

```
In [1]: import pandas as pd
```

```
# Load the uploaded dataset
file_path = "Flyzy Flight Cancellation - Sheet1.csv"
df = pd.read_csv(file_path)

# Display the first few rows and summary info
df.head(), df.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: 328.3+ KB
```

```
Out[1]: (
 0   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 ,
None)
```

```
In [2]: import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from scipy.stats import zscore

# Select numeric columns for outlier detection
numeric_cols = df.select_dtypes(include=[np.number]).columns.tolist()
numeric_cols.remove("Flight ID") # Exclude ID column

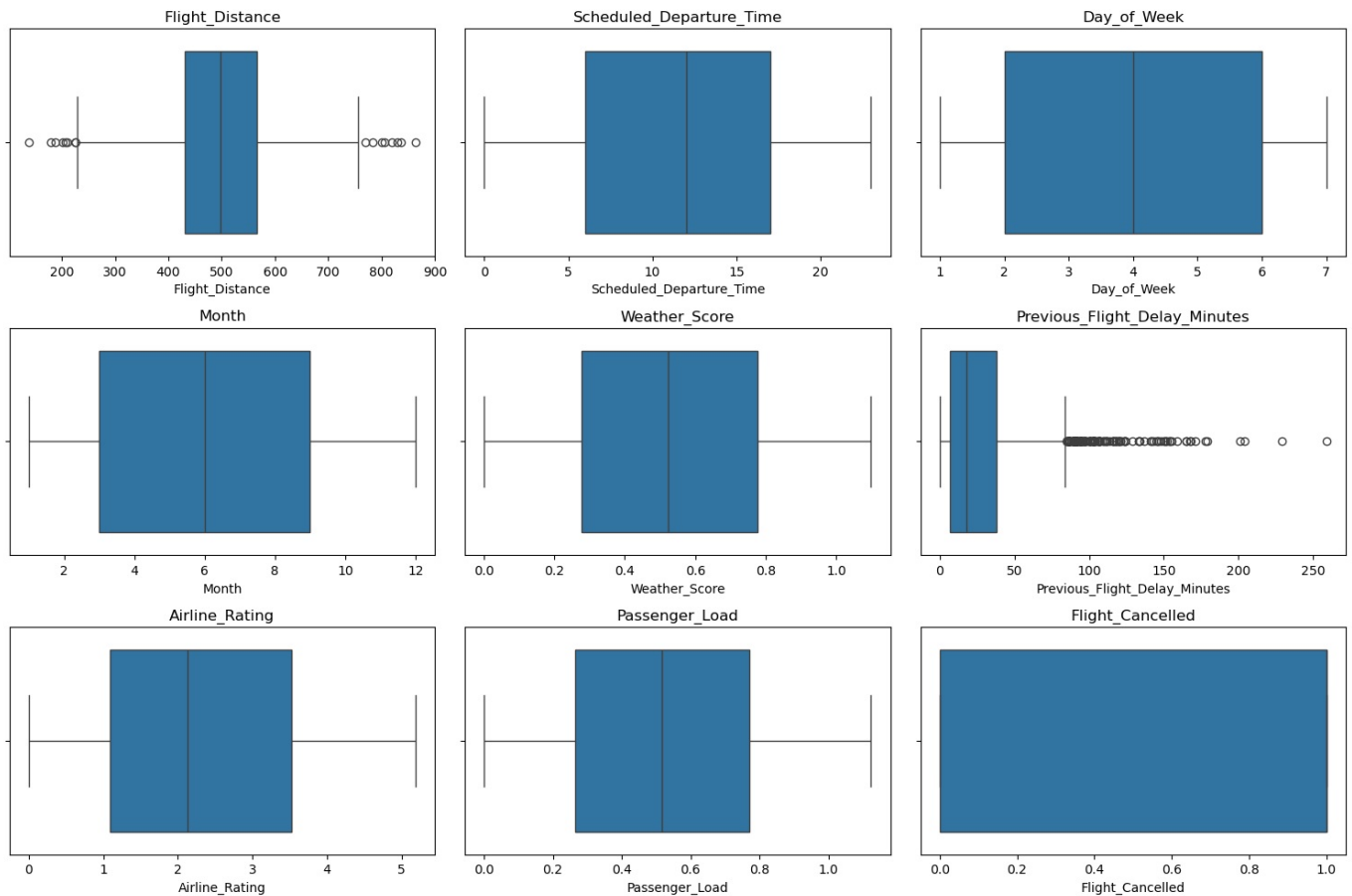
# Compute Z-scores
z_scores = np.abs(zscore(df[numeric_cols]))
outliers = (z_scores > 3)

# Count outliers per column
outlier_counts = outliers.sum(axis=0)

# Visualize outliers with boxplots
plt.figure(figsize=(15, 10))
for i, col in enumerate(numeric_cols, 1):
    plt.subplot(3, 3, i)
    sns.boxplot(x=df[col])
    plt.title(col)
```

```
plt.tight_layout()
plt.show()
```

outlier\_counts



```
Out[2]: Flight Distance          10
Scheduled_Departure_Time      0
Day_of_Week                   0
Month                         0
Weather_Score                 0
Previous_Flight_Delay_Minutes 51
Airline_Rating                0
Passenger_Load                0
Flight_Cancelled              0
dtype: int64
```

```
In [4]: # Reload the cleaned dataset
file_path = "monisha_data_cleaned_preprocessed.csv"
df = pd.read_csv(file_path)

# Display basic info to confirm successful load
df.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   float64
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(5), int64(5), object(4)
memory usage: 328.3+ KB
```

```
In [6]: # Step 1: Descriptive statistics
desc_stats = df.describe()

# Step 2: Boxplots for outlier and spread visualization
```

```

numerical_features = df.select_dtypes(include=['int64', 'float64']).drop(columns=["Flight_ID", "Flight_Canceled"])

# Create boxplots
plt.figure(figsize=(18, 12))
for i, col in enumerate(numerical_features, 1):
    plt.subplot(3, 3, i)
    sns.boxplot(y=df[col], color='skyblue')
    plt.title(f'Boxplot of {col}')
plt.tight_layout()
plt.show()

# Step 3: Correlation Matrix
plt.figure(figsize=(12, 8))
correlation_matrix = df.corr(numeric_only=True)
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f", linewidths=0.5)
plt.title("Correlation Matrix")
plt.show()

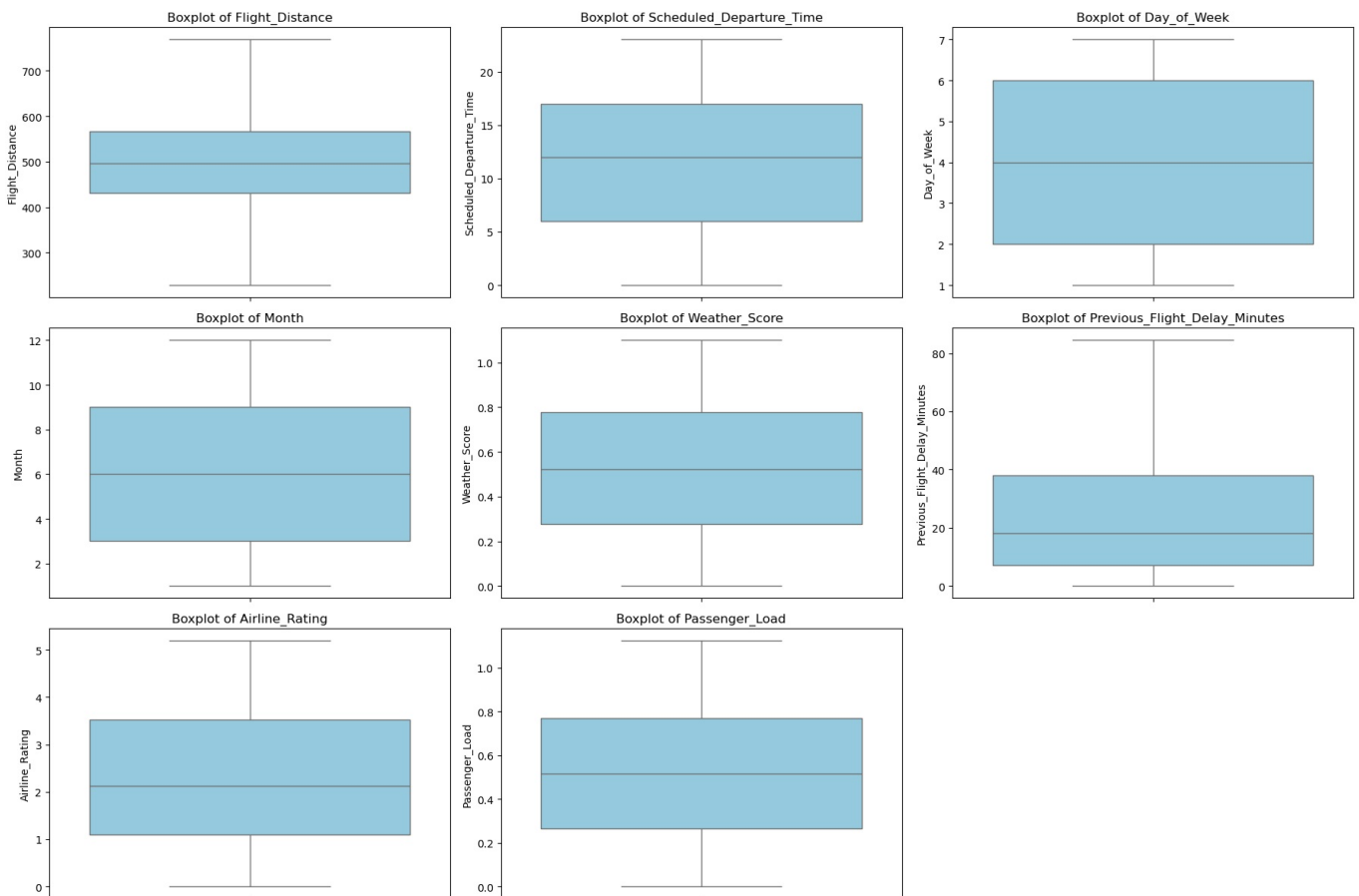
# Step 4: Relationship with Flight_Canceled
# Bar plots for categorical features
categorical_features = ['Airline', 'Origin_Airport', 'Destination_Airport', 'Airplane_Type']

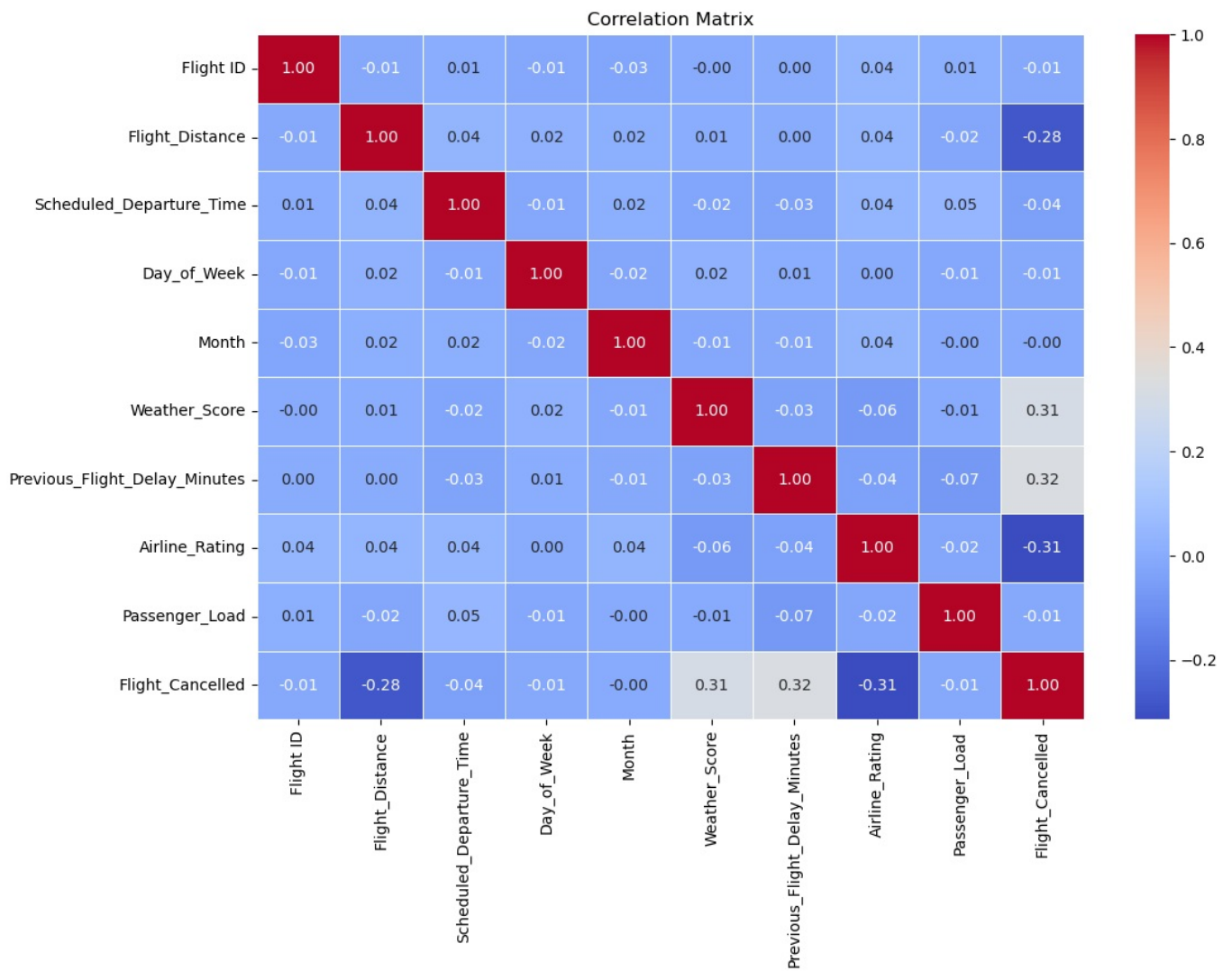
plt.figure(figsize=(20, 16))
for i, col in enumerate(categorical_features, 1):
    plt.subplot(2, 2, i)
    sns.countplot(x=col, hue='Flight_Canceled', data=df, palette='Set2')
    plt.xticks(rotation=45)
    plt.title(f'Flight Cancellation by {col}')
plt.tight_layout()
plt.show()

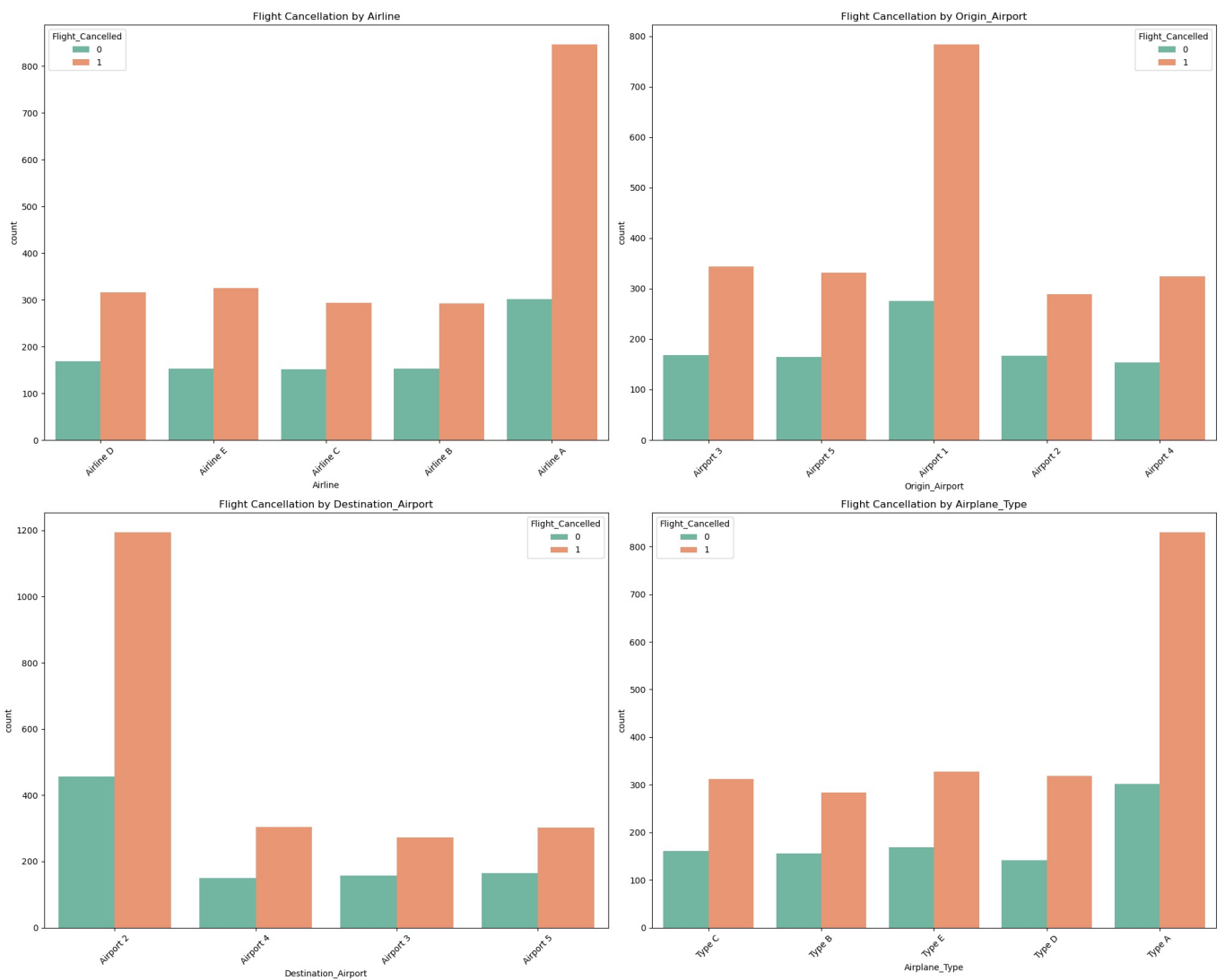
# Save descriptive stats to CSV
eda_csv_path = "monisha_eda_results_analysis.csv"
desc_stats.to_csv(eda_csv_path)

eda_csv_path

```







Out[6]: 'monisha\_eda\_results\_analysis.csv'

In [ ]:

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js