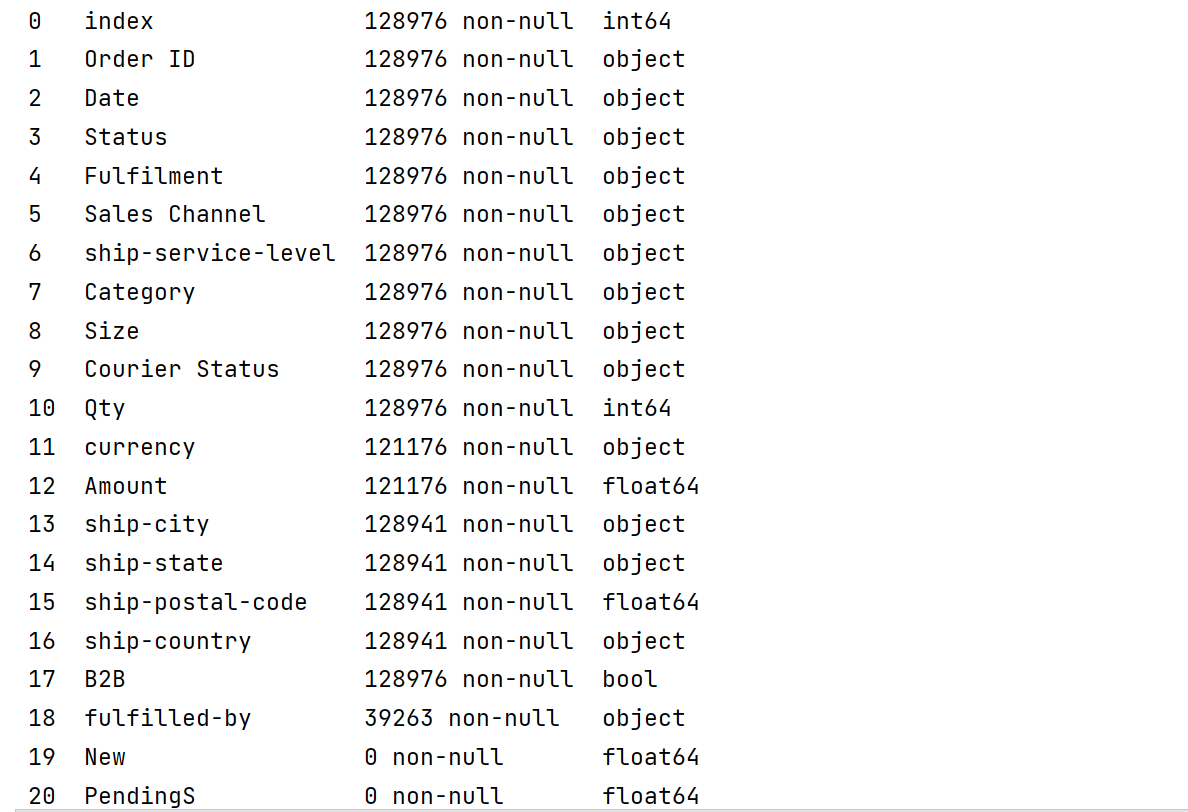
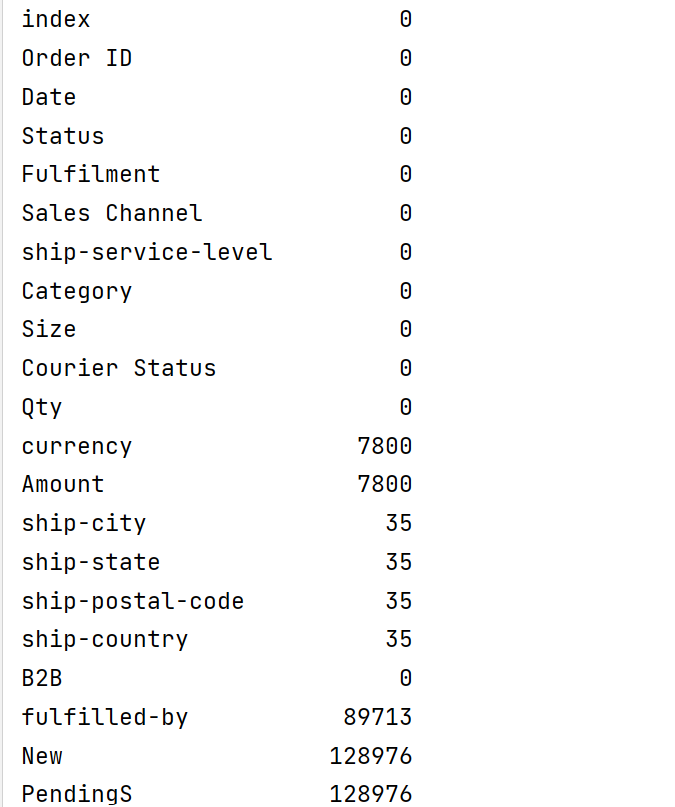
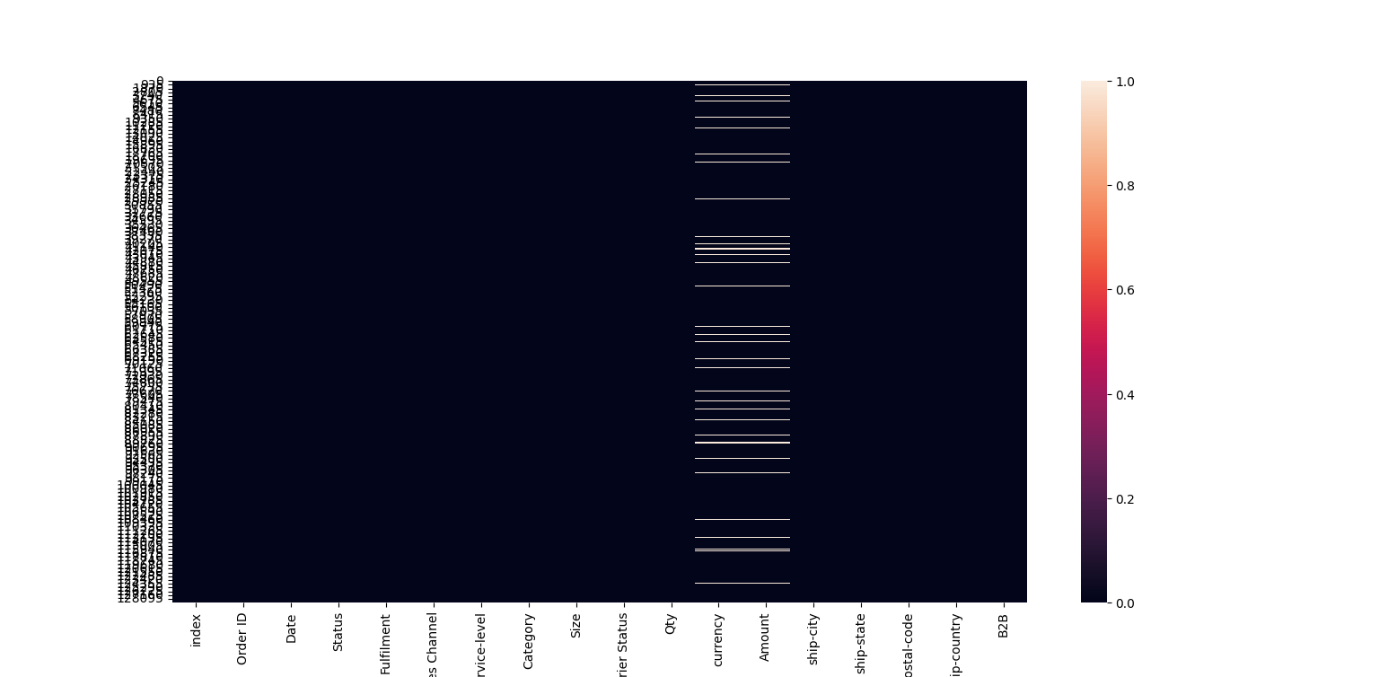
import pandas as pd  
import matplotlib.pyplot as plt  
import seaborn as sns  
df=pd.read\_excel("C:/Users/Lenovo/Downloads/Amazon Sale Report.xlsx")  
pd.set\_option('display.max\_rows', None)  
pd.set\_option('display.max\_columns', None)  
print(df.info())



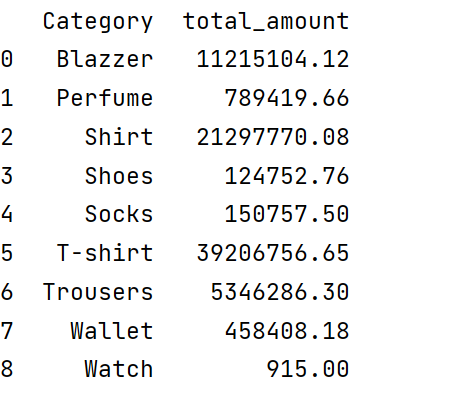
print(df.isnull().sum())

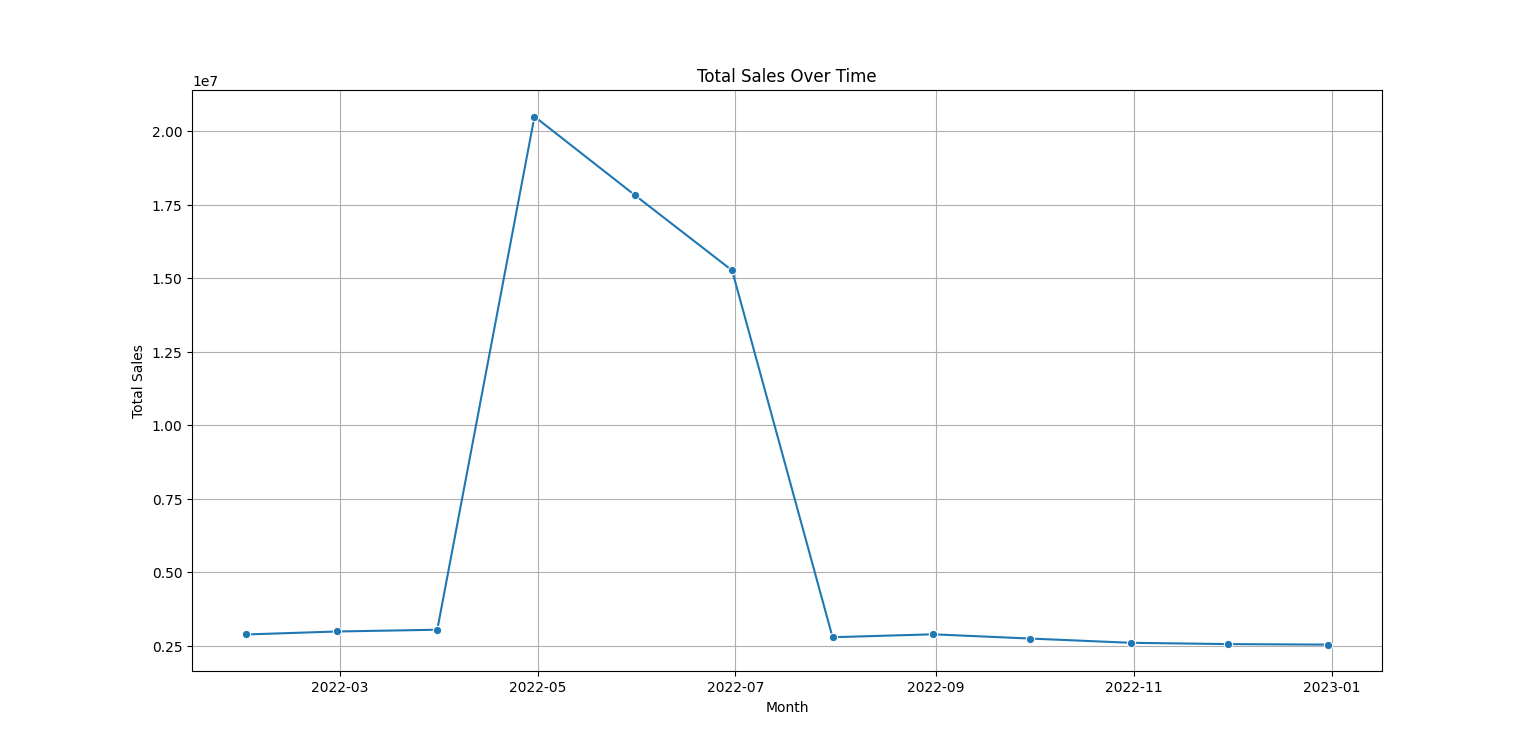


***#data cleaning***df = df.drop(columns=['New'])  
df = df.drop(columns=['PendingS'])  
print(df)  
percent=df.isnull().sum()/df.shape[0]\*100  
print(percent)  
drop\_columns = percent[percent >60].index  
print("\nColumns to be dropped:")  
print(drop\_columns)  
df.drop(columns=drop\_columns, inplace=True)  
print(df.info())  
plt.figure(figsize=(25,25))  
sns.heatmap(df.isnull())  
print(plt.show()

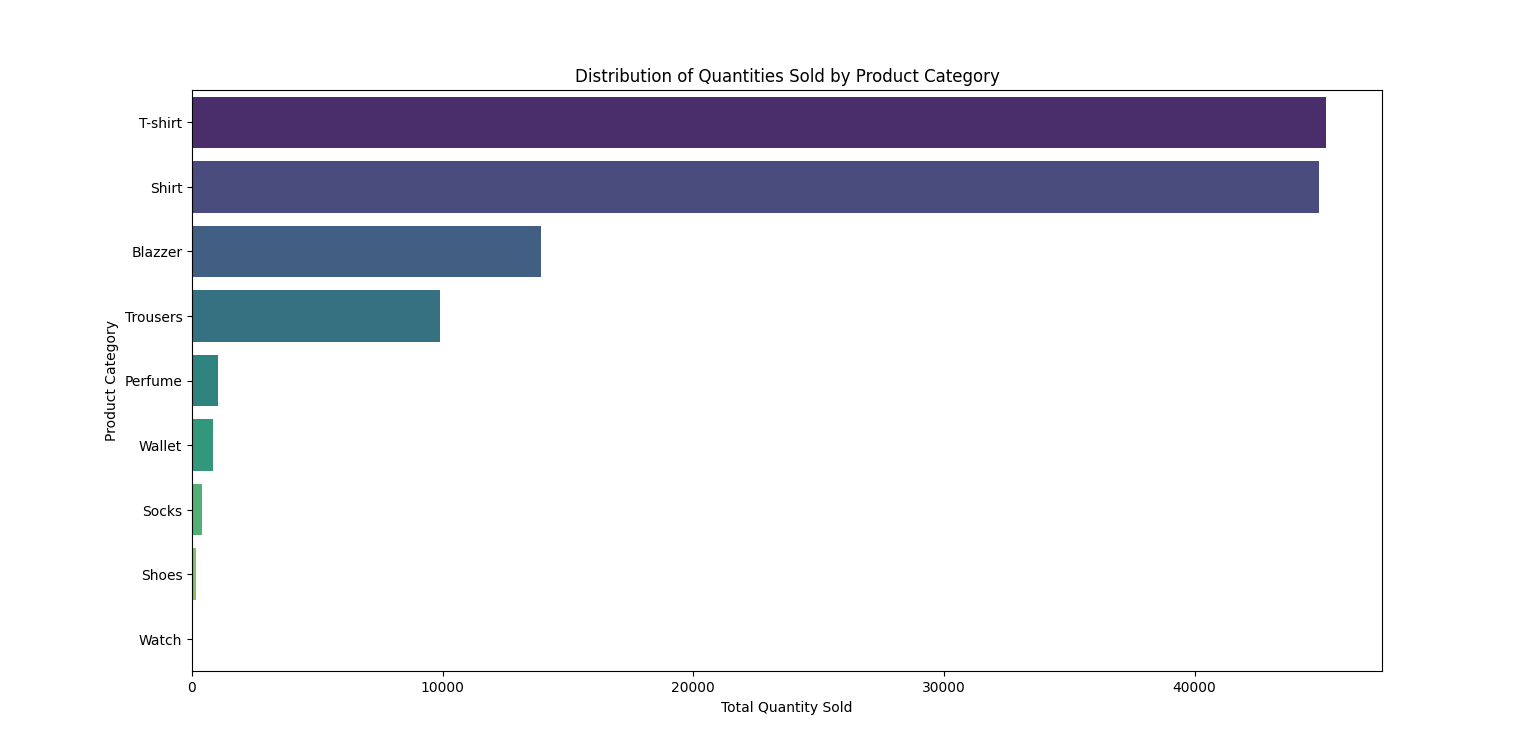


***# Group by 'Category' and sum 'Amount'***df\_grouped = df.groupby('Category').agg(total\_amount=('Amount', 'sum')).reset\_index()  
  
***# Display the grouped data***print(df\_grouped)  
df['Date'] = pd.to\_datetime(df['Date'])  
  
***# Handle missing values in 'Amount' column if necessary***df = df.dropna(subset=['Amount'])  
  
***# Aggregate sales data by month***sales\_over\_time = df.resample('M', on='Date').agg({'Amount': 'sum', 'Order ID': 'count'}).reset\_index()  
sales\_over\_time.columns = ['Month', 'Total Sales', 'Total Orders']  
 ***# Plot sales trends over time***plt.figure(figsize=(14, 7))  
sns.lineplot(data=sales\_over\_time, x='Month', y='Total Sales', marker='o')  
plt.title('Total Sales Over Time')  
plt.xlabel('Month')  
plt.ylabel('Total Sales')  
plt.grid(True)  
plt.show()

