EXPLORATORY DATA ANALYSIS

- EDA allows us to explore the data's characteristics, patterns, and outliers, facilitating the identification of crucial factors contributing to eCommerce.
- Through visualizations and analysis of numerical, categorical, and time-related variables, EDA allows for the identification of correlations, anomalies, and potential data quality issues, providing a solid foundation for subsequent modeling and decision-making in the eCommerce industry.
- EDA can uncover patterns and trends in different aspects of eCommerce, such as customer behavior, cart abandonment, or website performance issues. It allows for the identification of common factors contributing to these trends, including user demographics, product pricing, website design, seasonal demand, or promotional effectiveness.

OBJECTIVE

Task 1: Exploratory Data Analysis (EDA) and Business Insights

1. Perform EDA on the provided dataset.

2. Derive at least 5 business insights from the EDA.

Files Description:

1. Customers.csv

○ CustomerID: Unique identifier for each customer.

- CustomerName: Name of the customer.
- Region: Continent where the customer resides.
- SignupDate: Date when the customer signed up.

2. Products.csv

- ProductID: Unique identifier for each product.
- ProductName: Name of the product.
- Category: Product category.
- Price: Product price in USD.

3. Transactions.csv

- TransactionID: Unique identifier for each transaction.
- CustomerID: ID of the customer who made the transaction.
- ProductID: ID of the product sold.
- TransactionDate: Date of the transaction.
- Quantity: Quantity of the product purchased.
- TotalValue: Total value of the transaction.
- Price: Price of the product sold

1. IMPORTING LIBRARIES FOR Exploratory Data Analysis (EDA)

In [3]:
```
import warnings
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore", category=FutureWarning)
```

2. Loading Data With Pandas

In [4]:
```
customers_df    = pd.read_csv(r"C:\Users\Pruthviraj\Desktop\Zeotap\Custome

products_df     = pd.read_csv(r"C:\Users\Pruthviraj\Desktop\Zeotap\Product

transactions_df =  pd.read_csv(r"C:\Users\Pruthviraj\Desktop\Zeotap\Transa
```

Let's look at the first 3 rows of both dataframes to see what the data looks like

In [22]:
```
customers_df.head(3)
```

Out[22]:

	CustomerID	CustomerName	Region	SignupDate
0	C0001	Lawrence Carroll	South America	2022-07-10
1	C0002	Elizabeth Lutz	Asia	2022-02-13
2	C0003	Michael Rivera	South America	2024-03-07

In [15]:
```
products_df.head(3)
```

Out[15]:

	ProductID	ProductName	Category	Price
0	P001	ActiveWear Biography	Books	169.30
1	P002	ActiveWear Smartwatch	Electronics	346.30
2	P003	ComfortLiving Biography	Books	44.12

In [14]:
```
transactions_df.head(3)
```

Out[14]:

	TransactionID	CustomerID	ProductID	TransactionDate	Quantity	TotalValue	Price
0	T00001	C0199	P067	2024-08-25 12:38:23	1	300.68	300.68
1	T00112	C0146	P067	2024-05-27 22:23:54	1	300.68	300.68
2	T00166	C0127	P067	2024-04-25 07:38:55	1	300.68	300.68

3. Descriptive Statistics of Data

Data Types.

It is useful to first understand the data that you're dealing with along with the data types of each column. The data types may dictate how you transform and engineer features.

In [23]:
```
customers_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 4 columns):
 #   Column        Non-Null Count  Dtype 
---  ------        --------------  ----- 
 0   CustomerID    200 non-null    object
 1   CustomerName  200 non-null    object
 2   Region        200 non-null    object
 3   SignupDate    200 non-null    object
dtypes: object(4)
memory usage: 6.4+ KB
```

In [24]:
```
products_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 4 columns):
 #   Column       Non-Null Count  Dtype 
---  ------       --------------  ----- 
 0   ProductID    100 non-null    object
 1   ProductName  100 non-null    object
 2   Category     100 non-null    object
 3   Price        100 non-null    float64
dtypes: float64(1), object(3)
memory usage: 3.2+ KB
```

In [25]:
```
transactions_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 7 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   TransactionID    1000 non-null   object 
 1   CustomerID       1000 non-null   object 
 2   ProductID        1000 non-null   object 
 3   TransactionDate  1000 non-null   object 
 4   Quantity         1000 non-null   int64  
 5   TotalValue       1000 non-null   float64
 6   Price            1000 non-null   float64
dtypes: float64(2), int64(1), object(4)
memory usage: 54.8+ KB
```

You can see that all of the `TransactionDate`, & `SignupDate` related columns are not currently in datetime format. We will need to convert these later.

Statistics

Now let's look at some statistics about the datasets

In [26]:
```
customers_df.describe()
```

Out[26]:

	CustomerID	CustomerName	Region	SignupDate
count	200	200	200	200
unique	200	200	4	179
top	C0001	Lawrence Carroll	South America	2024-11-11
freq	1	1	59	3

In [27]:
```
products_df.describe()
```

Out[27]:

	Price
count	100.000000
mean	267.551700
std	143.219383
min	16.080000
25%	147.767500
50%	292.875000
75%	397.090000
max	497.760000

In [28]:
```
transactions_df.describe()
```

Out[28]:

	Quantity	TotalValue	Price
count	1000.000000	1000.000000	1000.00000
mean	2.537000	689.995560	272.55407
std	1.117981	493.144478	140.73639
min	1.000000	16.080000	16.08000
25%	2.000000	295.295000	147.95000
50%	3.000000	588.880000	299.93000
75%	4.000000	1011.660000	404.40000
max	4.000000	1991.040000	497.76000

Overall the customers, products,transactions data looks good.

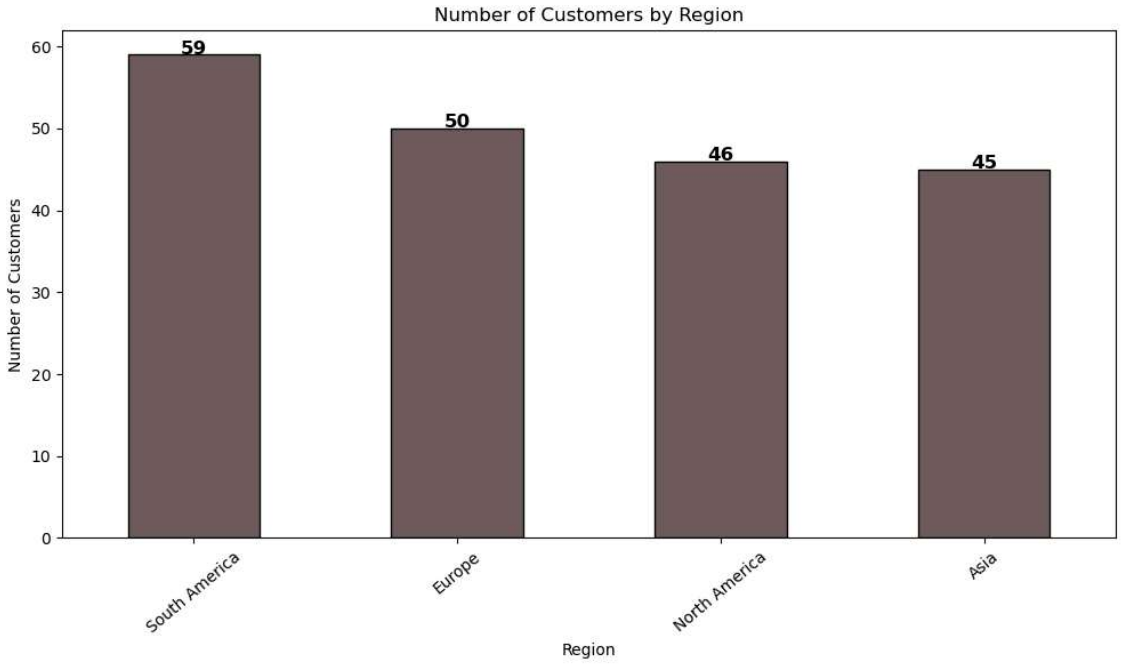
In transactions data for 'Quantity' Column Mean & 50% are not far they both look closer as same like for 'TotalValue & Price Column'.

A] take some Insights from customers Data

In [29]:
```
region_counts = customers_df["Region"].value_counts()
print("region_counts", region_counts)
```

```
region_counts South America    59
Europe           50
North America    46
Asia             45
Name: Region, dtype: int64
```

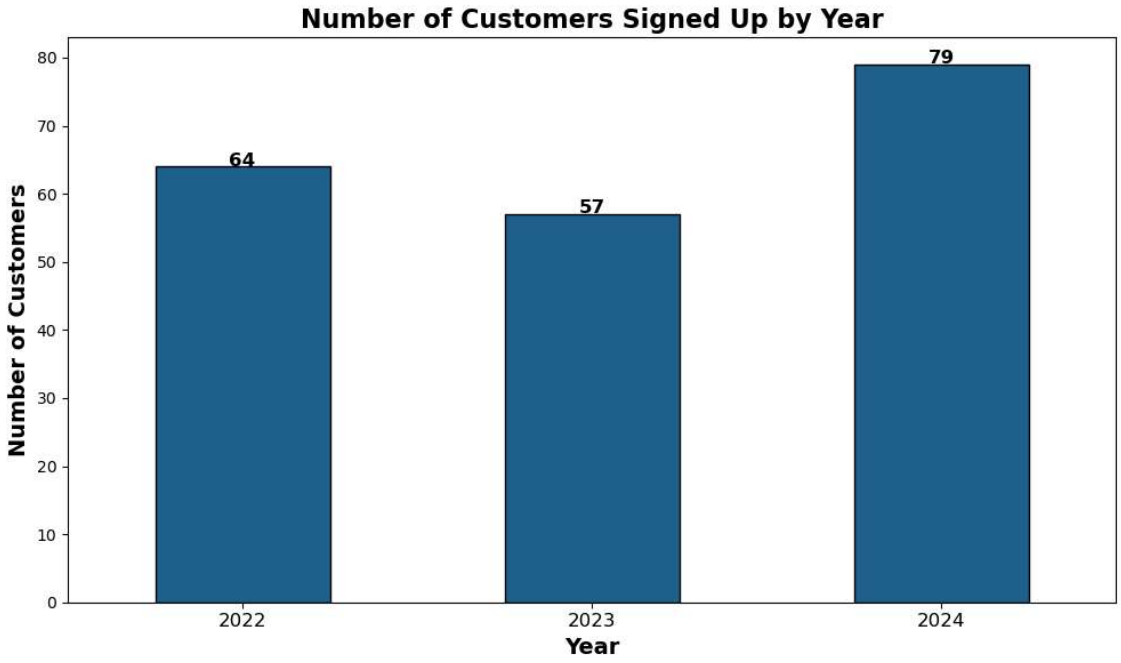
In [38]:
```
region_counts.plot(kind="bar", color = '#6F5E5C',edgecolor="black", figsiz
plt.title("Number of Customers by Region")
plt.xlabel("Region")
plt.ylabel("Number of Customers")
plt.xticks(rotation=40)
for index, value in enumerate(region_counts):
    plt.text(index, value + 0.1, str(value), ha="center", fontsize=12, wei

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



The customer distribution across regions shows minimal differences. To grow further, focus on attracting more customers from Asia, Europe, and North America through targeted marketing and region-specific strategies.

In [39]:

```
# Convert SignupDate to datetime format
customers_df["SignupDate"] = pd.to_datetime(customers_df["SignupDate"])

# Extract year from SignupDate
customers_df["Year"] = customers_df["SignupDate"].dt.year

yearly_counts = customers_df["Year"].value_counts().sort_index()
#=============================================================================

# Plot the bar chart
plt.figure(figsize=(10, 6))
bars = yearly_counts.plot(kind="bar", color="#1F618D", edgecolor="black")
#=============================================================================
# Add titles and labels
plt.title("Number of Customers Signed Up by Year", fontsize=16, weight="bo
plt.xlabel("Year", fontsize=14, weight="bold")
plt.ylabel("Number of Customers", fontsize=14, weight="bold")
plt.xticks(rotation=0, fontsize=12)

for index, value in enumerate(yearly_counts):
    plt.text(index, value + 0.1, str(value), ha="center", fontsize=12, wei

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



Customer signups were low in 2023, suggesting potential issues with our eCommerce services. The significant increase in signups in 2024 indicates successful strategies or improvements. Analyzing these factors will help attract more customers consistently and address challenges in underperforming years.

B] take some Insights from products & transactions Data

In [41]:
```
products_df.head(3)
```

Out[41]:

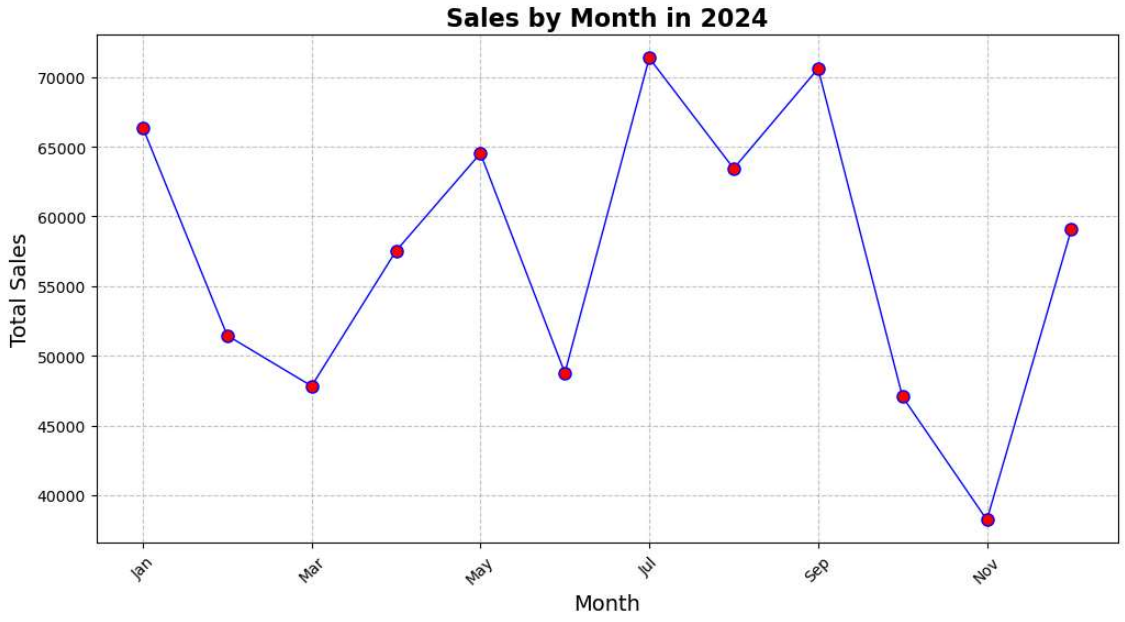
	ProductID	ProductName	Category	Price
0	P001	ActiveWear Biography	Books	169.30
1	P002	ActiveWear Smartwatch	Electronics	346.30
2	P003	ComfortLiving Biography	Books	44.12

In [42]:
```
transactions_df.head(3)
```

Out[42]:

	TransactionID	CustomerID	ProductID	TransactionDate	Quantity	TotalValue	Price
0	T00001	C0199	P067	2024-08-25 12:38:23	1	300.68	300.68
1	T00112	C0146	P067	2024-05-27 22:23:54	1	300.68	300.68
2	T00166	C0127	P067	2024-04-25 07:38:55	1	300.68	300.68

In [46]:
```
# Convert the TransactionDate to datetime format
transactions_df['TransactionDate'] = pd.to_datetime(transactions_df['Trans

# Filter transactions for 2024
transactions_2024 = transactions_df[transactions_df['TransactionDate'].dt.

# 1) Sell in every month of 2024
monthly_sales = transactions_2024.groupby(transactions_2024['TransactionDa

# Ensure all months (1-12) are included, even with 0 sales
monthly_sales = monthly_sales.reindex(range(1, 13), fill_value=0)

# Convert month number to month name
monthly_sales.index = monthly_sales.index.to_series().apply(lambda x: pd.t

# Plot: Sell in every month of 2024
plt.figure(figsize=(12, 6))
monthly_sales.plot(kind='line', color='b', marker='o', linewidth=1, marker

plt.title("Sales by Month in 2024", fontsize=16, fontweight='bold')
plt.xlabel("Month", fontsize=14)
plt.ylabel("Total Sales", fontsize=14)
plt.xticks(rotation=45)
plt.grid(True, linestyle='--', alpha=0.7)

plt.show()
```



Months with Sales Above 60,000:

In January, May, July, August, and September, the total sales exceeded 60,000, indicating strong sales performance in these months.

Months with Sales Below 60,000:

February, March, April, June, and October had lower sales, all falling below the 60,000 mark, suggesting a decrease in consumer activity during these months.

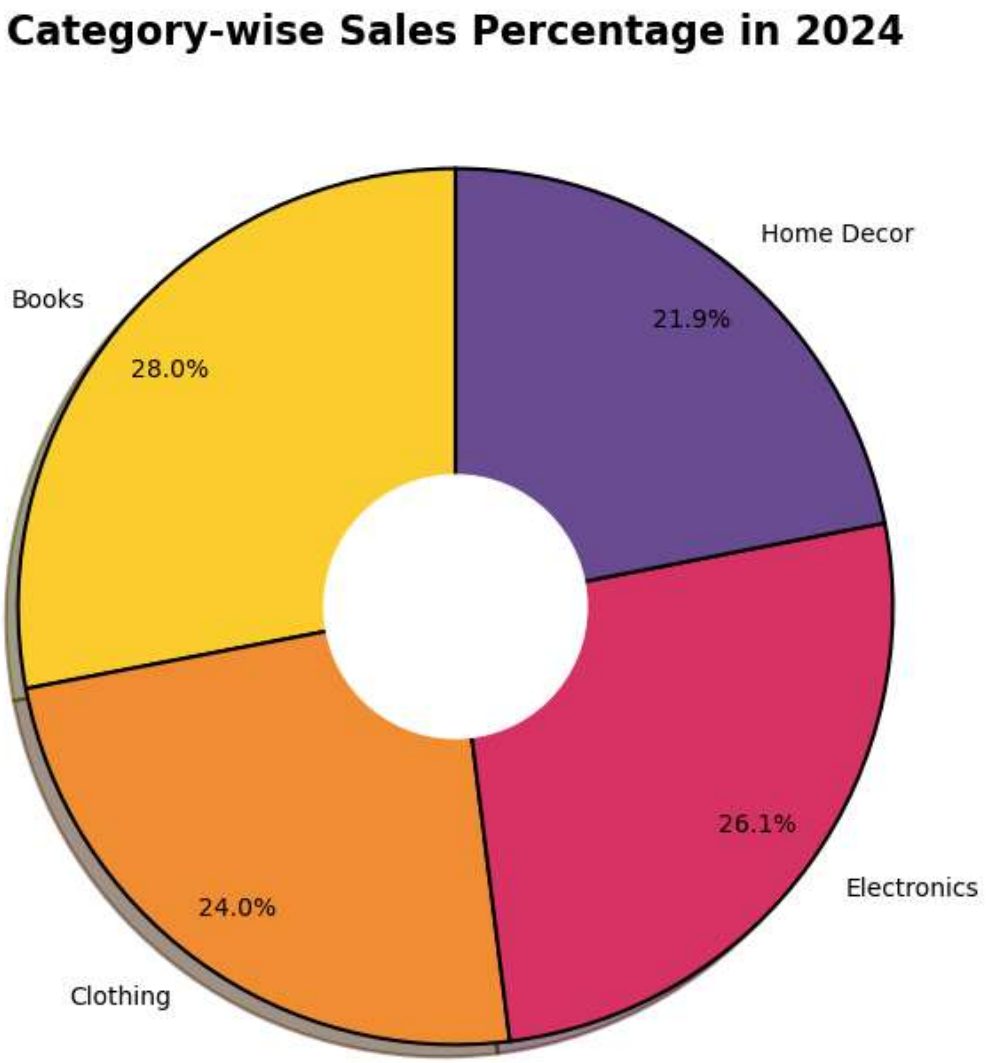
November Sales Drop:

November experienced a significant decline in sales, with total sales dipping below 20,000, indicating a notable drop in customer engagement.

Recovery in December:

By December, sales rebounded quickly, returning to near 60,000 levels, demonstrating a strong

In [53]:
```
# 2) Showing category-wise sales percentage in 2024
# Merge with the products dataframe to get the category information
merged_data = pd.merge(transactions_2024, products_df, on='ProductID', how
category_sales = merged_data.groupby('Category')['TotalValue'].sum()

# Donut plot: Category sales percentage with 3D-like effect
plt.figure(figsize=(8, 8))
category_sales.plot(kind='pie', autopct='%1.1f%%', figsize=(8, 8),
                    colors=['#FAD02E', '#F28D35', '#D83367', '#6A4C93'],
                    wedgeprops={'edgecolor': 'black', 'linewidth': 1.5, 'l
                    startangle=90, pctdistance=0.85, shadow=True)

# Create a hole in the center to make it a donut plot
plt.gca().add_artist(plt.Circle((0, 0), 0.30, color='white'))
plt.title("Category-wise Sales Percentage in 2024", fontsize=16, fontweigh
plt.ylabel("")

plt.show()
```



Balanced Sales Distribution:

Sales are evenly distributed across categories with no significant variations in percentage contributions.

Equal Category Contribution:

Each category contributes a similar share to total sales, indicating a broad interest in all product types.

Minimal Differences:

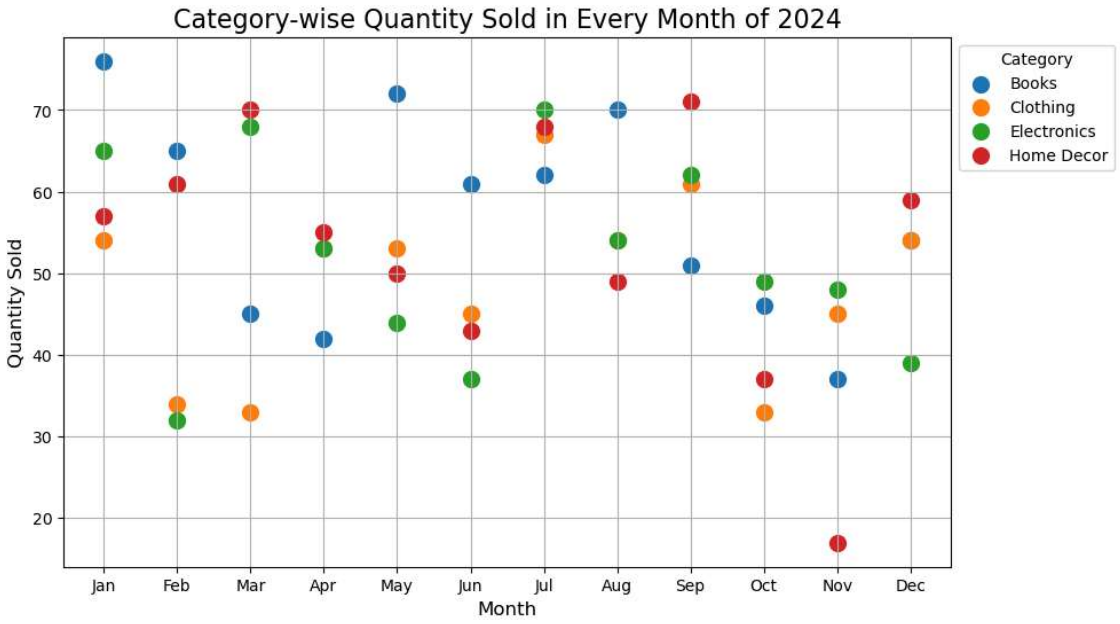
There is no dominant category; all product categories are performing almost equally.

Opportunity for Targeted Marketing:

There is potential to increase sales by focusing marketing efforts on categories with slightly lower performance.

In [55]:

```
# 3) Plot category-wise quantity sold in every month of 2024
monthly_quantity = merged_data.groupby([merged_data['TransactionDate'].dt.

# Convert month number to month name (Jan, Feb, Mar, etc.)
monthly_quantity.index = monthly_quantity.index.to_series().apply(lambda x

# Plot: Category-wise quantity sold in every month of 2024 (Scatter plot)
plt.figure(figsize=(10, 6))

# Scatter plot for each category
for category in monthly_quantity.columns:
    plt.scatter(monthly_quantity.index, monthly_quantity[category], label=

# Title and labels
plt.title("Category-wise Quantity Sold in Every Month of 2024", fontsize=1
plt.xlabel("Month", fontsize=12)
plt.ylabel("Quantity Sold", fontsize=12)
plt.grid(True)
plt.legend(title="Category", loc='upper left', bbox_to_anchor=(1, 1))

# Show the plot
plt.show()
```



High Sales in Jan, Jul, and Sep:

In these three months, all categories sold more than 50 units, indicating higher demand or successful marketing during these periods.

Top-Selling Categories:

Books and Home Decor are the most sold categories throughout the year, contributing significantly to total sales.

Here are the 5 key business insights summarized:

1. **High Sales in Jan, Jul, and Sep**: Strong sales in these months (over 50 units sold across categories) suggest successful marketing or high demand.
    - **Action**: Focus on marketing during these months.
2. **Top-Selling Categories: Books and Home Decor**: These categories performed the best across the year.
    - **Action**: Invest in more inventory and promotions for these categories.
3. **Sales Fluctuations by Month**: High sales in Jan, May, Jul, Aug, and Sep, with a drop in Nov.
    - **Action**: Analyze and address low sales months, especially November.
4. **Balanced Category Sales**: Sales are evenly distributed across categories.
    - **Action**: Continue promoting all categories to maintain balanced sales.
5. **Customer Signups Trend**: A significant increase in signups in 2024, after a drop in 2023.

- **Action**: Analyze the factors behind the increase to maintain growth.

In [ ]: