SUPPLY CHAIN MANAGEMENT

IMPORTING LIBRARIES

In [3]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.graph_objects as go
```

IMPORTING DATA SET

In [4]:
```
df = pd.read_csv('C:\\Users\\Gowthami Galla\\Desktop\\supply_chain_data.csv')
print(df.head())
```

```
  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]
```

About Dataset

In [7]:
```
print(df.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  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
```

In [188…

```python
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
```

DATA PREPROCESSING : Cleaning Missing Values, Removing Duplicates, and Standardizing Formats

In [190…

```python
# Drop rows with missing values (if it's minimal)
df.dropna(inplace=True)

# OR fill missing values (for numerical columns, with mean or median)
# df['Price'].fillna(df['Price'].mean(), inplace=True)

# Verify if there are any remaining missing values
print(df.isnull().sum())
```

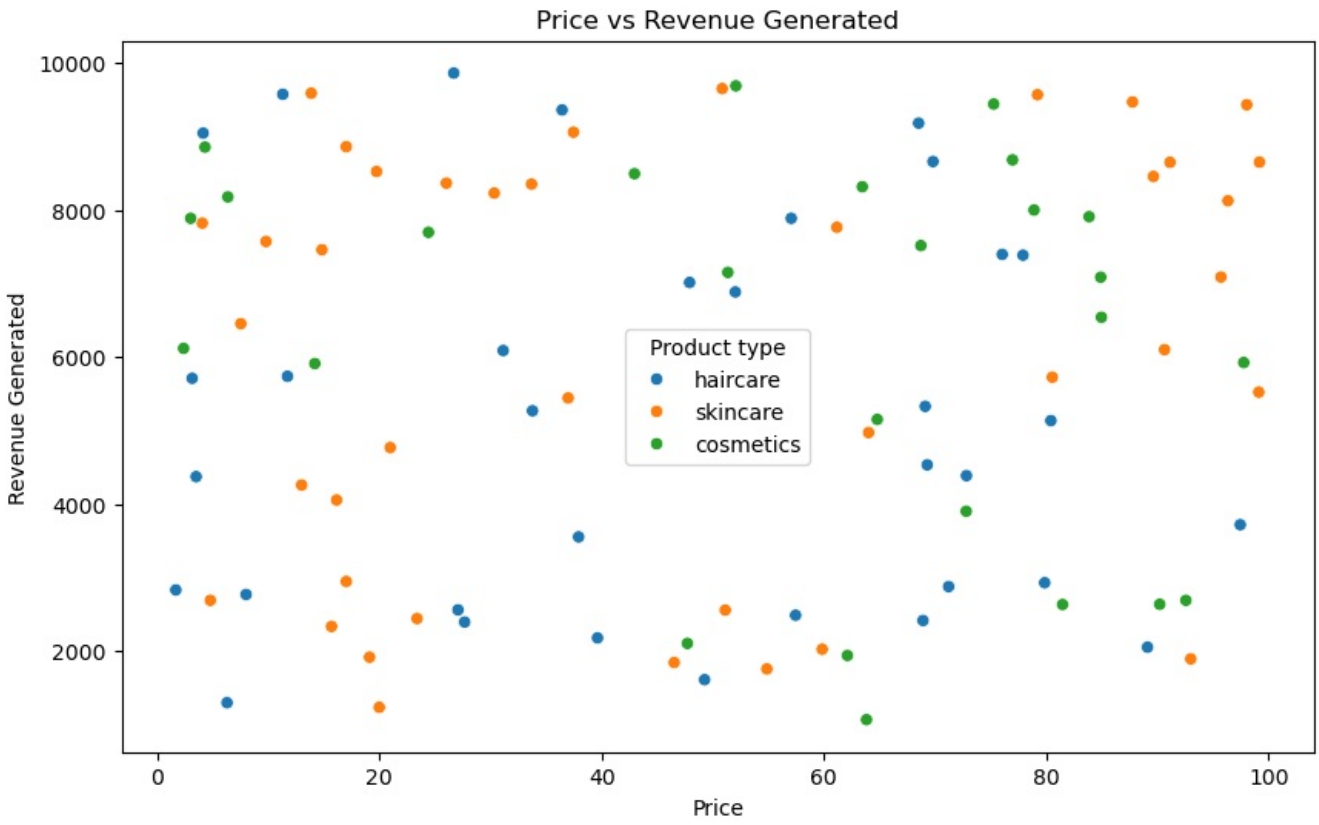
```
Product type              0
SKU                       0
Price                     0
Availability              0
Number of products sold   0
Revenue generated         0
Customer demographics     0
Stock levels              0
Lead times                0
Order quantities          0
Shipping times            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
```

EXPLORATORY DATA ANALYSIS(EDA)

Price vs Revenue Generated

In [192…

```python
# Price vs Revenue Generated
plt.figure(figsize=(10, 6))
sns.scatterplot(x=df['Price'], y=df['Revenue generated'], hue=df['Product type'])
plt.title('Price vs Revenue Generated')
plt.xlabel('Price')
plt.ylabel('Revenue Generated')
plt.show()
```



Total revenue generated

In [194…

```python
# Calculate total revenue generated
total_value_generated = df['Revenue generated'].sum()

print(f'Total Value Generated: {total_value_generated}')
```

```
Total Value Generated: 577604.8187399999
```

Total Order Quantity

In [196…
```
# Calculate total order quantity
total_order_quantity = df['Order quantities'].sum()
print(f'Total Order Quantity: {total_order_quantity}')
```

```
Total Order Quantity: 4922
```

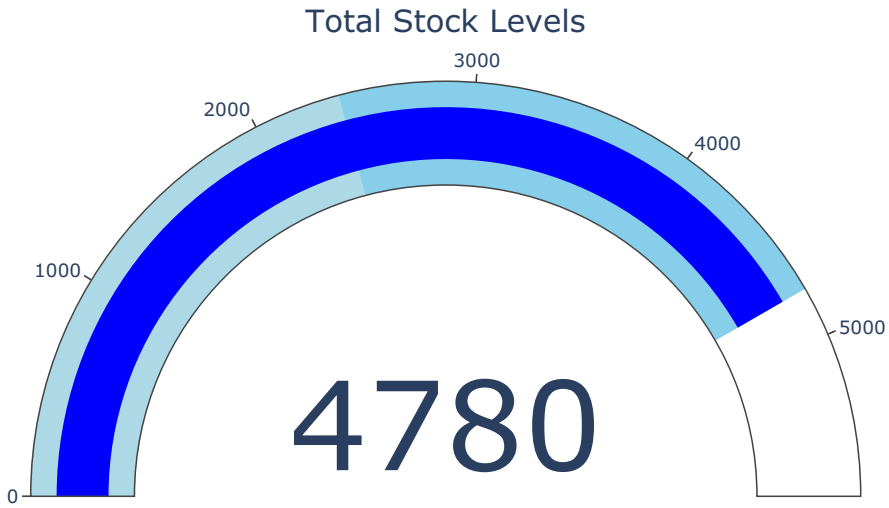
Total Availability

In [198…
```
# Calculate total availability
total_availability = df['Availability'].sum()

print(f'Total Availability: {total_availability}')
```

```
Total Availability: 4840
```

Total Stock Levels

In [178…
```
total_stock_levels = df['Stock levels'].sum()

# Create the gauge chart
fig = go.Figure(go.Indicator(
    mode="gauge+number",
    value=total_stock_levels,
    title={'text': "Total Stock Levels"},
    gauge={
        'axis': {'range': [0, total_stock_levels * 1.2]},  # Adjust range for visual clarity
        'bar': {'color': "blue"},
        'steps': [
            {'range': [0, total_stock_levels * 0.5], 'color': "lightblue"},
            {'range': [total_stock_levels * 0.5, total_stock_levels], 'color': "skyblue"},
        ]
    }
))

# Show the chart
fig.show()
```



Total Lead Times

In [206…
```
# Verify the column name for lead times
if 'Lead times' in df.columns:
    total_lead_times = df['Lead times'].sum()
elif 'Lead Time' in df.columns:  # Alternative column name
    total_lead_times = df['Lead Time'].sum()
```

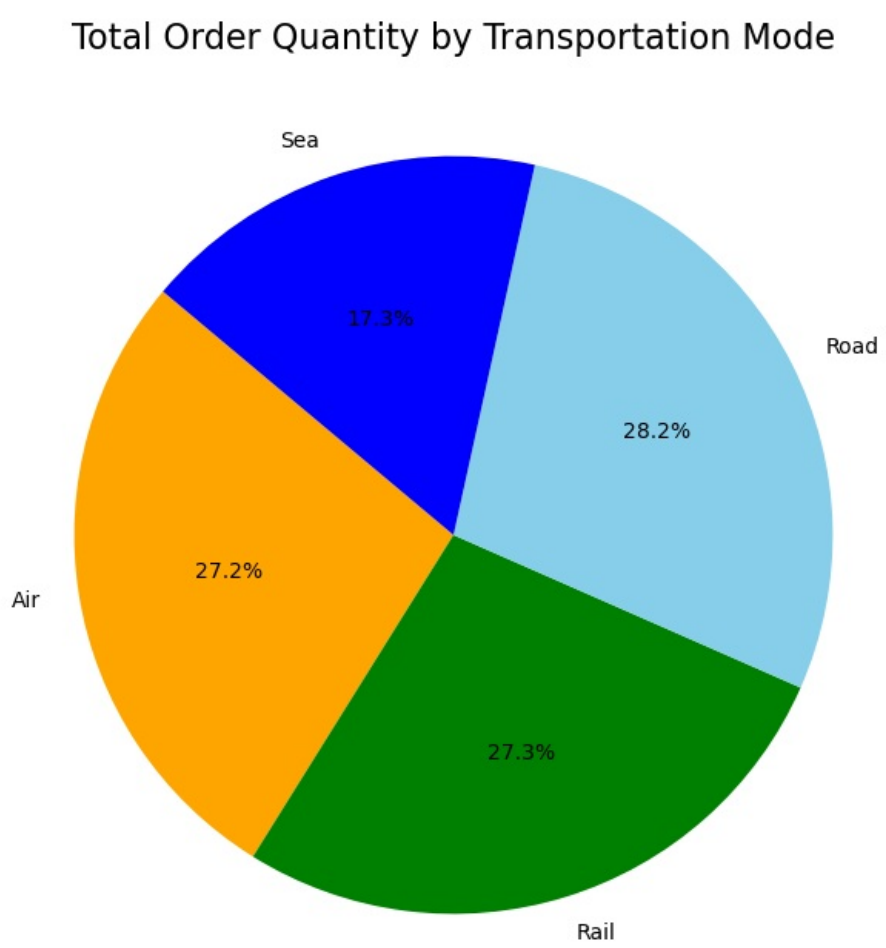
```
else:
    raise KeyError("Column 'Lead times' or 'Lead Time' not found in the dataset.")

print(f"Total Lead Times: {total_lead_times}")

# Create a gauge chart for total lead times
import plotly.graph_objects as go

fig = go.Figure(go.Indicator(
    mode="gauge+number",
    value=total_lead_times,
    title={'text': "Total Lead Times"},
    gauge={
        'axis': {'range': [0, total_lead_times * 1.2]},  # Adjust range
        'bar': {'color': "green"},
        'steps': [
            {'range': [0, total_lead_times * 0.5], 'color': "lightgreen"},
            {'range': [total_lead_times * 0.5, total_lead_times], 'color': "darkgreen"},
        ]
    }
))

fig.show()
```

```
Total Lead Times: 1596
```

Total Order Quantity By Transportation Mode

In [216…

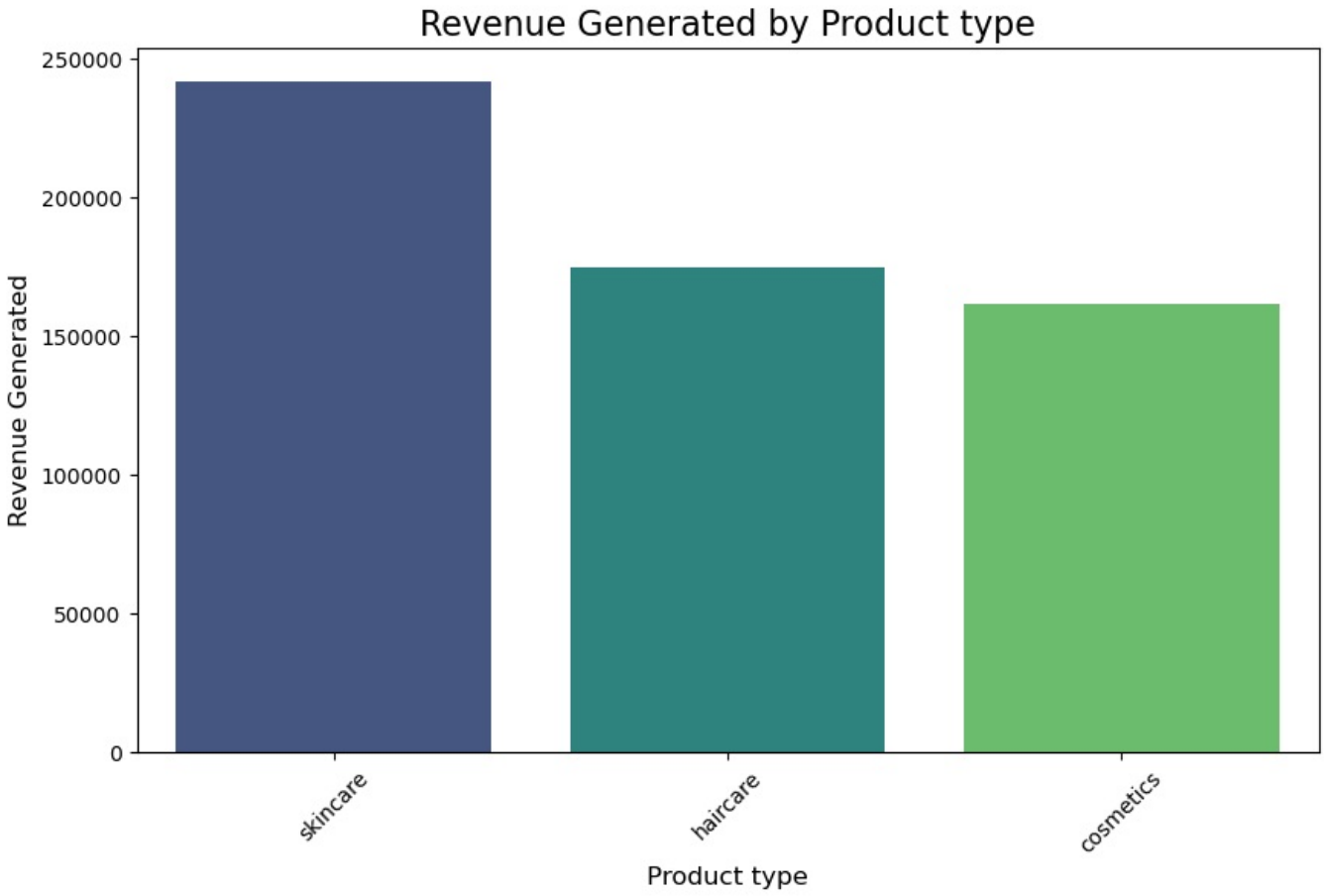
```
order_quantity_by_transportation = df.groupby('Transportation modes')['Order quantities'].sum()

# Define custom colors
custom_colors = ['orange', 'green', 'skyblue', 'blue']

# Plot the pie chart
plt.figure(figsize=(8, 8))
plt.pie(
    order_quantity_by_transportation,
    labels=order_quantity_by_transportation.index,
    autopct='%1.1f%%',
    startangle=140,
    colors=custom_colors[:len(order_quantity_by_transportation)]  # Use only as many colors as needed
)

# Add a title
plt.title('Total Order Quantity by Transportation Mode', fontsize=16)

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



Revenue Generated by Product type

In [144…

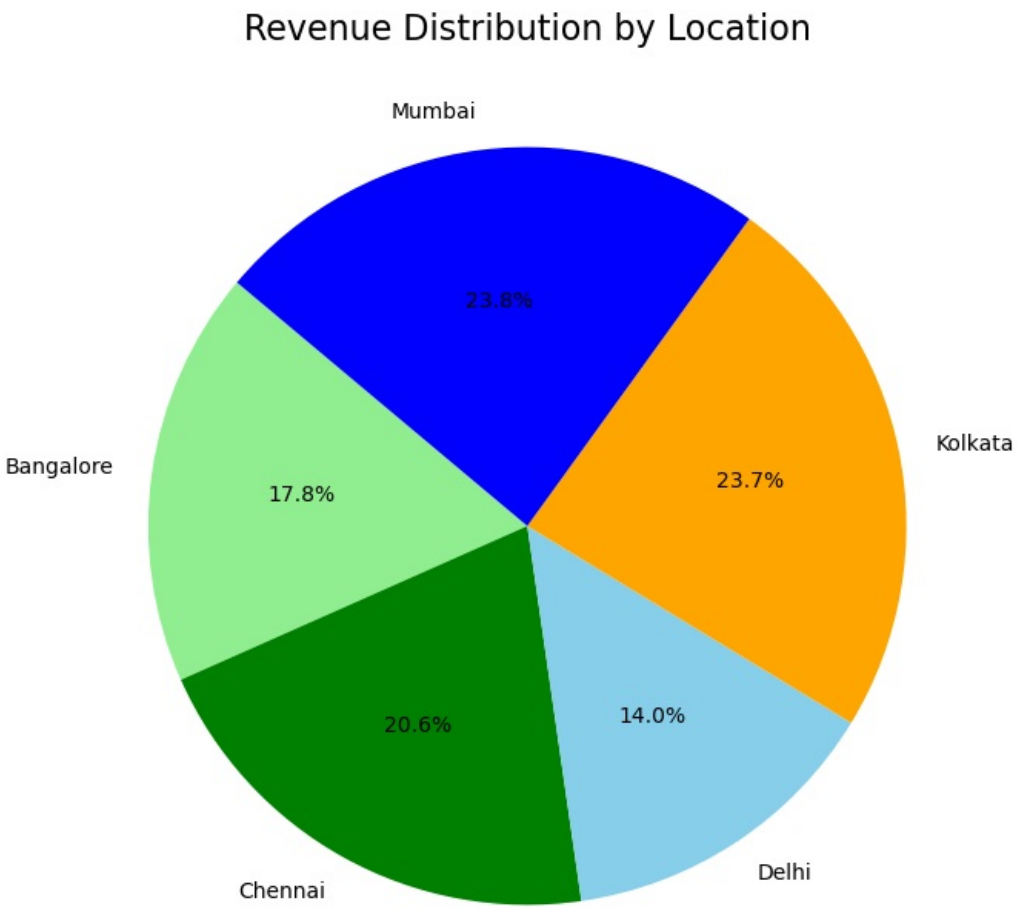
```
revenue_by_product = df.groupby('Product type')['Revenue generated'].sum().reset_index()

# Sort by revenue for better visualization
revenue_by_product = revenue_by_product.sort_values(by='Revenue generated', ascending=False)

# Plot the bar chart
plt.figure(figsize=(10, 6))
sns.barplot(
    data=revenue_by_product,
    x='Product type',
    y='Revenue generated',
    hue='Product type',  # Assign `x` variable to `hue`
    dodge=False,  # Avoid offsetting bars
    palette='viridis',  # Use the palette
    legend=False  # Disable legend
)

# Add labels and title
plt.title('Revenue Generated by Product type', fontsize=16)
plt.xlabel('Product type', fontsize=12)
plt.ylabel('Revenue Generated', fontsize=12)

# Rotate x-axis labels for better readability (if needed)
plt.xticks(rotation=45)

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



Revenue Distribution by Location

In [164…

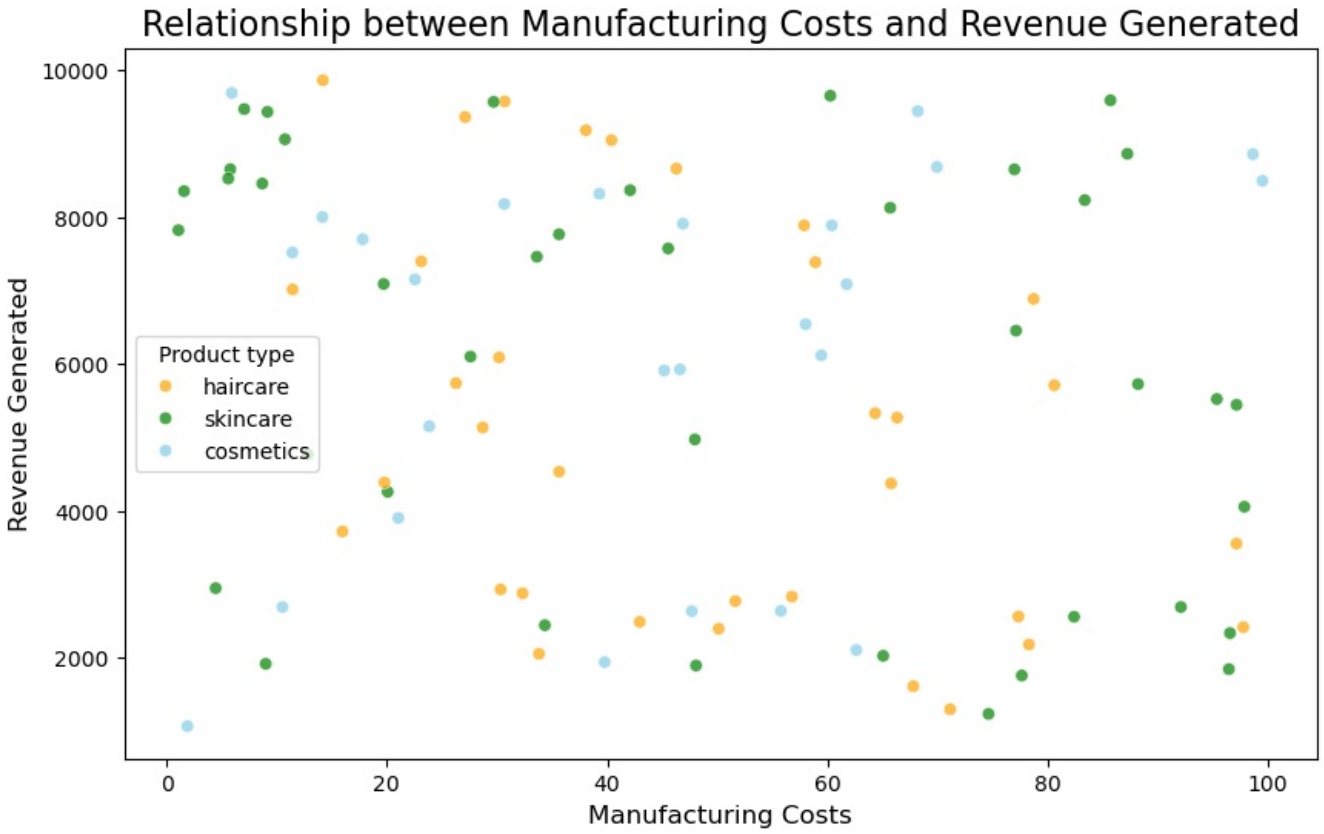
```
# Grouping the data by 'Location' and summing up the revenue
revenue_by_location = df.groupby('Location')['Revenue generated'].sum()

# Plotting the pie chart
plt.figure(figsize=(10, 8))
colors = ['lightgreen', 'green', 'skyblue', 'orange', 'blue']
plt.pie(
    revenue_by_location,
    labels=revenue_by_location.index,
    autopct='%1.1f%%',
    startangle=140,
    colors=colors
)

# Adding a title
plt.title('Revenue Distribution by Location', fontsize=16)
plt.show()
```



Relationship between Manufacturing Costs and Revenue Generated

In [234…

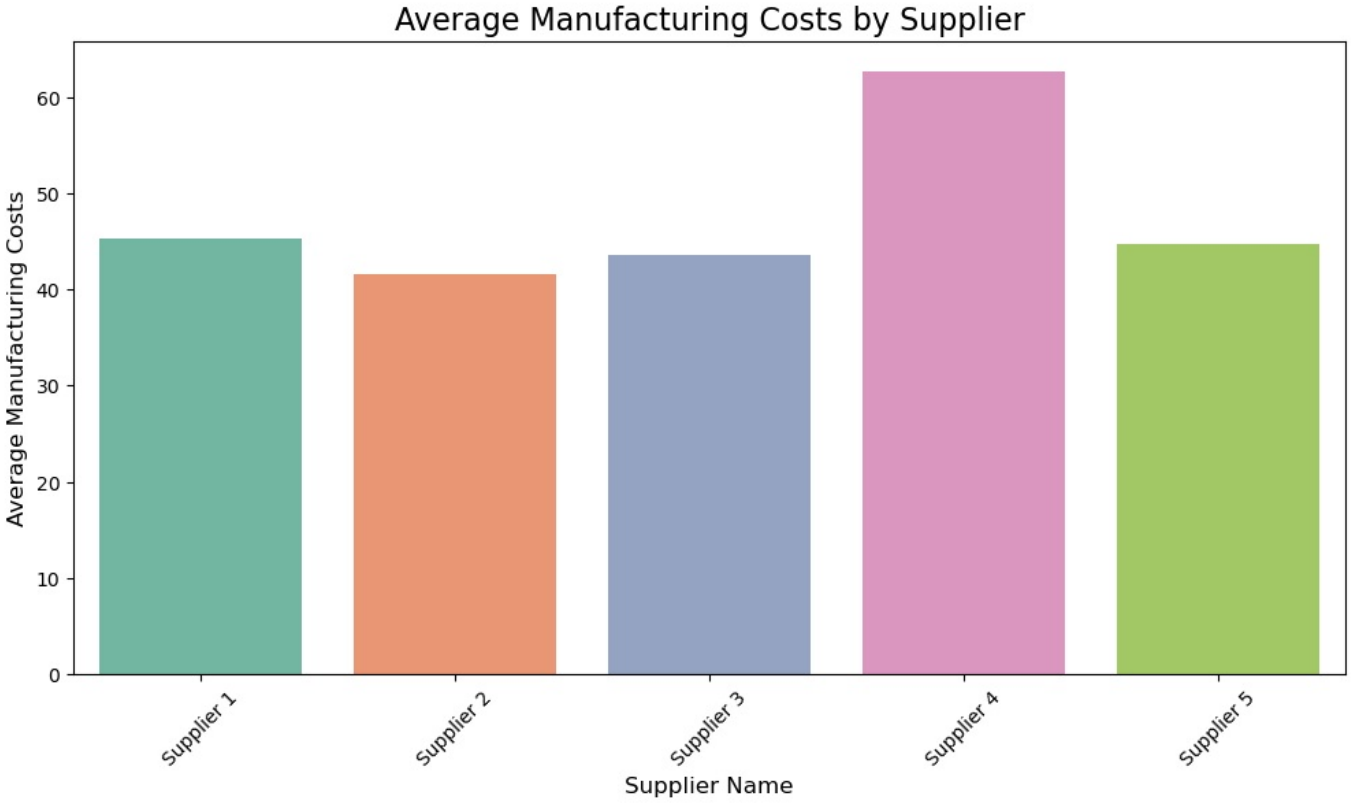
```
plt.figure(figsize=(10, 6))
sns.scatterplot(x=df['Manufacturing costs'],
                y=df['Revenue generated'],
                hue=df['Product type'],  # Color by 'Product Type'
                palette=['orange', 'green', 'skyblue'],  # Custom colors for the three categories
                alpha=0.7)  # Transparency of the points

# Set plot title and labels
plt.title('Relationship between Manufacturing Costs and Revenue Generated', fontsize=16)
plt.xlabel('Manufacturing Costs', fontsize=12)
plt.ylabel('Revenue Generated', fontsize=12)

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



Average Manufacturing Costs by Supplier

In [238…

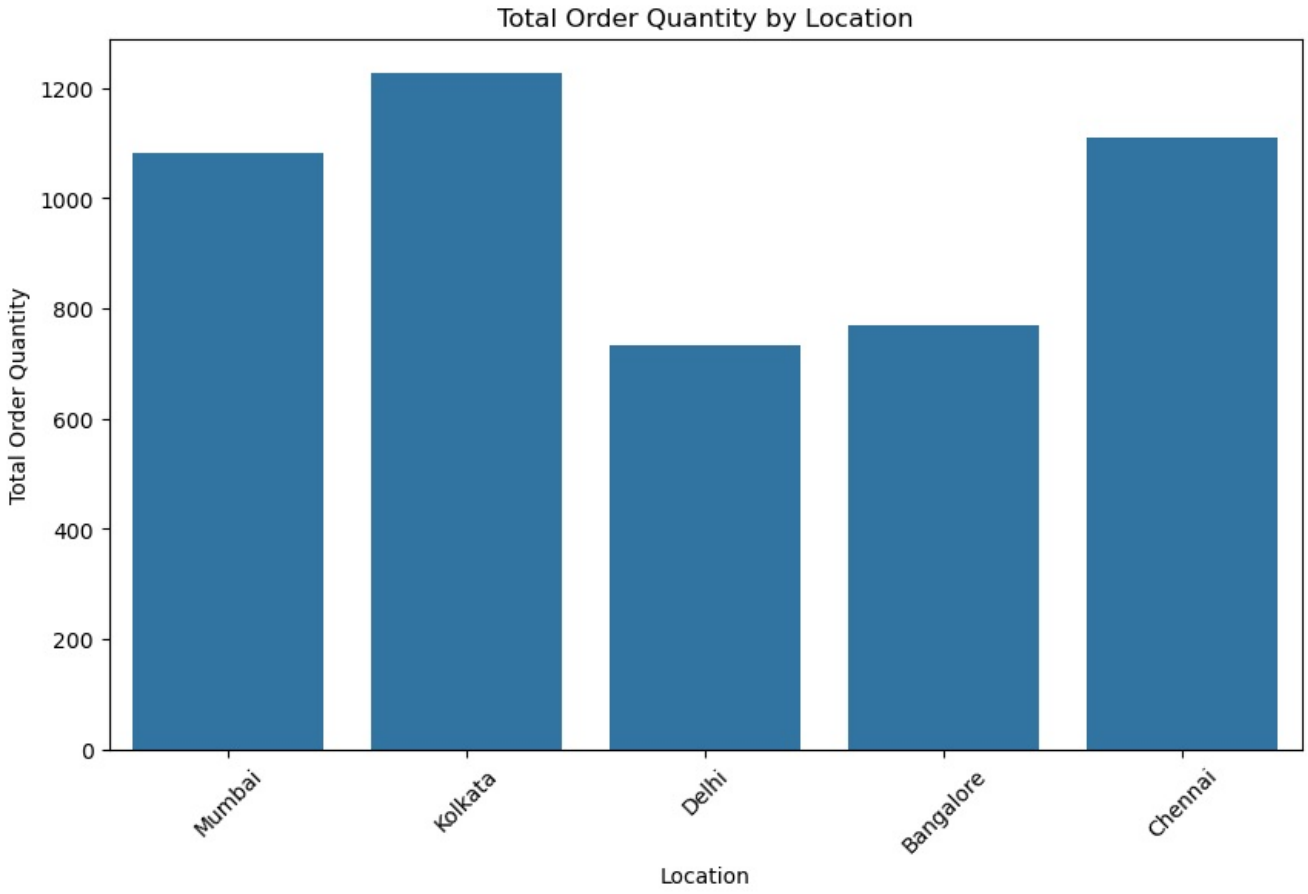
```
avg_manufacturing_costs = df.groupby('Supplier name')['Manufacturing costs'].mean().reset_index()

# Plotting the average manufacturing costs by suppliers using a bar chart
plt.figure(figsize=(12, 6))
sns.barplot(x='Supplier name', y='Manufacturing costs', data=avg_manufacturing_costs, hue='Supplier name', pale

# Set plot title and labels
plt.title('Average Manufacturing Costs by Supplier', fontsize=16)
plt.xlabel('Supplier Name', fontsize=12)
plt.ylabel('Average Manufacturing Costs', fontsize=12)
plt.xticks(rotation=45)  # Rotate x-axis labels for better readability

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



Total Order Quantity by Location

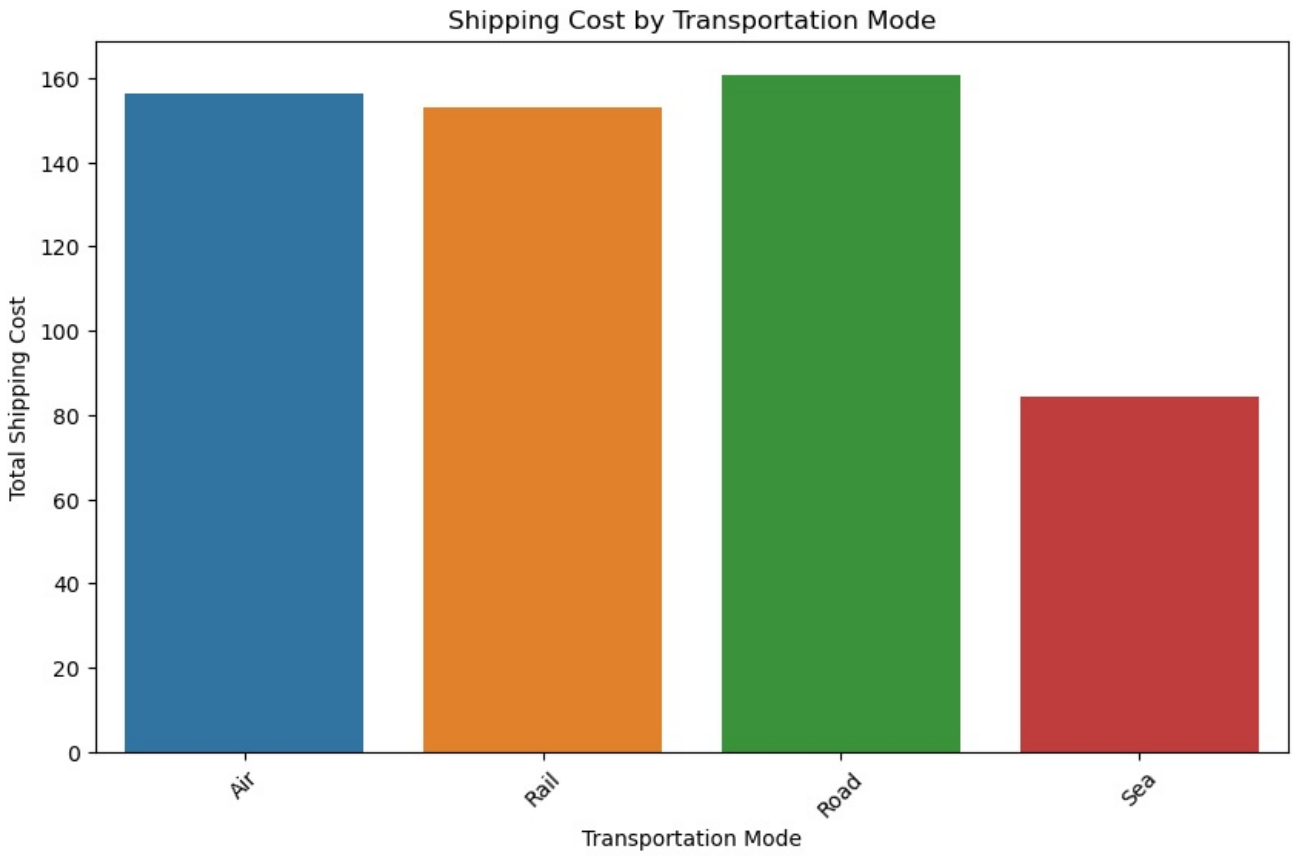
In [276…
```
plt.figure(figsize=(10, 6))

# Replace 'Order Quantity' and 'Location' with actual column names
sns.barplot(x='Location', y='Order quantities', data=df, estimator=sum,ci = None)

# Set the title and labels
plt.title('Total Order Quantity by Location')
plt.xlabel('Location')
plt.ylabel('Total Order Quantity')

# Show the plot
plt.xticks(rotation=45)  # Rotate labels if needed for better readability
plt.show()
```

```
C:\Users\Gowthami Galla\AppData\Local\Temp\ipykernel_21736\3946416223.py:4: FutureWarning:


The `ci` parameter is deprecated. Use `errorbar=None` for the same effect.
```



Shipping Cost by Transportation Mode

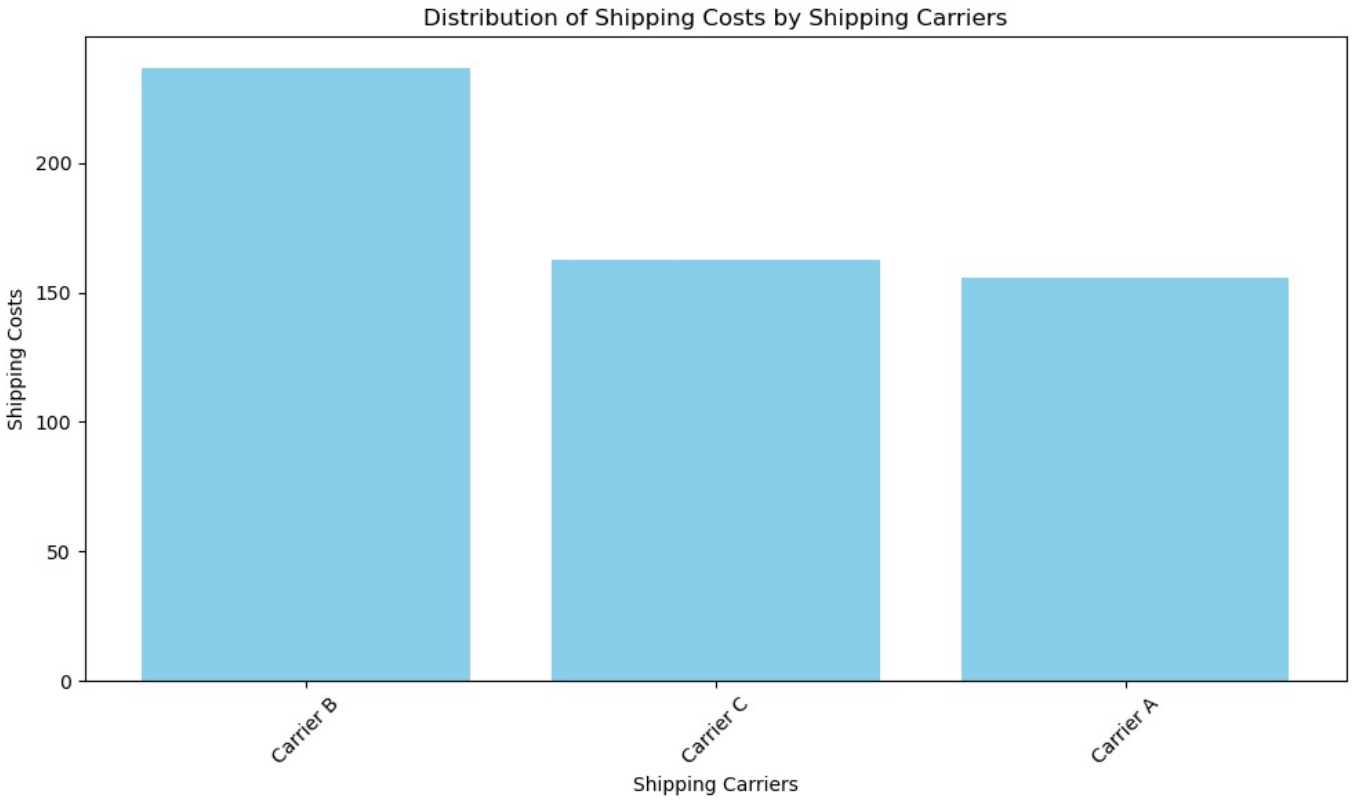
In [304…
```
plt.figure(figsize=(10, 6))
sns.barplot(x='Transportation modes', y='Shipping costs', data=shipping_cost_by_transportation_mode, hue='Transp

# Set the title and labels
plt.title('Shipping Cost by Transportation Mode')
plt.xlabel('Transportation Mode')
plt.ylabel('Total Shipping Cost')

# Show the plot
plt.xticks(rotation=45)  # Rotate labels if they are too long
plt.show()
```



Distribution of Shipping Costs by Shipping Carriers

In [314…

```
# Group the data by 'Shipping carriers' and sum the 'Shipping costs'
shipping_costs_by_carrier = df.groupby('Shipping carriers')['Shipping costs'].sum().reset_index()

# Sort values to make the plot clearer
shipping_costs_by_carrier = shipping_costs_by_carrier.sort_values('Shipping costs', ascending=False)

# Create a stacked bar chart using matplotlib
plt.figure(figsize=(10, 6))

# Plotting the stacked bar plot
plt.bar(shipping_costs_by_carrier['Shipping carriers'], shipping_costs_by_carrier['Shipping costs'], color='skyl

# Set the title and labels
plt.title('Distribution of Shipping Costs by Shipping Carriers')
plt.xlabel('Shipping Carriers')
plt.ylabel('Shipping Costs')

# Rotate x-axis labels for better readability if necessary
plt.xticks(rotation=45)

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

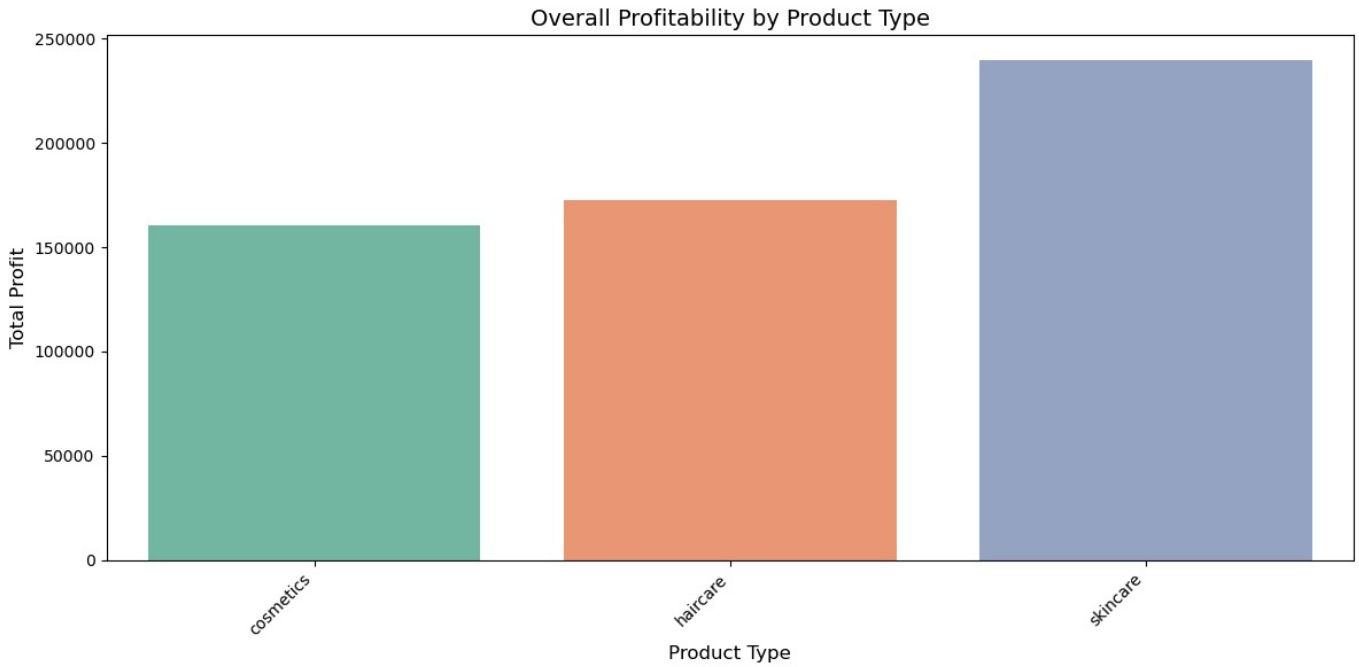


Overall Profitability by Product Type

In [332…

```
# Step 1: Calculate Profit for each row
df['Profit'] = df['Revenue generated'] - df['Manufacturing costs']

# Step 2: Group by 'Product Type' and calculate the total profit for each product type
profitability_by_product = df.groupby('Product type')['Profit'].sum().reset_index()

# Step 3: Create a bar plot to visualize the overall profitability by product type
plt.figure(figsize=(12, 6))
sns.barplot(x='Product type', y='Profit', data=profitability_by_product, palette='Set2')

# Adding title and labels
plt.title('Overall Profitability by Product Type', fontsize=14)
plt.xlabel('Product Type', fontsize=12)
plt.ylabel('Total Profit', fontsize=12)

# Rotate the x-axis labels if needed
plt.xticks(rotation=45, ha='right')

# Display the plot
plt.tight_layout()
plt.show()
```

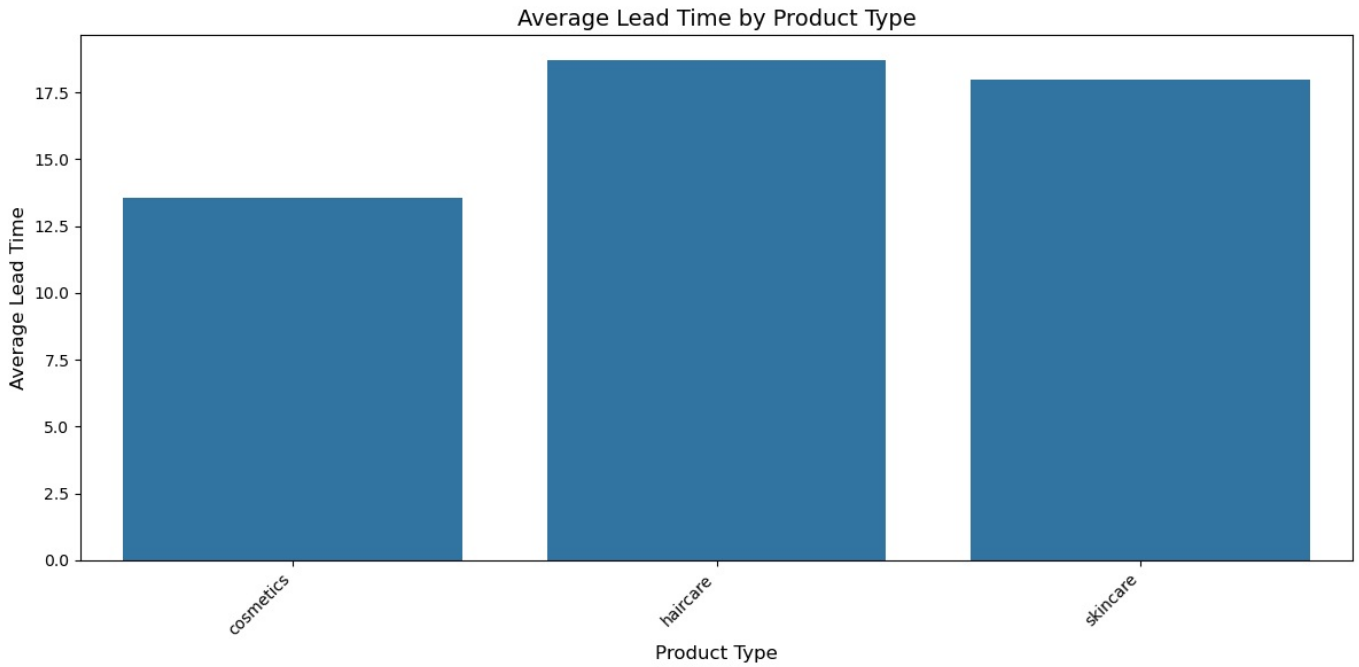
```
C:\Users\Gowthami Galla\AppData\Local\Temp\ipykernel_21736\1239111808.py:9: FutureWarning:


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



Average Lead Time by Product Type

In [362…

```
# Step 1: Group by 'Product Type' and calculate the average 'Leadtime'
avg_leadtime_by_product = df.groupby('Product type')['Lead time'].mean().reset_index()

# Step 2: Create a column chart (vertical bar plot) to visualize the average leadtime by product type
plt.figure(figsize=(12, 6))
sns.barplot(x='Product type', y='Lead time', data=avg_leadtime_by_product)

# Adding title and labels
plt.title('Average Lead Time by Product Type', fontsize=14)
plt.xlabel('Product Type', fontsize=12)
plt.ylabel('Average Lead Time', fontsize=12)

# Rotate the x-axis labels for better visibility
plt.xticks(rotation=45, ha='right')

# Display the plot
plt.tight_layout()
plt.show()
```



Average Defect Rate by Product Type

In [366…

```python
# Step 1: Group by 'Product Type' and calculate the average defect rate
avg_defect_rate_by_product = df.groupby('Product type')['Defect rates'].mean()

# Step 2: Create a pie chart
plt.figure(figsize=(8, 8))
colors = ['gold', 'skyblue', 'lightgreen', 'coral', 'violet']  # Customize colors as needed
plt.pie(
    avg_defect_rate_by_product,
    labels=avg_defect_rate_by_product.index,
    autopct='%1.1f%%',
    startangle=140,
    colors=colors
)

# Add title
plt.title('Average Defect Rate by Product Type', fontsize=14)

# Display the chart
plt.show()
```