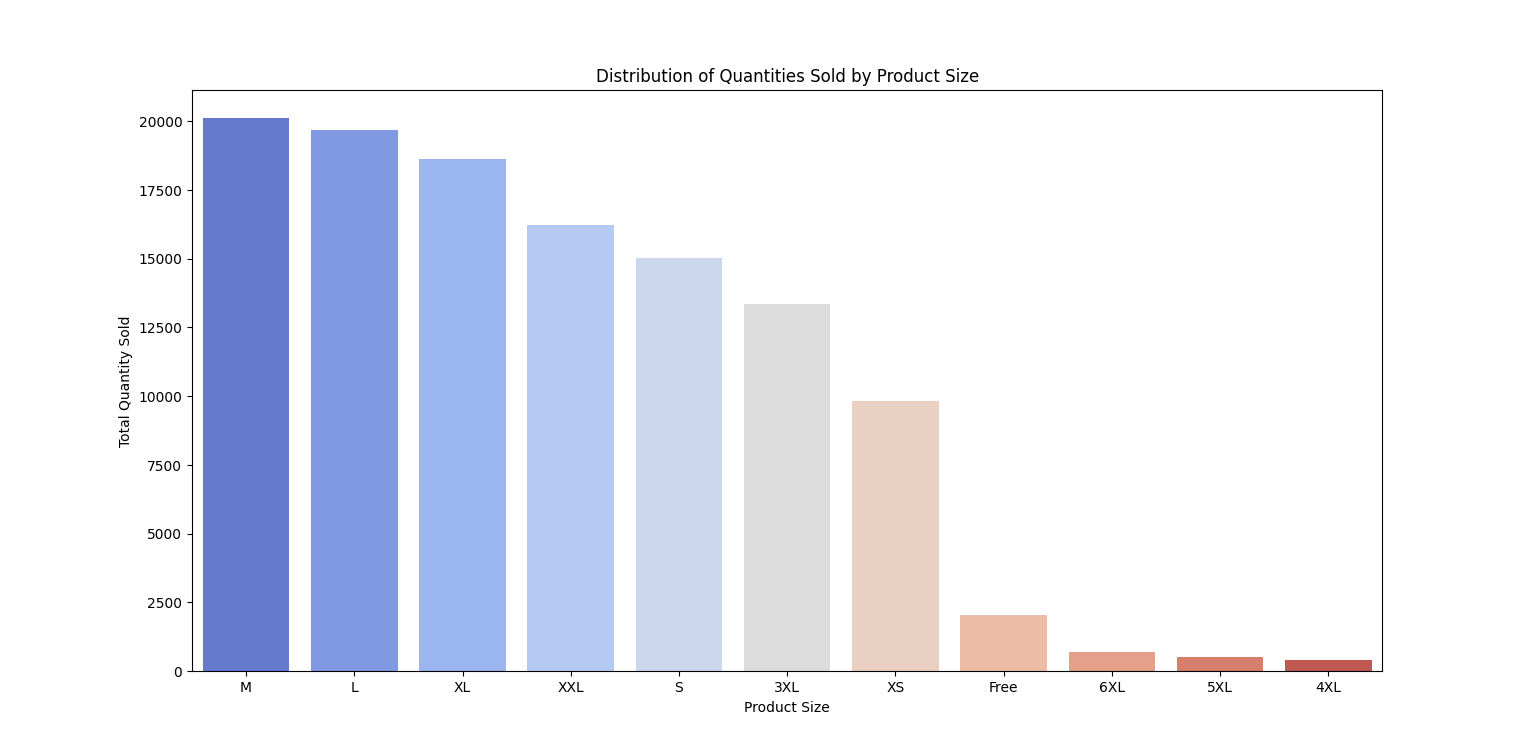




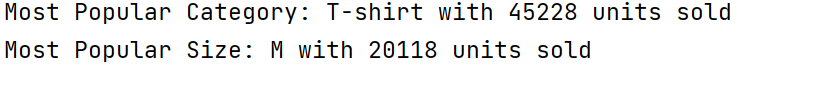
***# Analyze distribution by Category***category\_distribution = df.groupby('Category').agg(total\_quantity\_sold=('Qty', 'sum')).reset\_index()  
category\_distribution = category\_distribution.sort\_values(by='total\_quantity\_sold', ascending=False)  
  
plt.figure(figsize=(14, 7))  
sns.barplot(data=category\_distribution, x='total\_quantity\_sold', y='Category', palette='viridis')  
plt.title('Distribution of Quantities Sold by Product Category')  
plt.xlabel('Total Quantity Sold')  
plt.ylabel('Product Category')  
plt.show()



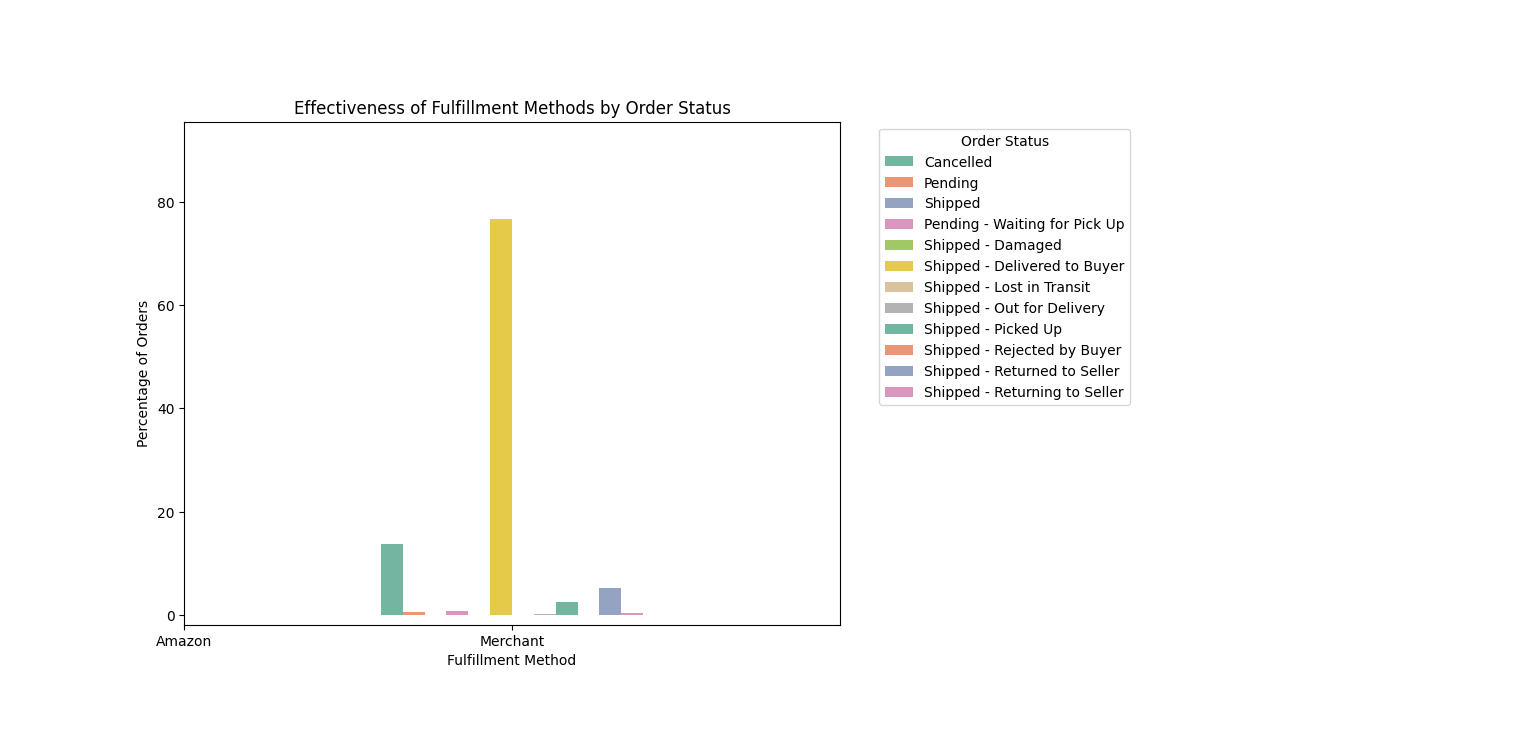
***# Analyze distribution by Size***size\_distribution = df.groupby('Size').agg(total\_quantity\_sold=('Qty', 'sum')).reset\_index()  
size\_distribution = size\_distribution.sort\_values(by='total\_quantity\_sold', ascending=False)  
  
plt.figure(figsize=(14, 7))  
sns.barplot(data=size\_distribution, x='Size', y='total\_quantity\_sold', palette='coolwarm')  
plt.title('Distribution of Quantities Sold by Product Size')  
plt.xlabel('Product Size')  
plt.ylabel('Total Quantity Sold')  
plt.show()



***# Identify the most popular products***most\_popular\_category = category\_distribution.iloc[0]  
print(f"Most Popular Category: {most\_popular\_category['Category']} with {most\_popular\_category['total\_quantity\_sold']} units sold")  
  
most\_popular\_size = size\_distribution.iloc[0]  
print(f"Most Popular Size: {most\_popular\_size['Size']} with {most\_popular\_size['total\_quantity\_sold']} units sold")

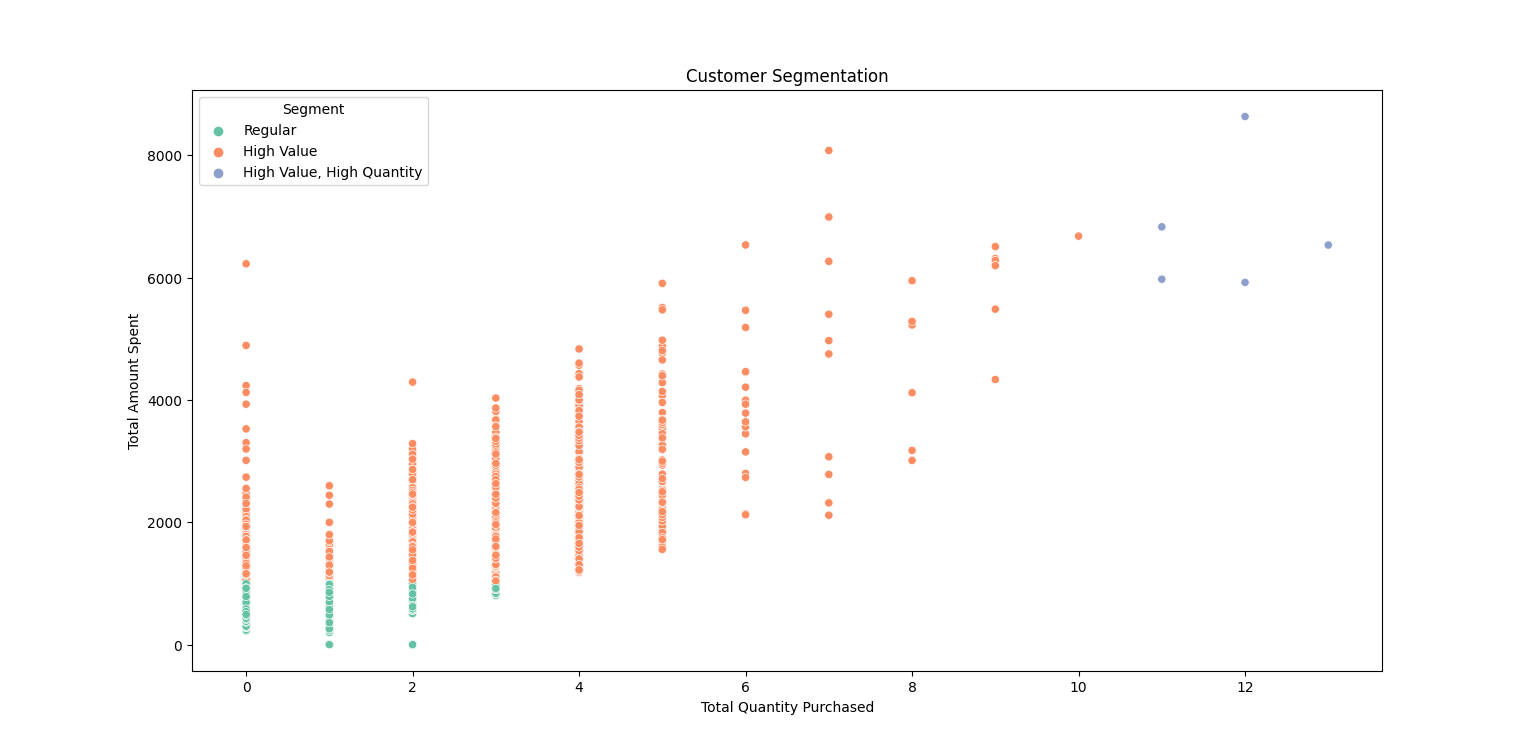


***# Check the unique fulfillment methods***print(df['Fulfilment'].unique())  
  
***# Analyze the effectiveness of fulfillment methods by order status***fulfillment\_analysis = df.groupby(['Fulfilment', 'Status']).agg(order\_count=('Order ID', 'count')).reset\_index()  
 ***# Calculate the percentage of each status within each fulfillment method***fulfillment\_analysis['percentage'] = fulfillment\_analysis.groupby('Fulfilment')['order\_count'].apply(lambda x: 100 \* x / x.sum())  
  
*#* ***Plot the effectiveness of fulfillment methods***plt.figure(figsize=(14, 7))  
sns.barplot(data=fulfillment\_analysis, x='Fulfilment', y='percentage', hue='Status', palette='Set2')  
plt.title('Effectiveness of Fulfillment Methods by Order Status')  
plt.xlabel('Fulfillment Method')  
plt.ylabel('Percentage of Orders')  
plt.legend(title='Order Status', bbox\_to\_anchor=(1.05, 1), loc='upper left')  
plt.show()

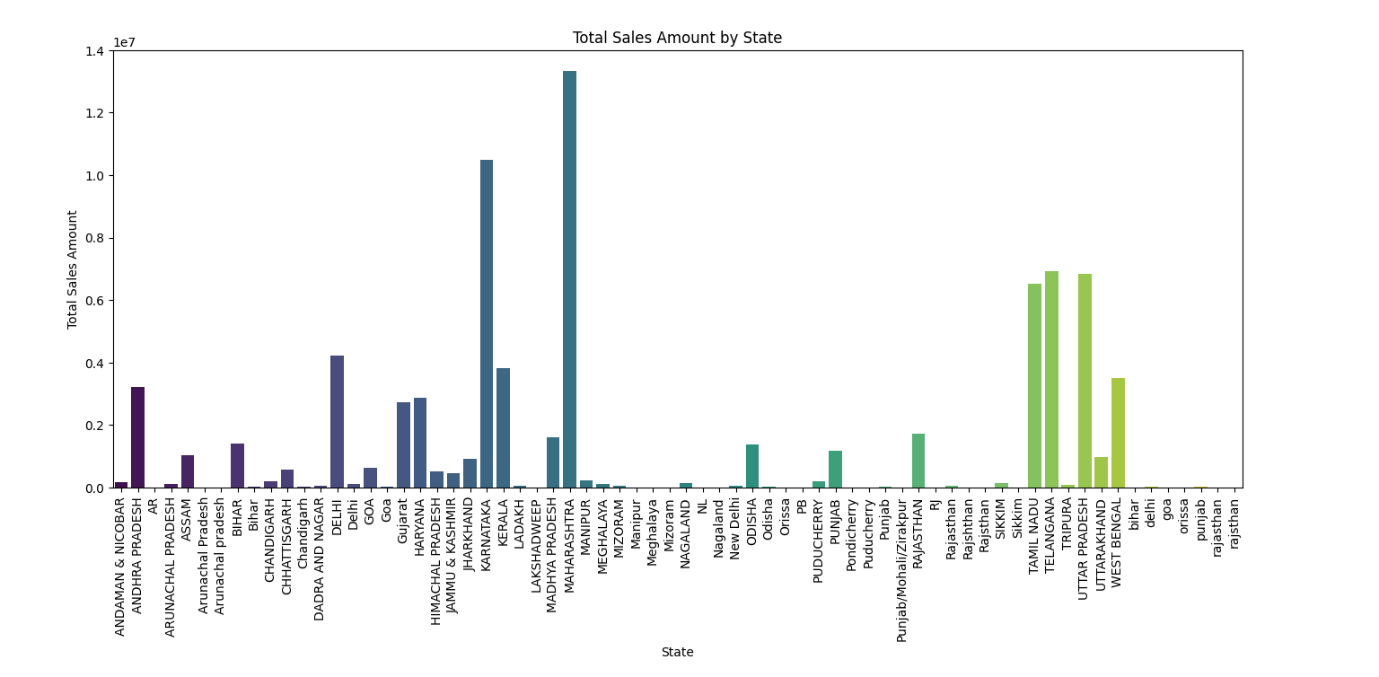


***# Aggregate features for each customer***customer\_data = df.groupby('Order ID').agg({  
 'Qty': 'sum',  
 'Amount': 'sum',  
 'ship-city': 'first',  
 'ship-state': 'first'  
}).reset\_index()

***# Define segmentation criteria***def segment\_customer(row):  
 if row['Amount'] > 1000 and row['Qty'] > 10:  
 return 'High Value, High Quantity'  
 elif row['Amount'] > 1000:  
 return 'High Value'  
 elif row['Qty'] > 10:  
 return 'High Quantity'  
 else:  
 return 'Regular'  
***# Apply the segmentation***customer\_data['Segment'] = customer\_data.apply(segment\_customer, axis=1)  
***# Analyze each segment***segment\_summary = customer\_data.groupby('Segment').agg({  
 'Qty': 'mean',  
 'Amount': 'mean',  
 'ship-city': lambda x: x.mode()[0],  
 'ship-state': lambda x: x.mode()[0]  
}).reset\_index()  
print("Segment Summary:")  
print(segment\_summary)  
***# Visualize the segments***import seaborn as sns  
import matplotlib.pyplot as plt  
plt.figure(figsize=(12, 8))  
sns.scatterplot(data=customer\_data, x='Qty', y='Amount', hue='Segment', palette='Set2', marker='o')  
plt.title('Customer Segmentation')  
plt.xlabel('Total Quantity Purchased')  
plt.ylabel('Total Amount Spent')  
plt.legend(title='Segment')  
plt.show()



df = df.dropna(subset=['ship-city', 'ship-state', 'Amount'])  
***# Aggregate sales by state***sales\_by\_state = df.groupby('ship-state').agg({  
 'Amount': 'sum',  
 'Qty': 'sum'  
}).reset\_index()  
***# Aggregate sales by city***sales\_by\_city = df.groupby('ship-city').agg({  
 'Amount': 'sum',  
 'Qty': 'sum'  
}).reset\_index()  
***# Plot total sales by state***plt.figure(figsize=(12, 6))  
sns.barplot(data=sales\_by\_state, x='ship-state', y='Amount', palette='viridis')  
plt.title('Total Sales Amount by State')  
plt.xlabel('State')  
plt.ylabel('Total Sales Amount')  
plt.xticks(rotation=90)  
plt.show()



***# Plot total sales by city***plt.figure(figsize=(12, 6))  
top\_cities = sales\_by\_city.nlargest(10, 'Amount') *# Show top 10 cities*sns.barplot(data=top\_cities, x='ship-city', y='Amount', palette='viridis')  
plt.title('Top 10 Cities by Total Sales Amount')  
plt.xlabel('City')  
plt.ylabel('Total Sales Amount')  
plt.xticks(rotation=90)  
plt.show()

