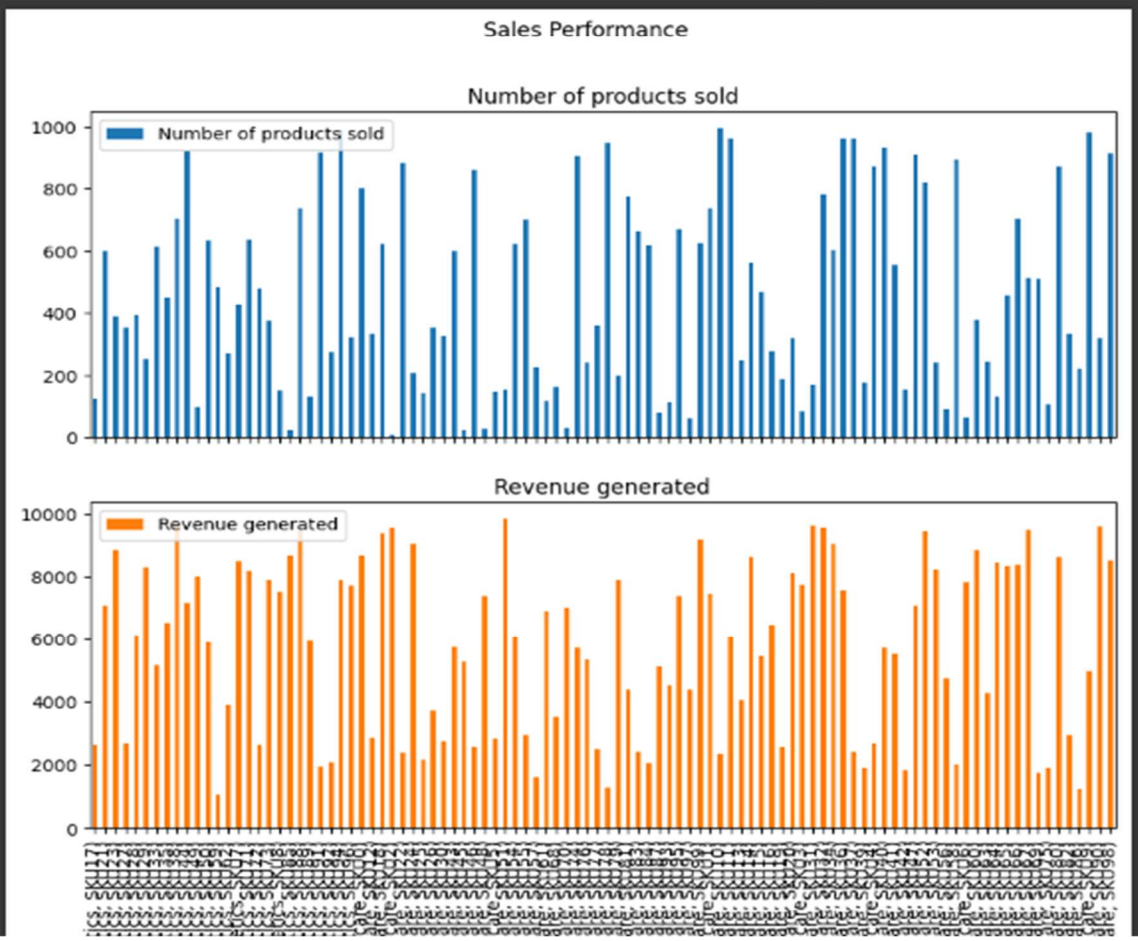
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
```

Load the Dataset

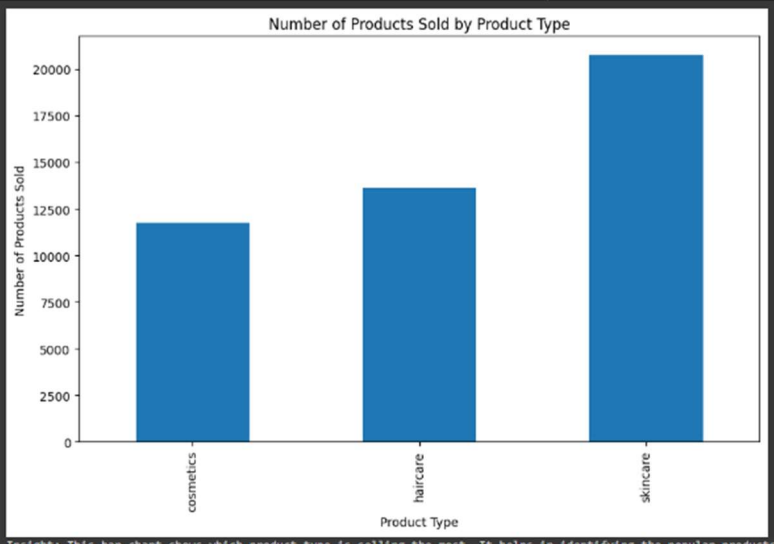
```
[2] df = pd.read_csv('15_Supply Chain Analysis.csv')
```

Sales and Performance Analysis

```
[3] sales_performance = df.groupby(['Product type', 'SKU'])[['Number of products sold', 'Revenue generated']].sum()
    sales_performance.plot(kind='bar', subplots=True, layout=(2,1), figsize=(10,8), title='Sales Performance')

    plt.show()
```

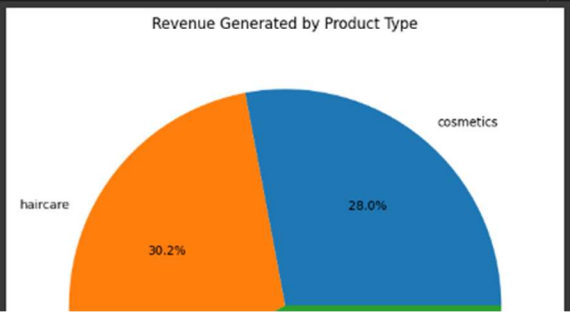


```
[4] products_sold = df.groupby('Product type')['Number of products sold'].sum()
    products_sold.plot(kind='bar', figsize=(10, 6), title='Number of Products Sold by Product Type')
    plt.xlabel('Product Type')
    plt.ylabel('Number of Products Sold')
    plt.show()
    print("Insight: This bar chart shows which product type is selling the most. It helps in identifying the popular products.")
```



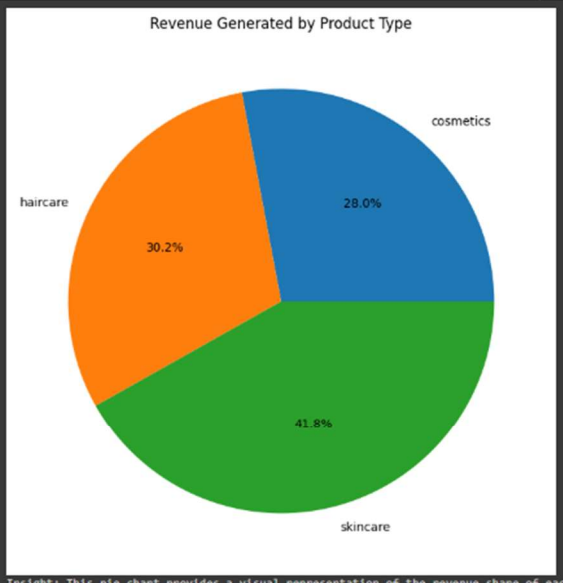

```
Insight: This bar chart shows which product type is selling the most. It helps in identifying the popular products.
```

```
[5] revenue_generated = df.groupby('Product type')['Revenue generated'].sum()
    revenue_generated.plot(kind='pie', figsize=(8, 8), title='Revenue Generated by Product Type', autopct='%1.1f%%')
    plt.ylabel('')
    plt.show()
    print("Insight: This pie chart provides a visual representation of the revenue share of each product type. It highlights the product types contributing most to the revenue.")
```



+ Code + Text

```
[5] revenue_generated = df.groupby('Product type')['Revenue generated'].sum()
    revenue_generated.plot(kind='pie', figsize=(8, 8), title='Revenue Generated by Product Type', autopct='%1.1f%%')
    plt.ylabel('')
    plt.show()
    print("Insight: This pie chart provides a visual representation of the revenue share of each product type. It highlights the product types contributing most to the revenue.")
```



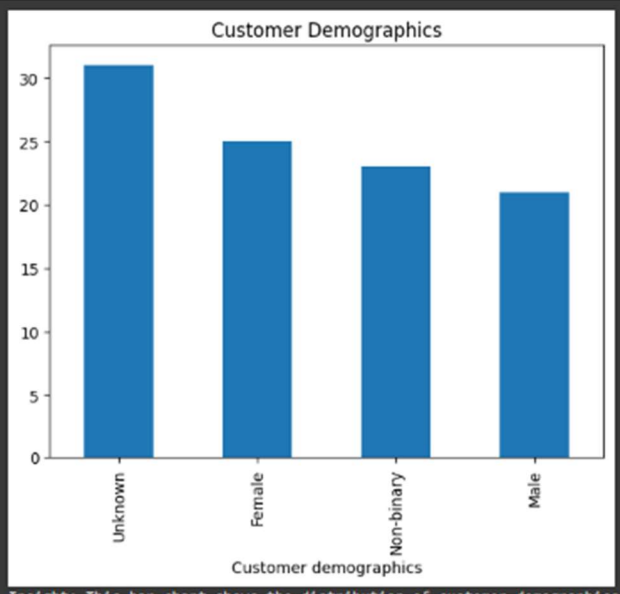

```
Insight: This pie chart provides a visual representation of the revenue share of each product type. It highlights the product types contributing most to the revenue.
```

Customer Demographics Insights

```
[6] customer_demographics = df['Customer demographics'].value_counts()
    customer_demographics.plot(kind='bar', title='Customer Demographics')

    plt.show()
    print("Insight: This bar chart shows the distribution of customer demographics. It helps in understanding the customer base better.")
```

+ Code + Text

```
customer_demographics = df['Customer demographics'].value_counts()
customer_demographics.plot(kind='bar', title='Customer Demographics')

plt.show()
print("Insight: This bar chart shows the distribution of customer demographics. It helps in understanding the customer base better.")
```




```
Insight: This bar chart shows the distribution of customer demographics. It helps in understanding the customer base better.
```

Inventory Management

```
[7] inventory_summary = df[['Stock levels', 'Lead times', 'Order quantities']].describe()
```

```
[8] threshold = 10  # Example threshold value
    low_stock = df[df['Stock levels'] < threshold]
    print(low_stock)
```

```
31        9655.135103                Male            6            17
33        5149.998350           Non-binary           4            17
34        9061.710896              Unknown           1            26
47        7089.474250                Male            4            15
57        1605.866900              Unknown           5            18
68        3550.218433           Non-binary           0             8
78        1292.458418              Unknown           5             4
87        5133.846701                Male            5             7

    Order quantities  ...   Location Lead time  Production volumes  \
2                 88  ...     Mumbai        12                 971
4                 56  ...      Delhi         5                 414
8                 15  ...     Mumbai        13                 769
15                69  ...  Bangalore        14                 580
16                78  ...  Bangalore         3                 399
24                 2  ...  Bangalore        28                 447
31                44  ...    Chennai        24                 461
33                95  ...    Chennai         1                 251
34                21  ...    Chennai         4                 452
47                51  ...    Kolkata        10                 964
57                51  ...      Delhi        21                 588
68                58  ...  Bangalore         2                 375
78                51  ...     Mumbai        25                 858
87                55  ...    Chennai        27                 523
```

```
threshold = 10  # Example threshold value
low_stock = df[df['Stock levels'] < threshold]
print(low_stock)
```

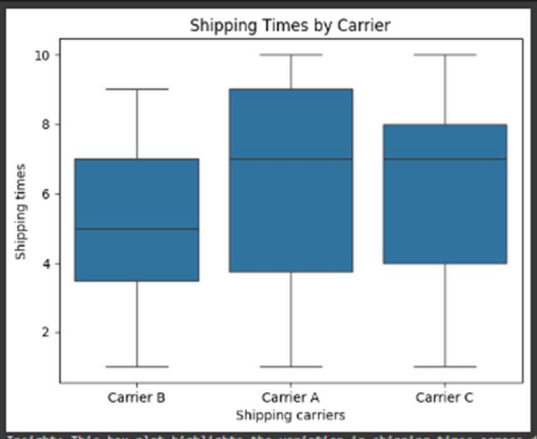
```
    Stock levels  Lead times  Order quantities  ...   Location Lead time  Production volumes  \
2         9655.135103                 Male           6          17                  88  ...     Mumbai        12                 971
33        5149.998350           Non-binary           4          17                  56  ...      Delhi         5                 414
34        9061.710896              Unknown           1          26                  15  ...     Mumbai        13                 769
47        7089.474250                 Male           4          15                  69  ...  Bangalore        14                 580
57        1605.866900              Unknown           5          18                  78  ...  Bangalore         3                 399
68        3550.218433           Non-binary           0           8                   2  ...  Bangalore        28                 447
78        1292.458418              Unknown           5           4                  44  ...    Chennai        24                 461
87        5133.846701                 Male           5           7                  95  ...    Chennai         1                 251
                                                                                    21  ...    Chennai         4                 452
                                                                                    51  ...    Kolkata        10                 964
                                                                                    51  ...      Delhi        21                 588
                                                                                    58  ...  Bangalore         2                 375
                                                                                    51  ...     Mumbai        25                 858
                                                                                    55  ...    Chennai        27                 523

    Manufacturing lead time  Manufacturing costs  Inspection results  \
2                        27           30.688019             Pending
4                         3           92.065161                Fail
8                         8           11.423027             Pending
15                        7           97.121282                Pass
16                       21           77.106342                Pass
24                        3           40.382360             Pending
31                        8           60.251146             Pending
33                       23           23.853428                Fail
34                       10           10.754273                Pass
47                       20           19.712993             Pending
57                       25           67.779623             Pending
68                       18           97.113582                Fail
78                       21           71.126515             Pending
87                       17           28.696997                Fail

    Defect rates  Transportation modes   Routes       Costs
2       4.580593                   Air  Route C  141.920282
4       3.145580                   Air  Route A  923.440632
8       2.709863                   Sea  Route B  505.557134
15      2.264406                   Sea  Route B  127.861800
16      1.012563                   Air  Route A  865.525780
24      3.691310                   Air  Route A  758.724773
31      2.989000                  Rail  Route C  609.379207
33      3.541046                   Sea  Route A  371.255296
34      0.646605                  Road  Route B  510.358000
47      0.380574                  Rail  Route A  581.602355
57      2.511175                  Rail  Route A  482.191239
68      1.983468                  Rail  Route A  299.706303
78      4.096881                   Sea  Route C  323.592203
87      3.693738                   Sea  Route B  879.359218

[14 rows x 24 columns]
```

Logistics and Shipping Efficiency

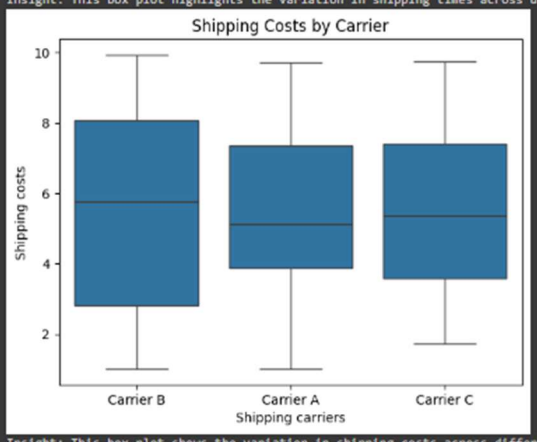
```
[9] shipping_performance = df.groupby('Shipping carriers')[['Shipping times', 'Shipping costs']].mean()

    sns.boxplot(x='Shipping carriers', y='Shipping times', data=df)
    plt.title('Shipping Times by Carrier')
    plt.show()
    print("Insight: This box plot highlights the variation in shipping times across different carriers, helping in identifying the most efficient shipping carrier.")
```

Logistics and Shipping Efficiency

```
shipping_performance = df.groupby('Shipping carriers')[['Shipping times', 'Shipping costs']].mean()

sns.boxplot(x='Shipping carriers', y='Shipping times', data=df)
plt.title('Shipping Times by Carrier')
plt.show()
print("Insight: This box plot highlights the variation in shipping times across different carriers, helping in identifying the most efficient shipping carrier.")

sns.boxplot(x='Shipping carriers', y='Shipping costs', data=df)
plt.title('Shipping Costs by Carrier')
plt.show()
print("Insight: This box plot shows the variation in shipping costs across different carriers, aiding in cost-efficiency analysis.")
```



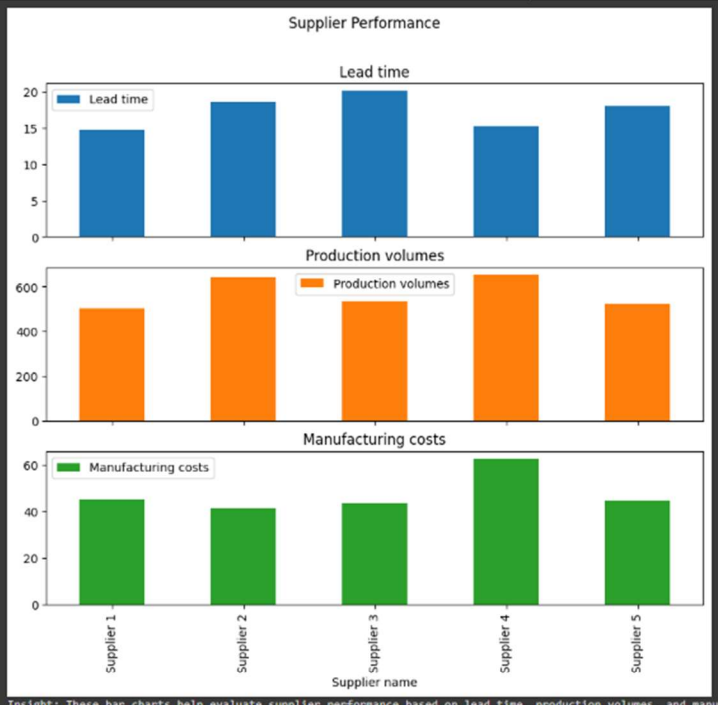

```
Insight: This box plot highlights the variation in shipping times across different carriers, helping in identifying the most efficient shipping carrier.
```




```
Insight: This box plot shows the variation in shipping costs across different carriers, aiding in cost-efficiency analysis.
```

Supplier Performance Evaluation

```
supplier_performance = df.groupby('Supplier name')[['Lead time', 'Production volumes', 'Manufacturing costs']].mean()
supplier_performance.plot(kind='bar', subplots=True, layout=(3, 1), figsize=(10, 8), title='Supplier Performance')
plt.show()
print("Insight: These bar charts help evaluate supplier performance based on lead time, production volumes, and manufacturing costs. It aids in supplier selection and management.")
```




```
Insight: These bar charts help evaluate supplier performance based on lead time, production volumes, and manufacturing costs. It aids in supplier selection and management.
```

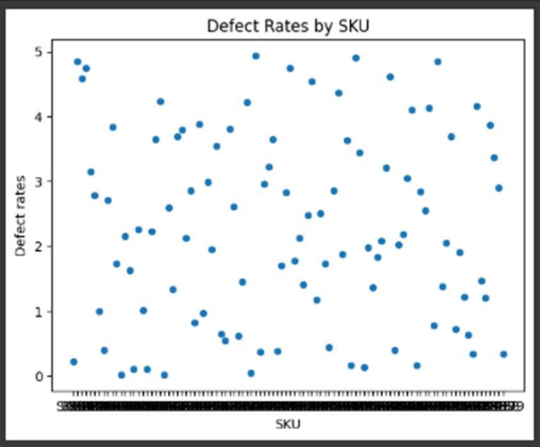
Quality Control

```
[11] quality_control = df.groupby('SKU')['Defect rates'].mean()
     sns.scatterplot(x='SKU', y='Defect rates', data=df)
     plt.title('Defect Rates by SKU')
     plt.show()
```

+ Code + Text

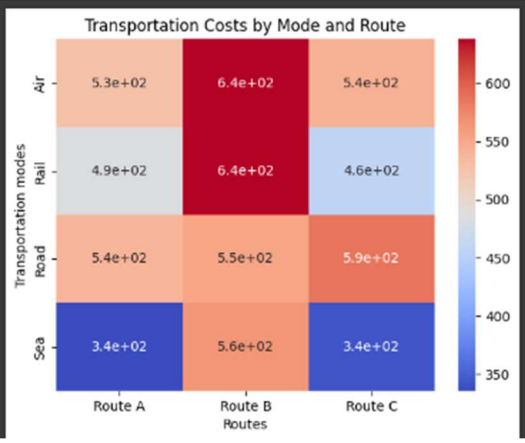
Quality Control

```
quality_control = df.groupby('SKU')['Defect rates'].mean()
sns.scatterplot(x='SKU', y='Defect rates', data=df)
plt.title('Defect Rates by SKU')
plt.show()
```



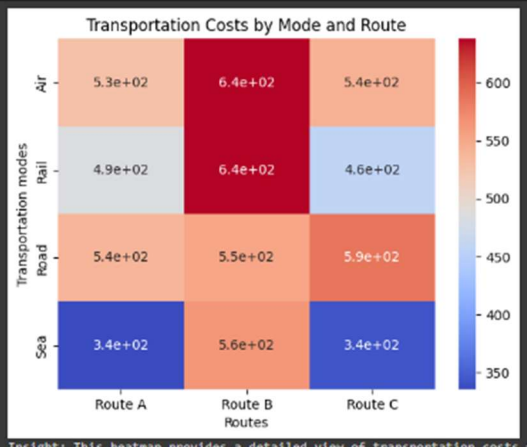
Transportation and Routing Efficiency

```
[12] transportation_efficiency = df.groupby(['Transportation modes', 'Routes'])['Costs'].mean().unstack()
sns.heatmap(transportation_efficiency, annot=True, cmap='coolwarm')
plt.title('Transportation Costs by Mode and Route')
plt.show()
print("Insight: This heatmap provides a detailed view of transportation costs for different modes and routes, useful for optimizing transportation strategy.")
```



Transportation and Routing Efficiency

```
transportation_efficiency = df.groupby(['Transportation modes', 'Routes'])['Costs'].mean().unstack()
sns.heatmap(transportation_efficiency, annot=True, cmap='coolwarm')
plt.title('Transportation Costs by Mode and Route')
plt.show()
print("Insight: This heatmap provides a detailed view of transportation costs for different modes and routes, useful for optimizing transportation strategy.")
```




```
Insight: This heatmap provides a detailed view of transportation costs for different modes and routes, useful for optimizing transportation strategy.
```

DATA ANALYSIS TECHNIQUES

Employ Descriptive Statistics

```
[13] print(df.head())
descriptive_stats = df.describe()
print(descriptive_stats)
```

```
  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
```

DATA ANALYSIS TECHNIQUES

Employ Descriptive Statistics

```
[13] print(df.head())
     descriptive_stats = df.describe()
     print(descriptive_stats)
```

```
  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]
              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
```

+ Code + Text

```
       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                100.00000           100.000000    100.000000  100.000000
mean                  14.77000            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
```

Implement Inferential Statistics

```
[14] haircare_revenue = df[df['Product type'] == 'haircare']['Revenue generated']
     skincare_revenue = df[df['Product type'] == 'skincare']['Revenue generated']

     # Perform an independent t-test
     t_stat, p_value = stats.ttest_ind(haircare_revenue, skincare_revenue)
     print(f'T-statistic: {t_stat}, P-value: {p_value}')
```

```
T-statistic: -1.4145258015562003, P-value: 0.16151767429524036
```

Develop a Predictive Model

```
[15] X = df[['Number of products sold', 'Stock levels', 'Lead times', 'Order quantities', 'Shipping times', 'Shipping costs']]
     y = df['Revenue generated']

     # Split the data into training and testing sets
     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

     # Create and train the linear regression model
     model = LinearRegression()
     model.fit(X_train, y_train)

     # Make predictions
     y_pred = model.predict(X_test)

     # Evaluate the model
     mse = mean_squared_error(y_test, y_pred)
     r2 = r2_score(y_test, y_pred)
     print(f'Mean Squared Error: {mse}')
     print(f'R-squared: {r2}')
```

```
Mean Squared Error: 7484459.001456099
R-squared: 0.06864818572406117
```