Project Name - Supermart Grocery Sales - Retail Analytics Dataset_ (Data Analyst) (Part 2)

Project Type - Data Analysis

Industry - Unified Mentor

Contribution - Individual

Member Name - Hare Krishana Mishra

Task - 2

Project Summary -

Project Description:

This project analyzes the Supermart Grocery Sales – Retail Analytics Dataset, which contains detailed records of grocery orders placed by customers in Tamil Nadu, India. The dataset includes attributes such as category, sub-category, sales, discount, profit, region, city, and order dates. The focus of this phase of the project is on exploratory data analysis (EDA) and data visualization, enabling insights into sales performance, profitability, seasonal trends, and regional contributions. By using Python libraries like Pandas, Matplotlib, and Seaborn, the data is cleaned, transformed, and visualized to uncover patterns that can help improve sales strategies and business decision-making.

Objective:

- To explore and understand the distribution of sales across various product categories, sub-categories, cities, and regions.
- To identify time-based sales trends (monthly, yearly) and detect seasonal variations.
- To analyze the relationship between sales and profit, highlighting the most profitable product lines.
- To discover high-performing locations and categories that drive revenue growth.
- To present actionable insights for marketing, inventory management, and strategic planning.

Key Project Details:

Dataset Origin: Fictional dataset created for data analytics practice.

Data Coverage: Orders from customers in Tamil Nadu, India.

Key Columns: Order ID, Customer Name, Category, Sub Category, City, Order Date, Region, Sales, Discount, Profit, State, Month, Year.

Tools Used: Python, Pandas, Matplotlib, Seaborn, NumPy.

Analysis Performed:

- Grouped sales by category, month, and year.
- Visualized sales contributions using bar charts, line charts, and pie charts.
- Identified top-performing cities based on total sales.
- Highlighted best-selling categories for strategic investment.

Outcome: Clear understanding of sales trends, regional performance, and category-wise profitability, enabling data-driven decisions.

Let's Begin:-

In [32]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
```

Load the Dataset

In [33]:
```
df=pd.read_csv('/content/Supermart Grocery Sales - Retail Analytics Dataset
```

Data Preprocessing

In [34]:
```
 #display the first five rows of the data
 df.head()
```

Out[34]:

	Order ID	Customer Name	Category	Sub Category	City	Order Date	Region	Sales	I
0	OD1	Harish	Oil & Masala	Masalas	Vellore	11-08-2017	North	1254	
1	OD2	Sudha	Beverages	Health Drinks	Krishnagiri	11-08-2017	South	749	
2	OD3	Hussain	Food Grains	Atta & Flour	Perambalur	06-12-2017	West	2360	
3	OD4	Jackson	Fruits & Veggies	Fresh Vegetables	Dharmapuri	10-11-2016	South	896	
4	OD5	Ridhesh	Food Grains	Organic Staples	Ooty	10-11-2016	South	2355	

In [35]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9994 entries, 0 to 9993
Data columns (total 11 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Order ID       9994 non-null   object 
 1   Customer Name  9994 non-null   object 
 2   Category       9994 non-null   object 
 3   Sub Category   9994 non-null   object 
 4   City           9994 non-null   object 
 5   Order Date     9994 non-null   object 
 6   Region         9994 non-null   object 
 7   Sales          9994 non-null   int64  
 8   Discount       9994 non-null   float64
 9   Profit         9994 non-null   float64
 10  State          9994 non-null   object 
dtypes: float64(2), int64(1), object(8)
memory usage: 859.0+ KB
```

In [36]:
```
df['Order Date'] = pd.to_datetime(df['Order Date'], errors='coerce')
```

In [37]:
```
#changed to date data type
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9994 entries, 0 to 9993
Data columns (total 11 columns):
 #   Column         Non-Null Count  Dtype         
---  ------         --------------  -----         
 0   Order ID       9994 non-null   object        
 1   Customer Name  9994 non-null   object        
 2   Category       9994 non-null   object        
 3   Sub Category   9994 non-null   object        
 4   City           9994 non-null   object        
 5   Order Date     4042 non-null   datetime64[ns]
 6   Region         9994 non-null   object        
 7   Sales          9994 non-null   int64         
 8   Discount       9994 non-null   float64       
 9   Profit         9994 non-null   float64       
 10  State          9994 non-null   object        
dtypes: datetime64[ns](1), float64(2), int64(1), object(7)
memory usage: 859.0+ KB
```

In [38]:
```
# applying groupby() function to
# group the data on Category.
da=df.groupby("Category")
da.first()
```

Out[38]:

	Order ID	Customer Name	Sub Category	City	Order Date	Region	Sales	Di
Category								
Bakery	OD9	Hafiz	Biscuits	Tirunelveli	2015-06-09	West	791	
Beverages	OD2	Sudha	Health Drinks	Krishnagiri	2017-11-08	South	749	
Eggs, Meat & Fish	OD12	Yadav	Eggs	Namakkal	2015-06-09	West	701	
Food Grains	OD3	Hussain	Atta & Flour	Perambalur	2017-06-12	West	2360	
Fruits & Veggies	OD4	Jackson	Fresh Vegetables	Dharmapuri	2016-10-11	South	896	
Oil & Masala	OD1	Harish	Masalas	Vellore	2017-11-08	North	1254	
Snacks	OD11	Ganesh	Chocolates	Karur	2015-06-09	West	1903	

