1. How can the raw, unnormalized dataset be transformed and normalized for consistency and usability?

```
import pandas as pd

import warnings

warnings.filterwarnings('ignore')


# Load the dataset

data = pd.read_csv('DSS Data.csv')


# Step 1: Data Cleaning

data = data.drop_duplicates()  # Remove duplicates

data.fillna(method='ffill', inplace=True)  # Handle missing values

data.columns = [col.strip().replace(" ", "_").lower() for col in data.columns]  # Rename columns


# Step 2: Transformation

data['profit'] = data['revenue_generated'] - data['costs']  # Create a derived column


# Step 3: Normalization

data_normalized = pd.get_dummies(data, columns=['product_type','shipping_carriers'])  # Convert categorical to dummies

# Validate the transformed data

print(data_normalized.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 30 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   sku                         100 non-null    object
 1   price                       100 non-null    float64
 2   availability                100 non-null    int64
 3   number_of_products_sold     100 non-null    int64
 4   revenue_generated           100 non-null    float64
 5   customer_demographics       100 non-null    object
 6   stock_levels                100 non-null    float64
 7   lead_times                  100 non-null    int64
 8   order_quantities            100 non-null    int64
 9   shipping_times              100 non-null    int64
 10  shipping_costs              100 non-null    float64
 11  supplier_name               100 non-null    object
 12  location                    100 non-null    object
 13  lead_time                   100 non-null    int64
 14  production_volumes          100 non-null    int64
 15  manufacturing_lead_time     100 non-null    int64
 16  manufacturing_costs         100 non-null    float64
 17  inspection_results          100 non-null    object
 18  defect_rates                100 non-null    float64
 19  transportation_modes        100 non-null    object
 20  routes                      100 non-null    object
 21  costs                       100 non-null    float64
 22  revenue_per_unit            100 non-null    float64
 23  profit                      100 non-null    float64
 24  product_type_cosmetics      100 non-null    bool
 25  product_type_haircare       100 non-null    bool
 26  product_type_skincare       100 non-null    bool
 27  shipping_carriers_Carrier A 100 non-null    bool
 28  shipping_carriers_Carrier B 100 non-null    bool
 29  shipping_carriers_Carrier C 100 non-null    bool
dtypes: bool(6), float64(9), int64(8), object(7)
memory usage: 19.5+ KB
None
```

data_normalized.head()

36]:

	sku	price	availability	number_of_products_sold	revenue_generated	customer_demographics	stock_levels	lead_times	order_quantities	shipping_times	...	rou
0	SKU0	0.698749	55	802	8661.996792	Non-binary	0.58	7	96	4	...	Rc
1	SKU1	0.134845	95	736	7460.900065	Female	0.53	30	37	2	...	Rc
2	SKU2	0.098693	34	8	9577.749626	Unknown	0.01	10	88	2	...	Rc
3	SKU3	0.610060	68	83	7766.836426	Non-binary	0.23	13	59	6	...	Rc
4	SKU4	0.031861	26	871	2686.505152	Non-binary	0.05	3	56	8	...	Rc

5 rows × 30 columns

[36]:

..	routes	costs	revenue_per_unit	profit	product_type_cosmetics	product_type_haircare	product_type_skincare	shipping_carriers_Carrier A	shipping_carriers_Ca
...	Route B	197.752075	10.800495	8464.244716	False	True	False	False	
...	Route B	513.065579	10.137092	6947.834486	False	False	True	True	
...	Route C	151.920282	1197.218703	9425.829344	False	True	False	False	
...	Route A	264.776159	93.576342	7502.060267	False	False	True	False	
...	Route A	933.440632	3.084392	1753.064520	False	False	True	True	

[ ]:

36]:

_per_unit	profit	product_type_cosmetics	product_type_haircare	product_type_skincare	shipping_carriers_Carrier A	shipping_carriers_Carrier B	shipping_carriers_Carrier C
10.800495	8464.244716	False	True	False	False	True	False
10.137092	6947.834486	False	False	True	True	False	False
97.218703	9425.829344	False	True	False	False	True	False
93.576342	7502.060267	False	False	True	False	False	True
3.084392	1753.064520	False	False	True	True	False	False

2. What steps are needed to create a database on MS SQL Server using the normalized data?

```
-- Step 1: Create and Use Database
create database SupplyChainDSS;
use SupplyChainDSS;

-- Step 2: Create Tables with Relationships
create table Products (SKU varchar(50) primary key,Product_Type varchar(50), Price
float);

 create table Sales (Sale_ID int identity (1,1) primary key, SKU
varchar(50),Revenue_Generated float,
 Number_Of_Products_Sold int,foreign key (SKU) references Products(SKU));

create table Inventory (SKU varchar(50) primary key,Stock_Levels int,Lead_Times int,
 Order_Quantities int, foreign key (SKU) references Products(SKU));

create table  Shipping (Shipping_ID int identity (1,1) primary key,SKU varchar(50),
Shipping_Carrier varchar(50),Shipping_Times int,Shipping_Costs float,
foreign key (SKU) references Products(SKU));
```

```sql
create table Customers ( Customer_ID int identity(1,1) primary key,
Demographics varchar(50),location varchar(50));

create table Manufacturing (SKU varchar(50) primary key,Production_Volumes int,
Manufacturing_Lead_Time int,Manufacturing_Costs float, foreign key (SKU) references
Products(SKU));

-- Step 3: Populate Tables
insert into Products (SKU, Product_Type, Price)
values ('SKU1', 'Skincare', 14.84), ('SKU2', 'Haircare', 11.32), ('SKU3', 'Cosmetics',
45.12);

insert into Sales (SKU, Revenue_Generated, Number_Of_Products_Sold)
values ('SKU1', 7460.90, 736), ('SKU2', 9577.75, 802), ('SKU3', 8650.00, 120);

insert into Inventory (SKU, Stock_Levels, Lead_Times, Order_Quantities)
values ('SKU1', 50, 7, 100), ('SKU2', 30, 10, 200), ('SKU3', 20, 5, 150);

insert into Shipping (SKU, Shipping_Carrier, Shipping_Times, Shipping_Costs)
values ('SKU1', 'Carrier A', 3, 200.00), ('SKU2', 'Carrier B', 5, 150.00), ('SKU3',
'Carrier C', 2, 100.00);

insert into Customers (Demographics, location)
values ('Female', 'Mumbai'), ('Male', 'Delhi'), ('Non-binary', 'Bangalore');

insert into Manufacturing (SKU, Production_Volumes, Manufacturing_Lead_Time,
Manufacturing_Costs)
values ('SKU1', 500, 10, 1500.00), ('SKU2', 300, 15, 1200.00), ('SKU3', 400, 12,
1300.00);

-- Step 4: Analyze Data with Queries
-- Join Products and Sales
select P.SKU, P.Product_Type, S.Revenue_Generated, S.Number_Of_Products_Sold
from Products P
join Sales S on P.SKU = S.SKU;

-- Analyze Inventory and Shipping
select I.SKU, I.Stock_Levels, I.Lead_Times, S.Shipping_Times, S.Shipping_Costs
from Inventory I
join Shipping S on I.SKU = S.SKU;

-- Analyze Manufacturing Data
select M.SKU, P.Product_Type, M.Production_Volumes, M.Manufacturing_Lead_Time,
M.Manufacturing_Costs
from Manufacturing M
join Products P on M.SKU = P.SKU;
```

Results | Messages

	SKU	Product_Type	Production_Volumes	Manufacturing_Lead_Time	Manufacturing_Costs
1	SKU1	Skincare	500	10	1500
2	SKU2	Haircare	300	15	1200
3	SKU3	Cosmetics	400	12	1300

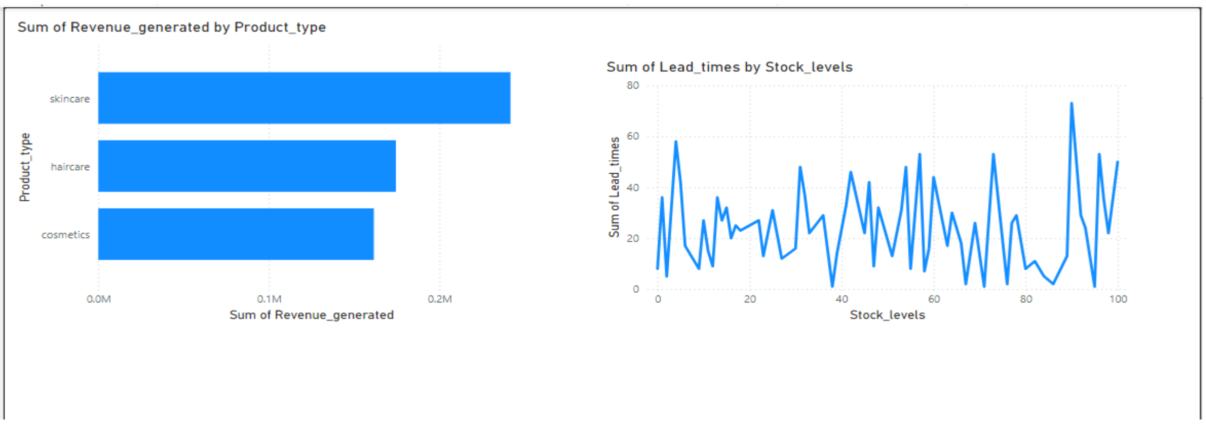
3. **How can we connect the MS SQL Server database to reporting tools like Excel, Power BI, or Tableau for effective data analysis and visualization?**

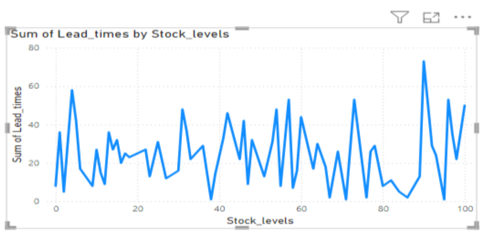
```sql
-- Step 1: Create the Database
create database SupplyChainDB;

-- Use the created database
use SupplyChainDB;

-- Step 2: Create a Table to Store the Dataset
CREATE TABLE SupplyChainData(
    Product_Type VARCHAR(50),
    SKU VARCHAR(50),
    Price FLOAT,
    Availability INT,
    Number_Of_Products_Sold INT,
    Revenue_Generated FLOAT,
    Customer_Demographics VARCHAR(50),
    Stock_Levels INT,
    Lead_Times INT,
    Order_Quantities INT,
    Shipping_Times INT,
    Shipping_Carriers VARCHAR(50),
    Shipping_Costs FLOAT,
    Supplier_Name VARCHAR(50),
    Location VARCHAR(50),
    Lead_Time INT,
    Production_Volumes INT,
    Manufacturing_Lead_Time INT,
    Manufacturing_Costs FLOAT,
    Inspection_Results VARCHAR(50),
    Defect_Rates FLOAT,
    Transportation_Modes VARCHAR(50),
    Routes VARCHAR(50),
    Costs FLOAT
);
-- Step 3: Import Data into the Table
```

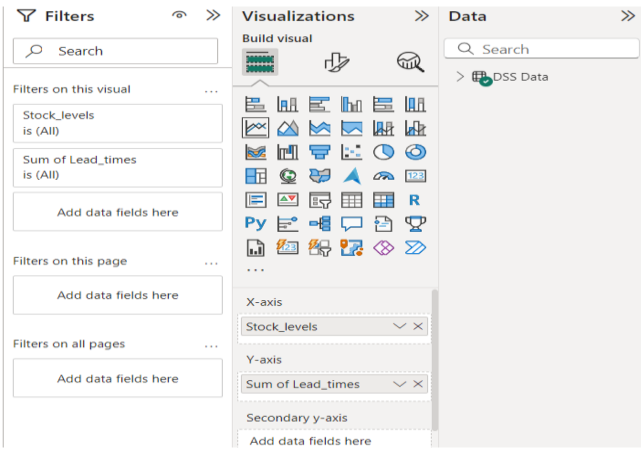
1. Open SQL Server Management Studio (SSMS).
2. Right-click on your database (SupplyChainDB) and select Tasks > Import Data.
3. Choose Flat File Source and select your CSV file.
4. Complete the wizard to load the data.

Connect MS SQL Server to Power BI

1. Open Power BI and click **Get Data > SQL Server**.
2. Enter the server name and database name (SupplyChainDB).
3. Authenticate and load the SupplyChainData table.
4. Create visualizations:
   - Bar chart: Product Type vs Revenue Generated.
   - Line chart: Stock Levels over Time .








```sql
-- Step 4: Example Queries for Analysis

-- 4.1 Total Revenue by Product Type
SELECT Product_Type, SUM(Revenue_Generated) AS Total_Revenue
FROM [DSS Data]
GROUP BY Product_Type;
```

Results | Messages

	Product_Type	Total_Revenue
1	cosmetics	161521.265991211
2	haircare	174455.392211914
3	skincare	241628.162231445

```sql
-- 4.2 Inventory with High Stock Levels
SELECT SKU, Product_Type, Stock_Levels
FROM [DSS Data]
WHERE Stock_Levels > 90;
```

Results | Messages

	SKU	Product_Type	Stock_Levels
1	SKU7	cosmetics	93
2	SKU12	haircare	100
3	SKU45	haircare	93
4	SKU46	haircare	92
5	SKU49	cosmetics	97
6	SKU51	haircare	100
7	SKU53	skincare	96
8	SKU55	haircare	97
9	SKU59	cosmetics	100
10	SKU69	skincare	95
11	SKU77	haircare	96
12	SKU91	cosmetics	98

```sql
-- 4.3 Shipping Costs by Carrier
SELECT Shipping_Carriers, AVG(Shipping_Costs) AS Average_Cost
FROM [DSS Data]
GROUP BY Shipping_Carriers;
```

Results | Messages

	Shipping_Carriers	Average_Cost
1	Carrier A	5.55492250408445
2	Carrier B	5.50924700082735
3	Carrier C	5.5992916288047

```sql
-- 4.4 Top 5 Products by Sales
SELECT TOP 5 SKU, Product_Type, Number_Of_Products_Sold
FROM [DSS Data]
ORDER BY Number_Of_Products_Sold DESC;
```

Results | Messages

	SKU	Product_Type	Number_Of_Products_Sold
1	SKU10	skincare	996
2	SKU94	cosmetics	987
3	SKU9	skincare	980
4	SKU36	skincare	963
5	SKU37	skincare	963

```sql
-- 4.5 Detailed Analysis: Join Key Metrics
SELECT
    Product_Type,
    SUM(Number_Of_Products_Sold) AS Total_Units_Sold,
    AVG(Price) AS Avg_Price,
    SUM(Revenue_Generated) AS Total_Revenue,
    AVG(Shipping_Costs) AS Avg_Shipping_Cost,
    AVG(Manufacturing_Costs) AS Avg_Manufacturing_Cost
FROM [DSS Data]
GROUP BY Product_Type;
```

Results | Messages

	Product_Type	Total_Units_Sold	Avg_Price	Total_Revenue	Avg_Shipping_Cost	Avg_Manufacturing_Cost
1	cosmetics	11757	57.361057804181	161521.265991211	6.06014089400952	43.052740743527
2	haircare	13611	46.0142791481579	174455.392211914	5.90775689307381	48.4579932268928
3	skincare	20731	47.259328854084	241628.162231445	4.90968776941299	48.993157428503

4. What processes are involved in cleaning the data through an Extract, Transform, and Load (ETL) process?

```python
import pandas as pd

import warnings

warnings.filterwarnings('ignore')

# Step 1: Load the Dataset (Extract)

data = pd.read_csv('DSS Data.csv')

# Step 2: Inspect the Dataset

print("Initial Dataset Info:")

data.info()
```

```
Initial Dataset 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
```

```
# Step 3: Data Transformation

# Step 3.1: Handle Missing Values

filled_data = data.fillna({

   "Price": data["Price"].median(),  # Replace missing prices with median

   "Stock levels": data["Stock levels"].mean(),  # Replace missing stock levels with mean

   "Shipping costs": data["Shipping costs"].median(),  # Replace missing shipping costs with median})


# Step 2: Inspect the Dataset

print("Initial Dataset Info:")

data.info()
```

```
Initial Dataset 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
```

```
# Step 3.2: Remove Duplicates

deduplicated_data = filled_data.drop_duplicates()

# Step 3.3: Rename Columns for Consistency

deduplicated_data.columns = [col.strip().replace(" ", "_").lower() for col in deduplicated_data.columns]

# Step 3.4: Add Derived Columns

if "revenue_generated" in deduplicated_data.columns and "costs" in deduplicated_data.columns:

   deduplicated_data["profit"] = deduplicated_data["revenue_generated"] - deduplicated_data["costs"]

# Step 3.5: Normalize Categorical Data

transformed_data = pd.get_dummies(
```

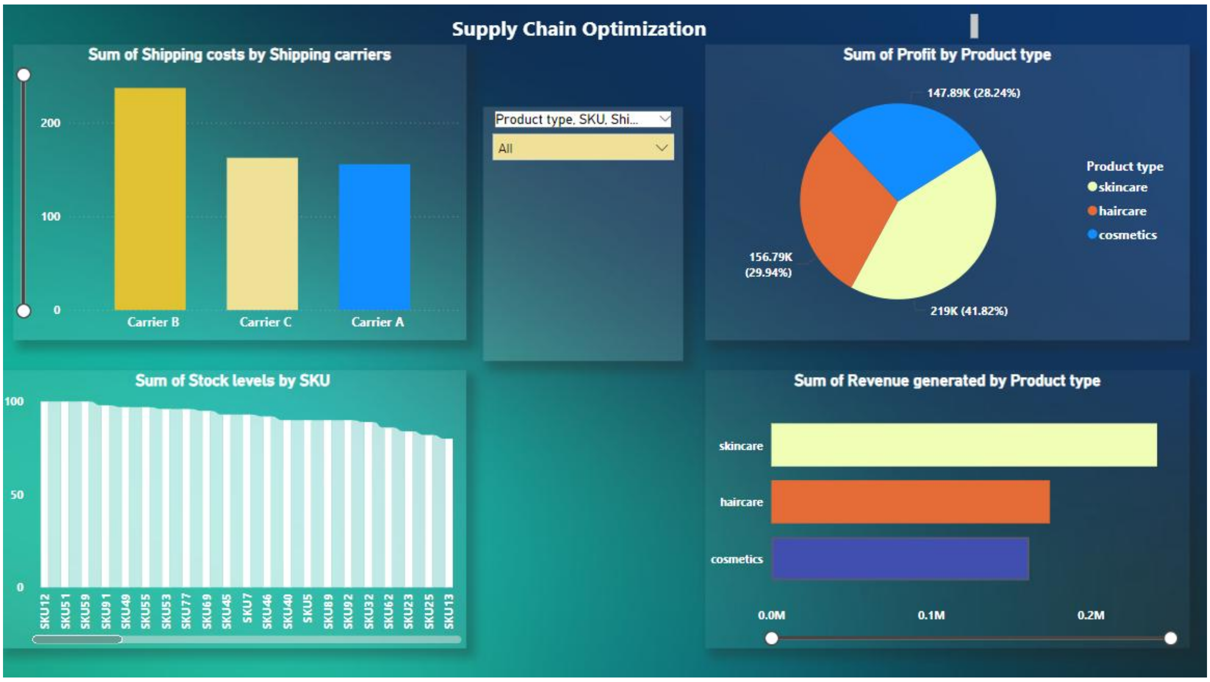
```
    deduplicated_data,
    columns=["product_type", "customer_demographics", "shipping_carriers"])
# Step 4: Save the Cleaned Data (Load)
cleaned_file_path = "Cleaned_DSS_Data.csv"
transformed_data.to_csv(cleaned_file_path, index=False)
# Preview the first few rows of the cleaned dataset
print("Cleaned Data Preview:")
print(transformed_data.head())
print(f"Cleaned dataset saved to: {cleaned_file_path}")
```

```
Cleaned Data Preview:
    sku      price  availability  number_of_products_sold  revenue_generated  \
0  SKU0  69.808006            55                      802        8661.996792
1  SKU1  14.843523            95                      736        7460.900065
2  SKU2  11.319683            34                        8        9577.749626
3  SKU3  61.163343            68                       83        7766.836426
4  SKU4   4.805496            26                      871        2686.505152

   stock_levels  lead_times  order_quantities  shipping_times  shipping_costs  \
0            58           7                96               4        2.956572
1            53          30                37               2        9.716575
2             1          10                88               2        8.054479
3            23          13                59               6        1.729569
4             5           3                56               8        3.890548

   ... product_type_cosmetics product_type_haircare  product_type_skincare  \
0  ...                  False                  True                  False
1  ...                  False                 False                   True
2  ...                  False                  True                  False
3  ...                  False                 False                   True
4  ...                  False                 False                   True

   customer_demographics_Female  customer_demographics_Male  \
0                         False                       False
1                          True                       False
2                         False                       False
3                         False                       False
4                         False                       False

   customer_demographics_Non-binary customer_demographics_Unknown  \
0                              True                         False
1                             False                         False
2                             False                          True
3                              True                         False
4                              True                         False

   shipping_carriers_Carrier A shipping_carriers_Carrier B  \
0                        False                        True
1                         True                       False
2                        False                        True
3                        False                       False
4                         True                       False

  shipping_carriers_Carrier C
0                       False
1                       False
2                       False
3                        True
4                       False

[5 rows x 32 columns]
Cleaned dataset saved to: Cleaned_DSS_Data.csv
```

```
]:
```

5. **How can we visualize the key supply chain metrics for better understanding and decision making?**

I used PowerBi and made Dashboard for Visualize the Supply chain Optimization



6. **How can Python be used to conduct statistical analysis to answer important supply chain related questions?**

```
import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

from scipy.stats import ttest_ind

import statsmodels.api as sm


# Step 1: Load the Dataset

data = pd.read_csv('DSS Data.csv')


# Step 2: Data Exploration
```

```
print("Dataset Information:")

print(data.info())

print("\nSummary Statistics:")

print(data.describe())

print("\nMissing Values:")

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

```
Dataset Information:
<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

Summary Statistics:
            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%     51.239830     43.500000               392.500000        6006.352023
75%     77.198228     75.000000               704.250000        8253.976920
max     99.171329    100.000000               996.000000        9866.465458

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

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

       Manufacturing lead time  Manufacturing costs  Defect rates        Costs
count                100.00000           100.000000    100.000000   100.000000
mean                  14.77000            47.266693      2.477158   539.245782
std                    8.91243            28.982841      1.461366   258.301696
min                    1.00000             1.085069      0.218608   113.916248
25%                    7.00000            22.983299      1.209650   328.778455
50%                   14.00000            45.905622      2.341863   530.430444
75%                   23.00000            68.621026      3.763995   773.078231
max                   30.00000            99.466109      5.139255  1007.413450

Missing Values:
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
```

```
# Step 3: Correlation Analysis
# Select relevant numerical columns for correlation
correlation_columns = ['price', 'stock_levels', 'shipping_costs', 'revenue_generated', 'profit']
if all(col in data.columns for col in correlation_columns):
    # Compute correlation matrix
    correlation_matrix = data[correlation_columns].corr()

    # Visualize the correlation matrix
    plt.figure(figsize=(8, 6))
    sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f")
    plt.title("Correlation Matrix of Supply Chain Metrics")
    plt.show()
else:
    print("Not all columns for correlation analysis are available in the dataset.")
```

```
Not all columns for correlation analysis are available in the dataset.
```

```
# Step 4: Hypothesis Testing
# Check if shipping cost differs significantly between carriers
if "shipping_carriers" in data.columns and "shipping_costs" in data.columns:
    # Assuming 'shipping_carriers' is a categorical column with carrier names
    unique_carriers = data['shipping_carriers'].unique()
    if len(unique_carriers) >= 2:
        carrier_1_data = data[data['shipping_carriers'] == unique_carriers[0]]['shipping_costs']
        carrier_2_data = data[data['shipping_carriers'] == unique_carriers[1]]['shipping_costs']

        # Perform a two-sample t-test
        t_stat, p_value = ttest_ind(carrier_1_data, carrier_2_data, equal_var=False)

        print(f"\nT-Test Results for Shipping Costs between {unique_carriers[0]} and
{unique_carriers[1]}:")
```

```python
        print(f"T-Statistic: {t_stat}, P-Value: {p_value}")
        if p_value < 0.05:
            print("Significant difference in shipping costs between the two carriers.")
        else:
            print("No significant difference in shipping costs between the two carriers.")
    else:
        print("Not enough unique carriers for hypothesis testing.")
else:
    print("Required columns for hypothesis testing are missing.")
```

```
Required columns for hypothesis testing are missing.
```

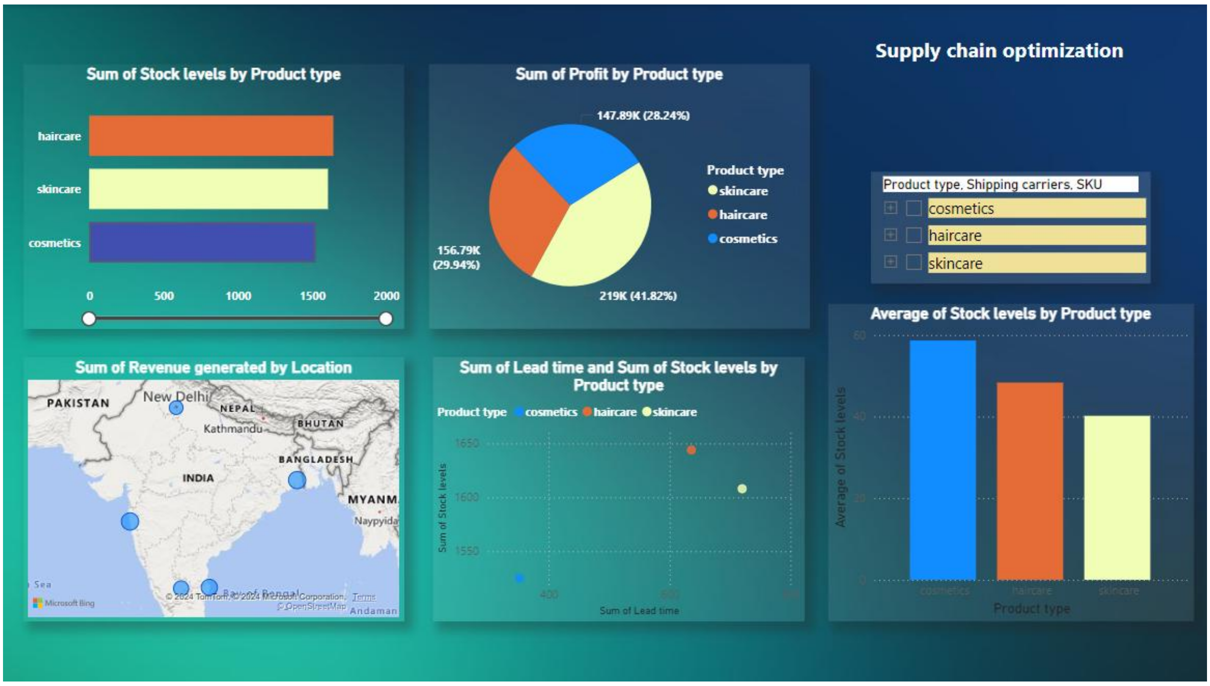
```python
# Step 6: Visualize Key Metrics
# Revenue by Product Type
if "product_type" in data.columns and "revenue_generated" in data.columns:
    plt.figure(figsize=(10, 6))
    sns.barplot(x='product_type', y='revenue_generated', data=data, ci=None, estimator=sum)
    plt.title('Total Revenue by Product Type')
    plt.xlabel('Product Type')
    plt.ylabel('Total Revenue')
    plt.xticks(rotation=45)
    plt.show()
else:
    print("Required columns for revenue analysis are missing.")
```

```
Required columns for revenue analysis are missing.
```

7. What key observations can be made regarding inventory holding, improvements in on-time delivery, and the provision of real-time insights through the implemented Decision Support System?

key observation:

1. Inventory Holding
2. On-Time Delivery
3. Real-Time Insights
4. Lead Times vs Stock Levels
5. Profit Analysis



Final Observation:

Recommendations Based on Observations

1. **Inventory Management:**
    - Use DSS to track inventory turnover and set alerts for slow-moving products.
    - Employ just-in-time (JIT) inventory practices supported by real-time stock monitoring.
2. **On-Time Delivery:**
    - Prioritize carriers with consistently lower shipping times and costs using DSS analysis.
    - Reduce supplier lead times by negotiating contracts and evaluating alternate suppliers.
3. **Real-Time Insights:**

- Continuously monitor and optimize KPIs like revenue, stock levels, and costs using DSS dashboards.