Problem Statement

Flyzy is a company focused on providing a smooth and hassle-free air travel experience. They offer personalized in-flight and airport recommendations, and they also provide real-time flight tracking, mobile check-in, and more. Flyzy aims to redefine the future of air travel with a more personalized and connected experience from the beginning of the trip to the end.

Flight cancellation is a significant issue in the aviation industry. It not only disrupts the customers' plans but also impacts the airlines' reputation and profitability. Therefore, predicting flight cancellations can help airlines take preventive measures and minimize disruptions.

Task -Data Checking

Python

Before developing the predictive model for hotel cancellations, we will conduct preliminary data analysis. This involves checking for missing values, identifying outliers, and ensuring appropriate data types for each column. Handling missing values and outliers strategically will ensure a reliable dataset for accurate modeling.

● First, load the dataset and check for:

● Missing values: Use the appropriate function to check if there are any missing values in the dataset. If there are, decide on the best strategy to handle them based on the nature of the data.

● Outliers: Check for outliers in the dataset. These can be identified using various techniques, such as boxplots, scatterplots, or Z-scores. If there are any outliers, decide on the best strategy to handle them.

● Data types: Check the data type of each column. Ensure that the data type is appropriate for the data it represents.

Importing My Libraries

```python
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from scipy import stats
```

- # First, load the dataset

```
# Load the dataset
Data = pd.read_csv('Flyzy Flight Cancellation - Sheet1 (1).csv')

# Display the first few rows of the dataset
Data.head()

   Flight ID    Airline  Flight_Distance Origin_Airport
Destination_Airport  \
0    7319483  Airline D              475      Airport 3
Airport 2
1    4791965  Airline E              538      Airport 5
Airport 4
2    2991718  Airline C              565      Airport 1
Airport 2
3    4220106  Airline E              658      Airport 5
Airport 3
4    2263008  Airline E              566      Airport 2
Airport 2

   Scheduled_Departure_Time  Day_of_Week  Month Airplane_Type
Weather_Score  \
0                         4            6      1        Type C
0.225122
1                        12            1      6        Type B
0.060346
2                        17            3      9        Type C
0.093920
3                         1            1      8        Type B
0.656750
4                        19            7     12        Type E
0.505211

   Previous_Flight_Delay_Minutes  Airline_Rating  Passenger_Load  \
0                            5.0        2.151974        0.477202
1                           68.0        1.600779        0.159718
2                           18.0        4.406848        0.256803
3                           13.0        0.998757        0.504077
4                            4.0        3.806206        0.019638

   Flight_Cancelled
0                 0
1                 1
2                 0
3                 1
4                 0
```

This dataset contains information about flights, including flight ID, airline, distance, origin and destination airports, scheduled departure time, day of the week, month, aircraft type, weather score, previous flight delay in minutes, airline rating, passenger load, and flight cancellation status.

```
# print the shape of the dataset
Data.shape
```

```
(3000, 14)
```

```
# print info about the dataset
print(Data.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3000 entries, 0 to 2999
Data columns (total 14 columns):
 #   Column                         Non-Null Count  Dtype
---  ------                         --------------  -----
 0   Flight ID                      3000 non-null   int64
 1   Airline                        3000 non-null   object
 2   Flight_Distance                3000 non-null   int64
 3   Origin_Airport                 3000 non-null   object
 4   Destination_Airport            3000 non-null   object
 5   Scheduled_Departure_Time       3000 non-null   int64
 6   Day_of_Week                    3000 non-null   int64
 7   Month                          3000 non-null   int64
 8   Airplane_Type                  3000 non-null   object
 9   Weather_Score                  3000 non-null   float64
 10  Previous_Flight_Delay_Minutes  3000 non-null   float64
 11  Airline_Rating                 3000 non-null   float64
 12  Passenger_Load                 3000 non-null   float64
 13  Flight_Cancelled               3000 non-null   int64
dtypes: float64(4), int64(6), object(4)
memory usage: 328.2+ KB
None
```

```
# print statistics
Data.describe()
```

```
          Flight ID  Flight_Distance  Scheduled_Departure_Time
Day_of_Week  \
count  3.000000e+03      3000.000000               3000.000000
3000.000000
mean   4.997429e+06       498.909333                 11.435000
3.963000
std    2.868139e+06        98.892266                  6.899298
2.016346
min    3.681000e+03       138.000000                  0.000000
1.000000
25%    2.520313e+06       431.000000                  6.000000
2.000000
50%    5.073096e+06       497.000000                 12.000000
4.000000
```

```
75%    7.462026e+06      566.000000                17.000000
6.000000
max    9.999011e+06      864.000000                23.000000
7.000000

             Month  Weather_Score  Previous_Flight_Delay_Minutes  \
count  3000.000000    3000.000000                    3000.000000
mean      6.381000       0.524023                      26.793383
std       3.473979       0.290694                      27.874733
min       1.000000       0.000965                       0.000000
25%       3.000000       0.278011                       7.000000
50%       6.000000       0.522180                      18.000000
75%       9.000000       0.776323                      38.000000
max      12.000000       1.099246                     259.000000

       Airline_Rating  Passenger_Load  Flight_Cancelled
count     3000.000000     3000.000000       3000.000000
mean         2.317439        0.515885          0.690667
std          1.430386        0.295634          0.462296
min          0.000103        0.001039          0.000000
25%          1.092902        0.265793          0.000000
50%          2.126614        0.517175          1.000000
75%          3.525746        0.770370          1.000000
max          5.189038        1.123559          1.000000
```

Check for • Missing values:

```python
# Check for missing values
Data_missing_values = Data.isnull().sum()

# Display the columns with missing values and their count
print(Data_missing_values[Data_missing_values > 0])
```

```
Series([], dtype: int64)
```

```python
missing_values = Data.isnull().sum()
print("Missing values in the dataset:")
print(missing_values)
```

```
Missing values in the dataset:
Flight ID                        0
Airline                          0
Flight_Distance                  0
Origin_Airport                   0
Destination_Airport              0
Scheduled_Departure_Time         0
Day_of_Week                      0
Month                            0
```

```
Airplane_Type                      0
Weather_Score                      0
Previous_Flight_Delay_Minutes      0
Airline_Rating                     0
Passenger_Load                     0
Flight_Cancelled                   0
dtype: int64
```

There are no missing values in the dataset. No further action is needed to handle missing values.

```
#Handling duplicate values
Data.duplicated(keep=False)
```

```
0       False
1       False
2       False
3       False
4       False
        ...
2995    False
2996    False
2997    False
2998    False
2999    False
Length: 3000, dtype: bool
```

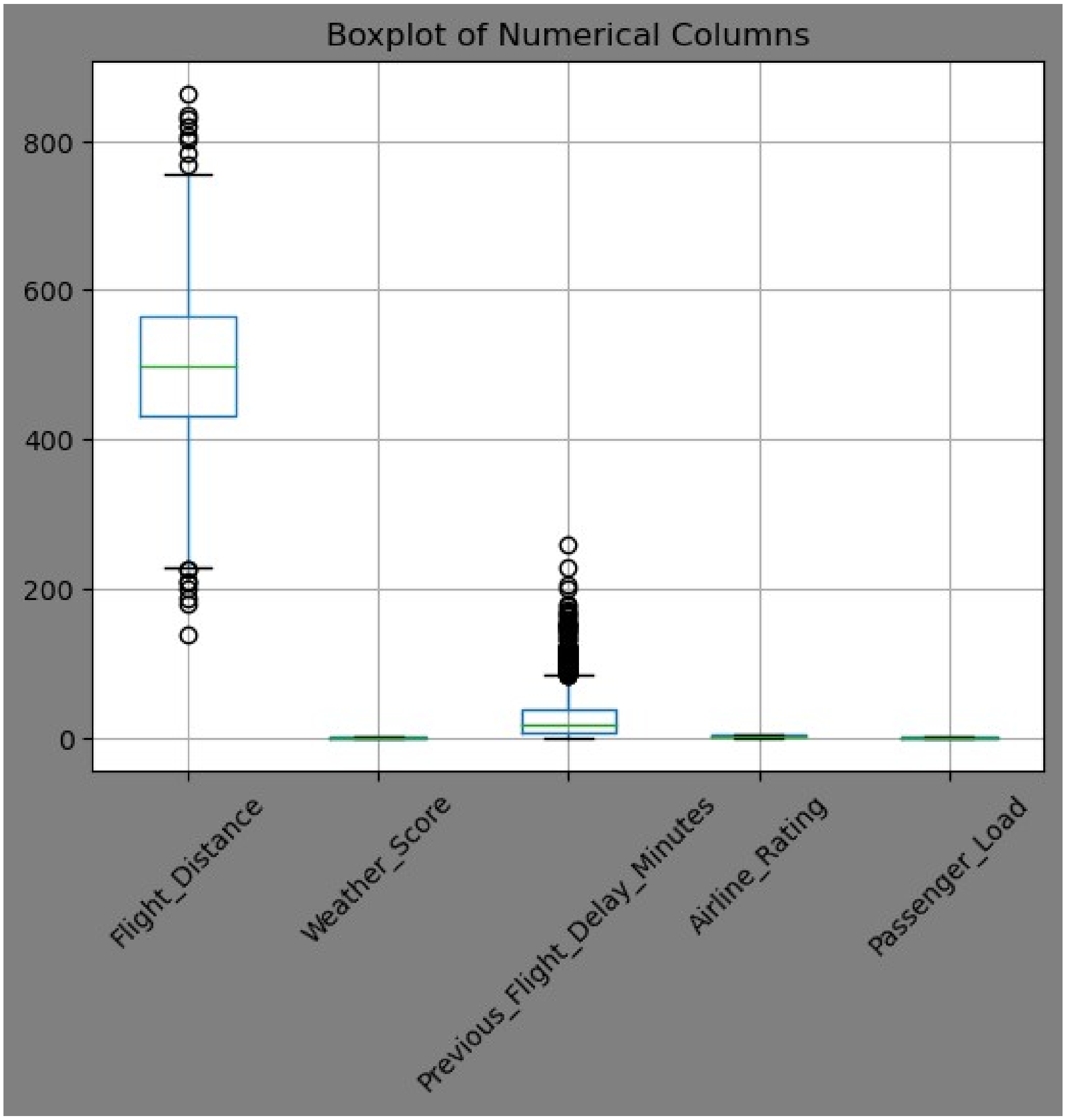
```
Data.duplicated(keep=False).sum()
```

```
0
```

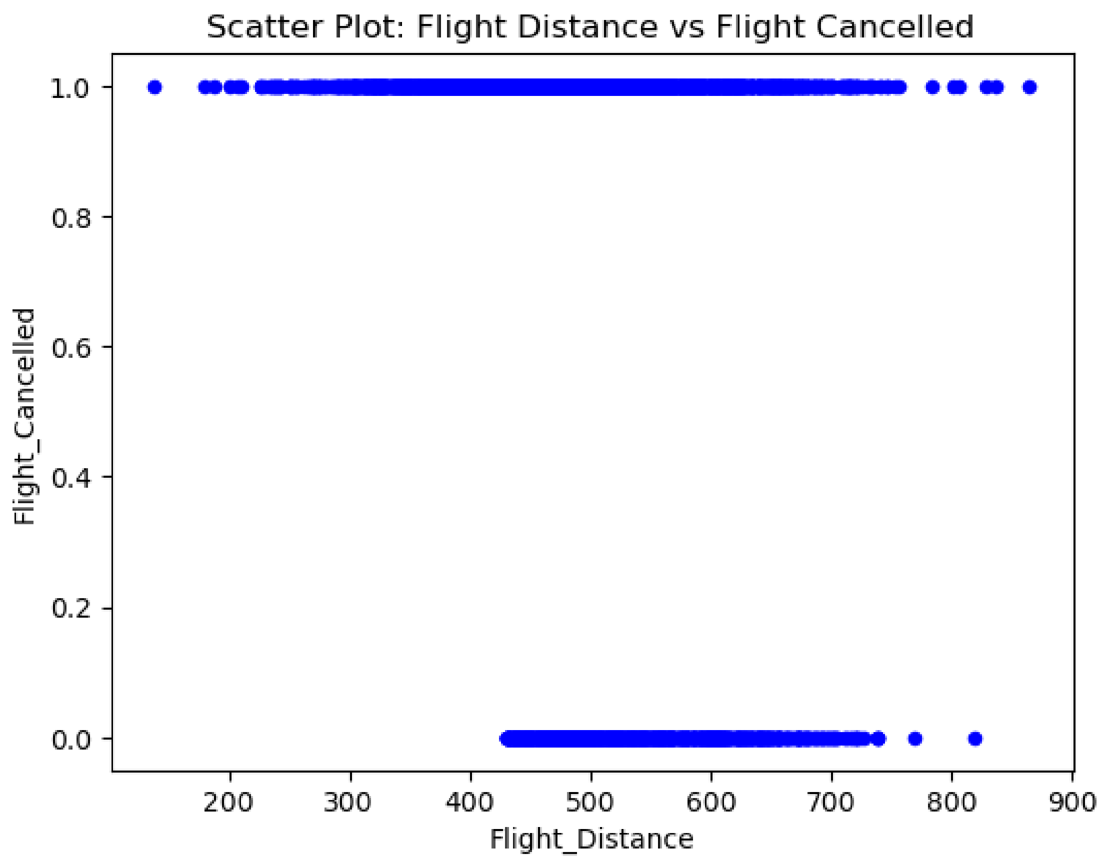
- # Outliers and Z-Score method

```
# Set the background color to white for visibility
plt.figure(facecolor='Grey')

# Select numerical columns for outlier detection
numerical_cols = ['Flight_Distance', 'Weather_Score',
'Previous_Flight_Delay_Minutes', 'Airline_Rating', 'Passenger_Load']

# Plot boxplots for numerical columns
Data[numerical_cols].boxplot()
plt.xticks(rotation=45)
plt.title('Boxplot of Numerical Columns')
plt.show()

# Calculate Z-scores for numerical columns
Data_z_scores = np.abs(stats.zscore(Data[numerical_cols]))
```

```
# Define a threshold for identifying outliers
threshold = 3

# Identify outliers
outliers = np.where(Data_z_scores > threshold)

# Display the number of outliers in each numerical column
outlier_counts = pd.Series(outliers[1]).value_counts().sort_index()
outlier_counts.index = [numerical_cols[i] for i in
outlier_counts.index]
print(outlier_counts)

# Plot scatter plot for an example numerical column against
Flight_Cancelled
Data.plot.scatter(x='Flight_Distance', y='Flight_Cancelled',
color='blue', title='Scatter Plot: Flight Distance vs Flight
Cancelled')
plt.show()
```




```
Flight_Distance                   10
Previous_Flight_Delay_Minutes     51
dtype: int64
```



The boxplot and the calculation of Z-scores have been used to identify outliers in the dataset.

The boxplot visualizes the distribution of numerical columns, indicating potential outliers beyond the whiskers. Based on Z-scores, with a threshold of 3 for identifying outliers, the following counts of outliers were found in the dataset: Flight Distance: 10 outliers Previous Flight Delay Minutes: 51 outliers

The scatter plot depicting Flight Distance against Flight Cancellation does not overtly reveal outliers, but rather illustrates the distribution of flight distances for both cancelled and non-cancelled flights.

Considering the inherent nature of the data, outliers in 'Previous Flight Delay Minutes' may be authentic due to the variability in flight delays. Regarding 'Flight Distance', outliers may signify long-haul flights. It is crucial to carefully consider the context before determining how to address these outliers. One potential approach could involve retaining these outliers if they represent valid scenarios, or imposing a cap at a specific threshold if they are found to significantly distort the analysis.

```
#find the limits
upper_limit=Data['Previous_Flight_Delay_Minutes'].mean() + 4 *
Data['Previous_Flight_Delay_Minutes'].std()
lower_limit=Data['Previous_Flight_Delay_Minutes'].mean() - 4 *
Data['Previous_Flight_Delay_Minutes'].std()
```

```
print("upper limit:", upper_limit)
print("lower limit:", lower_limit)

upper limit: 138.292314250808
lower limit: -84.70554874814133

#trimming - deleting the outlier data
new_Data=Data.loc[(Data['Previous_Flight_Delay_Minutes'] <
upper_limit) & (Data['Previous_Flight_Delay_Minutes'] > lower_limit)]
print('Before removing outliers:',len(Data))
print('after removing outliers:',len(new_Data))
print('Outliers:', len(Data)-len(new_Data))

Before removing outliers: 3000
after removing outliers: 2974
Outliers: 26
```

• Data types:

```
# Check the data types of each column
data_types = Data.dtypes
print(data_types)

Flight ID                            int64
Airline                             object
Flight_Distance                      int64
Origin_Airport                      object
Destination_Airport                 object
Scheduled_Departure_Time             int64
Day_of_Week                          int64
Month                                int64
Airplane_Type                       object
Weather_Score                      float64
Previous_Flight_Delay_Minutes      float64
Airline_Rating                     float64
Passenger_Load                     float64
Flight_Cancelled                     int64
dtype: object
```

The dataset has undergone a thorough check of data types for each column. While most data types are suitable for their respective data, there are a few considerations to note:

Scheduled_Departure_Time is represented as int64, indicating a format that may not be immediately interpretable as a time (e.g., an integer timestamp). Depending on the analysis, it may be beneficial to convert this to a datetime format for enhanced manipulation and interpretation.

Flight_Cancelled is an int64, which is appropriate for binary indication (0 for not cancelled, 1 for cancelled). However, for improved clarity and consistency in analysis, consideration may be given to converting this to a boolean type.

Day_of_Week and Month are also represented as int64, suitable for numerical analysis. For improved readability, mapping these to their respective names (e.g., Monday, January) at some point in the analysis may be beneficial.

Columns such as Airline, Origin_Airport, Destination_Airport, and Airplane_Type are of type object, which is typical for textual data. Numerical columns like Flight_Distance, Weather_Score, Previous_Flight_Delay_Minutes, Airline_Rating, and Passenger_Load have numerical types (int64 or float64), suitable for quantitative analysis.

Task -Exploratory Data Analysis (EDA)

In preparation for building the predictive model, Exploratory Data Analysis (EDA) will be conducted on the dataset. This will involve obtaining descriptive statistics, visualizing data distributions, exploring feature relationships through scatter plots or correlation matrices, and investigating how features relate to the target variable to extract valuable insights for accurate modeling.

Perform an EDA on the dataset to understand the data better and extract insights. This may involve:

● Descriptive Statistics: Use the appropriate function to get the descriptive statistics of the dataset.

● Distribution of data: Plot histograms or bar charts to see the distribution of data in each column.

● Relationship between features: Plot scatter plots, pair plots, or correlation matrices to see the relationship between different features.

● Relationship between features and target variable: Investigate how different features relate to the target variable.

● Descriptive Statistics

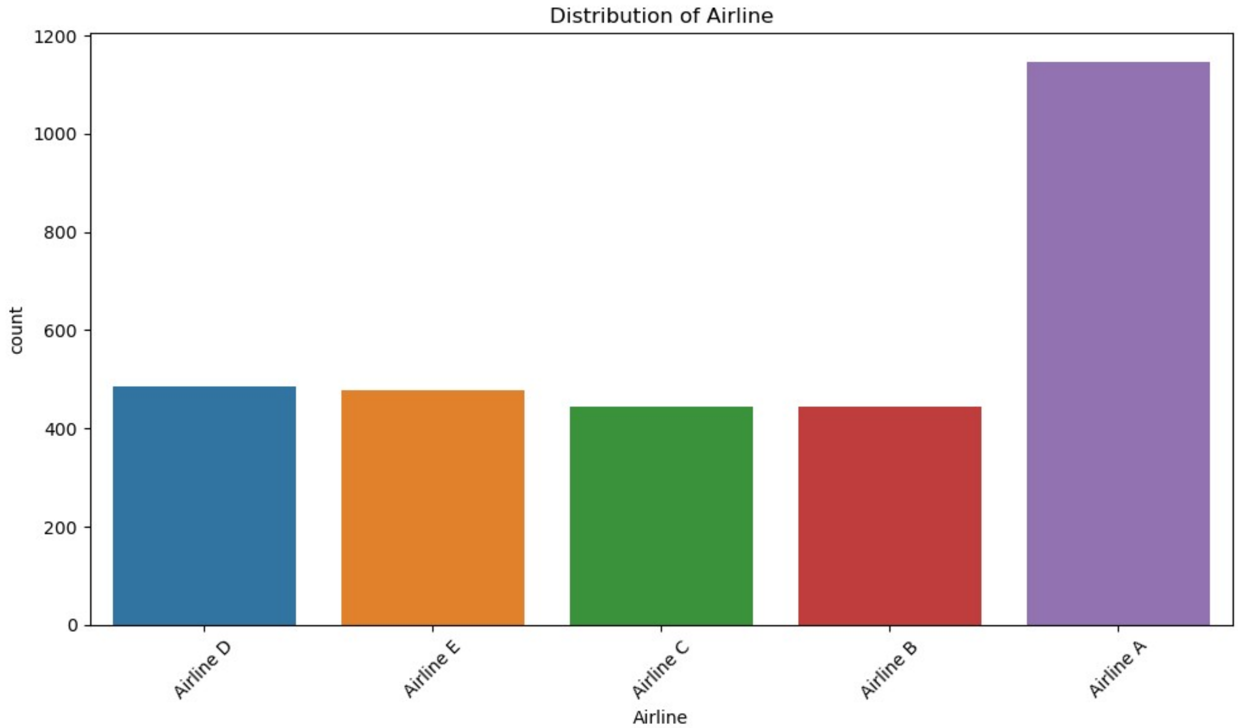
```
descriptive_stats = Data.describe()
print(descriptive_stats)

          Flight_ID  Flight_Distance  Scheduled_Departure_Time
Day_of_Week  \
count  3.000000e+03      3000.000000                3000.000000
3000.000000
mean   4.997429e+06       498.909333                  11.435000
3.963000
std    2.868139e+06        98.892266                   6.899298
```

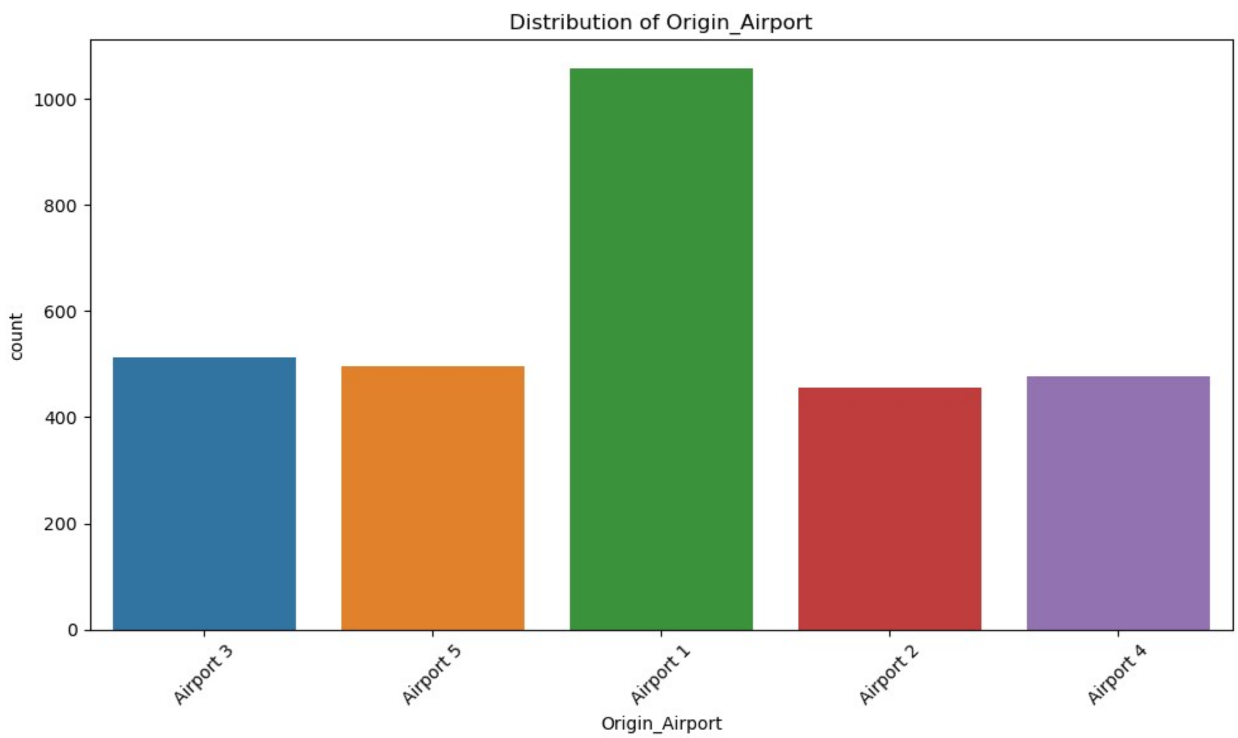
```
2.016346
min    3.681000e+03       138.000000                       0.000000
1.000000
25%    2.520313e+06       431.000000                       6.000000
2.000000
50%    5.073096e+06       497.000000                      12.000000
4.000000
75%    7.462026e+06       566.000000                      17.000000
6.000000
max    9.999011e+06       864.000000                      23.000000
7.000000

             Month  Weather_Score  Previous_Flight_Delay_Minutes  \
count  3000.000000    3000.000000                    3000.000000
mean      6.381000       0.524023                      26.793383
std       3.473979       0.290694                      27.874733
min       1.000000       0.000965                       0.000000
25%       3.000000       0.278011                       7.000000
50%       6.000000       0.522180                      18.000000
75%       9.000000       0.776323                      38.000000
max      12.000000       1.099246                     259.000000

       Airline_Rating  Passenger_Load  Flight_Cancelled
count     3000.000000     3000.000000       3000.000000
mean         2.317439        0.515885          0.690667
std          1.430386        0.295634          0.462296
min          0.000103        0.001039          0.000000
25%          1.092902        0.265793          0.000000
50%          2.126614        0.517175          1.000000
75%          3.525746        0.770370          1.000000
max          5.189038        1.123559          1.000000
```

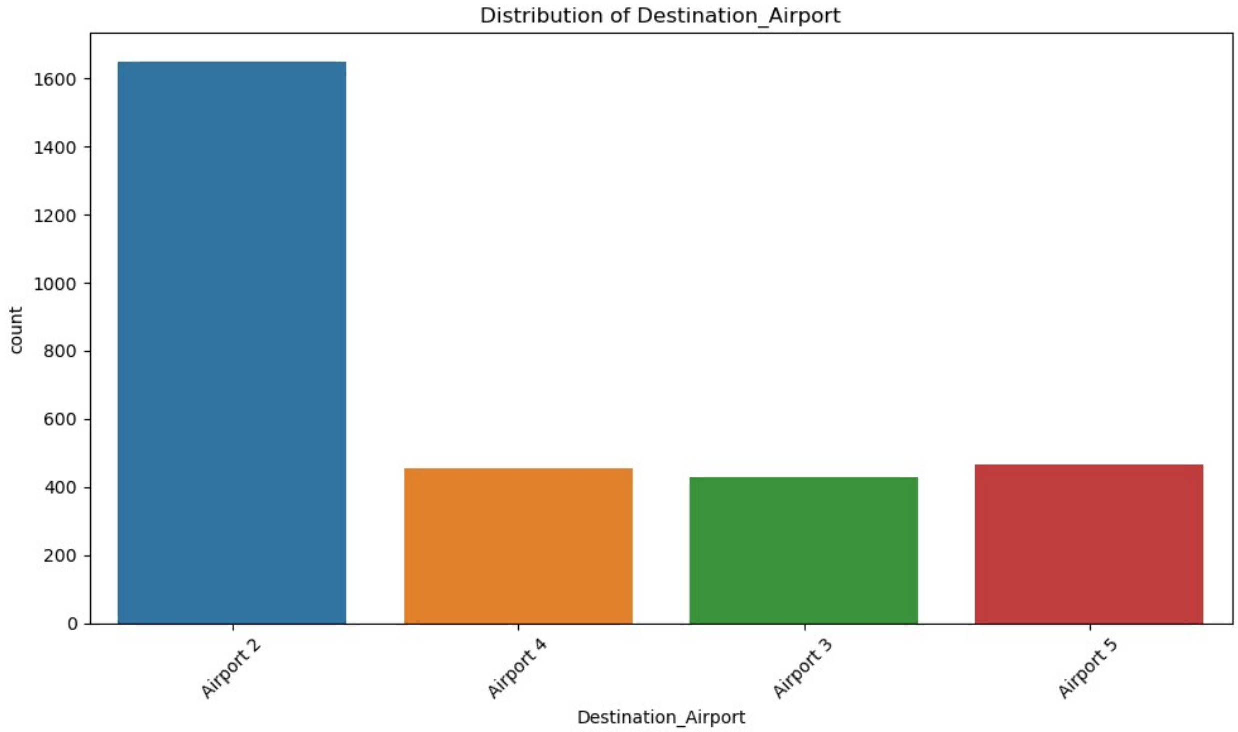
- # Distribution of data

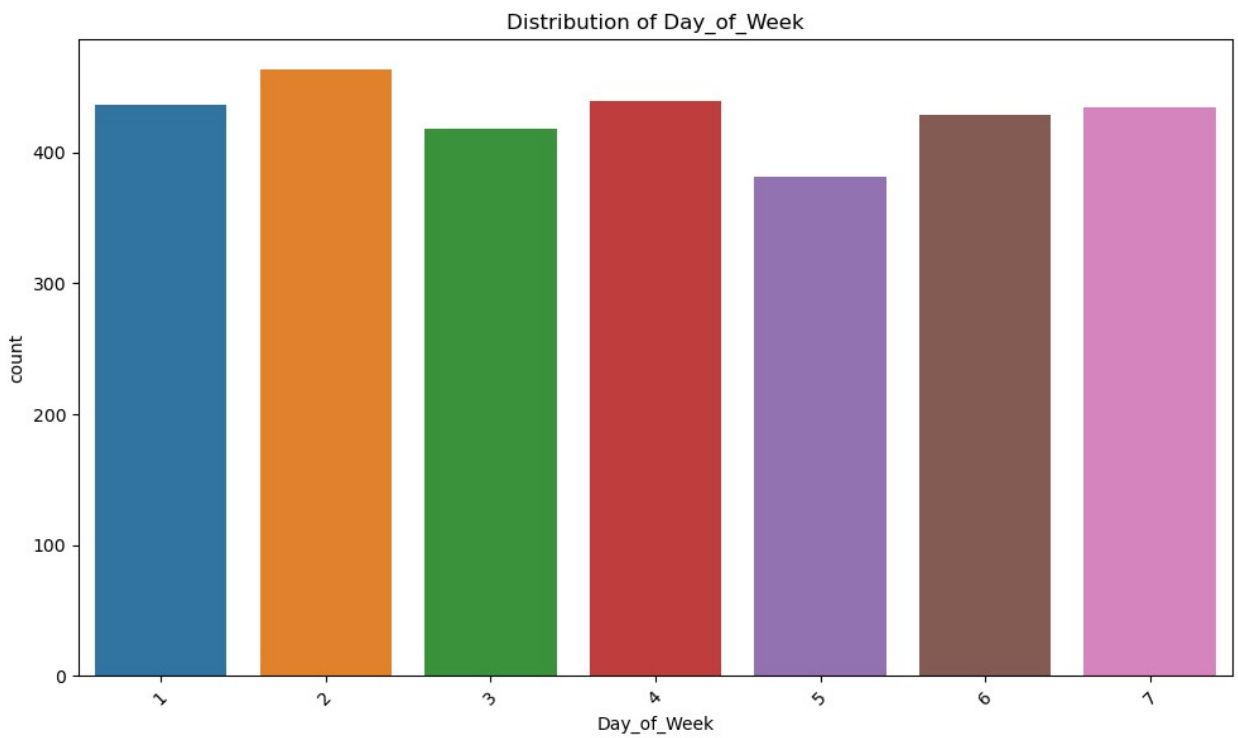
```
# Plotting histograms for numerical columns
Data.hist(figsize=(20, 15), bins=50)
plt.tight_layout()
plt.show()

# Plotting bar charts for categorical columns
categorical_columns = ['Airline', 'Origin_Airport',
'Destination_Airport', 'Day_of_Week', 'Month', 'Airplane_Type',
'Flight_Cancelled']

for column in categorical_columns:
    plt.figure(figsize=(10, 6))
    sns.countplot(x=column, data=Data)
    plt.title('Distribution of ' + column)
```

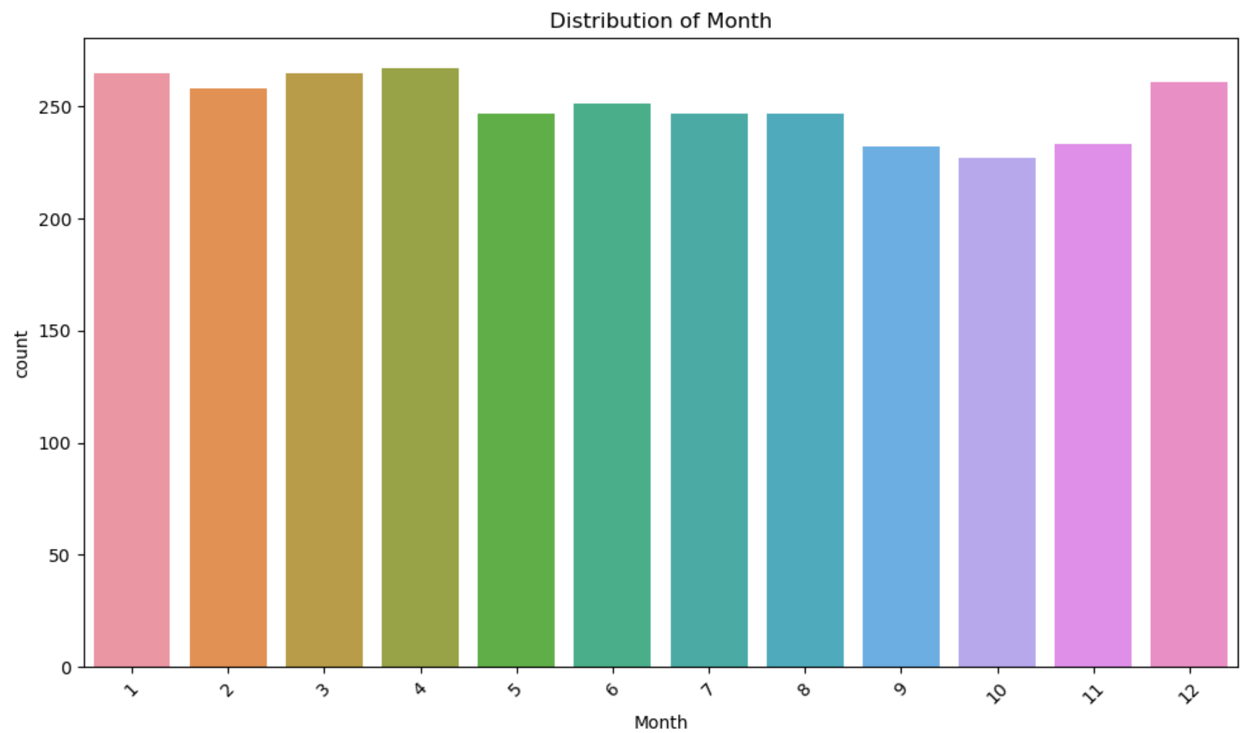
```
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```


Distribution of Airline
count
0
200
400
600
800
1000
1200
Airline D
Airline E
Airline C
Airline B
Airline A
Airline

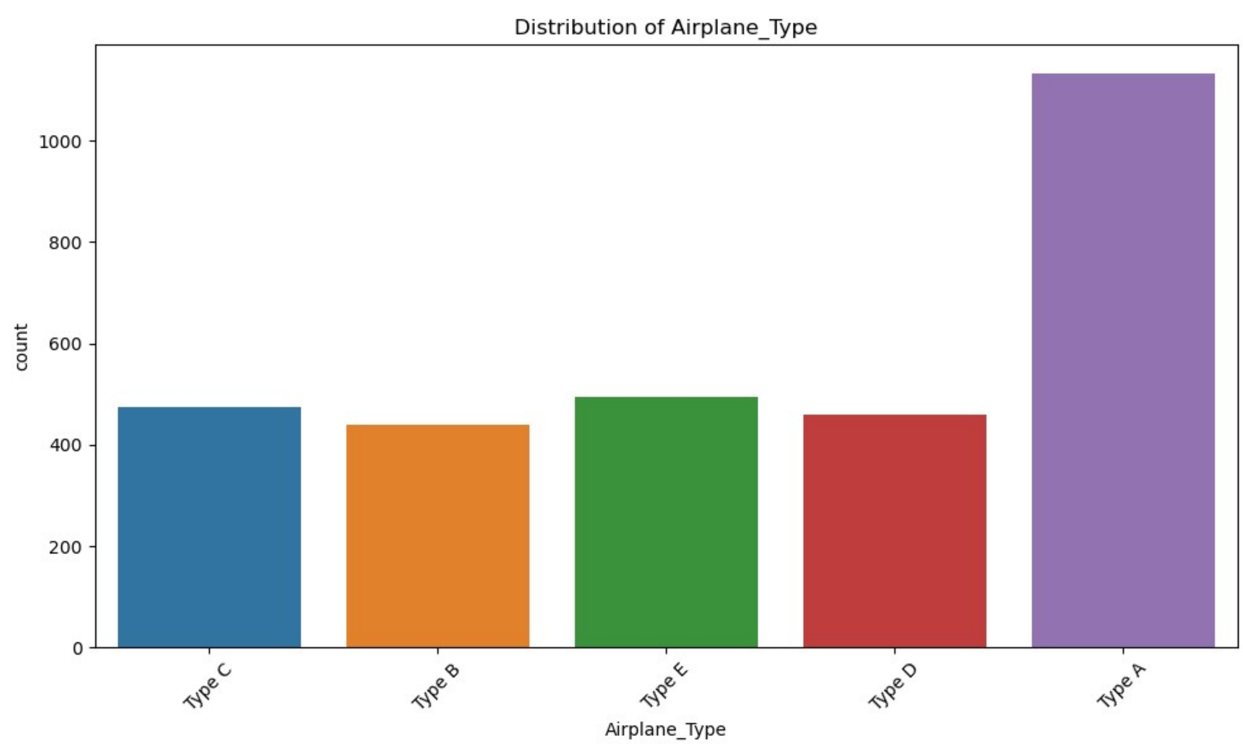

Distribution of Origin_Airport
count
0
200
400
600
800
1000
Airport 3
Airport 5
Airport 1
Airport 2
Airport 4
Origin_Airport


Distribution of Destination_Airport
count
0
200
400
600
800
1000
1200
1400
1600
Airport 2
Airport 4
Airport 3
Airport 5
Destination_Airport

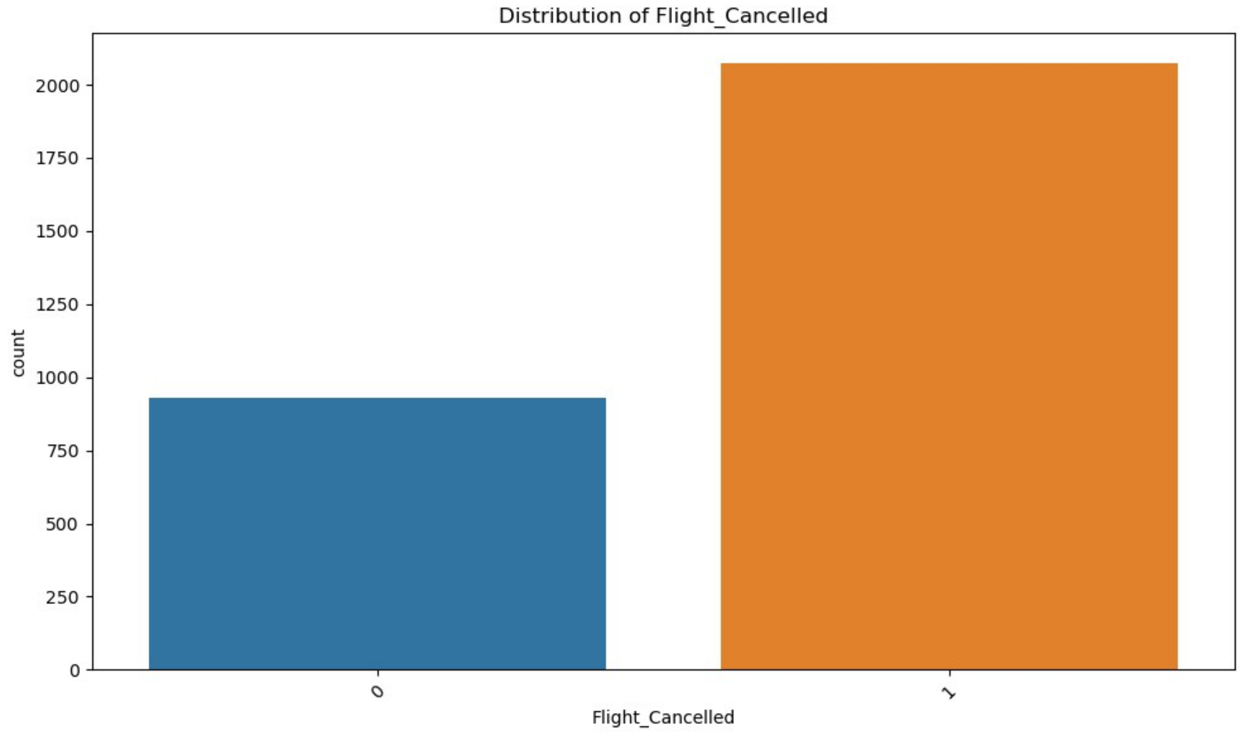

Distribution of Day_of_Week
count
0
100
200
300
400
1
2
3
4
5
6
7
Day_of_Week



Distribution of Month
count
0
50
100
150
200
250
1
2
3
4
5
6
7
8
9
10
11
12
Month



Distribution of Airplane_Type
count
0
200
400
600
800
1000
Type C
Type B
Type E
Type D
Type A
Airplane_Type

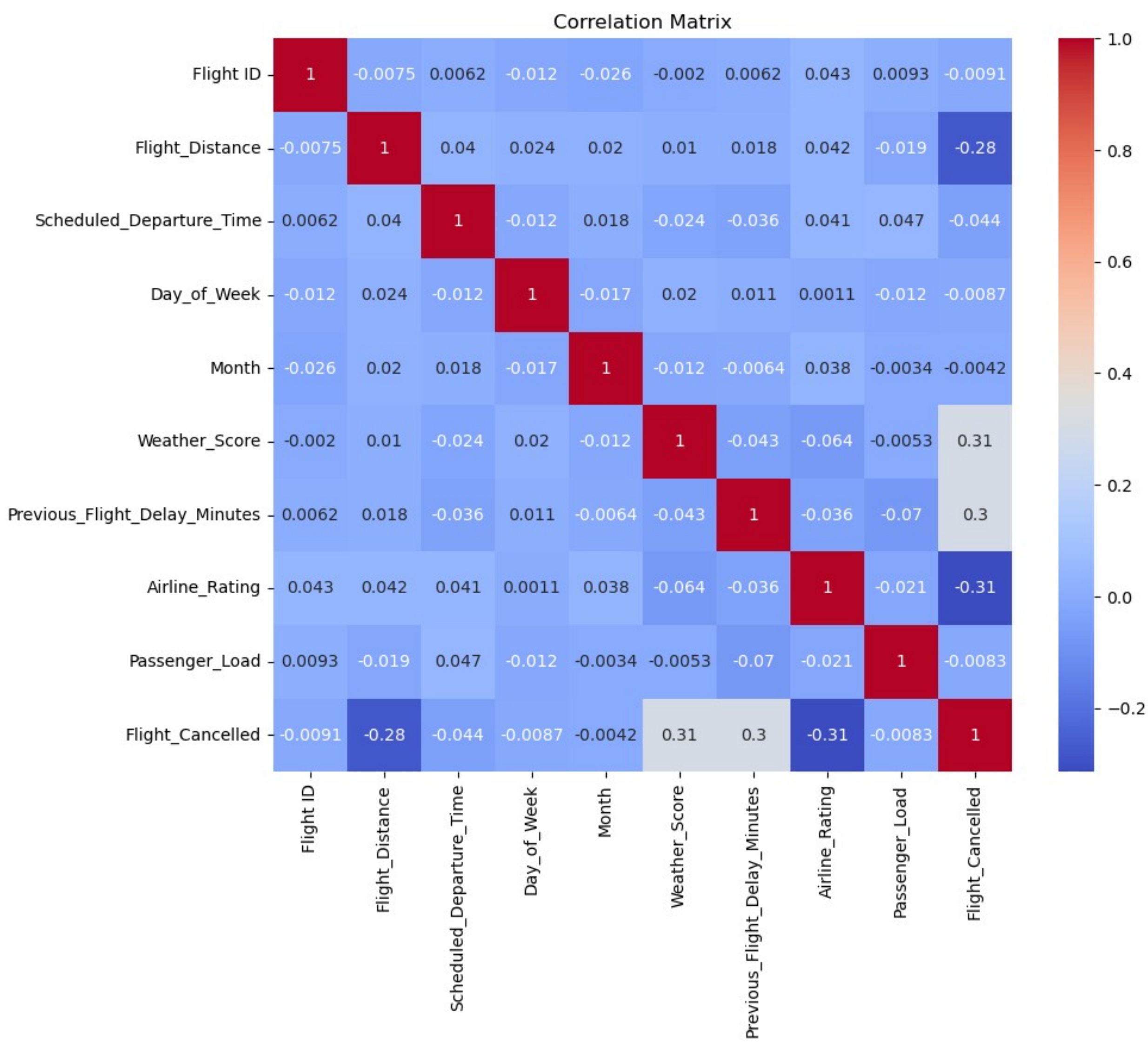


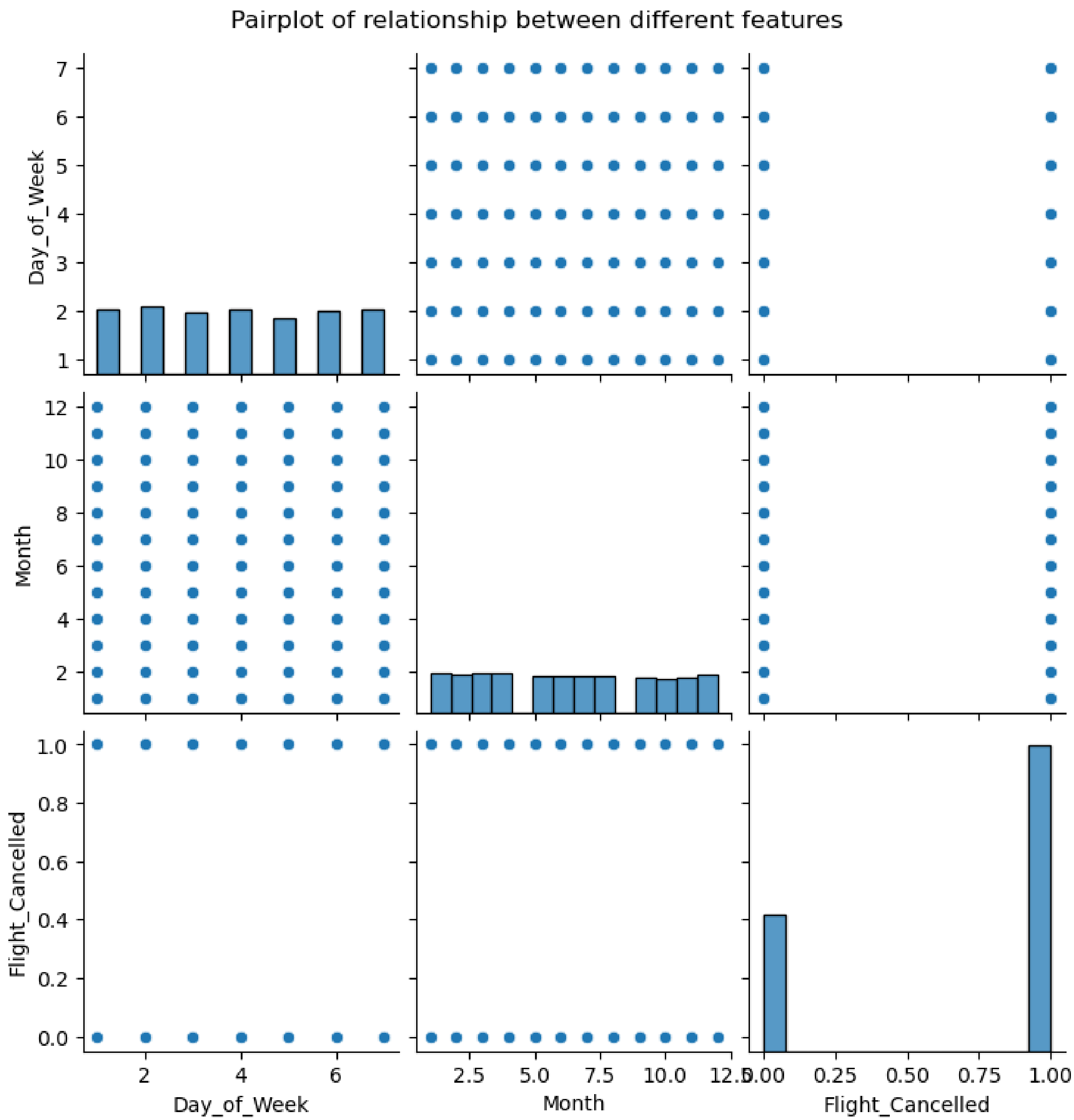
These visualizations provide a deeper understanding of the data's distribution across various features. For example, the histograms for numerical columns show the spread of values for features like flight distance, departure time, and weather score. The bar charts for categorical columns reveal the frequency of flights across different airlines, airports, days of the week, months, airplane types, and the proportion of flights cancelled.

• Relationship between features

```python
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Load the dataset
Data = pd.read_csv('Flyzy Flight Cancellation - Sheet1 (1).csv')

# Display the first few rows of the dataset
Data.head()

# Plotting correlation matrix
corr = Data.corr()
plt.figure(figsize=(10, 8))
sns.heatmap(corr, annot=True, cmap='coolwarm')
plt.title('Correlation Matrix')
plt.show()

# Pairplot for a subset of features
```

```
# Selecting a subset of features for clarity in visualization
features = ['Airline', 'Origin_Airport', 'Destination_Airport',
'Day_of_Week', 'Month', 'Airplane_Type', 'Flight_Cancelled']
sns.pairplot(Data[features])
plt.suptitle('Pairplot of relationship between different features',
y=1.02)
plt.show()

C:\Users\Student_0002\AppData\Local\Temp\
ipykernel_7120\2802903203.py:12: FutureWarning: The default value of
numeric_only in DataFrame.corr is deprecated. In a future version, it
will default to False. Select only valid columns or specify the value
of numeric_only to silence this warning.
  corr = Data.corr()
```



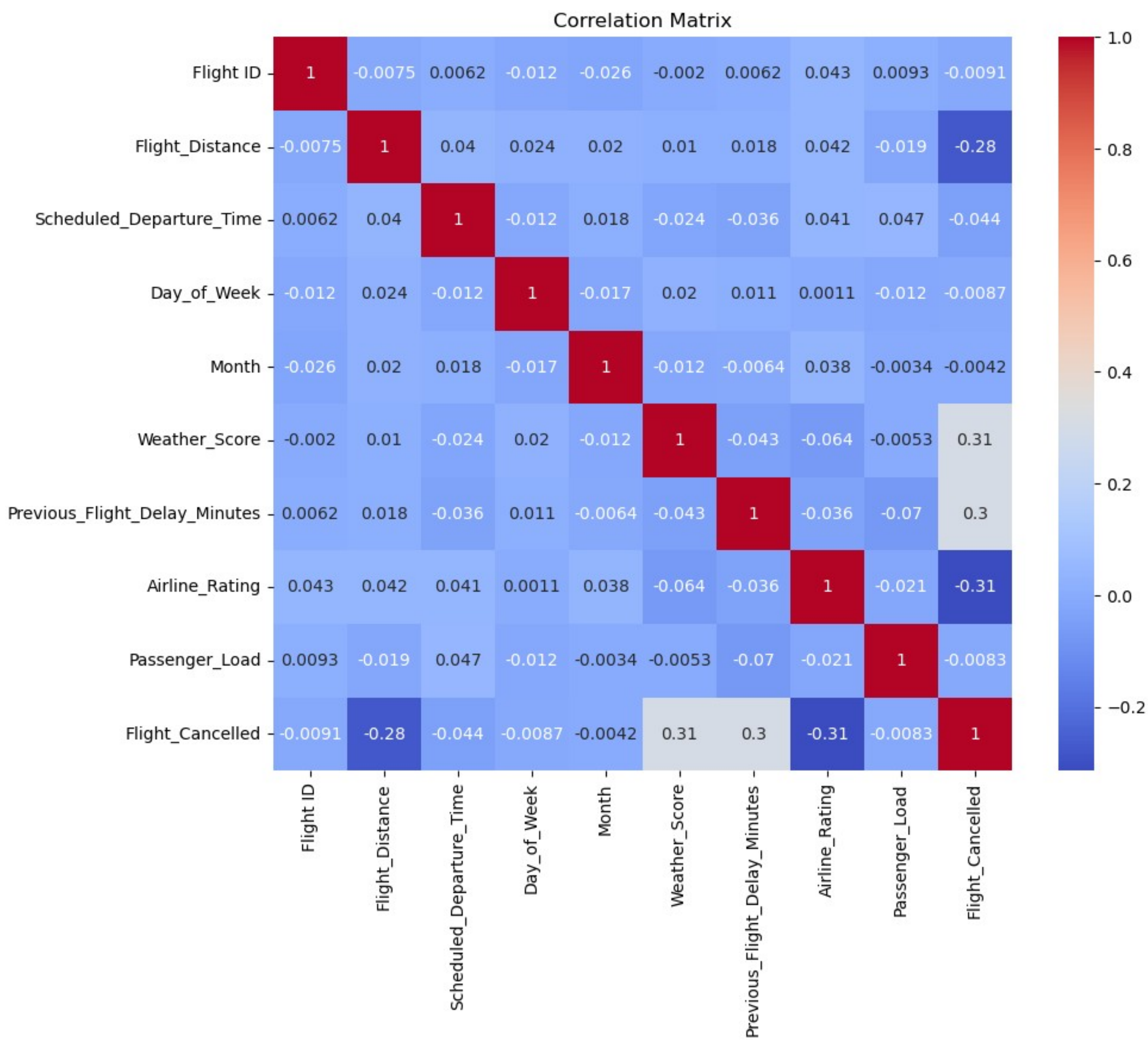


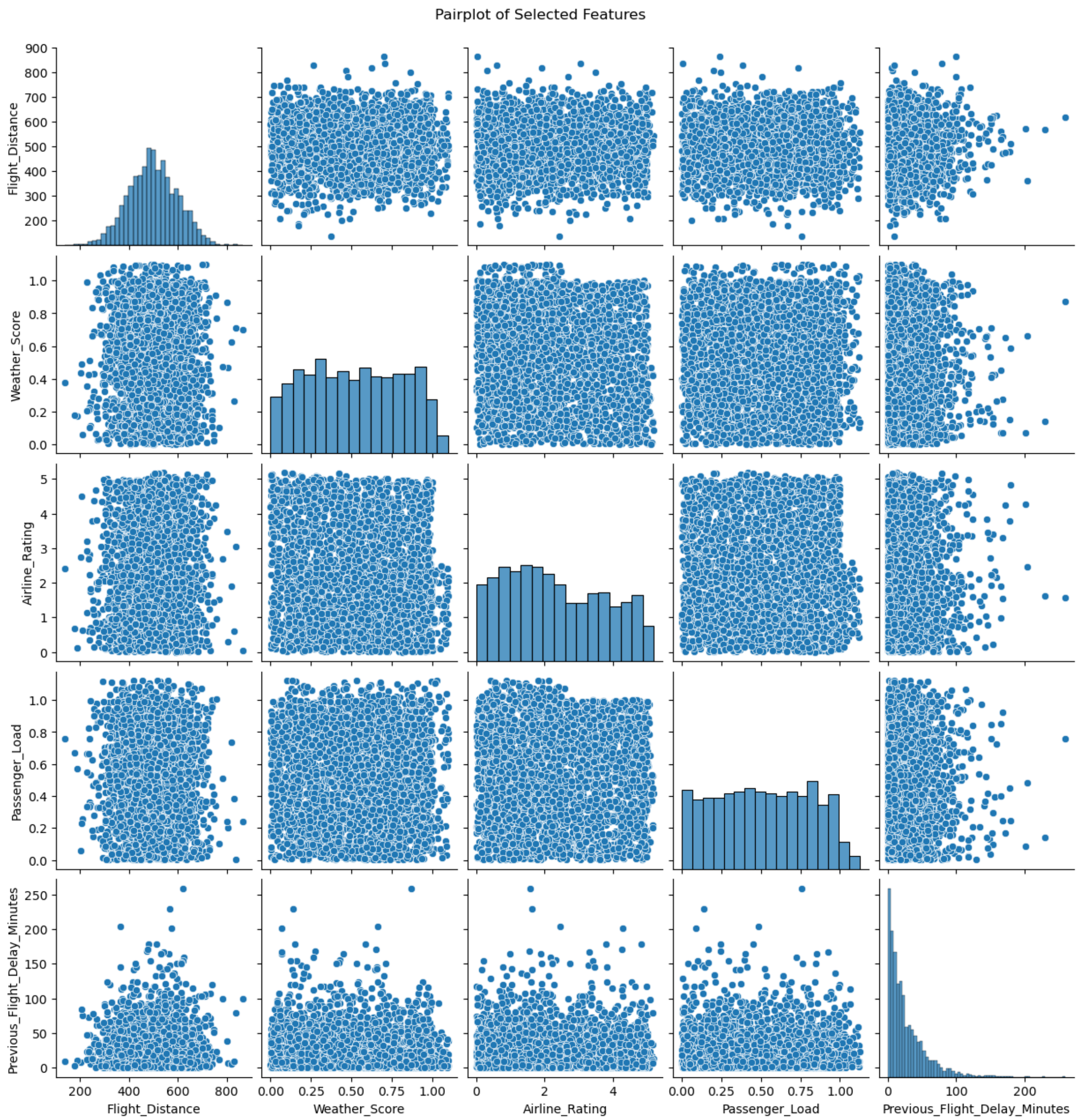
• Relationship between features and target variable

```
# Correcting the column names and plotting again
features = ['Flight_Distance', 'Weather_Score', 'Airline_Rating',
'Passenger_Load', 'Previous_Flight_Delay_Minutes']

# Plotting correlation matrix
corr = Data.corr()
```

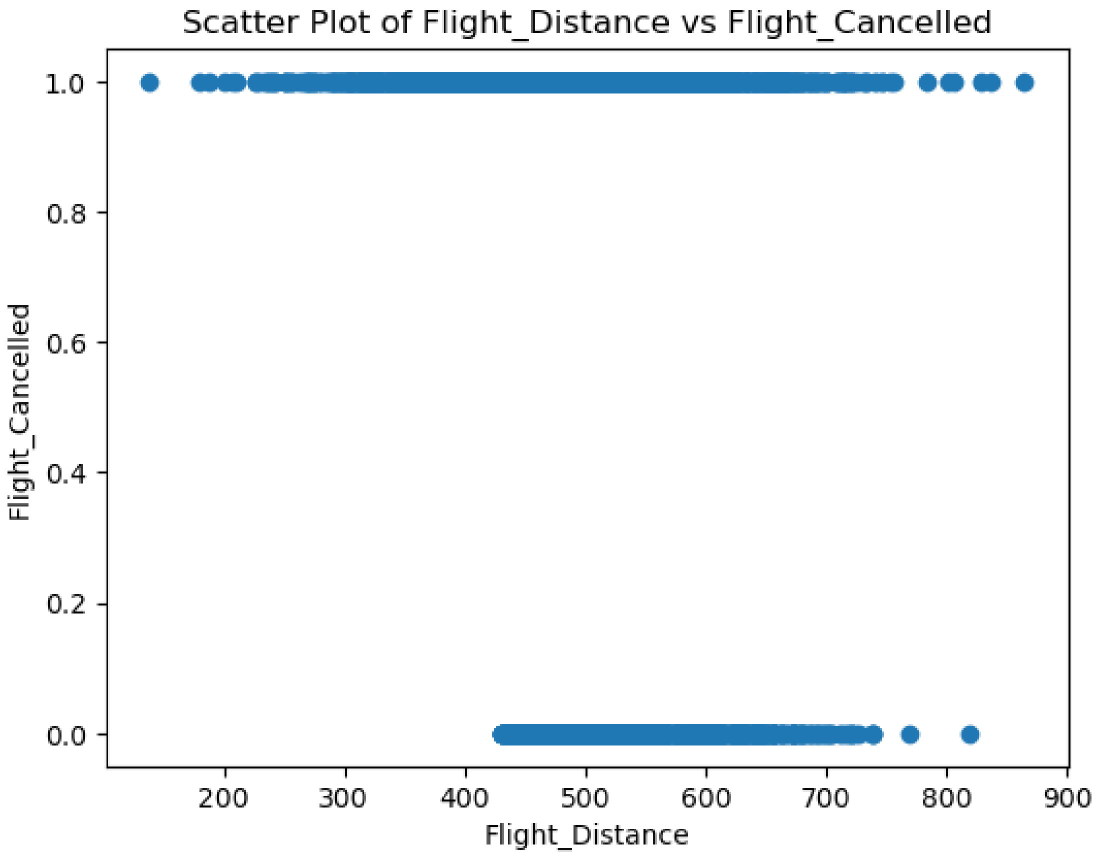
```
plt.figure(figsize=(10, 8))
sns.heatmap(corr, annot=True, cmap='coolwarm')
plt.title('Correlation Matrix')
plt.show()

# Pairplot for the corrected subset of features
sns.pairplot(Data[features])
plt.suptitle('Pairplot of Selected Features', y=1.02)
plt.show()
```

```
C:\Users\Student_0002\AppData\Local\Temp\
ipykernel_7120\2141227160.py:5: FutureWarning: The default value of
numeric_only in DataFrame.corr is deprecated. In a future version, it
will default to False. Select only valid columns or specify the value
of numeric_only to silence this warning.
  corr = Data.corr()
```






```
import matplotlib.pyplot as plt

# Scatter plot with target variable
plt.scatter(Data['Flight_Distance'], Data['Flight_Cancelled'])
plt.xlabel('Flight_Distance')
plt.ylabel('Flight_Cancelled')
plt.title('Scatter Plot of Flight_Distance vs Flight_Cancelled')
plt.show()
```


Scatter Plot of Flight_Distance vs Flight_Cancelled
Flight_Cancelled
1.0
0.8
0.6
0.4
0.2
0.0
200
300
400
500
600
700
800
900
Flight_Distance