

```
In [1]: import pandas as pd  
import numpy as np  
import matplotlib.pyplot as plt  
import seaborn as sns
```

```
In [2]: df = pd.read_csv("supply_chain_data.csv")
```

```
In [3]: print(df.head())  
print(df.info())  
print(df.describe())
```

	Product type	SKU	Price	Availability	Number of products sold	\
0	haircare	SKU0	69.808006	55	802	
1	skincare	SKU1	14.843523	95	736	
2	haircare	SKU2	11.319683	34	8	
3	skincare	SKU3	61.163343	68	83	
4	skincare	SKU4	4.805496	26	871	

	Revenue generated	Customer demographics	Stock levels	Lead times	\
0	8661.996792	Non-binary	58	7	
1	7460.900065	Female	53	30	
2	9577.749626	Unknown	1	10	
3	7766.836426	Non-binary	23	13	
4	2686.505152	Non-binary	5	3	

	Order quantities	...	Location	Lead time	Production volumes	\
0	96	...	Mumbai	29	215	
1	37	...	Mumbai	23	517	
2	88	...	Mumbai	12	971	
3	59	...	Kolkata	24	937	
4	56	...	Delhi	5	414	

	Manufacturing lead time	Manufacturing costs	Inspection results	\
0	29	46.279879	Pending	
1	30	33.616769	Pending	
2	27	30.688019	Pending	
3	18	35.624741	Fail	
4	3	92.065161	Fail	

	Defect rates	Transportation modes	Routes	Costs
0	0.226410	Road	Route B	187.752075
1	4.854068	Road	Route B	503.065579
2	4.580593	Air	Route C	141.920282
3	4.746649	Rail	Route A	254.776159
4	3.145580	Air	Route A	923.440632

[5 rows x 24 columns]

<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
None
          Price Availability Number of products sold Revenue generated \
count  100.000000  100.000000      100.000000  100.000000
mean   49.462461  48.400000      460.990000  5776.048187
std    31.168193  30.743317      303.780074  2732.841744
min    1.699976  1.000000       8.000000  1061.618523
25%   19.597823  22.750000      184.250000  2812.847151
50%   51.239831  43.500000      392.500000  6006.352023
75%   77.198228  75.000000      704.250000  8253.976921
max   99.171329  100.000000     996.000000  9866.465458

          Stock levels Lead times Order quantities Shipping times \
count  100.000000  100.000000      100.000000  100.000000
mean   47.770000  15.960000      49.220000  5.750000
std    31.369372  8.785801      26.784429  2.724283
min    0.000000  1.000000       1.000000  1.000000
25%   16.750000  8.000000      26.000000  3.750000
50%   47.500000  17.000000      52.000000  6.000000
75%   73.000000  24.000000      71.250000  8.000000
max   100.000000 30.000000      96.000000  10.000000

          Shipping costs Lead time Production volumes \
count  100.000000  100.000000      100.000000
mean   5.548149  17.080000      567.840000
std    2.651376  8.846251      263.046861
min    1.013487  1.000000      104.000000
25%   3.540248  10.000000      352.000000
50%   5.320534  18.000000      568.500000
75%   7.601695  25.000000      797.000000
max   9.929816  30.000000      985.000000

          Manufacturing lead time Manufacturing costs Defect rates      Costs
count        100.000000            100.000000  100.000000  100.000000
mean         14.770000            47.266693  2.277158  529.245782
std          8.91243             28.982841  1.461366  258.301696
min          1.00000              1.085069  0.018608  103.916248
25%          7.00000              22.983299  1.009650  318.778455
50%          14.00000             45.905622  2.141863  520.430444
75%          23.00000             68.621026  3.563995  763.078231
max          30.00000             99.466109  4.939255  997.413450

```

```
In [4]: print(df.isnull().sum())
```

```
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 [5]: print("Shape:", df.shape)
```

```
Shape: (100, 24)
```

```
In [6]: df.duplicated().sum()
```

```
Out[6]: np.int64(0)
```

```
In [7]: df = df.drop_duplicates()
```

```
In [8]: df.columns
```

```
Out[8]: 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 [9]: df.nunique()
```

```
Out[9]: 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
```

```
In [10]: cols = ["Price", "Shipping costs", "Manufacturing costs", "Costs", "Stock levels"]
summary = df[cols].agg(["min", "max"]).T
print(summary)
```

	min	max
Price	1.699976	99.171329
Shipping costs	1.013487	9.929816
Manufacturing costs	1.085069	99.466109
Costs	103.916248	997.413450
Stock levels	0.000000	100.000000
Defect rates	0.018608	4.939255

```
In [11]: # Price > 0
df = df[df["Price"] > 0]

# Costs >= 0
cost_cols = ["Shipping costs", "Manufacturing costs", "Costs"]
for col in cost_cols:
    df = df[df[col] >= 0]

# Stock >= 0
df = df[df["Stock levels"] >= 0]

# Defect rates
df = df[(df["Defect rates"] >= 0) & (df["Defect rates"] <= 5)]
```

```
In [12]: # Convert columns to float
numeric_cols = ["Price", "Revenue generated", "Shipping costs",
                 "Manufacturing costs", "Costs", "Lead times", "Shipping times", ""]

for col in numeric_cols:
    df[col] = pd.to_numeric(df[col], errors="coerce")
```

```
In [13]: # Convert columns to int
int_cols = ["Number of products sold", "Stock levels", "Order quantities", "Prod
for col in int_cols:
    df[col] = pd.to_numeric(df[col], errors="coerce").astype("Int64")
```



```
In [14]: text_cols = ["Supplier name", "Location", "Transportation modes",
                  "Routes", "Shipping carriers", "Customer demographics"]

for col in text_cols:
    df[col] = df[col].astype(str).str.strip()
```



```
In [15]: print(df.head(5))
```

	Product type	SKU	Price	Availability	Number of products sold	\
0	haircare	SKU0	69.808006	55		802
1	skincare	SKU1	14.843523	95		736
2	haircare	SKU2	11.319683	34		8
3	skincare	SKU3	61.163343	68		83
4	skincare	SKU4	4.805496	26		871

	Revenue generated	Customer demographics	Stock levels	Lead times	\
0	8661.996792	Non-binary	58	7	
1	7460.900065	Female	53	30	
2	9577.749626	Unknown	1	10	
3	7766.836426	Non-binary	23	13	
4	2686.505152	Non-binary	5	3	

	Order quantities	...	Location	Lead time	Production volumes	\
0	96	...	Mumbai	29		215
1	37	...	Mumbai	23		517
2	88	...	Mumbai	12		971
3	59	...	Kolkata	24		937
4	56	...	Delhi	5		414

	Manufacturing lead time	Manufacturing costs	Inspection results	\
0	29	46.279879	Pending	
1	30	33.616769	Pending	
2	27	30.688019	Pending	
3	18	35.624741	Fail	
4	3	92.065161	Fail	

	Defect rates	Transportation modes	Routes	Costs
0	0.226410	Road	Route B	187.752075
1	4.854068	Road	Route B	503.065579
2	4.580593	Air	Route C	141.920282
3	4.746649	Rail	Route A	254.776159
4	3.145580	Air	Route A	923.440632

[5 rows x 24 columns]

```
In [16]: # two decimal digits
two_decimal_cols = ["Price", "Revenue generated", "Shipping costs",
                     "Manufacturing costs", "Costs"]

for col in two_decimal_cols:
    df[col] = df[col].round(2)
```

```
# one decimal number
df["Defect rates"] = df["Defect rates"].round(1)
```

```
In [17]: #text_cols = ["Supplier name", "Location", "Transportation modes",
#                  "Routes", "Shipping carriers", "Customer demographics"]
```

```
In [18]: for col in text_cols:
    print(f"\n--- Values in column: {col} ---")
    print(df[col].value_counts(dropna=False).head(10))
```

```
--- Values in column: Supplier name ---
```

```
Supplier name
Supplier 1    27
Supplier 2    22
Supplier 5    18
Supplier 4    18
Supplier 3    15
Name: count, dtype: int64
```

```
--- Values in column: Location ---
```

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

```
--- Values in column: Transportation modes ---
```

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

```
--- Values in column: Routes ---
```

```
Routes
Route A      43
Route B      37
Route C      20
Name: count, dtype: int64
```

```
--- Values in column: Shipping carriers ---
```

```
Shipping carriers
Carrier B    43
Carrier C    29
Carrier A    28
Name: count, dtype: int64
```

```
--- Values in column: Customer demographics ---
```

```
Customer demographics
Unknown       31
Female        25
Non-binary    23
Male          21
Name: count, dtype: int64
```

```
In [19]: txt_cols = ["Supplier name", "Location", "Transportation modes",
                  "Routes", "Shipping carriers", "Customer demographics", "Product typ
```

```
for col in txt_cols:  
    # Remove spaces and unify capitalize letters  
    df[col] = df[col].astype(str).str.strip().str.capitalize()
```

```
In [20]: import re
```

```
for col in text_cols:  
    df[col] = df[col].apply(lambda x: re.sub(r"[^a-zA-Z0-9\s]", "", x))
```

```
In [21]: df.columns = df.columns.str.title()
```

```
In [22]: for col in df.select_dtypes(include='object').columns:  
    df[col] = df[col].fillna("Unknown")
```

```
In [23]: print(df.head())
```

	Product Type	Sku	Price	Availability	Number Of Products Sold	\
0	Haircare	SKU0	69.81	55		802
1	Skincare	SKU1	14.84	95		736
2	Haircare	SKU2	11.32	34		8
3	Skincare	SKU3	61.16	68		83
4	Skincare	SKU4	4.81	26		871

	Revenue Generated	Customer Demographics	Stock Levels	Lead Times	\
0	8662.00	Nonbinary	58	7	
1	7460.90	Female	53	30	
2	9577.75	Unknown	1	10	
3	7766.84	Nonbinary	23	13	
4	2686.51	Nonbinary	5	3	

	Order Quantities	...	Location	Lead Time	Production Volumes	\
0	96	...	Mumbai	29	215	
1	37	...	Mumbai	23	517	
2	88	...	Mumbai	12	971	
3	59	...	Kolkata	24	937	
4	56	...	Delhi	5	414	

	Manufacturing Lead Time	Manufacturing Costs	Inspection Results	\
0	29	46.28	Pending	
1	30	33.62	Pending	
2	27	30.69	Pending	
3	18	35.62	Fail	
4	3	92.07	Fail	

	Defect Rates	Transportation Modes	Routes	Costs
0	0.2	Road	Route b	187.75
1	4.9	Road	Route b	503.07
2	4.6	Air	Route c	141.92
3	4.7	Rail	Route a	254.78
4	3.1	Air	Route a	923.44

[5 rows x 24 columns]

```
In [24]: df.columns
```

```
Out[24]: 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 [25]: df['Profit']=(df['Revenue Generated']-df['Costs']).round(2)
```

```
In [26]: df['Stock out Risk']=((df['Stock Levels']/df['Order Quantities'])*100).round(0)
df['Stock out Risk']=pd.to_numeric(df['Stock out Risk'],errors='coerce').astype(
```

```
In [27]: df['Warehousing Costs']=(df['Costs']-(df['Manufacturing Costs']+df['Shipping Cos
```

```
In [28]: df["Defective Units"] = df["Production Volumes"] * (df["Defect Rates"] / 100)
```

```
In [29]: df["Accepted Units"] = df["Production Volumes"] - df["Defective Units"]
```

```
In [30]: df[["Accepted Units", "Defective Units"]] = df[["Accepted Units", "Defective Uni
```

```
In [31]: df.rename(columns={"Lead Times":"Customer Lead Time",'Lead Time':'Supplier Lead
          'Customer Demographics':'Customer Gender','Availability':'Avai
          'Costs':'Total Cost'}
          , inplace=True)
```

```
In [32]: df.columns
```

```
Out[32]: Index(['Product Type', 'Sku', 'Price', 'Available for Sale',
       'Number Of Products Sold', 'Revenue Generated', 'Customer Gender',
       'Stock Levels', 'Customer Lead Time', 'Order Quantities',
       'Shipping Times', 'Shipping Carriers', 'Shipping Costs',
       'Supplier Name', 'Location', 'Supplier Lead Time', 'Production Volumes',
       'Manufacturing Lead Time', 'Manufacturing Costs', 'Inspection Results',
       'Defect Rates', 'Transportation Modes', 'Routes', 'Total Cost',
       'Profit', 'Stock out Risk', 'Warehousing Costs', 'Defective Units',
       'Accepted Units'],
      dtype='object')
```

```
In [33]: df
```

Out[33]:

	Product Type	Sku	Price	Available for Sale	Number Of Products Sold	Revenue Generated	Customer Gender	Stock Levels	Custom L T
0	Haircare	SKU0	69.81	55	802	8662.00	Nonbinary	58	
1	Skincare	SKU1	14.84	95	736	7460.90	Female	53	
2	Haircare	SKU2	11.32	34	8	9577.75	Unknown	1	
3	Skincare	SKU3	61.16	68	83	7766.84	Nonbinary	23	
4	Skincare	SKU4	4.81	26	871	2686.51	Nonbinary	5	
...
95	Haircare	SKU95	77.90	65	672	7386.36	Unknown	15	
96	Cosmetics	SKU96	24.42	29	324	7698.42	Nonbinary	67	
97	Haircare	SKU97	3.53	56	62	4370.92	Male	46	
98	Skincare	SKU98	19.75	43	913	8525.95	Female	53	
99	Haircare	SKU99	68.52	17	627	9185.19	Unknown	55	

100 rows × 29 columns



In [34]: `df.loc[:, "Production Volumes"]`

Out[34]:

