Name: Komal Sabale
Class: D15C
Roll No.: 45

Experiment 2: Data Visualization & Exploratory Data Analysis Using Matplotlib and Seaborn

Introduction

Exploratory Data Analysis (EDA) is the first step in data analysis, developed by *John Tukey* in the 1970s. EDA is used to summarize datasets, detect patterns, identify anomalies, and ensure data quality before applying machine learning models.

When working with datasets, it is crucial to visualize data in different ways to understand relationships between variables. EDA helps identify **missing values, trends, and correlations** that may impact future analysis.

Why Perform EDA?

1. **Understanding Data Quality** – Identifies missing values, outliers, and inconsistencies.
2. **Feature Selection** – Determines key variables for modeling.
3. **Pattern Discovery** – Helps find trends and distributions in the dataset.
4. **Detecting Anomalies** – Highlights unusual data points.
5. **Improving Model Accuracy** – Ensures that only clean and relevant data is used.

Advantages of Data Visualization

1. **Improved Understanding:**
   In business, it is often necessary to compare the performance of different components or scenarios. Traditionally, this requires analyzing large volumes of data, which can be time-consuming. Data visualization simplifies this process by providing a clear, visual comparison.
2. **Enhanced Interpretation:**
   Converting data into graphical formats makes it easier to interpret and analyze. Visualization tools, such as **Google Trends**, help identify key insights and emerging patterns by presenting complex data in a digestible form.
3. **Efficient Data Sharing:**
   Visual representation of data facilitates better communication within organizations. Instead of dealing with lengthy reports or raw datasets, visually appealing charts and graphs make information easier to understand and convey effectively.
4. **Sales Analysis:**
   Visualization techniques, including heatmaps and trend charts, help sales professionals

quickly grasp product performance. By analyzing sales patterns, businesses can identify factors driving growth, customer preferences, repeat buyers, and regional impacts.

5. **Identifying Relationships Between Events:**
   Business performance is often influenced by multiple factors. Recognizing correlations between events enables decision-makers to pinpoint business trends. For example, in **e-commerce**, sales typically increase during festive seasons like **Christmas or Thanksgiving**. If an online business records an average of $1 million in quarterly revenue and sees a surge in the next quarter, visualization helps link this increase to specific events or promotions.
6. **Exploring Opportunities and Trends:**
   With vast amounts of available data, business leaders can uncover insights into industry trends and market opportunities. **Data visualization** enables analysts to detect customer behavior patterns, allowing businesses to adapt strategies and capitalize on emerging trends.

Technologies Used

Matplotlib

Matplotlib is a Python library for creating static, animated, and interactive visualizations. It provides various graphing tools, including bar graphs, histograms, and scatter plots.

Seaborn

Seaborn is a high-level visualization library built on Matplotlib, designed for statistical graphics. It simplifies complex plots like heatmaps, box plots, and scatter plots.

General Syntax in Python for Data Visualization

Python libraries like Matplotlib and Seaborn follow a general syntax for creating visualizations:

1. **Import the library:** Import the required libraries (e.g., import matplotlib.pyplot as plt).
2. **Prepare the data:** Use Pandas to manipulate and prepare the data for visualization.
3. **Create the plot:** Use functions like plot(), scatter(), boxplot(), etc., to create the visualization.
4. **Customize the plot:** Add titles, labels, legends, and other customizations.
5. **Display the plot:** Use plt.show() to display the visualization.

<------This doc is using up on the cleaned data of the previous experiment.------->

1. Bar Graph and Contingency Table

Theory

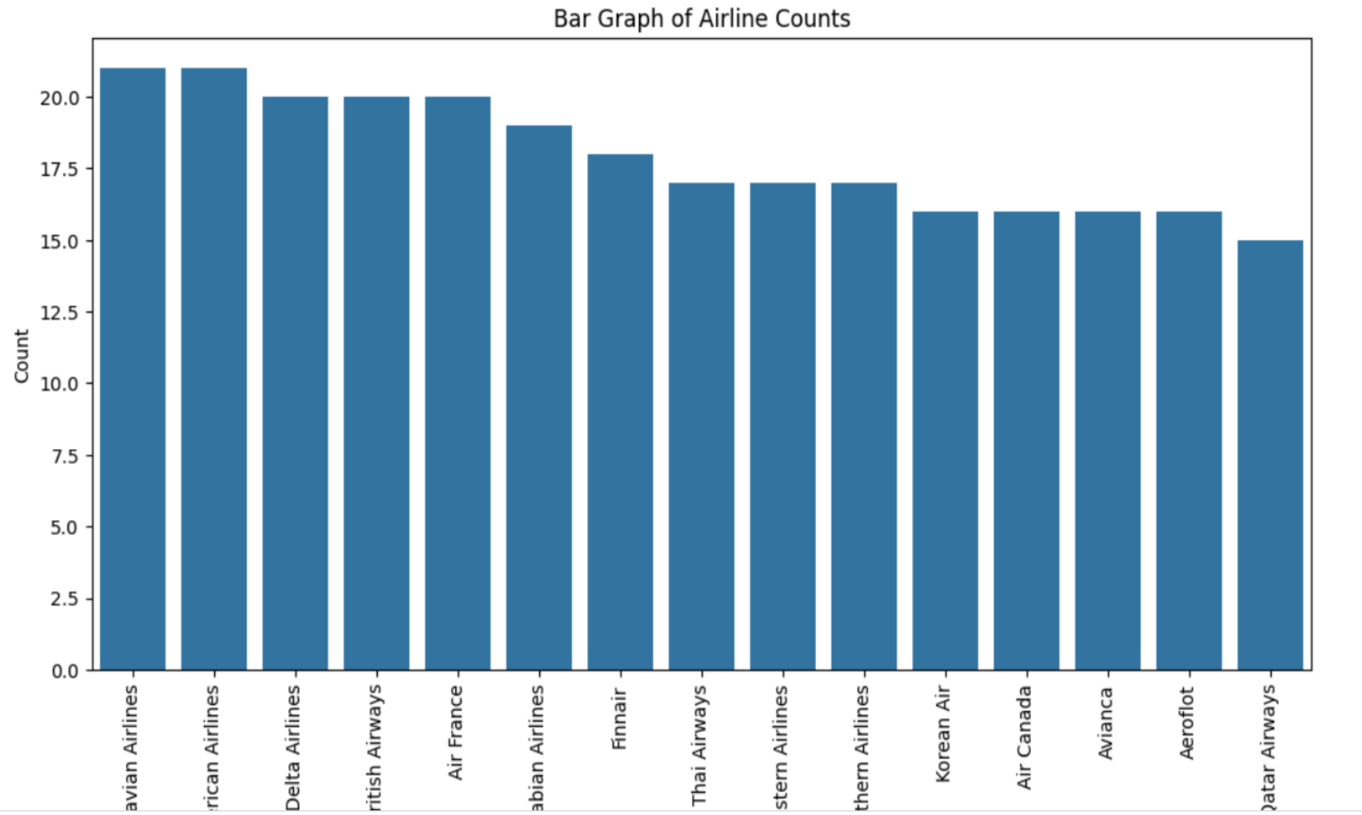
- **Bar Graph:** A bar graph is used to display categorical data with rectangular bars. The length of each bar represents the frequency of a category.
- **Contingency Table:** A table that summarizes the frequency distribution of two categorical variables, helping to analyze relationships between them.

Terms

- **Categorical Data:** Data divided into groups (e.g., Airline names).
- **Frequency:** The count of occurrences of each category.

```python
import seaborn as sns
import matplotlib.pyplot as plt
plt.figure(figsize=(12, 6))
sns.countplot(data=df, x="Airline", order=df["Airline"].value_counts().index[:15])  # Top 15 airlines for clarity
plt.xticks(rotation=90)
plt.title("Bar Graph of Airline Counts")
plt.xlabel("Airline")
plt.ylabel("Count")
plt.show()
max_total = df["Total"].max()
bins = [0, 10, 50, 100, 500]
if max_total > 500:
    bins.append(1000)
if max_total > 1000:
    bins.append(max_total + 1)

labels = ["0-10", "11-50", "51-100", "101-500"]
if max_total > 500:
    labels.append("501-1000")
if max_total > 1000:
    labels.append("1000+")

df["Total_Binned"] = pd.cut(df["Total"], bins=bins, labels=labels)

# Contingency Table: Airline vs Total Aircraft Count
contingency_table = pd.crosstab(df["Airline"], df["Total_Binned"])
contingency_table.head()
```

Output



Total_Binned Airline	0-10	11-50	51-100	101-500	501-1000
ABX Air	0	1	2	0	0
ANA Wings	0	2	0	0	0
Aegean Airlines	4	2	0	0	0
Aer Lingus	9	4	0	0	0
Aer Lingus Regional	0	1	0	0	0

Explanation

- The **bar graph** was created using sns.countplot() to visualize the number of records per **Airline**.
- The **contingency table** was created using pd.crosstab(), categorizing airlines based on **Total Aircraft Count** into binned intervals.

Observations

- Certain airlines have significantly more records, indicating larger fleet sizes.

- The contingency table helps understand which airlines operate more aircraft within specific fleet size ranges.

---

2. Scatter Plot, Box Plot, and Heatmap

Theory

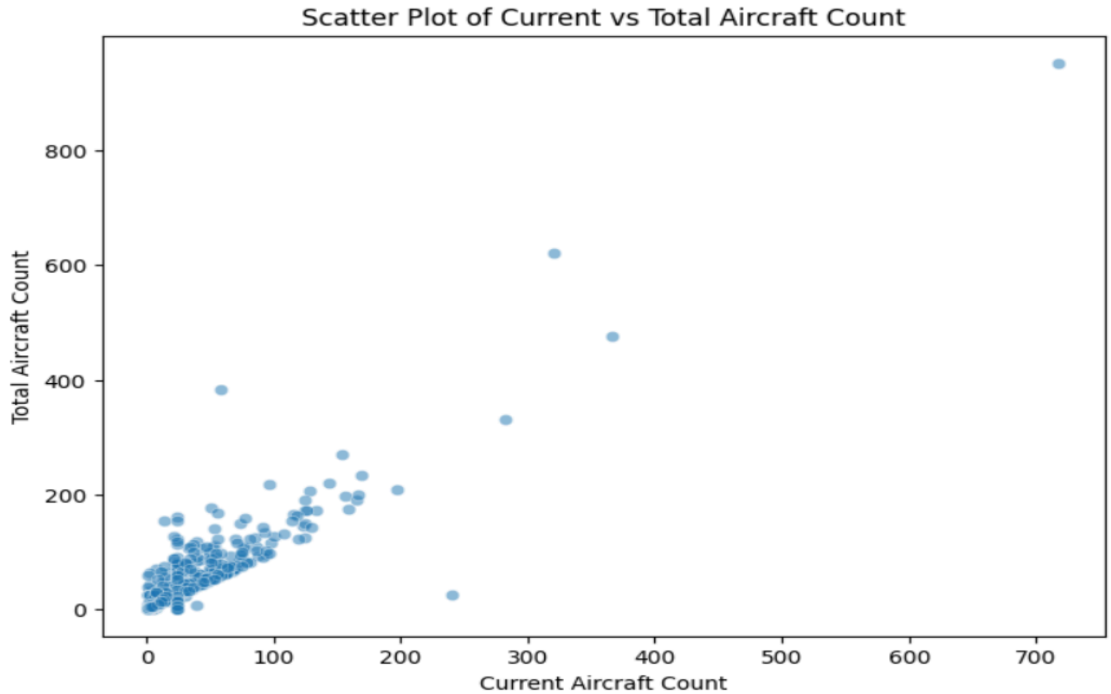
- **Scatter Plot:** A scatter plot visualizes the relationship between two numerical variables by plotting data points on an X-Y coordinate plane.
- **Box Plot:** A box plot displays the distribution of numerical data using quartiles and highlights outliers.
- **Heatmap:** A heatmap visually represents the correlation between numerical variables using colors.

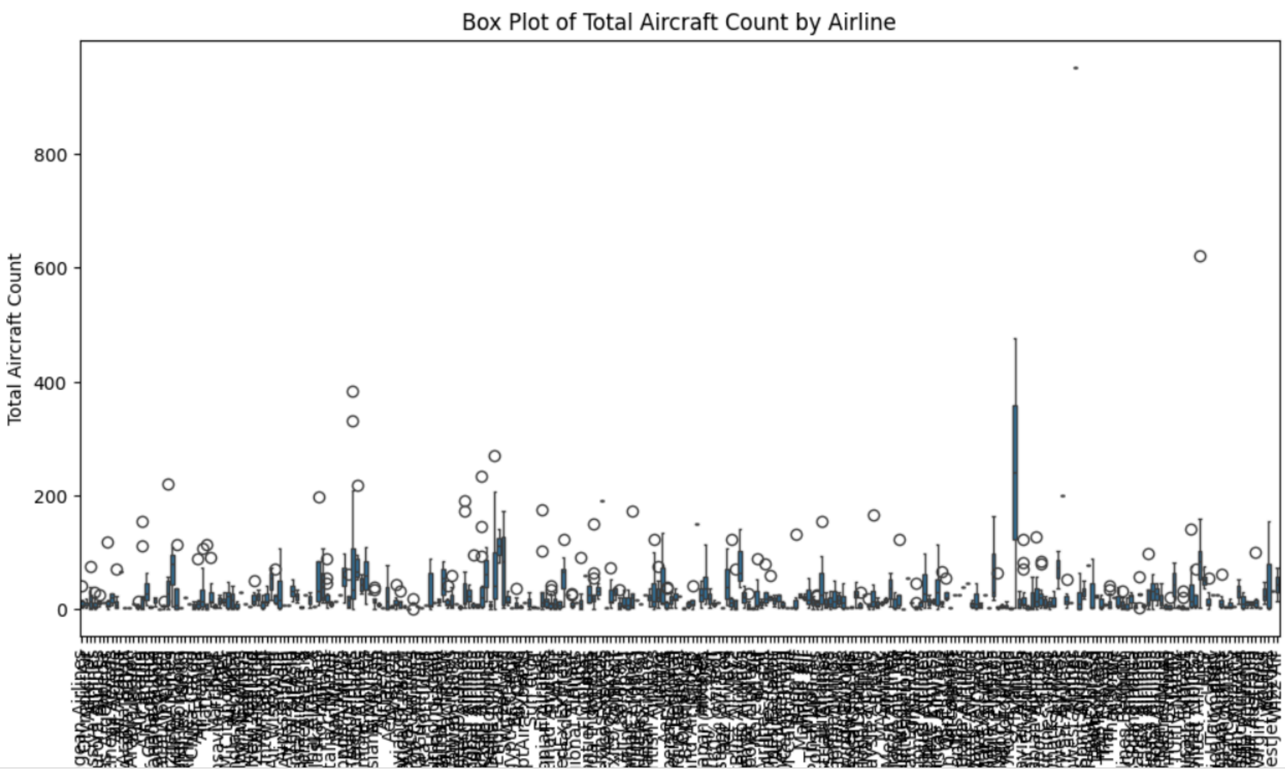
Terms

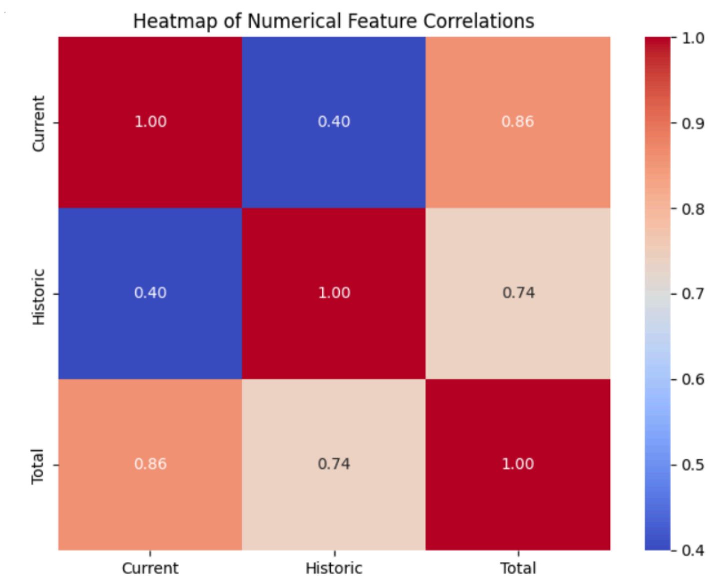
- **Numerical Data:** Data representing continuous quantities (e.g., aircraft count).
- **Quartiles:** Divides the dataset into four equal parts.
- **Correlation:** A measure of the strength of the relationship between two variables (values range from -1 to +1).

```
# Scatter plot: 'Current' vs 'Total' aircraft count
plt.figure(figsize=(8, 6))
sns.scatterplot(data=df, x="Current", y="Total", alpha=0.5)
plt.title("Scatter Plot of Current vs Total Aircraft Count")
plt.xlabel("Current Aircraft Count")
plt.ylabel("Total Aircraft Count")
plt.show()

# Box plot: Distribution of 'Total' aircraft count by 'Airline'
plt.figure(figsize=(12, 6))
sns.boxplot(data=df, x="Airline", y="Total")
plt.xticks(rotation=90)
plt.title("Box Plot of Total Aircraft Count by Airline")
plt.xlabel("Airline")
plt.ylabel("Total Aircraft Count")
plt.show()

# Heatmap of correlation between numerical features
plt.figure(figsize=(8, 6))
sns.heatmap(df[["Current", "Historic", "Total"]].corr(), annot=True, cmap="coolwarm", fmt=".2f")
plt.title("Heatmap of Numerical Feature Correlations")
plt.show()
```

Output


Scatter Plot of Current vs Total Aircraft Count
Total Aircraft Count
0
200
400
600
800
0
100
200
300
400
500
600
700
Current Aircraft Count


Box Plot of Total Aircraft Count by Airline
Total Aircraft Count
0
200
400
600
800



Explanation

- The **scatter plot** was created using sns.scatterplot() to visualize the relationship between **Current** and **Total** aircraft counts.
- The **box plot** was created using sns.boxplot() to analyze fleet size variations across different airlines.
- The **heatmap** was plotted using sns.heatmap() to display the correlation matrix between numerical features.

Observations

- A **positive correlation** between *Current* and *Total* aircraft count indicates that airlines with larger historic fleets still retain many aircraft.
- The **box plot** reveals significant variations in fleet sizes among airlines.
- The **heatmap** confirms strong correlations between fleet-related numerical variables.
- **● Total vs Quantity (High Positive Correlation)**
  ○ A high positive correlation (close to 1) suggests that the total fleet size increases as the number of aircraft in operation increases. This is expected in fleet management data.
- **● Historic vs Current Fleet (Strong Positive Correlation)**
  ○ A strong correlation between historic and current aircraft count indicates that airlines with larger historic fleets still operate many aircraft.
- **● Weak Correlations**
  ○ Some variables, such as fleet size and revenue (if applicable), may show weak or no correlation, suggesting external factors influence revenue generation beyond just fleet size.
- **● No Negative Correlations**
  ○ Since this dataset primarily deals with fleet size and operational data, most numerical features are likely positively correlated.

3. Histogram and Normalized Histogram

Theory

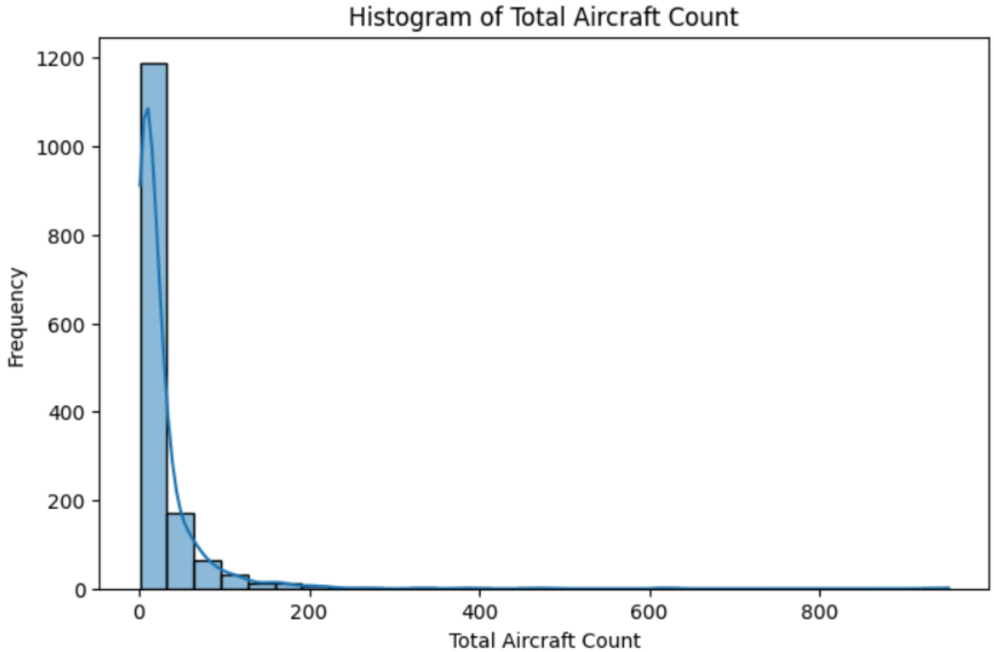
- **Histogram:** A histogram is used to represent the frequency distribution of numerical data by dividing it into bins.
- **Normalized Histogram:** Represents the probability distribution, where the total area sums to 1.

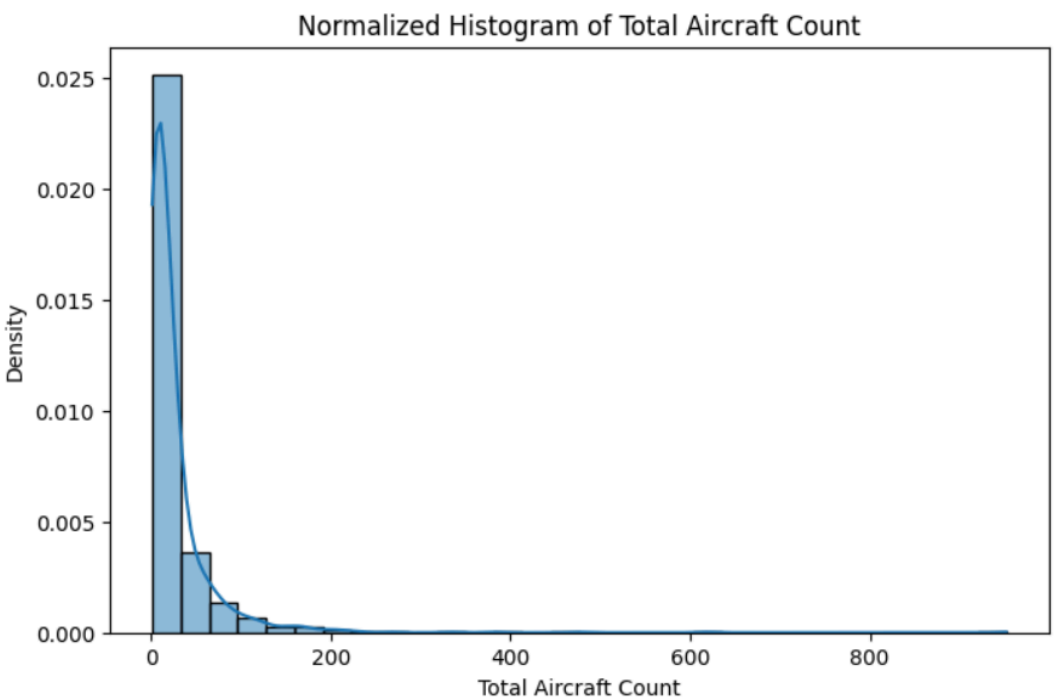
Terms

- **Bins:** Intervals that group continuous data.
- **Density:** The probability density of the data distribution.

```
[ ] # Histogram of 'Total' aircraft count
    plt.figure(figsize=(8, 5))
    sns.histplot(df["Total"], bins=30, kde=True)
    plt.title("Histogram of Total Aircraft Count")
    plt.xlabel("Total Aircraft Count")
    plt.ylabel("Frequency")
    plt.show()

    # Normalized Histogram
    plt.figure(figsize=(8, 5))
    sns.histplot(df["Total"], bins=30, kde=True, stat="density")  # Normalized version
    plt.title("Normalized Histogram of Total Aircraft Count")
    plt.xlabel("Total Aircraft Count")
    plt.ylabel("Density")
    plt.show()
```

Output





Explanation

- The **histogram** was created using sns.histplot() to visualize the frequency distribution of **Total Aircraft Count**.
- The **normalized histogram** was generated by setting stat="density" to normalize the values.

Observations

- Most airlines have **small to mid-sized fleets**, with fewer airlines operating large fleets.
- The distribution is slightly **right-skewed**, indicating a higher number of small airlines compared to larger ones.
- The histogram revealed that most airlines have **small to mid-sized fleets**, with fewer airlines operating large fleets.
- The normalized histogram confirmed that the **fleet size distribution is right-skewed**, meaning a few airlines dominate in terms of fleet numbers.
- This suggests that while many airlines operate on a smaller scale, a handful of airlines have significantly larger fleets, influencing the overall industry trends

4. Handling Outliers Using Box Plot and IQR

Theory

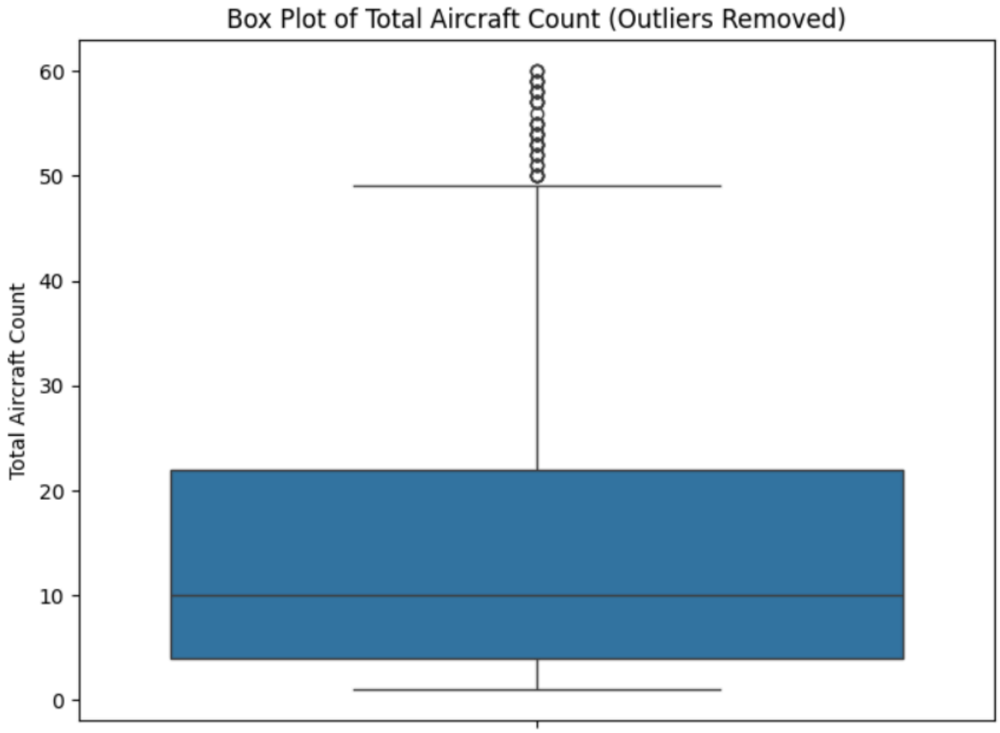
- **Outliers:** Data points that deviate significantly from the rest of the dataset.
- **Box Plot:** Identifies outliers using quartiles and whiskers.
- **IQR Method:** Detects outliers using the Interquartile Range (IQR) as follows:
  - Lower Bound = Q1 - (1.5 * IQR)
  - Upper Bound = Q3 + (1.5 * IQR)

Terms

- **Quartiles (Q1, Q3):** The 25th and 75th percentiles of the dataset.
- **IQR:** The range between Q1 and Q3.

```
[ ] # Calculate IQR for 'Total' aircraft count
    Q1 = df["Total"].quantile(0.25)
    Q3 = df["Total"].quantile(0.75)
    IQR = Q3 - Q1

    # Define lower and upper bounds
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR

    # Remove outliers
    df_no_outliers = df[(df["Total"] >= lower_bound) & (df["Total"] <= upper_bound)]

    # Box plot after outlier removal
    plt.figure(figsize=(8, 6))
    sns.boxplot(data=df_no_outliers, y="Total")
    plt.title("Box Plot of Total Aircraft Count (Outliers Removed)")
    plt.ylabel("Total Aircraft Count")
    plt.show()
```

Output



Explanation

- **Box Plot:** Used sns.boxplot() before and after outlier removal.
- **IQR Method:** Applied using df[(df["Total"] >= lower_bound) & (df["Total"] <= upper_bound)] to filter out extreme values.

Observations

- Outliers were detected in airlines with exceptionally high aircraft counts.
- Removing outliers **improves dataset reliability** by reducing distortions in analysis.
- The **box plot** helped identify airlines with extreme fleet sizes, either **exceptionally large or unusually small**.
- Using the **IQR method**, these outliers were removed, resulting in a dataset that better represents the majority of airlines.
- By eliminating extreme values, the analysis becomes more accurate and avoids distortions caused by a few exceptionally large airlines

Conclusion

This experiment conducted a detailed **Exploratory Data Analysis (EDA)** on airline fleet data. We used various visualization techniques to uncover trends, distributions, and anomalies within the dataset.

Key findings:

- **Bar Graph & Contingency Table** helped analyze airline fleet sizes.
- **Scatter Plot & Heatmap** confirmed strong correlations between aircraft count variables.
- **Box Plot & IQR Method** helped detect and remove outliers.
- **Histogram** provided insights into fleet size distributions.

By leveraging these statistical methods and visualizations, we gained meaningful insights that can assist in airline fleet management and operational strategies. This experiment highlights the **importance of EDA in data-driven decision-making** and provides a strong foundation for further predictive modeling.