In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
data=pd.read_csv('supply_chain_data.csv')
data.head()
```

Out[2]:

	Product type	SKU	Price	Availability	Number of products sold	Revenue generated	Customer demographics	Stock levels	Lead times	Or quanti
0	haircare	SKU0	69.808006	55	802	8661.996792	Non-binary	58	7	
1	skincare	SKU1	14.843523	95	736	7460.900065	Female	53	30	
2	haircare	SKU2	11.319683	34	8	9577.749626	Unknown	1	10	
3	skincare	SKU3	61.163343	68	83	7766.836426	Non-binary	23	13	
4	skincare	SKU4	4.805496	26	871	2686.505152	Non-binary	5	3	

5 rows × 24 columns

In [3]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 24 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   Product type             100 non-null    object 
 1   SKU                      100 non-null    object 
 2   Price                    100 non-null    float64
 3   Availability             100 non-null    int64  
 4   Number of products sold  100 non-null    int64  
 5   Revenue generated        100 non-null    float64
 6   Customer demographics    100 non-null    object 
 7   Stock levels             100 non-null    int64  
 8   Lead times               100 non-null    int64  
 9   Order quantities         100 non-null    int64  
 10  Shipping times           100 non-null    int64  
 11  Shipping carriers        100 non-null    object 
 12  Shipping costs           100 non-null    float64
 13  Supplier name            100 non-null    object 
 14  Location                 100 non-null    object 
 15  Lead time                100 non-null    int64  
 16  Production volumes       100 non-null    int64  
 17  Manufacturing lead time  100 non-null    int64  
 18  Manufacturing costs      100 non-null    float64
 19  Inspection results       100 non-null    object 
 20  Defect rates             100 non-null    float64
 21  Transportation modes     100 non-null    object 
 22  Routes                   100 non-null    object 
 23  Costs                    100 non-null    float64
dtypes: float64(6), int64(9), object(9)
memory usage: 18.9+ KB
```

In [4]:
```
print('The number of rows and columns in the datset',data.shape)
```

```
The number of rows and columns in the datset (100, 24)
```

In [5]:
```
data.describe().style.background_gradient(cmap='winter_r')
```

Out[5]:

	Price	Availability	Number of products sold	Revenue generated	Stock levels	Lead times	Order quantities	Shippi tim
count	100.000000	100.000000	100.000000	100.000000	100.000000	100.000000	100.000000	100.0000
mean	49.462461	48.400000	460.990000	5776.048187	47.770000	15.960000	49.220000	5.7500
std	31.168193	30.743317	303.780074	2732.841744	31.369372	8.785801	26.784429	2.7242
min	1.699976	1.000000	8.000000	1061.618523	0.000000	1.000000	1.000000	1.0000
25%	19.597823	22.750000	184.250000	2812.847151	16.750000	8.000000	26.000000	3.7500
50%	51.239831	43.500000	392.500000	6006.352023	47.500000	17.000000	52.000000	6.0000
75%	77.198228	75.000000	704.250000	8253.976921	73.000000	24.000000	71.250000	8.0000
max	99.171329	100.000000	996.000000	9866.465458	100.000000	30.000000	96.000000	10.0000

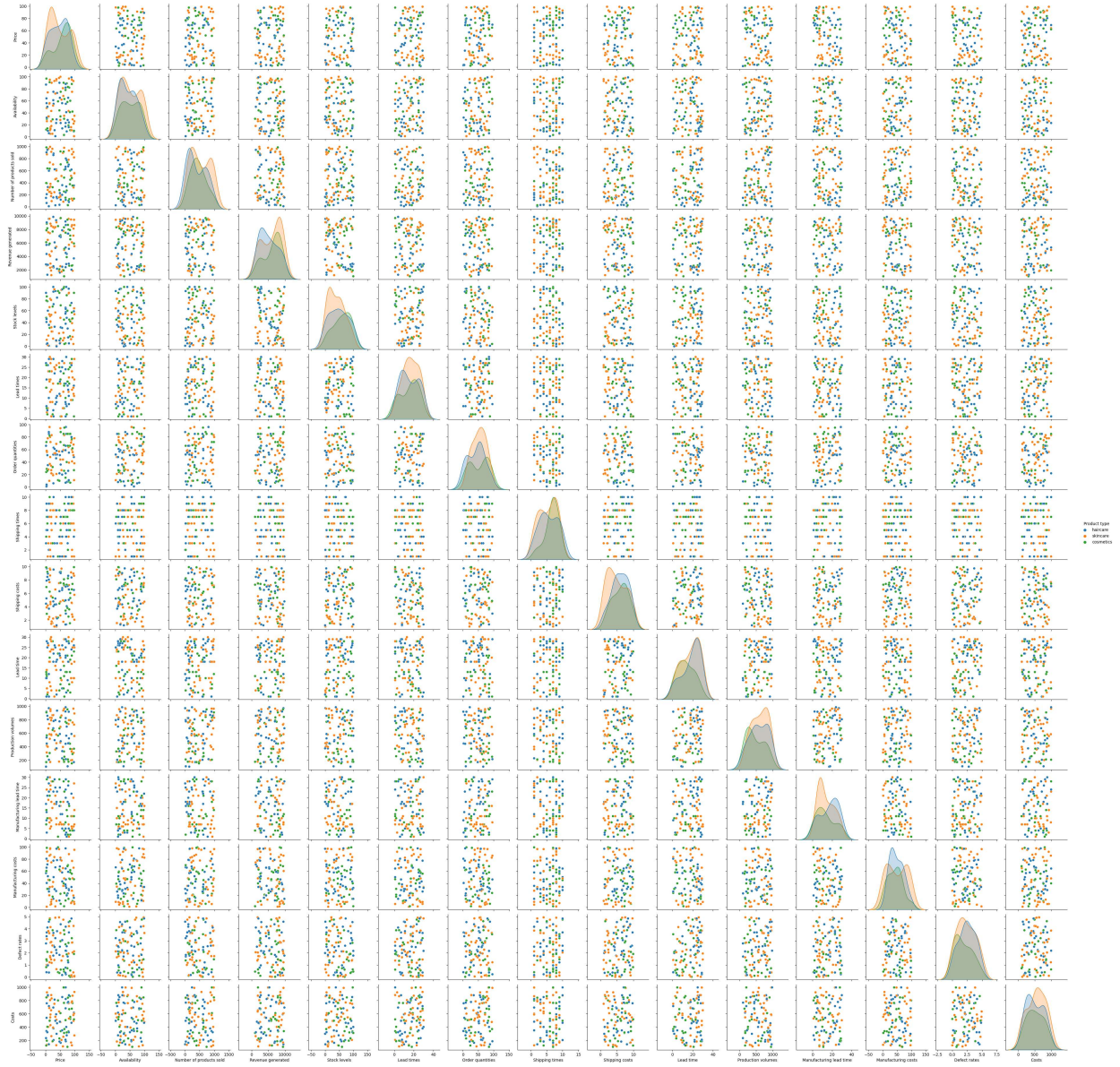
In [6]:
```
plt.figure(figsize=(16,7))
sns.heatmap(data.corr(),annot=True,cmap='BuPu_r')
plt.show()
```

In [7]:
```
#Checking the null values
data.isna().sum()/len(data)
```

Out[7]:
```
Product type                0.0
SKU                         0.0
Price                       0.0
Availability                0.0
Number of products sold     0.0
Revenue generated           0.0
Customer demographics       0.0
Stock levels                0.0
Lead times                  0.0
Order quantities            0.0
Shipping times              0.0
Shipping carriers           0.0
Shipping costs              0.0
Supplier name               0.0
Location                    0.0
Lead time                   0.0
Production volumes          0.0
Manufacturing lead time     0.0
Manufacturing costs         0.0
Inspection results          0.0
Defect rates                0.0
Transportation modes        0.0
Routes                      0.0
Costs                       0.0
dtype: float64
```

In [8]:
```
sns.pairplot(data,hue='Product type')
```

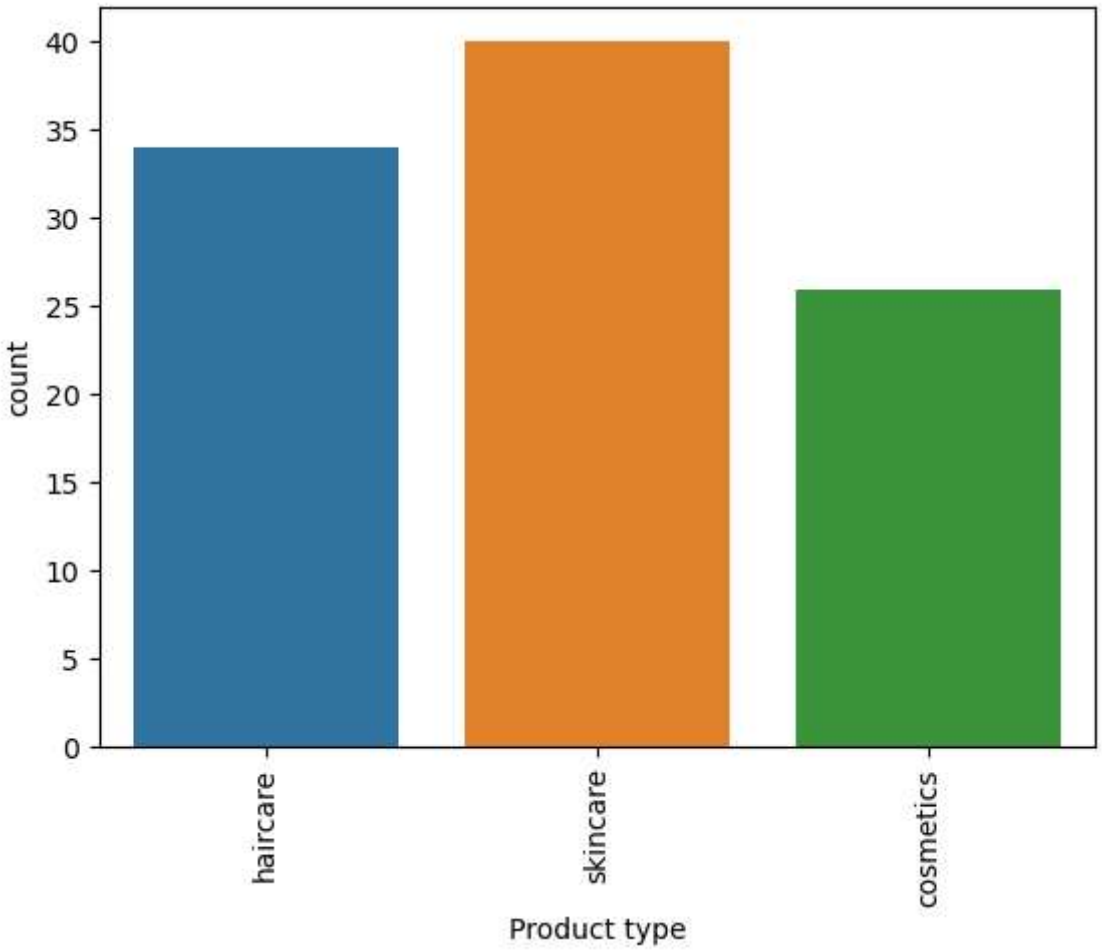
Out[8]:
```
<seaborn.axisgrid.PairGrid at 0x13ca9617be0>
```

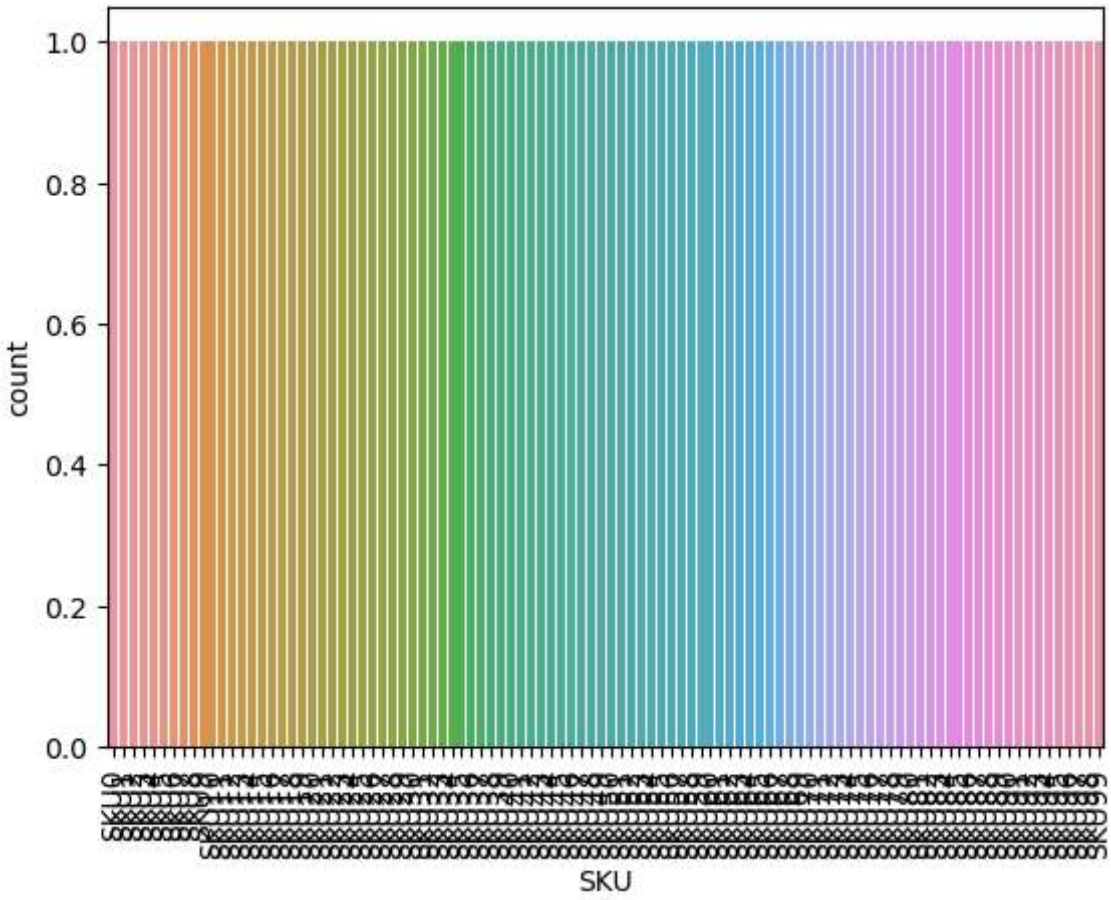


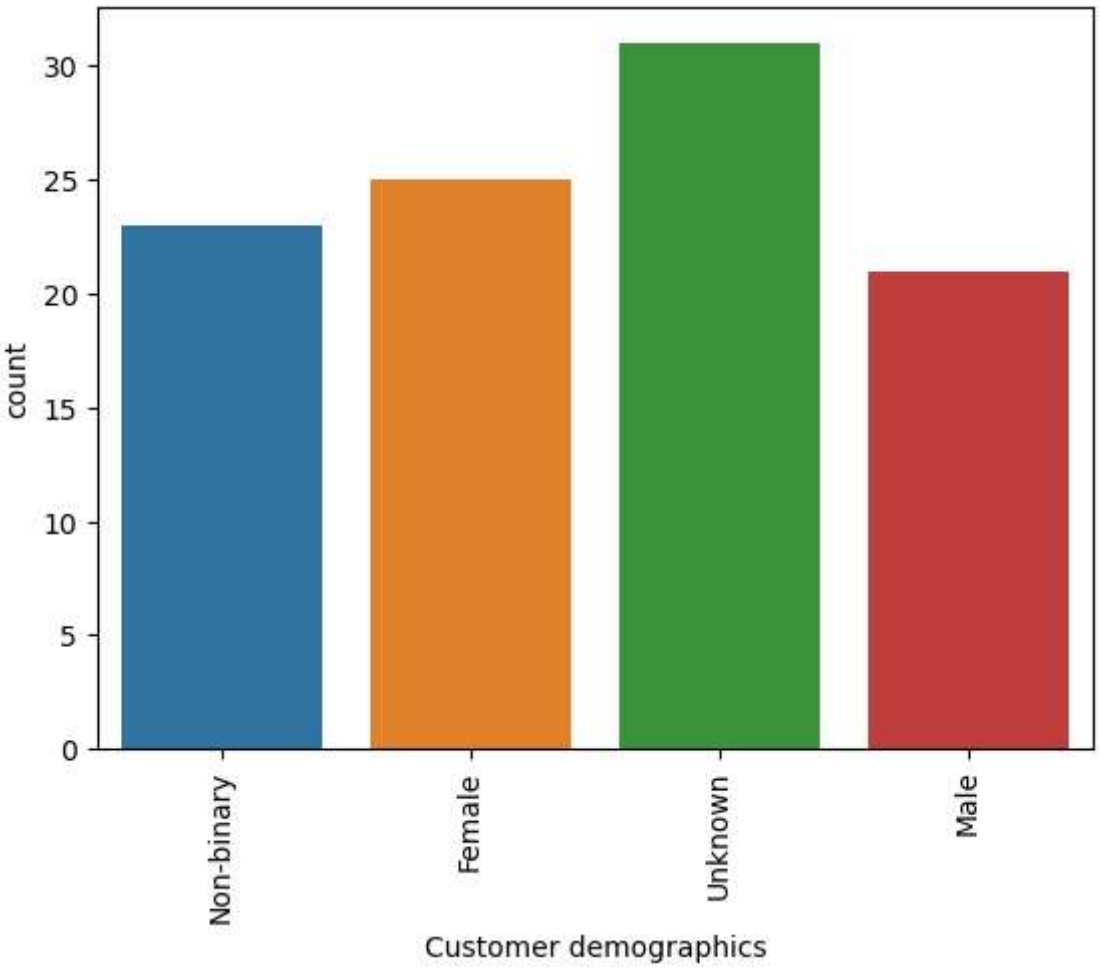
In [9]:
```
#Let's separate the categorical and numerical columns
categorical=[i for i in data.columns if data[i].dtypes=='object']
numerical=[i for i in data.columns if data[i].dtypes!='object']
```

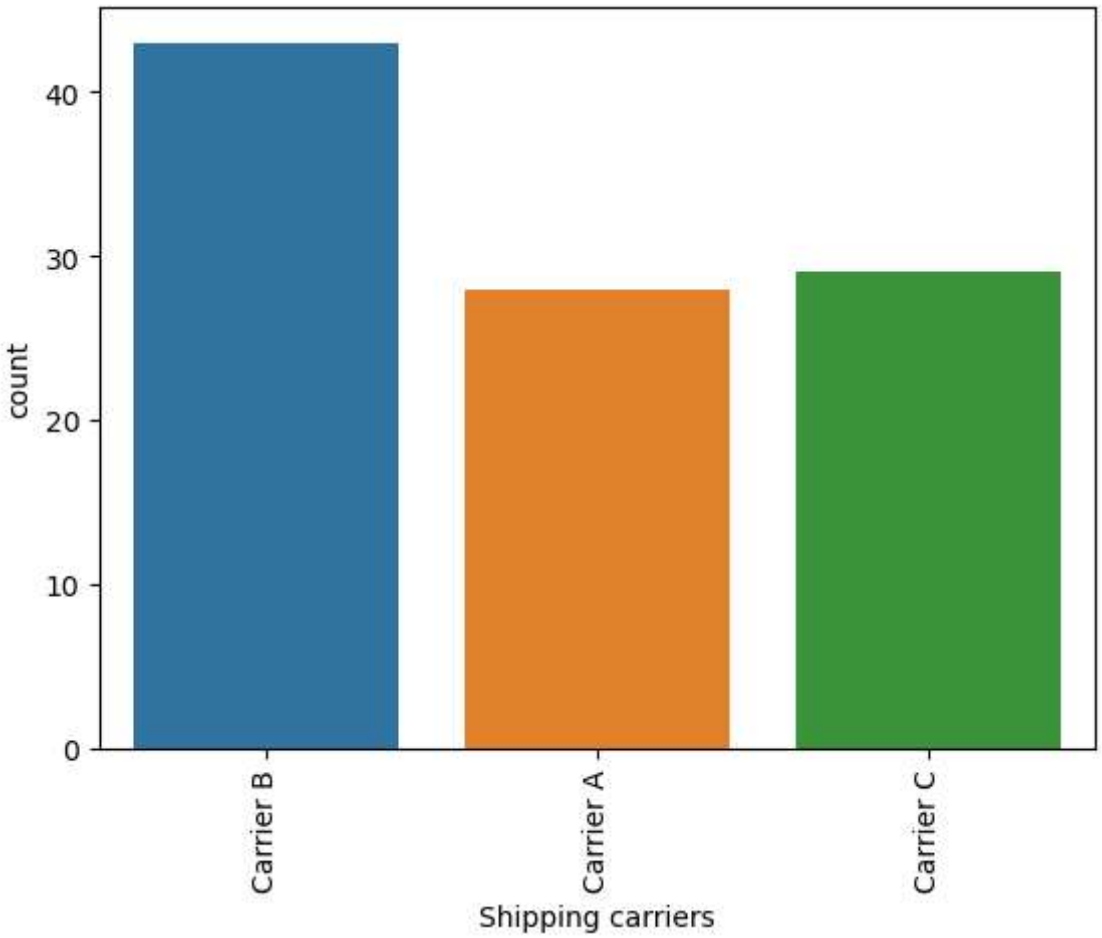
Explore Data Analysis (EDA)

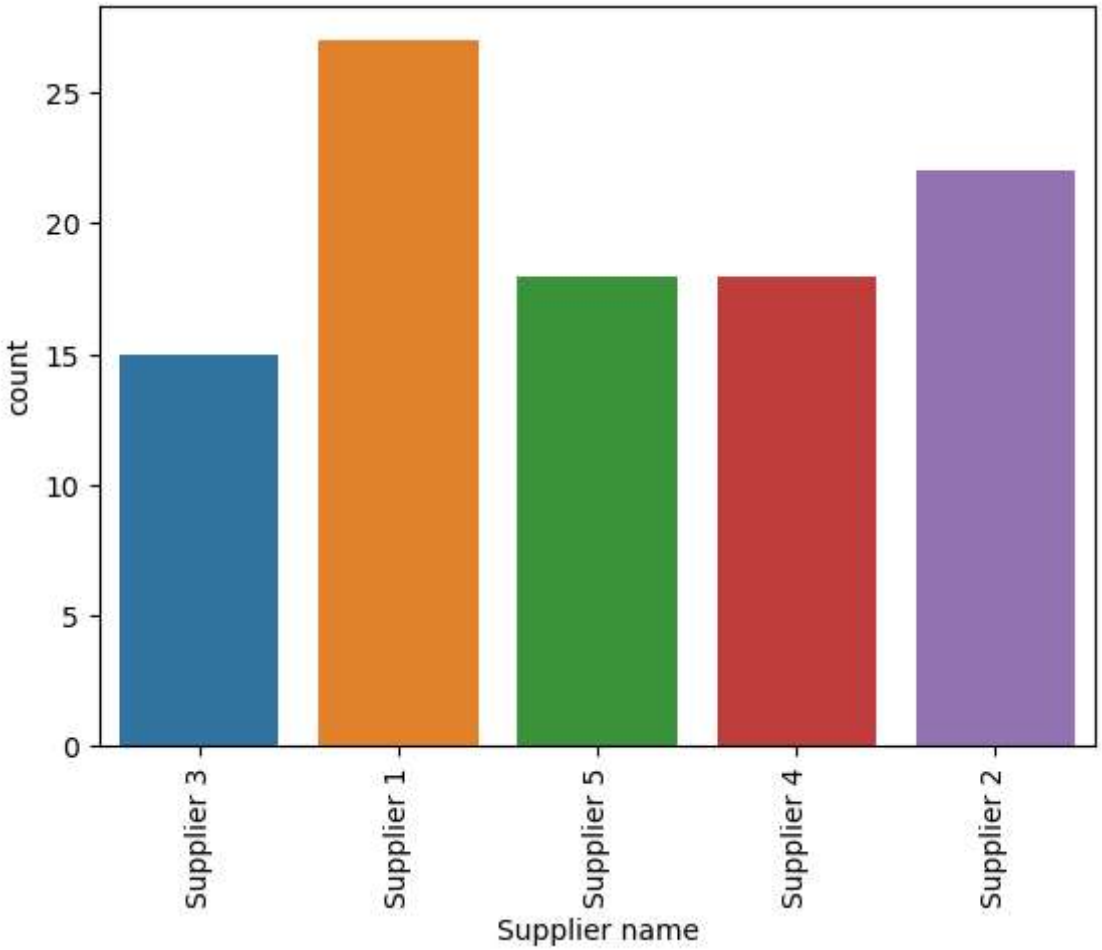
In [10]:
```
#Visualize the all the categorical columns value in the dataset using the countplots
for i in data.select_dtypes(include='object'):
    sns.countplot(data=data,x=data[i])
    plt.xticks(rotation=90)
    plt.show()
```

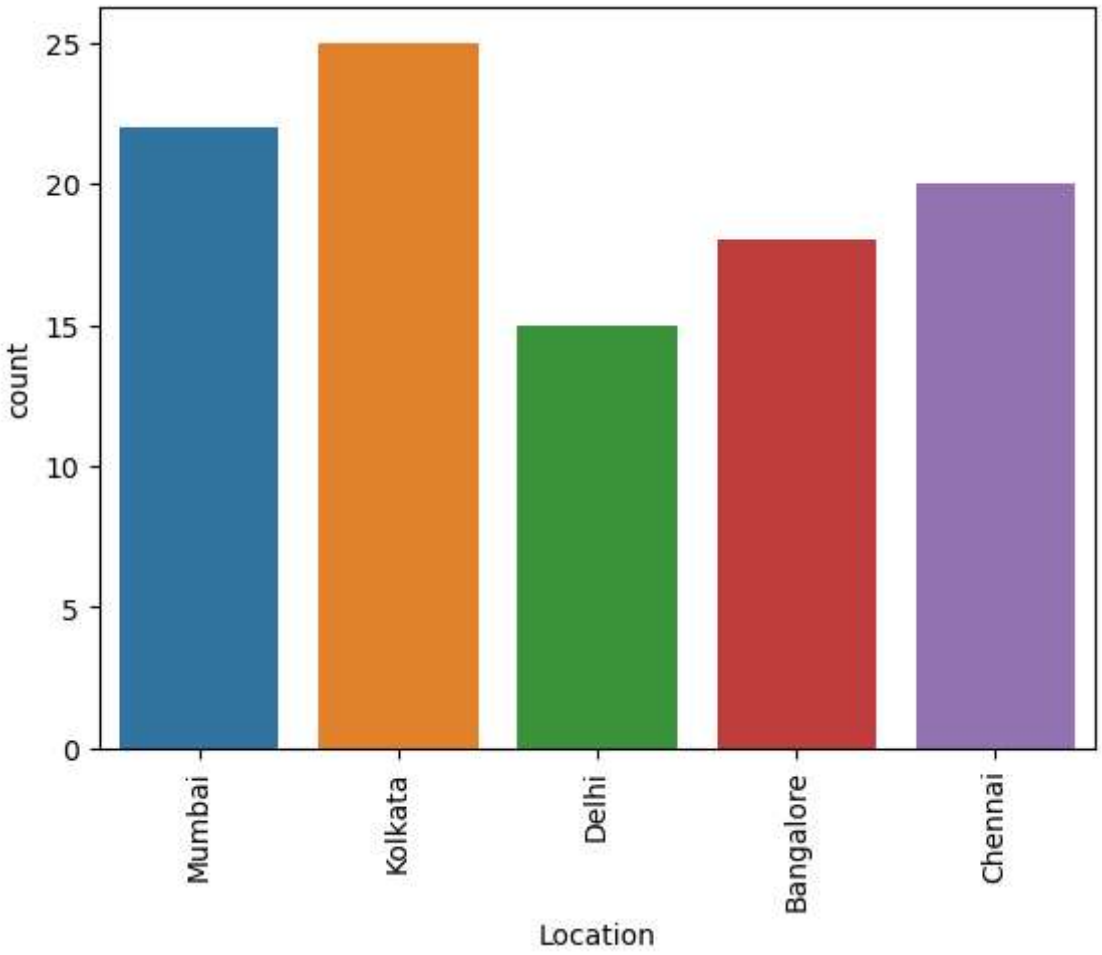

40
35
30
25
20
15
10
5
0
count
haircare
skincare
cosmetics
Product type

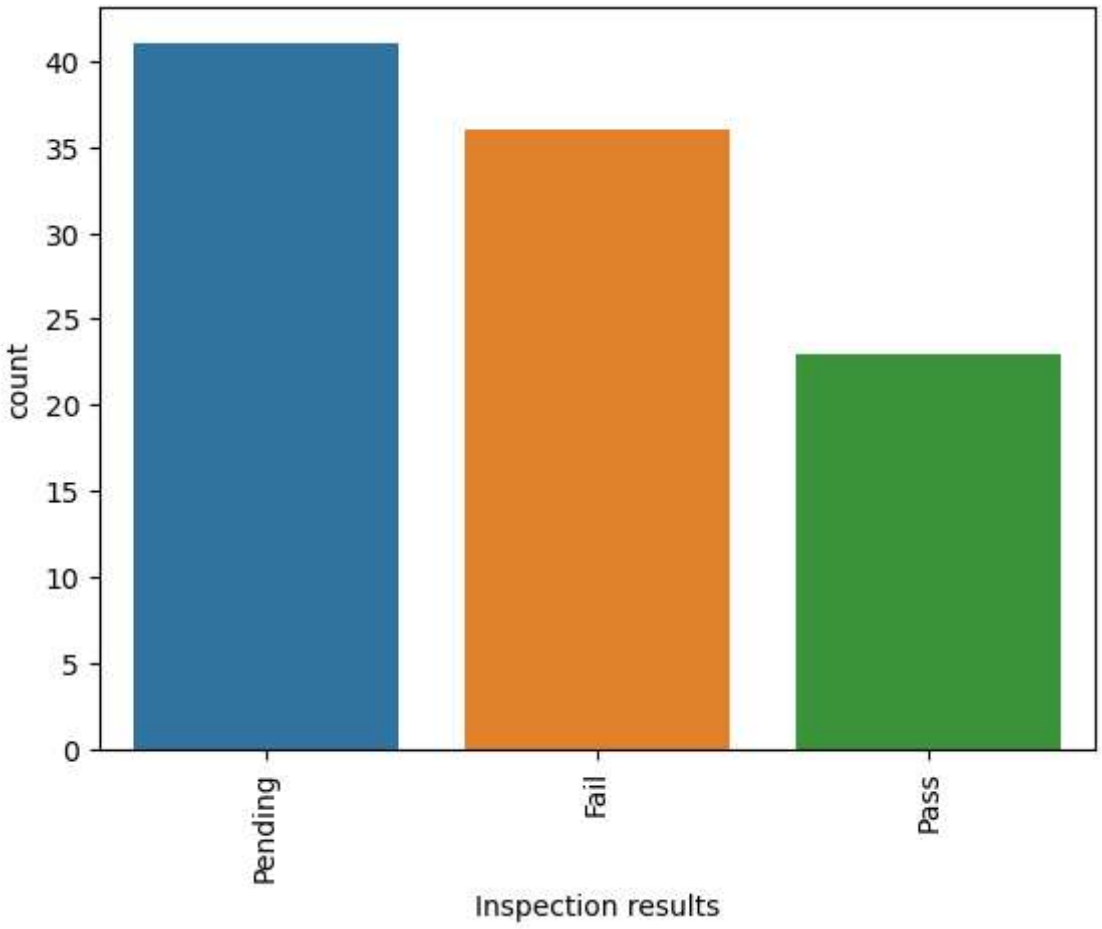

1.0
0.8
0.6
0.4
0.2
0.0
count
SKU

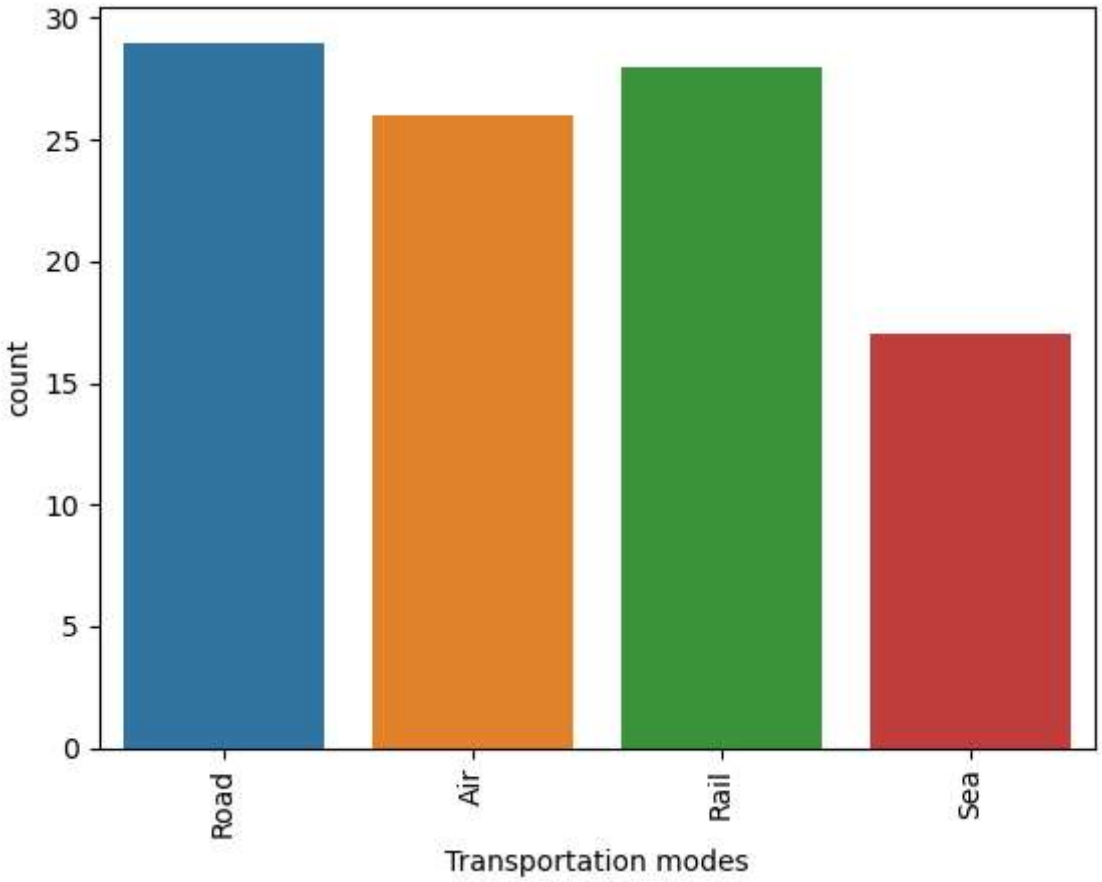

30
25
20
15
10
5
0
count
Non-binary
Female
Unknown
Male
Customer demographics

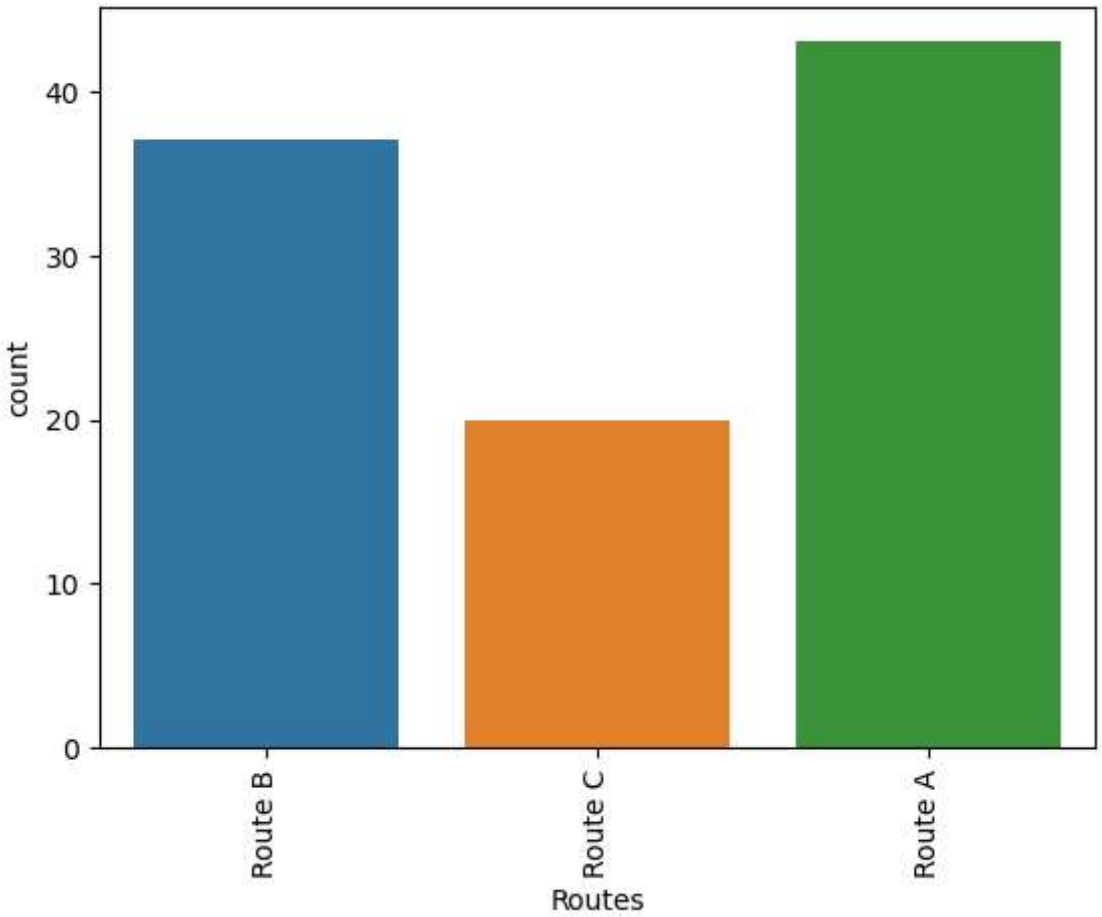

40
30
20
10
0
count
Carrier B
Carrier A
Carrier C
Shipping carriers


25
20
15
10
5
0
count
Supplier 3
Supplier 1
Supplier 5
Supplier 4
Supplier 2
Supplier name


25
20
15
10
5
0
count
Mumbai
Kolkata
Delhi
Bangalore
Chennai
Location

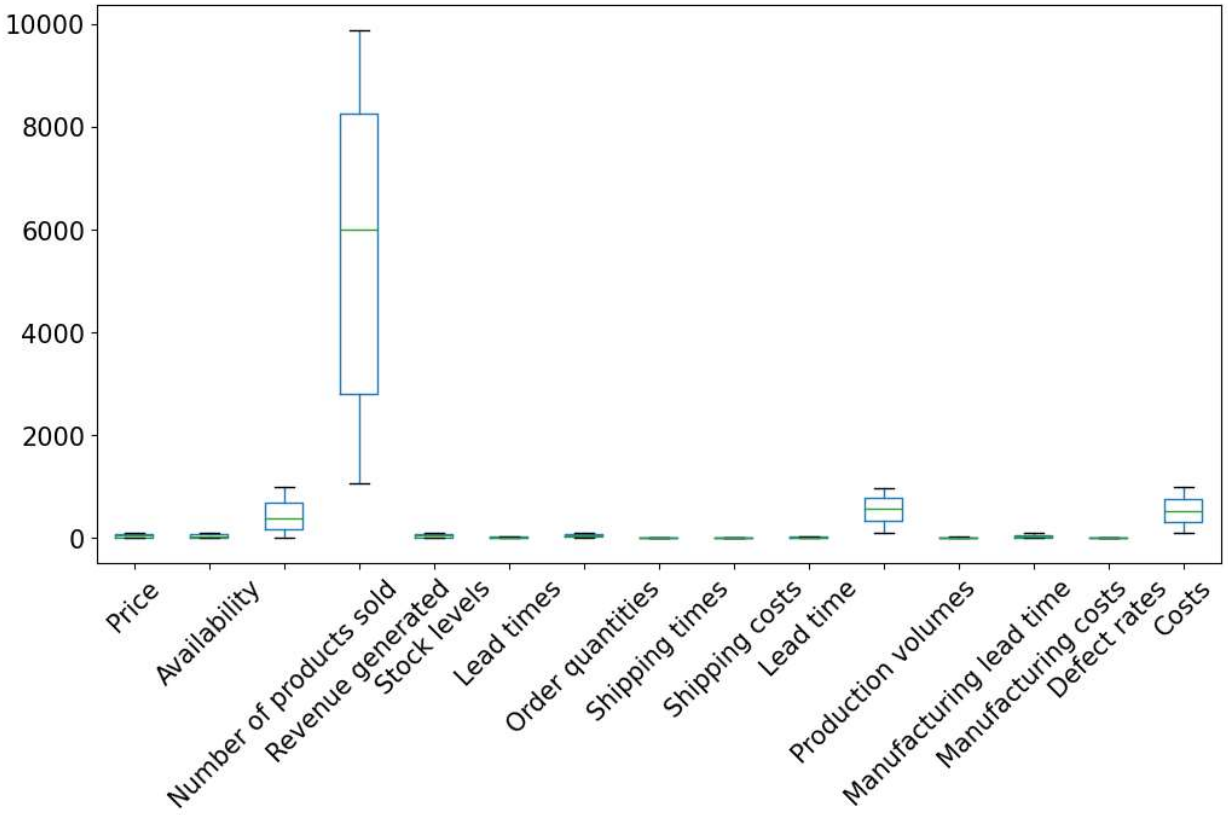

40
35
30
25
20
15
10
5
0
count
Pending
Fail
Pass
Inspection results


30
25
20
15
10
5
0
count
Road
Air
Rail
Sea
Transportation modes


40
30
20
10
0
count
Route B
Route C
Route A
Routes

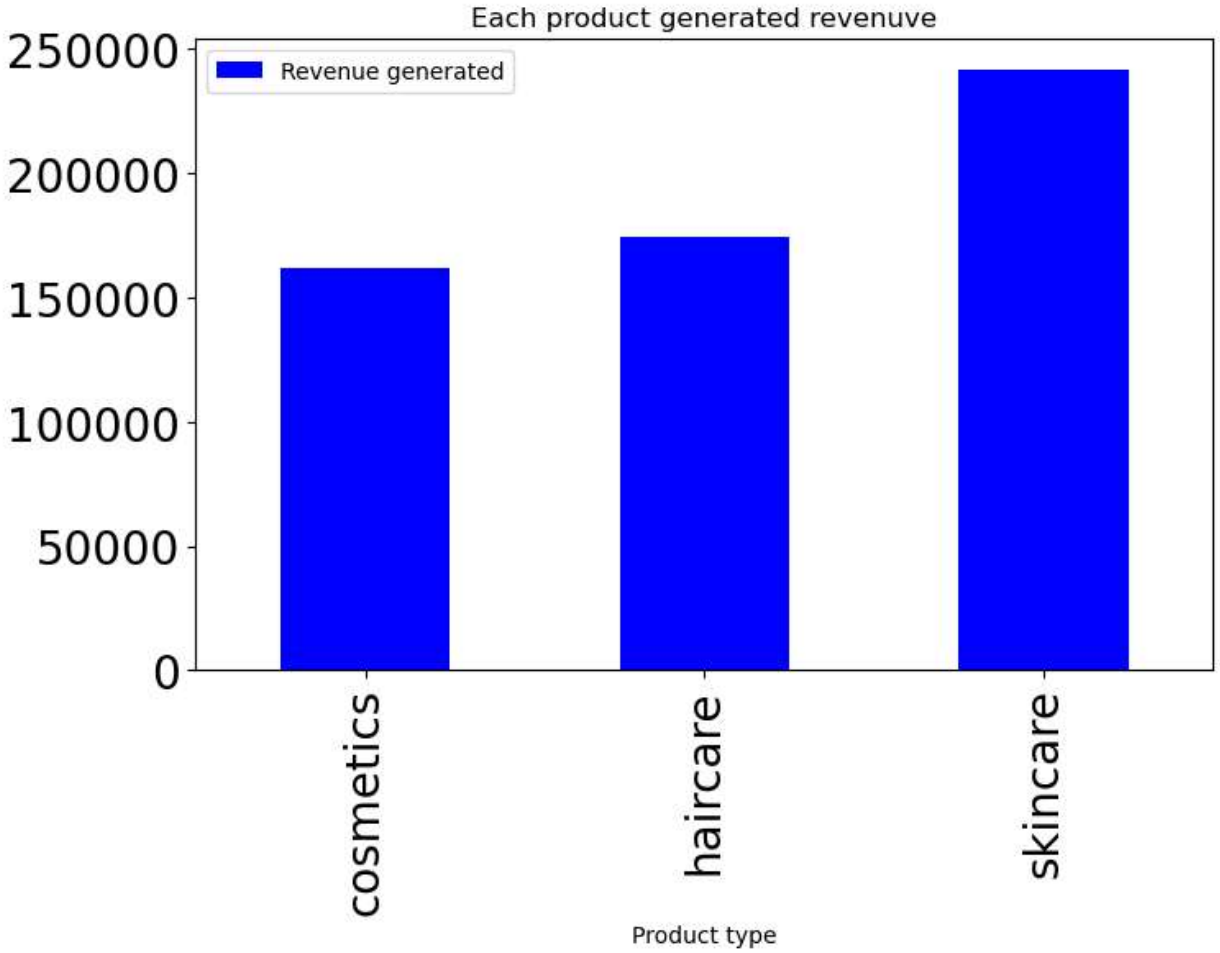
In [11]:
```
#visualize the boxplot with the data
plt.figure(figsize=(12,6))
data.boxplot(grid=False, rot=45, fontsize=15)
```

Out[11]:
```
<AxesSubplot:>
```



In [12]:
```
#we just visualize the each product type with revenue generated in the productd
data.groupby(['Product type'])[['Revenue generated']].sum()\
.plot(kind='bar',figsize=(8,5),title="Each product generated revenuve",fontsize=20,co
```

Out[12]:
```
<AxesSubplot:title={'center':'Each product generated revenuve'}, xlabel='Product typ
e'>
```



Observations:

1) The main concept of the above data to find the revenue of the products

2) Skincare item's get more revenue generated and then Hair products also gain good revenue

3) cosmetics get less revenue compare to other revenue item's

In [13]:
```
#identify the what are the location gain most revenue during the product items
data.groupby(['Product type','Location'])[['Revenue generated']].sum()\
.sort_index()\
.sort_values(by='Product type',ascending=False)\
.unstack()\
.style.background_gradient(cmap='winter_r')
```

Out[13]:

	Revenue generated				
Location	**Bangalore**	**Chennai**	**Delhi**	**Kolkata**	**Mumbai**
Product type					
cosmetics	19309.562880	31461.947457	37429.677331	24163.571855	49156.506477
haircare	51654.345696	28723.448932	14625.900767	35027.713247	44423.981964
skincare	31637.815307	58957.419359	28972.123128	77886.265903	44174.538437

Observations:

1)From the above information made create a data which product type get more revenued in location wise

2) As from the data skincare products get more revenue in kolkata and then chennai, mumbai,Bangalore and the last city was delhi

3) According to the haricare products Bangalore get more revenued then Mumbai,kolkatha,chennai and the delhi will be less revenued generated

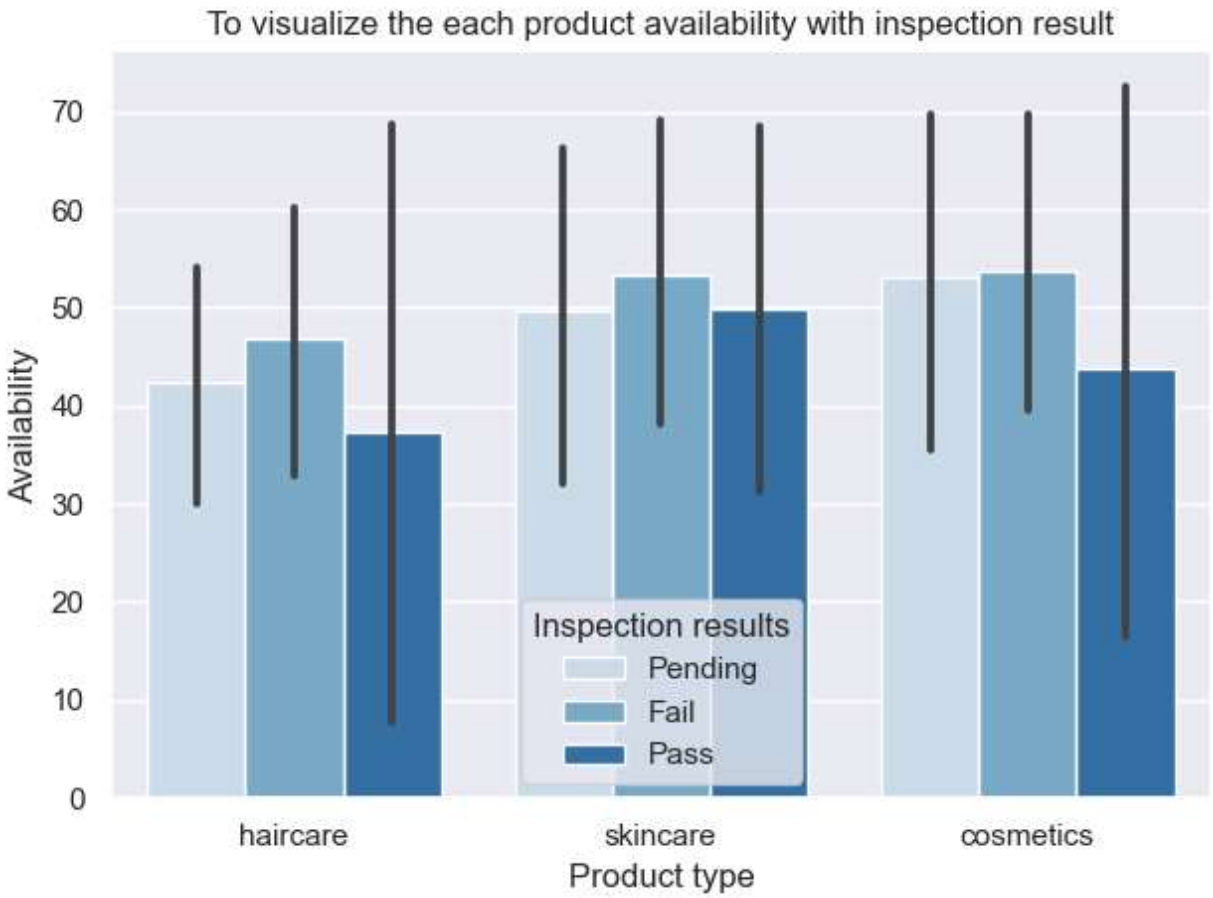
4) Coming to the cosmetics Mumbai get more profit and Delhi second place the chennai,Kotkata and the bangalore will be the less revenued generated.

In [14]:
```
#we create a data which contain each categorey wise with total price and order quantit
data.groupby(['Product type']).sum()\
.sort_index()\
.style.background_gradient(cmap='Dark2_r')
```

Out[14]:

Product type	Price	Availability	Number of products sold	Revenue generated	Stock levels	Lead times	Order quantities	Shipping times	Sl
cosmetics	1491.387498	1332	11757	161521.265999	1525	400	1343	171	157
haircare	1564.485482	1471	13611	174455.390605	1644	528	1480	191	200
skincare	1890.373155	2037	20731	241628.162133	1608	668	2099	213	196

In [15]:
```
#We visualize the each product type in with itemes availability columns hue with inspe
sns.set_theme(context='notebook',style='darkgrid')
sns.barplot(data=data,x='Product type',y='Availability',hue='Inspection results', pale
plt.title("To visualize the each product availability with inspection result")
plt.tight_layout()
plt.show()
```



Observations:

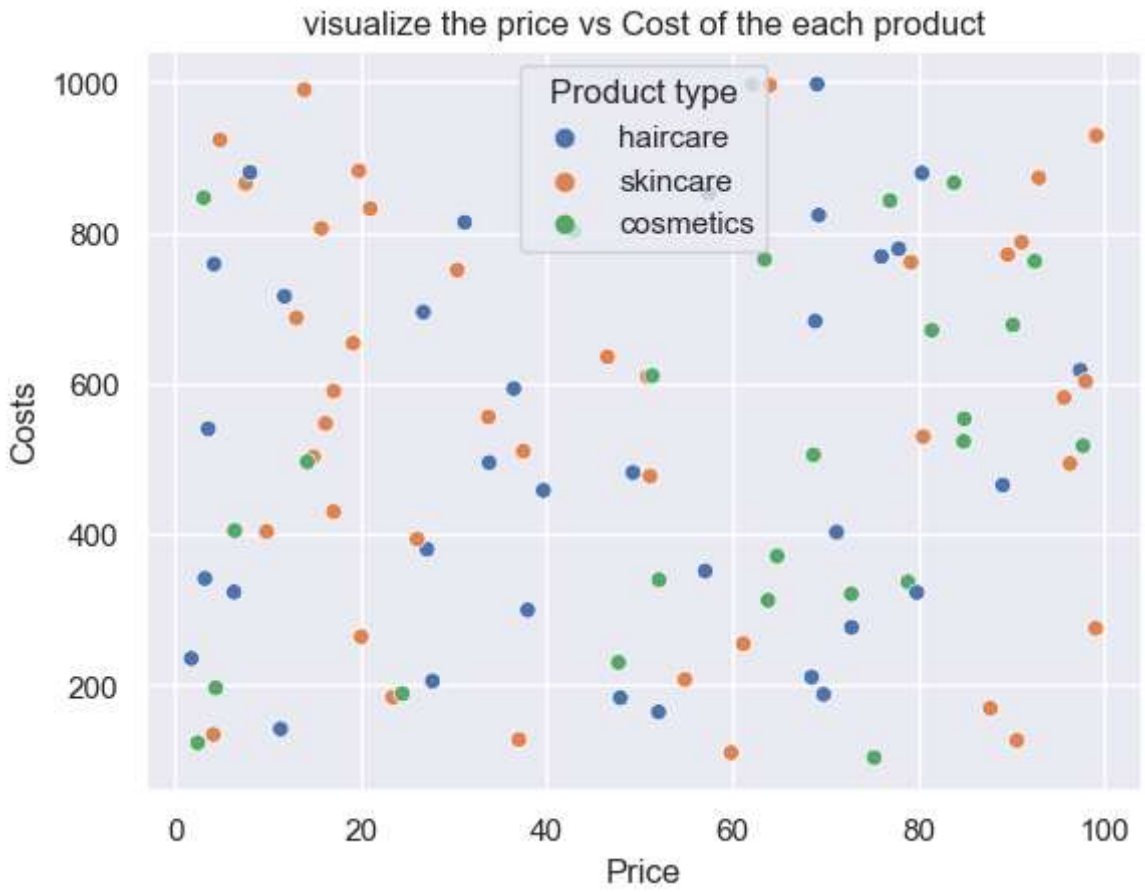
1) Form the bar chart we visualize the each product avalibality and how many products done inspection in the process we get the product is pending and either fail or pass

2) From hair care products Fail product are more compare to pending and less percentage product pass it.

3) From the skin care hear also same thing happend more product fali. both pending and pass percentage are equal.

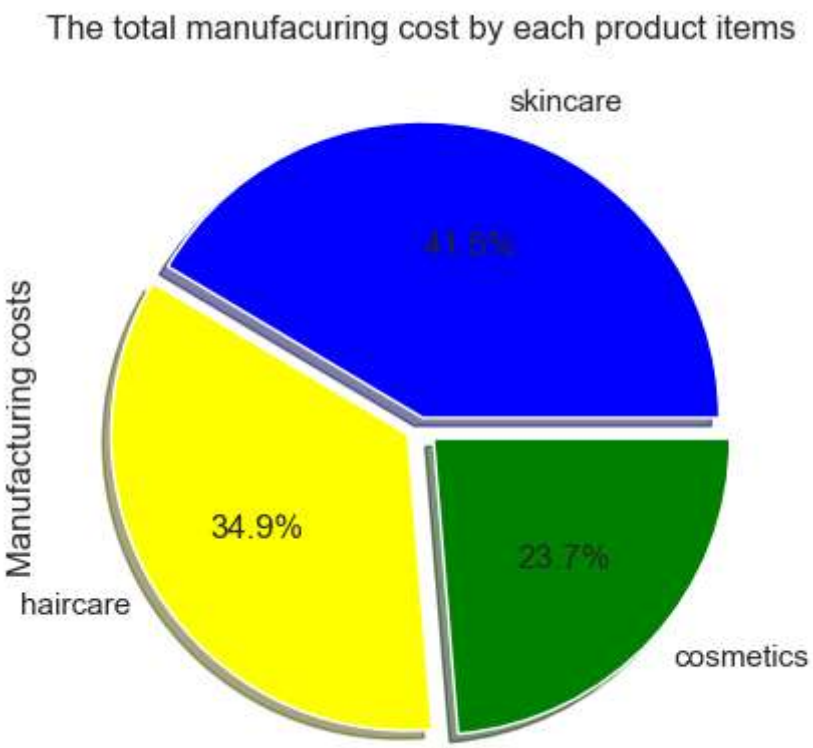
4) Let's talk about cosmetics the percentage of the fail and pending nearly equal and pass percentage less.

In [16]:
```
#Visualize the scatter plot with actually price and costs of the each product
sns.set_theme(style='darkgrid')
sns.scatterplot(data=data,x='Price',hue='Product type',y='Costs')
plt.title("visualize the price vs Cost of the each product")
plt.show()
```



In [17]:
```
#what was the manufacturing costs of the product
data.groupby(['Product type'])['Manufacturing costs'].sum()\
.sort_values(ascending=False)\
.plot(kind='pie',labels=['skincare','haircare','cosmetics'],autopct='%1.1f%%',title='
```

Out[17]:
```
<AxesSubplot:title={'center':'The total manufacuring cost by each product items'}, yl
abel='Manufacturing costs'>
```



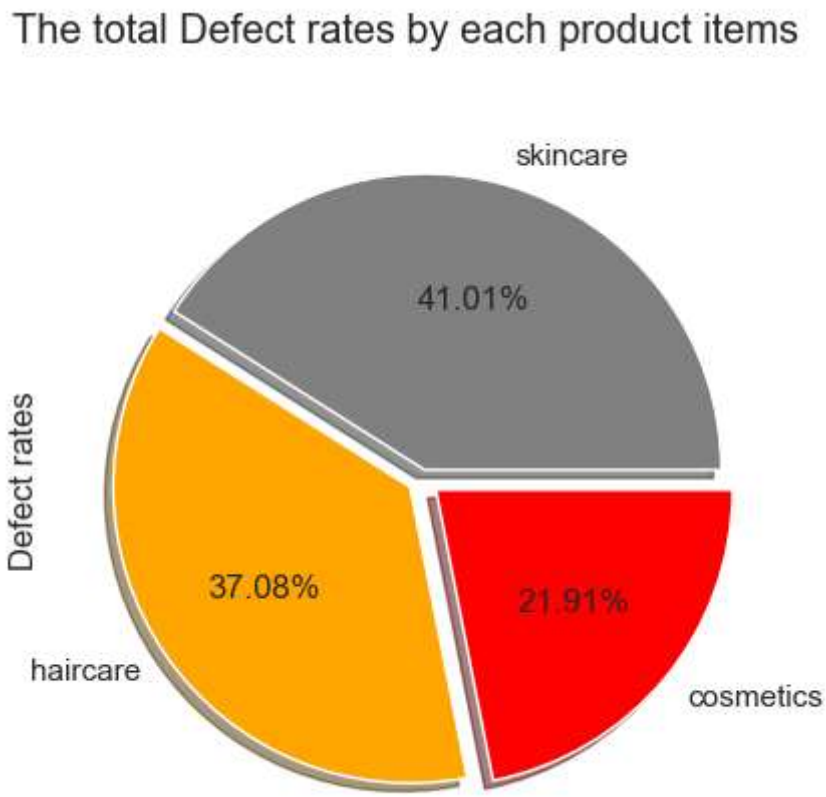
Observations:

1) From the above data skincare has more required for manufacturing costs

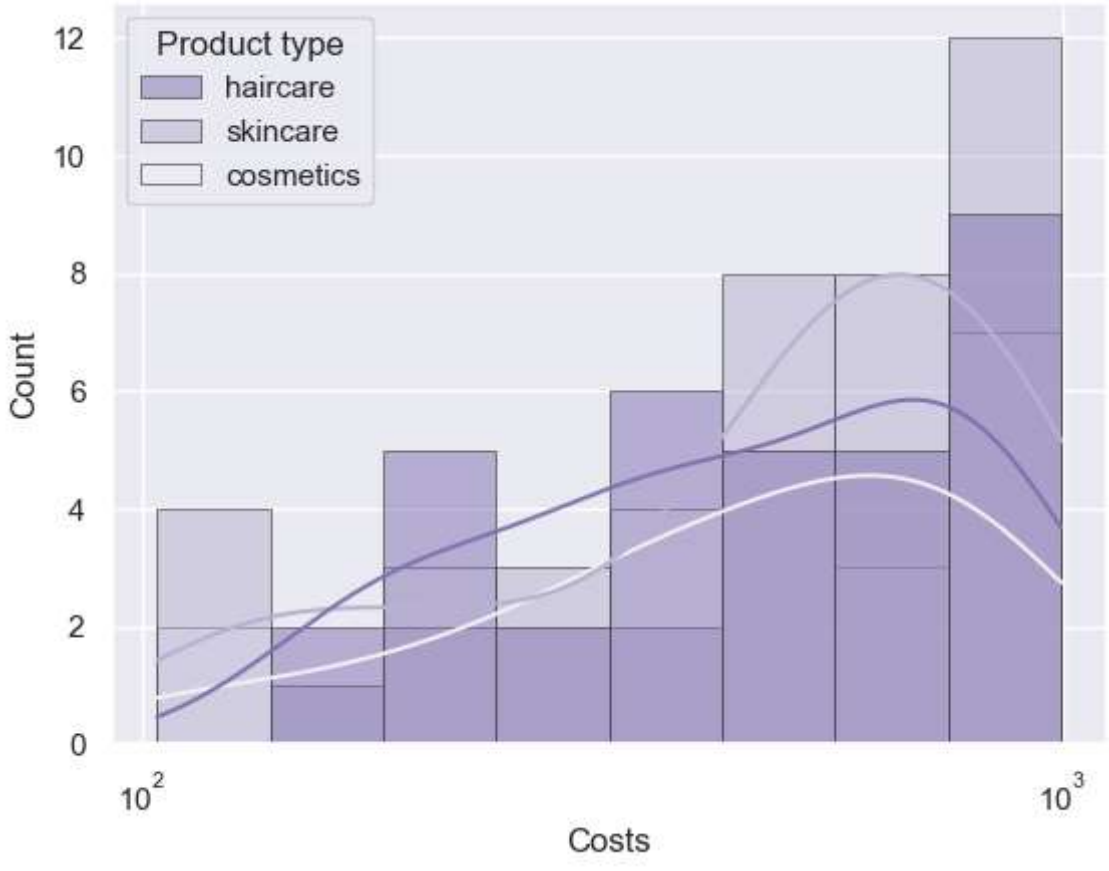
2) Then Hari care get less manufacruring cost compare to skin care and cosmetics get less manufacturing costs.

In [18]:
```
data.groupby(['Product type'])['Defect rates'].sum()\
.sort_values(ascending=False)\
.plot(kind='pie',labels=['skincare','haircare','cosmetics'],autopct='%1.2f%%',title='
```

Out[18]:
```
array([<AxesSubplot:ylabel='Defect rates'>], dtype=object)
```



In [19]:
```
sns.set_theme(style='darkgrid')
sns.histplot(data=data,x='Costs',hue='Product type',palette="light:m_r",edgecolor=".3'
plt.show()
```

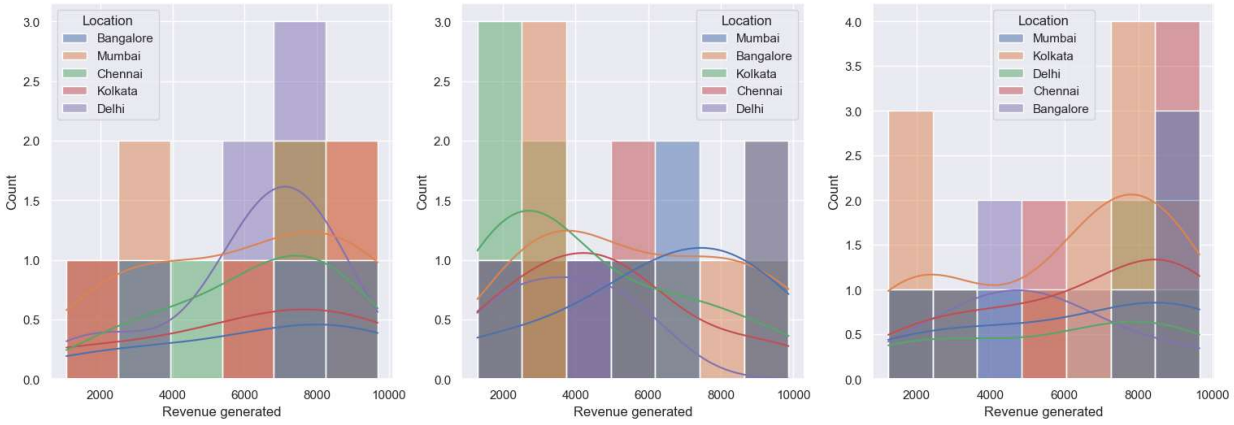


In [20]:
```
#create a piovt_table to under standing the total products sold in location wise with
pd.pivot_table(data,index='Product type',columns=['Location'],values='Number of produc
.style.background_gradient(cmap='twilight_shifted_r')
```

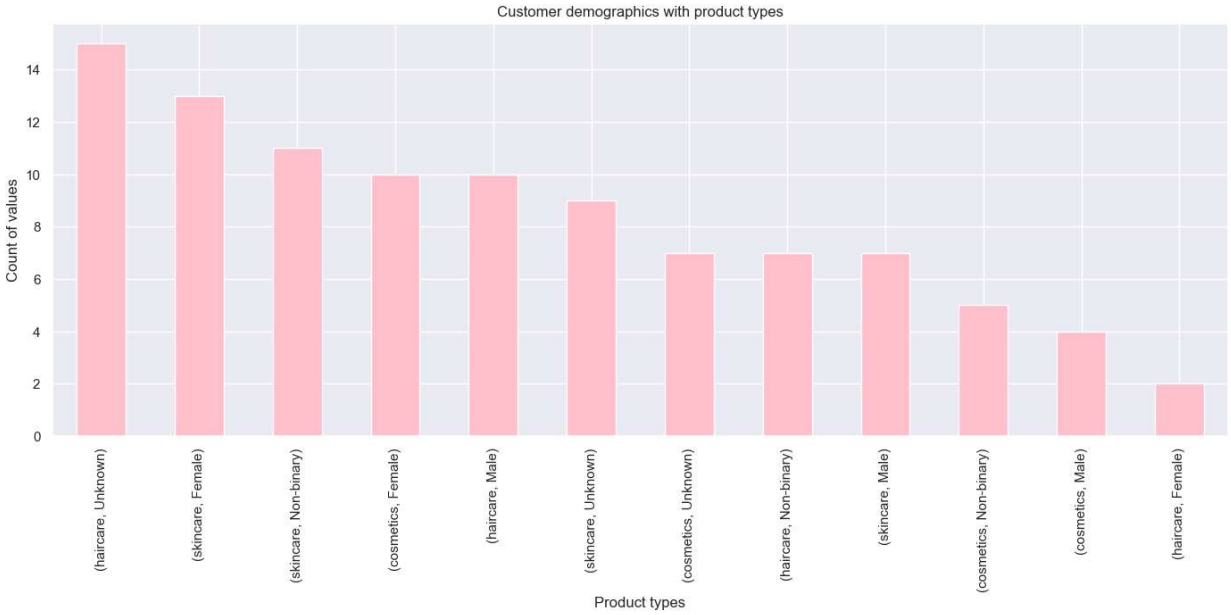
Out[20]:

Location	Bangalore	Chennai	Delhi	Kolkata	Mumbai
Product type					
cosmetics	513.666667	348.600000	667.166667	315.500000	401.000000
haircare	240.000000	386.833333	651.500000	425.875000	445.285714
skincare	286.500000	522.666667	621.200000	623.153846	443.000000

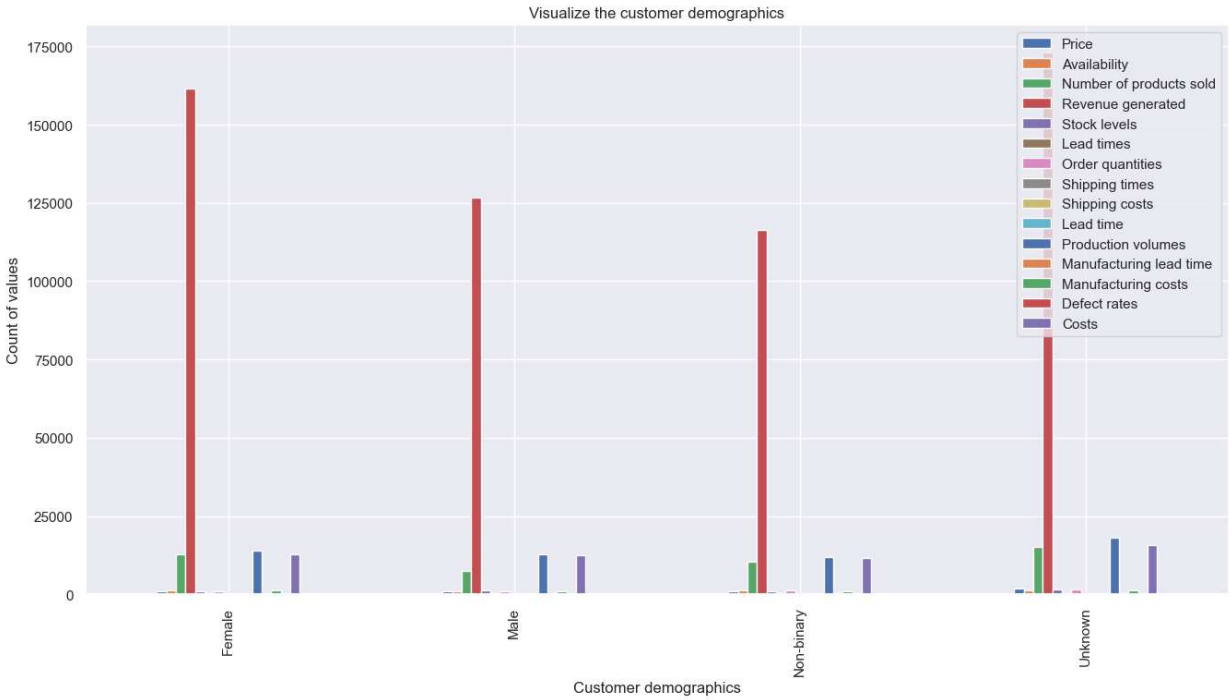
In [21]:
```
plt.subplots(1,3,figsize=(25,6))
plt.subplot(141)
ax =sns.histplot(data=data[data['Product type']=='cosmetics'],x='Revenue generated',kd
plt.subplot(142)
ax =sns.histplot(data=data[data['Product type']=='haircare'],x='Revenue generated',kde
plt.subplot(143)
ax =sns.histplot(data=data[data['Product type']=='skincare'],x='Revenue generated',kde
plt.show()
```



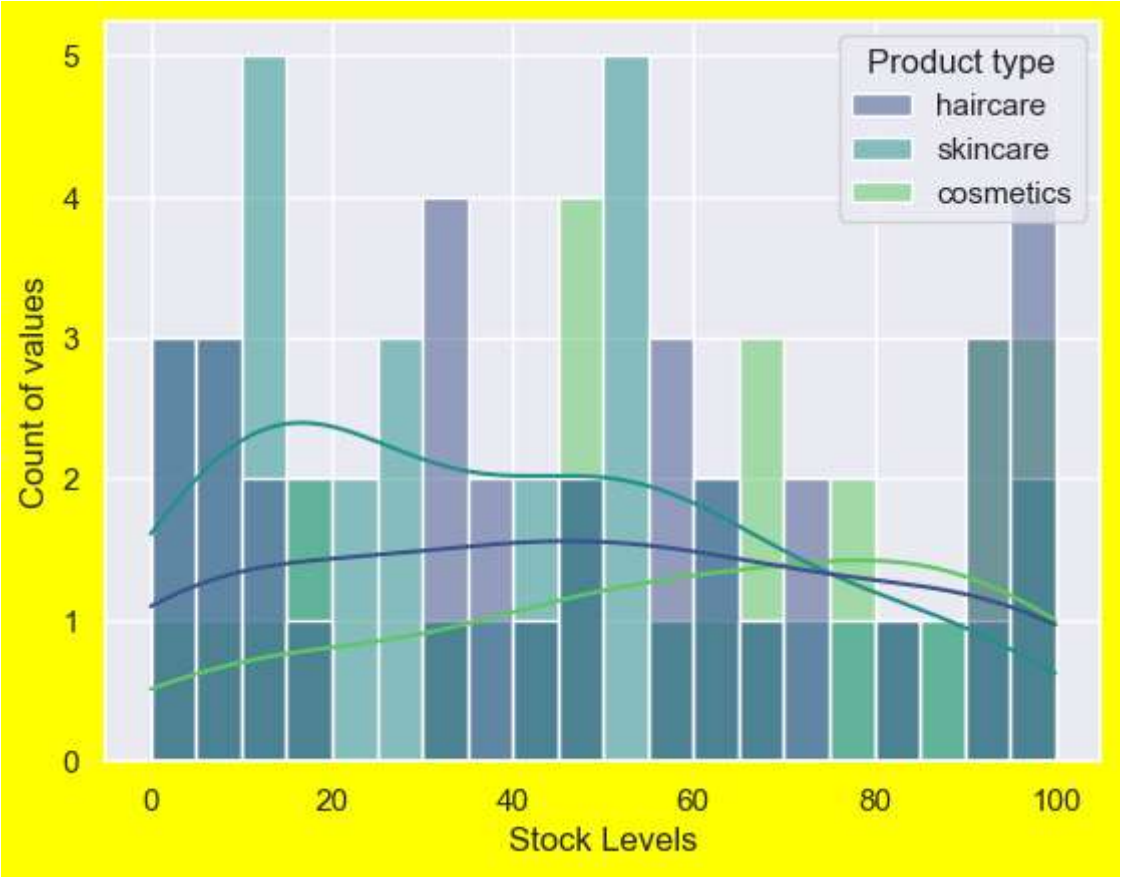
In [22]:
```
#To create a barchart for customers demographics
data.groupby(['Product type'])['Customer demographics'].value_counts()\
.sort_index()\
.sort_values(ascending=False)\
.plot(kind='bar',title="Customer demographics with product types",figsize=(17,6),color
plt.xlabel("Product types")
plt.ylabel("Count of values")
plt.show()
```



In [23]:
```
data.groupby(['Customer demographics']).sum()\
.sort_index()\
.plot(kind='bar',figsize=(16,8))
plt.title("Visualize the customer demographics")
plt.xlabel("Customer demographics",fontweight=20)
plt.ylabel("Count of values")
plt.show()
```



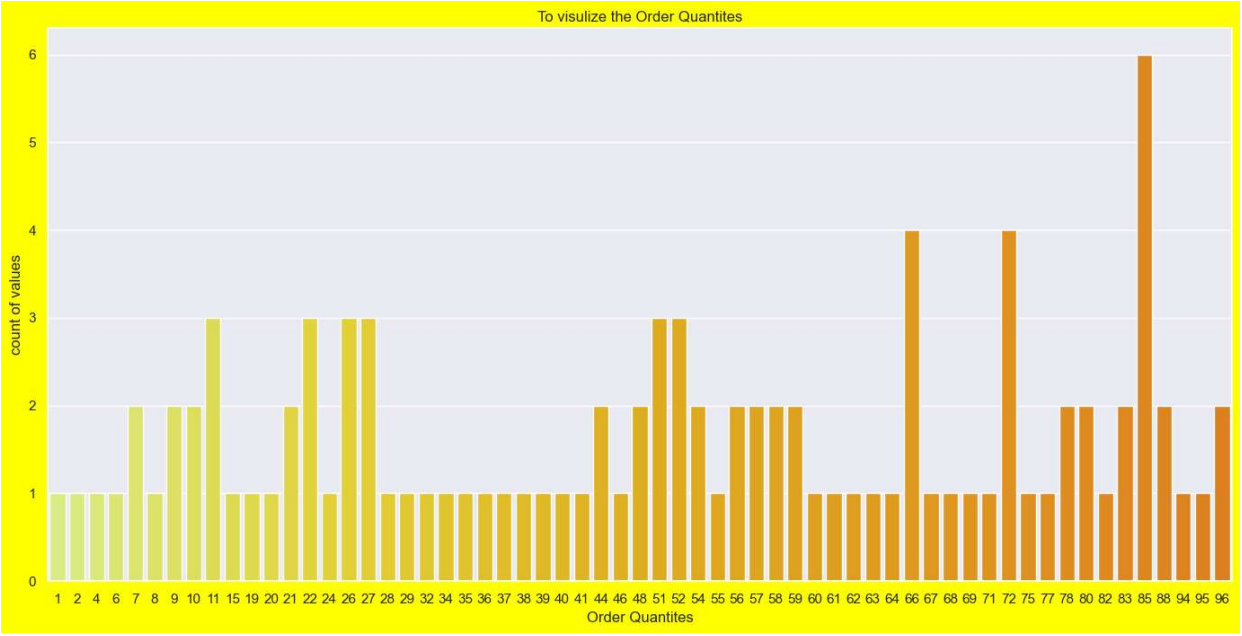
In [24]:
```
#distribution of stock level
sns.set_theme(style='darkgrid')
plt.rcParams['figure.facecolor']='yellow'
sns.histplot(data=data,x='Stock levels',hue='Product type',bins=20,palette='viridis',r
plt.xlabel("Stock Levels")
plt.ylabel("Count of values")
plt.show()
```



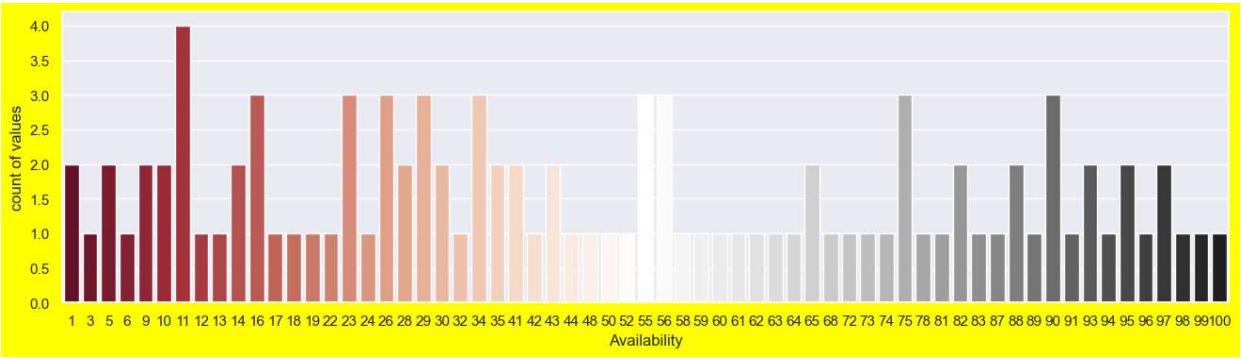
Observations:

1) From the above chart hightest stock lies on nearly 5 and 60

In [25]:
```
plt.figure(figsize=(17,8))
sns.countplot(data=data,x='Order quantities',palette='Wistia')
plt.title("To visulize the Order Quantites")
plt.xlabel("Order Quantites")
plt.ylabel("count of values")
plt.show()
```



In [26]:
```
plt.figure(figsize=(16,4))
sns.countplot(data=data,x='Availability',palette='RdGy')
plt.xlabel("Availability")
plt.ylabel("count of values")
plt.show()
```



Observations:

1)From above chart's the highest quantity is 85 later 62 and 72

2) In the second chart 11 was repeted 4 times which mean 11 products was avilble in 4 time and 55 and 56 availbel 3 times

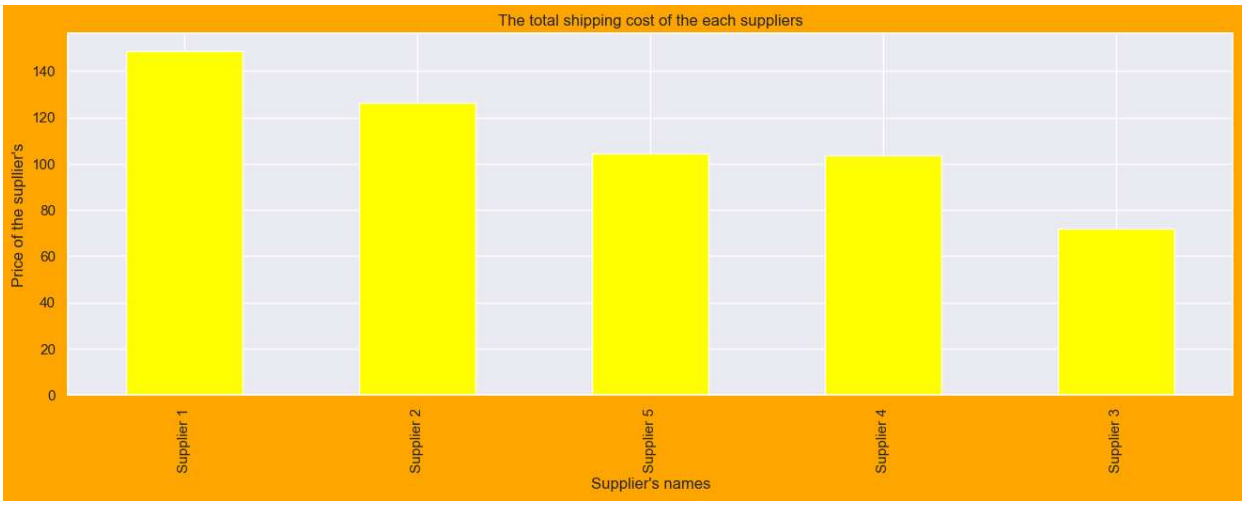
In [27]:
```
data.columns
```

Out[27]:
```
Index(['Product type', 'SKU', 'Price', 'Availability',
       'Number of products sold', 'Revenue generated', 'Customer demographics',
       'Stock levels', 'Lead times', 'Order quantities', 'Shipping times',
       'Shipping carriers', 'Shipping costs', 'Supplier name', 'Location',
       'Lead time', 'Production volumes', 'Manufacturing lead time',
       'Manufacturing costs', 'Inspection results', 'Defect rates',
       'Transportation modes', 'Routes', 'Costs'],
      dtype='object')
```

In [28]:
```
plt.rcParams['figure.facecolor']='orange'
data['Shipping times'].value_counts()\
```

```
.sort_values(ascending=False)\
.plot(kind='bar',title="Understanding the shipping times",figsize=(16,5),color='black'
plt.xlabel("Time")
plt.ylabel("Count of values")
plt.show()
```

In [29]:
```
data.groupby(['Supplier name'])['Shipping costs'].sum()\
.sort_values(ascending=False)\
.plot(kind='bar',title="The total shipping cost of the each suppliers",figsize=(16,5),
plt.xlabel("Supplier's names")
plt.ylabel("Price of the supllier's")
plt.show()
```



Observations:

1)From the first chart most of the suppliers send their product with 8 clock and 7 clock and less number of people send their product 2 clock

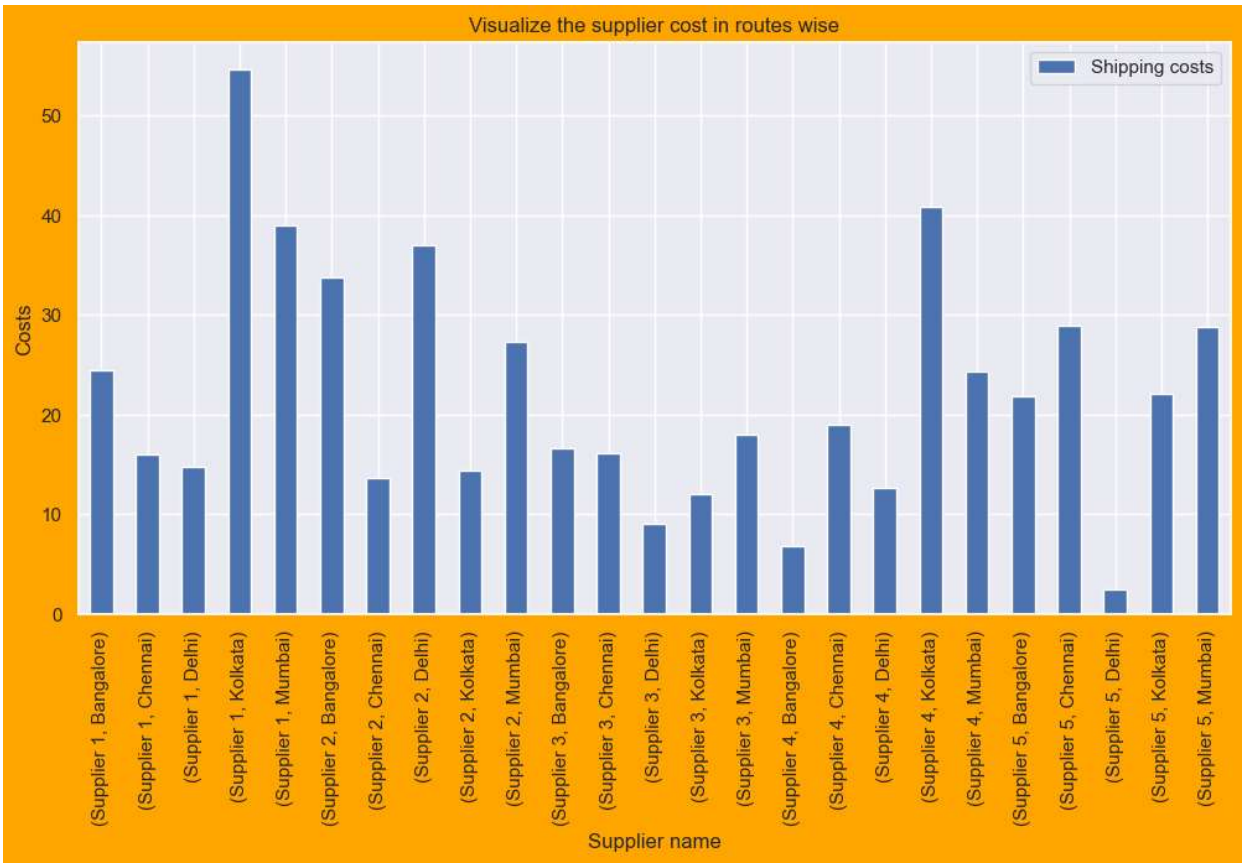
2)The second chart the supplier1 spend arround 148 price for shipping and supliers'3 less amount spend for shipping cost

In [30]:
```
#Shipping cost of the each supplies with location wise
data.groupby(['Supplier name','Location'])[['Shipping costs']].sum()\
.sort_index()\
.unstack()\
.style.background_gradient(cmap='Reds')
```

Out[30]:

	Shipping costs				
Location	**Bangalore**	**Chennai**	**Delhi**	**Kolkata**	**Mumbai**
Supplier name					
Supplier 1	24.478536	15.957139	14.777578	54.658678	38.960219
Supplier 2	33.725821	13.692824	37.051131	14.425672	27.366470
Supplier 3	16.628690	16.159605	9.004716	12.012523	18.026025
Supplier 4	6.792438	19.035269	12.709169	40.814152	24.321286
Supplier 5	21.846532	28.936740	2.505621	22.124321	28.803753

In [31]:
```
supplier=data.groupby(['Supplier name','Location'])[['Shipping costs']].sum()
supplier.plot(kind='bar',title="Visualize the supplier cost in routes wise",figsize=(1
plt.xlabel("Supplier name")
plt.ylabel("Costs")
plt.show()
```



Observations:

1)Another dataframe created for identify with which location each suplies spend most shipping price

2)Supplier 1 spend mores shipping cost in kolkata and mumbai.

3)Supplier 2 spend mores shipping cost in Delhi and Benguluru.

4)Supplier 3 spend mores shipping cost in Benguluru and mumbai.

5)Supplier 4 spend mores shipping cost in kolkata and mumbai.

6)Supplier 5 spend mores shipping cost in chennai and mumbai.

In [32]:
```
#Visualize the overal cost price of shipping carries in different location wise
'''
We use a groupby function with shipping and location and cost columns
\ used for filter and unstack function converted to rows
and finally visualize with background color
'''
data.groupby(['Shipping carriers','Location'])['Costs'].sum()\
.unstack()\
.style.background_gradient(cmap='nipy_spectral')
```

Out[32]:

Location	Bangalore	Chennai	Delhi	Kolkata	Mumbai
Shipping carriers					
Carrier A	4010.034588	3822.282532	2194.619202	2163.042576	1737.092806
Carrier B	5156.304957	4243.420799	3756.959961	5860.229558	3708.528990
Carrier C	1394.381893	4369.309861	2271.988994	4258.472904	3977.908594

Observations:

1)In the above chart carrier A received more cost in Bangalore and then Chennai get received costs.

2)In the above chart carrier B received more cost in Kolkata and then Bangalore get received costs.

3)In the above chart carrier C received more cost in Chennai and then Delhi get received costs.

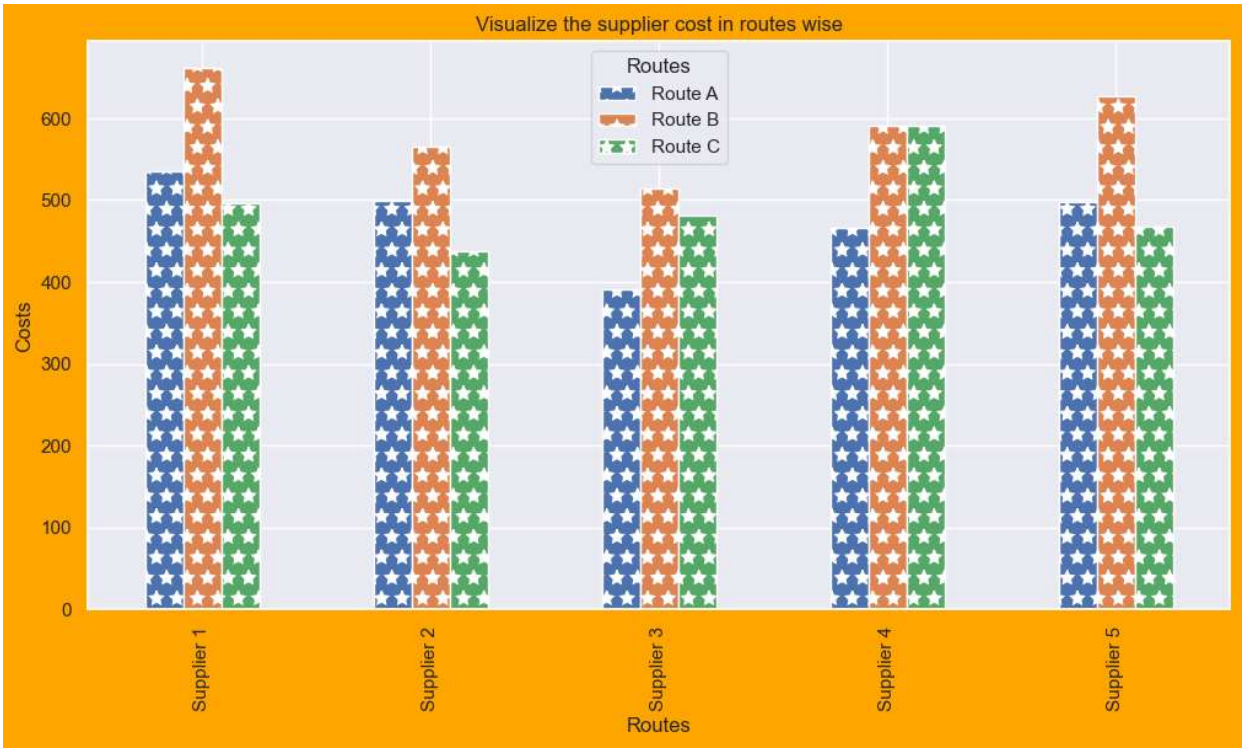
In [33]:
```
'''
We create a pivot tabel for each supliers spend most cost with routes
wise, we take index as supplier name columns routes and
values are costs once we done with visualize with background color

'''
pd.pivot_table(data,index='Supplier name',columns=['Routes'],values='Costs')\
.style.background_gradient(cmap='YlOrBr')
```

Out[33]:

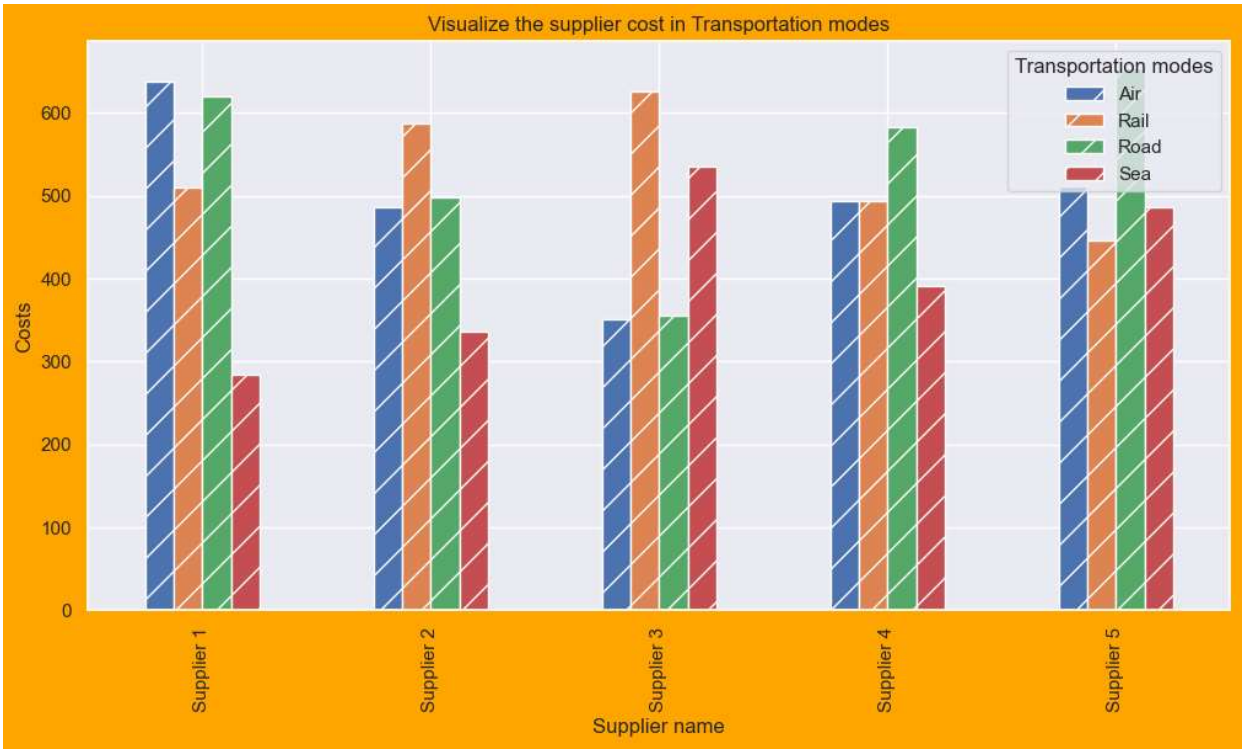
Routes	Route A	Route B	Route C
Supplier name			
Supplier 1	535.451837	661.026634	495.759134
Supplier 2	499.177114	565.627125	438.211349
Supplier 3	391.100859	514.451725	480.441713
Supplier 4	466.373041	590.960046	591.254232
Supplier 5	497.997427	627.270012	467.604074

In [34]:
```
supp=pd.pivot_table(data,index='Supplier name',columns=['Routes'],values='Costs')
supp.plot(kind='bar',title="Visualize the supplier cost in routes wise",figsize=(12,6)
plt.xlabel("Routes")
plt.ylabel("Costs")
plt.show()
```



In [35]:
```
'''Same as above pivot table but this time we find the cost of each supplies with trar
which mean each suplier which transpotaion spend more cost.
'''

trans=pd.pivot_table(data,index='Supplier name',columns=['Transportation modes'],value
trans.plot(kind='bar',title="Visualize the supplier cost in Transportation modes",figs
plt.xlabel("Supplier name")
plt.ylabel("Costs")
plt.show()
```



Observation:

1)Supplier 1 spend more cost in the Air and Road.and less transporation in Sea and Rails.

2)Supplier 2 spend more cost in the Air and Road.and less transporation in Sea and Rails.

3)Supplier 3 spend more cost in the Rail and Sea.and less transporation in Road and Air.

4)Supplier 4 spend more cost in the Road and Air,Rails.and less transporation in sea.

5)Supplier 5 spend more cost in the AIr and Road.and less transporation in Sea and Rails.

In [ ]: