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

```
sales_data = pd.read_csv('/content/Amazon Sales data.csv', parse_dates=['Order date','Ship date'])
```

```
sales_data.head()
```

	Region	Country	Item Type	Sales Channel	Order Priority	Order date	Ship date	Unit Price	Unit Cost	Total Revenue
0	Australia and Oceania	Tuvalu	Baby Food	Offline	H	2010-05-28	2010-06-27	255.28	159.42	2533654.0
1	Central America and the Caribbean	Grenada	Cereal	Online	C	2012-08-22	2012-09-15	205.70	117.11	576782.8
2	Europe	Russia	Office Supplies	Offline	L	2014-05-02	2014-05-08	651.21	524.96	1158502.5
3	Sub-Saharan Africa	Sao Tome and Principe	Fruits	Online	C	2014-06-20	2014-07-05	9.33	6.92	75591.6

```
sales_data.shape
```

```
(100, 14)
```

```
sales_data.columns
```

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

```
sales_data.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    datetime64[ns]
 6   Ship date       100 non-null    datetime64[ns]
 7   Unit Price      100 non-null    float64
 8   Unit Cost       100 non-null    float64
 9   Total Revenue   100 non-null    float64
 10  Total Cost      100 non-null    float64
 11  Total Profit    100 non-null    float64
 12  Order ID        100 non-null    int64
 13  Units Sold      100 non-null    int64
dtypes: datetime64[ns](2), float64(5), int64(2), object(5)
memory usage: 11.1+ KB
```

```
#checking no. of null values
sales_data.isnull().sum()
```

```
Region            0
Country           0
Item Type         0
Sales Channel     0
Order Priority    0
Order date        0
Ship date         0
```

```
Unit Price        0
Unit Cost         0
Total Revenue     0
Total Cost        0
Total Profit      0
Order ID          0
Units Sold        0
dtype: int64
```

```
sales_data.describe()
```

	Unit Price	Unit Cost	Total Revenue	Total Cost	Total Profit	Order ID	Un
count	100.000000	100.000000	1.000000e+02	1.000000e+02	1.000000e+02	1.000000e+02	10
mean	276.761300	191.048000	1.373488e+06	9.318057e+05	4.416820e+05	5.550204e+08	512
std	235.592241	188.208181	1.460029e+06	1.083938e+06	4.385379e+05	2.606153e+08	279
min	9.330000	6.920000	4.870260e+03	3.612240e+03	1.258020e+03	1.146066e+08	12
25%	81.730000	35.840000	2.687212e+05	1.688680e+05	1.214436e+05	3.389225e+08	283
50%	179.880000	107.275000	7.523144e+05	3.635664e+05	2.907680e+05	5.577086e+08	538
75%	437.200000	263.330000	2.212045e+06	1.613870e+06	6.358288e+05	7.907551e+08	736

```
sales_data1 = sales_data.copy()
```

```
# creating Year, Month, Quarter, Day Columns in sales_data1

sales_data1['Ship_Year']= sales_data['Ship date'].dt.year
sales_data1['Ship_Month']= sales_data['Ship date'].dt.month
sales_data1['Ship_Quarter']= sales_data['Ship date'].dt.quarter
sales_data1['Ship_Day']= sales_data['Ship date'].dt.day
```

```
sales_data1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 18 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    datetime64[ns]
 6   Ship date       100 non-null    datetime64[ns]
 7   Unit Price      100 non-null    float64
 8   Unit Cost       100 non-null    float64
 9   Total Revenue   100 non-null    float64
 10  Total Cost      100 non-null    float64
 11  Total Profit    100 non-null    float64
 12  Order ID        100 non-null    int64
 13  Units Sold      100 non-null    int64
 14  Ship_Year       100 non-null    int64
 15  Ship_Month      100 non-null    int64
 16  Ship_Quarter    100 non-null    int64
 17  Ship_Day        100 non-null    int64
dtypes: datetime64[ns](2), float64(5), int64(6), object(5)
memory usage: 14.2+ KB
```

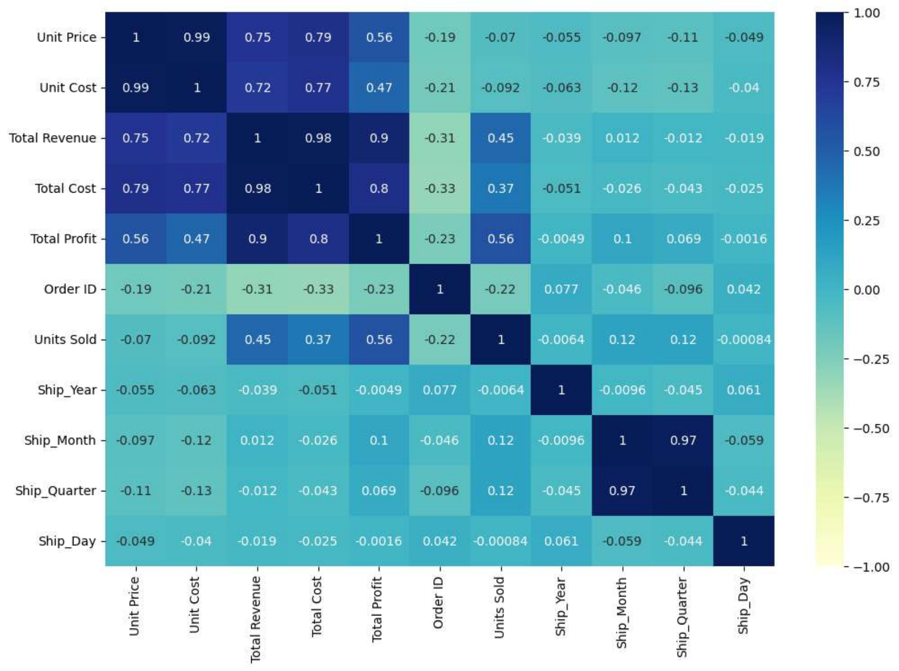
```
# Creating DataFrame only with necessary values.
sales_data2 = sales_data1[['Region', 'Country', 'Item Type', 'Sales Channel', 'Order Priority',
       'Order date', 'Ship date', 'Unit Price', 'Unit Cost', 'Total Revenue',
       'Total Cost', 'Total Profit', 'Order ID', 'Units Sold', 'Ship_Year', 'Ship_Month', 'Ship_Quarter', 'Ship_Day']]
```

```
sales_data2.isnull().sum()
```

```
Region            0
Country           0
Item Type         0
Sales Channel     0
Order Priority    0
Order date       0
Ship date        0
Unit Price       0
Unit Cost        0
Total Revenue    0
Total Cost       0
Total Profit     0
Order ID         0
Units Sold       0
Ship_Year        0
Ship_Month       0
Ship_Quarter     0
Ship_Day         0
dtype: int64
```

```
#checking the correlation
plt.figure(figsize=(12,8))
sns.heatmap(sales_data2.corr(method='pearson'),annot=True, vmin=-1, vmax=1, cmap='YlGnBu')
```

```
<ipython-input-15-a0931919ed9b>:3: FutureWarning: The default value of numeric_only in I
  sns.heatmap(sales_data2.corr(method='pearson'),annot=True, vmin=-1, vmax=1, cmap='YlGr
<Axes: >
```



	Unit Price	Unit Cost	Total Revenue	Total Cost	Total Profit	Order ID	Units Sold	Ship_Year	Ship_Month	Ship_Quarter	Ship_Day
Unit Price	1	0.99	0.75	0.79	0.56	-0.19	-0.07	-0.055	-0.097	-0.11	-0.049
Unit Cost	0.99	1	0.72	0.77	0.47	-0.21	-0.092	-0.063	-0.12	-0.13	-0.04
Total Revenue	0.75	0.72	1	0.98	0.9	-0.31	0.45	-0.039	0.012	-0.012	-0.019
Total Cost	0.79	0.77	0.98	1	0.8	-0.33	0.37	-0.051	-0.026	-0.043	-0.025
Total Profit	0.56	0.47	0.9	0.8	1	-0.23	0.56	-0.0049	0.1	0.069	-0.0016
Order ID	-0.19	-0.21	-0.31	-0.33	-0.23	1	-0.22	0.077	-0.046	-0.096	0.042
Units Sold	-0.07	-0.092	0.45	0.37	0.56	-0.22	1	-0.0064	0.12	0.12	-0.00084
Ship_Year	-0.055	-0.063	-0.039	-0.051	-0.0049	0.077	-0.0064	1	-0.0096	-0.045	0.061
Ship_Month	-0.097	-0.12	0.012	-0.026	0.1	-0.046	0.12	-0.0096	1	0.97	-0.059
Ship_Quarter	-0.11	-0.13	-0.012	-0.043	0.069	-0.096	0.12	-0.045	0.97	1	-0.044
Ship_Day	-0.049	-0.04	-0.019	-0.025	-0.0016	0.042	-0.00084	0.061	-0.059	-0.044	1

```
sales_data2.head()
```

	Region	Country	Item Type	Sales Channel	Order Priority	Order date	Ship date	Unit Price	Unit Cost	Total Revenu
0	Australia and Oceania	Tuvalu	Baby Food	Offline	H	2010-05-28	2010-06-27	255.28	159.42	2533654.0
1	Central America and the Caribbean	Grenada	Cereal	Online	C	2012-08-22	2012-09-15	205.70	117.11	576782.8

```
sales_data2.tail()
```

	Region	Country	Item Type	Sales Channel	Order Priority	Order date	Ship date	Unit Price	Unit Cost	Re
95	Sub-Saharan Africa	Mali	Clothes	Online	M	2011-07-26	2011-09-03	109.28	35.84	97
96	Asia	Malaysia	Fruits	Offline	L	2011-11-11	2011-12-28	9.33	6.92	58
97	Sub-Saharan Africa	Sierra Leone	Vegetables	Offline	C	2016-06-01	2016-06-29	154.06	90.93	228
98	North America	Mexico	Personal Care	Offline	M	2015-07-30	2015-08-08	81.73	56.67	471
99	Sub-Saharan Africa	Mozambique	Household	Offline	L	2012-02-10	2012-02-15	668.27	502.54	3586

```
sales_data2.rename(columns = {'Item Type':'Item'}, inplace = True)
```

```
sales_data2.head()
```

	Region	Country	Item	Sales Channel	Order Priority	Order date	Ship date	Unit Price	Unit Cost	Total Revenu
0	Australia and Oceania	Tuvalu	Baby Food	Offline	H	2010-05-28	2010-06-27	255.28	159.42	2533654.0
1	Central America and the Caribbean	Grenada	Cereal	Online	C	2012-08-22	2012-09-15	205.70	117.11	576782.8
2	Europe	Russia	Office Supplies	Offline	L	2014-05-02	2014-05-08	651.21	524.96	1158502.5
3	Sub-Saharan Africa	Sao Tome and Principe	Fruits	Online	C	2014-06-20	2014-07-05	9.33	6.92	75591.6
4	Sub-Saharan Africa	Rwanda	Office Supplies	Offline	L	2013-02-01	2013-02-06	651.21	524.96	3296425.0

```
sales_data2.Item.value_counts()
```

```
Clothes            13
Cosmetics          13
Office Supplies    12
Fruits             10
Personal Care      10
Household           9
Beverages           8
Baby Food           7
Cereal              7
Vegetables          6
Snacks              3
Meat                2
Name: Item, dtype: int64
```

```
sales_data2.describe()
```

	Unit Price	Unit Cost	Total Revenue	Total Cost	Total Profit	Order ID	Un
count	100.000000	100.000000	1.000000e+02	1.000000e+02	1.000000e+02	1.000000e+02	10
mean	276.761300	191.048000	1.373488e+06	9.318057e+05	4.416820e+05	5.550204e+08	512
std	235.592241	188.208181	1.460029e+06	1.083938e+06	4.385379e+05	2.606153e+08	279
min	9.330000	6.920000	4.870260e+03	3.612240e+03	1.258020e+03	1.146066e+08	12
25%	81.730000	35.840000	2.687212e+05	1.688680e+05	1.214436e+05	3.389225e+08	283
50%	179.880000	107.275000	7.523144e+05	3.635664e+05	2.907680e+05	5.577086e+08	538
75%	437.200000	263.330000	2.212045e+06	1.613870e+06	6.358288e+05	7.907551e+08	736
max	668.270000	524.960000	5.997055e+06	4.509794e+06	1.719922e+06	9.940222e+08	992

Yearly Sales Record:

```
sales_data2.columns
```

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

```
Yearly_Sales =  sales_data2[['Region', 'Country', 'Item', 'Sales Channel', 'Order Priority',
       'Order date', 'Ship date', 'Unit Price', 'Unit Cost', 'Total Revenue',
       'Total Cost', 'Total Profit', 'Order ID', 'Units Sold', 'Ship_Year',
       'Ship_Month',]]
```

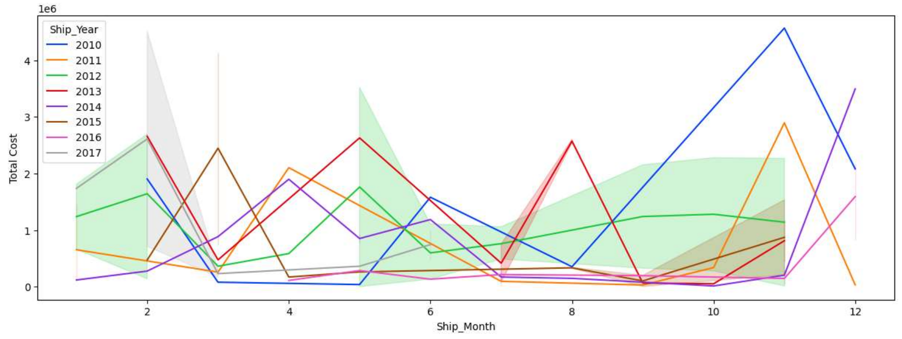
```
Yearly_Sales01 = Yearly_Sales.groupby('Ship_Year').sum().reset_index()
sns.catplot(y = 'Total Cost', x = 'Ship_Year', data = Yearly_Sales01, palette='Reds',kind="bar")
plt.xlabel('Year')
plt.ylabel('Sales Amount')
plt.title('Yearly Sales')
Yearly_Sales01[['Ship_Year', 'Total Cost']]
```

```
<ipython-input-24-e1d6983f8b03>:1: FutureWarning: The default value of numeric_only in D
  Yearly_Sales01 = Yearly_Sales.groupby('Ship_Year').sum().reset_index()
```

	Ship_Year	Total Cost
0	2010	10616258.38
1	2011	8513570.23
2	2012	24500421.12
3	2013	13494604.87
4	2014	10871176.50
5	2015	8421442.42

```
plt.figure(figsize = (15,5))
sns.lineplot(y ='Total Cost', x = 'Ship_Month',
             data= sales_data2.groupby(['Ship date','Ship_Year','Ship_Month']).sum(),
             hue = 'Ship_Year', palette='bright')
```

```
<ipython-input-25-3f26a3cd64c8>:3: FutureWarning: The default value of numeric_only in D
  data= sales_data2.groupby(['Ship date','Ship_Year','Ship_Month']).sum(),
<Axes: xlabel='Ship_Month', ylabel='Total Cost'>
```



```
#Plotting Piechart to know Sales Share among 3 years
plt.figure(figsize=(17,6))
plt.pie('Total Cost',labels='Ship_Year',data=Yearly_Sales01,
        autopct='%1.2f%%',shadow=True,startangle=90)
plt.axis('equal')
plt.title('Sales Contribution')
plt.legend(round(Yearly_Sales01['Total Cost'],2), loc=7, fontsize = 'xx-large')
plt.show()
```

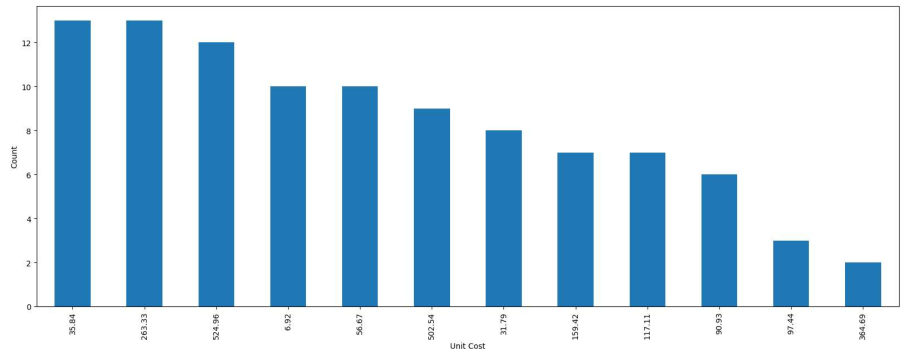
Sales Contribution

```
sales_data2.columns
```

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

```
plt.figure(figsize=(20, 7))
sales_data2['Unit Cost'].value_counts().plot.bar()
plt.xlabel('Unit Cost')
plt.ylabel('Count')
```

```
Text(0, 0.5, 'Count')
```



Yearly-Monthwise Records:

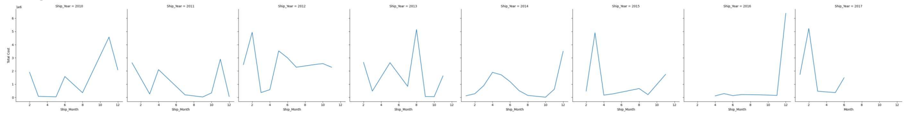
```
Yearly_Monthwise_Sales = sales_data2.groupby(['Ship_Year','Ship_Month']).sum().reset_index()
Yearly_Monthwise_Sales.describe()
```

```
<ipython-input-29-9044f006f830>:1: FutureWarning: The default value of numeric_only in [
  Yearly_Monthwise_Sales = sales_data2.groupby(['Ship_Year','Ship_Month']).sum().reset_i
```

	Ship_Year	Ship_Month	Unit Price	Unit Cost	Total Revenue	Total Cost	
count	63.000000	63.000000	63.000000	63.000000	6.300000e+01	6.300000e+01	6.30
mean	2013.301587	6.333333	439.303651	303.250794	2.180139e+06	1.479057e+06	7.01
std	2.129932	3.426651	444.971868	328.425455	2.279387e+06	1.588239e+06	7.63
min	2010.000000	1.000000	9.330000	6.920000	2.040471e+04	1.513404e+04	5.27
25%	2012.000000	3.000000	109.280000	56.670000	3.937805e+05	2.122014e+05	1.27
50%	2013.000000	6.000000	255.280000	159.420000	1.352371e+06	6.708028e+05	5.39
75%	2015.000000	9.000000	664.405000	524.960000	3.212633e+06	2.382881e+06	9.66
max	2017.000000	12.000000	1987.750000	1530.040000	1.037040e+07	6.372028e+06	3.99

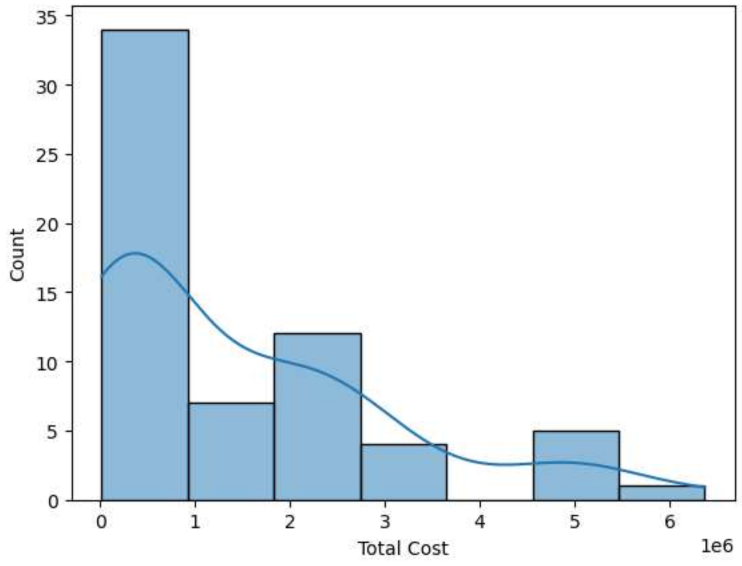
```
plt.figure(figsize=(20, 9))
sns.relplot(x ='Ship_Month',y = 'Total Cost', data=Yearly_Monthwise_Sales,height=5,
           kind = 'line', aspect = 1, col = 'Ship_Year')
plt.xlabel('Month')
plt.ylabel('Total Cost')
print('*'*50+'Yearly-Monthwise Sales Trend'+'*'*50)
```

```
**************************************************Yearly-Monthwise Sales Trend*********
<Figure size 2000x900 with 0 Axes>
```



```
sns.histplot(Yearly_Monthwise_Sales['Total Cost'], kde = True)
```

```
<Axes: xlabel='Total Cost', ylabel='Count'>
```



Monthly Records:

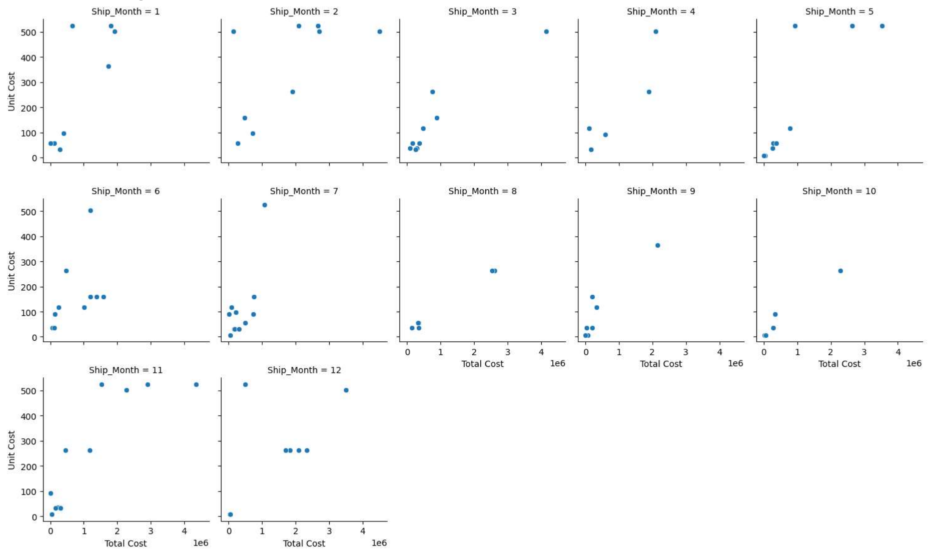
```
Monthly_sales = sales_data2.groupby(['Ship_Year','Ship_Month']).sum().reset_index()
Monthly_sales.describe()
```

```
<ipython-input-32-d05d0eb8a28c>:1: FutureWarning: The default value of numeric_only in D
  Monthly_sales = sales_data2.groupby(['Ship_Year','Ship_Month']).sum().reset_index()
```

	Ship_Year	Ship_Month	Unit Price	Unit Cost	Total Revenue	Total Cost	
count	63.000000	63.000000	63.000000	63.000000	6.300000e+01	6.300000e+01	6.30
mean	2013.301587	6.333333	439.303651	303.250794	2.180139e+06	1.479057e+06	7.01
std	2.129932	3.426651	444.971868	328.425455	2.279387e+06	1.588239e+06	7.63
min	2010.000000	1.000000	9.330000	6.920000	2.040471e+04	1.513404e+04	5.27
25%	2012.000000	3.000000	109.280000	56.670000	3.937805e+05	2.122014e+05	1.27
50%	2013.000000	6.000000	255.280000	159.420000	1.352371e+06	6.708028e+05	5.39
75%	2015.000000	9.000000	664.405000	524.960000	3.212633e+06	2.382881e+06	9.66
max	2017.000000	12.000000	1987.750000	1530.040000	1.037040e+07	6.372028e+06	3.99

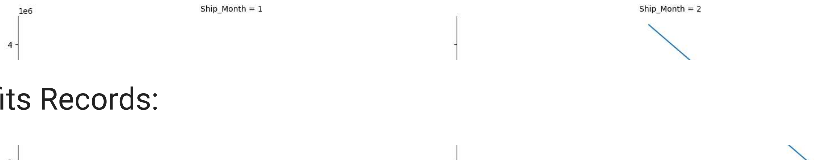
```
sns.relplot(y ='Unit Cost',x = 'Total Cost', data=sales_data2,height = 3,aspect=1,
          col = 'Ship_Month',col_wrap=5, palette='muted')
```

```
<ipython-input-33-0f39edd47b9a>:1: UserWarning: Ignoring `palette` because no `hue` variab
  sns.relplot(y ='Unit Cost',x = 'Total Cost', data=sales_data2,height = 3,aspect=1,
<seaborn.axisgrid.FacetGrid at 0x7f5a11cfa470>
```



```
plt.figure(figsize=(8, 20))
sns.relplot(x ='Ship_Day',y = 'Total Cost', data= sales_data2.query('Ship_Year == 2017'),
            kind = 'line', col = 'Ship_Month', col_wrap =2, height = 4, aspect = 2)
plt.ylabel('Total Cost')
print('*'*50+'Monthly Sales Trend in 2017'+'*'*50)
```

```
**************************************************Monthly Sales Trend in 2017***********
<Figure size 800x2000 with 0 Axes>
```



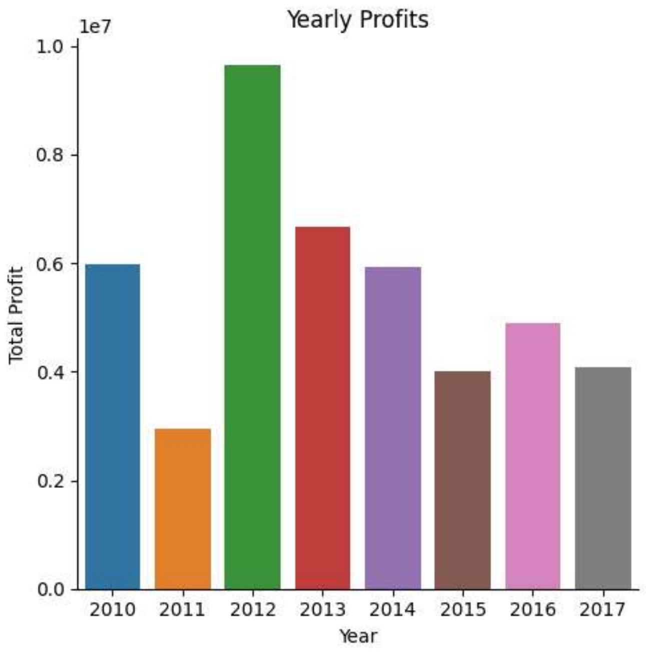
Profits Records:

```
sales_data2.columns
```

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

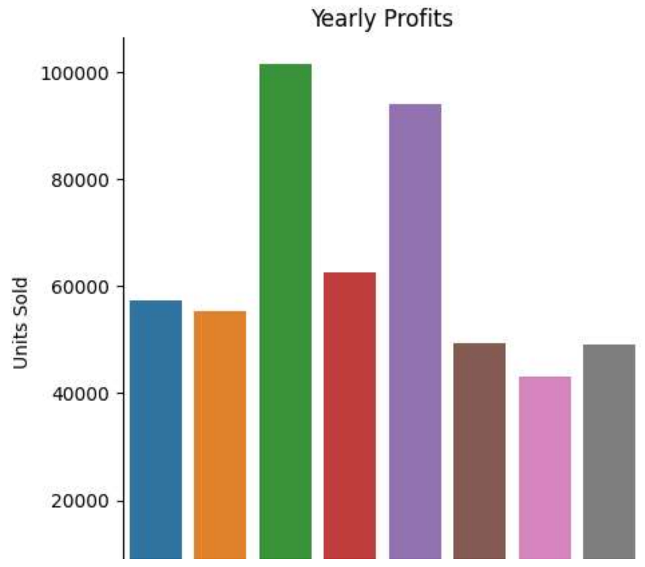
```
sns.catplot(y = 'Total Profit', x = 'Ship_Year', data = Yearly_Sales01,kind="bar")
plt.xlabel('Year')
plt.ylabel('Total Profit')
plt.title('Yearly Profits')
Yearly_Sales01[['Ship_Year', 'Total Profit']]
```

	Ship_Year	Total Profit
0	2010	5987980.15
1	2011	2946149.26
2	2012	9649456.37
3	2013	6662167.54
4	2014	5932714.18
5	2015	3996539.44
6	2016	4903838.01
7	2017	4089353.45



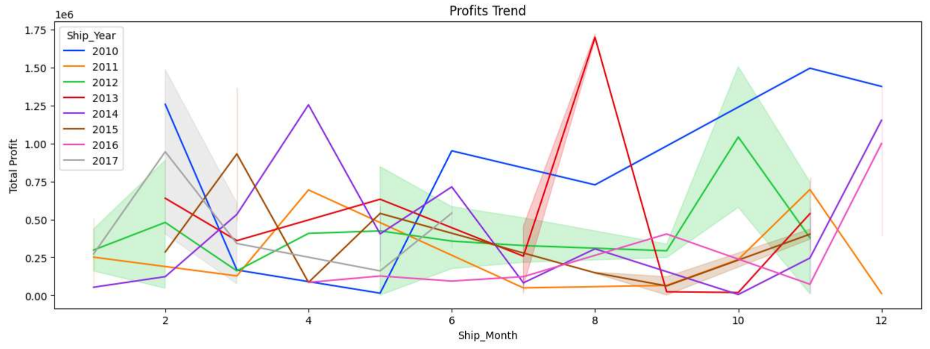
```
sns.catplot(y = 'Units Sold', x = 'Ship_Year', data = Yearly_Sales01,kind="bar")
plt.xlabel('Year')
plt.ylabel('Units Sold')
plt.title('Yearly Profits')
Yearly_Sales01[['Ship_Year', 'Units Sold']]
```

	Ship_Year	Units Sold
0	2010	57468
1	2011	55414
2	2012	101424
3	2013	62538
4	2014	94165
5	2015	49480
6	2016	43156
7	2017	49226

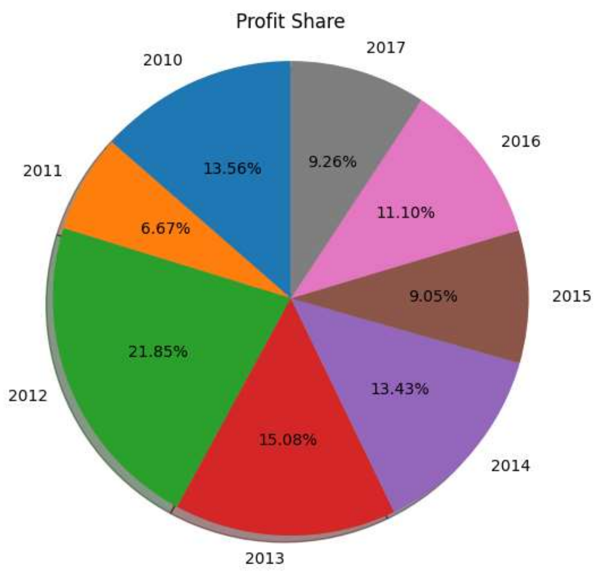


```
plt.figure(figsize = (15,5))
sns.lineplot(y ='Total Profit', x = 'Ship_Month',
             data= sales_data2.groupby(['Ship date','Ship_Year','Ship_Month']).sum(),
             hue = 'Ship_Year', palette='bright')
plt.title('Profits Trend')
plt.show()
```

```
<ipython-input-42-b46d43cd2da9>:3: FutureWarning: The default value of numeric_only in Dat
  data= sales_data2.groupby(['Ship date','Ship_Year','Ship_Month']).sum(),
```



```
plt.figure(figsize=(10,6))
plt.pie('Total Profit',labels='Ship_Year',data=Yearly_Sales01[['Ship_Year', 'Total Profit']],
        autopct='%1.2f%%',shadow=True,startangle=90)
plt.axis('equal')
plt.title('Profit Share')
plt.show()
```



Top 10 Records:

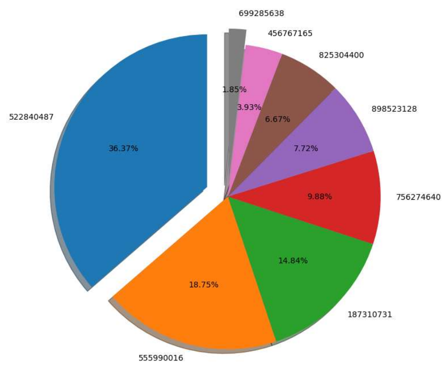
```
Top10byCustKey17 = Yearly_Sales[Yearly_Sales['Ship_Year']==2017].groupby(['Ship_Year','Order ID']).sum()
Top10byCustKey17 = Top10byCustKey17.sort_values('Total Profit',ascending = False).reset_index().head(10)
```

```
<ipython-input-56-3119354ad5a2>:1: FutureWarning: The default value of numeric_only in DataFrameGroupBy.sum is deprecated. In a future v
  Top10byCustKey17 = Yearly_Sales[Yearly_Sales['Ship_Year']==2017].groupby(['Ship_Year','Order ID']).sum()
```

```
plt.figure(figsize=(10,5))
sns.barplot(x='Order ID', y='Total Profit',data = Top10byCustKey17, palette = 'turbo')
plt.title('Top 10 Order ID by Total Profit')
Top10byCustKey17[['Order ID', 'Total Profit']]
```

	Order ID	Total Profit
0	522840487	1487261.02
1	555990016	766835.04
2	187310731	606834.72
3	756274640	404010.78
4	898523128	315574.05
5	825304400	272672.40

```
plt.figure(figsize=(20,8))
plt.pie('Total Profit',labels='Order ID',data=Top10byCustKey17,
        autopct='%1.2f%%',shadow=True,startangle=90, explode = ( 0.15, 0, 0, 0, 0, 0, 0, 0.1))
plt.axis('equal')
plt.show()
```

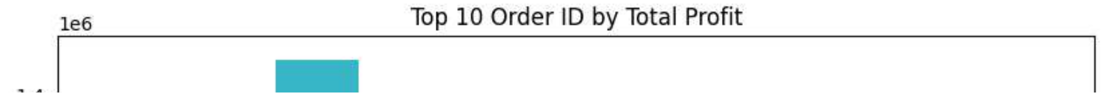


```
Top10byCustKey12 = Yearly_Sales[Yearly_Sales['Ship_Year']==2012].groupby(['Ship_Year','Order ID']).sum()
Top10byCustKey12 = Top10byCustKey12.sort_values('Total Profit',ascending = False).reset_index().head(10)
```

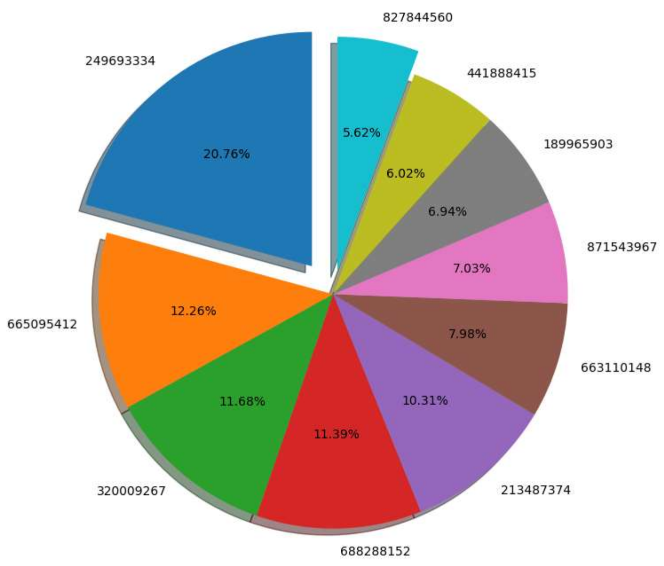
```
<ipython-input-69-62c07c2908ec>:1: FutureWarning: The default value of numeric_only in DataFrameGroupBy.sum is deprecated. In a future v
  Top10byCustKey12 = Yearly_Sales[Yearly_Sales['Ship_Year']==2012].groupby(['Ship_Year','Order ID']).sum()
```

```
plt.figure(figsize=(10,5))
sns.barplot(x='Order ID', y='Total Profit',data = Top10byCustKey12, palette = 'turbo')
plt.title('Top 10 Order ID by Total Profit')
Top10byCustKey12[['Order ID', 'Total Profit']]
```

	Order ID	Total Profit
0	249693334	1505888.07
1	665095412	889472.91
2	320009267	846885.00
3	688288152	825738.04
4	213487374	747939.49
5	663110148	579000.96
6	871543967	510216.66
7	189965903	503358.75
8	441888415	436446.25
9	827844560	407630.41



```
plt.figure(figsize=(20,8))
plt.pie('Total Profit',labels='Order ID',data=Top10byCustKey12,
        autopct='%1.2f%%',shadow=True,startangle=90, explode = ( 0.15, 0,0,0, 0, 0, 0, 0, 0, 0.1))
plt.axis('equal')
plt.show()
```


827844560
441888415
189965903
871543967
663110148
213487374
688288152
320009267
665095412
249693334
5.62%
6.02%
6.94%
7.03%
7.98%
10.31%
11.39%
11.68%
12.26%
20.76%