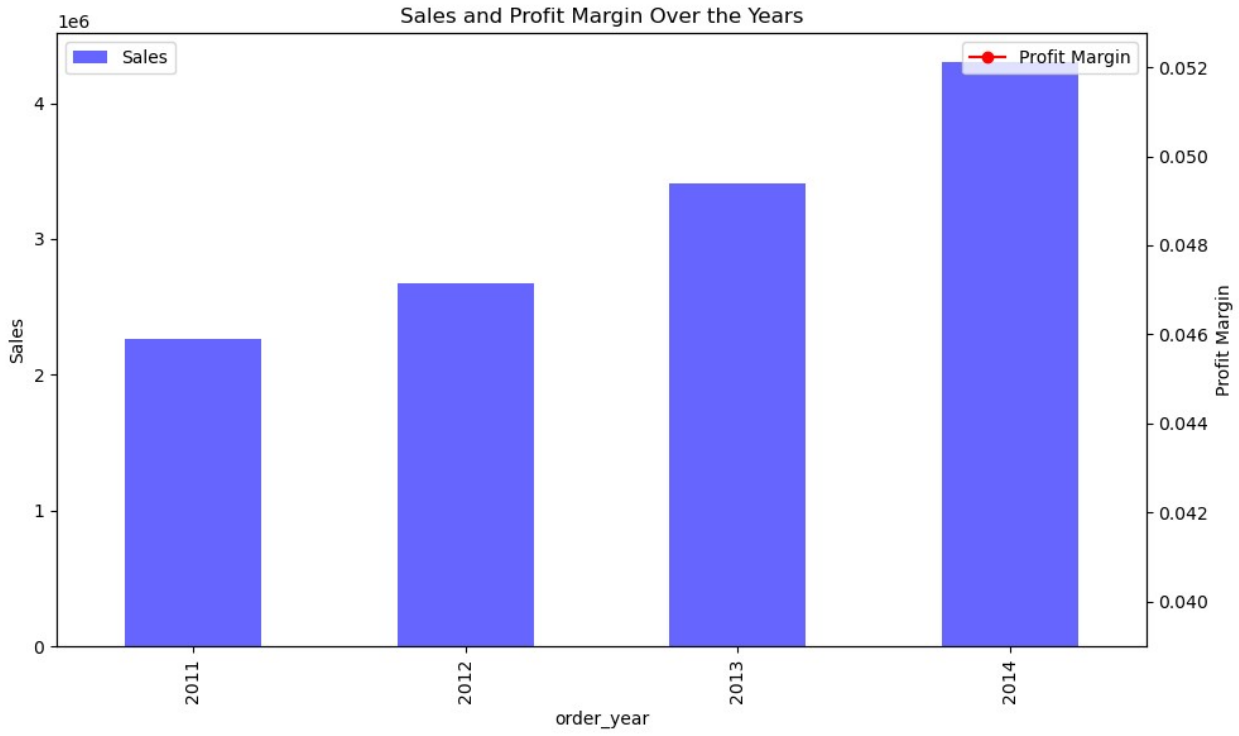
```
avg_shipping_time_market = data.groupby('market')
['shipping_time'].mean().reset_index()
print(avg_shipping_time_market)
```

```
   market  shipping_time
0    APAC       3.969097
1  Africa       3.910399
2  Canada       3.677083
3    EMEA       3.933386
4      EU       4.008300
5   LATAM       3.996794
6      US       3.958875
```

Sales & Profit Margin Over the years

```
data['order_year'] = data['order_date'].dt.year
sales_profit_yearly = data.groupby('order_year').agg({'sales': 'sum',
'profit_margin': 'mean'}).reset_index()

fig, ax1 = plt.subplots(figsize=(10, 6))
sales_profit_yearly.plot(kind='bar', x='order_year', y='sales',
ax=ax1, color='blue', alpha=0.6, label='Sales')
ax2 = ax1.twinx()
sales_profit_yearly.plot(kind='line', x='order_year',
y='profit_margin', ax=ax2, color='red', marker='o', label='Profit
Margin')
```

```
ax1.set_ylabel('Sales')
ax2.set_ylabel('Profit Margin')
plt.title('Sales and Profit Margin Over the Years')
ax1.legend(loc='upper left')
ax2.legend(loc='upper right')
plt.tight_layout()
plt.show()
```



