In [12]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

In [13]:
```
import warnings
warnings.filterwarnings('ignore')
```

In [15]:
```
data = pd.read_csv("C:\\Users\\mohan\\Downloads\\Supply Chain Project\\SupplyChainData.c
```

In [16]:
```
data.head(5)
```

Out[16]:

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

5 rows × 24 columns

In [17]:
```
data.head()
```

Out[17]:

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

5 rows × 24 columns

In [18]:
```
data.tail()
```

Out[18]:

| | Product type | SKU | Price | Availability | Number of products sold | Revenue generated | Customer demographics | Stock levels | Lead times | qua |
|---|---|---|---|---|---|---|---|---|---|---|
| **95** | Automotive Parts & Accessories | SKU95 | 77.903927 | 65 | 672 | 7386.363944 | Unknown | 15 | 14 | |
| **96** | Cell Phones & Accessories | SKU96 | 24.423131 | 29 | 324 | 7698.424766 | Non-binary | 67 | 2 | |
| **97** | Automotive Parts & Accessories | SKU97 | 3.526111 | 56 | 62 | 4370.916580 | Male | 46 | 19 | |
| **98** | Beauty & Personal Care | SKU98 | 19.754605 | 43 | 913 | 8525.952560 | Female | 53 | 1 | |
| **99** | Automotive Parts & Accessories | SKU99 | 68.517833 | 17 | 627 | 9185.185829 | Unknown | 55 | 8 | |

5 rows × 24 columns

In [ ]:
```
#Data Preparation and cleansing
#Loading the file using pandas
```

```
#Leveraging the  infromation about the data & the columns
#Data Cleansing for any missing or incorrect values
```

In [19]:
```
data.columns
```

Out[19]:
```
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', 'State',
       'Lead time', 'Production volumes', 'Manufacturing lead time',
       'Manufacturing costs', 'Inspection results', 'Defect rates',
       'Transportation modes', 'Routes', 'Costs'],
      dtype='object')
```

In [20]:
```
data.shape
```

Out[20]:
```
(100, 24)
```

In [21]:
```
data.describe()
```

Out[21]:

| | Price | Availability | Number of products sold | Revenue generated | Stock levels | Lead times | Order quantities | Shipping times | 1 |
|---|---|---|---|---|---|---|---|---|---|
| **count** | 100.000000 | 100.000000 | 100.000000 | 100.000000 | 100.000000 | 100.000000 | 100.000000 | 100.000000 | |
| **mean** | 49.462461 | 48.400000 | 460.990000 | 5776.048187 | 47.770000 | 15.960000 | 49.220000 | 5.750000 | |
| **std** | 31.168193 | 30.743317 | 303.780074 | 2732.841744 | 31.369372 | 8.785801 | 26.784429 | 2.724283 | |
| **min** | 1.699976 | 1.000000 | 8.000000 | 1061.618523 | 0.000000 | 1.000000 | 1.000000 | 1.000000 | |
| **25%** | 19.597823 | 22.750000 | 184.250000 | 2812.847151 | 16.750000 | 8.000000 | 26.000000 | 3.750000 | |
| **50%** | 51.239830 | 43.500000 | 392.500000 | 6006.352023 | 47.500000 | 17.000000 | 52.000000 | 6.000000 | |
| **75%** | 77.198228 | 75.000000 | 704.250000 | 8253.976920 | 73.000000 | 24.000000 | 71.250000 | 8.000000 | |
| **max** | 99.171329 | 100.000000 | 996.000000 | 9866.465458 | 100.000000 | 30.000000 | 96.000000 | 10.000000 | |

In [22]:
```
data.info()
```

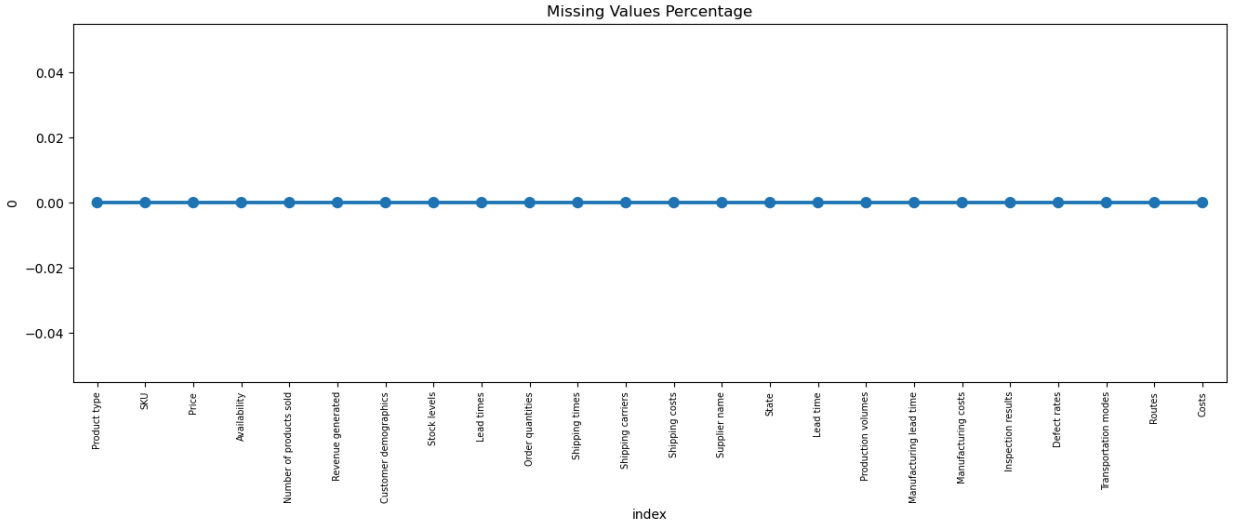
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 24 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   Product type             100 non-null    object 
 1   SKU                      100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

In [28]:
```
#Data Cleansing Missing and Duplicate values

data.isnull().sum()
```

Out[28]:
```
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
State                     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 [ ]:
```
```

In [31]:
```
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd

# Analyzing 'data' is your DataFrame with missing values
plt.figure(figsize=(16, 5))

# Calculating percentage of missing values
missing_values = pd.DataFrame(data.isnull().sum() * 100 / data.shape[0]).reset_index()

# Plot using sns.pointplot
ax = sns.pointplot(x='index', y=0, data=missing_values)

plt.xticks(rotation=90, fontsize=7)
plt.title('Missing Values Percentage')
plt.show()
```




In [32]:
```
#Checking Duplicate Values
len(data[data.duplicated()])
```

Out[32]:
```
0
```

In [33]:
```
#Identifying all unique values for each columns in the Dataset
data.nunique()
```

Out[33]:
```
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
State                        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 [ ]:
```
#Data Visualisation
#Sales Analysis
#Analyze number of products sold and revenue generated to understand sales performance c
#Identify customer demographics to determine which groups are purchasing the most produc
#Track availability and stock levels to ensure the right products are in stock when cust
```

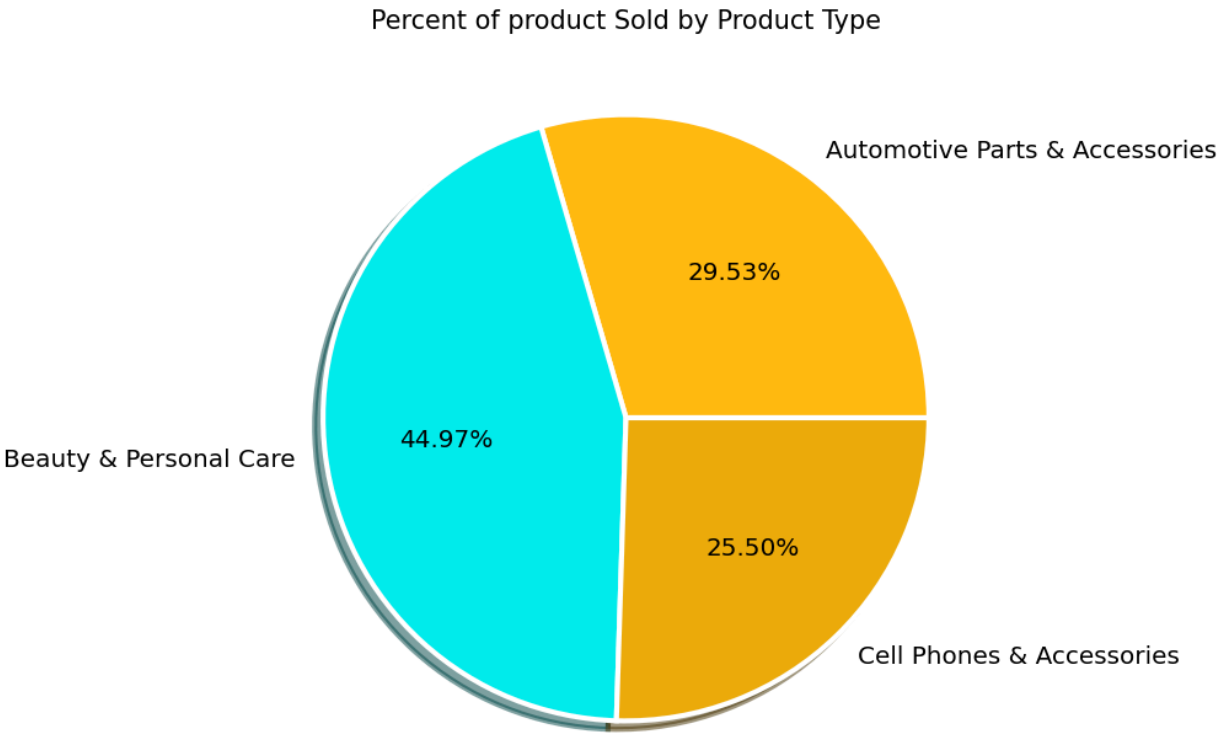
In [34]:
```
product_sold = data.groupby(['Product type'])['Number of products sold', 'Revenue genera
data['Revenue generated'] = data['Revenue generated'].round(2)
```

In [35]:
```
product_sold
```

Out[35]:

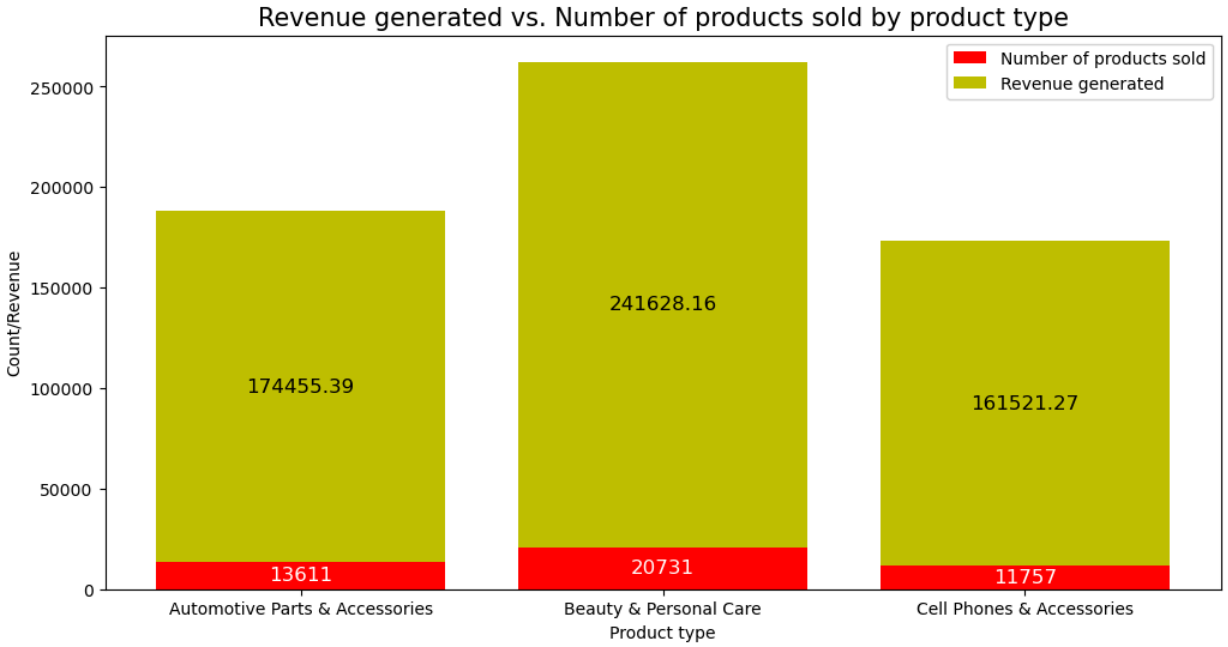
| | Product type | Number of products sold | Revenue generated |
|---|---|---|---|
| **0** | Automotive Parts & Accessories | 13611 | 174455.390606 |
| **1** | Beauty & Personal Care | 20731 | 241628.162133 |
| **2** | Cell Phones & Accessories | 11757 | 161521.266001 |

In [36]:
```
plt.figure(figsize = (12,8))
colors = ['#FFB90F', '#00EEEE', '#EEAD0E']
pie_chart = plt.pie(product_sold['Number of products sold'], labels = product_sold['Proc
       textprops={'size': 'x-large'}, shadow =True, colors = colors)
plt.title('Percent of product Sold by Product Type', fontsize= 15)
plt.show()
```




In [41]:
```
import matplotlib.pyplot as plt

# Assuming 'product_sold' is a dataframe
plt.figure(figsize=(12, 6))
bars1 = plt.bar(x=product_sold['Product type'], height=product_sold["Number of products
bars2 = plt.bar(x=product_sold['Product type'], height=product_sold["Revenue generated"]

plt.title("Revenue generated vs. Number of products sold by product type", fontsize=15)
plt.xlabel("Product type")
plt.ylabel("Count/Revenue")

# Add legend
plt.legend()

# Annotate totals inside the bars
```

```
for i, (bar1, bar2) in enumerate(zip(bars1, bars2)):
    height1 = bar1.get_height()
    height2 = bar2.get_height()
    plt.text(bar1.get_x() + bar1.get_width() / 2, height1 / 2, f'{round(height1, 2)}', h
    plt.text(bar2.get_x() + bar2.get_width() / 2, height1 + height2 / 2, f'{round(height

plt.show()
```




In [ ]:
```
#So, the highest number of products sold of the three product categories is Beauty & Per
#which means 45% of business comes from Beauty & Personal Care, 30% from Automotive, and
```

In [42]:
```
data['Customer demographics'].unique()
```

Out[42]:
```
array(['Non-binary', 'Female', 'Unknown', 'Male'], dtype=object)
```

In [44]:
```
demographics = data.groupby(['Customer demographics', 'Product type'])['Number of produc
```

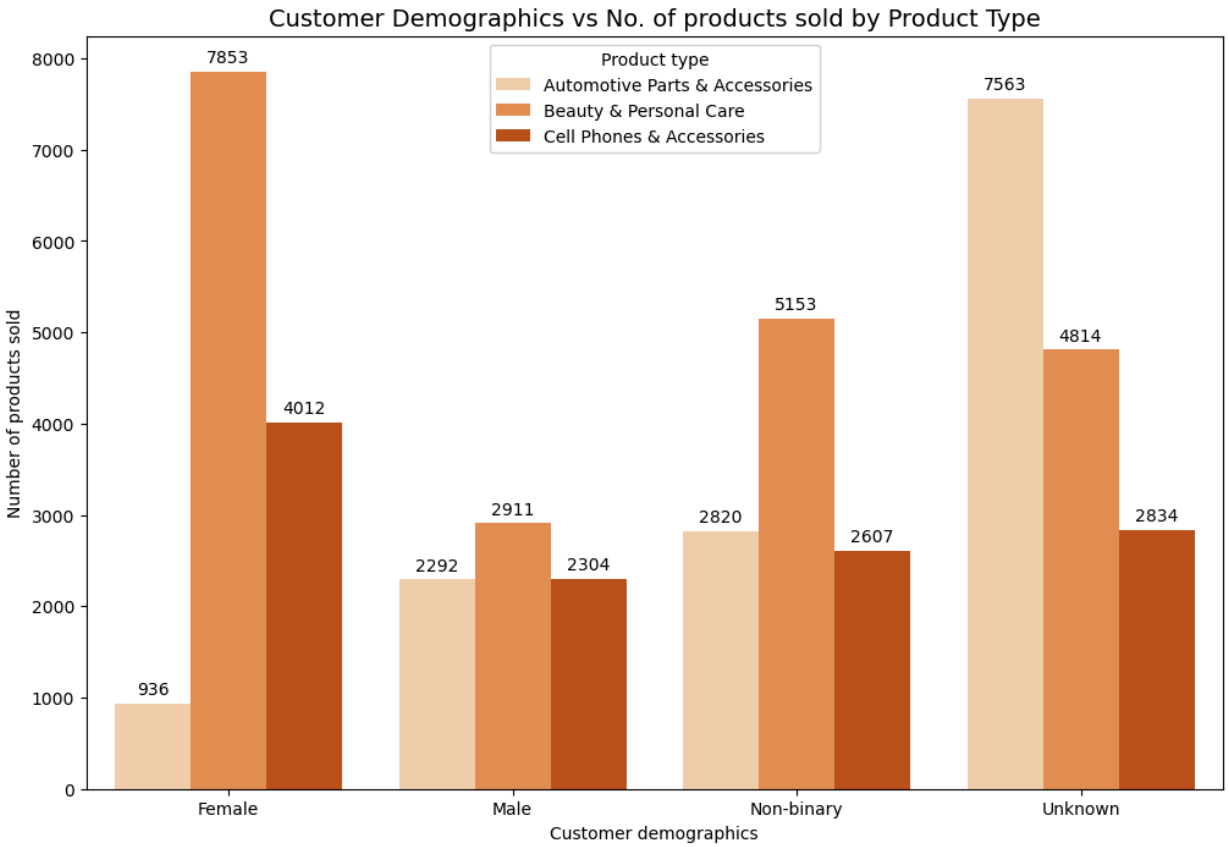
In [45]:
```
demographics
```

Out[45]:

| | Customer demographics | Product type | Number of products sold |
|---|---|---|---|
| 0 | Female | Automotive Parts & Accessories | 936 |
| 1 | Female | Beauty & Personal Care | 7853 |
| 2 | Female | Cell Phones & Accessories | 4012 |
| 3 | Male | Automotive Parts & Accessories | 2292 |
| 4 | Male | Beauty & Personal Care | 2911 |
| 5 | Male | Cell Phones & Accessories | 2304 |
| 6 | Non-binary | Automotive Parts & Accessories | 2820 |
| 7 | Non-binary | Beauty & Personal Care | 5153 |
| 8 | Non-binary | Cell Phones & Accessories | 2607 |
| 9 | Unknown | Automotive Parts & Accessories | 7563 |
| 10 | Unknown | Beauty & Personal Care | 4814 |
| 11 | Unknown | Cell Phones & Accessories | 2834 |

In [55]:

```
#plt.figure(figsize = (12,8))
#p = sns.barplot(x = demographics['Customer demographics'], y = demographics['Number of
#for container in p.containers:
#    p.bar_label(container,padding=-40, color='black', fontsize=10)
#plt.title("Customer Demographics vs No.of product sold by Product Type", fontsize = (14
#plt.show()

import matplotlib.pyplot as plt
import seaborn as sns

# Assuming 'demographics' is a dataframe
plt.figure(figsize=(12, 8))
p = sns.barplot(x=demographics['Customer demographics'], y=demographics['Number of produ

# Display total number of products sold above the bars
for container in p.containers:
    p.bar_label(container, padding=3, color='black', fontsize=10)

plt.title("Customer Demographics vs No. of products sold by Product Type", fontsize=14)
plt.show()
```




In [ ]:
```
#According to the graph, the female group purchases higher-quality Beauty & Personal Ca
#whereas the male group purchases products of about equal quality in terms of Automotive
#And an unknown group category purchases a higher quantity of all three products.
#Beauty & Personal Care products are the most popular among all four product categories.
```

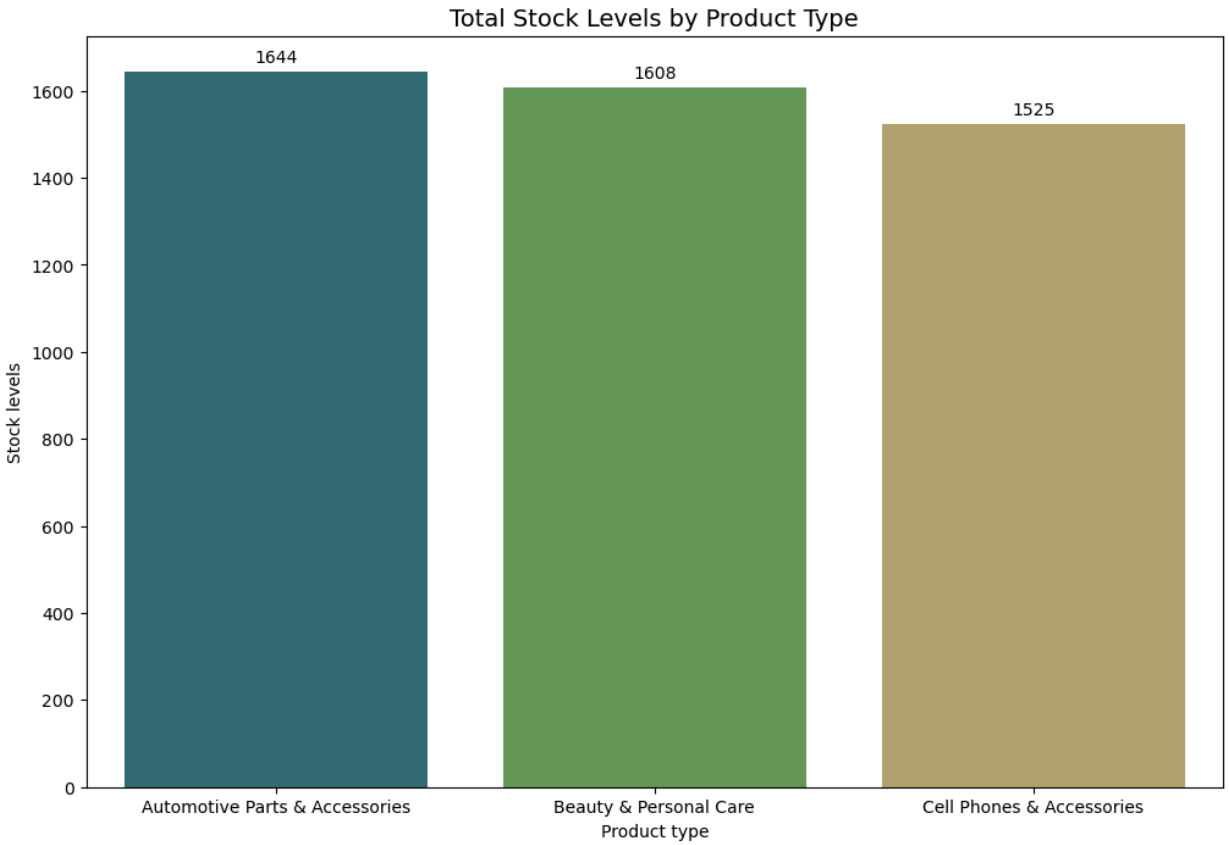
In [50]:
```
stock = data.groupby(['Product type'])['Stock levels','Availability'].sum().reset_index(
stock
```

Out[50]:

| | Product type | Stock levels | Availability |
|---|---|---|---|
| 0 | Automotive Parts & Accessories | 1644 | 1471 |
| 1 | Beauty & Personal Care | 1608 | 2037 |
| 2 | Cell Phones & Accessories | 1525 | 1332 |

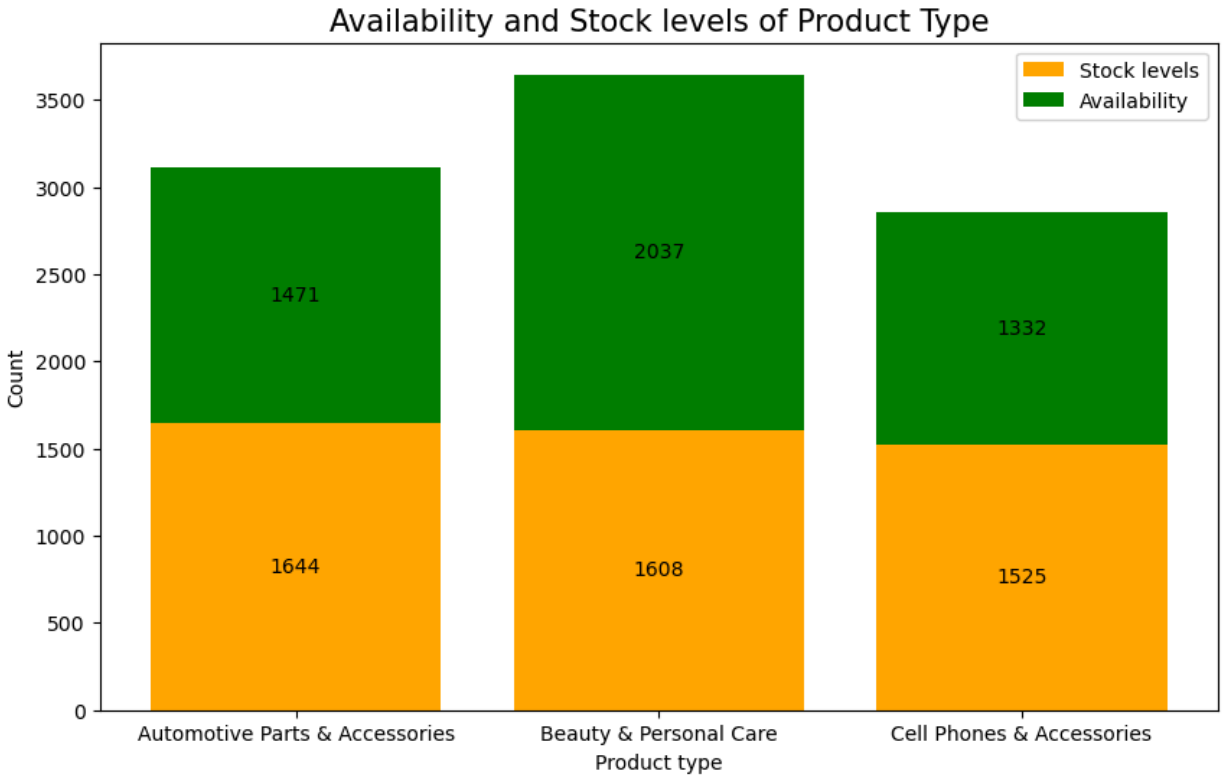
In [56]:
```
#p = sns.barplot(x ='Product type', y =('Stock levels') , data = stock, palette = 'gist_
#for container in p.containers:
#    p.bar_label(container,padding=-40, color='white', fontsize=10)



import matplotlib.pyplot as plt
import seaborn as sns

# Assuming 'stock' is a dataframe
plt.figure(figsize=(12, 8))
p = sns.barplot(x='Product type', y='Stock levels', data=stock, palette='gist_earth')

# Display total stock above the bars
for container in p.containers:
    p.bar_label(container, padding=3, color='black', fontsize=10)  # Adjust padding to p
```

```
plt.title("Total Stock Levels by Product Type", fontsize=14)
plt.show()
```




In [59]:
```
#plt.figure(figsize = (10,6))
#plt.bar(x ='Product type', height = 'Stock levels' , data = stock, color = 'brown')
#plt.bar(x ='Product type', height = 'Availability' , bottom = 'Stock levels' , data = s

#plt.title("Availability and Stock levels of Product Type", fontsize = (15))
#plt.show()



import matplotlib.pyplot as plt

# Assuming 'stock' is a dataframe
plt.figure(figsize=(10, 6))

# Plot the stock levels bar
bars1 = plt.bar(x=stock['Product type'], height=stock['Stock levels'], color='orange', l

# Plot the availability bar stacked on top of stock levels
bars2 = plt.bar(x=stock['Product type'], height=stock['Availability'], bottom=stock['Sto

plt.title("Availability and Stock levels of Product Type", fontsize=15)
plt.xlabel("Product type")
plt.ylabel("Count")

# Add legend
plt.legend()

# Annotate totals inside the bars
```

```
for i, (bar1, bar2) in enumerate(zip(bars1, bars2)):
    height1 = bar1.get_height()
    height2 = bar2.get_height()
    plt.text(bar1.get_x() + bar1.get_width() / 2, height1 / 2, f'{int(height1)}', ha='ce
    plt.text(bar2.get_x() + bar2.get_width() / 2, height1 + height2 / 2, f'{int(height2)

plt.show()
```




In [60]:
```
data.groupby(['Product type'])['Stock levels','Availability'].sum().reset_index()
```

Out[60]:

| | Product type | Stock levels | Availability |
|---|---|---|---|
| **0** | Automotive Parts & Accessories | 1644 | 1471 |
| **1** | Beauty & Personal Care | 1608 | 2037 |
| **2** | Cell Phones & Accessories | 1525 | 1332 |

In [ ]:
```
#In the graph, green represents the availability and orange represents the stock levels.
#So according to the graph, the company holds an high quantity of availability of Beauty
#Automotive products are less and a bit less stock of Cell Phone Accesories.
#So, Automotive products had a lower availability and higher stock level, which means we
#manufacture and ship products as needed. On the other hand,
#Beauty and Cell Phone Accesiores have a somehow resonable stock level and  availability
# has to gather raw material for processing and take time to ship product to the custome
```

In [ ]:
```
#Operations Analysis:
#1.Analyze lead times, order quantities, and production volumes to optimize inventory ma
#2.Track manufacturing lead time and costs to identify areas for improvement and cost sa
#3.Monitor inspection results and defect rates to identify quality issues and improve ma
```

In [61]:
```
data.columns
```

Out[61]:
```
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', 'State',
       'Lead time', 'Production volumes', 'Manufacturing lead time',
       'Manufacturing costs', 'Inspection results', 'Defect rates',
       'Transportation modes', 'Routes', 'Costs'],
      dtype='object')
```

In [62]:
```
product = data.groupby(['Product type'])['Lead time', 'Order quantities', 'Production vo
product['Order quantities'] = product['Order quantities'].round(2)
product['Lead time'] = product['Lead time'].round(2)
product['Production volumes'] = product['Production volumes'].round(2)
```

In [63]:
```
product
```

Out[63]:

| | Product type | Lead time | Order quantities | Production volumes |
|---|---|---|---|---|
| **0** | Automotive Parts & Accessories | 18.71 | 43.53 | 586.97 |
| **1** | Beauty & Personal Care | 18.00 | 52.48 | 609.15 |
| **2** | Cell Phones & Accessories | 13.54 | 51.65 | 479.27 |

In [ ]:
```
#Beauty & Personal Care products have higher order quantities and a longer lead time. Fu
#it has a higher production volume (production volume means the amount of products that
#which means higher production volumes may require longer lead times to ensure that ther
#the products and meet customer demand.
#Automotive Parts & Accessories products have a longer lead time and higher production v
```

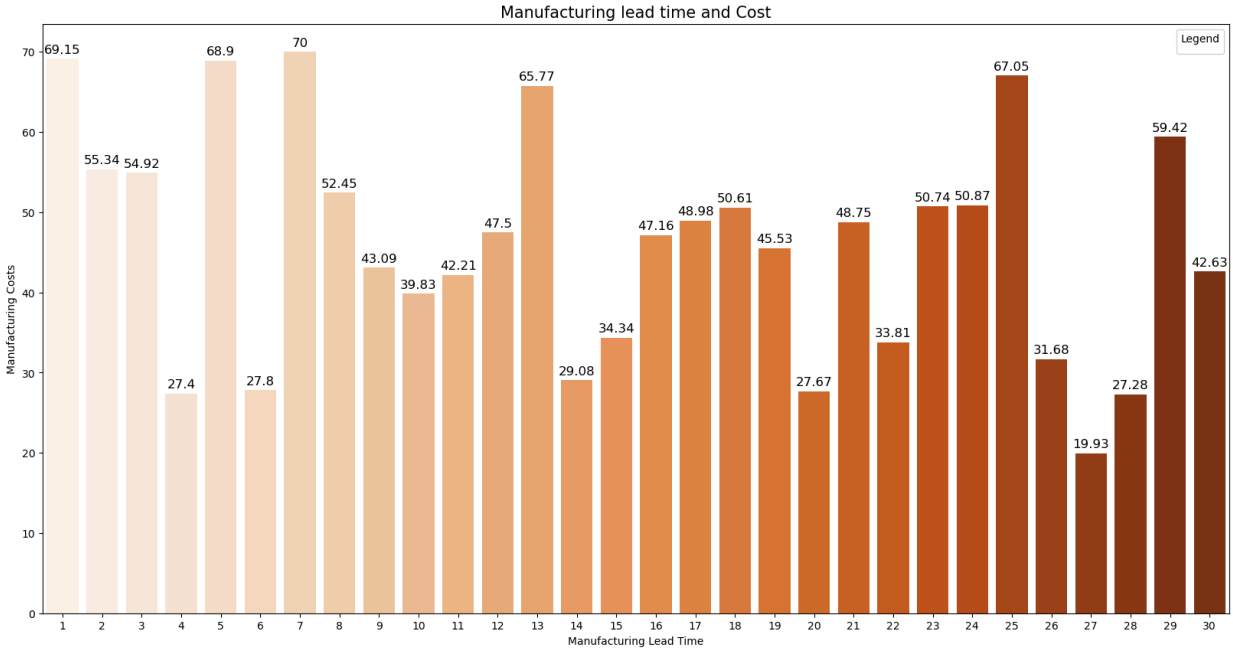
In [65]:
```
avg_costs = data.groupby(['Manufacturing lead time'])['Manufacturing costs'].mean().rese
avg_costs['Manufacturing costs'] = avg_costs['Manufacturing costs'].round(2)

avg_costs
```

Out[65]:

| | Manufacturing lead time | Manufacturing costs |
|---|---|---|
| **26** | 27 | 19.93 |
| **27** | 28 | 27.28 |
| **3** | 4 | 27.40 |
| **19** | 20 | 27.67 |
| **5** | 6 | 27.80 |
| **13** | 14 | 29.08 |
| **25** | 26 | 31.68 |
| **21** | 22 | 33.81 |
| **14** | 15 | 34.34 |
| **9** | 10 | 39.83 |
| **10** | 11 | 42.21 |
| **29** | 30 | 42.63 |
| **8** | 9 | 43.09 |
| **18** | 19 | 45.53 |
| **15** | 16 | 47.16 |
| **11** | 12 | 47.50 |
| **20** | 21 | 48.75 |
| **16** | 17 | 48.98 |
| **17** | 18 | 50.61 |
| **22** | 23 | 50.74 |
| **23** | 24 | 50.87 |
| **7** | 8 | 52.45 |
| **2** | 3 | 54.92 |
| **1** | 2 | 55.34 |
| **28** | 29 | 59.42 |
| **12** | 13 | 65.77 |
| **24** | 25 | 67.05 |
| **4** | 5 | 68.90 |
| **0** | 1 | 69.15 |
| **6** | 7 | 70.00 |

In [70]:
```
import matplotlib.pyplot as plt
import seaborn as sns

# Assuming 'avg_costs' is a dataframe
plt.figure(figsize=(20, 10))

# Create the barplot with orange color
```

```
p = sns.barplot(x=avg_costs['Manufacturing lead time'], y=avg_costs['Manufacturing costs

# Display the number above each bar
for container in p.containers:
    p.bar_label(container, padding=3, color='black', fontsize=12)  # Adjust padding to p

# Add title and legend with appropriate labels
plt.title('Manufacturing lead time and Cost', fontsize=15)
plt.xlabel('Manufacturing Lead Time')
plt.ylabel('Manufacturing Costs')

# Customize legend labels and title
handles, labels = p.get_legend_handles_labels()
plt.legend(handles, ['Manufacturing Lead Time', 'Manufacturing Costs'], title='Legend')

plt.show()
```




In [71]:
```
rate = data.groupby(['Product type', 'Inspection results'])['Defect rates'].mean().reset
rate['Defect rates'] = rate['Defect rates'].round(2)
```

In [72]:
```
rate
```

Out[72]:

| | Product type | Inspection results | Defect rates |
|---|---|---|---|
| **0** | Automotive Parts & Accessories | Fail | 2.53 |
| **1** | Automotive Parts & Accessories | Pass | 2.92 |
| **2** | Automotive Parts & Accessories | Pending | 2.27 |
| **3** | Beauty & Personal Care | Fail | 2.90 |
| **4** | Beauty & Personal Care | Pass | 1.68 |
| **5** | Beauty & Personal Care | Pending | 2.33 |
| **6** | Cell Phones & Accessories | Fail | 2.19 |
| **7** | Cell Phones & Accessories | Pass | 1.82 |
| **8** | Cell Phones & Accessories | Pending | 1.71 |

In [73]:
```
data['Defect rates'].mean()
```

Out[73]:
```
2.2771579927400003
```

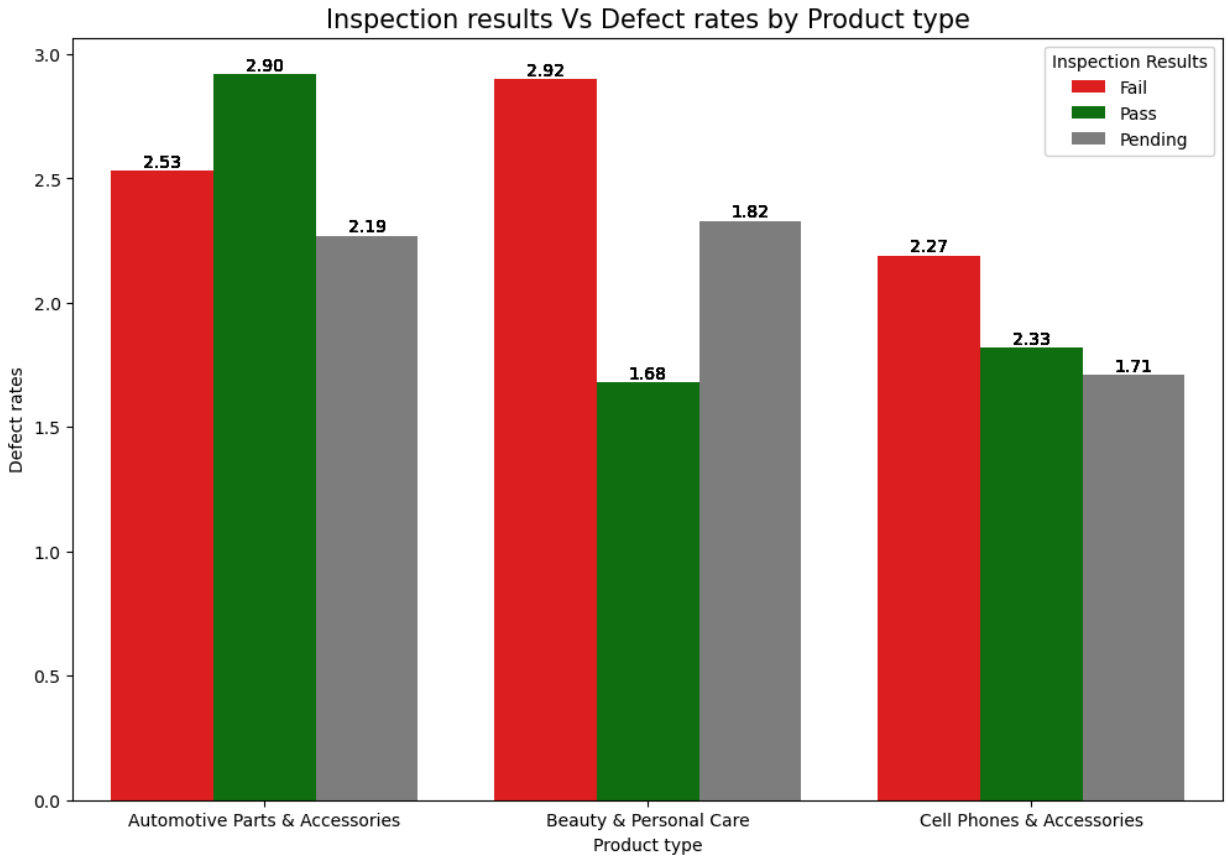
In [74]:
```
data['Defect rates'].max()
```

Out[74]:
```
4.939255289
```

In [75]:
```
data['Defect rates'].min()
```

Out[75]:
```
0.018607568
```

In [80]:
```
import matplotlib.pyplot as plt
import seaborn as sns

# Assuming 'rate' is a dataframe
plt.figure(figsize=(12, 8))

# Define custom colors for the legend categories
custom_palette = {'Fail': 'red', 'Pending': 'grey', 'Pass': 'green'}

# Create the barplot with custom colors
p = sns.barplot(x=rate['Product type'], y=rate['Defect rates'], hue=rate['Inspection res

# Add labels on top of each bar
for index, row in rate.iterrows():
    for bar, label in zip(p.patches, rate['Defect rates']):
        p.annotate(f'{label:.2f}', (bar.get_x() + bar.get_width() / 2, bar.get_height())

# Add title and labels
plt.title("Inspection results Vs Defect rates by Product type", fontsize=15)
plt.xlabel("Product type")
plt.ylabel("Defect rates")

# Show the plot with updated legend colors
plt.legend(title='Inspection Results', loc='upper right')

plt.show()
```




In [ ]:
```
All product categories have a resonable higher defect rates.
```

In [ ]:
```
#Shipping Analysis:
#Analyze costs, transportation modes, and routes to optimize logistics and reduce shipp
#Monitor shipping times, shipping carriers, modes of transportation to ensure timely de
#Track shipping costs associated with shipping carriers and revenue generated to identi
```

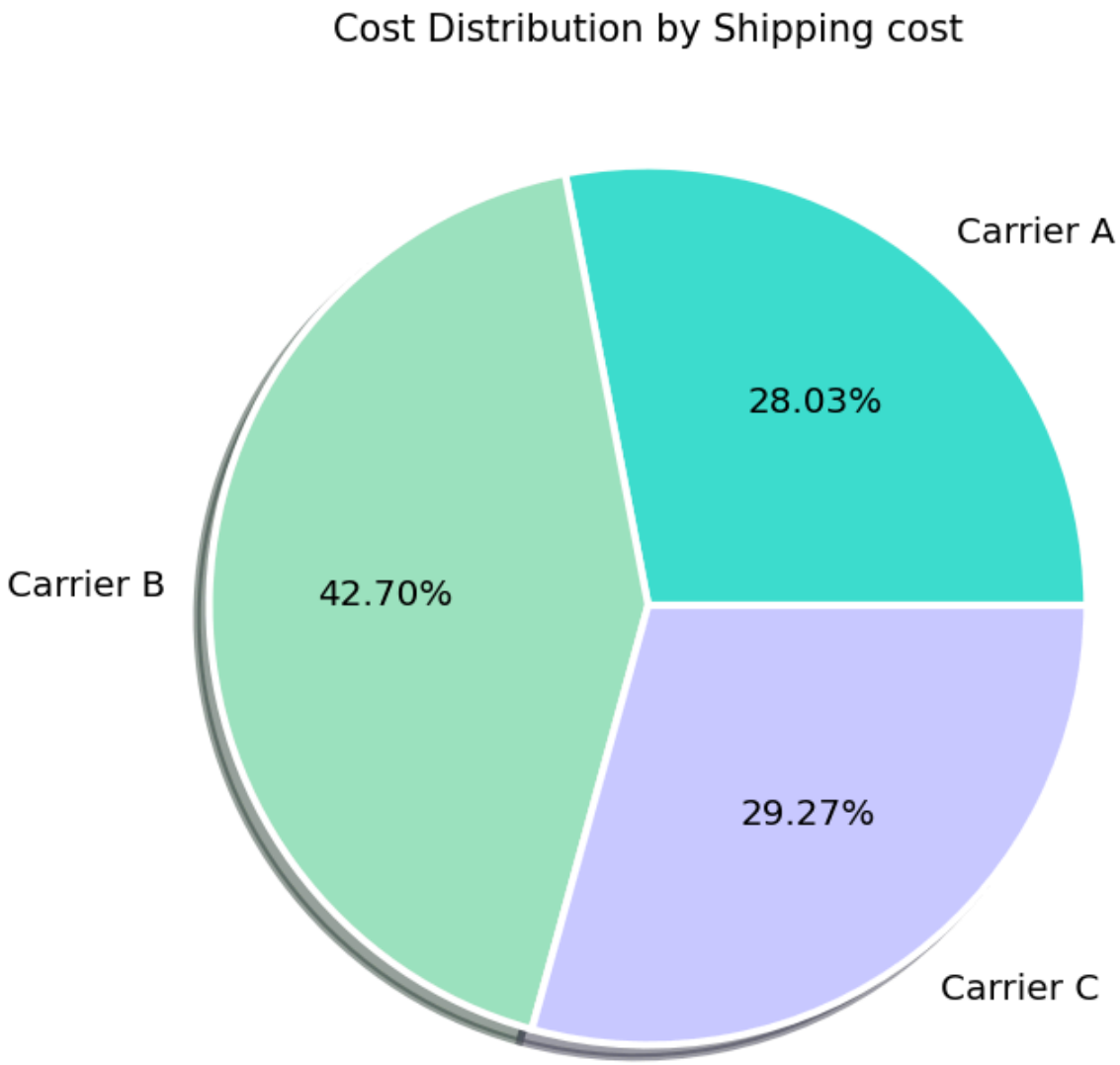
In [81]:
```
shipping = data.groupby(['Shipping carriers'])['Shipping costs'].sum().reset_index()
```

In [82]:
```
shipping
```

Out[82]:

| | Shipping carriers | Shipping costs |
|---|---|---|
| **0** | Carrier A | 155.537831 |
| **1** | Carrier B | 236.897620 |
| **2** | Carrier C | 162.379457 |

In [83]:
```
plt.figure(figsize = (12,8))
colors = ['#40E0D0', '#9FE2BF','#CCCCFF']
plt.pie( shipping['Shipping costs'], labels = shipping['Shipping carriers'],autopct = '%
       textprops={'size': 'x-large'}, shadow =True, colors = colors)
plt.title('Cost Distribution by Shipping cost', fontsize = (15))
plt.show()
```




In [ ]:
```
#Shipping of Carrier A is Airways, Carrier B is Roadways and Carrier C is Seaways.
```

In [84]:
```
carrier_revenue = data.groupby(['Shipping carriers'])['Revenue generated'].sum().reset_
carrier_revenue['Revenue generated'] = carrier_revenue['Revenue generated'].round(2)
```

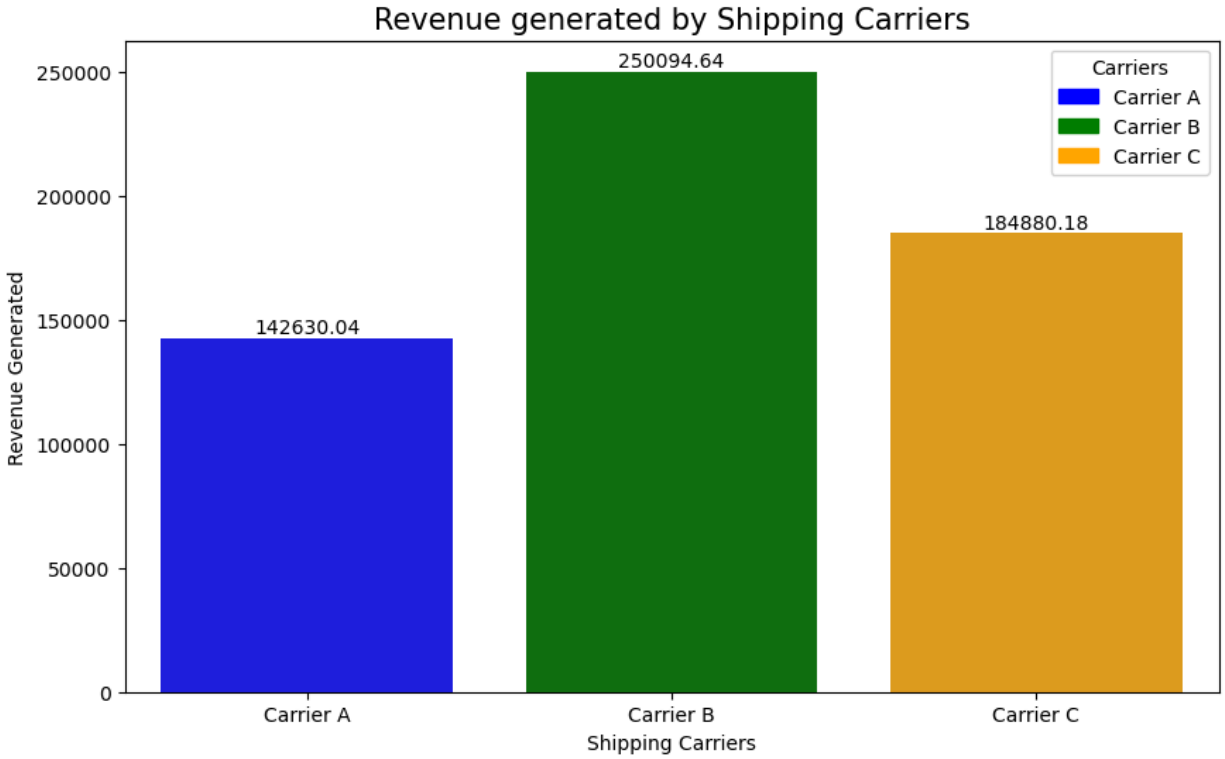
In [85]:
```
carrier_revenue
```

Out[85]:

| | Shipping carriers | Revenue generated |
|---|---|---|
| **0** | Carrier A | 142630.04 |
| **1** | Carrier B | 250094.64 |
| **2** | Carrier C | 184880.18 |

In [87]:
```
#plt.figure(figsize = (10,6))
#p = sns.barplot(x = carrier_revenue['Shipping carriers'], y = carrier_revenue['Revenue
#for container in p.containers:
#    p.bar_label(container,padding=-40, color='black', fontsize=10)

#plt.show()
```

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

# Assuming 'carrier_revenue' is a dataframe
plt.figure(figsize=(10, 6))

# Define custom colors for each carrier
custom_palette = {'Carrier A': 'blue', 'Carrier B': 'green', 'Carrier C': 'orange'}

# Create the barplot with custom colors
p = sns.barplot(x=carrier_revenue['Shipping carriers'], y=carrier_revenue['Revenue gener

# Display the total revenue on top of each bar
for container in p.containers:
    for bar, label in zip(p.patches, carrier_revenue['Revenue generated']):
        p.annotate(f'{label:.2f}', (bar.get_x() + bar.get_width() / 2, bar.get_height())

# Add title and labels
plt.title("Revenue generated by Shipping Carriers", fontsize=15)
plt.xlabel("Shipping Carriers")
plt.ylabel("Revenue Generated")

# Create custom legend with specified colors
legend_labels = ['Carrier A', 'Carrier B', 'Carrier C']
legend_handles = [plt.Rectangle((0,0),1,1, color=custom_palette[label]) for label in leg
plt.legend(legend_handles, legend_labels, title='Carriers')

plt.show()
```




In [ ]:
```
#Analyzing the graphs clearly show shipping carrier B is costly as well as generating ht
```

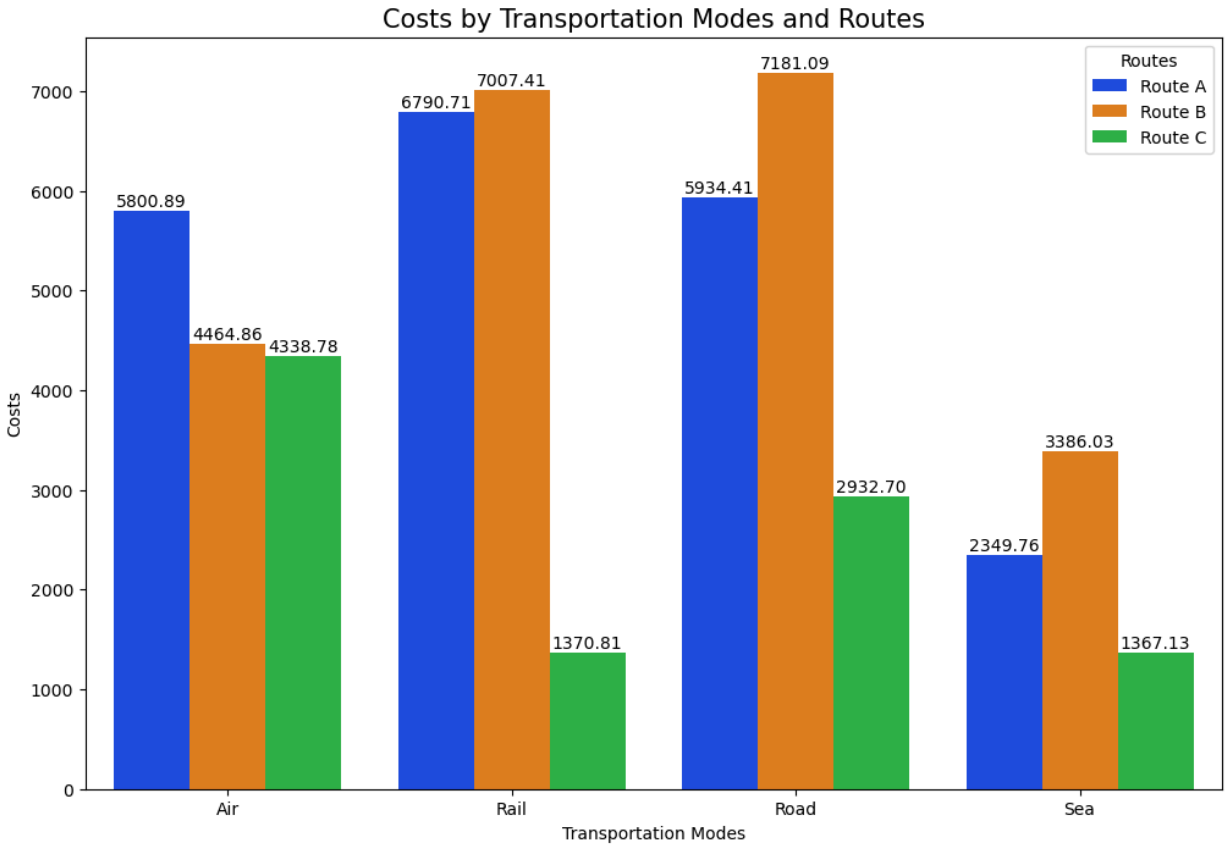
In [88]:
```
transport = data.groupby(['Transportation modes', 'Routes'])['Costs'].sum().reset_index(
```

In [89]:
```
transport
```

Out[89]:

| | Transportation modes | Routes | Costs |
|---|---|---|---|
| **0** | Air | Route A | 5800.887460 |
| **1** | Air | Route B | 4464.858025 |
| **2** | Air | Route C | 4338.782012 |
| **3** | Rail | Route A | 6790.710511 |
| **4** | Rail | Route B | 7007.410741 |
| **5** | Rail | Route C | 1370.810306 |
| **6** | Road | Route A | 5934.412107 |
| **7** | Road | Route B | 7181.085146 |
| **8** | Road | Route C | 2932.696385 |
| **9** | Sea | Route A | 2349.764416 |
| **10** | Sea | Route B | 3386.030113 |
| **11** | Sea | Route C | 1367.130992 |

In [103]:
```
import matplotlib.pyplot as plt
import seaborn as sns

# Assuming 'transport' is a dataframe
plt.figure(figsize=(12, 8))

# Create the barplot with a bright palette
p = sns.barplot(x=transport['Transportation modes'], y=transport['Costs'], hue=transport

# Add total cost above each bar
for patch in p.patches:
    height = patch.get_height()
    p.annotate(f'{height:.2f}', (patch.get_x() + patch.get_width() / 2, height + 5), ha=

# Add title and labels
plt.title("Costs by Transportation Modes and Routes", fontsize=15)
plt.xlabel("Transportation Modes")
plt.ylabel("Costs")

# Show the plot
plt.show()
```

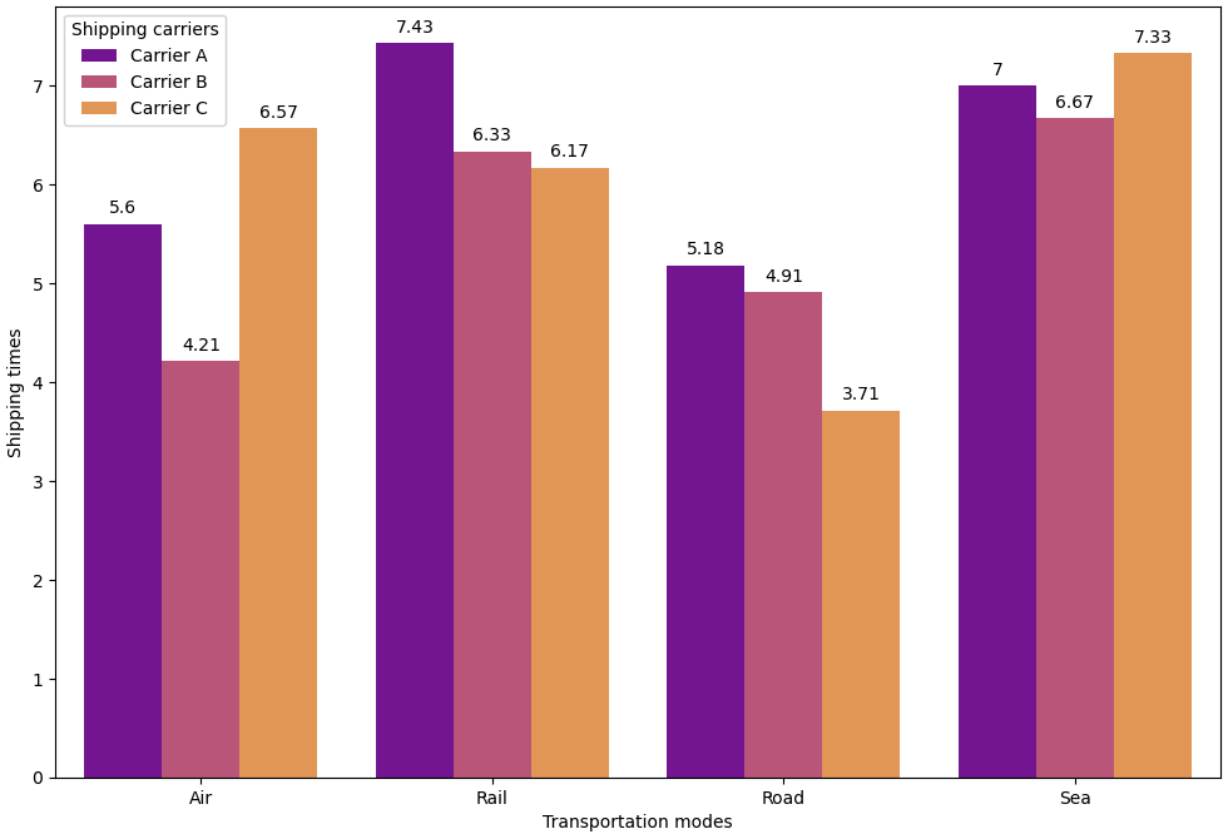



In [104]:
```
shipping = data.groupby(['Shipping carriers', 'Transportation modes'])['Shipping times']
shipping['Shipping times'] = shipping['Shipping times'].round(2)
shipping
```

Out[104]:

| | Shipping carriers | Transportation modes | Shipping times |
|---|---|---|---|
| **0** | Carrier A | Air | 5.60 |
| **1** | Carrier A | Rail | 7.43 |
| **2** | Carrier A | Road | 5.18 |
| **3** | Carrier A | Sea | 7.00 |
| **4** | Carrier B | Air | 4.21 |
| **5** | Carrier B | Rail | 6.33 |
| **6** | Carrier B | Road | 4.91 |
| **7** | Carrier B | Sea | 6.67 |
| **8** | Carrier C | Air | 6.57 |
| **9** | Carrier C | Rail | 6.17 |
| **10** | Carrier C | Road | 3.71 |
| **11** | Carrier C | Sea | 7.33 |

In [120]:
```
plt.figure(figsize = (12,8))
p = sns.barplot(x = shipping['Transportation modes'], y = shipping['Shipping times'], hu
for container in p.containers:
    p.bar_label(container,padding=4.1, color='black', fontsize=10)
plt.show()
```




In [ ]:
```
#According to the graph, the fastest and most efficient shipping option is Carrier B in
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: