Importing necessary libraries

In [ ]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from datetime import datetime
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error

sns.set(style='whitegrid') # setting seaborn style for the presentations like pie charts
```

Data Collection and Preparation

In [ ]:
```
data = pd.read_csv('Amazon Sales data.csv') # loading data

# Part of data Cleaning
# Here it is checking for any missing data in the datasets
print(data.isnull().sum())

# filling missing data
data.fillna(method='ffill', inplace=True)

# checking datatype is correct or not
data['Order Date'] = pd.to_datetime(data['Order Date'])
data['Ship Date'] = pd.to_datetime(data['Ship Date'])

# Remove duplicates
data.drop_duplicates(inplace=True)
```

```
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
dtype: int64
```

Descriptive Statistics

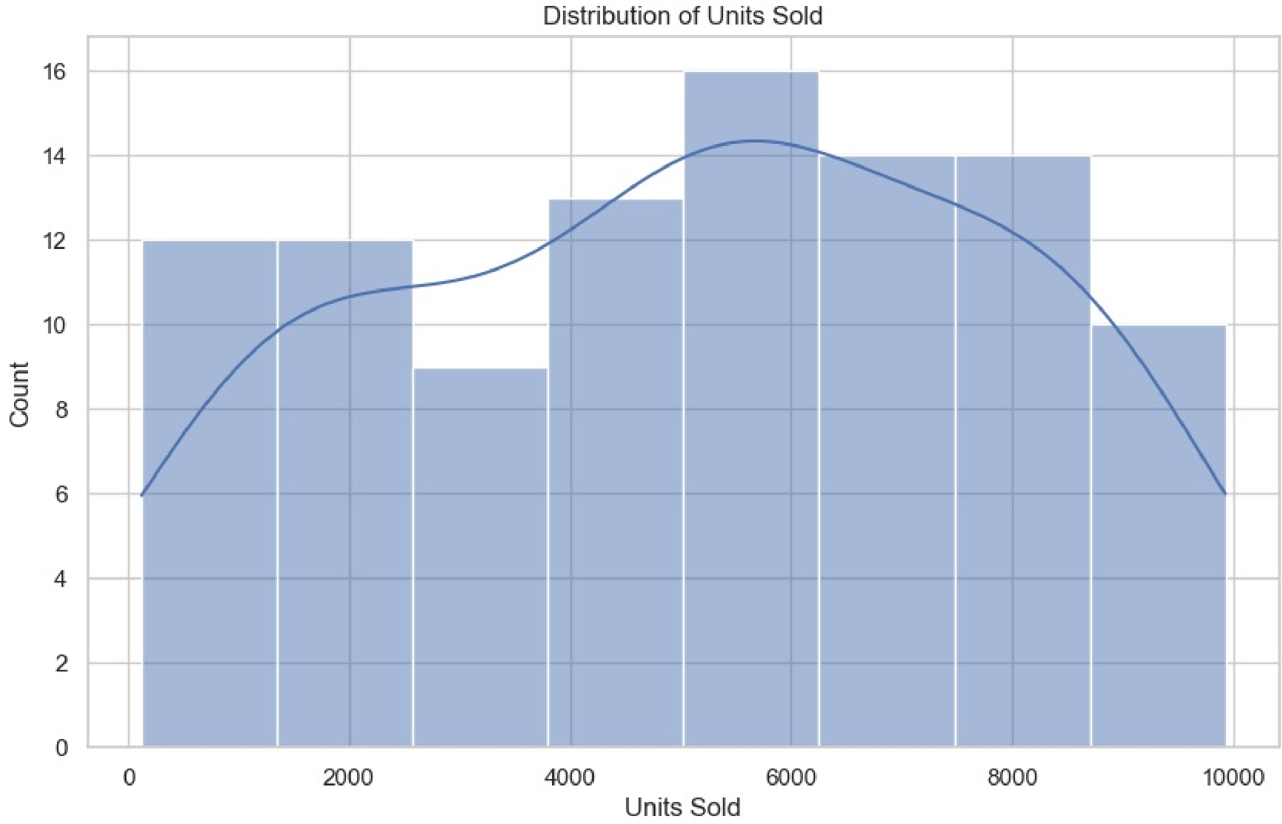
In [ ]:
```
print(data.describe())
```

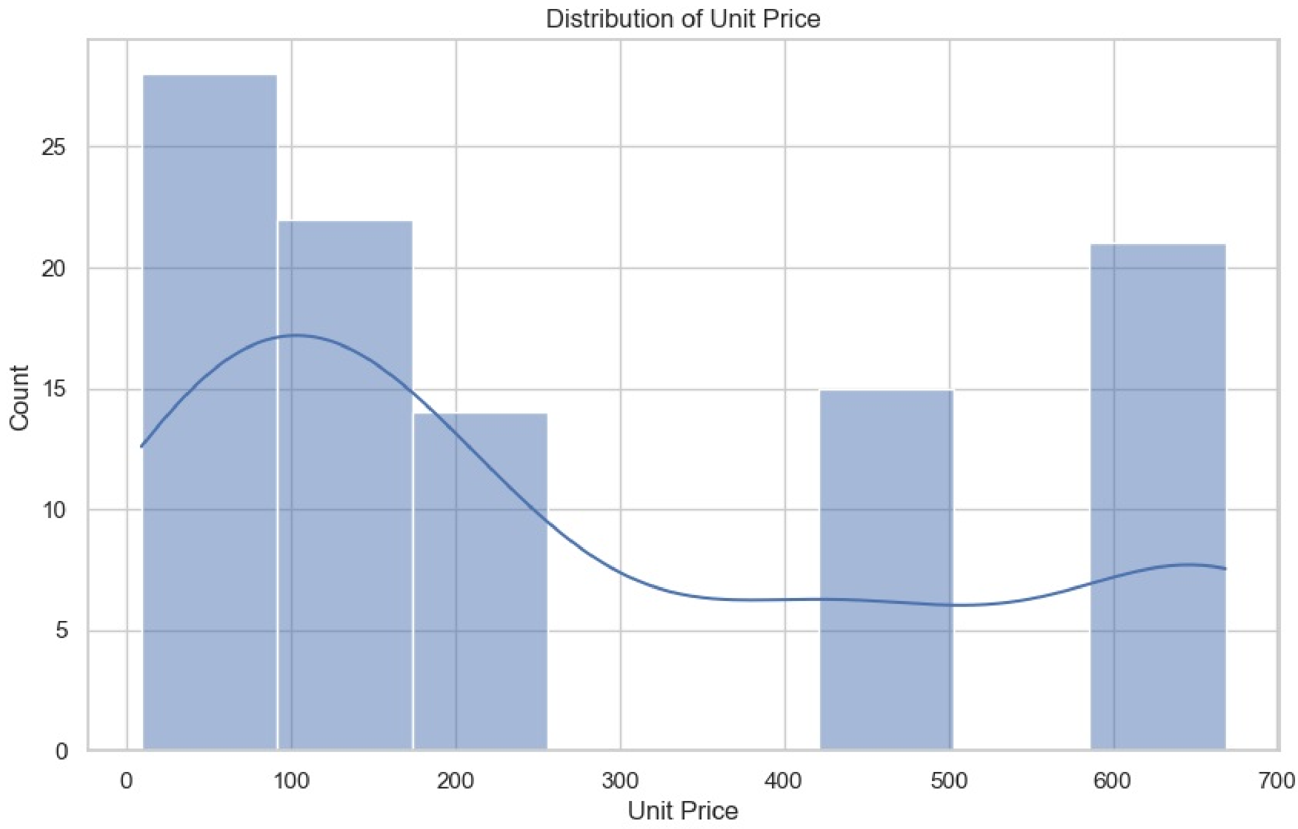
```
                Order Date      Order ID            Ship Date   Units Sold  \
count                  100  1.000000e+02                  100   100.000000
mean   2013-09-16 14:09:36  5.550204e+08  2013-10-09 22:48:00  5128.710000
min    2010-02-02 00:00:00  1.146066e+08  2010-02-25 00:00:00   124.000000
25%    2012-02-14 12:00:00  3.389225e+08  2012-02-24 18:00:00  2836.250000
50%    2013-07-12 12:00:00  5.577086e+08  2013-08-11 12:00:00  5382.500000
75%    2015-04-07 00:00:00  7.907551e+08  2015-04-28 00:00:00  7369.000000
max    2017-05-22 00:00:00  9.940222e+08  2017-06-17 00:00:00  9925.000000
std                    NaN  2.606153e+08                  NaN  2794.484562

       Unit Price   Unit Cost  Total Revenue    Total Cost  Total Profit
count  100.000000  100.000000   1.000000e+02  1.000000e+02  1.000000e+02
mean   276.761300  191.048000   1.373488e+06  9.318057e+05  4.416820e+05
min      9.330000    6.920000   4.870260e+03  3.612240e+03  1.258020e+03
25%     81.730000   35.840000   2.687212e+05  1.688680e+05  1.214436e+05
50%    179.880000  107.275000   7.523144e+05  3.635664e+05  2.907680e+05
75%    437.200000  263.330000   2.212045e+06  1.613870e+06  6.358288e+05
max    668.270000  524.960000   5.997055e+06  4.509794e+06  1.719922e+06
std    235.592241  188.208181   1.460029e+06  1.083938e+06  4.385379e+05
```

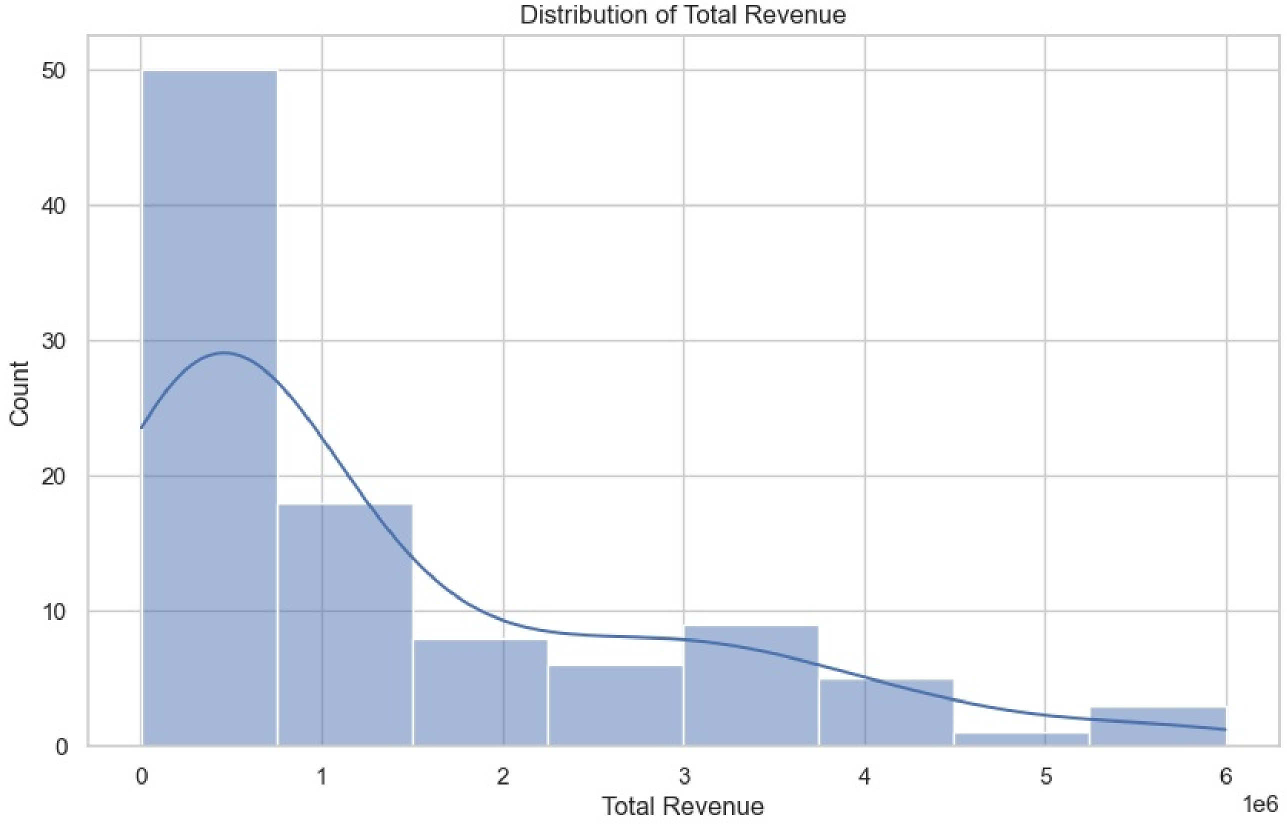
Data Distribution

In [ ]:
```
plt.figure(figsize=(10, 6))
sns.histplot(data['Units Sold'], kde=True)
plt.title('Distribution of Units Sold')
plt.show()
```

```
plt.figure(figsize=(10, 6))
sns.histplot(data['Unit Price'], kde=True)
plt.title('Distribution of Unit Price')
plt.show()

plt.figure(figsize=(10, 6))
sns.histplot(data['Total Revenue'], kde=True)
plt.title('Distribution of Total Revenue')
plt.show()
```

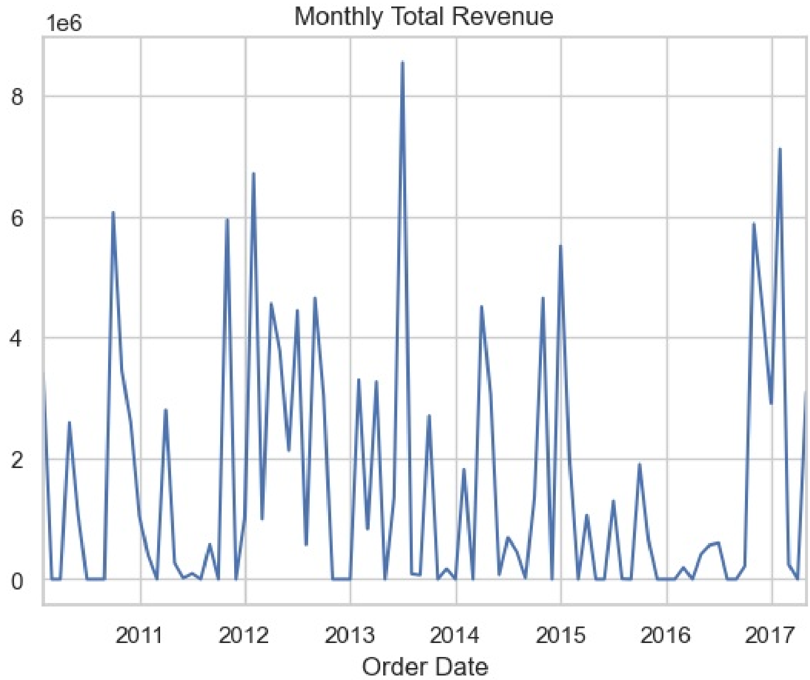






Time Series Analysis

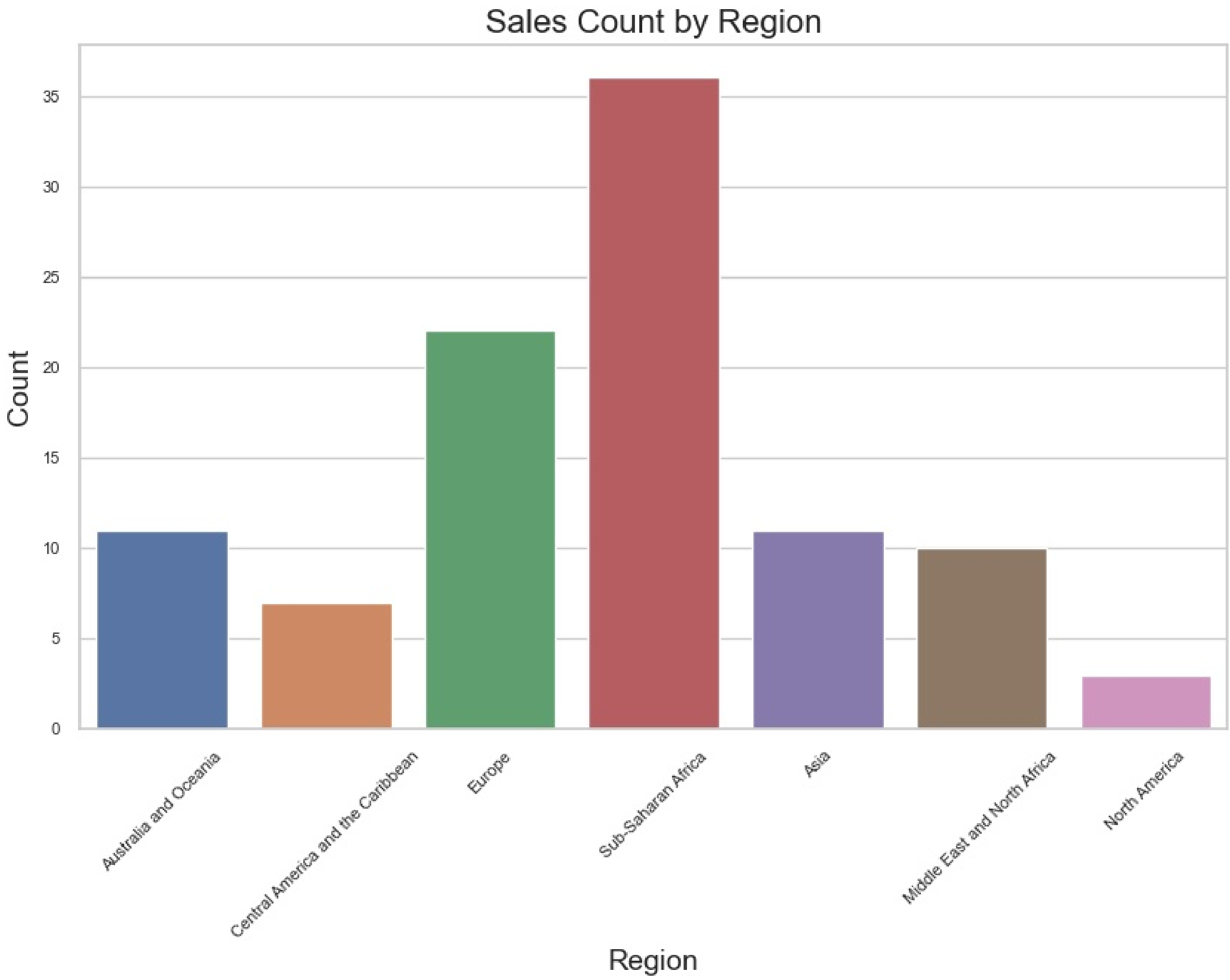
In [ ]:
```
data.set_index('Order Date', inplace=True)
data['Total Revenue'].resample('M').sum().plot()
plt.title('Monthly Total Revenue')
plt.show()
```

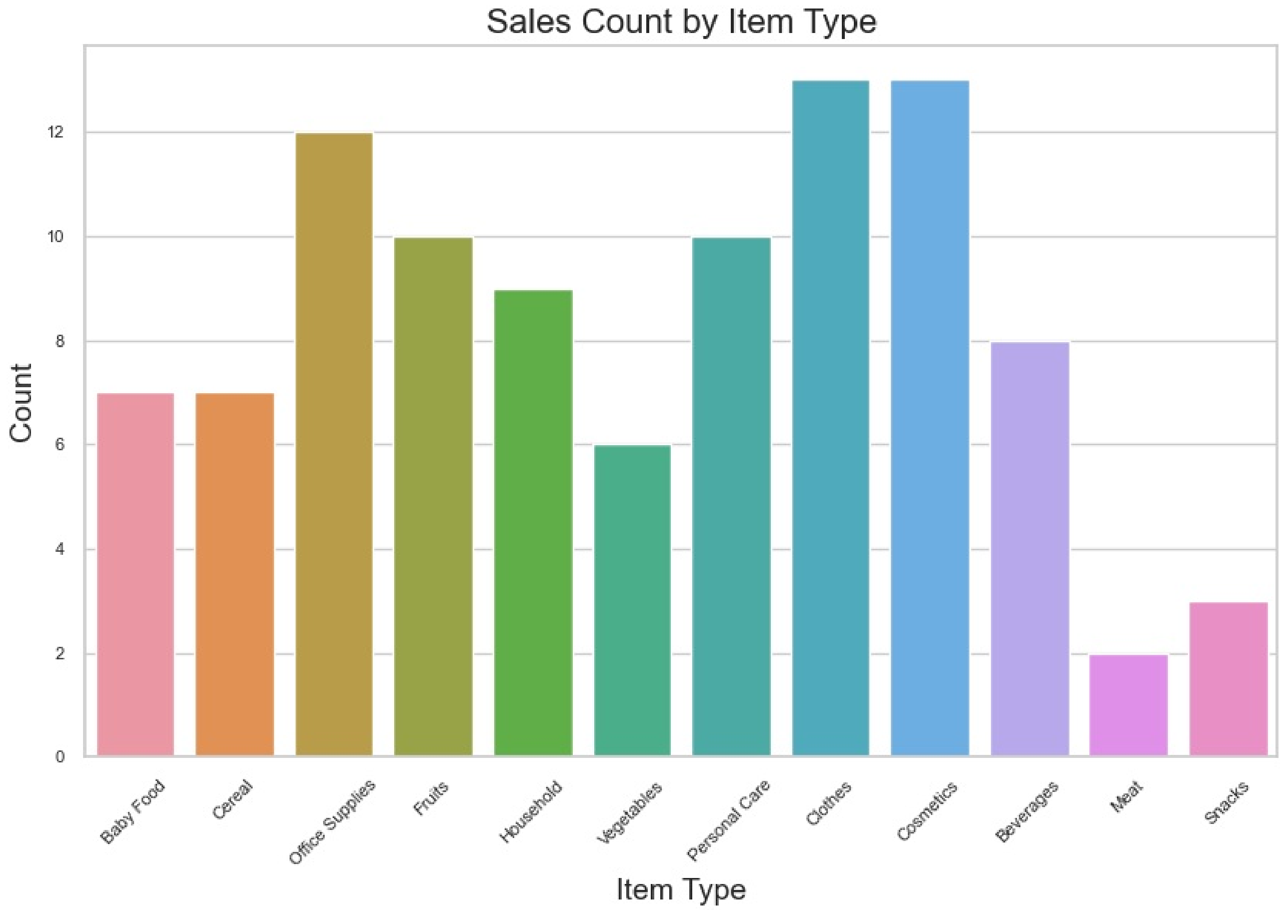


Categorical Analysis

In [ ]:
```
plt.figure(figsize=(10, 6))
sns.countplot(x='Region', data=data)
```

```python
plt.title('Sales Count by Region', fontsize=16)
plt.xlabel('Region', fontsize=14)
plt.ylabel('Count', fontsize=14)
plt.xticks(fontsize=8, rotation=45)
plt.yticks(fontsize=8)
plt.show()

plt.figure(figsize=(10, 6))
sns.countplot(x='Item Type', data=data)
plt.title('Sales Count by Item Type', fontsize=16)
plt.xlabel('Item Type', fontsize=14)
plt.ylabel('Count', fontsize=14)
plt.xticks(fontsize=8, rotation=45)
plt.yticks(fontsize=8)
plt.show()
```

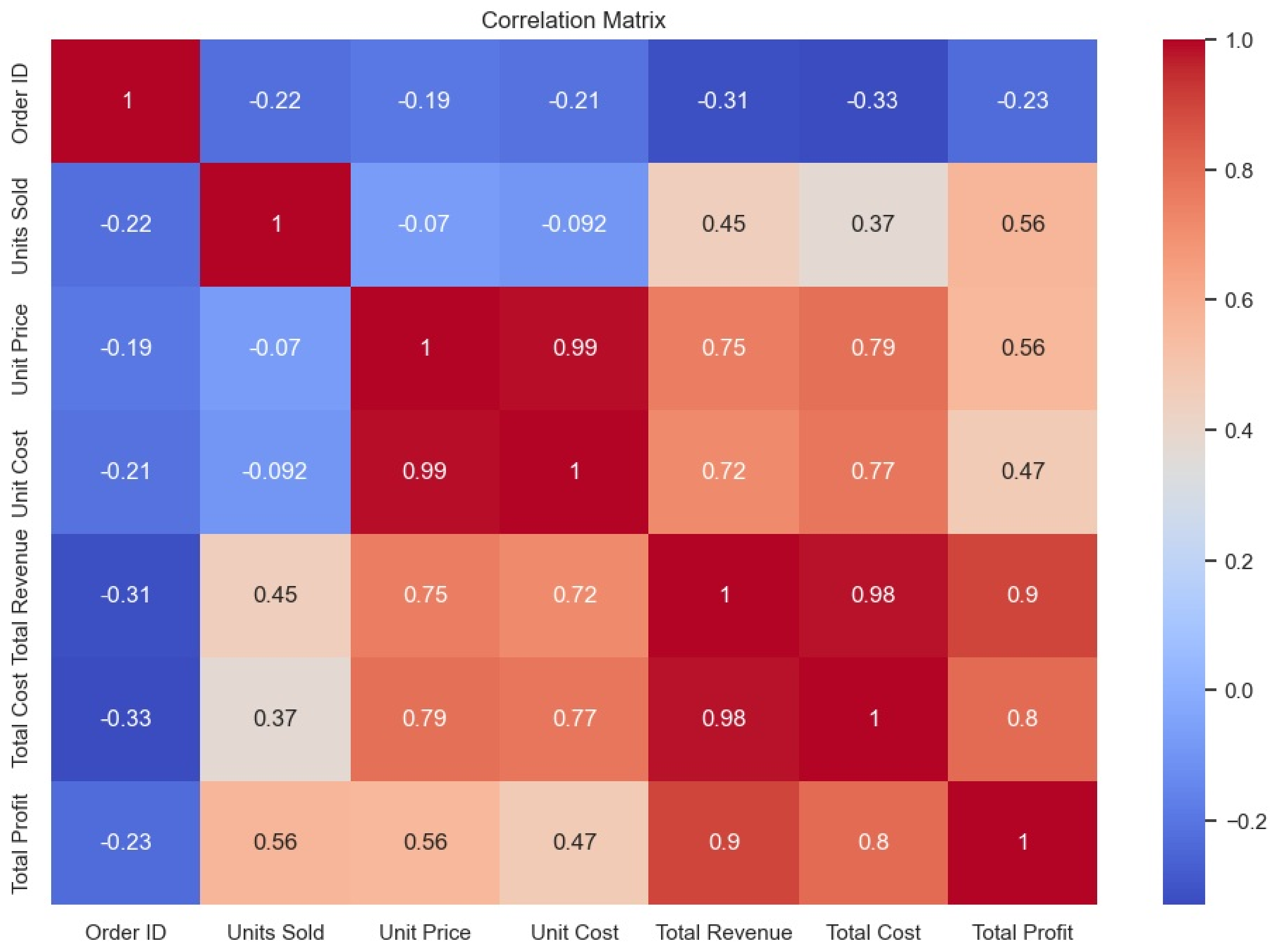




Correlation Analysis

In [ ]:

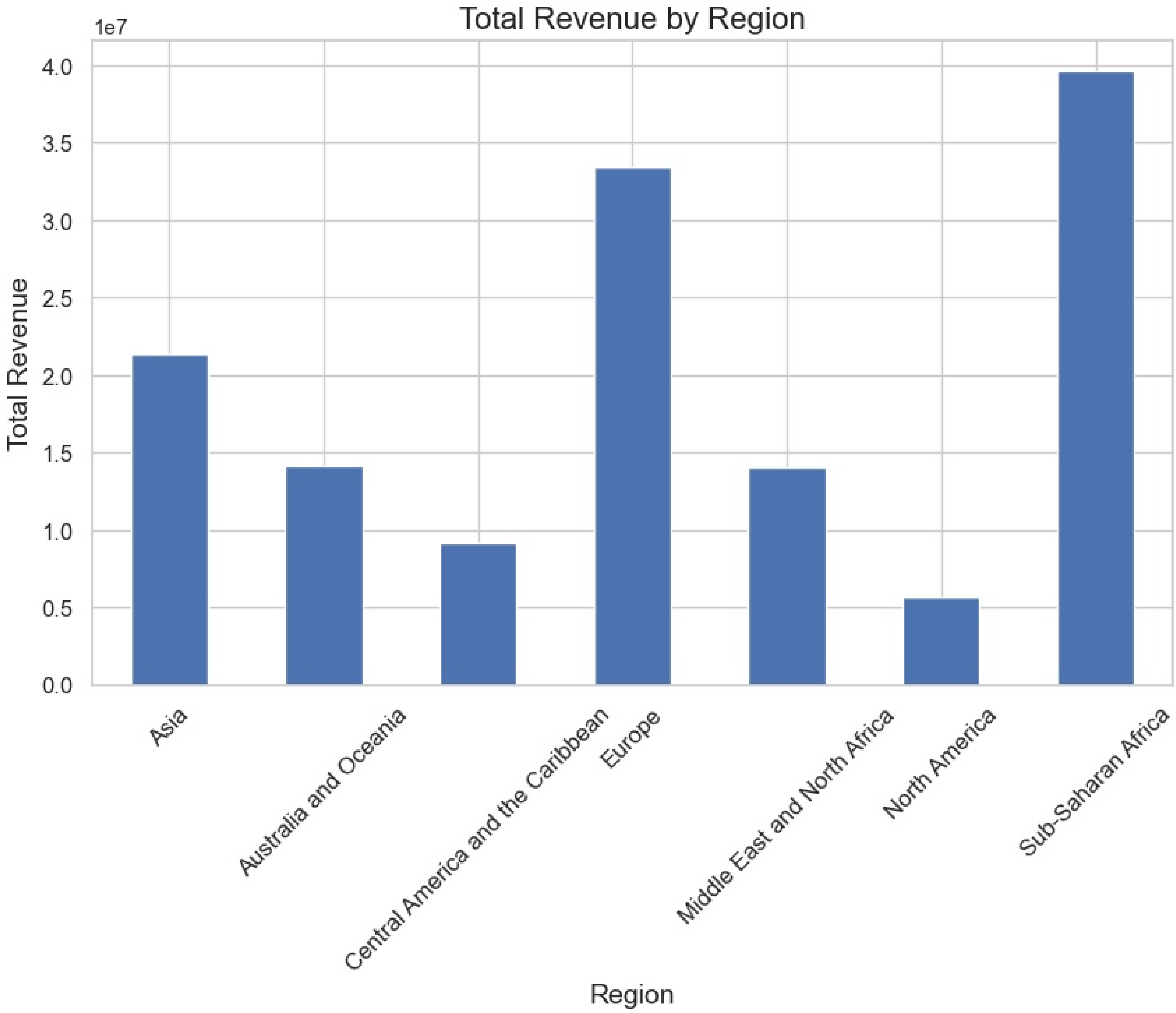
```
# Select only the numeric columns for correlation
numeric_data = data.select_dtypes(include=[np.number])

plt.figure(figsize=(12, 8))
sns.heatmap(numeric_data.corr(), annot=True, cmap='coolwarm')
plt.title('Correlation Matrix')
plt.show()
```

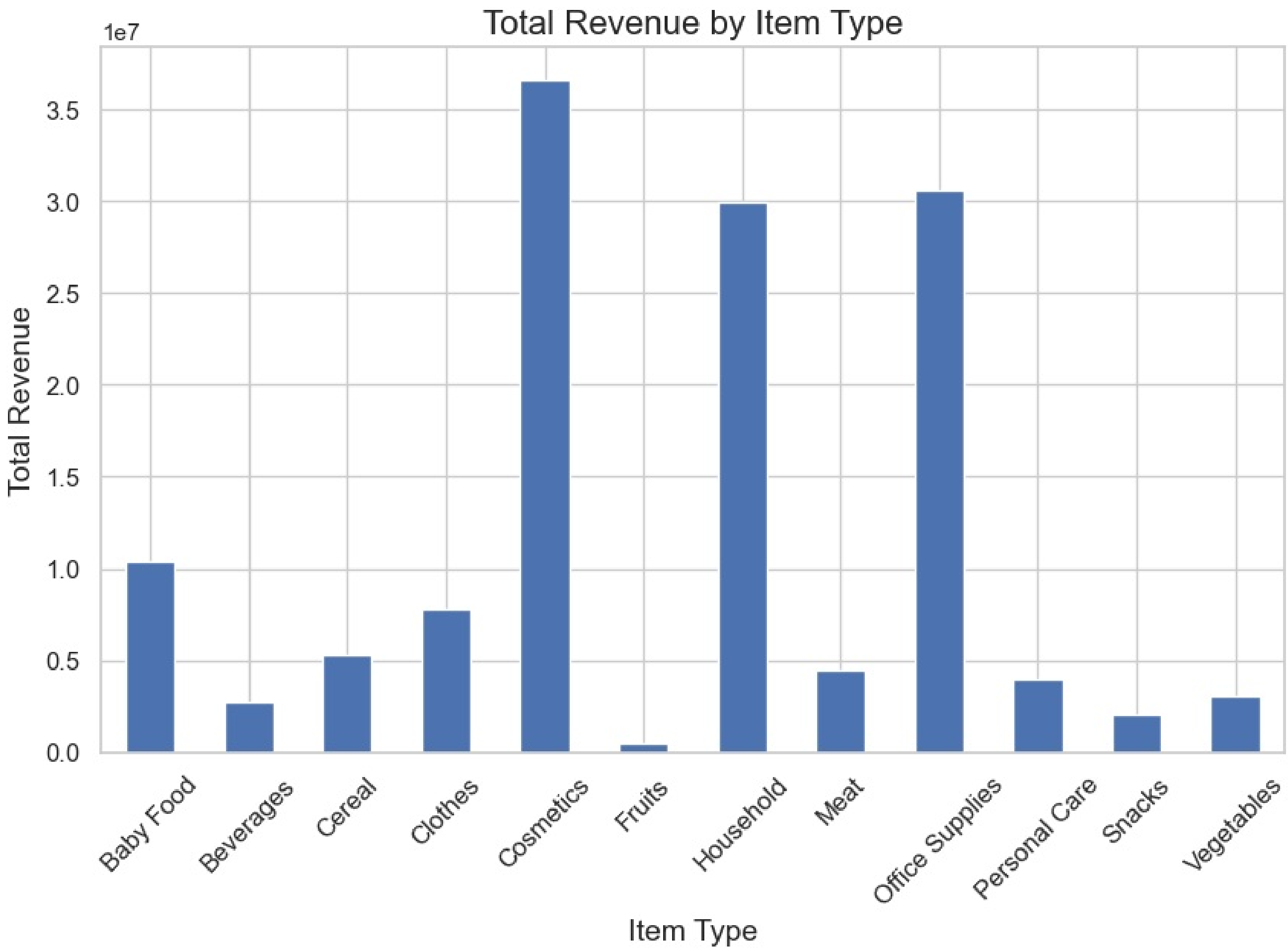


Bar Charts

In [ ]:

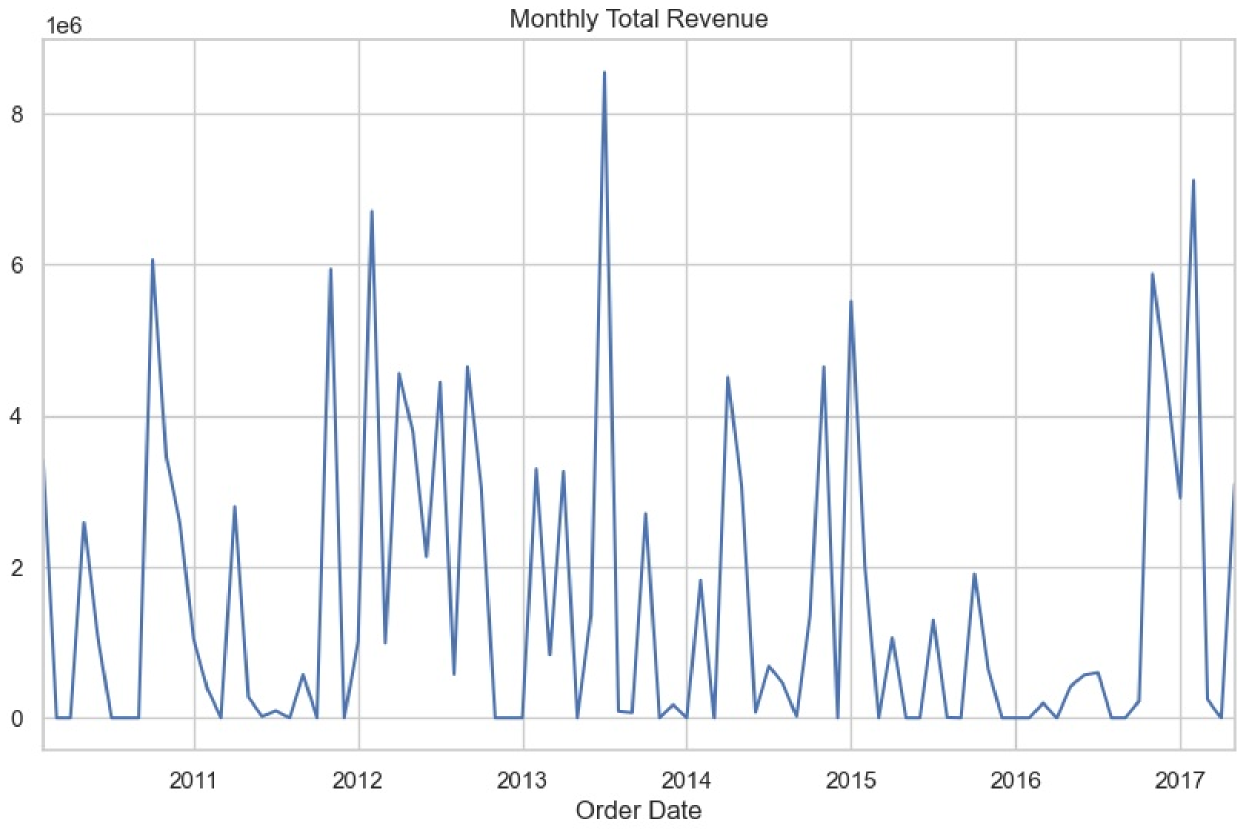
```
plt.figure(figsize=(10, 6))
data.groupby('Region')['Total Revenue'].sum().plot(kind='bar')
plt.title('Total Revenue by Region', fontsize=16)
plt.xlabel('Region', fontsize=14)
plt.ylabel('Total Revenue', fontsize=14)
plt.xticks(fontsize=12, rotation=45)
plt.yticks(fontsize=12)
plt.show()

plt.figure(figsize=(10, 6))
data.groupby('Item Type')['Total Revenue'].sum().plot(kind='bar')
plt.title('Total Revenue by Item Type', fontsize=16)
plt.xlabel('Item Type', fontsize=14)
plt.ylabel('Total Revenue', fontsize=14)
plt.xticks(fontsize=12, rotation=45)
plt.yticks(fontsize=12)
plt.show()
```


Total Revenue by Region
1e7
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Total Revenue
Asia
Australia and Oceania
Central America and the Caribbean
Europe
Middle East and North Africa
North America
Sub-Saharan Africa
Region


Total Revenue by Item Type
1e7
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Total Revenue
Baby Food
Beverages
Cereal
Clothes
Cosmetics
Fruits
Household
Meat
Office Supplies
Personal Care
Snacks
Vegetables
Item Type

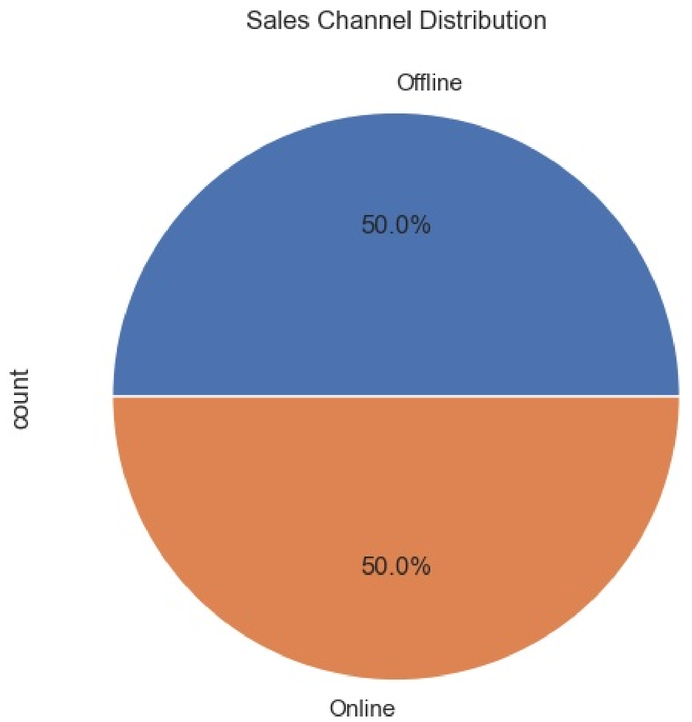
Line Charts

In [ ]:
```
plt.figure(figsize=(10, 6))
data['Total Revenue'].resample('M').sum().plot()
plt.title('Monthly Total Revenue')
plt.show()
```



Pie Charts

In [ ]:
```
plt.figure(figsize=(10, 6))
data['Sales Channel'].value_counts().plot(kind='pie', autopct='%1.1f%%')
plt.title('Sales Channel Distribution')
plt.show()
```



Insights and Patterns

In [ ]:
```
# Comparing Sales Channels
online_sales = data[data['Sales Channel'] == 'Online']['Total Revenue'].sum()
offline_sales = data[data['Sales Channel'] == 'Offline']['Total Revenue'].sum()
```

```
print(f"Online Sales Revenue: {online_sales}")
print(f"Offline Sales Revenue: {offline_sales}")

# Regional Performance
regional_performance = data.groupby('Region')['Total Revenue'].sum().sort_values(ascending=False)
print("Regional Performance:\n", regional_performance)

# Product Performance
product_performance = data.groupby('Item Type')['Total Revenue'].sum().sort_values(ascending=False)
print("Product Performance:\n", product_performance)
```

```
Online Sales Revenue: 58253959.11
Offline Sales Revenue: 79094809.2
Regional Performance:
 Region
Sub-Saharan Africa                   39672031.43
Europe                               33368932.11
Asia                                 21347091.02
Australia and Oceania                14094265.13
Middle East and North Africa         14052706.58
Central America and the Caribbean     9170385.49
North America                         5643356.55
Name: Total Revenue, dtype: float64
Product Performance:
 Item Type
Cosmetics          36601509.60
Office Supplies    30585380.07
Household          29889712.29
Baby Food          10350327.60
Clothes             7787292.80
Cereal              5322898.90
Meat                4503675.75
Personal Care       3980904.84
Vegetables          3089057.06
Beverages           2690794.60
Snacks              2080733.46
Fruits               466481.34
Name: Total Revenue, dtype: float64
```

Predictive Modelling - Sales Forecasting

In [ ]:
```
data['Order Date'] = pd.to_datetime(data.index)
data['Year'] = data['Order Date'].dt.year
data['Month'] = data['Order Date'].dt.month
data['Day'] = data['Order Date'].dt.day

# Feature Selection
features = ['Units Sold', 'Unit Price', 'Unit Cost', 'Total Cost']
X = data[features]
y = data['Total Revenue']

# Split the data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Build the model
model = LinearRegression()
model.fit(X_train, y_train)

# Make predictions
y_pred = model.predict(X_test)

# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
print(f'Mean Squared Error: {mse}')
```

```
Mean Squared Error: 6554207238.627237
```

Predictive Modelling - Profit Prediction

In [ ]:
```
y = data['Total Profit']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
print(f'Mean Squared Error for Profit Prediction: {mse}')
```

```
Mean Squared Error for Profit Prediction: 6554207238.635263
```