Supply Chain Analysis with Python

Problem Statement Supply chain analytics is a valuable part of data-driven decision-making in various industries such as manufacturing, retail, healthcare, and logistics. It is the process of collecting, analyzing and interpreting data related to the movement of products and services from suppliers to customers.

Import Library

In [3]:

```
import pandas as pd
```

In [4]:

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

Uploading Csv fle

In [5]:

```
df = pd.read_csv(r"C:\Users\Syed Arif\Desktop\supply_chain_data.csv")
```

Data Preprocessing

.head()

head is used show to the By default = 5 rows in the dataset

In [7]:

```
df.head()
```

Out[7]:

Customer demographics	Stock levels	Lead times	Order quantities	...	Location	Lead time	Production volumes	Manufacturing lead time	Manufactu c
Non-binary	58	7	96	...	Mumbai	29	215	29	46.279
Female	53	30	37	...	Mumbai	23	517	30	33.616
Unknown	1	10	88	...	Mumbai	12	971	27	30.688
Non-binary	23	13	59	...	Kolkata	24	937	18	35.624
Non-binary	5	3	56	...	Delhi	5	414	3	92.065

.tail()

tail is used to show rows by Descending order

In [8]:

```
df.tail()
```

Out[8]:

Customer demographics	Stock levels	Lead times	Order quantities	...	Location	Lead time	Production volumes	Manufacturing lead time	Manufactu c
Unknown	15	14	26	...	Mumbai	18	450	26	58.890
Non-binary	67	2	32	...	Mumbai	28	648	28	17.803
Male	46	19	4	...	Mumbai	10	535	13	65.765
Female	53	1	27	...	Chennai	28	581	9	5.604
Unknown	55	8	59	...	Chennai	29	921	2	38.072

.shape

It show the total no of rows & Column in the dataset

In [9]:
```
df.shape
```

Out[9]:
```
(100, 24)
```

.Columns

It show the no of each Column

In [10]:
```
df.columns
```

Out[10]:
```
Index(['Product type', 'SKU', 'Price', 'Availability',
       'Number of products sold', 'Revenue generated', 'Customer demographic
s',
       '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')
```

.dtypes

This Attribute show the data type of each column

In [11]:

```
df.dtypes
```

Out[11]:

```
Product type                object
SKU                         object
Price                      float64
Availability                 int64
Number of products sold      int64
Revenue generated          float64
Customer demographics       object
Stock levels                 int64
Lead times                   int64
Order quantities             int64
Shipping times               int64
Shipping carriers           object
Shipping costs             float64
Supplier name               object
Location                    object
Lead time                    int64
Production volumes           int64
Manufacturing lead time      int64
Manufacturing costs        float64
Inspection results          object
Defect rates               float64
Transportation modes        object
Routes                      object
Costs                      float64
dtype: object
```

.unique()

In a column, It show the unique value of specific column.

In [12]:

```
df["Location"].unique()
```

Out[12]:

```
array(['Mumbai', 'Kolkata', 'Delhi', 'Bangalore', 'Chennai'], dtype=object)
```

.nuique()

It will show the total no of unque value from whole data frame

In [13]:
```
df.nunique()
```

Out[13]:
```
Product type                3
SKU                       100
Price                     100
Availability               63
Number of products sold    96
Revenue generated         100
Customer demographics       4
Stock levels               65
Lead times                 29
Order quantities           61
Shipping times             10
Shipping carriers           3
Shipping costs            100
Supplier name               5
Location                    5
Lead time                  29
Production volumes         96
Manufacturing lead time    30
Manufacturing costs       100
Inspection results          3
Defect rates              100
Transportation modes        4
Routes                      3
Costs                     100
dtype: int64
```

.describe()

It show the Count, mean , median etc

In [14]:
```
df.describe()
```

Out[14]:

	Price	Availability	Number of products sold	Revenue generated	Stock levels	Lead times	Order quantities	S
count	100.000000	100.000000	100.000000	100.000000	100.000000	100.000000	100.000000	100
mean	49.462461	48.400000	460.990000	5776.048187	47.770000	15.960000	49.220000	5
std	31.168193	30.743317	303.780074	2732.841744	31.369372	8.785801	26.784429	2
min	1.699976	1.000000	8.000000	1061.618523	0.000000	1.000000	1.000000	1
25%	19.597823	22.750000	184.250000	2812.847151	16.750000	8.000000	26.000000	3
50%	51.239831	43.500000	392.500000	6006.352023	47.500000	17.000000	52.000000	6
75%	77.198228	75.000000	704.250000	8253.976921	73.000000	24.000000	71.250000	8
max	99.171329	100.000000	996.000000	9866.465458	100.000000	30.000000	96.000000	10

.value_counts

It Shows all the unique values with their count

In [15]:

```
df["Location"].value_counts()
```

Out[15]:

```
Kolkata      25
Mumbai       22
Chennai      20
Bangalore    18
Delhi        15
Name: Location, dtype: int64
```

.isnull()

It shows the how many null values

In [16]:

```
df.isnull()
```

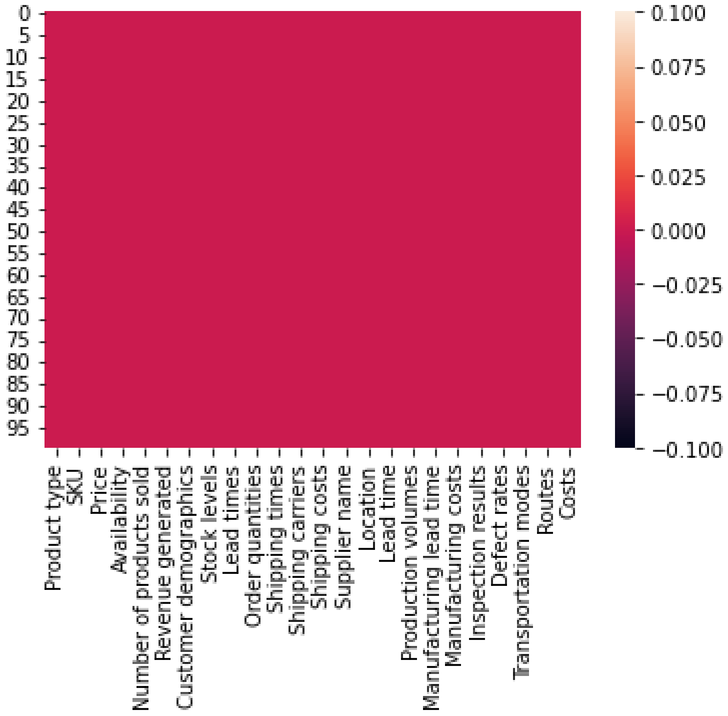
Out[16]:

	Product type	SKU	Price	Availability	Number of products sold	Revenue generated	Customer demographics	Stock levels	Lead times	Orc quantiti
0	False	False	False	False	False	False	False	False	False	Fa
1	False	False	False	False	False	False	False	False	False	Fa
2	False	False	False	False	False	False	False	False	False	Fa
3	False	False	False	False	False	False	False	False	False	Fa
4	False	False	False	False	False	False	False	False	False	Fa
...	...	...	...	...	...	...	...	...	...	
95	False	False	False	False	False	False	False	False	False	Fa
96	False	False	False	False	False	False	False	False	False	Fa
97	False	False	False	False	False	False	False	False	False	Fa
98	False	False	False	False	False	False	False	False	False	Fa
99	False	False	False	False	False	False	False	False	False	Fa

100 rows × 24 columns

```
In [17]: sns.heatmap(df.isnull())
```

```
Out[17]: <AxesSubplot:>
```



Sales By Product Type

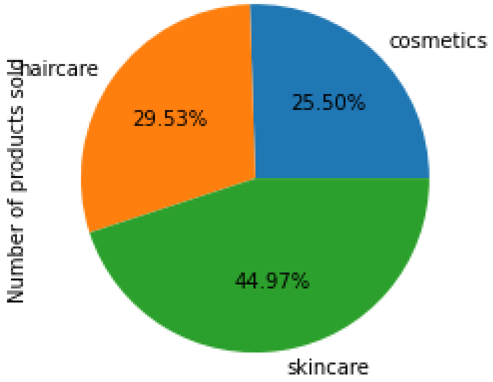
```
In [24]: sales_data = df.groupby('Product type')['Number of products sold'].sum().reset_
         sales_data
```

Out[24]:

	Product type	Number of products sold
0	cosmetics	11757
1	haircare	13611
2	skincare	20731

```
In [45]: Prod_type = df.groupby('Product type')['Number of products sold'].sum().plot(k
         Prod_type
```

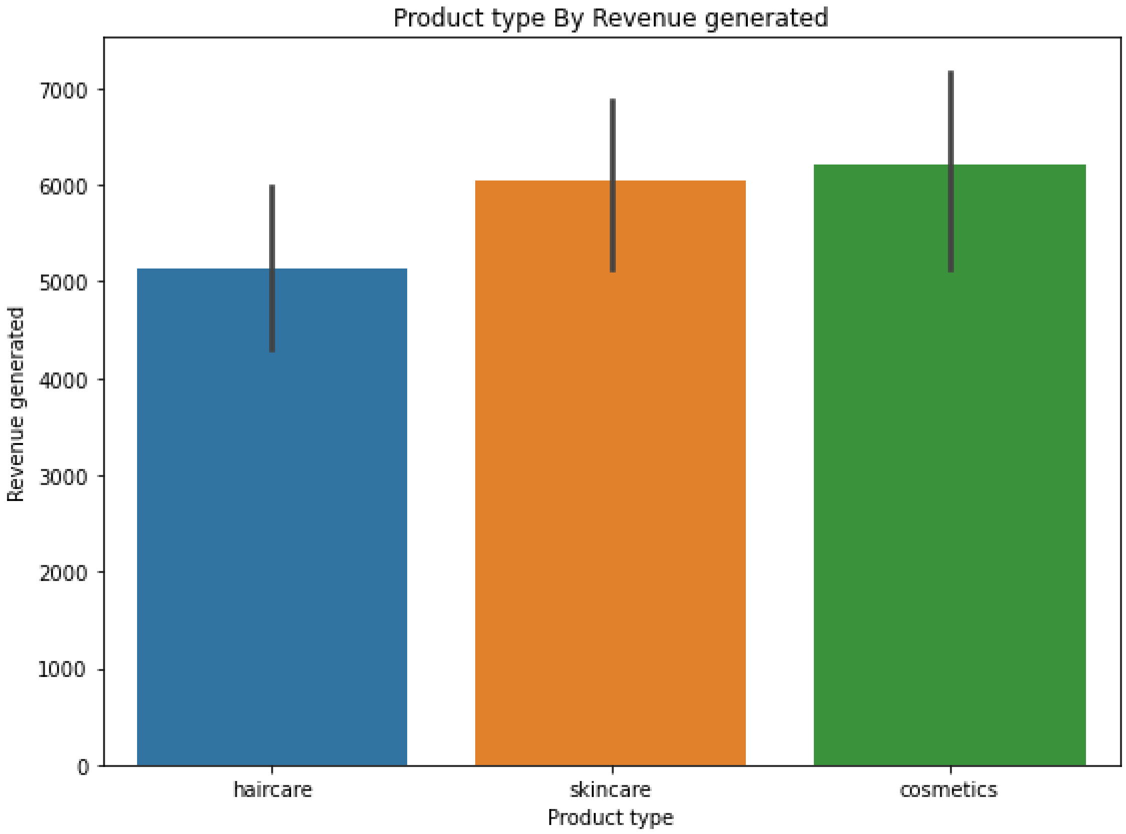
```
Out[45]: <AxesSubplot:ylabel='Number of products sold'>
```



Product Type By Revenue Generated

In [19]:

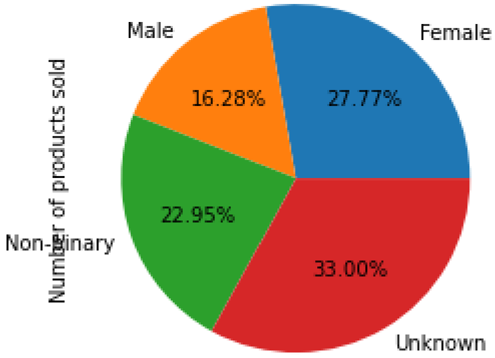
```
# Create a bar plot to visualize the correlation
plt.figure(figsize=(8, 6))
sns.barplot(data=df, x='Product type', y='Revenue generated')
plt.title('Product type By Revenue generated')
plt.xlabel('Product type')
plt.ylabel('Revenue generated')
plt.tight_layout()
plt.show()
```



Sales By Customer

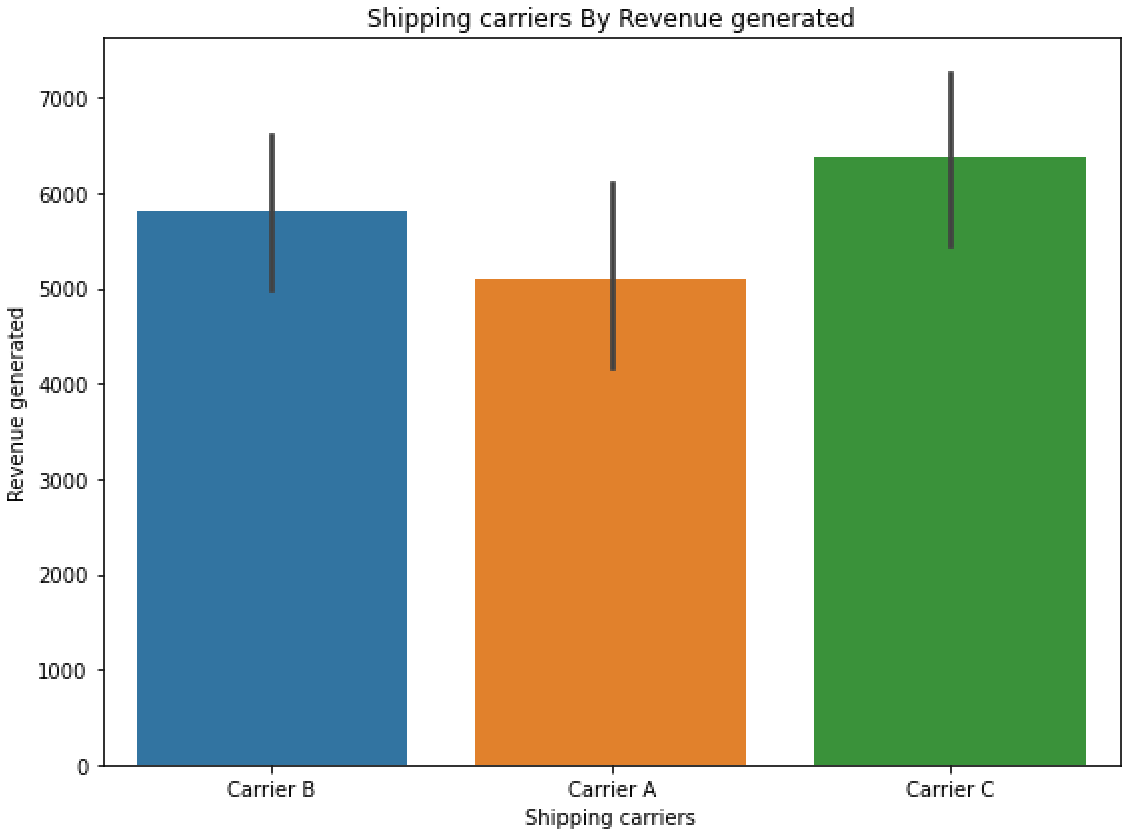
```
In [30]: Customer_Demograohics = df.groupby('Customer demographics')['Number of products
         Customer_Demograohics
```

```
Out[30]: <AxesSubplot:ylabel='Number of products sold'>
```



Revenue Generated By Shipping Carries

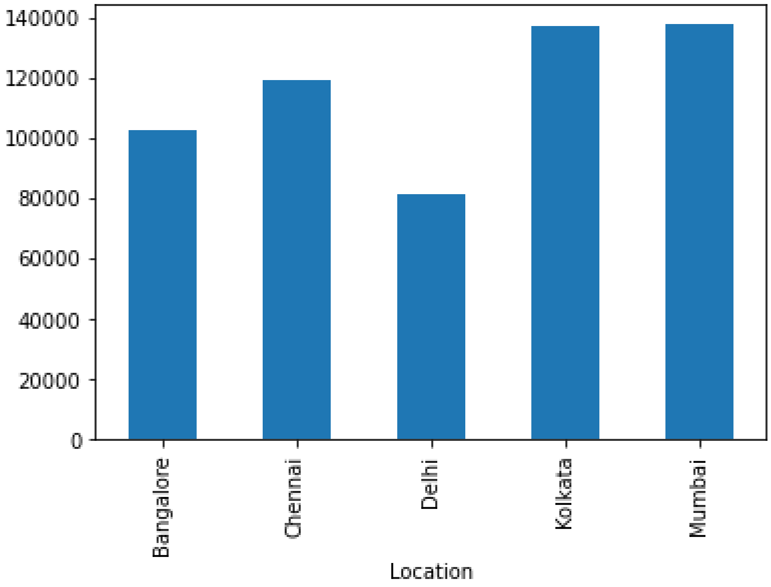
In [31]:
```python
# Create a bar plot to visualize the correlation
plt.figure(figsize=(8, 6))
sns.barplot(data=df, x='Shipping carriers', y='Revenue generated')
plt.title('Shipping carriers By Revenue generated')
plt.xlabel('Shipping carriers')
plt.ylabel('Revenue generated')
plt.tight_layout()
plt.show()
```



Revenue generated By Location

```
In [34]: Location_sales = df.groupby('Location')['Revenue generated'].sum().plot(kind='
         Location_sales
```

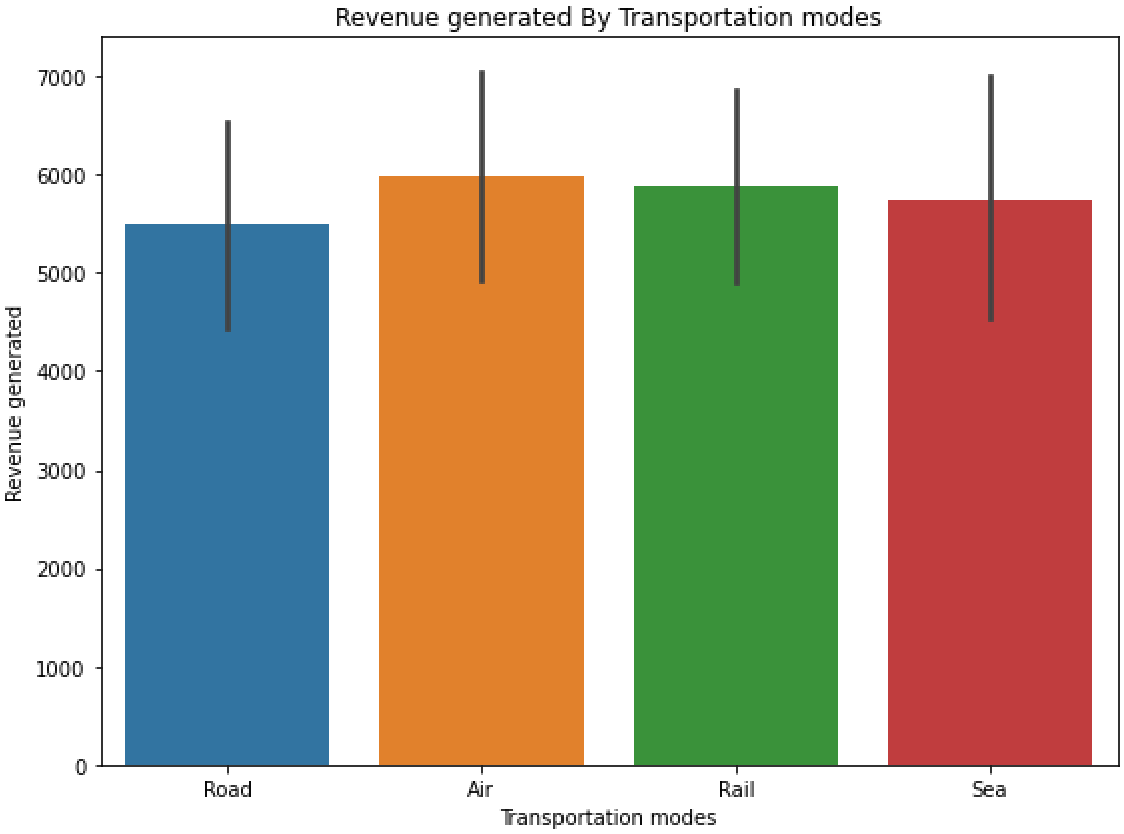
```
Out[34]: <AxesSubplot:xlabel='Location'>
```



Revenue generated By Transportation modes

In [38]:

```python
# Create a bar plot to visualize the correlation
plt.figure(figsize=(8, 6))
sns.barplot(data=df, x='Transportation modes', y='Revenue generated')
plt.title('Revenue generated By Transportation modes')
plt.xlabel('Transportation modes')
plt.ylabel('Revenue generated')
plt.tight_layout()
plt.show()
```



In [42]:

```python
avg_lead_time = df.groupby('Product type')['Lead time'].mean().reset_index()
avg_manufacturing_time = df.groupby('Product type')['Manufacturing costs'].mea
result = pd.merge(avg_lead_time,avg_manufacturing_time, on ='Product type')
result
```

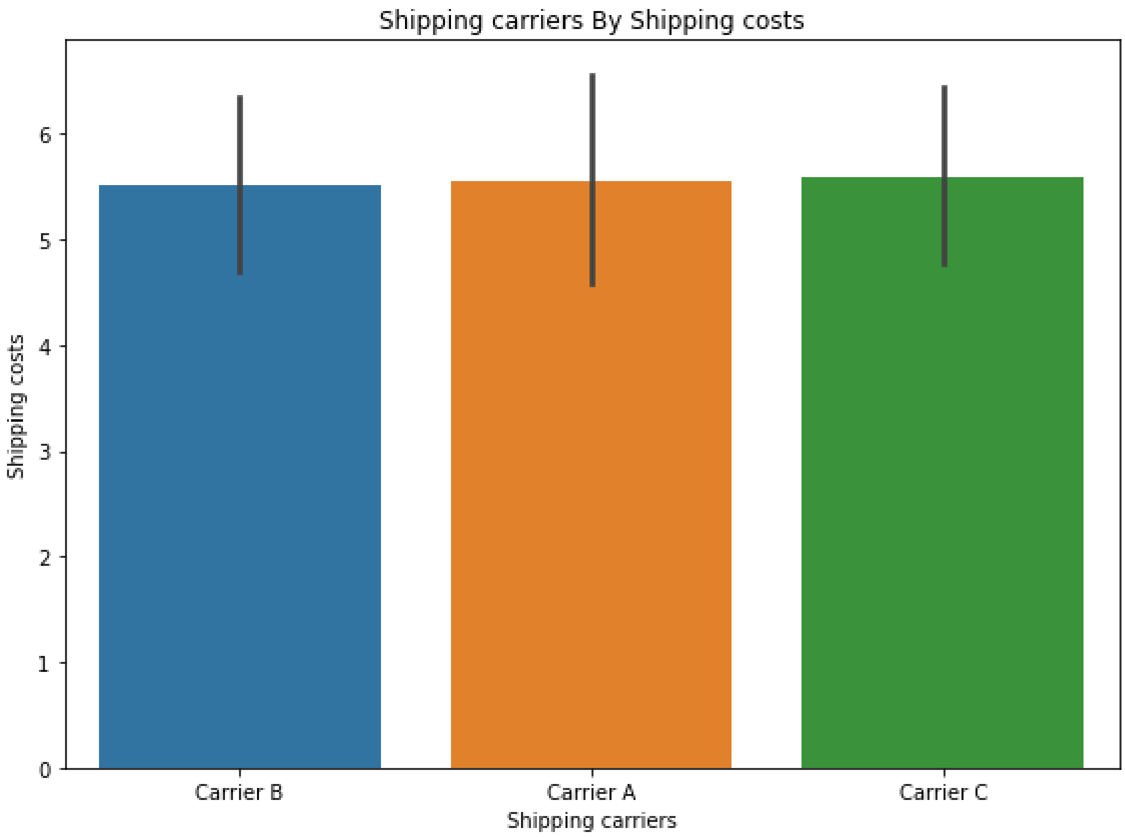
Out[42]:

	Product type	Lead time	Manufacturing costs
0	cosmetics	13.538462	43.052740
1	haircare	18.705882	48.457993
2	skincare	18.000000	48.993157

Shipping carriers By Shipping costs

In [43]:

```
# Create a bar plot to visualize the correlation
plt.figure(figsize=(8, 6))
sns.barplot(data=df, x='Shipping carriers', y='Shipping costs')
plt.title('Shipping carriers By Shipping costs')
plt.xlabel('Shipping carriers')
plt.ylabel('Shipping costs')
plt.tight_layout()
plt.show()
```



Average Defect Rates by Transportation modes

In [44]:

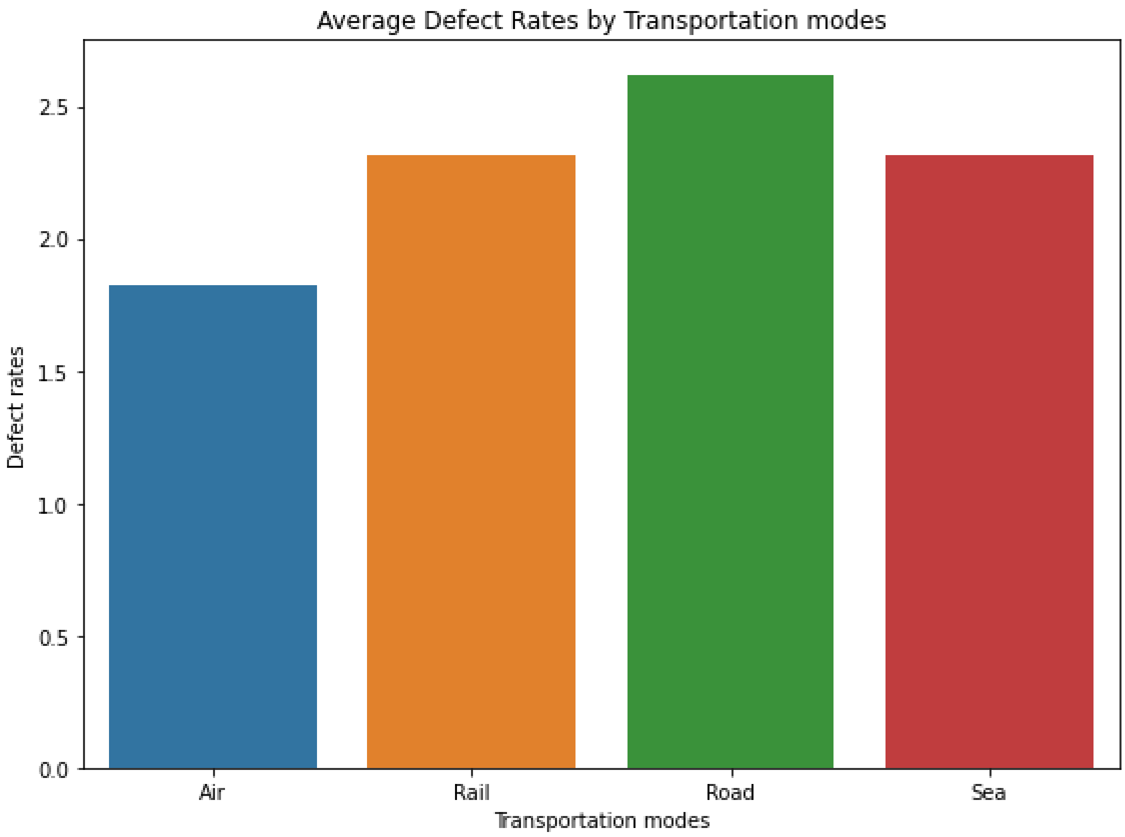
```
Avg_defeat_rate = df.groupby('Transportation modes')['Defect rates'].mean().res
Avg_defeat_rate
```

Out[44]:

	Transportation modes	Defect rates
0	Air	1.823924
1	Rail	2.318814
2	Road	2.620938
3	Sea	2.315281

In [57]:

```
# Create a bar plot to visualize the correlation
plt.figure(figsize=(8, 6))
sns.barplot(data= Avg_defeat_rate, x='Transportation modes', y='Defect rates')
plt.title('Average Defect Rates by Transportation modes')
plt.xlabel('Transportation modes')
plt.ylabel('Defect rates')
plt.tight_layout()
plt.show()
```



Average Defect Rates by Product Type

In [52]:

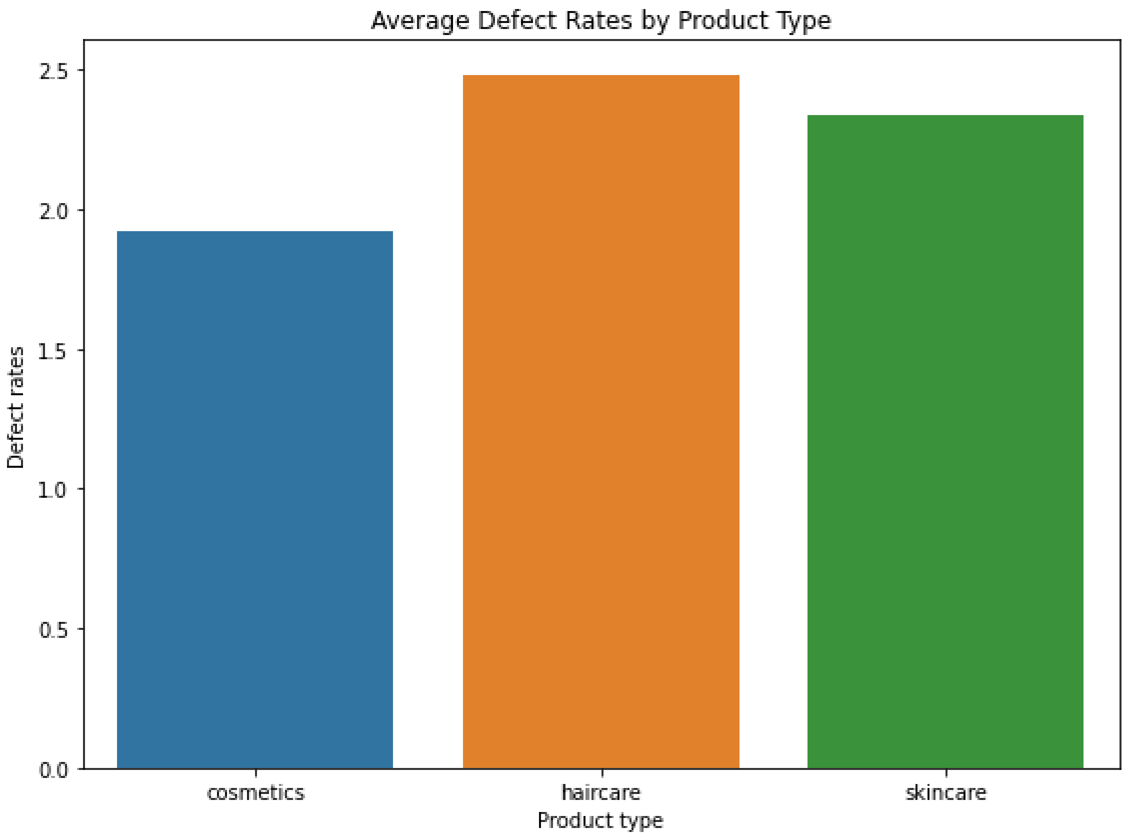
```
Avg_defeat_rate_Product = df.groupby('Product type')['Defect rates'].mean().res
Avg_defeat_rate_Product
```

Out[52]:

	Product type	Defect rates
0	cosmetics	1.919287
1	haircare	2.483150
2	skincare	2.334681

In [55]:

```
# Create a bar plot to visualize the correlation
plt.figure(figsize=(8, 6))
sns.barplot(data= Avg_defeat_rate_Product, x='Product type', y='Defect rates')
plt.title('Average Defect Rates by Product Type')
plt.xlabel('Product type')
plt.ylabel('Defect rates')
plt.tight_layout()
plt.show()
```



In [ ]: