```
from google.colab import files

# upload the file
uploaded = files.upload()
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving Customer Data.csv to Customer Data.csv
```

```
import pandas as pd

df = pd.read_csv("Customer Data.csv")

#view first 5 lines
df.head()
```

	Customer_ID	Name	Email	Phone_Number	Address	Country	City	Postal_Code	Date_of_Birth	Sig
0	260bdf8c-00d3-4f67-b409-2c5cee3d64cb	Nicole Ramos	lopezmorgan@cowan.com	941-554-3132	660 Gray Rapid, Annechester, HI 34762	Pakistan	Nelsonberg	72381	1/06/1983	
1	9519979b-1d76-4724-b0ef-444c370f0236	Kayla Miranda	bcasey@stewart.info	001-112-289-1767x88007	539 Hill Cliffs, Port Maryville, SC 51806	French Polynesia	South Charles	2039	5/02/2005	2
2	7d58930f-326f-47e8-9fd7-be87631e41d6	Paula Goodman	ortizcassandra@becker-lewis.net	(977)253-6780	9094 Amanda Pass, West Crystalstad, MN 40344	Puerto Rico	New Kayla	5141	25/04/1981	1
3	7e4897d4-ee3a-4462-92e5-9deb6d505ae2	Brandon Fisher	qbolton@yahoo.com	(395)637-7172x12097	Unit 5283 Box 8114, DPO AE 02718	Mauritius	South Josephhaven	91393	17/04/2006	
4	a1c82dfb-f3d5-4eae-bb34-b4d50a7b6c73	Christopher Hopkins	dianebanks@hotmail.com	053-523-9452x05828	6872 Mcdonald Corners Apt. 417, Kleinton, VA 9...	Cote d'Ivoire	North Jonathan	32299	17/03/1968	1

```
# Check data types
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 29965 entries, 0 to 29964
Data columns (total 17 columns):
 #   Column          Non-Null Count  Dtype 
---  ------          --------------  ----- 
 0   Customer_ID     10000 non-null  object
 1   Name            29965 non-null  object
 2   Email           29965 non-null  object
 3   Phone_Number    29965 non-null  object
 4   Address         29965 non-null  object
 5   Country         29965 non-null  object
 6   City            29965 non-null  object
 7   Postal_Code     29965 non-null  int64 
 8   Date_of_Birth   29965 non-null  object
 9   Signup_Date     29965 non-null  object
 10  Gender          29965 non-null  object
 11  Order_Date      29965 non-null  object
 12  Product_Name    29965 non-null  object
 13  Quantity        29965 non-null  int64 
 14  Unit_Price      29965 non-null  int64 
 15  Total_Amount    29965 non-null  int64 
 16  Payment_Method  29965 non-null  object
dtypes: int64(4), object(13)
memory usage: 3.9+ MB
```

```
# Converting date columns to datetime format
df["Date_of_Birth"] = pd.to_datetime(df["Date_of_Birth"], format="%d/%m/%Y", errors="coerce")
df["Signup_Date"] = pd.to_datetime(df["Signup_Date"], format="%d/%m/%Y", errors="coerce")
df["Order_Date"] = pd.to_datetime(df["Order_Date"], format="%d/%m/%Y", errors="coerce")

# For check
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 29965 entries, 0 to 29964
Data columns (total 17 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Customer_ID     10000 non-null  object
 1   Name            29965 non-null  object
 2   Email           29965 non-null  object
 3   Phone_Number    29965 non-null  object
 4   Address         29965 non-null  object
 5   Country         29965 non-null  object
 6   City            29965 non-null  object
 7   Postal_Code     29965 non-null  int64
 8   Date_of_Birth   29965 non-null  datetime64[ns]
 9   Signup_Date     29965 non-null  datetime64[ns]
 10  Gender          29965 non-null  object
 11  Order_Date      29965 non-null  datetime64[ns]
 12  Product_Name    29965 non-null  object
 13  Quantity        29965 non-null  int64
 14  Unit_Price      29965 non-null  int64
 15  Total_Amount    29965 non-null  int64
 16  Payment_Method  29965 non-null  object
dtypes: datetime64[ns](3), int64(4), object(10)
memory usage: 3.9+ MB
```

Start coding or generate with AI.

```
# Check number of missing data
df.isnull().sum()
```

	0
Customer_ID	19965
Name	0
Email	0
Phone_Number	0
Address	0
Country	0
City	0
Postal_Code	0
Date_of_Birth	0
Signup_Date	0
Gender	0
Order_Date	0
Product_Name	0
Quantity	0
Unit_Price	0
Total_Amount	0
Payment_Method	0

```
df[df["Customer_ID"].isnull()].head(20)
```

	Customer_ID	Name	Email	Phone_Number	Address	Country	City	Postal_Code	Date_of_Birth	Signup_Date	Gender	Or
10000	NaN	Unknown	Unknown	Unknown	Unknown	Unknown	Unknown	0	2025-02-09	2025-02-09	Other	2
10001	NaN	Unknown	Unknown	Unknown	Unknown	Unknown	Unknown	0	2025-02-09	2025-02-09	Other	2
10002	NaN	Unknown	Unknown	Unknown	Unknown	Unknown	Unknown	0	2025-02-09	2025-02-09	Other	2
10003	NaN	Unknown	Unknown	Unknown	Unknown	Unknown	Unknown	0	2025-02-09	2025-02-09	Other	2
10004	NaN	Unknown	Unknown	Unknown	Unknown	Unknown	Unknown	0	2025-02-09	2025-02-09	Other	2
10005	NaN	Unknown	Unknown	Unknown	Unknown	Unknown	Unknown	0	2025-02-09	2025-02-09	Other	2
10006	NaN	Unknown	Unknown	Unknown	Unknown	Unknown	Unknown	0	2025-02-09	2025-02-09	Other	2
10007	NaN	Unknown	Unknown	Unknown	Unknown	Unknown	Unknown	0	2025-02-09	2025-02-09	Other	2
10008	NaN	Unknown	Unknown	Unknown	Unknown	Unknown	Unknown	0	2025-02-09	2025-02-09	Other	2
10009	NaN	Unknown	Unknown	Unknown	Unknown	Unknown	Unknown	0	2025-02-09	2025-02-09	Other	2
10010	NaN	Unknown	Unknown	Unknown	Unknown	Unknown	Unknown	0	2025-02-09	2025-02-09	Other	2
10011	NaN	Unknown	Unknown	Unknown	Unknown	Unknown	Unknown	0	2025-02-09	2025-02-09	Other	2
10012	NaN	Unknown	Unknown	Unknown	Unknown	Unknown	Unknown	0	2025-02-09	2025-02-09	Other	2
10013	NaN	Unknown	Unknown	Unknown	Unknown	Unknown	Unknown	0	2025-02-09	2025-02-09	Other	2
10014	NaN	Unknown	Unknown	Unknown	Unknown	Unknown	Unknown	0	2025-02-09	2025-02-09	Other	2
10015	NaN	Unknown	Unknown	Unknown	Unknown	Unknown	Unknown	0	2025-02-09	2025-02-09	Other	2
10016	NaN	Unknown	Unknown	Unknown	Unknown	Unknown	Unknown	0	2025-02-09	2025-02-09	Other	2
10017	NaN	Unknown	Unknown	Unknown	Unknown	Unknown	Unknown	0	2025-02-09	2025-02-09	Other	2
10018	NaN	Unknown	Unknown	Unknown	Unknown	Unknown	Unknown	0	2025-02-09	2025-02-09	Other	2
10019	NaN	Unknown	Unknown	Unknown	Unknown	Unknown	Unknown	0	2025-02-09	2025-02-09	Other	2

```
# Delete rows with missing Customer_ID
df = df.dropna(subset=["Customer_ID"])
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 10000 entries, 0 to 9999
Data columns (total 17 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Customer_ID    10000 non-null  object
 1   Name           10000 non-null  object
 2   Email          10000 non-null  object
 3   Phone_Number   10000 non-null  object
 4   Address        10000 non-null  object
 5   Country        10000 non-null  object
 6   City           10000 non-null  object
 7   Postal_Code    10000 non-null  int64
 8   Date_of_Birth  10000 non-null  datetime64[ns]
 9   Signup_Date    10000 non-null  datetime64[ns]
 10  Gender         10000 non-null  object
 11  Order_Date     10000 non-null  datetime64[ns]
 12  Product_Name   10000 non-null  object
 13  Quantity       10000 non-null  int64
 14  Unit_Price     10000 non-null  int64
 15  Total_Amount   10000 non-null  int64
 16  Payment_Method 10000 non-null  object
dtypes: datetime64[ns](3), int64(4), object(10)
memory usage: 1.4+ MB
```

```
# Save the cleaned dataset as a new CSV file
df.to_csv("cleaned_customer_data.csv", index=False)
```