```
0    215
1    517
2    971
3    937
4    414
...
95   450
96   648
97   535
98   581
99   921
Name: Production Volumes, Length: 100, dtype: Int64
```

In [38]: `# هنديك على الاعمهه دي`

```
num_cols = ['Price',
            'Revenue Generated',
            'Shipping Costs',
            'Manufacturing Costs',
            'Total Cost',
```

```

'Defect Rates' ]

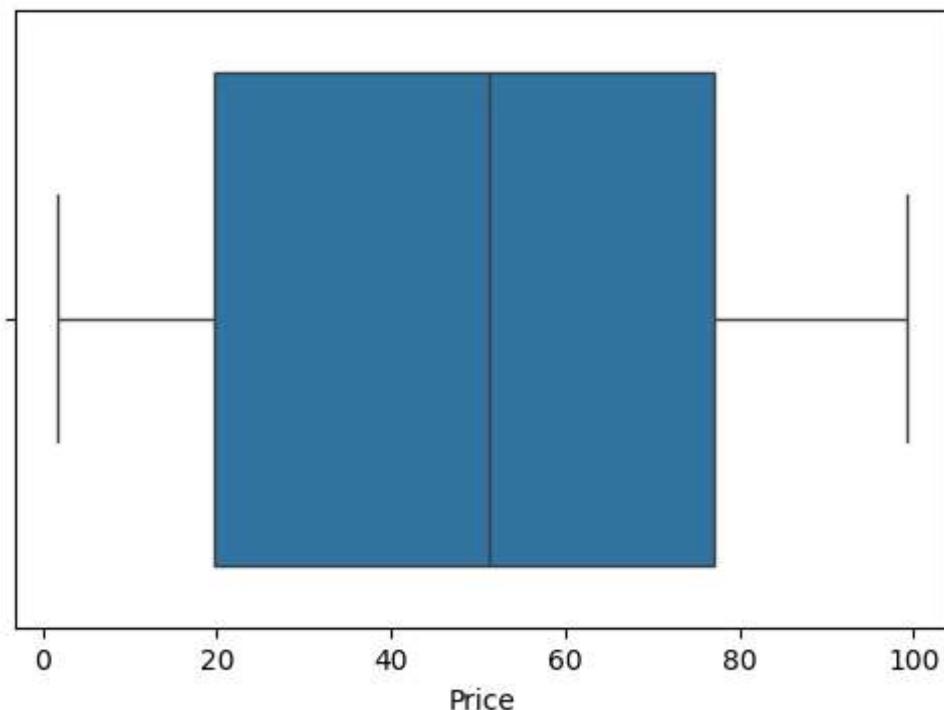
for col in num_cols:
    plt.figure(figsize=(6,4))
    sns.boxplot(x=df[col])
    plt.title(f"Outliers in {col}")
    plt.show()
# حساب IQR
    Q1 = df[col].quantile(0.25)
    Q3 = df[col].quantile(0.75)
    IQR = Q3 - Q1

    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR

    outliers = df[(df[col] < lower_bound) | (df[col] > upper_bound)]
    print(f"\nOutliers in {col}: {len(outliers)} rows")
    print(outliers[[col]].head(10))

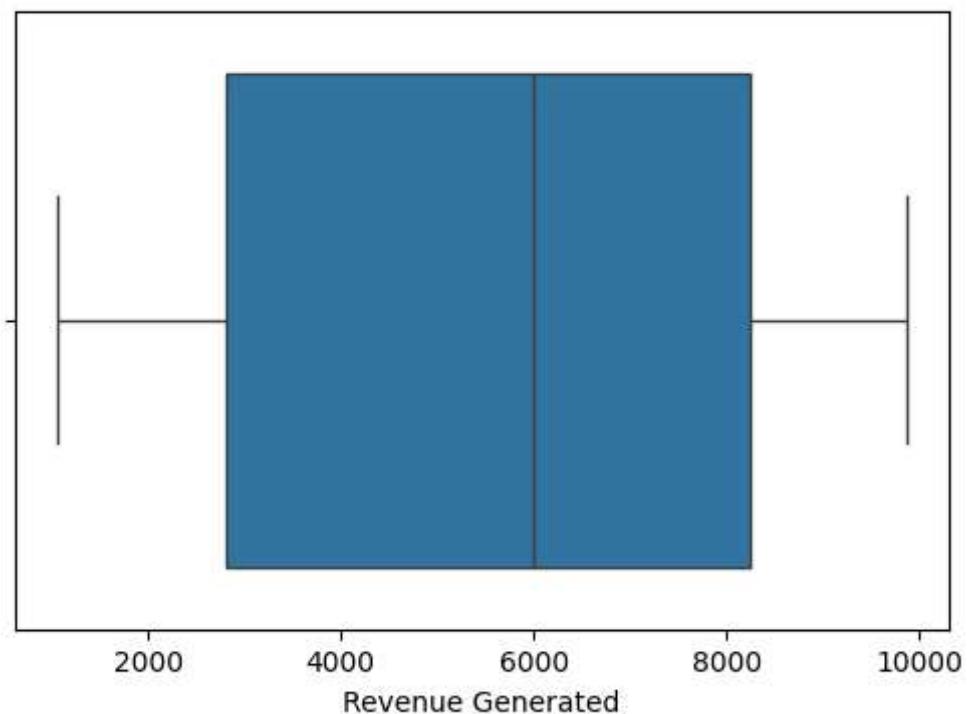
```

Outliers in Price



Outliers in Price: 0 rows
Empty DataFrame
Columns: [Price]
Index: []

Outliers in Revenue Generated



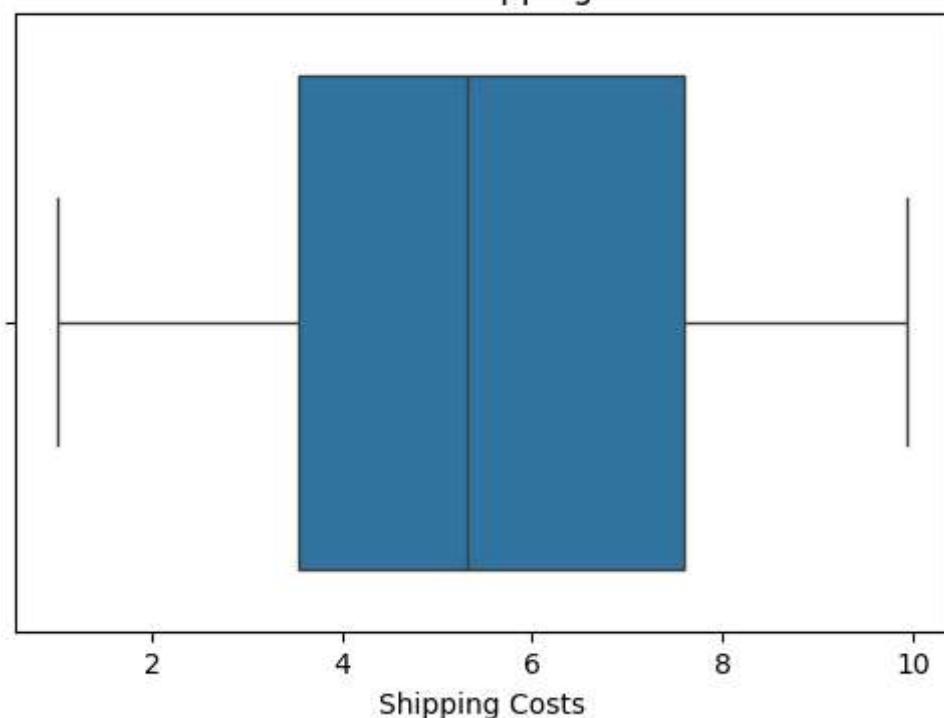
Outliers in Revenue Generated: 0 rows

Empty DataFrame

Columns: [Revenue Generated]

Index: []

Outliers in Shipping Costs



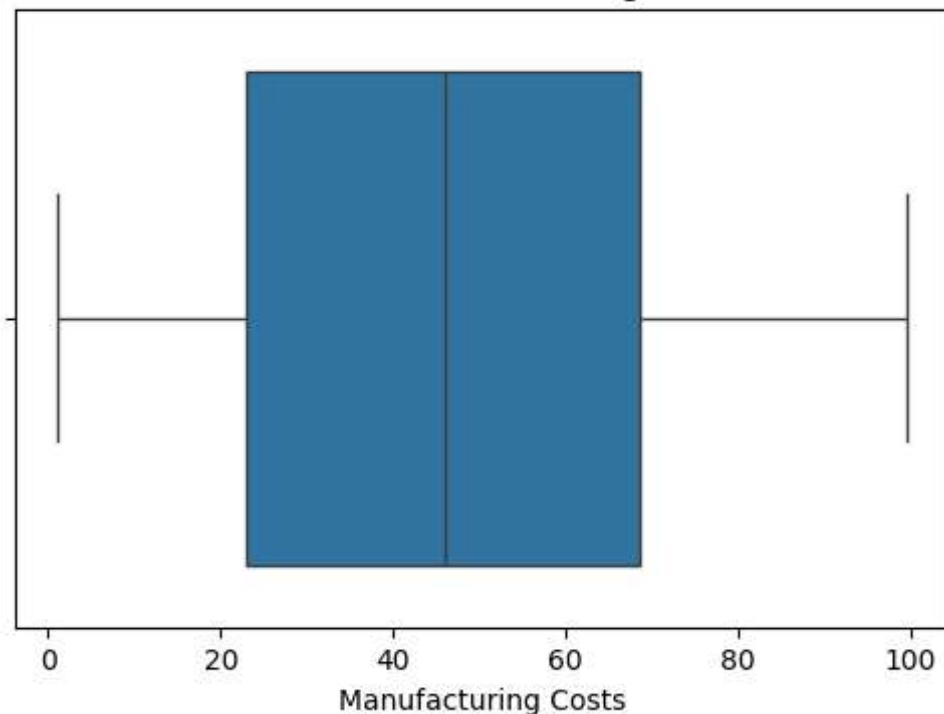
Outliers in Shipping Costs: 0 rows

Empty DataFrame

Columns: [Shipping Costs]

Index: []

Outliers in Manufacturing Costs



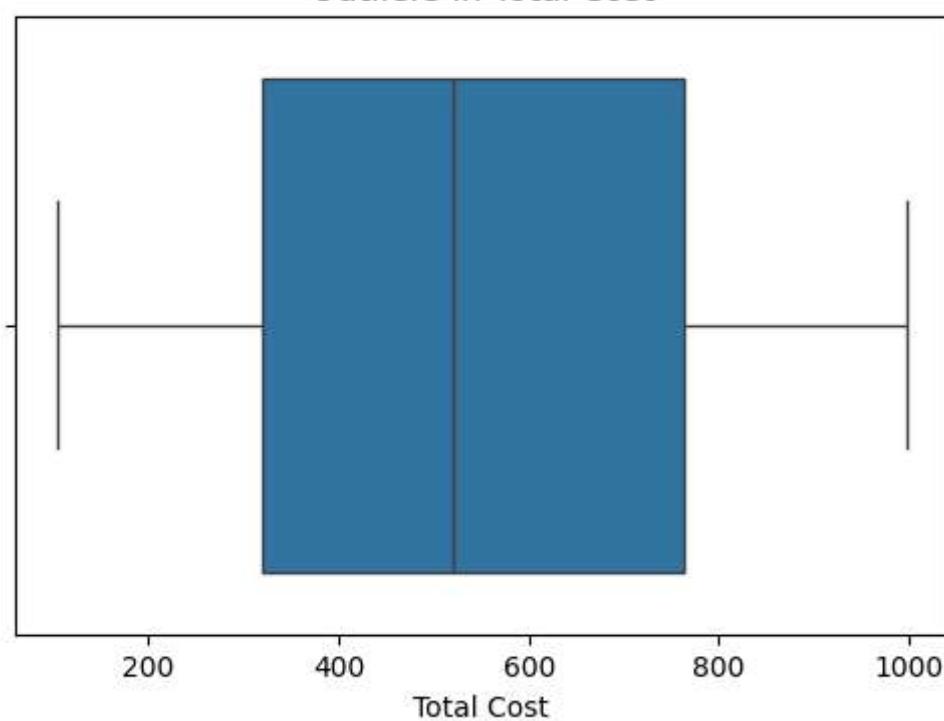
Outliers in Manufacturing Costs: 0 rows

Empty DataFrame

Columns: [Manufacturing Costs]

Index: []

Outliers in Total Cost



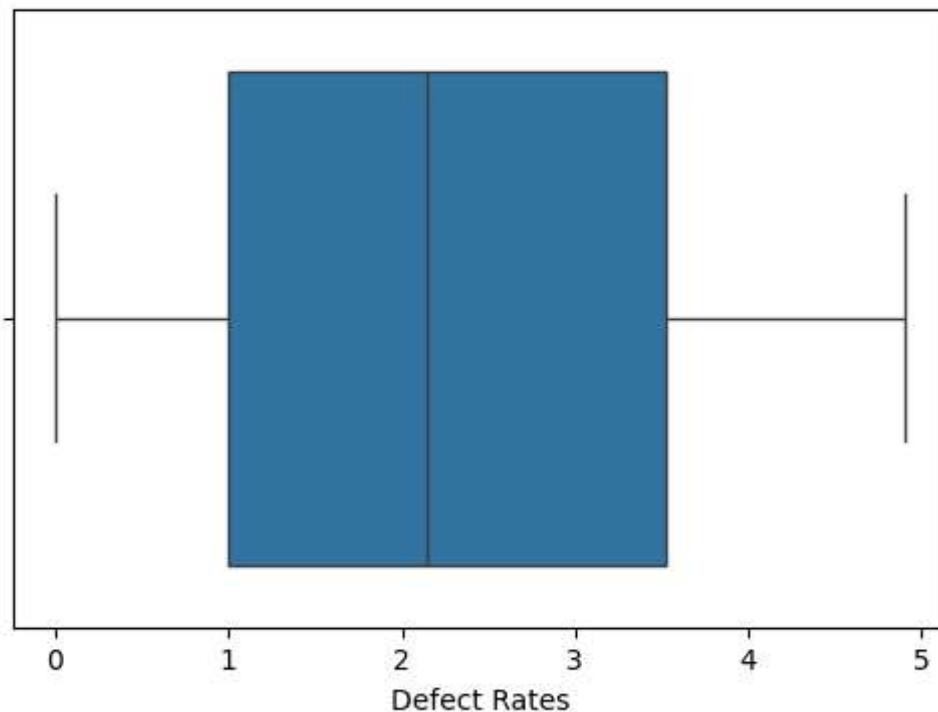
Outliers in Total Cost: 0 rows

Empty DataFrame

Columns: [Total Cost]

Index: []

Outliers in Defect Rates



```
Outliers in Defect Rates: 0 rows
```

```
Empty DataFrame
```

```
Columns: [Defect Rates]
```

```
Index: []
```

```
In [39]: df.columns
```

```
Out[39]: Index(['Product Type', 'Sku', 'Price', 'Available for Sale',
       'Number Of Products Sold', 'Revenue Generated', 'Customer Gender',
       'Stock Levels', 'Customer Lead Time', 'Order Quantities',
       'Shipping Times', 'Shipping Carriers', 'Shipping Costs',
       'Supplier Name', 'Location', 'Supplier Lead Time', 'Production Volumes',
       'Manufacturing Lead Time', 'Manufacturing Costs', 'Inspection Results',
       'Defect Rates', 'Transportation Modes', 'Routes', 'Total Cost',
       'Profit', 'Stock out Risk', 'Warehousing Costs', 'Defective Units',
       'Accepted Units'],
      dtype='object')
```

```
In [40]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 29 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   Available for Sale 100 non-null    int64  
 4   Number Of Products Sold 100 non-null    Int64  
 5   Revenue Generated 100 non-null    float64 
 6   Customer Gender   100 non-null    object  
 7   Stock Levels      100 non-null    Int64  
 8   Customer Lead Time 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  Supplier 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  Total Cost           100 non-null    float64 
 24  Profit               100 non-null    float64 
 25  Stock out Risk      100 non-null    int64  
 26  Warehousing Costs   100 non-null    float64 
 27  Defective Units     100 non-null    int64  
 28  Accepted Units       100 non-null    int64  
dtypes: Int64(4), float64(8), int64(8), object(9)
memory usage: 23.2+ KB
```

```
In [42]: df.to_csv('Cleaning Supply Chain Project 12.11.csv')
```

```
In [43]: df.to_excel('Cleaning Supply Chain Project 12.11 .xlsx')
```

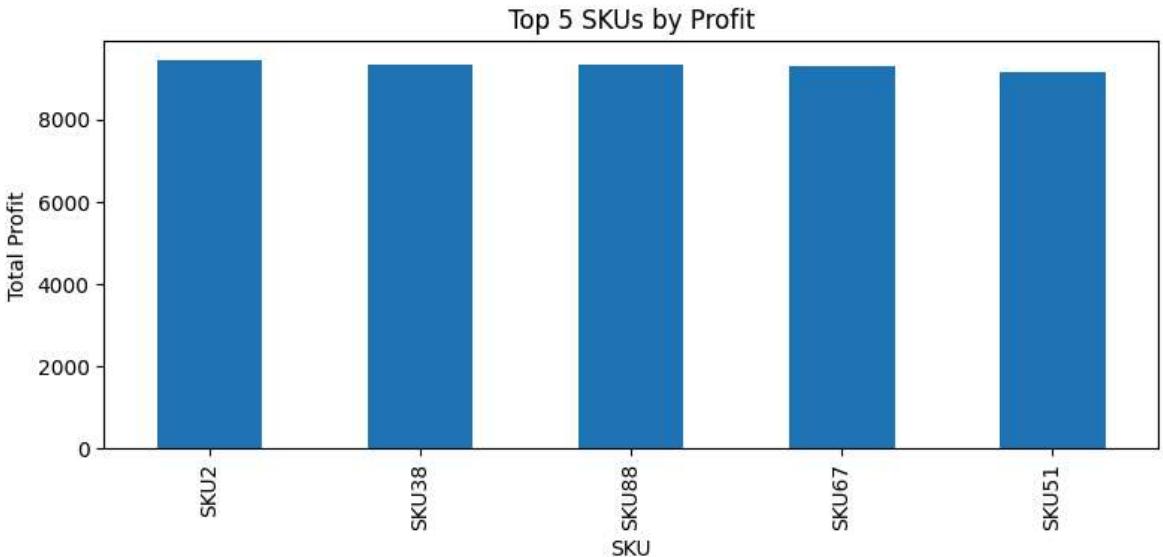
```
In [44]: #Product Performance
```

```
In [45]: #1.1 Top 5 SKUs by Profit
```

```
In [46]: top5_skus = df.groupby('Sku')['Profit'].sum().sort_values(ascending=False).head()
print(top5_skus)

# Visualization
top5_skus.plot(kind='bar', title='Top 5 SKUs by Profit', ylabel='Total Profit',
plt.tight_layout()
plt.show()
```

```
Sku
SKU2      9435.83
SKU38     9352.65
SKU88     9340.82
SKU67     9304.53
SKU51     9171.49
Name: Profit, dtype: float64
```



```
In [47]: # Top 3 Product Types by Order Quantities
```

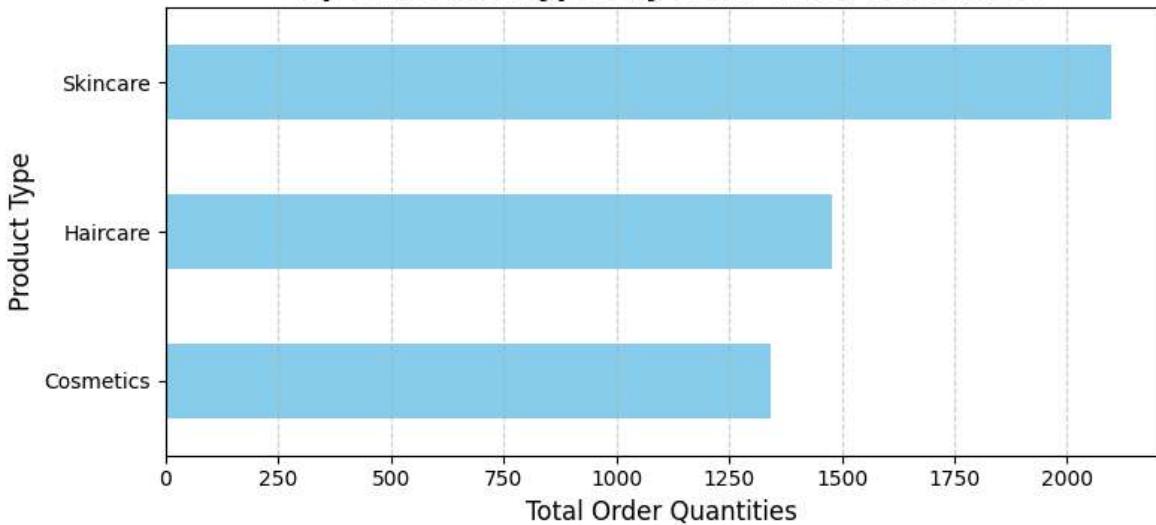
```
In [48]: top3_types_qty = (
    df.groupby('Product Type')['Order Quantities']
    .sum()
    .sort_values(ascending=False)
    .head(3)
)

print(top3_types_qty)

# Visualization
plt.figure(figsize=(8, 4))
top3_types_qty.plot(
    kind='barh',
    color='skyblue',
)
plt.title('Top 3 Product Types by Total Order Quantities', fontsize=14, weight='bold')
plt.xlabel('Total Order Quantities', fontsize=12)
plt.ylabel('Product Type', fontsize=12)
plt.gca().invert_yaxis() # يعكس أعلى نوع في الأعلى
plt.grid(axis='x', linestyle='--', alpha=0.6)
plt.tight_layout()
plt.show()
```

```
Product Type
Skincare      2099
Haircare      1480
Cosmetics     1343
Name: Order Quantities, dtype: Int64
```

Top 3 Product Types by Total Order Quantities

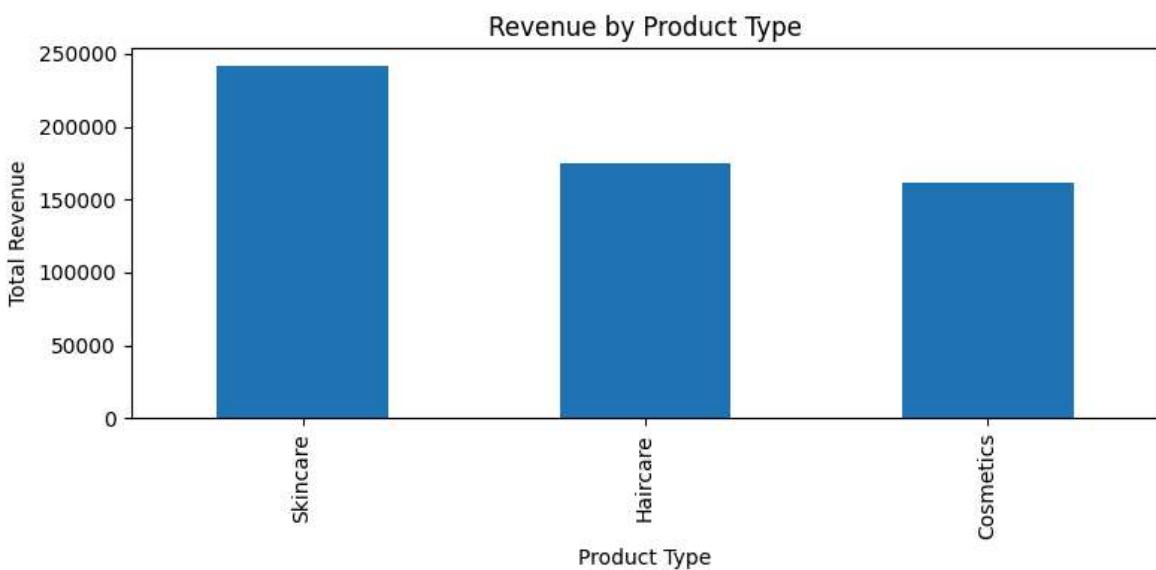


```
In [49]: #1.3 Revenue by Product Type
```

```
In [52]: revenue_by_type = df.groupby('Product Type')['Revenue Generated'].sum().sort_values()
print(revenue_by_type)

# Visualization
revenue_by_type.plot(kind='bar', title='Revenue by Product Type', ylabel='Total')
plt.tight_layout()
plt.show()
```

```
Product Type
Skincare      241628.17
Haircare      174455.42
Cosmetics     161521.27
Name: Revenue Generated, dtype: float64
```



```
In [53]: #1.4 Price vs Number Of Products Sold
```

```
In [54]: # Filter product types
types = ['Skincare', 'Haircare', 'Cosmetics']
data = df[df['Product Type'].isin(types)]

# Define consistent color palette
palette = sns.color_palette("Set2", n_colors=len(types))
```

```

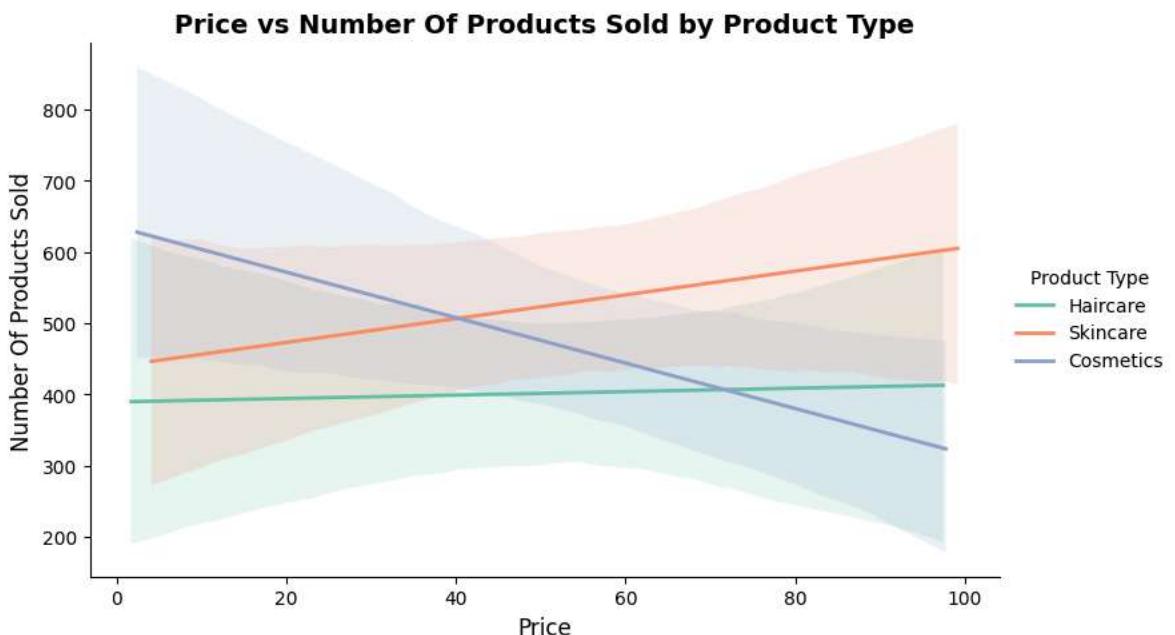
# Draw trendlines
g = sns.lmplot(
    x='Price',
    y='Number Of Products Sold',
    data=data,
    hue='Product Type',
    height=5,
    aspect=1.3,
    scatter=False,
    palette=palette,
    line_kws={'lw': 2}
)

