**INFOSYS MILESTONE-1 DOCUMENTATION**

**About Dataset:**

The provided dataset contains information about transactions made on an e-commerce platform. Each transaction is characterized by the following attributes:

* **Transaction ID:** A unique identifier for each transaction.
* **Date:** The date of the transaction.
* **Product Category:** The category to which the purchased product belongs (e.g., Electronics, Home Appliances, Clothing, Books, Beauty Products).
* **Product Name:** The specific name of the product purchased.
* **Units Sold:** The quantity of the product purchased.
* **Unit Price:** The price per unit of the product.
* **Total Revenue:** The total revenue generated by the transaction (Units Sold \* Unit Price).
* **Region:** The geographic region where the purchase was made (North America, Europe, Asia).
* **Payment Method:** The payment method used for the transaction (Credit Card, PayPal, Debit Card).

This dataset can be used for various analytical purposes, such as:

* **Sales Analysis:** Analyzing sales trends over time, identifying top-selling products, and understanding regional sales patterns.
* **Customer Segmentation:** Segmenting customers based on their purchasing behavior, preferences, and demographics.
* **Inventory Management:** Optimizing inventory levels based on sales data and demand forecasting.
* **Marketing Analysis:** Evaluating the effectiveness of marketing campaigns and promotions.

**Code:** print(df.isnull().sum()) #to check any missing values present in dataset

**Output:**

Transaction ID 0

Date 0

Product Category 0

Product Name 0

Units Sold 0

Unit Price 0

Total Revenue 0

Region 0

Payment Method 0

dtype: int64

**Code:**

#scatter plots used to identify correlation between two numerical variables

import matplotlib.pyplot as plt

plt.figure(figsize=(8,6))

plt.scatter(df['Unit Price'],df['Total Revenue'],color='green' ,alpha=0.6)

plt.title("Scatter plot of Total Revenue vs. Unit Price")

plt.xlabel("Unit Price")

plt.ylabel("Total Revenue")

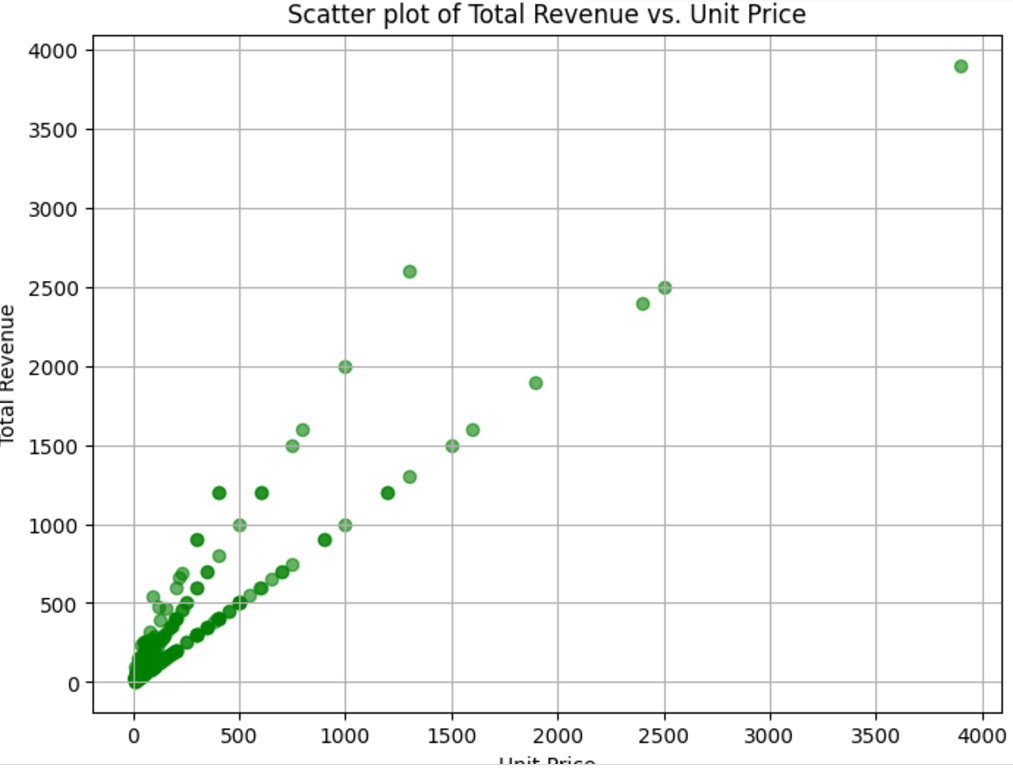
plt.grid(True)

plt.show()

**Explanation:**

This code visualizes the relationship between the unit price of a product and the total revenue generated from its sales, helping to understand if and how these two variables correlate. For example, you might observe a positive correlation where higher unit prices generally lead to higher total revenue.

**Output:**

****

**Code:**

import matplotlib.pyplot as plt

region\_counts=df['Region'].value\_counts()

plt.figure(figsize=(8,8))

plt.pie(region\_counts,labels=region\_counts.index,autopct='%1.1f%%',startangle=140,colors=plt.cm.viridis(np.linspace(0,1,len(region\_counts))))

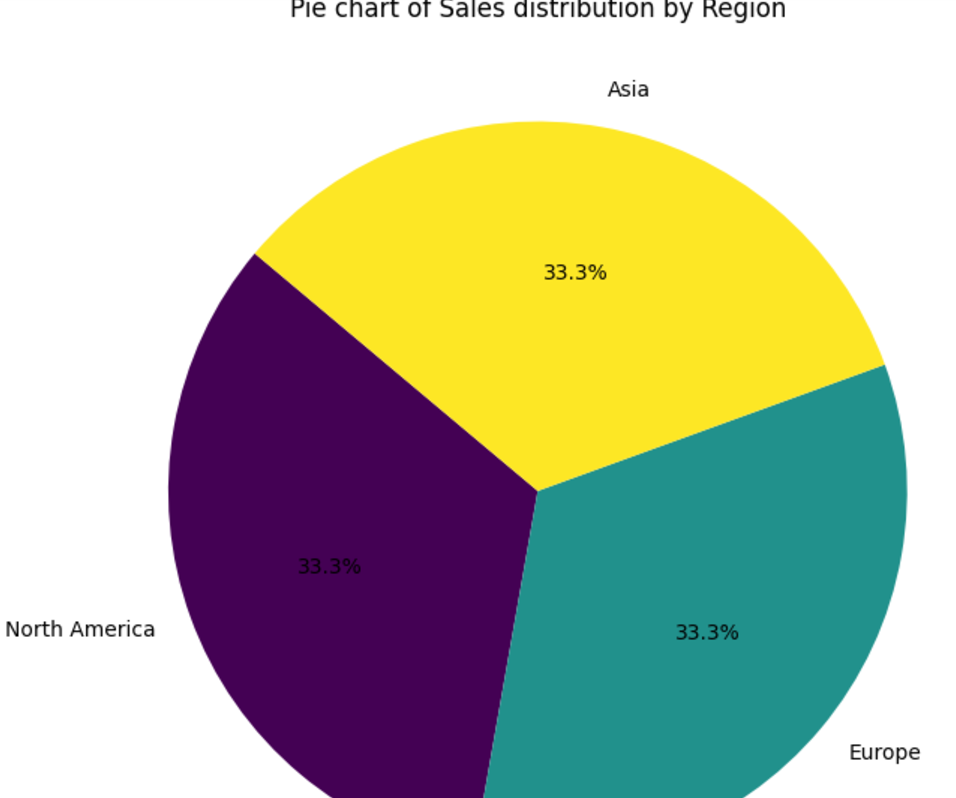
plt.title("Pie chart of Sales distribution by Region")

plt.show()

**Explanation:**

This code takes the sales data, calculates the total sales for each region, and then presents this information visually as a pie chart. Each slice of the pie represents a region, and its size corresponds to the proportion of total sales from that region**.** This allows for easy comparison of sales performance across different geographical areas.

**Output:**

****

**Code:**

import matplotlib.pyplot as plt

plt.figure(figsize=(10,6))

plt.plot(df['Date'],df['Total Revenue'],marker='o',linestyle='-',color='blue',label='Total Revenue')

plt.title("Line graph of Total Revenue over time")

plt.xlabel('Date')

plt.ylabel("Total Revenue")

plt.xticks(rotation=45) #for better readability of x labels

plt.legend()

plt.grid(True)

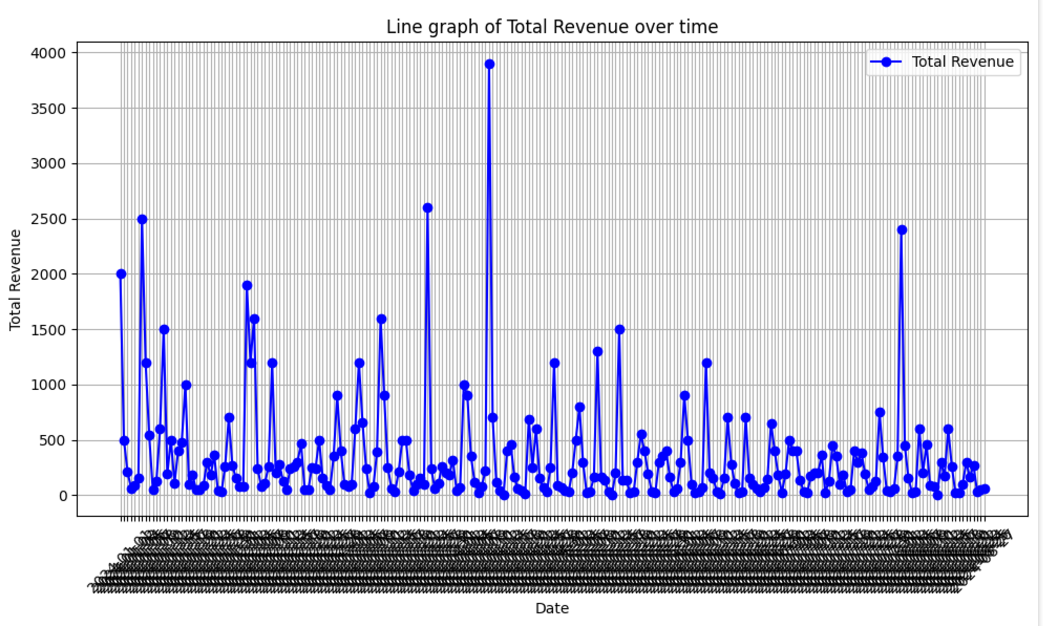
plt.tight\_layout()

plt.show()

**Explanation:**

This code visualizes how total revenue has changed over time, providing insights into sales trends and patterns. By observing the line's slope and fluctuations, you can identify periods of growth, decline, or stability in revenue**.**

**Output:**

****

**Code:**

#histogram of Total Revenue

plt.figure(figsize=(10,6))

sns.histplot(df['Total Revenue'],bins=20,kde=True,color="skyblue")

plt.title("Distribution of Total Revenue")

plt.xlabel("Total Revenue")

plt.ylabel("Frequency")

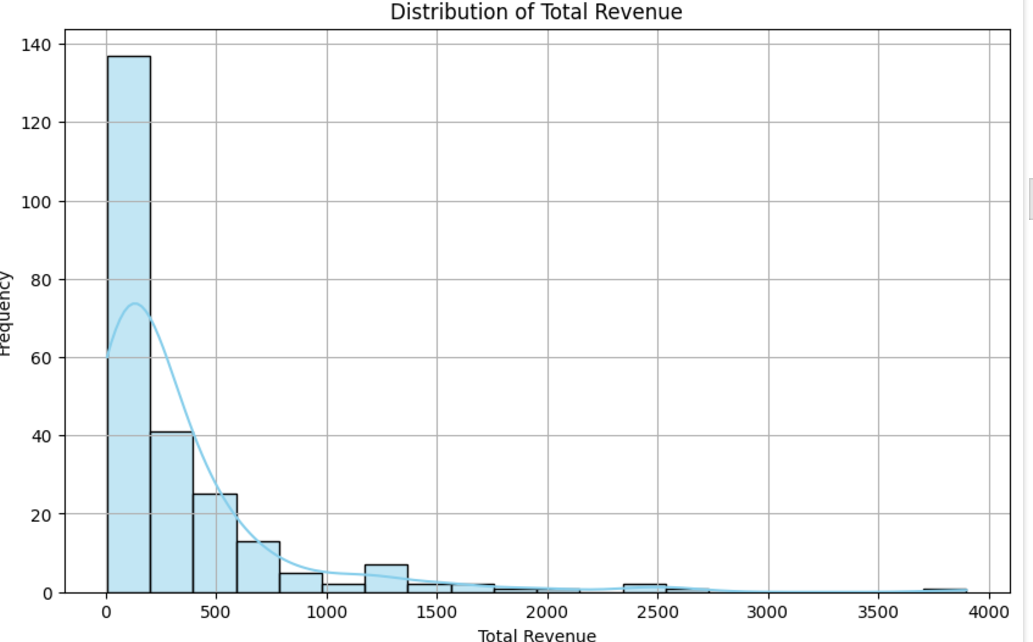
plt.grid(True)

plt.show()

**Explanation:**

This code visualizes the distribution of total revenue values, showing how many sales fall into different revenue ranges. By observing the shape of the histogram and the KDE curve, you can gain insights into the typical revenue generated from online sales, identify potential outliers, and understand the overall revenue distribution.

**Output:**

****

**Code:**

#box plot of Unit price

plt.figure(figsize=(8,6))

sns.boxplot(x=df['Unit Price'],color='lightgreen')

plt.title("Box plot of unit price")

plt.xlabel("Unit price")

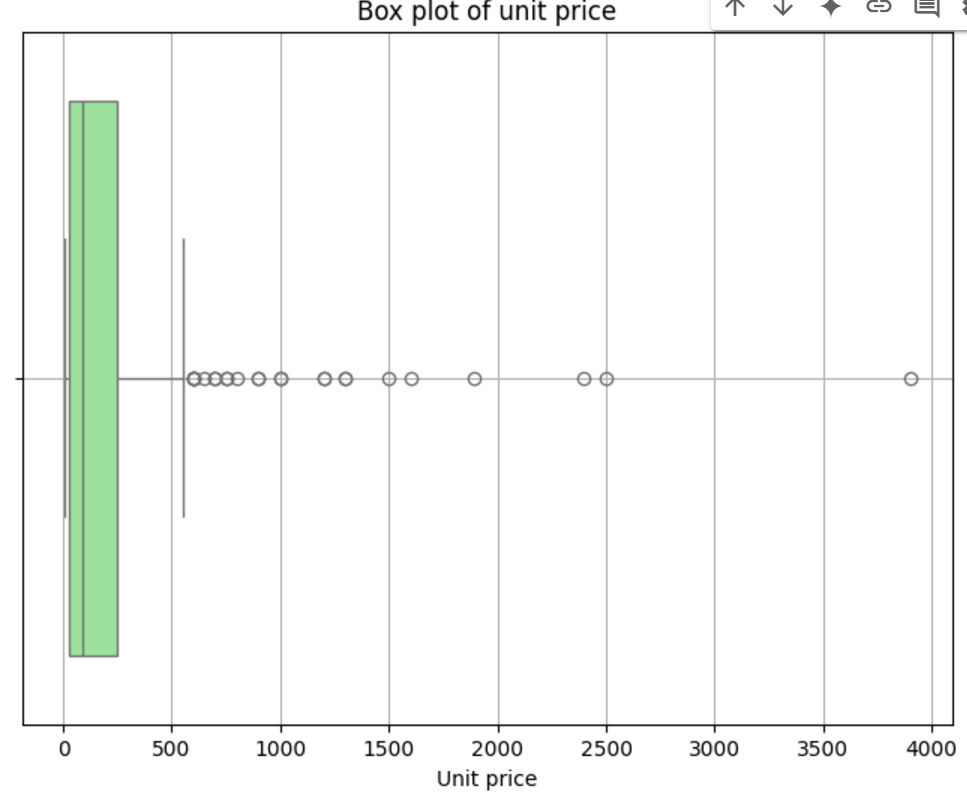
plt.grid(True)

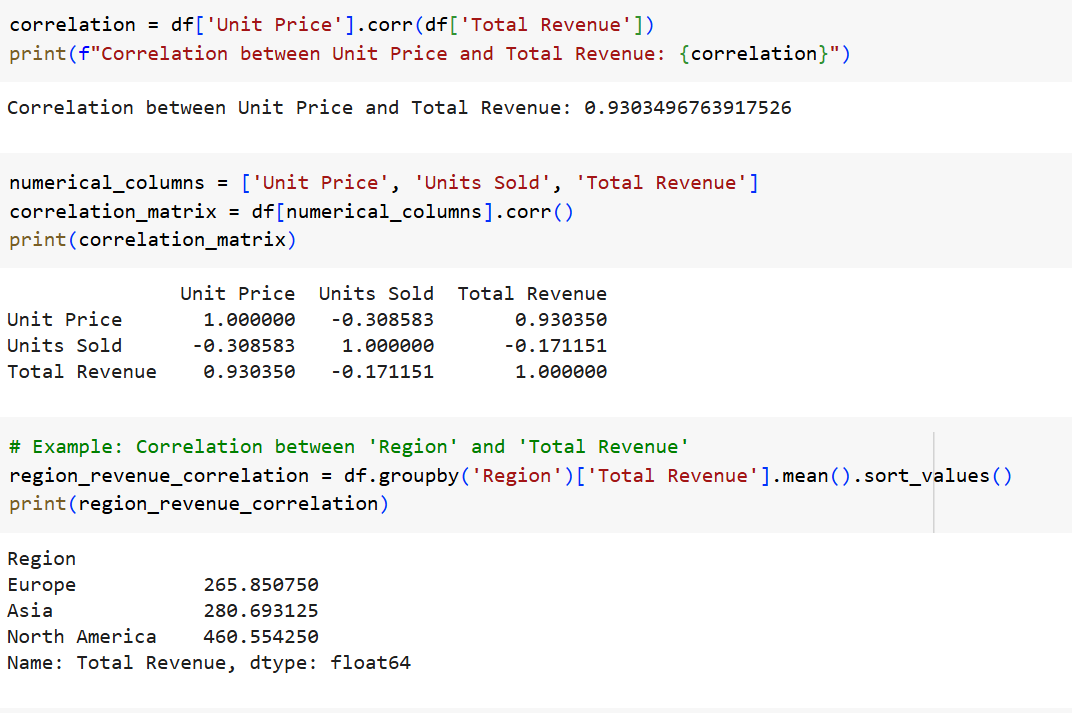
plt.show()

**Explanation:**

This code visualizes the distribution of unit prices, providing insights into their typical range, central tendency, and the presence of any outliers. By observing the box plot, you can quickly understand the spread and variability of unit prices within the online sales data.

**Output:**

****

****

**Code:**

import seaborn as sns

import matplotlib.pyplot as plt

correlation\_matrix = df[numerical\_columns].corr()

plt.figure(figsize=(10, 8))

sns.heatmap(correlation\_matrix, annot=True, cmap='coolwarm', fmt=".2f")

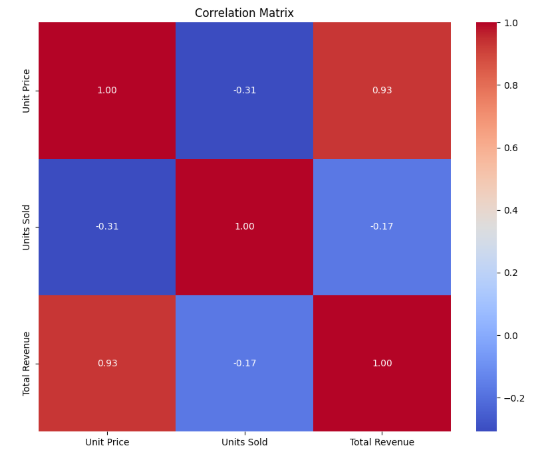
plt.title("Correlation Matrix")

plt.show()

**Explanation:**

This code visualizes the relationships between all numerical columns in the dataset using a heatmap. By observing the color patterns and the correlation values, you can identify which variables are strongly correlated (either positively or negatively) and which have weak or no correlation.

**Output:**

****

**Code:**

import matplotlib.pyplot as plt

import seaborn as sns

plt.figure(figsize=(12, 6))

sns.countplot(x='Product Category', data=df, order=df['Product Category'].value\_counts().index)

plt.title('Sales by Product Category')

plt.xlabel('Product Category')

plt.ylabel('Number of Sales')

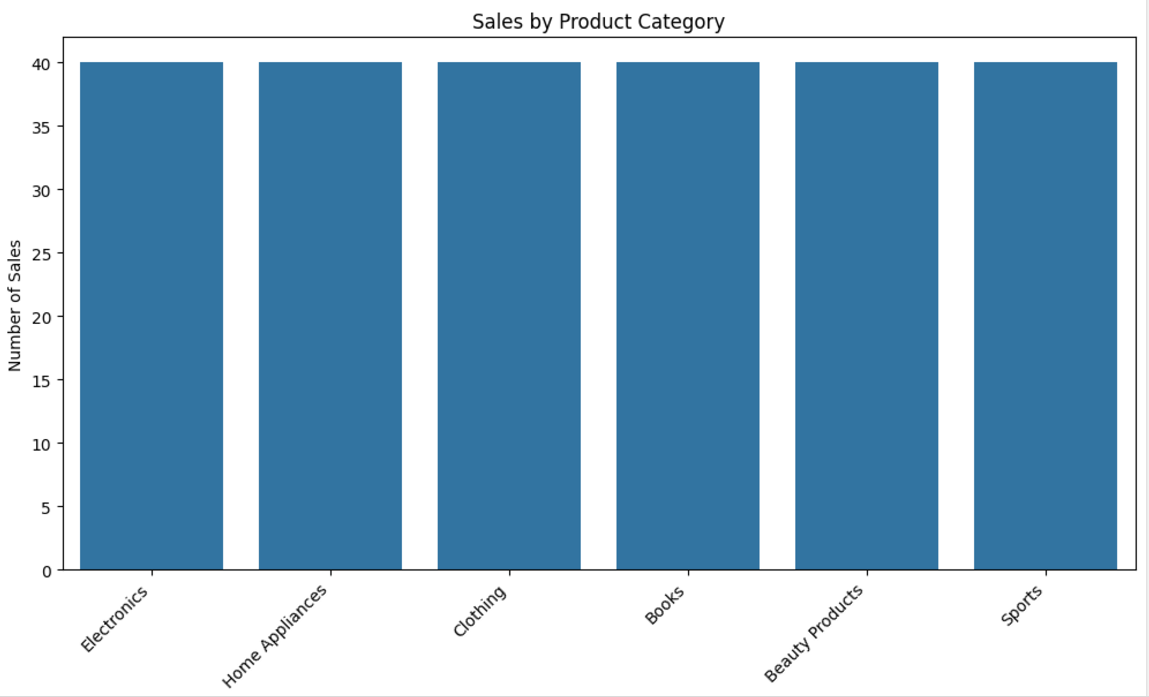
plt.xticks(rotation=45, ha='right')  # Rotate x-axis labels for readability

plt.show()

**Explanation:**

This code visualizes the sales distribution across different product categories. Each bar in the plot represents a product category, and its height corresponds to the number of sales for that category**.** This allows for easy comparison of sales performance across different product categories and helps identify the most popular and least popular categories.

**Output:**

****

**Code:**

import matplotlib.pyplot as plt

import seaborn as sns

plt.figure(figsize=(10, 6))

sns.boxplot(x='Payment Method', y='Unit Price', data=df)

plt.title('Unit Price Distribution by Payment Type')

plt.xlabel('Payment Method')

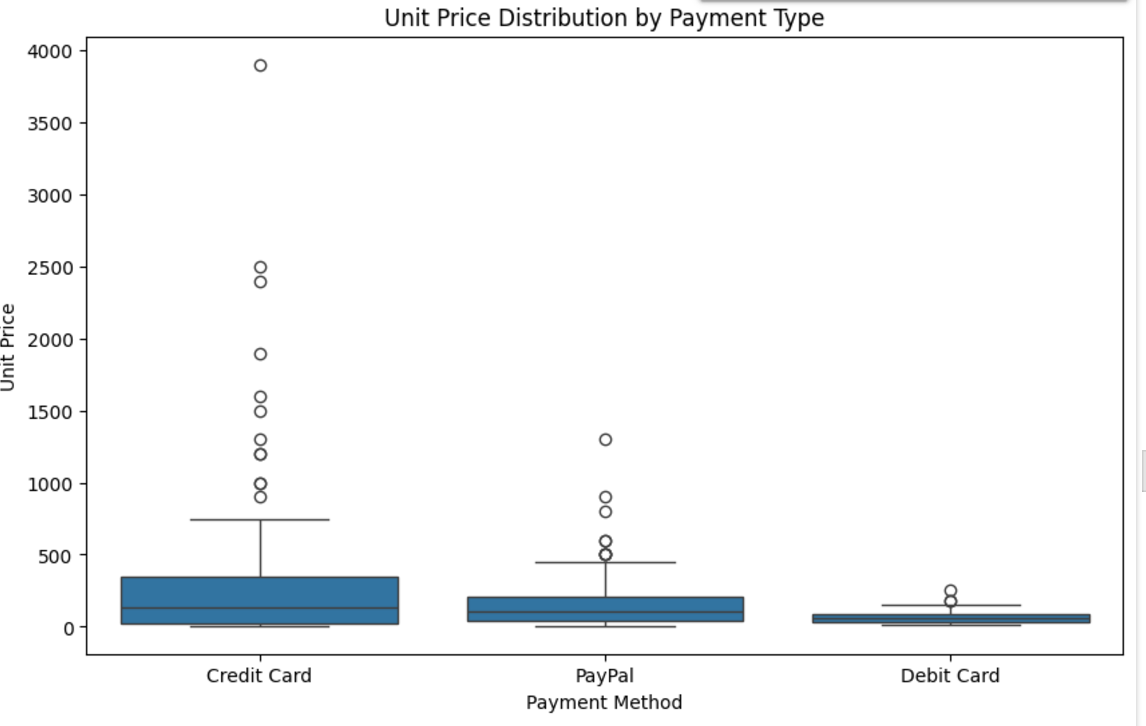
plt.ylabel('Unit Price')

plt.show()

**Explanation:**

This code visualizes how unit prices vary depending on the payment method used. By comparing the box plots for different payment methods, you can gain insights into potential pricing patterns or differences associated with each method. This visualization is useful for understanding customer behavior, pricing strategies, and potential relationships between payment methods and product prices.

**Output:**

****

**Code:**

import matplotlib.pyplot as plt

import pandas as pd

df['Date'] = pd.to\_datetime(df['Date'])

df['Month'] = df['Date'].dt.month

monthly\_revenue = df.groupby('Month')['Total Revenue'].sum()

plt.figure(figsize=(10, 6))

plt.plot(monthly\_revenue.index, monthly\_revenue.values, marker='o')

plt.title('Total Revenue Over Time')

plt.xlabel('Month')

plt.ylabel('Total Revenue')

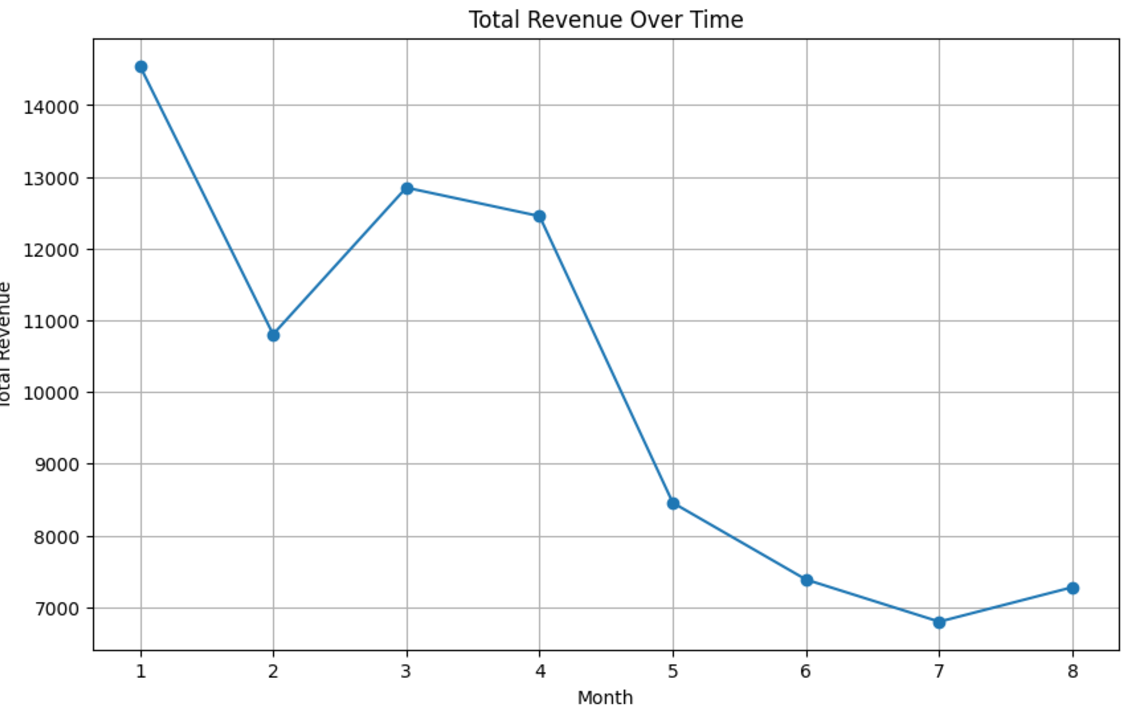
plt.grid(True)

plt.show()

**Explanation:**

This code visualizes the monthly revenue trend over time. By observing the line's slope and fluctuations, you can identify periods of high or low revenue, seasonal patterns, and overall revenue growth or decline.

**Output:**

****

**Code:**

import plotly.express as px

category\_sales = df.groupby('Product Category')['Total Revenue'].sum().reset\_index()

fig = px.treemap(category\_sales, path=['Product Category'], values='Total Revenue',

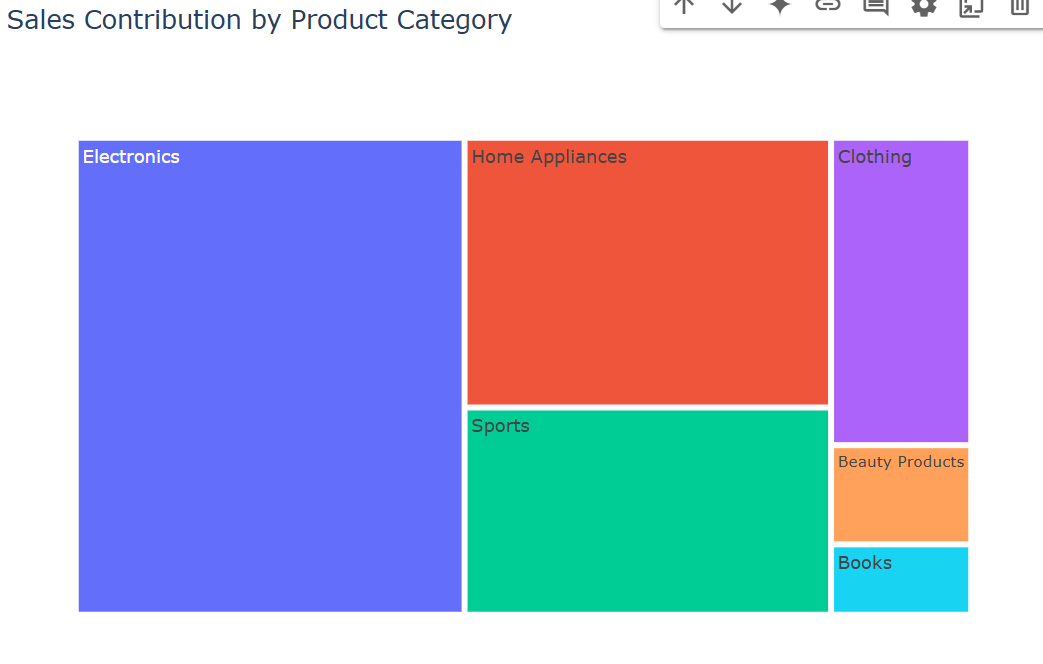
                  title='Sales Contribution by Product Category')

fig.show()

**Explanation:**

This code visualizes the contribution of different product categories to the total sales revenue using a treemap. By observing the size of the rectangles, you can easily identify the categories that generate the most revenue and their relative proportions.

**Output:**

****

**Code:**

import matplotlib.pyplot as plt

payment\_method\_counts = df['Payment Method'].value\_counts()

plt.figure(figsize=(8, 8))

plt.pie(payment\_method\_counts, labels=payment\_method\_counts.index, autopct='%1.1f%%', startangle=90)

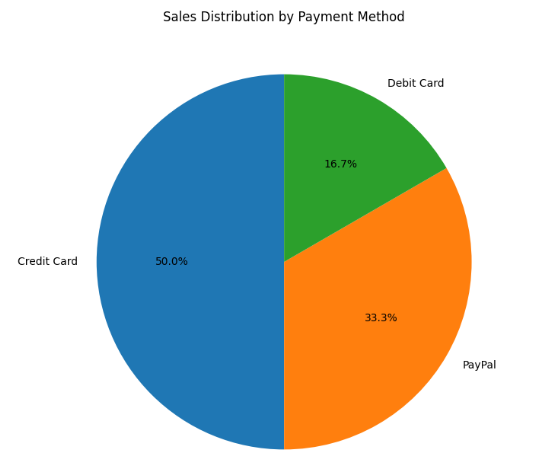
plt.title('Sales Distribution by Payment Method')

plt.show()

**Explanation:**

This code takes the sales data, calculates the total sales for each payment method, and then presents this information visually as a pie chart. Each slice of the pie represents a payment method, and its size corresponds to the proportion of total sales made using that method. This allows for easy comparison of payment method preferences among customers and helps identify the most and least popular payment options.

**Output:**

****

**Code:**

import matplotlib.pyplot as plt

import seaborn as sns

plt.figure(figsize=(12, 6))

sns.countplot(x='Product Category', hue='Payment Method', data=df)

plt.title('Sales by Product Category and Payment Method')

plt.xlabel('Product Category')

plt.ylabel('Number of Sales')

plt.xticks(rotation=45, ha='right')

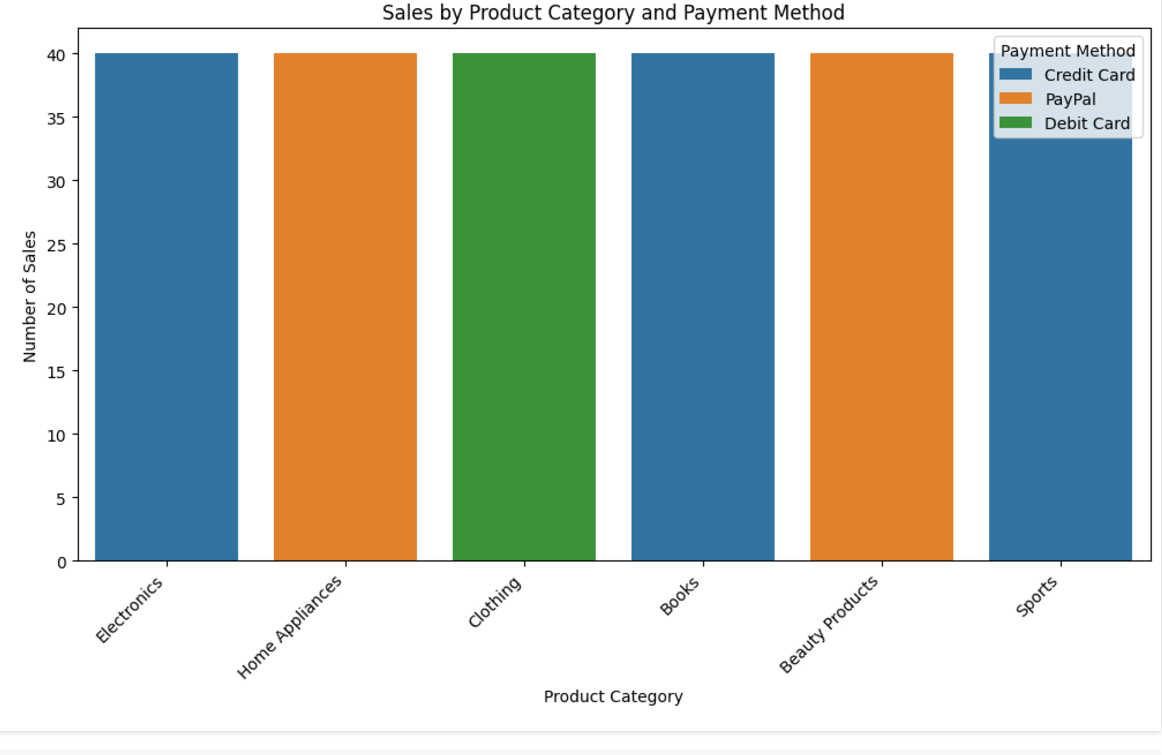
plt.legend(title='Payment Method')

plt.show()

**Explanation:**

This code visualizes the sales distribution across different product categories, further broken down by payment method. By comparing the heights of the bars within each category and across different payment methods, you can gain insights into which payment methods are most popular for each product category.

**Output:**

****

**Code:**

import matplotlib.pyplot as plt

import seaborn as sns

plt.figure(figsize=(10, 6))

sns.scatterplot(x='Unit Price', y='Units Sold', hue='Region', data=df)

plt.title('Unit Price vs. Quantity')

plt.xlabel('Unit Price')

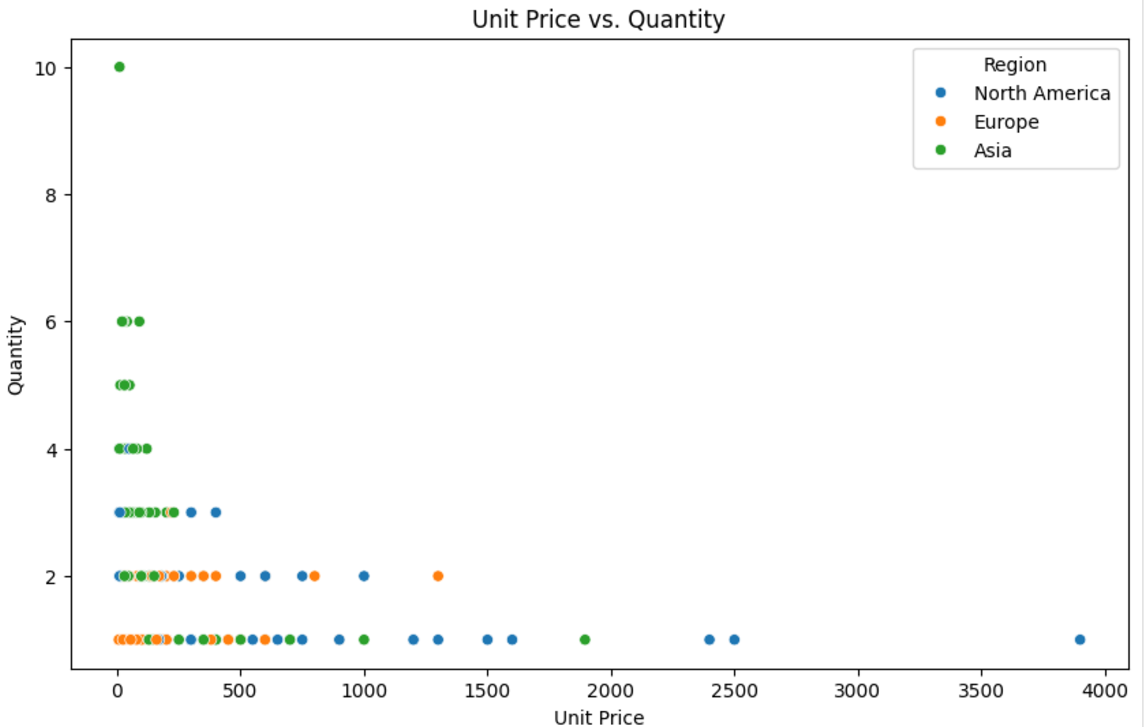
plt.ylabel('Quantity')

plt.show()

**Explanation:**

Thiscode visualizes how unit price and quantity sold are related, with regional variations highlighted by color. By observing the distribution of data points and their colors, you can gain insights into potential correlations, regional differences in pricing or sales volume, and overall sales patterns.

**Output:**

****

**COLAB\_LINK- https://colab.research.google.com/drive/1DRQ9SK6bPJiQiXvsHDuFAhGRSUbrQA11?usp=sharing**