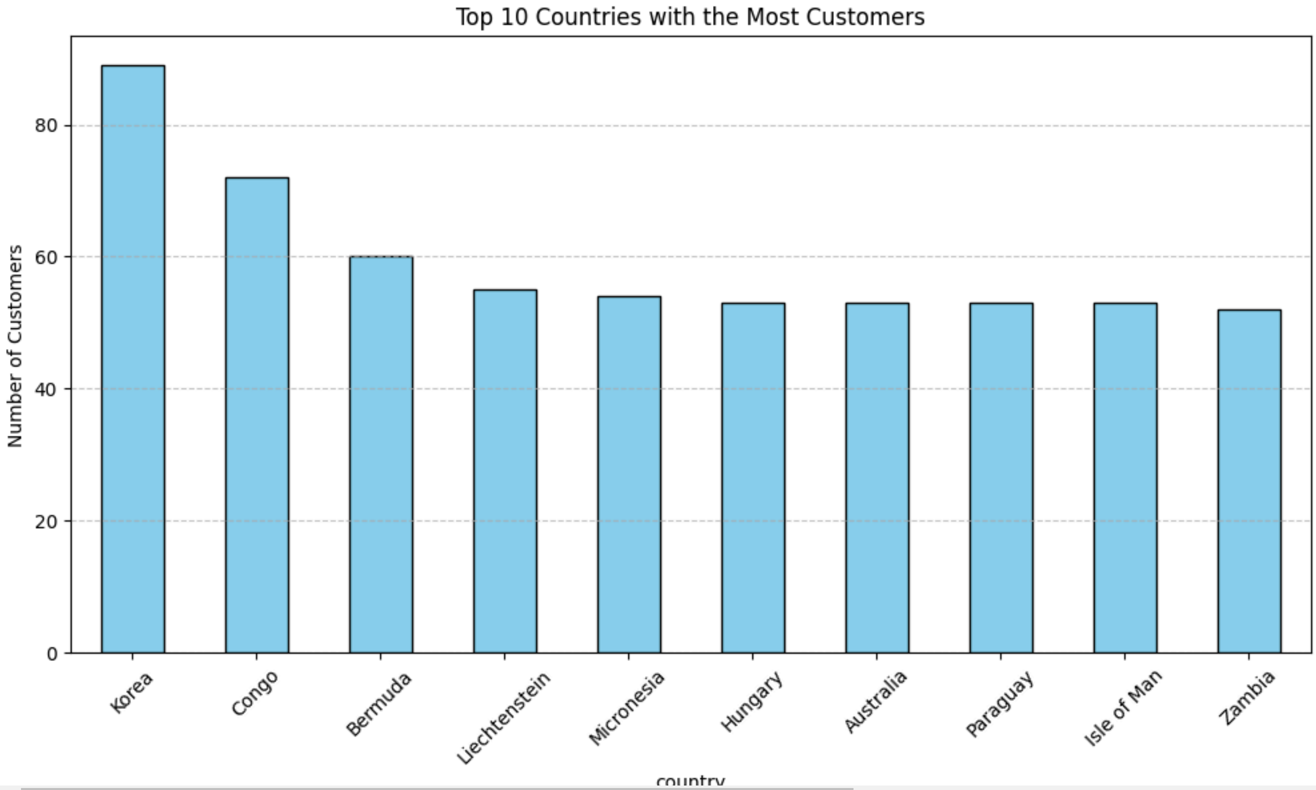
```
import os
os.listdir()
```

```
['.config', 'Customer Data.csv', 'cleaned_customer_data.csv', 'sample_data']
```

```
from google.colab import files
# Download the cleaned CSV file
```

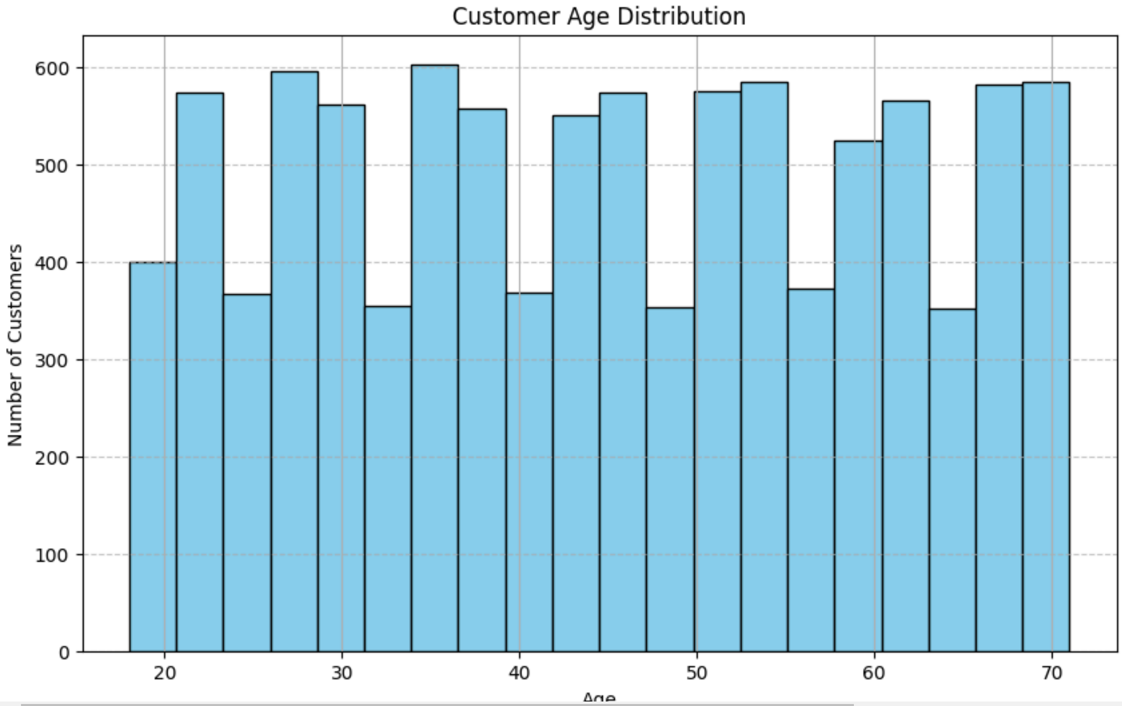
```
files.download("cleaned_customer_data.csv")
```

```
import matplotlib.pyplot as plt

# By number of customers by country
country_counts = df["Country"].value_counts().head(10)  # İlk 10 ülke

# visualize
plt.figure(figsize=(12,6))
country_counts.plot(kind="bar", color="skyblue", edgecolor="black")
plt.title("Top 10 Countries with the Most Customers")
plt.xlabel("country")
plt.ylabel("Number of Customers")
plt.xticks(rotation=45)
plt.grid(axis="y", linestyle="--", alpha=0.7)
plt.show()
```



In customer analysis, marketing efforts may be more effective in countries such as Korea and Congo.
There may be more sales in these countries because the customer base is larger.

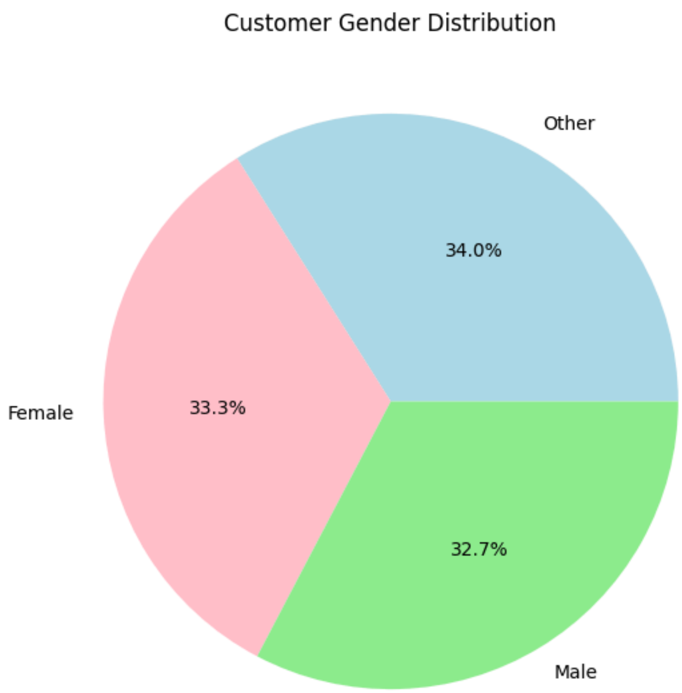
```
from datetime import datetime

# Get today's date
today = datetime.today()

#Calculate age (Today - Date of Birth)
df["Age"] = today.year - df["Date_of_Birth"].dt.year

#Visualizing age distribution
plt.figure(figsize=(10,6))
df["Age"].hist(bins=20, color="skyblue", edgecolor="black")
plt.title("Customer Age Distribution")
plt.xlabel("Age")
plt.ylabel("Number of Customers")
plt.grid(axis="y", linestyle="--", alpha=0.7)
plt.show()
```



The largest number of customers are between the ages of 25-30 and 55-60! There are almost no customers under the age of 18. The age distribution is generally balanced, but there are clear peaks in some age groups. What does this mean?

If a marketing strategy is to be created, the 25-30 age range can be focused on. Customers in the middle age group (55-60 years old) may also be an important target. The young age group (18-20 years old) customer base is low, perhaps special campaigns can be organized for this audience.

```
import matplotlib.pyplot as plt

# Count the number of customers by gender
gender_counts = df["Gender"].value_counts()

# Create a pie chart
plt.figure(figsize=(7,7))
plt.pie(gender_counts, labels=gender_counts.index, autopct="%1.1f%%", colors=["lightblue", "pink", "lightgreen"])
plt.title("Customer Gender Distribution")
plt.show()
```

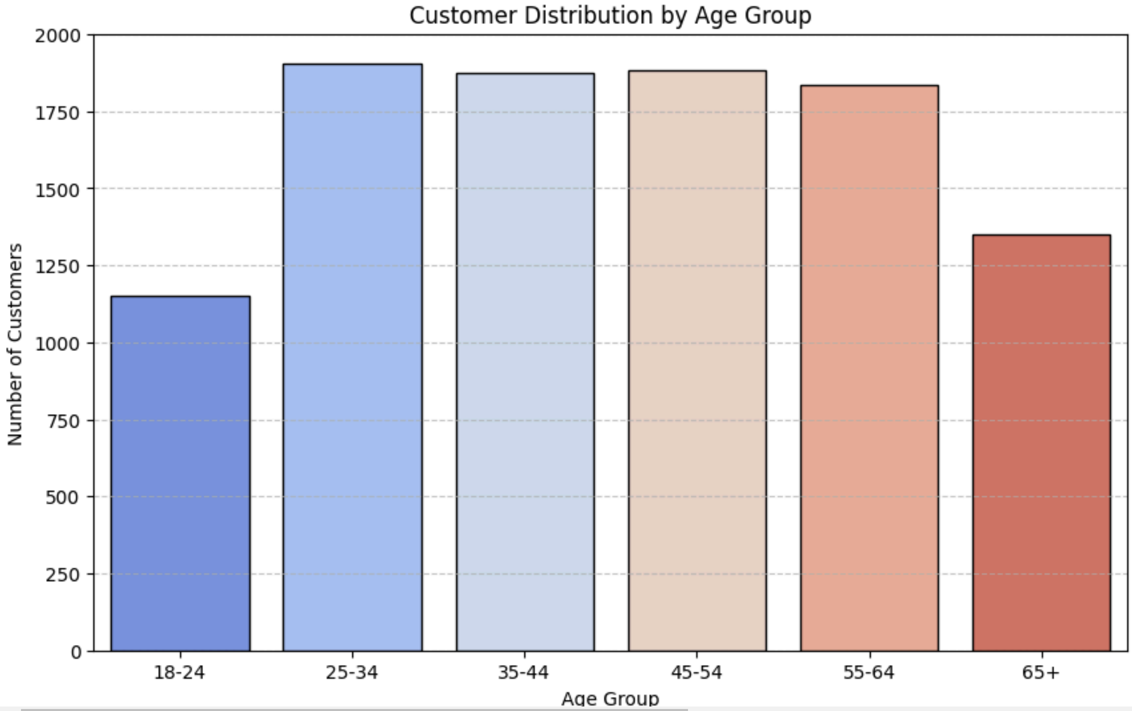


"Other" category has the highest percentage (34.0%) Female customers make up 33.3% Male customers account for 32.7% What does this mean?

The gender distribution is fairly balanced, with no single group dominating the customer base. Marketing campaigns should be designed inclusively since all gender categories have a significant presence. If needed, further investigation could reveal whether certain product categories are more popular among specific gender groups.

```
import seaborn as sns

# Define age groups
age_bins = [18, 25, 35, 45, 55, 65, 75]
age_labels = ["18-24", "25-34", "35-44", "45-54", "55-64", "65+"]
df["Age_Group"] = pd.cut(df["Age"], bins=age_bins, labels=age_labels, right=False)

# Count customers in each age group
age_group_counts = df["Age_Group"].value_counts().sort_index()

# Visualize the age group distribution
plt.figure(figsize=(10,6))
sns.barplot(x=age_group_counts.index, y=age_group_counts.values, palette="coolwarm", edgecolor="black")
plt.title("Customer Distribution by Age Group")
plt.xlabel("Age Group")
plt.ylabel("Number of Customers")
plt.grid(axis="y", linestyle="--", alpha=0.7)
plt.show()
```

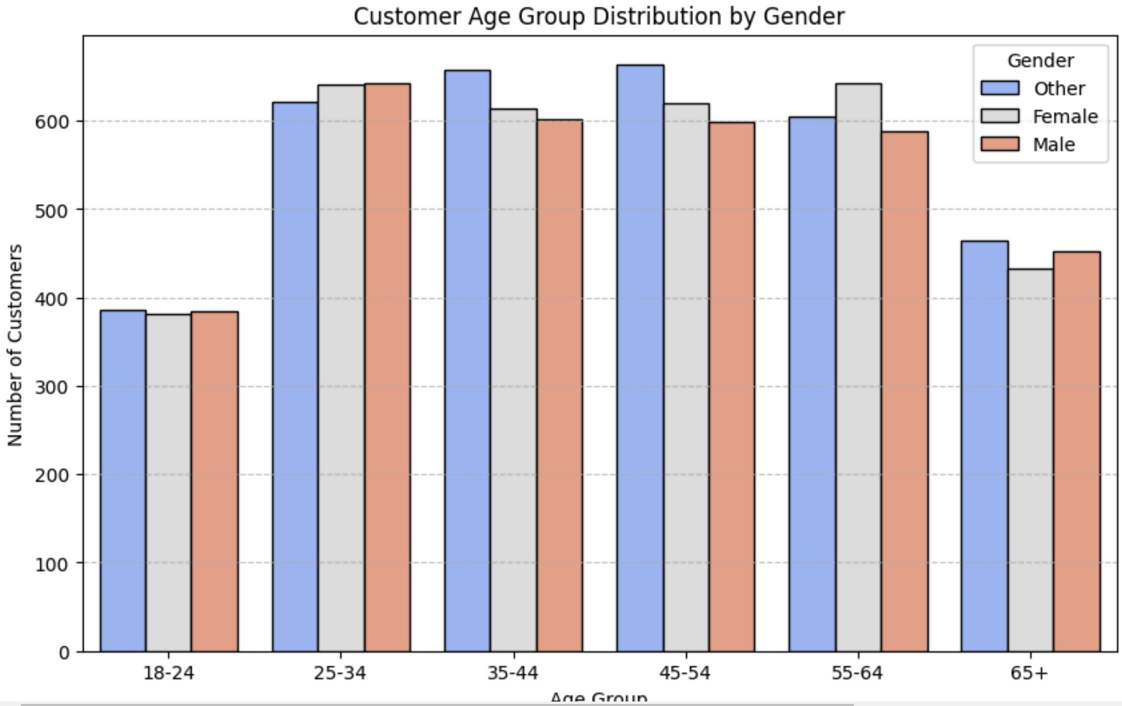
```
<ipython-input-14-0bc05037ae35>:13: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `le

  sns.barplot(x=age_group_counts.index, y=age_group_counts.values, palette="coolwarm", edgecolor="black")
```



Age Group Insights The 25-34 age group has the highest number of customers. 35-44, 45-54, and 55-64 age groups have a similar number of customers. The 18-24 and 65+ age groups have fewer customers compared to others.