Most Frequent Customers by Order Count

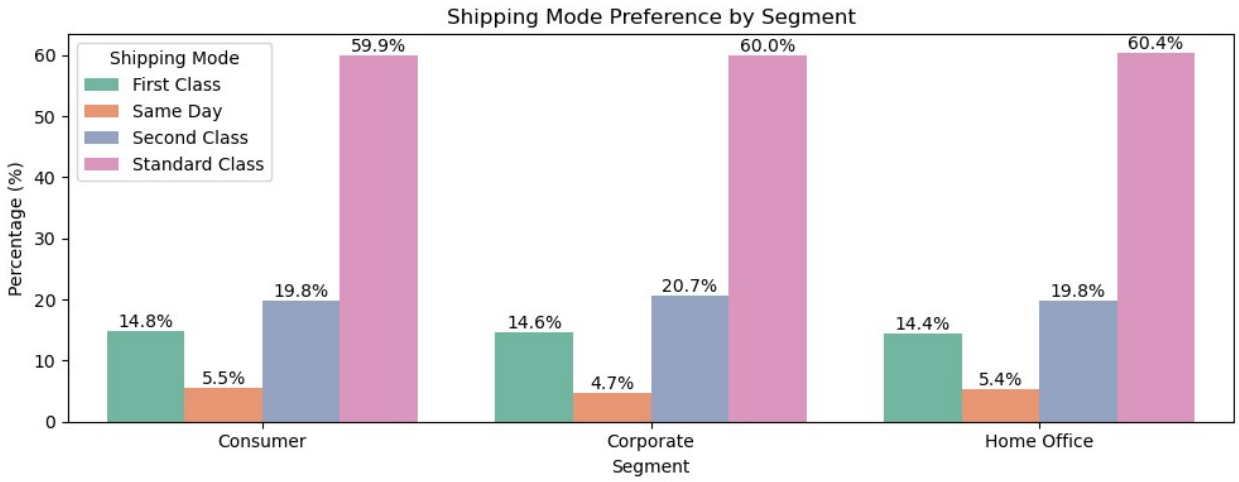
```
customer_order_count =
data['customer_name'].value_counts().reset_index()
customer_order_count.columns = ['customer_name', 'order_count']
most_frequent_customers = customer_order_count.head(10)

print(most_frequent_customers)
```

```
     customer_name  order_count
0  Muhammed Yedwab          108
1      Steven Ward          106
2       Gary Hwang          102
3  Patrick O'Brill          102
4      Bill Eplett          102
5     Harry Greene          101
6     Eric Murdock          100
7     Art Ferguson           98
```

```
8  Brosina Hoffman            97
9     Bart Watters            96
```

Shipping Mode Preference - by Segment

```
shipping_mode_segment = data.groupby(['segment',
'ship_mode']).size().reset_index(name='counts')

total_counts_segment = shipping_mode_segment.groupby('segment')
['counts'].sum().reset_index(name='total_counts')

shipping_mode_segment =
shipping_mode_segment.merge(total_counts_segment, on='segment')

shipping_mode_segment['percentage'] = (shipping_mode_segment['counts']
/ shipping_mode_segment['total_counts']) * 100

plt.figure(figsize=(10, 4))
bar_plot = sns.barplot(data=shipping_mode_segment, x='segment',
y='percentage', hue='ship_mode', palette='Set2')
plt.title('Shipping Mode Preference by Segment')
plt.xlabel('Segment')
plt.ylabel('Percentage (%)')
plt.legend(title='Shipping Mode')


for p in bar_plot.patches:
    height = p.get_height()
    bar_plot.annotate(f'{height:.1f}%', (p.get_x() + p.get_width() /
2., height),
                      ha='center', va='center', xytext=(0, 5),
textcoords='offset points')

plt.tight_layout()
plt.show()
```

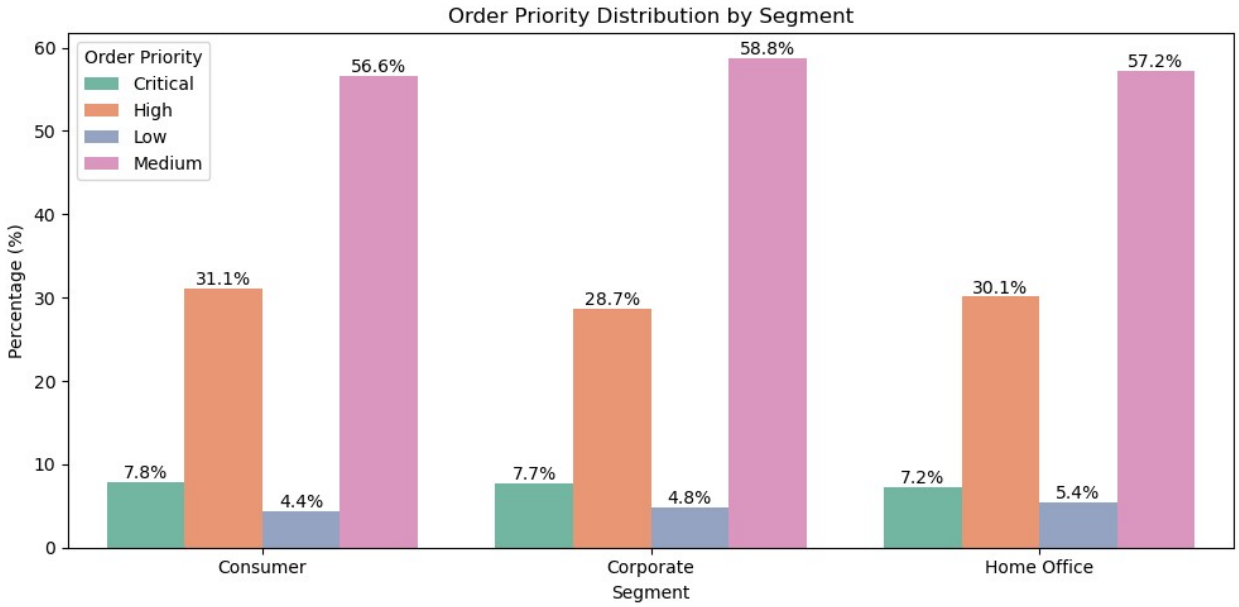


Across all segments, majority of customers prefer Standard Delivery

There is no significant difference between the preference of Delivery class among all segments

Order Priority Distribution - by Segment

```
order_priority_segment = data.groupby(['segment',
'order_priority']).size().reset_index(name='counts')

total_counts_segment = order_priority_segment.groupby('segment')
['counts'].sum().reset_index(name='total_counts')

order_priority_segment =
order_priority_segment.merge(total_counts_segment, on='segment')

order_priority_segment['percentage'] =
(order_priority_segment['counts'] /
order_priority_segment['total_counts']) * 100

plt.figure(figsize=(10, 5))
bar_plot = sns.barplot(data=order_priority_segment, x='segment',
y='percentage', hue='order_priority', palette='Set2')
plt.title('Order Priority Distribution by Segment')
plt.xlabel('Segment')
plt.ylabel('Percentage (%)')
plt.legend(title='Order Priority')

for p in bar_plot.patches:
    height = p.get_height()
    bar_plot.annotate(f'{height:.1f}%', (p.get_x() + p.get_width() /
2., height),
                      ha='center', va='center', xytext=(0, 5),
textcoords='offset points')

plt.tight_layout()
plt.show()
```

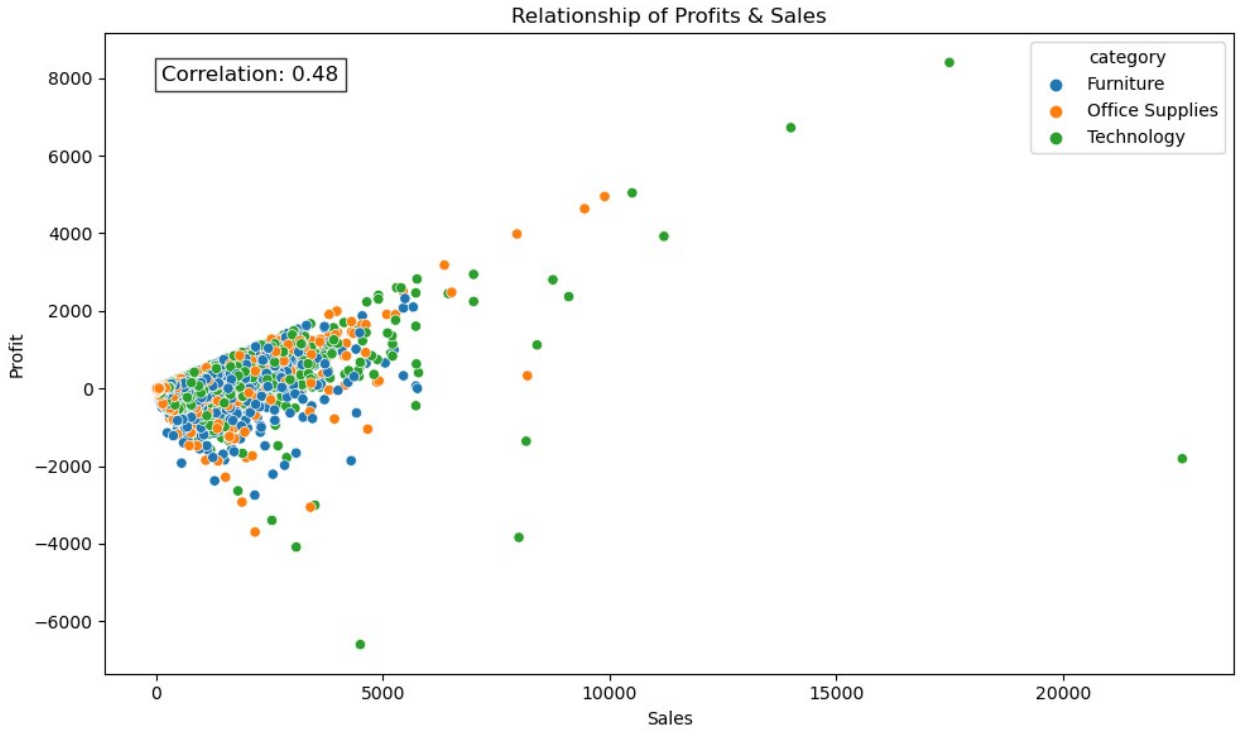


Across all segments, majority of Orders are of Medium Priority

There is no significant difference between the Priority of Order among all segments

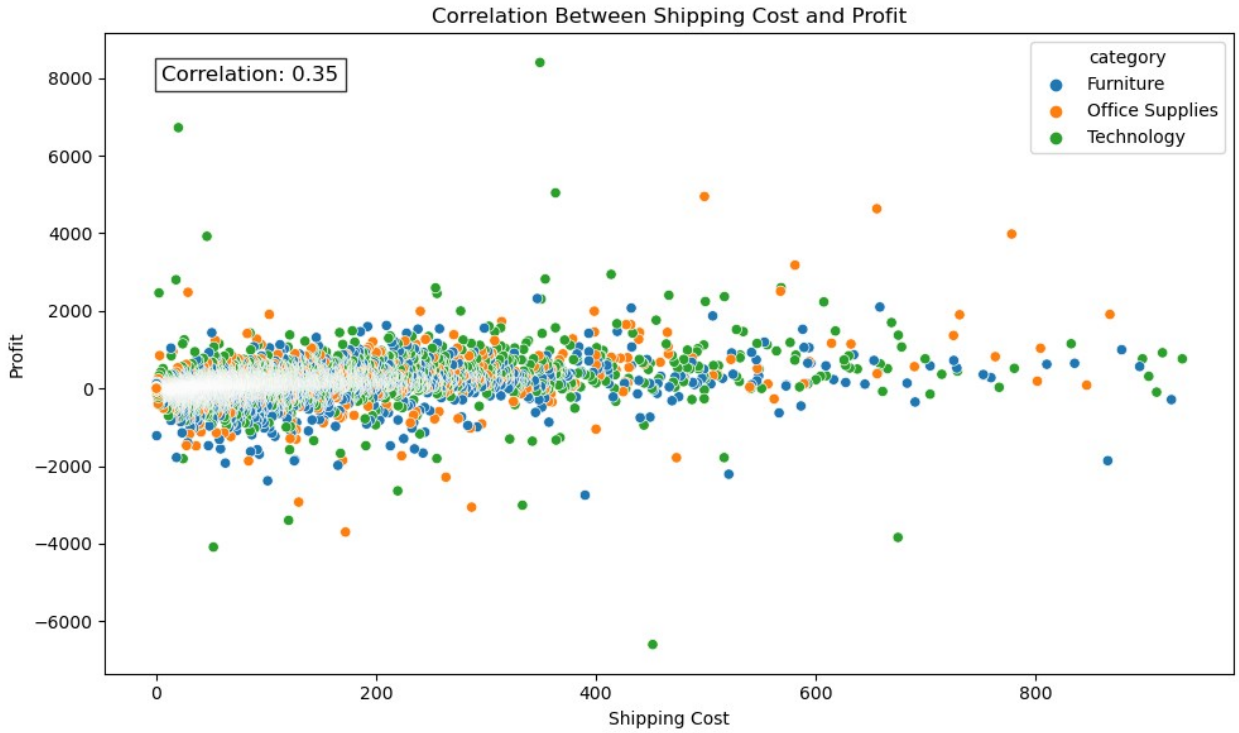
Relationship of Profits & Sales

```
correlation_coefficient = data['sales'].corr(data['profit'])

plt.figure(figsize=(10, 6))
sns.scatterplot(data=data, y = 'profit', x = 'sales', hue='category',
palette='tab10')
plt.title('Relationship of Profits & Sales')
plt.ylabel('Profit')
plt.xlabel('Sales')

plt.text(0.05, 0.95, f'Correlation: {correlation_coefficient:.2f}',
transform=plt.gca().transAxes,
         fontsize=12, verticalalignment='top',
bbox=dict(facecolor='white', alpha=0.8))

plt.tight_layout()
plt.show()
```



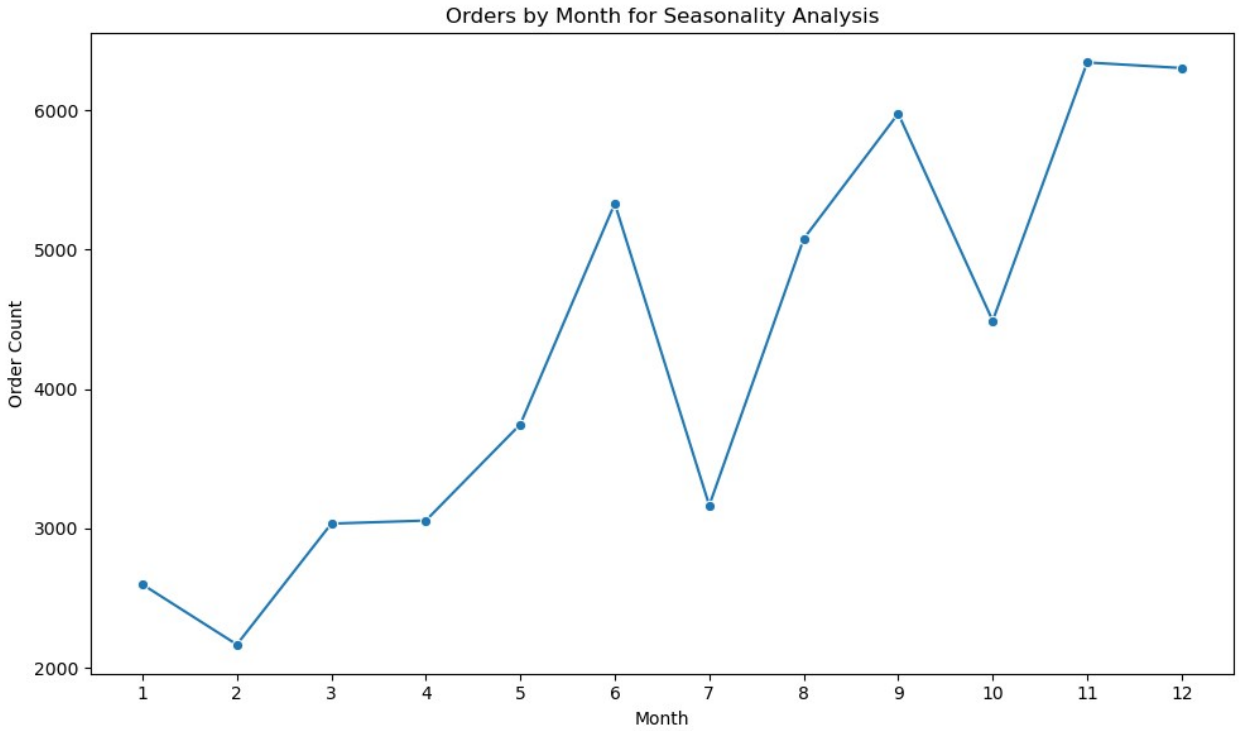
Correlation Between Shipping Cost and Profit

```
correlation_coefficient = data['shipping_cost'].corr(data['profit'])

plt.figure(figsize=(10, 6))
sns.scatterplot(data=data, x='shipping_cost', y='profit',
hue='category', palette='tab10')
plt.title('Correlation Between Shipping Cost and Profit')
plt.xlabel('Shipping Cost')
plt.ylabel('Profit')

plt.text(0.05, 0.95, f'Correlation: {correlation_coefficient:.2f}',
transform=plt.gca().transAxes,
         fontsize=12, verticalalignment='top',
bbox=dict(facecolor='white', alpha=0.8))

plt.tight_layout()
plt.show()
```

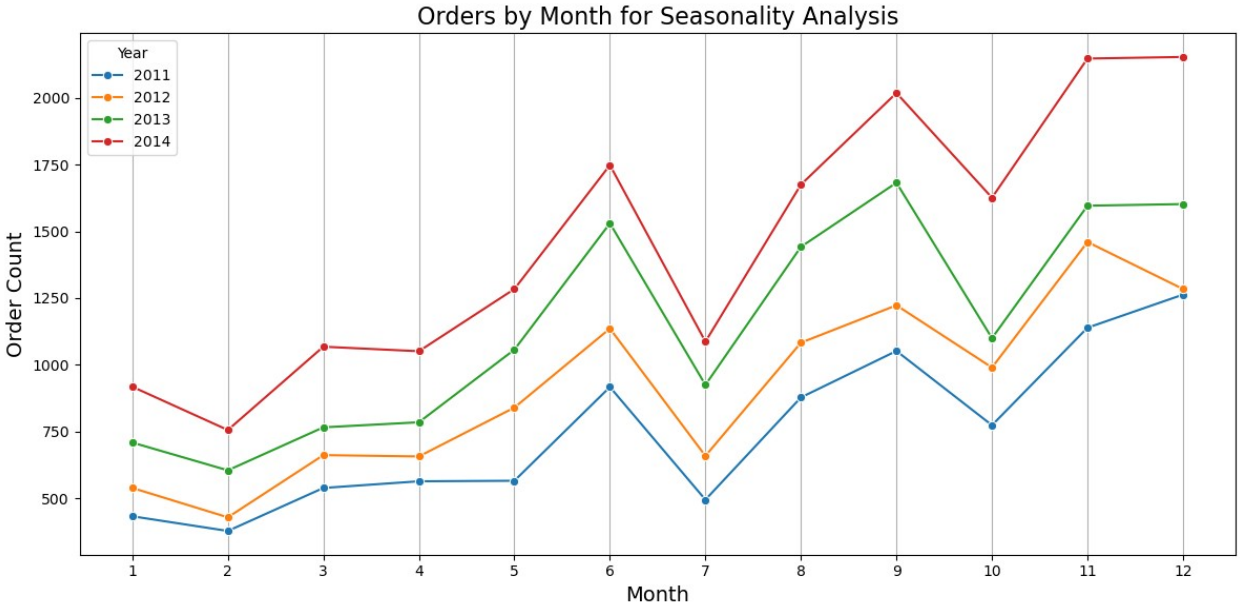


Orders by Month for Seasonality Analysis

```
data['order_month'] = data['order_date'].dt.month
monthly_orders = data.groupby('order_month')
['order_id'].count().reset_index()

plt.figure(figsize=(10, 6))
sns.lineplot(data=monthly_orders, x='order_month', y='order_id',
marker='o')
plt.title('Orders by Month for Seasonality Analysis')
plt.xlabel('Month')
plt.ylabel('Order Count')
plt.xticks(range(1, 13))
plt.tight_layout()
plt.show()
```




```
monthly_orders = data.groupby(['order_year', 'order_month'])
['order_id'].count().reset_index()

plt.figure(figsize=(12, 6))

for year in monthly_orders['order_year'].unique():
    year_data = monthly_orders[monthly_orders['order_year'] == year]
    sns.lineplot(data=year_data, x='order_month', y='order_id',
marker='o', label=year)

plt.title('Orders by Month for Seasonality Analysis', fontsize=16)
plt.xlabel('Month', fontsize=14)
plt.ylabel('Order Count', fontsize=14)
plt.xticks(range(1, 13))
plt.legend(title='Year')
plt.grid(axis='x')  # Adding horizontal grid lines
plt.tight_layout()
plt.show()
```


Orders by Month for Seasonality Analysis
Year
2011
2012
2013
2014
Order Count
2000
1750
1500
1250
1000
750
500
1
2
3
4
5
6
7
8
9
10
11
12
Month