🛍️ Retail Sales EDA & Customer Segmentation Report

Client (Project Type): EDA Case Study

Author: Shubh Pal

Date: July 2025

Tools Used: Python, Pandas, Matplotlib, Seaborn, Jupyter Notebook

Dataset: Superstore Sales Data (2011-2014)

📌 Objective:

Perform exploratory data analysis to uncover sales trends, customer behavior, and product performance. Deliver actionable insights to support data-driven decisions for marketing, inventory, and customer retention strategies.

📊 Executive Summary

This analysis explores retail sales data from 2011 to 2014 to uncover customer behavior, sales patterns, and product performance. The goal is to provide business-ready insights that inform marketing, inventory, and customer segmentation strategies.

🔍 Key Insights:

- 📈 **Seasonal Trends**: Sales peak consistently in **Q4**, suggesting a need to **intensify promotions in Q3**.
- 👥 **Customer Distribution**: The **top 15% of customers** contribute to **over 65% of revenue**, indicating a strong opportunity for **VIP loyalty programs**.
- 📦 **Product Performance**: Sub-categories like **"Phones" and "Chairs"** are top performers in both sales and profit.
- ⚠️ **Inventory Risk**: Some products show **high quantity but low sales**, signaling potential overstock or low demand.
- 🗂️ **Category Performance**: **Office Supplies** has high sales volume but low profit margins — consider reviewing pricing or discounting strategy.

📌 Recommendations Preview:

- **Increase marketing spend in Q3** to maximize Q4 peak season.
- Reassess **inventory strategy** for slow-moving products.
- Consider **re-pricing or bundling** low-margin sub-categories.

- Implementing a tiered **VIP loyalty program** is a strong recommendation with the schemes like **Exclusive Access**, **Personalized Offers**, **Personalized Communication**, **Track of Key Metrics**.

📂 Dataset Overview

This dataset contains order-level retail sales data from 2011 to 2015. It includes customer information, product categories, transaction details, and shipping information.

🔑 Key Information:

- **Time Range**: January 2011 – December 2014
- **Total Records**: 9,994 rows
- **Columns**: 21
- **Key Features**:
  - `Order ID`, `Order Date`, `Ship Date`, `Ship Mode`
  - `Customer ID`, `Customer Name`, `Segment`, `Region`
  - `Product ID`, `Category`, `Sub-Category`, `Product Name`
  - `Sales`, `Quantity`, `Discount`, `Profit`, `Margin`

In [27]:

```
import pandas as pd
import warnings
warnings.filterwarnings('ignore')

df = pd.read_csv(r"D:\Projects\first_sql.sql\Superstore.csv", encoding="windows-
df.head()
```

Out[27]:

	Row ID	Order ID	Order Date	Ship Date	Ship Mode	Customer ID	Customer Name	Segment	Country	
0	1	CA-2013-152156	09-11-2013	12-11-2013	Second Class	CG-12520	Claire Gute	Consumer	United States	Hen
1	2	CA-2013-152156	09-11-2013	12-11-2013	Second Class	CG-12520	Claire Gute	Consumer	United States	Hen
2	3	CA-2013-138688	13-06-2013	17-06-2013	Second Class	DV-13045	Darrin Van Huff	Corporate	United States	A
3	4	US-2012-108966	11-10-2012	18-10-2012	Standard Class	SO-20335	Sean O'Donnell	Consumer	United States	Lau
4	5	US-2012-108966	11-10-2012	18-10-2012	Standard Class	SO-20335	Sean O'Donnell	Consumer	United States	Lau

5 rows × 21 columns

In [28]:

```
df.shape
```

Out[28]:

```
(9994, 21)
```

In [29]:

```
# Check for nulls
df.isnull().sum()
```

Out[29]:
```
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 [30]:
```
#Removing any Duplicate (if Present)
df.drop_duplicates(inplace = True)
```

In [31]:
```
# Converting Date Columns from object to Datetime Format

df['Order Date'] = pd.to_datetime(df['Order Date'], format = "%d-%m-%Y", errors=
df['Ship Date'] = pd.to_datetime(df['Ship Date'], format = "%d-%m-%Y", errors="c
```

In [32]:
```
# Converting Numerical Columns From Float to Integer Data Type

df['Sales'] = df['Sales'].round().astype('Int64')
df['Discount'] = df['Discount'].round().astype('Int64')
df['Profit'] = df['Profit'].round().astype('Int64')
```

In [33]:
```
# Final Check for Data Types of all
df.dtypes
```

Out[33]:
```
Row ID                    int64
Order ID                 object
Order Date       datetime64[ns]
Ship Date        datetime64[ns]
Ship Mode                object
Customer ID              object
Customer Name            object
Segment                  object
Country                  object
City                     object
State                    object
Postal Code               int64
Region                   object
Product ID               object
Category                 object
Sub-Category             object
Product Name             object
Sales                     Int64
Quantity                  int64
Discount                  Int64
Profit                    Int64
dtype: object
```

In [34]:
```
# Checking Summary for Profit Column
df['Profit'].describe()
```

Out[34]:
```
count        9994.0
mean      28.651191
std      234.255752
min         -6600.0
25%             2.0
50%             9.0
75%            29.0
max          8400.0
Name: Profit, dtype: Float64
```

In [35]:
```
# Entries with negative profits maybe due to typos or heavy discounts (Cannot be
df[df['Profit']<0]
```

Out[35]:

	Row ID	Order ID	Order Date	Ship Date	Ship Mode	Customer ID	Customer Name	Segment	Country
3	4	US-2012-108966	2012-10-11	2012-10-18	Standard Class	SO-20335	Sean O'Donnell	Consumer	United States
14	15	US-2012-118983	2012-11-22	2012-11-26	Standard Class	HP-14815	Harold Pawlan	Home Office	United States
15	16	US-2012-118983	2012-11-22	2012-11-26	Standard Class	HP-14815	Harold Pawlan	Home Office	United States
23	24	US-2014-156909	2014-07-17	2014-07-19	Second Class	SF-20065	Sandra Flanagan	Consumer	United States
27	28	US-2012-150630	2012-09-17	2012-09-21	Standard Class	TB-21520	Tracy Blumstein	Consumer	United States
...	...	...	...	...	...	...	...	...	...
9920	9921	CA-2013-149272	2013-03-16	2013-03-20	Standard Class	MY-18295	Muhammed Yedwab	Corporate	United States
9921	9922	CA-2011-111360	2011-11-24	2011-11-30	Standard Class	AT-10435	Alyssa Tate	Home Office	United States
9931	9932	CA-2012-104948	2012-11-13	2012-11-17	Standard Class	KH-16510	Keith Herrera	Consumer	United States
9937	9938	CA-2013-164889	2013-06-04	2013-06-07	Second Class	CP-12340	Christine Phan	Corporate	United States
9962	9963	CA-2012-168088	2012-03-19	2012-03-22	First Class	CM-12655	Corinna Mitchell	Home Office	United States

	Row ID	Order ID	Order Date	Ship Date	Ship Mode	Customer ID	Customer Name	Segment	Countr

1865 rows × 21 columns

In [36]:
```
# Filtering out real Profits
df = df[df['Profit']>0]
df.shape
```

Out[36]: (7964, 21)

In [37]:
```
df.drop(columns = ['Row ID'], inplace=True)
```

In [38]:
```
df['Profit'].describe()
```

Out[38]:
```
count       7964.0
mean      55.559769
std      214.882169
min            1.0
25%            5.0
50%           14.0
75%           41.0
max         8400.0
Name: Profit, dtype: Float64
```

In [39]:
```
bins = [0, 1000, 5000,7000,float('inf')]
labels = ['low', 'medium', 'high', 'highest']

# Create Profit band column
df['Margin'] = pd.cut(df['Profit'], bins=bins, labels=labels)

df['Margin'].value_counts()
```

Out[39]:
```
Margin
low        7922
medium       39
high          2
highest       1
Name: count, dtype: int64
```

📊 Descriptive Statistics

🔍 Overview

To build an informed foundation for analysis, we examine both **numerical** and **categorical** variables.

📐 Numerical Features Summary

We focus on key continuous variables:

- **Sales**: Total value of each transaction.
- **Quantity**: Number of items sold.
- **Discount**: Discount applied.
- **Profit**: Net profit per order.

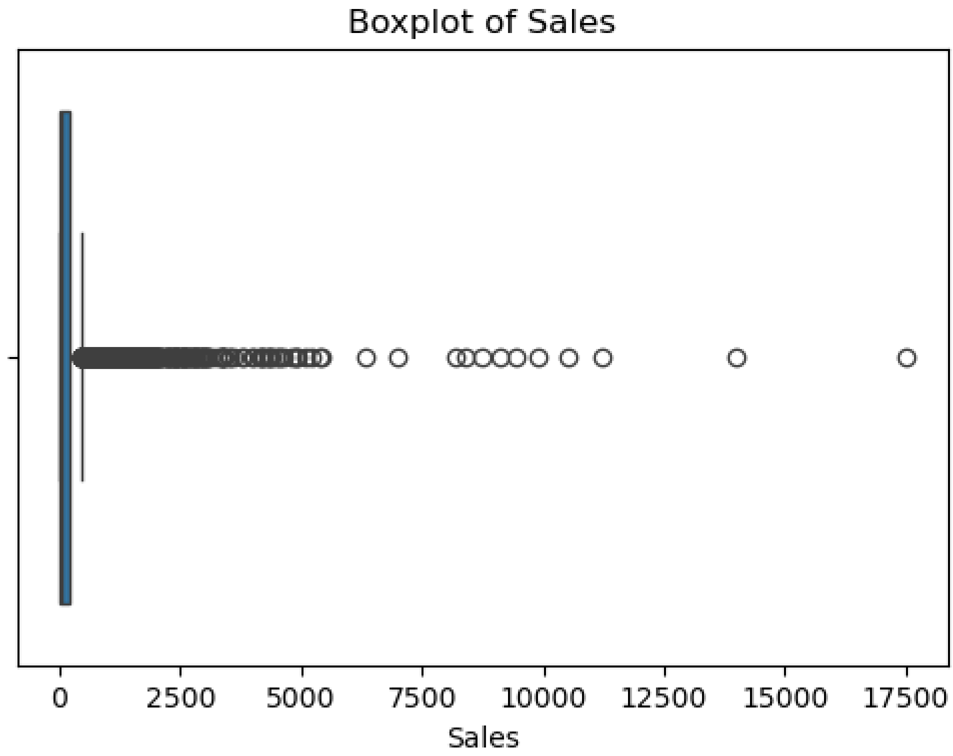
Key metrics examined: mean, median, standard deviation, min, max.

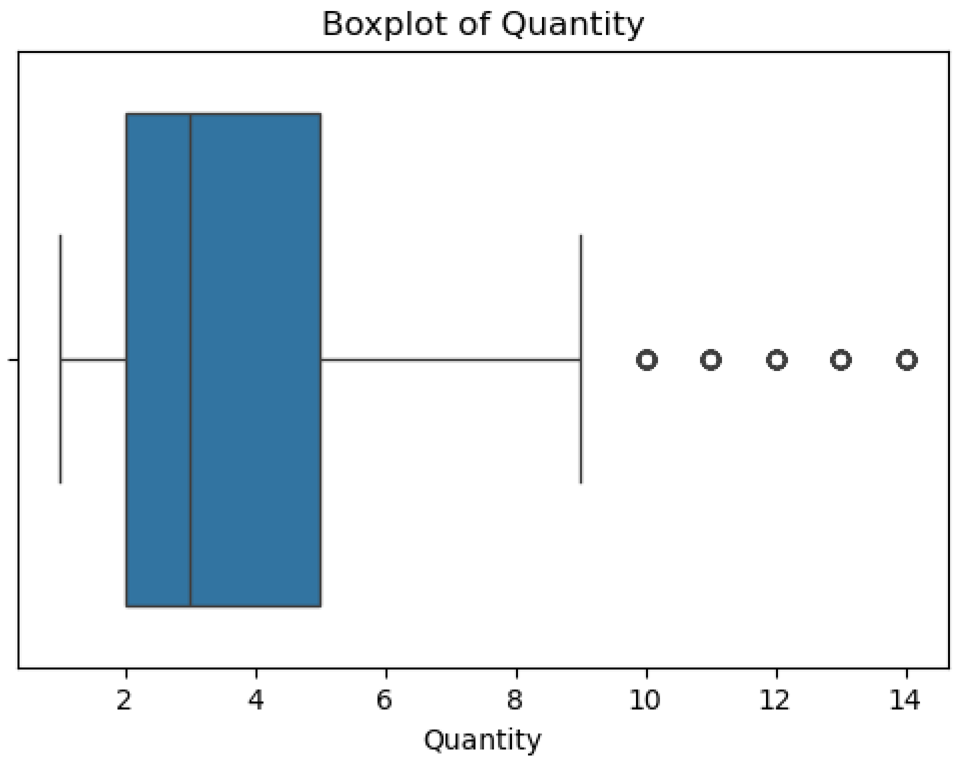
In [40]:
```
# Summary for Numerical Variables
df[['Sales', 'Quantity', 'Discount', 'Profit']].describe()
```

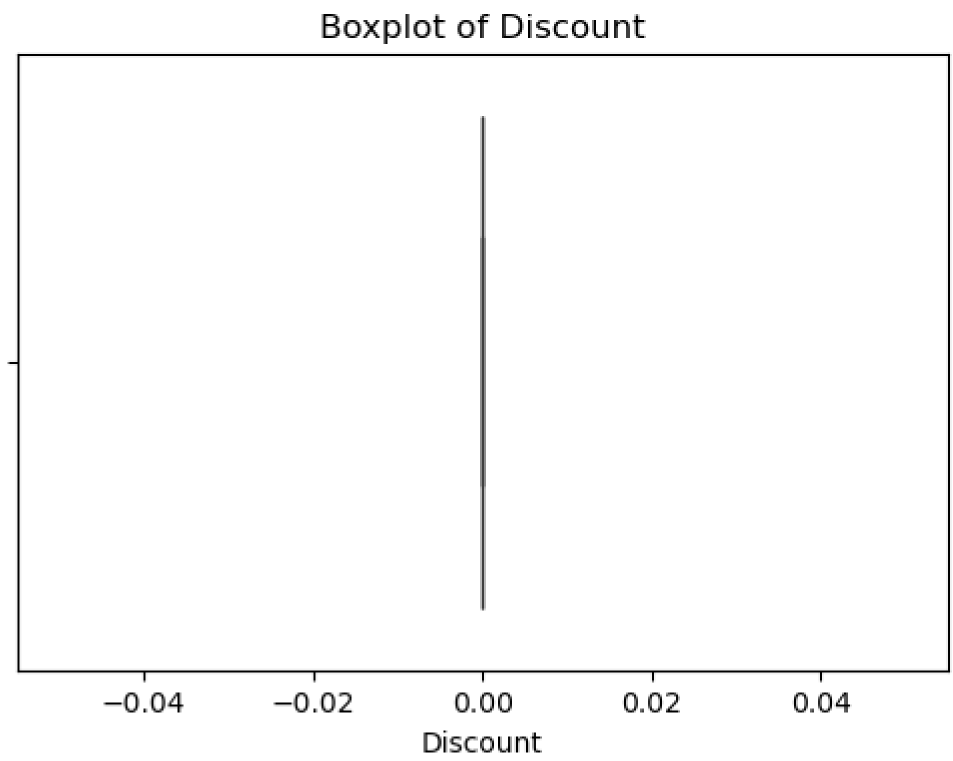
Out[40]:

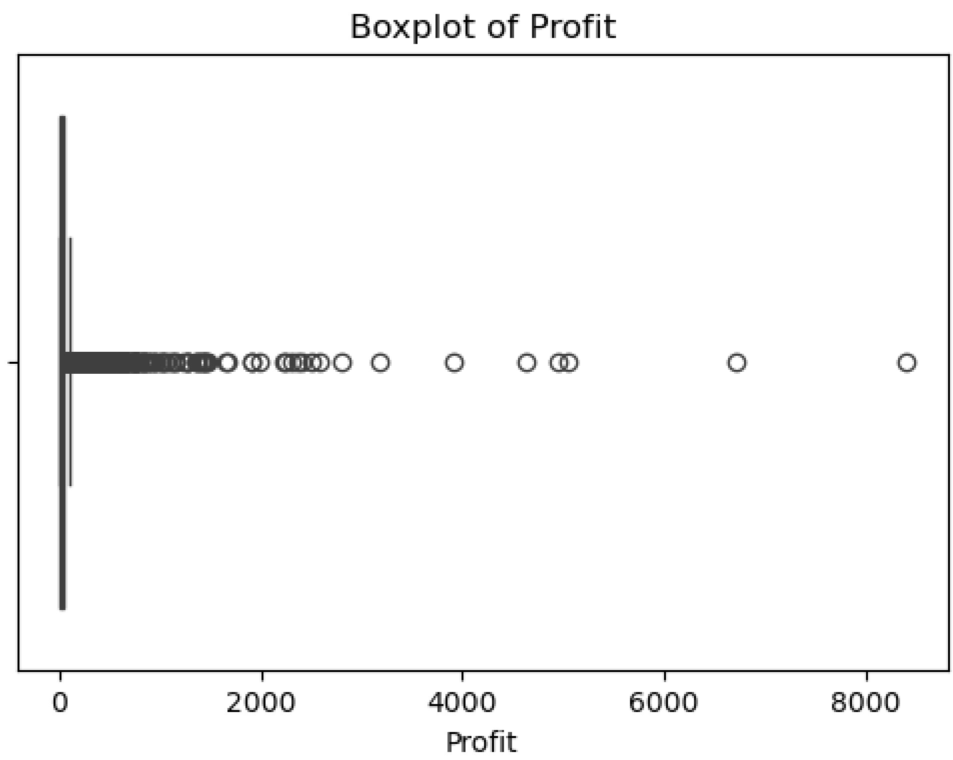
	Sales	Quantity	Discount	Profit
count	7964.0	7964.000000	7964.0	7964.0
mean	226.023857	3.813285	0.0	55.559769
std	603.426513	2.247617	0.0	214.882169
min	1.0	1.000000	0.0	1.0
25%	18.0	2.000000	0.0	5.0
50%	52.0	3.000000	0.0	14.0
75%	195.0	5.000000	0.0	41.0
max	17500.0	14.000000	0.0	8400.0

In [41]:
```
import matplotlib.pyplot as plt
import seaborn as sns

# Plotting Numerical Summaries
num_cols = ['Sales', 'Quantity', 'Discount', 'Profit']

for col in num_cols:
    plt.figure(figsize=(6, 4))
    sns.boxplot(x=df[col])
    plt.title(f'Boxplot of {col}')
    plt.show()
```


Boxplot of Sales
0
2500
5000
7500
10000
12500
15000
17500
Sales


Boxplot of Quantity
2
4
6
8
10
12
14
Quantity





🗂️ Categorical Features Overview

Categorical variables provide segmentation views of the data, such as:

- **Customer Segment** (Consumer, Corporate, Home Office)
- **Shipping Mode**
- **Product Category**
- **Geographic Region**

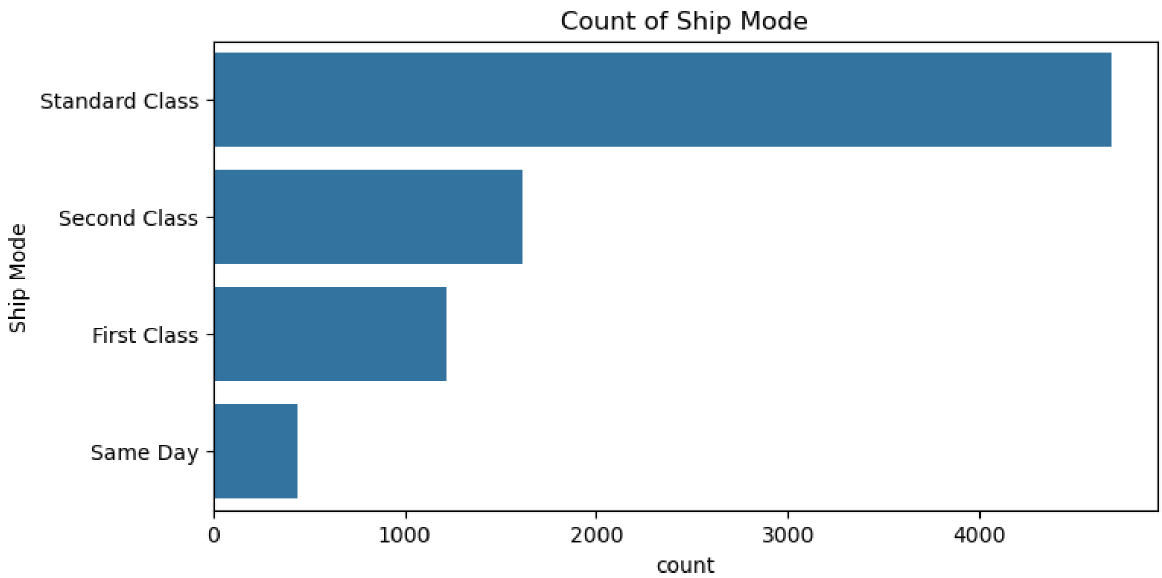
Understanding category distribution reveals where most transactions originate and which segments dominate.

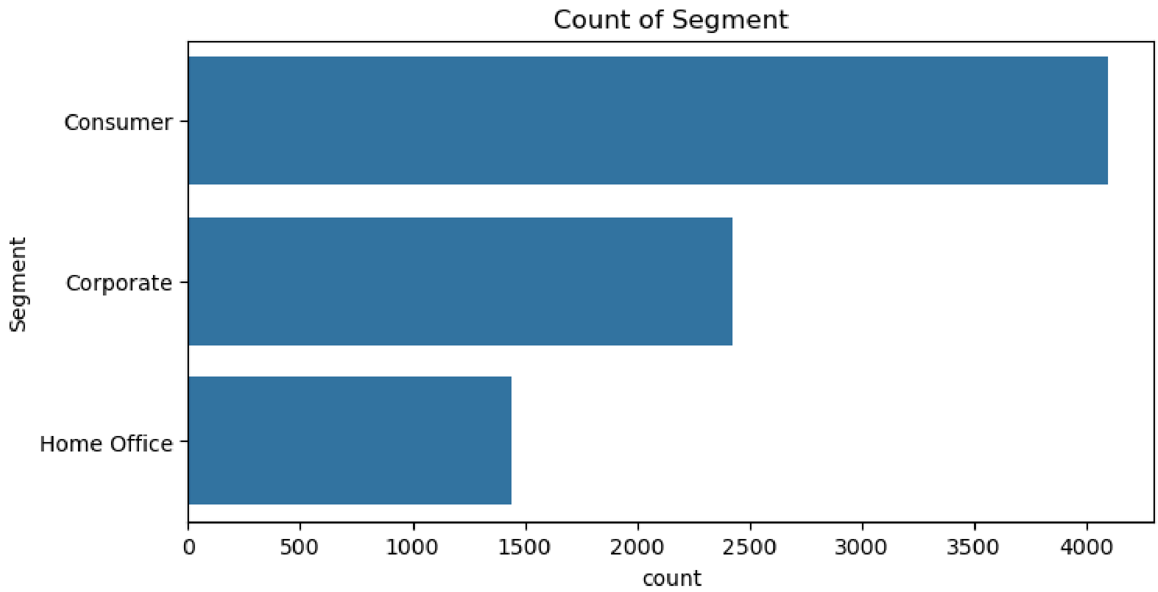
In [42]:
```
# Value counts for top categorical features
df[['Ship Mode', 'Segment', 'Country', 'City', 'State', 'Region', 'Category', 'S
```

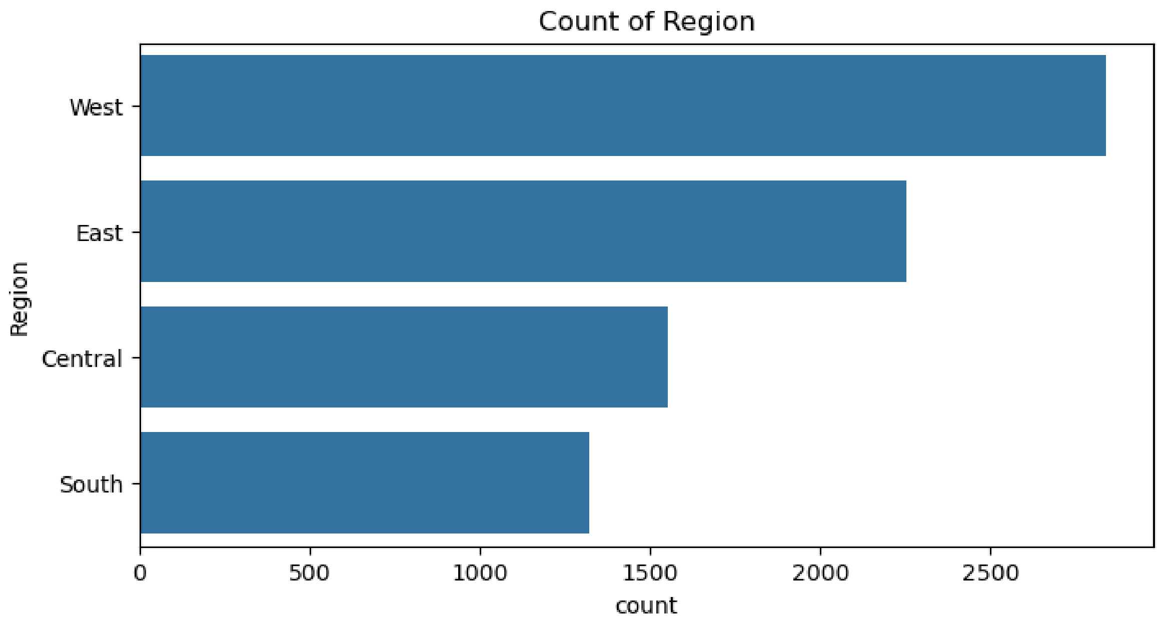
Out[42]:

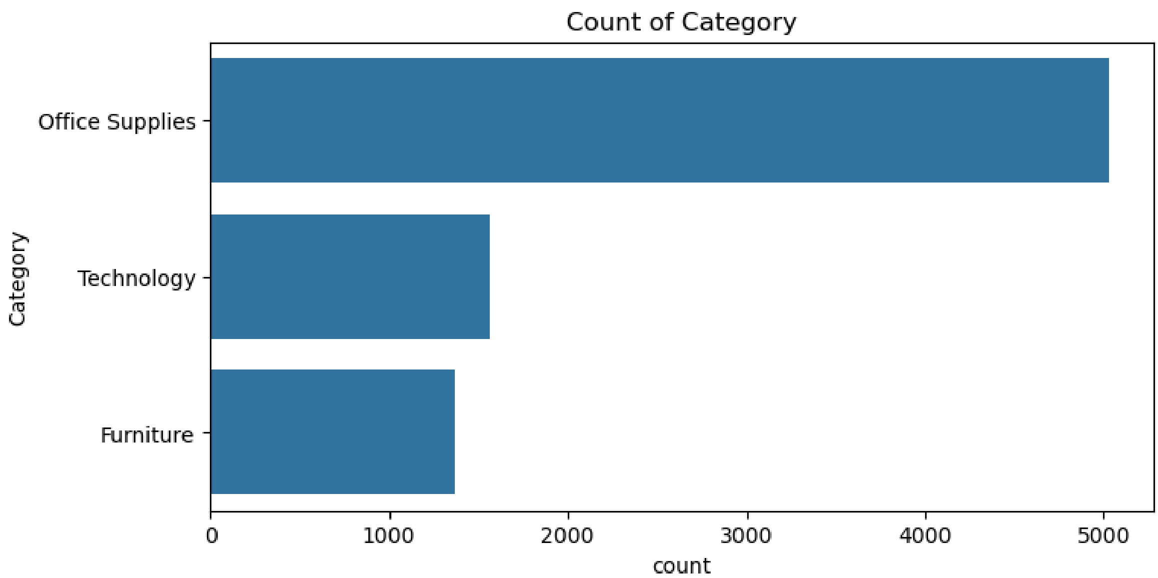
	Ship Mode	Segment	Country	City	State	Region	Category	Sub-Category
count	7964	7964	7964	7964	7964	7964	7964	7964
unique	4	3	1	512	49	4	3	17
top	Standard Class	Consumer	United States	New York City	California	West	Office Supplies	Paper
freq	4692	4097	7964	864	1872	2837	5027	1370

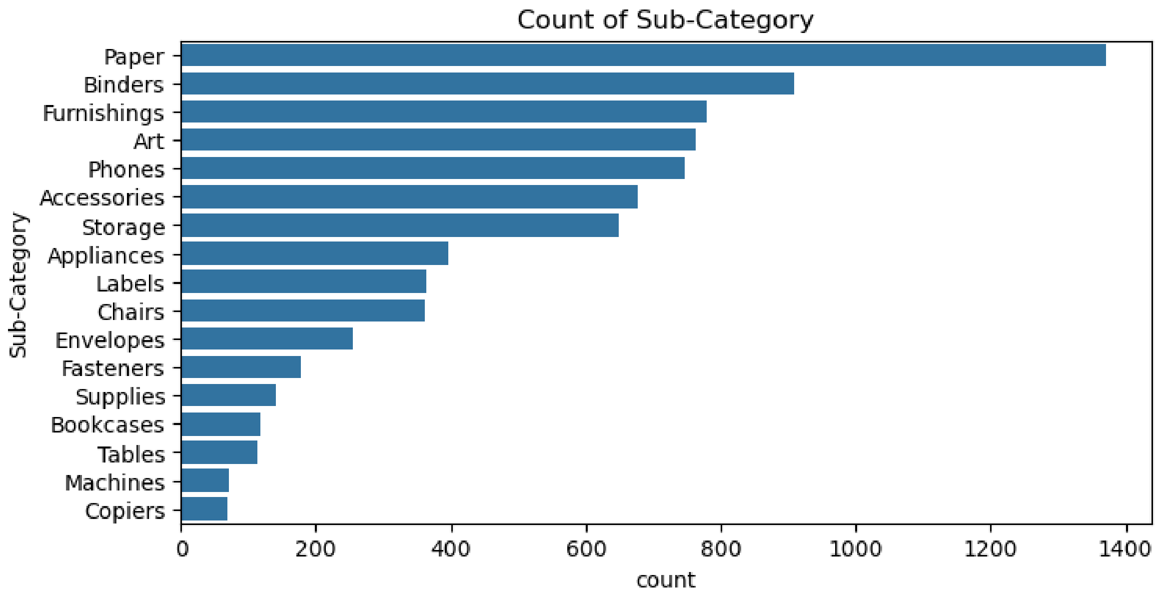
In [43]:
```
cat_cols = ['Ship Mode', 'Segment', 'Region', 'Category', 'Sub-Category']

# Plotting Categorical Distributions
for col in cat_cols:
    plt.figure(figsize=(8, 4))
    sns.countplot(data=df, y=col, order=df[col].value_counts().index)
    plt.title(f'Count of {col}')
    plt.show()
```




Count of Segment
Consumer
Corporate
Home Office
Segment
0
500
1000
1500
2000
2500
3000
3500
4000
count


Count of Region
West
East
Central
South
Region
0
500
1000
1500
2000
2500
count


Count of Category
Office Supplies
Technology
Furniture
Category
0
1000
2000
3000
4000
5000
count



🔎 Observations

- **Consumer** segment dominates sales volume.
- Most orders are shipped via **Standard Class**.
- **Office Supplies** is the most frequent product category.
- **West** region has the highest transaction volume.

📌 Recommendations

- While **Consumers dominate**, explore Corporate/Business segment opportunities through targeted offers.
- Audit **Standard Class** performance and costs. Explore ways to shift customers to more cost-efficient modes.
- Promote high-margin products in frequently ordered categories like **Office Supplies**.
- Use regional data to expand into less saturated markets like **Central** and **South** with focused marketing or regional pricing strategies.

📌 These observations will guide our deeper analysis into time trends, customer segmentation and product performance.

📅 Time-Series Analysis

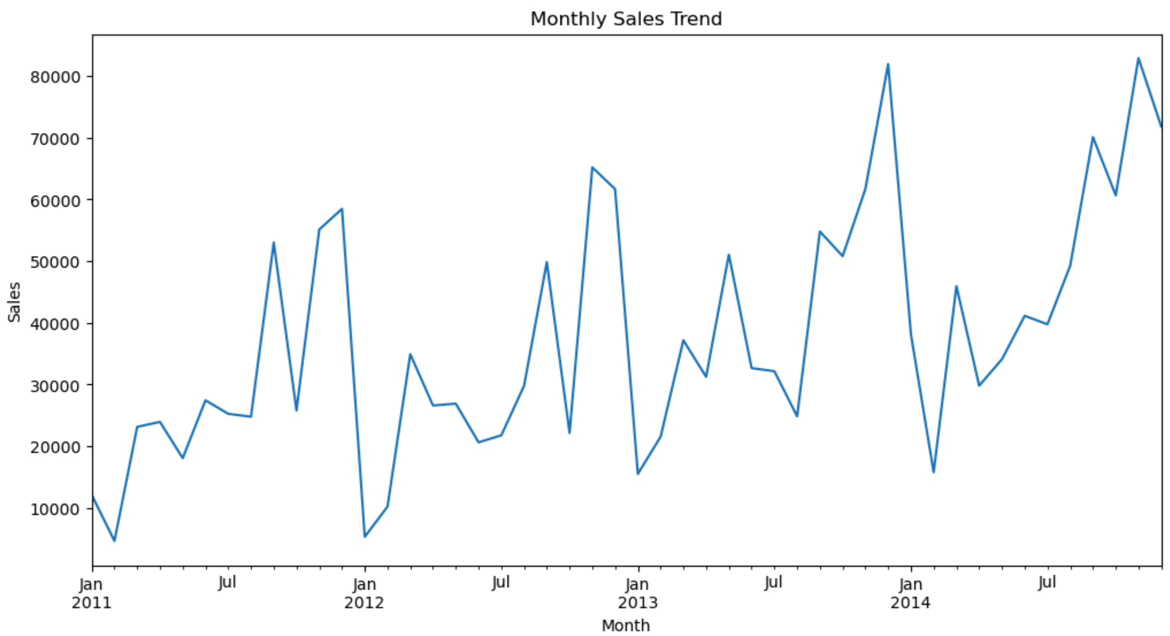
Understanding how sales evolve over time helps uncover seasonal patterns, peak months, and long-term trends. This can support inventory planning, marketing campaigns, and forecasting.

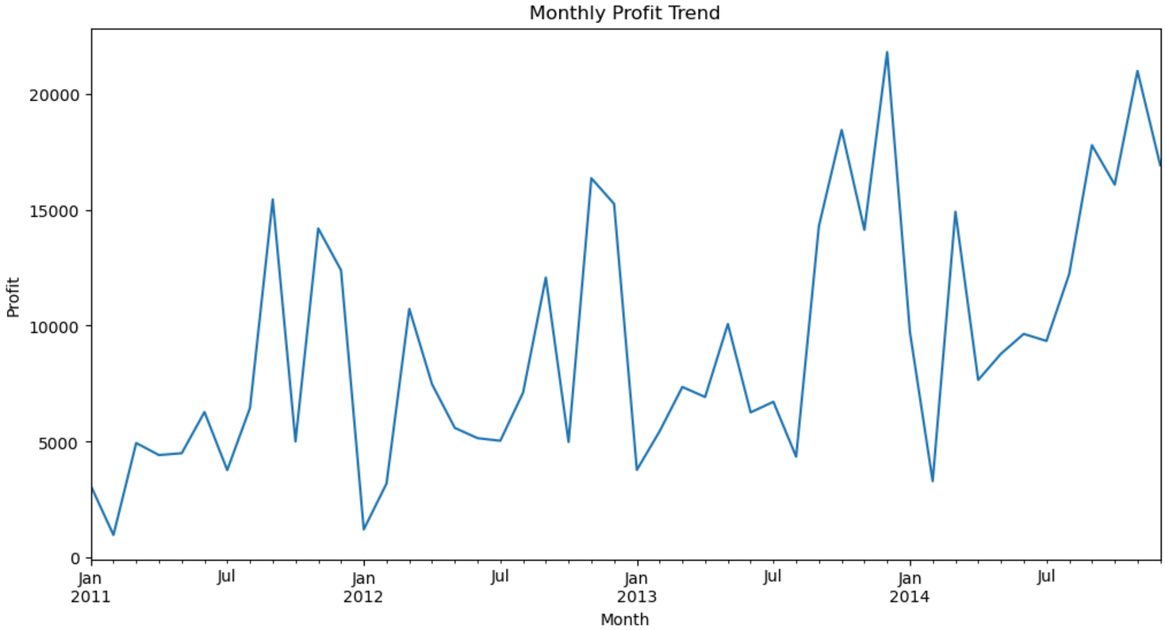
📌 Key Questions:

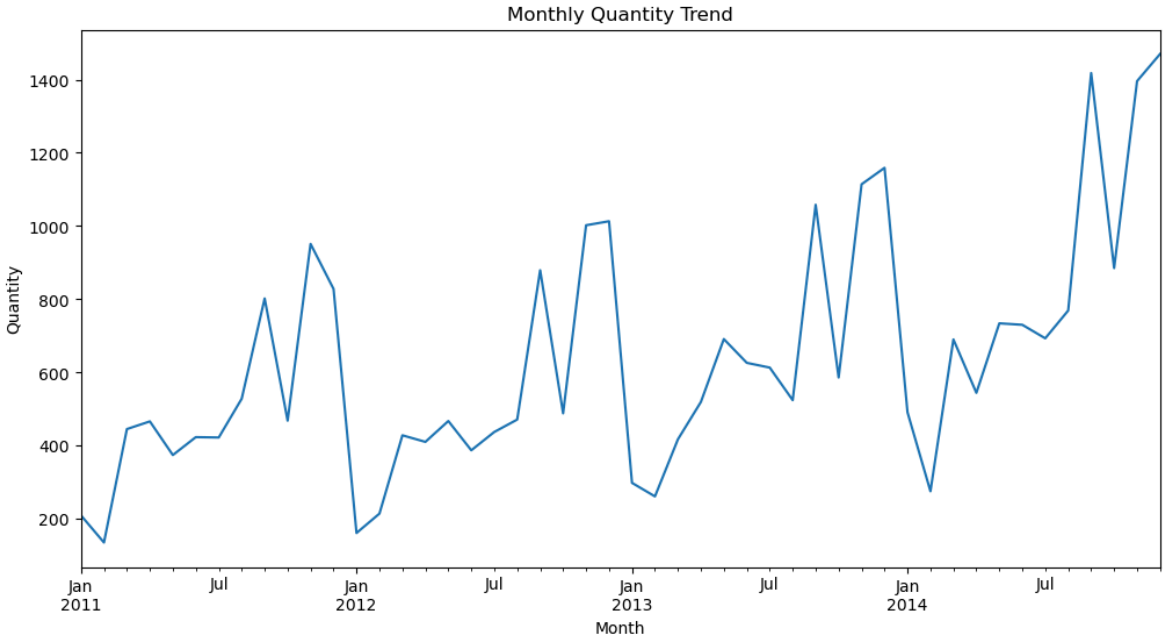
- Are there seasonal sales spikes?
- Which years or months show the highest growth?
- Is the sales trend improving over time?
- What are the Monthly Average Sales?
- What are the Sales Trend for different Categories?

In [44]:
```
df.set_index('Order Date', inplace=True)


monthly_sales = df['Sales'].resample('ME').sum()
monthly_discount = df['Discount'].resample('ME').sum()
monthly_profit = df['Profit'].resample('ME').sum()
monthly_quantity = df['Quantity'].resample('ME').sum()
```

In [45]:
```
plt.figure(figsize=(12, 6))
monthly_sales.plot()
plt.title('Monthly Sales Trend')
plt.xlabel('Month')
plt.ylabel('Sales')
plt.show()

plt.figure(figsize=(12, 6))
monthly_profit.plot()
plt.title('Monthly Profit Trend')
plt.xlabel('Month')
plt.ylabel('Profit')
plt.show()

plt.figure(figsize=(12, 6))
monthly_quantity.plot()
plt.title('Monthly Quantity Trend')
plt.xlabel('Month')
plt.ylabel('Quantity')
plt.show()
```

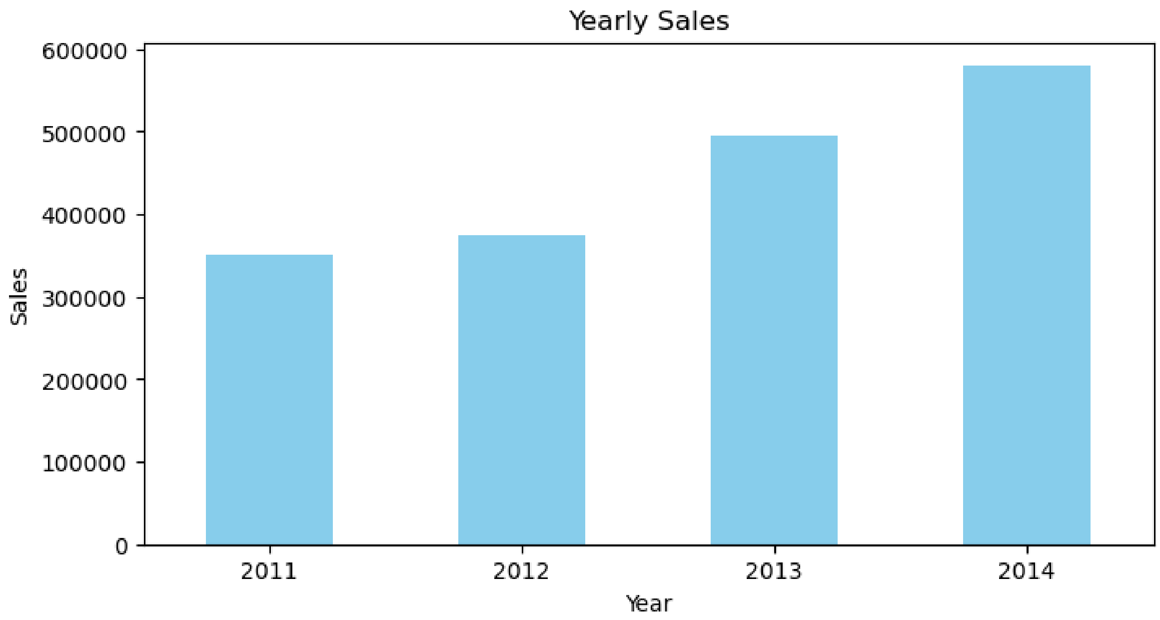


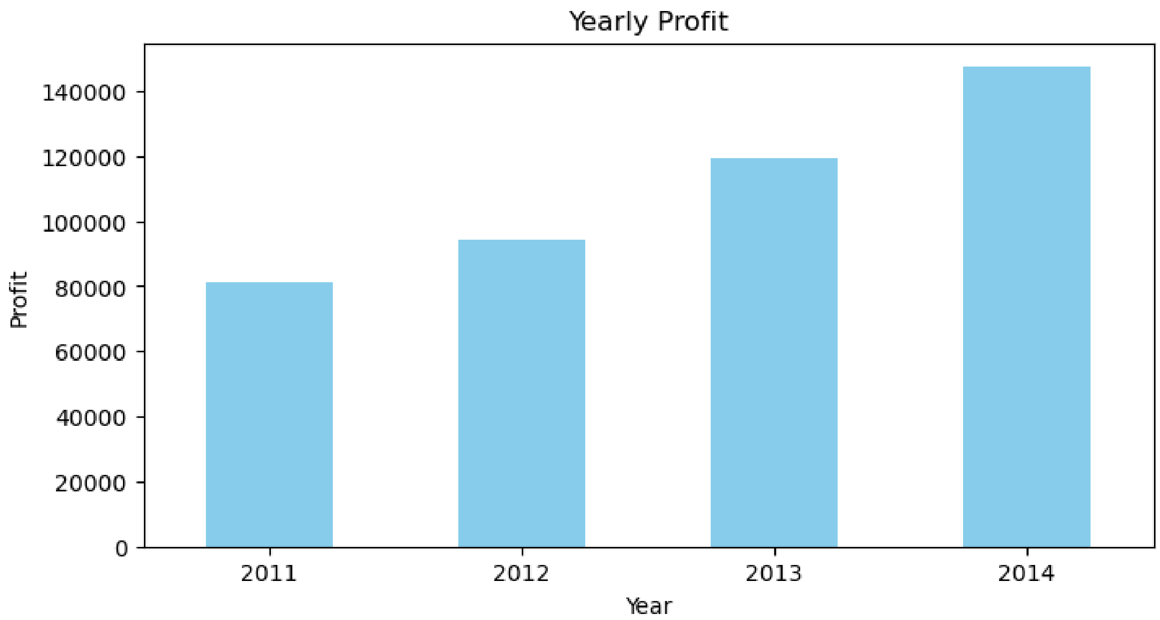




In [46]:

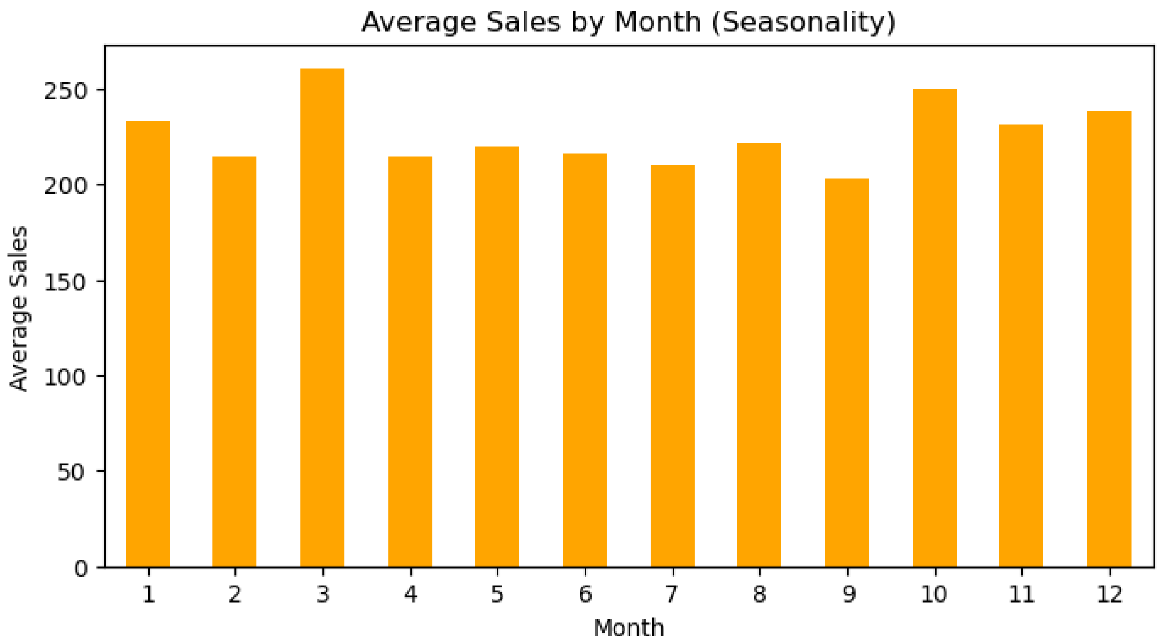
```
df['Year'] = df.index.year
yearly_sales = df.groupby('Year')['Sales'].sum()

yearly_sales.plot(kind='bar', figsize=(8, 4), color='skyblue')
plt.title('Yearly Sales')
plt.ylabel('Sales')
plt.xticks(rotation=0)
plt.show()

df['Year'] = df.index.year
yearly_profit = df.groupby('Year')['Profit'].sum()

yearly_profit.plot(kind='bar', figsize=(8, 4), color='skyblue')
plt.title('Yearly Profit')
plt.ylabel('Profit')
plt.xticks(rotation=0)
plt.show()
```



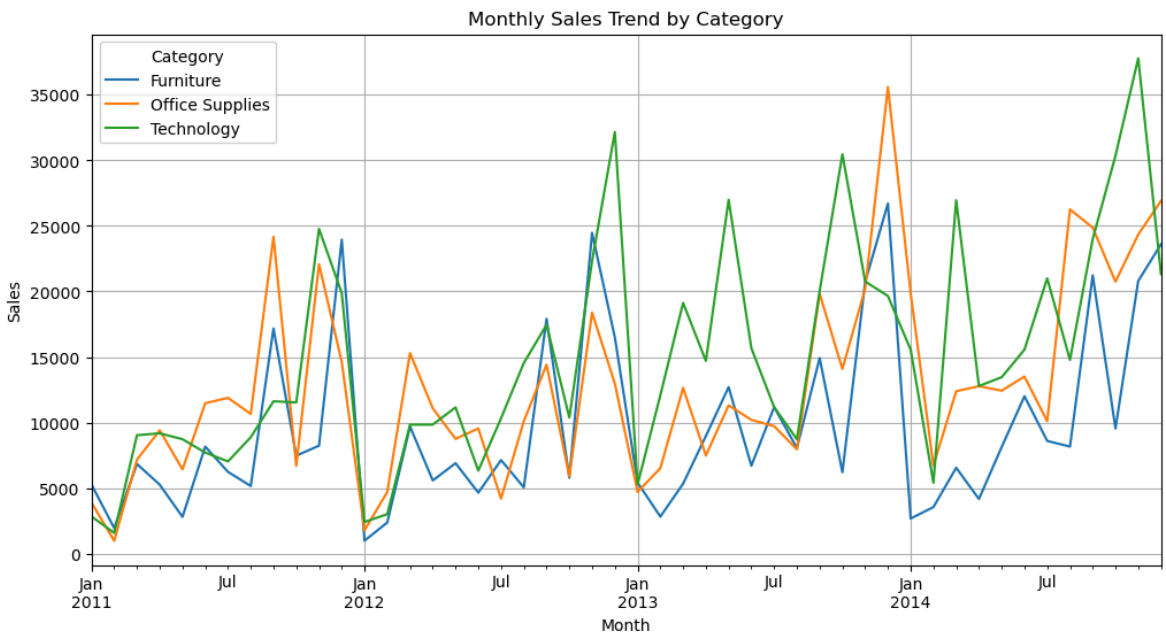


In [47]:

```
df['Month'] = df.index.month
monthly_avg_sales = df.groupby('Month')['Sales'].mean()

monthly_avg_sales.plot(kind='bar', figsize=(8, 4), color='orange')
plt.title('Average Sales by Month (Seasonality)')
plt.xlabel('Month')
plt.ylabel('Average Sales')
plt.xticks(rotation=0)
plt.show()
```



In [48]:
```
category_sales = df.groupby([df.index.to_period('M'), 'Category'])['Sales'].sum(

category_sales.plot(figsize=(12, 6))
plt.title('Monthly Sales Trend by Category')
plt.ylabel('Sales')
plt.xlabel('Month')
plt.legend(title='Category')
plt.grid(True)
plt.show()
```



🔍 Insight:

- Sales consistently **peak in Q4** (October to December), likely due to seasonal promotions or holidays.
- The trend shows **year-over-year growth**, especially between 2012 and 2014.
- Minor dips in mid-year months (May–July) suggest opportunities to boost promotions in those periods.

- Slower average performance in months like **July and September**.
- **Technology** shows the most **consistent and high sales**, especially in Q4.

📌 Recommendations

- **Increase marketing** spend in **Q3** to capitalize on predictable Q4 spikes.
- Consider adding **campaigns** or **discounts** in slower months to lift the baseline.
- Prioritize **inventory and marketing focus on Technology** during Q4.
- **Monitor Office Supplies trends** for school/business seasonality to time promotions better.

👥 Customer Segmentation Analysis

✅ Summary

Customer segmentation helps identify patterns in purchasing behavior across different customer groups. The dataset categorizes customers into three key segments:

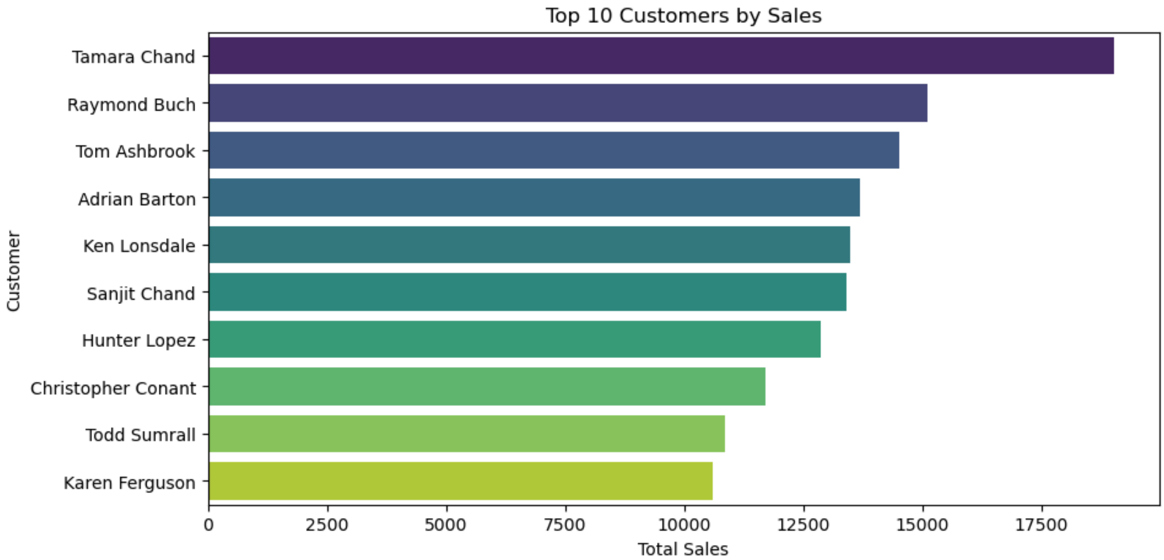
- **Consumer**
- **Corporate**
- **Home Office**

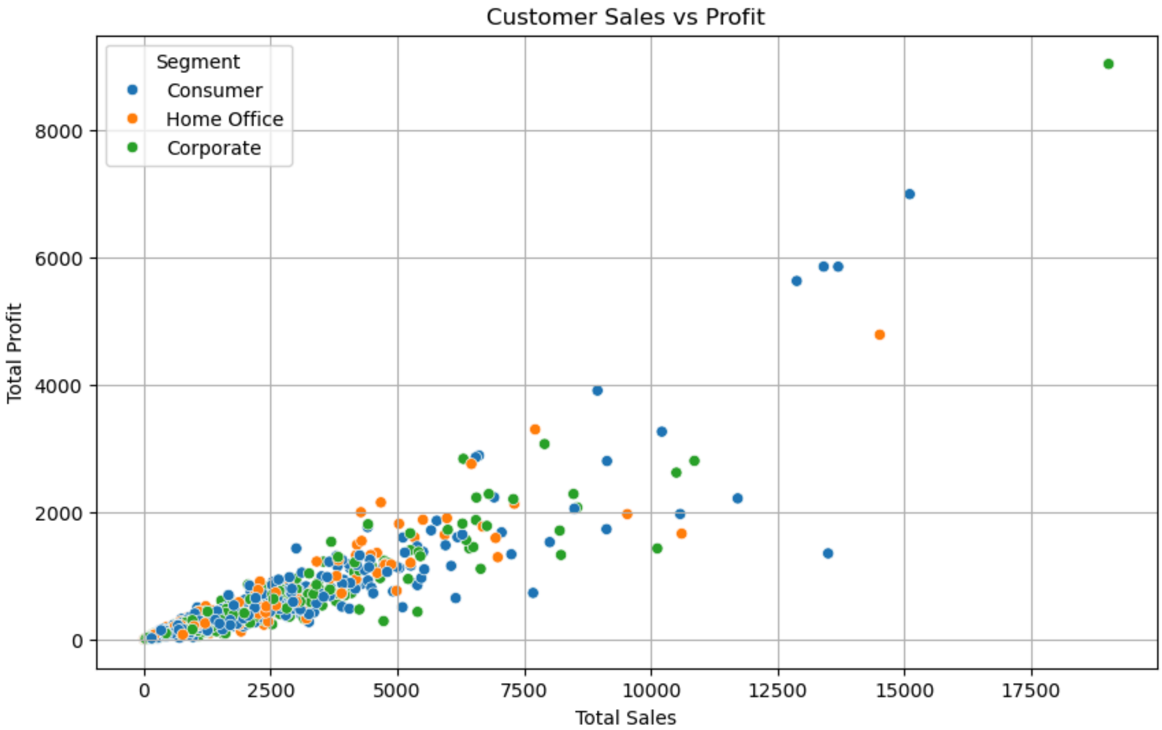
Analyzing their transaction volume and revenue contributions allows us to tailor business strategies for marketing, pricing, and retention.

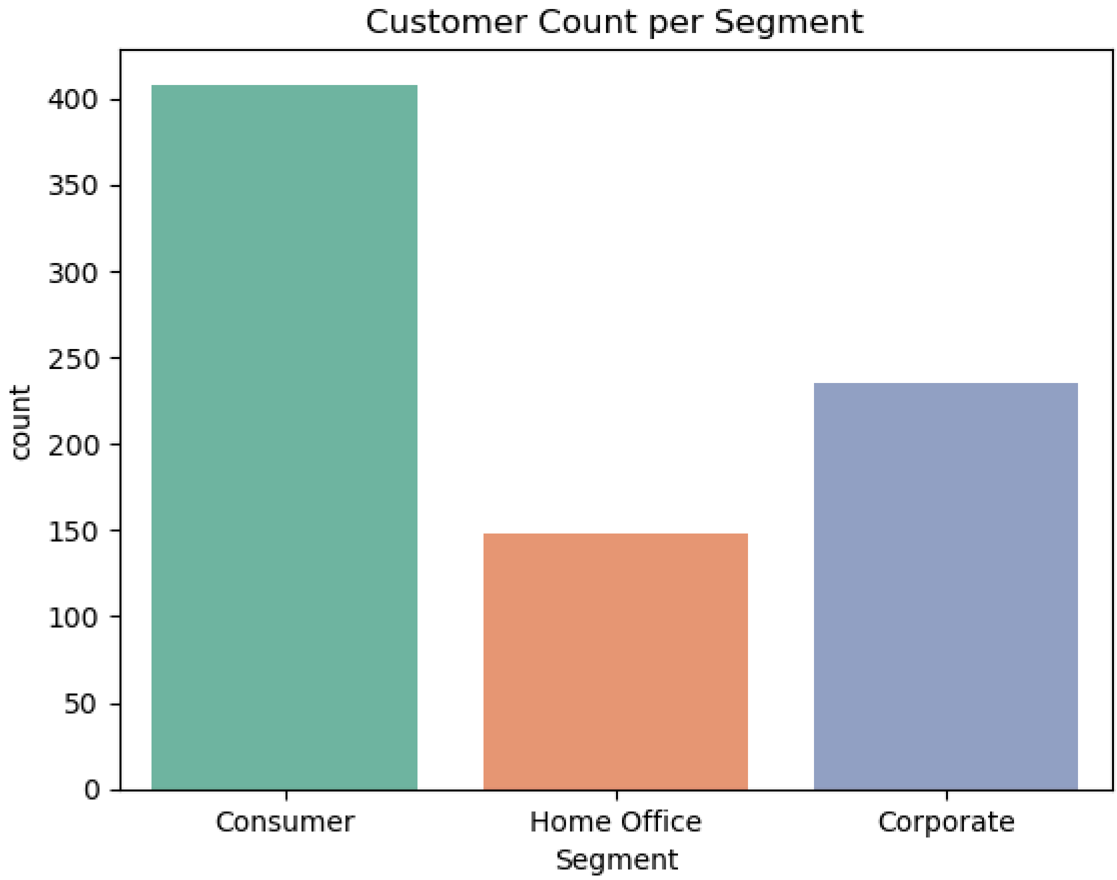
In [49]:
```
customer_df = df.groupby(['Customer ID', 'Customer Name']).agg({
    'Sales': 'sum',
    'Profit': 'sum',
    'Quantity': 'sum',
    'Segment': 'first'  # assumes one segment per customer
}).reset_index()

customer_df['Profit Margin %'] = (customer_df['Profit'] / customer_df['Sales'])
```

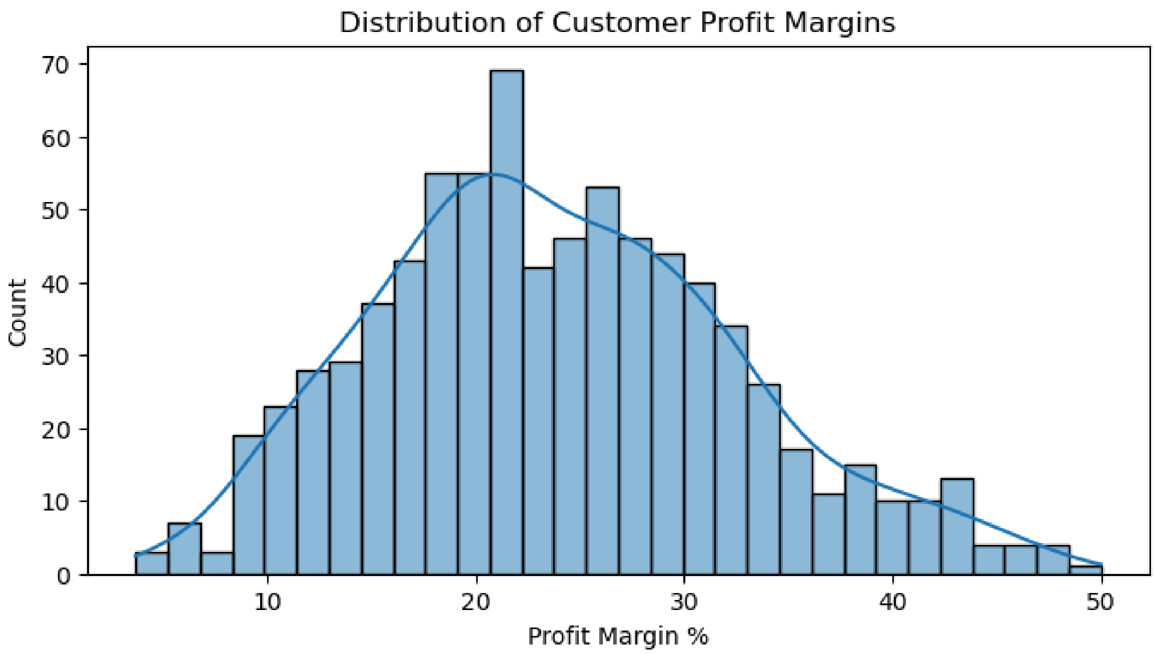
In [50]:
```
top_customers = customer_df.sort_values('Sales', ascending=False).head(10)

plt.figure(figsize=(10, 5))
sns.barplot(data=top_customers, x='Sales', y='Customer Name', palette='viridis')
plt.title('Top 10 Customers by Sales')
plt.xlabel('Total Sales')
plt.ylabel('Customer')
plt.show()



plt.figure(figsize=(10, 6))
sns.scatterplot(data=customer_df, x='Sales', y='Profit', hue='Segment')
plt.title('Customer Sales vs Profit')
```

```
plt.xlabel('Total Sales')
plt.ylabel('Total Profit')
plt.grid(True)
plt.show()



sns.countplot(data=customer_df, x='Segment', palette='Set2')
plt.title('Customer Count per Segment')
plt.show()

plt.figure(figsize=(8, 4))
sns.histplot(customer_df['Profit Margin %'], kde=True, bins=30)
plt.title('Distribution of Customer Profit Margins')
plt.xlabel('Profit Margin %')
plt.show()
```









🔍 Key Insights

- The **Consumer segment** contributes the **highest number of orders and sales**, indicating it's the primary revenue driver.
- **Corporate** and **Home Office** segments have **lower volume** but may represent **higher-value transactions** per order.
- Revenue per order may differ across segments — worth deeper profitability analysis.
- The segment mix may vary across regions or product categories, offering room for personalization.

📌 Recommendations

1. **Consumer Loyalty Programs**: Introduce a reward or referral system to maintain dominance in this segment.
2. **Corporate Upselling**: Bundle offers and enterprise packages can unlock greater value from corporate clients.
3. **Nurture Home Office Segment**: Target this underutilized group with flexible pricing or starter kits for small businesses.
4. **Segmented Marketing Campaigns**: Run personalized marketing (e.g., email campaigns) by segment to improve engagement and retention.

📦 Product Performance Analysis

✅ Summary

Analyzing product performance helps uncover which items drive revenue and which may hurt overall profitability. This includes examining product-level trends across:

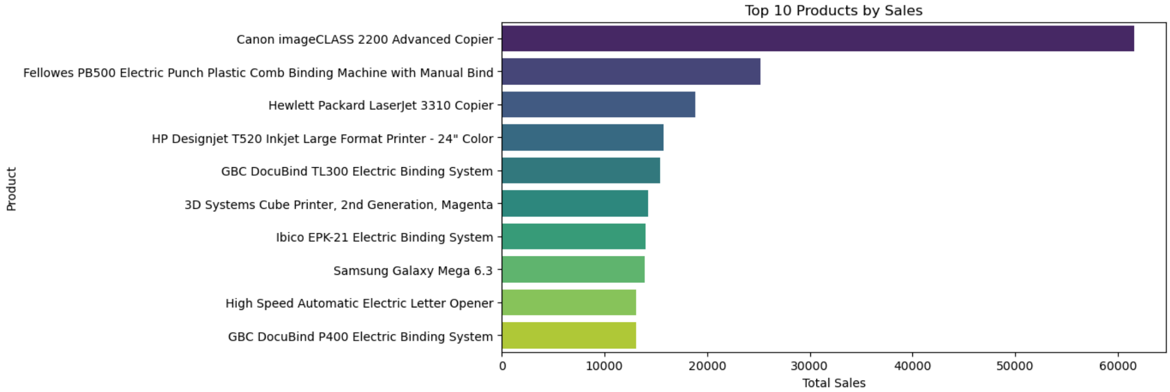
- 📁 Categories (Furniture, Office Supplies, Technology)
- 🧩 Sub-Categories (e.g., Chairs, Binders, Phones)
- 🏷️ Individual products (using Product Name & ID)

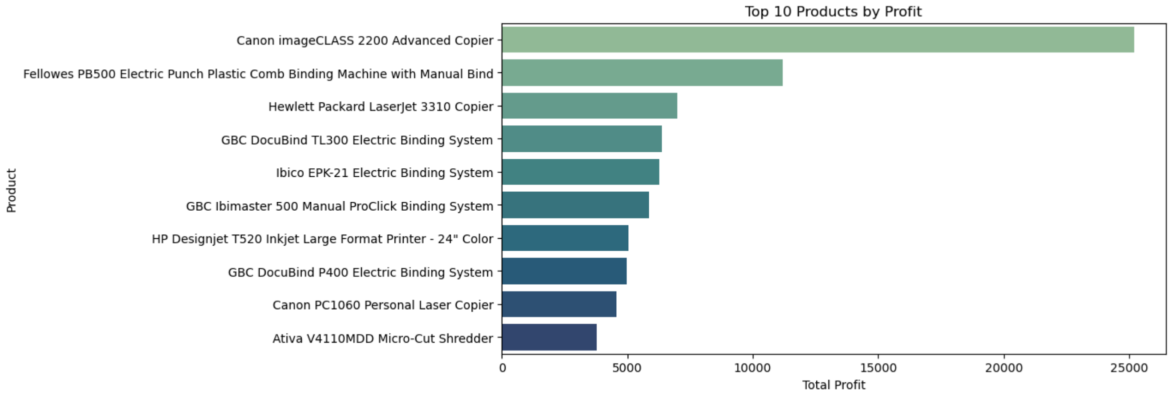
Both **sales volume** and **profit contribution** are analyzed to guide inventory, marketing, and pricing decisions.

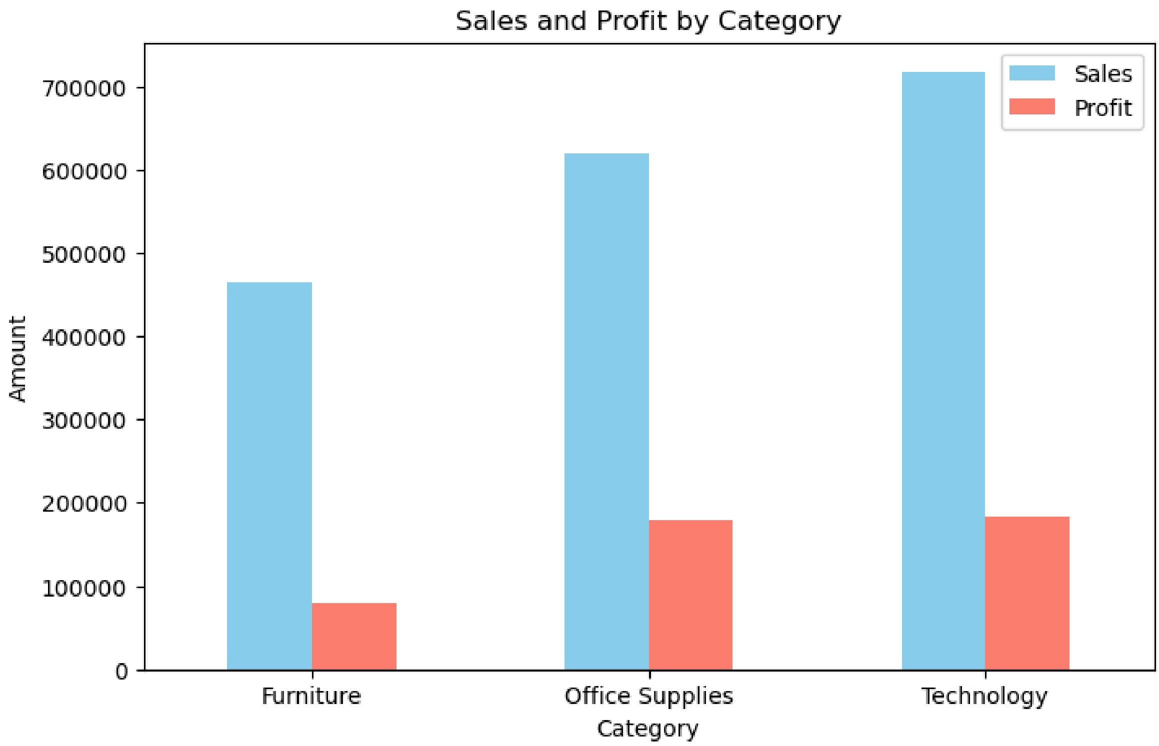
In [51]:
```
product_df = df.groupby(['Product ID', 'Product Name', 'Category', 'Sub-Category
    'Sales': 'sum',
    'Quantity': 'sum',
    'Profit': 'sum',
}).reset_index()

product_df['Profit Margin %'] = (product_df['Profit'] / product_df['Sales']) * 1
```

In [52]:
```
top_sales = product_df.sort_values('Sales', ascending=False).head(10)

plt.figure(figsize=(10, 5))
sns.barplot(data=top_sales, y='Product Name', x='Sales', palette='viridis')
plt.title('Top 10 Products by Sales')
plt.xlabel('Total Sales')
plt.ylabel('Product')
plt.show()



top_profit = product_df.sort_values('Profit', ascending=False).head(10)
```

```
plt.figure(figsize=(10, 5))
sns.barplot(data=top_profit, y='Product Name', x='Profit', palette='crest')
plt.title('Top 10 Products by Profit')
plt.xlabel('Total Profit')
plt.ylabel('Product')
plt.show()



category_perf = df.groupby('Category')[['Sales', 'Profit']].sum().reset_index()

category_perf.plot(kind='bar', x='Category', figsize=(8, 5), color=['skyblue', '
plt.title('Sales and Profit by Category')
plt.ylabel('Amount')
plt.xticks(rotation=0)
plt.show()
```







🔍 Key Insights

- **Office Supplies** category has the **highest number of orders** but relatively **lower profit margins**.
- **Technology** products (e.g., phones, copiers) bring **higher revenue and profitability** despite lower volume.
- Certain **sub-categories like Binders and Paper** are frequently sold but may yield **low or negative margins**.

📌 Recommendations

- **Focus on High-Margin Tech Products**
  Prioritize and promote top-performing tech items (e.g., phones, accessories) through bundling or featured placement.

- **Bundle Low-Profit Items**
  Combine items like paper and binders with higher-value tech products to increase average order profitability.

- **Seasonal Product Strategy**
  If certain products peak at specific times (e.g., office chairs), plan campaigns or procurement accordingly.
