superstore_sales_analysis

```
# The dataset contains 5901 rows and 23 columns, with details about
sales transactions. Here's a brief summary of the columns:
```

```
**Key Columns**:
1.Order Date, Ship Date: Dates related to order and shipping.
2.Ship Mode: Shipping method used.
3.Customer Name, Customer ID: Details about customers.
4.Segment: Customer segment (e.g., Corporate, Consumer).
5.City, State, Region: Geographic details.
6.Category, Sub-Category, Product Name: Product-related details.
7.Sales, Profit, Quantity: Financial and order metrics.
8.Returns: Return status (has many missing values).
9.Payment Mode: Payment method.
10.Columns like ind1 and ind2 are completely empty and irrelevant.
```

Step 1: Load and Inspect the Dataset

In [2]:
```python
import pandas as pd
```

```
C:\Users\priya\anaconda3\Lib\site-packages\pandas\core\arrays\masked.py:
60: UserWarning: Pandas requires version '1.3.6' or newer of 'bottlenec
k' (version '1.3.5' currently installed).
  from pandas.core import (
```

In [3]:
```python
# Load the dataset
file_path = "SuperStore_Sales_Dataset.csv"  # Replace with your file path
data = pd.read_csv(file_path)
```

In [4]:
```
# Display the first few rows and basic info
print(data.head())
print(data.info())
```

```
   Row ID+O6G3A1:R6        Order ID  Order Date   Ship Date       Ship M
ode  \
0              4918  CA-2019-160304  01-01-2019  07-01-2019  Standard Cl
ass
1              4919  CA-2019-160304  02-01-2019  07-01-2019  Standard Cl
ass
2              4920  CA-2019-160304  02-01-2019  07-01-2019  Standard Cl
ass
3              3074  CA-2019-125206  03-01-2019  05-01-2019     First Cl
ass
4              8604  US-2019-116365  03-01-2019  08-01-2019  Standard Cl
ass

  Customer ID      Customer Name    Segment        Country          City
...  \
0    BM-11575      Brendan Murry  Corporate  United States  Gaithersburg
...
1    BM-11575      Brendan Murry  Corporate  United States  Gaithersburg
...
2    BM-11575      Brendan Murry  Corporate  United States  Gaithersburg
...
3    LR-16915       Lena Radford   Consumer  United States   Los Angeles
...
4    CA-12310  Christine Abelman  Corporate  United States   San Antonio
...

          Category Sub-Category  \
0        Furniture    Bookcases
1        Furniture    Bookcases
2       Technology       Phones
3  Office Supplies      Storage
4       Technology  Accessories

                                        Product Name   Sales Quantity  \
0  Bush Westfield Collection Bookcases, Medium Ch...   73.94        1
1  Bush Westfield Collection Bookcases, Medium Ch...  173.94        3
2                                         GE 30522EE2  231.98        2
3  Recycled Steel Personal File for Hanging File ...  114.46        2
4                Imation Clip USB flash drive - 8 GB   30.08        2

    Profit  Returns  Payment Mode  ind1  ind2
0  28.2668      NaN        Online   NaN   NaN
1  38.2668      NaN        Online   NaN   NaN
2  67.2742      NaN         Cards   NaN   NaN
3  28.6150      NaN        Online   NaN   NaN
4  -5.2640      NaN        Online   NaN   NaN

[5 rows x 23 columns]
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5901 entries, 0 to 5900
Data columns (total 23 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   Row ID+O6G3A1:R6  5901 non-null   int64
 1   Order ID          5901 non-null   object
 2   Order Date        5901 non-null   object
 3   Ship Date         5901 non-null   object
 4   Ship Mode         5901 non-null   object
 5   Customer ID       5901 non-null   object
 6   Customer Name     5901 non-null   object
 7   Segment           5901 non-null   object
```

```
 8   Country          5901 non-null   object
 9   City             5901 non-null   object
 10  State            5901 non-null   object
 11  Region           5901 non-null   object
 12  Product ID       5901 non-null   object
 13  Category         5901 non-null   object
 14  Sub-Category     5901 non-null   object
 15  Product Name     5901 non-null   object
 16  Sales            5901 non-null   float64
 17  Quantity         5901 non-null   int64
 18  Profit           5901 non-null   float64
 19  Returns          287 non-null    float64
 20  Payment Mode     5901 non-null   object
 21  ind1             0 non-null      float64
 22  ind2             0 non-null      float64
dtypes: float64(5), int64(2), object(16)
memory usage: 1.0+ MB
None
```

Step 2: Data Cleaning

1.Drop irrelevant or empty columns. 2.Convert dates to datetime format. 3.Handle missing values in the "Returns" column.

In [5]:
```
# Drop irrelevant or completely empty columns
data = data.drop(columns=['ind1', 'ind2', 'Row ID+O6G3A1:R6'], errors='igr
```

In [6]:
```
# Convert date columns to datetime
data['Order Date'] = pd.to_datetime(data['Order Date'], format='%d-%m-%Y',
data['Ship Date'] = pd.to_datetime(data['Ship Date'], format='%d-%m-%Y', e
```

In [7]:
```
# Fill missing values in 'Returns' with 0 (indicating no returns)
data['Returns'] = data['Returns'].fillna(0)
```

In [8]:
```python
# Verify cleaning
print(data.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5901 entries, 0 to 5900
Data columns (total 20 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Order ID        5901 non-null   object
 1   Order Date      5901 non-null   datetime64[ns]
 2   Ship Date       5901 non-null   datetime64[ns]
 3   Ship Mode       5901 non-null   object
 4   Customer ID     5901 non-null   object
 5   Customer Name   5901 non-null   object
 6   Segment         5901 non-null   object
 7   Country         5901 non-null   object
 8   City            5901 non-null   object
 9   State           5901 non-null   object
 10  Region          5901 non-null   object
 11  Product ID      5901 non-null   object
 12  Category        5901 non-null   object
 13  Sub-Category    5901 non-null   object
 14  Product Name    5901 non-null   object
 15  Sales           5901 non-null   float64
 16  Quantity        5901 non-null   int64
 17  Profit          5901 non-null   float64
 18  Returns         5901 non-null   float64
 19  Payment Mode    5901 non-null   object
dtypes: datetime64[ns](2), float64(3), int64(1), object(14)
memory usage: 922.2+ KB
None
```

Step 3: Summary Statistics

Analyze the dataset for key metrics such as sales, profit, and quantity.

In [9]:
```python
# Summary statistics for numerical columns
print(data[['Sales', 'Profit', 'Quantity']].describe())
```

```
             Sales       Profit     Quantity
count  5901.000000  5901.000000  5901.000000
mean    265.345589    29.700408     3.781901
std     474.260645   259.589138     2.212917
min       0.836000 -6599.978000     1.000000
25%      71.976000     1.795500     2.000000
50%     128.648000     8.502500     3.000000
75%     265.170000    28.615000     5.000000
max    9099.930000  8399.976000    14.000000
```

Step 4: Sales and Profit Trends Over Time

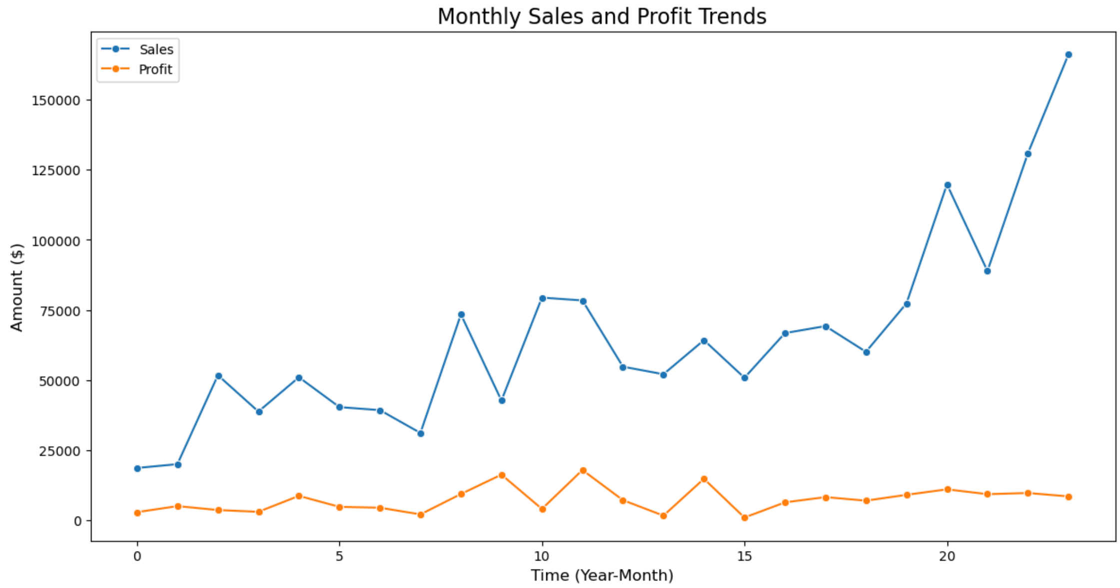
In [10]:
```
# Extract Year and Month
data['Year'] = data['Order Date'].dt.year
data['Month'] = data['Order Date'].dt.month
```

In [11]:
```
# Group by year and month
sales_trend = data.groupby(['Year', 'Month']).agg({'Sales': 'sum', 'Profit
```

In [12]:
```
# Visualize the trends
import matplotlib.pyplot as plt
import seaborn as sns
```

In [13]:
```
plt.figure(figsize=(14, 7))
sns.lineplot(x=sales_trend.index, y='Sales', data=sales_trend, label='Sale
sns.lineplot(x=sales_trend.index, y='Profit', data=sales_trend, label='Pro
plt.title("Monthly Sales and Profit Trends", fontsize=16)
plt.xlabel("Time (Year-Month)", fontsize=12)
plt.ylabel("Amount ($)", fontsize=12)
plt.legend()
plt.show()
```

```
C:\Users\priya\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: Fut
ureWarning: use_inf_as_na option is deprecated and will be removed in a
future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\priya\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: Fut
ureWarning: use_inf_as_na option is deprecated and will be removed in a
future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\priya\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: Fut
ureWarning: use_inf_as_na option is deprecated and will be removed in a
future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\priya\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: Fut
ureWarning: use_inf_as_na option is deprecated and will be removed in a
future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```

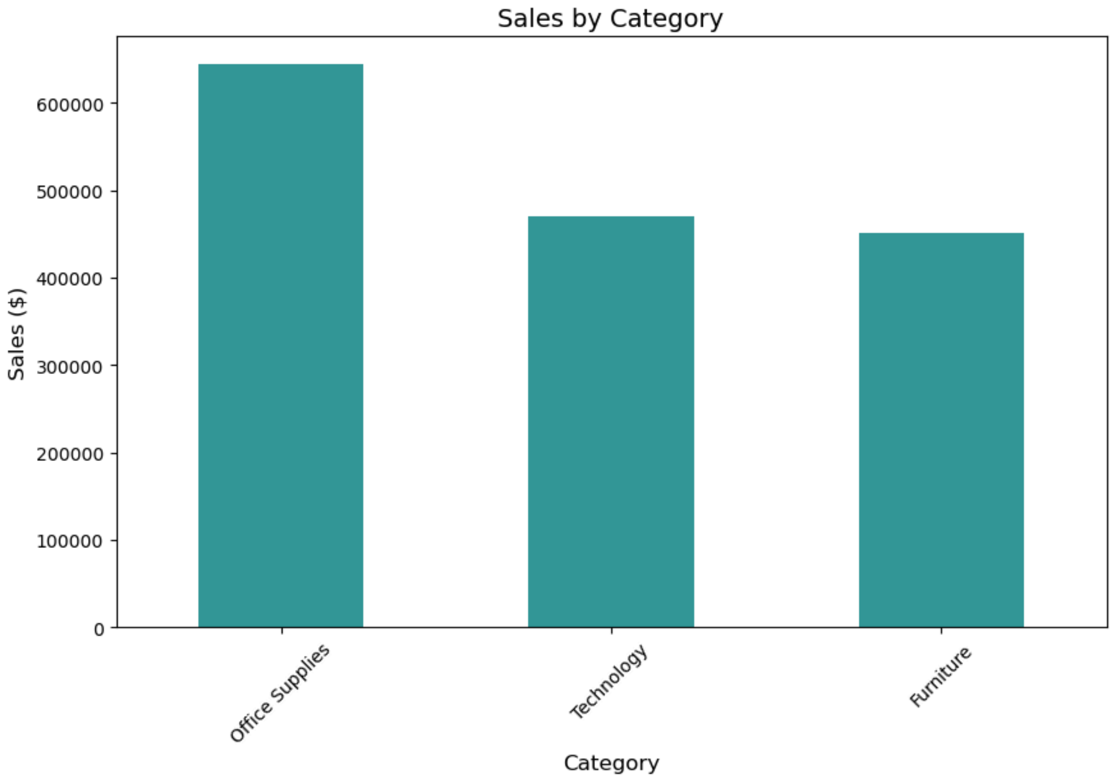


Step 5: Category and Regional Analysis

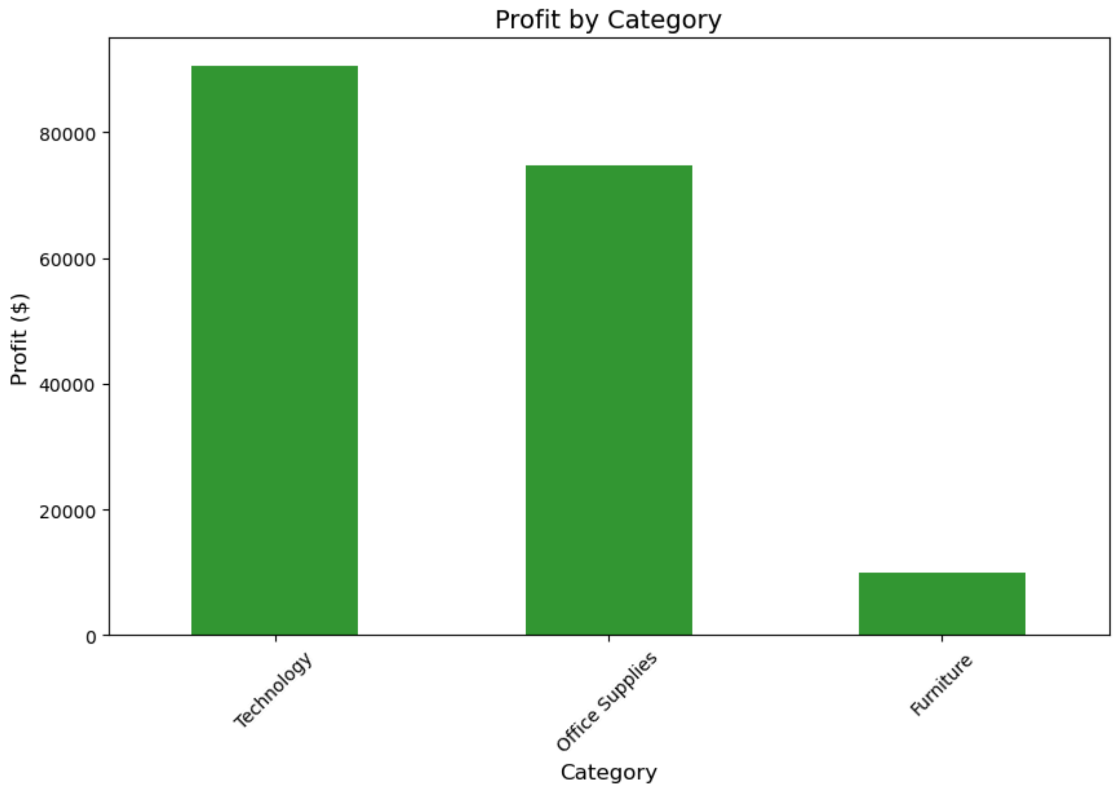
1.Analyze sales and profit by category and region. 2.Visualize the performance of different regions and categories.

In [14]:
```
# Sales and profit by category
category_sales = data.groupby('Category')['Sales'].sum().sort_values(ascen
category_profit = data.groupby('Category')['Profit'].sum().sort_values(asc
```

In [15]:
```
# Visualize category performance
plt.figure(figsize=(10, 6))
category_sales.plot(kind='bar', color='teal', alpha=0.8)
plt.title("Sales by Category", fontsize=14)
plt.xlabel("Category", fontsize=12)
plt.ylabel("Sales ($)", fontsize=12)
plt.xticks(rotation=45)
plt.show()
```

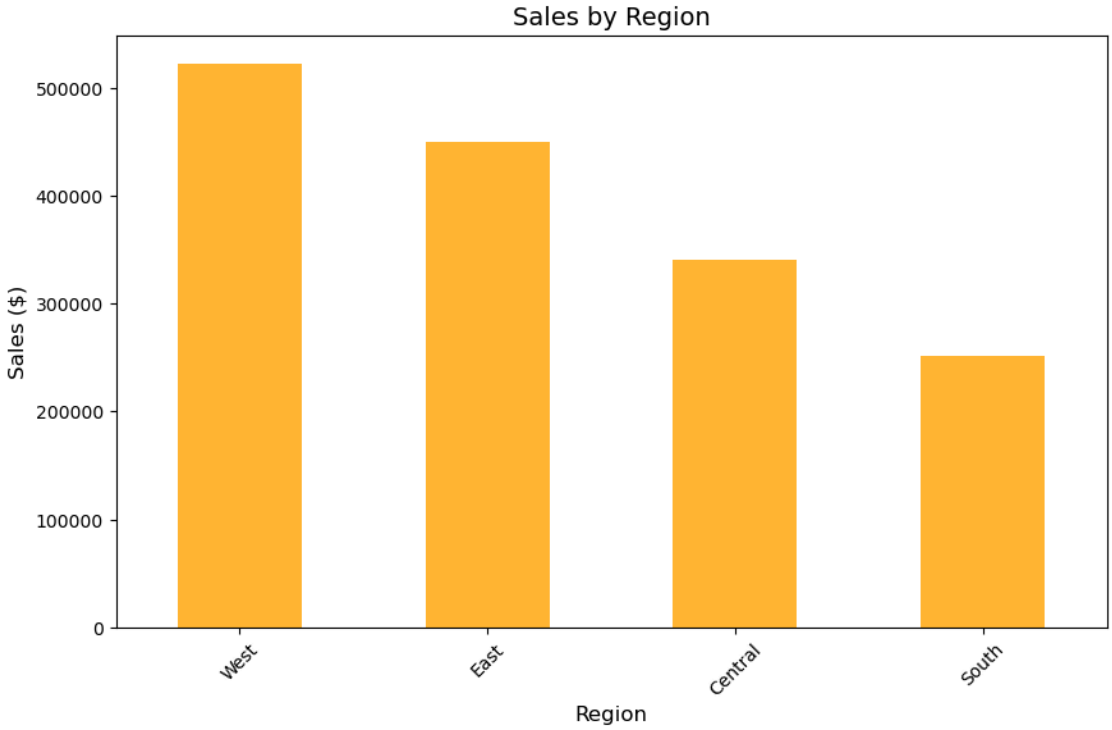


In [16]:
```
plt.figure(figsize=(10, 6))
category_profit.plot(kind='bar', color='green', alpha=0.8)
plt.title("Profit by Category", fontsize=14)
plt.xlabel("Category", fontsize=12)
plt.ylabel("Profit ($)", fontsize=12)
plt.xticks(rotation=45)
plt.show()
```

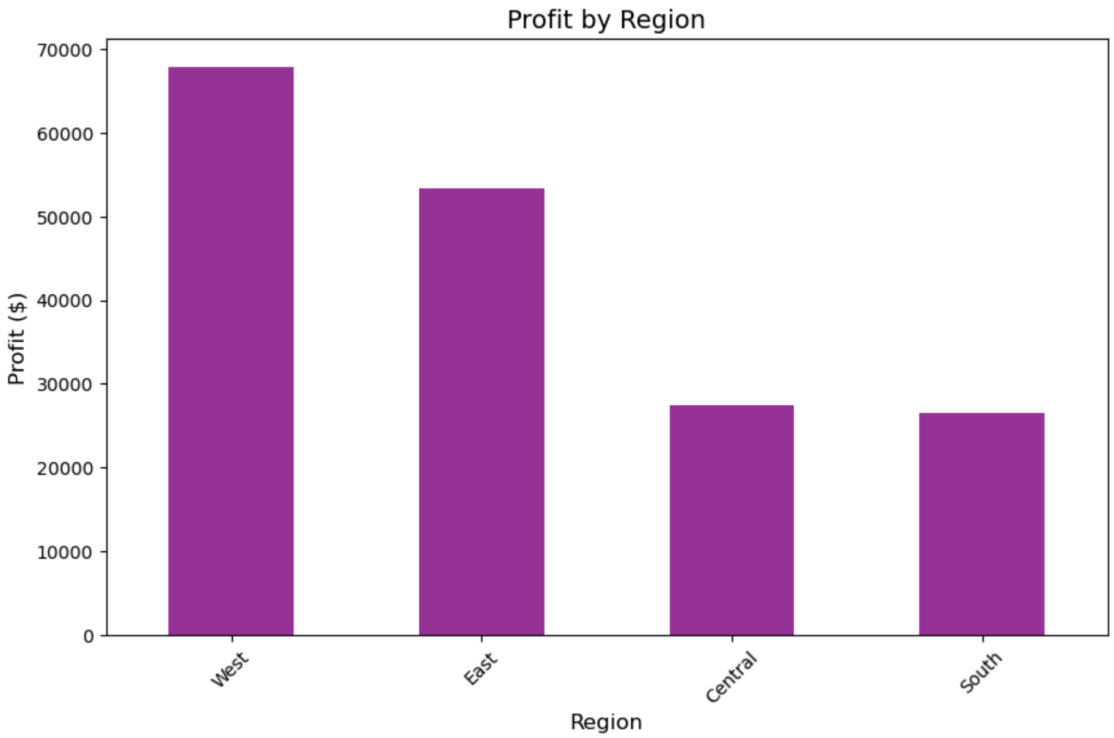


In [17]:
```
# Sales by region
region_sales = data.groupby('Region')['Sales'].sum().sort_values(ascending
region_profit = data.groupby('Region')['Profit'].sum().sort_values(ascendi
```

In [18]:
```
# Visualize region performance
plt.figure(figsize=(10, 6))
region_sales.plot(kind='bar', color='orange', alpha=0.8)
plt.title("Sales by Region", fontsize=14)
plt.xlabel("Region", fontsize=12)
plt.ylabel("Sales ($)", fontsize=12)
plt.xticks(rotation=45)
plt.show()
```



In [19]:
```
plt.figure(figsize=(10, 6))
region_profit.plot(kind='bar', color='purple', alpha=0.8)
plt.title("Profit by Region", fontsize=14)
plt.xlabel("Region", fontsize=12)
plt.ylabel("Profit ($)", fontsize=12)
plt.xticks(rotation=45)
plt.show()
```

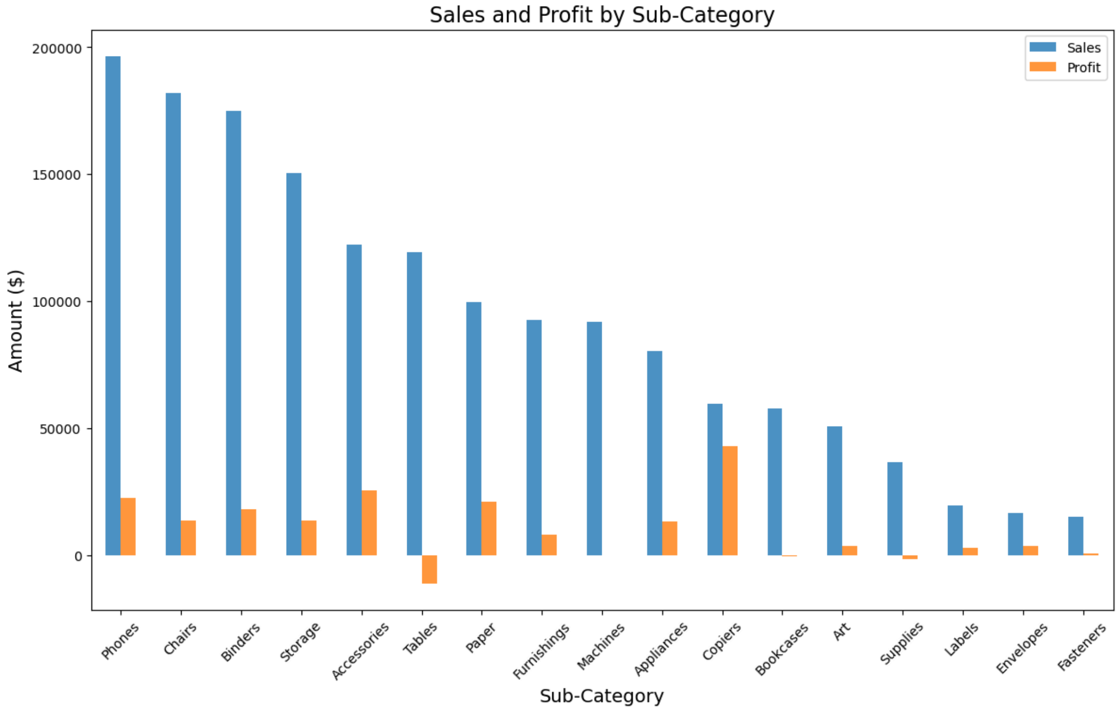


Step 6: Sub-Category and Segment Analysis

Analyze and visualize sales and profit by sub-category and customer segment.

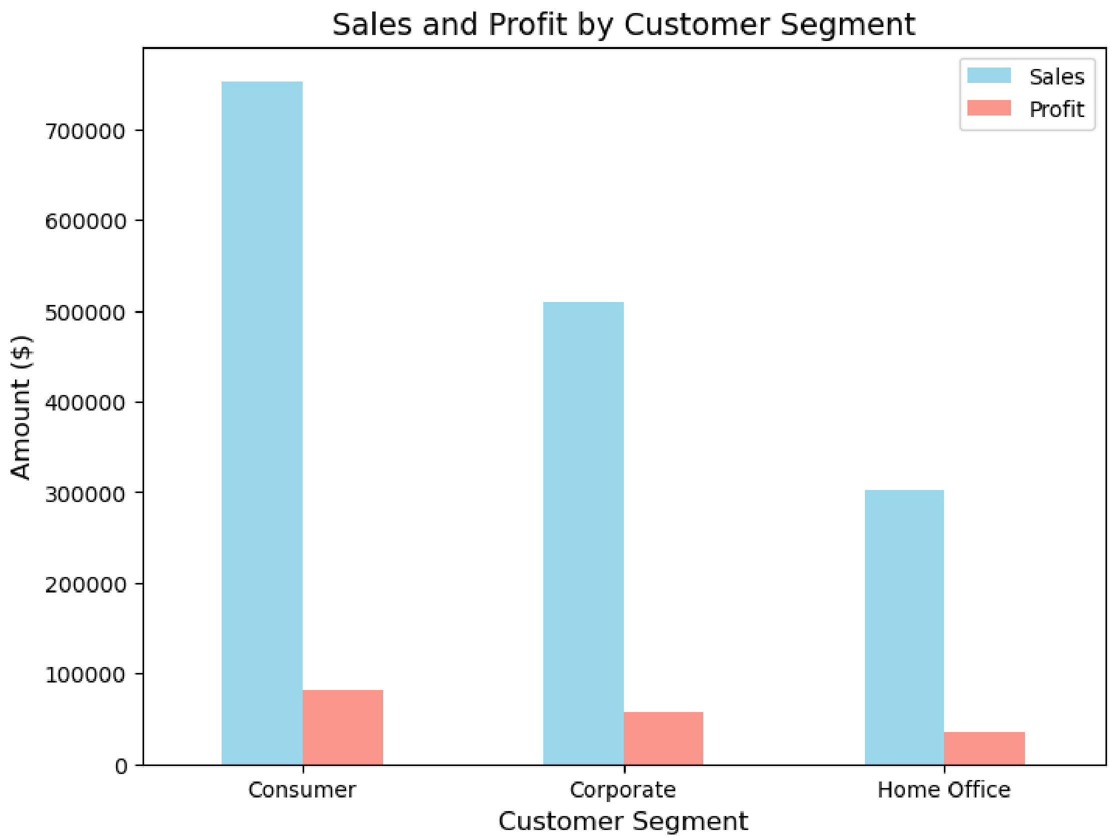
In [20]:
```
# Sales and profit by sub-category
subcategory_performance = data.groupby('Sub-Category').agg({'Sales': 'sum'
```

In [21]:
```
# Visualize sub-category performance
subcategory_performance.plot(kind='bar', figsize=(14, 8), alpha=0.8)
plt.title("Sales and Profit by Sub-Category", fontsize=16)
plt.xlabel("Sub-Category", fontsize=14)
plt.ylabel("Amount ($)", fontsize=14)
plt.xticks(rotation=45)
plt.show()
```



In [22]:
```
# Sales and profit by customer segment
segment_performance = data.groupby('Segment').agg({'Sales': 'sum', 'Profit
```

In [23]:
```
# Visualize customer segment performance
segment_performance.plot(kind='bar', color=['skyblue', 'salmon'], alpha=0.
plt.title("Sales and Profit by Customer Segment", fontsize=14)
plt.xlabel("Customer Segment", fontsize=12)
plt.ylabel("Amount ($)", fontsize=12)
plt.xticks(rotation=0)
plt.show()
```



Step 7: Insights and Recommendations

Generate insights based on your analysis:

1.Identify the most profitable categories and regions.

2.Determine any regions or categories with significant losses.

3.Highlight customer segments contributing the most to sales and profit.

4.Spot high-performing or underperforming sub-categories.

1. Identify the Most Profitable Categories and Regions

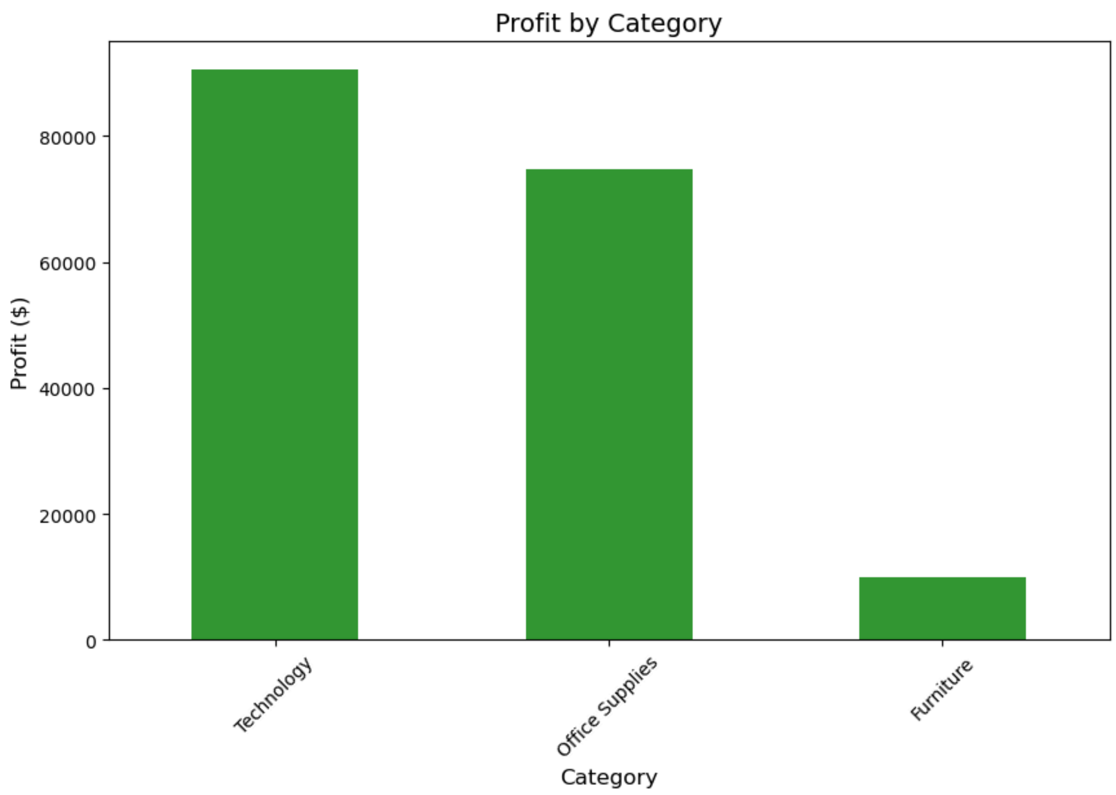
In [24]:
```
# Most profitable categories
most_profitable_categories = data.groupby('Category')['Profit'].sum().sort
```

In [25]:

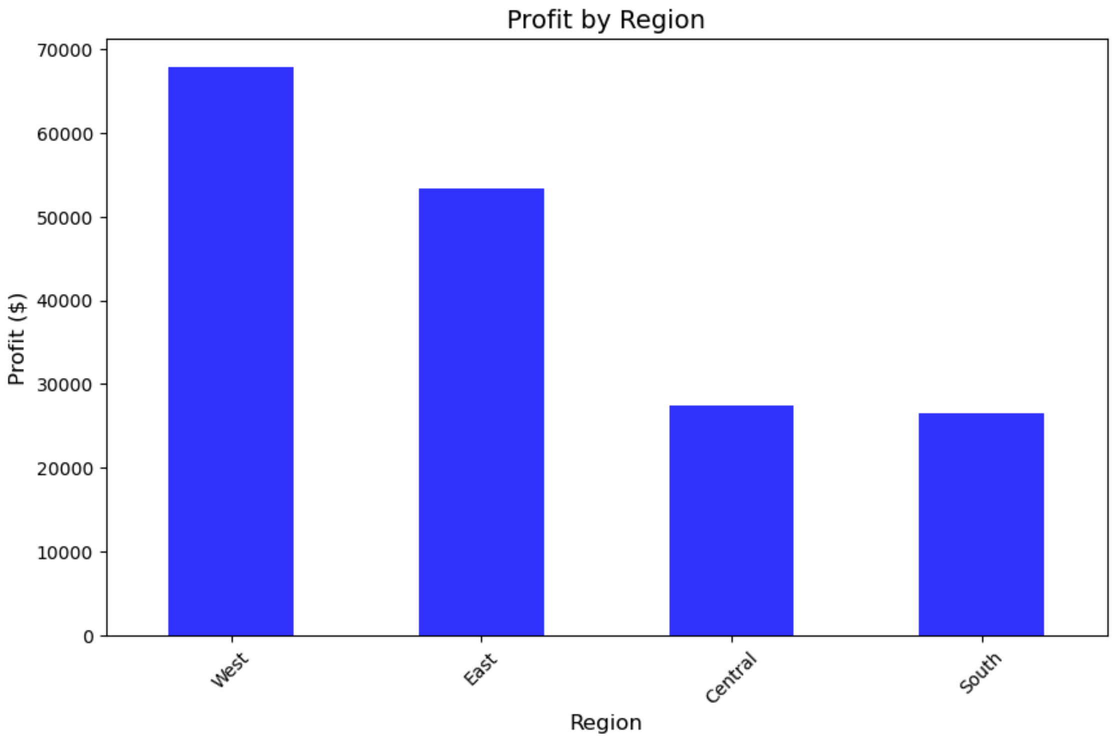
```
# Most profitable regions
most_profitable_regions = data.groupby('Region')['Profit'].sum().sort_valu
```

In [26]:

```
# Visualize profitable categories
plt.figure(figsize=(10, 6))
most_profitable_categories.plot(kind='bar', color='green', alpha=0.8)
plt.title("Profit by Category", fontsize=14)
plt.xlabel("Category", fontsize=12)
plt.ylabel("Profit ($)", fontsize=12)
plt.xticks(rotation=45)
plt.show()
```



In [27]:
```python
# Visualize profitable regions
plt.figure(figsize=(10, 6))
most_profitable_regions.plot(kind='bar', color='blue', alpha=0.8)
plt.title("Profit by Region", fontsize=14)
plt.xlabel("Region", fontsize=12)
plt.ylabel("Profit ($)", fontsize=12)
plt.xticks(rotation=45)
plt.show()
```



2. Determine Regions or Categories with Significant Losses

In [28]:
```python
# Categories with losses (negative profit)
categories_with_loss = data.groupby('Category')['Profit'].sum()
categories_with_loss = categories_with_loss[categories_with_loss < 0]
```

In [29]:
```python
# Regions with losses (negative profit)
regions_with_loss = data.groupby('Region')['Profit'].sum()
regions_with_loss = regions_with_loss[regions_with_loss < 0]
```

In [30]:
```
# Visualize categories with losses
if not categories_with_loss.empty:
    plt.figure(figsize=(10, 6))
    categories_with_loss.plot(kind='bar', color='red', alpha=0.8)
    plt.title("Categories with Losses", fontsize=14)
    plt.xlabel("Category", fontsize=12)
    plt.ylabel("Profit ($)", fontsize=12)
    plt.xticks(rotation=45)
    plt.show()
```

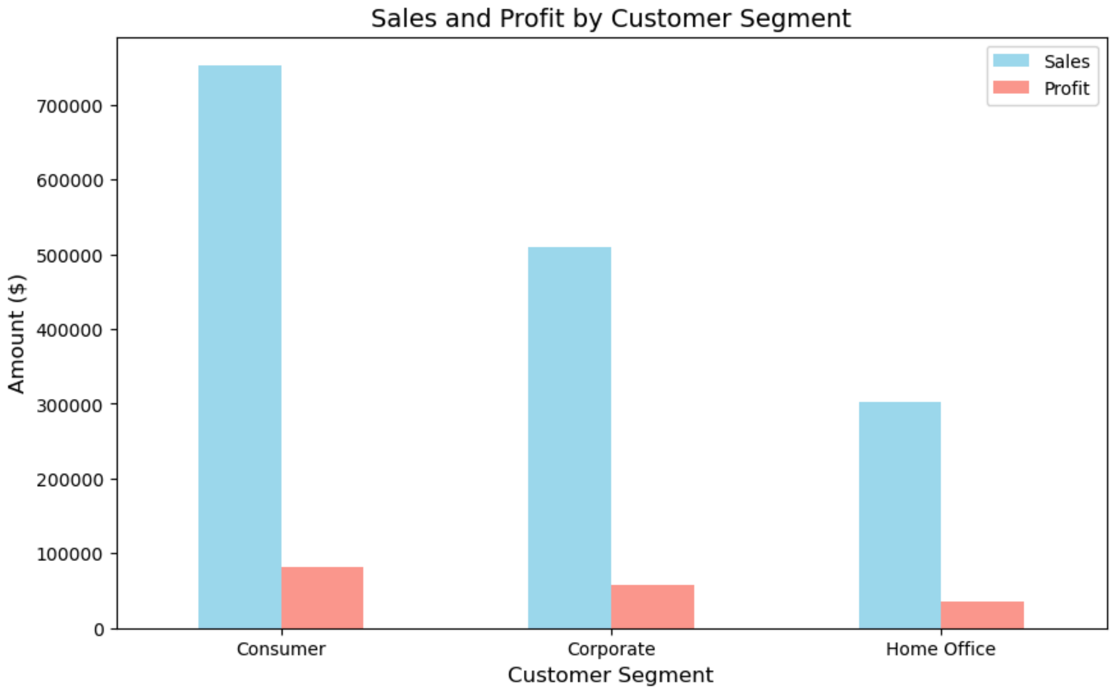
In [31]:
```
# Visualize regions with losses
if not regions_with_loss.empty:
    plt.figure(figsize=(10, 6))
    regions_with_loss.plot(kind='bar', color='orange', alpha=0.8)
    plt.title("Regions with Losses", fontsize=14)
    plt.xlabel("Region", fontsize=12)
    plt.ylabel("Profit ($)", fontsize=12)
    plt.xticks(rotation=45)
    plt.show()
```

3. Highlight Customer Segments Contributing the Most to Sales and Profit

In [32]:
```
# Sales and profit by customer segment
segment_sales_profit = data.groupby('Segment').agg({'Sales': 'sum', 'Profi
```

In [33]:

```
# Visualize sales and profit by segment
segment_sales_profit.plot(kind='bar', figsize=(10, 6), color=['skyblue', '
plt.title("Sales and Profit by Customer Segment", fontsize=14)
plt.xlabel("Customer Segment", fontsize=12)
plt.ylabel("Amount ($)", fontsize=12)
plt.xticks(rotation=0)
plt.legend(["Sales", "Profit"])
plt.show()
```



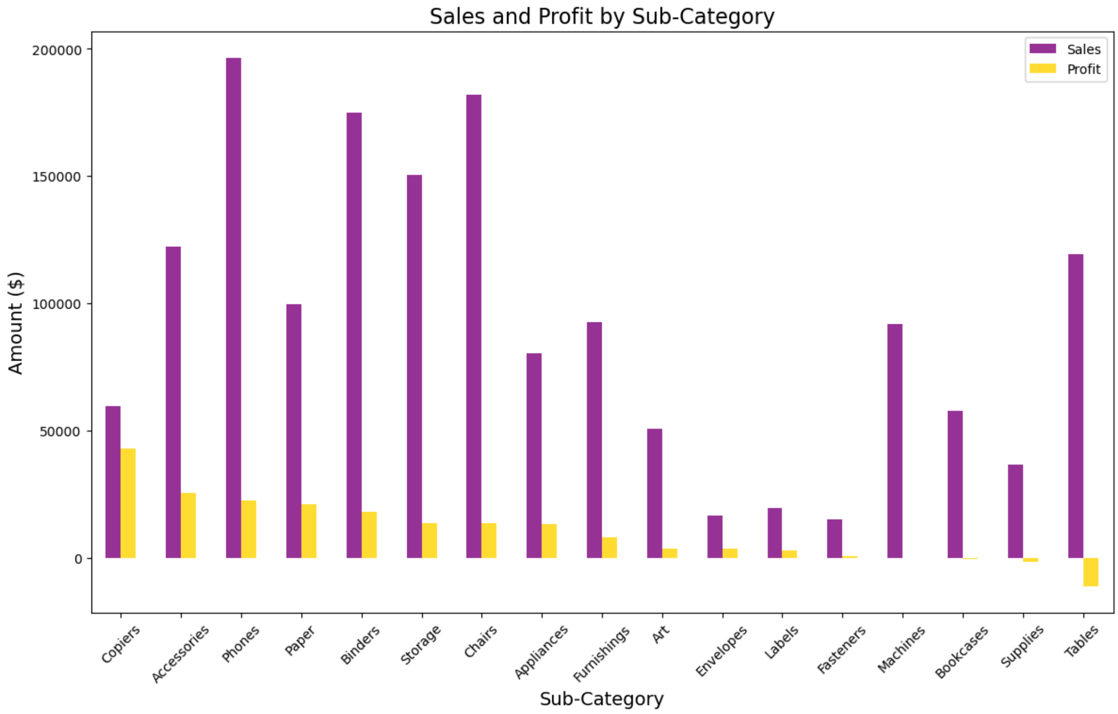
4. Spot High-Performing or Underperforming Sub-Categories

In [34]:

```
# Sales and profit by sub-category
subcategory_performance = data.groupby('Sub-Category').agg({'Sales': 'sum'
```

In [35]:
```
# Visualize high and low performing sub-categories
plt.figure(figsize=(14, 8))
subcategory_performance[['Sales', 'Profit']].plot(kind='bar', alpha=0.8, f
plt.title("Sales and Profit by Sub-Category", fontsize=16)
plt.xlabel("Sub-Category", fontsize=14)
plt.ylabel("Amount ($)", fontsize=14)
plt.xticks(rotation=45)
plt.legend(["Sales", "Profit"])
plt.show()
```

```
<Figure size 1400x800 with 0 Axes>
```



In [36]:
```
# Identify underperforming sub-categories (negative profit)
underperforming_subcategories = subcategory_performance[subcategory_perfor
print("Underperforming Sub-Categories (Negative Profit):")
print(underperforming_subcategories)
```

```
Underperforming Sub-Categories (Negative Profit):
                    Sales       Profit
Sub-Category
Bookcases       57577.6862    -342.8883
Supplies        36720.9860   -1654.2767
Tables         119293.7430  -11091.6365
```

```
You can use the above analysis to:

--Identify high-performing categories and regions to focus marketing
efforts.
---Address underperforming regions and categories to reduce losses.
---Prioritize customer segments driving the most sales and profit.
---Improve performance in underperforming sub-categories.
```