Importing Liabraries

In [11]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

Reading dataset

In [14]:
```
df= pd.read_csv("Amazon Sales data.csv")
```

In [16]:
```
df.head()
```

Out[16]:

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

Checking the Shape

In [19]:
```
df.shape
```

Out[19]:
```
(100, 14)
```

In [21]:
```
df.columns
```

Out[21]:
```
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 [23]:
```
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 [25]:
```
df.head()
```

Out[25]:

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

In [27]:
```
df.dtypes
```

Out[27]:
```
Region             object
Country            object
Item Type          object
Sales Channel      object
Order Priority     object
Order Date         object
Order ID            int64
Ship Date          object
Units Sold          int64
Unit Price        float64
Unit Cost         float64
Total Revenue     float64
Total Cost        float64
Total Profit      float64
dtype: object
```

Data Cleaning

In [30]:
```
# CHECKING MISSING VALUES
df.isnull().sum()
```

Out[30]:
```
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
```

There's no missing values or null ones in the dataset, hence the data is cleaned already.

In [33]:
```
df['Order Date'] = pd.to_datetime(df['Order Date'])
df['Ship Date'] = pd.to_datetime(df['Ship Date'])
```

In [35]:
```
df['Region'] = df['Region'].astype(str)
df['Country'] = df['Country'].astype(str)
df['Item Type'] = df['Item Type'].astype(str)
df['Sales Channel'] = df['Sales Channel'].astype(str)
df['Order Priority'] = df['Order Priority'].astype(str)
```

In [37]:
```
df.head()
```

Out[37]:

	Region	Country	Item Type	Sales Channel	Order Priority	Order Date	Order ID	Ship Date	Units Sold	Unit Price
0	Australia and Oceania	Tuvalu	Baby Food	Offline	H	2010-05-28	669165933	2010-06-27	9925	255.28
1	Central America and the Caribbean	Grenada	Cereal	Online	C	2012-08-22	963881480	2012-09-15	2804	205.70
2	Europe	Russia	Office Supplies	Offline	L	2014-05-02	341417157	2014-05-08	1779	651.21
3	Sub-Saharan Africa	Sao Tome and Principe	Fruits	Online	C	2014-06-20	514321792	2014-07-05	8102	9.33
4	Sub-Saharan Africa	Rwanda	Office Supplies	Offline	L	2013-02-01	115456712	2013-02-06	5062	651.21

In [39]:
```
df['Order Month'] = df['Order Date'].dt.month
df['Order Year'] = df['Order Date'].dt.year

df['Year-Month']= df['Order Date'].dt.to_period('M')

df = df.drop(columns=['Order Date'])
```

In [41]:
```
df.head()
```

Out[41]:

	Region	Country	Item Type	Sales Channel	Order Priority	Order ID	Ship Date	Units Sold	Unit Price	Unit Cost
0	Australia and Oceania	Tuvalu	Baby Food	Offline	H	669165933	2010-06-27	9925	255.28	159.42
1	Central America and the Caribbean	Grenada	Cereal	Online	C	963881480	2012-09-15	2804	205.70	117.11
2	Europe	Russia	Office Supplies	Offline	L	341417157	2014-05-08	1779	651.21	524.96
3	Sub-Saharan Africa	Sao Tome and Principe	Fruits	Online	C	514321792	2014-07-05	8102	9.33	6.92
4	Sub-Saharan Africa	Rwanda	Office Supplies	Offline	L	115456712	2013-02-06	5062	651.21	524.96

In [43]:
```
df['Ship Month'] = df['Ship Date'].dt.month
df['Ship Year'] = df['Ship Date'].dt.year

df['Year-Month']= df['Ship Date'].dt.to_period('M')

df = df.drop(columns=['Ship Date'])
```

In [45]:
```
df.head()
```

Out[45]:

	Region	Country	Item Type	Sales Channel	Order Priority	Order ID	Units Sold	Unit Price	Unit Cost	Rev
0	Australia and Oceania	Tuvalu	Baby Food	Offline	H	669165933	9925	255.28	159.42	25336
1	Central America and the Caribbean	Grenada	Cereal	Online	C	963881480	2804	205.70	117.11	5767
2	Europe	Russia	Office Supplies	Offline	L	341417157	1779	651.21	524.96	11585
3	Sub-Saharan Africa	Sao Tome and Principe	Fruits	Online	C	514321792	8102	9.33	6.92	755
4	Sub-Saharan Africa	Rwanda	Office Supplies	Offline	L	115456712	5062	651.21	524.96	32964

In [47]:
```
df.describe()
```

Out[47]:

	Order ID	Units Sold	Unit Price	Unit Cost	Total Revenue	Total Cost	T
count	1.000000e+02	100.000000	100.000000	100.000000	1.000000e+02	1.000000e+02	1.0
mean	5.550204e+08	5128.710000	276.761300	191.048000	1.373488e+06	9.318057e+05	4.4
std	2.606153e+08	2794.484562	235.592241	188.208181	1.460029e+06	1.083938e+06	4.3
min	1.146066e+08	124.000000	9.330000	6.920000	4.870260e+03	3.612240e+03	1.2
25%	3.389225e+08	2836.250000	81.730000	35.840000	2.687212e+05	1.688680e+05	1.2
50%	5.577086e+08	5382.500000	179.880000	107.275000	7.523144e+05	3.635664e+05	2.9
75%	7.907551e+08	7369.000000	437.200000	263.330000	2.212045e+06	1.613870e+06	6.3
max	9.940222e+08	9925.000000	668.270000	524.960000	5.997055e+06	4.509794e+06	1.7

In [49]:
```
regions = df['Region'].nunique()
print('Number of Regions:', regions)

countries = df['Country'].nunique()
print('Number of Countries:', countries)

item_type = df['Item Type'].nunique()
print('Number of Item Types:', item_type)
```

```
Number of Regions: 7
Number of Countries: 76
Number of Item Types: 12
```

In [51]:
```
df.loc[:,["Total Revenue","Total Profit"]].iloc[:]
```

Out[51]:

	Total Revenue	Total Profit
0	2533654.00	951410.50
1	576782.80	248406.36
2	1158502.59	224598.75
3	75591.66	19525.82
4	3296425.02	639077.50
...	...	...
95	97040.64	65214.72
96	58471.11	15103.47
97	228779.10	93748.05
98	471336.91	144521.02
99	3586605.09	889472.91

100 rows × 2 columns

In [53]:
```
# Display total values of all country
pd.set_option('display.max_rows', None)
df['Country'].value_counts()
```

Out[53]:

Country	
The Gambia	4
Sierra Leone	3
Sao Tome and Principe	3
Mexico	3
Australia	3
Djibouti	3
Switzerland	2
Myanmar	2
Norway	2
Turkmenistan	2
Cameroon	2
Bulgaria	2
Honduras	2
Azerbaijan	2
Libya	2
Rwanda	2
Mali	2
Gabon	1
Belize	1
Haiti	1
Lithuania	1
San Marino	1
United Kingdom	1
Austria	1
Fiji	1
Madagascar	1
Cote d'Ivoire	1
Tuvalu	1
Democratic Republic of the Congo	1
Zambia	1
Malaysia	1
Nicaragua	1
Romania	1
Slovenia	1
Kuwait	1
Kenya	1
Iran	1
Pakistan	1
Lebanon	1
Spain	1
Samoa	1
Monaco	1
Laos	1
Saudi Arabia	1
Federated States of Micronesia	1
Slovakia	1
Lesotho	1
Albania	1
Russia	1
Solomon Islands	1
Angola	1
Burkina Faso	1
Republic of the Congo	1
Senegal	1
Kyrgyzstan	1

```
Cape Verde                      1
Bangladesh                      1
Mongolia                        1
Sri Lanka                       1
East Timor                      1
Portugal                        1
New Zealand                     1
Moldova                         1
France                          1
Kiribati                        1
South Sudan                     1
Costa Rica                      1
Syria                           1
Brunei                          1
Niger                           1
Grenada                         1
Comoros                         1
Iceland                         1
Macedonia                       1
Mauritania                      1
Mozambique                      1
Name: count, dtype: int64
```

In [55]:
```
unit_sold = df['Units Sold'].sum()
print('Total Units Sold:', unit_sold)

unit_cost = df['Unit Cost'].sum()
print('Total Unit Cost:', unit_cost)

total_revenue = df['Total Revenue'].sum()
print('Total Total Revenue:', total_revenue)

total_profit = df['Total Profit'].sum()
print('Total Total Proft:', total_profit)

total_cost = df['Total Cost'].sum()
print('Total Cost:', total_cost)
```

```
Total Units Sold: 512871
Total Unit Cost: 19104.8
Total Total Revenue: 137348768.31
Total Total Proft: 44168198.39999999
Total Cost: 93180569.91000001
```

In [57]:
```
df.groupby(['Region', 'Sales Channel'])['Total Profit'].sum()
```

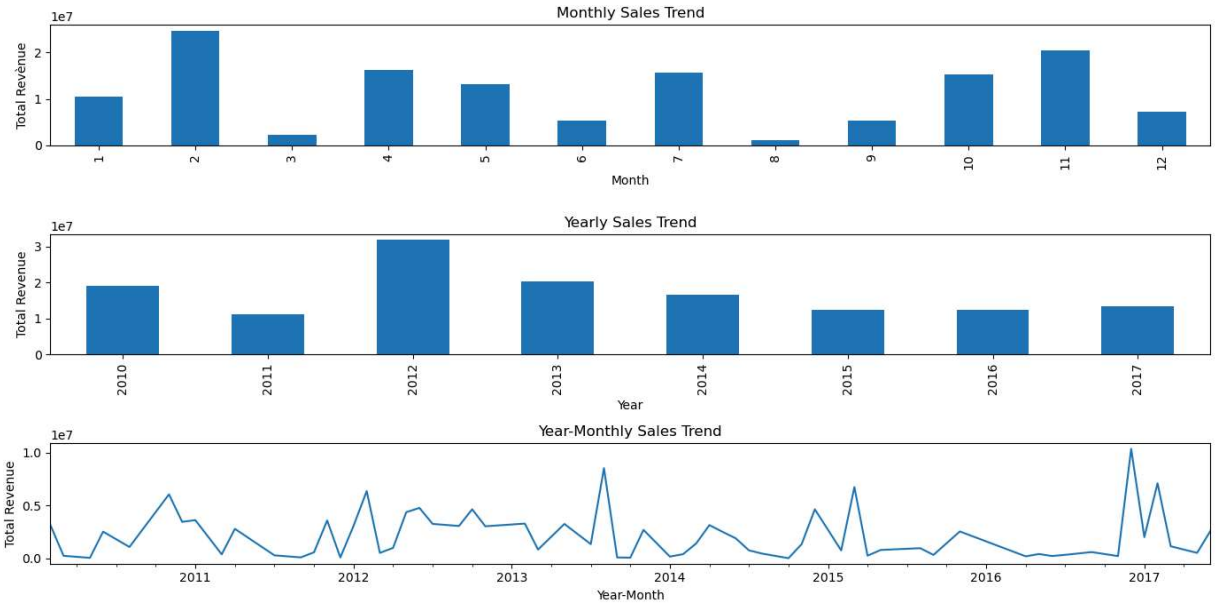
Out[57]:
```
Region                              Sales Channel
Asia                                Offline          3584286.33
                                    Online           2529559.54
Australia and Oceania               Offline          1886283.82
                                    Online           2835876.21
Central America and the Caribbean   Offline          2475814.99
                                    Online            371092.86
Europe                              Offline          5574539.91
                                    Online           5508398.72
Middle East and North Africa        Offline          2169081.08
                                    Online           3592110.78
North America                       Offline          1457942.76
Sub-Saharan Africa                  Offline          7772777.78
                                    Online           4410433.62
Name: Total Profit, dtype: float64
```

MORE PROFIT IN OFFLINE CHANNEL: Asia, Central America and the Caribbean, Europe and Sub-Saharan Africa

MORE PROFIT IN ONLINE CHANNEL: Australia and Oceania and Middle East and North Africa

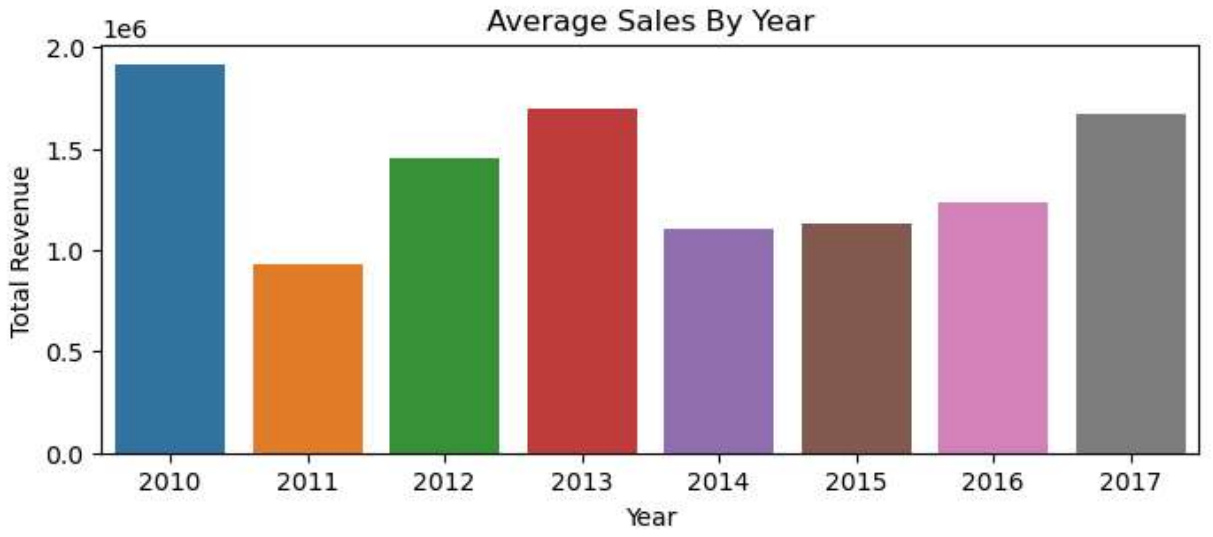
In [60]:
```
monthly_sales= df.groupby('Order Month')['Total Revenue'].sum()
yearly_sales= df.groupby('Order Year')['Total Revenue'].sum()
year_monthly_sales= df.groupby("Year-Month")[ 'Total Revenue'].sum()
```

In [62]:
```
plt.figure(figsize=(14,7))

plt.subplot(3, 1, 1)
monthly_sales.plot(kind='bar')
plt.title('Monthly Sales Trend')
plt.xlabel('Month')
plt.ylabel('Total Revènue')

plt.subplot(3, 1, 2)
yearly_sales.plot(kind="bar")
plt.title('Yearly Sales Trend')
plt.xlabel('Year')
plt.ylabel('Total Revenue')

plt.subplot(3, 1, 3)
year_monthly_sales.plot(kind='line')
plt.title('Year-Monthly Sales Trend')
plt.xlabel('Year-Month')
plt.ylabel('Total Revenue')

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

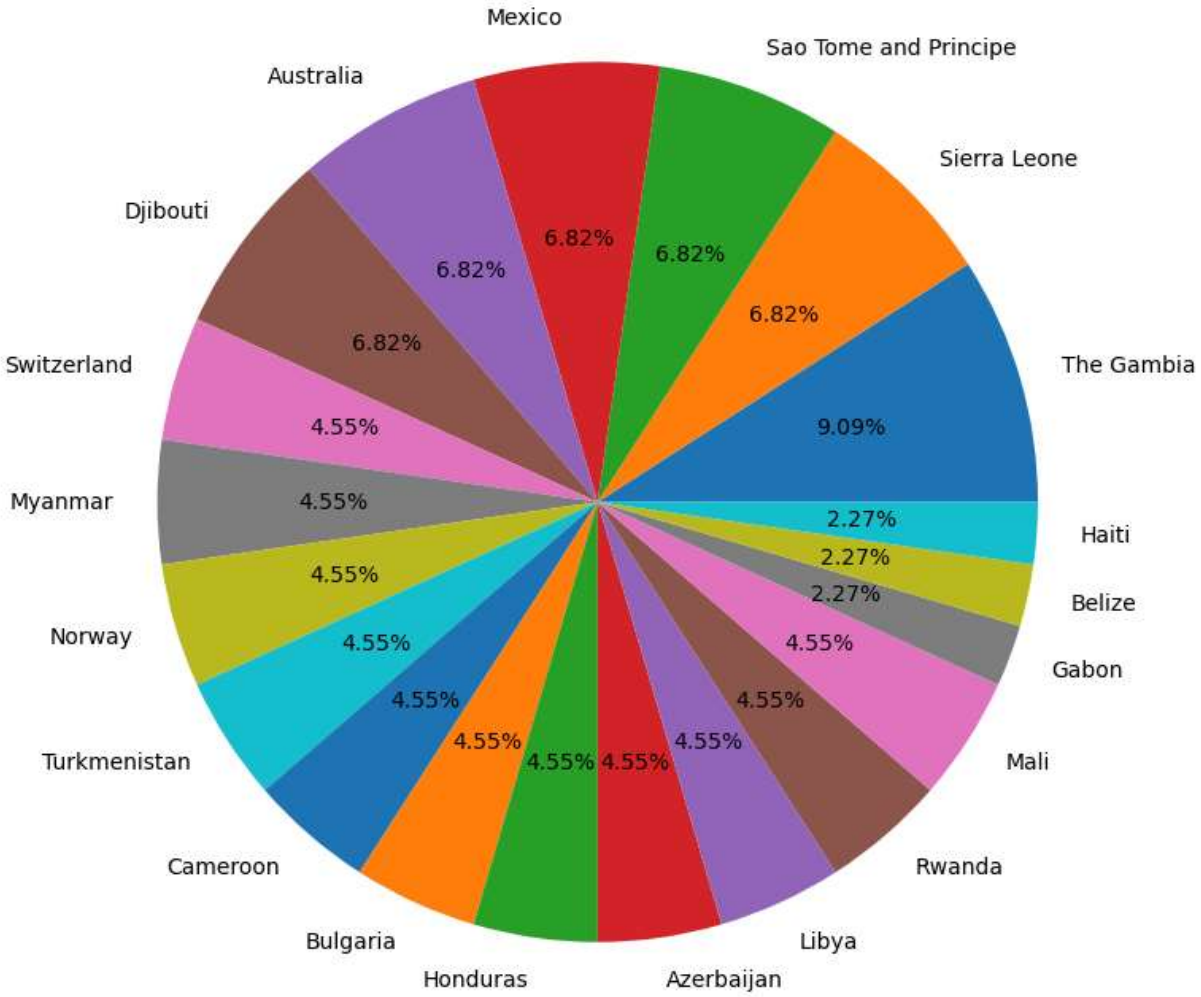


In [63]:
```
year_sales = df.groupby('Order Year')["Total Revenue"].mean()
plt.figure(figsize=(8,3))
sns.barplot (x=year_sales.index,y=year_sales.values,)
plt.title("Average Sales By Year")
plt.xlabel("Year")
plt.ylabel ("Total Revenue")
```

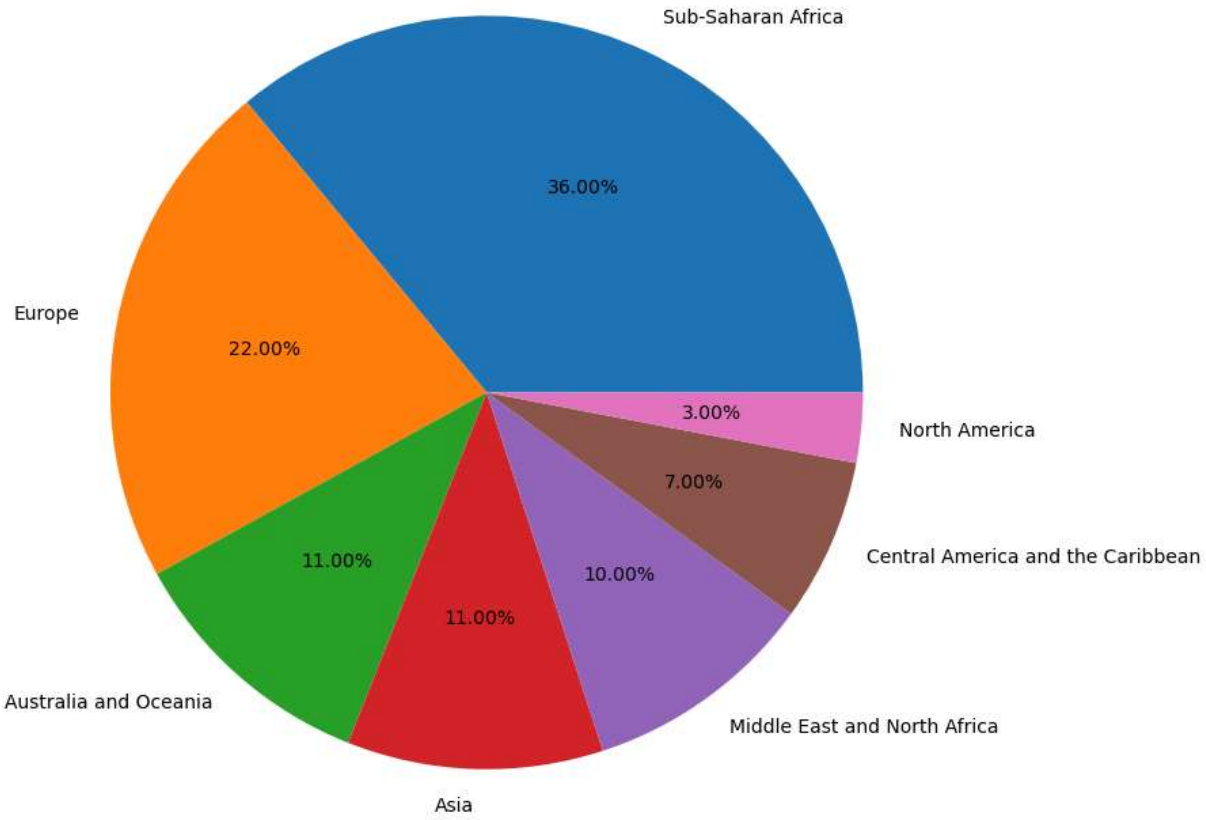
Out[63]: Text(0, 0.5, 'Total\xa0Revenue')



In [64]:
```
country_names = df.Country.value_counts().index
country_val = df.Country.value_counts().values
# Pie Chart for top 20 country
fig,ax = plt.subplots(figsize=(9,9))
ax.pie(country_val[:20],labels=country_names[:20],autopct='%1.2f%%')
plt.show()
```



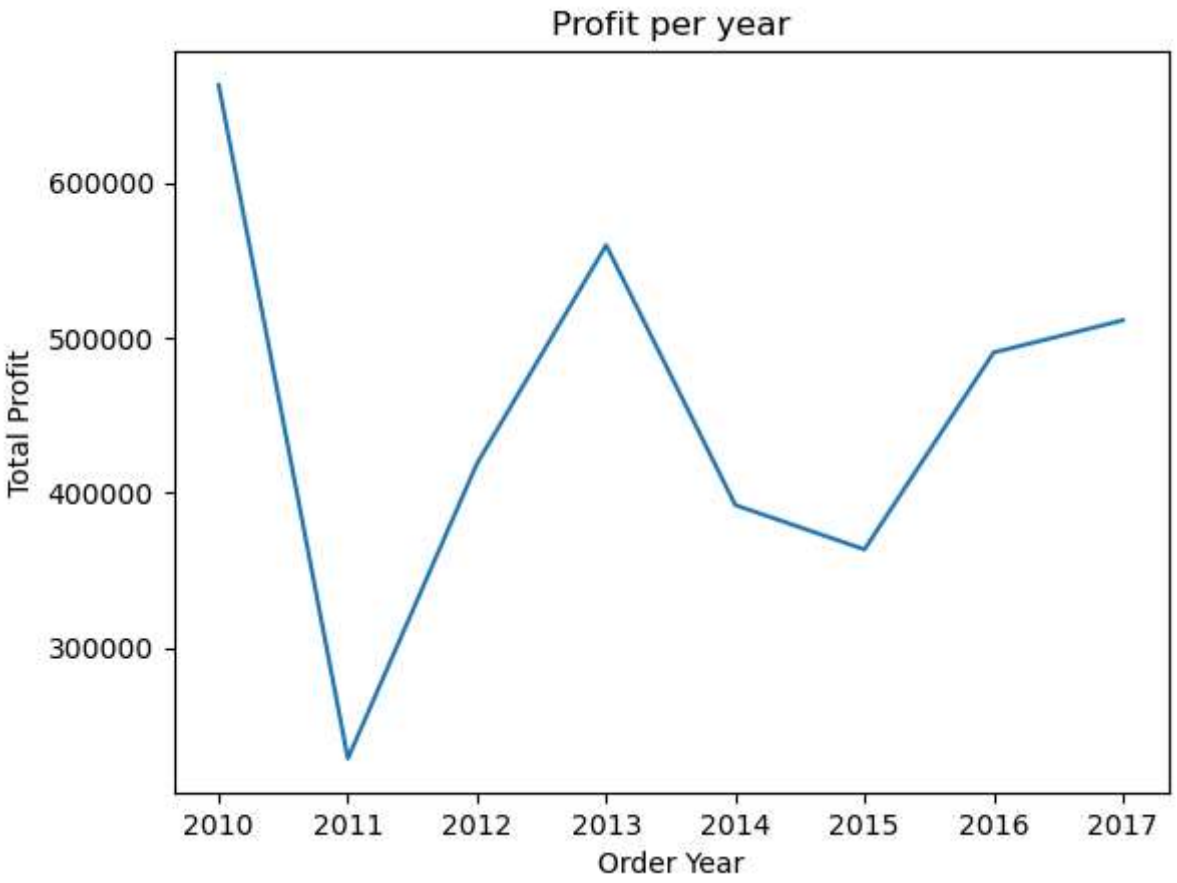
In [66]:
```
region_names = df.Region.value_counts().index
region_val = df.Region.value_counts().values

fig,ax = plt.subplots(figsize=(9,9))
ax.pie(region_val[:20],labels=region_names[:20],autopct='%1.2f%%')
plt.show()
```



In [68]:
```
# Creating a bar chart for Total Revenue and Order Month
plt.bar(df['Order Month'], df['Total Revenue'])

# Set the chart title and axis labels
plt.title('Number of Orders Purchased by Month and Year')
plt.xticks([1,2,3,4,5,6,7,8,9,10,11,12])
plt.xlabel('Order Month')
plt.ylabel('Total Revenue')

# Rotate the x-axis labels for better readability

# Display the chart
plt.show()
```

In [72]:
```
# Plot line graph of Total Profit and Order Year
df.groupby('Order Year')['Total Profit'].mean().plot()
plt.xlabel('Order Year')
plt.ylabel('Total Profit')
plt.title('Profit per year')
```

Out[72]:
```
Text(0.5, 1.0, 'Profit per year')
```

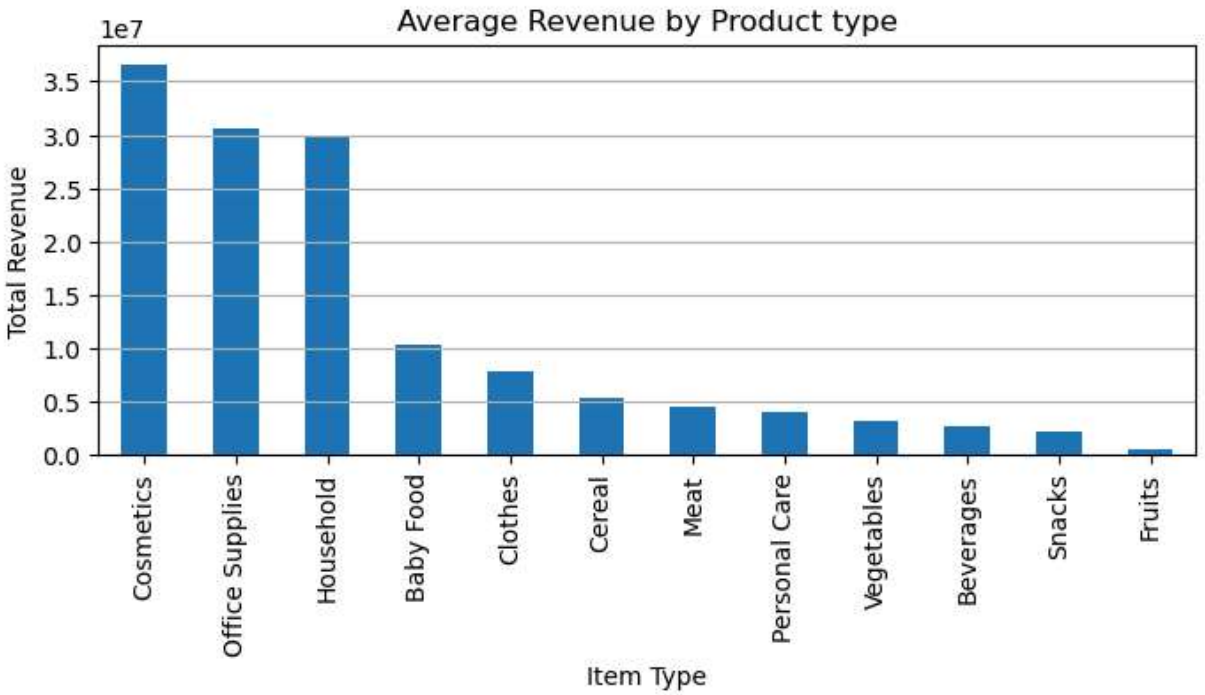


In [74]:
```
# Calculating the total revenue for each group with respect to Item Type and then s
revenue_by_category = df.groupby('Item Type')['Total Revenue'].sum().sort_values(as
revenue_by_category
```

Out[74]:
```
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
```

In [76]:
```
plt.figure(figsize=(8,3))

revenue_by_category.plot(kind='bar')

plt.xlabel('Item Type')

plt.ylabel('Total Revenue')

plt.title('Average Revenue by Product type')
```

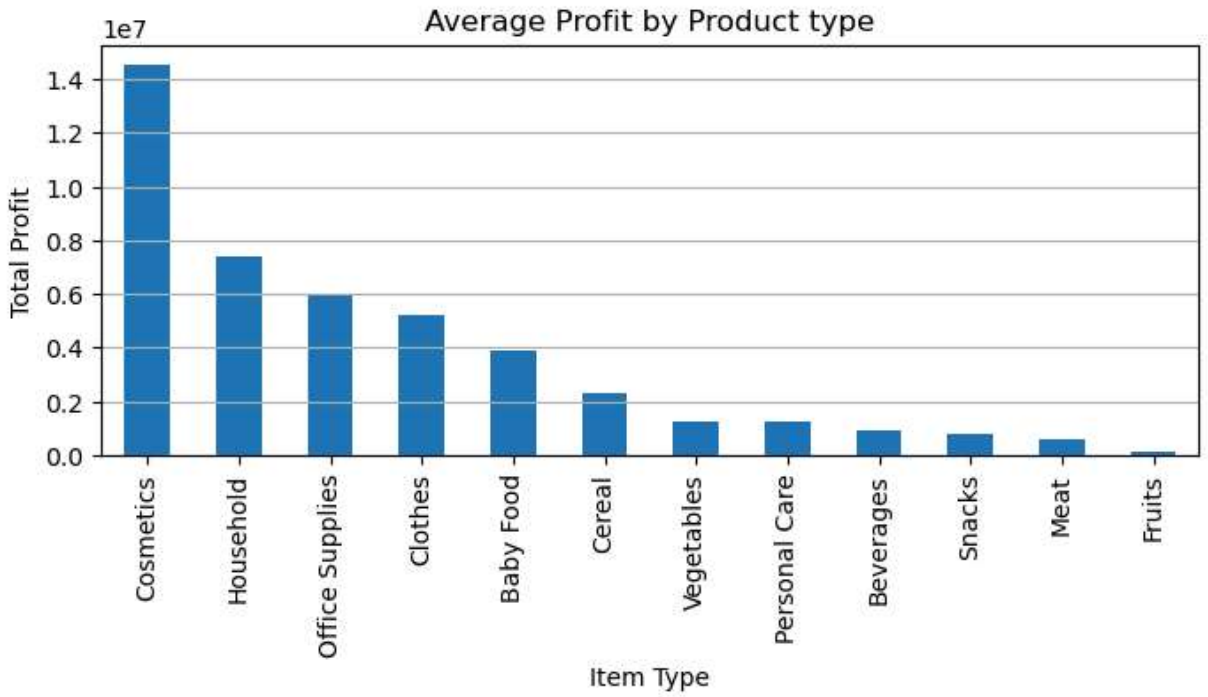
```
plt.grid(axis='y')
```



In [78]:
```
# Calculating the total profit for each group with respect to Item Type and then so
profit_by_category = df.groupby('Item Type')['Total Profit'].sum().sort_values(asce
profit_by_category
```

Out[78]:
```
Item Type
Cosmetics          14556048.66
Household           7412605.71
Office Supplies     5929583.75
Clothes             5233334.40
Baby Food           3886643.70
Cereal              2292443.43
Vegetables          1265819.63
Personal Care       1220622.48
Beverages            888047.28
Snacks               751944.18
Meat                 610610.00
Fruits               120495.18
Name: Total Profit, dtype: float64
```

In [80]:
```
plt.figure(figsize=(8,3))

profit_by_category.plot(kind='bar')

plt.xlabel('Item Type')

plt.ylabel('Total Profit')

plt.title('Average Profit by Product type')

plt.grid(axis='y')
```

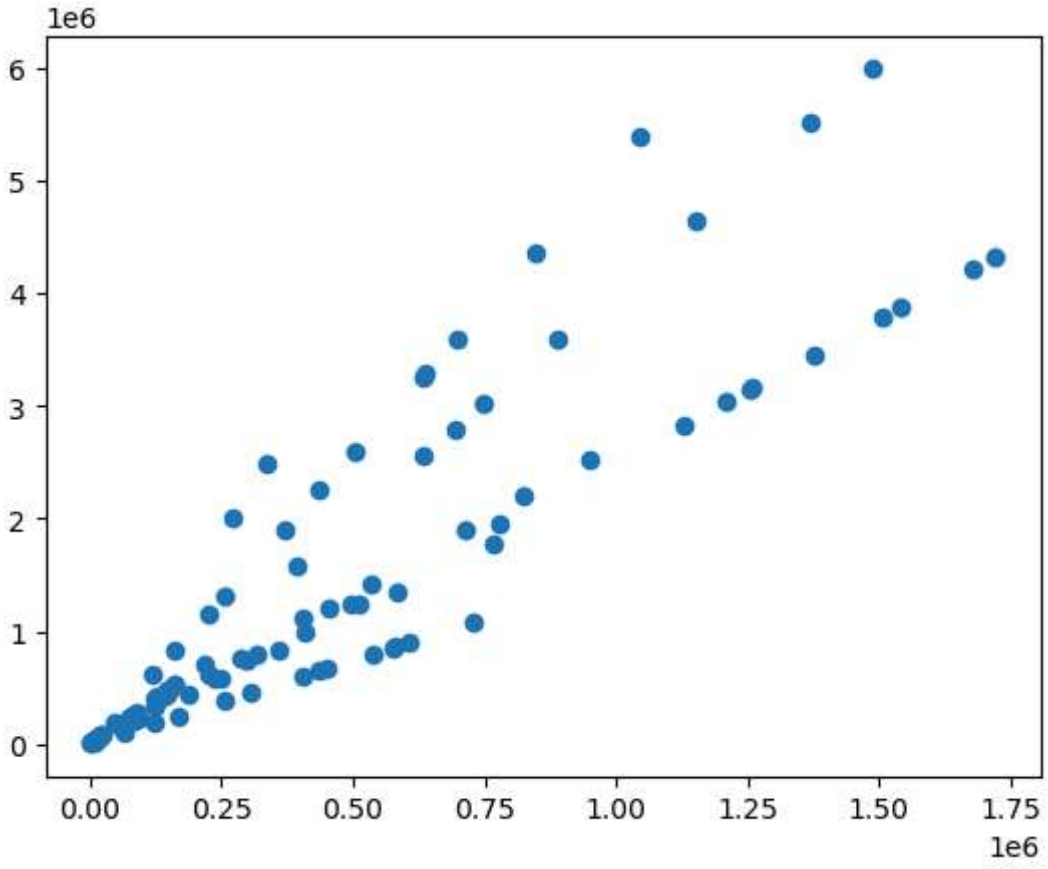


In [67]:
```
# Calculating correlation of 'Total Revenue', 'Total Cost' and 'Total Profit' colum
print(df[['Total Revenue', 'Total Cost', 'Total Profit']].corr())
```

```
                Total Revenue  Total Cost  Total Profit
Total Revenue        1.000000    0.983928      0.897327
Total Cost           0.983928    1.000000      0.804091
Total Profit         0.897327    0.804091      1.000000
```

In [69]:
```
plt.scatter(df['Total Profit'],df['Total Revenue'])
```

Out[69]:
```
<matplotlib.collections.PathCollection at 0x1da607a4510>
```



The scatter plot also suggests that total profit and total revenue are directly proportional to each other.

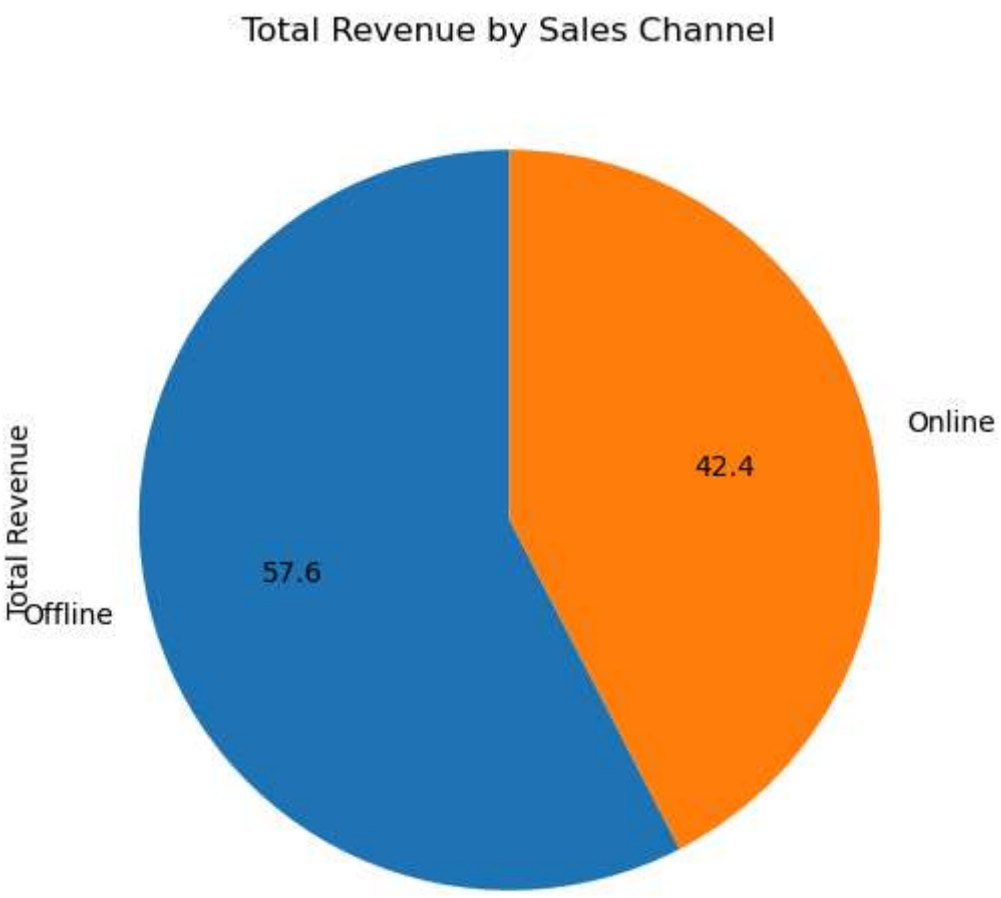
In [82]:
```
TotalRevenue_SalesChannel= df.groupby('Sales Channel') ['Total Revenue'].mean()
```

In [84]:
```
TotalRevenue_SalesChannel
```

Out[84]:
```
Sales Channel
Offline    1.581896e+06
Online     1.165079e+06
Name: Total Revenue, dtype: float64
```

In [88]:
```
plt.figure(figsize=(6,6))

plt.tight_layout()

TotalRevenue_SalesChannel.plot(kind='pie', autopct="%1.1f", startangle=90)

plt.title("Total Revenue by Sales Channel")
```

Out[88]:
```
Text(0.5, 1.0, 'Total Revenue by Sales Channel')
```

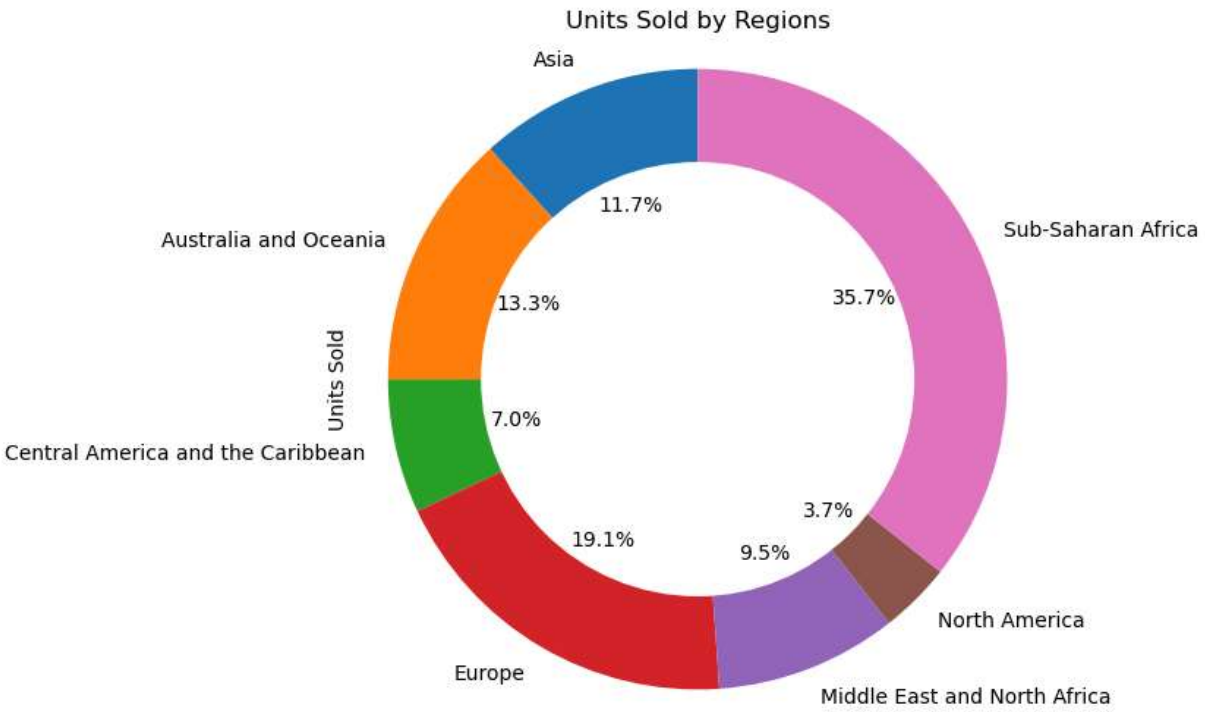


In [100…

```
Region_UnitSold= df.groupby('Region')["Units Sold"].sum()

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

Region_UnitSold.plot(kind='pie', labels= Region_UnitSold.index, autopct='%1.1f%%',

#Draw a circle at the centre of the pie Chart

cntr_circle = plt. Circle ((0,0), (0.70), fc='white')

fig= plt.gcf()

fig.gca().add_artist (cntr_circle)

#Equal aspect ratio ensures that pie is drawn as a circle

plt.title('Units Sold by Regions')

plt.axis('equal')
```

Out[100…

```
(-1.0999999530116766,
 1.0999990132545812,
 -1.0999995737000883,
 1.0999999797000042)
```



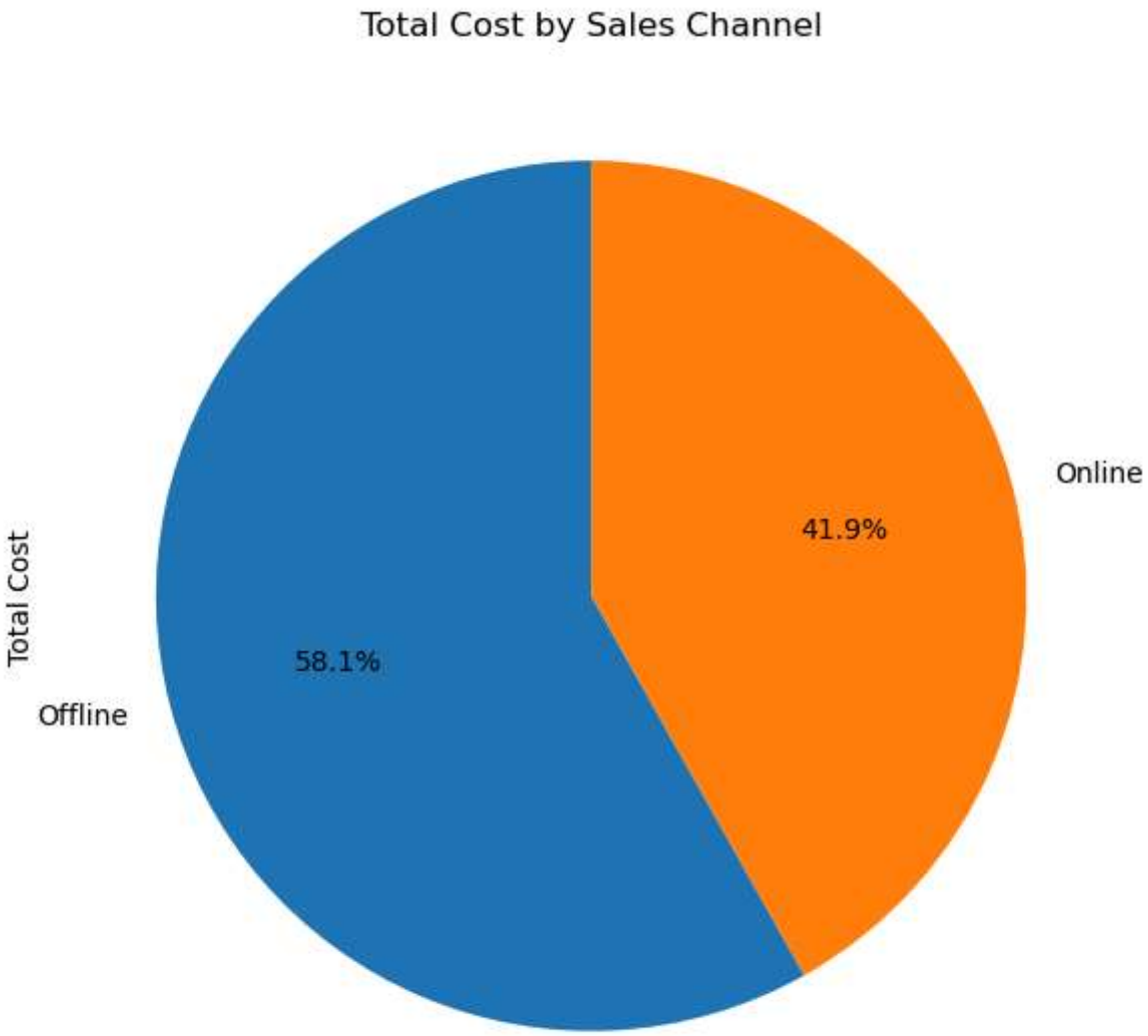
HIGHEST= Sub- Saharan Africa

LOWEST= North America

In [137…
```
TotalCost_SalesChannel = df.groupby('Sales Channel') ['Total Cost'].sum()
TotalCost_SalesChannel
```

Out[137…
```
Sales Channel
Offline    54174082.53
Online     39006487.38
Name: Total Cost, dtype: float64
```

In [139…
```
plt.figure(figsize=(6,6))

TotalCost_SalesChannel.plot(kind='pie', autopct='%1.1f%%', startangle=90)

plt.title("Total Cost by Sales Channel")

plt.tight_layout()
```



In [143…
```
np.average(df['Total Profit'])
```

Out[143…
```
441681.98399999994
```

At an average, the profit generated for a product is ₹441681.98.

In [152…
```
np.max(df['Total Profit'])
```

Out[152…
```
1719922.04
```

In [154…
```
np.min(df['Total Profit'])
```

Out[154…
```
1258.02
```

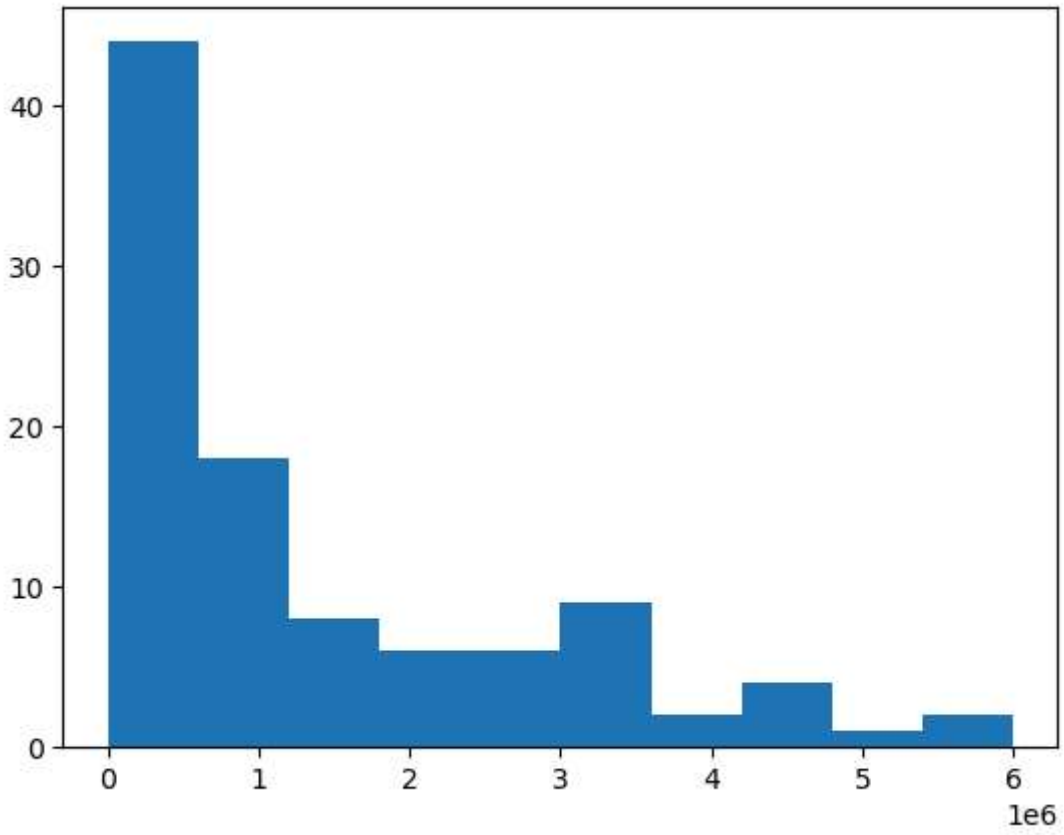
In [156…
```
np.var(df['Total Profit'])
```

Out[156…
```
190392340968.9648
```

Maximum and minimum profit generated are ₹ 1719922.04 and ₹ 1258.09 respectively.

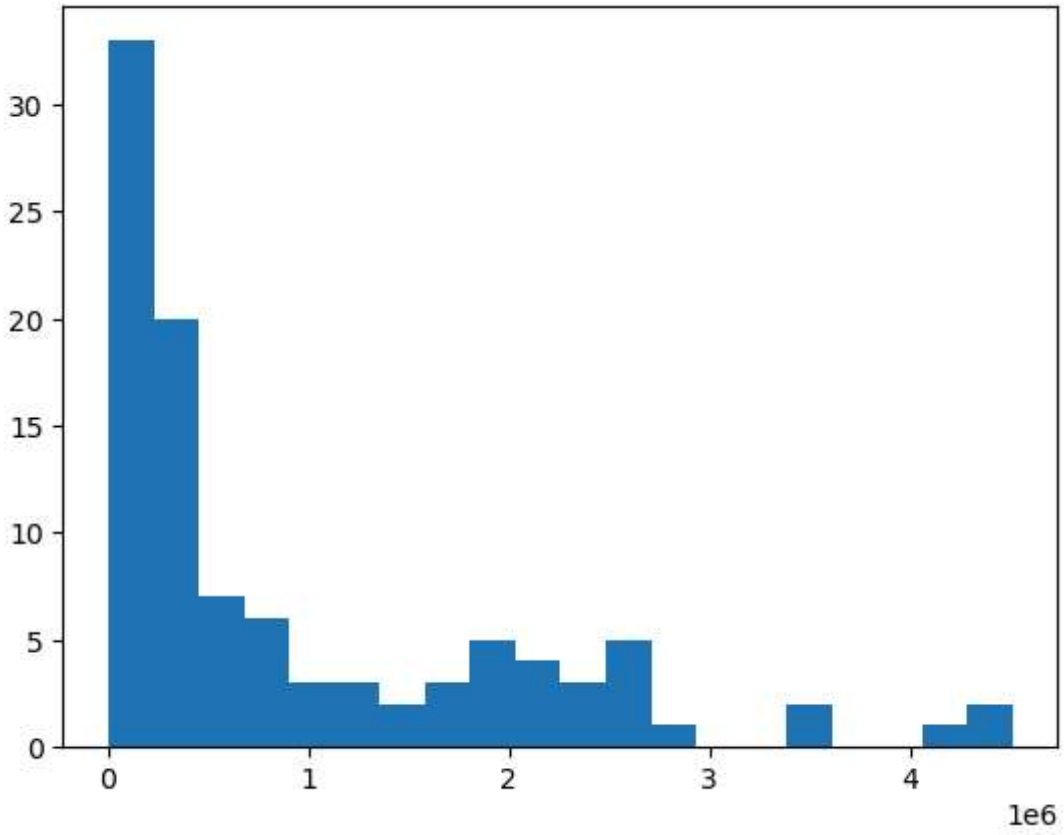
In [161…
```
plt.hist(df['Total Revenue'])
```

Out[161…
```
(array([44., 18.,  8.,  6.,  6.,  9.,  2.,  4.,  1.,  2.]),
 array([4.87026000e+03, 6.04088732e+05, 1.20330720e+06, 1.80252568e+06,
        2.40174415e+06, 3.00096262e+06, 3.60018109e+06, 4.19939956e+06,
        4.79861804e+06, 5.39783651e+06, 5.99705498e+06]),
 <BarContainer object of 10 artists>)
```



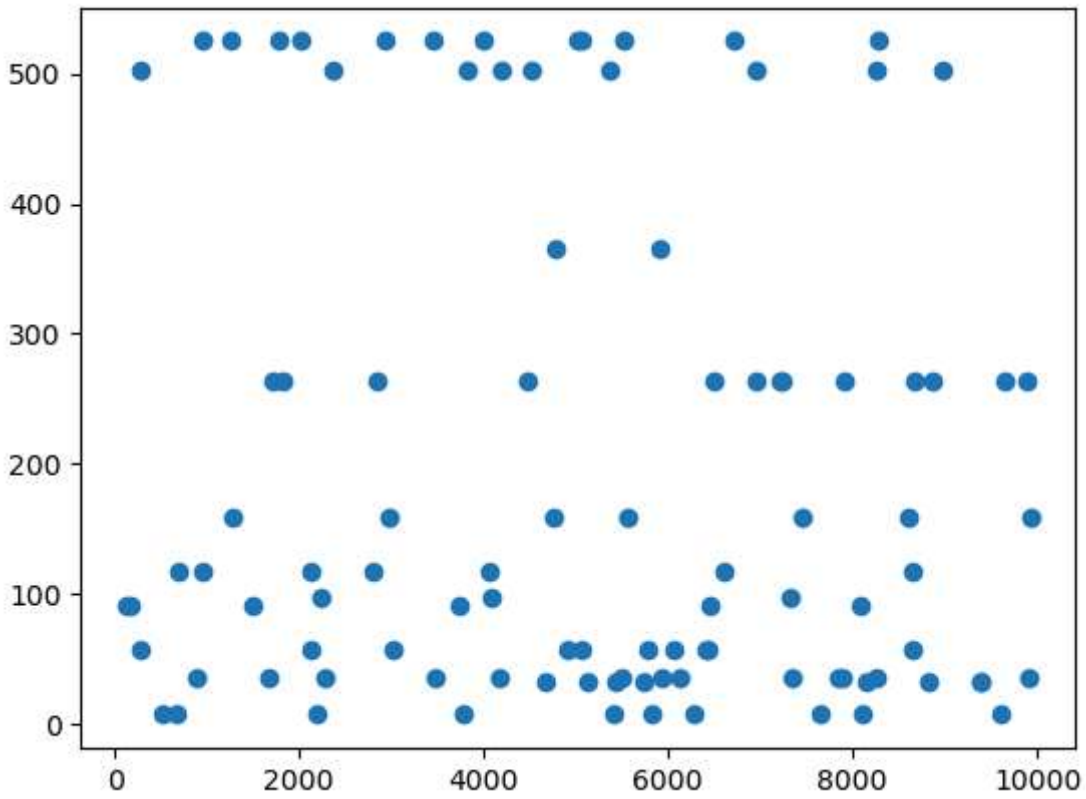
In [163…
```
plt.hist(df['Total Cost'],bins=20)
```

Out[163…
```
(array([33., 20.,  7.,  6.,  3.,  3.,  2.,  3.,  5.,  4.,  3.,  5.,  1.,
         0.,  0.,  2.,  0.,  0.,  1.,  2.]),
 array([3.61224000e+03, 2.28921326e+05, 4.54230412e+05, 6.79539498e+05,
        9.04848584e+05, 1.13015767e+06, 1.35546676e+06, 1.58077584e+06,
        1.80608493e+06, 2.03139401e+06, 2.25670310e+06, 2.48201219e+06,
        2.70732127e+06, 2.93263036e+06, 3.15793944e+06, 3.38324853e+06,
        3.60855762e+06, 3.83386670e+06, 4.05917579e+06, 4.28448487e+06,
        4.50979396e+06]),
 <BarContainer object of 20 artists>)
```



In [167…
```
plt.scatter(df['Units Sold'],df['Unit Cost'])
```

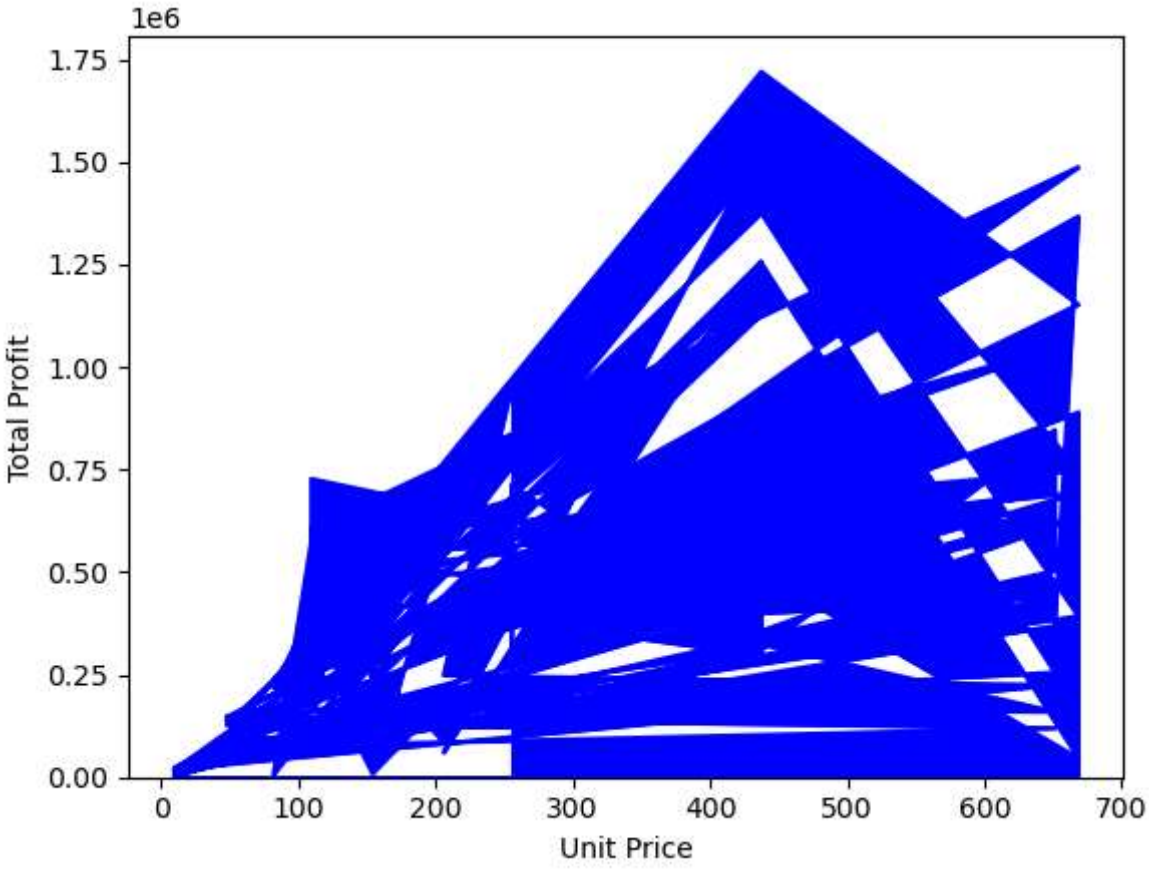
Out[167…
```
<matplotlib.collections.PathCollection at 0x1da64f2f550>
```



The above scatter plot implies that the two variables 'Units Sold' and 'Unit Cost' are inversely proportional to each other to some extent. When more units of a product are sold, the unit cost of that product becomes lesser and vice versa.

In [172…
```
area_plot = df.plot.area(x='Unit Price',y='Total Profit',color='blue',stacked=True,
plt.ylabel('Total Profit')
```

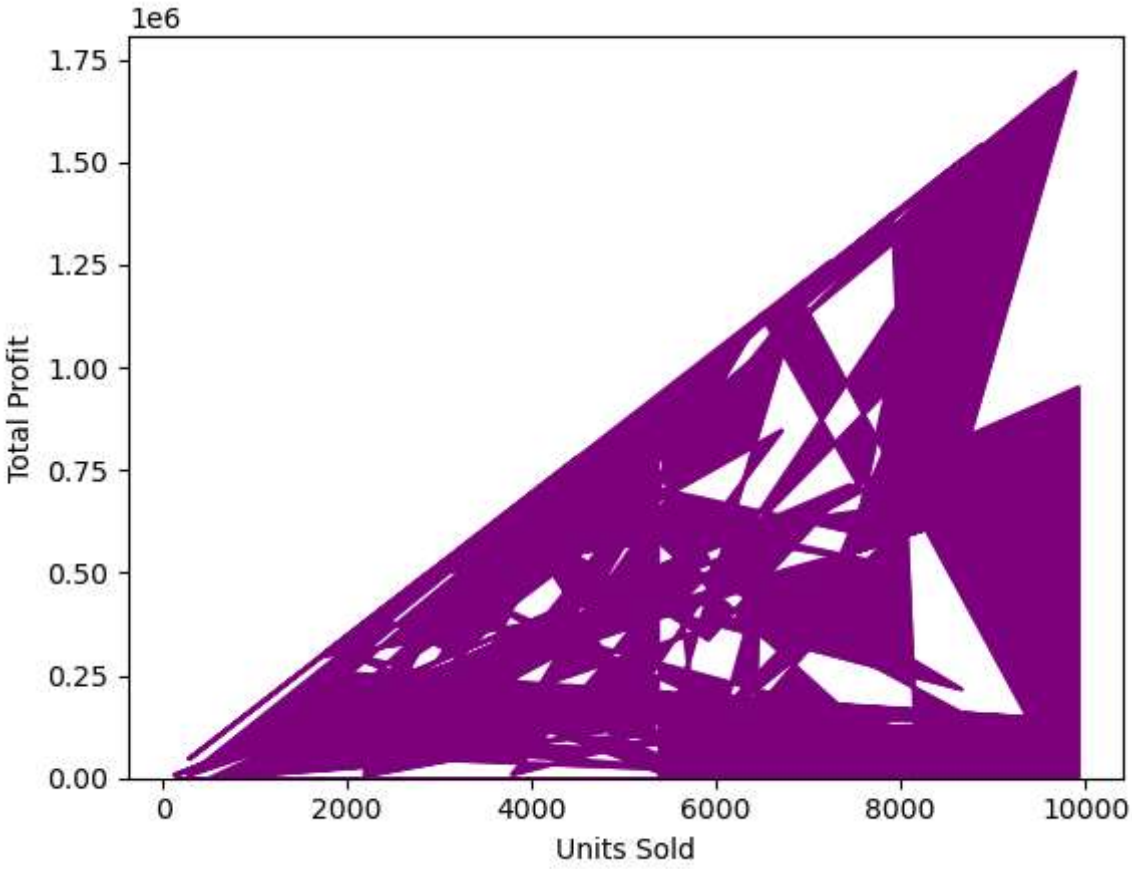
Out[172…
```
Text(0, 0.5, 'Total Profit')
```



Maximum profit has been generated in the unit price range of ₹400-₹500.

In [177…
```
df.plot.area(x='Units Sold',y='Total Profit',color='purple',legend=None)
plt.ylabel('Total Profit')
```

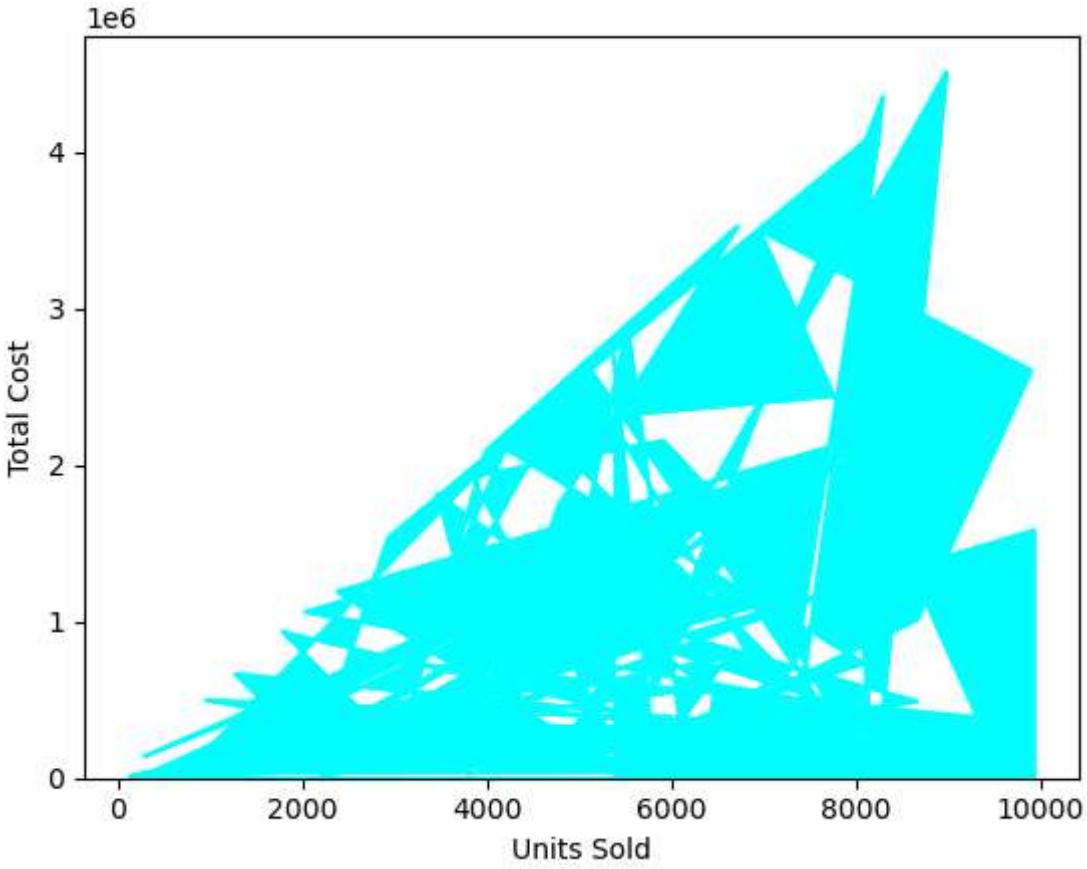
Out[177…
```
Text(0, 0.5, 'Total Profit')
```



Maximum profit has been generated when the number of units sold were between 8000 and 10000 i.e. more the number of units sold, more will be the profit generated.

In [182…
```
df.plot.area(x='Units Sold',y='Total Cost',color='aqua',legend=None)
plt.ylabel('Total Cost')
```

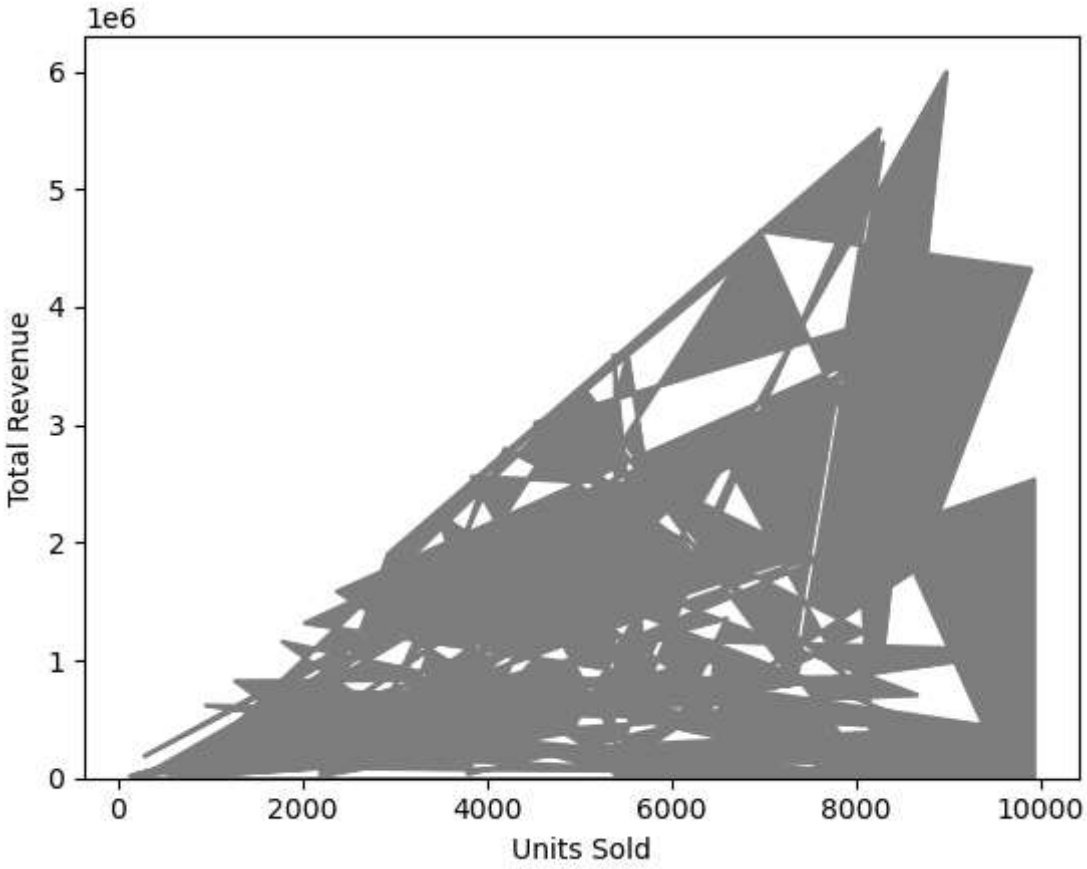
Out[182…
```
Text(0, 0.5, 'Total Cost')
```



Maximum cost has been generated when 8000-9000 units were sold.

In [185…
```
df.plot.area(x='Units Sold',y='Total Revenue',color='grey',legend=None)
plt.ylabel('Total Revenue')
```

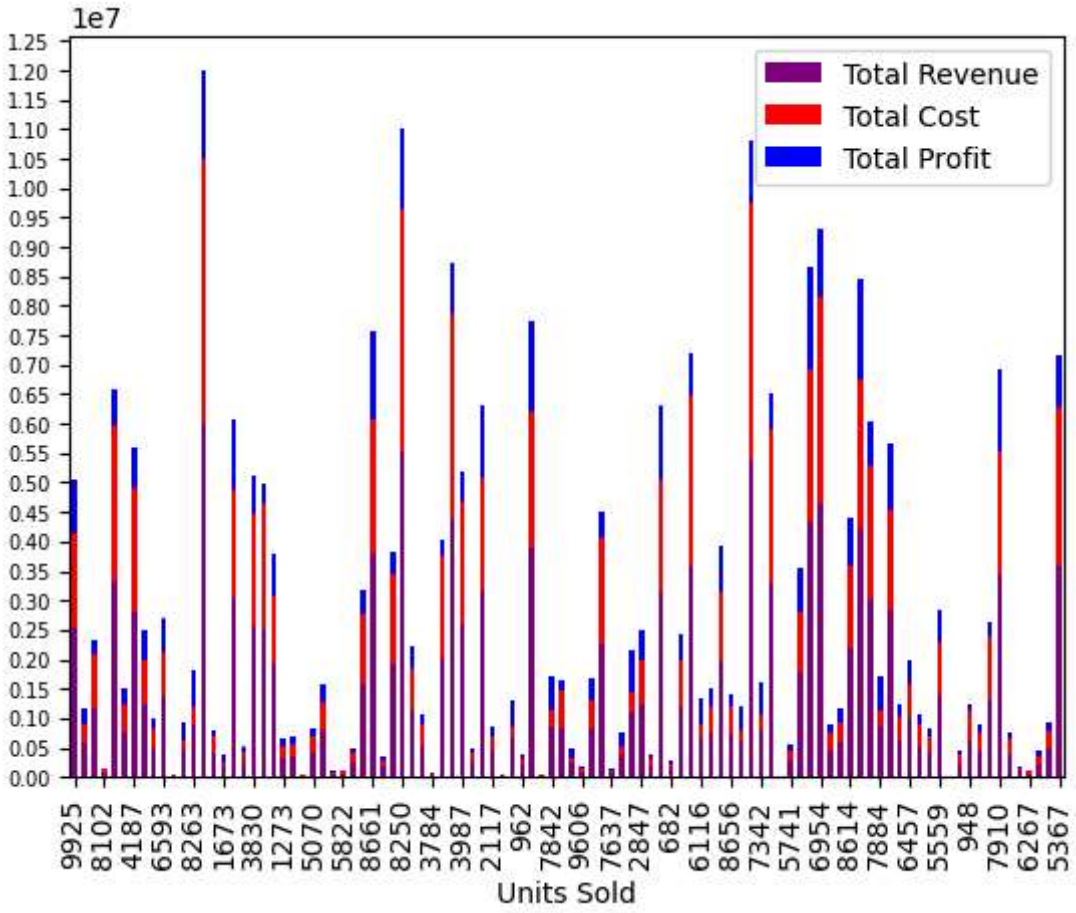
Out[185…
```
Text(0, 0.5, 'Total Revenue')
```



Maximum revenue has been generated when 5000-6500 units of a product were sold.

In [188…

```
bar_plot = df.plot.bar(x='Units Sold',y=['Total Revenue','Total Cost','Total Profit
plt.xticks(rotation=90)
plt.locator_params(nbins=38)
plt.tick_params(axis='y', which='major', labelsize=7)
```



In [190…
```
df['Item Type'].unique()
```

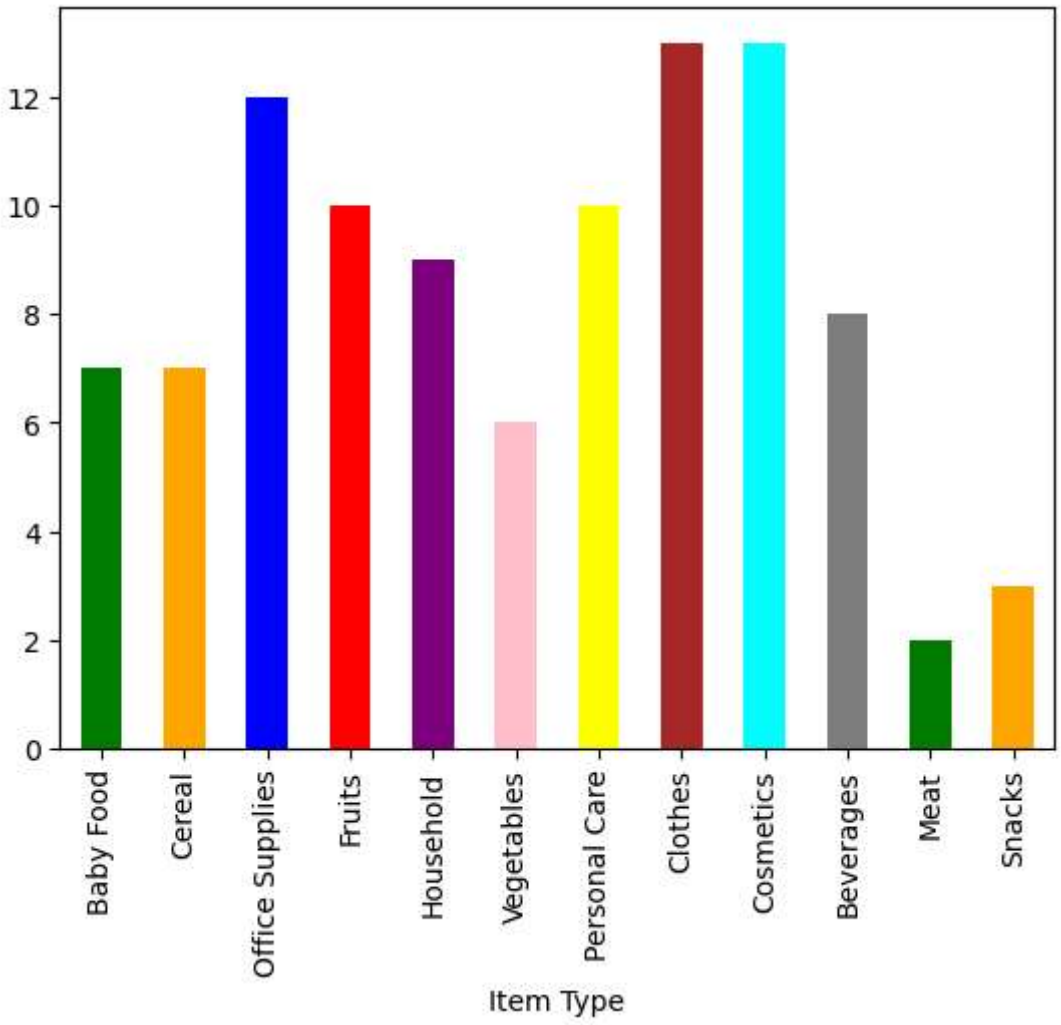
Out[190…
```
array(['Baby Food', 'Cereal', 'Office Supplies', 'Fruits', 'Household',
       'Vegetables', 'Personal Care', 'Clothes', 'Cosmetics', 'Beverages',
       'Meat', 'Snacks'], dtype=object)
```

In [198…
```
items = ['Baby Food', 'Cereal', 'Office Supplies', 'Fruits', 'Household',
       'Vegetables', 'Personal Care', 'Clothes', 'Cosmetics', 'Beverages',
       'Meat', 'Snacks']
```

In [200…
```
df['Item Type'] = pd.Categorical(df['Item Type'],categories=items,ordered=True)
df.groupby('Item Type')['Total Revenue'].count().plot(kind='bar',color=['green','or
```

```
C:\Users\nagpa\AppData\Local\Temp\ipykernel_24952\982783732.py:2: FutureWarning: The
default of observed=False is deprecated and will be changed to True in a future vers
ion of pandas. Pass observed=False to retain current behavior or observed=True to ad
opt the future default and silence this warning.
  df.groupby('Item Type')['Total Revenue'].count().plot(kind='bar',color=['green','o
range','blue','red','purple','pink','yellow','brown','aqua','grey'])
```

Out[200…
```
<Axes: xlabel='Item Type'>
```

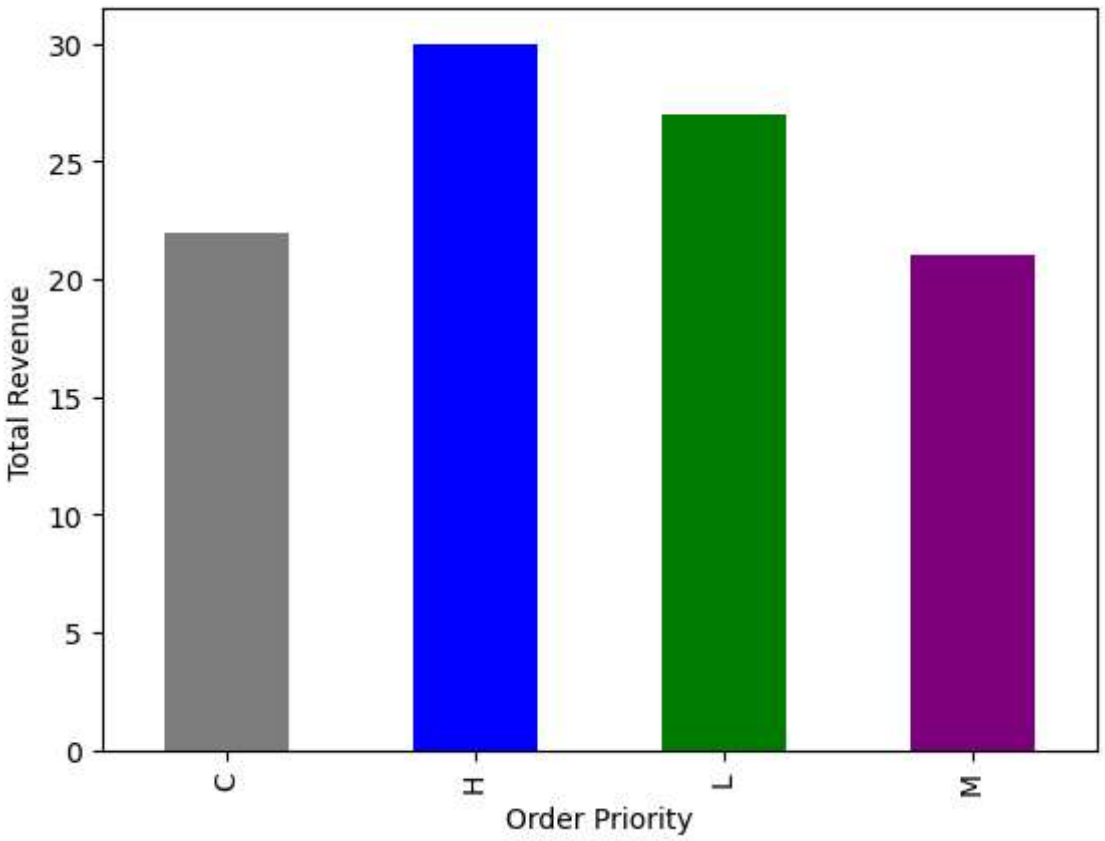


In [202…
```
df.loc[:,'Order Priority'].unique()
```

Out[202…
```
array(['H', 'C', 'L', 'M'], dtype=object)
```

In [119…
```
df.groupby('Order Priority')['Total Revenue'].count().plot(kind='bar',color=['grey'
plt.ylabel('Total Revenue')
```

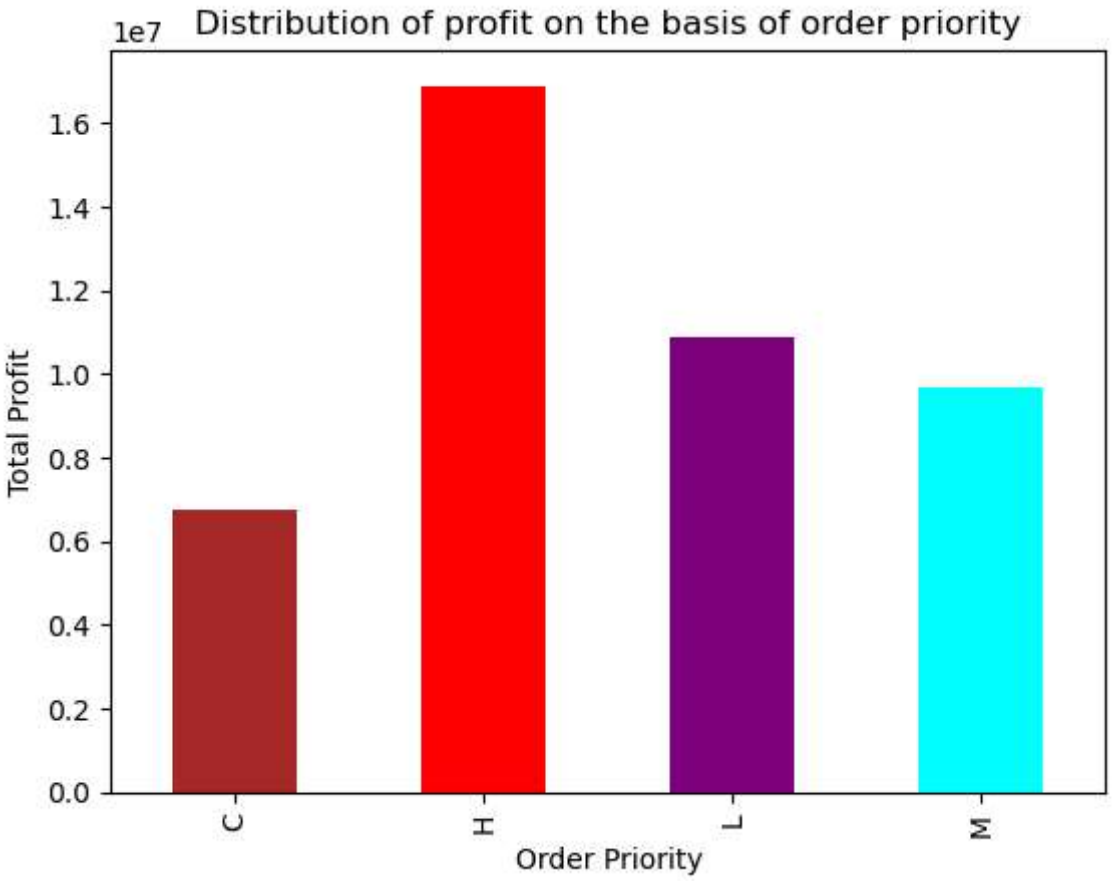
Out[119…
```
Text(0, 0.5, 'Total Revenue')
```



Maximum number of revenues has been generated by the products having order priority 'H' while minimum revenues has been generated by 'M' priority products.

In [122…
```
df.groupby('Order Priority')['Total Profit'].sum().plot(kind='bar',color=['brown','
plt.ylabel('Total Profit')
plt.title('Distribution of profit on the basis of order priority')
```

Out[122…
```
Text(0.5, 1.0, 'Distribution of profit on the basis of order priority')
```

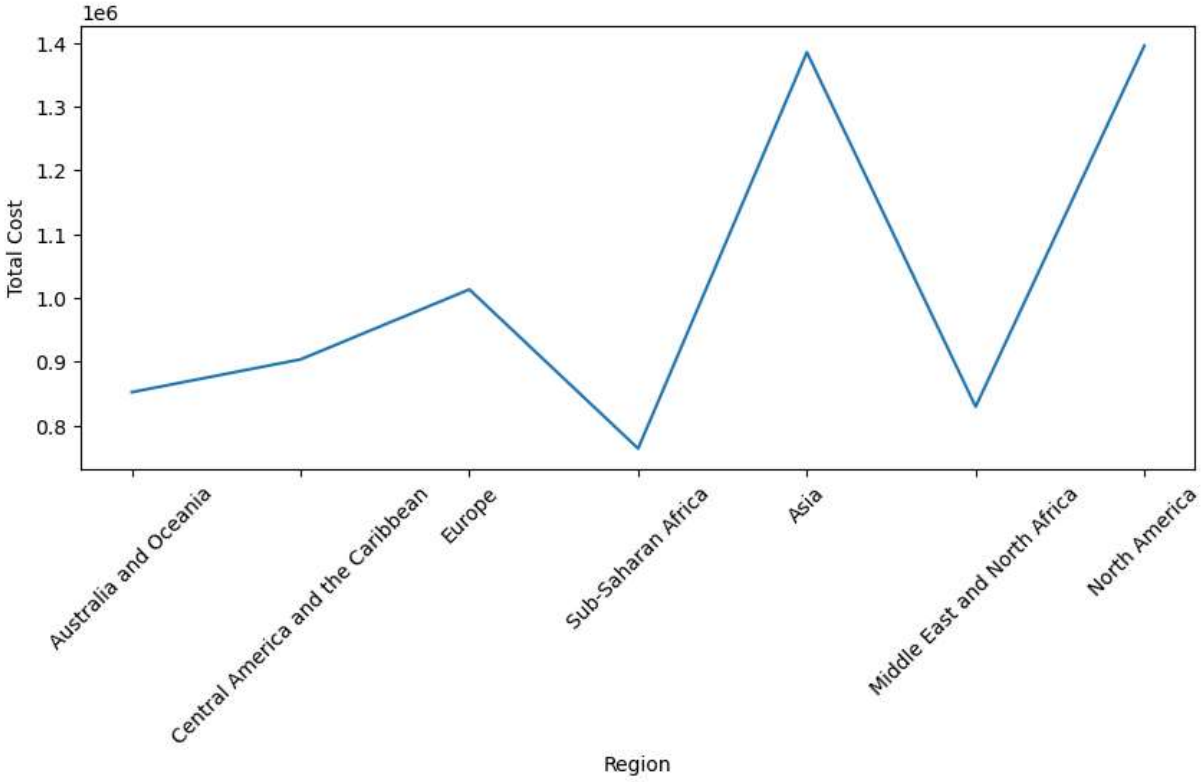


Maximum profit has been generated by products having order priority 'H' while minimum profit has been obtained in case of 'C' priority product orders.

In [86]:
```
plt.figure(figsize=(10,4))
sns.lineplot(x='Region',y='Total Cost',data=df,ci=None)
plt.xticks(rotation=45)
```

```
C:\Users\nagpa\AppData\Local\Temp\ipykernel_26680\831803164.py:2: FutureWarning:

The `ci` parameter is deprecated. Use `errorbar=None` for the same effect.

  sns.lineplot(x='Region',y='Total Cost',data=df,ci=None)
C:\Users\nagpa\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future version. Convert
inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\nagpa\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future version. Convert
inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```

Out[86]:
```
([0, 1, 2, 3, 4, 5, 6],
 [Text(0, 0, 'Australia and Oceania'),
  Text(1, 0, 'Central America and the Caribbean'),
  Text(2, 0, 'Europe'),
  Text(3, 0, 'Sub-Saharan Africa'),
  Text(4, 0, 'Asia'),
  Text(5, 0, 'Middle East and North Africa'),
  Text(6, 0, 'North America')])
```



Products have been much more expensive in Asia and North America in comparison to other continents.

In [112…
```
df['Country'].unique()
```

Out[112…
```
['Tuvalu', 'Grenada', 'Russia', 'Sao Tome and Principe', 'Rwanda', ..., 'Sloveni
a', 'Romania', 'Nicaragua', 'Malaysia', 'Mozambique']
Length: 76
Categories (76, object): ['Tuvalu' < 'Grenada' < 'Russia' < 'Sao Tome and Princip
e' ... 'Romania' < 'Nicaragua' < 'Malaysia' < 'Mozambique']
```

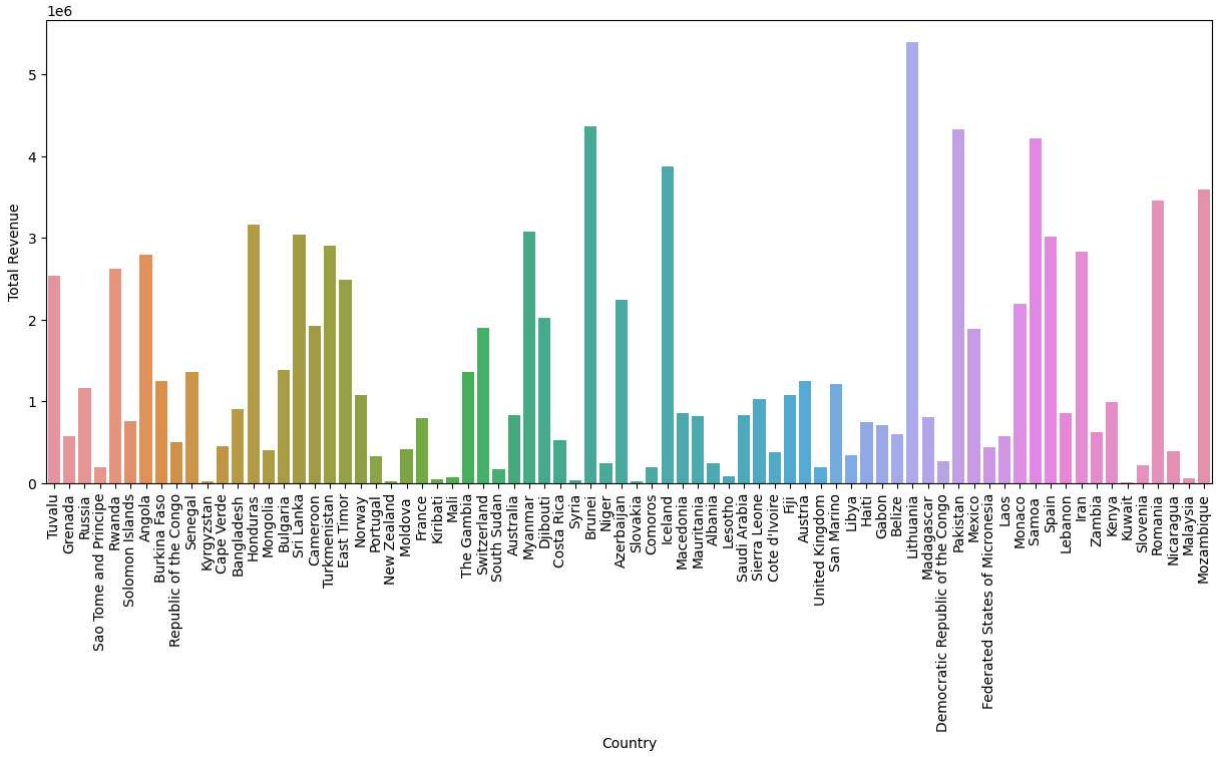
In [114…
```
countries = ['Tuvalu', 'Grenada', 'Russia', 'Sao Tome and Principe', 'Rwanda',
       'Solomon Islands', 'Angola', 'Burkina Faso',
       'Republic of the Congo', 'Senegal', 'Kyrgyzstan', 'Cape Verde',
       'Bangladesh', 'Honduras', 'Mongolia', 'Bulgaria', 'Sri Lanka',
       'Cameroon', 'Turkmenistan', 'East Timor', 'Norway', 'Portugal',
       'New Zealand', 'Moldova ', 'France', 'Kiribati', 'Mali',
       'The Gambia', 'Switzerland', 'South Sudan', 'Australia', 'Myanmar',
       'Djibouti', 'Costa Rica', 'Syria', 'Brunei', 'Niger', 'Azerbaijan',
       'Slovakia', 'Comoros', 'Iceland', 'Macedonia', 'Mauritania',
       'Albania', 'Lesotho', 'Saudi Arabia', 'Sierra Leone',
       "Cote d'Ivoire", 'Fiji', 'Austria', 'United Kingdom', 'San Marino',
       'Libya', 'Haiti', 'Gabon', 'Belize', 'Lithuania', 'Madagascar',
```

```
       'Democratic Republic of the Congo', 'Pakistan', 'Mexico',
       'Federated States of Micronesia', 'Laos', 'Monaco', 'Samoa ',
       'Spain', 'Lebanon', 'Iran', 'Zambia', 'Kenya', 'Kuwait',
       'Slovenia', 'Romania', 'Nicaragua', 'Malaysia', 'Mozambique']
```

In [116…

```
df['Country'] = pd.Categorical(df['Country'],categories=countries,ordered=True)
plt.figure(figsize=(15,6))
sns.barplot(x='Country', y='Total Revenue', data=df, ci=None)
plt.xticks(rotation=90)
plt.tick_params(axis='x', which='major', labelsize=10)
```

```
C:\Users\nagpa\AppData\Local\Temp\ipykernel_26680\3062323299.py:3: FutureWarning:

The `ci` parameter is deprecated. Use `errorbar=None` for the same effect.

  sns.barplot(x='Country', y='Total Revenue', data=df, ci=None)
C:\Users\nagpa\anaconda3\Lib\site-packages\seaborn\categorical.py:641: FutureWarnin
g: The default of observed=False is deprecated and will be changed to True in a futu
re version of pandas. Pass observed=False to retain current behavior or observed=Tru
e to adopt the future default and silence this warning.
  grouped_vals = vals.groupby(grouper)
```



Lithuania is the country where maximum revenue has been generated followed by Brunei.