In [39]:
```
# Convert Order Date to datetime format
df['Order Date'] = pd.to_datetime(df['Order Date'], errors='coerce')

# Extract month, month name, and year
df['month_no'] = df['Order Date'].dt.month
df['Month'] = df['Order Date'].dt.strftime('%B')
df['year'] = df['Order Date'].dt.year
```

```python
# Check the data to view the added columns
df.head()
```

Out[39]:

	Order ID	Customer Name	Category	Sub Category	City	Order Date	Region	Sales
0	OD1	Harish	Oil & Masala	Masalas	Vellore	2017-11-08	North	1254
1	OD2	Sudha	Beverages	Health Drinks	Krishnagiri	2017-11-08	South	749
2	OD3	Hussain	Food Grains	Atta & Flour	Perambalur	2017-06-12	West	2360
3	OD4	Jackson	Fruits & Veggies	Fresh Vegetables	Dharmapuri	2016-10-11	South	896
4	OD5	Ridhesh	Food Grains	Organic Staples	Ooty	2016-10-11	South	2355

Exploratory Data Analysis (EDA)

In [40]:

```python
# Sum up sales by month
monthly_sales = df.groupby('Month')['Sales'].sum().reset_index()
```

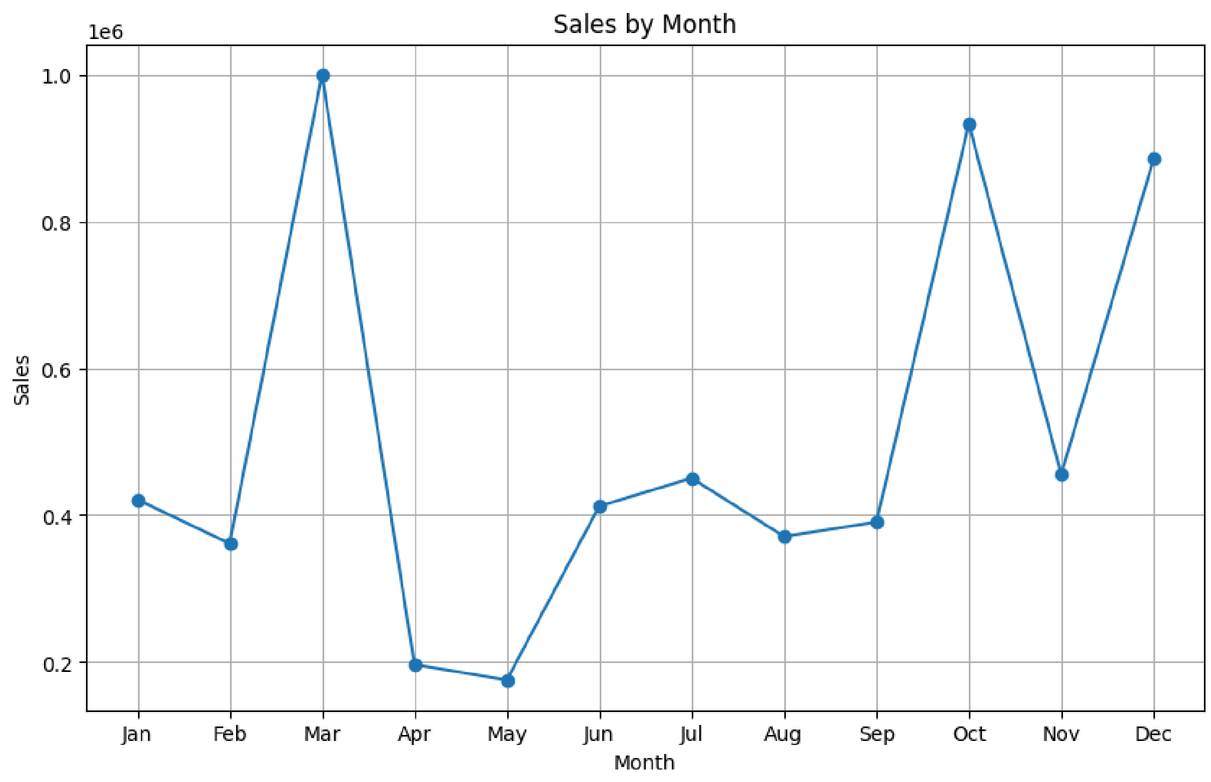
In [41]:

```python
# Sort the data by month
monthly_sales_sorted = monthly_sales.sort_values(by='Month')
```

Monthly Sales Growth Pattern

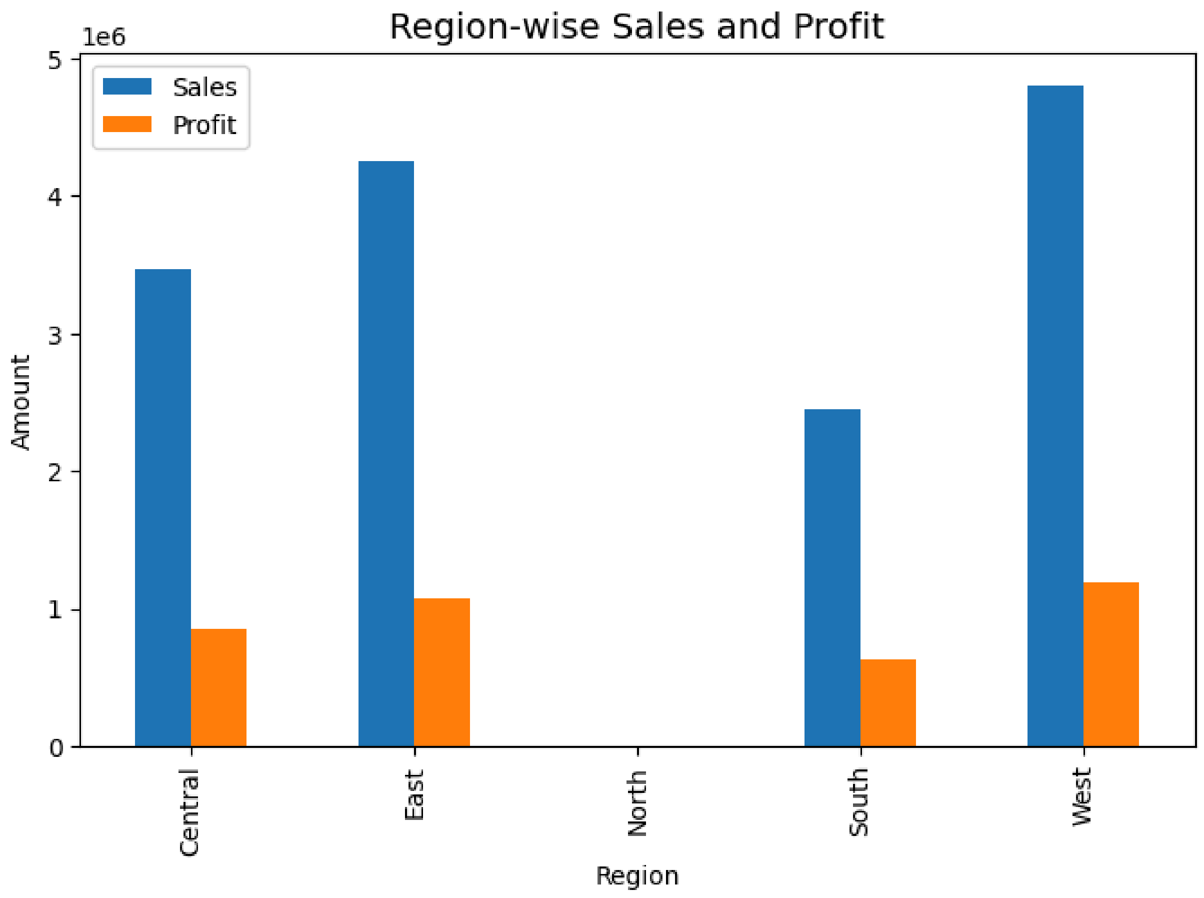
In [42]:

```python
# Create the line chart
plt.figure(figsize=(10, 6))
plt.plot(monthly_sales_sorted['Month'],
monthly_sales_sorted['Sales'], marker='o')
plt.title('Sales by Month')
plt.xlabel('Month')
plt.ylabel('Sales')
plt.xticks(monthly_sales_sorted['Month'], ['Jan', 'Feb', 'Mar',
'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec'])
plt.grid(True)
plt.show()
```



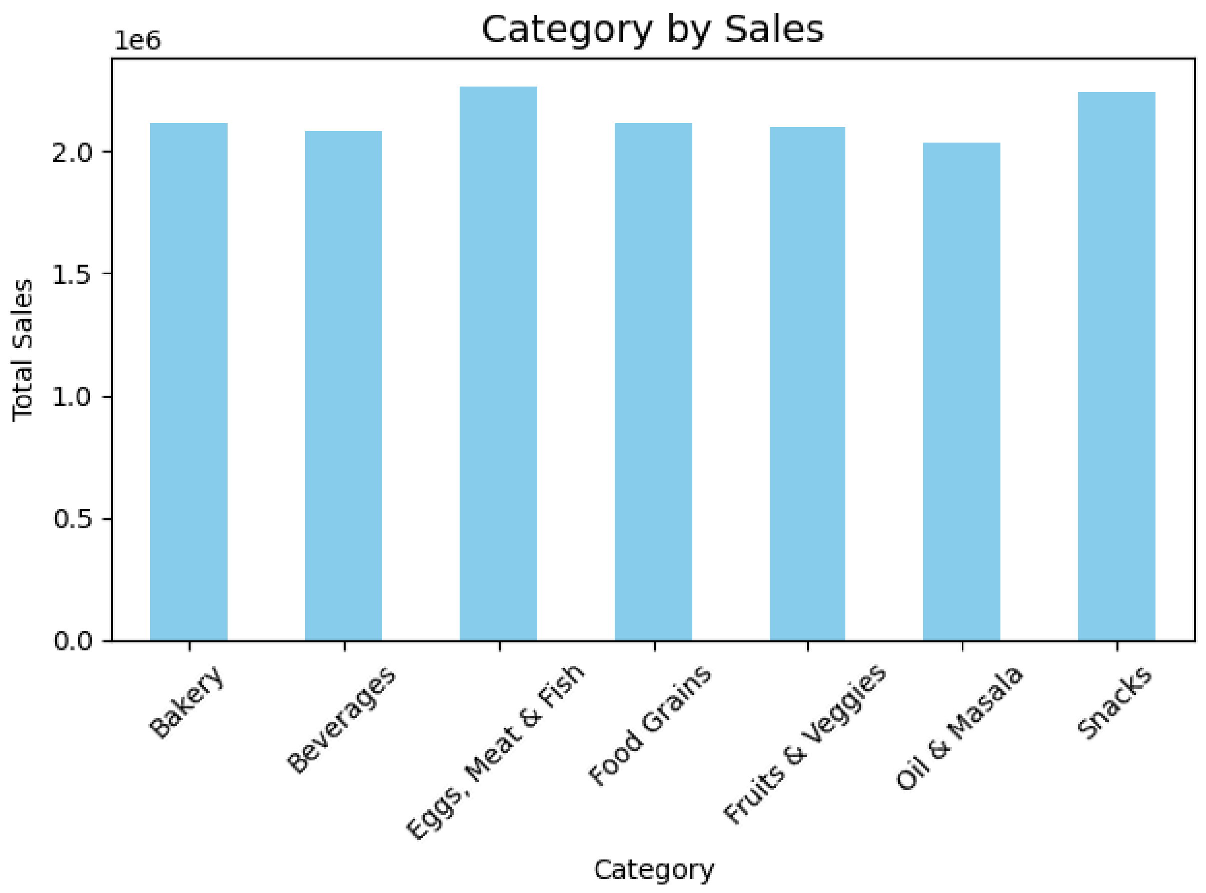
Region-Wise Sales & Profit (Grouped Bar Chart)

```
In [53]: region_stats = df.groupby('Region')[['Sales','Profit']].sum()

         region_stats.plot(kind='bar', figsize=(8,5))
         plt.title("Region-wise Sales and Profit", fontsize=14)
         plt.ylabel("Amount")
         plt.show()
```



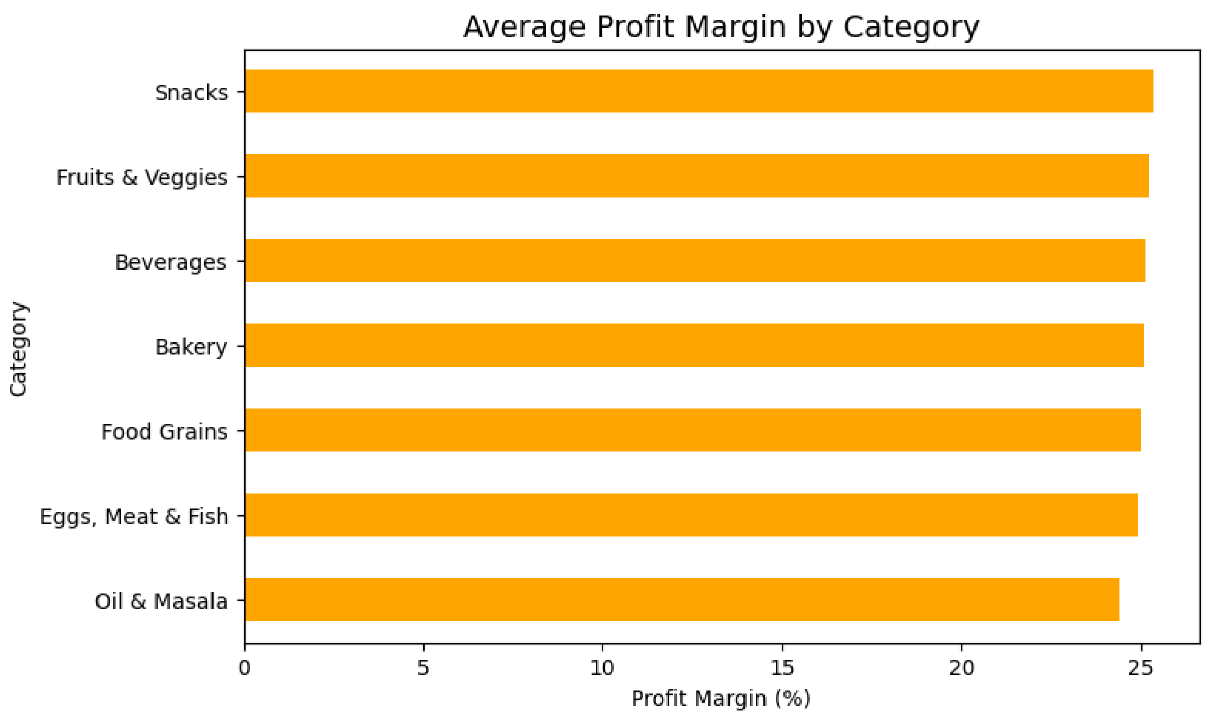
Total Sales by Product Category

```
In [43]: # Group by Category and get total sales
         Sales_category = df.groupby("Category")["Sales"].sum()

         # Plot sales by category
         Sales_category.plot(kind='bar', color='skyblue')
         plt.title('Category by Sales', fontsize=14)
         plt.xlabel('Category')
         plt.ylabel('Total Sales')
         plt.xticks(rotation=45)
         plt.tight_layout()
         plt.show()
```



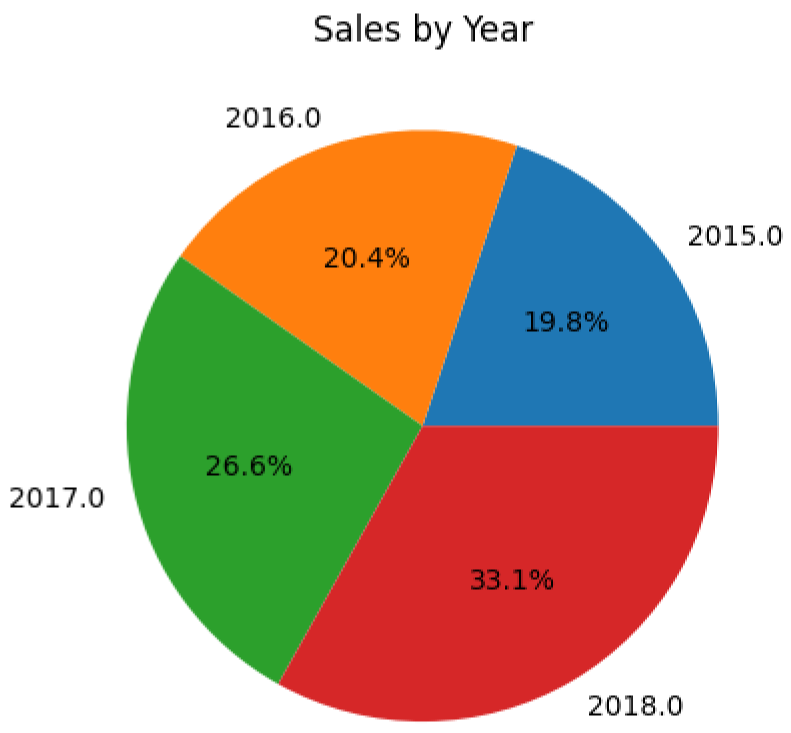
Profit Margin by Category (Horizontal Bar)

In [57]:
```
df['Profit Margin'] = (df['Profit'] / df['Sales']) * 100
profit_margin = df.groupby('Category')['Profit Margin'].mean().sort_values()

profit_margin.plot(kind='barh', color='orange', figsize=(8,5))
plt.title("Average Profit Margin by Category", fontsize=14)
plt.xlabel("Profit Margin (%)")
plt.ylabel("Category")
plt.show()
```



Yearly Sales Contribution

In [44]:
```
#we want to find the Yearly Sales
# we group by Year and get the total number of sales for each year
Yearly_Sales=df.groupby("year")["Sales"].sum()
# we create a pie chart with the sales by year
plt.pie(Yearly_Sales, labels=Yearly_Sales.index,
autopct='%1.1f%%')
plt.title('Sales by Year')
plt.show()
#Monthly_Sales.plot(kind='pie')
#plt.title('Yearly Sales', fontsize = 14)
#plt.show()
```



Top 5 Cities Generating the Highest Sales

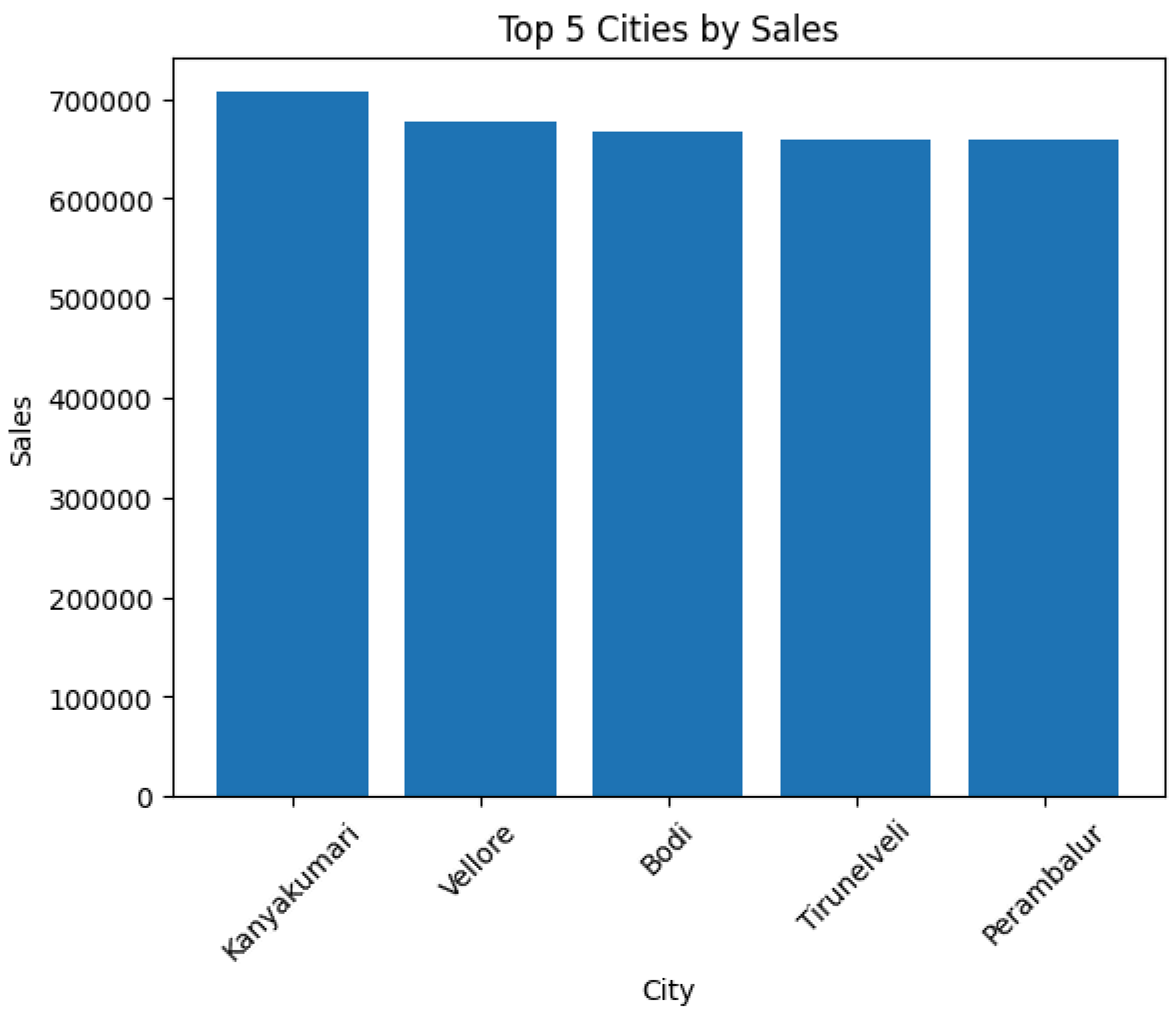
In [45]:
```
# Step 1: Extract relevant columns
city_sales = df[['City', 'Sales']]
```

In [46]:
```
# Step 2: Calculate total sales per city
total_sales = city_sales.groupby('City').sum()
```

In [47]:
```
# Step 3: Sort the cities by sales
sorted_cities = total_sales.sort_values(by='Sales',
ascending=False)
```

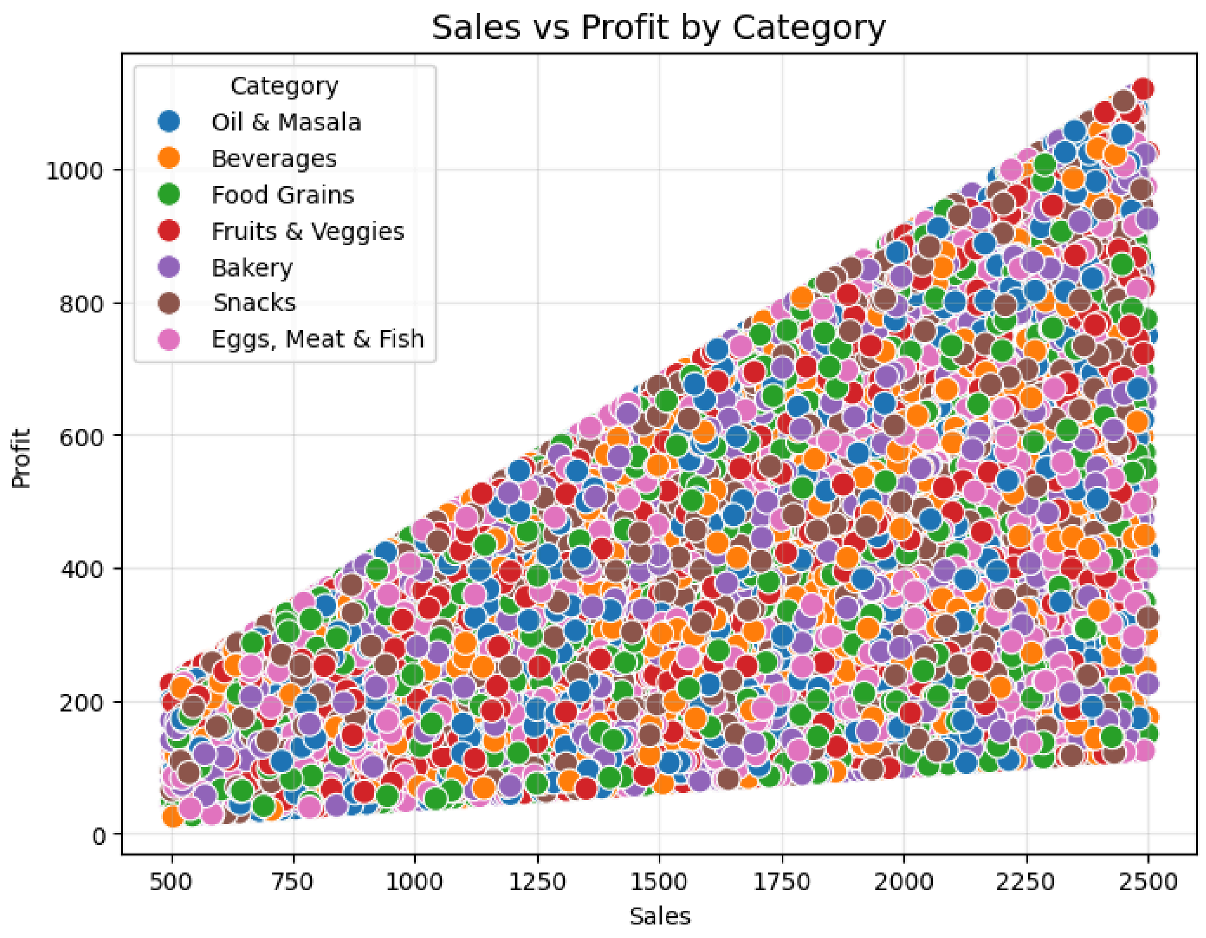
In [48]:
```
# Step 4: Select the top 5 cities
top_cities = sorted_cities.head(5)
```

In [49]:
```
# Step 5: Plot the bar chart
plt.bar(top_cities.index, top_cities['Sales'])
plt.xlabel('City')
plt.ylabel('Sales')
plt.title('Top 5 Cities by Sales')
plt.xticks(rotation=45)
plt.show()
```



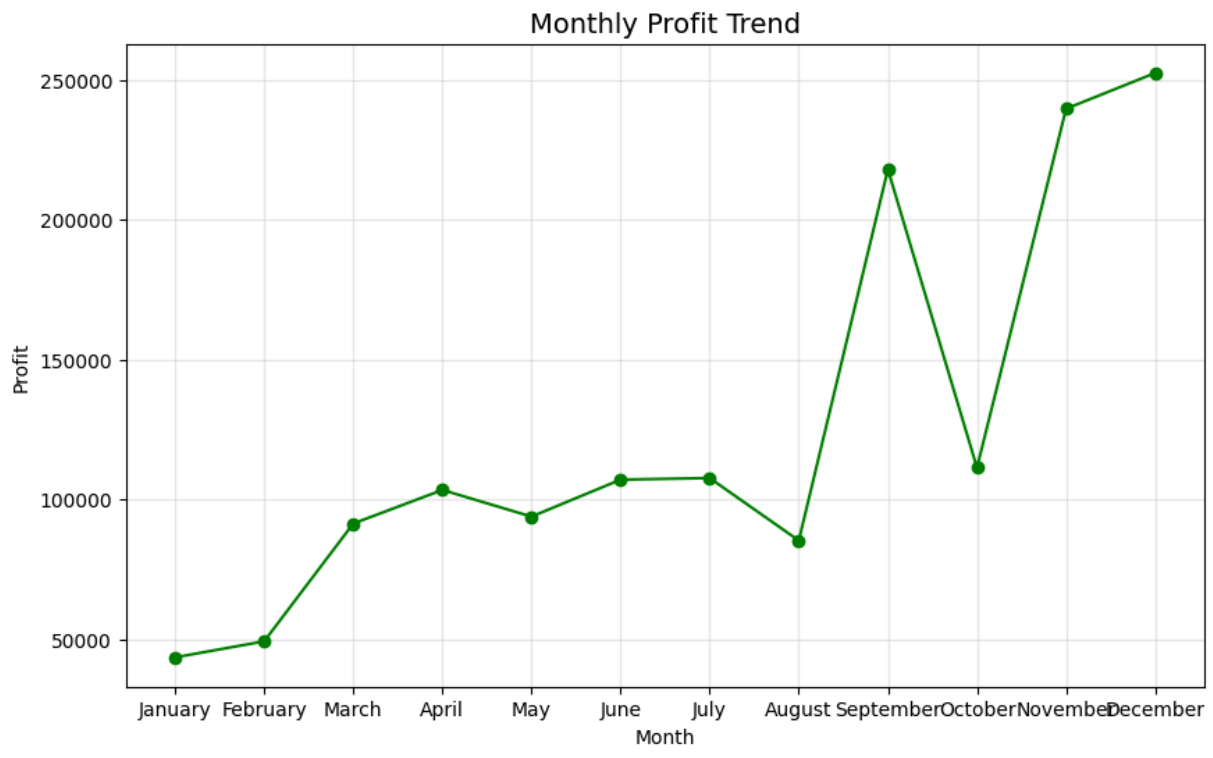
Sales vs. Profit by Category (Scatter Plot)

```
In [50]: plt.figure(figsize=(8,6))
         sns.scatterplot(data=df, x='Sales', y='Profit', hue='Category', s=100)
         plt.title("Sales vs Profit by Category", fontsize=14)
         plt.xlabel("Sales")
         plt.ylabel("Profit")
         plt.grid(True, alpha=0.3)
         plt.show()
```



Monthly Profit Trend (Line Chart)

```
In [52]: monthly_profit = df.groupby('Month')['Profit'].sum().reindex([
             'January','February','March','April','May','June',
             'July','August','September','October','November','December'
         ])

         plt.figure(figsize=(10,6))
         plt.plot(monthly_profit.index, monthly_profit.values, marker='o', color='gre
         plt.title("Monthly Profit Trend", fontsize=14)
         plt.xlabel("Month")
         plt.ylabel("Profit")
         plt.grid(True, alpha=0.3)
         plt.show()
```

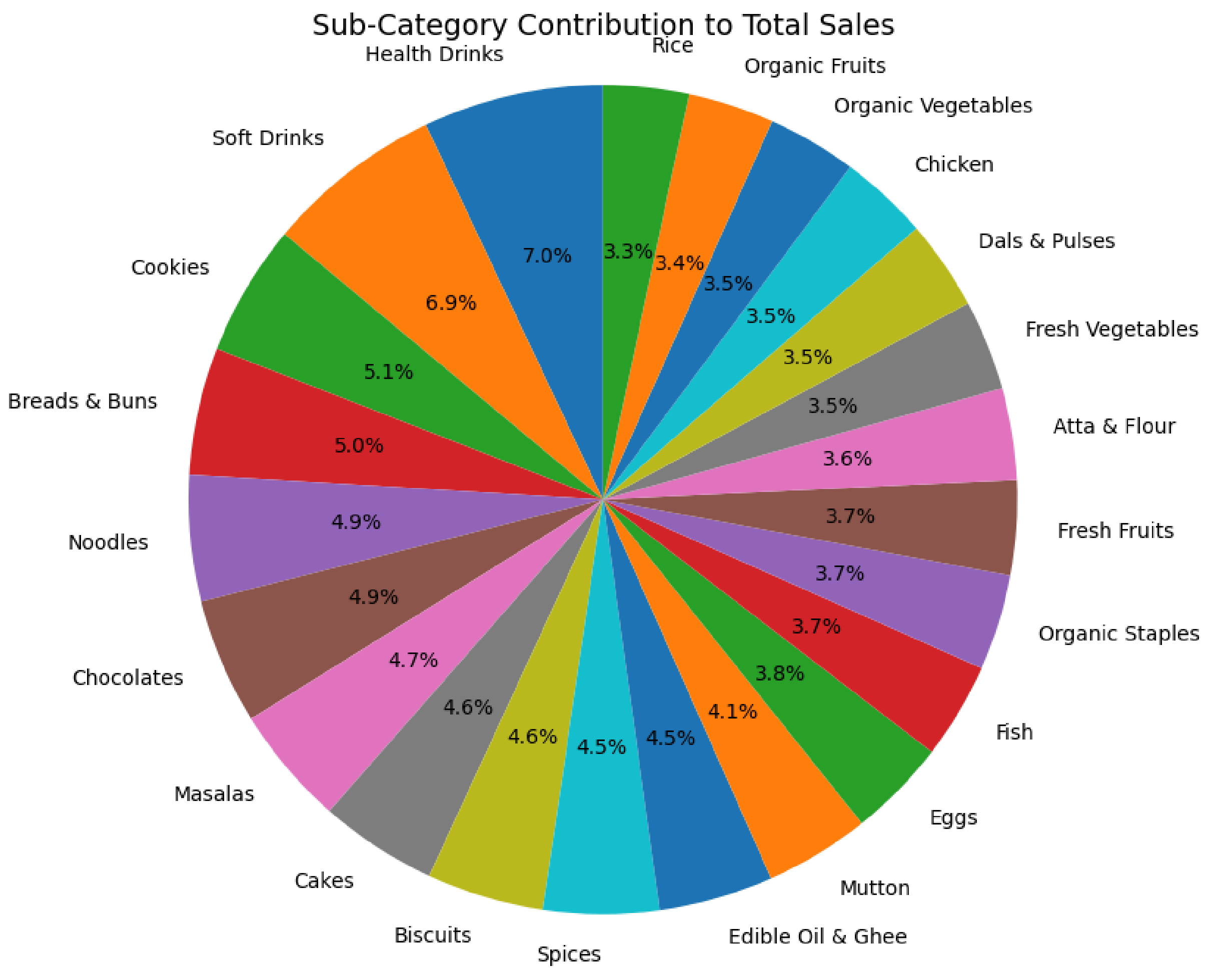


Sub-Category Contribution to Total Sales (Treemap)

```
In [55]: pip install squarify
```

```
Collecting squarify
  Downloading squarify-0.4.4-py3-none-any.whl.metadata (600 bytes)
Downloading squarify-0.4.4-py3-none-any.whl (4.1 kB)
Installing collected packages: squarify
Successfully installed squarify-0.4.4
```

```
In [56]: import matplotlib.pyplot as plt

# Prepare data
sub_sales = df.groupby('Sub Category')['Sales'].sum().sort_values(ascending=

# Create a pie chart styled as a "treemap alternative"
plt.figure(figsize=(8,8))
plt.pie(sub_sales, labels=sub_sales.index, autopct='%1.1f%%', startangle=90)
plt.title("Sub-Category Contribution to Total Sales", fontsize=14)
plt.axis('equal')
plt.show()
```


Sub-Category Contribution to Total Sales
Health Drinks 7.0%
Soft Drinks 6.9%
Cookies 5.1%
Breads & Buns 5.0%
Noodles 4.9%
Chocolates 4.9%
Masalas 4.7%
Cakes 4.6%
Biscuits 4.6%
Spices 4.5%
Edible Oil & Ghee 4.5%
Mutton 4.1%
Eggs 3.8%
Fish 3.7%
Organic Staples 3.7%
Fresh Fruits 3.7%
Atta & Flour 3.6%
Fresh Vegetables 3.5%
Dals & Pulses 3.5%
Chicken 3.5%
Organic Vegetables 3.5%
Organic Fruits 3.4%
Rice 3.3%