In [8]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [9]:
```
df = pd.read_csv(r'C:\Users\kriti\OneDrive\Desktop\UM\Amazon Sales data
```

EXPLORATORY DATA ANALYSIS

In [11]:
```
df.head()
```

Out[11]:

	Region	Country	Item Type	Sales Channel	Order Priority	Order Date	Order ID	Ship Date	Uni So
0	Australia and Oceania	Tuvalu	Baby Food	Offline	H	5/28/2010	669165933	6/27/2010	992
1	Central America and the Caribbean	Grenada	Cereal	Online	C	8/22/2012	963881480	9/15/2012	280
2	Europe	Russia	Office Supplies	Offline	L	05-02-2014	341417157	05-08-2014	177
3	Sub-Saharan Africa	Sao Tome and Principe	Fruits	Online	C	6/20/2014	514321792	07-05-2014	810
4	Sub-Saharan Africa	Rwanda	Office Supplies	Offline	L	02-01-2013	115456712	02-06-2013	506

In [12]:
```
df.shape
```

Out[12]:
```
(100, 14)
```

In [13]:
```
# check the columns
df.columns
```

Out[13]:
```
Index(['Region', 'Country', 'Item Type', 'Sales Channel', 'Order Prior
ity',
       'Order Date', 'Order ID', 'Ship Date', 'Units Sold', 'Unit Pric
e',
       'Unit Cost', 'Total Revenue', 'Total Cost', 'Total Profit'],
      dtype='object')
```

In [14]:
```
# information about the dataset
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 14 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Region          100 non-null    object
 1   Country         100 non-null    object
 2   Item Type       100 non-null    object
 3   Sales Channel   100 non-null    object
 4   Order Priority  100 non-null    object
 5   Order Date      100 non-null    object
 6   Order ID        100 non-null    int64
 7   Ship Date       100 non-null    object
 8   Units Sold      100 non-null    int64
 9   Unit Price      100 non-null    float64
 10  Unit Cost       100 non-null    float64
 11  Total Revenue   100 non-null    float64
 12  Total Cost      100 non-null    float64
 13  Total Profit    100 non-null    float64
dtypes: float64(5), int64(2), object(7)
memory usage: 11.1+ KB
```

In [15]:
```
# columns with categorical values
df.select_dtypes(include=['object']).columns
```

Out[15]:
```
Index(['Region', 'Country', 'Item Type', 'Sales Channel', 'Order Prior
ity',
       'Order Date', 'Ship Date'],
      dtype='object')
```

In [16]:
```
len(df.select_dtypes(include=['object']).columns)
```

Out[16]:
```
7
```

In [17]:
```
# columns with numerical values
df.select_dtypes(include=['int64', 'float64']).columns
```

Out[17]:
```
Index(['Order ID', 'Units Sold', 'Unit Price', 'Unit Cost', 'Total Rev
enue',
       'Total Cost', 'Total Profit'],
      dtype='object')
```

In [18]:
```
len(df.select_dtypes(include=['int64', 'float64']).columns)
```

Out[18]:
```
7
```

Dealing with Missing Data

In [19]:
```
# check if there are any null values
df.isnull().values.any() # this function returns true and false
```

Out[19]:
```
False
```

In [20]:
```
# check how many null values
df.isnull().values.sum()
```

Out[20]:
```
0
```

Categorical Data

In [21]:
```
df.head()
```

Out[21]:

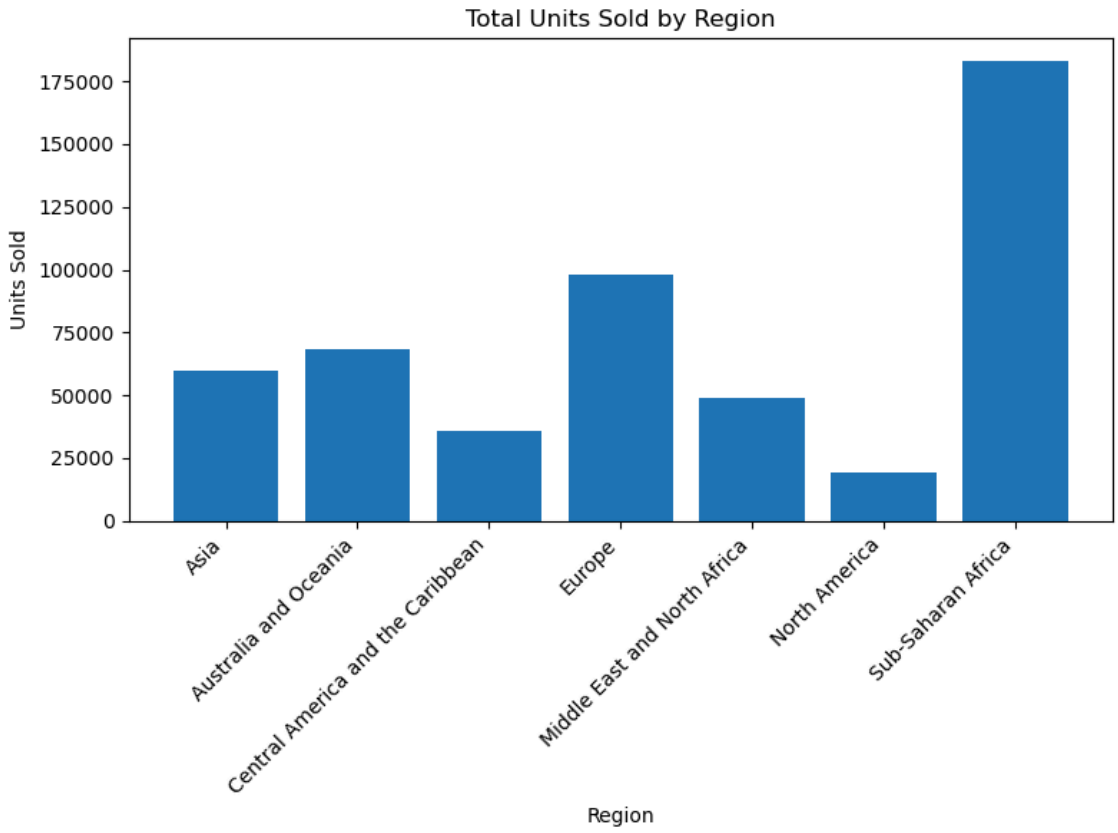
	Region	Country	Item Type	Sales Channel	Order Priority	Order Date	Order ID	Ship Date	Uni So
0	Australia and Oceania	Tuvalu	Baby Food	Offline	H	5/28/2010	669165933	6/27/2010	992
1	Central America and the Caribbean	Grenada	Cereal	Online	C	8/22/2012	963881480	9/15/2012	280
2	Europe	Russia	Office Supplies	Offline	L	05-02-2014	341417157	05-08-2014	177
3	Sub-Saharan Africa	Sao Tome and Principe	Fruits	Online	C	6/20/2014	514321792	07-05-2014	810
4	Sub-Saharan Africa	Rwanda	Office Supplies	Offline	L	02-01-2013	115456712	02-06-2013	506

In [56]:
```
# check the number of unique values in each column
print(df['Order ID'].nunique())
print(df['Ship Date'].nunique())
print(df['Order Priority'].nunique())
print(df['Country'].nunique())
```

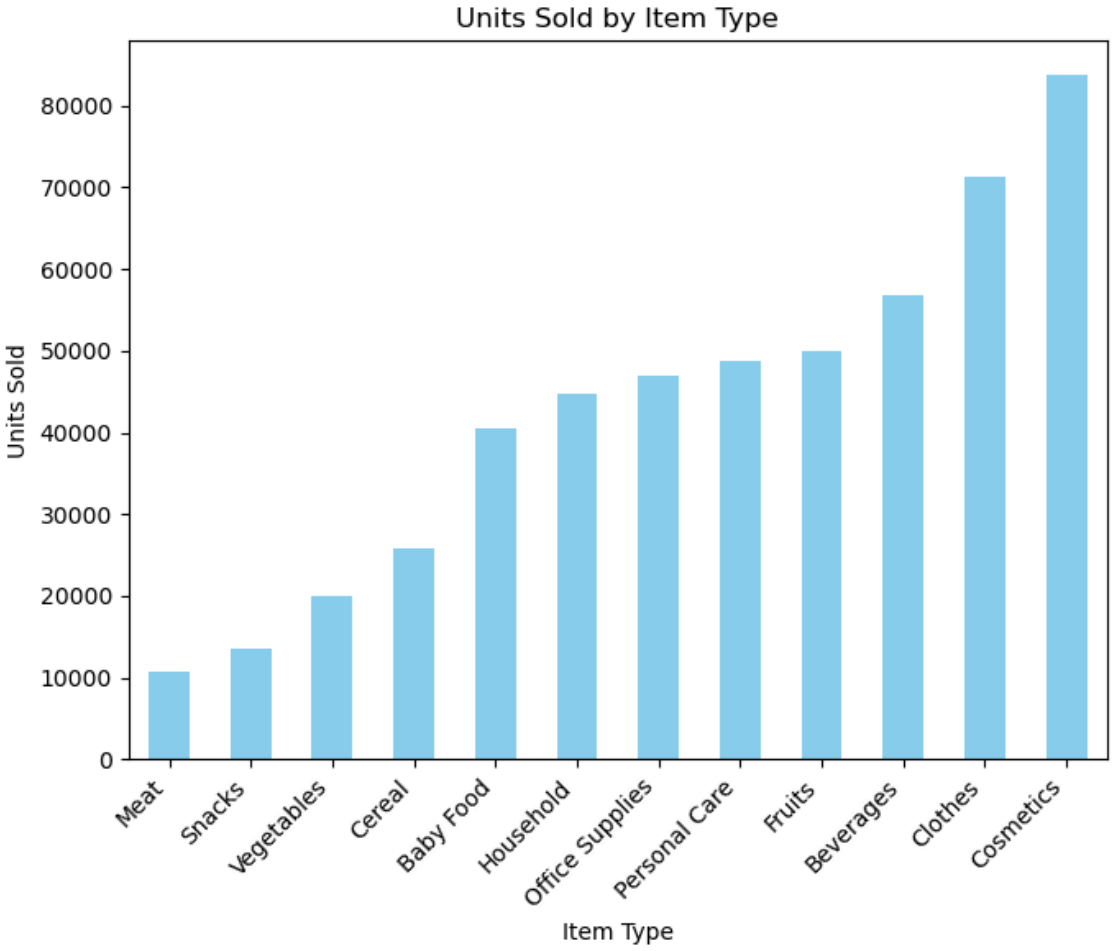
```
100
99
4
76
```

In [57]:

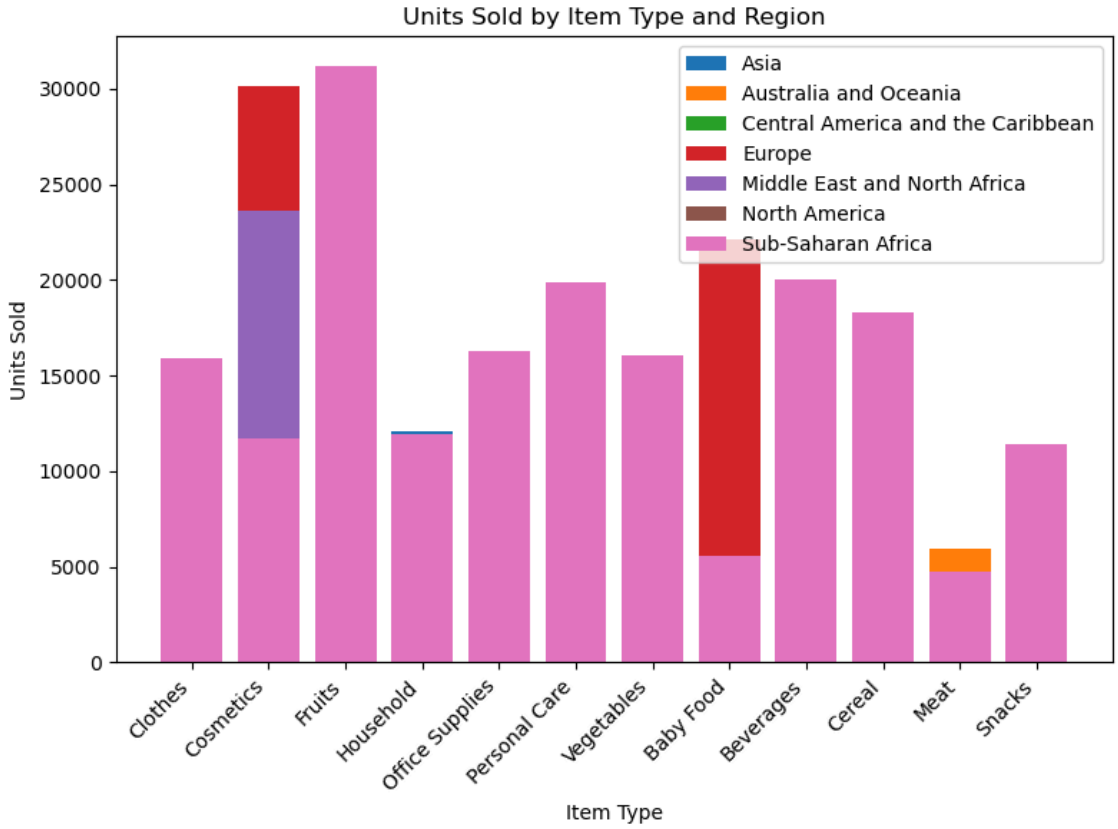
```
import pandas as pd
import matplotlib.pyplot as plt
region_units_sold = df[['Region', 'Units Sold']]
region_units_sold_sum = region_units_sold.groupby('Region').sum()
plt.figure(figsize=(8, 6))
plt.bar(region_units_sold_sum.index, region_units_sold_sum['Units Sold'
plt.xlabel('Region')
plt.ylabel('Units Sold')
plt.title('Total Units Sold by Region')
plt.xticks(rotation=10)
import textwrap
def wrap_text(text, width):
    return '\n'.join(textwrap.wrap(text, width))

# Set the desired width for wrapping
wrap_width = 40

# Wrap the x-axis labels
wrapped_labels = [wrap_text(label, wrap_width) for label in region_unit
plt.xticks(range(len(region_units_sold_sum.index)), wrapped_labels, rot

plt.tight_layout()
plt.show()
```



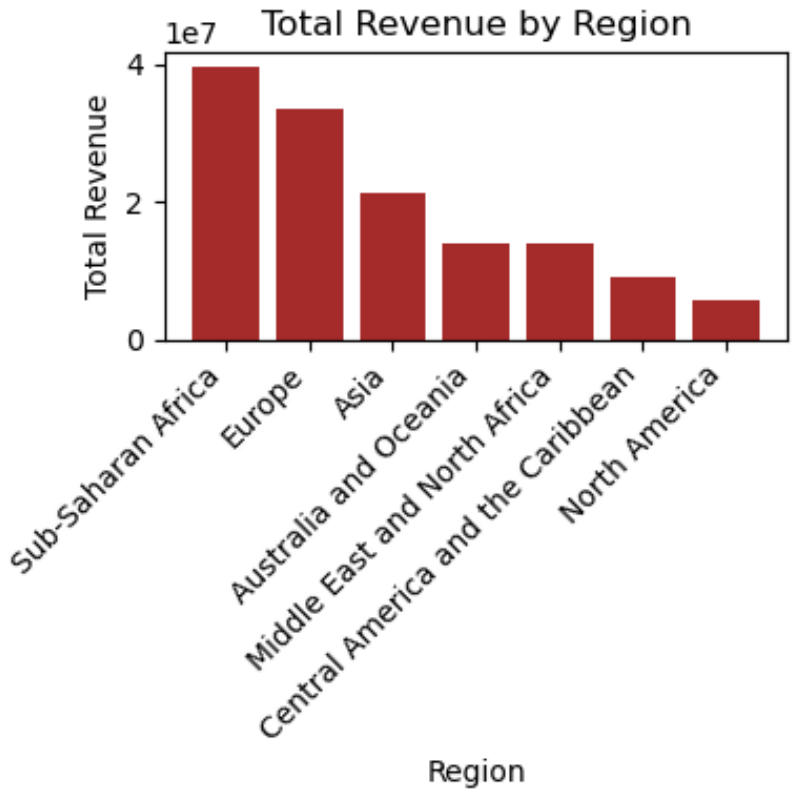
In [58]:
```
# Group by 'Item Type' and sum the 'Units_Sold' for each item type
item_sales = df.groupby('Item Type')['Units Sold'].sum().sort_values()

# Plotting the graph
plt.figure(figsize=(7, 6))
item_sales.plot(kind='bar', color='skyblue')
plt.title('Units Sold by Item Type')
plt.xlabel('Item Type')
plt.ylabel('Units Sold')
plt.xticks(rotation=45, ha='right')  # Rotate x-axis labels for better
plt.tight_layout()
plt.show()
```



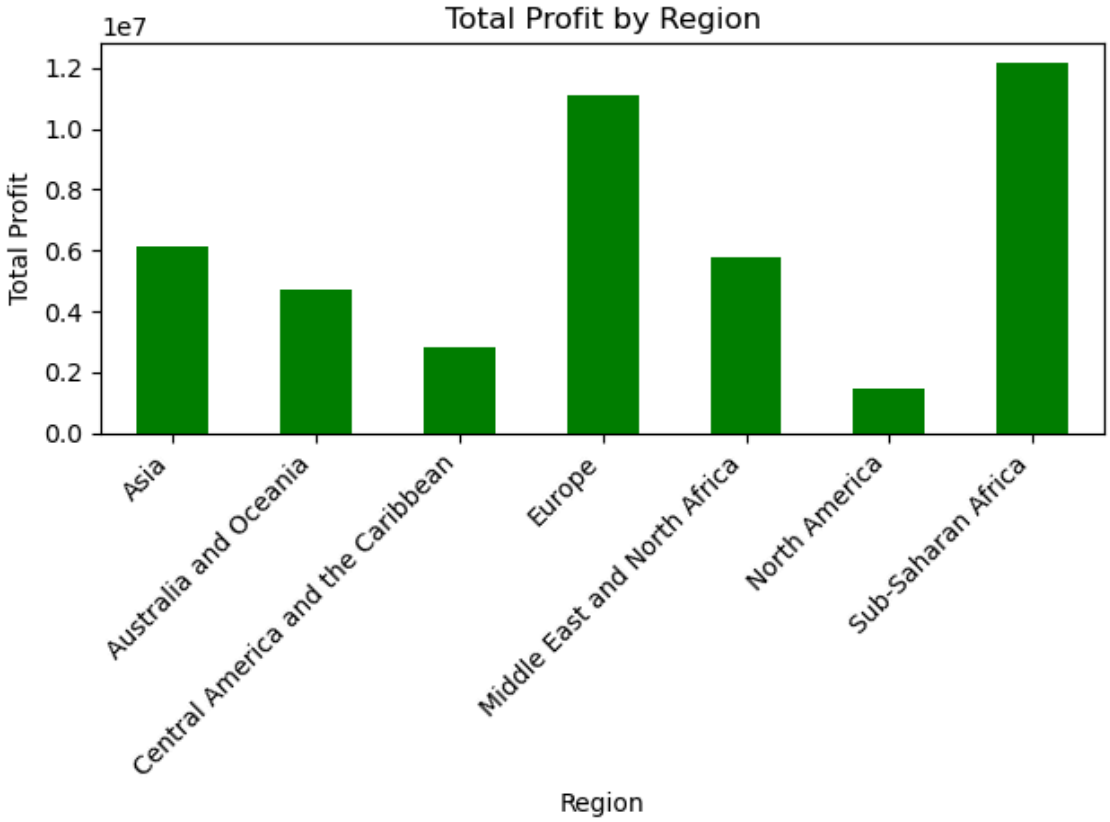
In [59]:
```
# Group by 'Region' and 'Item Type' and sum the 'Units Sold'
grouped_data = df.groupby(['Region', 'Item Type'])['Units Sold'].sum().

# Plotting the graph
plt.figure(figsize=(8, 6))
for region in grouped_data['Region'].unique():
    data = grouped_data[grouped_data['Region'] == region]
    plt.bar(data['Item Type'], data['Units Sold'], label=region)

plt.xlabel('Item Type')
plt.ylabel('Units Sold')
plt.title('Units Sold by Item Type and Region')
plt.xticks(rotation=45, ha='right')
plt.legend()
plt.tight_layout()
plt.show()
```



In [62]:
```
# Grouping the data by 'Region' and calculating the total revenue for e
revenue_by_region = df.groupby('Region')['Total Revenue'].sum().reset_i

# Sorting the data by total revenue in descending order for better visu
revenue_by_region = revenue_by_region.sort_values(by='Total Revenue', a

# Plotting the bar chart
plt.figure(figsize=(4, 4))
plt.bar(revenue_by_region['Region'], revenue_by_region['Total Revenue']
plt.xlabel('Region')
plt.ylabel('Total Revenue')
plt.title('Total Revenue by Region')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()  # Adjust layout to prevent clipping of labels
plt.show()
```



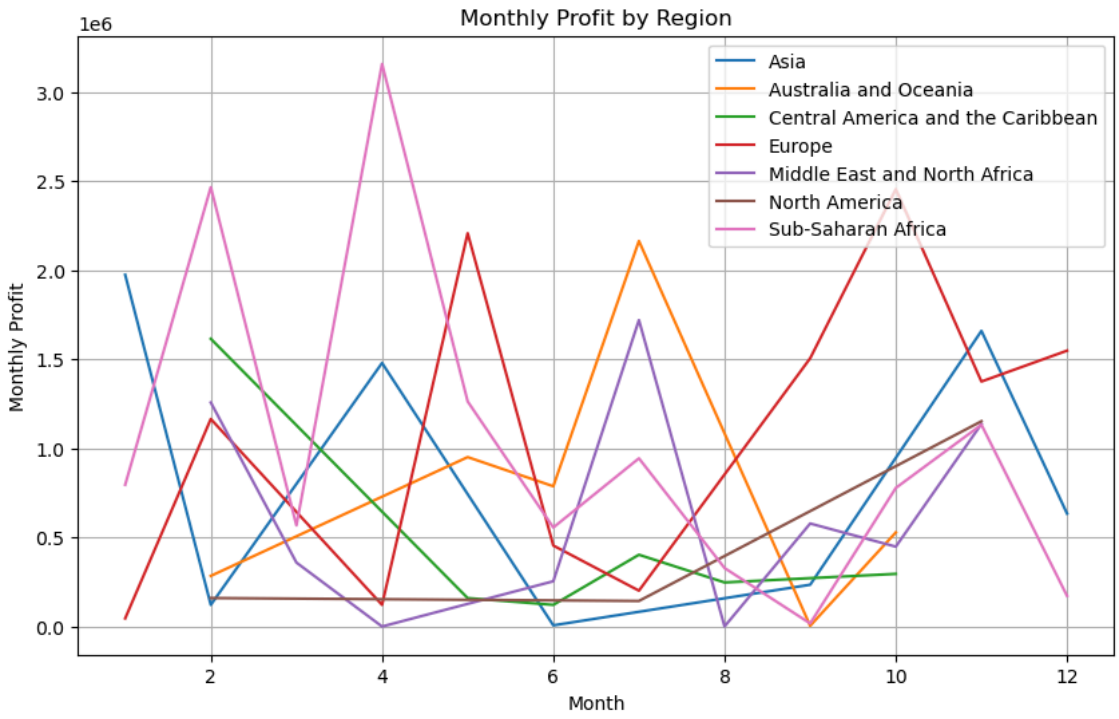
In [28]:
```
region_profit_df = df[['Region', 'Total Profit']]
region_profit_sum = region_profit_df.groupby('Region')['Total Profit'].
region_profit_sum.plot(kind='bar', color='green')
plt.xlabel('Region')
plt.ylabel('Total Profit')
plt.title('Total Profit by Region')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()
```



In [29]:
```
df['Order Date'] = pd.to_datetime(df['Order Date'])
df['Month'] = df['Order Date'].dt.month
monthly_profit = df.groupby('Month')['Total Profit'].sum()
print(monthly_profit)
```

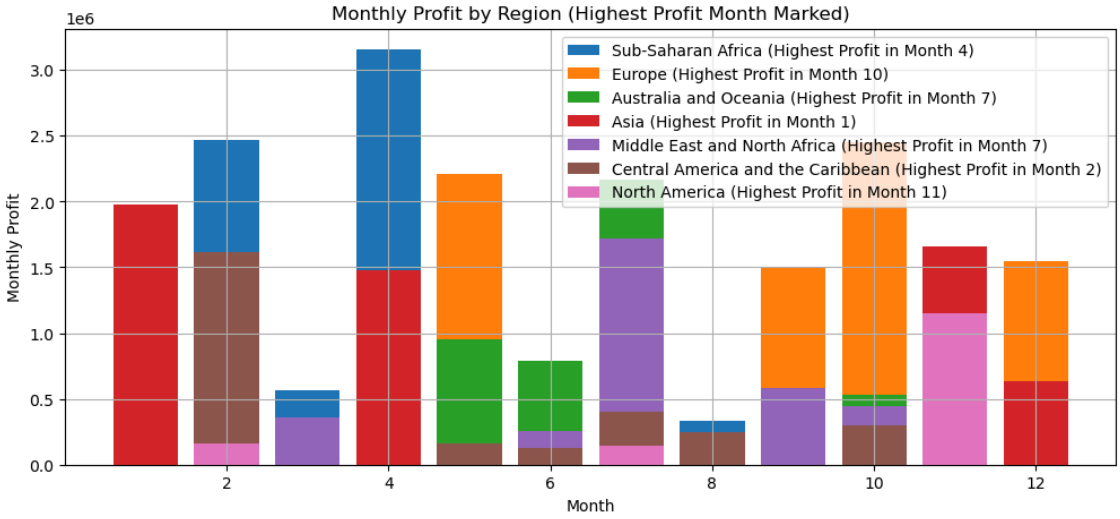
```
Month
1     2816857.02
2     7072050.51
3      928351.06
4     4760208.35
5     4582692.30
6     2185379.43
7     5578463.06
8      579276.67
9     2344166.03
10    4506923.25
11    6457600.65
12    2356230.07
Name: Total Profit, dtype: float64
```

In [30]:
```
import pandas as pd
import matplotlib.pyplot as plt

# Assuming you have calculated monthly_profit as shown in the previous

# Assuming df is your DataFrame containing the sales data
# If not, replace df with your actual DataFrame name

# Convert 'Order Date' column to datetime format if not done already
df['Order Date'] = pd.to_datetime(df['Order Date'])

# Extract the month from 'Order Date' column
df['Month'] = df['Order Date'].dt.month

# Group by 'Region' and 'Month' to get monthly profit by region
region_monthly_profit = df.groupby(['Region', 'Month'])['Total Profit']

# Plotting the graph
plt.figure(figsize=(10, 6))
for region in region_monthly_profit['Region'].unique():
    plt.plot(region_monthly_profit[region_monthly_profit['Region'] == r
             region_monthly_profit[region_monthly_profit['Region'] == r
             label=region)

plt.xlabel('Month')
plt.ylabel('Monthly Profit')
plt.title('Monthly Profit by Region')
plt.legend()
plt.grid(True)
plt.show()
```



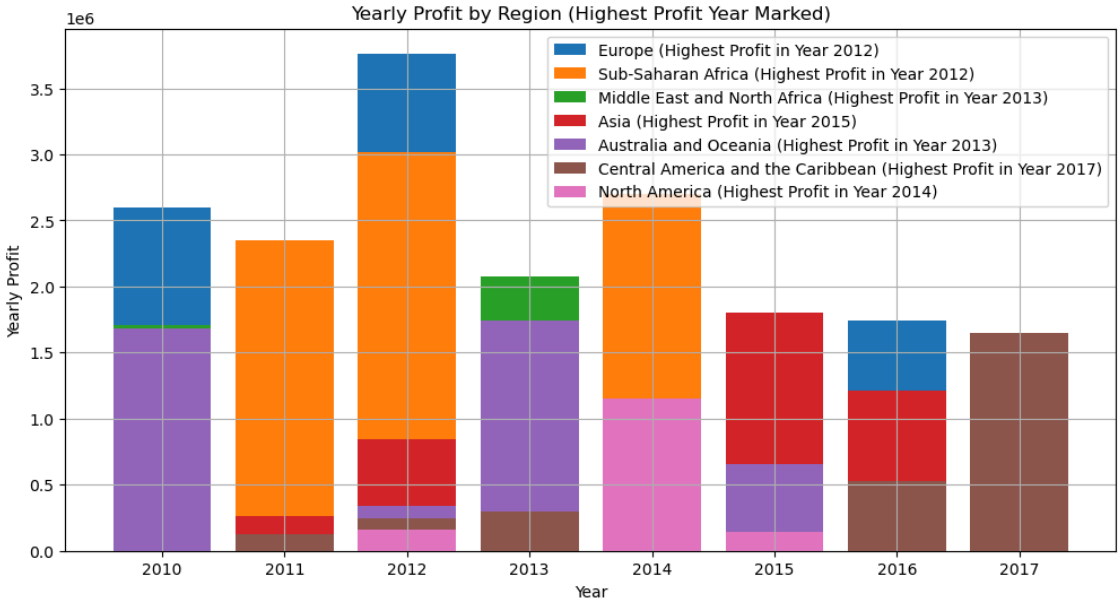
In [66]:

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

# Assuming you have calculated monthly_profit as shown in the previous

# Assuming df is your DataFrame containing the sales data
# If not, replace df with your actual DataFrame name

# Convert 'Order Date' column to datetime format if not done already
df['Order Date'] = pd.to_datetime(df['Order Date'])

# Extract the month from 'Order Date' column
df['Month'] = df['Order Date'].dt.month

# Group by 'Region' and 'Month' to get monthly profit by region
region_monthly_profit = df.groupby(['Region', 'Month'])['Total Profit']

# Get the month with the highest profit for each region
highest_profit_month = region_monthly_profit.loc[region_monthly_profit.

# Sort the highest profit months by profit in descending order
highest_profit_month_sorted = highest_profit_month.sort_values(by='Tota

# Plotting separate bar graphs for each region
plt.figure(figsize=(12, 5))

# Loop through each region and plot the bar graph
for idx, row in highest_profit_month_sorted.iterrows():
    region = row['Region']
    highest_profit = row['Total Profit']
    month = row['Month']
    region_data = region_monthly_profit[region_monthly_profit['Region']

    plt.bar(region_data['Month'], region_data['Total Profit'], label=f'

plt.xlabel('Month')
plt.ylabel('Monthly Profit')
plt.title('Monthly Profit by Region (Highest Profit Month Marked)')
plt.legend()
plt.grid(True)
plt.show()
```


Monthly Profit by Region (Highest Profit Month Marked)
1e6
Monthly Profit
Month
Sub-Saharan Africa (Highest Profit in Month 4)
Europe (Highest Profit in Month 10)
Australia and Oceania (Highest Profit in Month 7)
Asia (Highest Profit in Month 1)
Middle East and North Africa (Highest Profit in Month 7)
Central America and the Caribbean (Highest Profit in Month 2)
North America (Highest Profit in Month 11)

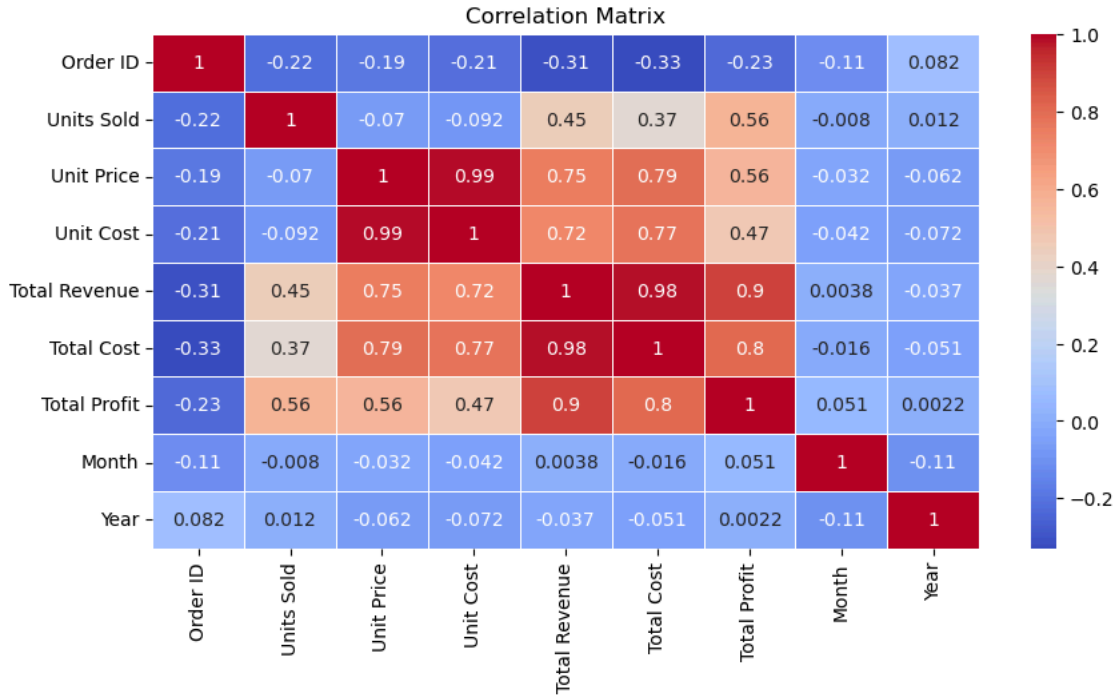
In [67]:

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

# Assuming you have calculated yearly_profit as shown in the previous c

# Assuming df is your DataFrame containing the sales data
# If not, replace df with your actual DataFrame name

# Convert 'Order Date' column to datetime format if not done already
df['Order Date'] = pd.to_datetime(df['Order Date'])

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

# Group by 'Region' and 'Year' to get yearly profit by region
region_yearly_profit = df.groupby(['Region', 'Year'])['Total Profit'].s

# Get the year with the highest profit for each region
highest_profit_year = region_yearly_profit.loc[region_yearly_profit.gro

# Sort the highest profit years by profit in descending order
highest_profit_year_sorted = highest_profit_year.sort_values(by='Total

# Plotting separate bar graphs for each region
plt.figure(figsize=(12, 6))

# Loop through each region and plot the bar graph
for idx, row in highest_profit_year_sorted.iterrows():
    region = row['Region']
    highest_profit = row['Total Profit']
    year = row['Year']
    region_data = region_yearly_profit[region_yearly_profit['Region'] =

    plt.bar(region_data['Year'], region_data['Total Profit'], label=f'{

plt.xlabel('Year')
plt.ylabel('Yearly Profit')
plt.title('Yearly Profit by Region (Highest Profit Year Marked)')
plt.legend()
plt.grid(True)
plt.show()
```



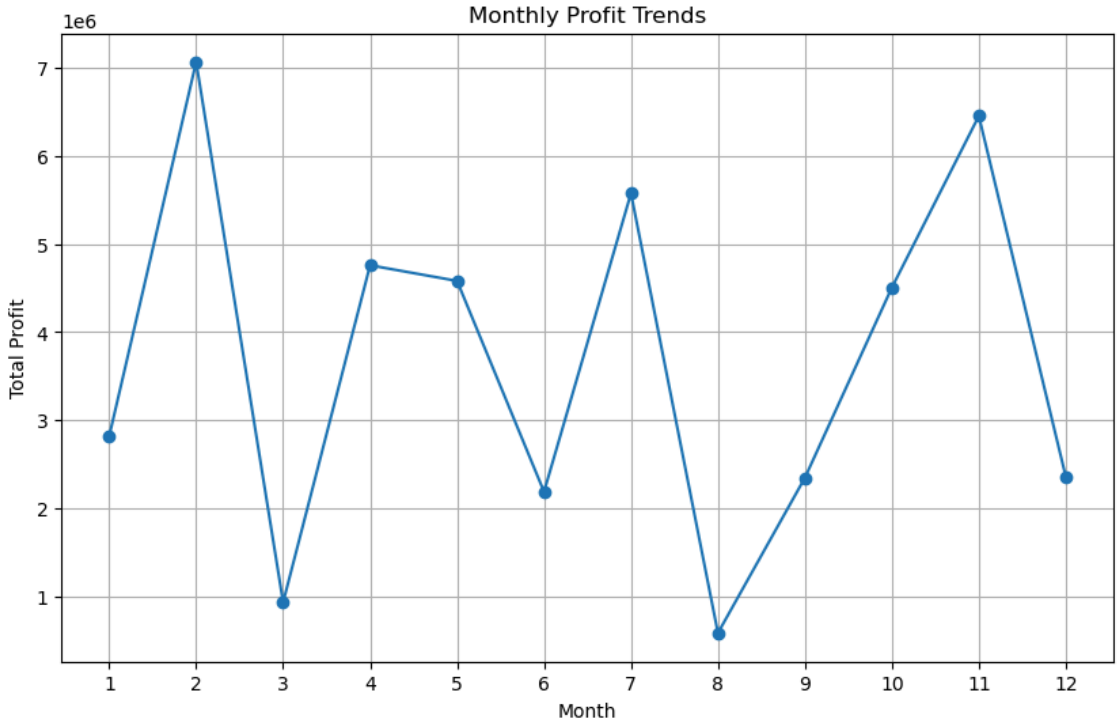
In [69]:
```
plt.figure(figsize=(10, 5))
sns.heatmap(df.corr(), annot=True, cmap='coolwarm', linewidths=0.5)
plt.title('Correlation Matrix')
plt.show()
```

```
C:\Users\kriti\AppData\Local\Temp\ipykernel_5904\911409688.py:2: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future version, it will default to False. Select only valid columns or specify the value of numeric_only to silence this warning.
  sns.heatmap(df.corr(), annot=True, cmap='coolwarm', linewidths=0.5)
```

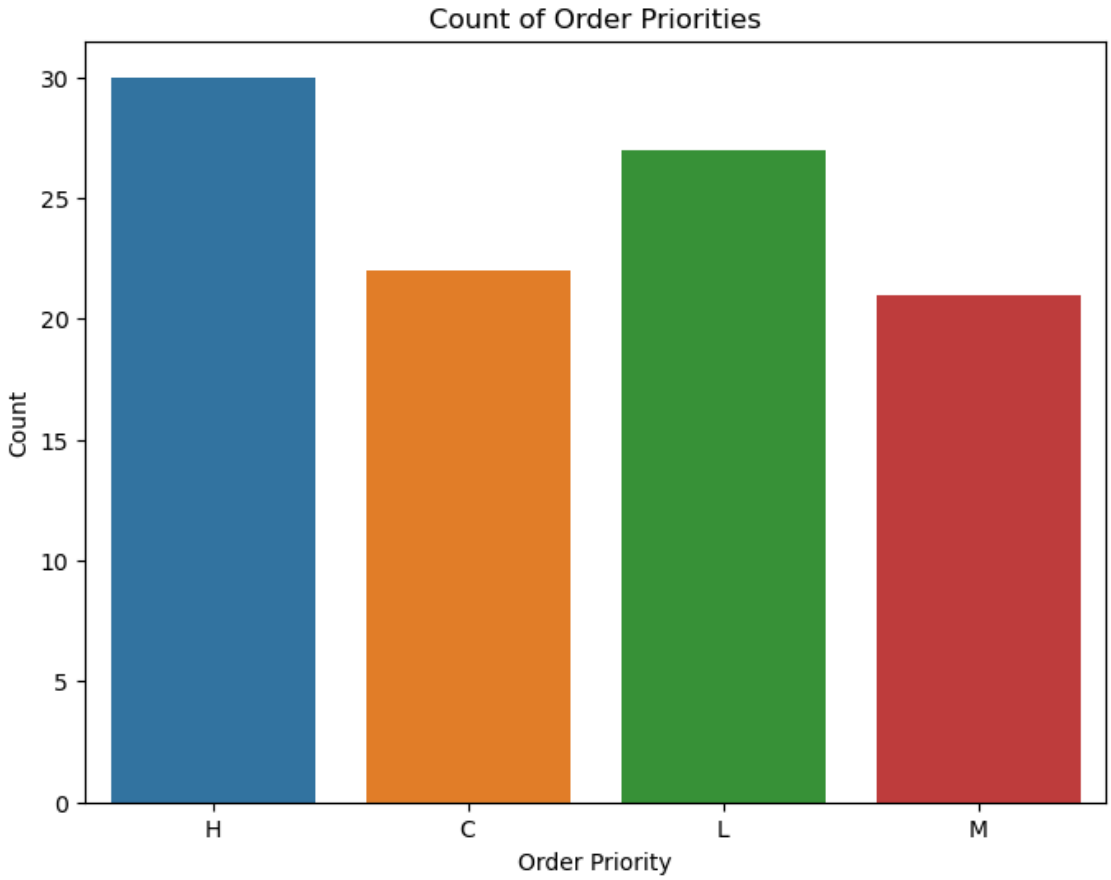


In [34]:
```
plt.figure(figsize=(10, 6))
sns.scatterplot(data=df, x='Sales Channel', y='Total Profit')
plt.title('Sales Channel vs. Total Profit')
plt.xlabel('Sales Channel')
plt.ylabel('Total Profit')
plt.xticks(rotation=45)
plt.show()
```

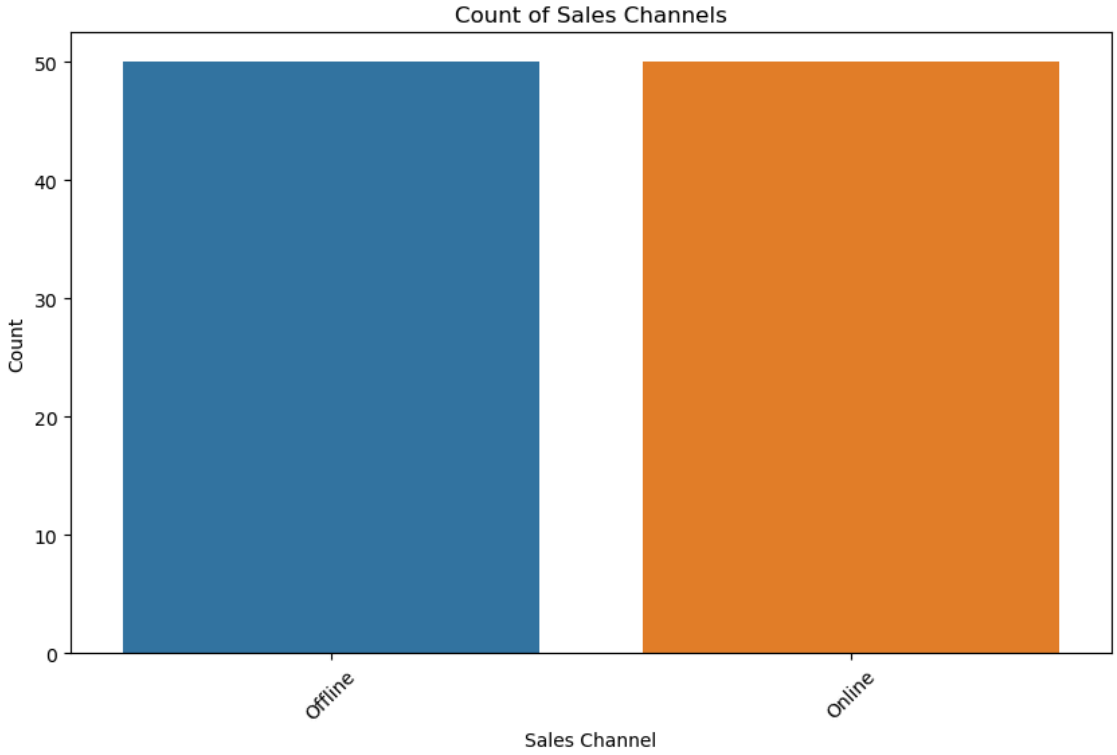
In [35]:
```
df['Order Date'] = pd.to_datetime(df['Order Date'])
df['Month'] = df['Order Date'].dt.month

monthly_profit = df.groupby('Month')['Total Profit'].sum()

plt.figure(figsize=(10, 6))
plt.plot(monthly_profit.index, monthly_profit.values, marker='o')
plt.title('Monthly Profit Trends')
plt.xlabel('Month')
plt.ylabel('Total Profit')
plt.xticks(monthly_profit.index)
plt.grid(True)
plt.show()
```



In [36]:
```
plt.figure(figsize=(8, 6))
sns.countplot(data=df, x='Order Priority')
plt.title('Count of Order Priorities')
plt.xlabel('Order Priority')
plt.ylabel('Count')
plt.show()
```



In [37]:
```
import matplotlib.pyplot as plt
import seaborn as sns

# Assuming df is your DataFrame containing the sales data
# If not, replace df with your actual DataFrame name

plt.figure(figsize=(10, 6))
sns.countplot(data=df, x='Sales Channel')
plt.title('Count of Sales Channels')
plt.xlabel('Sales Channel')
plt.ylabel('Count')
plt.xticks(rotation=45)
plt.show()
```



In [53]:
```
# Define features and target variable
X = df[['Unit Cost', 'Units Sold', 'Unit Price']]
Y = df['Total Profit']
```

In [54]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score



# Split the data into training and testing sets
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2

# Initialize and train the regression model
model = LinearRegression()
model.fit(X_train, Y_train)

# Make predictions on the test set
Y_pred = model.predict(X_test)

# Evaluate the model
mse = mean_squared_error(Y_test, Y_pred)
r2 = r2_score(Y_test, Y_pred)

# Print model evaluation metrics
print('Mean Squared Error:', mse)
print('R-squared Score:', r2)

# Optionally, visualize the predictions
# (e.g., plot actual vs predicted values)

# Save or export the model for future use
# (e.g., using pickle or joblib)
```

```
Mean Squared Error: 15118181009.87363
R-squared Score: 0.9095678745429016
```

In [ ]: