In [1]:
```
import pandas as pd
```

In [545…
```
data=pd.read_csv("supply_chain_data.csv")
```

In [220…
```
data.dropna(inplace=True)
```

In [108…
```
data
```

Out[108…

	Product type	SKU	Price	Availability	Number of products sold	Revenue generated	Customer demographics	Sto lev
0	haircare	SKU0	69.808006	55	802	8661.996792	Non-binary	
1	skincare	SKU1	14.843523	95	736	7460.900065	Female	
2	haircare	SKU2	11.319683	34	8	9577.749626	Unknown	
3	skincare	SKU3	61.163343	68	83	7766.836426	Non-binary	
4	skincare	SKU4	4.805496	26	871	2686.505152	Non-binary	
...	...	...	...	...	...	...	...	
95	haircare	SKU95	77.903927	65	672	7386.363944	Unknown	
96	cosmetics	SKU96	24.423131	29	324	7698.424766	Non-binary	
97	haircare	SKU97	3.526111	56	62	4370.916580	Male	
98	skincare	SKU98	19.754605	43	913	8525.952560	Female	
99	haircare	SKU99	68.517833	17	627	9185.185829	Unknown	

100 rows × 24 columns

In [5]:
```
data.describe()
```

Out[5]:

	Price	Availability	Number of products sold	Revenue generated	Stock levels	Lead times	O quant
count	100.000000	100.000000	100.000000	100.000000	100.000000	100.000000	100.00
mean	49.462461	48.400000	460.990000	5776.048187	47.770000	15.960000	49.22
std	31.168193	30.743317	303.780074	2732.841744	31.369372	8.785801	26.78
min	1.699976	1.000000	8.000000	1061.618523	0.000000	1.000000	1.00
25%	19.597823	22.750000	184.250000	2812.847151	16.750000	8.000000	26.00
50%	51.239831	43.500000	392.500000	6006.352023	47.500000	17.000000	52.00
75%	77.198228	75.000000	704.250000	8253.976921	73.000000	24.000000	71.25
max	99.171329	100.000000	996.000000	9866.465458	100.000000	30.000000	96.00

In [6]:
```
df=data.isnull().sum()
```

In [7]:
```
df
```

Out[7]:
```
Product type               0
SKU                        0
Price                      0
Availability               0
Number of products sold    0
Revenue generated          0
Customer demographics      0
Stock levels               0
Lead times                 0
Order quantities           0
Shipping times             0
Shipping carriers          0
Shipping costs             0
Supplier name              0
Location                   0
Lead time                  0
Production volumes         0
Manufacturing lead time    0
Manufacturing costs        0
Inspection results         0
Defect rates               0
Transportation modes       0
Routes                     0
Costs                      0
dtype: int64
```

In [8]:
```
duplicate_rows = data[data.duplicated()]
```

In [9]:
```
duplicate_rows
```

Out[9]:

Product type	SKU	Price	Availability	Number of products sold	Revenue generated	Customer demographics	Stock levels	Lead times

0 rows × 24 columns

```
In [10]: data.dtypes
```

```
Out[10]: 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
```

```
In [11]: data.astype("object")
```

Out[11]:

	Product type	SKU	Price	Availability	Number of products sold	Revenue generated	Customer demographics	Sto lev
0	haircare	SKU0	69.808006	55	802	8661.996792	Non-binary	
1	skincare	SKU1	14.843523	95	736	7460.900065	Female	
2	haircare	SKU2	11.319683	34	8	9577.749626	Unknown	
3	skincare	SKU3	61.163343	68	83	7766.836426	Non-binary	
4	skincare	SKU4	4.805496	26	871	2686.505152	Non-binary	
...	...	...	...	...	...	...	...	
95	haircare	SKU95	77.903927	65	672	7386.363944	Unknown	
96	cosmetics	SKU96	24.423131	29	324	7698.424766	Non-binary	
97	haircare	SKU97	3.526111	56	62	4370.91658	Male	
98	skincare	SKU98	19.754605	43	913	8525.95256	Female	
99	haircare	SKU99	68.517833	17	627	9185.185829	Unknown	

100 rows × 24 columns

In [12]:

```
data.head()["Product type"]
```

Out[12]:

```
0    haircare
1    skincare
2    haircare
3    skincare
4    skincare
Name: Product type, dtype: object
```

In [13]:

```
data
```

Out[13]:

	Product type	SKU	Price	Availability	Number of products sold	Revenue generated	Customer demographics	Sto lev
0	haircare	SKU0	69.808006	55	802	8661.996792	Non-binary	
1	skincare	SKU1	14.843523	95	736	7460.900065	Female	
2	haircare	SKU2	11.319683	34	8	9577.749626	Unknown	
3	skincare	SKU3	61.163343	68	83	7766.836426	Non-binary	
4	skincare	SKU4	4.805496	26	871	2686.505152	Non-binary	
...	...	...	...	...	...	...	...	
95	haircare	SKU95	77.903927	65	672	7386.363944	Unknown	
96	cosmetics	SKU96	24.423131	29	324	7698.424766	Non-binary	
97	haircare	SKU97	3.526111	56	62	4370.916580	Male	
98	skincare	SKU98	19.754605	43	913	8525.952560	Female	
99	haircare	SKU99	68.517833	17	627	9185.185829	Unknown	

100 rows × 24 columns

In [15]:

```
data.dtypes
```

Out[15]:

```
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                    datetime64[ns]
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
```

In [21]:

```
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import plotly.express as px
import plotly.graph_objects as go
```

1.What is the total revenue generated by each product type?

In [311…

```
grouped_df=data.groupby("Product type")["Revenue generated"].sum()
pd.DataFrame(grouped_df)
```

Out[311…

	Revenue generated
Product type	
cosmetics	161521.265999
haircare	174455.390605
skincare	241628.162133

How does the distribution of sales vary across different customer segments?

In [43]:

```
data["Customer demographics"].describe()
```

Out[43]:

```
count            100
unique             4
top          Unknown
freq              31
Name: Customer demographics, dtype: object
```

In [317…

```
grouped_df1=data.groupby("Customer demographics")["Revenue generated"].sum()
pd.DataFrame(grouped_df1)
```

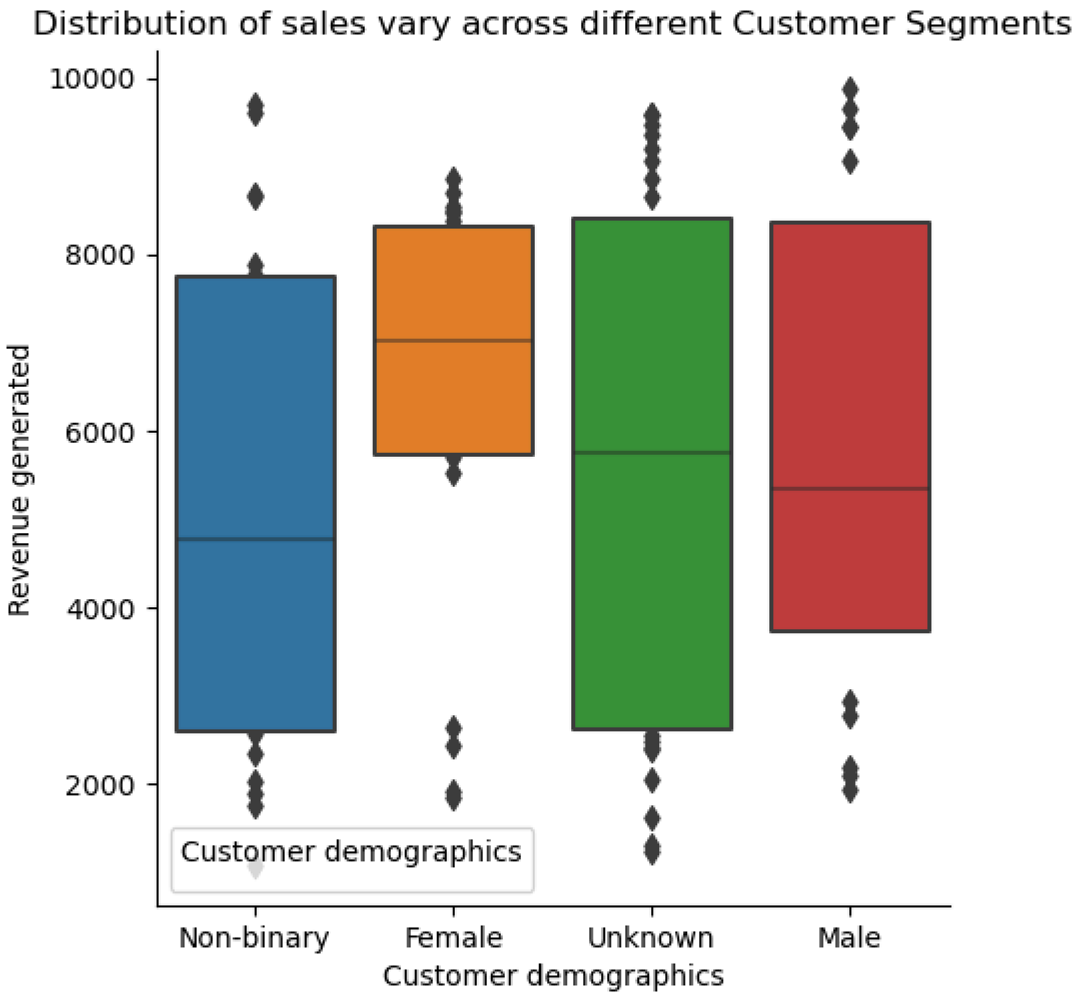
Out[317…

	Revenue generated
Customer demographics	
Female	161514.489122
Male	126634.394260
Non-binary	116365.801520
Unknown	173090.133837

In [75]:

```
sns.catplot(x = "Customer demographics",y="Revenue generated",data=data,kind="bc
plt.title("Distribution of sales vary across different Customer Segments")
plt.legend(title='Customer demographics')
plt.show()
```

```
C:\Users\skala\Desktop\New folder\Lib\site-packages\seaborn\categorical.py:1794:
FutureWarning:

use_inf_as_na option is deprecated and will be removed in a future version. Conve
rt inf values to NaN before operating instead.

C:\Users\skala\Desktop\New folder\Lib\site-packages\seaborn\categorical.py:1794:
FutureWarning:

use_inf_as_na option is deprecated and will be removed in a future version. Conve
rt inf values to NaN before operating instead.

C:\Users\skala\Desktop\New folder\Lib\site-packages\seaborn\categorical.py:1794:
FutureWarning:

use_inf_as_na option is deprecated and will be removed in a future version. Conve
rt inf values to NaN before operating instead.

C:\Users\skala\Desktop\New folder\Lib\site-packages\seaborn\categorical.py:1794:
FutureWarning:

use_inf_as_na option is deprecated and will be removed in a future version. Conve
rt inf values to NaN before operating instead.

No artists with labels found to put in legend.  Note that artists whose label sta
rt with an underscore are ignored when legend() is called with no argument.
```



Which product type has the highest and lowest average price?

In [136…
```
max_price=data[data["Price"]==data["Price"].max()]
```

In [137…
```
min_price=data[data["Price"]==data["Price"].min()]
```

In [135…
```
max_price_product
```

Out[135…

	Product type	SKU	Price	Availability	Number of products sold	Revenue generated	Customer demographics	Stoc leve
14	skincare	SKU14	99.171329	26	562	8653.570926	Non-binary	5

1 rows × 24 columns

In [154…
```
max_price_product=max_price["Product type"].values[0]
max_price2=max_price["Price"].values[0]
```

In [155…

```
min_price_product=min_price["Product type"].values[0]
min_price2=min_price["Price"].values[0]
```

In [338…

```
price=[max_price2,min_price2]
```

In [339…

```
product=[max_price_product,min_price_product]
```

In [343…

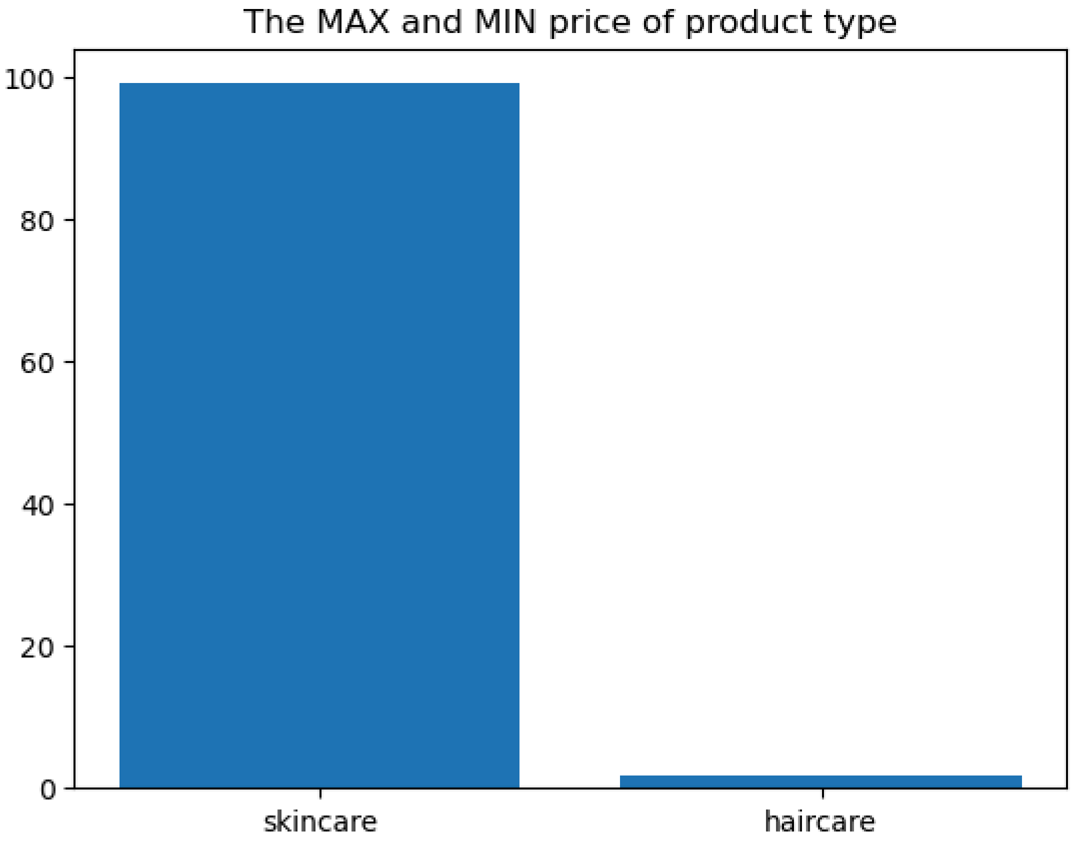
```
df =pd.DataFrame({'Product': product, 'high and low Price': price})

pd.DataFrame(df)
```

Out[343…

	Product	high and low Price
0	skincare	99.171329
1	haircare	1.699976

In [697…

```
plt.bar(product,price)
plt.title("The MAX and MIN price of product type")
plt.show()
```



In [222…

```
average_prices=data.groupby("Product type")["Price"].mean()
```

In [223…

```
average_prices
```

Out[223…

```
Product type
cosmetics    57.361058
haircare     46.014279
skincare     47.259329
Name: Price, dtype: float64
```

In [224…

```
highest_avg_price_product=average_prices.idxmax()
highest_avg_price=average_prices.max()
```

In [165…

```
highest_avg_price_product
```

Out[165…

```
'cosmetics'
```

In [174…

```
lowest_avg_price_product=average_prices.idxmin()
lowest_avg_price=average_prices.min()
```

In [175…

```
product2=[highest_avg_price_product,lowest_avg_price_product]
price2=[highest_avg_price,lowest_avg_price]
```

In [344…

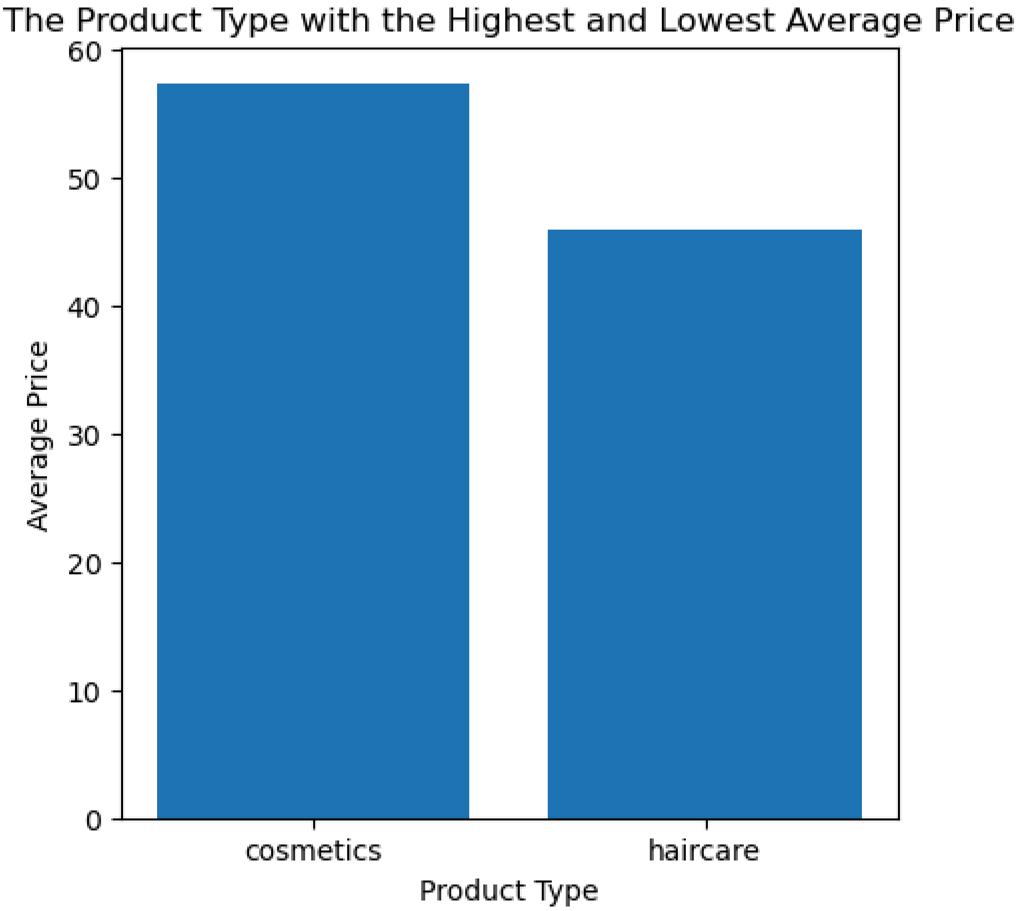
```
df2=pd.DataFrame({"Product Type":product2,"Highest and Lowest Average price":pri
pd.DataFrame(df2)
```

Out[344…

	Product Type	Highest and Lowest Average price
0	cosmetics	57.361058
1	haircare	46.014279

In [179…

```
plt.figure(figsize=[5,5])
plt.bar(product2,price2)
plt.title("The Product Type with the Highest and Lowest Average Price")
plt.xlabel("Product Type")
plt.ylabel("Average Price")
plt.show()
```



Is there a correlation between the number of products sold and the price?

In [360…

```
grouped_df2=data.groupby("Number of products sold")["Price"].sum().reset_index()
pd.DataFrame(grouped_df2)
```

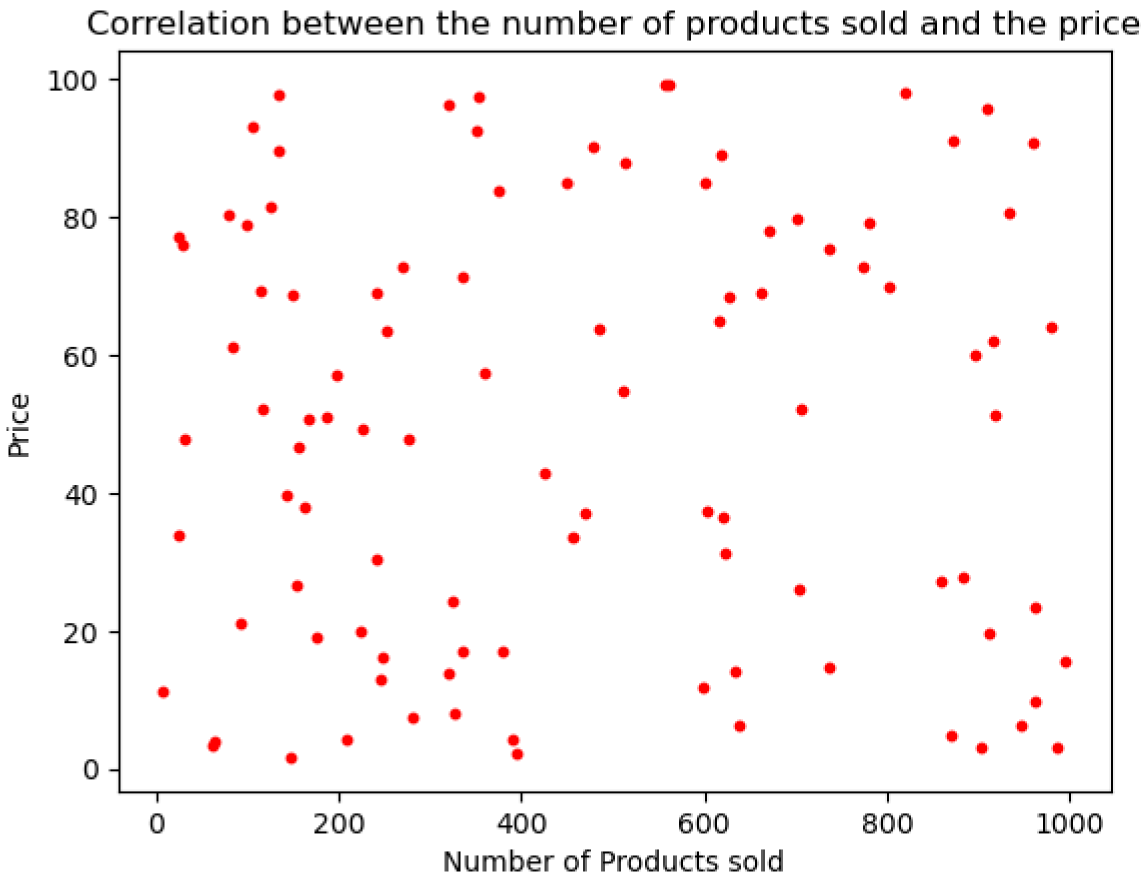
Out[360…

	Number of products sold	Price
0	8	11.319683
1	24	33.784138
2	25	76.962994
3	29	76.035544
4	32	47.914542
...	...	...
91	960	90.635460
92	963	33.212847
93	980	64.015733
94	987	3.037689
95	996	15.707796

96 rows × 2 columns

In [227…

```
nof=data["Number of products sold"]
price3=data["Price"]
plt.scatter(x=nof,y=price3,s=10,color="red")
plt.title("Correlation between the number of products sold and the price")
plt.xlabel("Number of Products sold")
plt.ylabel("Price")
plt.show()
```



How does the availability of products impact sales?

In [365…

```
grouped_df3=data.groupby("Availability")["Revenue generated"].sum().reset_index(
pd.DataFrame(grouped_df3)
```

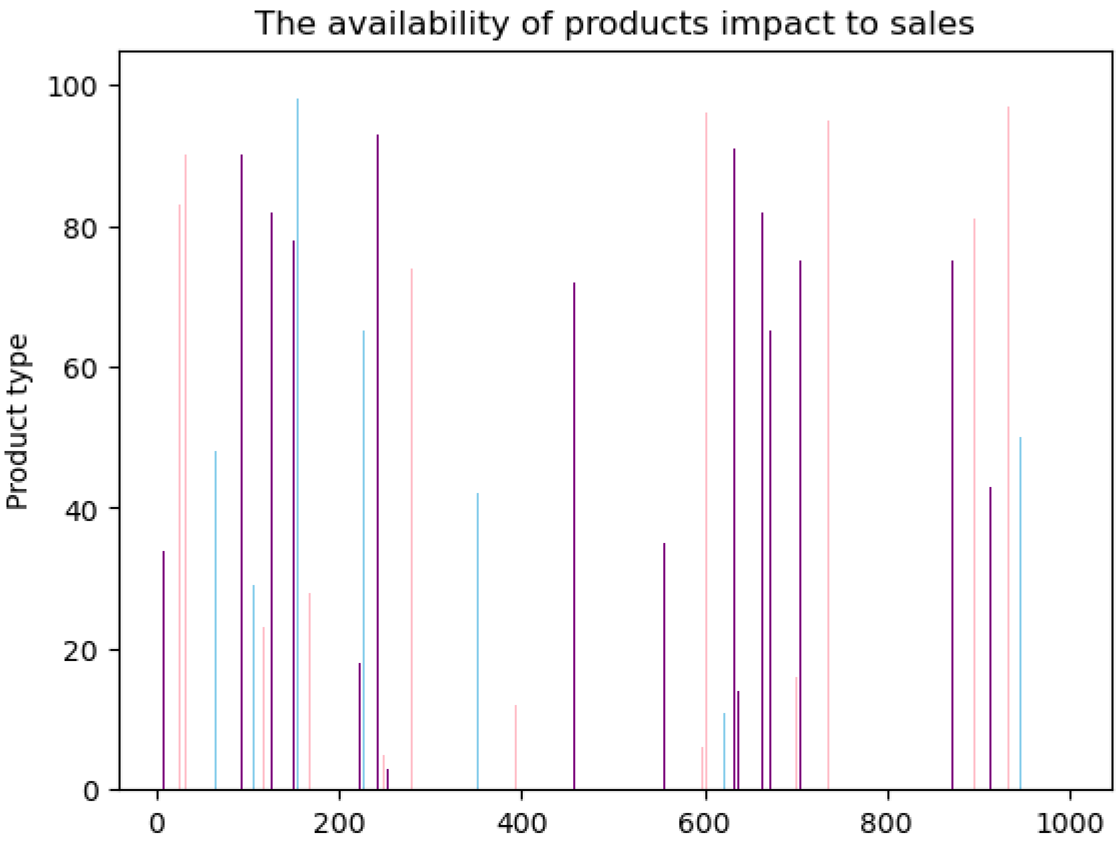
Out[365…

	Availability	Revenue generated
0	1	14703.719416
1	3	8318.903195
2	5	6491.078347
3	6	5737.425599
4	9	8100.906726
...	...	...
58	96	9061.710896
59	97	13613.315897
60	98	1839.609426
61	99	2048.290100
62	100	2553.495585

63 rows × 2 columns

In [274…

```
pro=data["Number of products sold"]
avail=data["Availability"]
plt.bar(pro,avail,color=["skyblue","pink","purple"])
plt.title("The availability of products impact to sales")
plt.xlabel=("Availability")
plt.ylabel("Product type")
plt.show()
```



How does the availability of products impact sales?

In [370…
```
grouped_df4=data.groupby("Availability")["Revenue generated"].sum().reset_index(
pd.DataFrame(grouped_df4)
```

Out[370…
```
     Availability  Revenue generated
 0              1       14703.719416
 1              3        8318.903195
 2              5        6491.078347
 3              6        5737.425599
 4              9        8100.906726
...           ...                ...
58             96        9061.710896
59             97       13613.315897
60             98        1839.609426
61             99        2048.290100
62            100        2553.495585

63 rows × 2 columns
```

What are the top-selling products in terms of revenue generated?

In [388…
```
grouped_df5=data.groupby("Product type")["Revenue generated"].sum().sort_values(
pd.DataFrame(grouped_df5)
```

Out[388…
```
              Revenue generated
Product type
    skincare      241628.162133
    haircare      174455.390605
   cosmetics      161521.265999
```

How does the revenue generated vary across different locations?

In [391…

```
grouped_df5=data.groupby("Location")["Revenue generated"].sum().reset_index()
pd.DataFrame(grouped_df5)
```

Out[391…

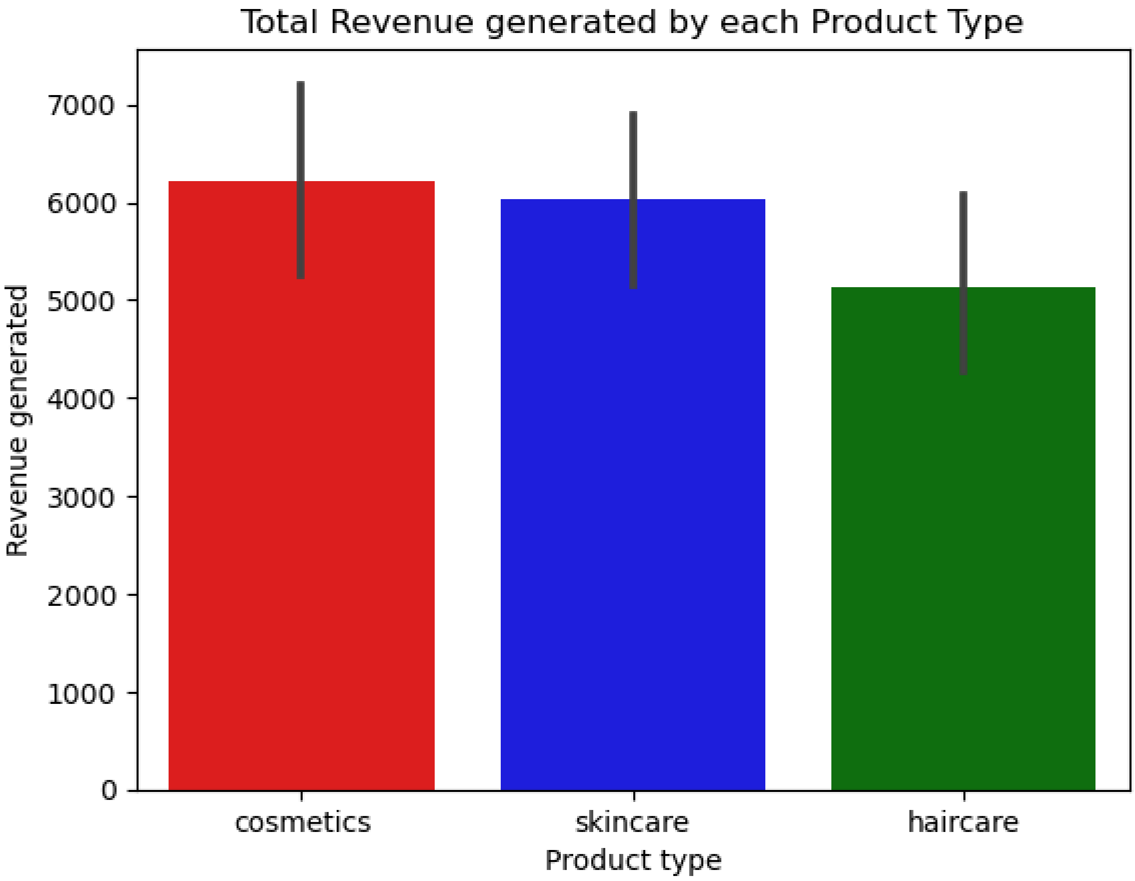
	Location	Revenue generated
0	Bangalore	102601.723882
1	Chennai	119142.815748
2	Delhi	81027.701225
3	Kolkata	137077.551005
4	Mumbai	137755.026877

Visualisations

1. Product revenue comparison

In [401…

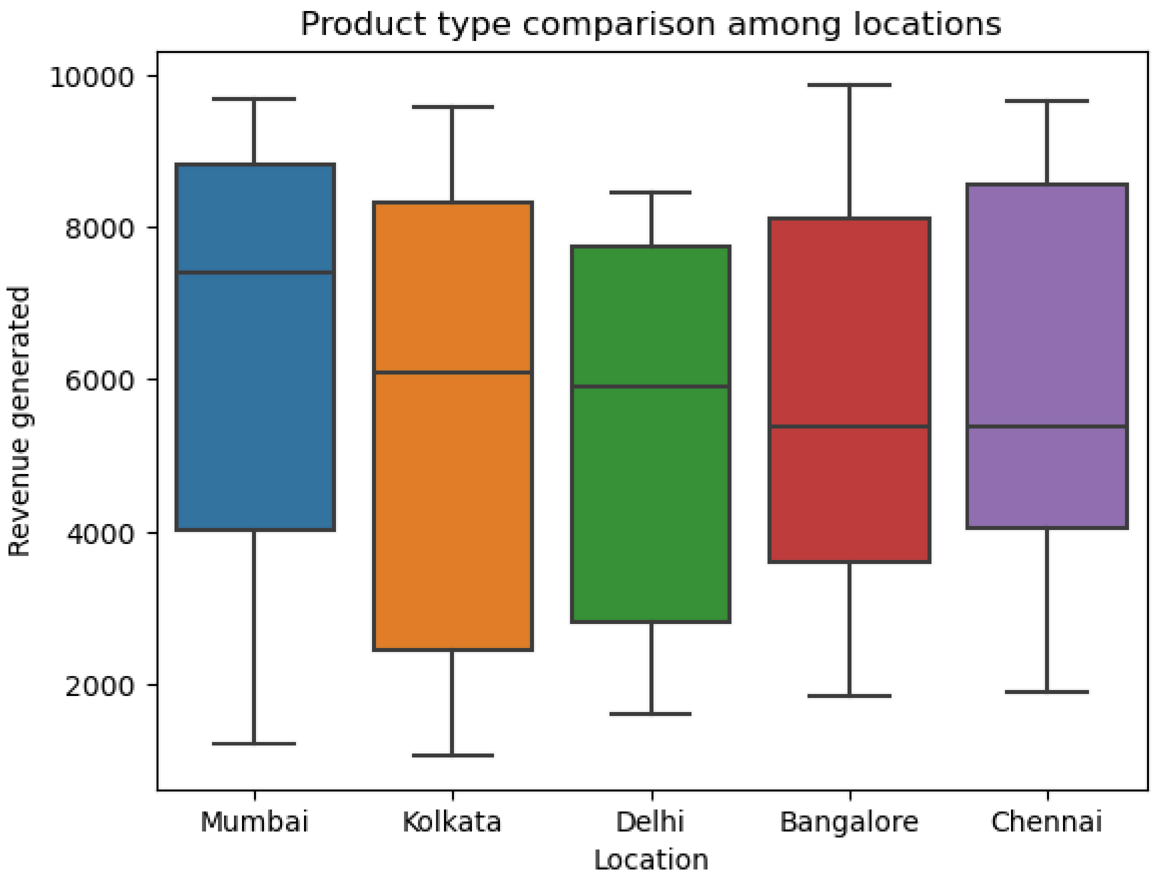
```
order1=["cosmetics","skincare","haircare"]
sns.barplot(x="Product type",y="Revenue generated",data=data,order=order1,palett
plt.title("Total Revenue generated by each Product Type")
plt.show()
```



2. Customer demographics analysis

In [421…

```
sns.boxplot(x="Location",y="Revenue generated",data=data)
plt.title("Product type comparison among locations")



plt.show()
```

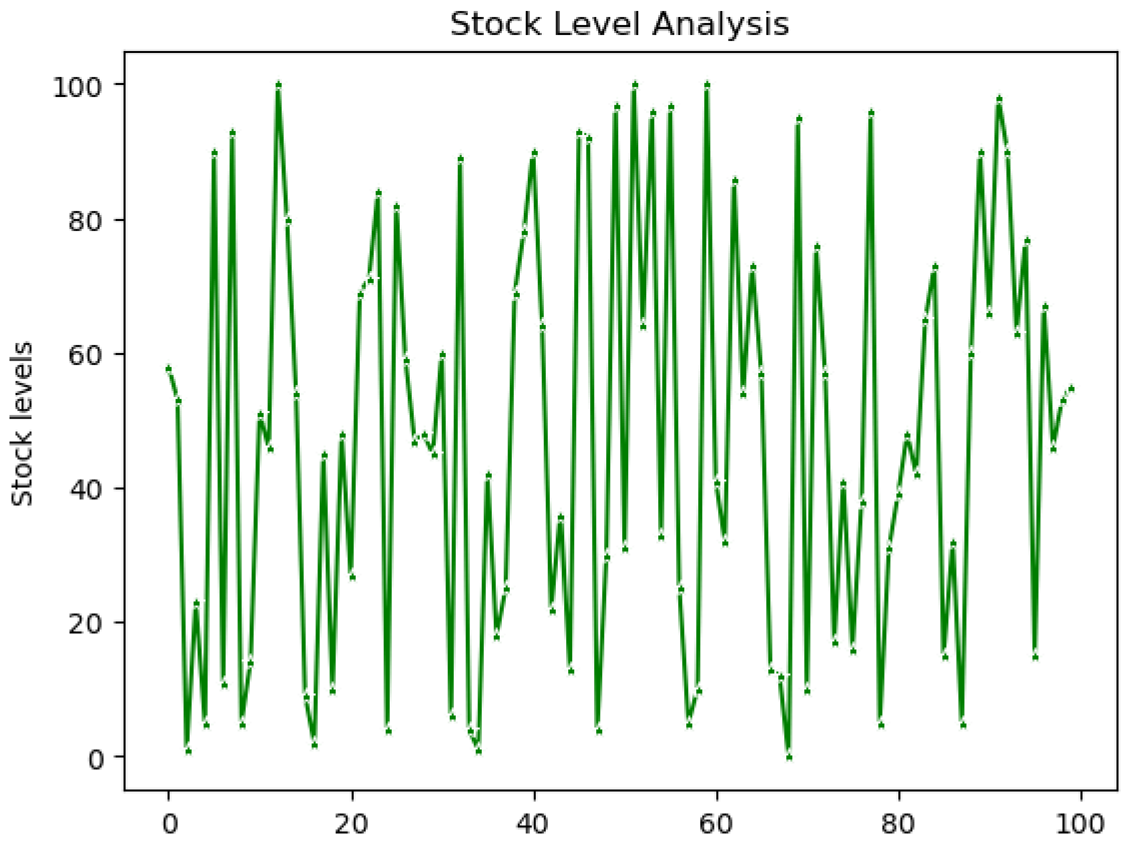


3. Stock level analysis

In [419…

```
sns.lineplot(data=data["Stock levels"],marker="*",color="green")
sns.dark_palette("#b285bc", as_cmap=True)

plt.title("Stock Level Analysis")
plt.show()
```

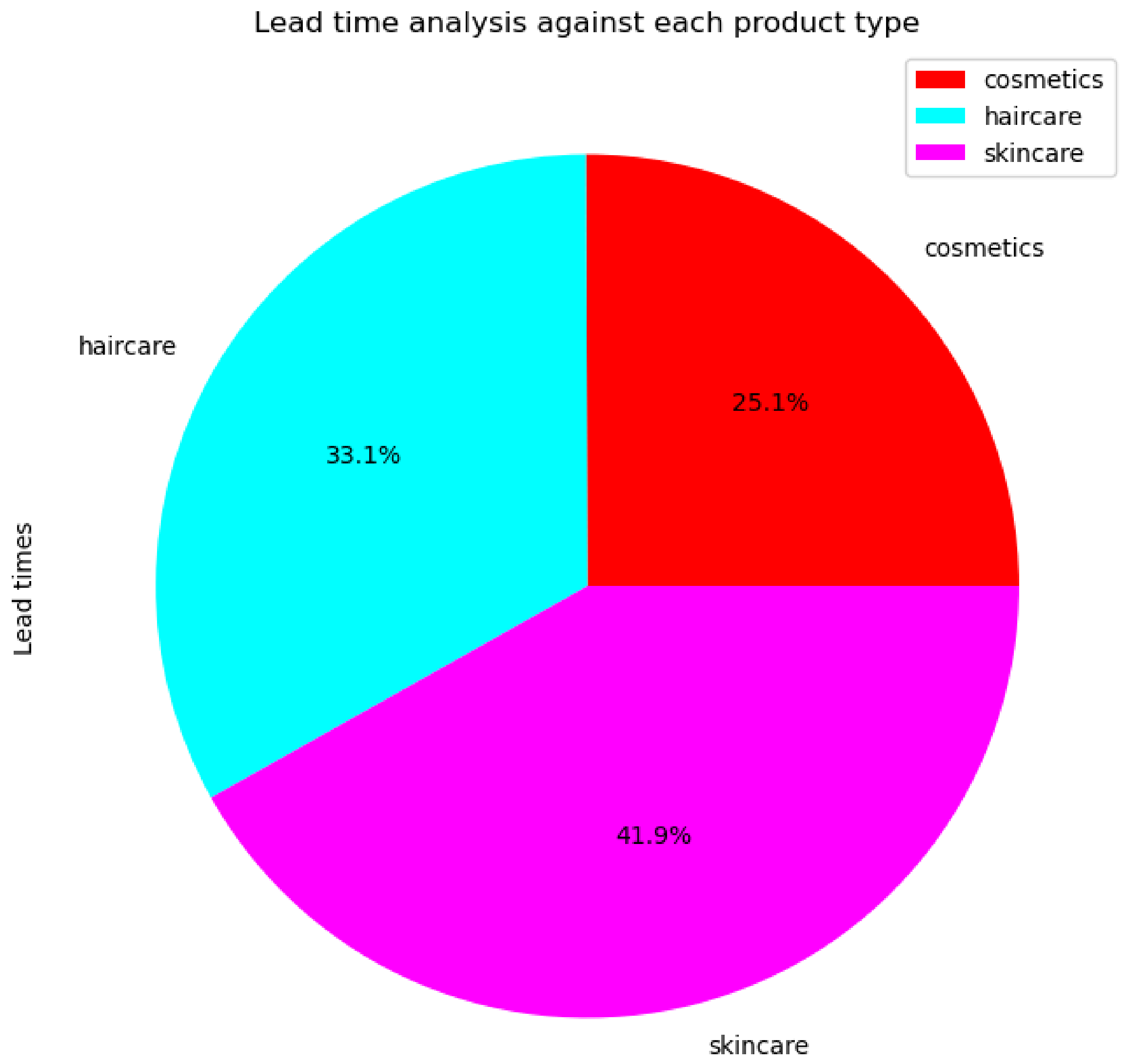
```
C:\Users\skala\Desktop\New folder\Lib\site-packages\seaborn\_oldcore.py:1119: Fut
ureWarning:

use_inf_as_na option is deprecated and will be removed in a future version. Conve
rt inf values to NaN before operating instead.

C:\Users\skala\Desktop\New folder\Lib\site-packages\seaborn\_oldcore.py:1119: Fut
ureWarning:

use_inf_as_na option is deprecated and will be removed in a future version. Conve
rt inf values to NaN before operating instead.
```



4. Lead time analysis

In [464…

```
grouped_df6=data.groupby("Product type")["Lead times"].sum()
c=["red", "cyan", "magenta"]
plt.figure(figsize=(10,8))
grouped_df6.plot(kind="pie",autopct="%1.1f%%",colors=c)
plt.title("Lead time analysis against each product type")
plt.legend(grouped_df6.index,loc="upper right",)
plt.show()
```

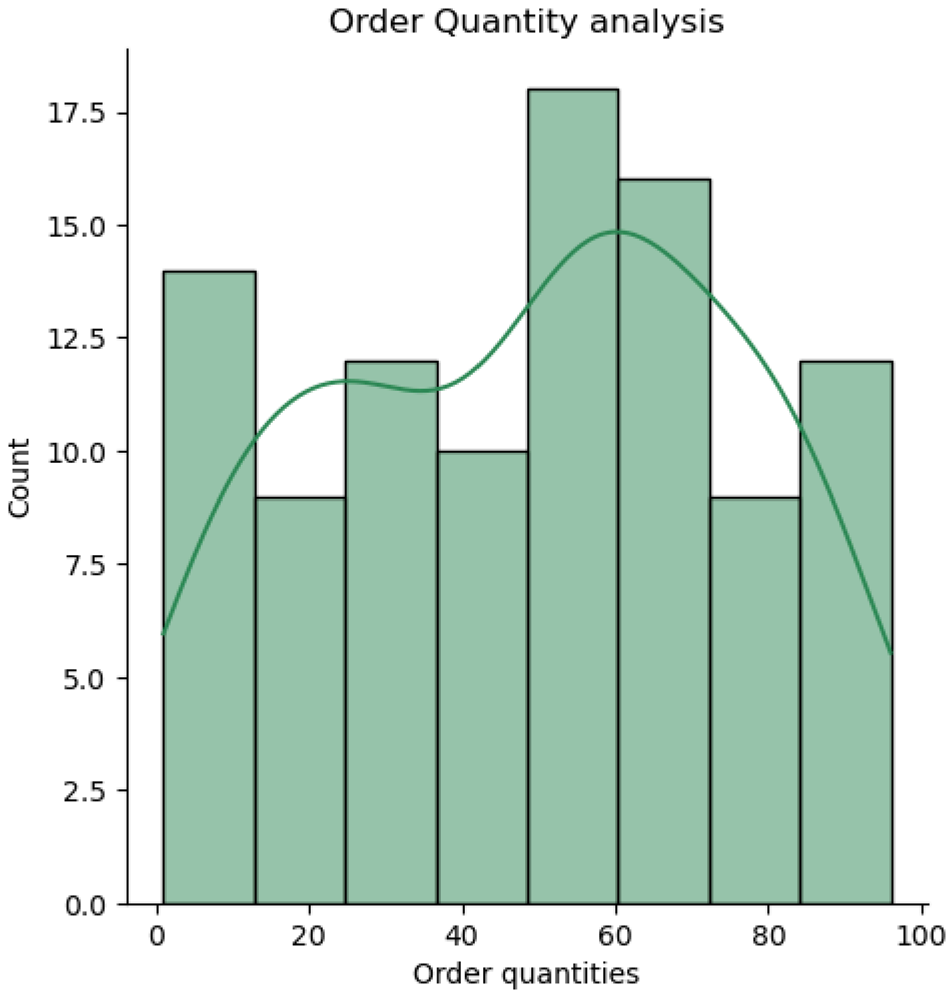


5. Order quantity Trend

In [490…

```
sns.displot(data["Order quantities"],kde=True,color="seagreen")
plt.title("Order Quantity analysis")


plt.show()
```

```
C:\Users\skala\Desktop\New folder\Lib\site-packages\seaborn\_oldcore.py:1119: Fut
ureWarning:

use_inf_as_na option is deprecated and will be removed in a future version. Conve
rt inf values to NaN before operating instead.
```



6. Shipping costs comparison

In [503…

```
grouped_df7
```

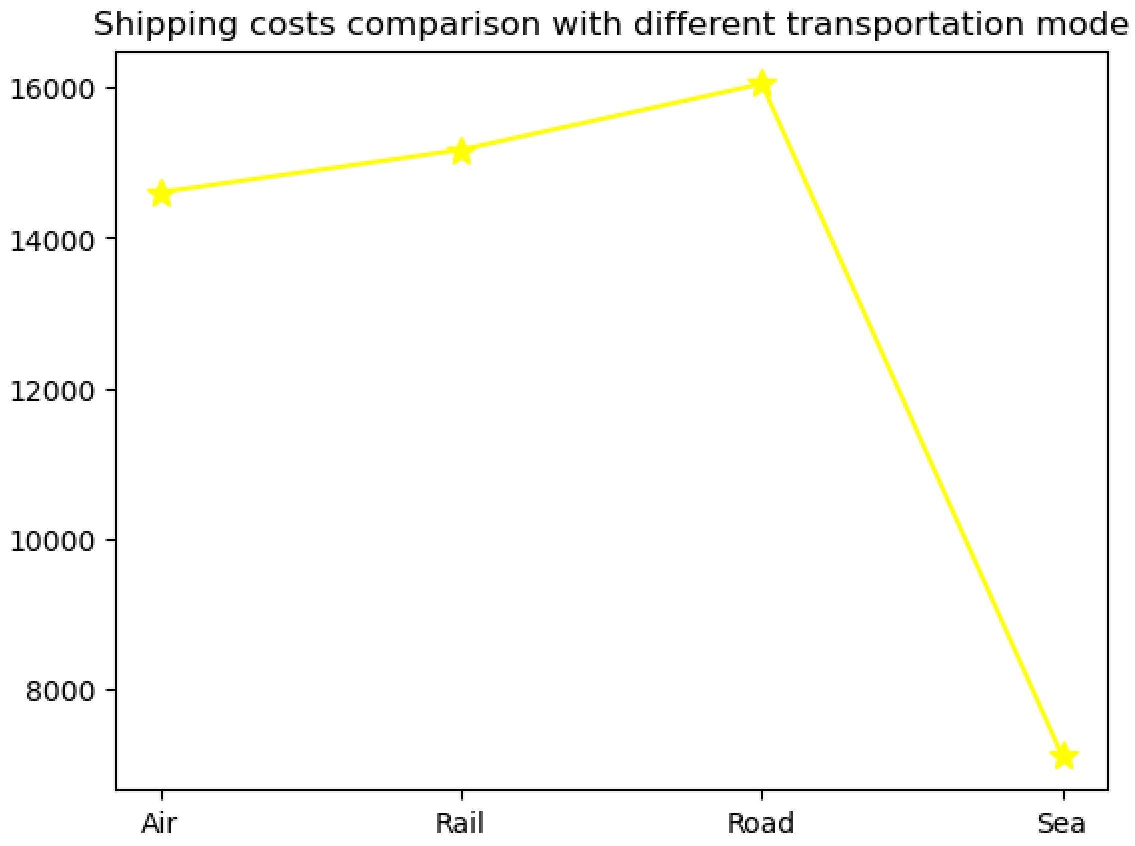
Out[503…

```
Transportation modes
Air     14604.527498
Rail    15168.931559
Road    16048.193639
Sea      7102.925520
Name: Costs, dtype: float64
```

In [526…

```
plt.plot(grouped_df7.index, grouped_df7.values, marker='*', markersize=10,label=
plt.title("Shipping costs comparison with different transportation mode")
plt.xlabel("Transportation")
plt.ylabel("Cost")
plt.show()
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[526], line 3
      1 plt.plot(grouped_df7.index, grouped_df7.values, marker='*', markersize=1
0,label=transport_mode,color="yellow")
      2 plt.title("Shipping costs comparison with different transportation mode")
----> 3 plt.xlabel("Transportation")
      4 plt.ylabel("Cost")
      5 plt.show()

TypeError: 'str' object is not callable
```



7. Supplier performance evaluation

In [551…

```
plt.scatter(data["Production volumes"],s=10)
plt.show()
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[551], line 1
----> 1 plt.scatter(data["Production volumes"],s=10)
      2 plt.show()

TypeError: scatter() missing 1 required positional argument: 'y'
```

In [542…

```
data['Production Volume'] = data['Production Volume'].astype(int)

# Display the DataFrame to verify the changes
print(data)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[542], line 1
----> 1 data['Production Volume'] = data['Production Volume'].astype(int)
      3 # Display the DataFrame to verify the changes
      4 print(data)

TypeError: 'float' object is not subscriptable
```

In [548…

```
data['Production volumes'] = data['Production volumes'].astype(int)
```

In [550…

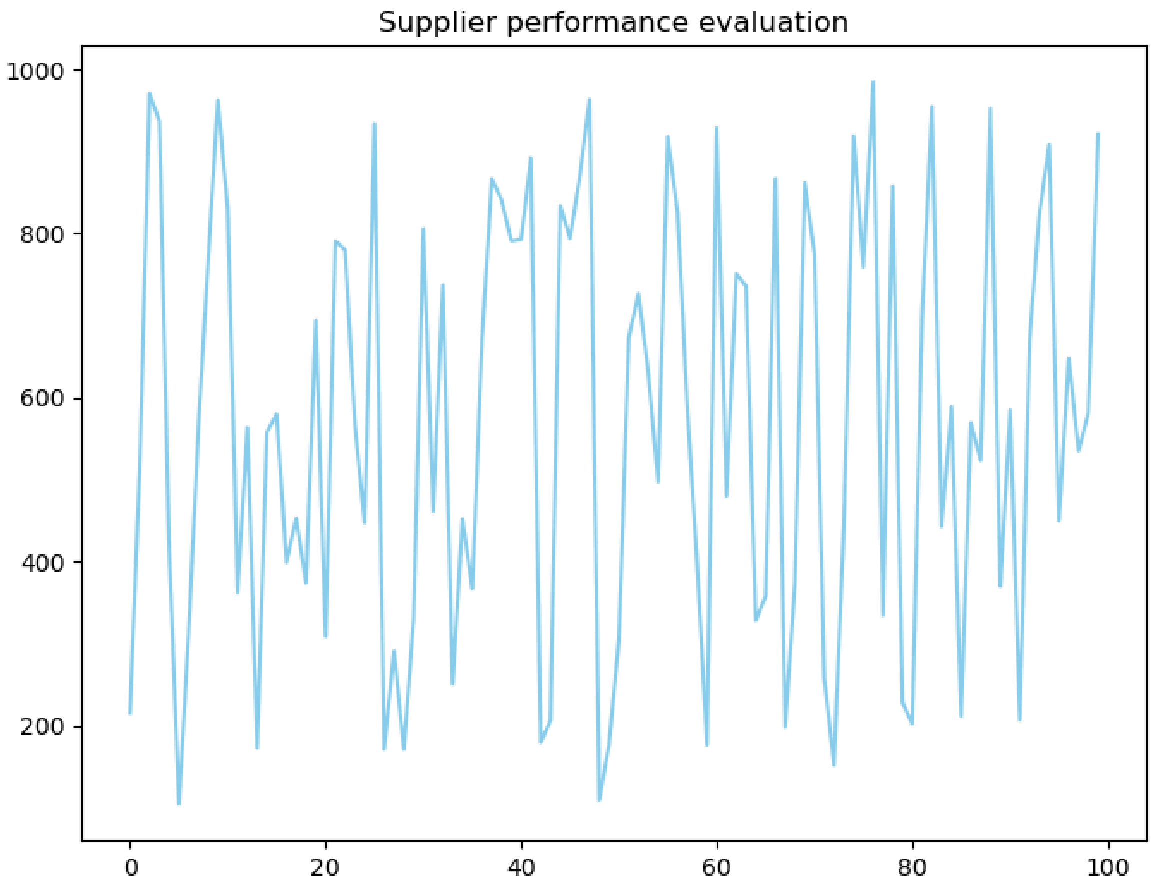
```
data.dtypes
```

Out[550…

```
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           int32
Manufacturing lead time      int64
Manufacturing costs        float64
Inspection results          object
Defect rates               float64
Transportation modes        object
Routes                      object
Costs                      float64
dtype: object
```

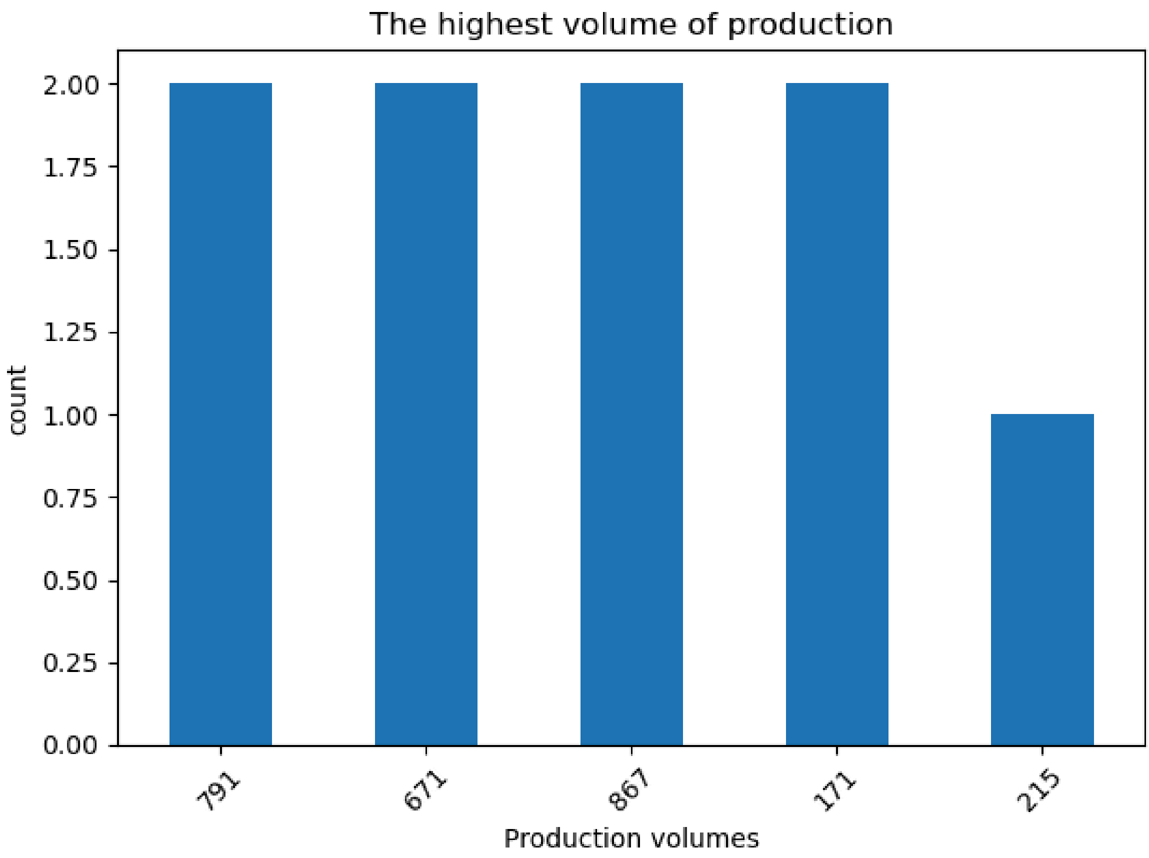
In [564…

```
df9=data["Production volumes"]
plt.figure(figsize=(8, 6))
plt.plot(range(len(df9)), df9, color='skyblue', label='Data Points')
plt.title("Supplier performance evaluation")

plt.show()
```



In [599…

```
max_pro = data['Production volumes'].value_counts().nlargest(5)
max_pro.plot(kind="bar")
plt.title("The highest volume of production")

plt.ylabel("count")
plt.xticks(rotation=45)

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



8. Transportation modes analysis

In [601…

```
import plotly.graph_objects as go
```

In [604…

```
trans=data["Transportation modes"].value_counts()
```

In [605…

```
trans
```

Out[605…

```
Transportation modes
Road    29
Rail    28
Air     26
Sea     17
Name: count, dtype: int64
```

In [615…

```
plt.figure(figsize=[50,50])
fig=go.Figure(go.Scatter(x=trans.index,y=trans.values,mode="markers",marker_size
fig.update_layout(title="Transportation modes analysis",xaxis_title="Transporati
fig.show()
```

```
<Figure size 5000x5000 with 0 Axes>
```

Routes efficiency assessment

In [633…

```
route=data["Routes"].unique()
count=data["Routes"].value_counts()
```

In [657…

```
fig=go.Figure(data=[go.Pie(labels=route,values=count)])
fig.update_layout(title="Transportation modes analysis")
fig.show()
```

10. Location based analysis

In [636…

```
grouped_df5=data.groupby("Location")["Revenue generated"].sum().reset_index()
pd.DataFrame(grouped_df5)
```

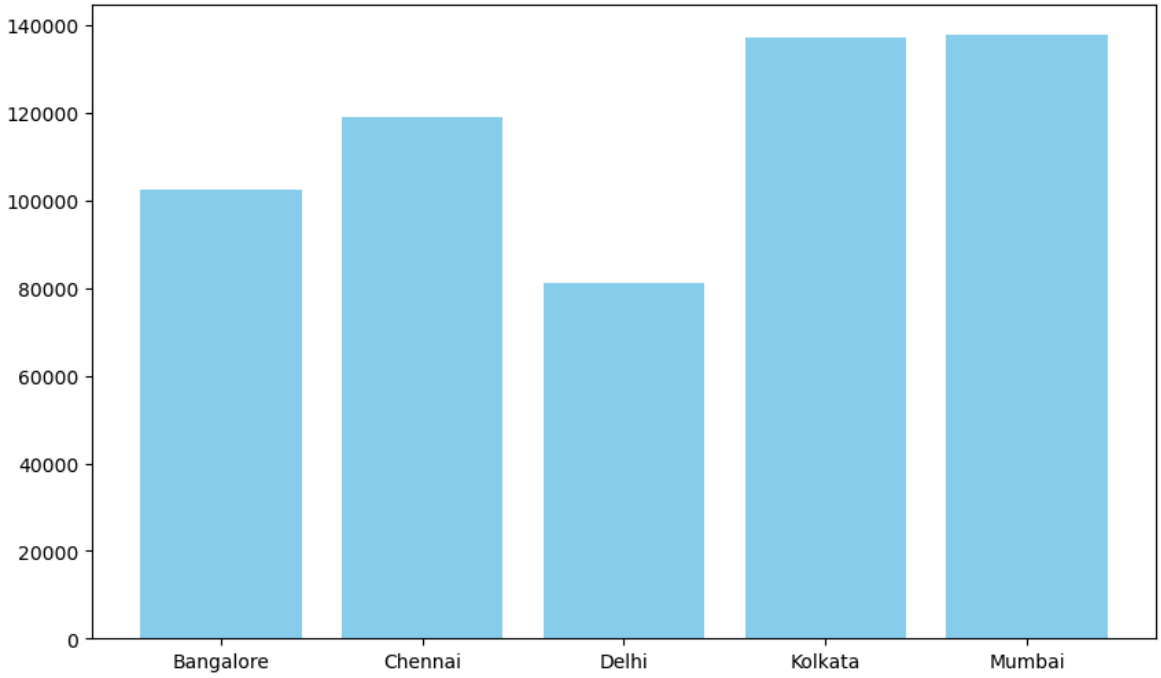
Out[636…

	Location	Revenue generated
0	Bangalore	102601.723882
1	Chennai	119142.815748
2	Delhi	81027.701225
3	Kolkata	137077.551005
4	Mumbai	137755.026877

In [655…

```
import matplotlib.pyplot as plt

grouped_df5 = data.groupby("Location")["Revenue generated"].sum().reset_index()

plt.figure(figsize=(10, 6))
plt.bar(grouped_df5["Location"], grouped_df5["Revenue generated"], color='skyblu

plt.xlabel("Location")
plt.ylabel("Revenue generated")
plt.title("Location-based Analysis")

plt.xticks(rotation=45)
plt.grid(True)
plt.show()
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[655], line 8
      5 plt.figure(figsize=(10, 6))
      6 plt.bar(grouped_df5["Location"], grouped_df5["Revenue generated"], color
='skyblue')
----> 8 plt.xlabel("Location")
      9 plt.ylabel("Revenue generated")
     10 plt.title("Location-based Analysis")

TypeError: 'str' object is not callable
```



```
In [ ]:
```