In [1]:

```
# Analysing Amazon Sales data
```

In [9]:

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

In [10]:

```
# load the dataset
sales = pd.read_csv(r'C:\Desktop\DataAnalytics\UnifiedMentor\Amazon1\sales.
csv')
```

In [11]:

```
# look at the data
sales.head()
```

Out[11]:

	Region	Country	Item Type	Sales Channel	Order Priority	Order Date	Order ID	Ship Date	Units Sold	U P
0	Australia and Oceania	Tuvalu	Baby Food	Offline	H	5/28/2010	669165933	6/27/2010	9925	255
1	Central America and the Caribbean	Grenada	Cereal	Online	C	8/22/2012	963881480	9/15/2012	2804	205
2	Europe	Russia	Office Supplies	Offline	L	5/2/2014	341417157	5/8/2014	1779	651
3	Sub-Saharan Africa	Sao Tome and Principe	Fruits	Online	C	6/20/2014	514321792	7/5/2014	8102	9
4	Sub-Saharan Africa	Rwanda	Office Supplies	Offline	L	2/1/2013	115456712	2/6/2013	5062	651

In [12]:

```
# Get the dimensions of the dataframe
sales.shape
```

Out[12]:

```
(100, 14)
```

In [13]:

```
# Get the row names of the dataframe
sales.index
```

Out[13]:

```
RangeIndex(start=0, stop=100, step=1)
```

In [14]:

```
# Get the column names of the dataframe
sales.columns
```

Out[14]:

```
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')
```

In [15]:

```
# Look at basic information about the dataframe

sales.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 [16]:

```
# checking null values
sales.isnull().sum()
```

Out[16]:

```
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
```

In [17]:

```
# Basic Statistics
sales.describe()
```

Out[17]:

	Order ID	Units Sold	Unit Price	Unit Cost	Total Revenue	Total Cost	Tot
count	1.000000e+02	100.000000	100.000000	100.000000	1.000000e+02	1.000000e+02	1.000
mean	5.550204e+08	5128.710000	276.761300	191.048000	1.373488e+06	9.318057e+05	4.416
std	2.606153e+08	2794.484562	235.592241	188.208181	1.460029e+06	1.083938e+06	4.385
min	1.146066e+08	124.000000	9.330000	6.920000	4.870260e+03	3.612240e+03	1.258
25%	3.389225e+08	2836.250000	81.730000	35.840000	2.687212e+05	1.688680e+05	1.214
50%	5.577086e+08	5382.500000	179.880000	107.275000	7.523144e+05	3.635664e+05	2.907
75%	7.907551e+08	7369.000000	437.200000	263.330000	2.212045e+06	1.613870e+06	6.358
max	9.940222e+08	9925.000000	668.270000	524.960000	5.997055e+06	4.509794e+06	1.719

In [18]:

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

In [19]:

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

In [20]:

```
# Extract year_month for better trend analysis
sales['Year_Month'] = sales['Order Date'].dt.to_period('M')
```

In [21]:

```
# Group by month, year, and year_month and calculate total sales
monthly_sales = sales.groupby(sales['Order Date'].dt.to_period('M'))['Total
Revenue'].sum()
yearly_sales = sales.groupby(sales['Order Date'].dt.year)['Total Revenue'].
sum()
yearly_monthly_sales = sales.groupby(['Order Year', 'Order Month'])['Total
Revenue'].sum()
```

In [22]:

```
# Print the results
print("Monthly Sales:")
print(monthly_sales)
```

```
Monthly Sales:
Order Date
2010-02    3410661.12
2010-05    2587973.26
2010-06    1082418.40
2010-10    6064933.75
2010-11    3458252.00
2010-12    2581786.39
2011-01    1042225.35
2011-02     387002.20
2011-04    2798046.49
2011-05     272410.45
2011-06      19103.44
2011-07      97040.64
2011-09     574951.92
2011-11    5938385.58
2012-01    1012884.00
2012-02    6707849.42
2012-03     994765.42
2012-04    4556012.38
2012-05    3782781.82
2012-06    2132075.27
2012-07    4445093.92
2012-08     576782.80
2012-09    4648152.72
2012-10    3042246.77
2013-02    3296425.02
2013-03     835759.10
2013-04    3262562.10
2013-06    1352867.40
2013-07    8545511.20
2013-08      89623.98
2013-09      71253.21
2013-10    2702770.40
2013-12     173676.25
2014-02    1819660.25
2014-04    4510578.10
2014-05    3060338.59
2014-06      75591.66
2014-07     688641.85
2014-08     455479.04
2014-09      20404.71
2014-10    1352370.65
2014-11    4647149.58
2015-01    5513227.50
2015-02    2003911.12
2015-04    1059987.26
2015-07    1292409.45
2015-08       6279.09
2015-10    1904138.04
2015-11     648030.40
2016-03     197883.40
2016-05     414371.10
2016-06     568269.60
2016-07     600821.44
2016-10     221117.00
2016-11    5876405.20
2016-12    4493999.48
2017-01    2914130.27
2017-02    7115008.64
2017-03     246415.95
```

```
2017-05    3097864.77
Freq: M, Name: Total Revenue, dtype: float64
```

In [23]:

```
print("\nYearly Sales:")
print(yearly_sales)
```

```
Yearly Sales:
Order Date
2010    19186024.92
2011    11129166.07
2012    31898644.52
2013    20330448.66
2014    16630214.43
2015    12427982.86
2016    12372867.22
2017    13373419.63
Name: Total Revenue, dtype: float64
```

In [24]:

```
print("\nYearly-Monthly Sales:")
print(yearly_monthly_sales)
```

```
Yearly-Monthly Sales:
Order Year  Order Month
2010        2              3410661.12
            5              2587973.26
            6              1082418.40
            10             6064933.75
            11             3458252.00
            12             2581786.39
2011        1              1042225.35
            2               387002.20
            4              2798046.49
            5               272410.45
            6                19103.44
            7                97040.64
            9               574951.92
            11             5938385.58
2012        1              1012884.00
            2              6707849.42
            3               994765.42
            4              4556012.38
            5              3782781.82
            6              2132075.27
            7              4445093.92
            8               576782.80
            9              4648152.72
            10             3042246.77
2013        2              3296425.02
            3               835759.10
            4              3262562.10
            6              1352867.40
            7              8545511.20
            8                89623.98
            9                71253.21
            10             2702770.40
            12              173676.25
2014        2              1819660.25
            4              4510578.10
            5              3060338.59
            6                75591.66
            7               688641.85
            8               455479.04
            9                20404.71
            10             1352370.65
            11             4647149.58
2015        1              5513227.50
            2              2003911.12
            4              1059987.26
            7              1292409.45
            8                 6279.09
            10             1904138.04
            11              648030.40
2016        3               197883.40
            5               414371.10
            6               568269.60
            7               600821.44
            10              221117.00
            11             5876405.20
            12             4493999.48
2017        1              2914130.27
            2              7115008.64
            3               246415.95
```

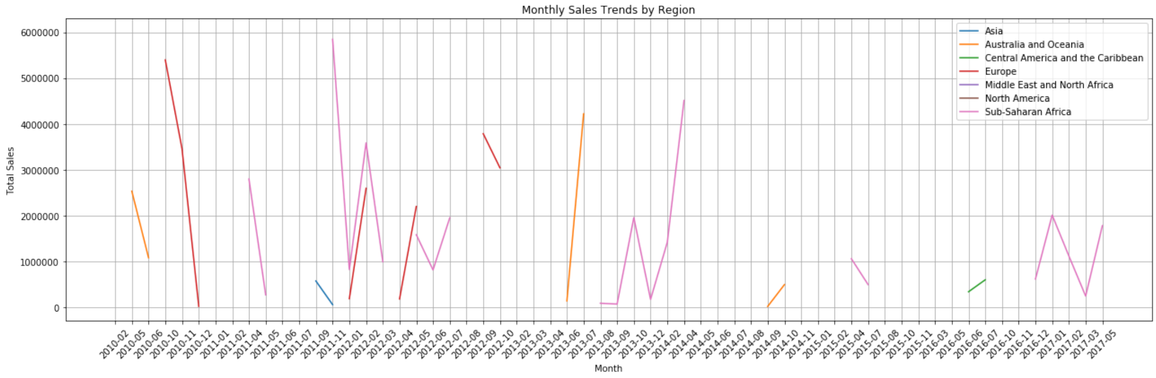
```
5            3097864.77
Name: Total Revenue, dtype: float64
```

In [25]:

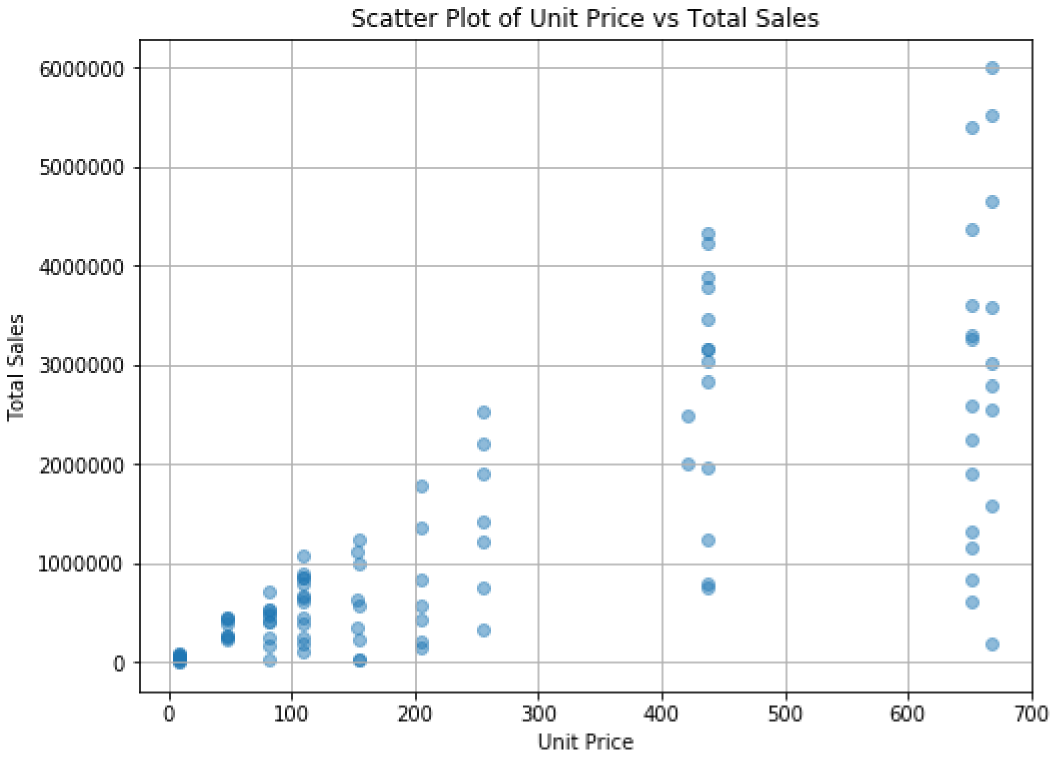
```
# Group by month and region, and calculate total sales
monthly_sales_region = sales.groupby([sales['Order Date'].dt.to_period
('M'), 'Region'])['Total Revenue'].sum().unstack()

monthly_sales_region.index = monthly_sales_region.index.strftime('%Y-%m')
```

In [26]:

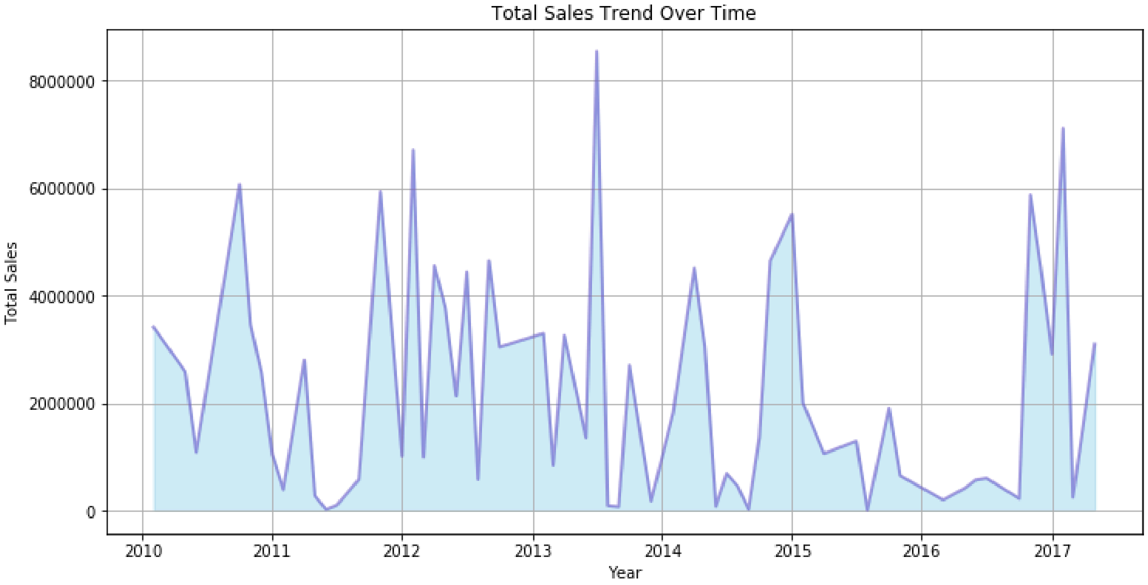
```
plt.figure(figsize=(18, 6))
for region in monthly_sales_region.columns:
    plt.plot(monthly_sales_region.index, monthly_sales_region[region], labe
l=region)

plt.title('Monthly Sales Trends by Region')
plt.xlabel('Month')
plt.ylabel('Total Sales')
plt.legend()
plt.grid(True)
plt.xticks(rotation=45)  # Rotate x-axis labels for better readability
plt.tight_layout()  # Adjust layout to prevent clipping of labels
plt.show()
```



```
In [27]: # Plot the scatter plot of unit price Vs Total Sales
         plt.figure(figsize=(8, 6))
         plt.scatter(sales['Unit Price'], sales['Total Revenue'], alpha=0.5)
         plt.title('Scatter Plot of Unit Price vs Total Sales')
         plt.xlabel('Unit Price')
         plt.ylabel('Total Sales')
         plt.grid(True)
         plt.show()
```



In [28]:
```
# Total sales Trend over time
# Group by month and calculate total sales
monthly_total_sales = sales.groupby(sales['Order Date'].dt.to_period('M'))
['Total Revenue'].sum()

# Plot the area chart
plt.figure(figsize=(12, 6))
plt.fill_between(monthly_total_sales.index.to_timestamp(), monthly_total_sa
les.values, color="skyblue", alpha=0.4)
plt.plot(monthly_total_sales.index.to_timestamp(), monthly_total_sales.valu
es, color="Slateblue", alpha=0.6, linewidth=2)

plt.title('Total Sales Trend Over Time')
plt.xlabel('Year')
plt.ylabel('Total Sales')
plt.grid(True)
plt.show()
```

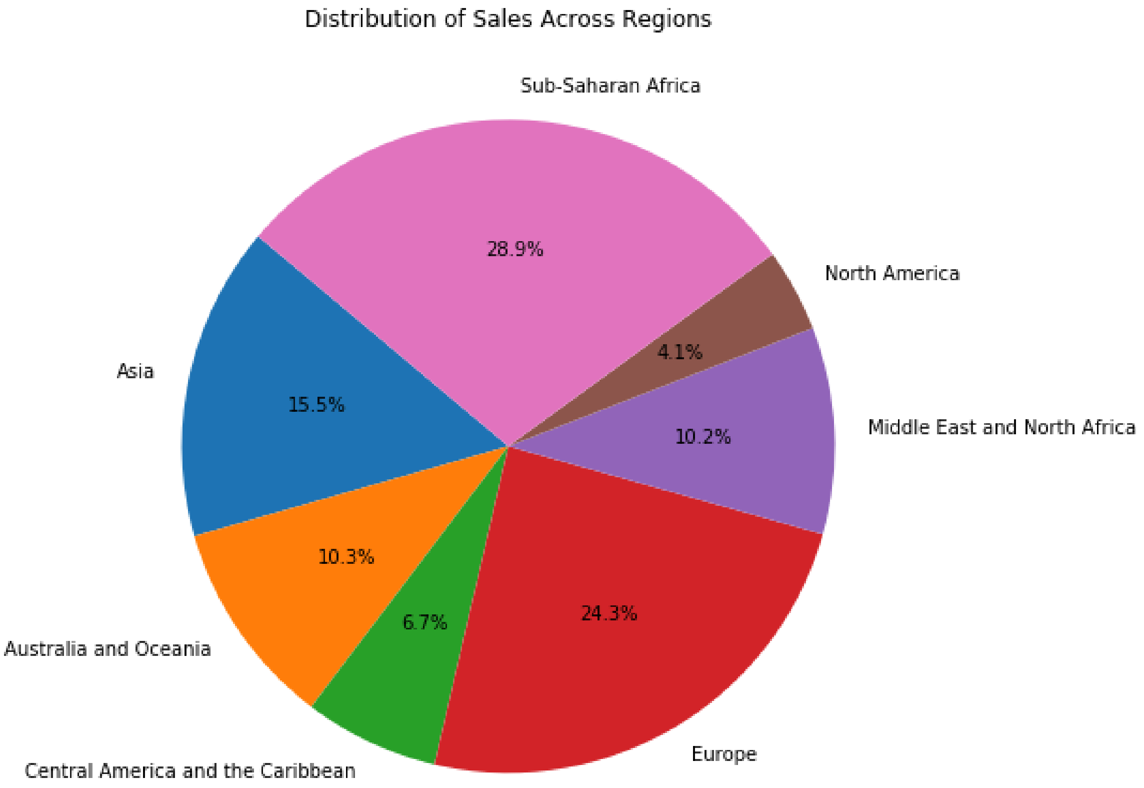


In [29]:
```
#Observation : Total sales is high in the mid of 2013
```

In [30]:
```
# Observation
```

In [31]:

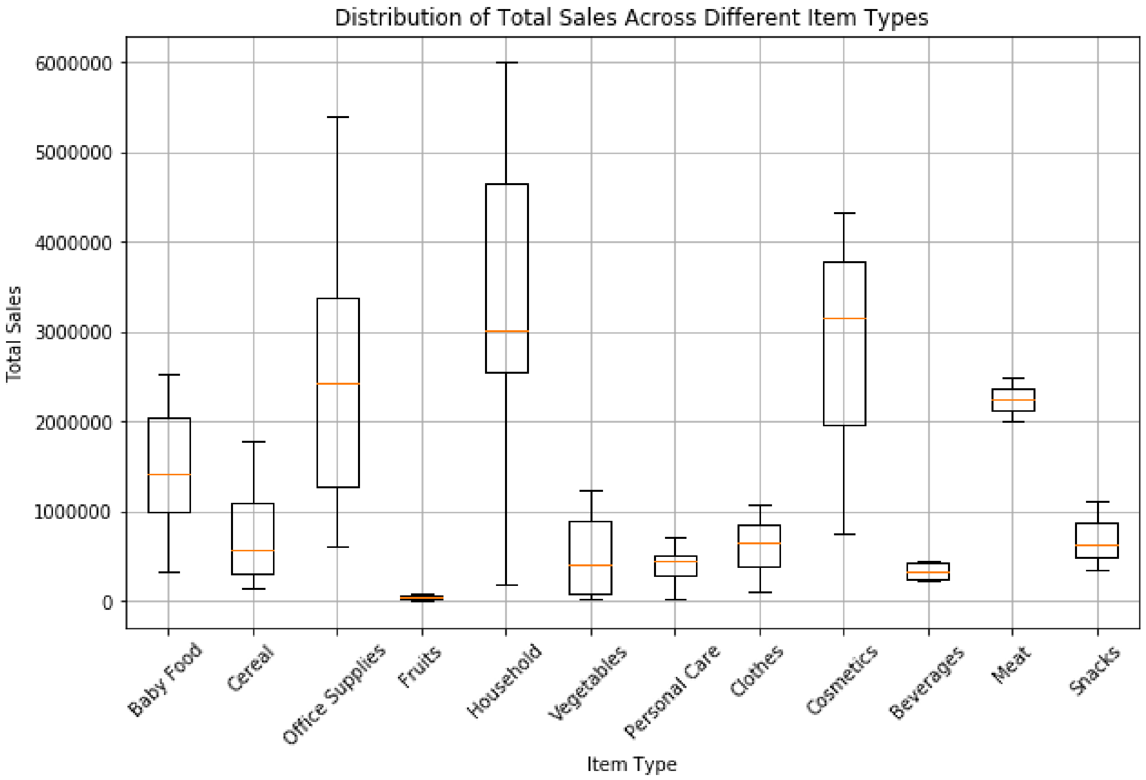
```
# Group by region and calculate total sales
region_sales = sales.groupby('Region')['Total Revenue'].sum()

# Plot the pie chart
plt.figure(figsize=(8, 8))
plt.pie(region_sales, labels=region_sales.index, autopct='%1.1f%%', startangle=140)
plt.title('Distribution of Sales Across Regions')
plt.show()
```



In [32]:

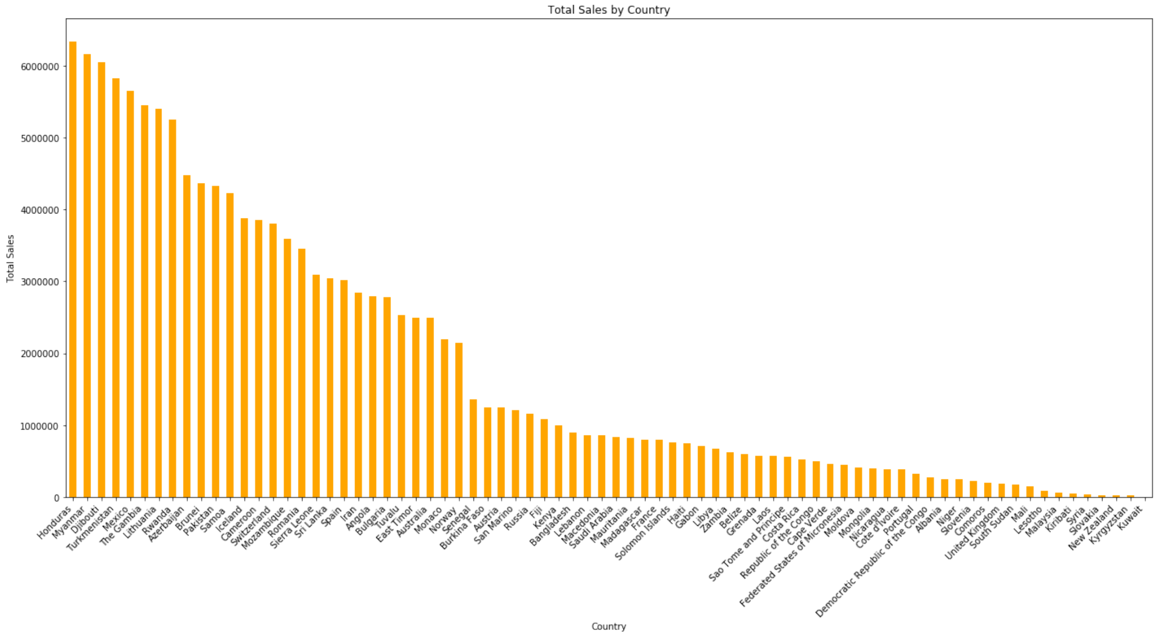
```
# Observation : sub saharan Africa region with highest total revenue of 28.9 %
```

In [33]:
```
# Plot the box plot
plt.figure(figsize=(10, 6))
plt.boxplot([sales[sales['Item Type'] == item]['Total Revenue'] for item in
sales['Item Type'].unique()],
            labels=sales['Item Type'].unique())
plt.title('Distribution of Total Sales Across Different Item Types')
plt.xlabel('Item Type')
plt.ylabel('Total Sales')
plt.xticks(rotation=45)
plt.grid(True)
plt.show()
```



In [34]:

```
sales_by_country = sales.groupby('Country')['Total Revenue'].sum().sort_values(ascending=False)

# Plot the bar chart
plt.figure(figsize=(18, 10))
sales_by_country.plot(kind='bar', color='orange')
plt.title('Total Sales by Country')
plt.xlabel('Country')
plt.ylabel('Total Sales')
plt.xticks(rotation=45, ha='right')  # Rotate x-axis labels for better readability
plt.tight_layout()  # Adjust layout to prevent clipping of labels
plt.show()
```

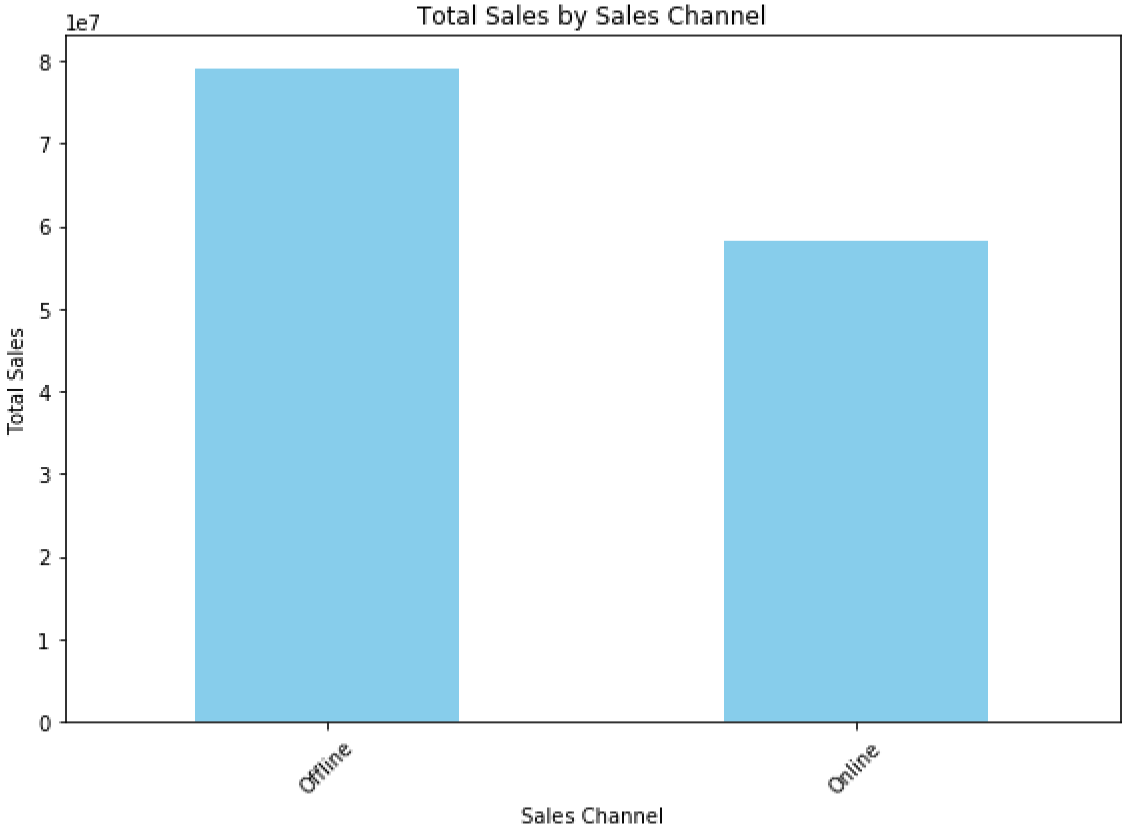


In [35]:

```
# Observation : Country with the Highest Total Revenue

#Upon analyzing the sales data, it is observed that the country Honduras has the highest total revenue among all countries represented in the dataset.
#This indicates that sales activities in Honduras have contributed significantly to the overall
#revenue generated by the company.

#This observation suggests that Honduras may be a key market for the company,
```

In [36]:

```
sales_by_channel = sales.groupby('Sales Channel')['Total Revenue'].sum().so
rt_values(ascending=False)

# Plot the bar chart
plt.figure(figsize=(8, 6))
sales_by_channel.plot(kind='bar', color='skyblue')
plt.title('Total Sales by Sales Channel')
plt.xlabel('Sales Channel')
plt.ylabel('Total Sales')
plt.xticks(rotation=45)  # Rotate x-axis labels for better readability
plt.tight_layout()  # Adjust layout to prevent clipping of labels
plt.show()
```

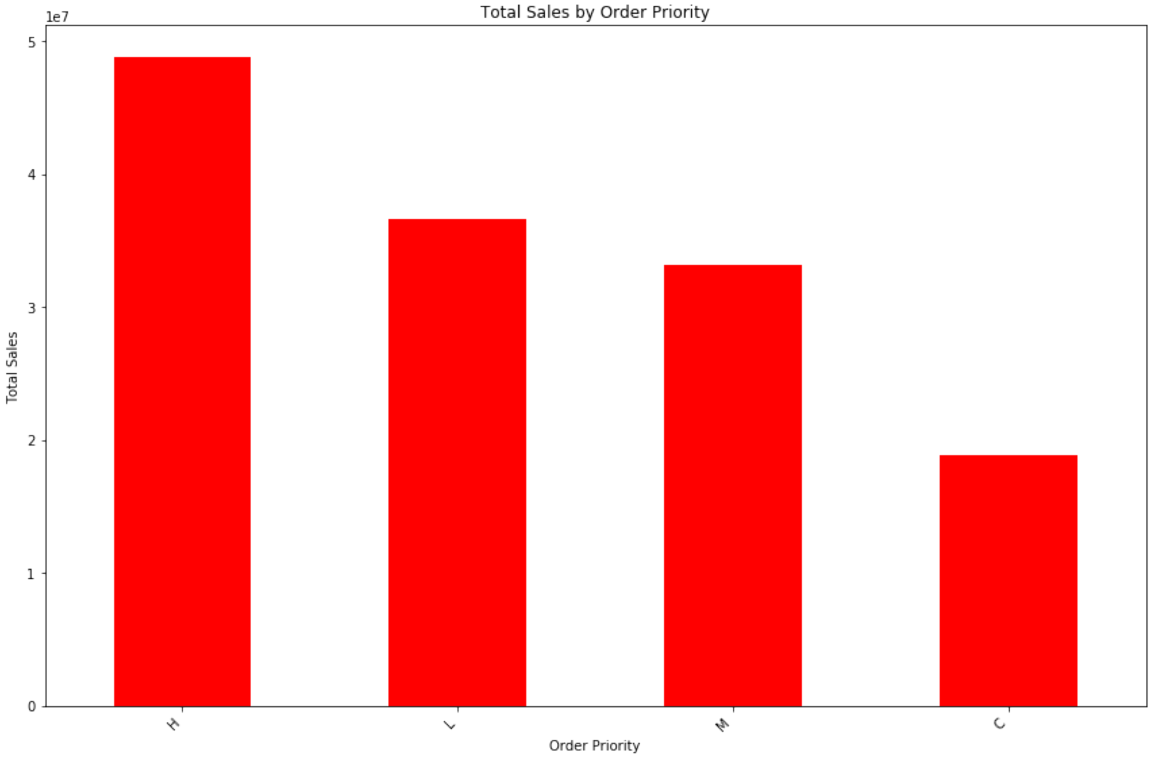


In [37]:

```
#Observation Upon analyzing the sales data, it is evident that the offline
sales channel
#consistently outperforms the online sales channel in terms of total sales.
```

In [38]:

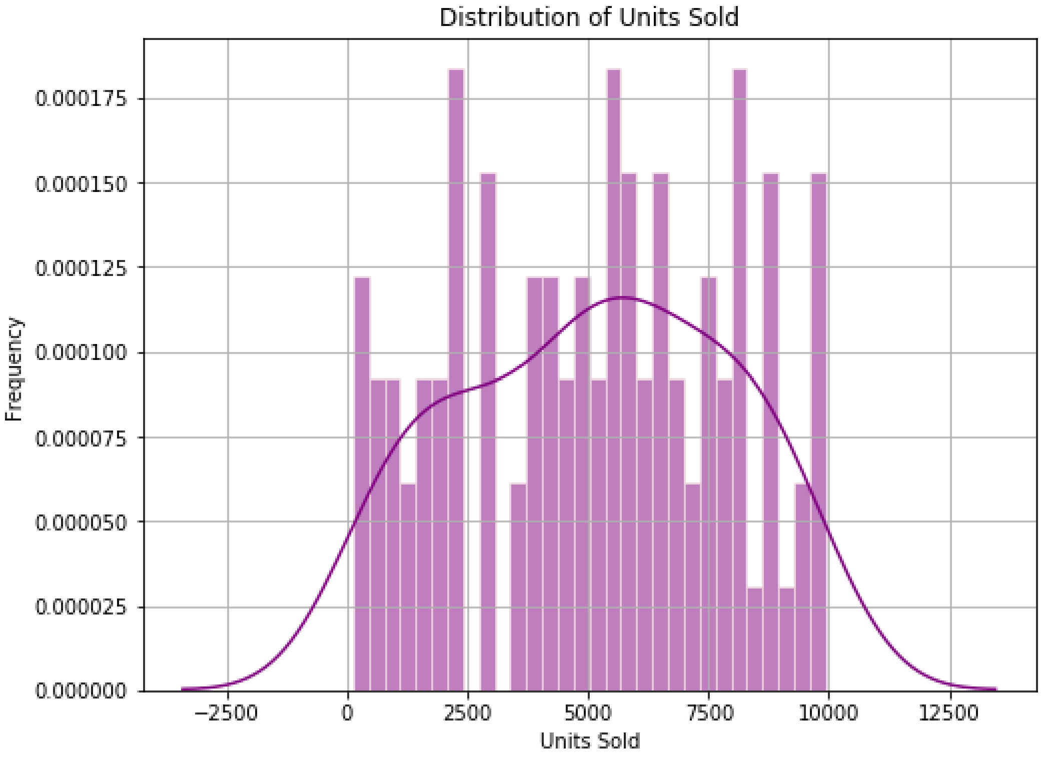
```
# Sales by Order Priority

sales_by_OrderPriority = sales.groupby('Order Priority')['Total Revenue'].sum().sort_values(ascending=False)

# Plot the bar chart
plt.figure(figsize=(12, 8))
sales_by_OrderPriority.plot(kind='bar', color='red')
plt.title('Total Sales by Order Priority')
plt.xlabel('Order Priority')
plt.ylabel('Total Sales')
plt.xticks(rotation=45, ha='right')  # Rotate x-axis labels for better readability
plt.tight_layout()  # Adjust layout to prevent clipping of labels
plt.show()
```



In [39]:

```
# Observation : Orders with a higher priority level may lead to increased sales volume or revenue.
```

In [40]:
```
plt.figure(figsize=(8, 6))
sns.distplot(sales['Units Sold'], color='purple', hist_kws={'edgecolor': 'l
inen', 'alpha': 0.5}, bins=30)

plt.title('Distribution of Units Sold')
plt.xlabel('Units Sold')
plt.ylabel('Frequency')
plt.grid(True)
plt.show()
```

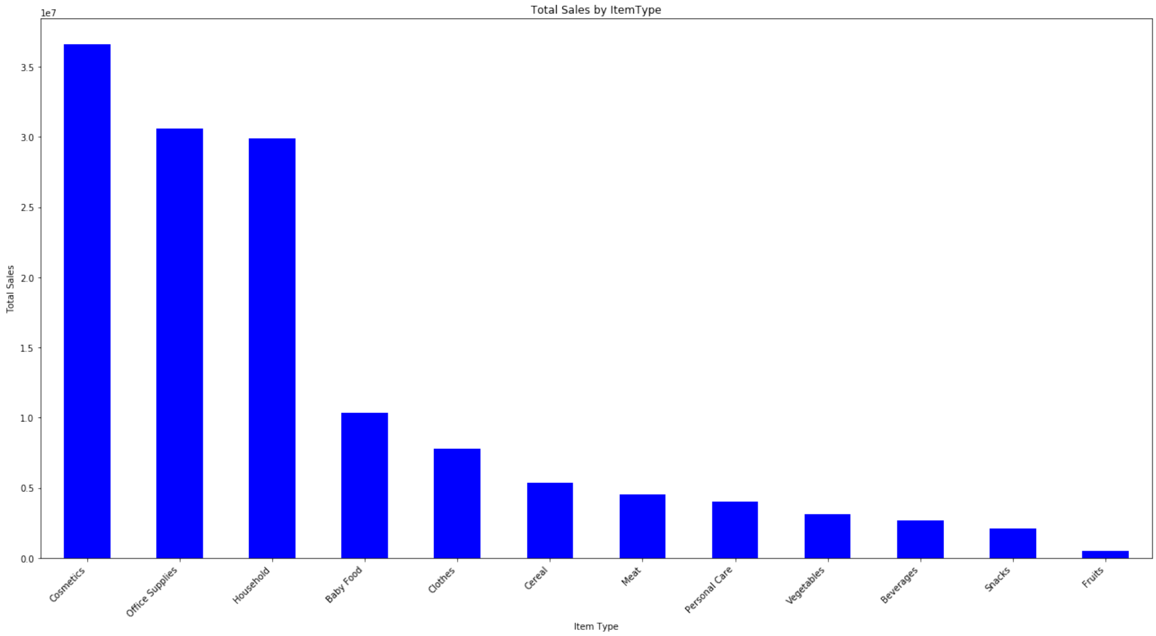


In [41]:
```
#Obs
```

In [42]:
```
# Product in high demand
```

In [43]:

```
sales_by_Itemtype = sales.groupby('Item Type')['Total Revenue'].sum().sort_values(ascending=False)

# Plot the bar chart
plt.figure(figsize=(18, 10))
sales_by_Itemtype.plot(kind='bar', color='blue')
plt.title('Total Sales by ItemType')
plt.xlabel('Item Type')
plt.ylabel('Total Sales')
plt.xticks(rotation=45, ha='right')  # Rotate x-axis labels for better readability
plt.tight_layout()  # Adjust layout to prevent clipping of labels
plt.show()
```

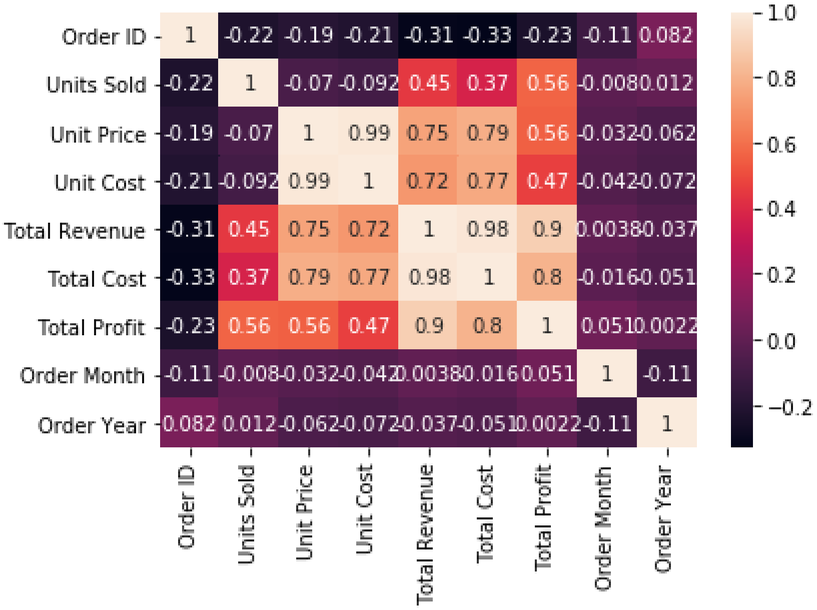


In [44]:

```
# Observation :
```

In [45]:

```
sales_CorrMatrix=sales.corr(method='pearson')
sns.heatmap(sales_CorrMatrix,annot=True)
plt.show()
```



In [46]:

```
#Final Conclusion
# From the given data, we can use simple visualisations to get a sense of how data are distributed.
```