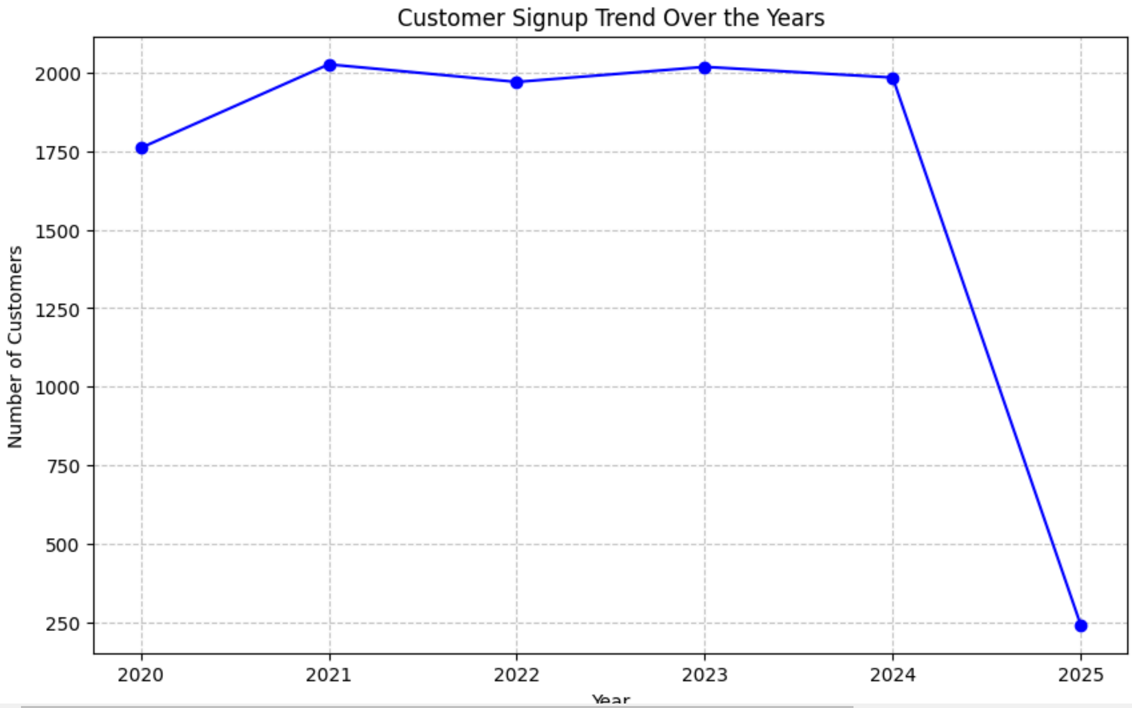
The 25-34 age group is the most active customer segment. This group should be a focus for marketing efforts. The 65+ and 18-24 groups are underrepresented. If the company wants to expand its reach, it may need different marketing strategies to attract these groups.

```
plt.figure(figsize=(10,6))
sns.countplot(x="Age_Group", hue="Gender", data=df, palette="coolwarm", edgecolor="black")
plt.title("Customer Age Group Distribution by Gender")
plt.xlabel("Age Group")
plt.ylabel("Number of Customers")
plt.legend(title="Gender")
plt.grid(axis="y", linestyle="--", alpha=0.7)
plt.show()
```



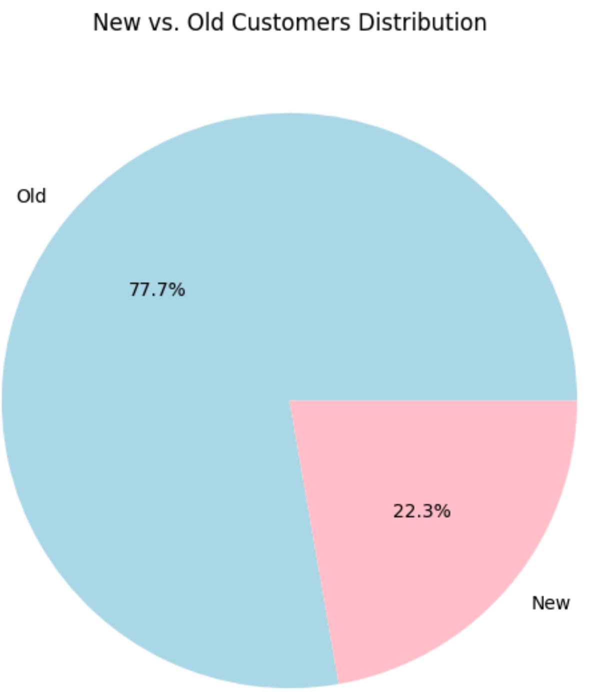
The 25-34 and 45-54 age groups are the most active customer base, so these segments can be targeted primarily. Special promotions or more accessible services may be offered to increase the 65+ age group.

```
import matplotlib.pyplot as plt

# Extract the year from the signup date
df["Signup_Year"] = df["Signup_Date"].dt.year

# Count the number of customers per year
signup_counts = df["Signup_Year"].value_counts().sort_index()

# Plot the customer signup trend
plt.figure(figsize=(10,6))
plt.plot(signup_counts.index, signup_counts.values, marker="o", linestyle="-", color="blue")
plt.title("Customer Signup Trend Over the Years")
plt.xlabel("Year")
plt.ylabel("Number of Customers")
plt.grid(True, linestyle="--", alpha=0.7)
plt.show()
```



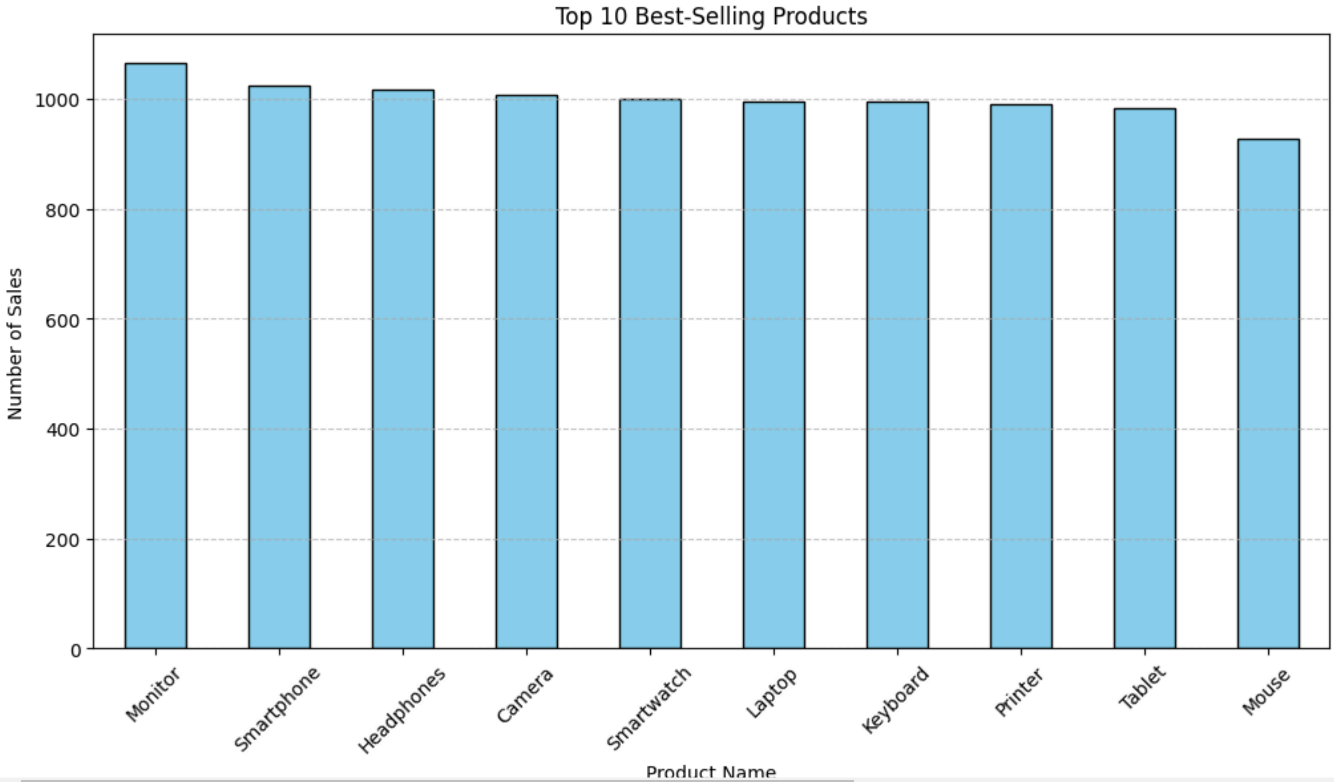
Customer sign-ups in 2025 have dropped significantly compared to previous years. If the 2025 data is incomplete, this drop might be due to missing records. The company may need to review its customer acquisition strategies.

```
import numpy as np

# Define new and old customers based on signup year
df["Customer_Type"] = np.where(df["Signup_Year"] >= 2024, "New", "Old")

# Count new and old customers
customer_type_counts = df["Customer_Type"].value_counts()

# Visualize the distribution
plt.figure(figsize=(7,7))
plt.pie(customer_type_counts, labels=customer_type_counts.index, autopct="%1.1f%%", colors=["lightblue", "pink"])
plt.title("New vs. Old Customers Distribution")
plt.show()
```



