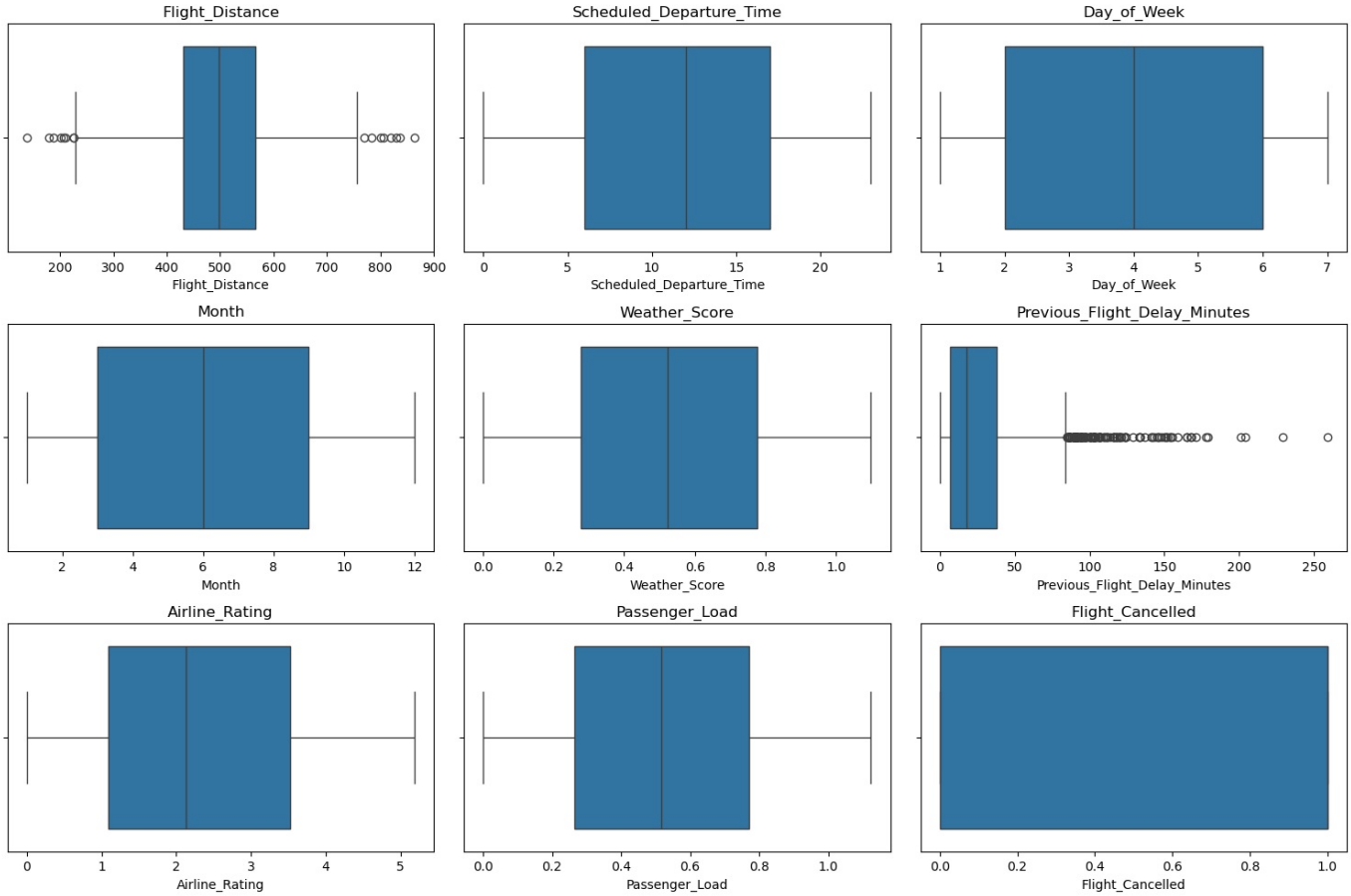
In [1]:
```
import pandas as pd

# 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]:
```
(   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
```



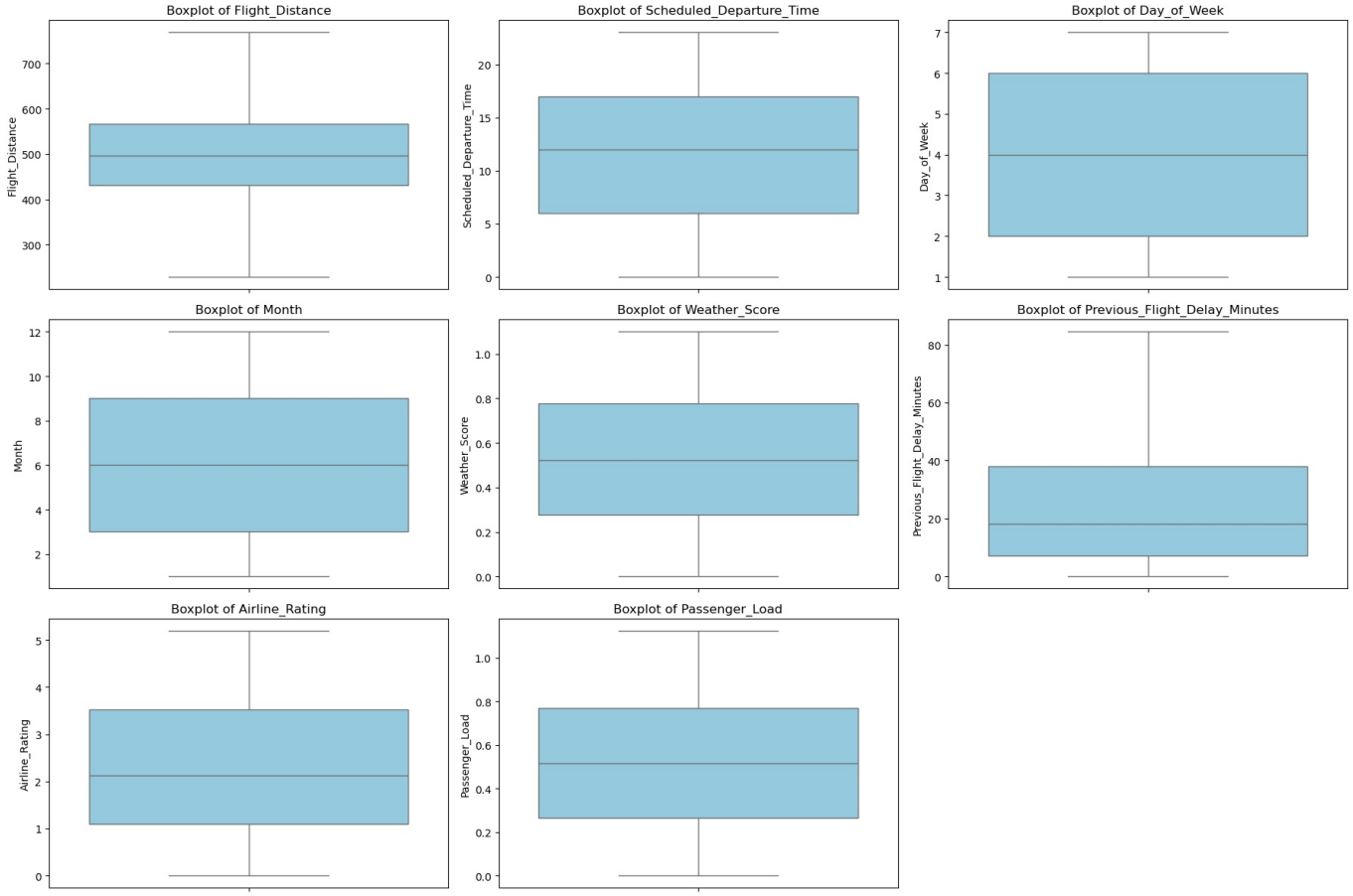
Out[2]:
```
Flight_Distance                10
Scheduled_Departure_Time         0
Day_of_Week                      0
Month                            0
Weather_Score                    0
Previous_Flight_Delay_Minutes    51
Airline_Rating                   0
Passenger_Load                   0
Flight_Cancelled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

