Amazon Sales Analysis

Python Data Analyst Project

About Amazon:

Amazon is a multinational technology company that specializes in e-commerce, cloud computing, digital streaming, and artificial intelligence. Founded in 1994 by Jeff Bezos, Amazon has grown to become one of the largest companies in the world.

Industry Scope:

Amazon operates in a wide range of industries, including:

- **E-commerce:** Retailing a vast array of products, from books and electronics to groceries and clothing.
- **Cloud Computing:** Providing cloud computing services through Amazon Web Services (AWS).
- **Digital Streaming:** Offering streaming services for video (Prime Video), music (Amazon Music), and audiobooks (Audible).
- **Artificial Intelligence:** Developing and utilizing AI technologies for various applications, such as Alexa, autonomous vehicles, and robotics.

Purpose of Analysis:

Analyzing the provided dataset can help Amazon gain valuable insights into its operations and customer behavior. Some key areas of analysis include:

- **Sales Performance:** Understanding sales trends, identifying top-selling products, and analyzing customer purchasing patterns.
- **Customer Behavior:** Analyzing customer demographics, preferences, and purchase history to tailor marketing strategies.
- **Inventory Management:** Optimizing inventory levels to avoid stockouts and reduce holding costs.
- **Logistics and Shipping:** Analyzing shipping performance, identifying bottlenecks, and improving delivery times.
- **Marketing Effectiveness:** Evaluating the impact of marketing campaigns on sales and customer acquisition.

Dataset Overview:

The provided dataset appears to contain information about Amazon orders. Some of the key columns include:

- **Order ID:** Unique identifier for each order.
- **Date:** Date of the order.
- **Status:** Current status of the order (e.g., Shipped, Cancelled).

- **Fulfillment:** Fulfillment method used for the order (e.g., Amazon, Merchant).
- **Sales Channel:** Sales channel through which the order was placed (e.g., Website, Mobile App).
- **Category:** Product category (e.g., T-shirt, Shirt, Blazer).
- **Size:** Product size.
- **Courier Status:** Status of the courier delivery (e.g., Shipped, On the Way).
- **Quantity:** Number of items ordered.
- **Currency:** Currency of the order.
- **Amount:** Total amount of the order.
- **Shipping Information:** Shipping address details.
- **B2B:** Indicates whether the order is B2B or B2C.
- **Fulfilled By:** Entity that fulfilled the order.

Import Necessary Libraries

[12]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Optional: Style the plots
sns.set(style="whitegrid")
```

Load the Data

[13]:
```
 # Load data data_path = "/content/drive/MyDrive/Data
Analysis/Python Project/Amazon Sales␣
 ↪Analysis/Amazon Sale DataSet.csv" data =
pd.read_csv(data_path,
encoding='unicode_escape')
```

Initial Data Exploration

[14]:
```
# View first
data.head()
```

[14]:
```
   index            Order ID     Date                       Status \
0      0 405-8078784-5731545 04-30-22  Cancelled
1      1 171-9198151-1101146 04-30-22 Shipped - Delivered to Buyer
2      2 404-0687676-7273146 04-30-22  Shipped
3      3 403-9615377-8133951 04-30-22  Cancelled
4      4 407-1069790-7240320 04-30-22  Shipped

  Fulfilment Sales Channel ship-service-level Category Size Courier
  Status \
```

```
0   Merchant     Amazon.in  Standard   T-shirt    S     On the Way
1   Merchant     Amazon.in  Standard   Shirt 3XL  Shipped
2     Amazon    Amazon.in        Expedited   Shirt  XL      Shipped
3   Merchant    Amazon.in        Standard  Blazzer   L     On    the
                                                          Way
4     Amazon    Amazon.in        Expedited Trousers 3XL     Shipped
   … currency Amount   ship-city ship-state ship-postal-code \
0 …       INR 647.62      MUMBAI MAHARASHTRA      400081.0
1 …       INR 406.00   BENGALURU   KARNATAKA      560085.0
2 …      INR 329.00 NAVI MUMBAI MAHARASHTRA      410210.0
3 …       INR 753.33  PUDUCHERRY  PUDUCHERRY      605008.0
4 …       INR 574.00     CHENNAI TAMIL NADU      600073.0

  ship-country  B2B fulfilled-by New PendingS
0           IN False   Easy Ship NaN    NaN
1           IN False   Easy Ship NaN    NaN
2           IN True    NaN NaN    NaN
3           IN False   Easy Ship NaN    NaN
4           IN False   NaN NaN    NaN
[5 rows x 21 columns]
```

[15]:
```
# and last rows
data.tail()
```

[15]:
```
        index           Order ID     Date  Status Fulfilment \
128971 128970 406-6001380-7673107 05-31-22 Shipped     Amazon
128972 128971 402-9551604-7544318 05-31-22 Shipped     Amazon
128973 128972 407-9547469-3152358 05-31-22 Shipped     Amazon
128974 128973 402-6184140-0545956 05-31-22 Shipped     Amazon
128975 128974 408-7436540-8728312 05-31-22 Shipped     Amazon

       Sales Channel ship-service-level Category Size Courier Status … \
128971    Amazon.in    Expedited  Shirt XL    Shipped …
128972    Amazon.in    Expedited T-shirt     M     Shipped …
128973    Amazon.in    Expedited Blazzer XXL Shipped …
128974    Amazon.in    Expedited T-shirt     XS    Shipped …
128975    Amazon.in    Expedited T-shirt     S     Shipped …

       currency Amount ship-city  ship-state ship-postal-code \
128971 INR 517.0 HYDERABAD TELANGANA 500013.0 128972 INR 999.0
GURUGRAM HARYANA 122004.0
128973      INR 690.0 HYDERABAD   TELANGANA  500049.0
128974      INR 1199.0 Halol Gujarat    389350.0
128975      INR 696.0  Raipur CHHATTISGARH   492014.0

       ship-country  B2B fulfilled-by New PendingS
```

```
128971          IN False    NaN NaN    NaN
128972          IN False    NaN NaN    NaN
128973          IN False    NaN NaN    NaN
128974          IN False    NaN NaN    NaN
128975          IN False    NaN NaN    NaN

[5 rows x 21 columns]
```

[16]:
```
# Get a summary of the data types and missing values
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 128976 entries, 0 to
128975 Data columns (total 21
columns):
#   Column            Non-Null Count Dtype
--- ------            -------------- -----
0   index             128976      non-null
                      int64
1   Order ID          128976      non-null
                      object
2   Date              128976      non-null
                      object
3   Status            128976      non-null
                      object
4   Fulfilment        128976      non-null
                      object
5   Sales Channel     128976      non-null
                      object
 6 ship-service-level 128976 non-null object
 7  Category          128976 non-null
                      object
 8   Size             128976 non-null
                      object
 9  Courier Status    128976 non-null
                      object
 10 Qty               128976 non-null
                      int64
 11 currency          121176 non-null
                      object
 12 Amount            121176       non-null
                      float64
 13 ship-city         128941 non-null
                      object
 14 ship-state        128941 non-null
                      object
```

```
 15 ship-postal-code  128941       non-null
                      float64
 16 ship-country      128941 non-null
                      object
 17 B2B               128976 non-null bool
 18 fulfilled-by      39263 non-null object
 19 New               0 non-null    float64
 20 PendingS          0 non-null    float64
dtypes: bool(1), float64(4), int64(2),
object(14) memory usage: 19.8+ MB
```

Data Cleaning

[17]:
```
# Drop unnecessary columns
data.drop(['New', 'PendingS'], axis=1, inplace=True)
```

[18]:
```
# Check for missing values and drop rows with missing values
print(data.isnull().sum())
data.dropna(inplace=True)
```

```
index                   0
Order ID                0
Date                    0
Status                  0
Fulfilment              0
Sales Channel           0
ship-service-level      0
Category                0
Size                    0
Courier Status          0
Qty                     0
currency             7800
Amount               7800
ship-city              35
ship-state             35
ship-postal-code       35
ship-country           35
B2B                     0
fulfilled-by        89713
dtype: int64
```

[19]:
```
 # Convert data types where needed data['ship-postal-
code'] = data['ship-postal-code'].astype(int)
data['Date'] = pd.to_datetime(data['Date'])
```

```
<ipython-input-19-30b9c13a928c>:3: UserWarning: Could not infer
format, so each element will be parsed individually, falling back to
```

`dateutil`. To ensure parsing is consistent and as-expected, please specify a format. data['Date'] = pd.to_datetime(data['Date'])

[20]:
```
# Rename columns for clarity
data.rename(columns={'Qty': 'Quantity'},
inplace=True)
```

Exploratory Data Analysis (EDA)

Descriptive Statistics

[22]:
```
# Display descriptive statistics
data.describe(include='number')
```

[22]:

	index	Quantity	Amount	ship-postal-code
count	37514.000000	37514.000000	37514.000000	37514.000000
mean	60953.809858	0.867383	646.553960	463291.552754
std	36844.853039	0.354160	279.952414	194550.425637
min	0.000000	0.000000	0.000000	110001.000000
25%	27235.250000	1.000000	458.000000	370465.000000
50%	63470.500000	1.000000	629.000000	500019.000000
75%	91790.750000	1.000000	771.000000	600042.000000
max	128891.000000	5.000000	5495.000000	989898.000000

Categorical Data Analysis

[23]:
```
# View unique values in categorical
columns for col in
data.select_dtypes(include=['object']):
    data[col].unique()
```

Data Visualization

Size Distribution

[29]:
```
# Use a vibrant color palette for the count
plot sns.countplot(x='Size', data=data,
palette="viridis") plt.xlabel('Size')
plt.ylabel('Number of Orders')
plt.title('Distribution of Orders by Size')
plt.show()
```

<ipython-input-29-6b939c7a015c>:2: FutureWarning:

```
Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `x` variable to `hue` and set
`legend=False` for the same effect.
```

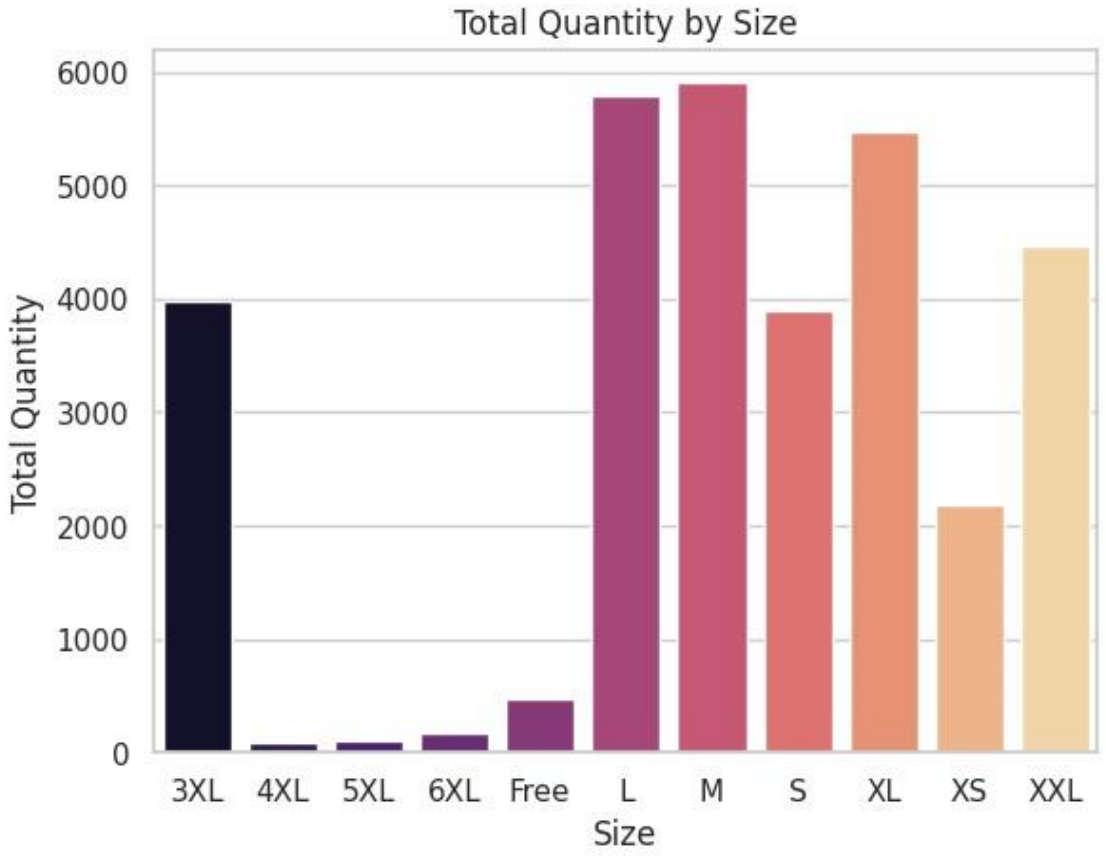
```
  sns.countplot(x='Size', data=data, palette="viridis")
```

** Quantity by Size **

[31]:
```
 # Bar plot with a "magma" color palette size_qty_sum =
data.groupby('Size')['Quantity'].sum().reset_index()
sns.barplot(x='Size', y='Quantity', data=size_qty_sum,
palette="magma") plt.xlabel('Size') plt.ylabel('Total
Quantity') plt.title('Total Quantity by Size') plt.show()

<ipython-input-31-f24068ef015b>:3: FutureWarning:
```

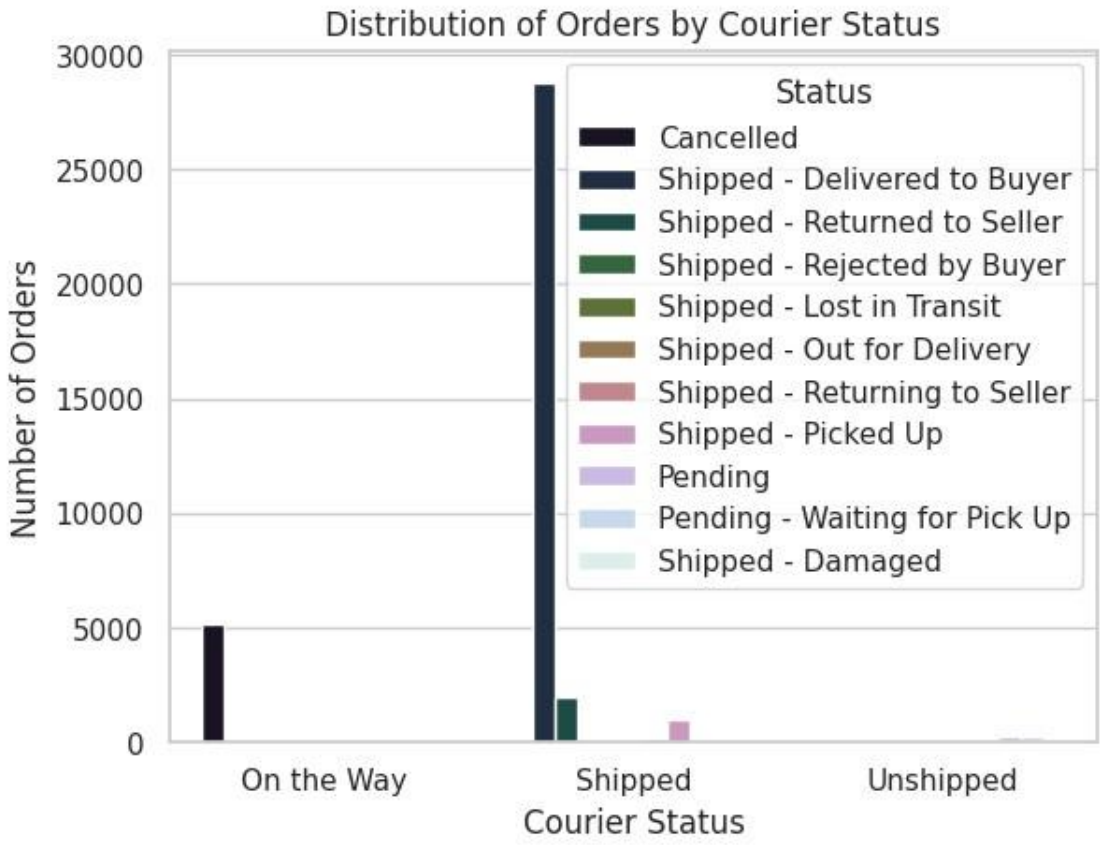
```
Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `x` variable to `hue` and set
`legend=False` for the same effect.
```

```
sns.barplot(x='Size', y='Quantity', data=size_qty_sum,
palette="magma")
```



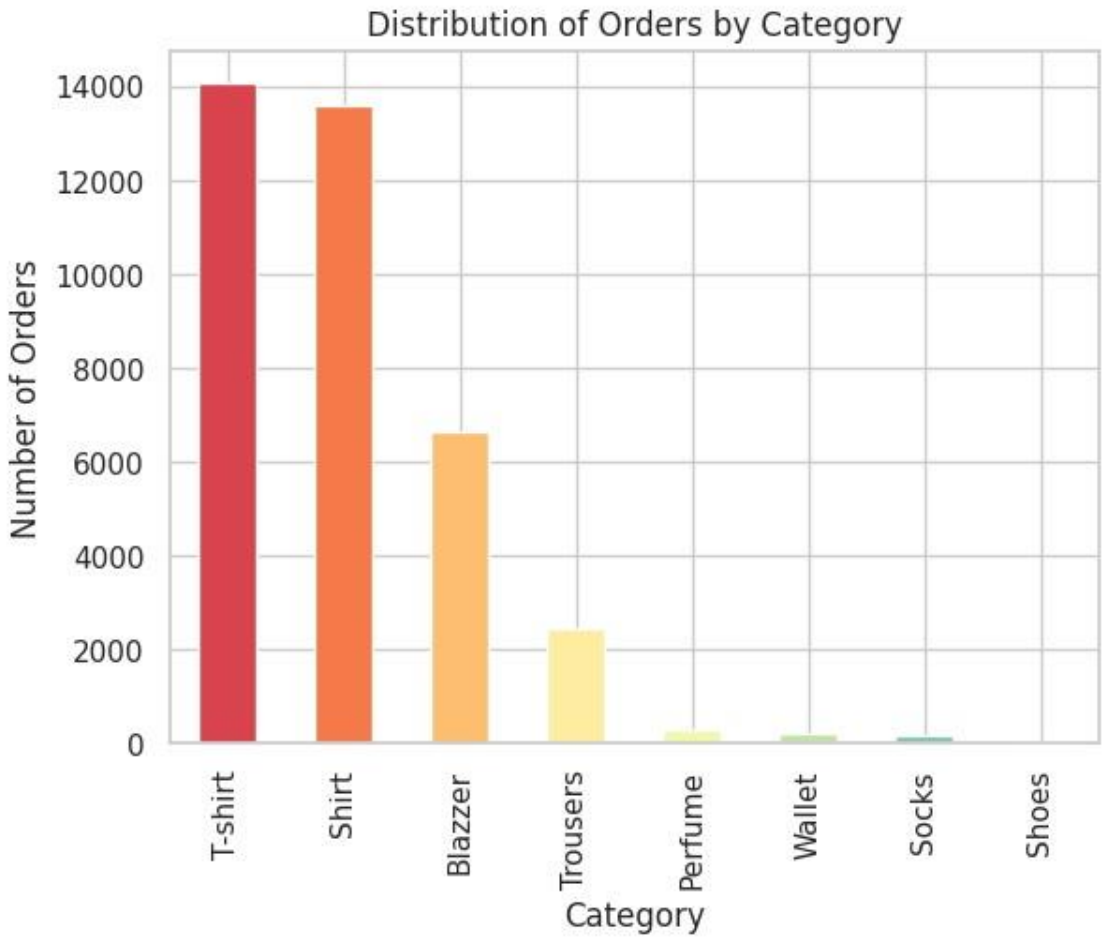
Courier Status Distribution

[32]:
```
# Count plot with a "cubehelix" color palette
sns.countplot(data=data, x='Courier Status', hue='Status',
palette="cubehelix") plt.xlabel('Courier Status') plt.ylabel('Number
of Orders') plt.title('Distribution of Orders by Courier Status')
plt.show()
```



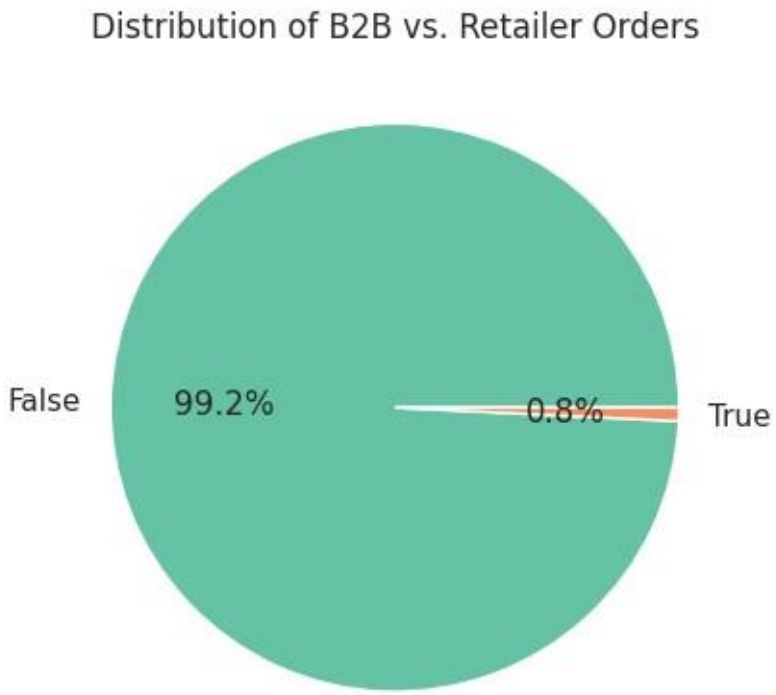
** Category Distribution **

[33]:
```
# Bar plot using a "Spectral" color palette
data['Category'].value_counts().plot(kind='bar', color=sns.
 ↪color_palette("Spectral",
len(data['Category'].unique())))
plt.xlabel('Category') plt.ylabel('Number of
Orders') plt.title('Distribution of Orders by
Category') plt.show()
```



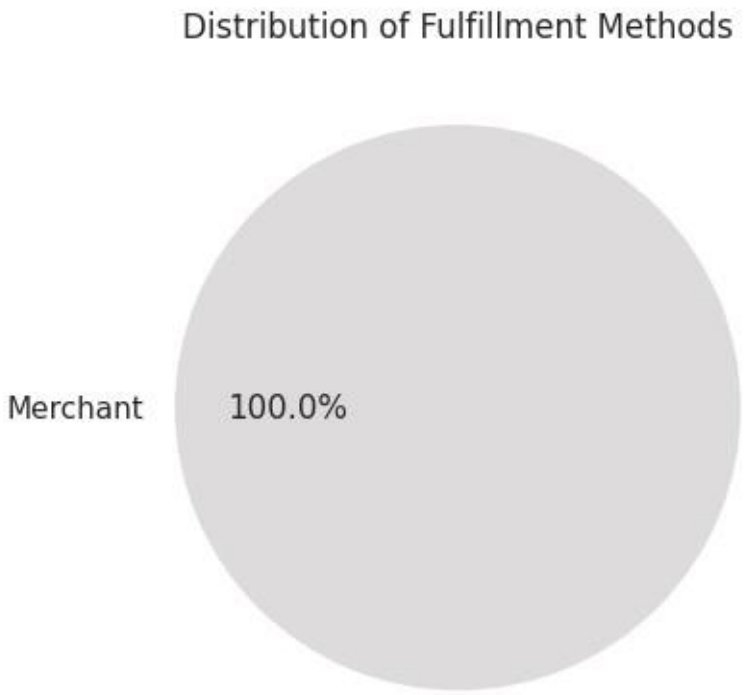
B2B vs. Retailer Orders

[34]:
```
 # Pie chart with custom colors b2b_counts =
data['B2B'].value_counts() plt.pie(b2b_counts,
labels=b2b_counts.index, autopct='%1.1f%%', colors=sns.
 ↪color_palette("Set2"))
plt.title('Distribution of B2B vs. Retailer
Orders') plt.show()
```



Fulfillment Method

[35]:
```
 # Pie chart with "coolwarm" colors fulfillment_counts =
data['Fulfilment'].value_counts() plt.pie(fulfillment_counts,
labels=fulfillment_counts.index, autopct='%1.1f%%',␣
 ↪colors=sns.color_palette("coolwarm",
len(fulfillment_counts))) plt.title('Distribution of
Fulfillment Methods') plt.show()
```



Customer Segmentation (Optional)

[38]:
```
# Check the first few rows of the data and list column names
data.columns
```

[38]:
```
Index(['index', 'Order ID', 'Date', 'Status', 'Fulfilment', 'Sales
Channel',
  'ship-service-level', 'Category', 'Size', 'Courier Status', 'Quantity',
           'currency', 'Amount', 'ship-city', 'ship-state', 'ship-postal-
           code',
           'ship-country', 'B2B', 'fulfilled-by'],
          dtype='object')
```

Create Purchase Frequency and Average Order Value for Each Category

[40]:
```
# Import necessary libraries
import pandas as pd

# Calculate purchase frequency by counting the number of orders
per Category data['Purchase Frequency'] =
data.groupby('Category')['Order ID']. ↪transform('count')
```

```
# Calculate the average order value for each category
data['Average Order Value'] =
data.groupby('Category')['Amount'].
 ↪transform('mean')
# Create a segmentation based on the average order value # (e.g.,
'High', 'Medium', 'Low' based on quartiles or thresholds)
data['Customer Segment'] = pd.cut(data['Average Order Value'],
bins=[-float('inf'), data['Average Order␣
 ↪Value'].quantile(0.33), data['Average Order Value'].quantile(0.
 ↪66), float('inf')], labels=['Low', 'Medium', 'High'])
```

Verify New Columns

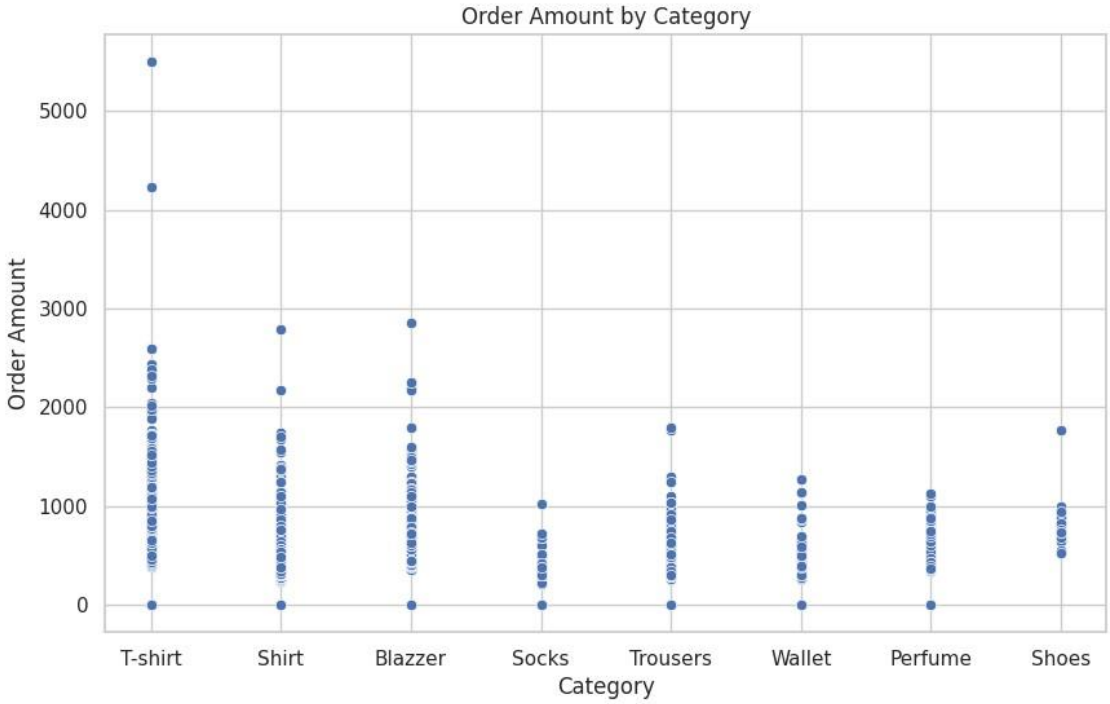
[41]:
```
data[['Category', 'Purchase Frequency', 'Average Order Value',
 'Customer␣ ↪Segment']].head()
```

[41]:

	Category	Purchase Frequency	Average Order Value	Customer Segment
0	T-shirt	14062	822.372824	Medium
1	Shirt	13595	450.287097	Low
3	Blazzer	6661	741.678745	Medium
7	Shirt	13595	450.287097	Low
12	Shirt	13595	450.287097	Low

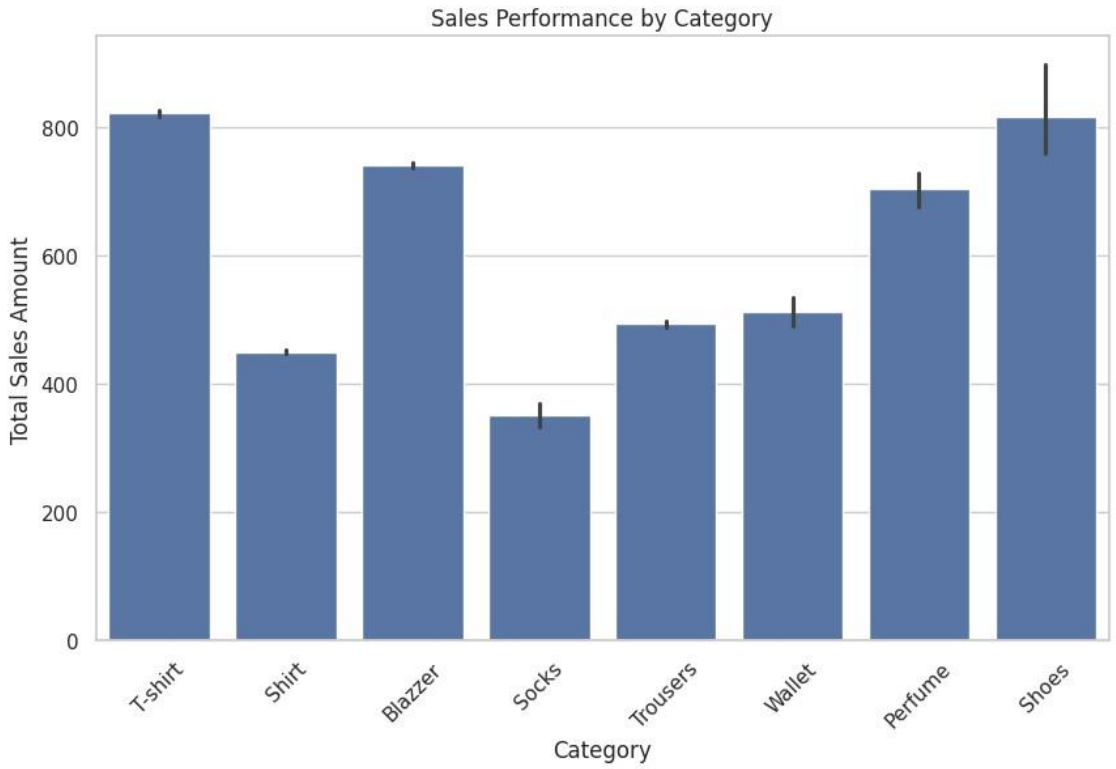
[42]:
```
# Assuming 'Category' represents customer segmentation and
 'Amount' is the␣ ↪value for each order

# Scatter plot to show order amount by category for
segmentation plt.figure(figsize=(10, 6))
sns.scatterplot(x='Category', y='Amount', data=data,
palette="plasma") plt.title('Order Amount by Category')
plt.xlabel('Category')
plt.ylabel('Order Amount')
plt.show()
```

```
<ipython-input-42-4a84187fe74c>:5: UserWarning: Ignoring `palette`
because no `hue` variable has been assigned.
sns.scatterplot(x='Category', y='Amount', data=data, palette="plasma")
```



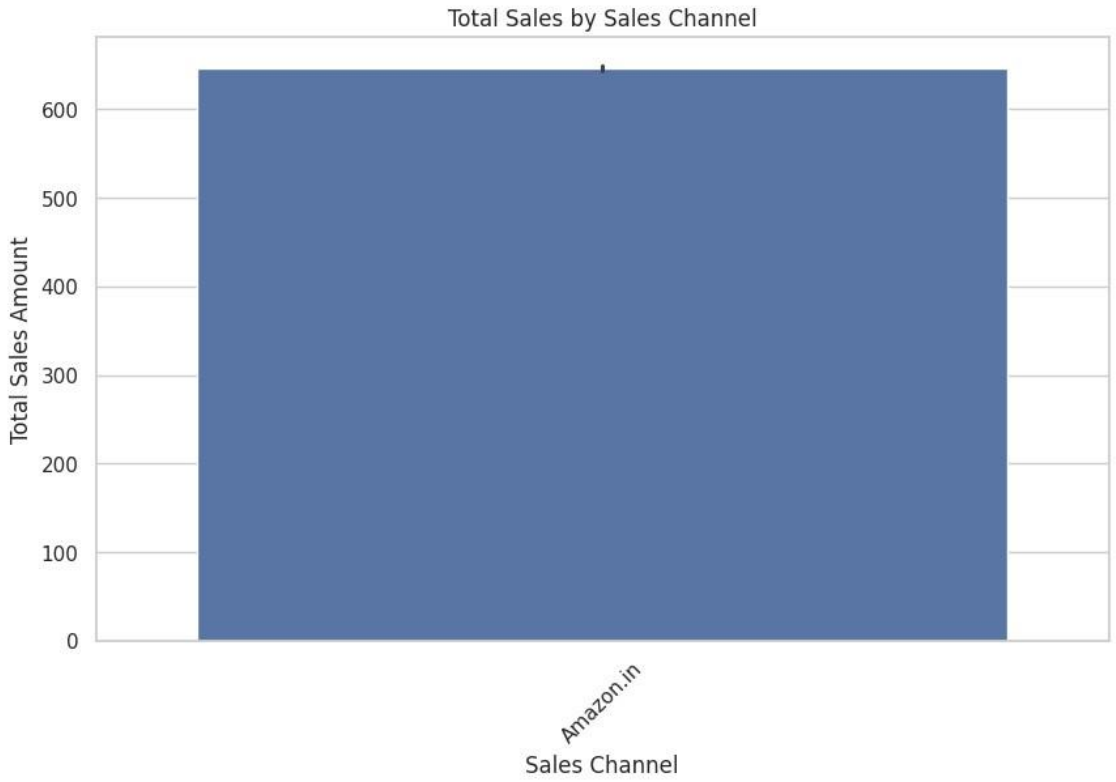
Sales Performance by Category:

[47]:
```
# Bar plot to visualize sales performance by category
plt.figure(figsize=(10, 6))
sns.barplot(x='Category',  ='Amount', data=data)
plt.title('Sales Performance by Category')
plt.xlabel('Category')
plt.ylabel('Total Sales Amount')
plt.xticks(rotation=45)
plt.show()
```



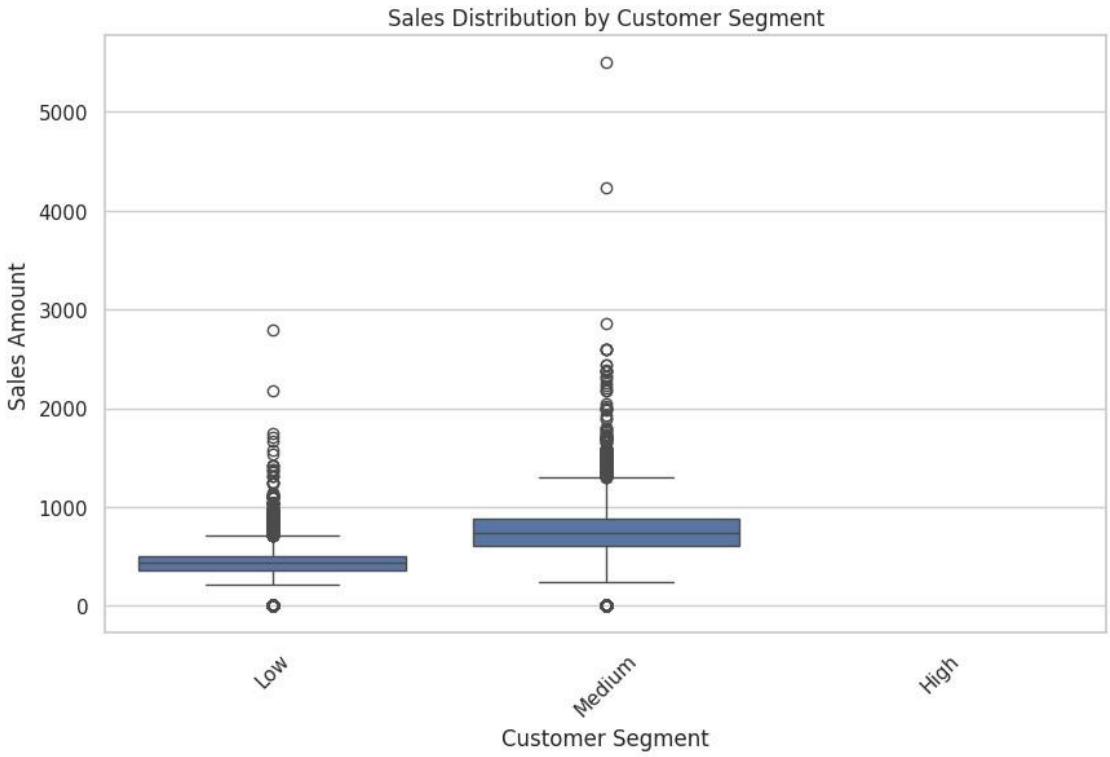
Sales Channel Analysis

```
[48]: # Bar plot to visualize total sales by sales
channel plt.figure(figsize=(10, 6))
sns.barplot(x='Sales Channel', y='Amount',
data=data) plt.title('Total Sales by Sales
Channel') plt.xlabel('Sales Channel')
plt.ylabel('Total Sales Amount')
plt.xticks(rotation=45) plt.show()
```



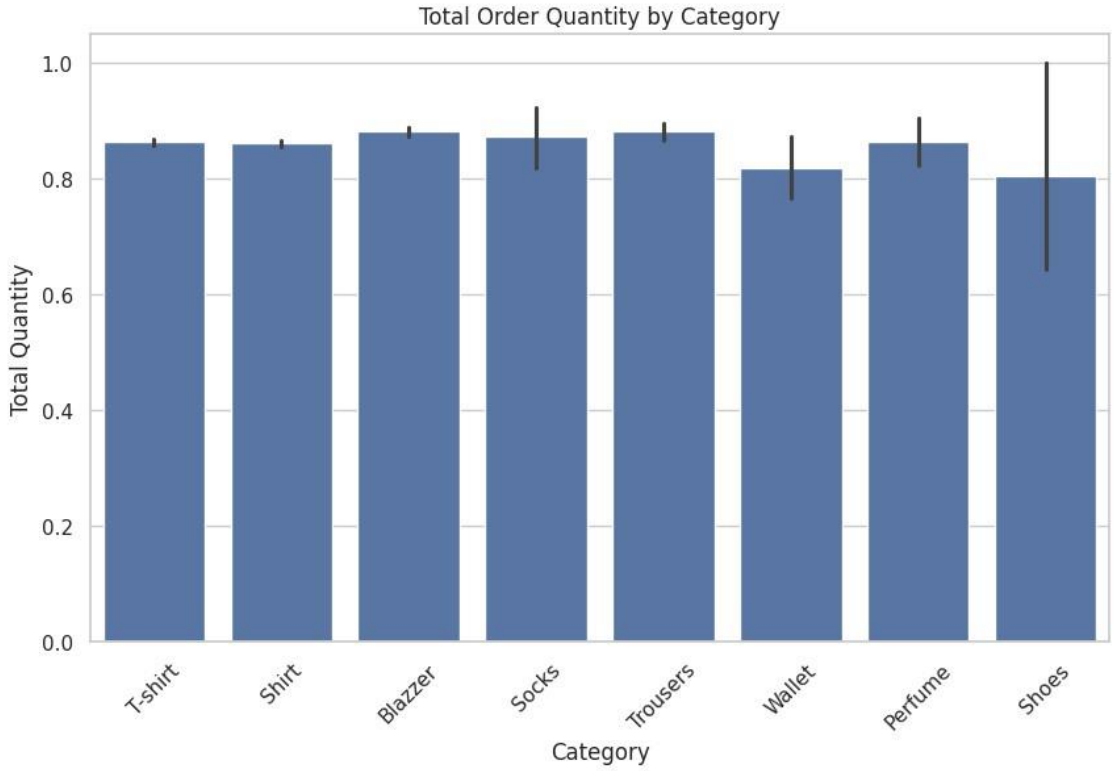
Customer Segment Analysis

[49]:
```
# Box plot to visualize sales distribution by
customer segment plt.figure(figsize=(10, 6))
sns.boxplot(x='Customer Segment', y='Amount',
data=data) plt.title('Sales Distribution by Customer
Segment') plt.xlabel('Customer Segment')
plt.ylabel('Sales Amount') plt.xticks(rotation=45)
plt.show()
```



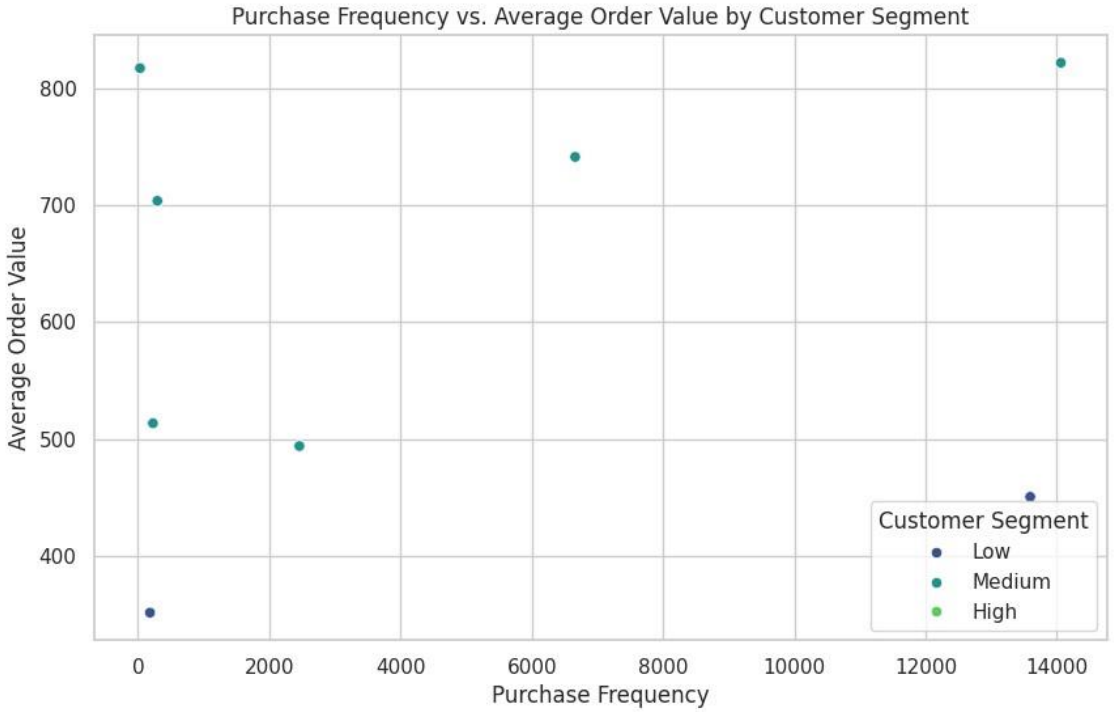
Order Quantity Analysis:

[50]:
```
# Bar plot to visualize total order quantity by
category plt.figure(figsize=(10, 6))
sns.barplot(x='Category', y='Quantity',
data=data) plt.title('Total Order Quantity by
Category')
plt.xlabel('Category')
plt.ylabel('Total Quantity')
plt.xticks(rotation=45)
plt.show()
```



** Purchase Frequency vs. Average Order Value **

[51]:
```
# Scatter plot for Purchase Frequency vs. Average Order Value
plt.figure(figsize=(10, 6)) sns.scatterplot(x='Purchase
Frequency', y='Average Order Value', data=data,␣
↪hue='Customer Segment', palette="viridis") plt.title('Purchase
Frequency vs. Average Order Value by Customer Segment')
plt.xlabel('Purchase Frequency') plt.ylabel('Average Order
Value') plt.legend(title='Customer Segment') plt.show()
```



Conclusion

In this analysis, we explored various dimensions of the sales data, leveraging visualizations to gain insights into key performance indicators. Here are some key takeaways based on the visualizations we generated:

1. **Sales Performance by Category:** The bar plot of sales performance by category highlighted which categories contribute the most to overall sales. This insight can guide inventory management and marketing strategies.
2. **Sales Channel Analysis:** Analyzing total sales by sales channel revealed the effectiveness of different sales channels. Understanding which channels yield the highest sales can help optimize resource allocation and enhance sales strategies.
3. **Customer Segment Analysis:** The box plot of sales distribution by customer segment provided insights into how different segments perform financially. Identifying high-performing segments can aid in targeted marketing efforts and customer relationship management.
4. **Order Quantity Analysis:** Visualizing total order quantity by category helped us understand which categories are most popular among customers. This information is crucial for managing stock levels and planning promotional activities.
5. **Purchase Frequency vs. Average Order Value:** The scatter plot illustrating the relationship between purchase frequency and average order value revealed trends in customer purchasing behavior.

Identifying segments with high purchase frequency but low average order value can present opportunities for upselling or cross-selling.

Final Thoughts

The visualizations provide a comprehensive overview of the sales data, allowing for a deeper understanding of customer behavior, product performance, and overall sales strategies. By utilizing these insights, businesses can make informed decisions, enhance operational efficiency, and ultimately drive growth.

For further analysis, consider exploring additional metrics, such as customer retention rates, seasonal trends, or the impact of marketing campaigns on sales performance. Continuous data monitoring and analysis will help adapt strategies to meet changing market demands and improve business outcomes.

Shaun Mia | LinkedIn