AMAZON SALES PROJE

NAME : DEVAGUPTAPU BHANU SRI

COURSE : DATA ANALYSIS

DURATION : 1 month

Information regarding the dataset:

SOURCE CODE:

```
#1STEP.First know the problem statements.

#2STEP.Collect the data from the respective sources.

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import random

import seaborn as sns

df = pd.read_csv('Amazon Sales data intern.csv')

print(df)

#Find the bottom columns and rows in dataset

df.head(10)

#Find the type of the data in dataset

type(df)

df.info()

#Shape of data

df.shape

#To find the dimensions of the data set

#Find the bottom columns and rows in dataset

df.tail(10)

#3STEP.Clean and Prepare the data

###To find null values in the dataset

f=df.isna().sum(axis=0)

f
```

OUTPUT:

```
Region            0
Country           0
Item Type         0
```

```
Sales Channel     0
Order Priority    0
Order Date        0
Order ID          0
Ship Date         0
Units Sold        0
Unit Price        0
Unit Cost         0
Total Revenue     0
Total Cost        0
Total Profit      0
```

```
#4STEP.Analyzing of the given dataset.

#######################

#We can find the count of unique values of dataset

df.nunique()

#So, we can find the different unique values of the columns in the dataset.

#We can see that there are 7 different regions with 76 different countries.

#There are 12 different items are sold in 2 channels that is ONLINE and OFFLINE.

#There are 4 types of order priorities

#to find the unique values in the dataset

df['Region'].unique()

df['Country'].unique()

df['Item Type'].unique()

df['Sales Channel'].unique()

df['Order Priority'].unique()

#As we can see that there are 4 order priorities like H, C, L and M.

#H means High Priority

#C means Critical Priority

#L means Low priority

#M means Medium priority
```

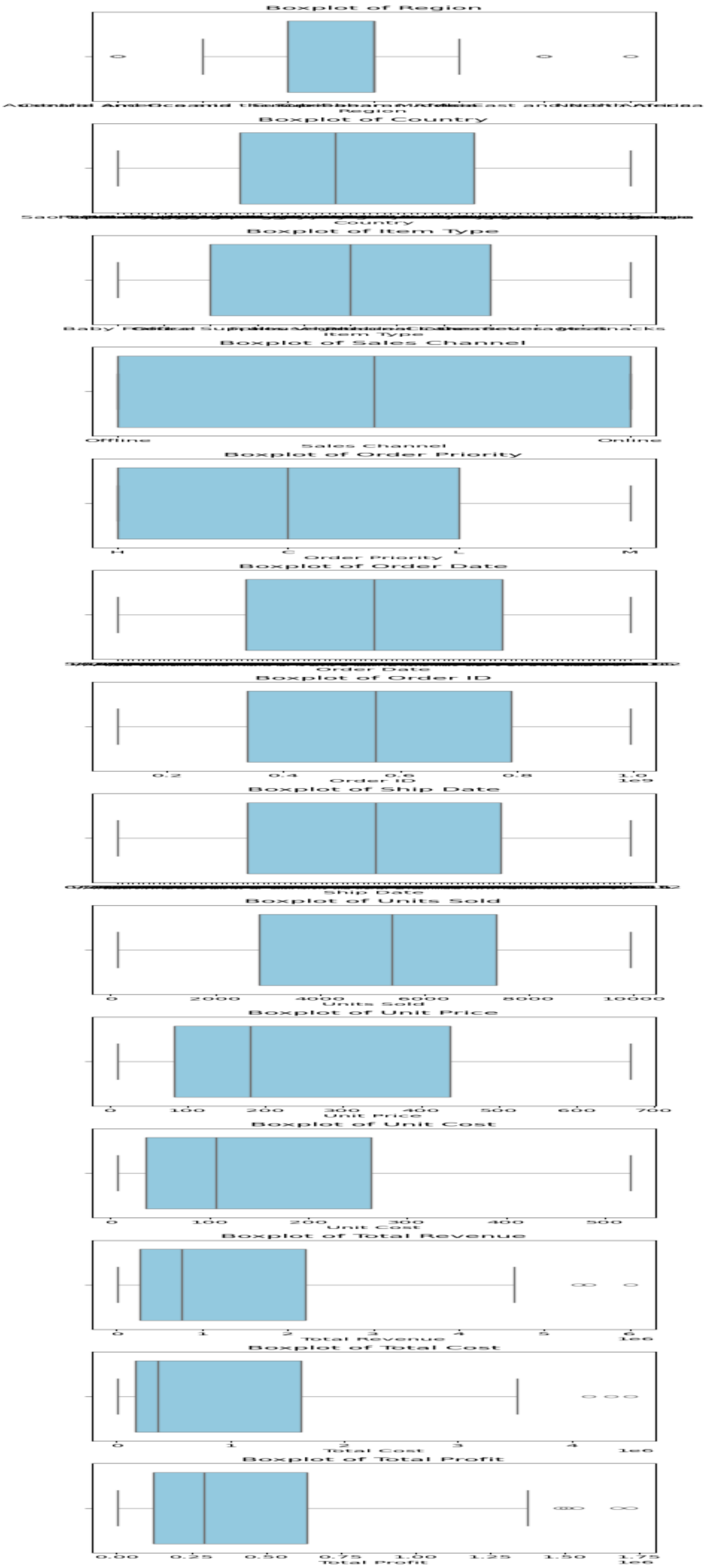
DESCRIBE:

```
#To know some statistical details regarding the dataset

df.describe()
```

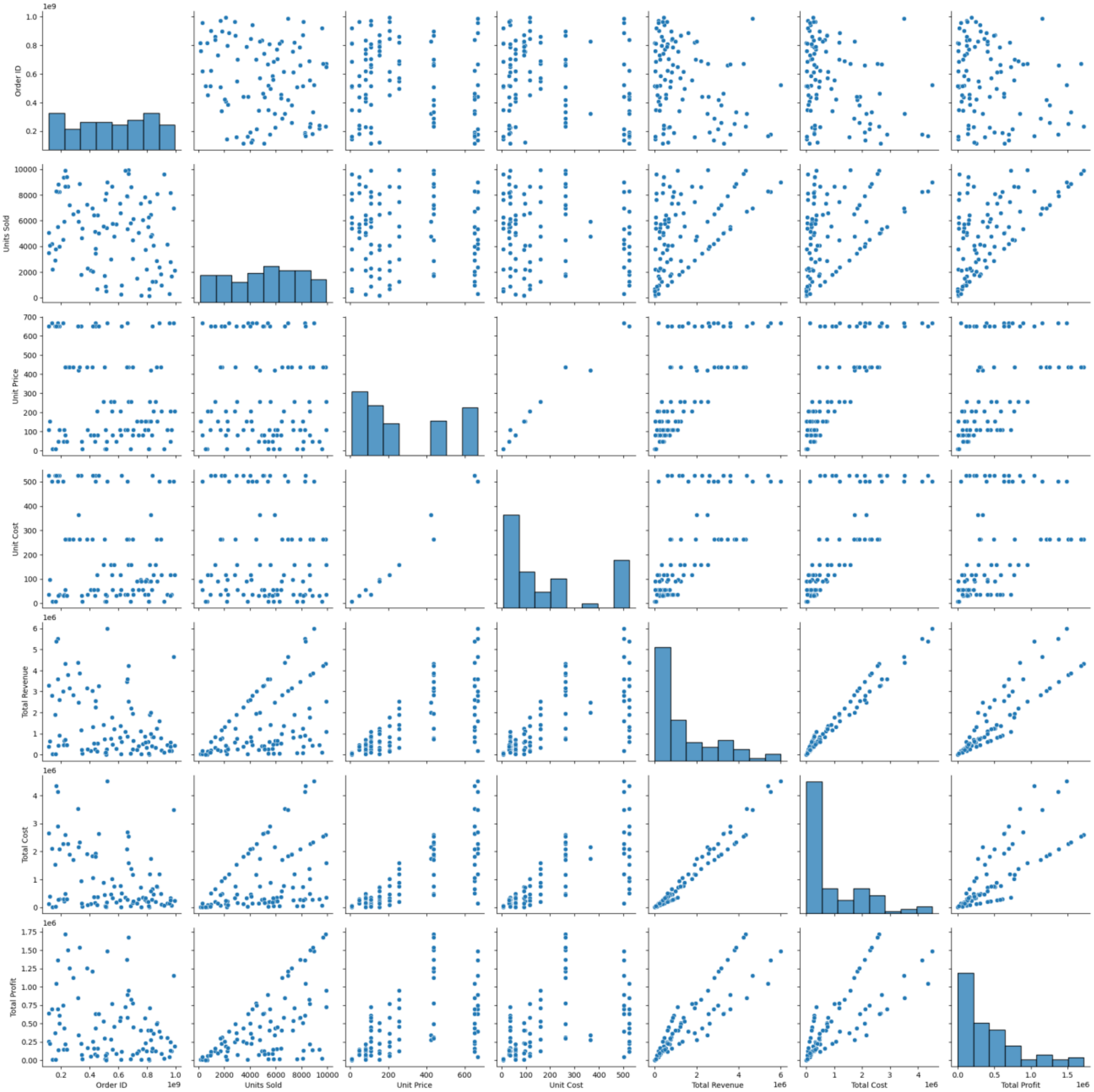
OUTPUT:

Order ID	Units Sold	Unit Price	Unit Cost	Total Revenue	Total Cost	Total Profit	
count	1.000000e+02	100.000000	100.000000	100.000000	1.000000e+02	1.000000e+02	1.000000e+02
mean	5.550204e+08	5128.710000	276.761300	191.048000	1.373488e+06	9.318057e+05	4.416820e+05
std	2.606153e+08	2794.484562	235.592241	188.208181	1.460029e+06	1.083938e+06	4.385379e+05
min	1.146066e+08	124.000000	9.330000	6.920000	4.870260e+03	3.612240e+03	1.258020e+03
25%	3.389225e+08	2836.250000	81.730000	35.840000	2.687212e+05	1.688680e+05	1.214436e+05
50%	5.577086e+08	5382.500000	179.880000	107.275000	7.523144e+05	3.635664e+05	2.907680e+05
75%	7.907551e+08	7369.000000	437.200000	263.330000	2.212045e+06	1.613870e+06	6.358288e+05
max	9.940222e+08	9925.000000	668.270000	524.960000	5.997055e+06	4.509794e+06	1.719922e+06

```
#To find if there are any outliers in the given data
#To show the outliers in graph
plt.figure(figsize=(5, 4 * len(df.columns)))

# Iterate through each column and plot boxplot
for i, column in enumerate(df.columns):
   plt.subplot(len(df.columns), 1, i+1)
   sns.boxplot(x=df[column], color='skyblue')
   plt.title(f'Boxplot of {column}')

plt.tight_layout()
plt.show()
#By using these boxplots we can observe that there are no much outliers in the given data
```

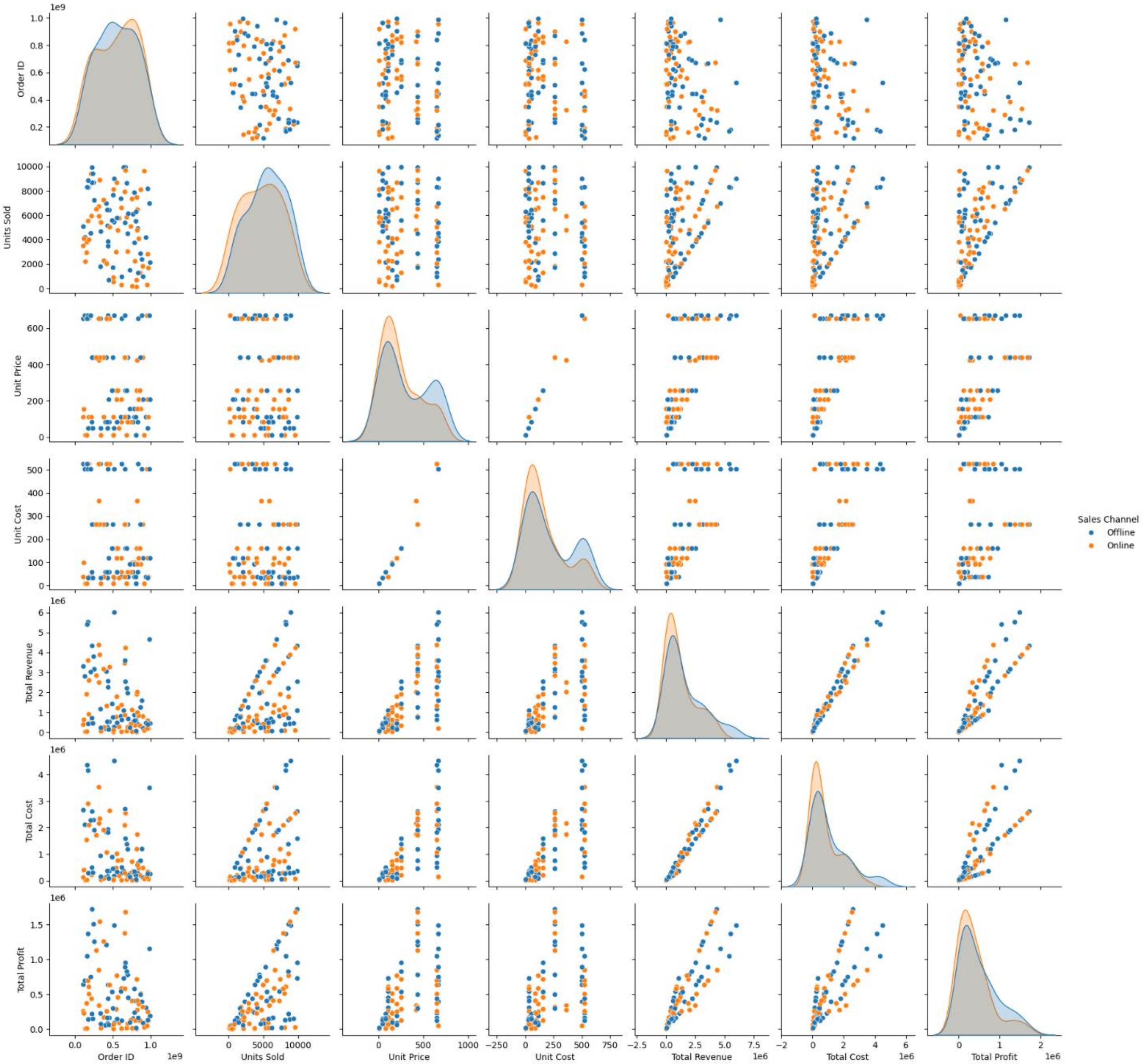

Boxplot of Region
Region
Boxplot of Country
Country
Boxplot of Item Type
Baby Food
Snacks
Item Type
Boxplot of Sales Channel
Offline
Online
Sales Channel
Boxplot of Order Priority
H
C
L
M
Order Priority
Boxplot of Order Date
Order Date
Boxplot of Order ID
0.2
0.4
0.6
0.8
1.0
1e9
Order ID
Boxplot of Ship Date
Ship Date
Boxplot of Units Sold
0
2000
4000
6000
8000
10000
Units Sold
Boxplot of Unit Price
0
100
200
300
400
500
600
700
Unit Price
Boxplot of Unit Cost
0
100
200
300
400
500
Unit Cost
Boxplot of Total Revenue
0
1
2
3
4
5
6
1e6
Total Revenue
Boxplot of Total Cost
0
1
2
3
4
1e6
Total Cost
Boxplot of Total Profit
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
1e6
Total Profit

```
import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

from ipywidgets import interactive


# To know the correlation and distribution between the variables

def size_widget(height=2.5, aspect=1):

   sns.pairplot(df, height=height, aspect=aspect)
```



```
interactive(size_widget, height=(1, 3.5, 0.5), aspect=(0.5, 2, 0.25))
```

```
# To know the correlation and distribution between the variables

sns.pairplot(df,hue = 'Sales Channel')

# Convert 'Order Date' column to datetime

df['Order Date'] = pd.to_datetime(df['Order Date'])
```

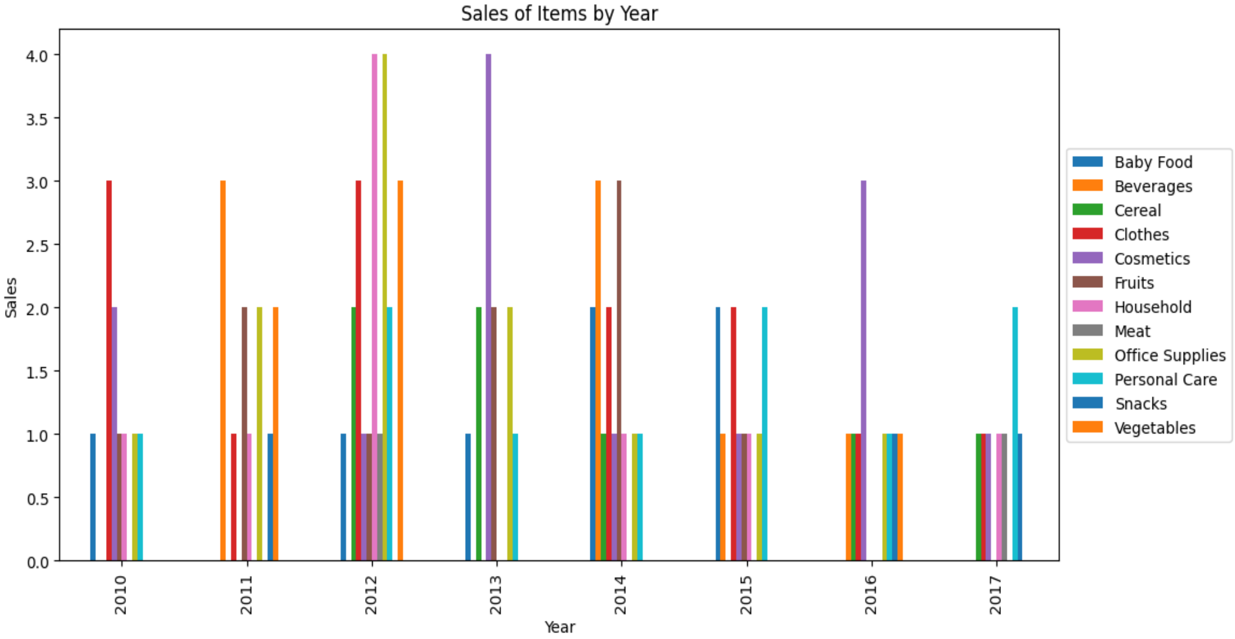


```
# Extract years and months from 'Order Date' column
```

```
df['Year'] = df['Order Date'].dt.year
df['Month'] = df['Order Date'].dt.month

# Display the DataFrame with years and months
print(df[['Order Date', 'Year', 'Month']])
```

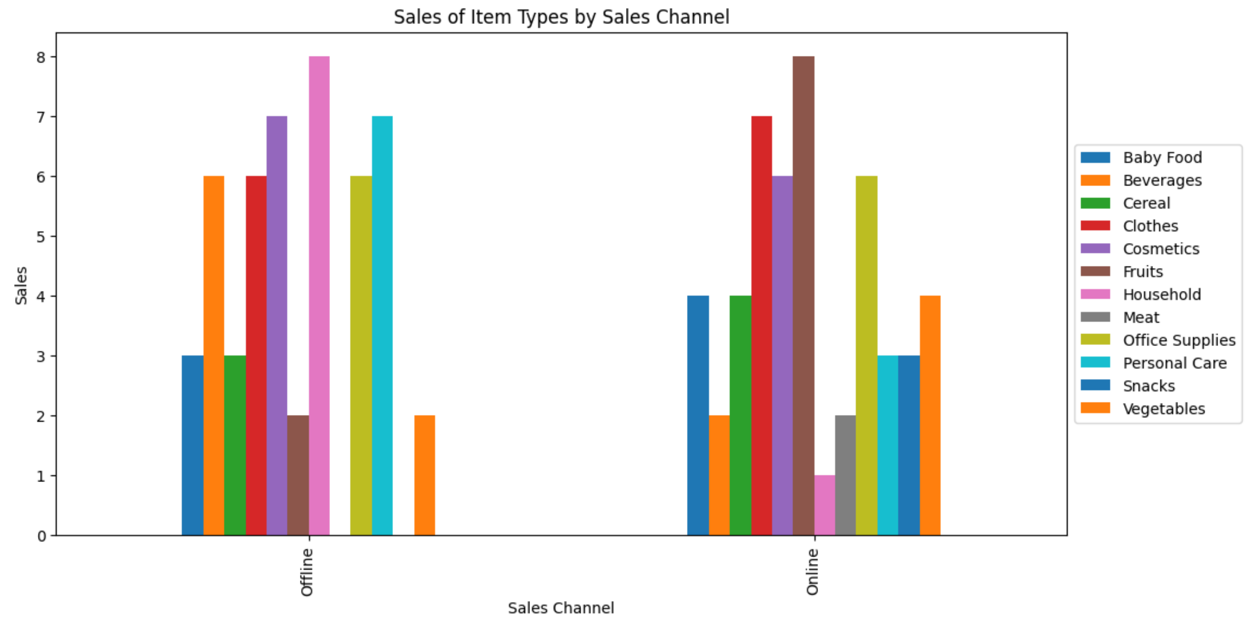
```
#Convert 'Order Date' column to datetime
df['Order Date'] = pd.to_datetime(df['Order Date'])

# Sort the dataset by the 'Order Date' column
df_sorted = df.sort_values(by='Order Date')

# Display the sorted DataFrame
print(df_sorted)
```

```
Convert 'Order Date' column to datetime
df['Order Date'] = pd.to_datetime(df['Order Date'])

# Extract year from 'Order Date' column
df['Year'] = df['Order Date'].dt.year

# Group by 'Year' and 'Item', then sum the sales
sales_by_year_item = df.groupby(['Year', 'Item Type']).size().unstack(fill_value=0)

# Plot the sales for each item across different years
sales_by_year_item.plot(kind='bar', figsize=(12, 6))

# Set plot labels and title
plt.xlabel('Year')
plt.ylabel('Sales')
plt.title('Sales of Items by Year')


# Show legend outside of the plot

plt.legend(loc='center left', bbox_to_anchor=(1, 0.5))


# Show the plot

plt.show()

#In this graph we can observe the Sales of different items according to years

# Group by 'Sales Channel' and 'Item Type', then sum the sales
```



```
sales_by_channel_item = df.groupby(['Sales Channel', 'Item Type']).size().unstack(fill_value=0)


# Plot the sales for each item type by sales channel

sales_by_channel_item.plot(kind='bar', figsize=(12, 6))


# Set plot labels and title

plt.xlabel('Sales Channel')

plt.ylabel('Sales')

plt.title('Sales of Item Types by Sales Channel')
```

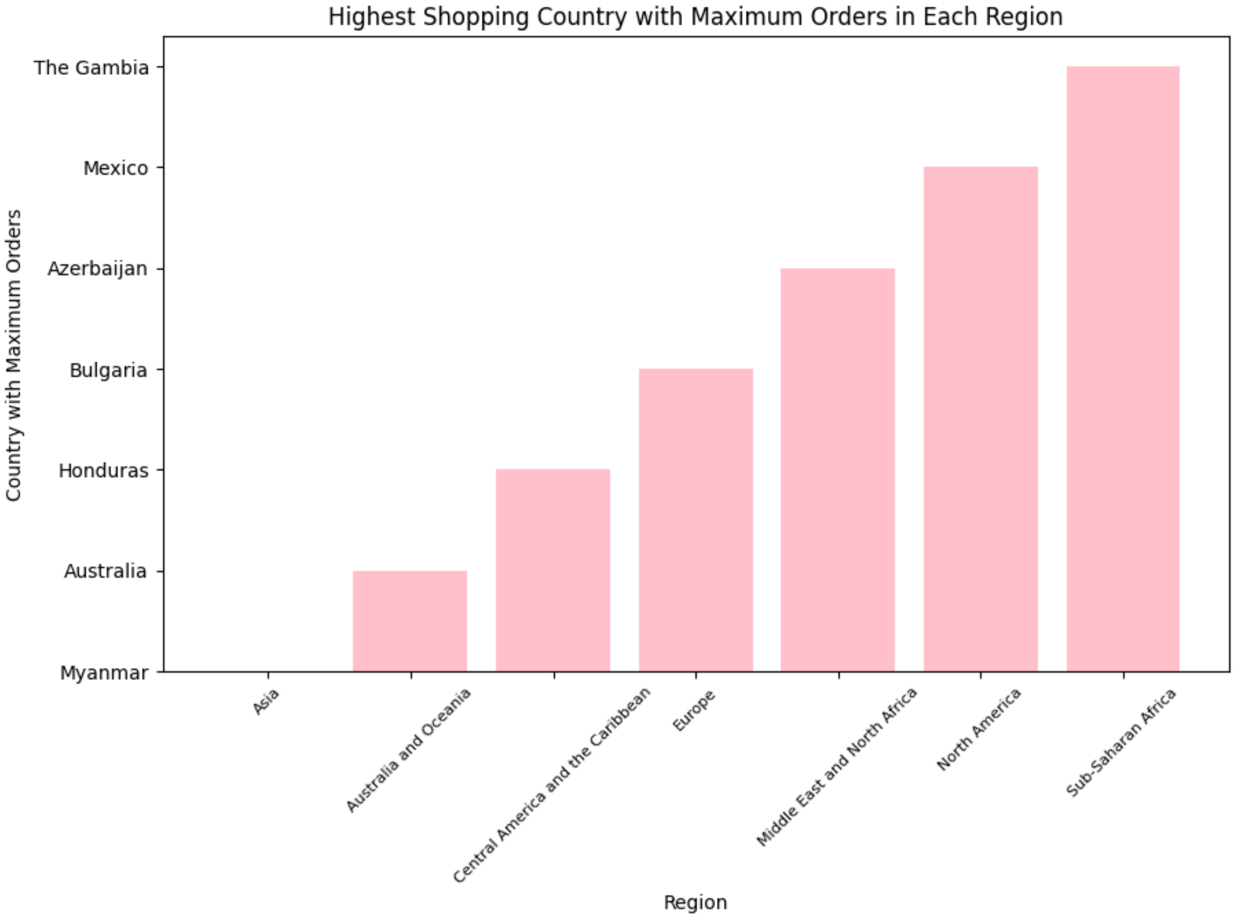
```
# Show legend outside of the plot

plt.legend(loc='center left', bbox_to_anchor=(1, 0.5))


# Show the plot

plt.show()

#From this graph we can observe that different item types are sold in 2 categories offline and Online
```



```
# Group by 'Region' and aggregate unique values of 'Country'

countries_by_region = df.groupby('Region')['Country'].unique()


# Print countries for each region

for region, countries in countries_by_region.items():

    print(f"Region: {region}")

    print("Countries:", ", ".join(countries))

    print()

#We can observe that there are different countries belongs to different regions.
```

```
# Group by 'Region' and 'Country', then sum the number of orders
```

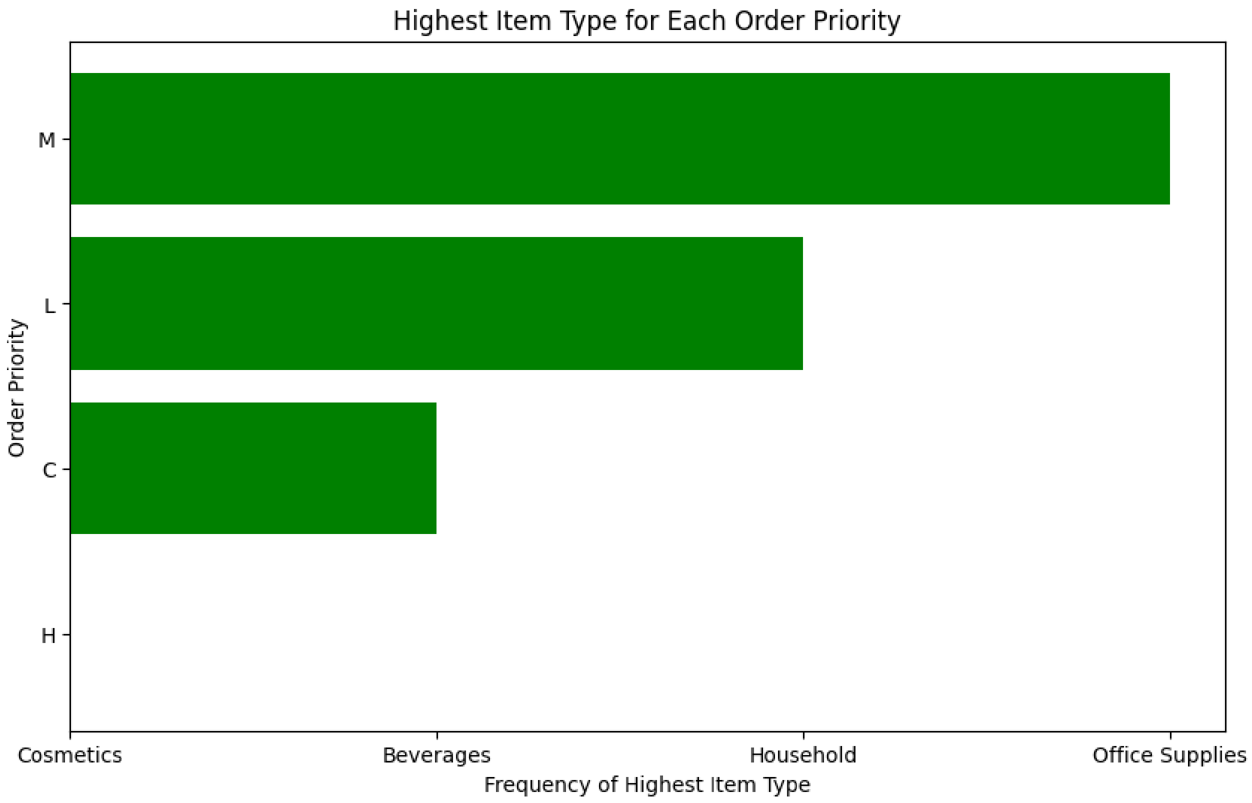
```
orders_by_region_country = df.groupby(['Region', 'Country']).size().unstack(fill_value=0)


# Find the country with the maximum ordering history in each region

highest_ordering_country = orders_by_region_country.idxmax(axis=1)


# Plot a bar graph showing the highest shopping country with the maximum ordering history for each region

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

plt.bar(highest_ordering_country.index, highest_ordering_country.values, color='pink')


# Set plot labels and title

plt.xlabel('Region')

plt.ylabel('Country with Maximum Orders')
```



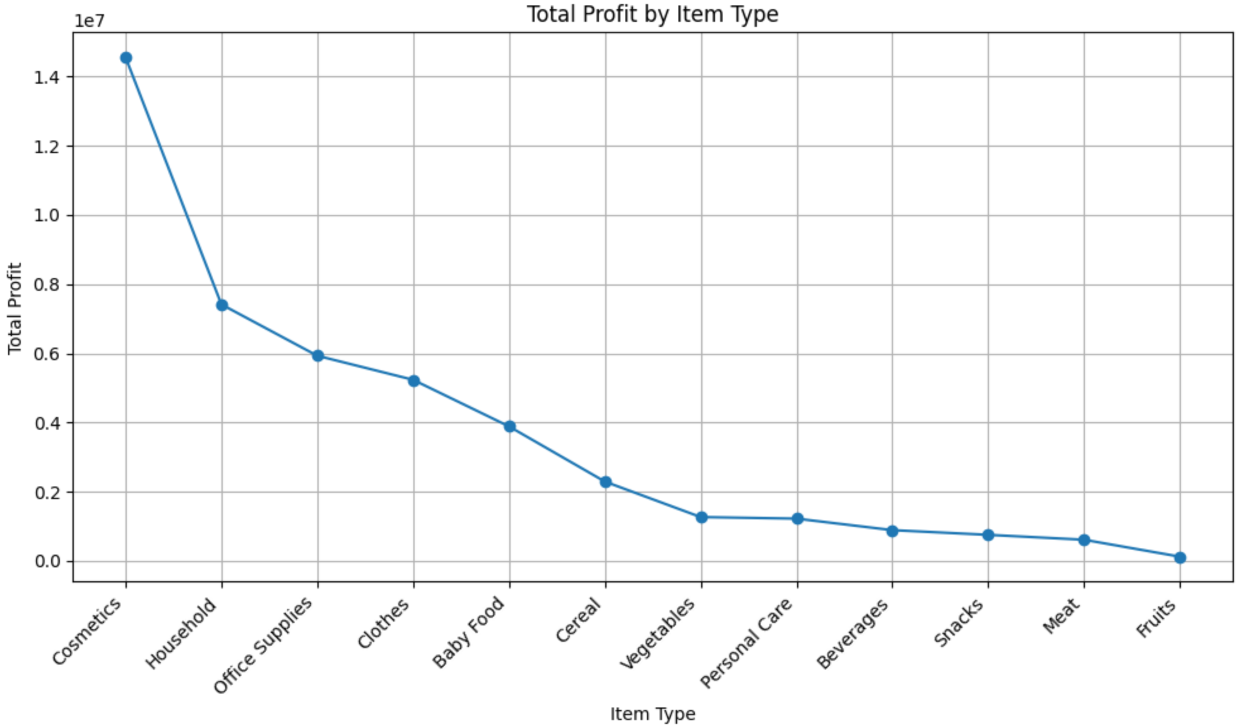
```
plt.title('Highest Shopping Country with Maximum Orders in Each Region')
# Decrease the size of x-axis labels
plt.xticks(rotation=45, fontsize=8)
```

```
# Create a cross-tabulation between 'Item Type' and 'Order Priority'
cross_tab = pd.crosstab(df['Item Type'], df['Order Priority'])

# Display the cross-tabulation
print(cross_tab)
```

```
# Find the item type with the highest frequency for each order priority
highest_item_types = {}
for priority in df['Order Priority'].unique():
    highest_item_types[priority] = df[df['Order Priority'] == priority]['Item Type'].value_counts().idxmax()

# Plot a bar graph showing the frequency of the highest item type for each order priority
plt.figure(figsize=(10, 6))
plt.barh(range(len(highest_item_types)), list(highest_item_types.values()), color='green')

# Set the y-axis ticks and labels to be the order priorities
plt.yticks(range(len(highest_item_types)), list(highest_item_types.keys()))

# Set plot labels and title
plt.ylabel('Order Priority')
plt.xlabel('Frequency of Highest Item Type')
plt.title('Highest Item Type for Each Order Priority')

# Show the plot
plt.show()
# Assuming your dataset is stored in a variable named 'df'
```

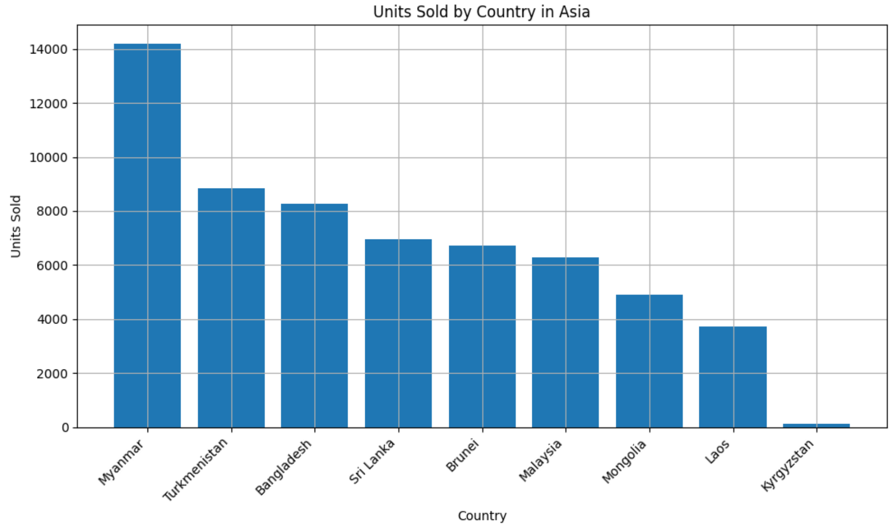


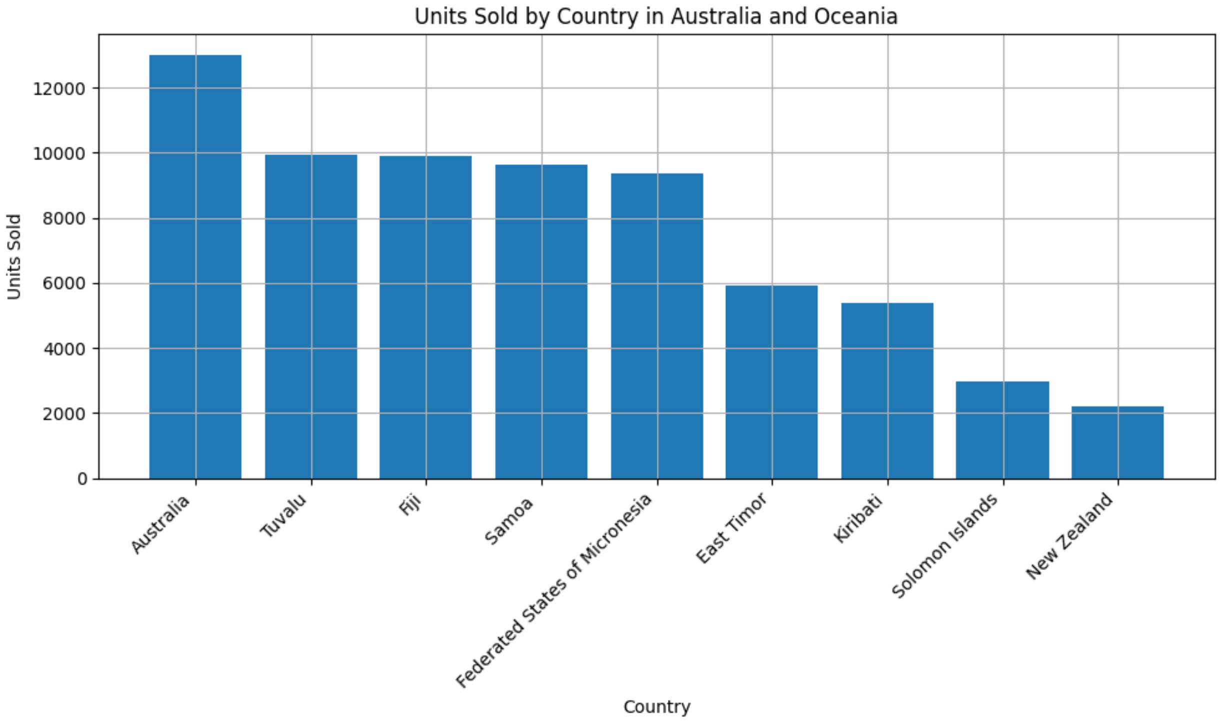
```
# Grouping by 'Item Type' and calculating total profit
item_profit = df.groupby('Item Type')['Total Profit'].sum().reset_index()

# Sorting by total profit in descending order
item_profit = item_profit.sort_values(by='Total Profit', ascending=False)

# Plotting the line graph
plt.figure(figsize=(10, 6))
plt.plot(item_profit['Item Type'], item_profit['Total Profit'], marker='o', linestyle='-')
plt.title('Total Profit by Item Type')
plt.xlabel('Item Type')
plt.ylabel('Total Profit')
plt.xticks(rotation=45, ha='right')
plt.grid(True)
plt.tight_layout()

plt.show()
```

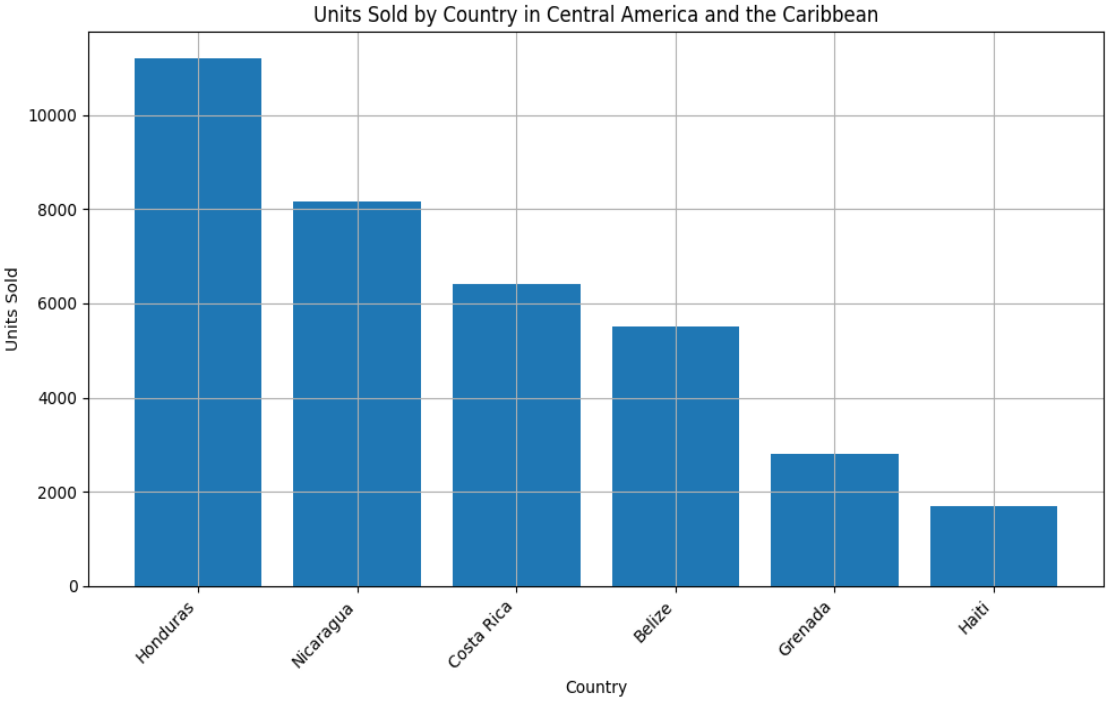


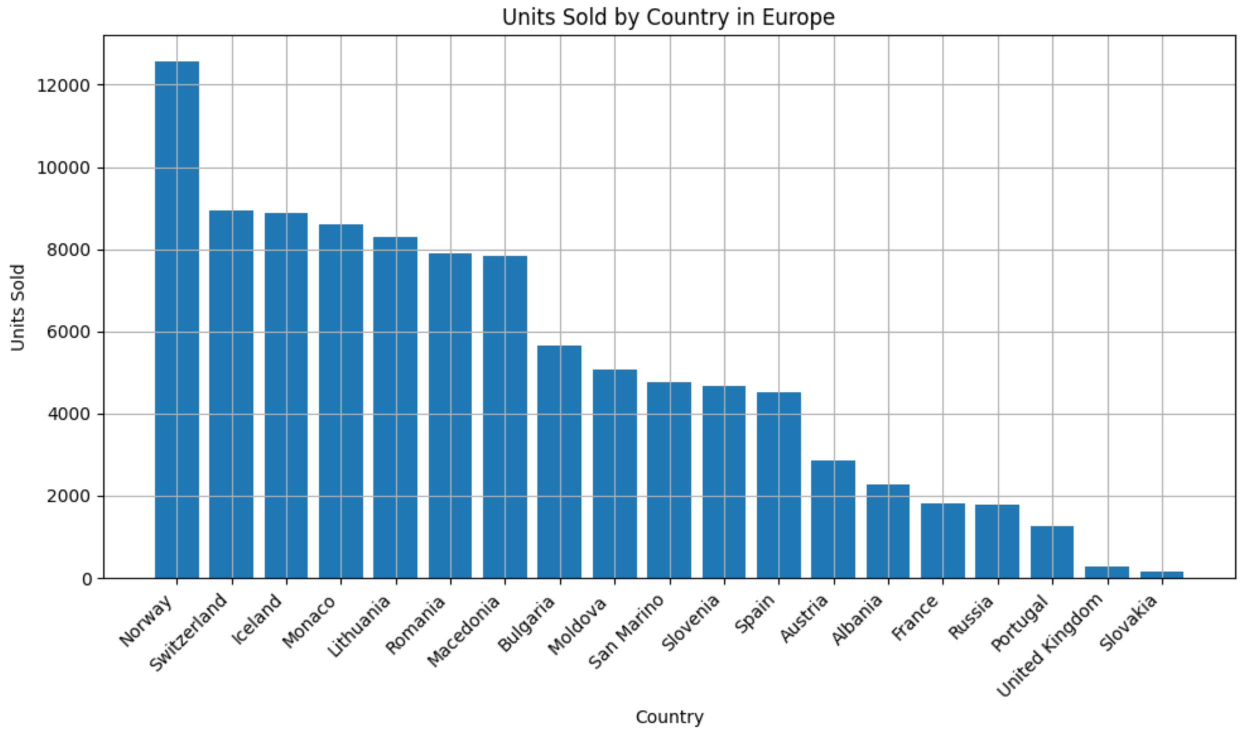
```
# Grouping by 'Region', 'Country', and 'Item Type', and calculating total units sold

country_item_sales = df.groupby(['Region', 'Country', 'Item Type'])['Units Sold'].sum().reset_index()


# Sorting by total units sold in descending order within each region and country

country_item_sales_sorted = country_item_sales.groupby(['Region', 'Country']).apply(lambda x: x.sort_values(by='Units Sold', ascending=False))


# Resetting index after sorting

country_item_sales_sorted.reset_index(drop=True, inplace=True)


# Displaying the top item types in each country within each region

for (region, country), data in country_item_sales_sorted.groupby(['Region', 'Country']):

    print(f"Top item types in {country} ({region}):")

    print(data.head())  # Adjust the number in head() to show more item types if needed

    print()
```

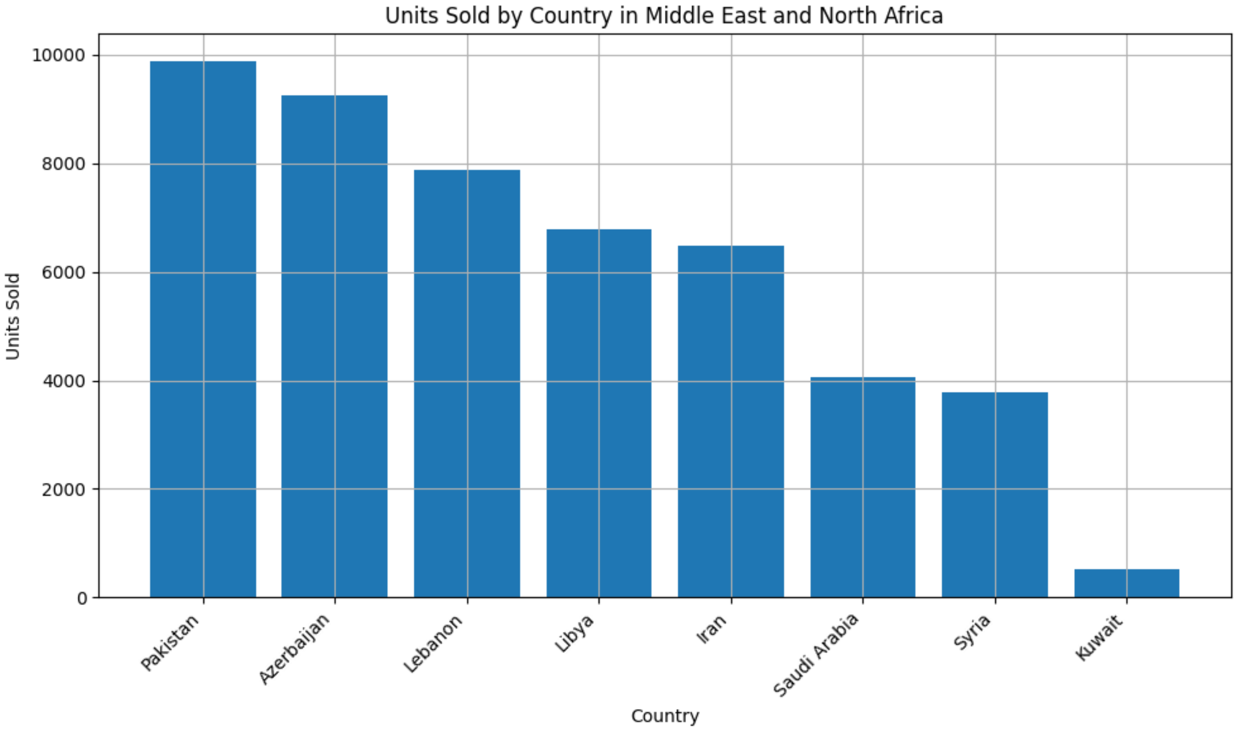
```
# Grouping by 'Region', 'Country', and calculating total units sold
region_country_sales = df.groupby(['Region', 'Country'])['Units Sold'].sum().reset_index()

# Sorting by total units sold in descending order within each region
region_country_sales_sorted = region_country_sales.sort_values(by=['Region', 'Units Sold'],
ascending=[True, False])

# Plotting the data for each region
regions = region_country_sales_sorted['Region'].unique()
for region in regions:
    data = region_country_sales_sorted[region_country_sales_sorted['Region'] == region]
    plt.figure(figsize=(10, 6))
    plt.bar(data['Country'], data['Units Sold'])
    plt.title(f'Units Sold by Country in {region}')
    plt.xlabel('Country')
    plt.ylabel('Units Sold')
    plt.xticks(rotation=45, ha='right')
    plt.grid(True)
    plt.tight_layout()
```

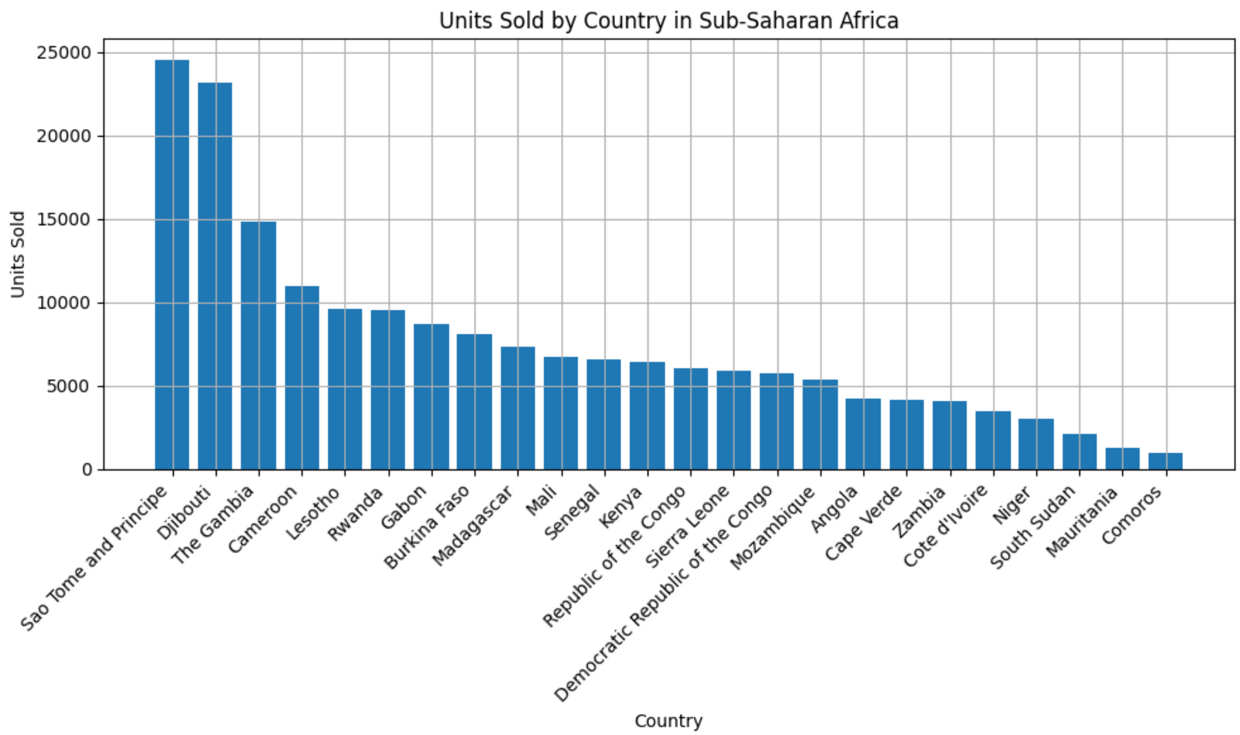



Units Sold by Country in Australia and Oceania
Units Sold
0
2000
4000
6000
8000
10000
12000
Australia
Tuvalu
Fiji
Samoa
Federated States of Micronesia
East Timor
Kiribati
Solomon Islands
New Zealand
Country

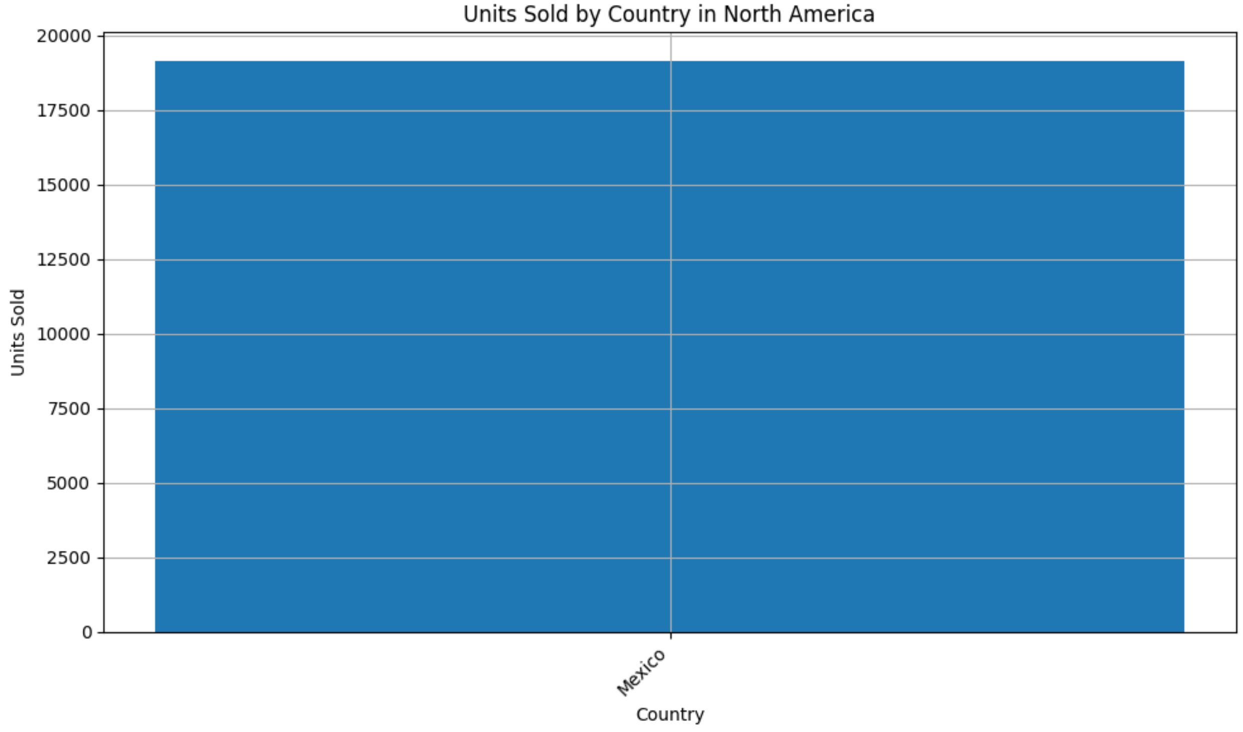

Units Sold by Country in Central America and the Caribbean
Units Sold
0
2000
4000
6000
8000
10000
Honduras
Nicaragua
Costa Rica
Belize
Grenada
Haiti
Country


Units Sold by Country in Europe
Units Sold
12000
10000
8000
6000
4000
2000
0
Norway
Switzerland
Iceland
Monaco
Lithuania
Romania
Macedonia
Bulgaria
Moldova
San Marino
Slovenia
Spain
Austria
Albania
France
Russia
Portugal
United Kingdom
Slovakia
Country


Units Sold by Country in Middle East and North Africa
Units Sold
10000
8000
6000
4000
2000
0
Pakistan
Azerbaijan
Lebanon
Libya
Iran
Saudi Arabia
Syria
Kuwait
Country


Units Sold by Country in Sub-Saharan Africa
Units Sold
25000
20000
15000
10000
5000
0
Sao Tome and Principe
Djibouti
The Gambia
Cameroon
Lesotho
Rwanda
Gabon
Burkina Faso
Madagascar
Mali
Senegal
Kenya
Republic of the Congo
Sierra Leone
Democratic Republic of the Congo
Mozambique
Angola
Cape Verde
Zambia
Cote d'Ivoire
Niger
South Sudan
Mauritania
Comoros
Country

```
plt.show()
```


Units Sold by Country in North America
Units Sold
20000
17500
15000
12500
10000
7500
5000
2500
0
Mexico
Country

Thank You