In [19]:

```
# Importing necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [20]:

```
# Step 1: Loading the dataset
# For this example, we'll load a sample dataset (you can replace this with your dataset)
dataset=pd.read_csv("Kolkata Flight.csv")
```

In [21]:

```
# Step 2: Basic EDA
# Display the first few rows of the dataset
dataset.head()
```

Out[21]:

	Airline	Flight No.	Source	Departure	No. of stops	Arrival	Destination	Ticket Class	Flight Duration (hrs)	Days left	Price
0	SpiceJet	SG-8264	Kolkata	Night	0	Late_Night	Delhi	Economy	2.50	1	6488
1	AirAsia	I5-582	Kolkata	Morning	1	Evening	Delhi	Economy	9.25	1	6353
2	AirAsia	I5-2473	Kolkata	Morning	1	Night	Delhi	Economy	12.42	1	6353
3	AirAsia	I5-1563	Kolkata	Evening	2_or_more	Afternoon	Delhi	Economy	18.33	1	6353
4	Indigo	6E-2009	Kolkata	Night	0	Late_Night	Delhi	Economy	2.50	1	6489

In [22]:

```
# Get general info about the dataset
print(dataset.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 46347 entries, 0 to 46346
Data columns (total 11 columns):
 #   Column                 Non-Null Count  Dtype  
---  ------                 --------------  -----  
 0   Airline                46347 non-null  object 
 1   Flight No.             46347 non-null  object 
 2   Source                 46347 non-null  object 
 3   Departure              46347 non-null  object 
 4   No. of stops           46347 non-null  object 
 5   Arrival                46347 non-null  object 
 6   Destination            46347 non-null  object 
 7   Ticket Class           46347 non-null  object 
 8   Flight Duration (hrs)  46347 non-null  float64
 9   Days left              46347 non-null  int64  
 10  Price                  46347 non-null  int64  
dtypes: float64(1), int64(2), object(8)
memory usage: 3.9+ MB
None
```

In [23]:

```
# Summary statistics
print(dataset.describe())
```

```
       Flight Duration (hrs)     Days left          Price
count           46347.000000  46347.000000   46347.000000
mean               13.249898     26.013162   21746.235679
std                 7.223163     13.511999   23439.972854
min                 2.080000      1.000000    2436.000000
25%                 7.670000     15.000000    5853.000000
50%                12.250000     26.000000    7958.000000
75%                17.500000     38.000000   49207.000000
max                40.500000     49.000000  123071.000000
```

In [24]:

```
# Step 3: Handling missing data
# Checking for missing values
print(dataset.isnull().sum())
```

```
Airline                  0
Flight No.               0
Source                   0
Departure                0
No. of stops             0
Arrival                  0
Destination              0
Ticket Class             0
Flight Duration (hrs)    0
Days left                0
Price                    0
dtype: int64
```

In [29]:

```
# Checking categorical columns
categorical_columns = dataset.select_dtypes(include=['object', 'category']).columns.tolist()
```

In [30]:

```
# Checking numerical columns
numerical_columns = dataset.select_dtypes(include=['number']).columns.tolist()
```

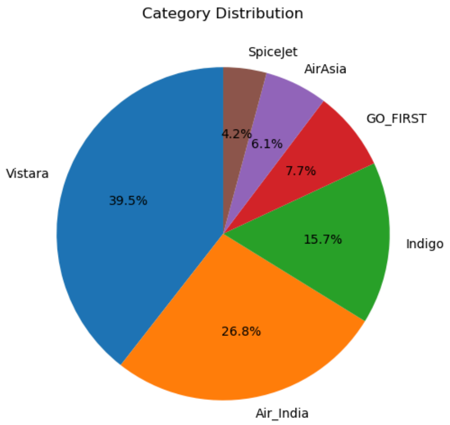
In [31]:

```
print("Categorical columns:", categorical_columns)
print("Numerical columns:", numerical_columns)
```

```
Categorical columns: ['Airline', 'Flight No.', 'Source', 'Departure', 'No. of stops', 'Arrival', 'Destination', 'Ticket Class']
Numerical columns: ['Flight Duration (hrs)', 'Days left', 'Price']
```

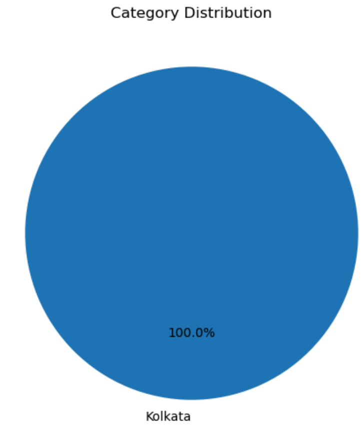
In [33]:

```
category_counts = dataset['Airline'].value_counts()
plt.figure(figsize=(6, 6))
category_counts.plot.pie(autopct='%1.1f%%', startangle=90)
plt.title('Category Distribution')
plt.ylabel('')
plt.show()
```



In [36]:

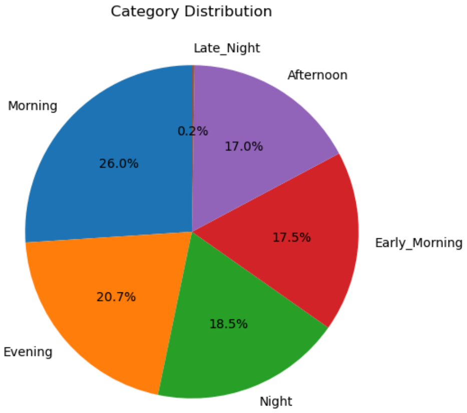
```
category_counts = dataset['Source'].value_counts()
plt.figure(figsize=(6, 6))
category_counts.plot.pie(autopct='%1.1f%%', startangle=90)
plt.title('Category Distribution')
plt.ylabel('')
plt.show()
```



Category Distribution

In [38]:

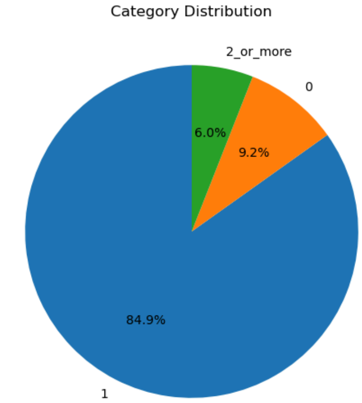
```
category_counts = dataset['Departure'].value_counts()
plt.figure(figsize=(6, 6))
category_counts.plot.pie(autopct='%1.1f%%', startangle=90)
plt.title('Category Distribution')
plt.ylabel('')
plt.show()
```



Category Distribution

In [39]:

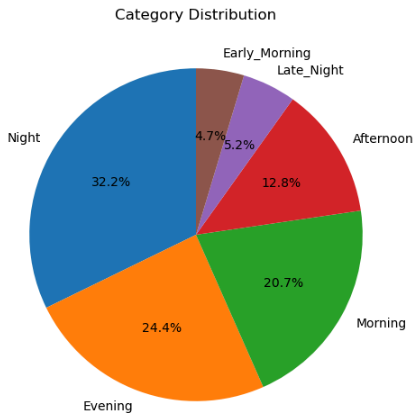
```
category_counts = dataset['No. of stops'].value_counts()
plt.figure(figsize=(6, 6))
category_counts.plot.pie(autopct='%1.1f%%', startangle=90)
plt.title('Category Distribution')
plt.ylabel('')
plt.show()
```



Category Distribution

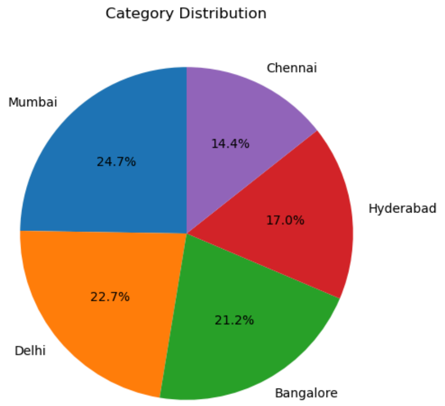
In [40]:

```
category_counts = dataset['Arrival'].value_counts()
plt.figure(figsize=(6, 6))
category_counts.plot.pie(autopct='%1.1f%%', startangle=90)
plt.title('Category Distribution')
plt.ylabel('')
plt.show()
```



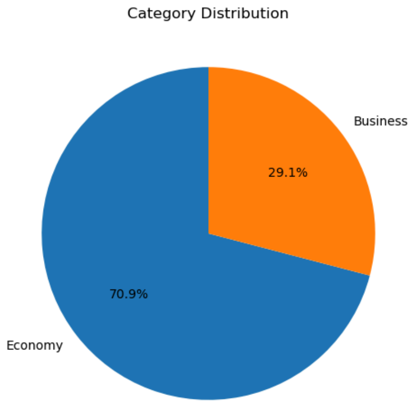
In [41]:

```
category_counts = dataset['Destination'].value_counts()
plt.figure(figsize=(6, 6))
category_counts.plot.pie(autopct='%1.1f%%', startangle=90)
plt.title('Category Distribution')
plt.ylabel('')
plt.show()
```



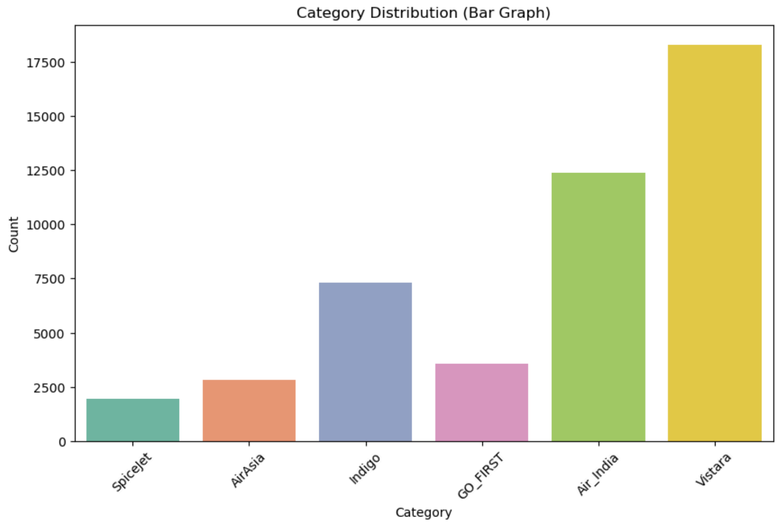
In [42]:

```
category_counts = dataset['Ticket Class'].value_counts()
plt.figure(figsize=(6, 6))
category_counts.plot.pie(autopct='%1.1f%%', startangle=90)
plt.title('Category Distribution')
plt.ylabel('')
plt.show()
```

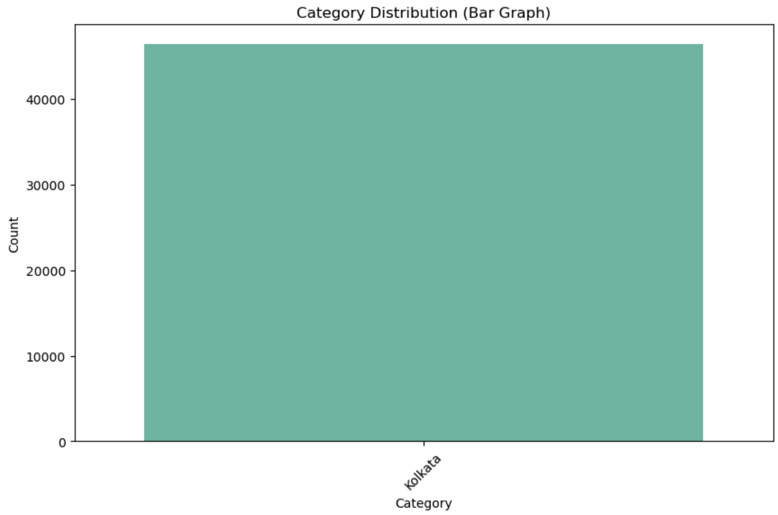


In [43]:

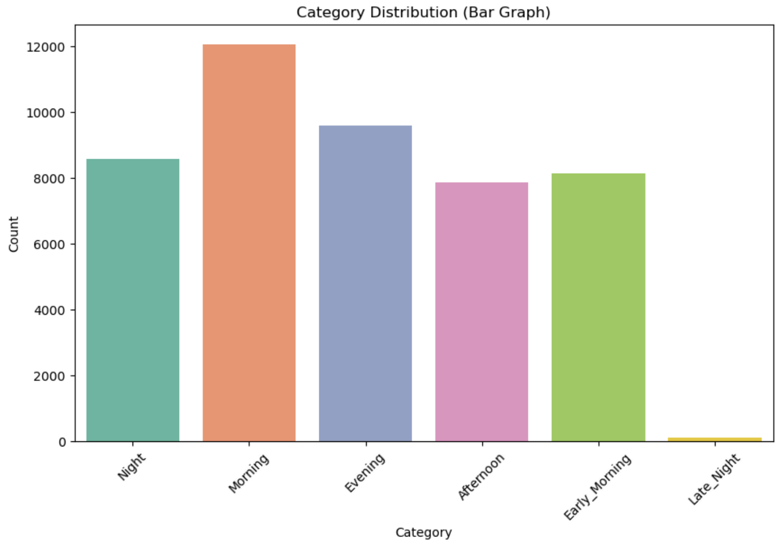
```
# 2. Bar Graph (for categorical data)
plt.figure(figsize=(10, 6))
sns.countplot(data=dataset, x='Airline', palette='Set2')
plt.title('Category Distribution (Bar Graph)')
plt.xlabel('Category')
plt.ylabel('Count')
plt.xticks(rotation=45)
plt.show()
```



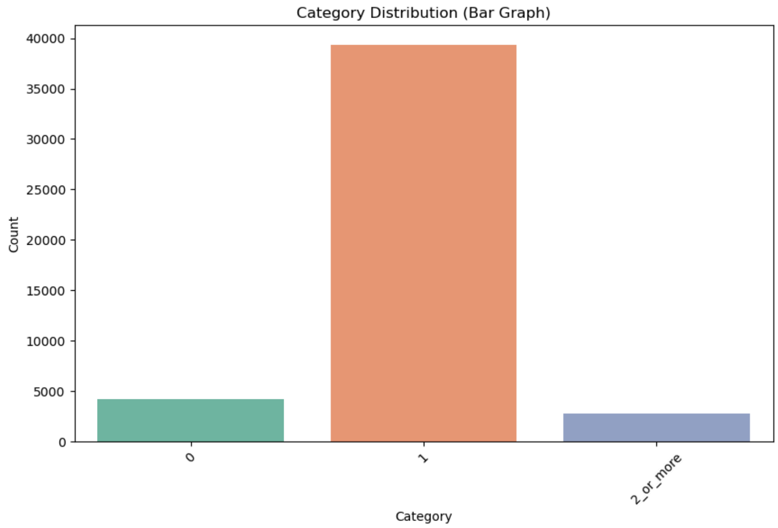
In [44]:
```python
plt.figure(figsize=(10, 6))
sns.countplot(data=dataset, x='Source', palette='Set2')
plt.title('Category Distribution (Bar Graph)')
plt.xlabel('Category')
plt.ylabel('Count')
plt.xticks(rotation=45)
plt.show()
```



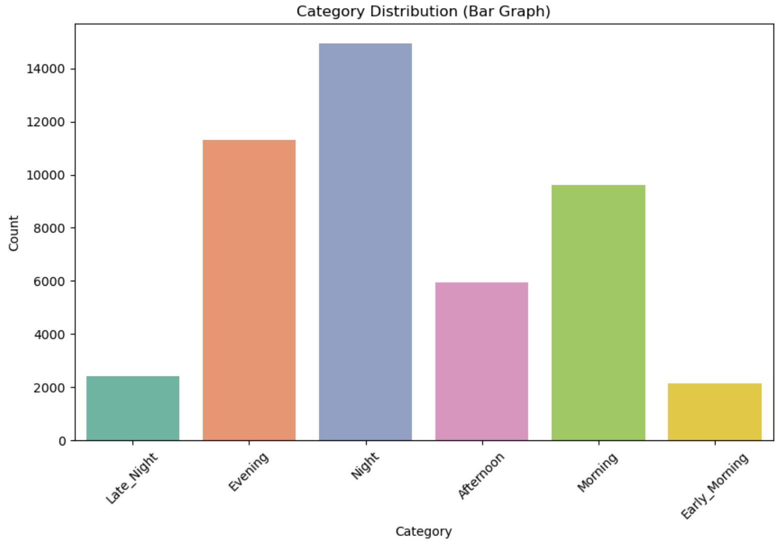
In [46]:
```python
plt.figure(figsize=(10, 6))
sns.countplot(data=dataset, x='Departure', palette='Set2')
plt.title('Category Distribution (Bar Graph)')
plt.xlabel('Category')
plt.ylabel('Count')
plt.xticks(rotation=45)
plt.show()
```



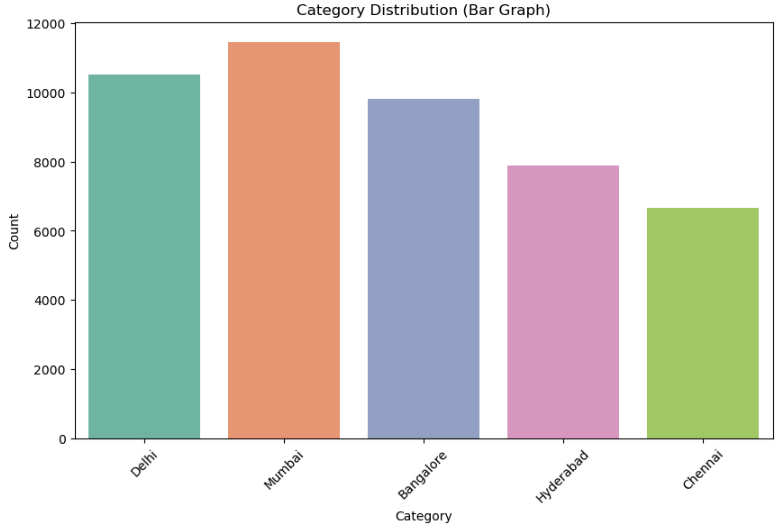
In [49]:
```python
plt.figure(figsize=(10, 6))
sns.countplot(data=dataset, x='No. of stops', palette='Set2')
plt.title('Category Distribution (Bar Graph)')
plt.xlabel('Category')
plt.ylabel('Count')
plt.xticks(rotation=45)
plt.show()
```



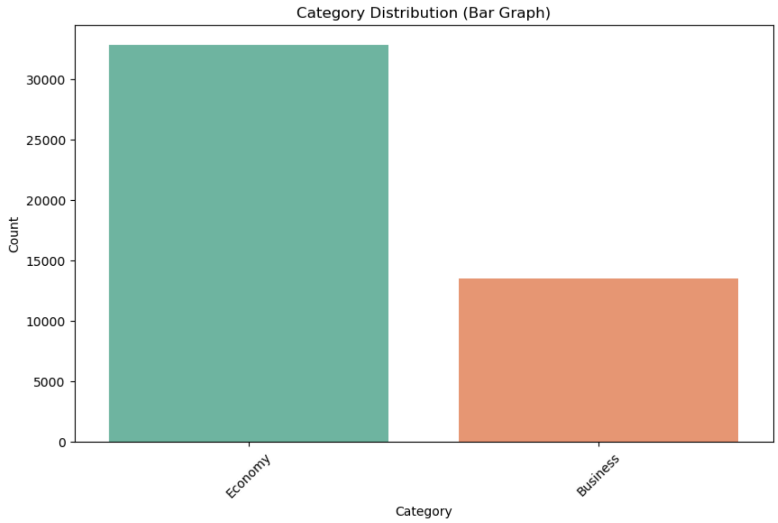
In [50]:
```
plt.figure(figsize=(10, 6))
sns.countplot(data=dataset, x='Arrival', palette='Set2')
plt.title('Category Distribution (Bar Graph)')
plt.xlabel('Category')
plt.ylabel('Count')
plt.xticks(rotation=45)
plt.show()
```



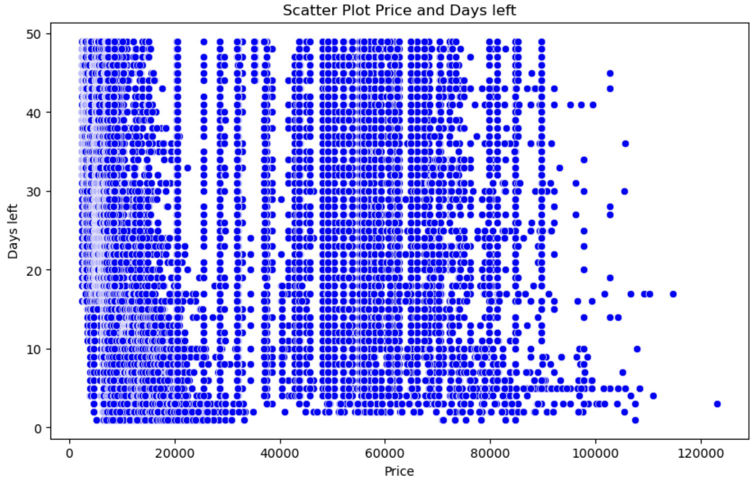
In [51]:
```
plt.figure(figsize=(10, 6))
sns.countplot(data=dataset, x='Destination', palette='Set2')
plt.title('Category Distribution (Bar Graph)')
plt.xlabel('Category')
plt.ylabel('Count')
plt.xticks(rotation=45)
plt.show()
```



In [52]:
```
plt.figure(figsize=(10, 6))
sns.countplot(data=dataset, x='Ticket Class', palette='Set2')
plt.title('Category Distribution (Bar Graph)')
plt.xlabel('Category')
plt.ylabel('Count')
plt.xticks(rotation=45)
plt.show()
```



In [56]:
```
# 3. Scatter Plot (for relationships between numerical columns)
plt.figure(figsize=(10, 6))
sns.scatterplot(data=dataset, x='Price', y='Days left', color='blue')
plt.title('Scatter Plot Price and Days left')
plt.xlabel('Price')
plt.ylabel('Days left')
plt.show()
```



In [61]:
```
from sklearn.preprocessing import LabelEncoder

# Create a label encoder object
label_encoder = LabelEncoder()

# Example: Convert a categorical column to numeric
dataset['Airline_encoded'] = label_encoder.fit_transform(dataset['Airline'])
```

In [62]:
```
dataset['Flight No._encoded'] = label_encoder.fit_transform(dataset['Flight No.'])
dataset['Source_encoded'] = label_encoder.fit_transform(dataset['Source'])
dataset['Departure_encoded'] = label_encoder.fit_transform(dataset['Departure'])
dataset['No. of stops_encoded'] = label_encoder.fit_transform(dataset['No. of stops'])
dataset['Arrival_encoded'] = label_encoder.fit_transform(dataset['Arrival'])
dataset['Destination_encoded'] = label_encoder.fit_transform(dataset['Destination'])
dataset['Ticket Class_encoded'] = label_encoder.fit_transform(dataset['Ticket Class'])
```

In [64]:
```
dataset.dtypes
```

Out[64]:
```
Airline                   object
Flight No.                object
Source                    object
Departure                 object
No. of stops              object
Arrival                   object
Destination               object
Ticket Class              object
Flight Duration (hrs)    float64
Days left                  int64
Price                      int64
Airline_encoded            int32
Flight No._encoded         int32
Source_encoded             int32
Departure_encoded          int32
No. of stops_encoded       int32
Arrival_encoded            int32
Destination_encoded        int32
Ticket Class_encoded       int32
dtype: object
```

In [73]:
```
label_encoder = LabelEncoder()

# Apply label encoding to all categorical columns
for column in dataset.select_dtypes(include=['object']).columns:
    dataset[column] = label_encoder.fit_transform(dataset[column])

# View the first few rows of the updated DataFrame
print(dataset.head())
```

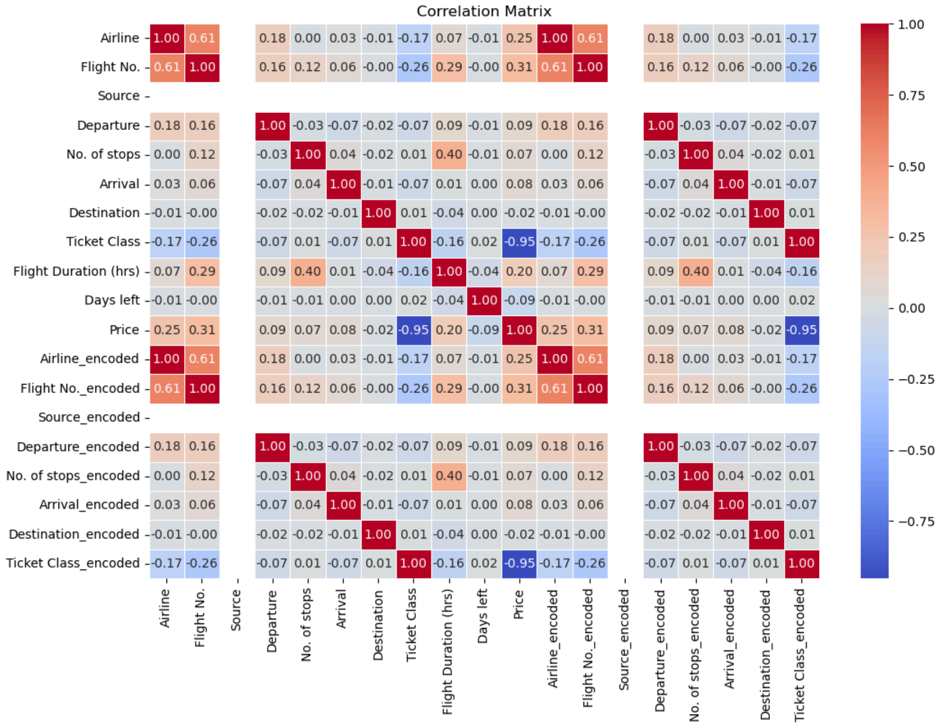
```
   Airline  Flight No.  Source  Departure  No. of stops  Arrival  Destination  \
0        4         196       0          5             0        3            2   
1        0         159       0          4             1        2            2   
2        0         148       0          4             1        5            2   
3        0         147       0          2             2        0            2   
4        3          16       0          5             0        3            2   

   Ticket Class  Flight Duration (hrs)  Days left  Price  Airline_encoded  \
0             1                   2.50          1   6488                4   
1             1                   9.25          1   6353                0   
2             1                  12.42          1   6353                0   
3             1                  18.33          1   6353                0   
4             1                   2.50          1   6489                3   

   Flight No._encoded  Source_encoded  Departure_encoded  \
0                 196               0                  5   
1                 159               0                  4   
2                 148               0                  4   
3                 147               0                  2   
4                  16               0                  5   

   No. of stops_encoded  Arrival_encoded  Destination_encoded  \
0                     0                3                    2   
1                     1                2                    2   
2                     1                5                    2   
3                     2                0                    2   
4                     0                3                    2   

   Ticket Class_encoded  
0                     1  
1                     1  
2                     1  
3                     1  
4                     1  
```

In [74]:
```
dataset.dtypes
```

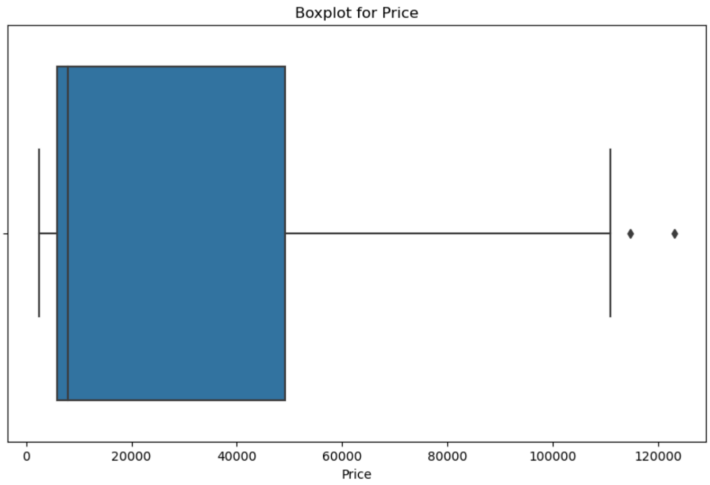
Out[74]:
```
Airline                    int32
Flight No.                 int32
Source                     int32
Departure                  int32
No. of stops               int32
Arrival                    int32
Destination                int32
Ticket Class               int32
Flight Duration (hrs)    float64
Days left                  int64
Price                      int64
Airline_encoded            int32
Flight No._encoded         int32
Source_encoded             int32
Departure_encoded          int32
No. of stops_encoded       int32
Arrival_encoded            int32
Destination_encoded        int32
Ticket Class_encoded       int32
dtype: object
```

In [75]:
```
# Step 5: Correlation Analysis
# Checking correlations between numerical variables
correlation_matrix = dataset.corr()
```

In [76]:
```
plt.figure(figsize=(12, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt='.2f', linewidths=0.5)
plt.title('Correlation Matrix')
plt.show()
```

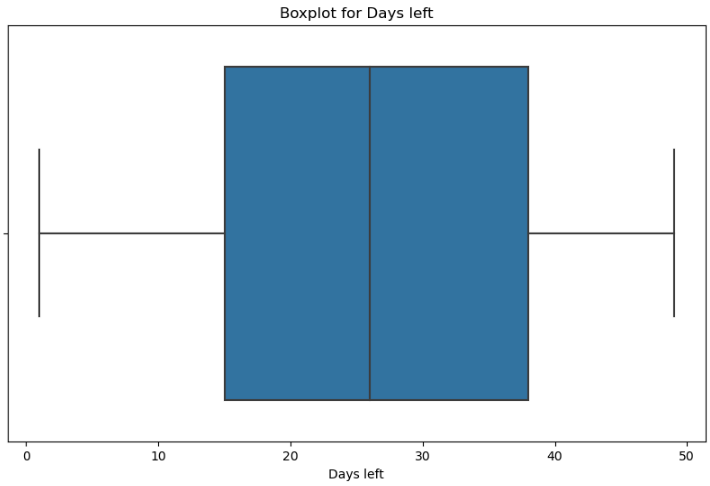


In [79]:
```
# Step 6: Outlier Detection
# Using boxplots to identify outliers
plt.figure(figsize=(10, 6))
sns.boxplot(data=dataset, x='Price')
plt.title('Boxplot for Price')
plt.show()
```



In [80]:

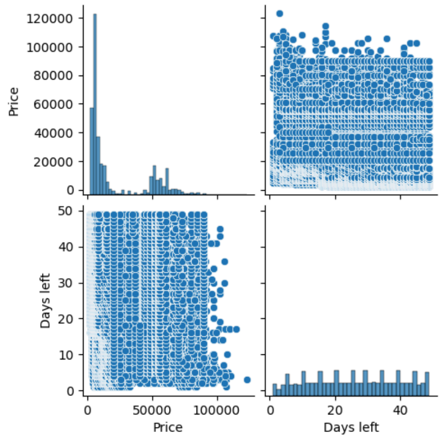
```
# Step 6: Outlier Detection
# Using boxplots to identify outliers
plt.figure(figsize=(10, 6))
sns.boxplot(data=dataset, x='Days left')
plt.title('Boxplot for Days left')
plt.show()
```



In [83]:

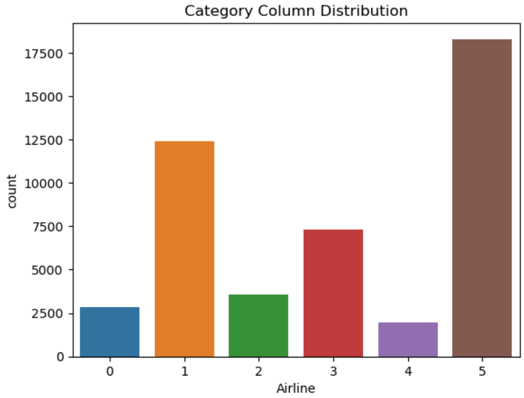
```
# Step 7: Data Transformation (if necessary)
# Normalize or scale data if needed
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
dataset_scaled = pd.DataFrame(scaler.fit_transform(dataset[['Price', 'Days left']]), columns=['Price', 'Days left'])

# Step 8: More Visualizations
# Pairplot (for relationships between multiple numeric variables)
sns.pairplot(df[['Price', 'Days left']])
plt.show()
```

```
C:\Users\Deviare User\anaconda3\Lib\site-packages\seaborn\axisgrid.py:118: UserWarning: The figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```



In [85]:

```
# Step 9: Analyzing Categorical Data
# Visualize categorical data distributions with countplots
sns.countplot(data=dataset, x='Airline')
plt.title('Category Column Distribution')
plt.show()
```


Category Column Distribution
count
0
2500
5000
7500
10000
12500
15000
17500
0
1
2
3
4
5
Airline

```
In [ ]:
```