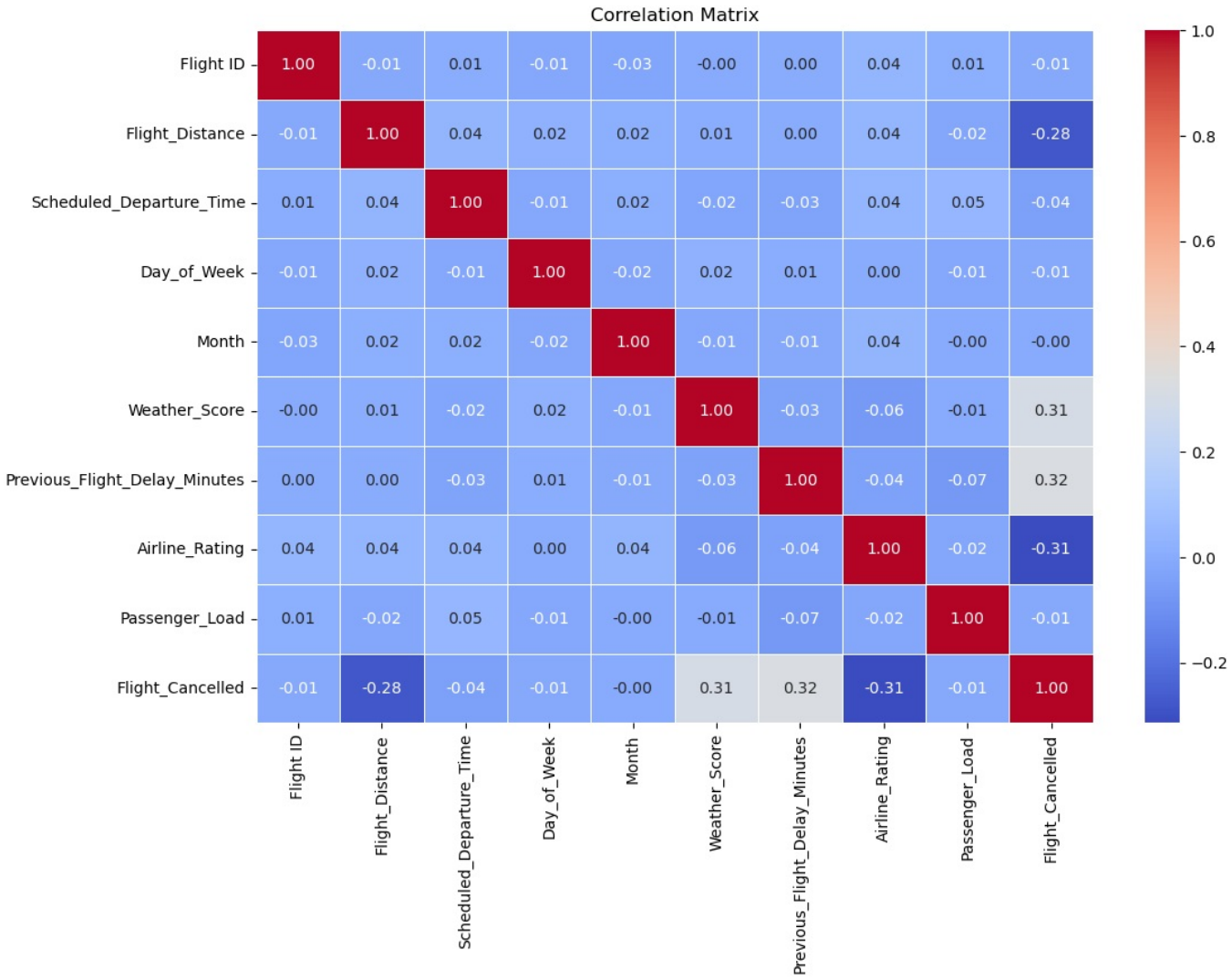
In [4]:
```
# Reload the cleaned dataset
file_path = "monisha_data_cleaned_preprocessed.csv"
df = pd.read_csv(file_path)

# Display basic info to confirm successful load
df.info()
```

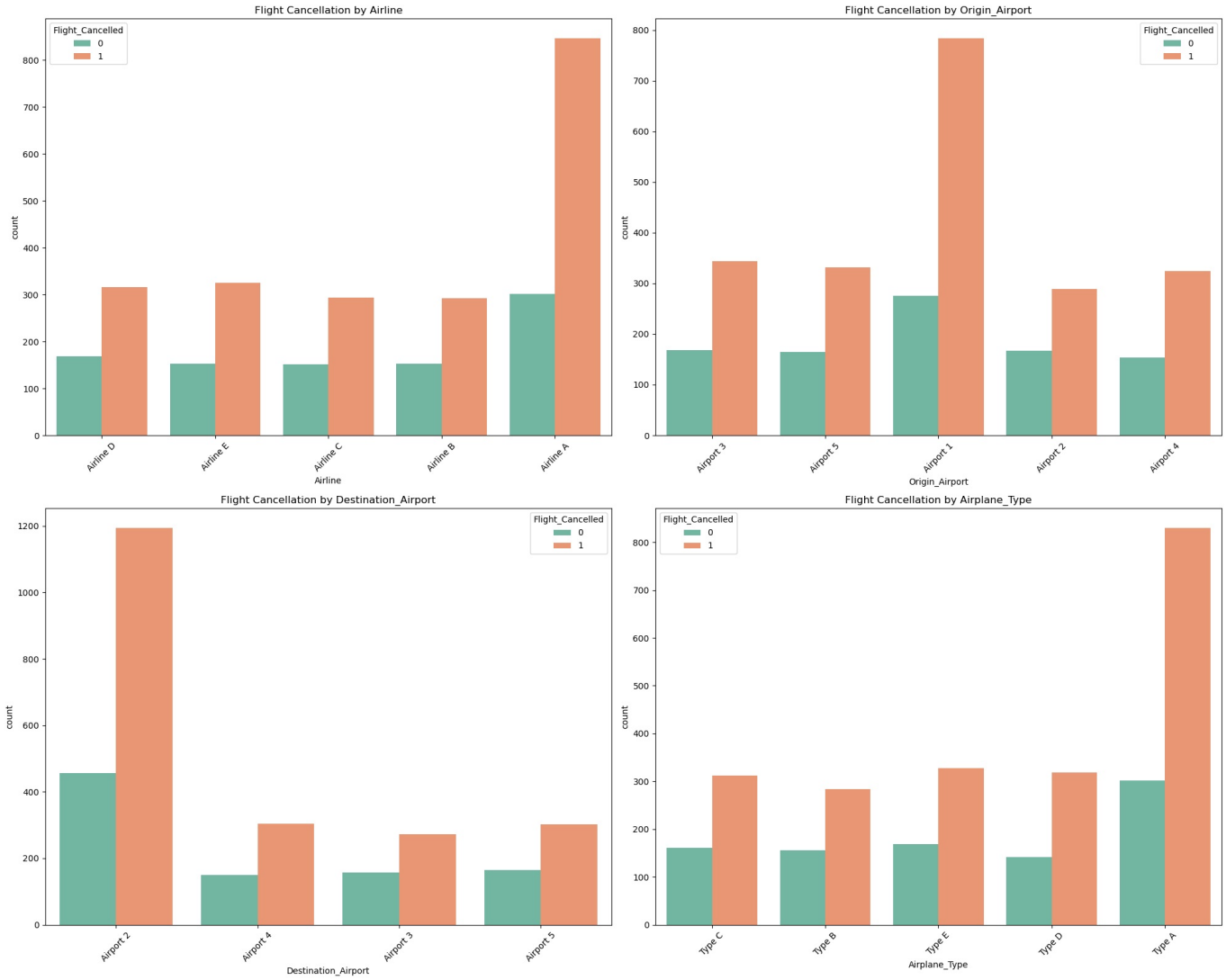
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3000 entries, 0 to 2999
Data columns (total 14 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   Flight ID                       3000 non-null   int64
 1   Airline                         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
```

```python
numerical_features = df.select_dtypes(include=['int64', 'float64']).drop(columns=["Flight ID", "Flight_Cancelle

# 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_Cancelled
# 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_Cancelled', 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
```




Correlation Matrix
Flight ID
Flight_Distance
Scheduled_Departure_Time
Day_of_Week
Month
Weather_Score
Previous_Flight_Delay_Minutes
Airline_Rating
Passenger_Load
Flight_Cancelled
1.00 -0.01 0.01 -0.01 -0.03 -0.00 0.00 0.04 0.01 -0.01
-0.01 1.00 0.04 0.02 0.02 0.01 0.00 0.04 -0.02 -0.28
0.01 0.04 1.00 -0.01 0.02 -0.02 -0.03 0.04 0.05 -0.04
-0.01 0.02 -0.01 1.00 -0.02 0.02 0.01 0.00 -0.01 -0.01
-0.03 0.02 0.02 -0.02 1.00 -0.01 -0.01 0.04 -0.00 -0.00
-0.00 0.01 -0.02 0.02 -0.01 1.00 -0.03 -0.06 -0.01 0.31
0.00 0.00 -0.03 0.01 -0.01 -0.03 1.00 -0.04 -0.07 0.32
0.04 0.04 0.04 0.00 0.04 -0.06 -0.04 1.00 -0.02 -0.31
0.01 -0.02 0.05 -0.01 -0.00 -0.01 -0.07 -0.02 1.00 -0.01
-0.01 -0.28 -0.04 -0.01 -0.00 0.31 0.32 -0.31 -0.01 1.00
1.0
0.8
0.6
0.4
0.2
0.0
−0.2



Out[6]: 'monisha_eda_results_analysis.csv'

In [10]:
```
# Import necessary libraries for feature engineering and modeling
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import OneHotEncoder, StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score, precision_score, recall_sc

# Reload the cleaned dataset
df = pd.read_csv("monisha_data_cleaned_preprocessed.csv")
```

```
# Separate features and target
X = df.drop(columns=["Flight ID", "Flight_Cancelled"])
y = df["Flight_Cancelled"]

# One-hot encode categorical features
categorical_cols = X.select_dtypes(include=['object']).columns.tolist()
X_encoded = pd.get_dummies(X, columns=categorical_cols, drop_first=True)

# Feature scaling
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X_encoded)

# Split the data
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42, stratify=y)

# Model building - Logistic Regression
log_model = LogisticRegression(max_iter=1000)
log_model.fit(X_train, y_train)

# Model prediction
y_pred = log_model.predict(X_test)

# Model evaluation metrics
metrics = {
    "Accuracy": accuracy_score(y_test, y_pred),
    "Precision": precision_score(y_test, y_pred),
    "Recall": recall_score(y_test, y_pred),
    "F1-Score": f1_score(y_test, y_pred)
}
# More readable output
print("Accuracy :", accuracy_score(y_test, y_pred))
print("Precision:", precision_score(y_test, y_pred))
print("Recall   :", recall_score(y_test, y_pred))
print("F1-Score :", f1_score(y_test, y_pred))
```

```
Accuracy : 0.8116666666666666
Precision: 0.8491879350348028
Recall   : 0.8840579710144928
F1-Score : 0.8662721893491124
```

In [13]:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

# Load the feature engineered data
df = pd.read_csv("monisha_feature_engineered_data.csv")

# Separate features and target
X = df.drop(columns=["Flight_Cancelled"])
y = df["Flight_Cancelled"]

# Scale the features again (safe check in case it's needed)
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Split the data using same method from Task 3
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42, stratify=y)

# Initialize models
models = {
    "Logistic Regression": LogisticRegression(max_iter=1000),
    "Decision Tree": DecisionTreeClassifier(random_state=42),
    "Random Forest": RandomForestClassifier(n_estimators=100, random_state=42),
    "Support Vector Machine": SVC()
}

# Train and evaluate each model
results = []

for name, model in models.items():
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    results.append({
        "Model": name,
        "Accuracy": accuracy_score(y_test, y_pred),
        "Precision": precision_score(y_test, y_pred),
        "Recall": recall_score(y_test, y_pred),
        "F1-Score": f1_score(y_test, y_pred)
    })
```

```
# Convert results to DataFrame
results_df = pd.DataFrame(results)
results_df.sort_values(by="F1-Score", ascending=False, inplace=True)
results_df.reset_index(drop=True, inplace=True)
print(results_df)
```

```
                   Model  Accuracy  Precision    Recall  F1-Score
0          Random Forest  0.975000   0.987775  0.975845  0.981774
1          Decision Tree  0.968333   0.978208  0.975845  0.977025
2  Support Vector Machine  0.863333   0.884259  0.922705  0.903073
3     Logistic Regression  0.811667   0.849188  0.884058  0.866272
```

In [ ]: