Supply Chain Dataset Analysis

DEPI Final Project

Table of Contents

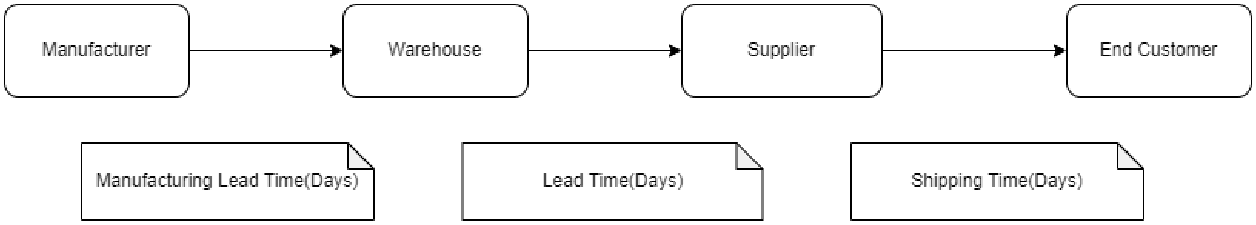
Team Members and Job Description:

1. **Aly Mohamed Salama Elsharkawy:** Aly was the team leader. He performed the initial data pre-processing and oversaw the other team members. Furthermore, he wrote the documentation.
2. **Amira Osama Abdelghany:** Amira created the Power BI dashboard. Furthermore, she also performed the data analysis in Python using Jupyter.
3. **Marina Magdy Khalaf:** Marina assisted in the database side of the project. She helped create the scripts for data retrieval from the database.
4. **Ramy Albert Mouris:** Ramy was in charge of the database side of the project. He normalized the pre-processed data, created a database, and inserted the pre-processed and normalized data.
5. **Sarah Hamed Abdelhakim Muhammed:** Sarah created the presentation for the project. She also assisted in the data analysis phase. Lastly, she was the person who created the team.

Overview:

Dataset Summary:

The dataset consists of data related to a supply chain. It has 100 rows and 24 columns. The columns contain various metrics about the state of a supply chain and its products within. Each row contains row contains information about a product’s SKU (Stock Keeping Unit). There are several groups in the supply chain, namely manufacturers, suppliers, carriers, and customers. Furthermore, there are several delays between the transport of goods and materials between each of them. This relationship is explained in a diagram below.



Analysis Objectives:

The scope of this Analysis project is to improve the overall efficiency of the supply chain contained in the source dataset.

1. **Improving Inventory Management:** By analyzing stock levels, availability percentages, and lead times, the project aims to identify patterns that help optimize inventory management. The goal is to ensure that availability is aligned with demand while reducing overstocking and stockouts.
2. **Reducing Supply Chain Costs:** The project aims to streamline the logistics process by analyzing shipping times, shipping costs, total costs, and transportation modes. This could involve selecting more cost-effective transport routes and modes or upgrading existing transport routes and modes.
3. **Reducing Defect Product Rate:** The project aims to analyze defective products and to find trends, insights, and causes as to why the products became defective.
4. **Evaluating Manufacturer and Production Efficiency:** The project aims to analyze trends in manufacturing time and cost to evaluate the performance of the manufacturers.

Technologies Used:

1. **Python:** Python was used for various helper scripts during the project. It was mainly used to read and merge data from the database. Furthermore, it was also used to pre-process and clean the data.

2. **Jupyter Notebook**: This was used to analyze the data. It was used for its ability to provide a consistent environment with all required libraries (seaborn, matplotlib, pandas, etc.) preinstalled
3. **Microsoft SQL Server**: This was used during the database phase of the project. However, it was not used in any other phases.

Column Meanings:

Column Name	Column Description
SKU	Unique identifier for each product entry.
Type	The category or type of the product (e.g., Haircare, Skincare).
Price	The selling price of the product.
Availability	Percentage of time product was in stock.
Quantity_Sold	Number of units sold for the product.
Revenue	Total revenue generated from the product sales.
Customer_Gender	Gender of the customer (e.g., Male, Female, Non-binary, Unknown).
Stock_Level	Current number of units remaining in stock.
Lead_Times	Average of Lead times in days for the product to be ready for sale.
Total_Orders	Total number of orders placed for the product.
Manufacturer_Location	The location of the manufacturer.
Lead_Time	Time taken by the manufacturer to produce and deliver the product.
Production_Volume	Quantity of the product produced by the manufacturer
Manufacturing_LT	Time taken to manufacture the product.
Manufacturing_Cost	The cost incurred to manufacture the product.
Inspection_Result	Quality inspection status for the product (e.g., Pass, Fail, Pending).
Defect_Rate	The percentage of defective products produced.
Transport_Mode	Mode of transportation for shipping the product (e.g., Road, Air, Rail).
Route_ID	Identifier for the shipping route.
Total_Cost	Total cost associated with the product (including production, shipping, etc.).
Shipping_Time	Time taken to ship the product to its destination.
Carrier_ID	Identifier for the shipping carrier.
Shipping_Cost	The cost of shipping the product.
Supplier_ID	Unique identifier for the supplier providing the product.

Challenges Encountered:

One of the main challenges encountered was understanding the dataset. The column names in the original dataset had obscure names. Furthermore, using the dataset correctly requires knowledge of supply chain terminology. This challenge was surmounted in the pre-processing stage. In this stage, columns were renamed and their meanings were documented. The above table is the result of the second stage of the project. In this stage, the columns were renamed into a database friendly format and uploaded to a Microsoft SQL server.

Methodology:

Overview:

The project was divided into several sections: pre-processing, database insertion and retrieval, analysis, and visualization. The pre-processing stage consisted of understanding the dataset's numerous columns. The database stage consisted of normalizing the dataset into several tables, inserting the data into a suitable database, and reassembling the data from the database using a python helper script. Each phase was dependent on its previous phases. In other words, the output of one phase was the input of another.

Pre-Processing:

As stated above, this stage was related to dealing with and understanding the dataset. The dataset contained no explanation of the meaning of its columns. Thus, we had to perform research and find a best effort answer. Furthermore, there was also no indication of units in the dataset. Thus, the units had to be assumed on a best effort basis. This resulted in almost all of the columns being renamed. An excerpt of the python script responsible is below. Furthermore, some decimals in the dataset were horrendously long. They were thus rounded to two digits of precision.

Note: The name of each script is always the first comment in each file unless it was used to create a plot and was taken from the Jupyter notebook

```
# Graduation_Project_Clean.py
# Importing dataset
df = pd.read_csv("./supply_chain_data.csv")

# Rename columns for more sensible capitalization and naming
df.rename(columns={"Product type": "Type", "Number of products sold": "Quantity Sold
(Units)","Price": "Price(USD)","Revenue generated": "Revenue(USD)", "Customer
demographics": "Customer Gender", "Stock levels": "Stock Level(Units)", "Shipping
carriers": "Carrier ID","Production volumes": "Production Volume(Units)",
"Manufacturing lead time": "Manufacturing LT(Days)","Inspection results": "Inspection
Result","Defect rates": "Defect Rate(%)", "Transportation modes": "Transport
Mode","Routes": "Route ID", "Costs": "Total Cost(USD)", "Supplier name": "Supplier
ID","Shipping times": "Shipping Time(Days)", "Shipping costs": "Shipping
Cost(USD)","Lead time": "Lead Time(Days)", "Location": "Manufacturer Location","Order
quantities": "Total Orders", "Availability": "Availability(%)","Manufacturing costs":
"Manufacturing Cost(USD)"}, inplace=True)
```