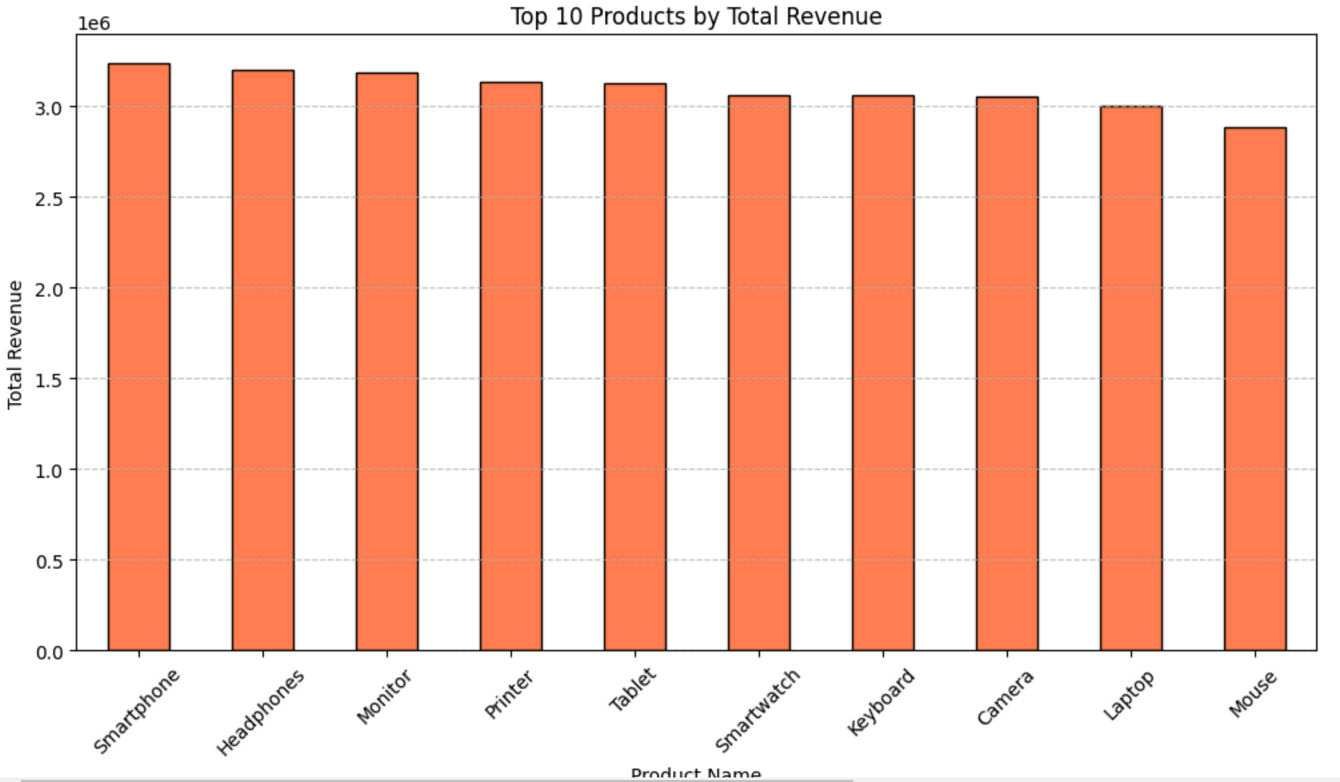
The customer base relies heavily on old customers. New customer acquisition is at a low level. The company may need more aggressive marketing strategies to attract new customers.

```
import matplotlib.pyplot as plt

# Count the number of sales per product
top_products = df["Product_Name"].value_counts().head(10)  # Top 10 products

# Visualize the data
plt.figure(figsize=(12,6))
top_products.plot(kind="bar", color="skyblue", edgecolor="black")
plt.title("Top 10 Best-Selling Products")
plt.xlabel("Product Name")
plt.ylabel("Number of Sales")
plt.xticks(rotation=45)
plt.grid(axis="y", linestyle="--", alpha=0.7)
plt.show()
```



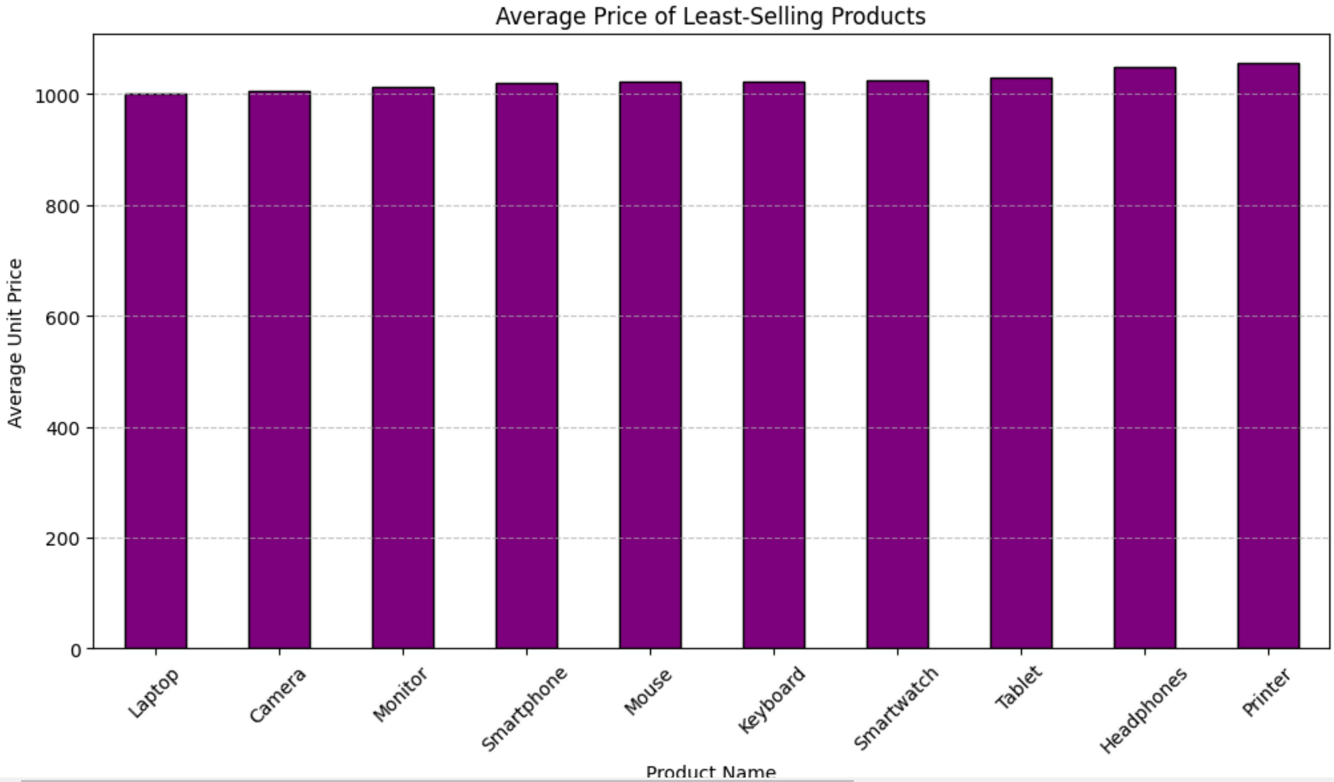
Monitor is the best-selling product. Smartphone, Headphones, Camera, and Smartwatch have high sales. Sales volume is very close among the top 10 products. Stock management is crucial for monitors and other top-selling products. The reasons for low sales of other products should be analyzed.

```
# Calculate total revenue per product
product_revenue = df.groupby("Product_Name")["Total_Amount"].sum().sort_values(ascending=False).head(10)

# Visualize revenue distribution
plt.figure(figsize=(12,6))
product_revenue.plot(kind="bar", color="coral", edgecolor="black")
plt.title("Top 10 Products by Total Revenue")
plt.xlabel("Product Name")
plt.ylabel("Total Revenue")
plt.xticks(rotation=45)
plt.grid(axis="y", linestyle="--", alpha=0.7)
plt.show()
```



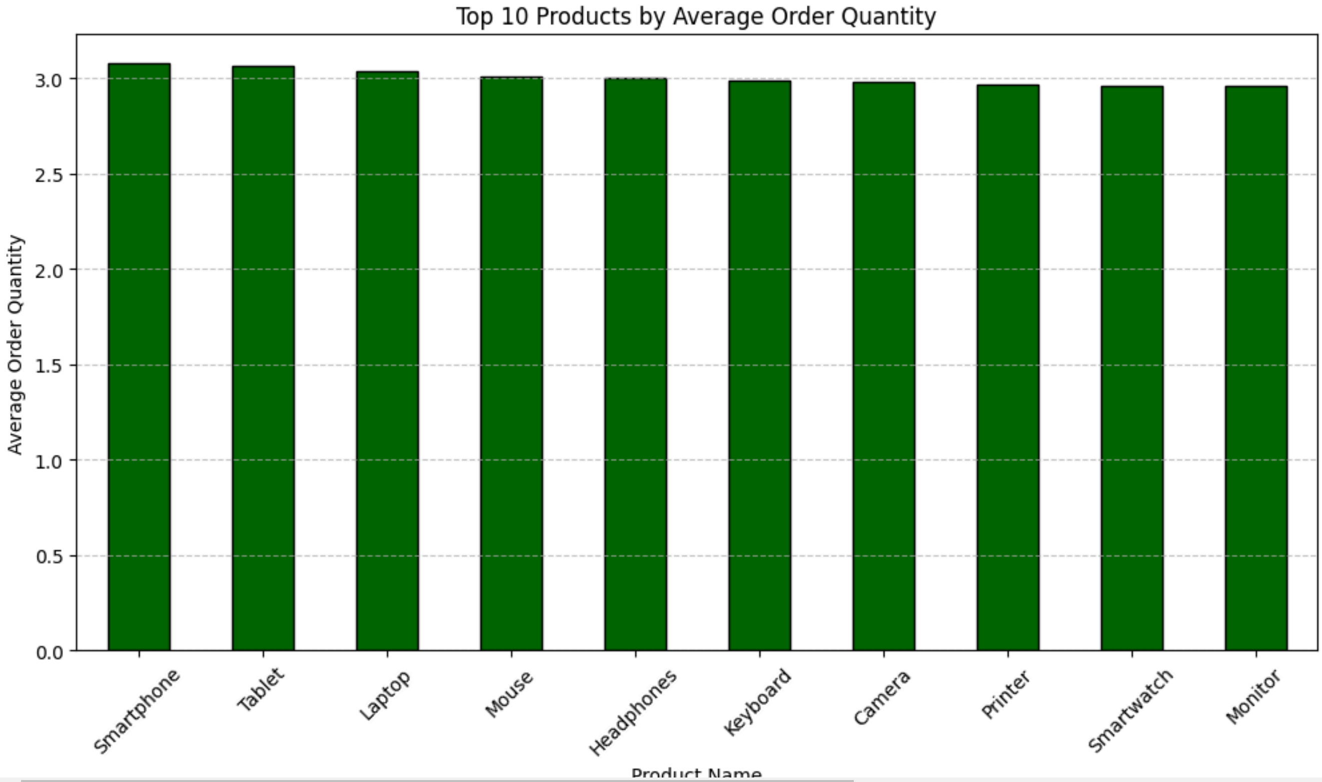
Smartphone is the highest revenue-generating product. Headphones, Monitor, and Printer are also among the top revenue-generating products. Revenue is evenly distributed among the top 10 products. Stock management and promotions are crucial for these products.

```
# Find the average price of least-selling products
least_selling_avg_price = df.groupby("Product_Name")["Unit_Price"].mean().sort_values().head(10)

# Visualize the price distribution
plt.figure(figsize=(12,6))
least_selling_avg_price.plot(kind="bar", color="purple", edgecolor="black")
plt.title("Average Price of Least-Selling Products")
plt.xlabel("Product Name")
plt.ylabel("Average Unit Price")
plt.xticks(rotation=45)
plt.grid(axis="y", linestyle="--", alpha=0.7)
plt.show()
```



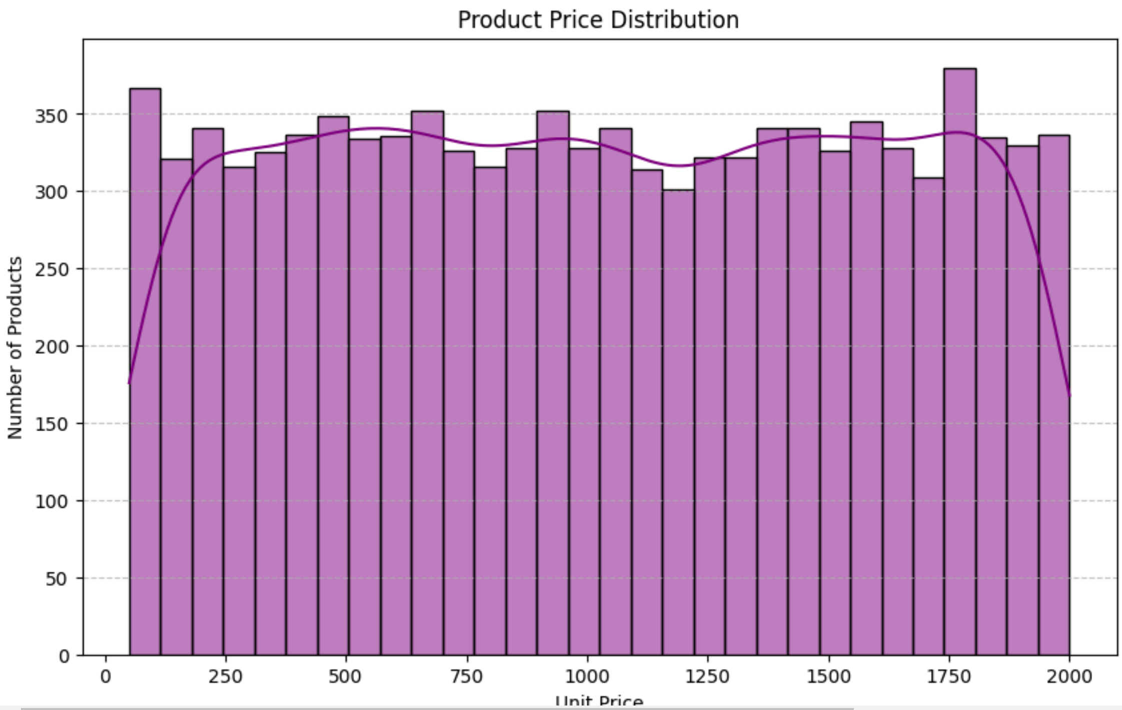
Low-selling products are generally high-priced. More affordable alternatives could be offered for price-sensitive customers. Installment payment options could improve accessibility.

```
# Calculate the average order quantity per product
avg_order_quantity = df.groupby("Product_Name")["Quantity"].mean().sort_values(ascending=False)

# Visualize the data
plt.figure(figsize=(12,6))
avg_order_quantity.head(10).plot(kind="bar", color="darkgreen", edgecolor="black")
plt.title("Top 10 Products by Average Order Quantity")
plt.xlabel("Product Name")
plt.ylabel("Average Order Quantity")
plt.xticks(rotation=45)
plt.grid(axis="y", linestyle="--", alpha=0.7)
plt.show()
```



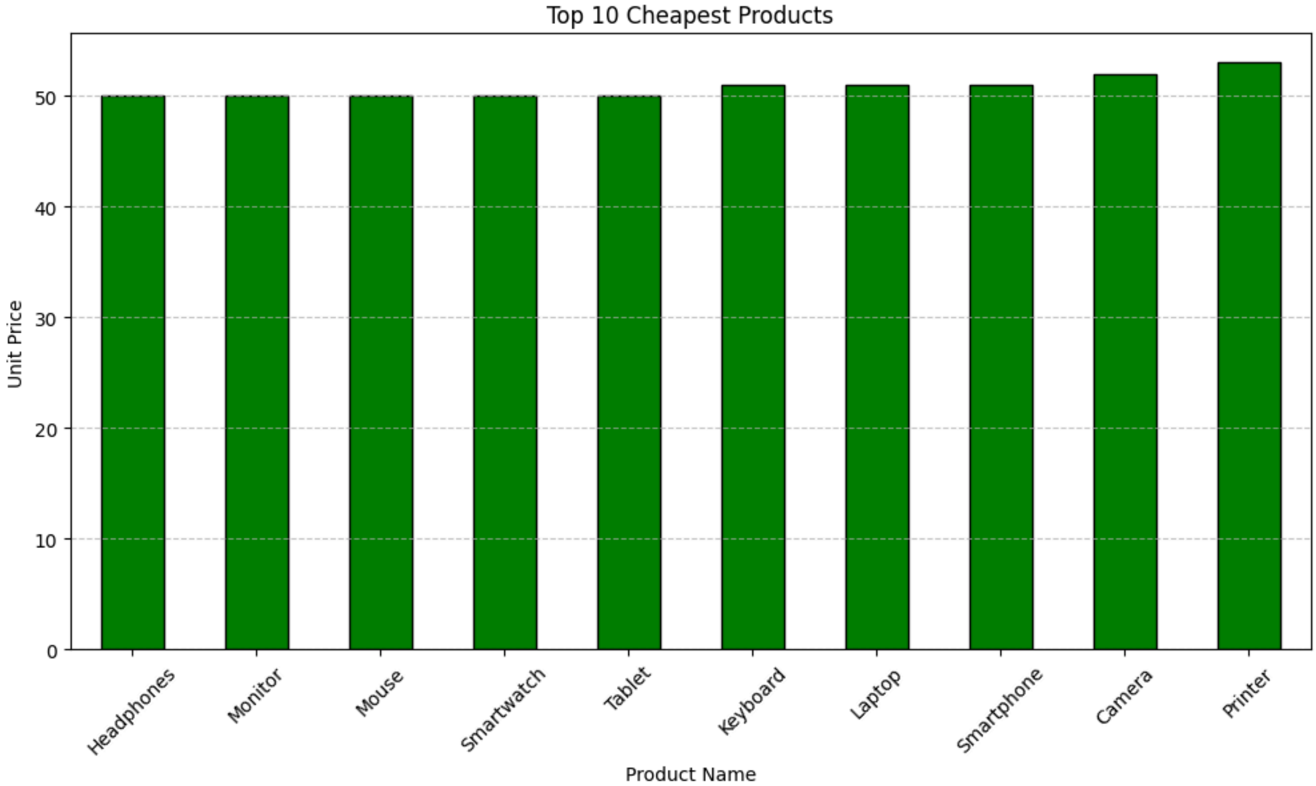
Smartphone, Tablet, and Laptop have the highest average order quantities. Mouse, Headphones, and Keyboard also have high order quantities. This suggests a bulk purchase trend for certain products.

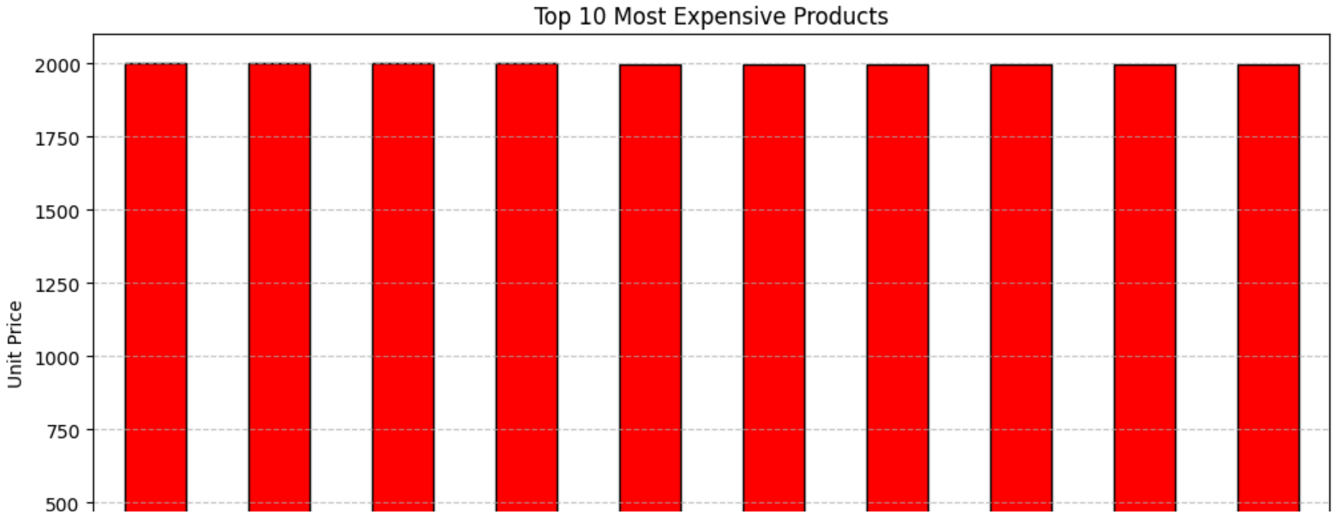
```
# Visualize the distribution of product prices
plt.figure(figsize=(10,6))
sns.histplot(df["Unit_Price"], bins=30, kde=True, color="purple", edgecolor="black")
plt.title("Product Price Distribution")
plt.xlabel("Unit Price")
plt.ylabel("Number of Products")
plt.grid(axis="y", linestyle="--", alpha=0.7)
plt.show()
```



Product prices are spread over a wide range. The price distribution seems relatively balanced.

```
# Identify the cheapest and most expensive products
cheapest_products = df.groupby("Product_Name")["Unit_Price"].min().sort_values().head(10)
most_expensive_products = df.groupby("Product_Name")["Unit_Price"].max().sort_values(ascending=False).head(10)

# Visualize the cheapest products
plt.figure(figsize=(12,6))
cheapest_products.plot(kind="bar", color="green", edgecolor="black")
plt.title("Top 10 Cheapest Products")
plt.xlabel("Product Name")
plt.ylabel("Unit Price")
plt.xticks(rotation=45)
plt.grid(axis="y", linestyle="--", alpha=0.7)
plt.show()

# Visualize the most expensive products
plt.figure(figsize=(12,6))
most_expensive_products.plot(kind="bar", color="red", edgecolor="black")
plt.title("Top 10 Most Expensive Products")
plt.xlabel("Product Name")
plt.ylabel("Unit Price")
plt.xticks(rotation=45)
plt.grid(axis="y", linestyle="--", alpha=0.7)
plt.show()
```

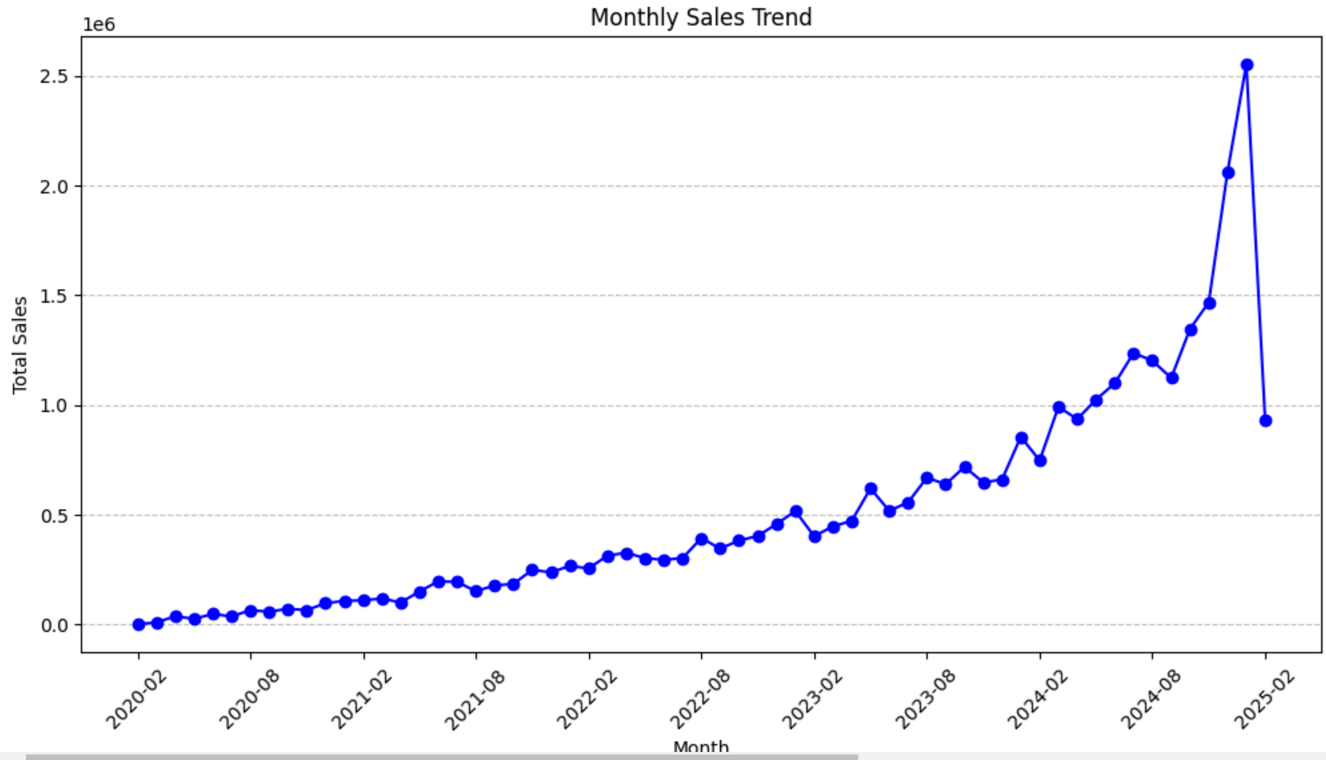




The cheapest products are usually low-cost and commonly used items. The most expensive products are technology-based (e.g., laptops, cameras, smartwatches). Cheap products have higher sales potential due to low prices, while expensive products provide higher profit margins.

```
#Convert Order_Date column to date format. Thus, let's find the total sales amount made each month.
df['Order_Date'] = pd.to_datetime(df['Order_Date'])
monthly_sales = df.groupby(df['Order_Date'].dt.to_period("M"))['Total_Amount'].sum()
monthly_sales.index = monthly_sales.index.astype(str)
```

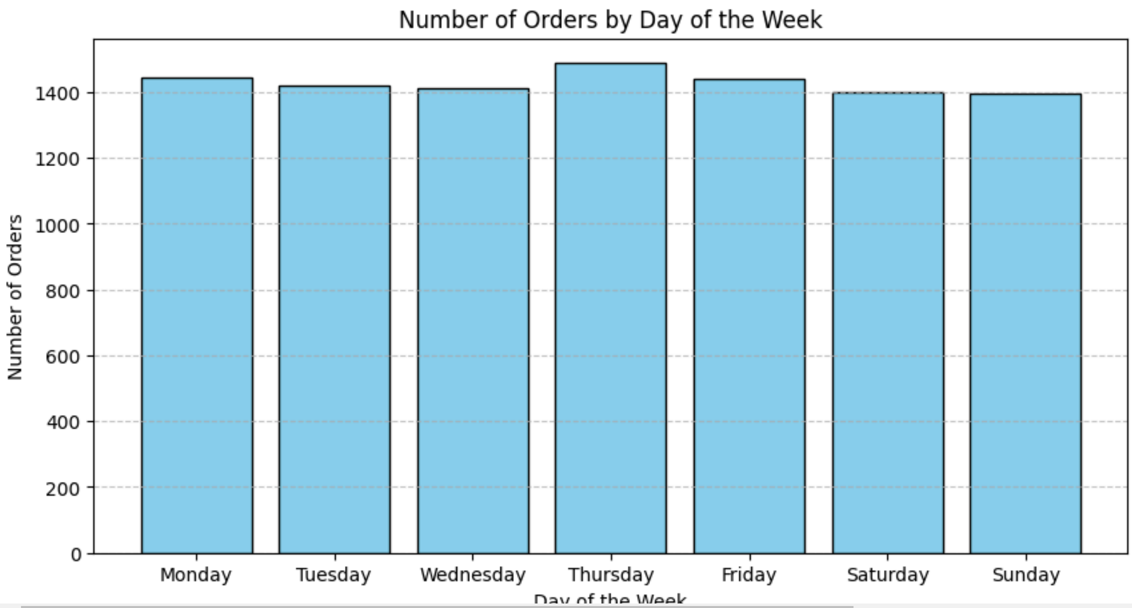
```
plt.figure(figsize=(12,6))  # Set figure size
plt.plot(monthly_sales.index, monthly_sales.values, marker='o', linestyle='-', color='b')  # Line chart

plt.xlabel("Month")  # X-axis label
plt.ylabel("Total Sales")  # Y-axis label
plt.title("Monthly Sales Trend")  # Chart title

plt.xticks(range(0, len(monthly_sales), 6), monthly_sales.index[::6], rotation=45)  # Reduce x-axis labels
plt.grid(axis="y", linestyle="--", alpha=0.7)  # Add grid lines on y-axis

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



Steady growth trend since 2020 Sharp increase from mid-2024 Sudden drop at the beginning of 2025 There might have been a discount period or a seasonal shopping boom at the end of 2024. The drop in 2025 could be due to missing or incorrect data. (New competitors or economic changes might have caused a sudden drop in sales.

```
import matplotlib.pyplot as plt
import pandas as pd

# Convert 'Order_Date' to datetime format
df["Order_Date"] = pd.to_datetime(df["Order_Date"])

# Extract day of the week and count the number of orders
df["Day_of_Week"] = df["Order_Date"].dt.day_name()
order_counts = df["Day_of_Week"].value_counts()

# Sort the days to follow the correct weekly order
order_counts = order_counts.reindex(
    ["Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday", "Sunday"]
)

# Plot the data
plt.figure(figsize=(10,5))
plt.bar(order_counts.index, order_counts.values, color="skyblue", edgecolor="black")
plt.xlabel("Day of the Week")
plt.ylabel("Number of Orders")
plt.title("Number of Orders by Day of the Week")
plt.grid(axis="y", linestyle="--", alpha=0.7)

# Show the plot
```

```
plt.show()
```



The most orders seem to be placed on Thursday. During business days, customers may be more inclined to place orders. If there is a drop in orders on weekends, it can be useful to analyze this situation and plan special campaigns on weekends.

```
import matplotlib.pyplot as plt

# Group orders by payment method
payment_counts = df["Payment_Method"].value_counts()

# Create the plot
plt.figure(figsize=(10, 5))
plt.bar(payment_counts.index, payment_counts.values, color='blue', alpha=0.7, edgecolor='black')
plt.xlabel("Payment Method")  #  X-axis label
plt.ylabel("Number of Orders")  #  Y-axis label
```