Supply Chain

June 25, 2025

1 Supply Chain Management

1.1 Project Overview

This project aims to perform a comprehensive analysis of a supply chain dataset to uncover key business insights and performance metrics across various operational dimensions. The dataset includes product information, sales performance, customer demographics, supplier data, manufacturing costs, transportation details, and quality control outcomes.

The objective of this analysis is to identify patterns, trends, and potential areas for optimization in the supply chain. Insights gained from this study will support data-driven decision-making in areas such as pricing strategy, inventory management, supplier evaluation, logistics planning, and quality assurance.

1.2 Dataset Overview

The dataset includes the following key features:

- **Product Information:** Product type, SKU, price, availability, and stock levels.
- **Sales & Revenue:** Number of products sold and total revenue generated.
- **Customer Demographics:** Gender and identity segments contributing to sales.
- **Manufacturing Data:** Production volumes and associated manufacturing costs.
- **Logistics & Transportation:** Shipping carriers, transportation modes, routes, and transportation costs.
- **Supplier Details:** Supplier name and location.
- **Quality Control:** Inspection results and defect rates.

1.3 Key Analytical Goals

- Measure sales and revenue performance by product type and customer demographics.
- Examine pricing effectiveness and its impact on product sales.
- Evaluate supplier contributions and manufacturing cost efficiency.
- Analyze transportation modes and routes in relation to shipping costs.

- Investigate quality metrics including inspection outcomes and defect rates.
- Compare performance metrics across different regional locations.

1.3.1 Load Required libraries

[53]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

1.3.2 Load Data From CSV File

[2]:
```
df = pd.read_csv("supply_chain_data.csv")
df.head(10)
```

[2]:
```
  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
5     haircare  SKU5   1.699976            87                      147
6     skincare  SKU6   4.078333            48                       65
7    cosmetics  SKU7  42.958384            59                      426
8    cosmetics  SKU8  68.717597            78                      150
9     skincare  SKU9  64.015733            35                      980

   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
5        2828.348746            Non-binary            90          27
6        7823.476560                  Male            11          15
7        8496.103813                Female            93          17
8        7517.363211                Female             5          10
9        4971.145988               Unknown            14          27

   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
5                66  …  Bangalore        10                 104
6                58  …    Kolkata        14                 314
```

```
7                 11  …  Bangalore        22                   564
8                 15  …     Mumbai        13                   769
9                 83  …    Chennai        29                   963

  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
5                      17           56.766476                Fail
6                      24            1.085069             Pending
7                       1           99.466109                Fail
8                       8           11.423027             Pending
9                      23           47.957602             Pending

   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      2.779194                  Road  Route A  235.461237
6      1.000911                   Sea  Route A  134.369097
7      0.398177                  Road  Route C  802.056312
8      2.709863                   Sea  Route B  505.557134
9      3.844614                  Rail  Route B  995.929461

[10 rows x 24 columns]
```

1.3.3 Data Inspection

[3]:
```
# Get all columns name
df.columns
```

[3]:
```
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')
```

[4]:
```
#Drop un-usefull columns and update main data frame
df.drop(['Lead times','Lead time','Shipping times','Manufacturing lead␣
 ↪time'],axis=1,inplace=True)
```

[5]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 20 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   Order quantities         100 non-null    int64
 9   Shipping carriers        100 non-null    object
 10  Shipping costs           100 non-null    float64
 11  Supplier name            100 non-null    object
 12  Location                 100 non-null    object
 13  Production volumes       100 non-null    int64
 14  Manufacturing costs      100 non-null    float64
 15  Inspection results       100 non-null    object
 16  Defect rates             100 non-null    float64
 17  Transportation modes     100 non-null    object
 18  Routes                   100 non-null    object
 19  Costs                    100 non-null    float64
dtypes: float64(6), int64(5), object(9)
memory usage: 15.8+ KB
```

[6]:
```
df.isnull().sum()
```

[6]:
```
Product type               0
SKU                        0
Price                      0
Availability               0
Number of products sold    0
Revenue generated          0
Customer demographics      0
Stock levels               0
Order quantities           0
Shipping carriers          0
Shipping costs             0
Supplier name              0
Location                   0
Production volumes         0
Manufacturing costs        0
Inspection results         0
```

```
Defect rates              0
Transportation modes      0
Routes                    0
Costs                     0
dtype: int64
```

[7]:
```
df.describe()
```

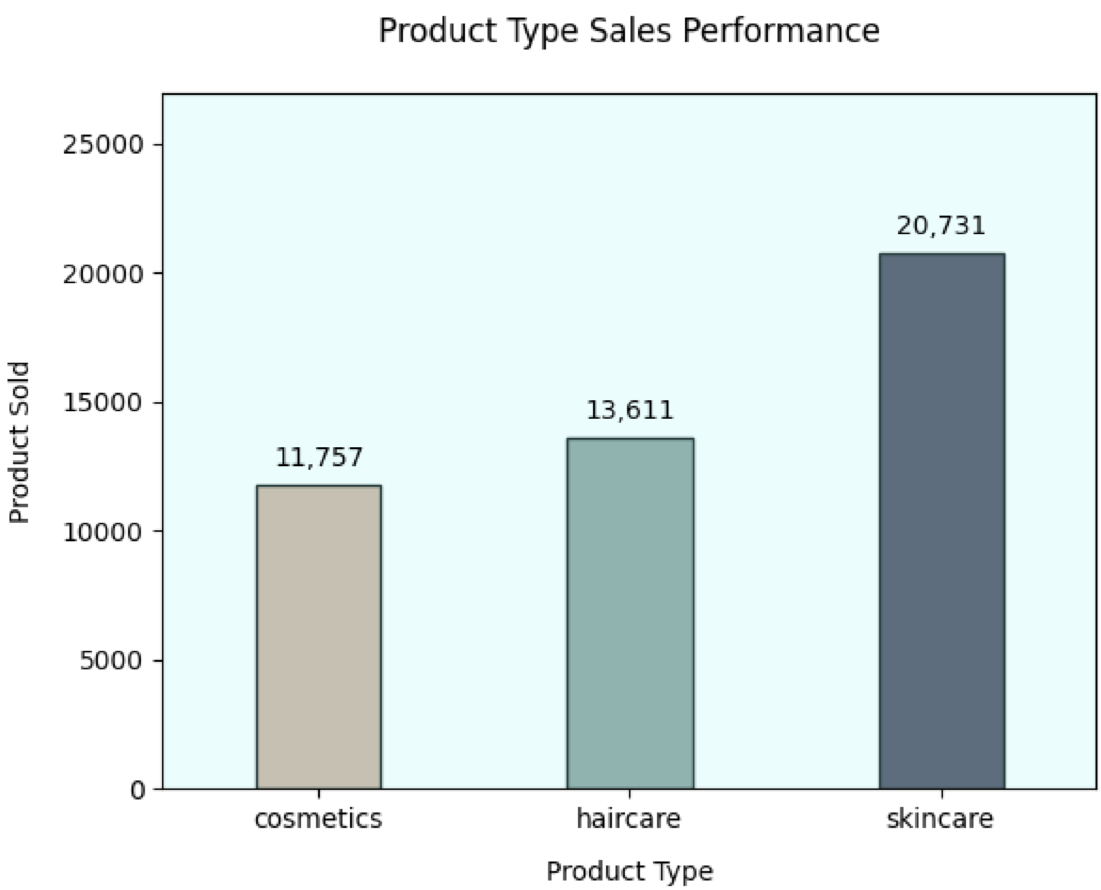
[7]:
```
            Price  Availability  Number of products sold  Revenue generated  \
count  100.000000    100.000000               100.000000         100.000000
mean    49.462461     48.400000               460.990000        5776.048187
std     31.168193     30.743317               303.780074        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  Order quantities  Shipping costs  Production volumes  \
count    100.000000        100.000000      100.000000          100.000000
mean      47.770000         49.220000        5.548149          567.840000
std       31.369372         26.784429        2.651376          263.046861
min        0.000000          1.000000        1.013487          104.000000
25%       16.750000         26.000000        3.540248          352.000000
50%       47.500000         52.000000        5.320534          568.500000
75%       73.000000         71.250000        7.601695          797.000000
max      100.000000         96.000000        9.929816          985.000000

       Manufacturing costs  Defect rates       Costs
count           100.000000    100.000000  100.000000
mean             47.266693      2.277158  529.245782
std              28.982841      1.461366  258.301696
min               1.085069      0.018608  103.916248
25%              22.983299      1.009650  318.778455
50%              45.905622      2.141863  520.430444
75%              68.621026      3.563995  763.078231
max              99.466109      4.939255  997.413450
```

2 Exploratory Analysis

2.0.1 Product Type Sales Performance

[178]:
```
plt.margins(0.3)

sold = df.groupby('Product type')['Number of products sold'].sum().
 ↪reset_index().sort_values(by='Number of products sold')
```

```
ax = sns.barplot(data = sold,
                 x = "Product type",
                 y = "Number of products sold",
                 palette = ['#c8c2ae','#8ab9b5','#586d83'],
                 hue = "Product type",
                 errorbar = None,
                 width = 0.4,
                 edgecolor="#2b4141")

ax.set_facecolor("#ebfdfd")
ax.set_title('Product Type Sales Performance',y=1.05)
ax.set_xlabel('Product Type',labelpad=10)
ax.set_ylabel('Product Sold',labelpad=10)


for container in ax.containers:
    ax.bar_label(container,fmt='{:,.0f}',padding=5)
plt.savefig('sale.png')
plt.show()
```



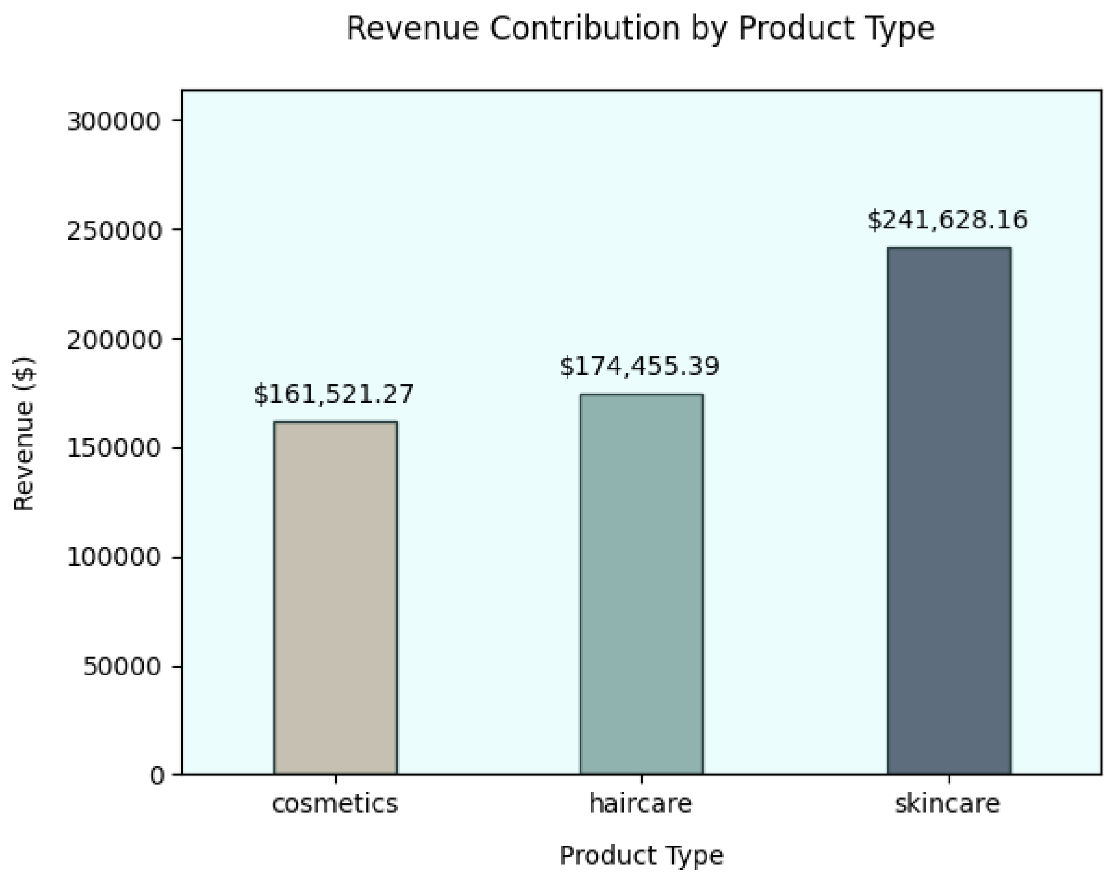
The bar chart illustrates the total number of units sold for three product type: **cosmetics, haircare, and skincare**.

- **Skincare** products achieved the highest sales volume, with **20,731** units sold, indicating strong consumer demand in this category.
- **Haircare** followed with **13,611** units sold, showing moderate market performance.
- **Cosmetics** registered the lowest sales among the three, with **11,757** units sold.

This data highlights a clear consumer preference for skincare products, significantly outperforming both haircare and cosmetics. The insights suggest potential for increased investment or marketing focus on the skincare category to further leverage its market traction. and For both Haircare and Cosmetics, a dual approach of strategic niche identification and aggressive, targeted promotions is crucial for revenue growth.

2.0.2 Revenue Contribution by Product Type

[179]:
```python
plt.margins(0.3)

revenue = df.groupby('Product type')['Revenue generated'].sum().reset_index().
 ↪sort_values(by='Revenue generated')

ax = sns.barplot(data = revenue,
                 x = "Product type",
                 y = "Revenue generated",
                 palette = ['#c8c2ae','#8ab9b5','#586d83'],
                 hue = "Product type",
                 errorbar = None,
                 width = 0.4,
                 edgecolor="#2b4141")

ax.set_facecolor("#ebfdfd")
ax.set_title('Revenue Contribution by Product Type',y=1.05)
ax.set_xlabel('Product Type',labelpad=10)
ax.set_ylabel('Revenue ($)',labelpad=10)

for container in ax.containers:
    ax.bar_label(container,fmt='${:,.2f}',padding=5)

plt.show()
```



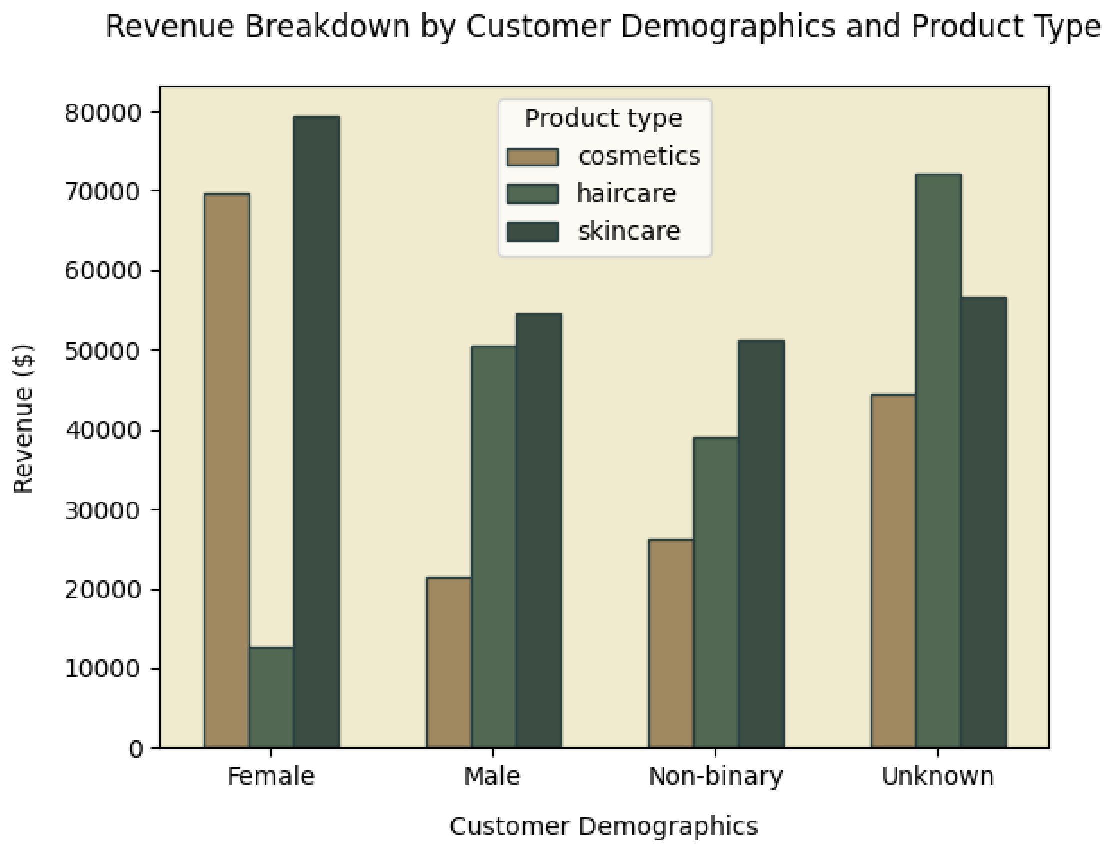
The bar chart presents a clear picture of revenue distribution across three key product types: **cosmetics, haircare, and skincare**.

- **Skincare** products stand out, generating the highest revenue at **$241,628.16** , which indicates robust consumer interest.
- **Haircare** products secured the second spot with **$174,455.39**, reflecting a solid market presence.
- **Cosmetics** recorded the lowest revenue at **$161,521.27**.

We've observed that skincare product sales and their corresponding revenue are dominant, surpassing both haircare and cosmetic categories. The insights suggest potential for increased investment or marketing focus on the skincare category to further leverage its market traction. and For both Haircare and Cosmetics, a dual approach of strategic niche identification and aggressive, targeted promotions is crucial for revenue growth.

2.0.3 Revenue Breakdown by Customer Demographics and Product Type

[135]:
```
revenue = df.groupby(['Customer demographics','Product type'])['Revenue␣
↪generated'].sum().reset_index()

ax = sns.barplot(data = revenue,
                 x = "Customer demographics",
                 y = "Revenue generated",
                 palette = ['#AA8B56','#4E6C50','#395144'],
                 hue = "Product type",
                 errorbar = None,
                 width = 0.6,
                 edgecolor="#2b4141")

ax.set_facecolor("#F0EBCE")
ax.set_title('Revenue Breakdown by Customer Demographics and Product Type',y=1.
↪05)
ax.set_xlabel('Customer Demographics',labelpad=10)
ax.set_ylabel('Revenue ($)',labelpad=10)
plt.show()
```



This clustered bar chart breaks down revenue by customer demographics (Female, Male, Non-binary, Unknown) across three product types: **Cosmetics, Haircare, and Skincare**.

Female customers are the primary revenue drivers, especially for Skincare and Cosmetics.

Male customers show higher revenue in Skincare and Haircare compared to cosmetics.

Non-binary and **Unknown** demographics contribute lower, but still noticeable, revenue across all categories, with Skincare being generally higher.

Overall, Skincare consistently generates high revenue across all known demographics, while Cosmetics is strong with Female customers and Haircare has a broader appeal.

The insights suggest to prioritize female-centric marketing for Cosmetics and Skincare and male marketing for Skincare. Capitalize on haircare's universal appeal by marketing it to a diverse customer base.

2.0.4 Relationship Between Price And Number Of Product Sold

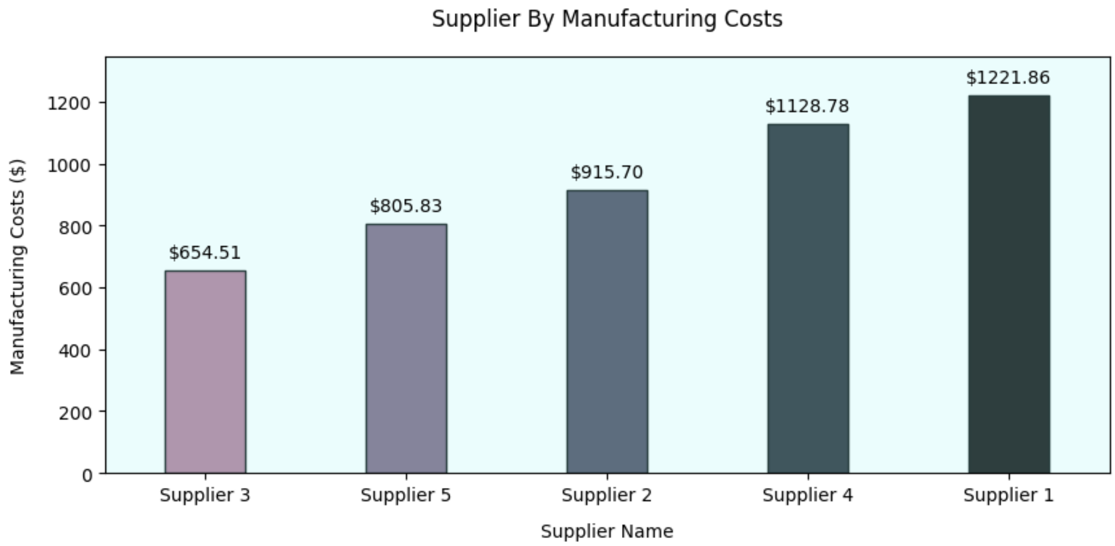
[136]:
```
plt.figure(figsize=(10, 8))
ax = sns.scatterplot(data=df,
                     x='Price',
                     y='Number of products sold',
                     hue='Product type',
                     palette=['#FF7D29','#5F8D4E','#E63E6D'],)

sns.move_legend(
    ax, "lower center",
     bbox_to_anchor=(0.5, -0.20), ncol=3, title='Product type', frameon=True,
)

ax.set_title('Relationship Between Price And Number Of Product Sold',y=1.05)
ax.set_ylabel('Number Of Products Sold',labelpad=10)
ax.set_xlabel('Price ($)',labelpad=10)

plt.grid(True)
plt.tight_layout()
plt.show()

print(f"The correlation between Price and number of product sold is:␣
 ↪{df['Price'].corr(df['Number of products sold']):.4f}")
```

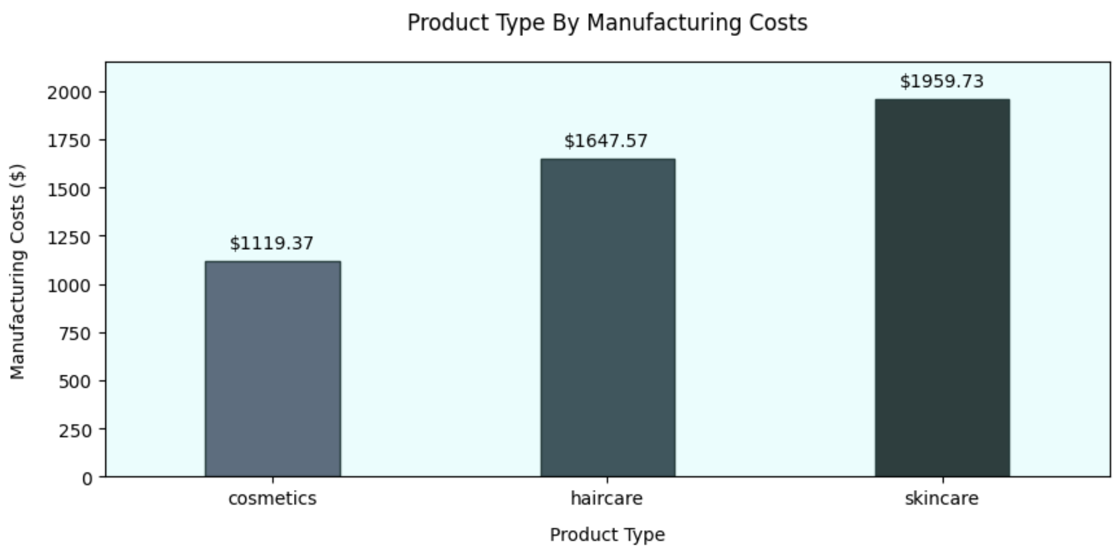

```
The correlation between Price and number of product sold is: 0.0057
```

2.0.5 Manufacturing Cost Breakdown by Supplier and Product

[143]:
```python
plt.figure(figsize = (10,10))
plt.subplot(2,1,1)
plt.margins(0.1)

manufacturing_cost_supplier = df.groupby('Supplier name')['Manufacturing␣
↪costs'].sum().reset_index().sort_values(by="Manufacturing costs")

ax = sns.barplot(manufacturing_cost_supplier,
                 x = 'Supplier name',
                 y = 'Manufacturing costs',
                 palette = ['#b392b1','#82809f','#586d83','#3b5862','#2b4141'],
                 hue = "Supplier name",
                 errorbar = None,
                 width = 0.4,
                 edgecolor="#2b4141")
```

```python
ax.set_facecolor("#ebfdfd")
ax.set_title('Supplier By Manufacturing Costs',y=1.05)
ax.set_xlabel('Supplier Name',labelpad=10)
ax.set_ylabel('Manufacturing Costs ($)',labelpad=10)

for container in ax.containers:
    ax.bar_label(container,fmt='${:.2f}',padding=5)


plt.subplot(2,1,2)
plt.subplots_adjust(hspace=0.4)
plt.margins(0.1)

manufacturing_cost_product = df.groupby('Product type')['Manufacturing costs'].
↪sum().reset_index().sort_values(by="Manufacturing costs")

ax = sns.barplot(manufacturing_cost_product,
                 x = 'Product type',
                 y = 'Manufacturing costs',
                 palette = ['#586d83','#3b5862','#2b4141'],
                 hue = "Product type",
                 errorbar = None,
                 width = 0.4,
                 edgecolor="#2b4141")

ax.set_facecolor("#ebfdfd")
ax.set_title('Product Type By Manufacturing Costs',y=1.05)
ax.set_xlabel('Product Type',labelpad=10)
ax.set_ylabel('Manufacturing Costs ($)',labelpad=10)

for container in ax.containers:
    ax.bar_label(container,fmt='${:.2f}',padding=5)

plt.show()
```

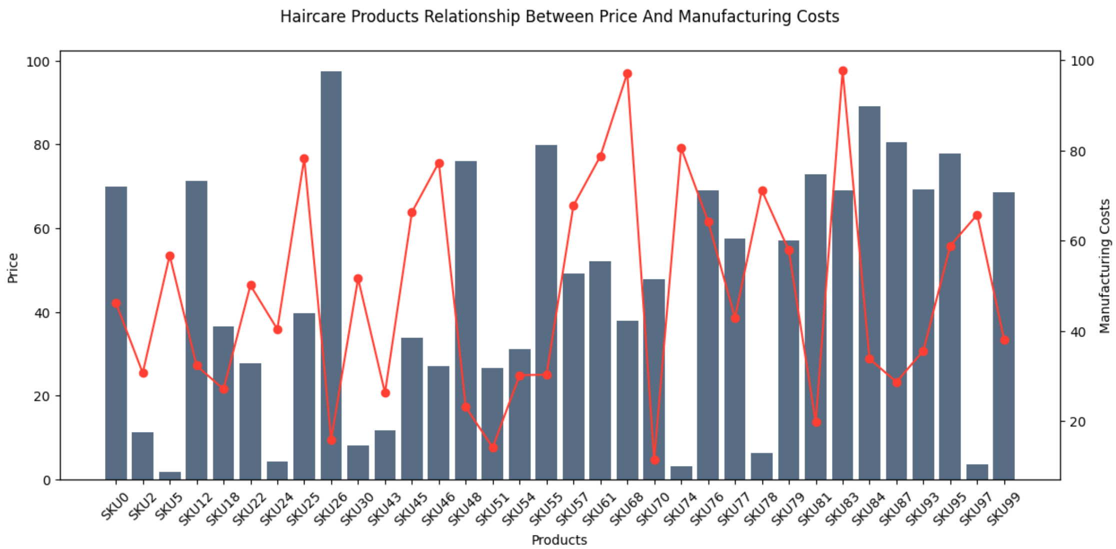




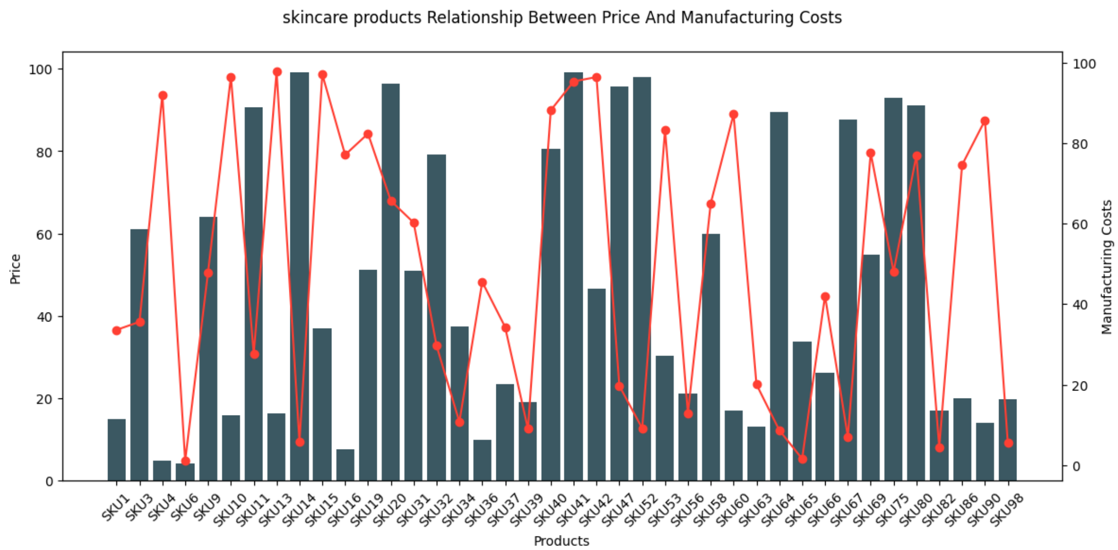
2.0.6 Price Vs Manufacturing Cost

[139]:
```
product = df[df['Product type'] == "haircare"]
fig, ax1 = plt.subplots(figsize=(12, 6))
ax2 = ax1.twinx()
ax1.bar(product['SKU'], product['Price'],color="#586d83")
ax2.plot(product['SKU'], product['Manufacturing costs'], 'o-', color="#FF3F33" )

ax1.set_xlabel('Products')
ax1.tick_params(axis='x', labelbottom=True,rotation=45)
ax1.set_ylabel('Price')
ax2.set_ylabel('Manufacturing Costs')
```

```
plt.title('Haircare Products Relationship Between Price And Manufacturing␣
↪Costs',y=1.05)
fig.tight_layout(h_pad=10)
plt.show()
```

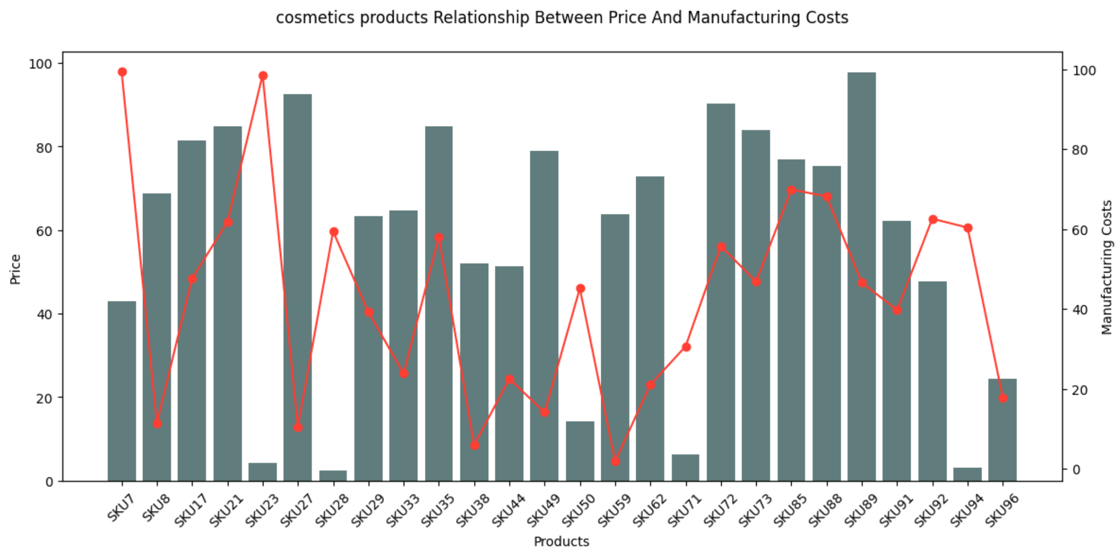


[140]:
```
product = df[df['Product type'] == "skincare"]
fig, ax1 = plt.subplots(figsize=(12, 6))
ax2 = ax1.twinx()
ax1.bar(product['SKU'], product['Price'],color="#3b5862")
ax2.plot(product['SKU'], product['Manufacturing costs'], 'o-', color="#FF3F33" )

ax1.set_xlabel('Products')
ax1.tick_params(axis='x', labelbottom=True,rotation=45)
ax1.set_ylabel('Price')
ax2.set_ylabel('Manufacturing Costs')
plt.title('skincare products Relationship Between Price And Manufacturing␣
↪Costs',y=1.05)
fig.tight_layout(h_pad=10)
plt.show()
```



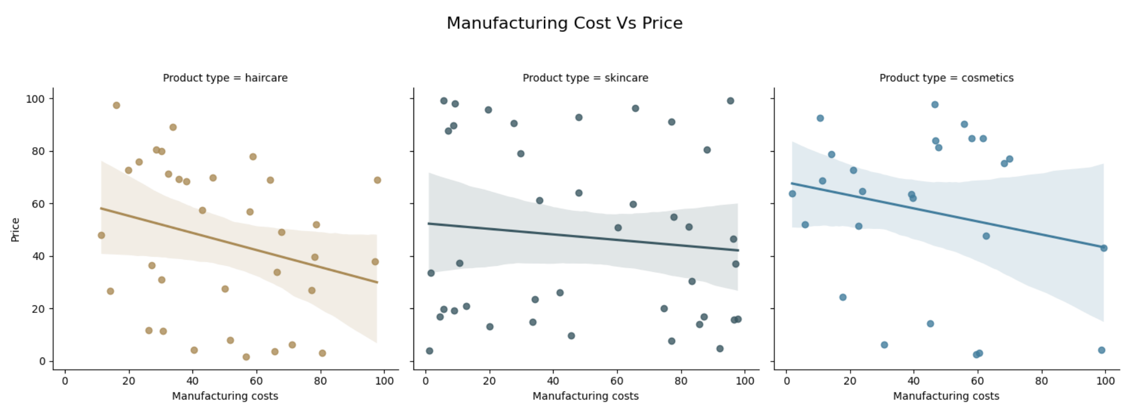
[141]:

```
product = df[df['Product type'] == "cosmetics"]
fig, ax1 = plt.subplots(figsize=(12, 6))
ax2 = ax1.twinx()
ax1.bar(product['SKU'], product['Price'],color="#617c7c")
ax2.plot(product['SKU'], product['Manufacturing costs'], 'o-', color="#FF3F33" )

ax1.set_xlabel('Products')
ax1.tick_params(axis='x', labelbottom=True,rotation=45)
ax1.set_ylabel('Price')
ax2.set_ylabel('Manufacturing Costs')
plt.title('cosmetics products Relationship Between Price And Manufacturing␣
 ↪Costs',y=1.05)
fig.tight_layout(h_pad=10)
plt.show()
```



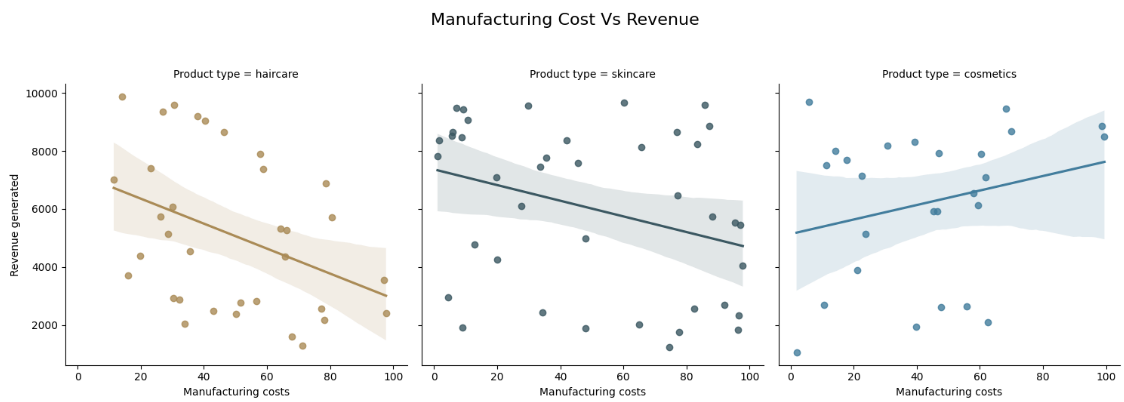
2.0.7 Manufacturing Cost vs Price

[148]:
```
ax = sns.lmplot(df,
                x="Manufacturing costs",
                y="Price",
                hue="Product type",
                col="Product type",
                palette = ['#AA8B56','#3b5862','#427D9D'])

fig = ax.fig
fig.suptitle('Manufacturing Cost Vs Price', fontsize=16,y=1.05)
plt.tight_layout()
plt.show()
```



2.0.8 Manufacturing Cost vs Revenue

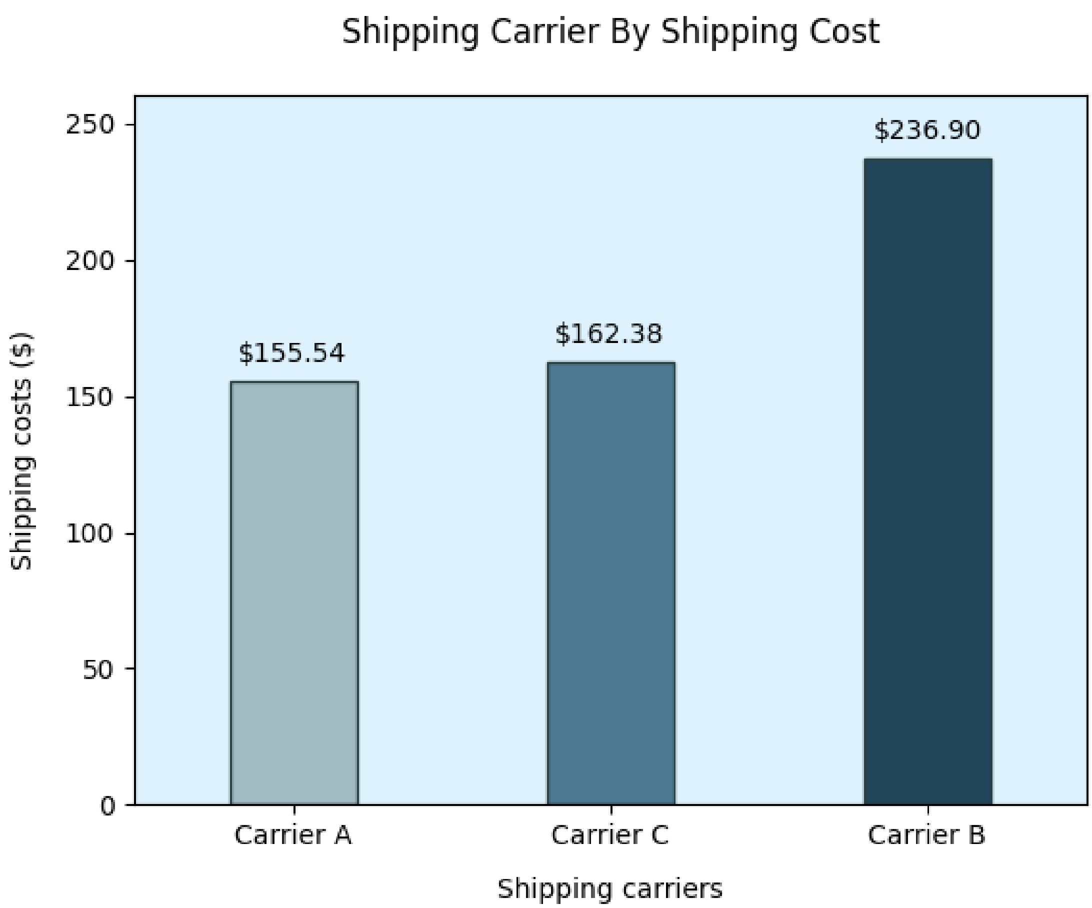
[147]:
```
ax = sns.lmplot(df,
                x="Manufacturing costs",
                y="Revenue generated",
                hue="Product type",
                col="Product type",
                palette = ['#AA8B56','#3b5862','#427D9D'])

fig = ax.fig
fig.suptitle('Manufacturing Cost Vs Revenue', fontsize=16,y=1.05)
plt.tight_layout()
plt.show()
```



2.0.9 Shipping Carrier By Shipping Cost

[149]:
```
shipping_cost = df.groupby('Shipping carriers')['Shipping costs'].sum().
  ↪reset_index().sort_values(by="Shipping costs")

plt.margins(0.1)

ax = sns.barplot(shipping_cost,
                 x = 'Shipping carriers',
                 y = 'Shipping costs',
                 palette = ['#9BBEC8','#427D9D','#164863'],
                 hue = "Shipping carriers",
                 errorbar = None,
                 width = 0.4,
                 edgecolor="#2b4141")

ax.set_facecolor("#DDF2FD")
ax.set_title('Shipping Carrier By Shipping Cost',y=1.05)
ax.set_xlabel('Shipping carriers',labelpad=10)
```

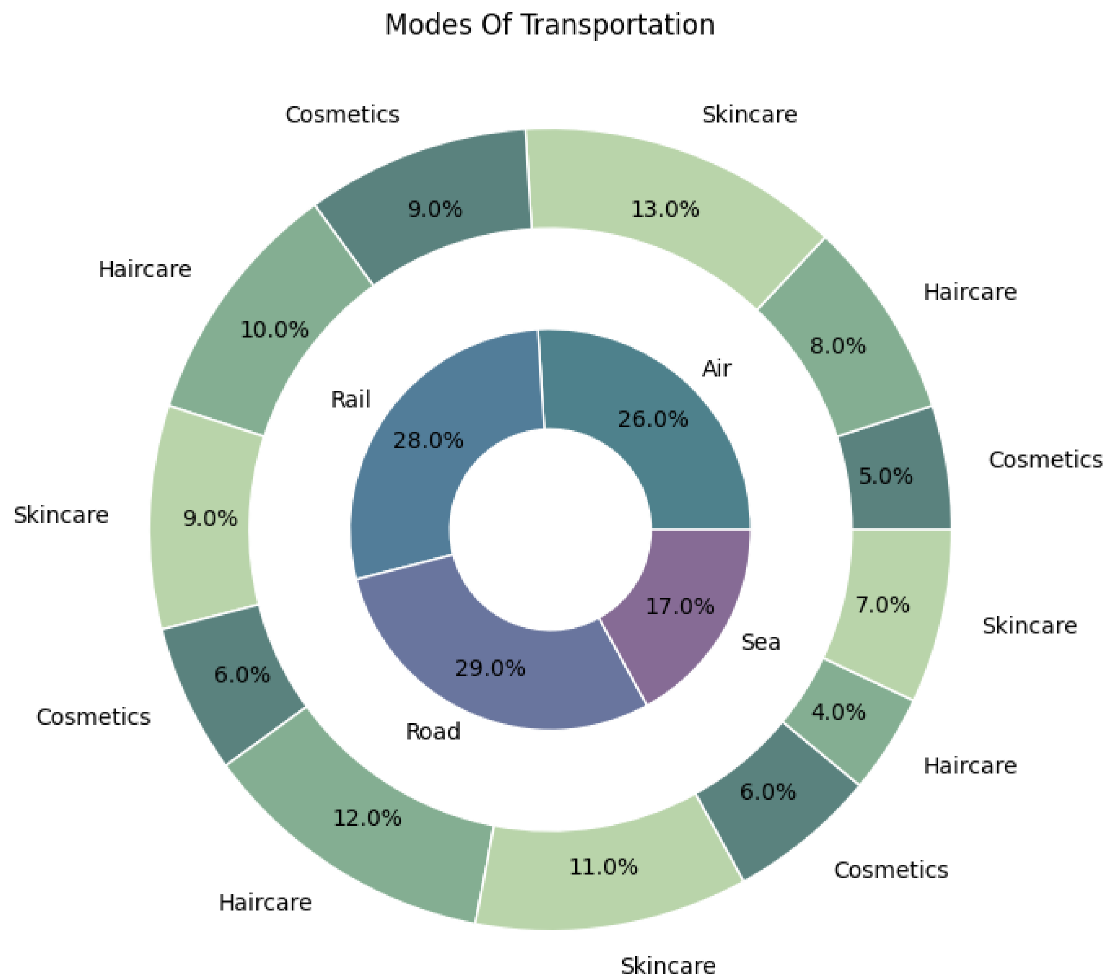
```
ax.set_ylabel('Shipping costs ($)',labelpad=10)

for container in ax.containers:
    ax.bar_label(container,fmt='${:.2f}',padding=5)

plt.show()
```



2.0.10 Modes Of Transportation

[155]:
```
transportation = df.groupby(['Transportation modes','Product␣
↪type'])['Transportation modes'].value_counts().reset_index()
fig, ax = plt.subplots(figsize=(7, 7))

# Outer pie chart data
wedges_outer, texts_outer, autotexts_outer = ax.pie(transportation['count'],
                                                    radius=1.2,
                                                   ␣
↪labels=transportation['Product type'].str.title(),
```

```
                                                  autopct='%1.1f%%',
                                                  pctdistance=0.85,
                                                  ␣
↪colors=['#5A827E','#84AE92','#B9D4AA'],
                                                  wedgeprops=dict(width=0.3,␣
↪edgecolor='w'))

# Inner pie chart data
wedges_inner, texts_inner, autotexts_inner = ax.pie(transportation.
↪groupby(['Transportation modes'])['count'].sum(),
                                                  radius=0.6,
                                                  ␣
↪labels=transportation['Transportation modes'].unique(),
                                                  autopct='%1.1f%%',
                                                  pctdistance=0.75,
                                                  ␣
↪colors=['#4e818c','#527d99','#69759e','#866b95'],
                                                  wedgeprops=dict(width=0.3,␣
↪edgecolor='w'))

plt.title('Modes Of Transportation', y=1.05)
plt.axis('equal')
plt.show()
```

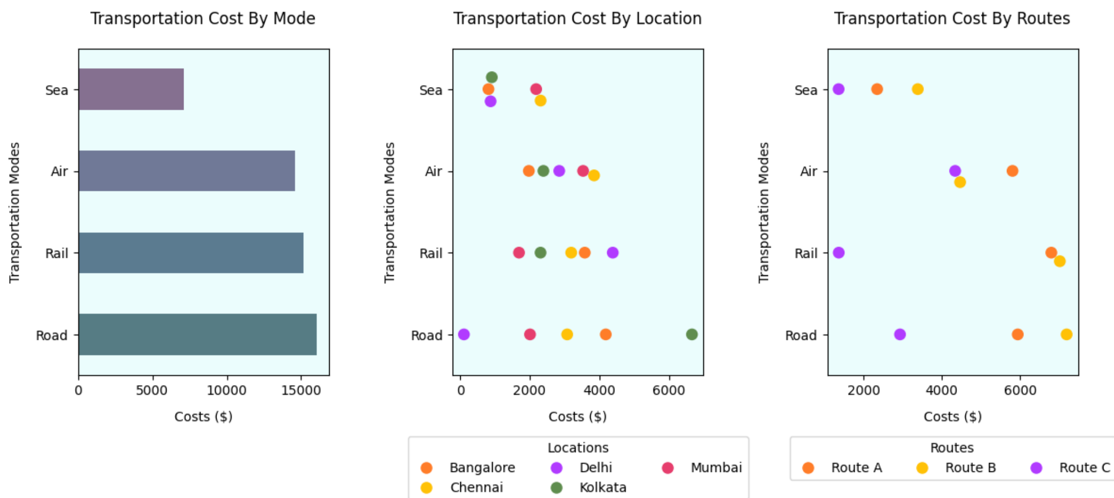


2.0.11 Transportation Costs Relationship With Mode And Location

[156]:
```
plt.figure(figsize=(12,6))
plt.subplot(1,3,1)

mode_cost = df.groupby(['Transportation modes'])['Costs'].sum().reset_index().
↪sort_values(by="Costs")
ax = sns.barplot(mode_cost,
            x='Costs',
            y='Transportation modes',
            hue='Transportation modes',
            palette=['#866b95','#69759e','#527d99','#4e818c'],
            width=.5
           )

ax.set_facecolor("#ebfdfd")
```

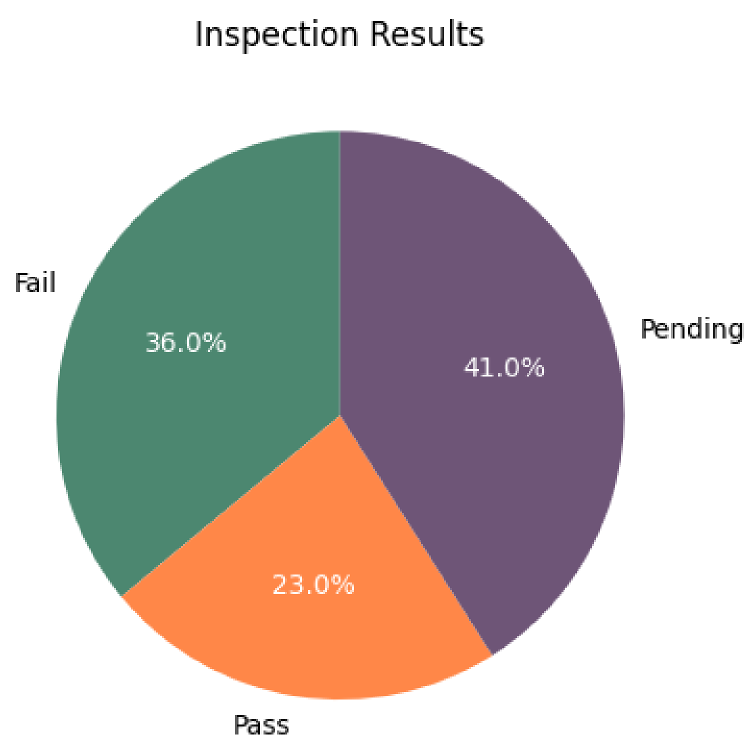
```
ax.set_title('Transportation Cost By Mode',y=1.05)
ax.set_ylabel('Transportation Modes',labelpad=10)
ax.set_xlabel('Costs ($)',labelpad=10)

plt.subplot(1,3,2)

cost = df.groupby(['Transportation modes','Location'])['Costs'].sum().
 ↪reset_index()

ax = sns.swarmplot(data=cost,
                   x="Costs",
                   y="Transportation modes",
                   hue="Location",
                   size=9,
                   palette=['#FF7D29','#FFC107','#B13BFF','#5F8D4E','#E63E6D'],
                   order=mode_cost['Transportation modes'])

sns.move_legend(
    ax, "lower center",
     bbox_to_anchor=(0.5, -0.40,), ncol=3, title='Locations', frameon=True,
)

ax.set_facecolor("#ebfdfd")
ax.set_title('Transportation Cost By Location',y=1.05)
ax.set_ylabel('Transportation Modes',labelpad=10)
ax.set_xlabel('Costs ($)',labelpad=10)

plt.subplot(1,3,3)

cost = df.groupby(['Transportation modes','Routes'])['Costs'].sum().
 ↪reset_index()

ax = sns.swarmplot(data=cost,
                   x="Costs",
                   y="Transportation modes",
                   hue="Routes",
                   size=9,
                   palette=['#FF7D29','#FFC107','#B13BFF'],
                   order=mode_cost['Transportation modes'])

sns.move_legend(
    ax, "lower center",
     bbox_to_anchor=(0.5, -0.34), ncol=3, title='Routes', frameon=True,
)

ax.set_facecolor("#ebfdfd")
ax.set_title('Transportation Cost By Routes',y=1.05)
```

```
ax.set_ylabel('Transportation Modes',labelpad=10)
ax.set_xlabel('Costs ($)',labelpad=10)
plt.tight_layout()
plt.show()
```



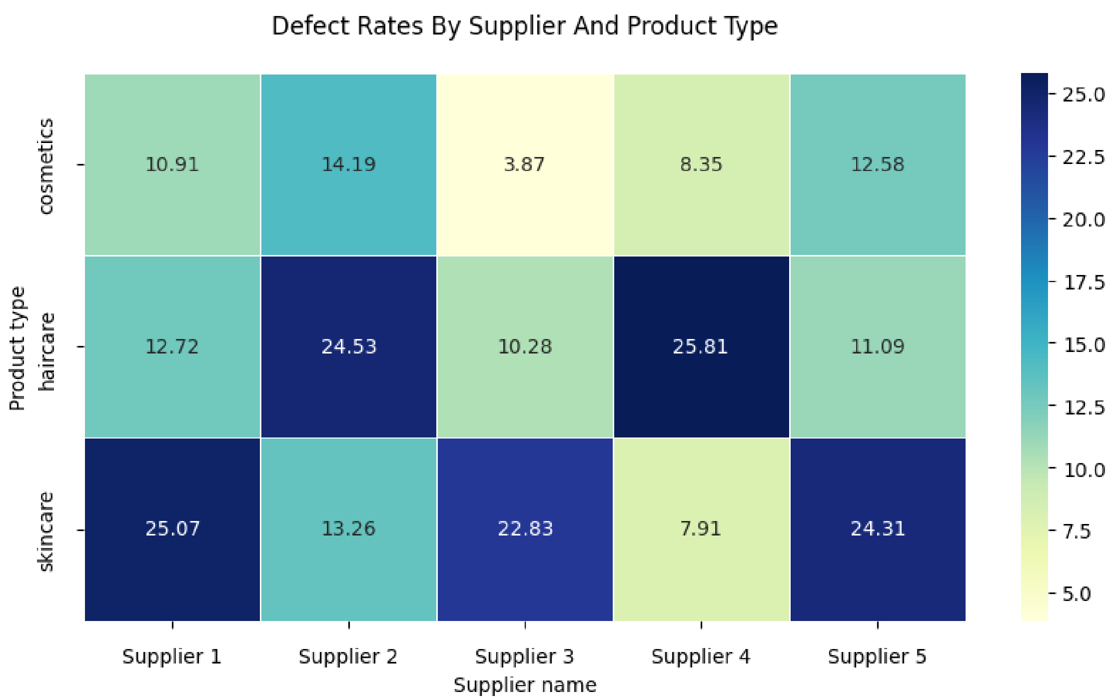
2.0.12 Inspection Results

[158]:
```
inspection_results = df.groupby(['Inspection results'])['Inspection results'].
  ↪value_counts().reset_index()

wedges, texts, autotexts = plt.pie(inspection_results['count'],
                                   labels=inspection_results['Inspection␣
  ↪results'],
                                   autopct='%1.1f%%',
                                   startangle=90,
                                   colors=['#4c8770','#ff8748','#6e5577'])

for autotext in autotexts:
    autotext.set_color('white')

plt.title('Inspection Results')
plt.show()
```



2.0.13 Defect Rates By Supplier And Product Type

[157]:
```
defect_rate = df.groupby(['Supplier name','Product type'])['Defect rates'].
  ↪sum().reset_index()

plt.figure(figsize=(10, 5))
ax = sns.heatmap(defect_rate.pivot(index = 'Product type',
                            columns = 'Supplier name',
                            values = 'Defect rates'),
                            annot=True,
                            cmap="YlGnBu",
                            fmt=".2f",
                            linewidths=.5,)

plt.title('Defect Rates By Supplier And Product Type', y=1.05)
ax.tick_params(axis='both',pad=10)
plt.show()
```



2.0.14 Relationship Between Manufacturing Costs and Defect Rates

[159]:
```
plt.figure(figsize=(10, 8))
ax = sns.scatterplot(data=df,
                x="Manufacturing costs",
                y="Defect rates",
                hue="Product type",
                palette=['#FF7D29','#5F8D4E','#E63E6D'])

sns.move_legend(
    ax, "lower center",
     bbox_to_anchor=(0.5, -0.20), ncol=3, title='Product type', frameon=True,
)

plt.title("Relationship Between Manufacturing Costs and Defect Rates",y=1.05)
plt.xlabel("Manufacturing Costs",labelpad=10)
plt.ylabel("Defect Rates",labelpad=10)
plt.grid(True)
plt.tight_layout()
plt.show()

print(f"The correlation between Price and number of product sold is:␣
 ↪{df['Manufacturing costs'].corr(df['Defect rates']):.4f}")
```


```
The correlation between Price and number of product sold is: -0.0078
```

2.0.15 Location-Based Performance Variance Comparison

[166]:
```python
regional_performance = df.groupby('Location').agg({
    'Price':'mean',
    'Revenue generated': 'mean',
    'Defect rates': 'mean',
    'Shipping costs': 'mean',
    'Costs': 'mean'
}).round(2)
print(regional_performance)
```

```
           Price  Revenue generated  Defect rates  Shipping costs   Costs
Location
Bangalore  38.03            5700.10          2.09            5.75  586.71
Chennai    68.15            5957.14          2.64            4.69  621.75
Delhi      46.27            5401.85          2.23            5.07  548.24
Kolkata    49.45            5483.10          2.29            5.76  491.27
Mumbai     44.01            6261.59          2.12            6.25  428.34
```

[ ]:

```

```