Database Insertion and Retrieval:

This phase consisted of first renaming the columns of the dataset in order to make the names more sensible for a database. The next phase consisted of normalizing the dataset into several smaller tables. This was followed by the creation of the database. The dataset was then bulk inserted into the database and a python script was used to reassemble the dataset. The first python script is shown below, followed by the SQL queries, and the final reassemble script.

```python
# split_data.py
import pandas as pd
import re
# Import original pre-processed dataset
original = pd.read_csv(r"C:\Users\Aly\Desktop\DEPI Assignmnets\Graduation Project\Cleaned_Data.csv")
# Create subsets
Product_DF = original[["SKU", "Type", "Price(USD)", "Availability(%)"]]
Sales_DF = original[["SKU", "Quantity Sold (Units)", "Revenue(USD)", "Customer Gender", "Stock Level(Units)", "Lead Times", "Total Orders"]]
Shipping_DF = original[["SKU", "Shipping Time(Days)", "Carrier ID", "Shipping Cost(USD)", "Supplier ID"]]
Manufacturing_DF = original[["SKU", "Manufacturer Location", "Lead Time(Days)", "Production Volume(Units)", "Manufacturing LT(Days)", "Manufacturing Cost(USD)", "Inspection Result", "Defect Rate(%)", "Transport Mode", "Route ID", "Total Cost(USD)"]]

# We need to rename the columns to give them a more consistent database naming scheme
def getCleanedName(inputName):
    # Remove spaces
    newName = inputName.replace(" ", "_", 1)
    # Remove parentheses
    regularExpression = r'\s*\(.*\)\s*'
    newNameRegexed = re.sub(regularExpression, "", newName)
    return newNameRegexed

original_DFCleaned = original.copy()
original_DFCleaned.columns = [getCleanedName(column) for column in original_DFCleaned.columns]
original_DFCleaned.to_csv("Ramy_Names.csv", index=False)
```

```
Product_DF.columns = [getCleanedName(column) for column in Product_DF.columns]
Sales_DF.columns = [getCleanedName(column) for column in Sales_DF.columns]
Shipping_DF.columns = [getCleanedName(column) for column in Shipping_DF.columns]
Manufacturing_DF.columns = [getCleanedName(column) for column in
Manufacturing_DF.columns]

# Write CSVs to disk
Product_DF.to_csv("PRODUCTS.csv", index=False)
Sales_DF.to_csv("SALES.csv", index=False)
Shipping_DF.to_csv("SHIPPING.csv", index=False)
Manufacturing_DF.to_csv("MANUFACTURING.csv", index=False)
```

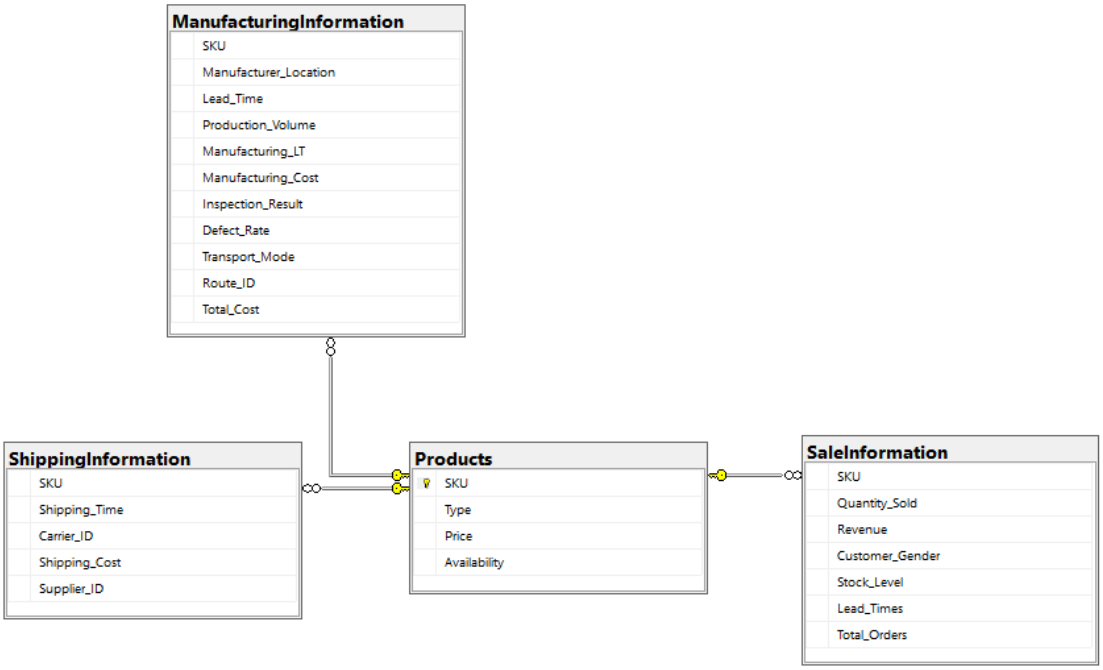
The database was then created using a series of SQL queries:

```
use DEPI_GRAD_PROJECT
CREATE TABLE Products(SKU INT PRIMARY KEY, Type VARCHAR(50), Price DECIMAL (10,2),
Availability INT);

CREATE TABLE SaleInformation(SKU INT, Quantity_Sold INT, Revenue DECIMAL(10,2),
Customer_Gender VARCHAR(30), Stock_Level INT, Lead_Times INT, Total_Orders
INT,FOREIGN KEY (SKU) REFERENCES Products(SKU));

CREATE TABLE ShippingInformation(SKU INT, Shipping_Time INT, Carrier_ID VARCHAR(5),
Shipping_Cost DECIMAL (10,2), Supplier_ID INT, FOREIGN KEY (SKU) REFERENCES
Products(SKU)
);

CREATE TABLE ManufacturingInformation( SKU INT, Manufacturer_Location VARCHAR(50),
Lead_Time INT, Production_Volume INT, Manufacturing_LT INT, Manufacturing_Cost
DECIMAL (10,2), Inspection_Result VARCHAR(50), Defect_Rate DECIMAL (10,2),
Transport_Mode VARCHAR(50), Route_ID VARCHAR(5), Total_Cost DECIMAL (10,2), FOREIGN
KEY (SKU) REFERENCES Products(SKU));
```

Data was then bulk inserted into the database using another series of SQL queries. The queries are identical except for the table name. Thus, only 1 query will be shown for brevity.

```
use DEPI_GRAD_PROJECT
BULK INSERT Products
FROM 'C:\Users\Aly\Desktop\DEPI Assignmnets\Graduation Project\PRODUCTS.csv'
WITH
(
    FIELDTERMINATOR =',',
    ROWTERMINATOR = '\n',
    FIRSTROW =2
)
```

The above operations result in a database with a star schema containing all of the dataset's data. This allows us to begin reassembling the dataset from the database. This was the final step in this phase. The dataset resulting from this phase will then be used in the analysis phase. A diagram of the database schema is below.



The below script will read and merge the data from the database. This marks the end of the Database phase.

```
import pandas as pd

import pyodbc as pyd


DRIVER_NAME = "ODBC Driver 18 for SQL Server"

SERVER_NAME = "DESKTOP-PF9MVGB"

DATABASE_NAME = "DEPI_GRAD_PROJECT"

CONNECTION_STRING = f"""
```

```
DRIVER={{{DRIVER_NAME}}};
SERVER={SERVER_NAME};
DATABASE={DATABASE_NAME};
Trusted_Connection=yes;
Encrypt=no;"""
CONNECTION_OBJECT = pyd.connect(CONNECTION_STRING)

Products_DF = pd.read_sql_query("SELECT * FROM Products", CONNECTION_OBJECT)
Sales_DF = pd.read_sql_query("SELECT * FROM SaleInformation", CONNECTION_OBJECT)
Manufacturing_DF = pd.read_sql_query("SELECT * FROM ManufacturingInformation",
CONNECTION_OBJECT)
Shipping_DF = pd.read_sql_query("SELECT * FROM ShippingInformation",
CONNECTION_OBJECT)

Final_DF = Products_DF.merge(Sales_DF, on="SKU", how="outer").merge(Manufacturing_DF,
on="SKU", how="outer").merge(Shipping_DF, on="SKU", how="outer")
Final_DF.to_csv("Reassembled_CSV.csv", index=False)
```

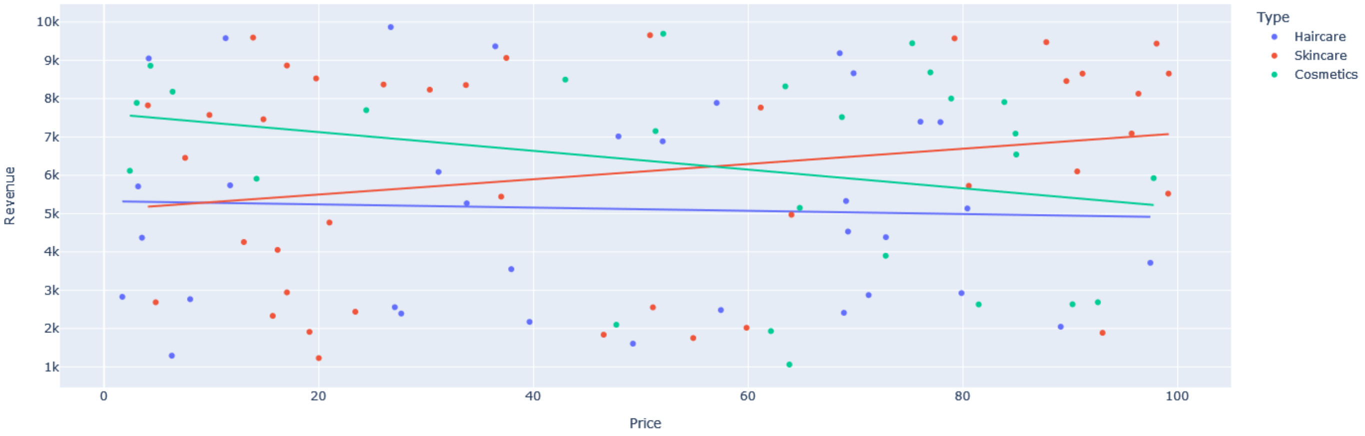
Analysis Phase:

In this phase, we used the results of previous phases to extract insights from the reassembled dataset.

Product-Revenue Relationship:

It was discovered that “Skincare” is the most profitable category of product. There is also a relation between the price of a product and the revenue from that product.

```
fig = px.scatter(df,x='Price',y='Revenue',
                 color='Type',
                 hover_data = ['Quantity_Sold'],
                 trendline='ols')
fig.show()
```

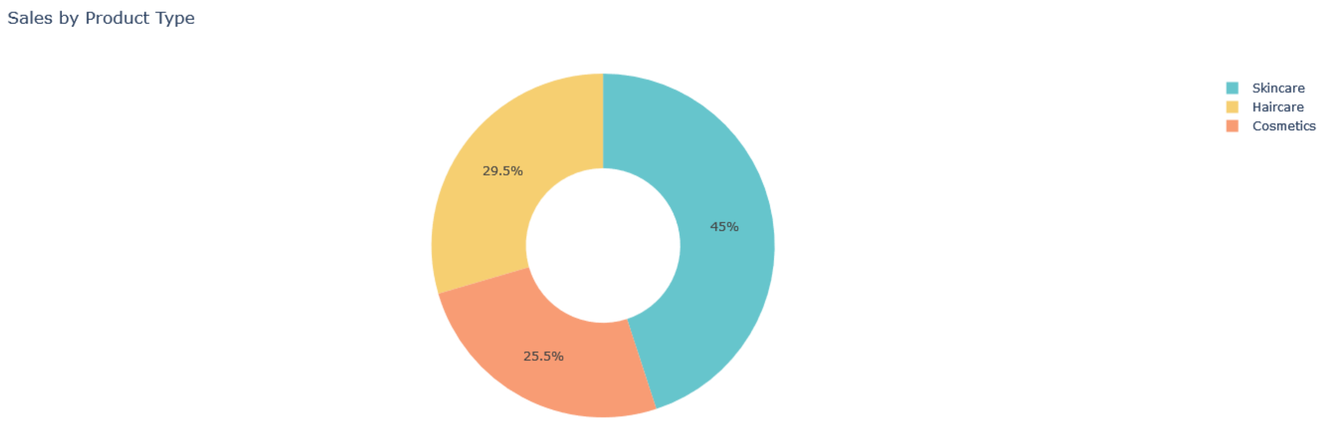


Sales by Product Type:

The previous conclusion is logical because Skincare is the most popular category of products sold:

1. Cosmetics: 11757 Units
2. Haircare: 13611 Units
3. Skincare: 20731 Units

```
pie_chart = px.pie(df, values='Quantity_Sold', names='Type',
                   title='Sales by Product Type',
                   hover_data=['Quantity_Sold'],
                   hole=0.45,
                   color_discrete_sequence=px.colors.qualitative.Pastel)
pie_chart.update_traces(textposition='inside',textinfo='percent')
pie_chart.show()
```



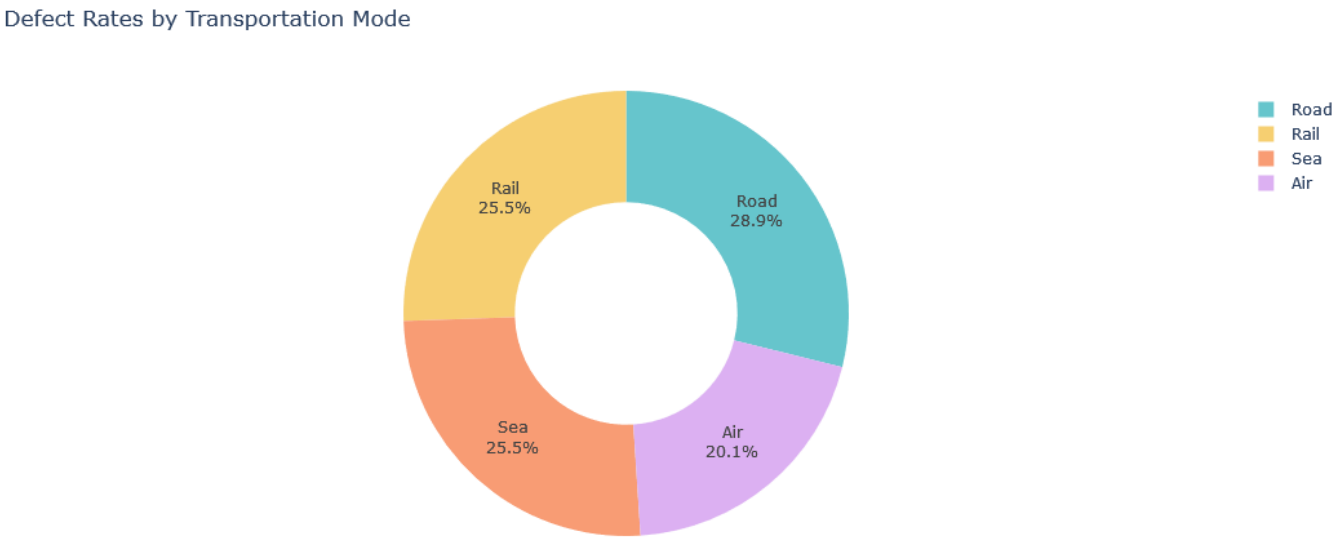
Shipping Analysis:

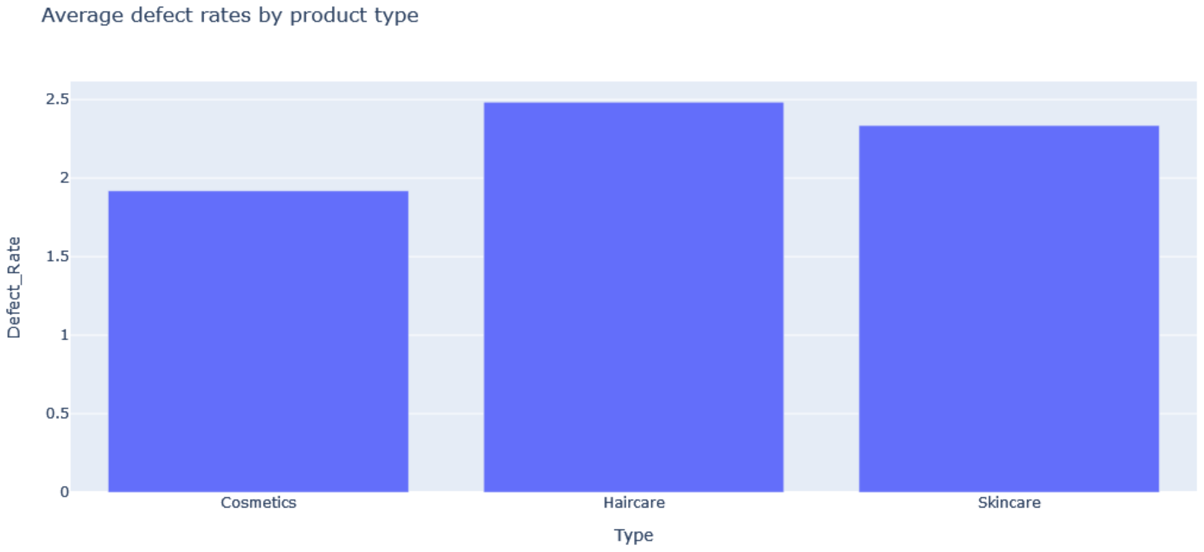
It was discovered that the most popular form of shipping was road. Furthermore, this mode of transportation also had the most defective products. Lastly, the most defective product was “Haircare”.

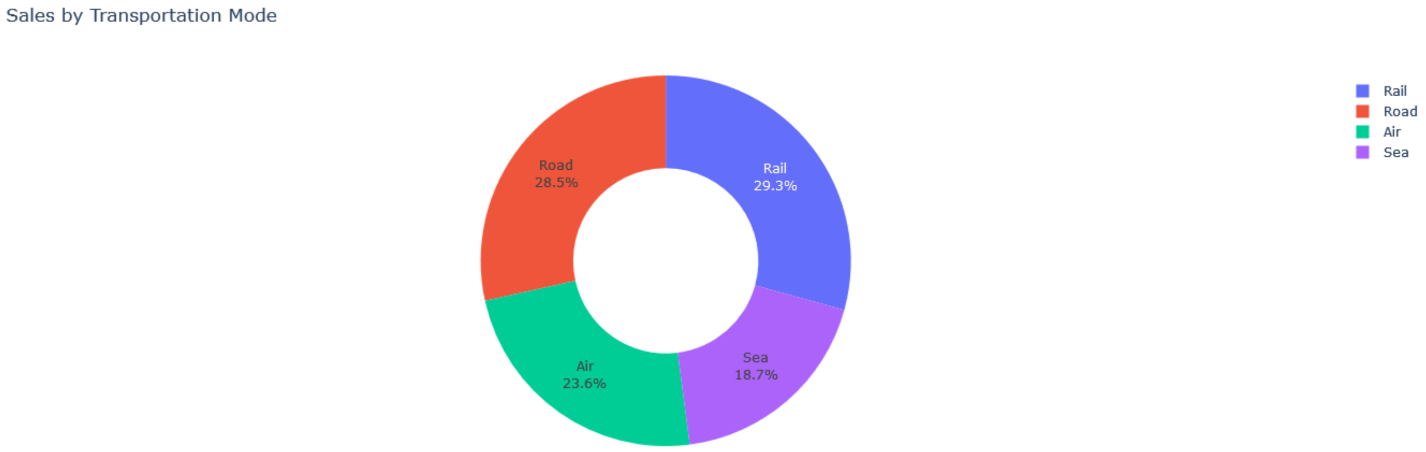
```
trans_data =
df.groupby('Transport_Mode')['Quantity_Sold'].sum().reset_index()

pie_chart = px.pie(trans_data,values='Quantity_Sold', names='Transport_Mode',
                  title='Sales by Transportation Mode',
                  hover_data=['Quantity_Sold'],
                  hole=0.5,
#                   color_discrete_sequence=px.colors.qualitative.Pastel
                  )
pie_chart.update_traces(textposition='inside',textinfo='percent+label')
pie_chart.show()
```

```
total_revenue = df.groupby('Carrier_ID')['Revenue'].sum().reset_index()

fig = go.Figure()
fig.add_trace(go.Bar(x=total_revenue['Carrier_ID'],
                     y=total_revenue['Revenue']))
fig.update_layout(title='Total revenue by shipping carrier',
                 xaxis_title='Shipping carrier',
                 yaxis_title='Revenue Generated')
fig.show()
```

```
defect_rate_by_product =
df.groupby('Type')['Defect_Rate'].mean().reset_index()

fig = px.bar(defect_rate_by_product, x='Type', y='Defect_Rate',
             title='Average defect rates by product type')
fig.show()
```

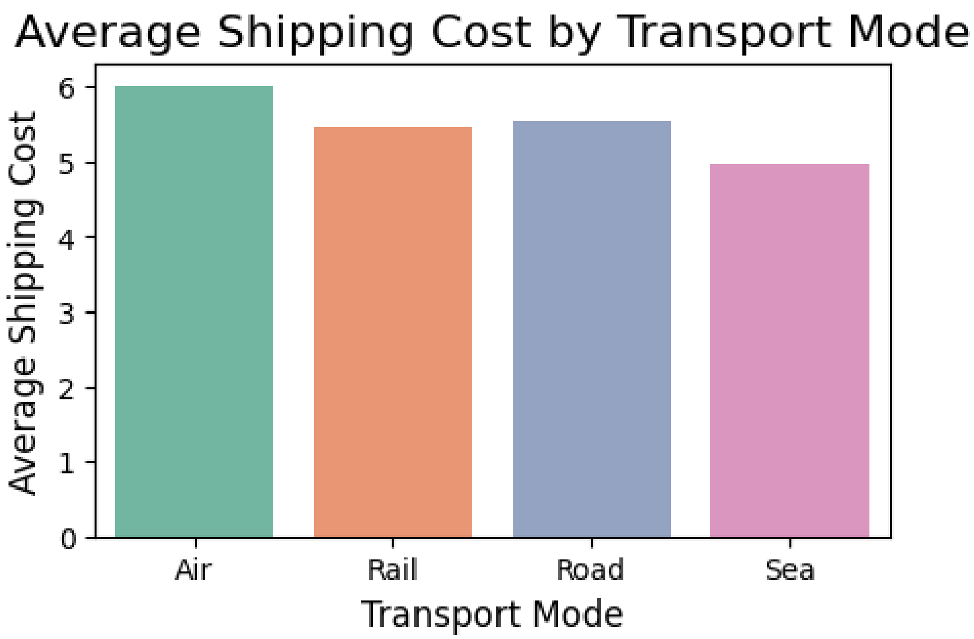






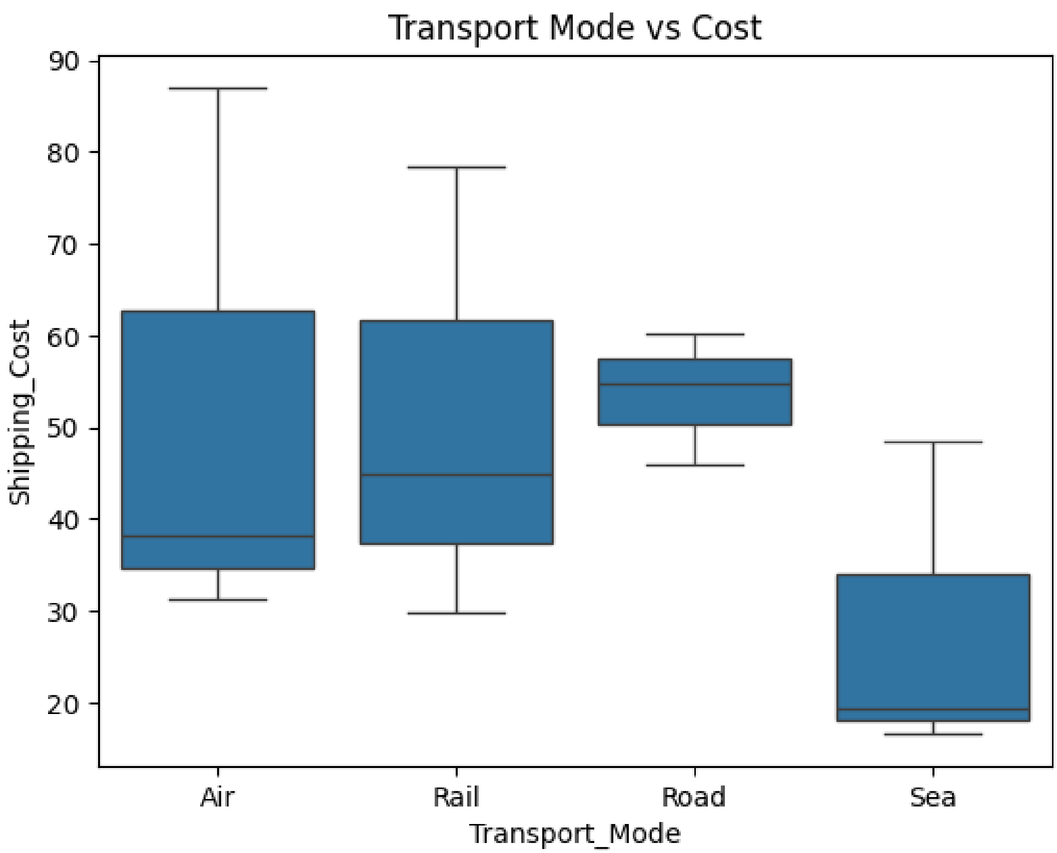
This insight is quite surprising. Road transport is very ubiquitous. Thus, it is usually the easiest method of transportation. However, it also appears to be the most detrimental form of transportation for the products. Road transport is also usually quite cheap. It is not as cheap as the cheapest form of transportation, which is transport by sea, but it is still cheap enough to be the default mode of transportation. The safest mode of transportation is air. Shockingly, the difference of average cost between both modes is very minor.

```
# Shipping Cost by Transport Mode
shipping_cost_by_mode =
df.groupby('Transport_Mode')['Shipping_Cost'].mean().reset_index()
plt.figure(figsize=(5, 3))
sns.barplot(data=shipping_cost_by_mode, x='Transport_Mode',
y='Shipping_Cost', palette='Set2')
plt.title('Average Shipping Cost by Transport Mode', fontsize=16)
plt.xlabel('Transport Mode', fontsize=12)
plt.ylabel('Average Shipping Cost', fontsize=12)
plt.show()
```



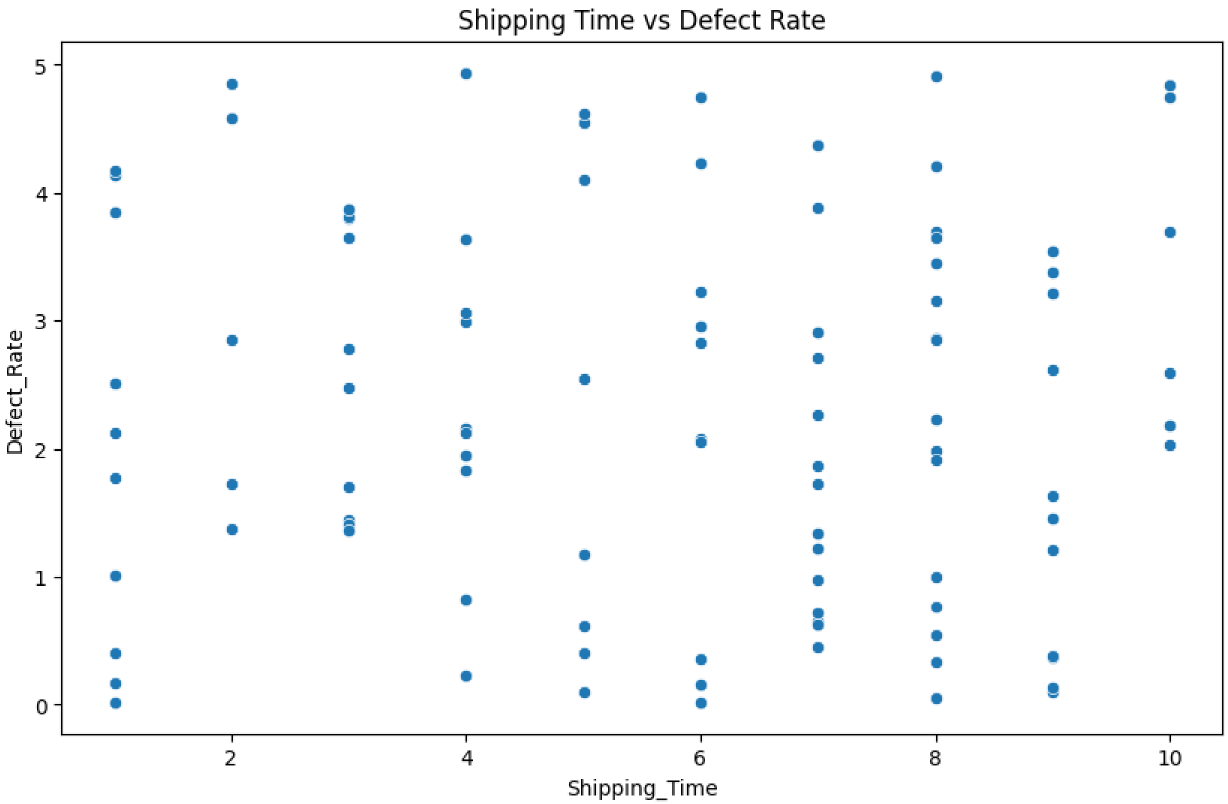
The above plot seems implies that the cost of transportation was minimal. However, it was revealed upon further analysis that there were large differences in the cost of each transport mode. Sea has the possibility to be by far the cheapest. The cost is road is quite stable. Rail and Air have the potential to be extremely expensive, however.

```
# Correlation between transport mode and revenue
carrier_revenue = df.groupby(['Carrier_ID',
'Transport_Mode'])['Shipping_Cost'].sum().reset_index()

# Plot transport mode vs revenue without limiting to the "mode"
sns.boxplot(x='Transport_Mode', y='Shipping_Cost', data=carrier_revenue)
plt.title('Transport Mode vs Cost')
plt.show()
```



The reason for defective products is unknown. We assumed that it could be caused by long shipping times. This makes sense as more time during shipping means more time for a product to become defective. However, there was no correlation between shipping time and the defect rate.

```
# Correlation between Shipping Time and Defect Rate
correlation = df[['Shipping_Time', 'Defect_Rate']].corr().iloc[0, 1]
print(f"Correlation between Shipping Time and Defect Rate: {correlation}")

# Scatter plot to visualize the correlation
plt.figure(figsize=(10,6))
sns.scatterplot(x='Shipping_Time', y='Defect_Rate', data=df)
plt.title('Shipping Time vs Defect Rate')
plt.show()
```

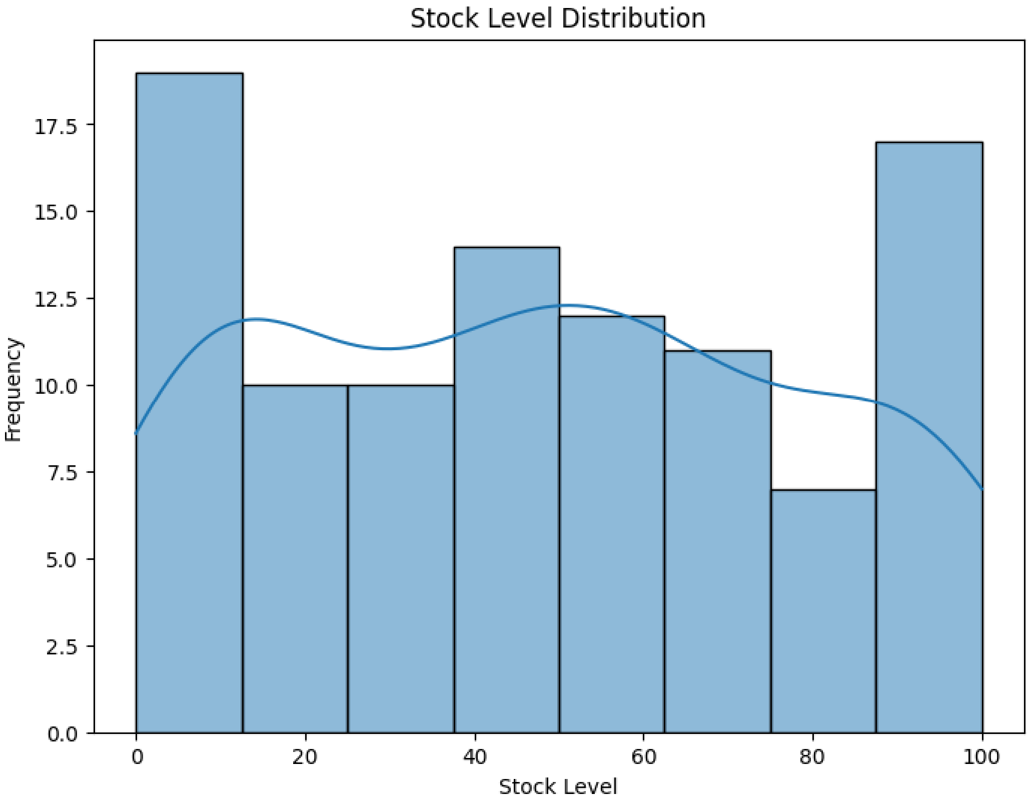


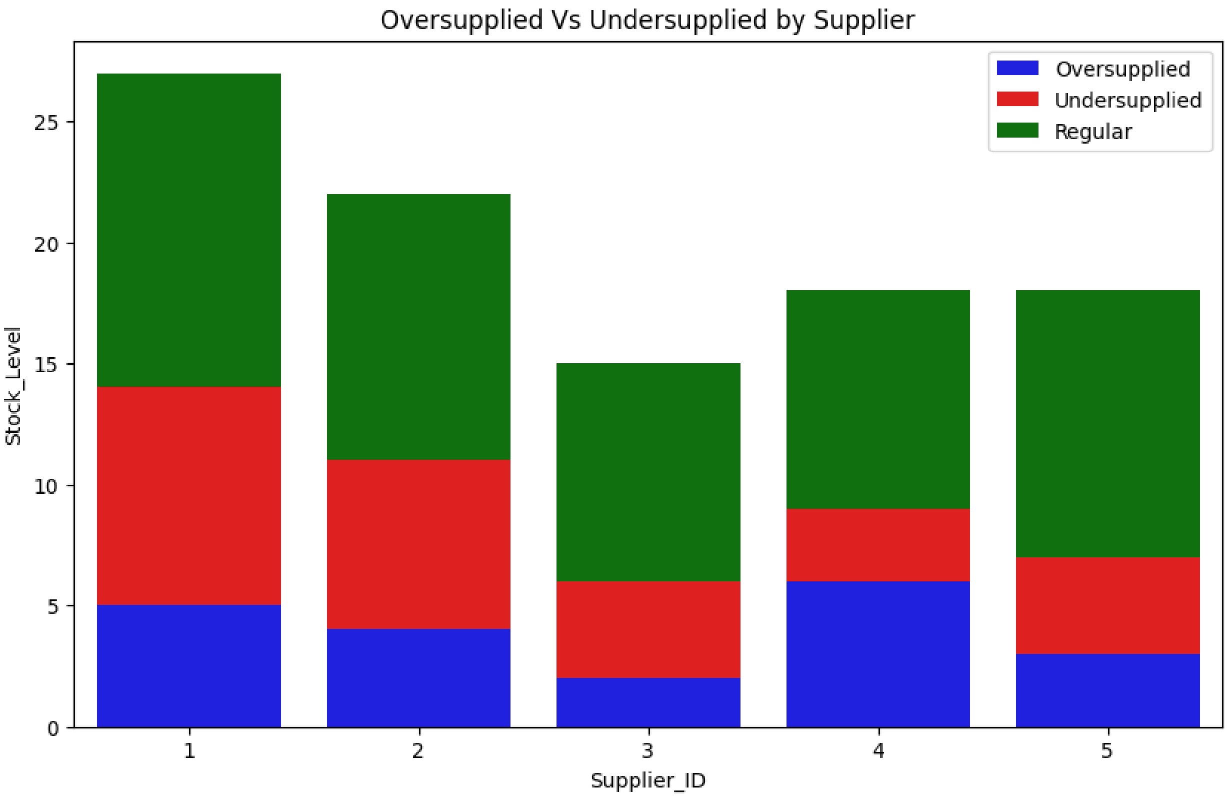
Stock Analysis:

The stock levels of the supply chain follow an irregular pattern. Products were most commonly stocked near 100, which was the max in the dataset, or close to 0. This bimodal distribution possibly indicates a chaotic and unregulated supply chain. In an ideal scenario, objects should not be close to 0 (understocking), which would lead to missed sales, or be close to the maximum which ties up capital in assets that will be sold slowly.

```
# Histogram for stock level distribution
plt.figure(figsize=(8, 6))
sns.histplot(df['Stock_Level'], kde=True)
plt.title('Stock Level Distribution')
plt.xlabel('Stock Level')
plt.ylabel('Frequency')
plt.show()
```

```
# Oversupplied and undersupplied items by supplier
oversupplied = df[df['Stock_Level'] >
80].groupby('Supplier_ID')['Stock_Level'].count().reset_index()
undersupplied = df[df['Stock_Level'] <
20].groupby('Supplier_ID')['Stock_Level'].count().reset_index()
regularsupplied = df[(df['Stock_Level'] >= 20) & (df['Stock_Level'] <=
80)].groupby('Supplier_ID')['Stock_Level'].count().reset_index()


plt.figure(figsize=(10,6))
sns.barplot(x='Supplier_ID', y='Stock_Level', data=oversupplied,
color='blue', label='Oversupplied')
sns.barplot(x='Supplier_ID', y='Stock_Level', data=undersupplied,
color='red', label='Undersupplied',
            bottom=oversupplied['Stock_Level'].values)
sns.barplot(x='Supplier_ID', y='Stock_Level', data=regularsupplied,
color='green', label='Regular',
            bottom=(oversupplied['Stock_Level'] +
undersupplied['Stock_Level']).values)
plt.title('Oversupplied Vs Undersupplied by Supplier')
plt.legend()
plt.show()
```



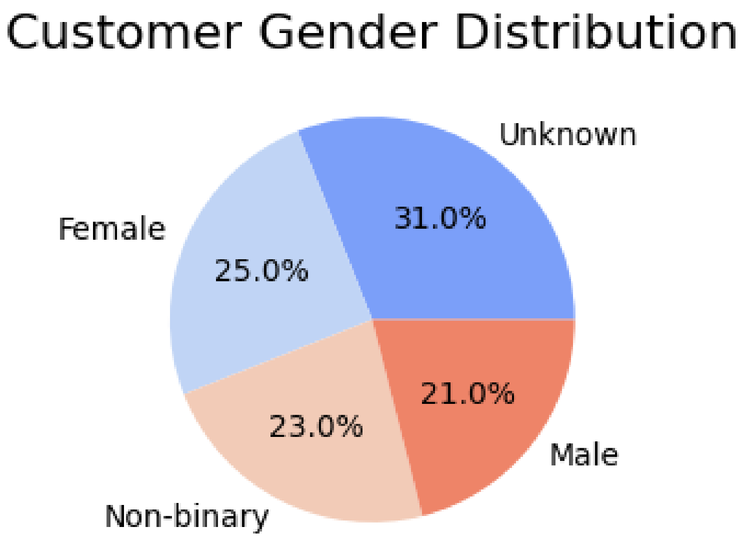


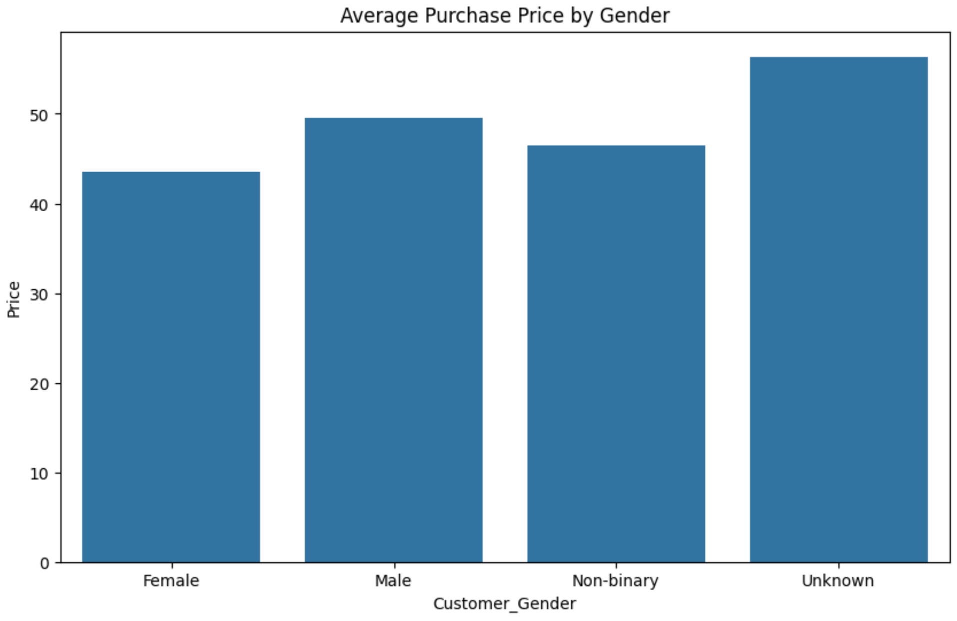
Customer Analysis:

Most customers did not disclose their gender while making a purchase. However, of those who did, their distribution was roughly equal. Furthermore, customers who did not disclose their gender where the most profitable customer segment followed by males.

```
# Customer Gender Distribution
gender_counts = df['Customer_Gender'].value_counts()
plt.figure(figsize=(3, 3))
plt.pie(gender_counts, labels=gender_counts.index, autopct='%1.1f%%',
colors=sns.color_palette('coolwarm', len(gender_counts)))
plt.title('Customer Gender Distribution', fontsize=16)
plt.show()
# Average purchase price by gender
avg_price_by_gender =
df.groupby('Customer_Gender')['Price'].mean().reset_index()
```

```
# Most common product type by gender
common_type_by_gender = df.groupby('Customer_Gender')['Type'].agg(lambda x:
x.mode()[0]).reset_index()

# Plot average price
plt.figure(figsize=(10,6))
sns.barplot(x='Customer_Gender', y='Price', data=avg_price_by_gender)
plt.title('Average Purchase Price by Gender')
plt.show()

# Display most common type by gender
print(common_type_by_gender)
```

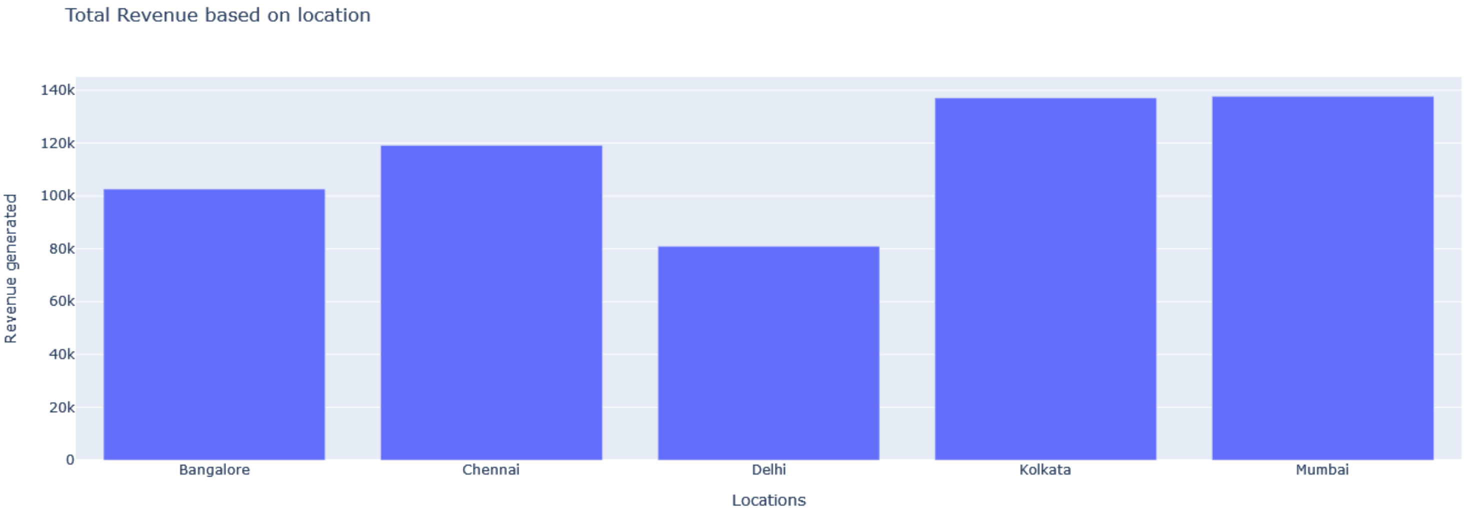




Manufacturer Analysis:

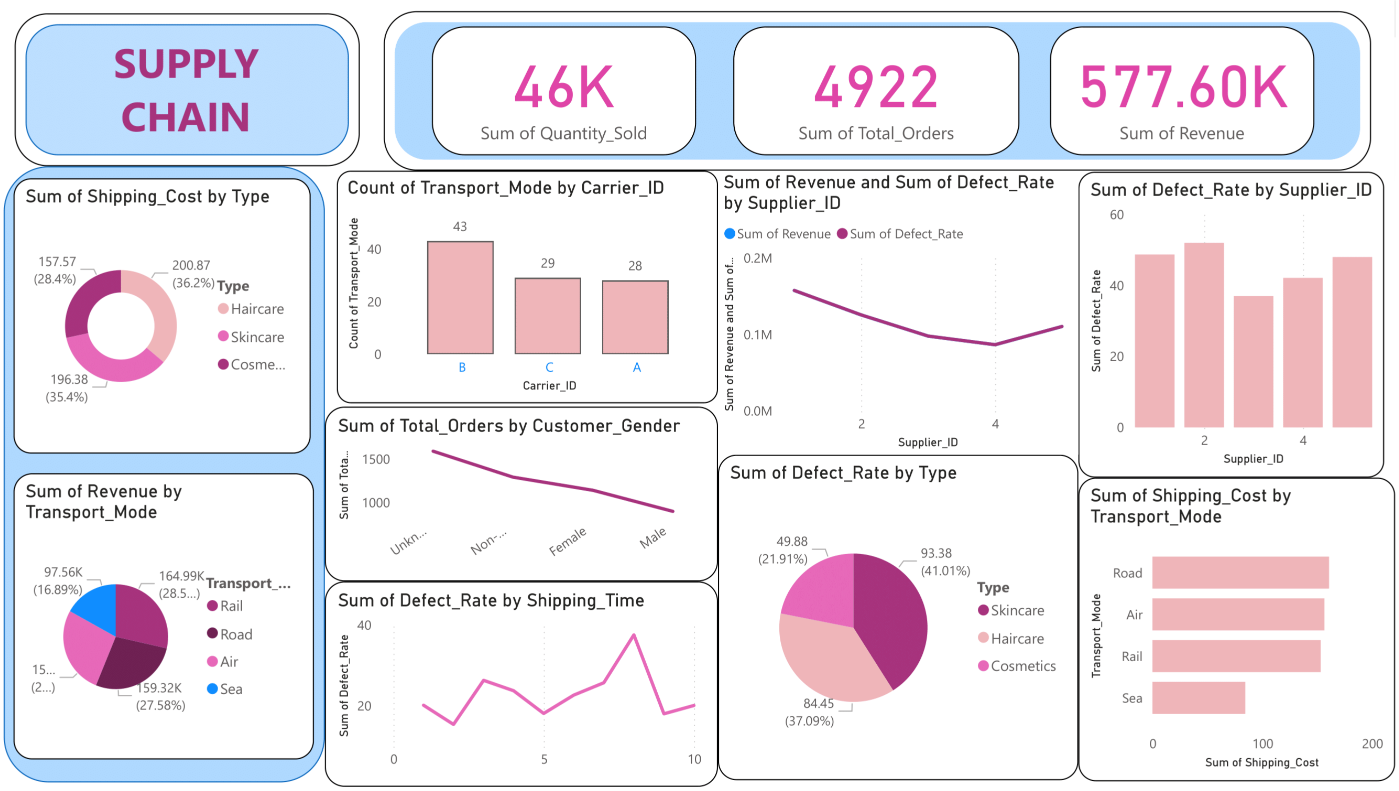
This result was quite surprising. Delhi had the least revenue, despite it having the largest real-world population. Mumbai had the highest revenue with a slight margin over Kolkata. Delhi has around 34 million while Delhi and Kolkata 15 and 21 million respectively. This implies that population does not correlate with revenue. This means, that at least in this dataset, the manufacturers got their revenue from distant customers instead of local ones.

```
total_revenue =
df.groupby('Manufacturer_Location')['Revenue'].sum().reset_index()

fig = go.Figure()
fig.add_trace(go.Bar(x=total_revenue['Manufacturer_Location'],
                    y=total_revenue['Revenue']))
fig.update_layout(title='Total Revenue based on location',
                 xaxis_title='Locations',
                 yaxis_title='Revenue generated')
fig.show()
```



Visualization Phase:

After the analysis phase was completed, a Power BI dashboard was constructed to interactively display the insights gained. A screenshot of this dashboard is below.



Conclusion:

The dataset consisted of only one hundred rows. More data is needed to draw more accurate and detailed conclusions. However, there are some insights that can be taken.

1. **Defects:** Shipping time does not correlate with defect rate. This implies that most defective products in the dataset are as a result of manufacturing rather than shipping. However, this does not deny that defective products can arise as a result of poor shipping. More data is needed is investigate this further.

2. **Shipping Costs:** The average costs of each shipping mode in the dataset were roughly equal. However, there was a large spread of data. Sea was by far the cheapest. Road was predictable. Its price had a very low spread. Rail and Air had similar costs but Air was more expensive. Both exhibited that their median price was fairly reasonable. However, their price could quickly skyrocket.
3. **Customer Demographics:** Most customers did not disclose their gender. However, it was revealed that the distribution of demographics was roughly equal and that the most profitable demographic was men. This is unusual as the products in the supply chain are *usually* marketed towards females.
4. **Product Revenue:** It was also discovered that there exists a relationship (although weak) between a products price and the total revenue. This means that in this supply chain selling fewer higher priced products is more profitable when compared to selling more lower priced products.
5. **Product Stock Levels:** There are irregular patterns inside the stock levels of the products. Some products are stocked abnormally high while others are stocked abnormally low. The stock graph level exhibited a bimodal distribution, with peaks at the high and low ends of the stock levels. Further analysis and data are required to investigate this. Lastly, this problem is not unique to a certain supplier but rather to all of them.