Model Evaluation

October 1, 2024

0.1 Data Checking

[1]:
```
import pandas as pd
import numpy as np
```

[2]:
```
data = pd.read_csv('Flyzy Flight Cancellation - Sheet1.csv')
data.iloc[1:4, 2:3]= np.NaN
data.iloc[1:4, 3:4]= "NA"
data.iloc[1:4, 4:5]= ""
data["None_col"]= None
data.head()
```

[2]:
```
   Flight ID    Airline  Flight_Distance Origin_Airport Destination_Airport  \
0    7319483  Airline D            475.0      Airport 3           Airport 2
1    4791965  Airline E              NaN             NA
2    2991718  Airline C              NaN             NA
3    4220106  Airline E              NaN             NA
4    2263008  Airline E            566.0      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 None_col
0                 0     None
1                 1     None
2                 0     None
3                 1     None
```

```
4                  0     None
```

```
[3]: null= pd.isnull(data)
     null.head()
```

```
[3]:    Flight ID  Airline  Flight_Distance  Origin_Airport  Destination_Airport  \
     0      False    False            False           False                False
     1      False    False             True           False                False
     2      False    False             True           False                False
     3      False    False             True           False                False
     4      False    False            False           False                False

        Scheduled_Departure_Time  Day_of_Week  Month  Airplane_Type  Weather_Score  \
     0                     False        False  False          False          False
     1                     False        False  False          False          False
     2                     False        False  False          False          False
     3                     False        False  False          False          False
     4                     False        False  False          False          False

        Previous_Flight_Delay_Minutes  Airline_Rating  Passenger_Load  \
     0                          False           False           False
     1                          False           False           False
     2                          False           False           False
     3                          False           False           False
     4                          False           False           False

        Flight_Cancelled  None_col
     0             False      True
     1             False      True
     2             False      True
     3             False      True
     4             False      True
```

```
[4]: pd.isnull(data).sum().sum()
```

```
[4]: 3003
```

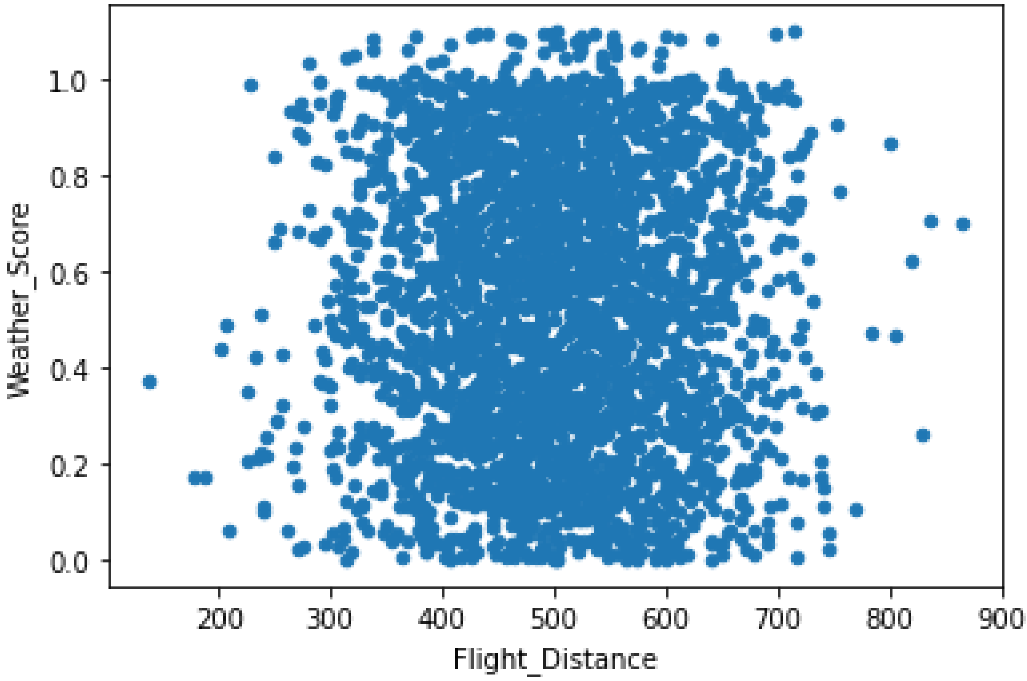
```
[5]: missing_values = data.isnull().sum()
     print("Missing values per column (before handling):")
     print(missing_values)
```

```
Missing values per column (before handling):
Flight ID                        0
Airline                          0
Flight_Distance                  3
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
None_col                         3000
dtype: int64
```

[6]:

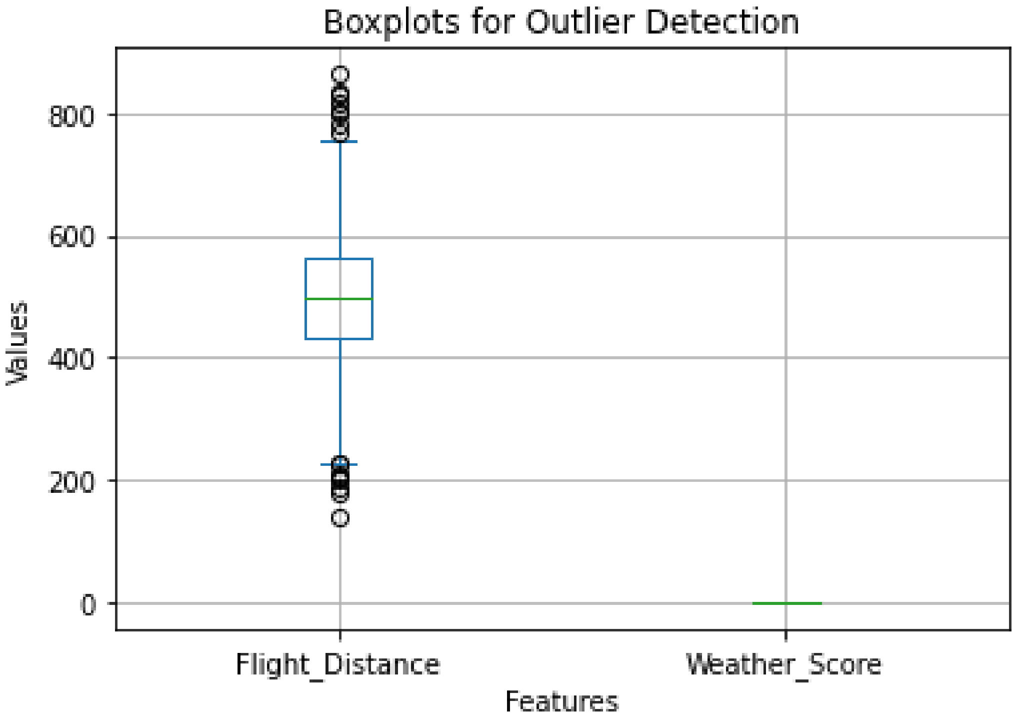
```python
import pandas as pd
import matplotlib.pyplot as plt
data.plot(kind='scatter' , x= 'Flight_Distance', y= 'Weather_Score')
plt.show()
```



[7]:

```python
import matplotlib.pyplot as plt
import pandas as pd
columns_to_check = ['Flight_Distance', 'Weather_Score']
plt.figure(figsize=(10, 6))
data[columns_to_check].plot(kind= 'box')
plt.title('Boxplots for Outlier Detection')
plt.xlabel('Features')
plt.ylabel('Values')
plt.grid(True)
plt.show()
```

```
<Figure size 720x432 with 0 Axes>
```



[8]:
```
data_types = data.dtypes
print("Data types of each column:")
print(data_types)
```

```
Data types of each column:
Flight ID                          int64
Airline                           object
Flight_Distance                  float64
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
None_col                               object
dtype: object
```

```
[9]: Data =pd.read_csv('Flyzy Flight Cancellation - Sheet1.csv')
```

```
[10]: print(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
```

```
[11]: print(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
```

EDA

[12]:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

[13]:

```
Data =pd.read_csv('Flyzy Flight Cancellation - Sheet1.csv')
```

[14]:

```
print(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
```

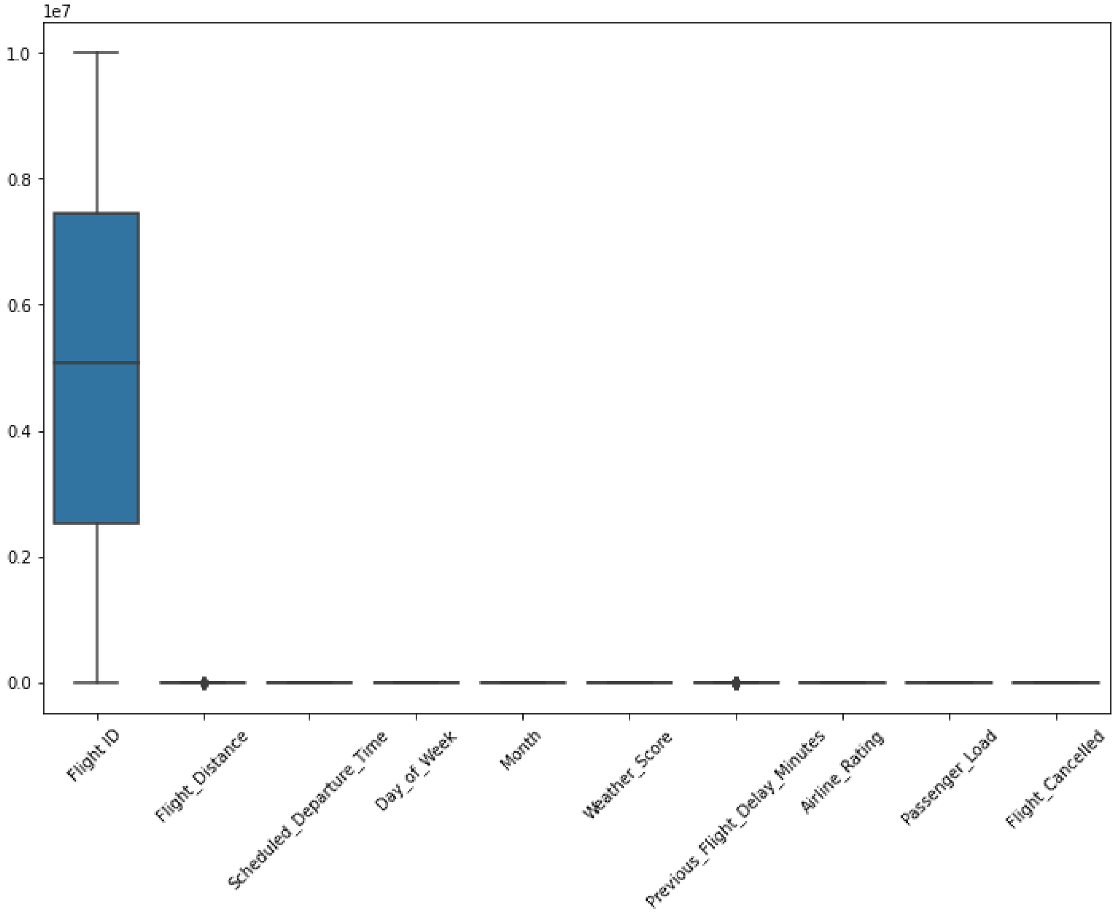
```
[15]: print(data.isnull().sum())
```

```
Flight ID                          0
Airline                            0
Flight_Distance                    3
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
None_col                        3000
dtype: int64
```

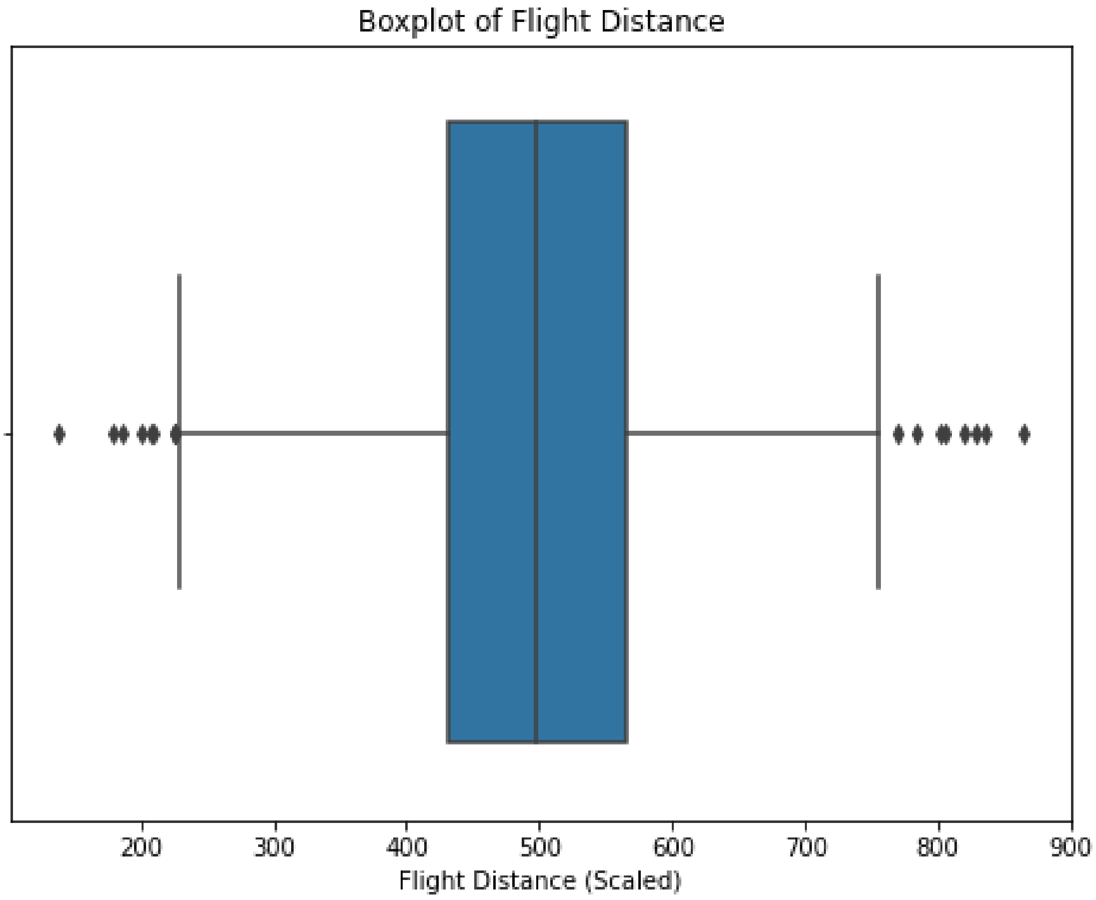
```
[16]: print(Data.dtypes)
```

```
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
```

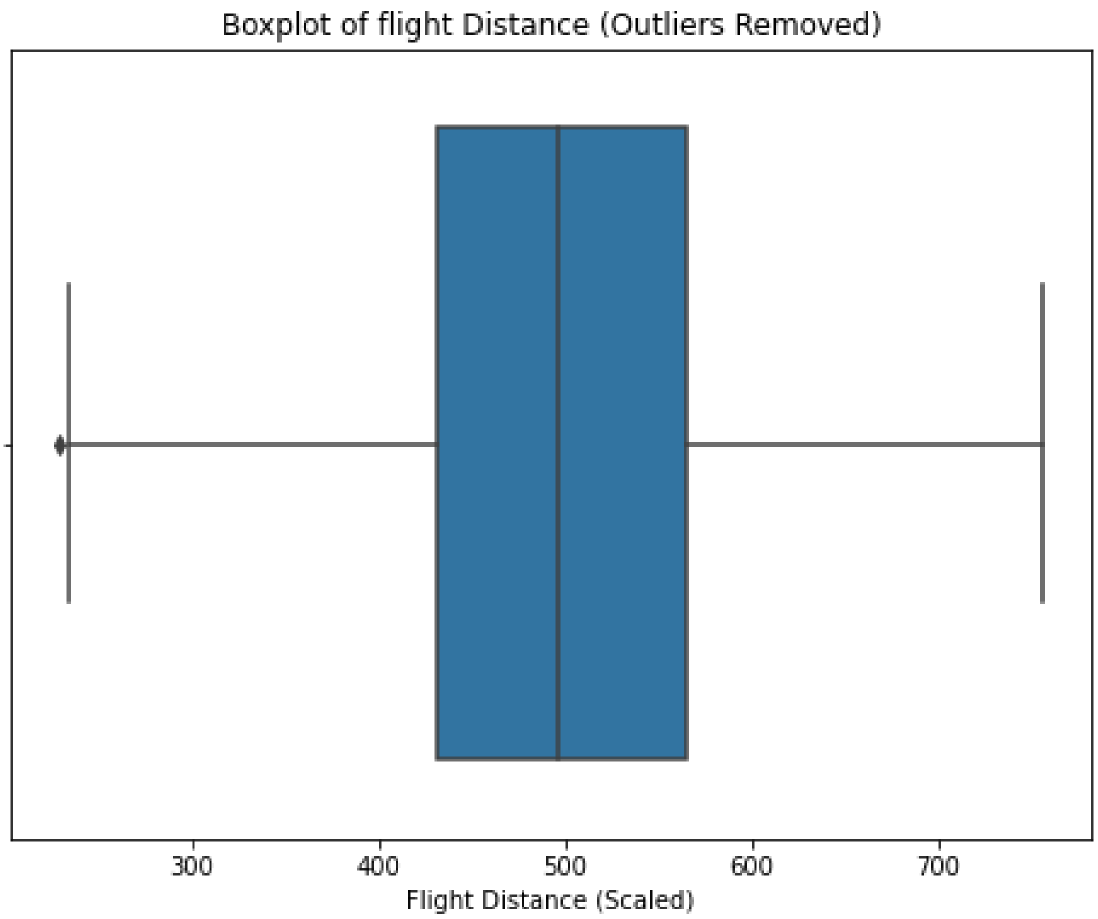
[17]:
```
plt.figure(figsize=(12,8))
sns.boxplot(data=Data)
plt.xticks(rotation=45)
plt.show()
```



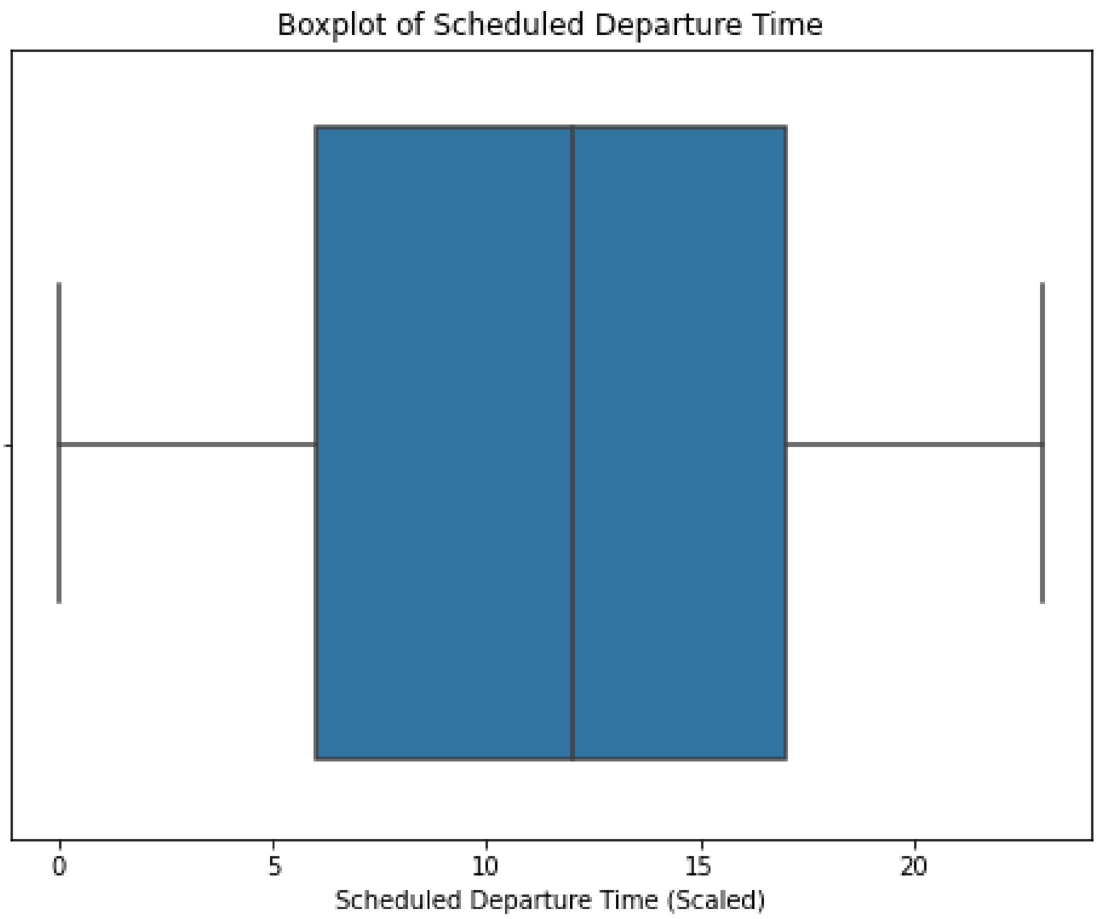
[18]:
```
import seaborn as sns
import matplotlib.pyplot as plt
plt.figure(figsize=(8, 6))
sns.boxplot(x=Data['Flight_Distance'])
plt.title('Boxplot of Flight Distance')
plt.xlabel('Flight Distance (Scaled)')
plt.show()
```



```
[19]: Q1 = Data['Flight_Distance'].quantile(0.25)
      Q3 = Data['Flight_Distance'].quantile(0.75)
      IQR = Q3 - Q1
      lower_bound = Q1 - 1.5 * IQR
      upper_bound = Q3 + 1.5 * IQR
      filtered_Data = Data[(Data['Flight_Distance'] >= lower_bound) &␣
        ↪(Data['Flight_Distance'] <= upper_bound)]
      plt.figure(figsize=(8, 6))
      sns.boxplot(x=filtered_Data['Flight_Distance'])
      plt.title('Boxplot of flight Distance (Outliers Removed)')
      plt.xlabel('Flight Distance (Scaled)')
      plt.show()
```

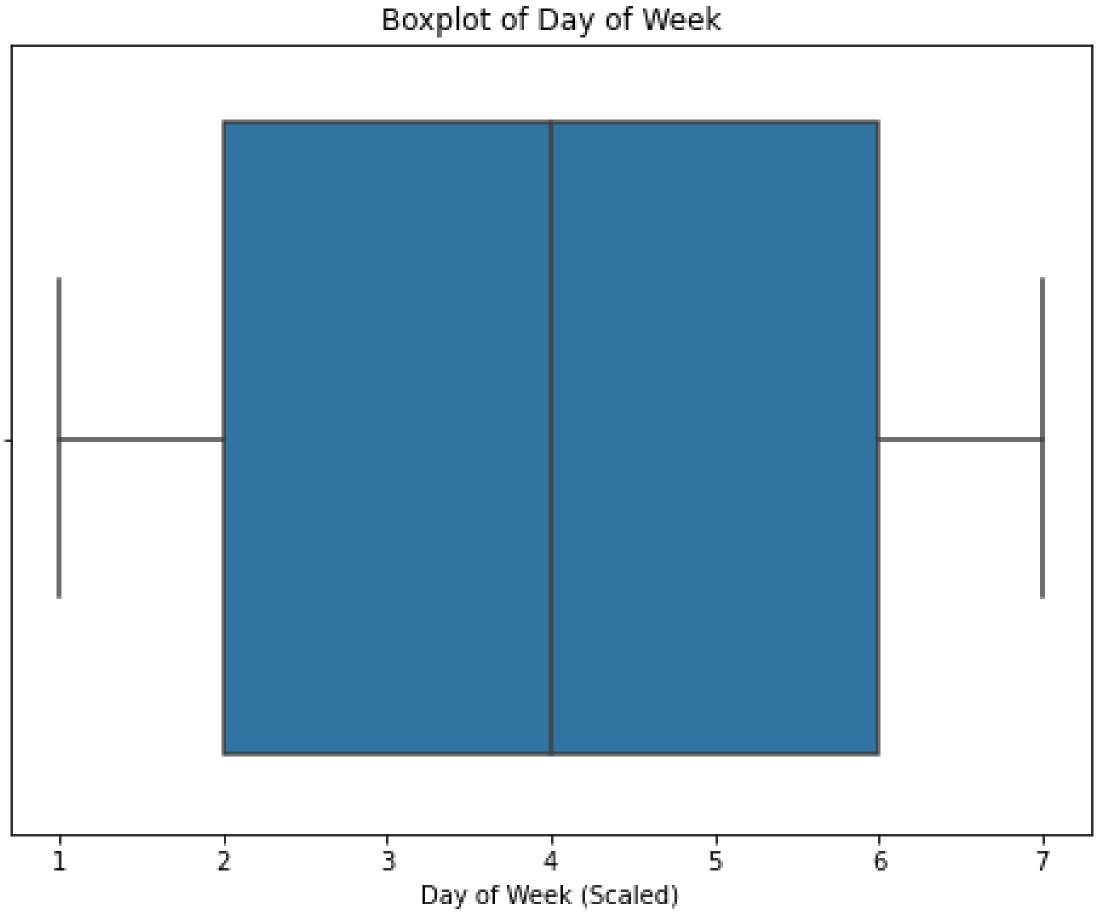


[20]:
```
import seaborn as sns
import matplotlib.pyplot as plt
plt.figure(figsize=(8, 6))
sns.boxplot(x=Data['Scheduled_Departure_Time'])
plt.title('Boxplot of Scheduled Departure Time')
plt.xlabel('Scheduled Departure Time (Scaled)')
plt.show()
```

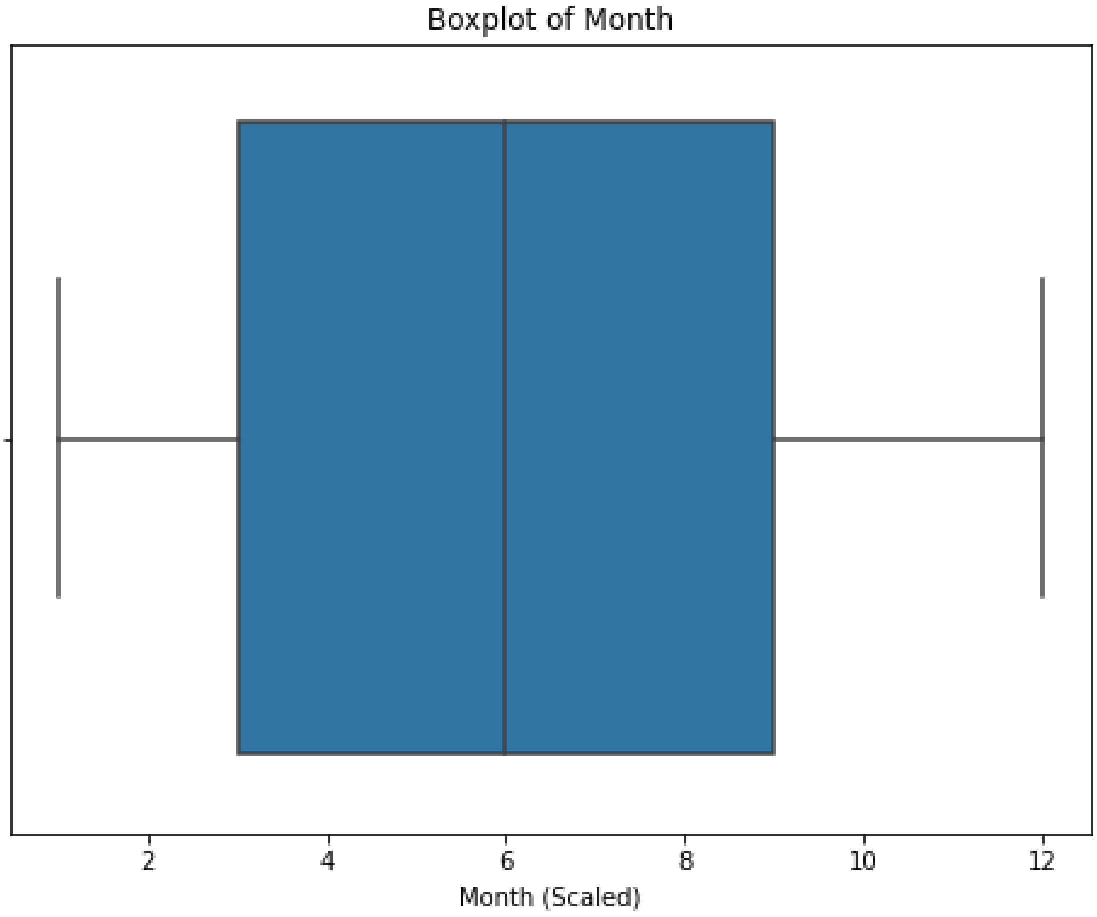


[21]:

```
import seaborn as sns
import matplotlib.pyplot as plt
plt.figure(figsize=(8, 6))
sns.boxplot(x=Data['Day_of_Week'])
plt.title('Boxplot of Day of Week')
plt.xlabel('Day of Week (Scaled)')
plt.show()
```

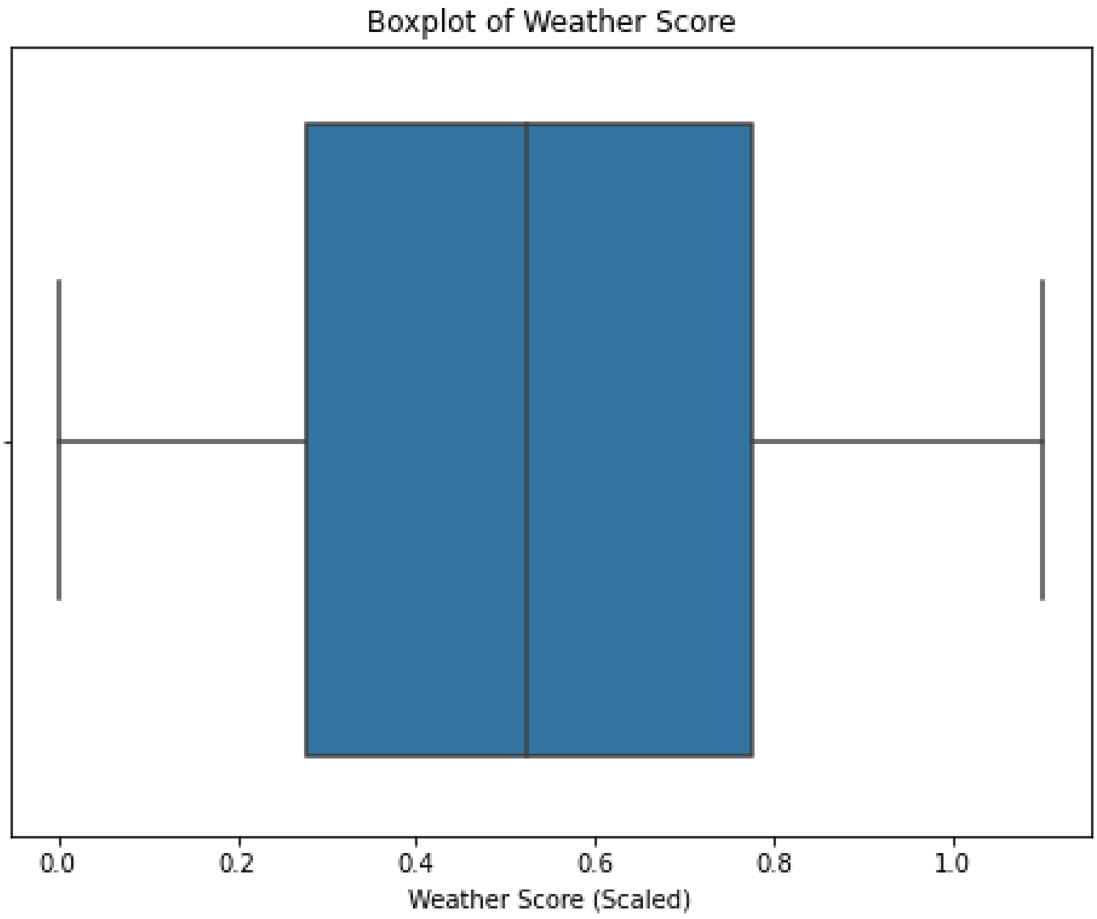


[22]:
```
plt.figure(figsize=(8, 6))
sns.boxplot(x=Data['Month'])
plt.title('Boxplot of Month')
plt.xlabel('Month (Scaled)')
plt.show()
```



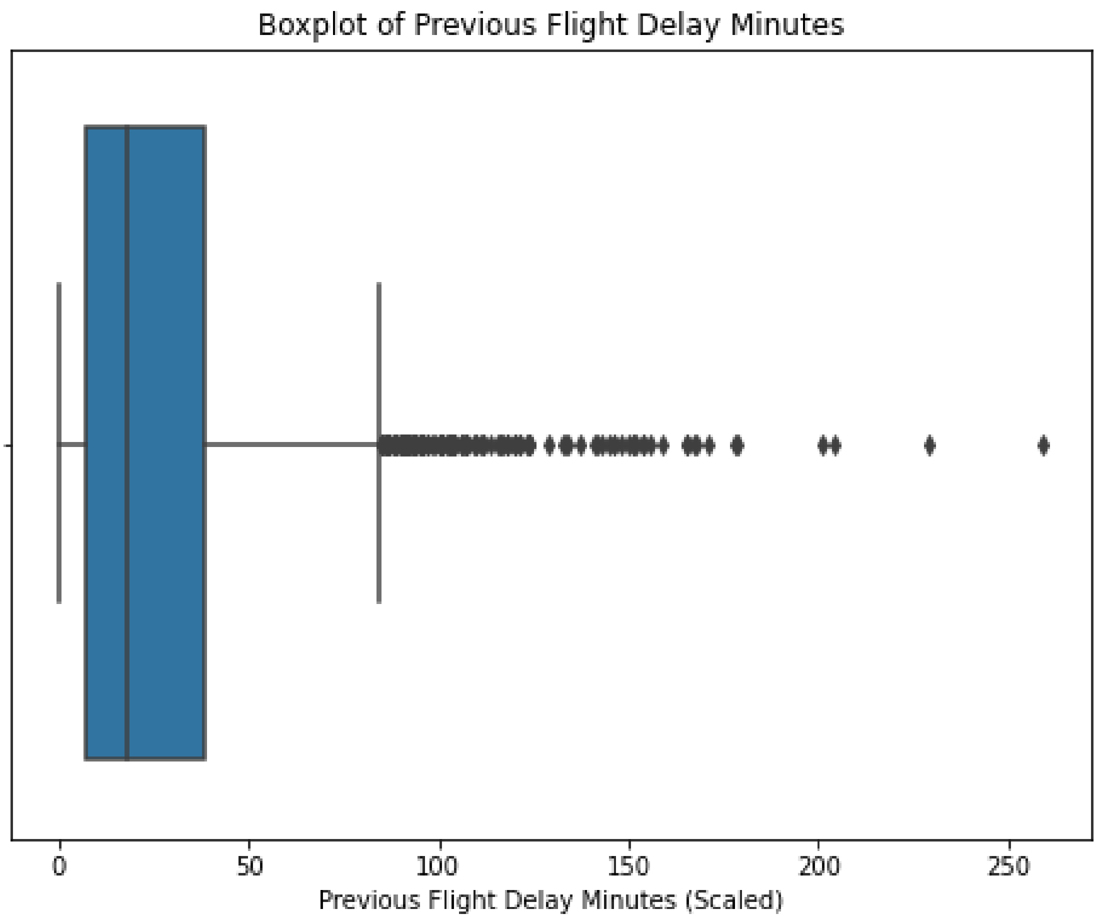
[23]:

```
import seaborn as sns
import matplotlib.pyplot as plt
plt.figure(figsize=(8, 6))
sns.boxplot(x=Data['Weather_Score'])
plt.title('Boxplot of Weather Score')
plt.xlabel('Weather Score (Scaled)')
plt.show()
```

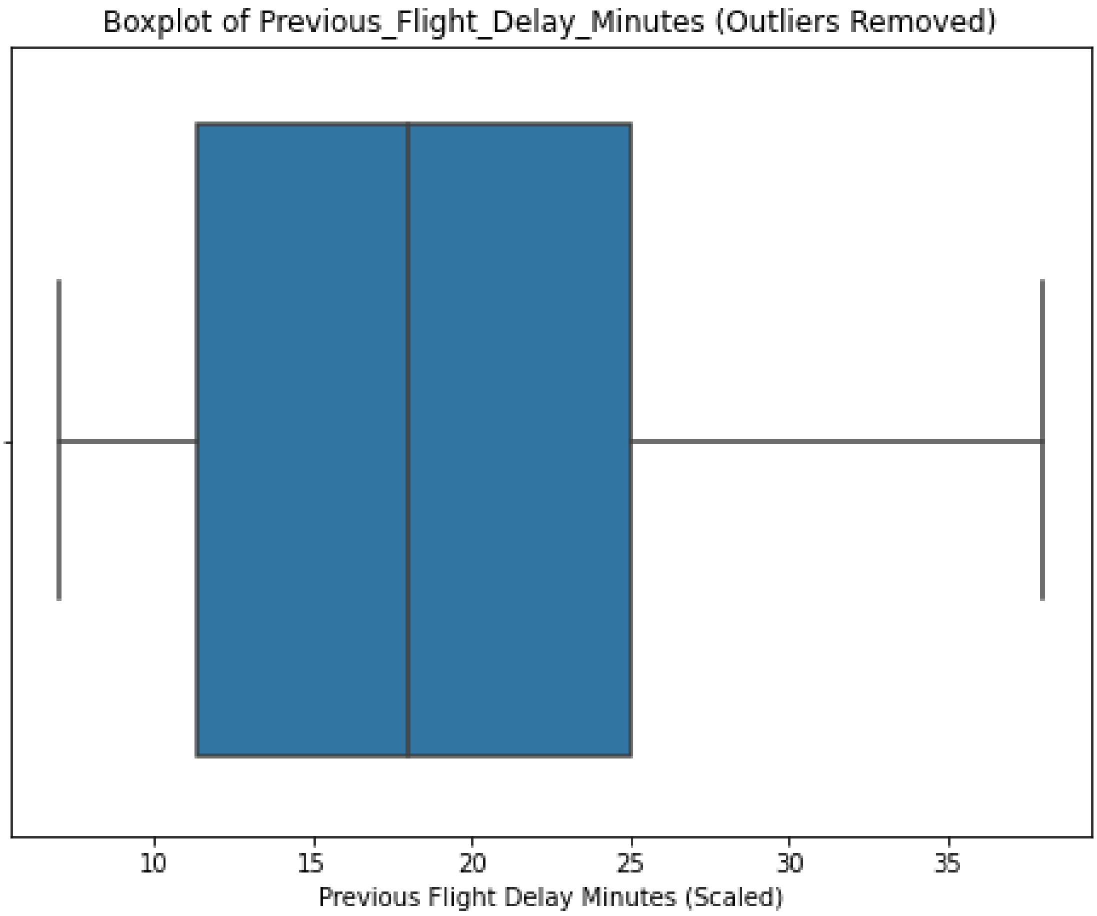


[24]:

```
import seaborn as sns
import matplotlib.pyplot as plt
plt.figure(figsize=(8, 6))
sns.boxplot(x=Data['Previous_Flight_Delay_Minutes'])
plt.title('Boxplot of Previous Flight Delay Minutes')
plt.xlabel('Previous Flight Delay Minutes (Scaled)')
plt.show()
```

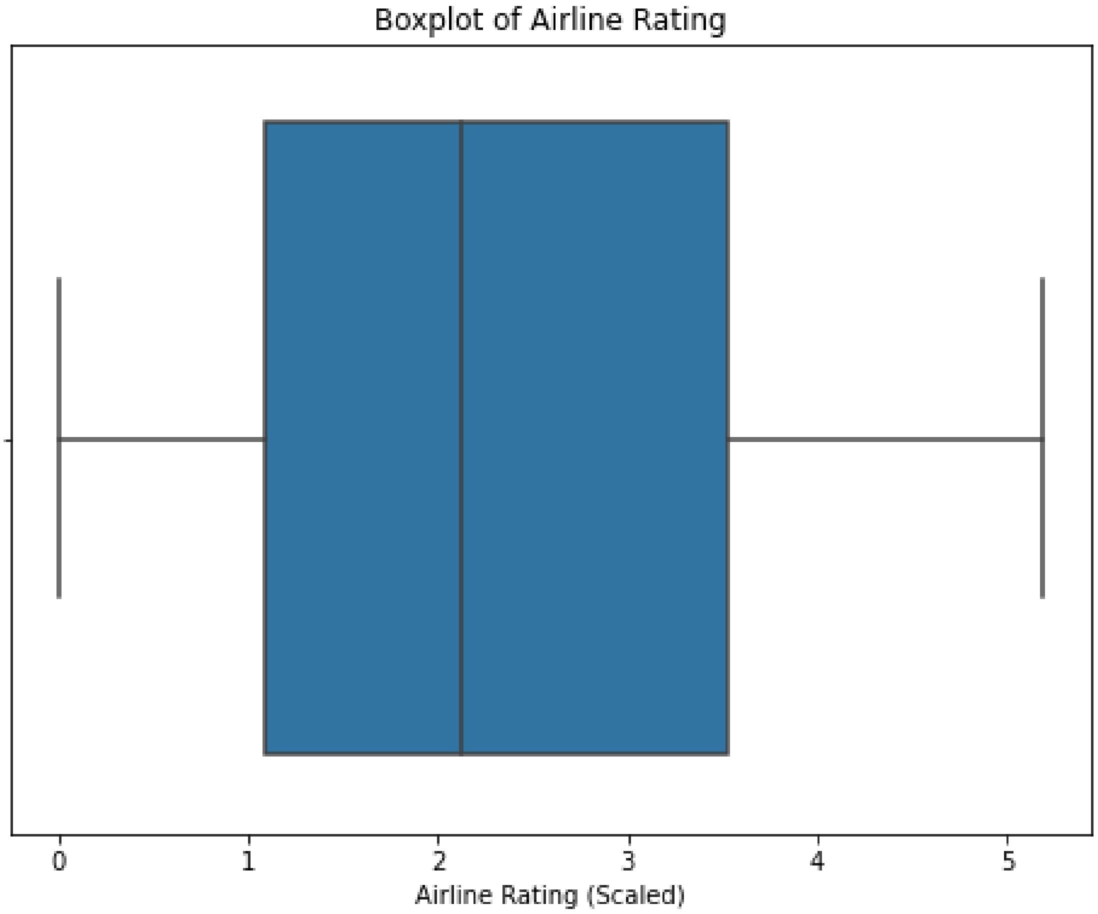


[25]:
```
Q1 = Data['Previous_Flight_Delay_Minutes'].quantile(0.25)
Q3 = Data['Previous_Flight_Delay_Minutes'].quantile(0.75)
IQR = Q3 - Q1
lower_bound = Q1 - 0.0 * IQR
upper_bound = Q3 + 0.0 * IQR
filtered_Data = Data[(Data['Previous_Flight_Delay_Minutes'] >= lower_bound) &␣
 ↪(Data['Previous_Flight_Delay_Minutes'] <= upper_bound)]
plt.figure(figsize=(8, 6))
sns.boxplot(x=filtered_Data['Previous_Flight_Delay_Minutes'])
plt.title('Boxplot of Previous_Flight_Delay_Minutes (Outliers Removed)')
plt.xlabel('Previous Flight Delay Minutes (Scaled)')
plt.show()
```

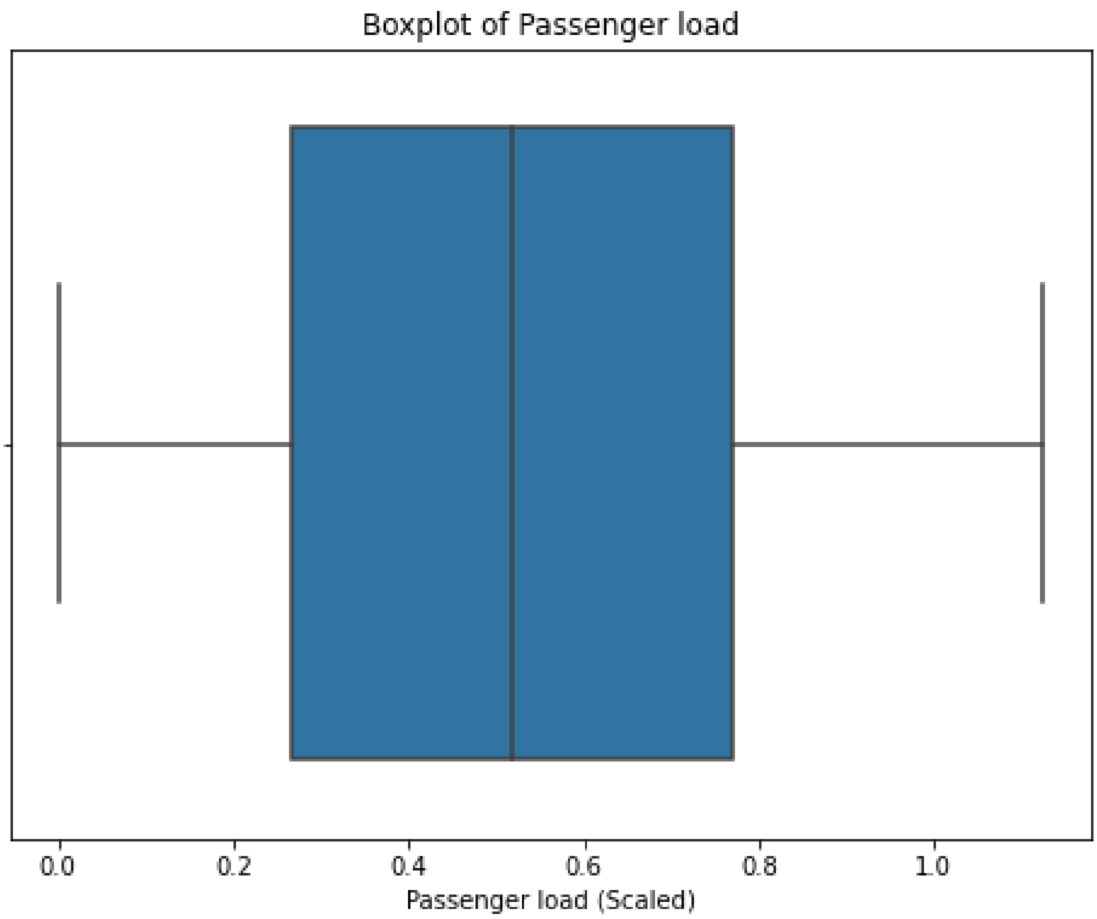


[26]:

```
plt.figure(figsize=(8, 6))
sns.boxplot(x=Data['Airline_Rating'])
plt.title('Boxplot of Airline Rating')
plt.xlabel('Airline Rating (Scaled)')
plt.show()
```

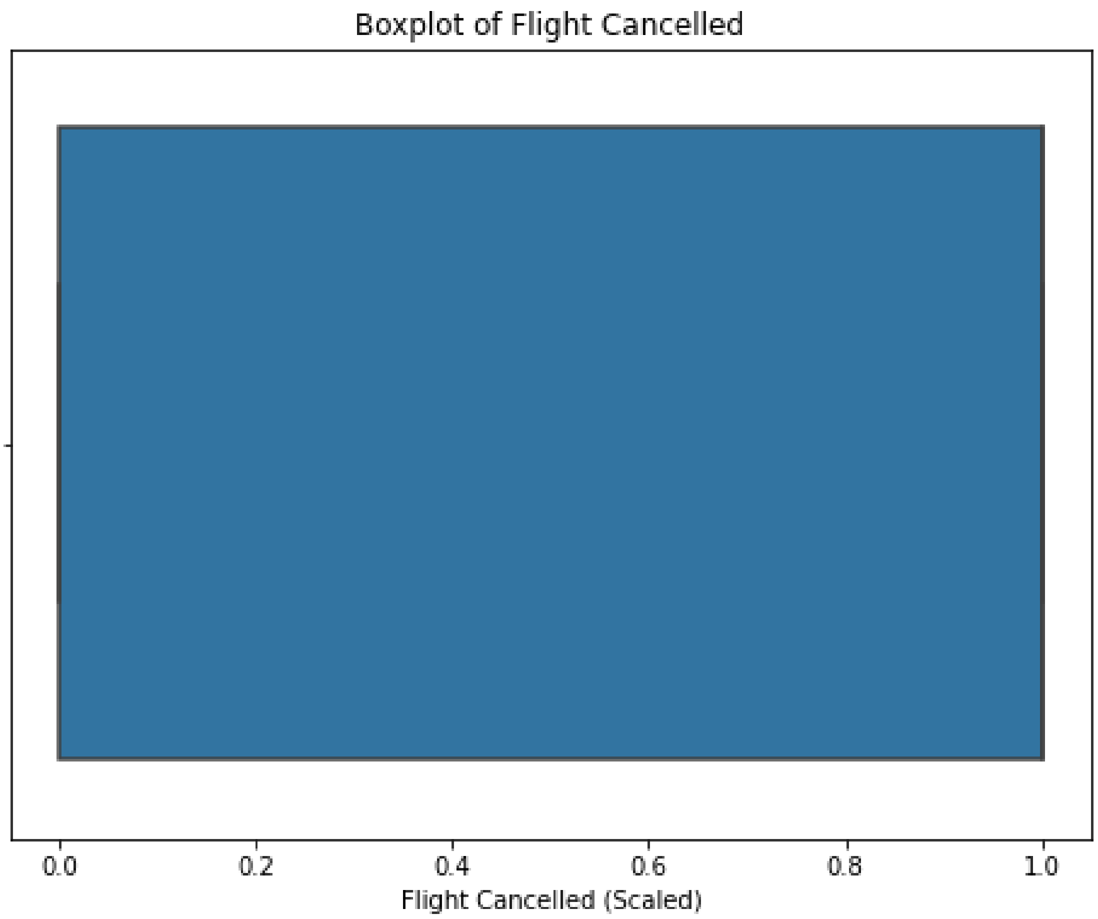


[27]:
```
plt.figure(figsize=(8, 6))
sns.boxplot(x=Data['Passenger_Load'])
plt.title('Boxplot of Passenger load')
plt.xlabel('Passenger load (Scaled)')
plt.show()
```



[28]:

```
plt.figure(figsize=(8, 6))
sns.boxplot(x=Data['Flight_Cancelled'])
plt.title('Boxplot of Flight Cancelled')
plt.xlabel('Flight Cancelled (Scaled)')
plt.show()
```



```
[29]: data = pd.read_csv('Flyzy Flight Cancellation - Sheet1.csv')
```

```
[30]: data.describe()
```

```
[30]:           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
```

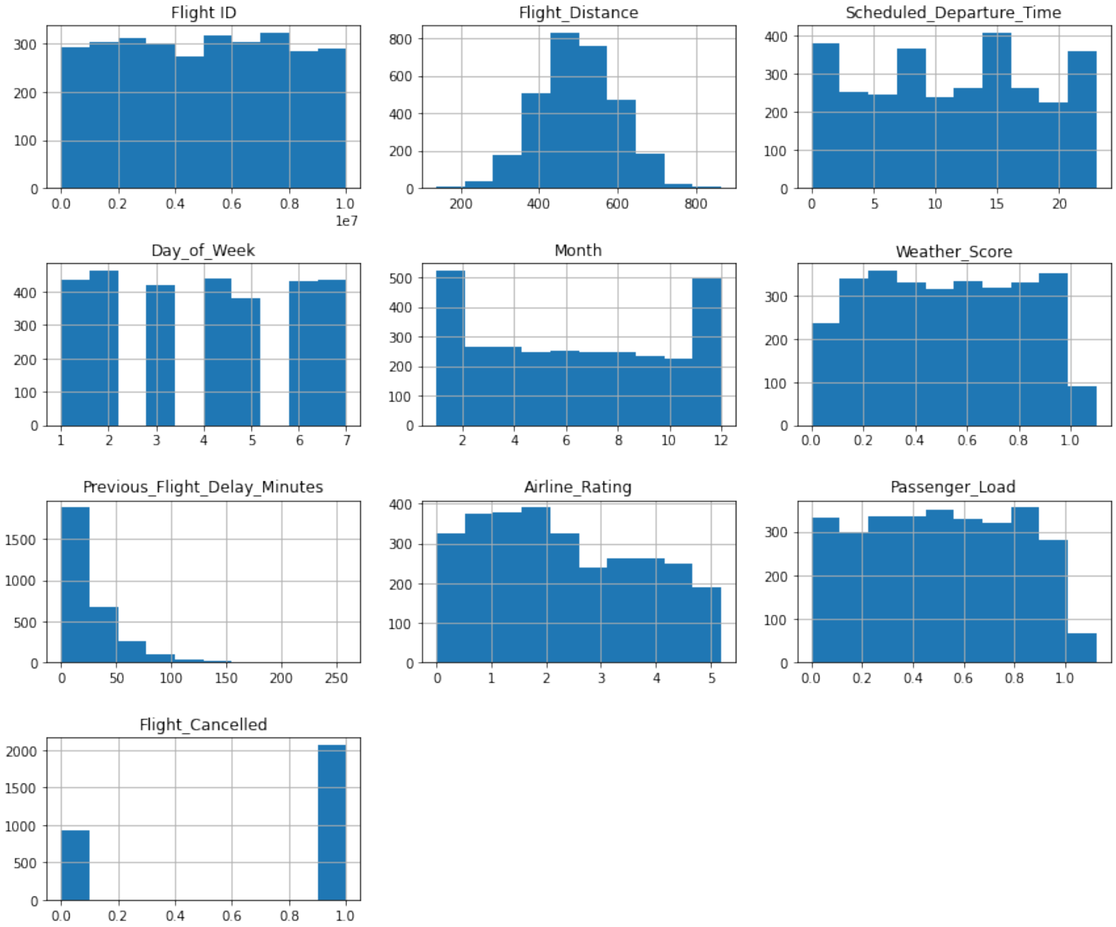
[31]:

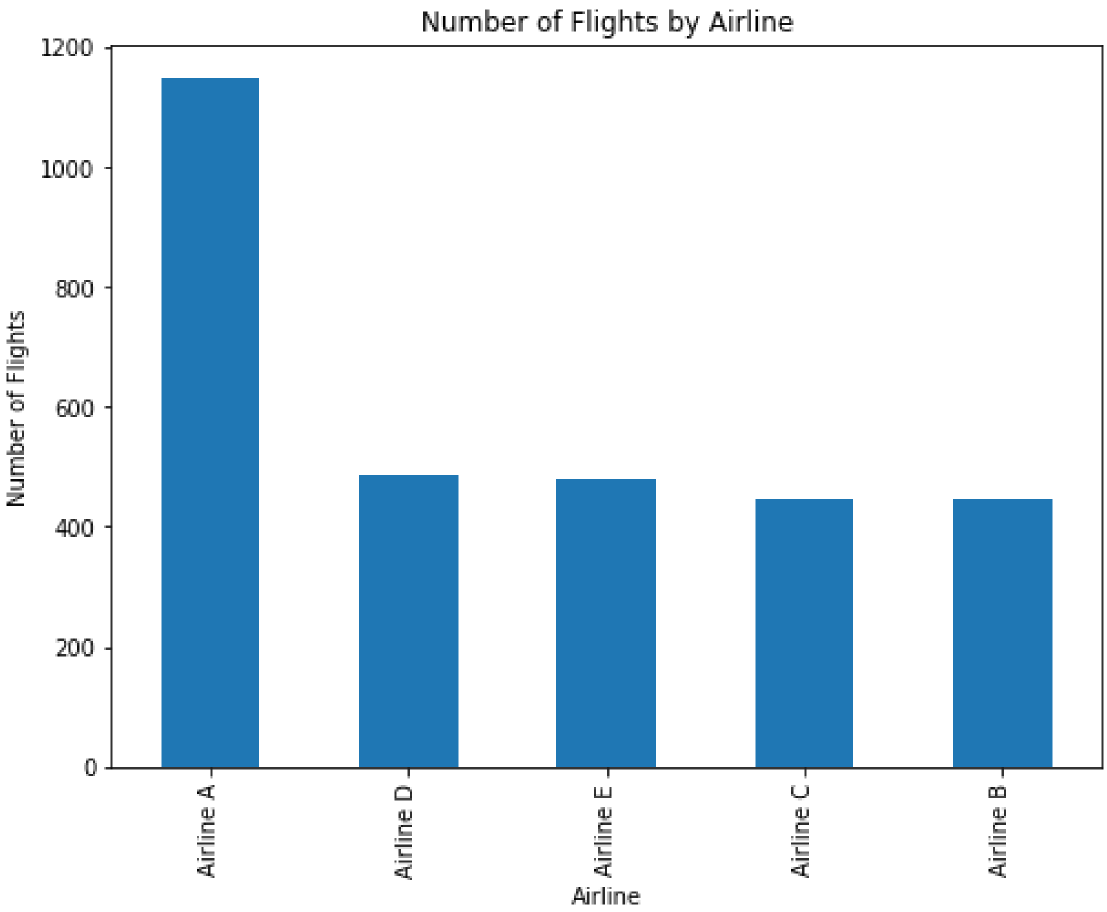
```python
Data.info()
```

```
<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
```

[32]:

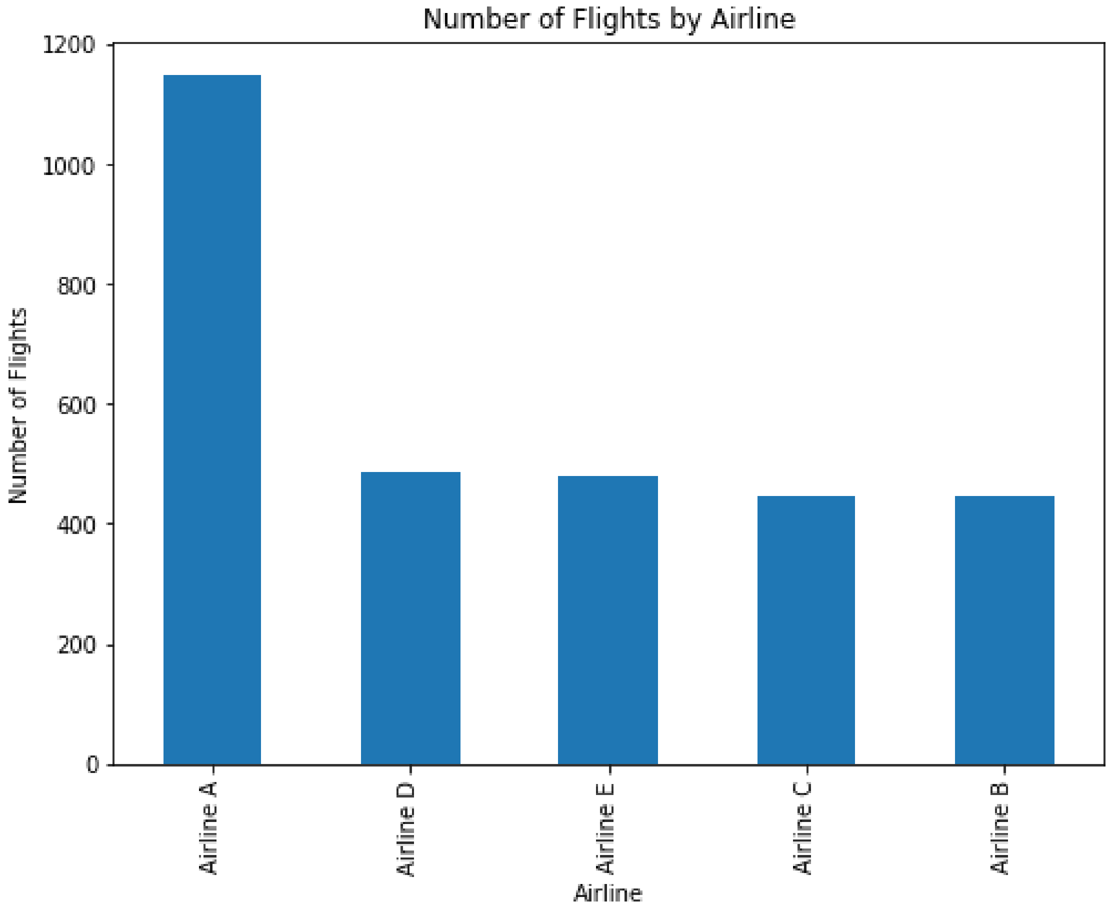
```python
Data.hist(figsize=(12, 10))
plt.tight_layout()
plt.show()
plt.figure(figsize=(8, 6))
Data['Airline'].value_counts().plot(kind='bar')
plt.title('Number of Flights by Airline')
plt.xlabel('Airline')
plt.ylabel('Number of Flights')
plt.show()
print("\n")
```



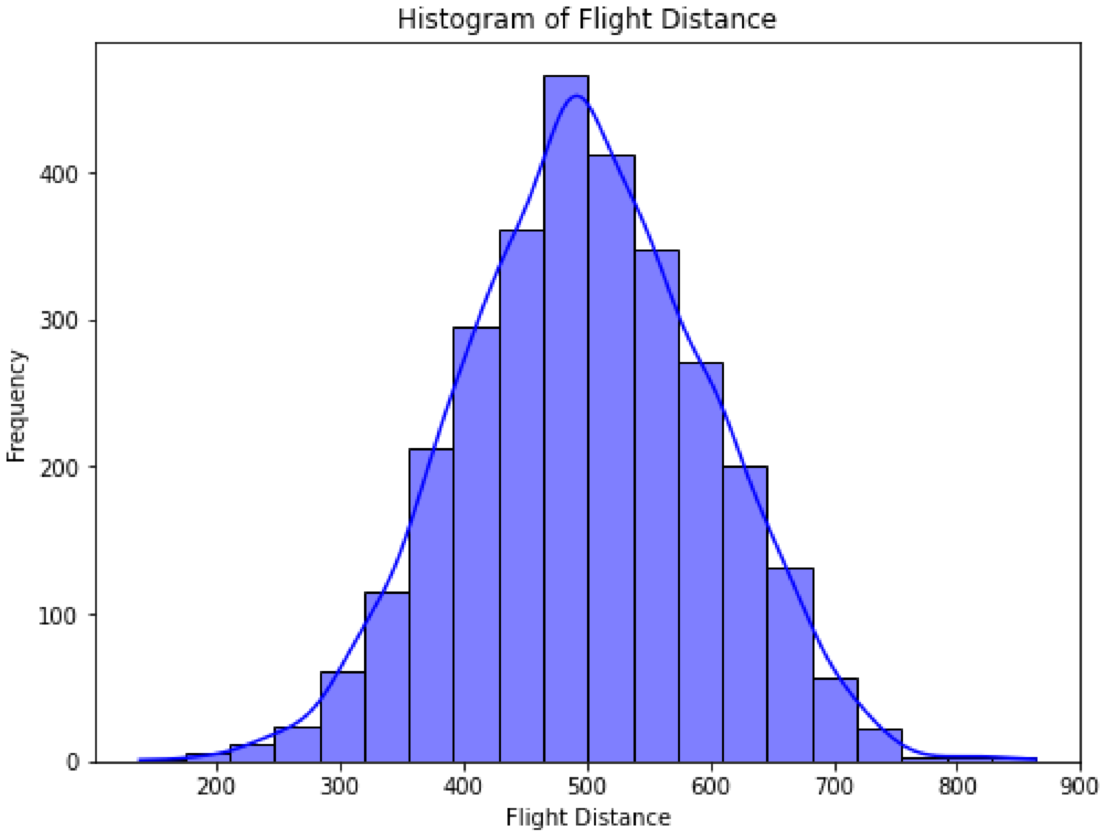


[33]:

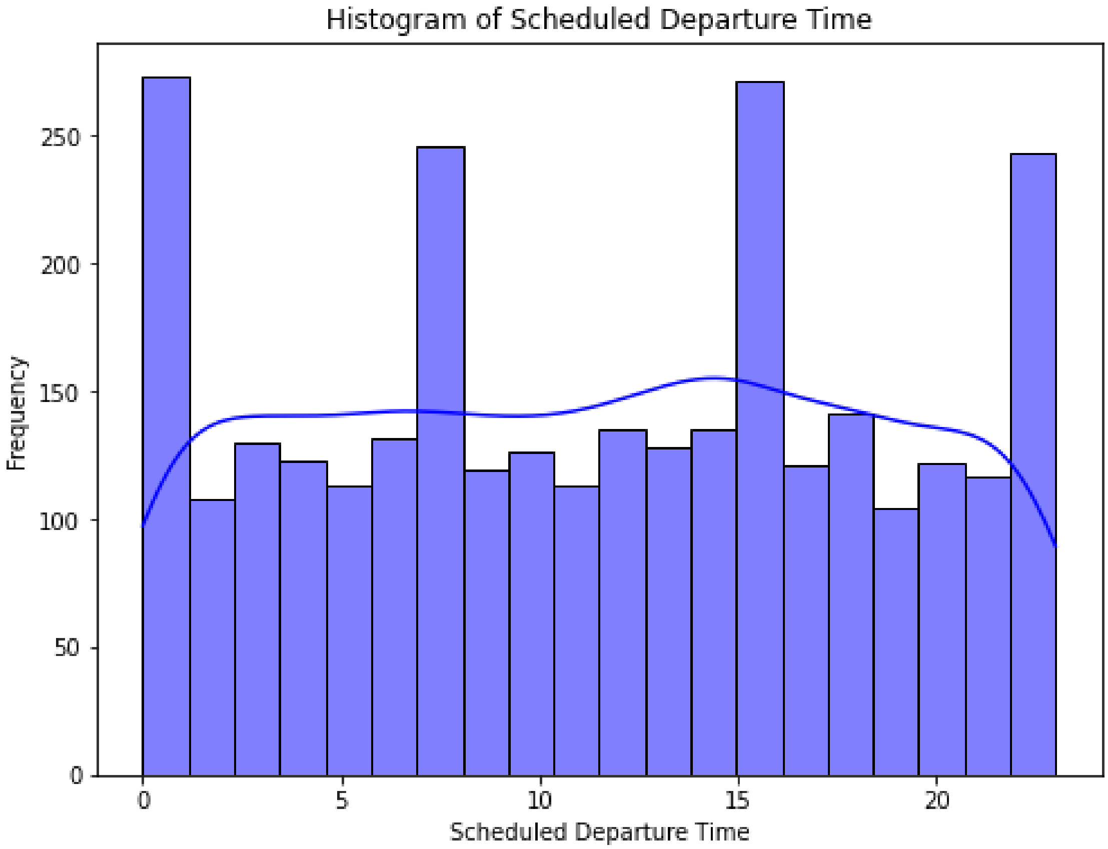
```
plt.figure(figsize=(8, 6))
Data['Airline'].value_counts().plot(kind='bar')
plt.title('Number of Flights by Airline')
plt.xlabel('Airline')
plt.ylabel('Number of Flights')
plt.show()
```



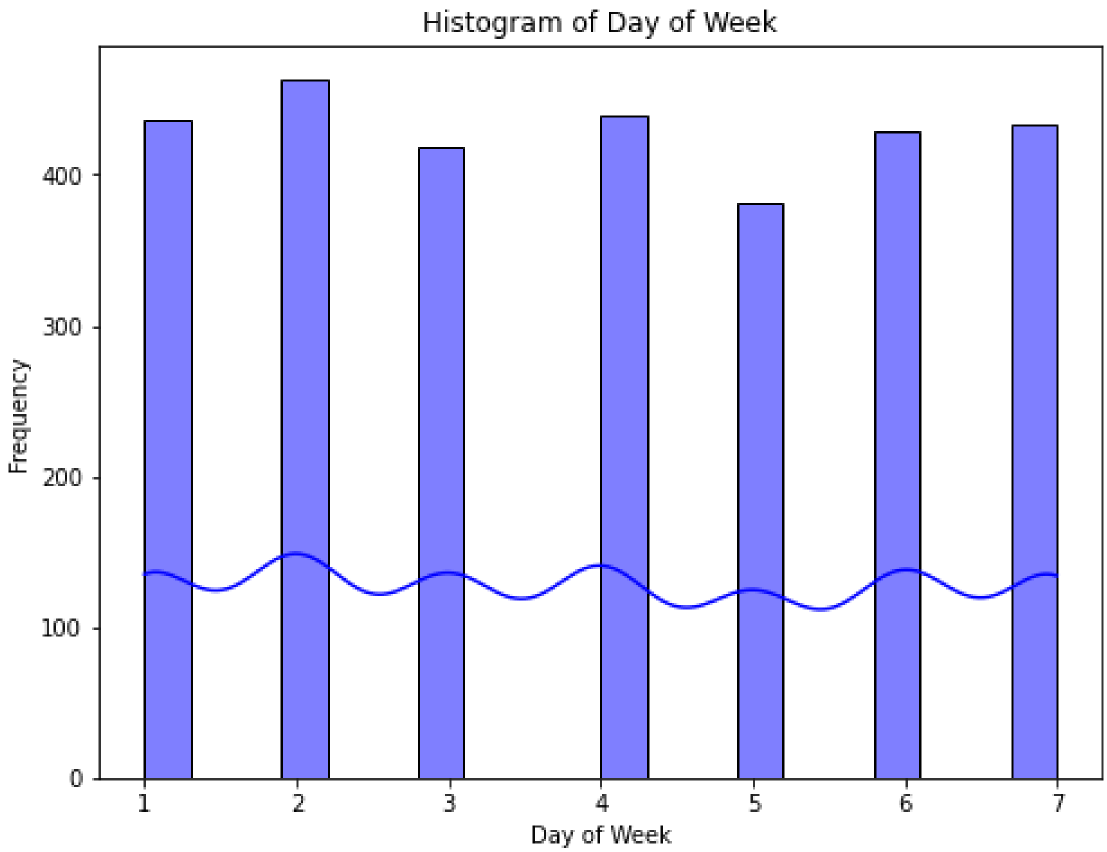
[34]:
```
plt.figure(figsize=(8, 6))
sns.histplot(Data['Flight_Distance'], kde=True, bins=20, color='blue')
plt.title('Histogram of Flight Distance')
plt.xlabel('Flight Distance')
plt.ylabel('Frequency')
plt.show()
```



[35]:
```
plt.figure(figsize=(8, 6))
sns.histplot(Data['Scheduled_Departure_Time'], kde=True, bins=20, color='blue')
plt.title('Histogram of Scheduled Departure Time')
plt.xlabel('Scheduled Departure Time')
plt.ylabel('Frequency')
plt.show()
```

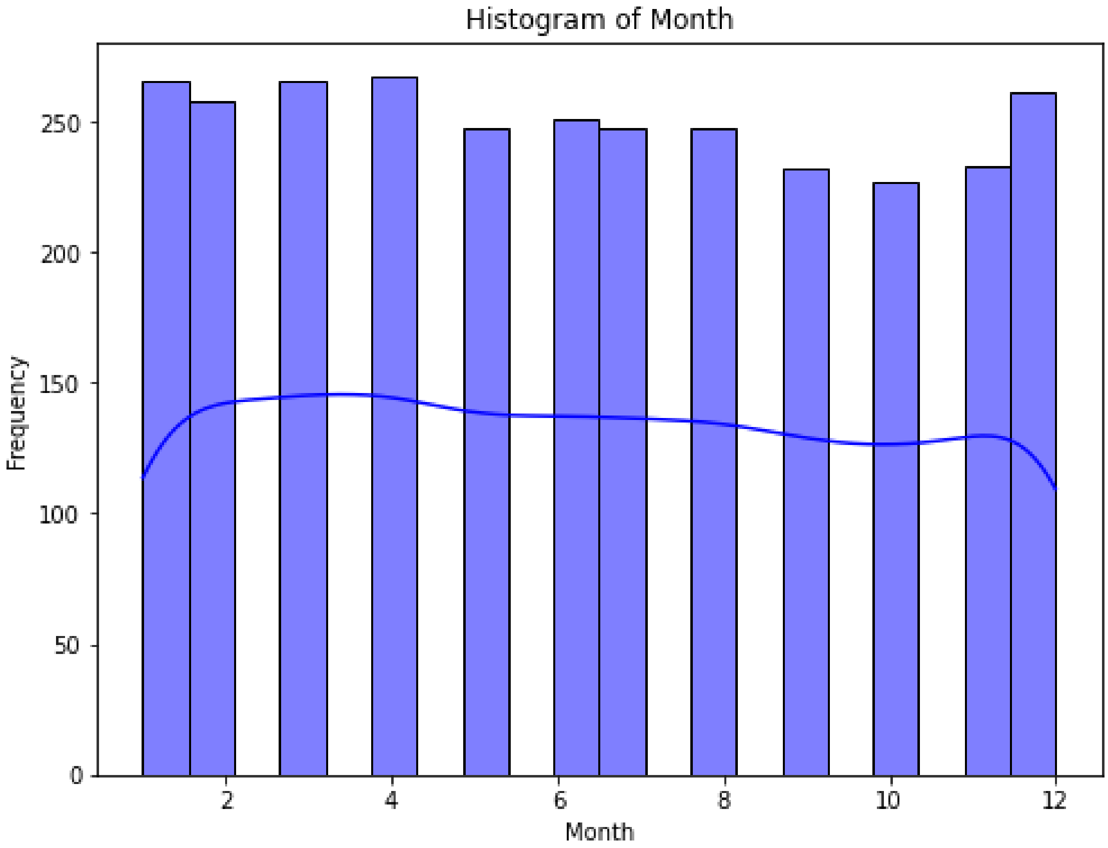


[36]:
```
plt.figure(figsize=(8, 6))
sns.histplot(Data['Day_of_Week'], kde=True, bins=20, color='blue')
plt.title('Histogram of Day of Week')
plt.xlabel('Day of Week')
plt.ylabel('Frequency')
plt.show()
```



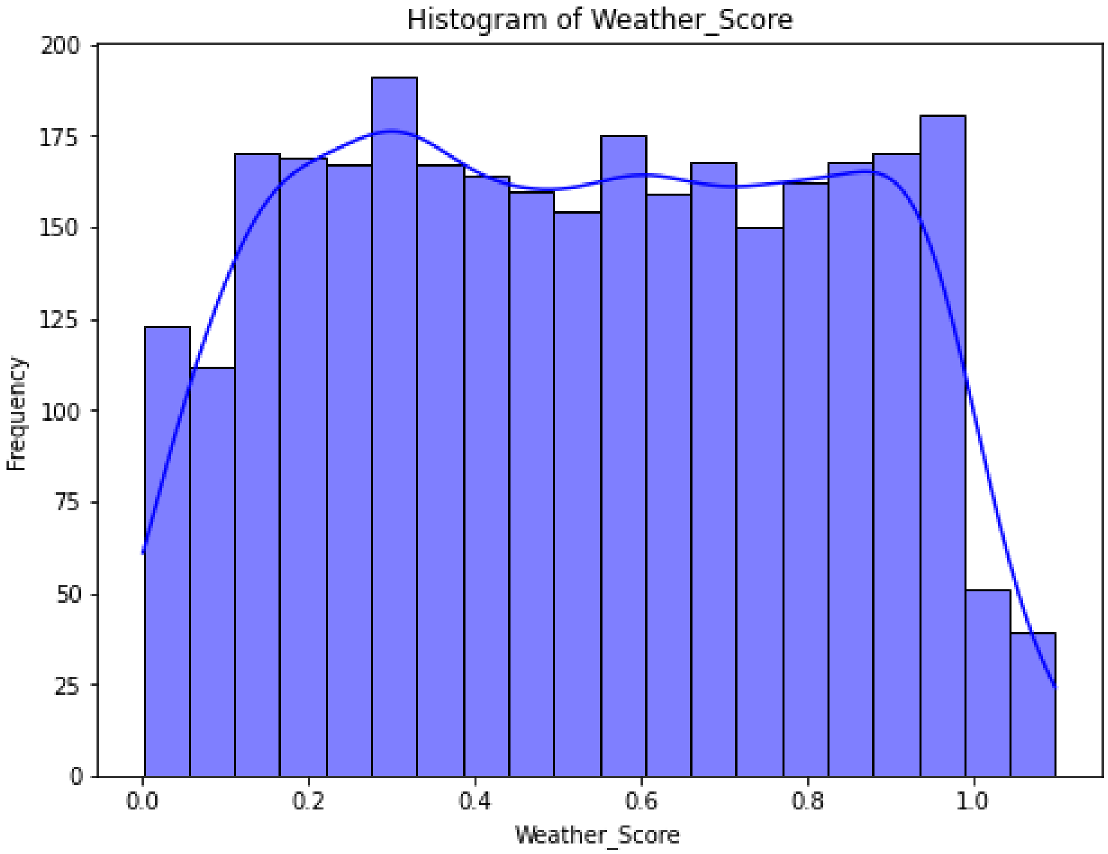
[37]:

```
plt.figure(figsize=(8, 6))
sns.histplot(Data['Month'], kde=True, bins=20, color='blue')
plt.title('Histogram of Month')
plt.xlabel('Month')
plt.ylabel('Frequency')
plt.show()
```



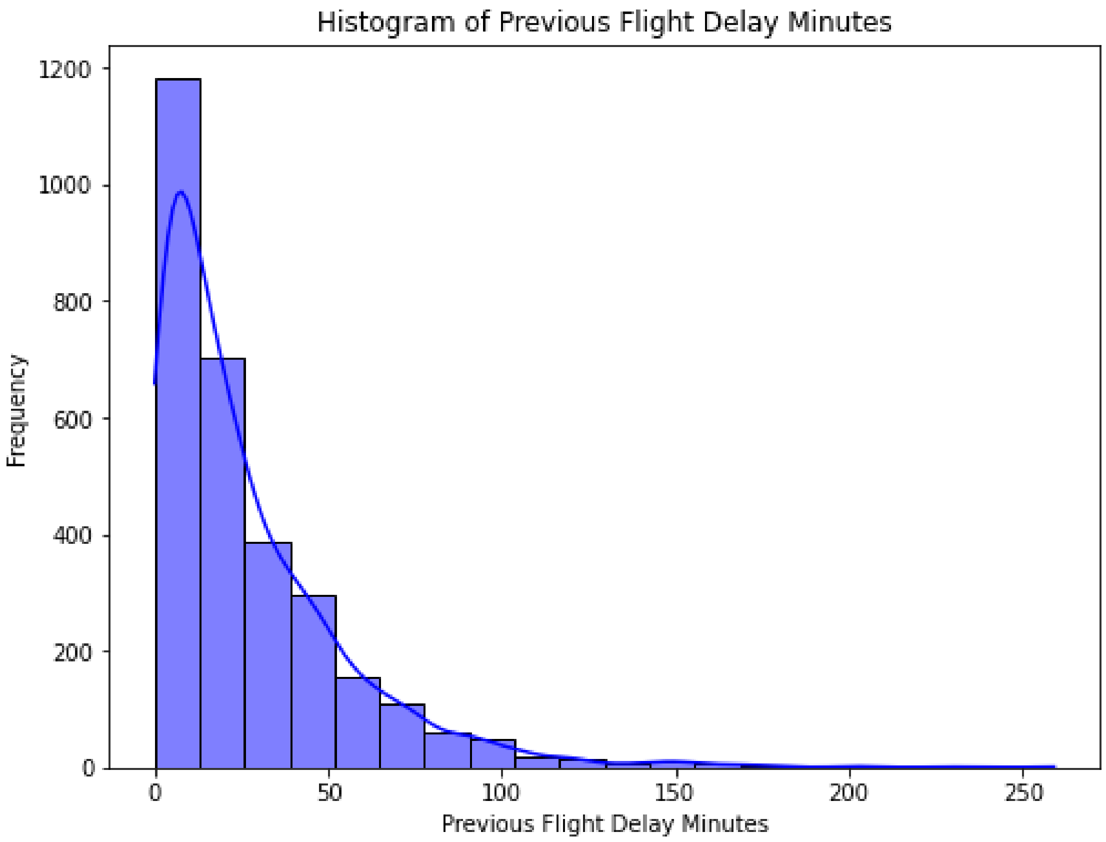
[38]:

```
plt.figure(figsize=(8, 6))
sns.histplot(Data['Weather_Score'], kde=True, bins=20, color='blue')
plt.title('Histogram of Weather_Score')
plt.xlabel('Weather_Score')
plt.ylabel('Frequency')
plt.show()
```



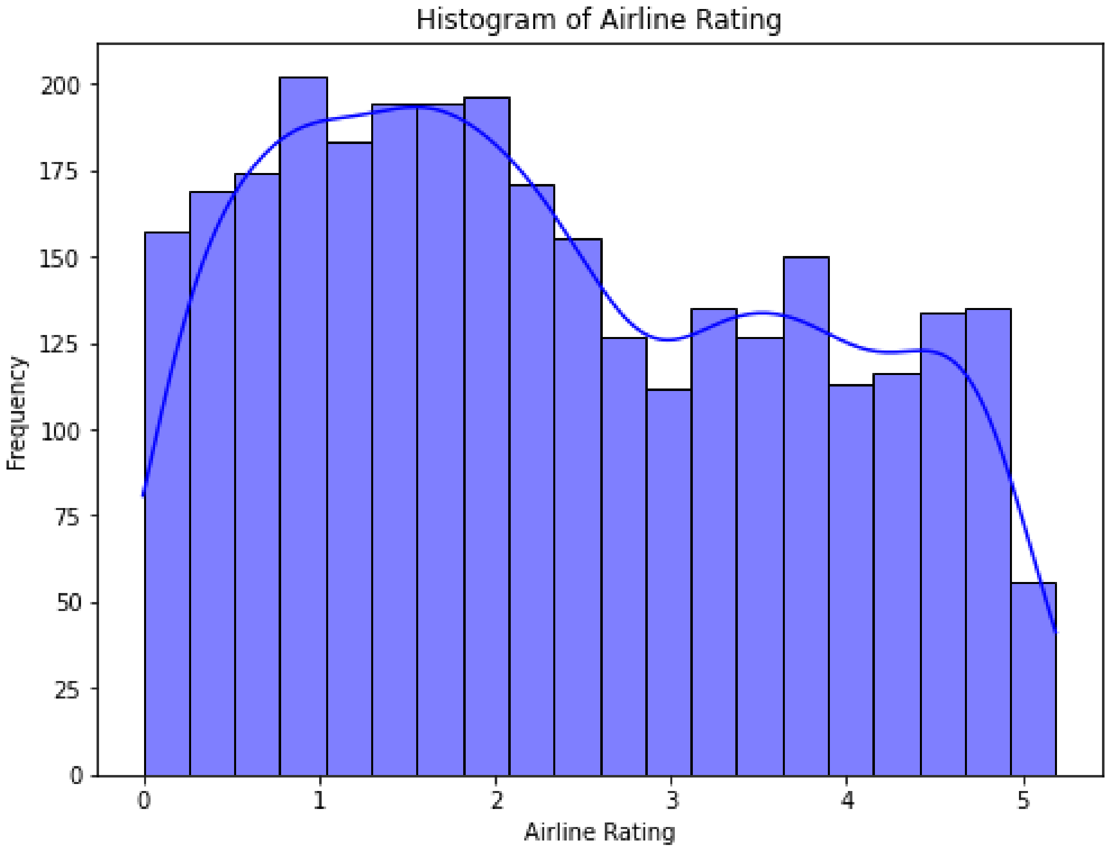
[39]:

```
plt.figure(figsize=(8, 6))
sns.histplot(Data['Previous_Flight_Delay_Minutes'], kde=True, bins=20,␣
  ↪color='blue')
plt.title('Histogram of Previous Flight Delay Minutes')
plt.xlabel('Previous Flight Delay Minutes')
plt.ylabel('Frequency')
plt.show()
```

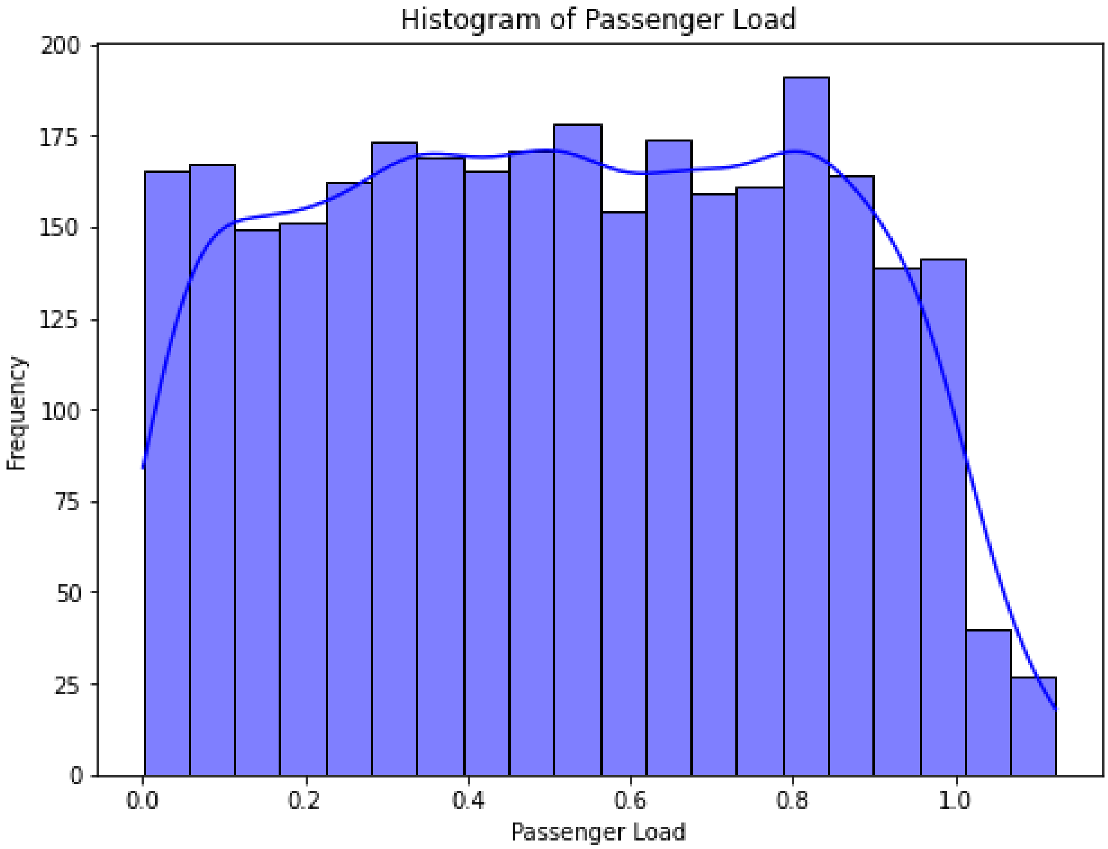


[40]:

```
plt.figure(figsize=(8, 6))
sns.histplot(Data['Airline_Rating'], kde=True, bins=20, color='blue')
plt.title('Histogram of Airline Rating')
plt.xlabel('Airline Rating')
plt.ylabel('Frequency')
plt.show()
```

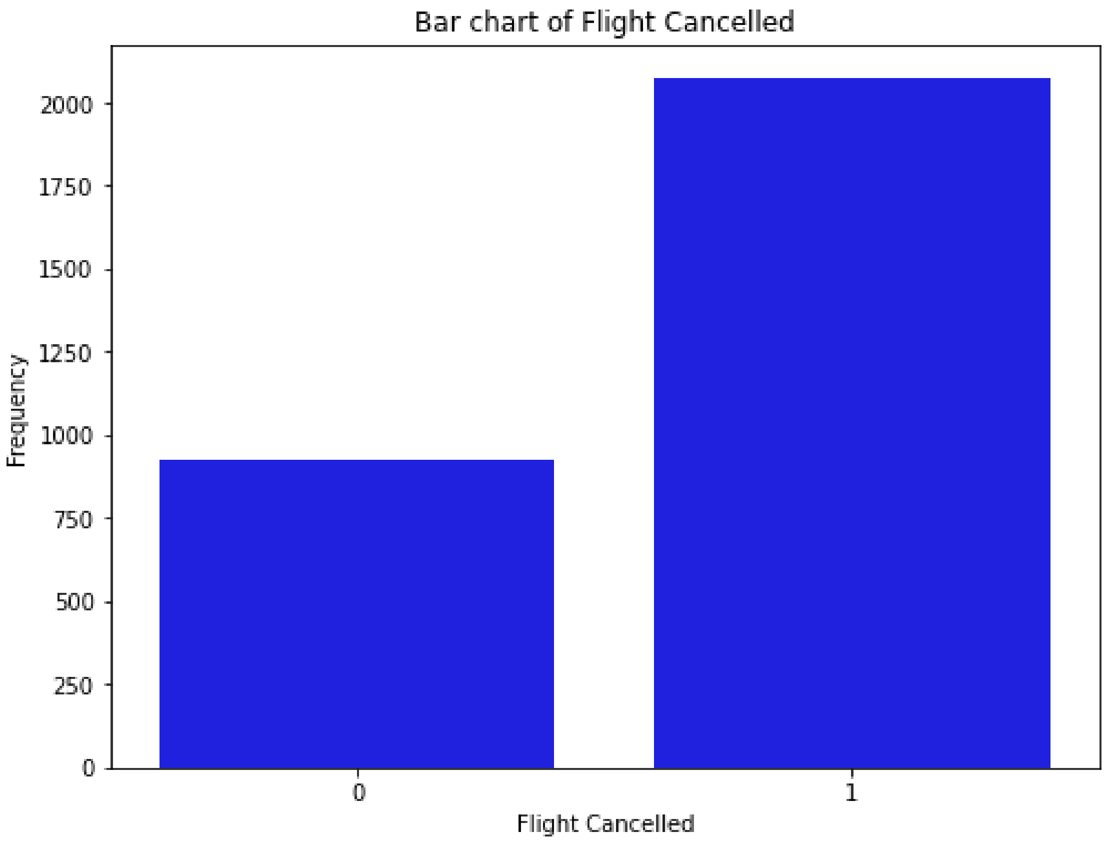


[41]:
```
plt.figure(figsize=(8, 6))
sns.histplot(Data['Passenger_Load'], kde=True, bins=20, color='blue')
plt.title('Histogram of Passenger Load')
plt.xlabel('Passenger Load')
plt.ylabel('Frequency')
plt.show()
```



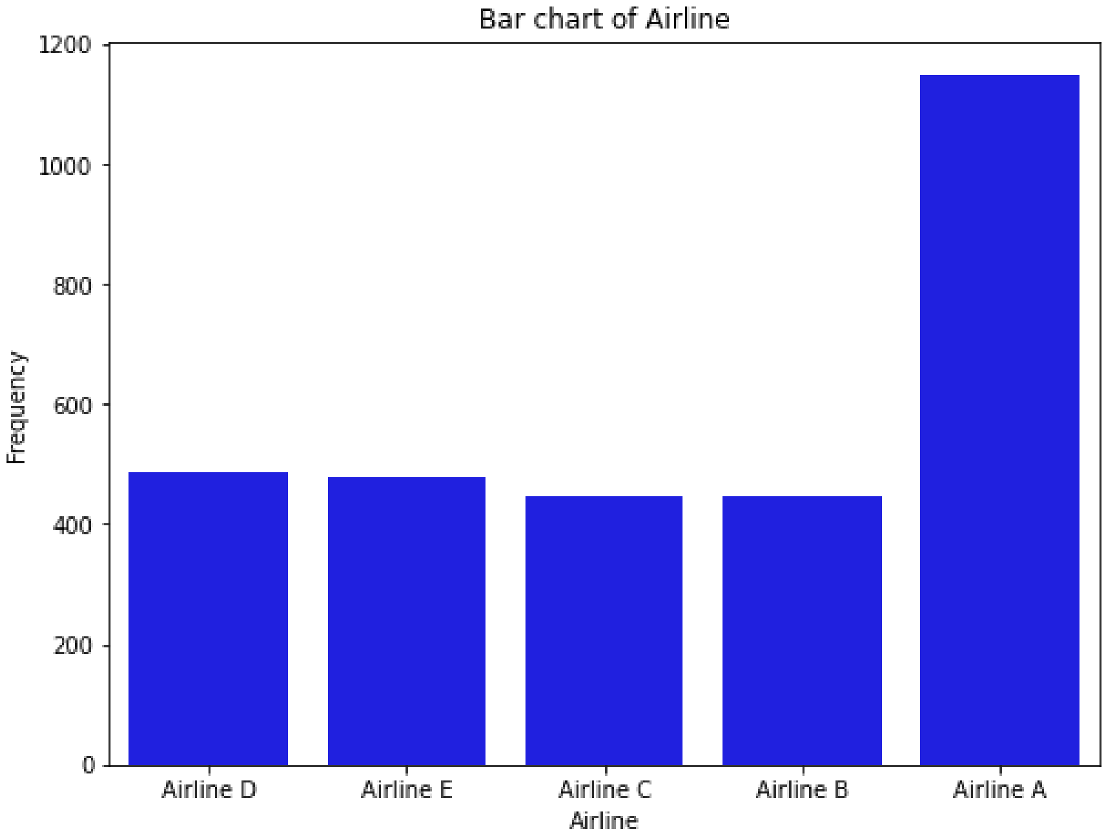
[42]:

```
plt.figure(figsize=(8, 6))
sns.countplot(data=Data, x= 'Flight_Cancelled', color='blue')
plt.title('Bar chart of Flight Cancelled')
plt.xlabel('Flight Cancelled')
plt.ylabel('Frequency')
plt.show()
```

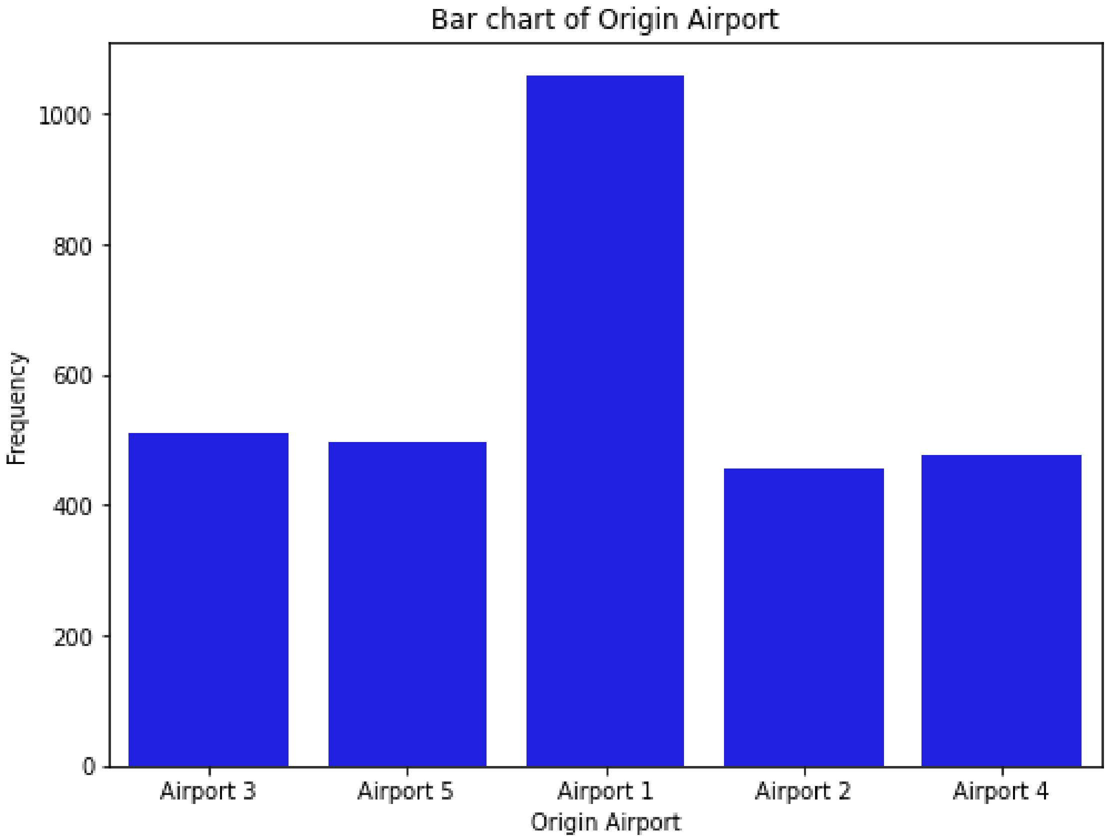


[43]:

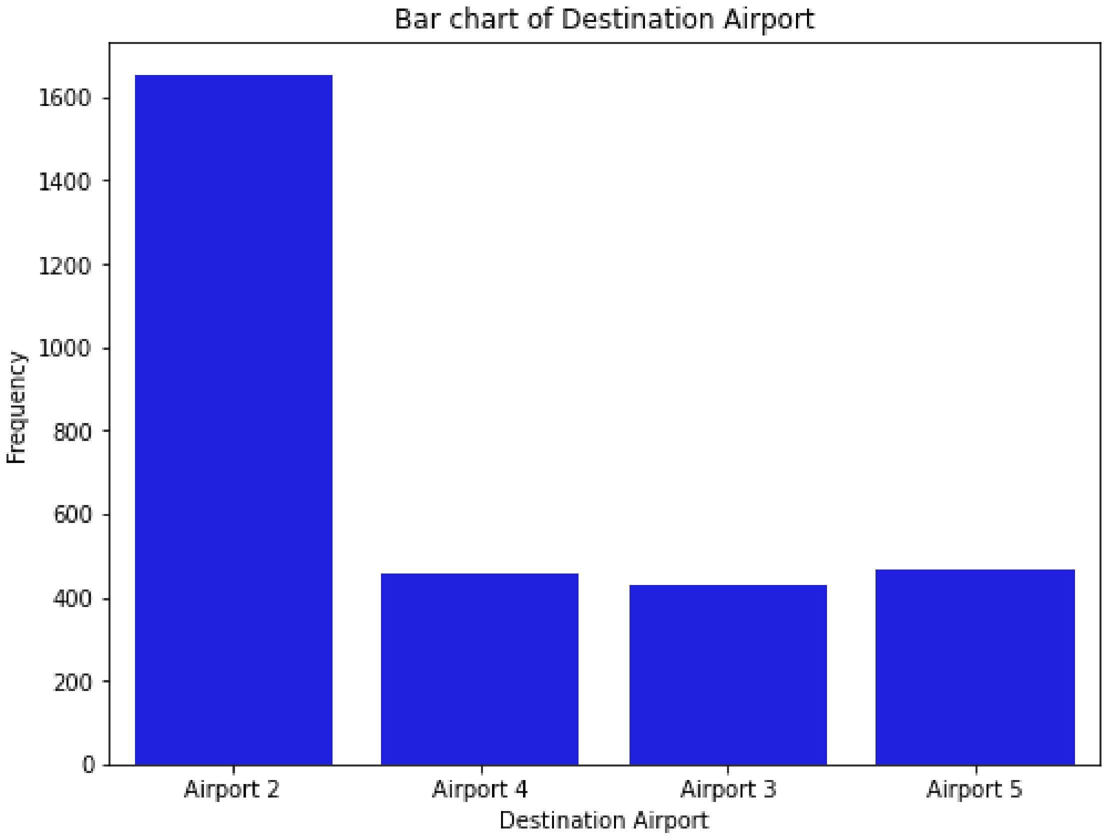
```
plt.figure(figsize=(8, 6))
sns.countplot(data=Data, x= 'Airline', color='blue')
plt.title('Bar chart of Airline')
plt.xlabel('Airline')
plt.ylabel('Frequency')
plt.show()
```



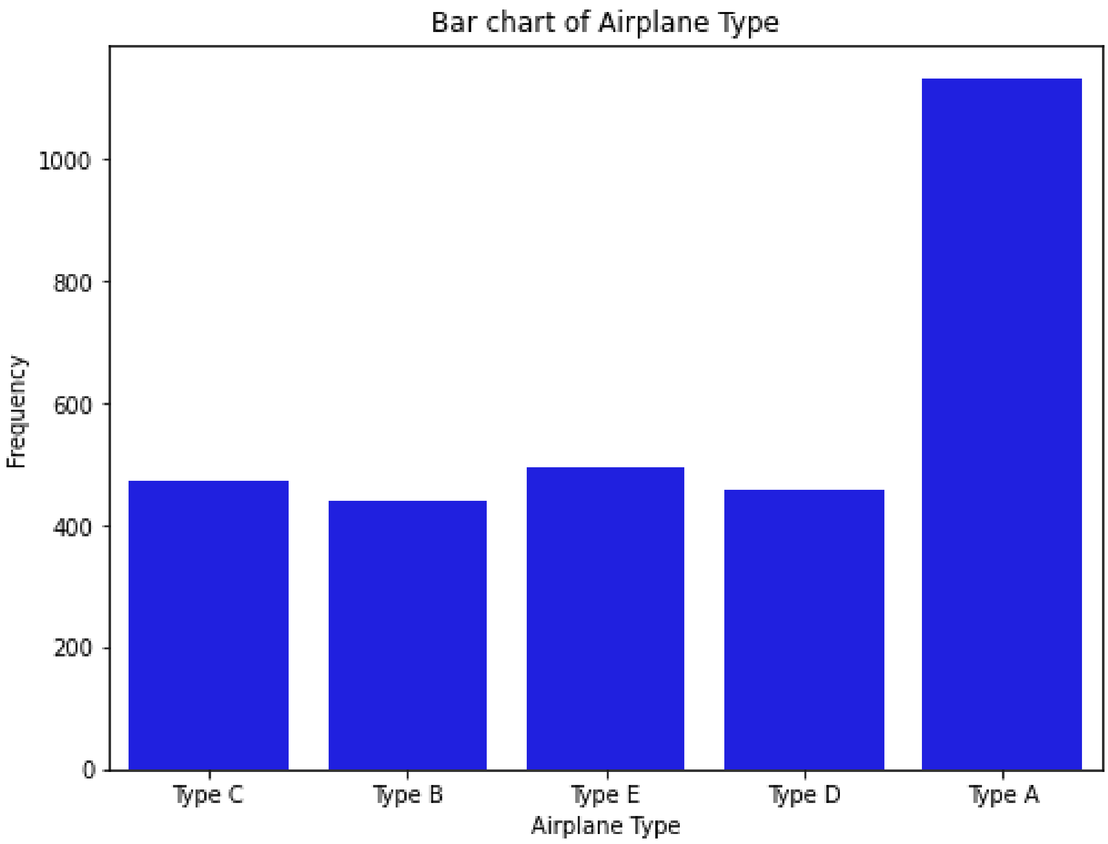
```
[44]: plt.figure(figsize=(8, 6))
      sns.countplot(data=Data, x= 'Origin_Airport', color='blue')
      plt.title('Bar chart of Origin Airport')
      plt.xlabel('Origin Airport')
      plt.ylabel('Frequency')
      plt.show()
```



[45]:
```
plt.figure(figsize=(8, 6))
sns.countplot(data=Data, x= 'Destination_Airport', color='blue')
plt.title('Bar chart of Destination Airport')
plt.xlabel('Destination Airport')
plt.ylabel('Frequency')
plt.show()
```

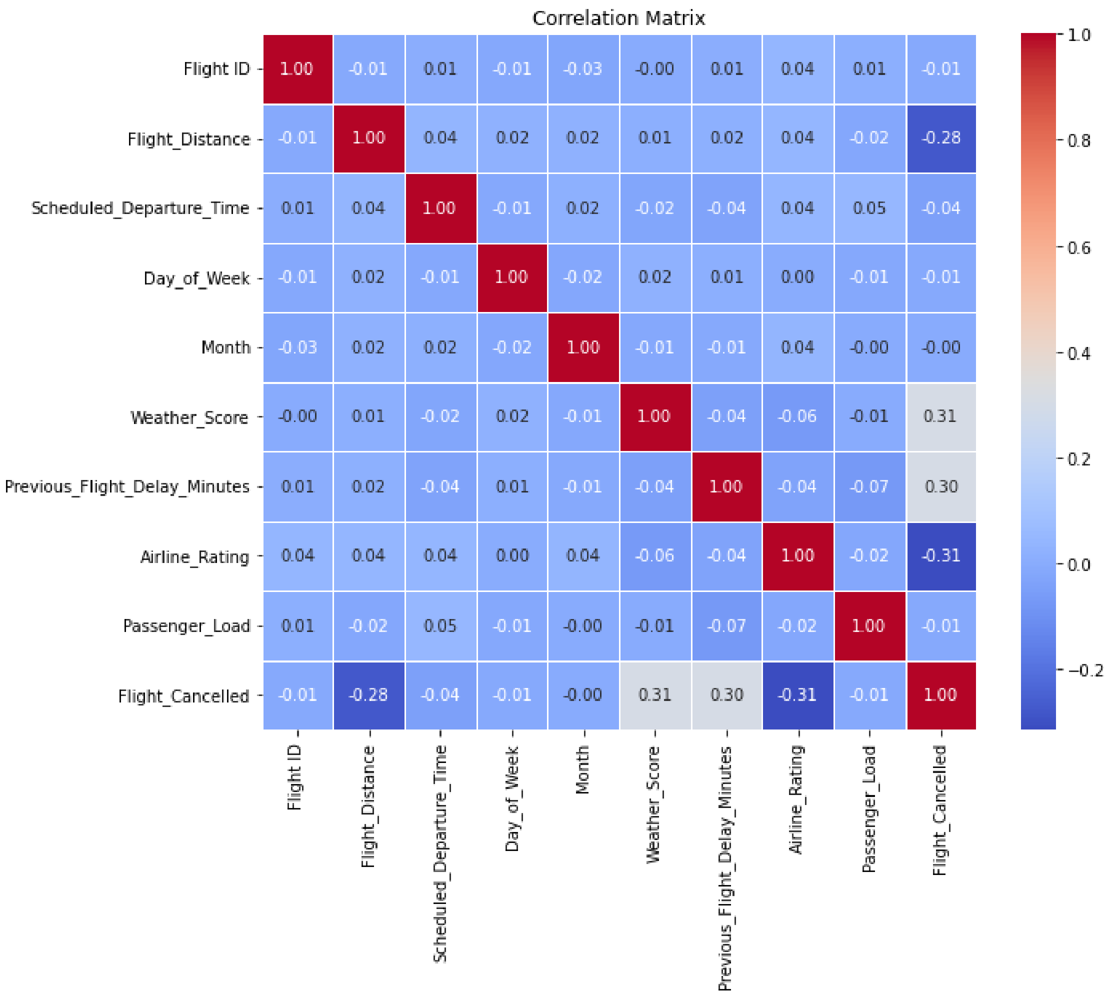


[46]:
```
plt.figure(figsize=(8, 6))
sns.countplot(data=Data, x= 'Airplane_Type', color='blue')
plt.title('Bar chart of Airplane Type')
plt.xlabel('Airplane Type')
plt.ylabel('Frequency')
plt.show()
```



[47]:
```
plt.figure(figsize=(10, 8))
sns.heatmap(Data.corr(), annot=True, cmap= 'coolwarm', fmt=".2f", linewidths=0.
  ↪5)
plt.title('Correlation Matrix')
plt.show()
print("\n")
```

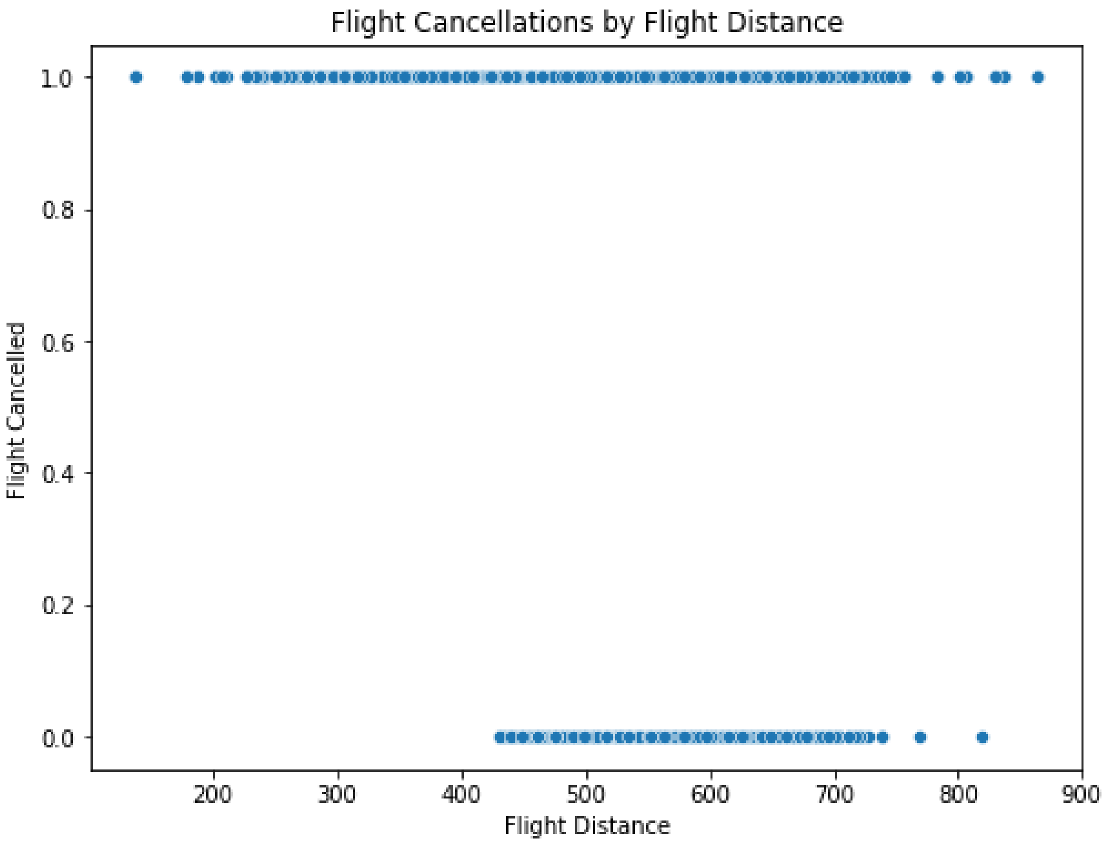
```
/tmp/ipykernel_91/3230157905.py:2: FutureWarning: The default value of
numeric_only in DataFrame.corr is deprecated. In a future version, it will
default to False. Select only valid columns or specify the value of numeric_only
to silence this warning.
  sns.heatmap(Data.corr(), annot=True, cmap= 'coolwarm', fmt=".2f",
linewidths=0.5)
```



	Flight ID	Flight_Distance	Scheduled_Departure_Time	Day_of_Week	Month	Weather_Score	Previous_Flight_Delay_Minutes	Airline_Rating	Passenger_Load	Flight_Cancelled
Flight ID	1.00	-0.01	0.01	-0.01	-0.03	-0.00	0.01	0.04	0.01	-0.01
Flight_Distance	-0.01	1.00	0.04	0.02	0.02	0.01	0.02	0.04	-0.02	-0.28
Scheduled_Departure_Time	0.01	0.04	1.00	-0.01	0.02	-0.02	-0.04	0.04	0.05	-0.04
Day_of_Week	-0.01	0.02	-0.01	1.00	-0.02	0.02	0.01	0.00	-0.01	-0.01
Month	-0.03	0.02	0.02	-0.02	1.00	-0.01	-0.01	0.04	-0.00	-0.00
Weather_Score	-0.00	0.01	-0.02	0.02	-0.01	1.00	-0.04	-0.06	-0.01	0.31
Previous_Flight_Delay_Minutes	0.01	0.02	-0.04	0.01	-0.01	-0.04	1.00	-0.04	-0.07	0.30
Airline_Rating	0.04	0.04	0.04	0.00	0.04	-0.06	-0.04	1.00	-0.02	-0.31
Passenger_Load	0.01	-0.02	0.05	-0.01	-0.00	-0.01	-0.07	-0.02	1.00	-0.01
Flight_Cancelled	-0.01	-0.28	-0.04	-0.01	-0.00	0.31	0.30	-0.31	-0.01	1.00

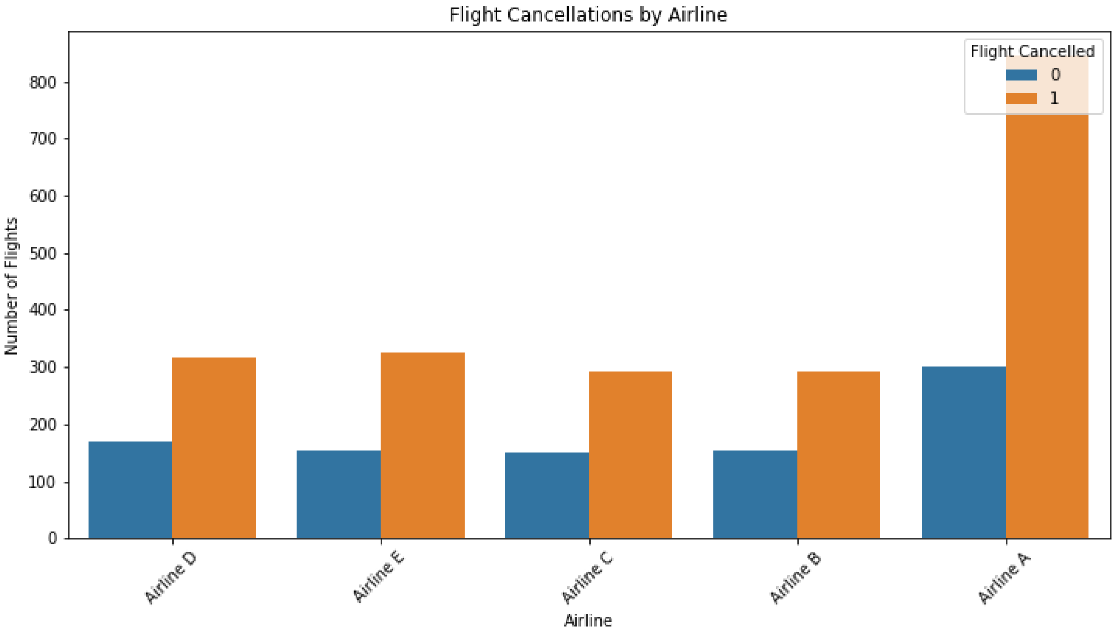
[48]:

```
plt.figure(figsize=(8, 6))
sns.scatterplot(data=Data, x='Flight_Distance', y='Flight_Cancelled')
plt.title('Flight Cancellations by Flight Distance')
plt.xlabel('Flight Distance')
plt.ylabel('Flight Cancelled')
plt.show()
```

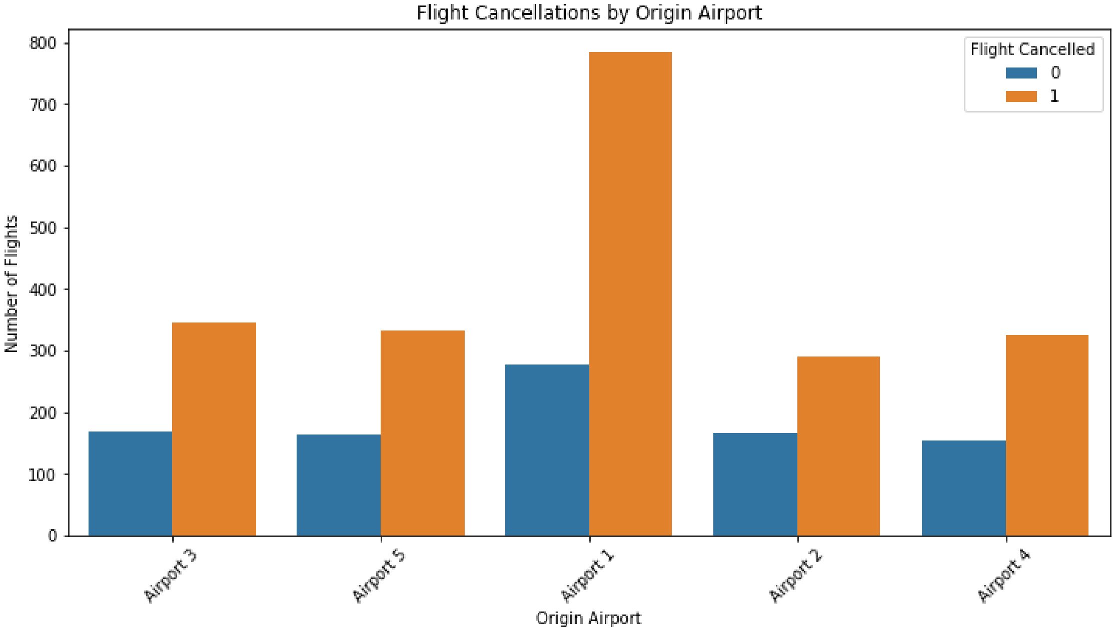


[49]:

```
plt.figure(figsize=(12, 6))
sns.countplot(data=Data, x='Airline', hue='Flight_Cancelled')
plt.title('Flight Cancellations by Airline')
plt.xlabel('Airline')
plt.ylabel('Number of Flights')
plt.xticks(rotation=45)
plt.legend(title='Flight Cancelled', loc='upper right')
plt.show()
```

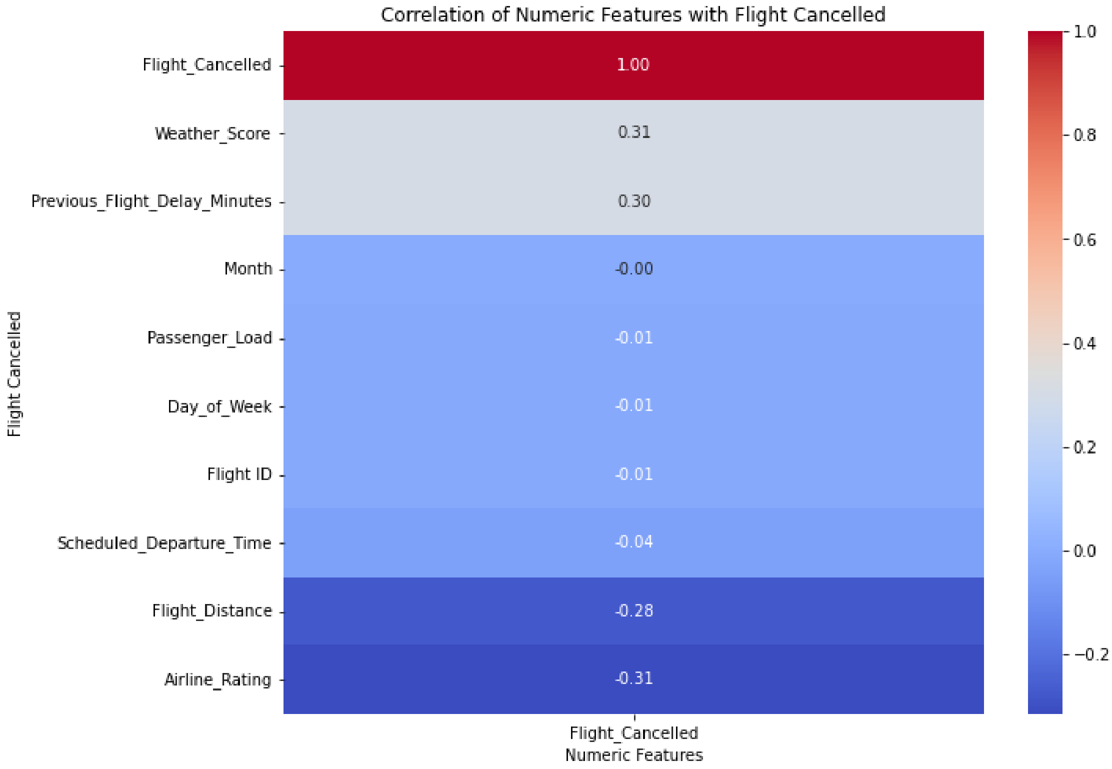


[50]:
```
plt.figure(figsize=(12, 6))
sns.countplot(data=Data, x='Origin_Airport', hue='Flight_Cancelled')
plt.title('Flight Cancellations by Origin Airport')
plt.xlabel('Origin Airport')
plt.ylabel('Number of Flights')
plt.xticks(rotation=45)
plt.legend(title='Flight Cancelled', loc='upper right')
plt.show()
```



[51]:

```
numeric_data = Data.select_dtypes(include='number')
correlation_with_target = numeric_data.corr()['Flight_Cancelled'].
  ↪sort_values(ascending=False)
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_with_target.to_frame(), annot=True, cmap='coolwarm',␣
  ↪fmt=".2f")
plt.title('Correlation of Numeric Features with Flight Cancelled')
plt.xlabel('Numeric Features')
plt.ylabel('Flight Cancelled')
plt.show()
```



Preprocessing and Model Building

[52]:

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import OneHotEncoder, MinMaxScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, precision_score, recall_score,␣
  ↪f1_score, roc_auc_score
```

```
[53]: X = Data.drop(['Flight ID', 'Flight_Cancelled'], axis=1)  # Features
      y = Data['Flight_Cancelled']
```

```
[59]: x_encoded = pd.get_dummies(X)
```

```
[61]: print(x_encoded.head())
```

```
   Flight_Distance  Scheduled_Departure_Time  Day_of_Week  Month  \
0              475                         4            6      1
1              538                        12            1      6
2              565                        17            3      9
3              658                         1            1      8
4              566                        19            7     12

   Weather_Score  Previous_Flight_Delay_Minutes  Airline_Rating  \
0       0.225122                            5.0        2.151974
1       0.060346                           68.0        1.600779
2       0.093920                           18.0        4.406848
3       0.656750                           13.0        0.998757
4       0.505211                            4.0        3.806206

   Passenger_Load  Airline_Airline A  Airline_Airline B  …  \
0        0.477202                  0                  0  …
1        0.159718                  0                  0  …
2        0.256803                  0                  0  …
3        0.504077                  0                  0  …
4        0.019638                  0                  0  …

   Origin_Airport_Airport 5  Destination_Airport_Airport 2  \
0                         0                              1
1                         1                              0
2                         0                              1
3                         1                              0
4                         0                              1

   Destination_Airport_Airport 3  Destination_Airport_Airport 4  \
0                              0                              0
1                              0                              1
2                              0                              0
3                              1                              0
4                              0                              0

   Destination_Airport_Airport 5  Airplane_Type_Type A  Airplane_Type_Type B  \
0                              0                     0                     0
1                              0                     0                     1
2                              0                     0                     0
3                              0                     0                     1
```

```
4                              0                     0                     0

   Airplane_Type_Type C  Airplane_Type_Type D  Airplane_Type_Type E
0                     1                     0                     0
1                     0                     0                     0
2                     1                     0                     0
3                     0                     0                     0
4                     0                     0                     1

[5 rows x 27 columns]
```

[63]:

```
scaler = MinMaxScaler()
x_scaled = scaler.fit_transform(x_encoded)
```

[64]:

```
print(x_scaled)
```

```
[[0.46418733 0.17391304 0.83333333 … 1.         0.         0.        ]
 [0.55096419 0.52173913 0.         … 0.         0.         0.        ]
 [0.58815427 0.73913043 0.33333333 … 1.         0.         0.        ]
 …
 [0.44490358 0.34782609 0.33333333 … 0.         0.         0.        ]
 [0.44903581 0.2173913  0.66666667 … 0.         0.         1.        ]
 [0.31818182 0.04347826 0.         … 0.         0.         0.        ]]
```

[66]:

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x_scaled, y, test_size=0.2,␣
 ↪random_state=42)
print("Training set - Features:", x_train.shape, "Target:", y_train.shape)
print("Test set - Features:", x_test.shape, "Target:", y_test.shape)
```

```
Training set - Features: (2400, 27) Target: (2400,)
Test set - Features: (600, 27) Target: (600,)
```

[67]:

```
from sklearn.linear_model import LogisticRegression
```

[68]:

```
logreg_model = LogisticRegression(random_state=42)
```

[70]:

```
logreg_model.fit(x_train, y_train)
```

[70]:

```
LogisticRegression(random_state=42)
```

[71]:

```
print("Model Coefficients:", logreg_model.coef_)
```

```
Model Coefficients: [[-5.44813124  0.01029947 -0.08940905  0.16600798
3.36691911  8.39711858
  -2.91703632  0.05163797 -0.02845402 -0.14100312  0.0460956   0.03333066
   0.09012382 -0.02711148 -0.05280328  0.01537221  0.10468177 -0.04004627
  0.05851232 -0.03401603  0.07954138 -0.10394473  0.04275748 -0.21807963
  0.02008184  0.22047041 -0.06513717]]
```

[72]:
```
from sklearn.metrics import accuracy_score, precision_score, recall_score,␣
↪f1_score, roc_auc_score
```

[73]:
```
y_pred = logreg_model.predict(x_test)
```

[74]:
```
from sklearn.metrics import accuracy_score, precision_score, recall_score,␣
↪f1_score, roc_auc_score
```

0.2 Build other Classification Models

[75]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
```

[76]:
```
X = Data.drop(columns=['Scheduled_Departure_Time', 'Flight_Cancelled'])
y = Data['Flight_Cancelled']
numerical_features = ['Flight_Distance', 'Scheduled_Departure_Time',␣
↪'Previous_Flight_Delay_Minutes', 'Airline_Rating', 'Passenger_Load',␣
↪'Weather_Score']
categorical_features = ['Airline', 'Origin_Airport', 'Destination_Airport',␣
↪'Airplane_Type', 'Day_of_Week', 'Month']
preprocessor = ColumnTransformer(
    transformers=[
        ('num', StandardScaler(), numerical_features),
        ('cat', OneHotEncoder(), categorical_features)
    ])
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
↪random_state=42)
```

[77]:
```
numerical_features = ['Flight_Distance', 'Sched_Departure_Time',␣
↪'Previous_Flight_Delay_Minutes', 'Airline_Rating', 'Passenger_Load',␣
↪'Weather_Score']
```

[80]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, roc_auc_score,␣
↪precision_score, recall_score, f1_score
```

[ ]:

```

```

[81]:

```
data = pd.read_csv('Flyzy Flight Cancellation - Sheet1.csv')
data.iloc[1:4, 2:3]= np.NaN
data.iloc[1:4, 3:4]= "NA"
data.iloc[1:4, 4:5]= ""
data["None_col"]= None
data.head()
```

[81]:

```
   Flight ID    Airline  Flight_Distance Origin_Airport Destination_Airport  \
0    7319483  Airline D            475.0      Airport 3           Airport 2
1    4791965  Airline E              NaN             NA
2    2991718  Airline C              NaN             NA
3    4220106  Airline E              NaN             NA
4    2263008  Airline E            566.0      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 None_col
0                 0     None
1                 1     None
2                 0     None
3                 1     None
4                 0     None
```

[83]:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, roc_auc_score,␣
 ↪precision_score, recall_score, f1_score
```

```
Data = pd.read_csv('Flyzy Flight Cancellation - Sheet1.csv')


print(Data.columns)


X = Data.drop(columns=['Flight_Cancelled', 'Flight_Cancelled'])
y = Data['Flight_Cancelled']


numerical_features = ['Flight_Distance', 'Scheduled_Departure_Time',␣
 ↪'Previous_Flight_Delay_Minutes', 'Airline_Rating', 'Passenger_Load',␣
 ↪'Weather_Score']
categorical_features = ['Airline', 'Origin_Airport', 'Destination_Airport',␣
 ↪'Airplane_Type', 'Day_of_Week', 'Month']

preprocessor = ColumnTransformer(
    transformers=[
        ('num', StandardScaler(), numerical_features),
        ('cat', OneHotEncoder(), categorical_features)
    ])


X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)
```

```
Index(['Flight ID', 'Airline', 'Flight_Distance', 'Origin_Airport',
       'Destination_Airport', 'Scheduled_Departure_Time', 'Day_of_Week',
       'Month', 'Airplane_Type', 'Weather_Score',
       'Previous_Flight_Delay_Minutes', 'Airline_Rating', 'Passenger_Load',
       'Flight_Cancelled'],
      dtype='object')
```

[84]:

```
numerical_features = ['Flight_Distance', 'Sched_Departure_Time',␣
 ↪'Previous_Flight_Delay_Minutes', 'Airline_Rating', 'Passenger_Load',␣
 ↪'Weather_Score']
```

[85]:

```
lr_pipeline = Pipeline(steps=[('preprocessor', preprocessor),
                              ('classifier',␣
 ↪LogisticRegression(random_state=42))])


lr_pipeline.fit(X_train, y_train)
```

[85]:

```
Pipeline(steps=[('preprocessor',
                 ColumnTransformer(transformers=[('num', StandardScaler(),
                                                  ['Flight_Distance',
                                                   'Scheduled_Departure_Time',
'Previous_Flight_Delay_Minutes',
                                                   'Airline_Rating',
                                                   'Passenger_Load',
                                                   'Weather_Score']),
                                                 ('cat', OneHotEncoder(),
                                                  ['Airline', 'Origin_Airport',
                                                   'Destination_Airport',
                                                   'Airplane_Type',
                                                   'Day_of_Week', 'Month'])])),
                ('classifier', LogisticRegression(random_state=42))])
```

[86]:

```
from sklearn.linear_model import LogisticRegression

# Build pipeline
lr_pipeline = Pipeline(steps=[('preprocessor', preprocessor),
                              ('classifier', LogisticRegression())])

# Train and evaluate
lr_pipeline.fit(X_train, y_train)
y_pred = lr_pipeline.predict(X_test)

# Evaluation
print("Logistic Regression Classification Report:")
print(classification_report(y_test, y_pred))
print("ROC-AUC Score:", roc_auc_score(y_test, lr_pipeline.
 ↪predict_proba(X_test)[:, 1]))
```

```
Logistic Regression Classification Report:
              precision    recall  f1-score   support

           0       0.70      0.59      0.64       187
           1       0.83      0.89      0.86       413

    accuracy                           0.79       600
   macro avg       0.76      0.74      0.75       600
weighted avg       0.79      0.79      0.79       600

ROC-AUC Score: 0.8652613587808006
```

[89]:

```
from sklearn.tree import DecisionTreeClassifier


dt_pipeline = Pipeline(steps=[('preprocessor', preprocessor),
                              ('classifier',␣
 ↪DecisionTreeClassifier(random_state=42))])

```

```
dt_pipeline.fit(X_train, y_train)
y_pred = dt_pipeline.predict(X_test)

print("Decision Tree Classification Report:")
print(classification_report(y_test, y_pred))
print("ROC-AUC Score:", roc_auc_score(y_test, dt_pipeline.
 ↪predict_proba(X_test)[:, 1]))
```

```
Decision Tree Classification Report:
              precision    recall  f1-score   support

           0       0.97      0.94      0.95       187
           1       0.97      0.99      0.98       413

    accuracy                           0.97       600
   macro avg       0.97      0.96      0.96       600
weighted avg       0.97      0.97      0.97       600

ROC-AUC Score: 0.9606505159845139
```

[90]:

```
from sklearn.ensemble import GradientBoostingClassifier


gb_pipeline = Pipeline(steps=[('preprocessor', preprocessor),
                              ('classifier',␣
 ↪GradientBoostingClassifier(random_state=42))])


gb_pipeline.fit(X_train, y_train)
y_pred = gb_pipeline.predict(X_test)


print("Gradient Boosting Classification Report:")
print(classification_report(y_test, y_pred))
print("ROC-AUC Score:", roc_auc_score(y_test, gb_pipeline.
 ↪predict_proba(X_test)[:, 1]))
```

```
Gradient Boosting Classification Report:
              precision    recall  f1-score   support

           0       0.96      1.00      0.98       187
           1       1.00      0.98      0.99       413

    accuracy                           0.99       600
   macro avg       0.98      0.99      0.98       600
weighted avg       0.99      0.99      0.99       600
```

```
ROC-AUC Score: 0.9917779130142041
```

[98]:

```
from sklearn.ensemble import GradientBoostingClassifier


gb_pipeline = Pipeline(steps=[('preprocessor', preprocessor),
                              ('classifier',␣
 ↪GradientBoostingClassifier(random_state=42))])


gb_pipeline.fit(X_train, y_train)
y_pred = gb_pipeline.predict(X_test)


print("Gradient Boosting Classification Report:")
print(classification_report(y_test, y_pred))
print("ROC-AUC Score:", roc_auc_score(y_test, gb_pipeline.
 ↪predict_proba(X_test)[:, 1]))
```

```
Gradient Boosting Classification Report:
              precision    recall  f1-score   support

           0       0.96      1.00      0.98       187
           1       1.00      0.98      0.99       413

    accuracy                           0.99       600
   macro avg       0.98      0.99      0.98       600
weighted avg       0.99      0.99      0.99       600

ROC-AUC Score: 0.9917779130142041
```

[1]:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
from sklearn.svm import SVC
from sklearn.metrics import precision_score, recall_score, f1_score,␣
 ↪roc_auc_score, classification_report
import matplotlib.pyplot as plt


Data = pd.read_csv('Flyzy Flight Cancellation - Sheet1.csv')
```

```
X = Data.drop(columns=['Flight_Cancelled', 'Flight_Cancelled'])
y = Data['Flight_Cancelled']


numerical_features = ['Flight_Distance', 'Scheduled_Departure_Time',␣
 ↪'Previous_Flight_Delay_Minutes', 'Airline_Rating', 'Passenger_Load',␣
 ↪'Weather_Score']
categorical_features = ['Airline', 'Origin_Airport', 'Destination_Airport',␣
 ↪'Airplane_Type', 'Day_of_Week', 'Month']

preprocessor = ColumnTransformer(
    transformers=[
        ('num', StandardScaler(), numerical_features),
        ('cat', OneHotEncoder(), categorical_features)
    ])


X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)


models = {
    "Logistic Regression": LogisticRegression(random_state=42),
    "Decision Tree": DecisionTreeClassifier(random_state=42),
    "Random Forest": RandomForestClassifier(random_state=42),
    "Gradient Boosting": GradientBoostingClassifier(random_state=42),
    "SVM": SVC(probability=True, random_state=42)
}


results = {}

for model_name, model in models.items():
    pipeline = Pipeline(steps=[('preprocessor', preprocessor),
                               ('classifier', model)])
    pipeline.fit(X_train, y_train)
    y_pred = pipeline.predict(X_test)
    y_proba = pipeline.predict_proba(X_test)[:, 1] if hasattr(model,␣
 ↪"predict_proba") else None
    results[model_name] = {
        "Precision": precision_score(y_test, y_pred),
        "Recall": recall_score(y_test, y_pred),
        "F1 Score": f1_score(y_test, y_pred),
        "ROC-AUC": roc_auc_score(y_test, y_proba) if y_proba is not None else␣
 ↪None
```

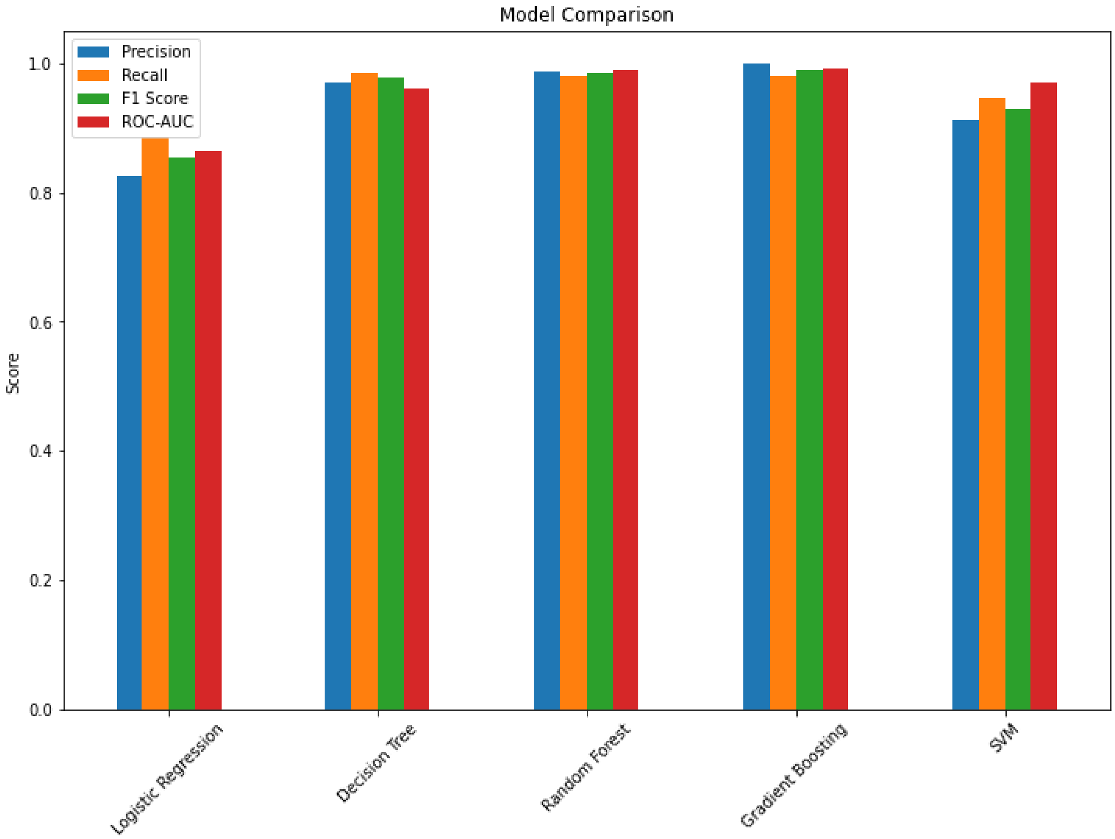
```
    }

results_df = pd.DataFrame(results).T
print(results_df)
```

```
                     Precision    Recall  F1 Score   ROC-AUC
Logistic Regression   0.826185  0.886199  0.855140  0.865261
Decision Tree         0.971360  0.985472  0.978365  0.960651
Random Forest         0.987805  0.980630  0.984204  0.990703
Gradient Boosting     1.000000  0.980630  0.990220  0.991778
SVM                   0.913551  0.946731  0.929845  0.970996
```

[2]:

```
results_df.plot(kind='bar', figsize=(12, 8))
plt.title('Model Comparison')
plt.ylabel('Score')
plt.xticks(rotation=45)
plt.show()
```



[ ]:

```
```