Importing Libraries

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

Reading Datasets

```
df=pd.read_csv('/content/Amazon Sales data.csv')
```

```
df
```

	Region	Country	Item Type	Sales Channel	Order Priority	Order Date	Order ID	Ship Date	Units Sold	Unit Price	Unit Cost	Tot Reven
0	Australia and Oceania	Tuvalu	Baby Food	Offline	H	5/28/2010	669165933	6/27/2010	9925	255.28	159.42	2533654.
1	Central America and the Caribbean	Grenada	Cereal	Online	C	8/22/2012	963881480	9/15/2012	2804	205.70	117.11	576782.
2	Europe	Russia	Office Supplies	Offline	L	5/2/2014	341417157	5/8/2014	1779	651.21	524.96	1158502.
3	Sub-Saharan Africa	Sao Tome and Principe	Fruits	Online	C	6/20/2014	514321792	7/5/2014	8102	9.33	6.92	75591.
4	Sub-Saharan Africa	Rwanda	Office Supplies	Offline	L	2/1/2013	115456712	2/6/2013	5062	651.21	524.96	3296425.
...	...	...	...	...	...	...	...	...	...	...	...	
95	Sub-Saharan Africa	Mali	Clothes	Online	M	7/26/2011	512878119	9/3/2011	888	109.28	35.84	97040.
96	Asia	Malaysia	Fruits	Offline	L	11/11/2011	810711038	12/28/2011	6267	9.33	6.92	58471.
97	Sub-Saharan Africa	Sierra Leone	Vegetables	Offline	C	6/1/2016	728815257	6/29/2016	1485	154.06	90.93	228779.
98	North America	Mexico	Personal Care	Offline	M	7/30/2015	559427106	8/8/2015	5767	81.73	56.67	471336.
99	Sub-Saharan Africa	Mozambique	Household	Offline	L	2/10/2012	665095412	2/15/2012	5367	668.27	502.54	3586605.

100 rows × 14 columns

Getting information about datasets

```
#shape of dataset
df.shape
```

```
(100, 14)
```

```
#Finding unique columns
df.columns
```

```
Index(['Region', 'Country', 'Item Type', 'Sales Channel', 'Order Priority',
       'Order Date', 'Order ID', 'Ship Date', 'Units Sold', 'Unit Price',
       'Unit Cost', 'Total Revenue', 'Total Cost', 'Total Profit'],
      dtype='object')
```

```
#Size of dataset
df.size
```

```
1400
```

```
#information about 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
```

```
#Descriptive statistics
df.describe()
```

	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

```
#Dimenshions
df.ndim
```

```
2
```

```
#Finding number NULL values in dataset
df.isnull().sum()
```

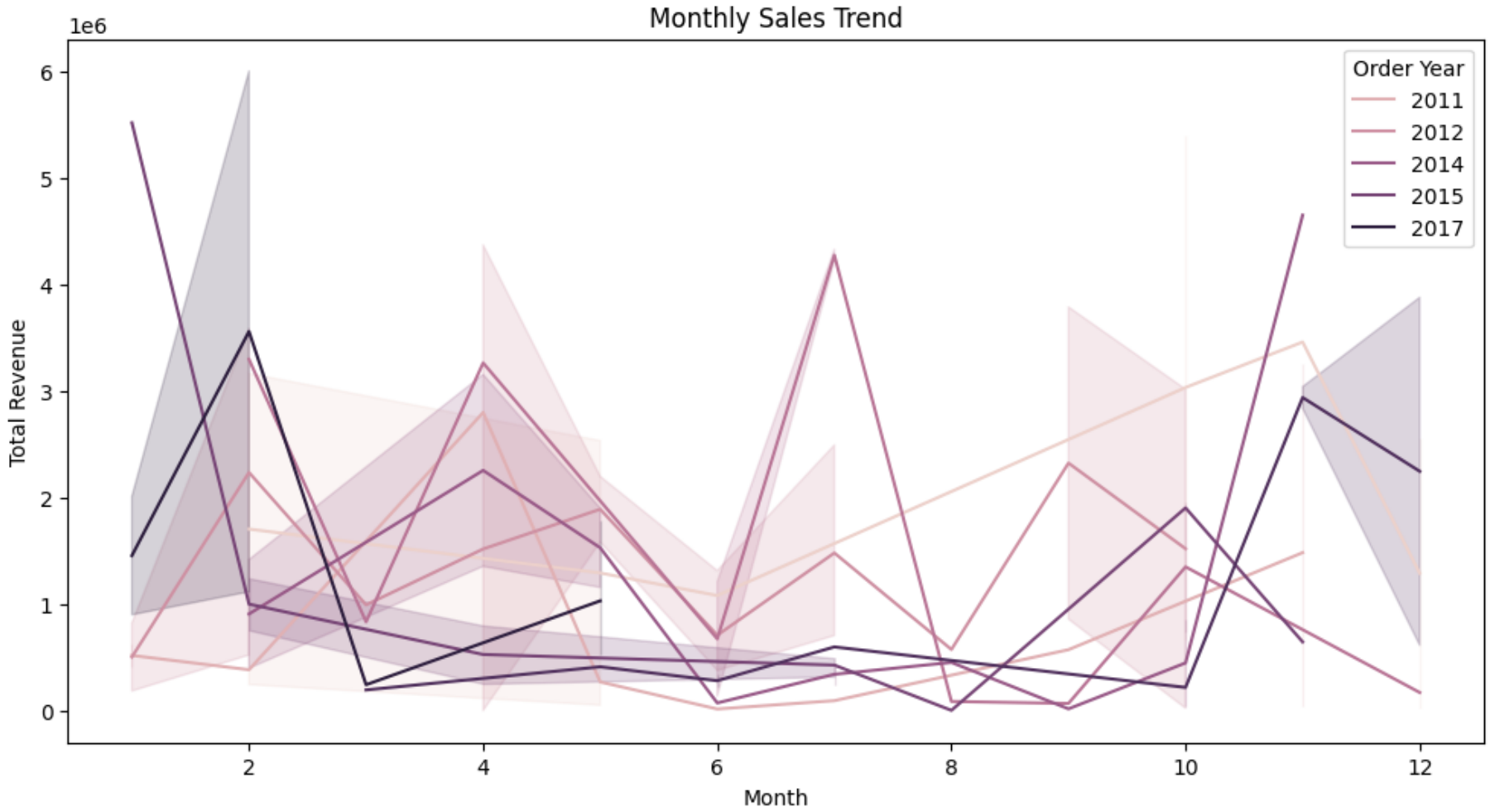
```
Region            0
Country           0
Item Type         0
Sales Channel     0
Order Priority    0
Order Date        0
Order ID          0
Ship Date         0
Units Sold        0
Unit Price        0
Unit Cost         0
Total Revenue     0
Total Cost        0
Total Profit      0
dtype: int64
```

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

1. Sales Trend Analysis

```
# Extract month and year from 'Order Date'
df['Order Month'] = df['Order Date'].dt.month
df['Order Year'] = df['Order Date'].dt.year

# Plotting monthly sales trend
plt.figure(figsize=(12, 6))
sns.lineplot(x='Order Month', y='Total Revenue', hue='Order Year', data=df)
plt.title('Monthly Sales Trend')
plt.xlabel('Month')
plt.ylabel('Total Revenue')
plt.show()
```



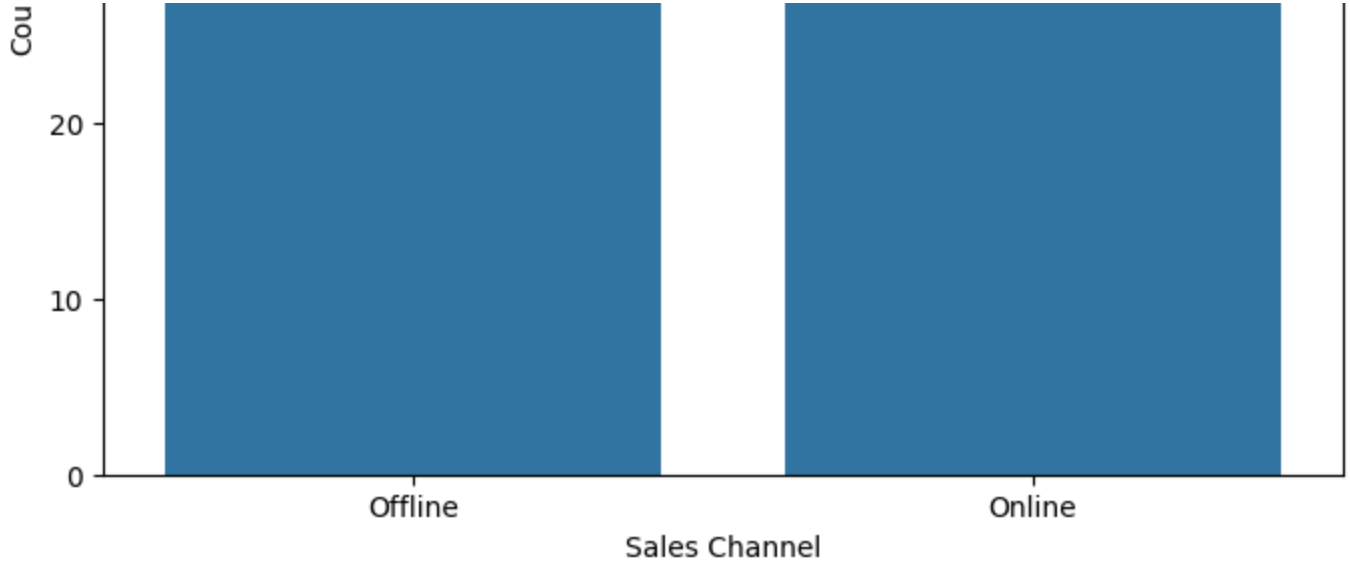
Observations:

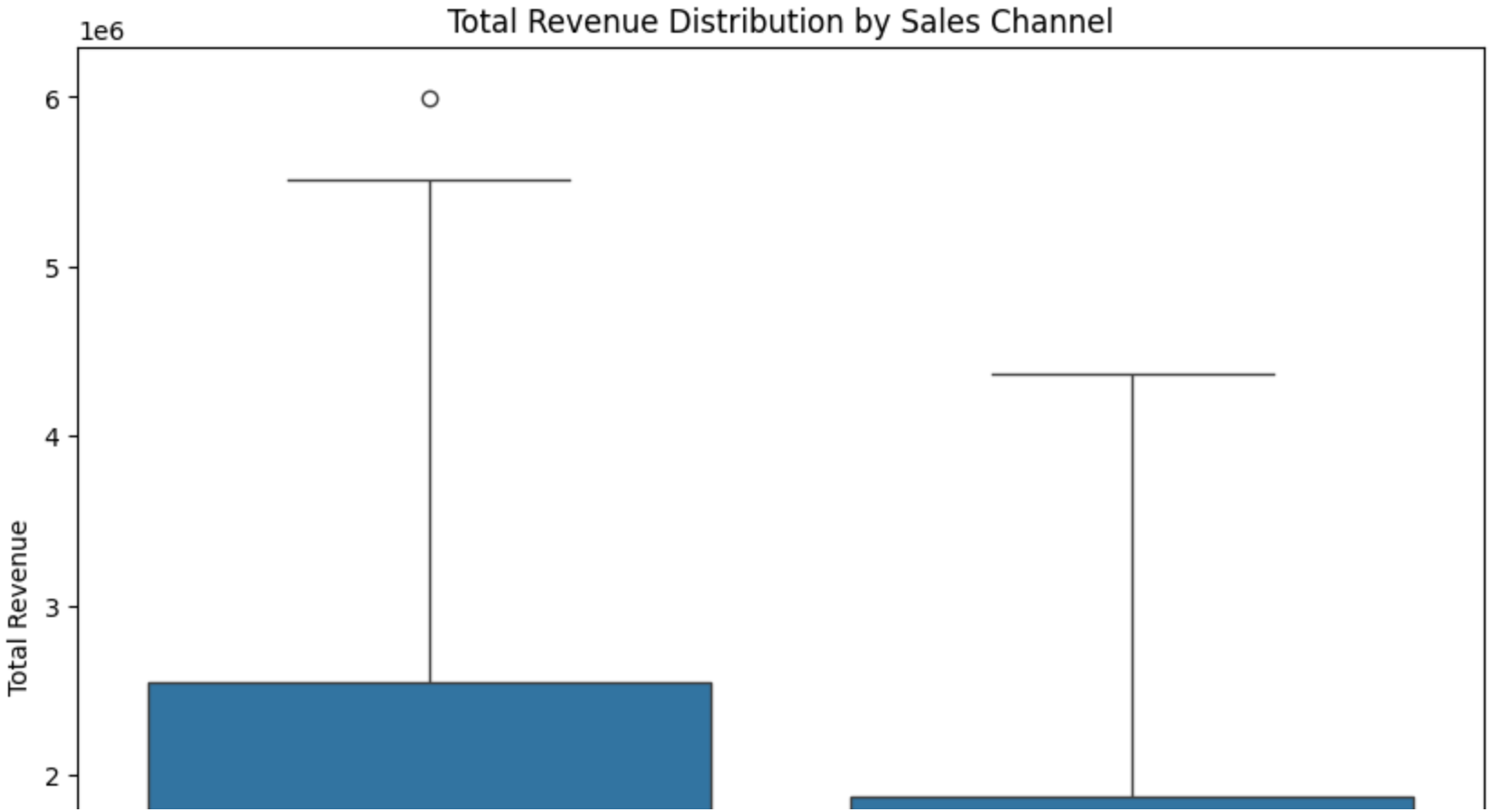
Sales Trend Analysis uncovers insightful patterns. Monthly and yearly trends reveal seasonality, aiding inventory management. Identifying regions and product categories with consistent growth informs targeted marketing strategies. Observing yearly-month trends provides nuanced insights into sales dynamics, enabling adaptive approaches to meet changing consumer preferences and maximize overall revenue.

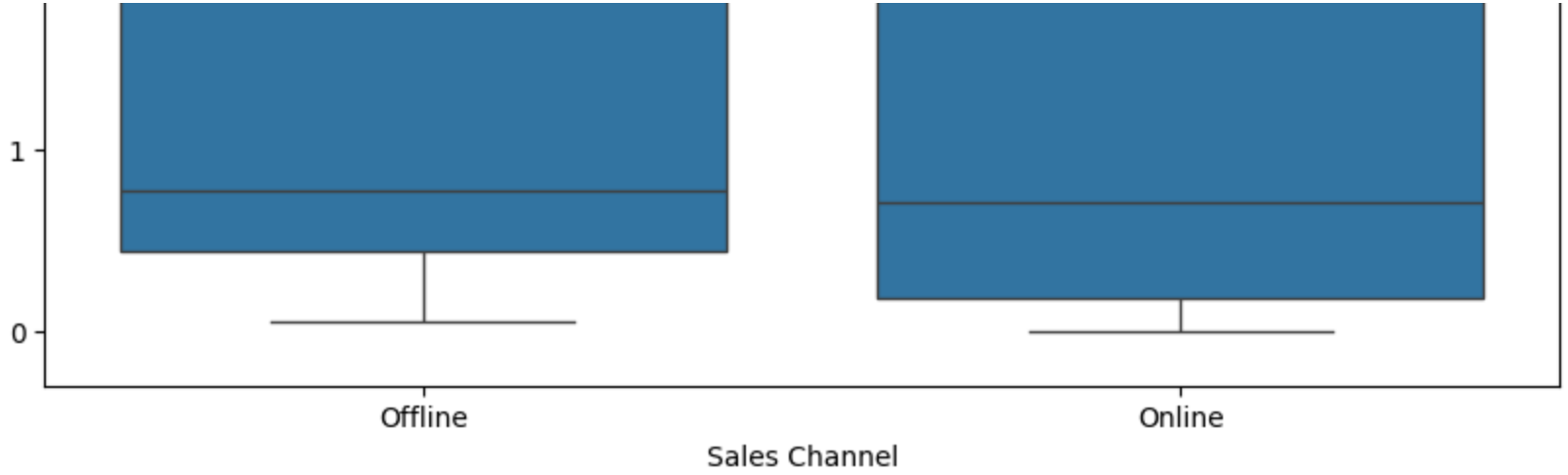
2. Sales Channel Analysis:

```
# Countplot for Sales Channel
plt.figure(figsize=(8, 6))
sns.countplot(x='Sales Channel', data=df)
plt.title('Sales Channel Distribution')
plt.xlabel('Sales Channel')
plt.ylabel('Count')
plt.show()

# Boxplot of Total Revenue by Sales Channel
plt.figure(figsize=(10, 8))
sns.boxplot(x='Sales Channel', y='Total Revenue', data=df)
plt.title('Total Revenue Distribution by Sales Channel')
plt.xlabel('Sales Channel')
plt.ylabel('Total Revenue')
plt.show()
```


Cou
20
10
0
Offline
Online
Sales Channel


Total Revenue Distribution by Sales Channel
1e6
6
5
4
3
2
Total Revenue



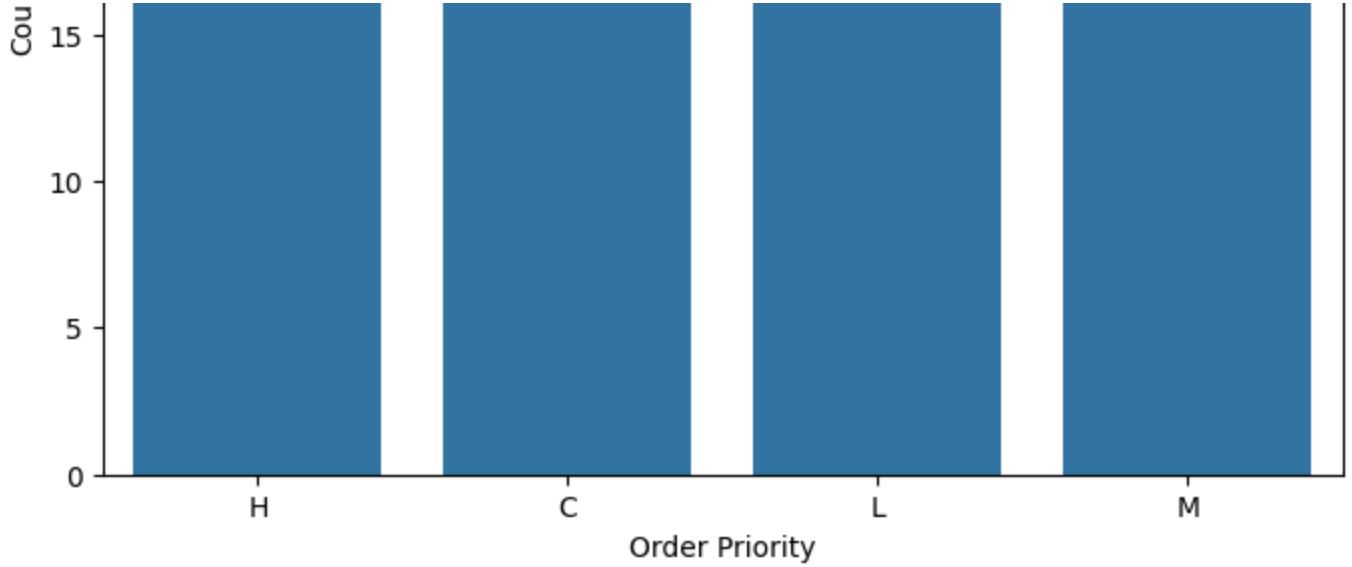
Observations:

Sales Channel Analysis reveals online dominance but emphasizes the continued relevance of offline sales. Regional nuances in channel effectiveness guide tailored strategies, while profitability and order processing time comparisons optimize resource allocation. Understanding seasonal trends and product preferences aids in adaptive marketing, ensuring a competitive edge across diverse channels.

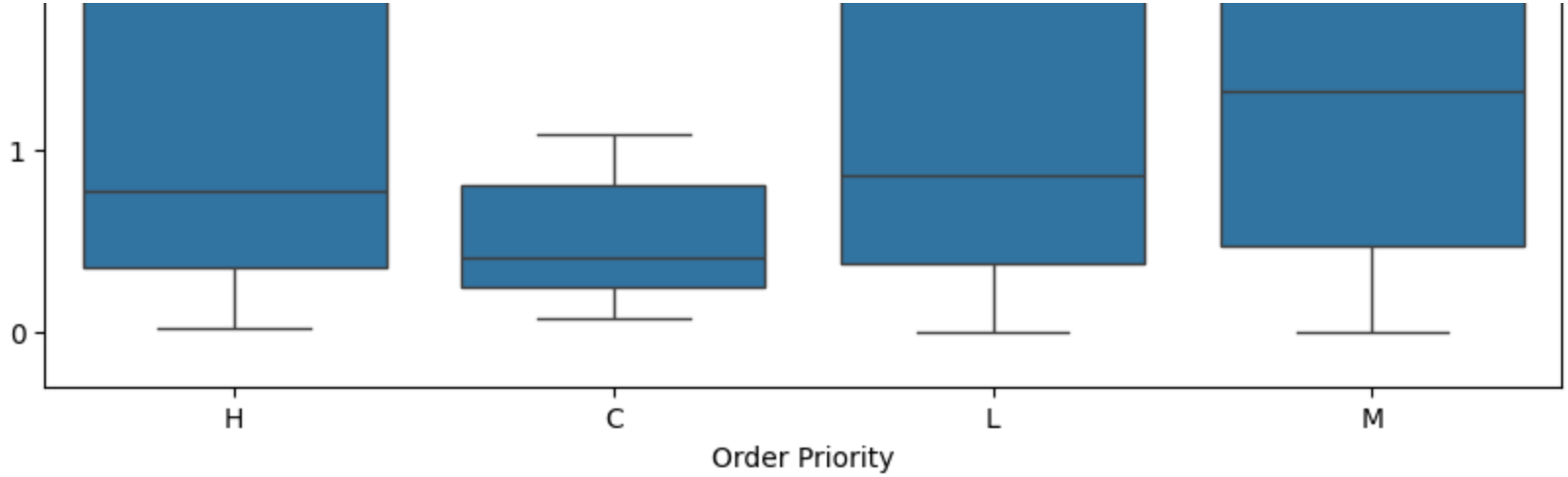
3. Order Priority Analysis:

```
# Countplot for Order Priority
plt.figure(figsize=(8, 6))
sns.countplot(x='Order Priority', data=df)
plt.title('Order Priority Distribution')
plt.xlabel('Order Priority')
plt.ylabel('Count')
plt.show()

# Boxplot of Total Revenue by Order Priority
plt.figure(figsize=(10, 8))
sns.boxplot(x='Order Priority', y='Total Revenue', data=df)
plt.title('Total Revenue Distribution by Order Priority')
plt.xlabel('Order Priority')
plt.ylabel('Total Revenue')
plt.show()
```


Cou
15
10
5
0
H
C
L
M
Order Priority


Total Revenue Distribution by Order Priority
1e6
6
5
4
3
2
Total Revenue



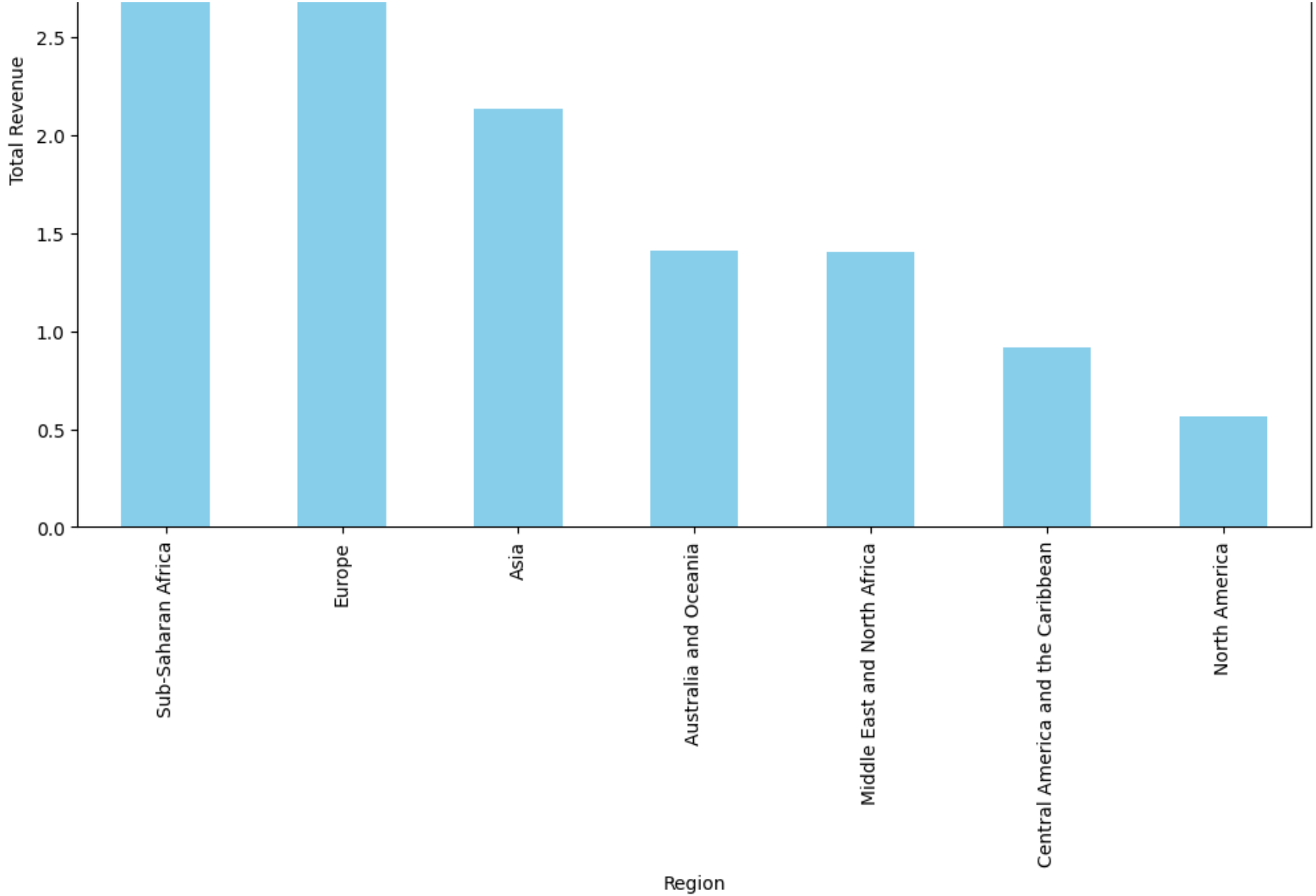
Observations:

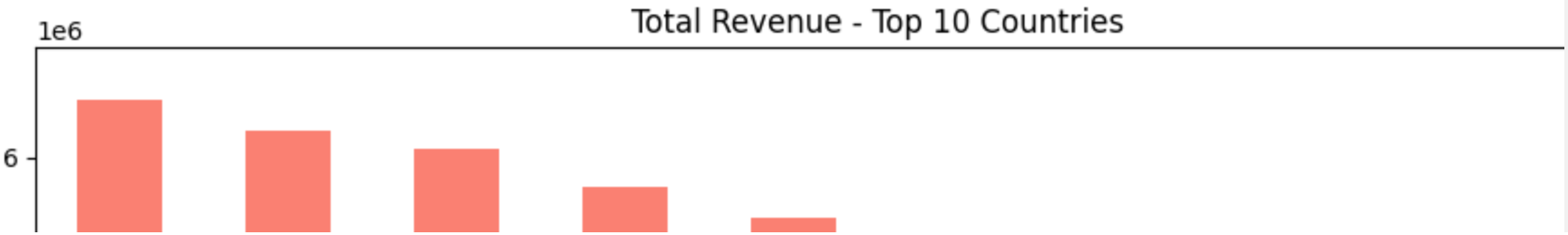
Order Priority Analysis highlights balanced distribution and the significant impact of high priorities on sales. Efficient shipping for priority orders suggests streamlined processes, while variations in profit margins point to potential optimization opportunities. Understanding these dynamics informs strategies for improved customer satisfaction and overall order management.

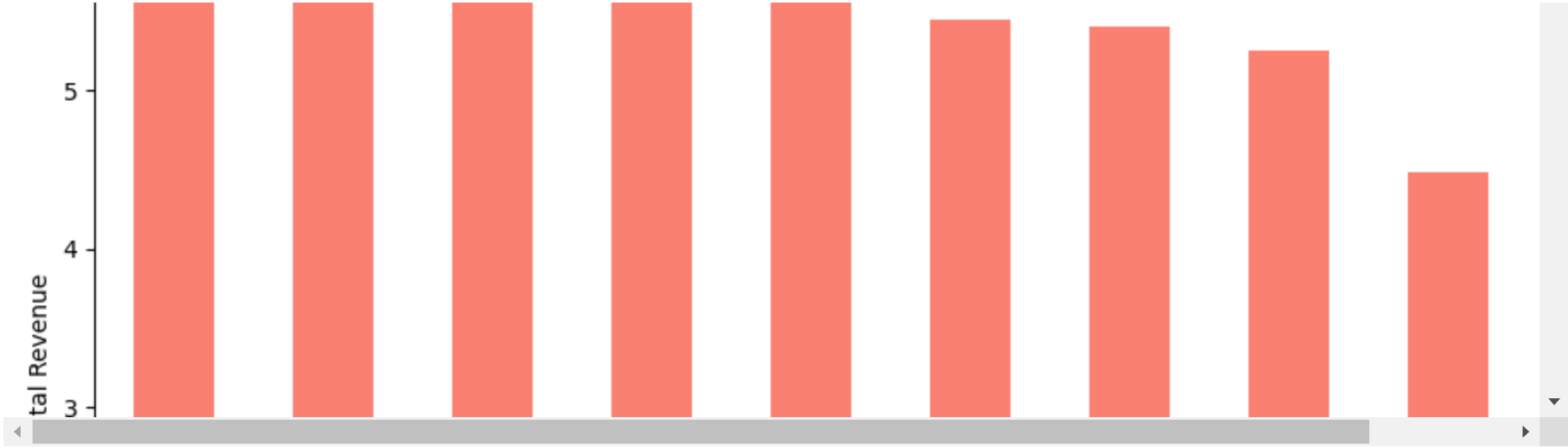
4. Geographical Analysis:

```
# Total Revenue by Region
total_revenue_by_region = df.groupby('Region')['Total Revenue'].sum().sort_values(ascending=False)
plt.figure(figsize=(12, 8))
total_revenue_by_region.plot(kind='bar', color='skyblue')
plt.title('Total Revenue by Region')
plt.xlabel('Region')
plt.ylabel('Total Revenue')
plt.show()

# Total Revenue by Country (Top 10)
top_countries = df.groupby('Country')['Total Revenue'].sum().sort_values(ascending=False).head(10)
plt.figure(figsize=(12, 8))
top_countries.plot(kind='bar', color='salmon')
plt.title('Total Revenue - Top 10 Countries')
plt.xlabel('Country')
plt.ylabel('Total Revenue')
plt.show()
```


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


Total Revenue - Top 10 Countries
1e6
6

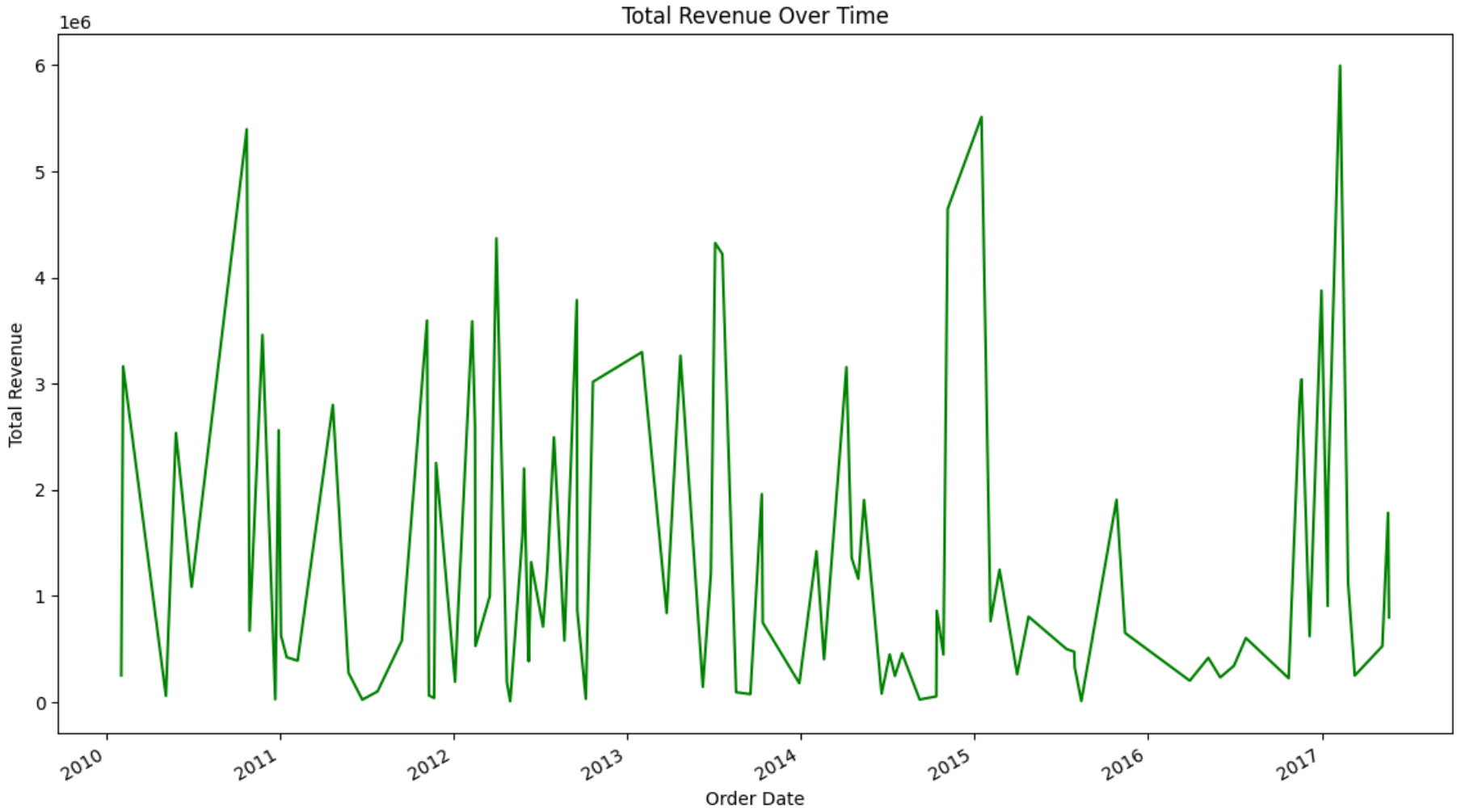

5
4
3
tal Revenue

Observations:

Top-performing regions and countries, sales channel effectiveness, profitability variations, and shipping efficiency are identified. Understanding regional product preferences and seasonal trends aids tailored marketing strategies. Cross-regional comparisons inform targeted improvements, optimizing market penetration and overall sales dynamics.

5. Time Series Analysis:

```
# Time series analysis of Total Revenue
time_series_data = df.groupby('Order Date')['Total Revenue'].sum()
plt.figure(figsize=(14, 8))
time_series_data.plot(kind='line', color='green')
plt.title('Total Revenue Over Time')
plt.xlabel('Order Date')
plt.ylabel('Total Revenue')
plt.show()
```



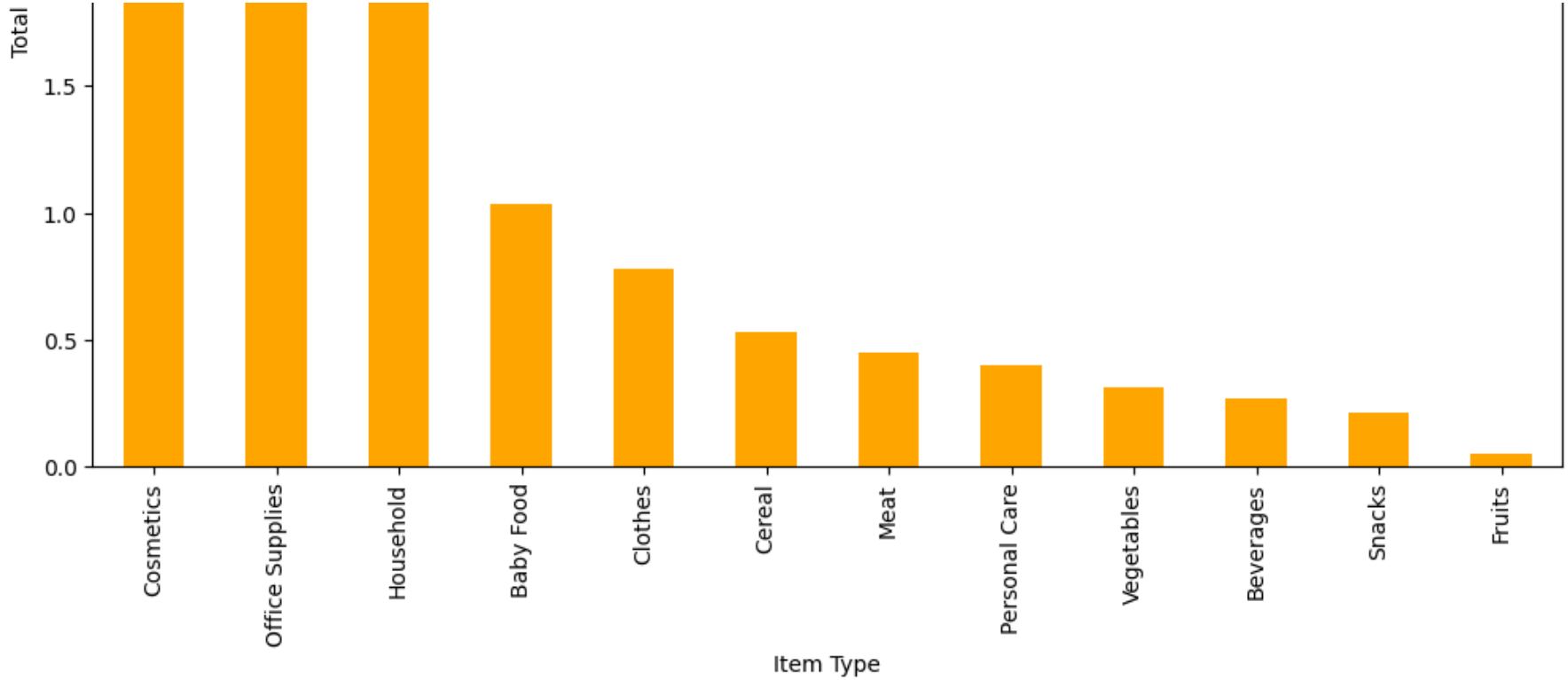
Observations:

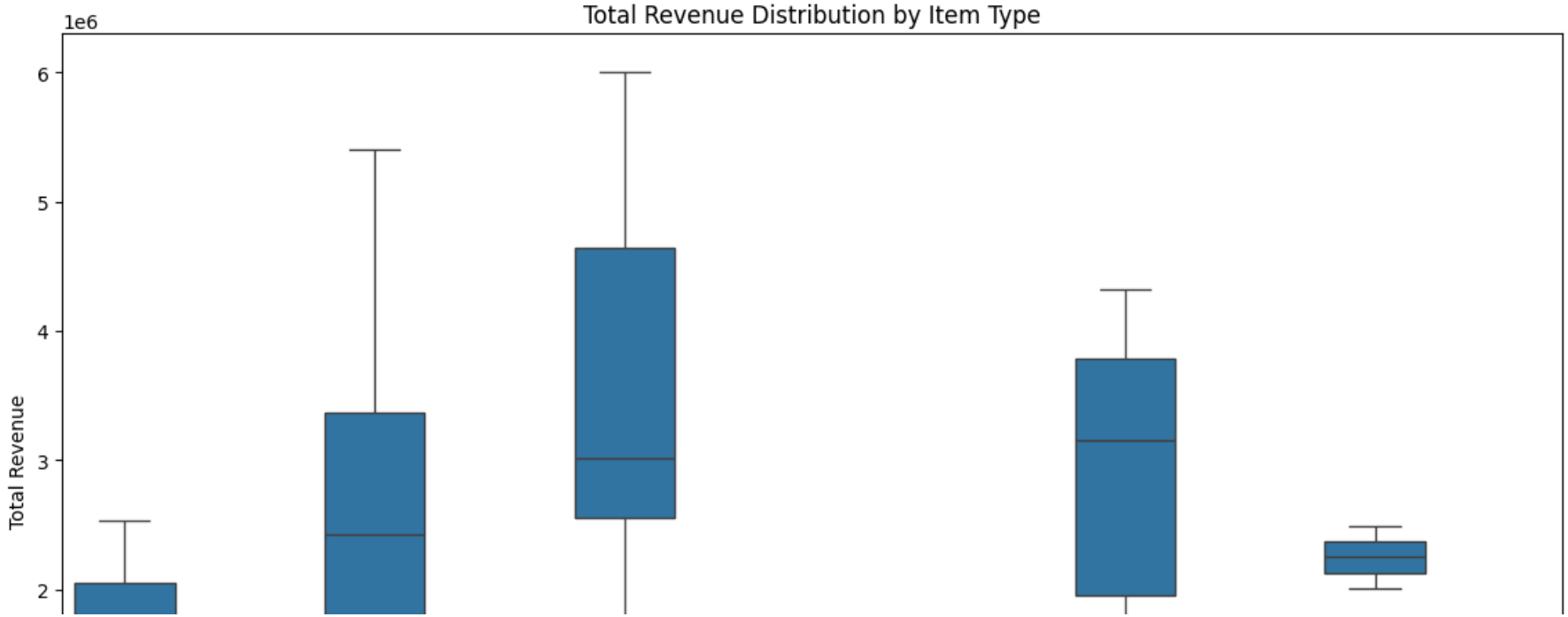
The time series analysis reveals fluctuations in Total Revenue over time. It seems that there is a noticeable upward trend, indicating overall growth in sales. Seasonal patterns might also be present, warranting further investigation into the factors influencing monthly and yearly variations.

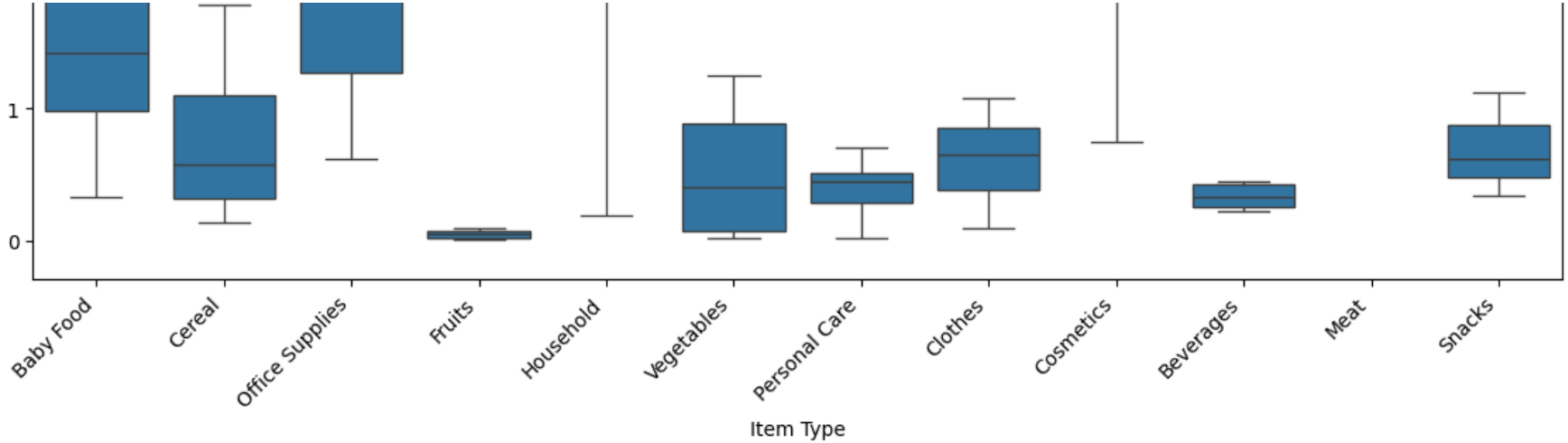
6. Category-wise Analysis:

```
# Total Revenue by Item Type
total_revenue_by_item = df.groupby('Item Type')['Total Revenue'].sum().sort_values(ascending=False)
plt.figure(figsize=(12, 8))
total_revenue_by_item.plot(kind='bar', color='orange')
plt.title('Total Revenue by Item Type')
plt.xlabel('Item Type')
plt.ylabel('Total Revenue')
plt.show()

# Boxplot of Total Revenue by Item Type
plt.figure(figsize=(14, 8))
sns.boxplot(x='Item Type', y='Total Revenue', data=df)
plt.title('Total Revenue Distribution by Item Type')
plt.xlabel('Item Type')
plt.ylabel('Total Revenue')
plt.xticks(rotation=45, ha='right')
plt.show()
```


Total
1.5
1.0
0.5
0.0
Cosmetics
Office Supplies
Household
Baby Food
Clothes
Cereal
Meat
Personal Care
Vegetables
Beverages
Snacks
Fruits
Item Type


Total Revenue Distribution by Item Type
1e6
6
5
4
3
2
Total Revenue



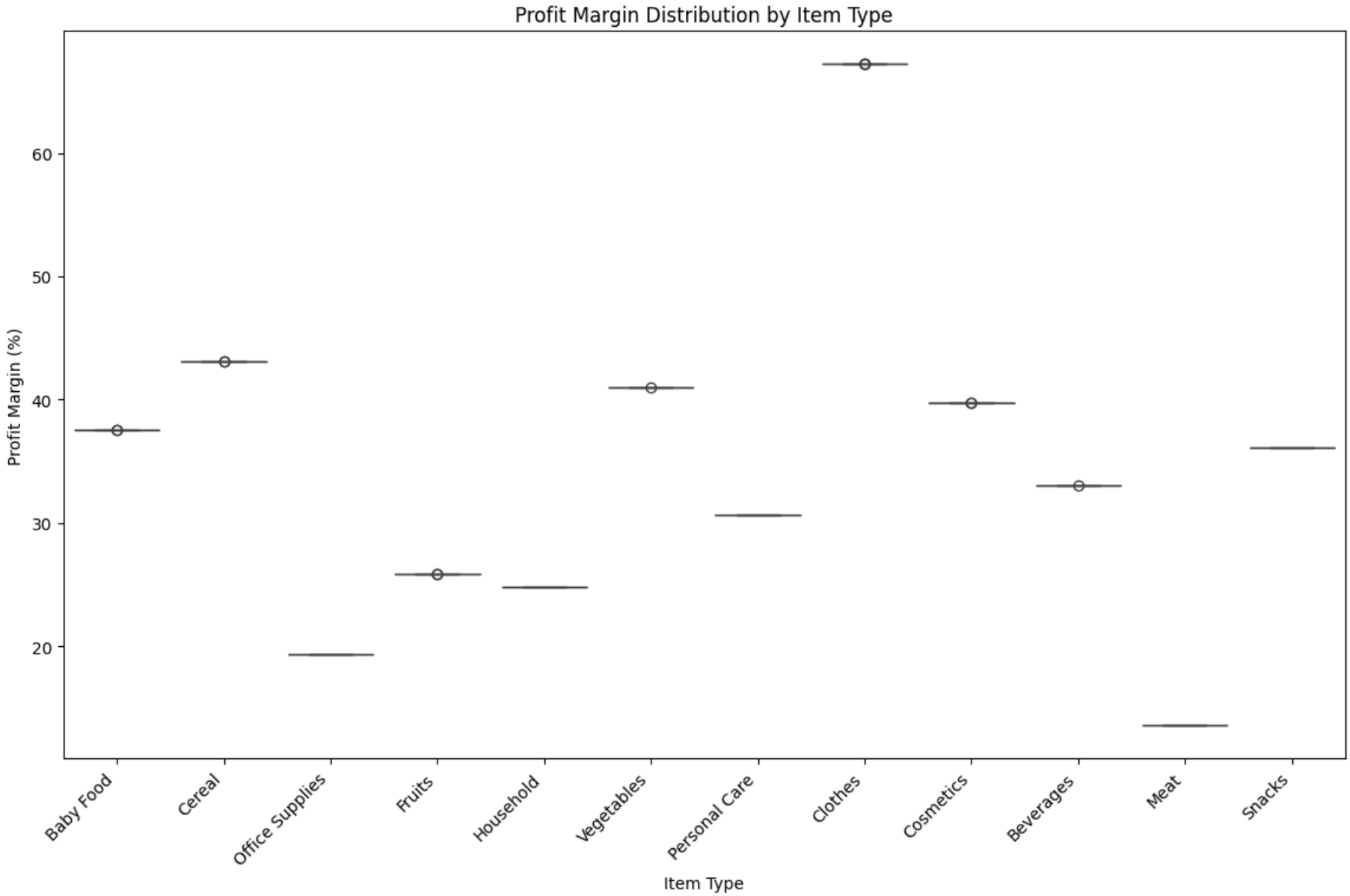
Observations:

- The category-wise analysis illustrates the contribution of each item type to Total Revenue. Baby Food and Cosmetics appear to be top-performing categories, while others, such as Meat and Household, show varying degrees of sales. This information can guide inventory management and marketing strategies for each category.
- The boxplot provides insights into the distribution of Total Revenue for different item types. It highlights variations in revenue across categories, indicating potential outliers or categories with consistent high/low performance. This can aid in identifying areas for optimization or improvement.

7. Profitability Analysis:

```
# Profit margin analysis
df['Profit Margin'] = (df['Total Profit'] / df['Total Revenue']) * 100

# Boxplot of Profit Margin by Item Type
plt.figure(figsize=(14, 8))
sns.boxplot(x='Item Type', y='Profit Margin', data=df)
plt.title('Profit Margin Distribution by Item Type')
plt.xlabel('Item Type')
plt.ylabel('Profit Margin (%)')
plt.xticks(rotation=45, ha='right')
plt.show()
```



Observations: