Documentation of Logistics Project

(SPREADSHEETS, PYTHON)

- *Ask & Prepare Phase*

During this phase, I searched for a detailed logistics dataset. I found a good one on kaggle, and decided to analyze it.

The scope of the analysis is improving the efficiency of the supply chain with possible solutions that are based on data.

- *Data Cleaning*

This dataset is complex, however it's not that large, only having 100 rows, meaning that both spreadsheets and python are viable solutions for cleaning this dataset.

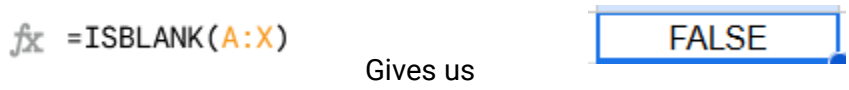
The first step I decided to make is to ensure that all the columns are of the correct type, I did that using this code:

```
import pandas as pd

file_path = r'D:\Work\Datasets\supply_chain_data.csv'
data = pd.read_csv(file_path)

data.info()
```

Which provided the following conclusion, the data types are appropriate for this dataset.

```
<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
```

Now, I'll check the dataset for missing values, duplicates, outliers.



Gives us

```
import pandas as pd

file_path = r'D:\Work\Datasets\supply_chain_data.csv'
data = pd.read_csv(file_path)

if data.duplicated().any():
   print(f"There is {data.duplicated().sum()} duplicate data.")
else:
   print("No duplicates")
```

Returns

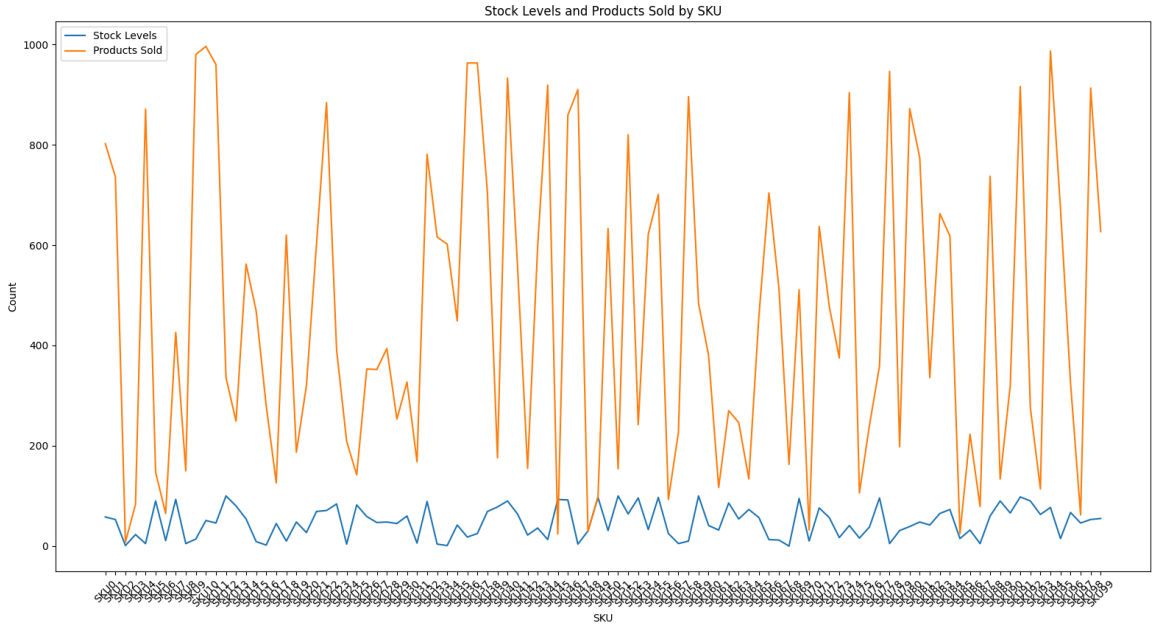
```
"D:\software\Python Projects\Python_DataTest
No duplicates

Process finished with exit code 0
```

Using functions such as MAX and MIN, I found no outliers.

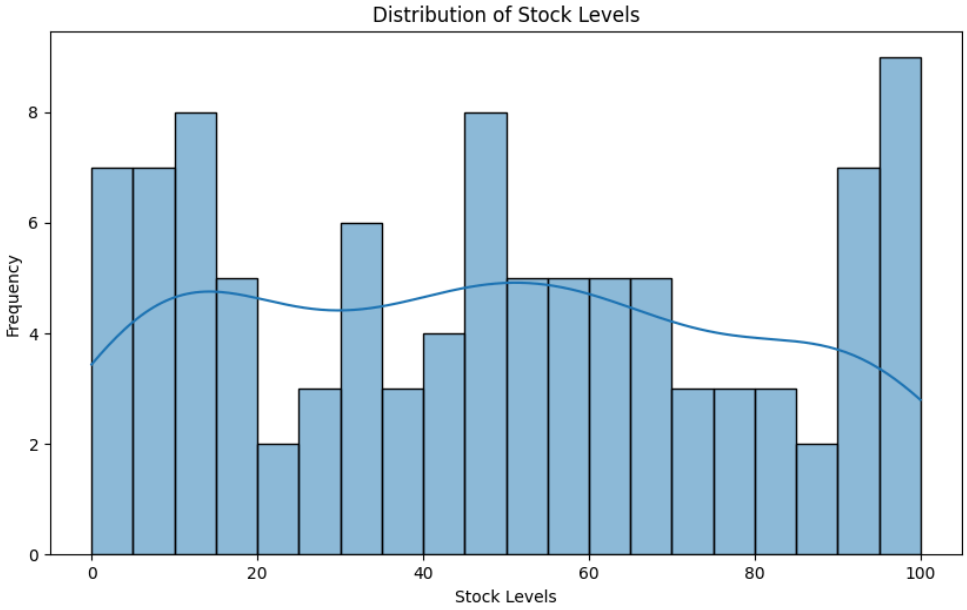
Therefore, the dataset is clean and no modifying is required to perform EDA.

- *Data Analysis*

After writing a python script that compares stock to products sold, we can see one big issue.



The stock levels in a lot of cases are way too low compared to the products sold, this causes massive delays for customers.

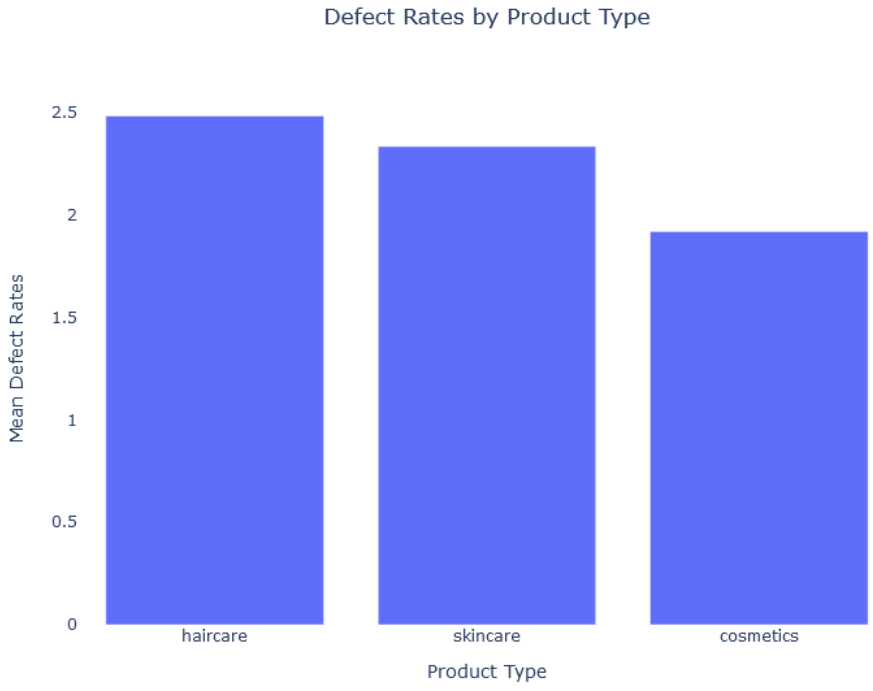


The distribution of stock is all over the place, some stock that isn't needed as much is kept at 100, while needed stock is extremely low.

code:

```
plt.figure(figsize=(12, 6))
sns.lineplot(data=data, x='SKU', y='Stock levels', label='Stock Levels')
sns.lineplot(data=data, x='SKU', y='Number of products sold', label='Products
Sold')
plt.title('Stock Levels and Products Sold by SKU')
plt.xlabel('SKU')
plt.ylabel('Count')
plt.xticks(rotation=45)
plt.legend()
plt.show()

plt.figure(figsize=(10, 6))
sns.histplot(data=data['Stock levels'], bins=20, kde=True)
plt.title('Distribution of Stock Levels')
plt.xlabel('Stock Levels')
plt.ylabel('Frequency')
plt.show()
```

For the next part, I wanted to analyze the defect rates by product type.

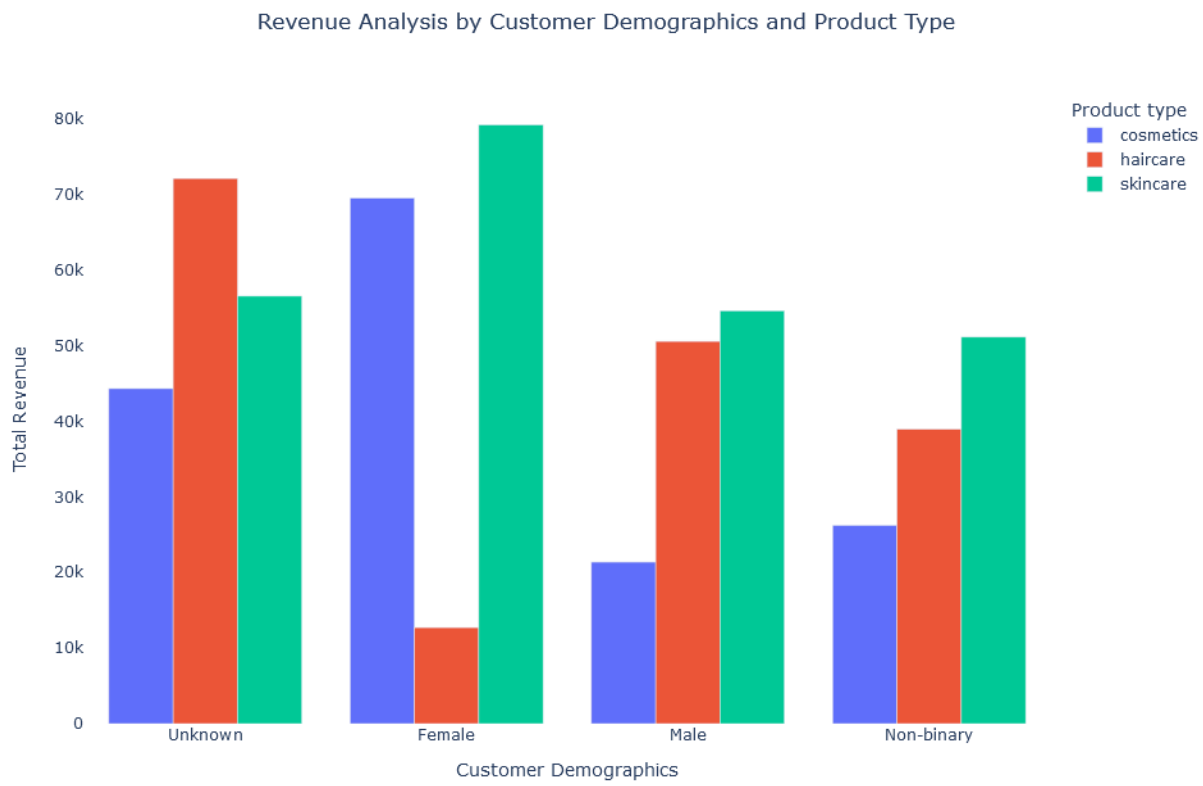


Haircare has the most defects.

Code:

```
defect_rates_by_product = data.groupby("Product type")['Defect
rates'].mean().reset_index()

fig = px.bar(defect_rates_by_product, x='Product type', y='Defect rates',
title='Defect Rates by Product Type')

fig.update_layout(
   xaxis_title="Product Type",
   yaxis_title="Mean Defect Rates",
   xaxis=dict(categoryorder='total descending'),
   yaxis=dict(title='Mean Defect Rates'),
   plot_bgcolor='white',
   title_x=0.5,
   showlegend=True
)


fig.show()
```

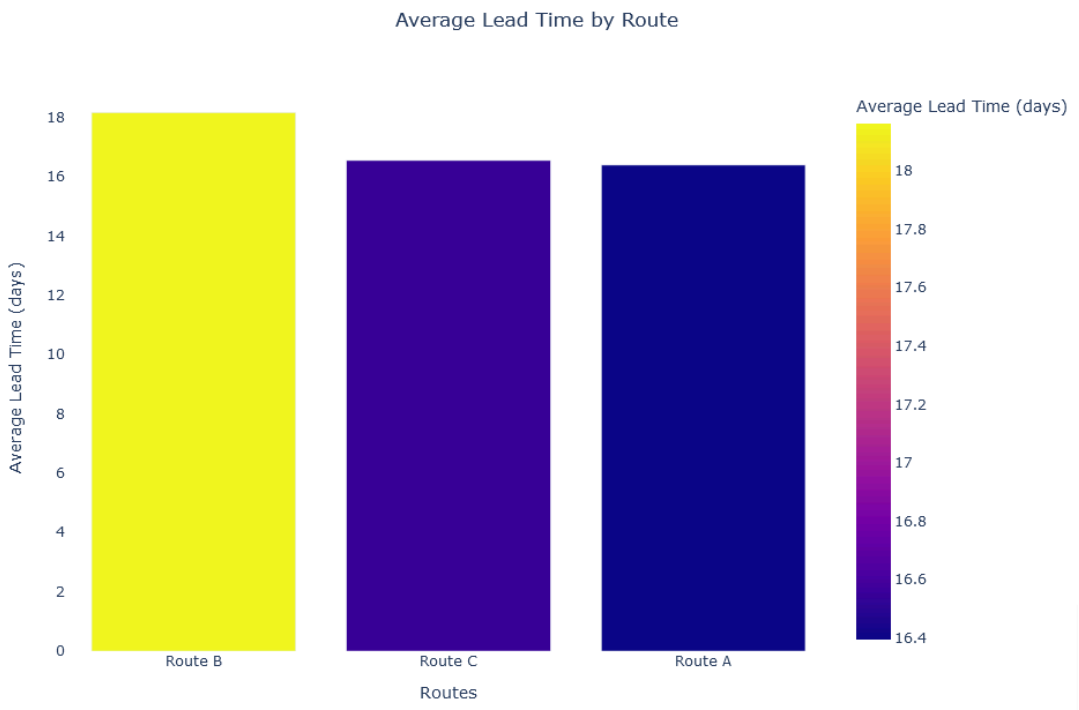
Next, customer demographics based on product type is also important, this graph could help with stock or even business strategies, sending targeted ads/emails to groups that are more likely to buy that product.



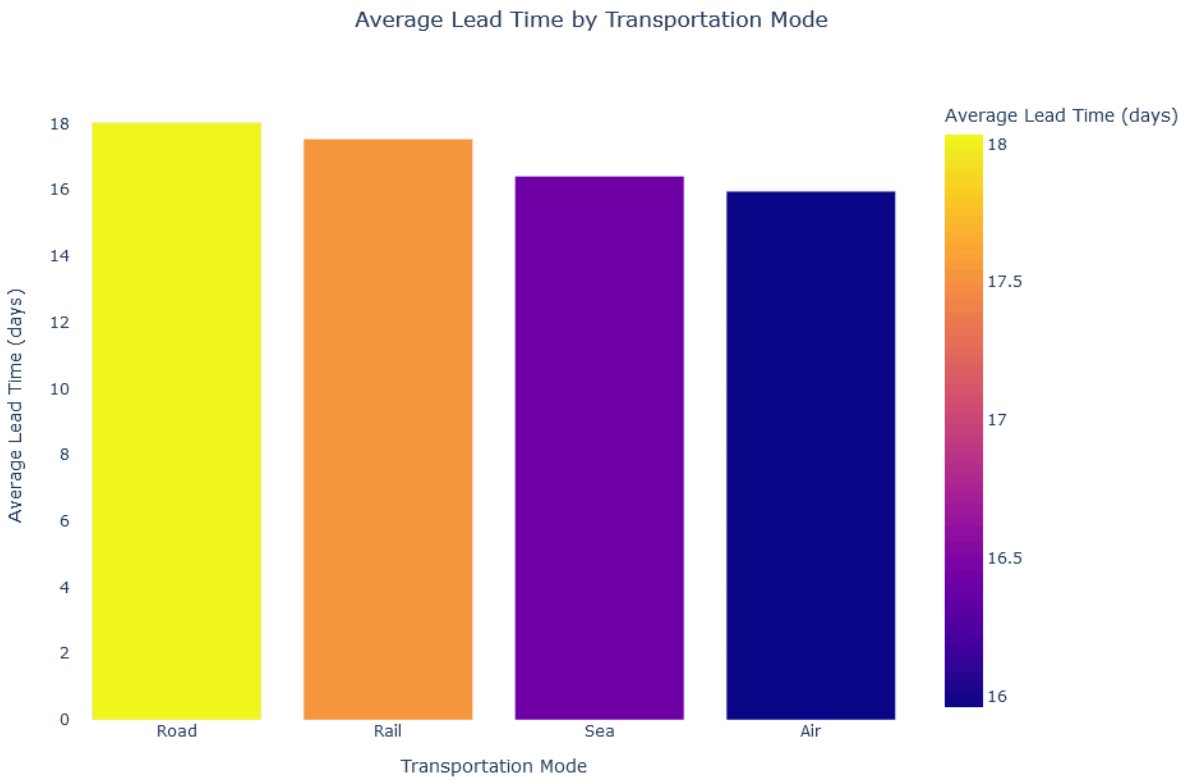
Code:

```
revenue_analysis = data.groupby(['Customer demographics', 'Product
type'])['Revenue generated'].sum().reset_index()


fig = px.bar(
    revenue_analysis,
    x='Customer demographics',
    y='Revenue generated',
    color='Product type',
    title='Revenue Analysis by Customer Demographics and Product Type',
    labels={'Customer demographics': 'Customer Demographics', 'Revenue
generated': 'Total Revenue'},
    barmode='group'
)

fig.update_layout(
    xaxis_title="Customer Demographics",
    yaxis_title="Total Revenue",
    plot_bgcolor='white',
    title_x=0.5,
    xaxis=dict(categoryorder='total descending'),
)

fig.show()
```

Figuring out how routes affect shipments is also important.



Here we can see that route B is the slowest, taking an average of 18 days, versus route A which takes 16.

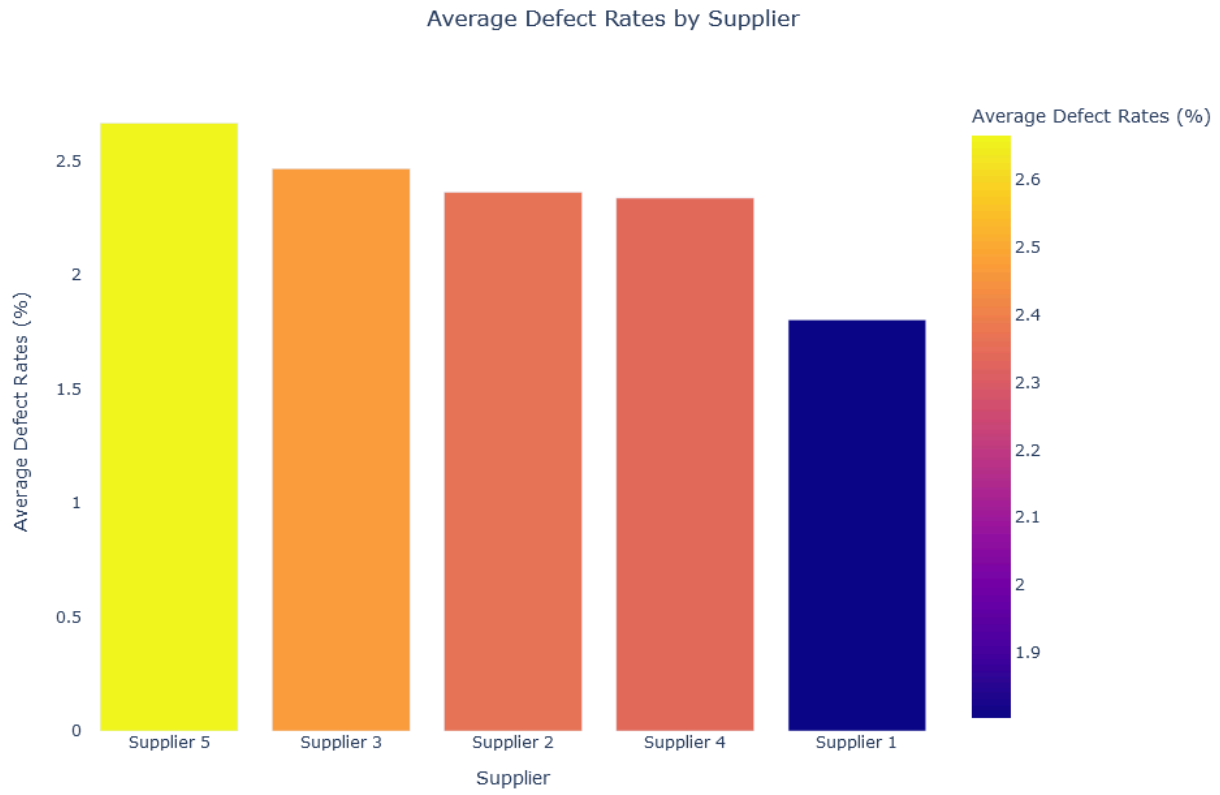


Transportation mode also makes a difference, although it's not that significant, for important shipments planes could reduce the delivery times by a few days.

Code:

```
lead_time_by_route = data.groupby('Routes')['Lead time'].mean().reset_index()

fig_route = px.bar(
    lead_time_by_route,
    x='Routes',
    y='Lead time',
    title='Average Lead Time by Route',
    labels={'Lead time': 'Average Lead Time (days)', 'Routes': 'Route'},
    color='Lead time'
)

fig_route.update_layout(
    xaxis_title="Routes",
```

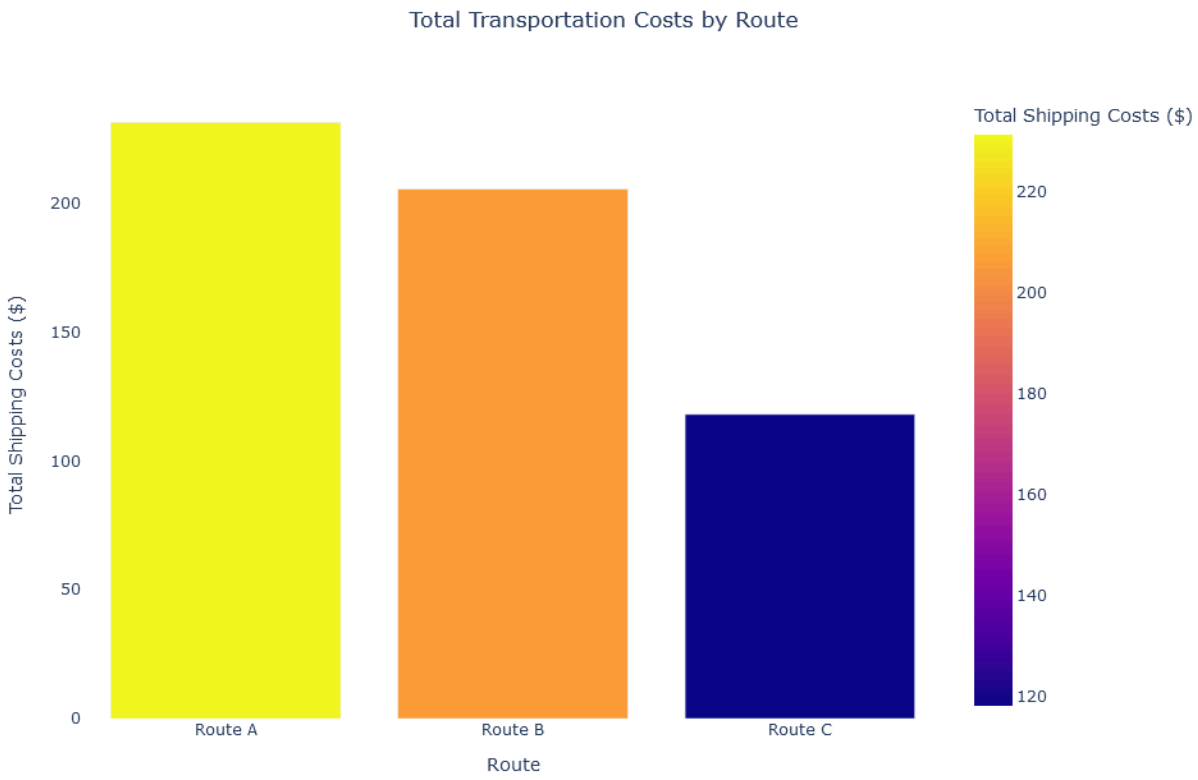
```
    yaxis_title="Average Lead Time (days)",
    plot_bgcolor='white',
    title_x=0.5,
    xaxis=dict(categoryorder='total descending'),
)

fig_route.show()

lead_time_by_transportation = data.groupby('Transportation modes')['Lead
time'].mean().reset_index()

fig_transport = px.bar(
    lead_time_by_transportation,
    x='Transportation modes',
    y='Lead time',
    title='Average Lead Time by Transportation Mode',
    labels={'Lead time': 'Average Lead Time (days)', 'Transportation modes':
'Transportation Mode'},
    color='Lead time'
)

fig_transport.update_layout(
    xaxis_title="Transportation Mode",
    yaxis_title="Average Lead Time (days)",
    plot_bgcolor='white',
    title_x=0.5,
    xaxis=dict(categoryorder='total descending'),
)

fig_transport.show()
```



Code:

```
defect_rates_by_supplier = data.groupby('Supplier name')['Defect
rates'].mean().reset_index()

fig_supplier = px.bar(
    defect_rates_by_supplier,
    x='Supplier name',
    y='Defect rates',
    title='Average Defect Rates by Supplier',
    labels={'Defect rates': 'Average Defect Rates (%)', 'Supplier name':
'Supplier'},
    color='Defect rates'
)

fig_supplier.update_layout(
    xaxis_title="Supplier",
    yaxis_title="Average Defect Rates (%)",
    plot_bgcolor='white',
    title_x=0.5,
    xaxis=dict(categoryorder='total descending'),
)

fig_supplier.show()
```

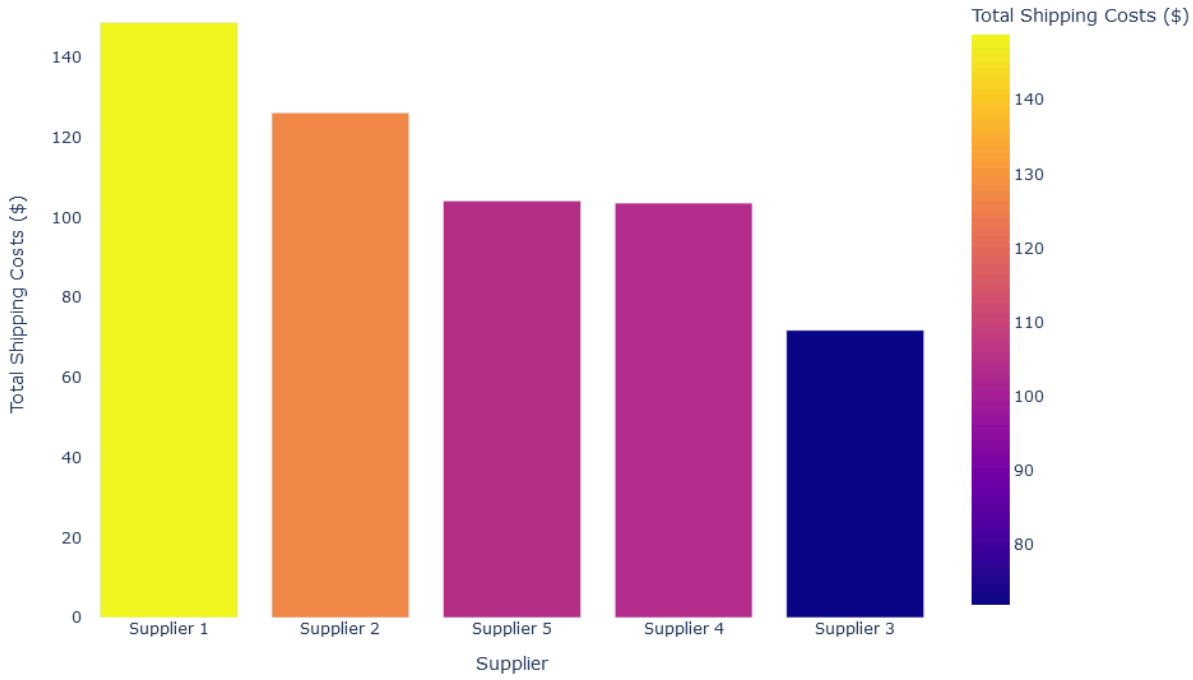
```
defect_rates_by_inspection = data.groupby('Inspection results')['Defect
rates'].mean().reset_index()
```



Code:

```
transportation_costs_by_mode = data.groupby('Transportation modes')['Shipping
costs'].sum().reset_index()

fig_mode = px.bar(
    transportation_costs_by_mode,
    x='Transportation modes',
    y='Shipping costs',
    title='Total Transportation Costs by Mode',
    labels={'Shipping costs': 'Total Shipping Costs ($)', 'Transportation
modes': 'Transportation Mode'},
    color='Shipping costs'
)

fig_mode.update_layout(
    xaxis_title="Transportation Mode",
    yaxis_title="Total Shipping Costs ($)",
    plot_bgcolor='white',
    title_x=0.5,
```

```
    xaxis=dict(categoryorder='total descending'),
)

fig_mode.show()

transportation_costs_by_route = data.groupby('Routes')['Shipping
costs'].sum().reset_index()

fig_route_cost = px.bar(
    transportation_costs_by_route,
    x='Routes',
    y='Shipping costs',
    title='Total Transportation Costs by Route',
    labels={'Shipping costs': 'Total Shipping Costs ($)', 'Routes': 'Route'},
    color='Shipping costs'
)

fig_route_cost.update_layout(
    xaxis_title="Route",
    yaxis_title="Total Shipping Costs ($)",
    plot_bgcolor='white',
    title_x=0.5,
    xaxis=dict(categoryorder='total descending'),
)

fig_route_cost.show()
```


Total Shipping Costs by Supplier
Total Shipping Costs ($)
140
120
100
80
60
40
20
0
Supplier 1
Supplier 2
Supplier 5
Supplier 4
Supplier 3
Supplier
Total Shipping Costs ($)
140
130
120
110
100
90
80