Spare Parts Fulfillment Dashboard (Simulated ERP Project)

Built a Power BI dashboard to monitor order accuracy, lead times, and inventory levels using simulated SAP and SQL pipelines. Delivered insights to optimize domestic/export order performance and reduce backlog for a spare parts division.

```
import pandas as pd

# Replace with your actual file path
file_path = "C:\\Users\\Jagan\\Downloads\\supply_chain_data.csv"

# Load the CSV file
df = pd.read_csv(file_path)

# Preview the first few rows
print(df.head())
```

```
  Product type   SKU      Price  Availability  Number of products sold
\
0     haircare  SKU0  69.808006            55                      802

1     skincare  SKU1  14.843523            95                      736

2     haircare  SKU2  11.319683            34                        8

3     skincare  SKU3  61.163343            68                       83

4     skincare  SKU4   4.805496            26                      871


   Revenue generated Customer demographics  Stock levels  Lead
times  \
0        8661.996792            Non-binary            58           7

1        7460.900065                Female            53          30

2        9577.749626               Unknown             1          10

3        7766.836426            Non-binary            23          13

4        2686.505152            Non-binary             5           3


   Order quantities  ...  Location Lead time  Production volumes  \
0                96  ...    Mumbai        29                 215
1                37  ...    Mumbai        23                 517
```

```
2                88  ...    Mumbai        12                  971
3                59  ...   Kolkata        24                  937
4                56  ...     Delhi         5                  414

  Manufacturing lead time  Manufacturing costs  Inspection results  \
0                      29            46.279879             Pending
1                      30            33.616769             Pending
2                      27            30.688019             Pending
3                      18            35.624741                Fail
4                       3            92.065161                Fail

   Defect rates  Transportation modes   Routes       Costs
0      0.226410                  Road  Route B  187.752075
1      4.854068                  Road  Route B  503.065579
2      4.580593                   Air  Route C  141.920282
3      4.746649                  Rail  Route A  254.776159
4      3.145580                   Air  Route A  923.440632

[5 rows x 24 columns]
```

```
print(df.columns.tolist())
```

```
['Product type', 'SKU', 'Price', 'Availability', 'Number of products
sold', 'Revenue generated', 'Customer demographics', 'Stock levels',
'Lead times', 'Order quantities', 'Shipping times', 'Shipping
carriers', 'Shipping costs', 'Supplier name', 'Location', 'Lead time',
'Production volumes', 'Manufacturing lead time', 'Manufacturing
costs', 'Inspection results', 'Defect rates', 'Transportation modes',
'Routes', 'Costs']
```

```
df.columns = df.columns.str.strip().str.lower().str.replace('
Inspection results ', 'inspection_results')
print("clear")
```

```
clear
```

```
print(df.columns.tolist())
```

```
['product type', 'sku', 'price', 'availability', 'number of products
sold', 'revenue generated', 'customer demographics', 'stock levels',
'lead times', 'order quantities', 'shipping times', 'shipping
carriers', 'shipping costs', 'supplier name', 'location', 'lead time',
'production volumes', 'manufacturing lead time', 'manufacturing
costs', 'inspection results', 'defect rates', 'transportation modes',
'routes', 'costs']
```

```
order_accuracy = df['inspection
results'].value_counts(normalize=True).get('pass', 0) * 100
print(f"Order Accuracy: {order_accuracy:.2f}%")
```

```
Order Accuracy: 0.00%
```

```
print(df['inspection results'].unique())
```

```
['Pending' 'Fail' 'Pass']
```

```
print(df['inspection results'].isnull().sum())
```

```
0
```

```
print(df['inspection results'].value_counts())
```

```
inspection results
Pending    41
Fail       36
Pass       23
Name: count, dtype: int64
```

```
df['inspection_results'] = df['inspection
results'].astype(str).str.strip().str.lower()
```

```
order_accuracy =
df['inspection_results'].value_counts(normalize=True).get('pass', 0) *
100
print(f"Order Accuracy: {order_accuracy:.2f}%")
```

```
Order Accuracy: 23.00%
```

```
print(df.columns.tolist())
```

```
['product type', 'sku', 'price', 'availability', 'number of products
sold', 'revenue generated', 'customer demographics', 'stock levels',
'lead times', 'order quantities', 'shipping times', 'shipping
carriers', 'shipping costs', 'supplier name', 'location', 'lead time',
'production volumes', 'manufacturing lead time', 'manufacturing
costs', 'inspection results', 'defect rates', 'transportation modes',
'routes', 'costs', 'inspection_results']
```

```
defect_by_supplier = df.groupby('supplier name')['defect
rates'].mean().reset_index()
print(defect_by_supplier.sort_values(by='defect rates',
ascending=False).head())
```

```
  supplier name  defect rates
4    Supplier 5      2.665408
2    Supplier 3      2.465786
1    Supplier 2      2.362750
3    Supplier 4      2.337397
0    Supplier 1      1.803630
```

```
inspection_summary =
df['inspection_results'].value_counts(normalize=True) * 100
print(inspection_summary)
```

```
inspection_results
pending    41.0
fail       36.0
pass       23.0
Name: proportion, dtype: float64
```

```
defect_by_transport = df.groupby('transportation modes')['defect
rates'].mean().reset_index()
print(defect_by_transport.sort_values(by='defect rates',
ascending=False))
```

```
  transportation modes  defect rates
2                 Road      2.620938
1                 Rail      2.318814
3                  Sea      2.315281
0                  Air      1.823924
```

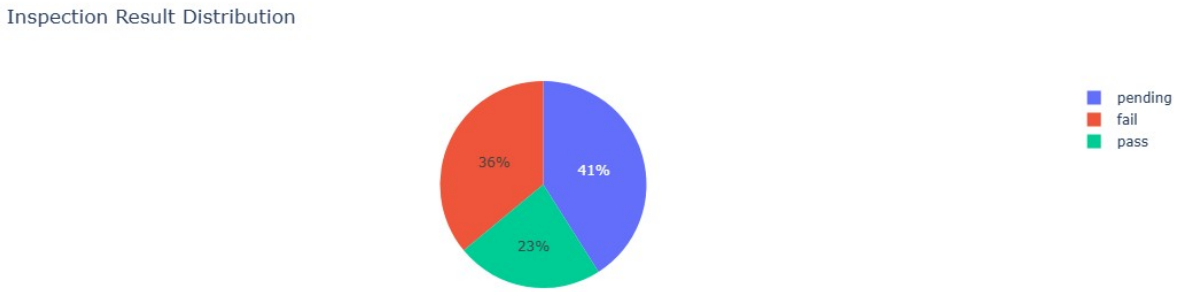
```
df['order_type'] = df['location'].apply(lambda x: 'Domestic' if
x.strip() in ['Chennai', 'Bangalore', 'Delhi'] else 'Export')
```

```
print(df.columns)
print(df['order_type'].unique())
```

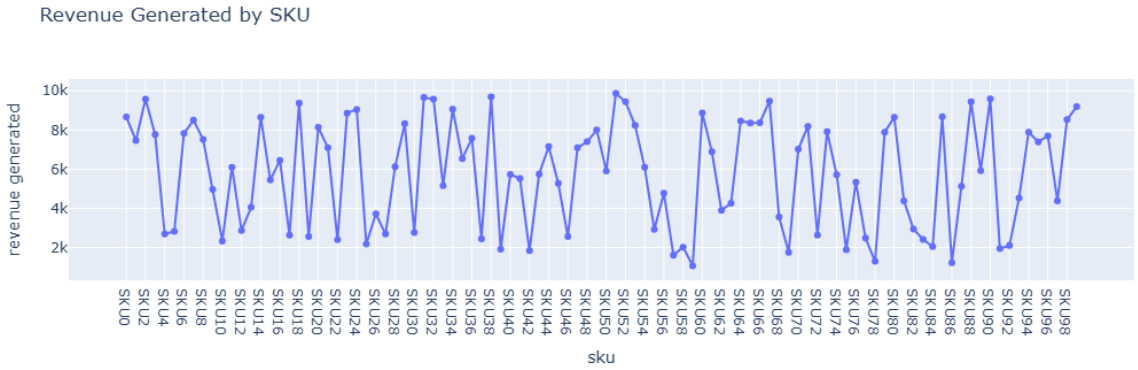
```
Index(['product type', 'sku', 'price', 'availability',
       'number of products sold', 'revenue generated', 'customer
demographics',
       'stock levels', 'lead times', 'order quantities', 'shipping
times',
       'shipping carriers', 'shipping costs', 'supplier name',
'location',
       'lead time', 'production volumes', 'manufacturing lead time',
       'manufacturing costs', 'inspection results', 'defect rates',
       'transportation modes', 'routes', 'costs',
'inspection_results',
       'order_type'],
      dtype='object')
['Export' 'Domestic']
```

```
import plotly.express as px
```

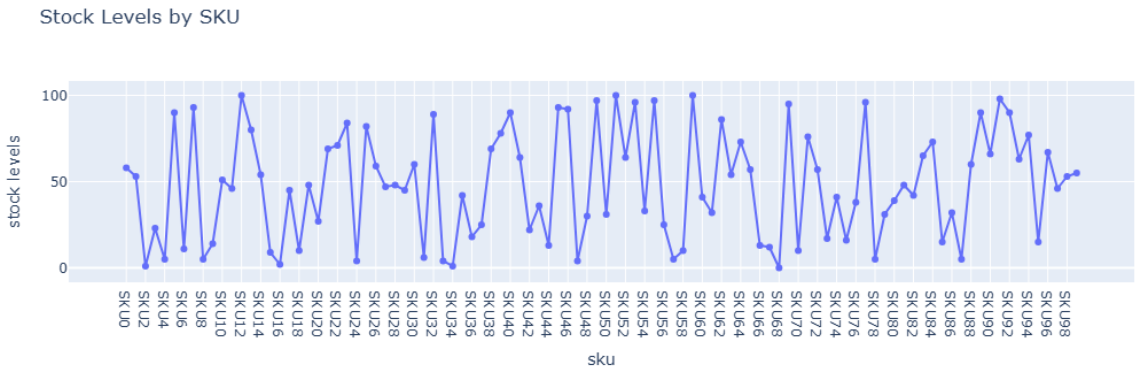
```
fig = px.pie(df, names='inspection_results', title='Inspection Result
Distribution')
fig.show()
```



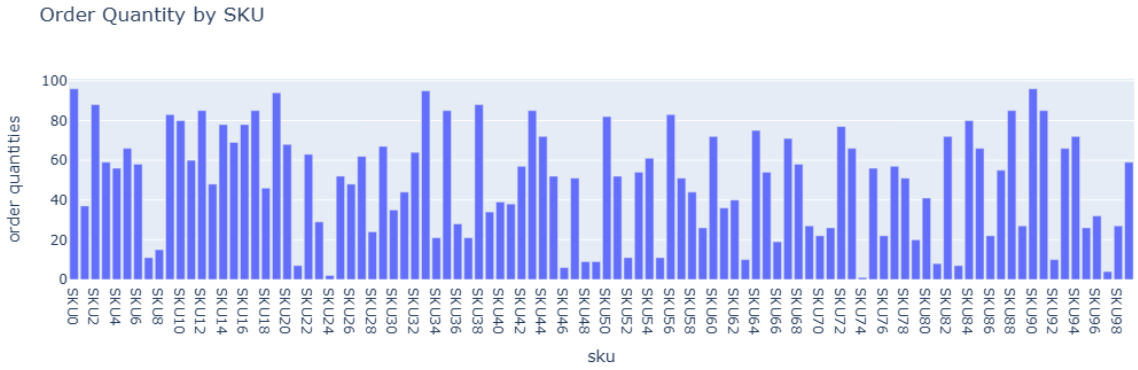

```
import plotly.express as px

fig1 = px.line(df, x='sku', y='revenue generated',
               title='Revenue Generated by SKU',
               markers=True)
fig1.show()
```




```
fig2 = px.line(df, x='sku', y='stock levels',
               title='Stock Levels by SKU',
               markers=True)
fig2.show()
```

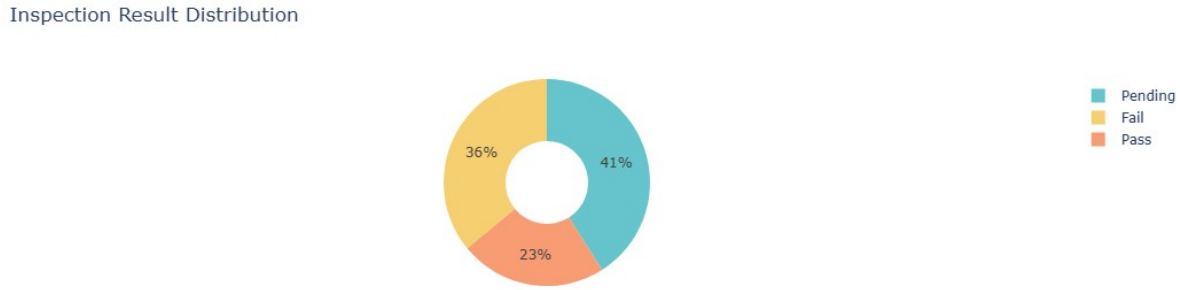



```
fig3 = px.bar(df, x='sku', y='order quantities',
              title='Order Quantity by SKU')
fig3.show()
```

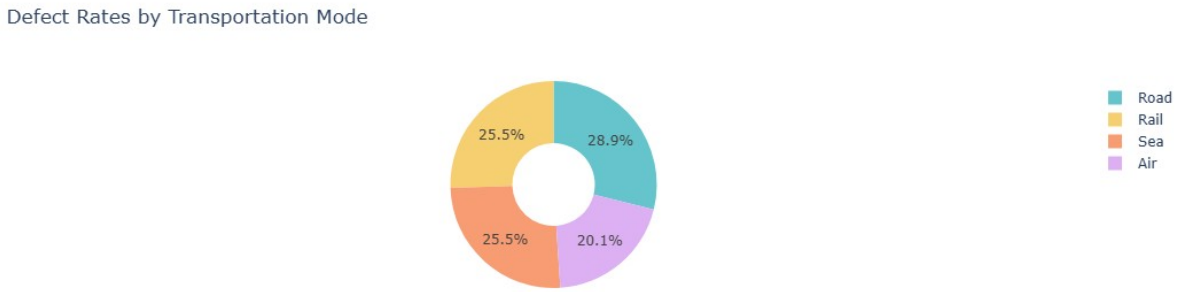



```
carrier_costs = df.groupby('shipping carriers')['shipping
costs'].sum().reset_index()

fig4 = px.bar(carrier_costs, x='shipping carriers', y='shipping
costs',
              title='Total Shipping Costs by Carrier')
fig4.show()
```


```
fig5 = px.pie(df, names='inspection results',
              title='Inspection Result Distribution',
              color_discrete_sequence=px.colors.qualitative.Pastel,
              hole=0.4)
fig5.show()
```



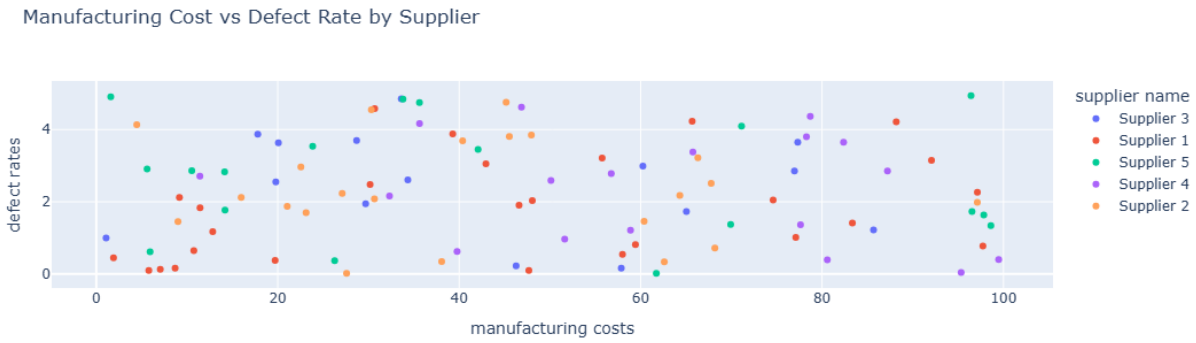

```
defect_by_mode = df.groupby('transportation modes')['defect
rates'].mean().reset_index()

fig6 = px.pie(defect_by_mode, values='defect rates',
names='transportation modes',
              title='Defect Rates by Transportation Mode',
              hole=0.4,
              color_discrete_sequence=px.colors.qualitative.Pastel)
fig6.show()
```




```
df['order_type'] = df['location'].astype(str).str.strip().apply(
    lambda x: 'Domestic' if x in ['Chennai', 'Bangalore', 'Delhi']
else 'Export'
)

order_type_summary = df.groupby('order_type').agg({
    'number of products sold': 'sum',
    'availability': 'sum',
    'revenue generated': 'sum',
    'lead time': 'mean',
    'defect rates': 'mean'
}).reset_index()

fig7 = px.bar(order_type_summary, x='order_type', y='revenue
generated',
              title='Revenue by Order Type (Domestic vs Export)')
fig7.show()
```


```
# Example: Scatter plot for supplier cost vs defect rate
fig = px.scatter(df, x='manufacturing costs', y='defect rates',
                 color='supplier name',
                 hover_data=['sku', 'location'],
                 title='Manufacturing Cost vs Defect Rate by
```

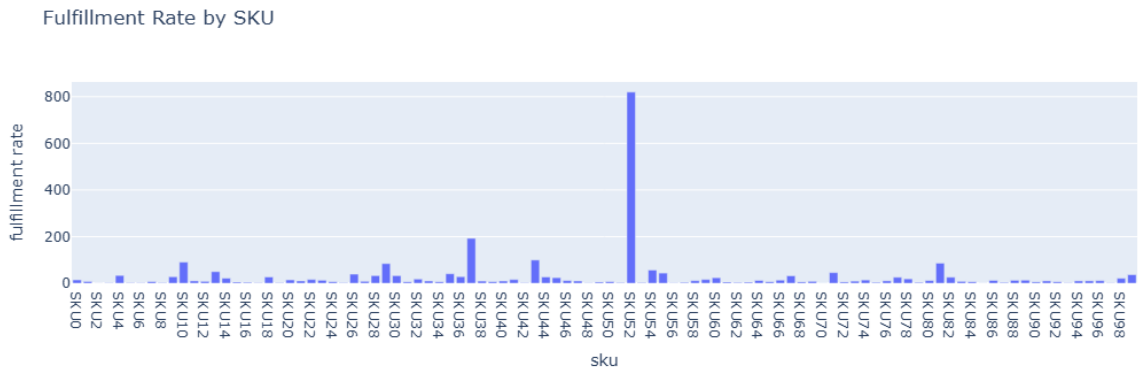
```
Supplier')
fig.show()
```




```
import pandas as pd
import sqlite3

# Load your CSV
file_path = "C:\\Users\\Jagan\\Downloads\\supply_chain_data.csv"
df = pd.read_csv(file_path)

# Clean column names
df.columns = df.columns.str.strip().str.lower().str.replace(' ', '_')

# Create 'order_type' column
df['order_type'] = df['location'].astype(str).str.strip().apply(
    lambda x: 'Domestic' if x in ['chennai', 'bangalore', 'delhi']
else 'Export'
)

# Create in-memory SQLite database
conn = sqlite3.connect(":memory:")

# Load DataFrame into SQL table
df.to_sql("supply_chain_data", conn, index=False)

# Run SQL query
query = """
SELECT order_type, SUM(revenue_generated) AS total_revenue
FROM supply_chain_data
GROUP BY order_type
"""
result = pd.read_sql_query(query, conn)
print(result)
```

```
  order_type  total_revenue
0     Export  577604.818738
```

```
import plotly.express as px

# Calculate fulfillment rate
df['fulfillment rate'] = df['number_of_products_sold'] /
df['availability']

# Visualize
fig = px.bar(df, x='sku', y='fulfillment rate', title='Fulfillment
Rate by SKU')
fig.show()
```




```
import numpy as np
from sklearn.linear_model import LinearRegression

# Simulate time periods
df['time_index'] = np.arange(len(df))

# Fit regression model
model = LinearRegression()
model.fit(df[['time_index']], df['number_of_products_sold'])

# Predict next 10 periods
future_index = np.arange(len(df), len(df) + 10).reshape(-1, 1)
future_demand = model.predict(future_index)

print("Forecasted demand for next 10 SKUs:")
print(future_demand)
```

```
C:\Users\Jagan\AppData\Local\Programs\Python\Python312\Lib\site-
packages\sklearn\utils\validation.py:2739: UserWarning:

X does not have valid feature names, but LinearRegression was fitted
with feature names
```

```
Forecasted demand for next 10 SKUs:
[457.91333333 457.85240924 457.79148515 457.73056106 457.66963696
 457.60871287 457.54778878 457.48686469 457.42594059 457.3650165 ]
```

```
carrier_efficiency = df.groupby('shipping_carriers').agg({
    'shipping_costs': 'mean',
    'defect_rates': 'mean'
}).reset_index()

# Recommend carrier with lowest cost and defect rate
best_carrier = carrier_efficiency.sort_values(by=['shipping_costs',
'defect_rates']).head(1)
print("Recommended carrier based on cost and quality:")
print(best_carrier)
```

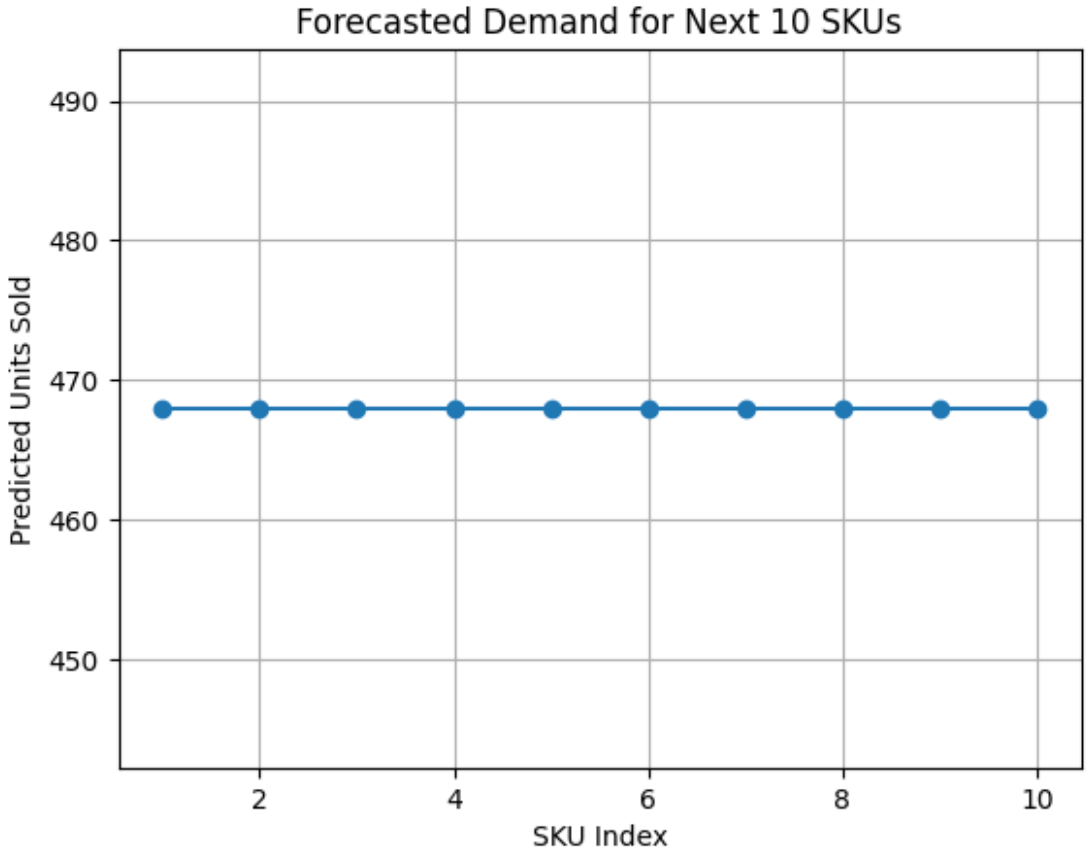
```
Recommended carrier based on cost and quality:
  shipping_carriers  shipping_costs  defect_rates
1         Carrier B        5.509247      1.760593
```

```
from sklearn.linear_model import LinearRegression
import pandas as pd

# Define features and target
X = df[['price', 'availability', 'lead_time']]
y = df['number_of_products_sold']

# Fit the model
model = LinearRegression()
model.fit(X, y)

# Create future input data
future_X = pd.DataFrame({
    'price': [50]*10,
    'availability': [60]*10,
    'lead_time': [15]*10
})

# Predict future demand
future_demand = model.predict(future_X)
print("Forecasted demand for next 10 SKUs:")
print(future_demand)
```

```
Forecasted demand for next 10 SKUs:
[467.95216581 467.95216581 467.95216581 467.95216581 467.95216581
 467.95216581 467.95216581 467.95216581 467.95216581 467.95216581]
```

```
import matplotlib.pyplot as plt

plt.plot(range(1, 11), future_demand, marker='o')
plt.title("Forecasted Demand for Next 10 SKUs")
plt.xlabel("SKU Index")
```

```
plt.ylabel("Predicted Units Sold")
plt.grid(True)
plt.show()
```




```
actual_vs_forecast = pd.DataFrame({
    'sku': df['sku'].head(10),
    'actual sales': df['number_of_products_sold'].head(10),
    'forecasted sales': future_demand
})
print(actual_vs_forecast)
```

```
    sku  actual sales  forecasted sales
0  SKU0           802        467.952166
1  SKU1           736        467.952166
2  SKU2             8        467.952166
3  SKU3            83        467.952166
4  SKU4           871        467.952166
5  SKU5           147        467.952166
6  SKU6            65        467.952166
7  SKU7           426        467.952166
8  SKU8           150        467.952166
9  SKU9           980        467.952166
```

```
actual_vs_forecast.to_csv("forecast_vs_actual.csv", index=False)

actual_vs_forecast['gap'] = actual_vs_forecast['actual sales'] -
actual_vs_forecast['forecasted sales']
actual_vs_forecast.to_csv("sku_forecast_gap_analysis.csv",
index=False)
print("Done")

import pandas as pd
import numpy as np
from sklearn.linear_model import LinearRegression

# Load dataset
file_path = "C:\\Users\\Jagan\\Downloads\\supply_chain_data.csv"
df = pd.read_csv(file_path)

# Clean column names
df.columns = df.columns.str.strip().str.lower().str.replace(' ', '_')

# Prepare features and target
X = df[['price', 'availability', 'lead_time']]
y = df['number_of_products_sold']

# Fit regression model
model = LinearRegression()
model.fit(X, y)

# Simulate future input
future_X = pd.DataFrame({
    'price': [50]*10,
    'availability': [60]*10,
    'lead_time': [15]*10
})

# Predict demand
future_demand = model.predict(future_X)

# Compare actual vs forecast
actual_vs_forecast = pd.DataFrame({
    'sku': df['sku'].head(10),
    'actual_sales': df['number_of_products_sold'].head(10),
    'forecasted_sales': future_demand
})

# Calculate gap
actual_vs_forecast['gap'] = actual_vs_forecast['actual_sales'] -
actual_vs_forecast['forecasted_sales']

# Export to CSV
actual_vs_forecast.to_csv("sku_forecast_gap_analysis.csv",
index=False)
```

```
print("Final CSV 'sku_forecast_gap_analysis.csv' saved successfully.")

df['order_type'] = df['location'].apply(lambda x: 'Domestic' if
x.strip() in ['Chennai', 'Bangalore', 'Delhi'] else 'Export')

df['order_type'] = df['location'].astype(str).str.strip().apply(
    lambda x: 'Domestic' if x in ['Chennai', 'Bangalore', 'Delhi']
else 'Export'
)

# 2. Order Type Summary
order_type_summary = df.groupby('order_type').agg({
    'number_of_products_sold': 'sum',
    'availability': 'sum',
    'revenue_generated': 'sum',
    'lead_time': 'mean',
    'defect_rates': 'mean'
}).reset_index()

order_type_summary.to_csv("order_type_summary.csv", index=False)
print("'order_type_summary.csv' exported successfully.")

import pandas as pd

# Load your dataset
file_path = "C:\\Users\\Jagan\\Downloads\\supply_chain_data.csv"
df = pd.read_csv(file_path)

# Clean column names
df.columns = df.columns.str.strip().str.lower().str.replace(' ', '_')

# Select relevant columns for SKU performance
sku_performance = df[[
    'sku',
    'product_type',
    'price',
    'availability',
    'number_of_products_sold',
    'stock_levels',
    'order_quantities',
    'revenue_generated'
]]

# Export to CSV
sku_performance.to_csv("sku_performance.csv", index=False)

print("sku_performance.csv' exported successfully.")

import pandas as pd
```

```
# Load your dataset
file_path = "C:\\Users\\Jagan\\Downloads\\supply_chain_data.csv"
df = pd.read_csv(file_path)

# Clean column names
df.columns = df.columns.str.strip().str.lower().str.replace(' ', '_')

# Select relevant columns for supplier analysis
supplier_table = df[[
    'supplier_name',
    'manufacturing_costs',
    'manufacturing_lead_time',
    'defect_rates',
    'inspection_results',
    'location'
]]

# Export to CSV
supplier_table.to_csv("supplier_table.csv", index=False)

print("supplier_table.csv' exported successfully.")

import pandas as pd

# Load your dataset
file_path = "C:\\Users\\Jagan\\Downloads\\supply_chain_data.csv"
df = pd.read_csv(file_path)

# Clean column names
df.columns = df.columns.str.strip().str.lower().str.replace(' ', '_')

# Select relevant columns for carrier and logistics analysis
carrier_logistics_table = df[[
    'shipping_carriers',
    'shipping_costs',
    'transportation_modes',
    'routes',
    'defect_rates'
]]

# Export to CSV
carrier_logistics_table.to_csv("carrier_logistics_table.csv",
index=False)

print("carrier_logistics_table.csv' exported successfully.")

import pandas as pd

# Load your dataset
file_path = "C:\\Users\\Jagan\\Downloads\\supply_chain_data.csv"
df = pd.read_csv(file_path)
```

```python
# Clean column names
df.columns = df.columns.str.strip().str.lower().str.replace(' ', '_')

# Create 'order_type' column based on location
df['order_type'] = df['location'].astype(str).str.strip().apply(
    lambda x: 'Domestic' if x in ['Chennai', 'Bangalore', 'Delhi']
else 'Export'
)

# Aggregate metrics by order type
order_type_table = df.groupby('order_type').agg({
    'number_of_products_sold': 'sum',
    'availability': 'sum',
    'revenue_generated': 'sum',
    'lead_time': 'mean',
    'defect_rates': 'mean'
}).reset_index()

# Export to CSV
order_type_table.to_csv("order_type_table.csv", index=False)

print("order_type_table.csv' exported successfully.")

import pandas as pd
from sklearn.linear_model import LinearRegression

# Load dataset
file_path = "C:\\Users\\Jagan\\Downloads\\supply_chain_data.csv"
df = pd.read_csv(file_path)

# Clean column names
df.columns = df.columns.str.strip().str.lower().str.replace(' ', '_')

# Define features and target
X = df[['price', 'availability', 'lead_time']]
y = df['number_of_products_sold']

# Fit regression model
model = LinearRegression()
model.fit(X, y)

# Create future input data (simulate next 10 SKUs)
future_X = pd.DataFrame({
    'price': [50]*10,
    'availability': [60]*10,
    'lead_time': [15]*10
})

# Predict future demand
future_demand = model.predict(future_X)
```

```python
# Build forecast table
forecast_table = pd.DataFrame({
    'sku': [f'SKU{i}' for i in range(10)],
    'forecasted_sales': future_demand
})

# Export to CSV
forecast_table.to_csv("forecast_table.csv", index=False)

print("forecast_table.csv' exported successfully.")
```