MINI PROJECT TO CLEAN AND BUILD A DASH BOARD

In [113]:
```
#The aim  of this project is to clean and preprocess the data, uncover the insights in
```

In [114]:
```
# importing the data to notebook
import pandas as pd
import numpy as np
import os
import matplotlib.pyplot as plt
import seaborn as sns
```

In [115]:
```
df = pd.read_csv(r"C:\Users\USER\Documents\WORKSPACE\Visualization\Data\superstore_sale
```

In [116]:
```
df
```

Out[116]:

	order_id	order_date	ship_date	ship_mode	customer_name	segment	state	country	market	
0	AG-2011-2040	1/1/2011	1/6/2011	Standard Class	Toby Braunhardt	Consumer	Constantine	Algeria	Africa	
1	IN-2011-47883	1/1/2011	1/8/2011	Standard Class	Joseph Holt	Consumer	New South Wales	Australia	APAC	C
2	HU-2011-1220	1/1/2011	1/5/2011	Second Class	Annie Thurman	Consumer	Budapest	Hungary	EMEA	
3	IT-2011-3647632	1/1/2011	1/5/2011	Second Class	Eugene Moren	Home Office	Stockholm	Sweden	EU	
4	IN-2011-47883	1/1/2011	1/8/2011	Standard Class	Joseph Holt	Consumer	New South Wales	Australia	APAC	C
...	...	...	...	...	...	...	...	...	...	
51399	CA-2014-115427	12/31/2014	1/4/2015	Standard Class	Erica Bern	Corporate	California	United States	US	
51400	MO-2014-2560	12/31/2014	1/5/2015	Standard Class	Liz Preis	Consumer	Souss-Massa-Draâ	Morocco	Africa	
51401	MX-2014-110527	12/31/2014	1/2/2015	Second Class	Charlotte Melton	Consumer	Managua	Nicaragua	LATAM	
51402	MX-2014-114783	12/31/2014	1/6/2015	Standard Class	Tamara Dahlen	Consumer	Chihuahua	Mexico	LATAM	
51403	CA-2014-156720	12/31/2014	1/4/2015	Standard Class	Jill Matthias	Consumer	Colorado	United States	US	

51404 rows × 21 columns

In [117]:
```
#inspecting the structure and properties of our data .
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 51404 entries, 0 to 51403
Data columns (total 21 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   order_id        51404 non-null  object 
 1   order_date      51404 non-null  object 
 2   ship_date       51404 non-null  object 
 3   ship_mode       51404 non-null  object 
 4   customer_name   51404 non-null  object 
 5   segment         51404 non-null  object 
 6   state           51404 non-null  object 
 7   country         51404 non-null  object 
 8   market          51404 non-null  object 
 9   region          51404 non-null  object 
 10  product_id      51404 non-null  object 
 11  category        51404 non-null  object 
 12  sub_category    51404 non-null  object 
 13  product_name    51404 non-null  object 
 14  sales           51399 non-null  object 
 15  quantity        51401 non-null  float64
 16  discount        51400 non-null  float64
 17  profit          51401 non-null  float64
 18  shipping_cost   51402 non-null  float64
 19  order_priority  51404 non-null  object 
 20  year            51404 non-null  int64  
dtypes: float64(4), int64(1), object(16)
memory usage: 8.2+ MB
```

In [118]:
```
# To get the list of my columns to know what is going on in my data
df.columns
```

Out[118]:
```
Index(['order_id', 'order_date', 'ship_date', 'ship_mode', 'customer_name',
       'segment', 'state', 'country', 'market', 'region', 'product_id',
       'category', 'sub_category', 'product_name', 'sales', 'quantity',
       'discount', 'profit', 'shipping_cost', 'order_priority', 'year'],
      dtype='object')
```

In [119]:
```
# Getting quantity columns and checking the nature of the data in that column
df['quantity']
```

Out[119]:
```
0        2.0
1        3.0
2        4.0
3        3.0
4        5.0
        ...
51399    2.0
51400    1.0
51401    3.0
51402    1.0
51403    3.0
Name: quantity, Length: 51404, dtype: float64
```

In [120]:
```
# Getting discountcolumns and checking the nature of the data in that column
df['discount']
```

Out[120]:
```
0        0.0
1        0.1
2        0.0
3        0.5
4        0.1
        ...
51399    0.2
51400    0.0
51401    0.0
51402    0.0
51403    0.2
Name: discount, Length: 51404, dtype: float64
```

In [121]:
```
# Getting year columns and checking the nature of the data in that column
df['year']
```

Out[121]:
```
0        2011
1        2011
2        2011
3        2011
4        2011
         ...
51399    2014
51400    2014
51401    2014
51402    2014
51403    2014
Name: year, Length: 51404, dtype: int64
```

In [122]:
```
# Inspecting the column for their unique values
df['profit']
```

Out[122]:
```
0        106.1400
1         36.0360
2         29.6400
3        -26.0550
4         37.7700
           ...
51399      4.5188
51400      0.4200
51401     12.3600
51402      0.5600
51403     -0.6048
Name: profit, Length: 51404, dtype: float64
```

In [158]:
```
#checking for unique value for sales
df['sales'].unique()
```

Out[158]:
```
array([ 408.,  120.,   66., ..., 1763., 1821., 1831.])
```

In [125]:
```
# converting the sales colounm from strings to float
```

In [126]:
```
df.sales.unique()
```

Out[126]:
```
array(['408', '120', '66', ..., '1,763', '1,821', '1,831'], dtype=object)
```

In [127]:
```
df['sales'] = df['sales'].str.replace(',', '').astype(float)
```

In [128]:
```
# cheaking to see that the sales column is converted to flaot
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 51404 entries, 0 to 51403
Data columns (total 21 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   order_id        51404 non-null  object 
 1   order_date      51404 non-null  object 
 2   ship_date       51404 non-null  object 
 3   ship_mode       51404 non-null  object 
 4   customer_name   51404 non-null  object 
 5   segment         51404 non-null  object 
 6   state           51404 non-null  object 
 7   country         51404 non-null  object 
 8   market          51404 non-null  object 
 9   region          51404 non-null  object 
 10  product_id      51404 non-null  object 
 11  category        51404 non-null  object 
 12  sub_category    51404 non-null  object 
 13  product_name    51404 non-null  object 
 14  sales           51399 non-null  float64
 15  quantity        51401 non-null  float64
 16  discount        51400 non-null  float64
 17  profit          51401 non-null  float64
 18  shipping_cost   51402 non-null  float64
 19  order_priority  51404 non-null  object 
 20  year            51404 non-null  int64  
dtypes: float64(5), int64(1), object(15)
memory usage: 8.2+ MB
```

In [129]:
```
#checking for missing values
df.isna().sum()
```

Out[129]:
```
order_id          0
order_date        0
ship_date         0
ship_mode         0
customer_name     0
segment           0
state             0
country           0
market            0
region            0
product_id        0
category          0
sub_category      0
product_name      0
sales             5
quantity          3
discount          4
profit            3
shipping_cost     2
order_priority    0
year              0
dtype: int64
```

In [130]:
```
#droping the missing values because it will affect the outcome of my work
df=df.dropna()
```

In [131]:
```
df.isna().sum()
```

Out[131]:
```
order_id          0
order_date        0
ship_date         0
ship_mode         0
customer_name     0
segment           0
state             0
country           0
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
year              0
dtype: int64
```

In [132]:
```
#checking for duplicate
df.duplicated().sum()
```

Out[132]:
```
114
```

In [133]:
```
#droping duplicate
df = df.drop_duplicates()
```

In [136]:
```
#Grouping the data for total sales
df["Total_sales"]= df["sales"]*df["quantity"]
```

In [137]:
```
#checking to see if the total sales have been added to the coloum
print(df.columns)
```

```
Index(['order_id', 'order_date', 'ship_date', 'ship_mode', 'customer_name',
       'segment', 'state', 'country', 'market', 'region', 'product_id',
       'category', 'sub_category', 'product_name', 'sales', 'quantity',
       'discount', 'profit', 'shipping_cost', 'order_priority', 'year',
       'Total_sales'],
      dtype='object')
```

In [138]:
```
#Grouping the total sales by years
sales_df = df.groupby("year")["Total_sales"].sum().reset_index()
profit_df= df.groupby("year")["profit"].sum().reset_index()
```

In [139]:
```
1 df.head()
```

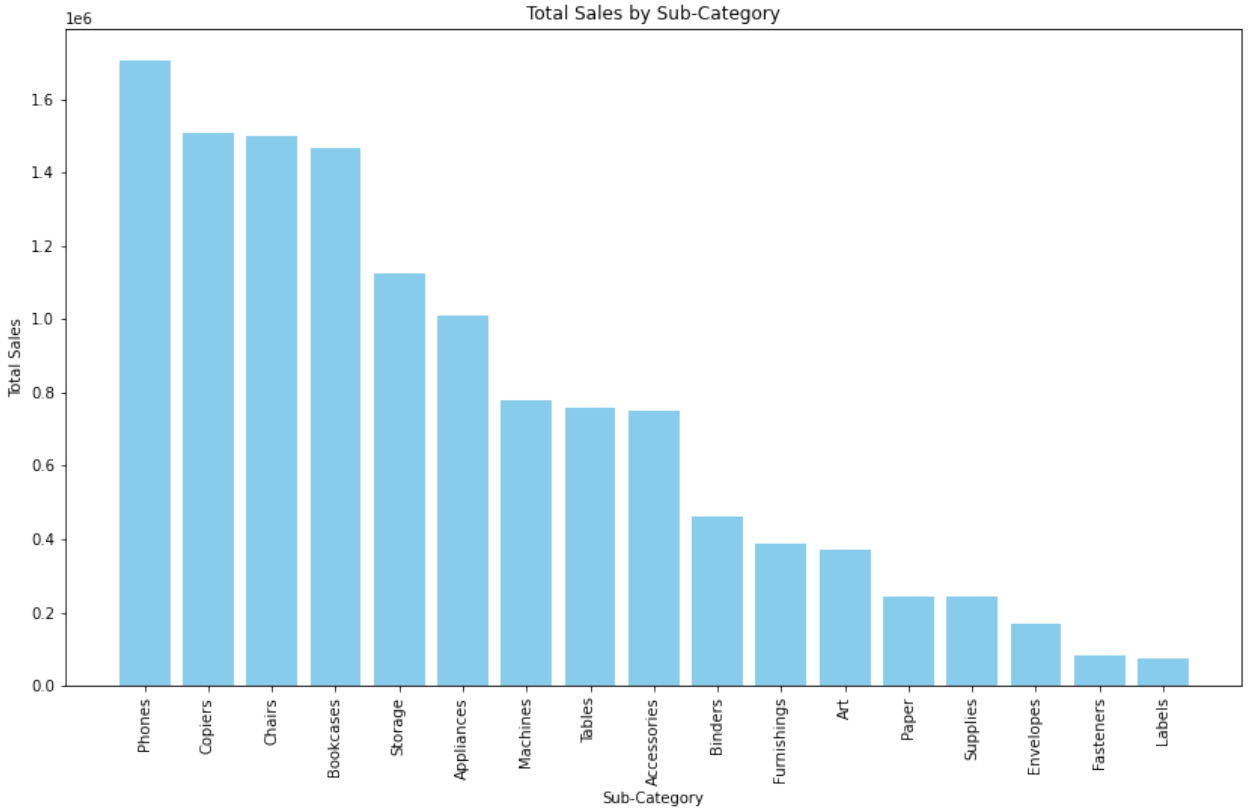
Out[139]:

	order_id	order_date	ship_date	ship_mode	customer_name	segment	state	country	market	regio
0	AG-2011-2040	1/1/2011	1/6/2011	Standard Class	Toby Braunhardt	Consumer	Constantine	Algeria	Africa	Afric
1	IN-2011-47883	1/1/2011	1/8/2011	Standard Class	Joseph Holt	Consumer	New South Wales	Australia	APAC	Oceani
2	HU-2011-1220	1/1/2011	1/5/2011	Second Class	Annie Thurman	Consumer	Budapest	Hungary	EMEA	EME
3	IT-2011-3647632	1/1/2011	1/5/2011	Second Class	Eugene Moren	Home Office	Stockholm	Sweden	EU	Nort
4	IN-2011-47883	1/1/2011	1/8/2011	Standard Class	Joseph Holt	Consumer	New South Wales	Australia	APAC	Oceani

5 rows × 22 columns

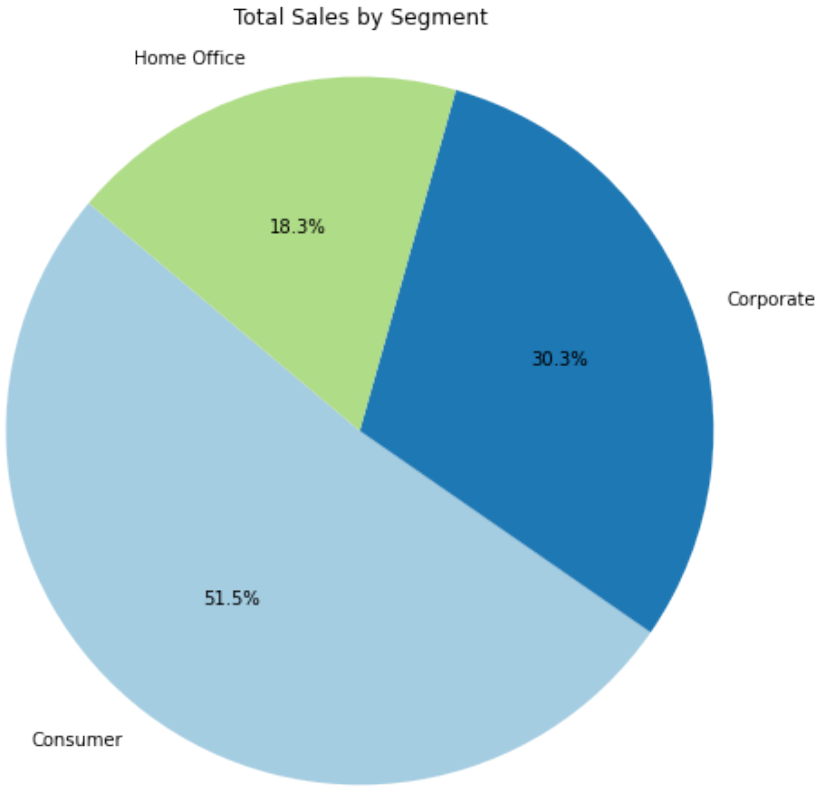
In [27]:
```
1 # Group by sub-category and sum the sales
2 total_sales_by_sub_category = df.groupby('sub_category')['sales'].sum().reset_index()
3
4 # Sort the data by total sales for better visualization
5 total_sales_by_sub_category = total_sales_by_sub_category.sort_values(by='sales', asce
```

In [140]:
```
# ploting the total sales by sub-category
plt.figure(figsize=(14, 8))
plt.bar(total_sales_by_sub_category['sub_category'], total_sales_by_sub_category['sales
plt.title('Total Sales by Sub-Category')
plt.xlabel('Sub-Category')
plt.ylabel('Total Sales')
plt.xticks(rotation=90)
plt.show()
```



In [141]:
```
# Group by segment and sum the sales
total_sales_by_segment = df.groupby('segment')['sales'].sum().reset_index()
```

In [142]:
```
# Ploting the data
plt.figure(figsize=(10, 8))
plt.pie(total_sales_by_segment['sales'], labels=total_sales_by_segment['segment'], aut
plt.title('Total Sales by Segment')
plt.axis('equal')
plt.show()
```



In [143]:
```
# Aggregate sales data by month
monthly_sales = df.groupby('year')['sales'].sum().reset_index()
```

In [144]:
```
# Ploting the data
plt.figure(figsize=(12, 6))
plt.plot(monthly_sales['year'], monthly_sales['sales'], marker='o', linestyle='-', col
plt.title('Sales by Month')
plt.xlabel('Month')
plt.ylabel('Sales')
plt.xticks(rotation=45)
plt.grid(True)
plt.tight_layout()
plt.show()
```

In [145]:
```
# Convert the order_date column to datetime
df['order_date'] = pd.to_datetime(df['order_date'])
```

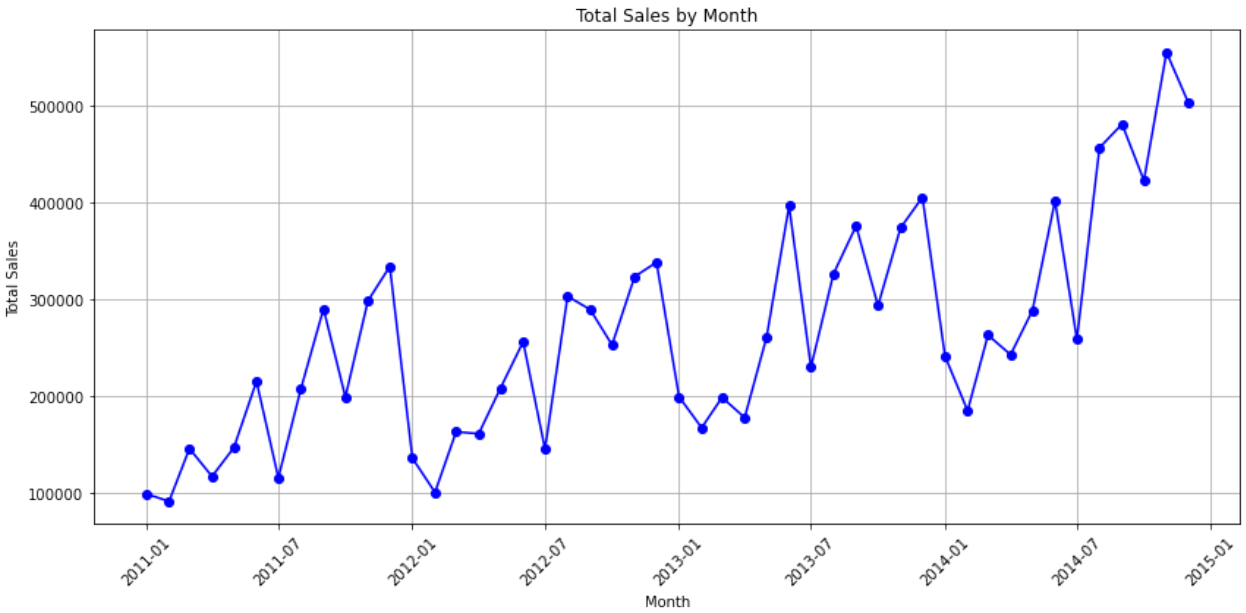
In [146]:
```
# Extract the month and year from the date column
df['year_month'] = df['order_date'].dt.to_period('M')
```

In [147]:
```
# Aggregate sales data by month
monthly_sales = df.groupby('year_month')['sales'].sum().reset_index()
```

In [148]:
```
# Convert the 'year_month' to a datetime object for plotting
monthly_sales['year_month'] = monthly_sales['year_month'].dt.to_timestamp()
```

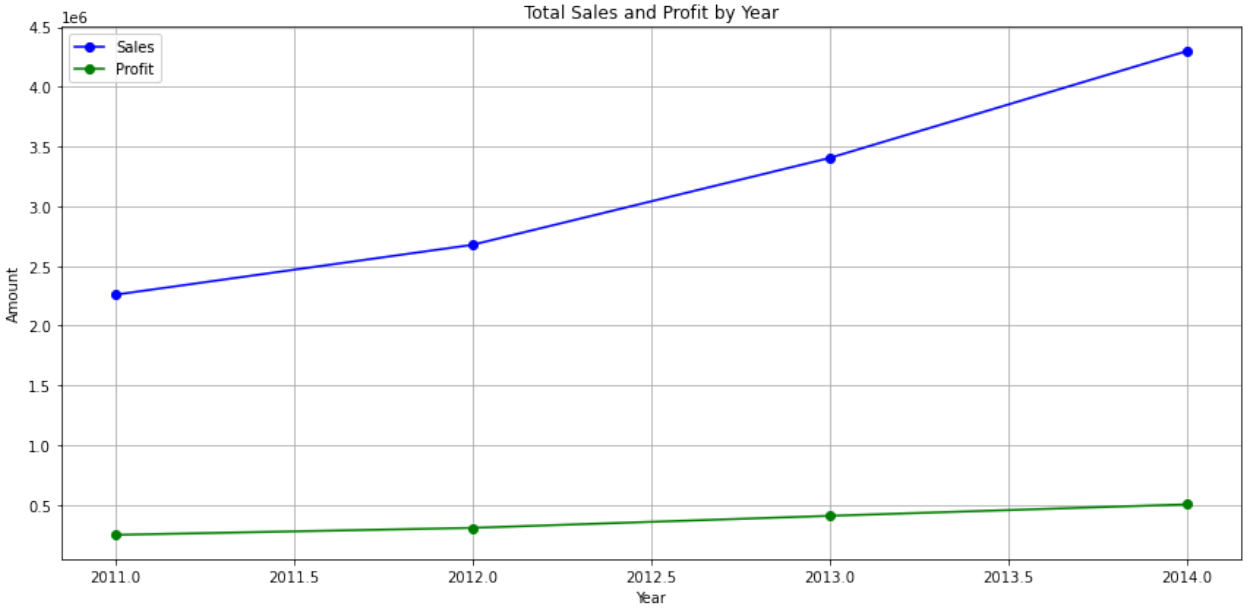
In [149]:

```
# Ploting the data
plt.figure(figsize=(12, 6))
plt.plot(monthly_sales['year_month'], monthly_sales['sales'], marker='o', linestyle='-
plt.title('Total Sales by Month')
plt.xlabel('Month')
plt.ylabel('Total Sales')
plt.xticks(rotation=45)
plt.grid(True)
plt.tight_layout()
plt.show()
```



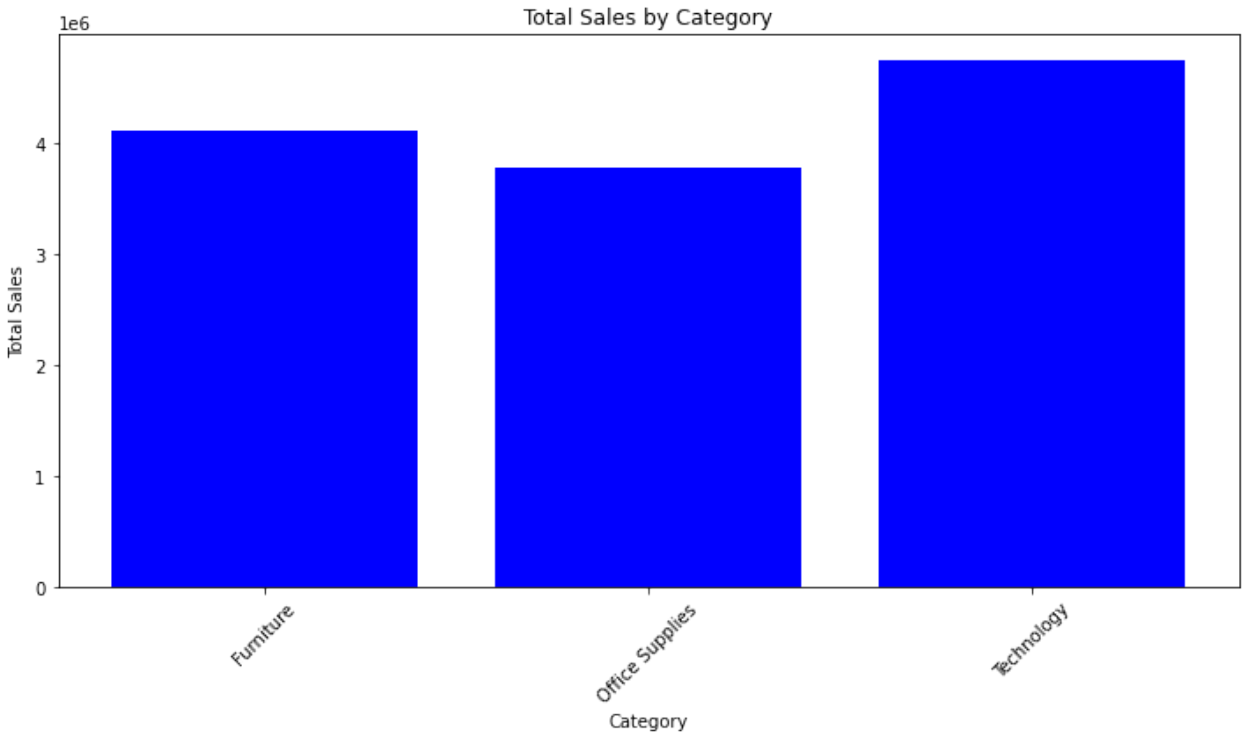
In [150]:

```
# Convert the order_date column to datetime
df['order_date'] = pd.to_datetime(df['order_date'])

# Extract the year from the date column
df['year'] = df['order_date'].dt.year

# Aggregate sales and profit data by year
yearly_data = df.groupby('year')[['sales', 'profit']].sum().reset_index()

# Plot the data
plt.figure(figsize=(12, 6))
plt.plot(yearly_data['year'], yearly_data['sales'], marker='o', linestyle='-', color='
plt.plot(yearly_data['year'], yearly_data['profit'], marker='o', linestyle='-', color=
plt.title('Total Sales and Profit by Year')
plt.xlabel('Year')
plt.ylabel('Amount')
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```



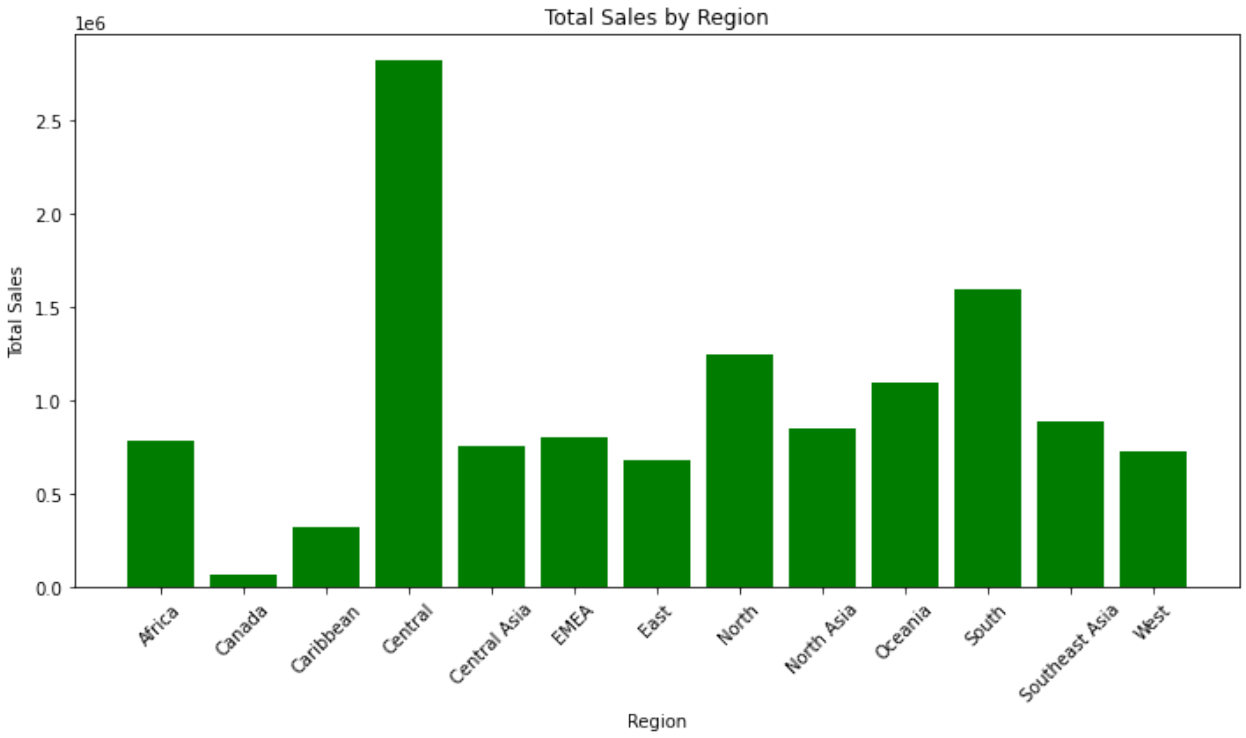
In [151]:

```
#Aggregate total sales by category
category_sales = df.groupby('category')['sales'].sum().reset_index()

# Ploting the data
plt.figure(figsize=(10, 6))
plt.bar(category_sales['category'], category_sales['sales'], color='blue')
plt.title('Total Sales by Category')
plt.xlabel('Category')
plt.ylabel('Total Sales')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



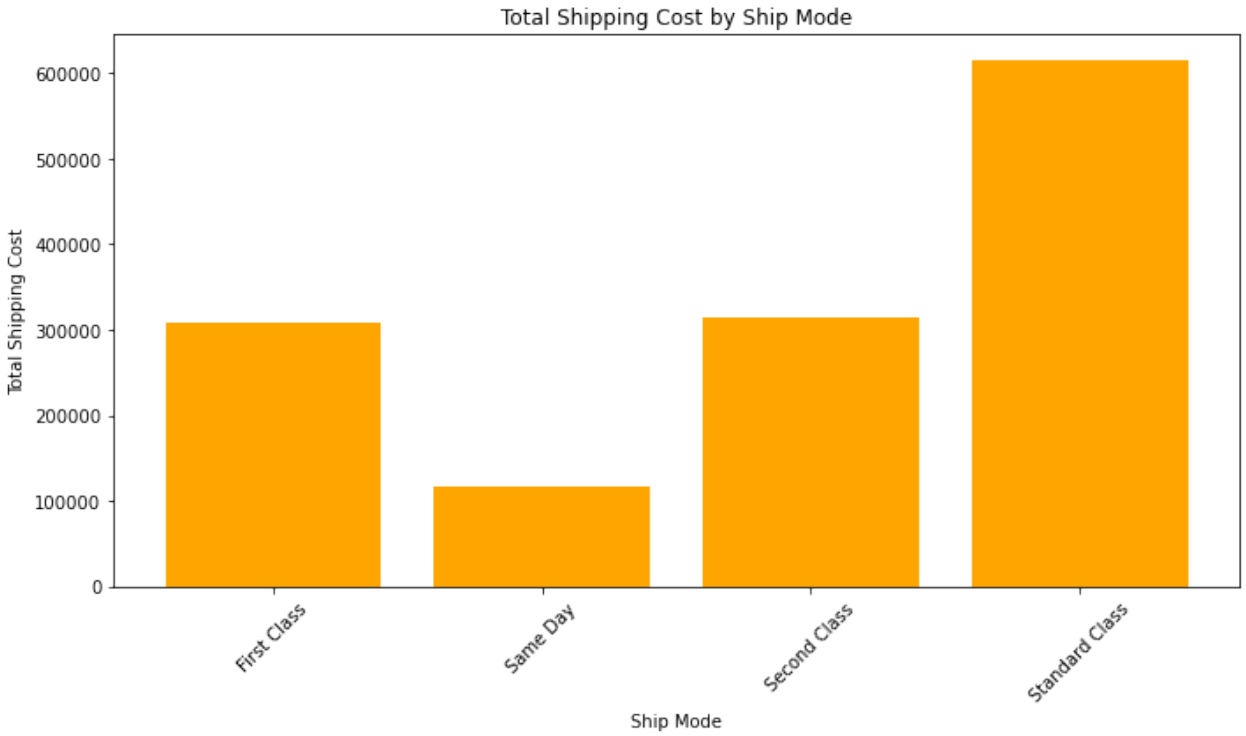
In [152]:

```
# Aggregate total sales by region
region_sales = df.groupby('region')['sales'].sum().reset_index()

# Plot the data
plt.figure(figsize=(10, 6))
plt.bar(region_sales['region'], region_sales['sales'], color='green')
plt.title('Total Sales by Region')
plt.xlabel('Region')
plt.ylabel('Total Sales')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

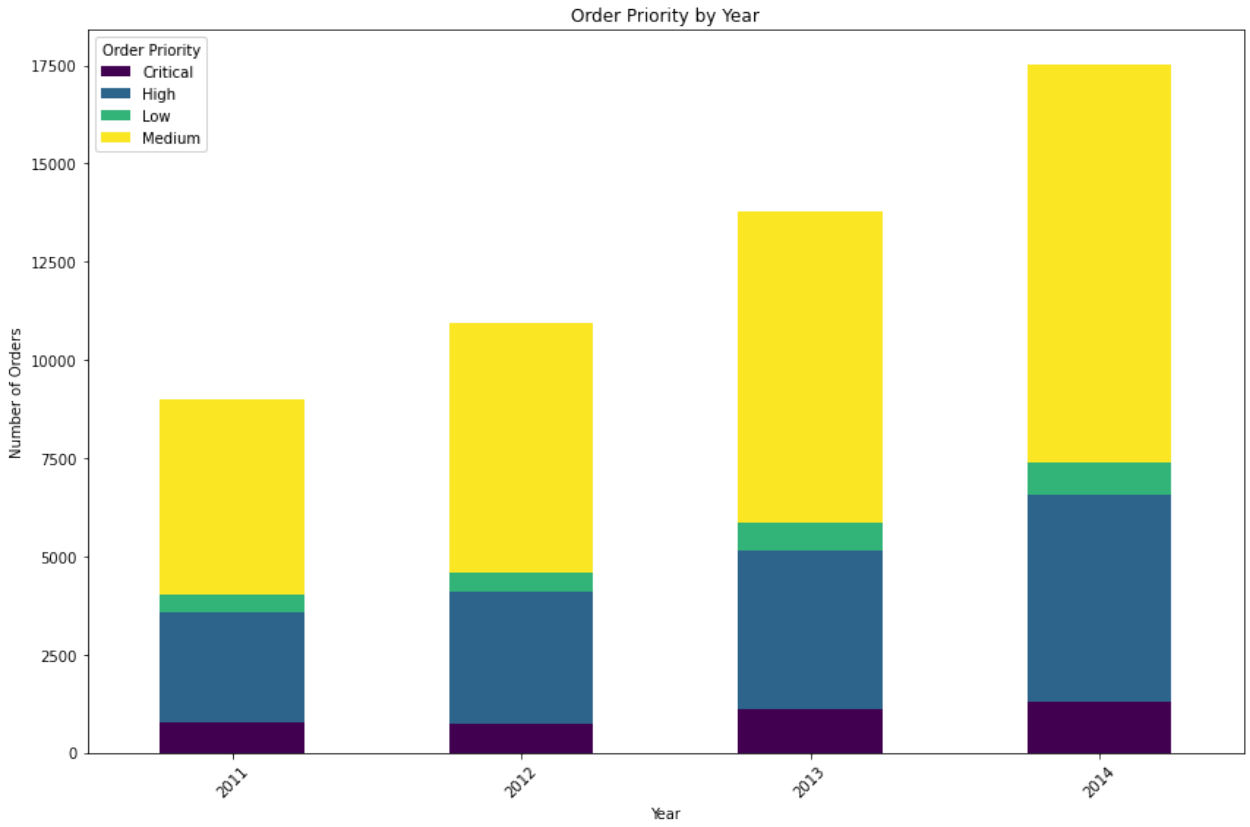


In [153]:

```
# Aggregate total shipping cost by ship mode
ship_mode_cost = df.groupby('ship_mode')['shipping_cost'].sum().reset_index()

# Plot the data
plt.figure(figsize=(10, 6))
plt.bar(ship_mode_cost['ship_mode'], ship_mode_cost['shipping_cost'], color='orange')
plt.title('Total Shipping Cost by Ship Mode')
plt.xlabel('Ship Mode')
plt.ylabel('Total Shipping Cost')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



In [154]:
```
# Aggregate the count of orders by year and order priority
order_priority_by_year = df.groupby(['year', 'order_priority']).size().unstack(fill_va

# Plot the data
order_priority_by_year.plot(kind='bar', stacked=True, figsize=(12, 8), colormap='virid
plt.title('Order Priority by Year')
plt.xlabel('Year')
plt.ylabel('Number of Orders')
plt.legend(title='Order Priority')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

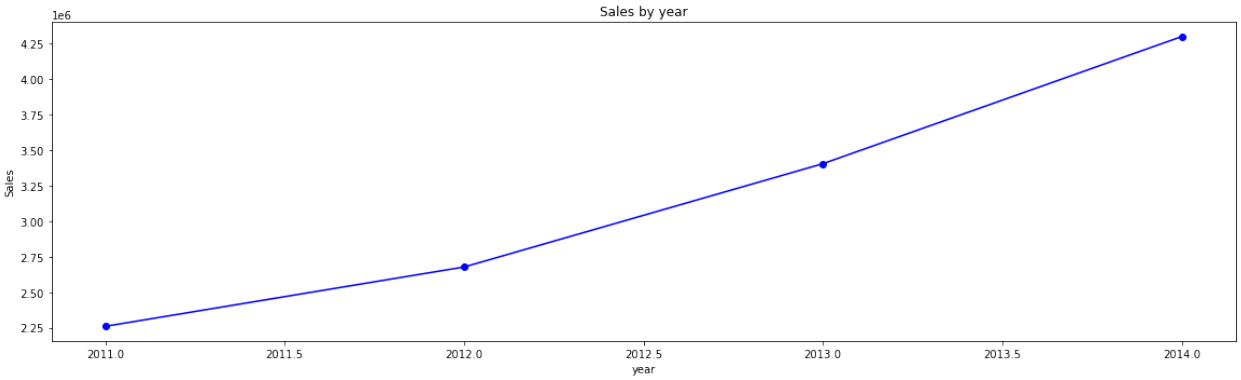


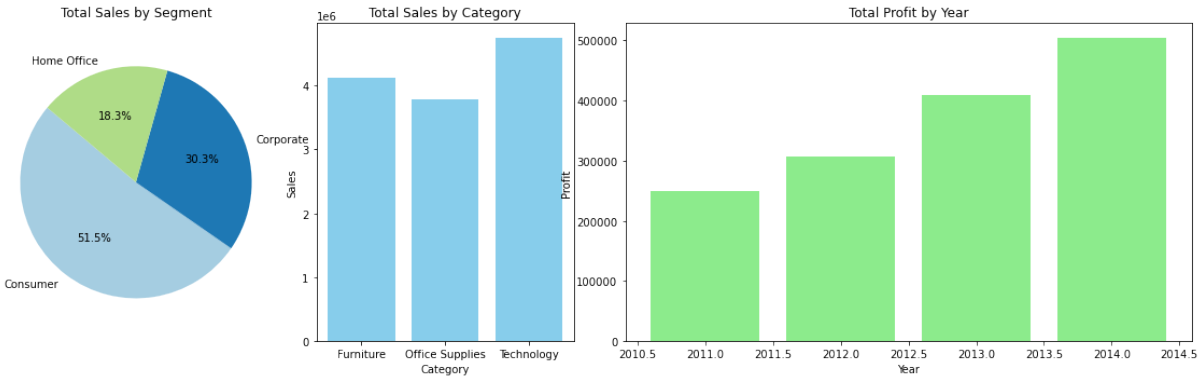
In [ ]:
```
# CREATING DASHBOARD
```

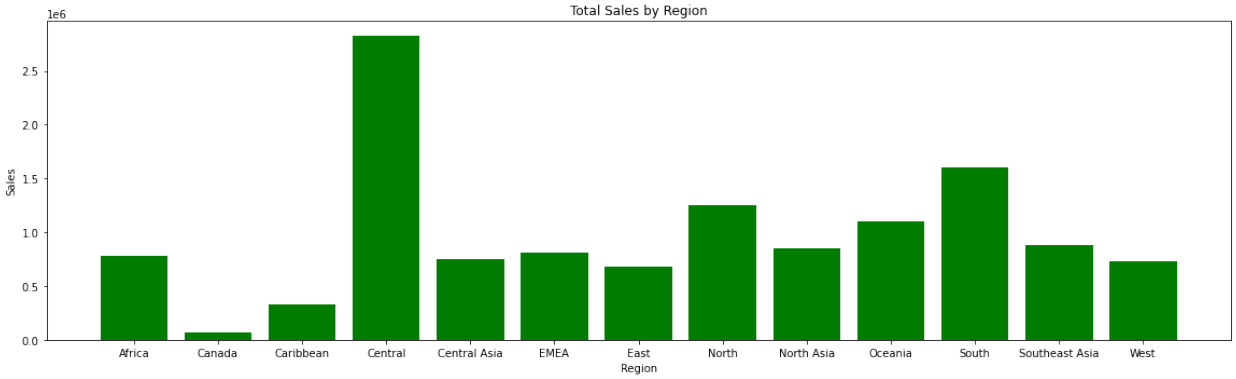
In [156]:

```
fig = plt.figure(figsize=(30,25))

ax = [None for _ in range(8)]

ax[0] = plt.subplot2grid((4,6), (0,0), colspan=4)
ax[1] = plt.subplot2grid((4,6), (1,0), colspan=1)
ax[2] = plt.subplot2grid((4 ,6), (1,1), colspan=1)
ax[3] = plt.subplot2grid((4,6), (1,2), colspan=2)
ax[4] = plt.subplot2grid((4,6), (2,3), colspan=3)
ax[5] = plt.subplot2grid((4,6), (2,0), colspan=4)



# to load content into the content
sales_by_month = df.groupby('year')['sales'].sum().reset_index()
ax[0].plot(sales_by_month['year'], sales_by_month['sales'], marker='o', color='b')
ax[0].set_title('Sales by year')
ax[0].set_xlabel('year')
ax[0].set_ylabel('Sales')


ax[1].pie(total_sales_by_segment['sales'], labels=total_sales_by_segment['segment'], a
ax[1].set_title('Total Sales by Segment')
ax[1].axis('equal')

ax[2].bar(total_sales_by_category['category'], total_sales_by_category['sales'], color
ax[2].set_title('Total Sales by Category')
ax[2].set_xlabel('Category')
ax[2].set_ylabel('Sales')

ax[3].bar(profit_by_year['year'], profit_by_year['profit'], color='lightgreen')
ax[3].set_title('Total Profit by Year')
ax[3].set_xlabel('Year')
ax[3].set_ylabel('Profit')

ax[5].bar(total_sales_by_region['region'], total_sales_by_region['sales'], color='gree
ax[5].set_title('Total Sales by Region')
ax[5].set_xlabel('Region')
ax[5].set_ylabel('Sales')



```

Out[156]: Text(0, 0.5, 'Sales')


Sales by year
1e6
Sales
year


Total Sales by Segment
Home Office
18.3%
30.3%
51.5%
Consumer
Total Sales by Category
1e6
Corporate
Sales
Furniture
Office Supplies
Technology
Category
Total Profit by Year
Profit
Year


Total Sales by Region
1e6
Sales
Africa
Canada
Caribbean
Central
Central Asia
EMEA
East
North
North Asia
Oceania
South
Southeast Asia
West
Region

In [ ]:
```
1
```