# Title & labels
plt.title('Price vs Number Of Products Sold by Product Type', fontsize=14, weight='bold')
plt.xlabel('Price', fontsize=12)
plt.ylabel('Number Of Products Sold', fontsize=12)

# Move single legend outside the plot
g._legend.set_bbox_to_anchor((1.15, 0.5)) # يحركها لليمين برا الرسم
g._legend.set_title('Product Type') # العنوان
g._legend.set_frame_on(False) # بدون إطار
g._legend.set_title('Product Type')
for text in g._legend.texts:
    text.set_fontsize(10)

plt.tight_layout()
plt.show()

```



In [55]: `#Supplier Performance`

In [56]: `#2.1 Average Supplier Lead Time`

In [57]: `avg_lead_time = df.groupby('Supplier Name')['Supplier Lead Time'].mean().sort_values()
print(avg_lead_time)`

`# Visualization`

`avg_lead_time.plot(kind='bar', title='Average Supplier Lead Time', ylabel='Days')`

```

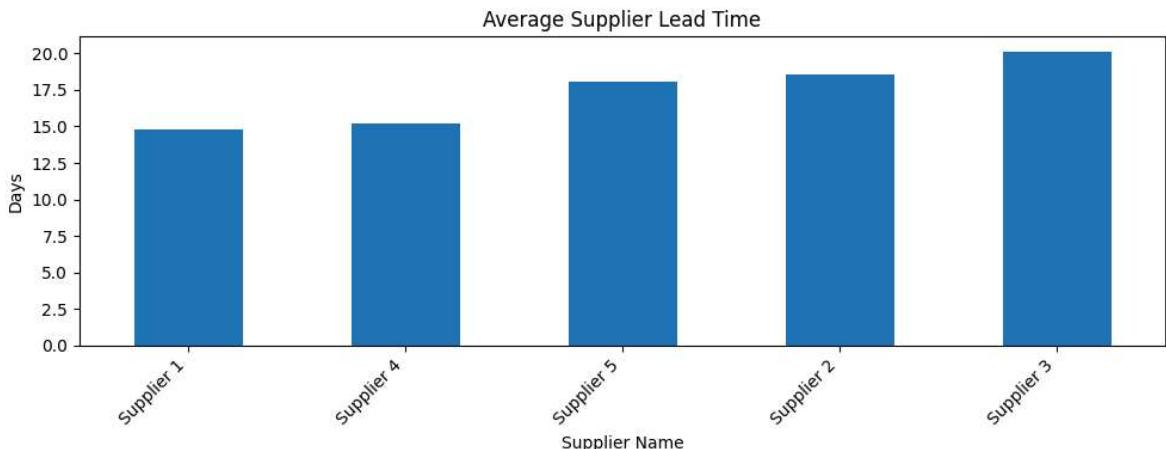
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()

```

Supplier Name

Supplier 1	14.777778
Supplier 4	15.222222
Supplier 5	18.055556
Supplier 2	18.545455
Supplier 3	20.133333

Name: Supplier Lead Time, dtype: float64



In [58]: # 2.2 Defect Rate vs Supplier Lead Time per Supplier

```

In [59]: defect_vs_lead = df.groupby('Supplier Name')[['Defect Rates', 'Supplier Lead Time']]

# ترتيب الموردين حسب معدل العيوب
defect_vs_lead = defect_vs_lead.sort_values('Defect Rates', ascending=False)

# إنشاء الشكل
fig, ax1 = plt.subplots(figsize=(10,5))

# العمود الأول: Defect Rate (Bar)
sns.barplot(
    data=defect_vs_lead,
    x='Supplier Name',
    y='Defect Rates',
    color='#2E86AB',
    ax=ax1
)
ax1.set_ylabel('Average Defect Rate', color="#2E86AB")
ax1.tick_params(axis='y', labelcolor="#2E86AB")
ax1.set_xticklabels(defect_vs_lead['Supplier Name'], rotation=45, ha='right')

# العمود الثاني: Supplier Lead Time (Line)
ax2 = ax1.twinx()
sns.lineplot(
    data=defect_vs_lead,
    x='Supplier Name',
    y='Supplier Lead Time',
    color="#E74C3C",
    marker='o',
    linewidth=2,
    ax=ax2
)
ax2.set_ylabel('Average Supplier Lead Time', color="#E74C3C")

```

```

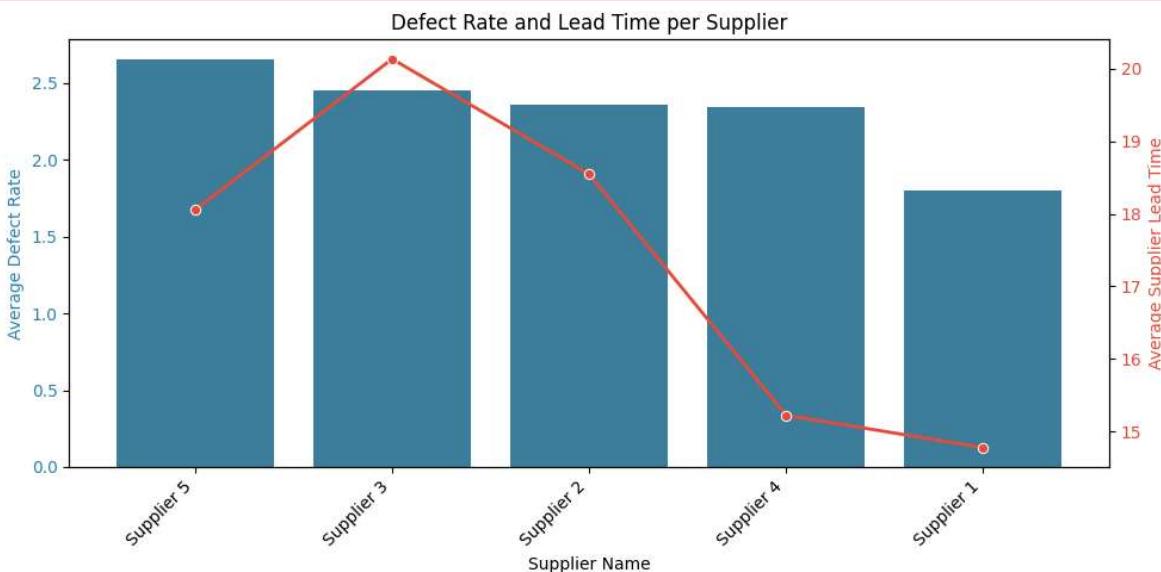
ax2.tick_params(axis='y', labelcolor="#E74C3C")

plt.title('Defect Rate and Lead Time per Supplier')
plt.tight_layout()
plt.show()
print(defect_vs_lead)

```

C:\Users\dell\AppData\Local\Temp\ipykernel_10008\2377119272.py:19: UserWarning: set_ticklabels() should only be used with a fixed number of ticks, i.e. after set_ticks() or using a FixedLocator.

```
    ax1.set_xticklabels(defect_vs_lead['Supplier Name'], rotation=45, ha='right')
```



	Supplier Name	Defect Rates	Supplier Lead Time
4	Supplier 5	2.650000	18.055556
2	Supplier 3	2.453333	20.133333
1	Supplier 2	2.359091	18.545455
3	Supplier 4	2.344444	15.222222
0	Supplier 1	1.800000	14.777778

In [60]: #2.4 Supplier Reliability (Accepted Units / Total Units) *****Panding*****

```

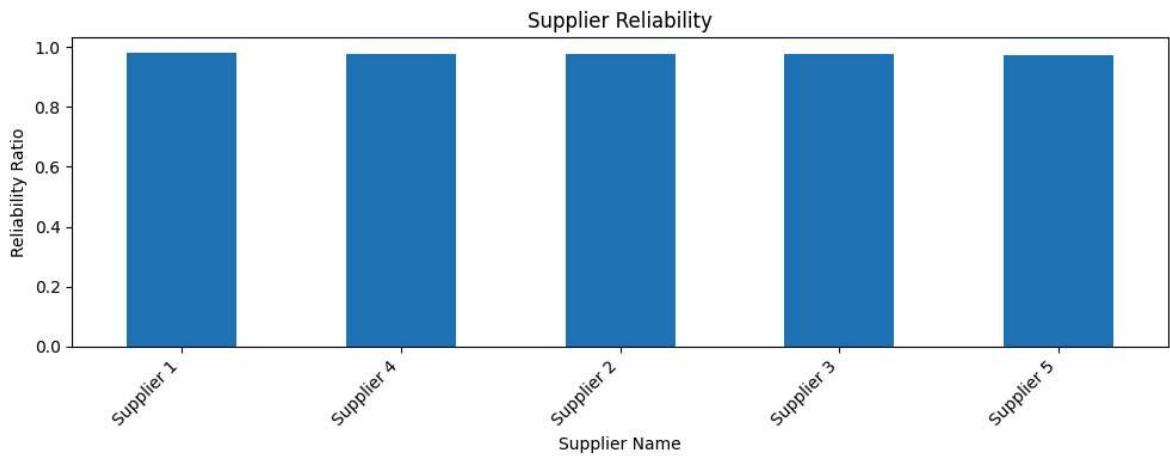
In [61]: df['Total Units'] = df['Accepted Units'] + df['Defective Units']
df['Supplier Reliability'] = df['Accepted Units'] / df['Total Units']
reliability = df.groupby('Supplier Name')['Supplier Reliability'].mean().sort_values()
print(reliability)

# Visualization
reliability.plot(kind='bar', title='Supplier Reliability', ylabel='Reliability R'
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()

```

Supplier Name	Supplier Reliability
Supplier 1	0.981969
Supplier 4	0.976763
Supplier 2	0.976175
Supplier 3	0.975775
Supplier 5	0.973359

Name: Supplier Reliability, dtype: float64



```
In [62]: #Logistics Analysis
```

```
In [63]: #3.1 Avg Shipping Time per Carrier
```

```
In [64]: avg_shipping_time = df.groupby('Shipping Carriers')['Shipping Times'].mean().sort_values()
print(avg_shipping_time)

# Visualization
# Calculate average shipping time per carrier
avg_shipping_time = df.groupby('Shipping Carriers')['Shipping Times'].mean().sort_values()

# Plot
plt.figure(figsize=(8,4))
sns.barplot(
    x=avg_shipping_time.values,
    y=avg_shipping_time.index,
    palette='crest'
)

# Add value Labels
for i, v in enumerate(avg_shipping_time.values):
    plt.text(v + 0.1, i, f"{v:.1f} days", va='center', fontsize=9, weight='bold')

plt.title('Average Shipping Time per Carrier', fontsize=14, weight='bold')
plt.xlabel('Average Shipping Time (Days)', fontsize=12)
plt.ylabel('Shipping Carrier', fontsize=12)
plt.tight_layout()
plt.show()
```

```
Shipping Carriers
Carrier b      5.302326
Carrier c      6.034483
Carrier a      6.142857
Name: Shipping Times, dtype: float64
```

```
C:\Users\dell\AppData\Local\Temp\ipykernel_10008\220646903.py:10: FutureWarning:
```

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same effect.

```
sns.barplot(
```



```
In [65]: #3.2 Avg Shipping Cost by Transport Mode
```

```
In [66]: avg_shipping_cost = df.groupby('Transportation Modes')[['Shipping Costs']].mean()
print(avg_shipping_cost)

# Visualization
plt.figure(figsize=(8,4))
sns.barplot(
    x=avg_shipping_cost.values,
    y=avg_shipping_cost.index,
    palette='crest'
)

# Add value labels
for i, v in enumerate(avg_shipping_cost.values):
    plt.text(v + 0.05, i, f"${v:.2f}", va='center', fontsize=9, fontweight='bold')

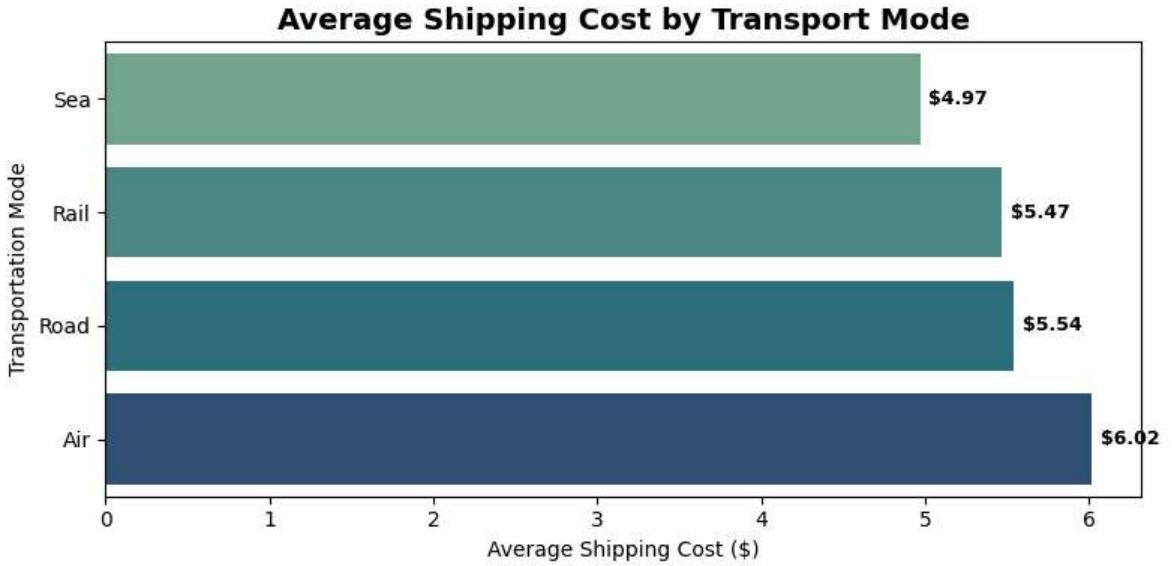
plt.title('Average Shipping Cost by Transport Mode', fontsize=14, fontweight='bold')
plt.xlabel('Average Shipping Cost ($)')
plt.ylabel('Transportation Mode')
plt.tight_layout()
plt.show()
```

```
Transportation Modes
Sea      4.970000
Rail     5.469286
Road     5.542414
Air      6.017692
Name: Shipping Costs, dtype: float64
```

```
C:\Users\dell\AppData\Local\Temp\ipykernel_10008\1227519717.py:6: FutureWarning:
```

```
Passing `palette` without assigning `hue` is deprecated and will be removed in v
0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same effe
ct.
```

```
sns.barplot(
```



```
In [67]: #3.3 Route Efficiency (Time / Cost)
```

```
In [68]: df['Route Efficiency'] = df['Shipping Times'] / (df['Shipping Costs'] + 1e-6)

route_summary = df.groupby('Routes').agg({
    'Shipping Times': 'mean',
    'Shipping Costs': 'mean',
    'Route Efficiency': 'mean'
}).round(2)

route_summary = route_summary.sort_values('Route Efficiency')

route_summary = route_summary.rename(columns={
    'Shipping Times': 'Avg Shipping Time (Days)',
    'Shipping Costs': 'Avg Shipping Cost ($)',
    'Route Efficiency': 'Time per Cost Unit'
})

print("\n📊 Route Efficiency Summary:\n")
print(route_summary)

# Visualization
plt.figure(figsize=(10,5))
sns.barplot(
    x=route_summary.index,
    y='Time per Cost Unit',
    data=route_summary,
    palette="crest"
)
plt.title('Route Efficiency (Time / Cost)', fontsize=14, weight='bold')
plt.xlabel('Route', fontsize=12)
plt.ylabel('Days per Cost Unit', fontsize=12)
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()
```

📊 Route Efficiency Summary:

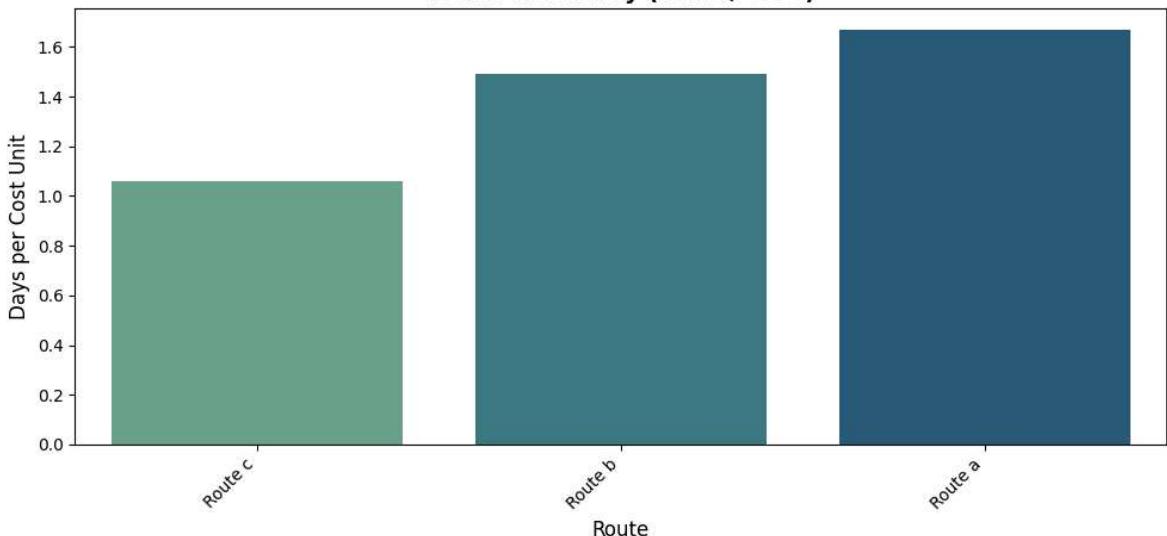
Routes	Avg Shipping Time (Days)	Avg Shipping Cost (\$)	Time per Cost Unit
Route c	5.25	5.90	1.06
Route b	5.70	5.55	1.49
Route a	6.02	5.38	1.67

C:\Users\dell\AppData\Local\Temp\ipykernel_10008\4067178423.py:22: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v 0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

```
sns.barplot(
```

Route Efficiency (Time / Cost)



In [69]: #3.4 Carrier Performance (Time vs Cost)

```
# حساب المتوسط لكل Carrier
carrier_summary = df.groupby('Shipping Carriers').agg({
    'Shipping Times': 'mean',
    'Shipping Costs': 'mean'
}).round(2).reset_index()

# Efficiency Score (كل ما يقل → Eff++)
carrier_summary['Efficiency Score'] = (carrier_summary['Shipping Costs'] / carrier_summary['Shipping Times'])
carrier_summary = carrier_summary.sort_values('Efficiency Score')

# عرض الجدول
print("📊 Average Carrier Performance (Time vs Cost):\n")
print(carrier_summary)

# Scatter Plot + Trendline + Labels
plt.figure(figsize=(9,6))
sns.regrplot(
    data=carrier_summary,
    x='Shipping Times',
    y='Shipping Costs',
    scatter=False,
    ci=None,
    line_kws={'color':'gray', 'lw':1, 'ls': '--'}
)
```

```

sns.scatterplot(
    data=carrier_summary,
    x='Shipping Times',
    y='Shipping Costs',
    hue='Shipping Carriers',
    s=150,
    palette='Set2'
)

for i in range(len(carrier_summary)):
    plt.text(
        carrier_summary['Shipping Times'][i] + 0.05,
        carrier_summary['Shipping Costs'][i],
        carrier_summary['Shipping Carriers'][i],
        fontsize=10,
        weight='bold'
    )

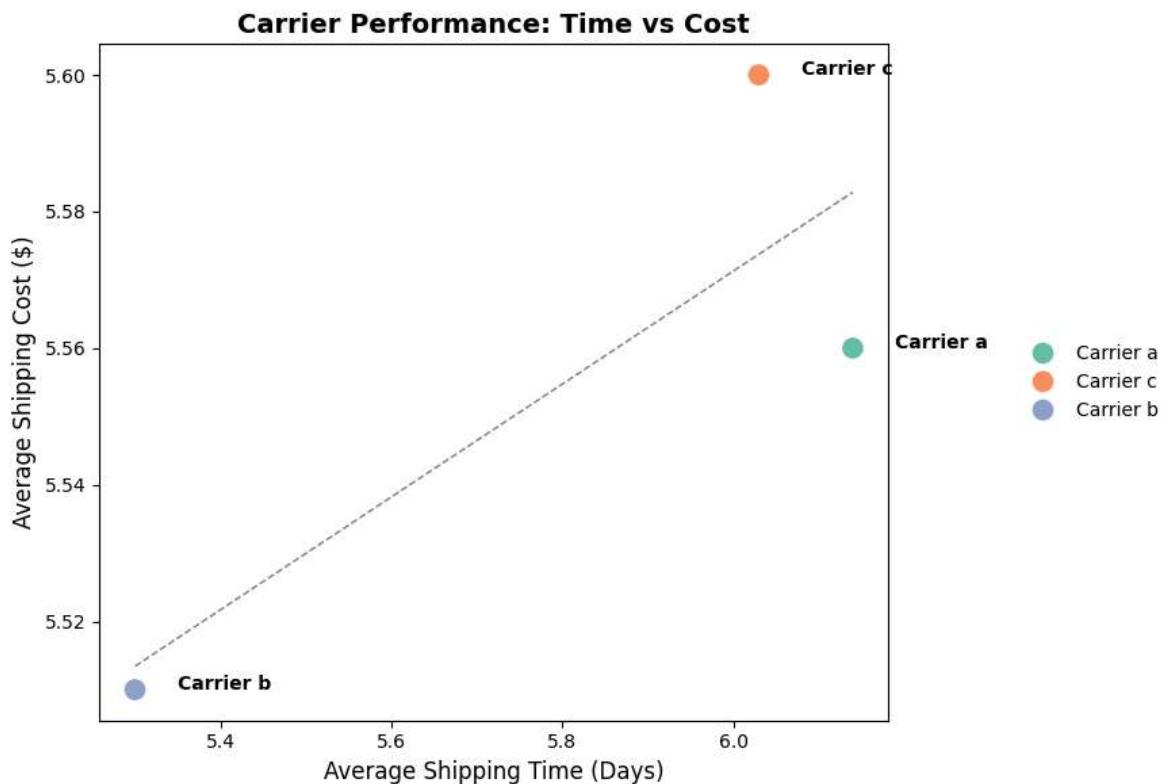
plt.title('Carrier Performance: Time vs Cost', fontsize=14, weight='bold')
plt.xlabel('Average Shipping Time (Days)', fontsize=12)
plt.ylabel('Average Shipping Cost ($)', fontsize=12)
plt.legend(bbox_to_anchor=(1.15, 0.5), loc='center left', frameon=False)
plt.tight_layout()
plt.show()

# Heatmap
plt.figure(figsize=(6,4))
sns.heatmap(
    carrier_summary[['Shipping Times', 'Shipping Costs', 'Efficiency Score']].se
    annot=True,
    cmap='YlGnBu',
    fmt='.2f'
)
plt.title('Carrier Efficiency Overview', fontsize=13, weight='bold')
plt.ylabel('Shipping Carriers')
plt.tight_layout()
plt.show()

```

 Average Carrier Performance (Time vs Cost):

	Shipping Carriers	Shipping Times	Shipping Costs	Efficiency Score
0	Carrier a	6.14	5.56	0.91
2	Carrier c	6.03	5.60	0.93
1	Carrier b	5.30	5.51	1.04



```
In [71]: #Manufacturing Analysis
```

```
In [72]: #4.1 Manufacturing Lead Time vs Cost vs Production Volumes
```

```
In [73]: # إلى فئات محددة
bins = [0, 5, 10, 15, 20, 25, 30]
labels = ['1-5', '6-10', '11-15', '16-20', '21-25', '26-30']
df['Lead Time Group'] = pd.cut(df['Manufacturing Lead Time'], bins=bins, labels=labels)

# حساب المتوسطات لكل نوع منتج
summary = df.groupby(['Product Type', 'Lead Time Group']).agg({
    'Manufacturing Lead Time': 'mean',
```

```

        'Manufacturing Costs': 'mean',
        'Production Volumes': 'mean'
    }).round(2).reset_index()

summary['Production Volumes'] = summary['Production Volumes'].round().astype(int)

print("📊 Manufacturing Summary by Product Type and Lead Time Group:\n")
print(summary)

# Visualization 1: Bubble Chart (كل Product Type)
plt.figure(figsize=(10,6))
sns.scatterplot(
    data=summary,
    x='Manufacturing Lead Time',
    y='Manufacturing Costs',
    size='Production Volumes',
    sizes=(100, 1000),
    hue='Lead Time Group',
    style='Product Type',
    palette='viridis',
    alpha=0.8
)

plt.title('Manufacturing Lead Time vs Cost vs Production Volumes (by Product Type)')
plt.xlabel('Avg Manufacturing Lead Time (Days)', fontsize=12)
plt.ylabel('Avg Manufacturing Costs ($)', fontsize=12)
plt.legend(bbox_to_anchor=(1.2, 0.5), loc='center left', frameon=False)
plt.tight_layout()
plt.show()

# Visualization 2: Line Chart توضح الاتجاه العام لكل نوع منتج
plt.figure(figsize=(10,6))
sns.lineplot(
    data=summary,
    x='Lead Time Group',
    y='Manufacturing Costs',
    hue='Product Type',
    marker='o'
)
plt.title('Trend: Lead Time vs Manufacturing Cost (by Product Type)', fontsize=14)
plt.xlabel('Lead Time Group', fontsize=12)
plt.ylabel('Avg Manufacturing Cost ($)', fontsize=12)
plt.tight_layout()
plt.show()

# Correlation Analysis
corr_time_cost = df['Manufacturing Lead Time'].corr(df['Manufacturing Costs'])
corr_time_volume = df['Manufacturing Lead Time'].corr(df['Production Volumes'])
corr_cost_volume = df['Manufacturing Costs'].corr(df['Production Volumes'])

print("\n📈 Correlation Analysis (Overall):")
print(f"Lead Time ↔ Cost: {corr_time_cost:.3f}")
print(f"Lead Time ↔ Production Volume: {corr_time_volume:.3f}")
print(f"Cost ↔ Production Volume: {corr_cost_volume:.3f}")

```

C:\Users\dell\AppData\Local\Temp\ipykernel_10008\2375021959.py:7: FutureWarning:
The default of observed=False is deprecated and will be changed to True in a future version of pandas. Pass observed=False to retain current behavior or observed=True to adopt the future default and silence this warning.

```
summary = df.groupby(['Product Type', 'Lead Time Group']).agg({
```

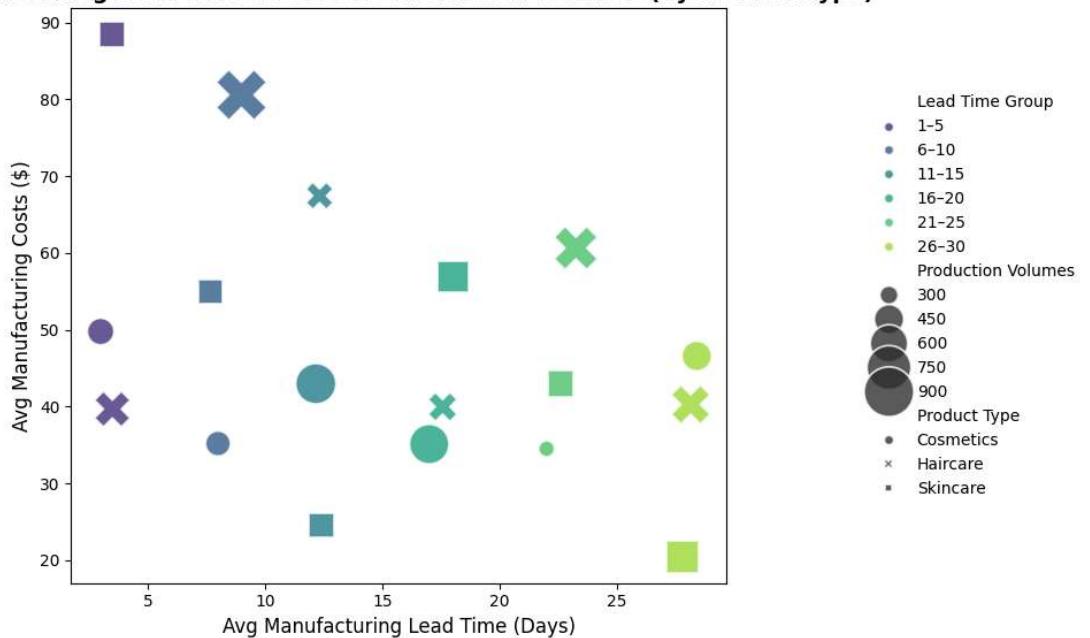
 Manufacturing Summary by Product Type and Lead Time Group:

	Product Type	Lead Time Group	Manufacturing Lead Time	Manufacturing Costs \
0	Cosmetics	1-5	3.00	49.78
1	Cosmetics	6-10	8.00	35.20
2	Cosmetics	11-15	12.17	42.99
3	Cosmetics	16-20	17.00	35.12
4	Cosmetics	21-25	22.00	34.52
5	Cosmetics	26-30	28.40	46.59
6	Haircare	1-5	3.50	39.69
7	Haircare	6-10	9.00	80.58
8	Haircare	11-15	12.33	67.45
9	Haircare	16-20	17.57	39.98
10	Haircare	21-25	23.25	60.65
11	Haircare	26-30	28.14	40.28
12	Skincare	1-5	3.50	88.44
13	Skincare	6-10	7.67	54.99
14	Skincare	11-15	12.40	24.57
15	Skincare	16-20	18.00	56.86
16	Skincare	21-25	22.60	42.99
17	Skincare	26-30	27.80	20.44

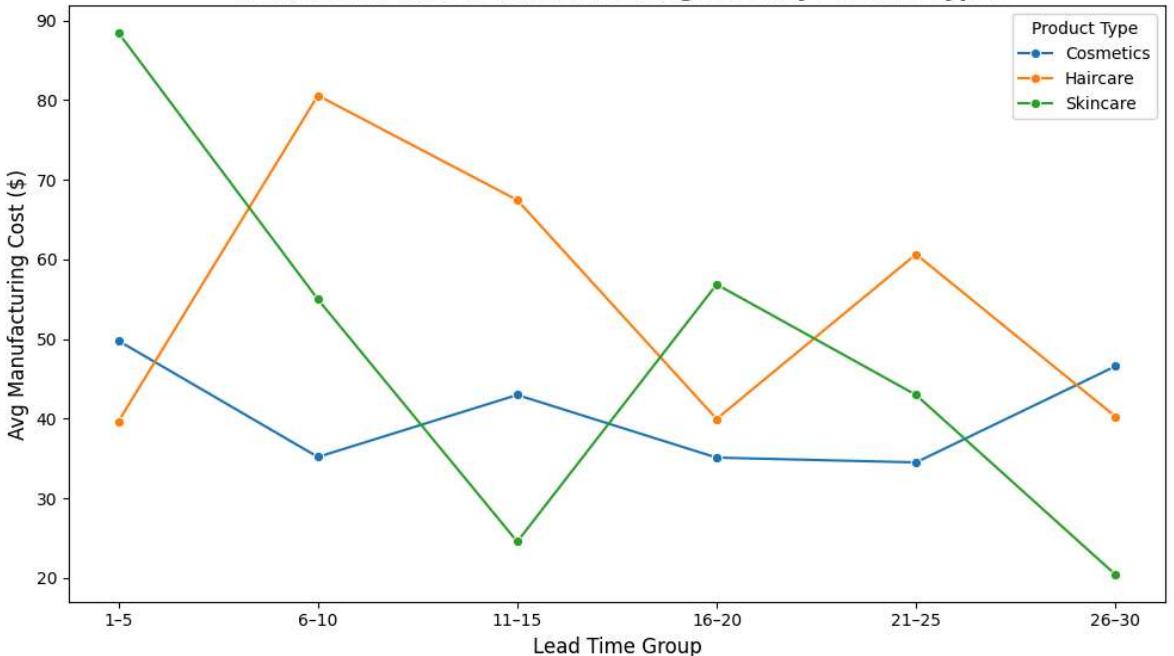
Production Volumes

0	406
1	382
2	662
3	644
4	278
5	454
6	562
7	919
8	414
9	441
10	728
11	627
12	560
13	537
14	549
15	744
16	593
17	778

Manufacturing Lead Time vs Cost vs Production Volumes (by Product Type)



Trend: Lead Time vs Manufacturing Cost (by Product Type)



📈 Correlation Analysis (Overall):

Lead Time ↔ Cost: -0.158

Lead Time ↔ Production Volume: 0.184

Cost ↔ Production Volume: 0.051

In [74]: #4.2 Defect Rate vs Production Volume

```
# حساب المتوسط لكل جروب
summary = df.groupby('Production Volume Group').agg({
    'Defect Rates': 'mean'
}).round(2).reset_index()

print("📊 Summary Table:\n", summary)
```

```

# Visualization:
fig, ax1 = plt.subplots(figsize=(9,5))

# العمود الأساسي (Bar Chart)
sns.barplot(
    data=summary,
    x='Production Volume Group',
    y='Defect Rates',
    color='lightblue',
    ax=ax1
)

# الخطوط (Line)
sns.lineplot(
    data=summary,
    x='Production Volume Group',
    y='Defect Rates',
    marker='o',
    linewidth=2.5,
    color='darkblue',
    ax=ax1
)

ax1.set_title('Average Defect Rate by Production Volume Group', fontsize=14, weight='bold')
ax1.set_xlabel('Production Volume Group', fontsize=12)
ax1.set_ylabel('Average Defect Rate (%)', fontsize=12)
ax1.grid(alpha=0.3)
plt.tight_layout()
plt.show()

# Correlation
corr = df['Production Volumes'].corr(df['Defect Rates'])
print(f"\n🔍 Correlation between Production Volumes and Defect Rates: {corr:.3f}")

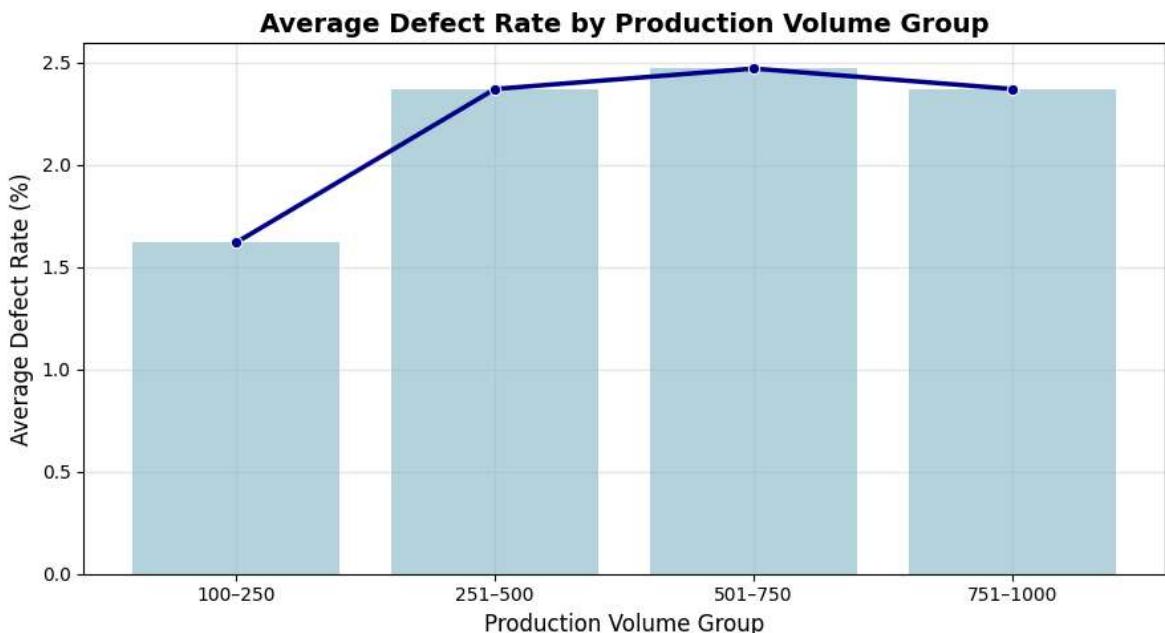
```

📊 Summary Table:

	Production Volume Group	Defect Rates
0	100–250	1.62
1	251–500	2.37
2	501–750	2.47
3	751–1000	2.37

C:\Users\dell\AppData\Local\Temp\ipykernel_10008\3399302648.py:7: FutureWarning:
The default of observed=False is deprecated and will be changed to True in a future version of pandas. Pass observed=False to retain current behavior or observed=True to adopt the future default and silence this warning.

```
summary = df.groupby('Production Volume Group').agg({
```



🔍 Correlation between Production Volumes and Defect Rates: 0.119

In [76]: `#Inventory & Warehouse Analysis`

In [77]: `#5.1 Stock Levels vs Order Quantities`

```
category_corr = df.groupby('Product Type').apply(
    lambda x: x['Stock Levels'].corr(x['Order Quantities']))
).reset_index(name='Correlation')

category_corr = category_corr.sort_values('Correlation', ascending=False)

plt.figure(figsize=(6,3))
sns.heatmap(
    category_corr.pivot_table(values='Correlation', index='Product Type'),
    annot=True, fmt='.2f', cmap='RdBu', cbar_kws={'label': 'Correlation Value'}
)
plt.title('Correlation between Stock Levels and Order Quantities by Category', f
plt.xlabel('')
plt.ylabel('Product Type')
plt.tight_layout()
plt.show()
```

C:\Users\dell\AppData\Local\Temp\ipykernel_10008\2095759728.py:1: FutureWarning:
DataFrameGroupBy.apply operated on the grouping columns. This behavior is deprecated, and in a future version of pandas the grouping columns will be excluded from the operation. Either pass `include_groups=False` to exclude the groupings or explicitly select the grouping columns after groupby to silence this warning.

```
category_corr = df.groupby('Product Type').apply(
```



```
In [79]: #5.2 Stockout Risk Index
```

```
In [80]: df['Stockout Risk Index'] = df['Order Quantities'] * (df['Stock Levels'] / 100 + 1)

# حساب المخاطر من نوع حساب المخاطر
stockout_risk = (
    df.groupby('Product Type')['Stockout Risk Index']
    .mean()
    .sort_values(ascending=False)
)
print(stockout_risk)

fig, ax = plt.subplots(figsize=(10, 4))

colors = plt.cm.viridis(
    (stockout_risk - stockout_risk.min()) / (stockout_risk.max() - stockout_risk)
)

bars = ax.bar(stockout_risk.index, stockout_risk.values, color=colors)

ax.set_title('Stockout Risk by Product Type', fontsize=12)
ax.set_xlabel('Product Type', fontsize=10)
ax.set_ylabel('Risk Index', fontsize=10)
ax.set_xticklabels(stockout_risk.index, rotation=45, ha='right')

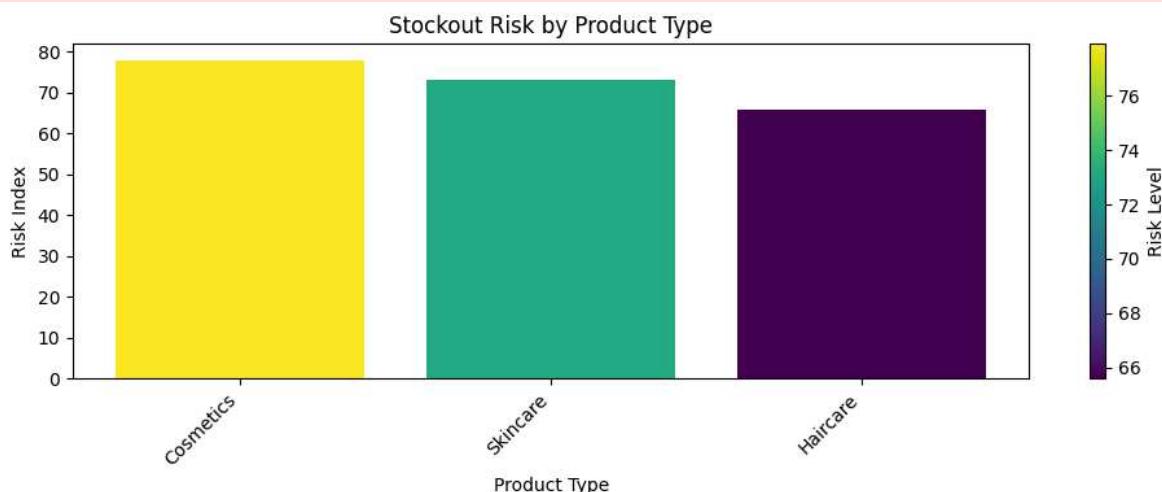
sm = plt.cm.ScalarMappable(
    cmap='viridis',
    norm=plt.Normalize(vmin=stockout_risk.min(), vmax=stockout_risk.max())
)
sm.set_array([])

fig.colorbar(sm, ax=ax, label='Risk Level')

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

```
Product Type
Cosmetics      77.908077
Skincare       73.11975
Haircare        65.592059
Name: Stockout Risk Index, dtype: Float64
```

```
C:\Users\dell\AppData\Local\Temp\ipykernel_10008\1169210866.py:22: UserWarning: set_xticklabels() should only be used with a fixed number of ticks, i.e. after set_ticks() or using a FixedLocator.  
    ax.set_xticklabels(stockout_risk.index, rotation=45, ha='right')
```



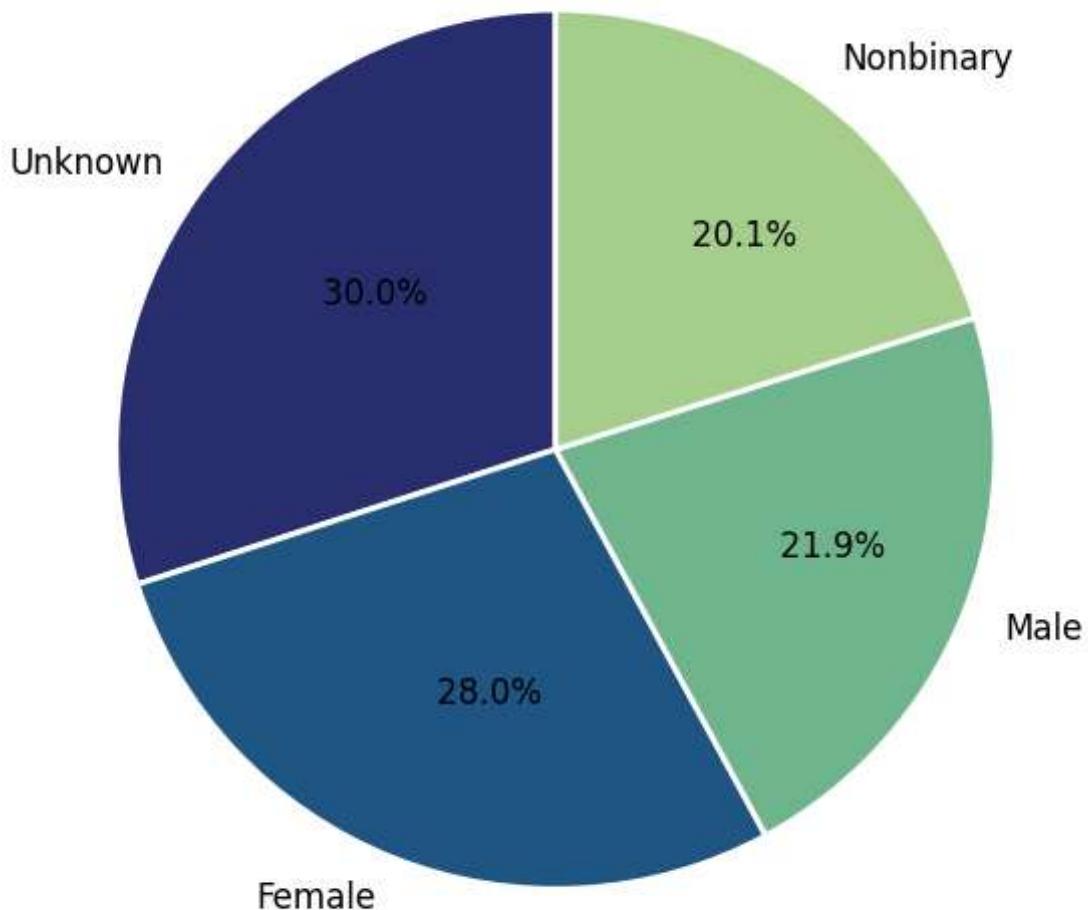
```
In [81]: #Customer Insights
```

```
In [82]: #6.1 Sales by Customer Gender
```

```
In [84]: sales_by_gender = df.groupby('Customer Gender')[ 'Revenue Generated'].sum().sort_  
print(sales_by_gender)  
  
cmap = plt.get_cmap('crest')  
  
normalized_values = (sales_by_gender - sales_by_gender.min()) / (sales_by_gender  
colors = cmap(normalized_values)  
  
sales_by_gender.plot(  
    kind='pie',  
    title='Sales by Customer Gender',  
    autopct='%10.1f%%',  
    figsize=(6, 6),  
    colors=colors,  
    startangle=90,  
    textprops={'fontsize': 12, 'color': 'black'},  
    wedgeprops={'edgecolor': 'white', 'linewidth': 2}  
)  
plt.ylabel('')  
plt.tight_layout()  
plt.show()
```

```
Customer Gender  
Unknown      173090.14  
Female       161514.49  
Male         126634.42  
Nonbinary    116365.81  
Name: Revenue Generated, dtype: float64
```

Sales by Customer Gender



```
In [85]: #Efficiency & Risk Analysis
```

```
In [86]: #7.1 Stockout Risk vs Supplier Lead Time
```

```
In [87]: stockout_vs_lead = df.groupby('Supplier Name')[['Stock out Risk', 'Supplier Lead Time']]  
print(stockout_vs_lead)  
  
# Visualization  
plt.figure(figsize=(8,5))  
sns.set(style='whitegrid')  
  
sns.scatterplot(  
    data=stockout_vs_lead,  
    x='Supplier Lead Time',  
    y='Stock out Risk',  
    hue='Supplier Name',  
    palette='Set2',  
    s=120,  
    edgecolor='black',  
    alpha=0.85  
)  
  
sns.regplot(
```

```

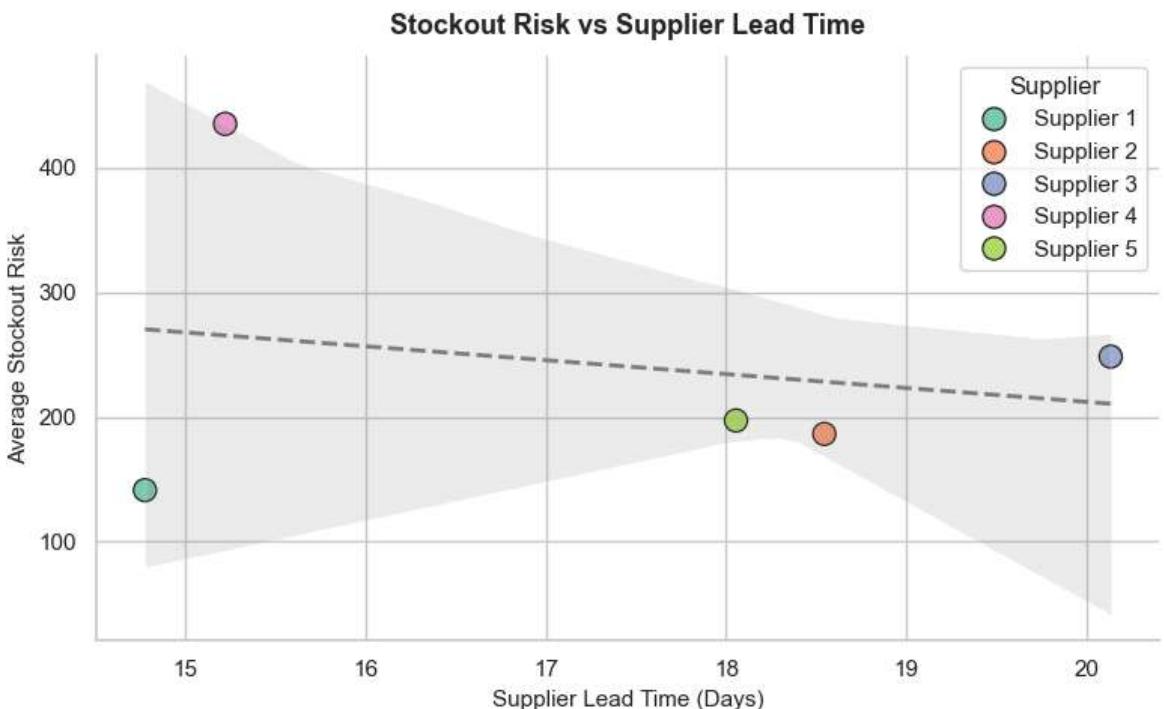
        data=stockout_vs_lead,
        x='Supplier Lead Time',
        y='Stock out Risk',
        scatter=False,
        color='gray',
        line_kws={'linewidth':2, 'linestyle':'--'}
    )

plt.title('Stockout Risk vs Supplier Lead Time', fontsize=13, fontweight='bold',
plt.xlabel('Supplier Lead Time (Days)', fontsize=11)
plt.ylabel('Average Stockout Risk', fontsize=11)

plt.legend(title='Supplier', loc='upper right', frameon=True)
sns.despine()
plt.tight_layout()
plt.show()

```

	Supplier Name	Stock out Risk	Supplier Lead Time
0	Supplier 1	141.259259	14.777778
1	Supplier 2	186.409091	18.545455
2	Supplier 3	248.466667	20.133333
3	Supplier 4	435.277778	15.222222
4	Supplier 5	197.166667	18.055556



In [88]: #Cross-Functional Analysis

In [89]: #8.1 Production Volume vs Sales

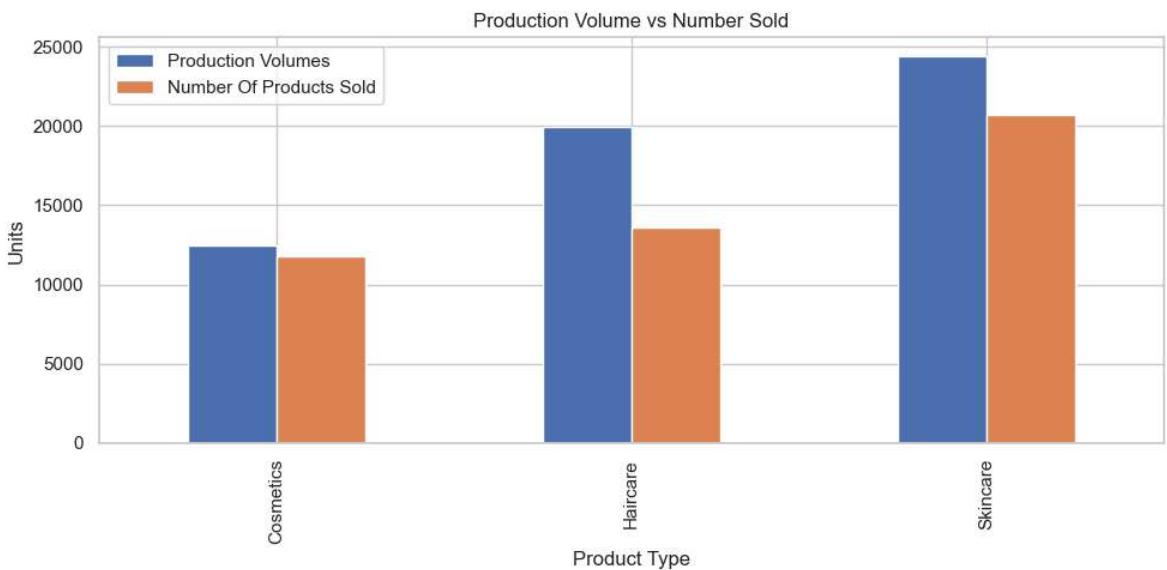
In [90]: prod_sales = df.groupby('Product Type')[['Production Volumes','Number Of Product
print(prod_sales)

```

# Visualization
prod_sales.set_index('Product Type').plot(kind='bar', title='Production Volume v
plt.ylabel('Units')
plt.tight_layout()
plt.show()

```

	Product Type	Production Volumes	Number Of Products Sold
0	Cosmetics	12461	11757
1	Haircare	19957	13611
2	Skincare	24366	20731



```
In [91]: #8.2 Profit Margin vs Location
```

```
In [92]: df['Profit Margin'] = (df['Profit'] / (df['Revenue Generated'] + 1e-6)) * 100

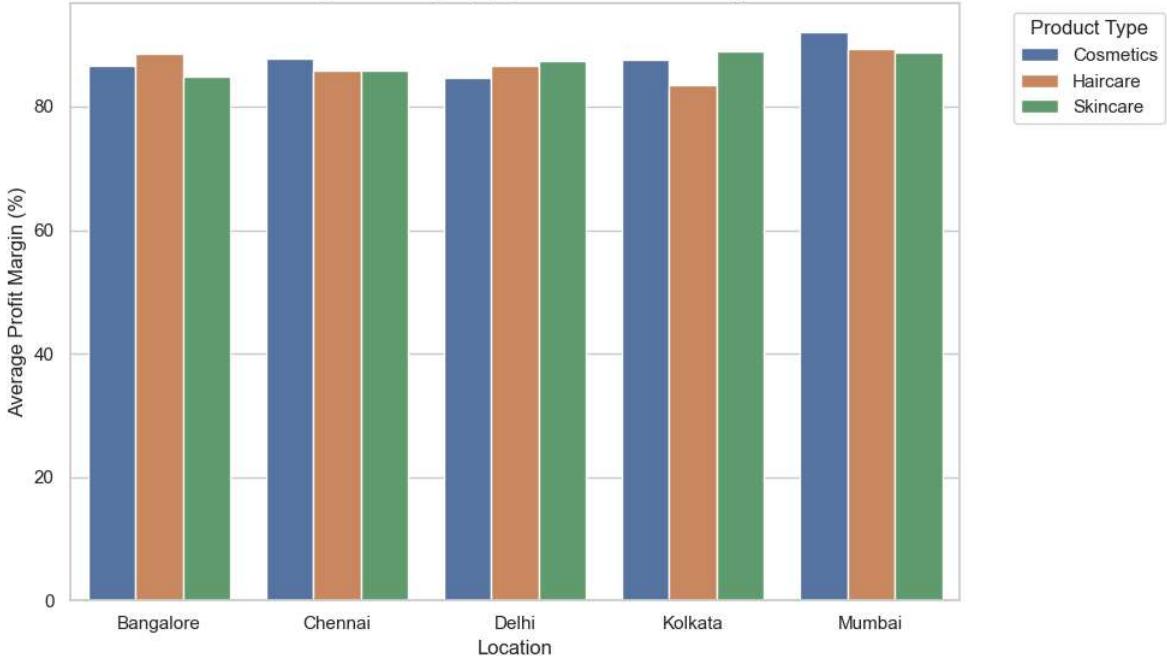
margin_loc = (
    df.groupby(['Product Type', 'Location'])['Profit Margin']
    .mean()
    .reset_index()
)

print(margin_loc)

# Visualization
plt.figure(figsize=(10,6))
sns.barplot(
    data=margin_loc,
    x='Location',
    y='Profit Margin',
    hue='Product Type',
    errorbar=None
)
plt.title('Average Profit Margin (%) by Location and Product Type')
plt.xlabel('Location')
plt.ylabel('Average Profit Margin (%)')
plt.legend(title='Product Type', bbox_to_anchor=(1.05, 1), loc='upper left')
plt.tight_layout()
plt.show()
```

	Product Type	Location	Profit Margin
0	Cosmetics	Bangalore	86.620018
1	Cosmetics	Chennai	87.846746
2	Cosmetics	Delhi	84.693012
3	Cosmetics	Kolkata	87.602771
4	Cosmetics	Mumbai	92.079421
5	Haircare	Bangalore	88.563774
6	Haircare	Chennai	85.899502
7	Haircare	Delhi	86.659572
8	Haircare	Kolkata	83.507351
9	Haircare	Mumbai	89.362945
10	Skincare	Bangalore	84.952659
11	Skincare	Chennai	85.843412
12	Skincare	Delhi	87.319519
13	Skincare	Kolkata	89.020078
14	Skincare	Mumbai	88.747884

Average Profit Margin (%) by Location and Product Type



In [93]: #8.3 Cost Breakdown (Operational Cost Distribution)

```
costs = {
    'Shipping Costs': df['Shipping Costs'].sum(),
    'Warehousing Costs': df['Warehousing Costs'].sum(),
    'Manufacturing Costs': df['Manufacturing Costs'].sum()
}

total_cost = sum(costs.values())

labels = list(costs.keys())
sizes = list(costs.values())
colors = ['#2E86AB', '#F18F01', '#C73E1D']

fig, ax = plt.subplots(figsize=(7,6))
wedges, texts, autotexts = ax.pie(
    sizes,
    labels=labels,
    autopct=lambda p: f'{p:.1f}%\n({p*total_cost/100:.0f})',
    startangle=120,
    colors=colors,
    textprops={'fontsize':10, 'color':'black'},
```

```

        pctdistance=0.8,
        wedgeprops={'edgecolor':'white', 'linewidth':2}
    )

for t in autotexts:
    t.set_fontweight('bold')
    t.set_fontsize(10)

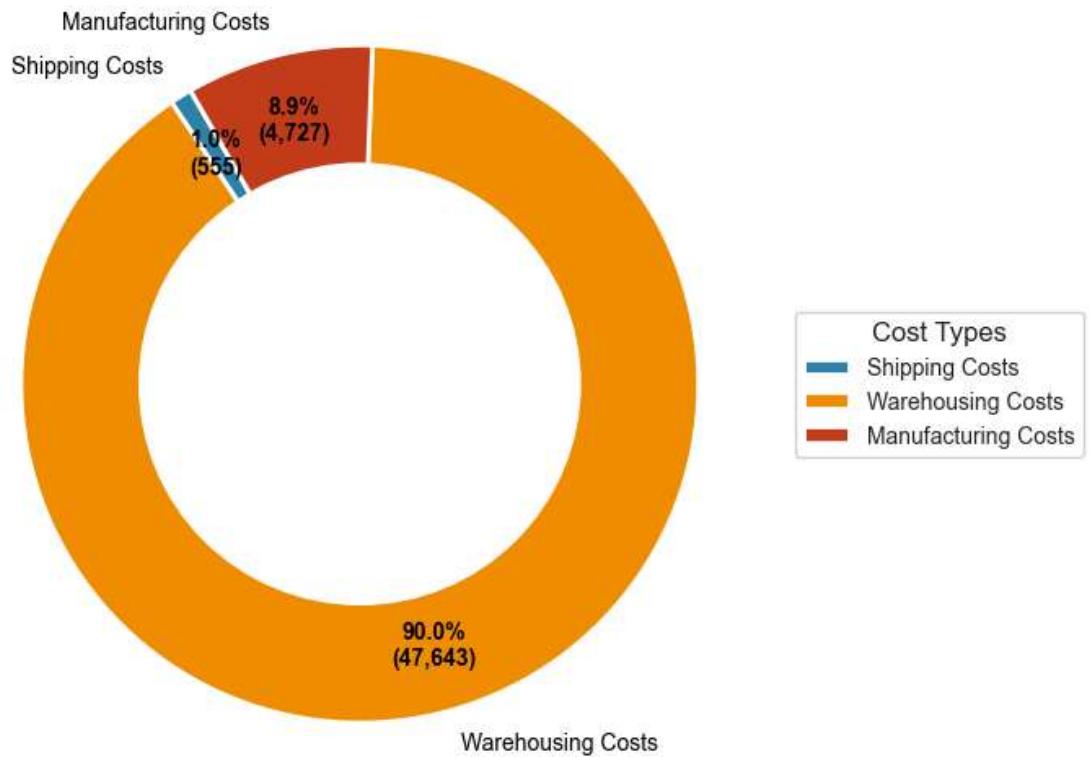
centre_circle = plt.Circle((0,0), 0.65, fc='white')
fig.gca().add_artist(centre_circle)

plt.title('Operational Cost Distribution', fontsize=15, fontweight='bold', pad=2)

plt.legend(
    wedges, labels,
    title="Cost Types",
    loc="center left",
    bbox_to_anchor=(1, 0, 0.5, 1),
    fontsize=10
)
plt.tight_layout()
plt.show()

```

Operational Cost Distribution



In [95]: #8.4 Full Cycle Time (Supplier → Manufacturing → Shipping → Customer)

In [96]: حساب إجمالي الوقت لكل مرحلة

```

lead_times = {
    'Supplier Lead Time': df['Supplier Lead Time'].sum(),
    'Manufacturing Lead Time': df['Manufacturing Lead Time'].sum(),
    'Shipping Times': df['Shipping Times'].sum()
}

```

```

# تحويلها إلى DataFrame
lead_df = pd.DataFrame(list(lead_times.items()), columns=['Stage', 'Full Lead Time (Days)'])

# حساب الإجمالي الكلي
total_lead_time = lead_df['Full Lead Time (Days)'].sum()

# حساب النسبة المئوية من الإجمالي
lead_df['% of Total Cycle Time'] = (lead_df['Full Lead Time (Days)'] / total_lead_time) * 100

# ترتيب تنازلي حسب القيمة
lead_df = lead_df.sort_values(by='Full Lead Time (Days)', ascending=False)

# عرض النتائج
print(lead_df)

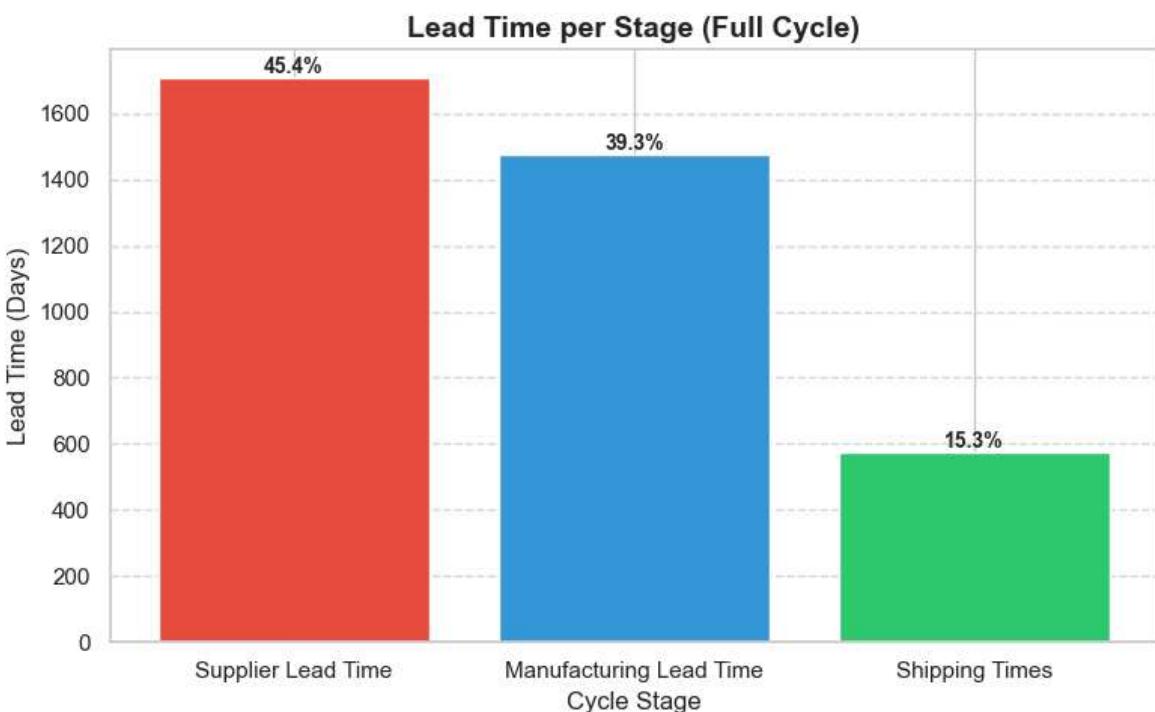
# Visualization
plt.figure(figsize=(8,5))
bars = plt.bar(lead_df['Stage'], lead_df['Full Lead Time (Days)'], color=[ '#E74C3C', '#3498DB', '#2ECC71'])
plt.title('Lead Time per Stage (Full Cycle)', fontsize=14, fontweight='bold')
plt.xlabel('Cycle Stage')
plt.ylabel('Lead Time (Days)')
plt.grid(axis='y', linestyle='--', alpha=0.7)

for bar, pct in zip(bars, lead_df['% of Total Cycle Time']):
    plt.text(bar.get_x() + bar.get_width()/2, bar.get_height(), f'{pct:.1f}%', ha='center', va='bottom', fontsize=10, fontweight='bold')

plt.tight_layout()
plt.show()

```

	Stage	Full Lead Time (Days)	% of Total Cycle Time
0	Supplier Lead Time	1708	45.425532
1	Manufacturing Lead Time	1477	39.281915
2	Shipping Times	575	15.292553



In []: