Supply Chain Management Project

* Dataset is divided into **7 tables**, one table for all orders that needs to be assigned a route – *OrderList* table, and 6 additional files specifying the problem and restrictions.
* For instance, the *FreightRates* table describes all available couriers, the weight gaps for each individual lane and rates associated.
* The *PlantPorts* table describes the allowed links between the warehouses and shipping ports in real world. Furthermore, the *ProductsPerPlant* table lists all supported warehouse-product combinations.
* The V*miCustomers* lists all special cases, where warehouse is only allowed to support specific customer, while any other non-listed warehouse can supply any customer.
* Moreover, the *WhCapacities* lists warehouse capacities measured in number of orders per day and the *WhCosts* specifies the cost associated in storing the products in given warehouse measured in dollars per unit.
* Order ID is ID of the order made by the customer, product ID is the specific product ID customer ordered.
* "tpt\_day\_cnt" in the FrieghtRates table means transportation day count, i.e. estimated shipping time.
* WhCapacities correspond to the number of orders. For example, let's say Customer 1 requests 10 units of X, Customer 2 requests 20 units of Y. The total number of orders is 2, thus total capacity in "whCapacity" is 2.
* WhCapacities table is the maximum number of orders that can be processed per each plant, it is not dependant on specific products.
* The OrderList contains historical records of how the orders were routed and demand satisfied. The whCapacities and rest of the tables are the current state constraints of the network. Thus, we can calculate the costs of historical network and also optimize for the new constraints.
* In order to build Linear Programming (LP) model, you would take the following from the OrderList: the product ID that needs to be shipped, the destination port, unit quantity (for cost) and unit weight (for weight constraints). And then use the limits of those constraints from other tables.

**Questions:** There is a Carrier V44\_3 in OrderList table, but it is missing in the FreightRates table? V44\_3 is a carrier that was historically used for supplying given demand, but since it has been discontinued and therefore do not appear in the Freight Rates List. Also, all of the V44\_3 instances are CRF - i.e. customer arranges their own shipping and hence cost is not calculated either way.

**OrderList table:**

Here’s an explanation of each column in your dataset for the OrderList table:

1. **OrderDate:** The date when the order was placed. This is useful for understanding order timelines and scheduling routes based on the time of the order.
2. **Origin Port:** The port from where the shipment or order is originating. Example: PORT09 represents a specific origin port, which could be a shipping dock, distribution center, or warehouse.
3. **Carrier:** The carrier responsible for transporting the order. Example: V44\_3 could represent a specific carrier (such as a trucking company, shipping line, or courier service) in your dataset.
4. **Transport Mode:** If TPT refers to transport type, 1 could represent a specific mode of transportation, such as:

0 = No transport needed

1 = Road transport (trucks)

2 = Sea transport (ships)

3 = Air transport (planes)

4 = Rail transport (trains)

1. **Service Level (CRF):** The level of service requested for the delivery, such as standard, express, or premium. CRF could be a specific code or categorization within your organization’s service levels.
2. **Ship Ahead Day:** This refers to the number of days the shipment is scheduled to ship ahead of the actual required delivery date. This can help with managing buffer time in case of delays.

**Day Count:** The total number of days estimated or allocated for the order to be shipped and delivered.

1. **Ship Late Day Count:** This could indicate how many days late a shipment is from its originally scheduled ship date.
2. **Customer:** Identifies the customer placing the order. Example: V55555\_53 represents a specific customer, often encoded in such a way to identify them uniquely.
3. **Product ID:** A unique identifier for the product being ordered. This helps in tracking different types of products across orders.
4. **Plant Code:** The code representing the manufacturing or fulfillment plant where the product is sourced or produced. Example: PLANT16 could be a specific factory or location.
5. **Destination Port:** The port where the order is to be delivered. Like Origin Port, PORT09 represents a specific location, often a port for shipment.
6. **Unit Quantity:** The number of units of the product ordered. This gives an idea of the order size or volume.
7. **Weight:** The total weight of the order. This is important for logistics, as it affects transportation costs, vehicle load planning, and route optimization.

These columns are primarily used for logistics and route optimization in shipping and supply chain management. Each field plays a crucial role in ensuring that orders are delivered efficiently and on time.

**FreightRates table:**

FreightRates refers to the cost or pricing structure associated with shipping goods from one location to another, typically based on factors such as the weight of the goods, distance between origin and destination, transport mode (e.g., air, sea, road), and service level. These rates are determined by the carrier (the company responsible for transporting the goods) and may vary depending.

**Here’s a breakdown of the columns in the FreightRates table:**

1. **Carrier:** Identifies the transport company or service provider responsible for shipping the goods. Example: V444\_6 represents a specific carrier.
2. **orig\_port\_cd (Origin Port Code):** The code for the port or location from where the shipment originates. Example: PORT08.
3. **dest\_port\_cd (Destination Port Code):** The code for the port or location where the shipment is to be delivered. Example: PORT09.
4. **minm\_wgh\_qty (Minimum Weight Quantity):** The minimum weight of the shipment (in a particular weight range) for which the rate applies. Example: 250.0 means the rate is applicable for shipments of at least 250 kg.
5. **max\_wgh\_qty (Maximum Weight Quantity):** The maximum weight of the shipment for which the rate applies. Example: 499.99 means the rate applies to shipments up to 499.99 kg.
6. **svc\_cd (Service Code):** A code indicating the type of service being offered. Example: DTD might stand for "Door-to-Door" service.
7. **minimum cost:** The minimum charge for shipping within the specified weight range, regardless of the exact weight. Example: 43.2272 is the minimum cost.
8. **rate:** The cost per unit weight for the shipment (usually per kilogram or ton). Example: 0.7132 means the cost per unit weight is 0.7132 currency units (e.g., dollars, euros).
9. **mode\_dsc (Mode Description):** Describes the mode of transport used for shipping, such as AIR, SEA, ROAD. Example: AIR indicates air transport.
10. **tpt\_day\_cnt (Transport Day Count):** The number of days required to transport the goods. Example: 2 means the transport will take 2 days.
11. **Carrier type:** A secondary identifier or classification for the carrier. Example: V88888888\_0 might represent a specific type or subcategory of carrier within the company.

These columns help define the shipping rates based on the carrier, weight, service level, transport mode, and other factors.

**WhCosts table:**

**WH (Warehouse):** This column lists different plant codes (such as PLANT15, PLANT17, etc.). Each plant code represents a specific warehouse or production plant where goods are stored or manufactured.

**Cost/unit:** This column represents the cost per unit for each plant. It indicates how much it costs to handle or process a single unit of goods at each plant. The values provided (e.g., 1.42, 0.43) are the costs per unit, which might reflect warehousing costs, operational costs, or production costs specific to that plant.

**WhCapacity table:**

Here’s a breakdown of the two columns in the table you provided:

**Plant ID:** This column lists plant codes (such as PLANT15, PLANT17, etc.). Each plant code represents a specific manufacturing or storage facility within your supply chain network. These codes help identify individual plants where production or warehousing occurs.

**Daily Capacity:** This column indicates the daily capacity of each plant, expressed as a numeric value. It represents the maximum number of units or items that a plant can produce or handle in a single day. For example, 11 for PLANT15 means that this plant can process or manufacture a maximum of 11 units daily.In summary, this table provides information about the production or handling capacity of each plant, which is crucial for understanding how much product can be processed and for planning purposes within the supply chain.

**ProductPerPlant table:**

This table establishes a relationship between plants and the products they handle, showing which products are associated with each manufacturing or storage facility.

**VMIcustomers table:**

In summary, this table links plants with the customers they serve, showing which customers are assigned to each plant for order fulfillment.

**PLANT PORTtable:**

This table establishes a relationship between plants and the ports they are associated with, indicating which ports are linked to each manufacturing or storage facility for shipping purposes.

The other 6 tables describe the restrictions imposed on the system.

* some customers can only be serviced by a specific plant
* plants and ports have to be physically connected.
* plants can only handle specific items

**Problem Statement in Simple Language**:

There is a carrier named **V44\_3 in the OrderList table, but you can't find it in the FreightRates table**. This is because V44\_3 was used in the past for deliveries, but it is no longer in use. Therefore, it doesn’t show up in the list of current freight rates. Additionally, all orders using V44\_3 are marked as CRF, which means that the customers are taking care of their own shipping. As a result, the shipping cost for these orders is not calculated.

**The presence of the discontinued carrier V44\_3 in the OrderList but not in the FreightRates table creates several issues, such as data integrity problems, operational inefficiencies, and customer service challenges.** This inconsistency can lead to delays in processing orders and financial discrepancies due to unclear shipping costs. If not addressed, these problems may result in increased customer complaints, confusion in logistics operations, financial losses, and damage to the company’s reputation. In summary, resolving the issues related to V44\_3 is crucial to ensure smooth operations, maintain customer satisfaction, and prevent negative impacts on the business.

**Type of Problem Generated by V44\_3**

The presence of the discontinued carrier V44\_3 in the OrderList table but not in the FreightRates table generates several types of problems, primarily:

* **Data Integrity Issues:** The inconsistency between the order records and freight rate availability can lead to confusion in logistics operations.
* **Operational Inefficiencies:** Orders linked to a non-existent carrier may delay processing, as there would be no clear way to calculate shipping costs or arrange transport.
* **Customer Service Challenges:** Customers may be unaware that V44\_3 is no longer in use, leading to potential frustrations when they attempt to arrange shipping.
* **Financial Implications:** Without clear guidelines for shipping costs, the organization may face financial discrepancies, impacting budgeting and forecasting.

**Why This is a Problem Statement**

The original question is a problem statement because it clearly identifies:

**The Issue:** The presence of a discontinued carrier (V44\_3) in the OrderList but absent from the FreightRates table.

**The Context:** V44\_3 was historically used but is now discontinued, leading to inconsistencies in the system.

**The Consequences:** The impact on cost calculations and the arrangements made by customers for their shipping needs.

**Challenges if the Problem is Not Solved**

If the problem related to V44\_3 is not addressed, several challenges could arise:

* **Increased Customer Complaints**: Customers may become frustrated if they try to arrange shipping using a carrier that is no longer available, leading to dissatisfaction and potential loss of business.
* **Delayed Order Processing:** Orders that still reference V44\_3 may face delays due to the lack of available shipping options, which can disrupt delivery schedules and lead to penalties or chargebacks.
* **Operational Confusion:** Logistics and operations teams may experience confusion over how to handle orders associated with the discontinued carrier, leading to errors in shipping arrangements and cost calculations.
* **Financial Losses:** Inaccurate freight cost calculations can result in unexpected expenses that can hurt profitability, as unaccounted costs may arise from late shipments or missed delivery deadlines.
* **Damage to Reputation:** Consistently poor service or confusion in order fulfillment may damage the company's reputation in the market, leading to long-term loss of customers.
* **Inability to Adapt:** Failure to resolve this issue may hinder the organization’s ability to adapt to future changes in carriers or logistics processes, limiting growth and operational efficiency.
* By addressing the problem, organizations can mitigate these challenges, improve operational efficiency, and enhance customer satisfaction.

**Solution**

To solve the problem related to the discontinued carrier V44\_3 using Python in Google Colab, we will follow a systematic approach. Here’s how you can structure your solution:

Step-by-Step Solution

Set Up Google Colab Environment:

Open Google Colab and create a new notebook.

Make sure to upload the Excel dataset containing the OrderList and FreightRates tables.

Import Necessary Libraries:

We'll need libraries like pandas for data manipulation and analysis.

Load the Data:

Read the relevant sheets (like OrderList and FreightRates) from the uploaded Excel file.

Identify and Handle Discontinued Carrier:

Check for instances of V44\_3 in the OrderList.

Update those entries to reflect the discontinuation (e.g., remove or flag them).

Communicate Changes:

Provide alternatives or document the changes for any impacted customers.

Check Data Integrity:

Verify the data integrity after the changes to ensure no further issues arise.

Output Results:

Display the updated tables for verification.

Sample Code in Python

Here’s how you can implement the above steps in Google Colab:

python

Copy code

# Step 1: Set Up Google Colab Environment

# You can upload your Excel file via the file upload feature in Google Colab

# Step 2: Import Necessary Libraries

import pandas as pd

# Step 3: Load the Data

# Load the Excel file

file\_path = '/content/Supply chain logisitcs problem.xlsx' # Change this to your uploaded file path

xls = pd.ExcelFile(file\_path)

# Load the relevant sheets

order\_list\_df = pd.read\_excel(xls, sheet\_name='OrderList')

freight\_rates\_df = pd.read\_excel(xls, sheet\_name='FreightRates')

# Display the data for verification

print("Order List:")

print(order\_list\_df.head())

print("\nFreight Rates:")

print(freight\_rates\_df.head())

# Step 4: Identify and Handle Discontinued Carrier

# Check for occurrences of V44\_3 in the OrderList

v44\_3\_orders = order\_list\_df[order\_list\_df['carrier'] == 'V44\_3']

print("\nOrders with V44\_3:")

print(v44\_3\_orders)

# Handling the discontinued carrier

# Here, we can mark or remove these orders

# For this example, we will create a new column to flag them

order\_list\_df['Discontinued Carrier'] = order\_list\_df['carrier'].apply(lambda x: 'Yes' if x == 'V44\_3' else 'No')

# Step 5: Communicate Changes

# You can create a list of affected customers or orders to communicate the changes

affected\_customers = v44\_3\_orders['customer'].unique()

print("\nAffected Customers:")

print(affected\_customers)

# Step 6: Check Data Integrity

# Verify the updated data

print("\nUpdated Order List:")

print(order\_list\_df.head())

# Step 7: Output Results

# Save the updated Order List to a new Excel file

output\_file\_path = '/content/Updated\_Order\_List.xlsx'

order\_list\_df.to\_excel(output\_file\_path, index=False)

print(f"\nUpdated Order List saved to: {output\_file\_path}")

Explanation of Each Step

Set Up Google Colab Environment: Start by creating a new notebook and uploading your Excel dataset.

Import Necessary Libraries: Use pandas for data manipulation, which provides powerful tools for data analysis.

Load the Data: Load the Excel file and read the specific sheets. You can check the first few rows of each sheet to confirm the data structure.

Identify and Handle Discontinued Carrier: Search the OrderList for any instances of V44\_3. You can flag these entries in a new column or remove them based on your requirements.

Communicate Changes: Create a list of affected customers who used V44\_3. This step helps ensure they are informed about the discontinuation and can make necessary arrangements.

Check Data Integrity: After making changes, review the updated OrderList to confirm that the adjustments have been applied correctly.

Output Results: Finally, save the modified OrderList to a new Excel file for further use or distribution.

Conclusion

By following these steps, you effectively handle the issue of the discontinued carrier V44\_3 in your dataset. This approach ensures data integrity, improves operational efficiency, and helps maintain customer satisfaction. You can customize the code further based on specific requirements or additional analyses you want to perform.

Yes, you can definitely create problem statements based on the additional six sheets in your dataset. Since you've already worked on issues related to the OrderList and FreightRates sheets, we can now look at the other sheets to generate new problem statements and conduct insightful analysis. Here's a structured approach:

Overview of Additional Sheets

Based on your descriptions and earlier questions, the additional sheets in the dataset include information like:

WHCosts (Warehouse Costs): Contains plant codes and warehouse costs per unit.

Customer-Plant Mapping: Shows which plants serve which customers.

Plant Capacities: Contains plant IDs and daily capacities.

Product Mapping: Maps plant codes to product IDs.

Port Mapping: Maps plant codes to specific ports.

Potential Problem Statements and Insights

Here are some potential problem statements you could explore for each of the sheets:

**1. Warehouse Costs (WHCosts) :**

* Problem Statement:

How can we minimize logistics and warehousing costs while maintaining or improving delivery speed?

What are the most cost-efficient plants for fulfilling customer orders based on warehouse costs per unit?

* Insights/Analysis:

Cost Analysis: Analyze which plants have the highest warehouse costs per unit and compare this to the volume of orders they fulfill. High-cost plants may need further optimization or alternate shipping routes.

Optimization: Identify plants with lower costs but under-utilized capacity and suggest redistributing orders to these plants.

* Visualizations:

Bar plot of WHCosts per unit for each plant.

Scatter plot of plant capacity vs. WHCosts per unit to find cost-effective plants.

* **SOLUTION:**

### Step-by-Step Explanation

1. Import Libraries:  
   We start by bringing in the necessary libraries that will help us manipulate data and create visualizations. This is important because we need tools to handle our data and display it in charts.
2. Load Excel File:  
   We load our Excel file and select the sheets we need, which in this case are the "WhCosts" (warehouse costs) and "OrderList" (orders data). This step allows us to access the data we want to analyze.
3. Inspect Column Names:  
   We print the names of the columns in both data sets. This helps us understand the structure of our data and identify what information we have.
4. Clean Column Names:  
   We remove any extra spaces in the column names. This is important because extra spaces can cause problems when we try to use these names in our code later.
5. Rename Column:  
   We change the name of the 'WH' column to 'Plant Code' in the warehouse costs data. This makes it clearer what the data represents, helping us understand our analysis better.
6. Summarize Order Volume:  
   We calculate the total quantity of units ordered from each plant by grouping the order data. This helps us know how much business each plant is handling.
7. Merge DataFrames:  
   We combine the warehouse costs data with the order volume data into one table. This allows us to see both costs and order volumes side by side, making comparisons easier.
8. Calculate Cost Efficiency:  
   We calculate how efficient each plant is by dividing its cost per unit by the total order volume. This helps us identify which plants are costing too much compared to the amount they are delivering.
9. Identify Inefficient Plants:  
   We sort the combined data by cost efficiency to find plants that have high costs and low order volumes. This is important because it highlights areas that need improvement.
10. Plot Warehouse Costs:  
    We create a bar chart that shows the warehouse costs per unit for each plant. This visual representation makes it easier to see which plants have higher costs at a glance.
11. Scatter Plot:  
    We create a scatter plot to show the relationship between total order volume and cost efficiency for each plant. This helps us see if there’s a pattern or trend in how plants perform.
12. Print Inefficient Plants:  
    Finally, we print a list of the inefficient plants along with their costs and order volumes. This provides a clear overview of which plants require further analysis and potential changes to improve efficiency.

### Summary

This code performs a detailed analysis of warehouse costs related to orders from different plants. By visualizing the data and calculating efficiency, it helps identify plants that are underperforming and need optimization. Each step builds on the previous one, leading to insights that can help improve the overall supply chain efficiency.

**2. Customer-Plant Mapping :**

* Problem Statement:

Are customers optimally assigned to plants based on proximity, plant capacity, and product availability?

What are the most efficient customer-plant pairings for improving delivery time and reducing costs?

* Insights/Analysis:

Customer Clustering: Analyze the geographical proximity between plants and customers to determine if customers are being served by the most efficient plant.

Capacity Alignment: Ensure that customers are assigned to plants with adequate capacity, avoiding overloading certain plants while others remain underutilized.

* Visualizations:

Heatmap of customer locations vs. plant capacity.

Network Graph showing customer-plant connections with weights for efficiency (based on costs, distance, and capacity).

**3. Plant Capacities :**

* Problem Statement:

Are plants being used to their full capacity, or is there a mismatch between demand and available capacity?

How can we better distribute orders across plants to optimize capacity utilization?

* Insights/Analysis:

Capacity Utilization: Compare daily capacities of plants to their actual usage from the order data. Plants with higher unused capacity can take on more orders.

Redistribution: Analyze which plants are over-capacity and suggest redistributing the workload to other plants with available capacity.

* Visualizations:

Bar chart of daily capacity vs. actual order volume for each plant.

Gantt Chart for plant utilization over time.

* **SOLUTION:-**

**Step-by-Step Explanation**

1. \*\*Import Libraries\*\*:

We start by importing the necessary libraries:

- `pandas` for data manipulation,

- `matplotlib.pyplot` and `seaborn` for visualizations,

- `plotly.express` for interactive charts.

This is crucial as these libraries provide the functions we need to work with data and create visual representations.

2. \*\*Load Excel File\*\*:

We load the Excel file containing multiple sheets. This enables us to access the different datasets stored within the file for analysis.

3. \*\*Load Specific Sheets into DataFrames\*\*:

We read specific sheets from the Excel file into separate DataFrames:

- `order\_list\_df` for order data,

- `plant\_capacity\_df` for plant capacities.

This allows us to work with only the relevant data we need for our analysis.

4. \*\*Preprocess OrderList for Plant Usage\*\*:

We calculate the total quantity of units ordered for each plant. We group the order data by 'Plant Code' and sum the 'Unit quantity' to get the total order volume per plant. This step helps us understand how much business each plant is processing.

5. \*\*Merge with Plant Capacities\*\*:

We rename the columns in the `plant\_capacity\_df` to ensure consistency and then merge it with the `order\_volume\_df` on 'Plant Code'. This creates a combined DataFrame that contains both the total order volumes and daily capacities for each plant.

6. \*\*Fill Missing Values\*\*:

We fill any missing values in the 'Total Order Volume' column with 0 for plants that have no orders. This ensures that all plants are represented in the analysis, even if they did not receive any orders.

7. \*\*Calculate Capacity Utilization\*\*:

We calculate the capacity utilization for each plant by dividing the total order volume by the daily capacity and multiplying by 100 to express it as a percentage. This metric indicates how effectively each plant is utilizing its capacity.

8. \*\*Visualizations\*\*:

- \*\*Visualization 1\*\*: We create a bar chart comparing daily capacity and actual order volume for each plant. The blue bars represent actual order volumes, while the orange bars represent daily capacities. This visual comparison helps identify how well each plant is performing against its capacity.

- \*\*Visualization 2\*\*: We create a Gantt chart to visualize plant utilization over time. We first convert the 'Order Date' to a datetime format and then group the order data by 'Plant Code' and 'Order Date' to prepare it for the chart. This interactive chart allows us to see how much work each plant is handling over time.

9. \*\*Analysis & Insights\*\*:

We identify plants that are either over-capacity (capacity utilization greater than 100%) or underutilized (capacity utilization less than 50%). This analysis helps pinpoint areas that need improvement.

- We print a list of plants that are over capacity and those that are underutilized to facilitate further examination and decision-making.

**Summary**

This code analyzes the capacity utilization of different plants in a supply chain by comparing order volumes to their capacities. By visualizing the data, we can identify plants that are performing well, those that are overworked, and those that are underutilized. Each step in the code builds upon the previous one to provide insights into how well the supply chain is functioning and where adjustments may be needed.

**4. Product Mapping :**

* Problem Statement:

Are products being sourced from the most optimal plants based on production capabilities and plant capacity?

Is there a need to consolidate product sourcing to specific plants for better efficiency?

* Insights/Analysis:

Product Concentration: Analyze whether certain products are being disproportionately sourced from a small number of plants, potentially causing bottlenecks.

Product-Plant Alignment: Ensure that plants with lower capacities are not overwhelmed with high-demand products.

* Visualizations:

Product Sourcing Bar Chart showing the number of products each plant handles.

Pie Chart of product distribution by plant.

* **SOLUTION:**

### Step-by-Step Explanation

1. Import Libraries:  
   The code begins by importing the necessary libraries:
   * pandas for data manipulation,
   * matplotlib.pyplot for creating static visualizations,
   * seaborn for enhanced visualization options.  
     These libraries provide the tools required for data analysis and visualization.
2. Load Excel File:  
   The code loads the Excel file that contains various data sheets relevant to the analysis. This is done using pd.ExcelFile, which allows us to access specific sheets within the Excel file.
3. Load Data from Relevant Sheets:  
   We read the specific sheet named 'ProductsPerPlant' into a DataFrame called product\_mapping\_df. This DataFrame contains the product mapping data necessary for the analysis.
4. Data Cleaning:  
   We print the column names of product\_mapping\_df to check their format. If the column names do not match the expected format, we rename them to ensure consistency. In this case, 'Plant Code' is renamed to 'Plant\_Code' and 'Product ID' to 'Product\_ID'.
5. Analyze Product Concentration per Plant:  
   We analyze how many unique products each plant handles by grouping the data by 'Plant\_Code' and counting the unique 'Product\_IDs'. This results in a new DataFrame, product\_counts\_per\_plant, which shows the number of unique products for each plant.
6. Visualize Product Concentration - Bar Chart:  
   We create a bar chart using seaborn to visualize the number of unique products each plant handles. The x-axis represents the plant codes, and the y-axis represents the count of unique products. This chart helps identify how many different products are being managed by each plant.
7. Visualize Product Distribution - Pie Chart:  
   We calculate the distribution of products across plants by counting how many products each plant handles. A pie chart is created to visualize this distribution, allowing us to see the proportion of products handled by each plant. This visualization helps identify which plants are handling the most products.
8. Analyze Total Products per Plant:  
   We count the total number of products per plant by using the value\_counts method on the 'Plant\_Code' column and resetting the index. This results in a DataFrame, total\_products\_per\_plant, which displays the count of products each plant handles.
9. Identify Overloaded Plants:  
   We analyze whether certain plants are handling a disproportionately high number of products. We create a new DataFrame, overloaded\_plants, that filters out plants handling more products than the average. This helps identify potential bottlenecks in the supply chain where certain plants may be overwhelmed with too many products.
10. Print Overloaded Plants:  
    Finally, the code prints the list of overloaded plants, providing insights into which facilities might require additional support or resources to manage their workloads effectively.

### Summary

This code analyzes product mapping across various plants in a supply chain. By visualizing product concentration and distribution, we can gain insights into how products are sourced and identify any plants that may be overloaded. Each step builds upon the previous one to provide a comprehensive view of product distribution, ultimately aiding in better supply chain management and optimization.

**5. Port Mapping :**

* Problem Statement:
* Are the plants assigned to the most optimal ports based on shipment volume and geographical proximity?

How can we reduce shipping times and costs by optimizing port assignments?

* Insights/Analysis:

Shipping Optimization: Analyze whether the ports being used are optimal for the geographical distribution of customers.

Port Congestion: Check if certain ports are being over-utilized while others are under-utilized, which can lead to delays and higher costs.

* Visualizations:

Geographical Map of plants, ports, and customers, showing shipment routes.

Port Utilization Bar Chart comparing the number of shipments per port.

* **SOLUTION:-**

### Step-by-Step Explanation

1. Import Libraries:  
   The code begins by importing necessary libraries:
   * pandas for data manipulation,
   * matplotlib.pyplot for creating static visualizations,
   * seaborn for enhanced visualization options,
   * geopandas for geographical data processing and visualization.  
     These libraries provide tools for data analysis and visualization.
2. Load Excel File:  
   The code specifies the path to the Excel file and loads it using pd.ExcelFile. This enables access to multiple sheets within the Excel file for analysis.
3. Load Relevant Sheets:  
   Two specific sheets are loaded into DataFrames: plant\_port\_df (for plant-port mapping) and order\_list\_df (for order details). This prepares the necessary data for the analysis.
4. Check Data Structure:  
   The column names of the loaded DataFrames are printed to verify the structure of the data. This step helps ensure that the data is correctly formatted and contains the expected columns.
5. Data Processing and Analysis:
   * Aggregate Shipment Volume by Port:  
     The code groups the order\_list\_df by 'Destination Port' and sums the 'Unit quantity' to calculate the total shipment volume for each port. This creates a new DataFrame, port\_shipments, that reflects the total shipments handled by each port.
6. Merge Port Shipment Data:  
   The port\_shipments data is merged with the plant\_port\_df using the 'Port' from plant\_port\_df and 'Destination Port' from port\_shipments. This creates a new DataFrame, plant\_port\_shipments, which consolidates shipment data with plant-port mapping.
7. Visualization 1 - Port Utilization Bar Chart:  
   A bar chart is created using seaborn to visualize the shipment volume for each port. The x-axis represents the ports, while the y-axis represents the total shipment volume. This visualization helps identify which ports are utilized the most.
8. Visualization 2 - Geographical Map of Plants, Ports, and Shipments:  
   The code prepares to create a geographical map to visualize the locations of plants and ports, assuming that geographical coordinates (latitude and longitude) are available in the plant\_port\_df.
   * A GeoDataFrame (gdf\_ports) would be created using the coordinates of the ports and plants.
   * A world map is read as a base layer using geopandas.
   * The ports and plants would be plotted on the map.  
     (Note: This section is commented out and requires actual coordinates to function properly.)
9. Insights Based on Analysis:
   * Analyze Port Congestion:  
     The mean shipment volume across all ports is calculated, and ports with shipment volumes above the average are identified as over-utilized, while those below the average are deemed under-utilized.
   * The results are printed, showing which ports are experiencing high or low shipment volumes.

### Summary

This code analyzes port mapping and shipment volumes in a supply chain setting. By visualizing port utilization and examining geographical data, it provides insights into which ports are operating effectively and which may require attention due to congestion or under-utilization. Each step is designed to build a comprehensive understanding of port performance within the supply chain context.

**6. Combining All Data Sources for Holistic Insights :**

* ***5th Problem Statement:-***

***Holistic Supply Chain Optimization::***

* ***How can we optimize the entire supply chain by integrating key factors such as warehousing costs, plant capacities, customer assignments, shipping costs, and port assignments?***

***Solution:-***

***Total Cost Optimization:  
Conduct a thorough, end-to-end analysis of the supply chain to pinpoint opportunities for reducing overall costs, including warehousing, shipping, and capacity utilization.***

***Delivery Speed Optimization:  
Examine how various combinations of plants, ports, and carriers impact delivery speed and customer satisfaction, aiming to identify the configurations that best enhance performance.***

***Visualization:-***

* Problem Statement:

How can we holistically optimize the entire supply chain by considering warehousing, plant capacities, customer assignments, shipping costs, and port assignments?

* Insights/Analysis:

Total Cost Optimization: Perform an end-to-end analysis of the supply chain to identify areas where the overall cost (warehousing, shipping, capacity utilization) can be reduced.

Delivery Speed Optimization: Analyze how different combinations of plants, ports, and carriers affect overall delivery speed and customer satisfaction.

* Visualizations:

Supply Chain Flow Diagram: Showing the flow of products from plants to customers via different ports and carriers.

Heatmap of Total Costs: A heatmap showing how different routes (plant-port-customer combinations) affect total costs.

* **SOME NEW PROBLEM STATEMENTS:**

### 1. Carrier Utilization and Shipping Costs

* Problem Statement: There is a carrier named V44\_3 in the OrderList, which is no longer active in the FreightRates table. Orders using V44\_3 are marked as CRF, indicating that customers manage their shipping, resulting in uncalculated shipping costs.
* Insight: This issue highlights the need for maintaining up-to-date records of active carriers to ensure accurate shipping cost calculations and improve financial tracking. Investigating the volume of orders associated with inactive carriers can help reallocate shipments to more reliable carriers.

### 2. Plant Overutilization

* Problem Statement: Some plants are operating at over 100% capacity, which may lead to delays in fulfilling orders and decreased service levels.
* Insight: Identifying plants that are overutilized allows for better distribution of orders across available plants, enhancing overall capacity utilization and ensuring timely order fulfillment. This can reduce backlogs and improve customer satisfaction.

### 3. Order Fulfillment Delays

* Problem Statement: A significant percentage of orders are delayed beyond the expected delivery time, impacting customer satisfaction and operational efficiency.
* Insight: Analyzing the reasons behind these delays, whether due to shipping, processing, or inventory issues, can provide actionable insights for improving operational processes and enhancing customer satisfaction.

### 4. Freight Rate Anomalies

* Problem Statement: There are discrepancies between freight rates in the FreightRates table and actual shipping costs observed in the OrderList, leading to budget overruns.
* Insight: Conducting a thorough comparison of current freight rates and actual costs can identify anomalies and help negotiate better rates with carriers, ultimately optimizing shipping expenses.

### 5. Port Utilization Inefficiencies

* Problem Statement: Some ports are experiencing congestion while others remain underutilized, leading to increased shipping times and costs.
* Insight: Analyzing shipment volumes per port can reveal opportunities for reallocating resources and optimizing shipping routes, thereby reducing congestion and improving overall shipping efficiency.

### 6. Product Availability Issues

* Problem Statement: Certain products frequently go out of stock or are backordered, resulting in lost sales and dissatisfied customers.
* Insight: By analyzing product availability against order demand, businesses can implement better inventory management practices, ensuring that high-demand products are consistently in stock to meet customer needs.

### 7. Customer-Specific Shipping Patterns

* Problem Statement: Different customers exhibit varied shipping preferences, leading to inefficiencies in fulfillment processes.
* Insight: Understanding these patterns can facilitate tailored shipping strategies, improving service levels and optimizing logistics operations for different customer segments.

### 8. Geographic Distribution of Orders

* Problem Statement: Orders are not evenly distributed geographically, which leads to inefficiencies in routing and shipping costs.
* Insight: Mapping out order distribution can help identify regions where demand is high but supply chain capabilities are limited, allowing for better strategic planning and resource allocation.

### 9. Seasonal Demand Fluctuations

* Problem Statement: There are significant variations in order volumes during different seasons, making it difficult to manage inventory and production schedules.
* Insight: Analyzing historical order data to identify seasonal trends can enhance capacity planning and inventory management, ensuring that resources are allocated effectively during peak periods.

### 10. Impact of External Factors on Logistics

* Problem Statement: External factors, such as economic changes or natural disasters, are affecting logistics performance, causing delays and increased costs.
* Insight: By correlating logistics data with external events, companies can develop contingency plans and enhance resilience in their supply chain operations.

These problem statements provide a strong foundation for identifying key areas for improvement in your logistics operations while offering insights that can guide strategic decisions and optimize overall efficiency.

**TOP 5 Problem Statements :-**

Here are the top 4-5 problem statements from your supply chain logistics data that provide significant insights for improvement:

### 1. Carrier Utilization and Shipping Costs

* **Problem Statement:** There is a carrier named V44\_3 in the OrderList that is no longer active in the FreightRates table. Orders using V44\_3 are marked as CRF, indicating that customers manage their own shipping, resulting in uncalculated shipping costs.
* **Insight:** This highlights the need for maintaining up-to-date records of active carriers to ensure accurate shipping cost calculations. Investigating the volume of orders associated with inactive carriers can help reallocate shipments to more reliable carriers, improving cost tracking and financial management.

### 2. Plant Overutilization

* **Problem Statement:** Some plants are operating at over 100% capacity, which may lead to delays in fulfilling orders and decreased service levels.
* **Insight:** Identifying overutilized plants allows for better distribution of orders across available plants, enhancing overall capacity utilization and ensuring timely order fulfillment. This can reduce backlogs and improve customer satisfaction, ultimately leading to better operational efficiency.

### 3. Freight Rate Anomalies

* **Problem Statement:** Discrepancies exist between the freight rates in the FreightRates table and actual shipping costs observed in the OrderList, leading to budget overruns.
* **Insight:** Conducting a thorough comparison of current freight rates and actual costs can identify anomalies and help negotiate better rates with carriers. This can optimize shipping expenses, leading to significant cost savings and improved budgeting accuracy.

### 4. Port Utilization Inefficiencies

* **Problem Statement:** Some ports experience congestion while others remain underutilized, leading to increased shipping times and costs.
* **Insight:** Analyzing shipment volumes per port can reveal opportunities for reallocating resources and optimizing shipping routes. By addressing port congestion, businesses can enhance shipping efficiency and reduce overall delivery times and costs.

### 5. Customer-Specific Shipping Patterns

* **Problem Statement:** Different customers exhibit varied shipping preferences, leading to inefficiencies in fulfillment processes.
* **Insight:** Understanding these shipping patterns allows for tailored logistics strategies that enhance service levels. Optimizing shipping based on customer preferences can improve overall customer satisfaction and streamline operations.

These problem statements not only identify critical issues within your supply chain logistics but also offer actionable insights that can guide strategic improvements for greater efficiency and customer satisfaction.