In [1]:
```
#installing & importing modules and packages
%pip install --upgrade pip
%pip install pandas numpy openpyxl seaborn matplotlib

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

#loading training and testing dataset
train_data = pd.read_excel("TEAM15_DATASET1(TRAIN).xlsx")
test_data = pd.read_excel("TEAM15_DATASET2(TEST).xlsx")

#saving 'price' column in a variable
y = train_data['Price']

#removing price column
train_data = train_data.drop('Price', axis=1)

#flagging to differentiate tran and test dataset
train_data['is_train'] = 1
test_data['is_train'] = 0

#merging both training and testing dataset
df = pd.concat([train_data, test_data], ignore_index=True)

print(f"Combined dataset shape: {df.shape}")
print(f"Training samples: {df['is_train'].sum()}")
print(f"Test samples: {(df['is_train'] == 0).sum()}")
print("\nFirst few rows of combined dataset:")
df.head()
```

```
Requirement already satisfied: pip in c:\users\jishn\anaconda3\lib\site-packages
(25.2)
Note: you may need to restart the kernel to use updated packages.
Requirement already satisfied: pandas in c:\users\jishn\anaconda3\lib\site-packag
es (2.2.2)
Requirement already satisfied: numpy in c:\users\jishn\anaconda3\lib\site-package
s (1.26.4)
Requirement already satisfied: openpyxl in c:\users\jishn\anaconda3\lib\site-pack
ages (3.1.5)
Requirement already satisfied: seaborn in c:\users\jishn\anaconda3\lib\site-packa
ges (0.13.2)
Requirement already satisfied: matplotlib in c:\users\jishn\anaconda3\lib\site-pa
ckages (3.9.2)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\jishn\anaconda3
\lib\site-packages (from pandas) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in c:\users\jishn\anaconda3\lib\site-
packages (from pandas) (2024.1)
Requirement already satisfied: tzdata>=2022.7 in c:\users\jishn\anaconda3\lib\sit
e-packages (from pandas) (2023.3)
Requirement already satisfied: et-xmlfile in c:\users\jishn\anaconda3\lib\site-pa
ckages (from openpyxl) (1.1.0)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\jishn\anaconda3\lib\s
ite-packages (from matplotlib) (1.2.0)
Requirement already satisfied: cycler>=0.10 in c:\users\jishn\anaconda3\lib\site-
packages (from matplotlib) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\jishn\anaconda3\lib
\site-packages (from matplotlib) (4.51.0)
Requirement already satisfied: kiwisolver>=1.3.1 in c:\users\jishn\anaconda3\lib
\site-packages (from matplotlib) (1.4.4)
Requirement already satisfied: packaging>=20.0 in c:\users\jishn\anaconda3\lib\si
te-packages (from matplotlib) (24.1)
Requirement already satisfied: pillow>=8 in c:\users\jishn\anaconda3\lib\site-pac
kages (from matplotlib) (10.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in c:\users\jishn\anaconda3\lib\s
ite-packages (from matplotlib) (3.1.2)
Requirement already satisfied: six>=1.5 in c:\users\jishn\anaconda3\lib\site-pack
ages (from python-dateutil>=2.8.2->pandas) (1.16.0)
Note: you may need to restart the kernel to use updated packages.
Combined dataset shape: (13354, 11)
Training samples: 10683
Test samples: 2671

First few rows of combined dataset:
```

Out[1]:

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Du
0	IndiGo	24/03/2019	Banglore	New Delhi	BLR → DEL	22:20	01:10 22 Mar	2
1	Air India	1/05/2019	Kolkata	Banglore	CCU → IXR → BBI → BLR	05:50	13:15	7
2	Jet Airways	9/06/2019	Delhi	Cochin	DEL → LKO → BOM → COK	09:25	04:25 10 Jun	
3	IndiGo	12/05/2019	Kolkata	Banglore	CCU → NAG → BLR	18:05	23:30	5
4	IndiGo	01/03/2019	Banglore	New Delhi	BLR → NAG → DEL	16:50	21:35	4

In [2]:
```
#before preprocessing
df.shape
```

Out[2]:
```
(13354, 11)
```

In [3]:
```
#removing 'Route' column
df.drop('Route', axis=1, inplace=True)
```

In [4]:
```
##checking for null values
df.isnull().sum()
```

Out[4]:
```
Airline            0
Date_of_Journey    0
Source             0
Destination        0
Dep_Time           0
Arrival_Time       0
Duration           0
Total_Stops        1
Additional_Info    0
is_train           0
dtype: int64
```

In [5]:
```
#accessing row with null value
rows_with_nulls = df[df.isnull().any(axis=1)]
rows_with_nulls
```

Out[5]:

	Airline	Date_of_Journey	Source	Destination	Dep_Time	Arrival_Time	Duration
9039	Air India	6/05/2019	Delhi	Cochin	09:45	09:25 07 May	23h 40m

In [6]:
```
df.shape
```

Out[6]:
```
(13354, 10)
```

In [7]:
```
#getting count of variuos data in 'additional info'
df['Additional_Info'].value_counts()
```

Out[7]:
```
Additional_Info
No info                         10493
In-flight meal not included      2426
No check-in baggage included      396
1 Long layover                     20
Change airports                     8
Business class                      5
No Info                             3
1 Short layover                     1
Red-eye flight                      1
2 Long layover                      1
Name: count, dtype: int64
```

In [8]:
```
#dividing it to two categories
df['Additional_Info'] = df['Additional_Info'].apply(lambda x: 'No info' if x ==
df['Additional_Info'].value_counts()
```

Out[8]:
```
Additional_Info
No info    10493
Others      2861
Name: count, dtype: int64
```

In [9]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 13354 entries, 0 to 13353
Data columns (total 10 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   Airline          13354 non-null  object
 1   Date_of_Journey  13354 non-null  object
 2   Source           13354 non-null  object
 3   Destination      13354 non-null  object
 4   Dep_Time         13354 non-null  object
 5   Arrival_Time     13354 non-null  object
 6   Duration         13354 non-null  object
 7   Total_Stops      13353 non-null  object
 8   Additional_Info  13354 non-null  object
 9   is_train         13354 non-null  int64
dtypes: int64(1), object(9)
memory usage: 1.0+ MB
```

In [10]:
```
df.head()
```

Out[10]:

	Airline	Date_of_Journey	Source	Destination	Dep_Time	Arrival_Time	Duration
0	IndiGo	24/03/2019	Banglore	New Delhi	22:20	01:10 22 Mar	2h 50m
1	Air India	1/05/2019	Kolkata	Banglore	05:50	13:15	7h 25m
2	Jet Airways	9/06/2019	Delhi	Cochin	09:25	04:25 10 Jun	19h
3	IndiGo	12/05/2019	Kolkata	Banglore	18:05	23:30	5h 25m
4	IndiGo	01/03/2019	Banglore	New Delhi	16:50	21:35	4h 45m

In [11]:

```
#feature engineering
df['Date'] = df['Date_of_Journey'].str.split('/').str[0]
df['Month'] = df['Date_of_Journey'].str.split('/').str[1]
df['Year'] = df['Date_of_Journey'].str.split('/').str[2]
df.drop('Date_of_Journey',axis=1,inplace=True)
df.head()
```

Out[11]:

	Airline	Source	Destination	Dep_Time	Arrival_Time	Duration	Total_Stops	Addit
0	IndiGo	Banglore	New Delhi	22:20	01:10 22 Mar	2h 50m	non-stop	
1	Air India	Kolkata	Banglore	05:50	13:15	7h 25m	2 stops	
2	Jet Airways	Delhi	Cochin	09:25	04:25 10 Jun	19h	2 stops	
3	IndiGo	Kolkata	Banglore	18:05	23:30	5h 25m	1 stop	
4	IndiGo	Banglore	New Delhi	16:50	21:35	4h 45m	1 stop	

In [12]:

```
df['Dep_hour'] = df['Dep_Time'].str.split(':').str[0]
df['Dep_minute'] = df['Dep_Time'].str.split(':').str[1]
df.drop('Dep_Time', axis=1, inplace=True)

df['Arrival_hour'] = df['Arrival_Time'].str.split(' ').str[0].str.split(':').str
df['Arrival_minute'] = df['Arrival_Time'].str.split(' ').str[0].str.split(':').s
df.drop('Arrival_Time', axis=1, inplace=True)

df.head()
```

Out[12]:

	Airline	Source	Destination	Duration	Total_Stops	Additional_Info	is_train	Date
0	IndiGo	Banglore	New Delhi	2h 50m	non-stop	No info	1	24
1	Air India	Kolkata	Banglore	7h 25m	2 stops	No info	1	1
2	Jet Airways	Delhi	Cochin	19h	2 stops	No info	1	9
3	IndiGo	Kolkata	Banglore	5h 25m	1 stop	No info	1	12
4	IndiGo	Banglore	New Delhi	4h 45m	1 stop	No info	1	01

In [13]:

```
#splitting duration to hours and minutes
def extract_duration_hours(duration_str):
    try:
        if 'h' in duration_str:
            return int(duration_str.split('h')[0])
        else:
            return 0
    except:
        return 0

def extract_duration_minutes(duration_str):
    try:
        if 'm' in duration_str:
            parts = duration_str.split(' ')
            for part in parts:
                if 'm' in part:
                    return int(part.replace('m', ''))
            return 0
        else:
            return 0
    except:
        return 0

# Apply the functions
df['Duration_hour'] = df['Duration'].apply(extract_duration_hours)
df['Duration_minute'] = df['Duration'].apply(extract_duration_minutes)
df.drop('Duration', axis=1, inplace=True)

#converting hours to minutes to redcue the features
df['Duration_minutes'] = df['Duration_hour']*60 + df['Duration_minute']
df.drop(['Duration_hour','Duration_minute'], axis=1, inplace=True)

df.head()
```

Out[13]:

	Airline	Source	Destination	Total_Stops	Additional_Info	is_train	Date	Month
0	IndiGo	Banglore	New Delhi	non-stop	No info	1	24	03
1	Air India	Kolkata	Banglore	2 stops	No info	1	1	05
2	Jet Airways	Delhi	Cochin	2 stops	No info	1	9	06
3	IndiGo	Kolkata	Banglore	1 stop	No info	1	12	05
4	IndiGo	Banglore	New Delhi	1 stop	No info	1	01	03

In [14]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 13354 entries, 0 to 13353
Data columns (total 14 columns):
 #   Column            Non-Null Count  Dtype 
---  ------            --------------  ----- 
 0   Airline           13354 non-null  object
 1   Source            13354 non-null  object
 2   Destination       13354 non-null  object
 3   Total_Stops       13353 non-null  object
 4   Additional_Info   13354 non-null  object
 5   is_train          13354 non-null  int64 
 6   Date              13354 non-null  object
 7   Month             13354 non-null  object
 8   Year              13354 non-null  object
 9   Dep_hour          13354 non-null  object
 10  Dep_minute        13354 non-null  object
 11  Arrival_hour      13354 non-null  object
 12  Arrival_minute    13354 non-null  object
 13  Duration_minutes  13354 non-null  int64 
dtypes: int64(2), object(12)
memory usage: 1.4+ MB
```

In [15]:
```
df[['Date','Month','Year','Dep_hour','Dep_minute','Arrival_hour','Arrival_minute
```

Out[15]:
```
Date                0
Month               0
Year                0
Dep_hour            0
Dep_minute          0
Arrival_hour        0
Arrival_minute      0
Duration_minutes    0
dtype: int64
```

In [16]:
```
#converting to int datatype
col_to_convert = ['Date','Month','Year','Dep_hour','Dep_minute','Arrival_hour','
df[col_to_convert] = df[col_to_convert].astype(int)
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 13354 entries, 0 to 13353
Data columns (total 14 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   Airline           13354 non-null  object
 1   Source            13354 non-null  object
 2   Destination       13354 non-null  object
 3   Total_Stops       13353 non-null  object
 4   Additional_Info   13354 non-null  object
 5   is_train          13354 non-null  int64
 6   Date              13354 non-null  int32
 7   Month             13354 non-null  int32
 8   Year              13354 non-null  int32
 9   Dep_hour          13354 non-null  int32
 10  Dep_minute        13354 non-null  int32
 11  Arrival_hour      13354 non-null  int32
 12  Arrival_minute    13354 non-null  int32
 13  Duration_minutes  13354 non-null  int64
dtypes: int32(7), int64(2), object(5)
memory usage: 1.1+ MB
```

In [17]:
```
df.head()
```

Out[17]:

	Airline	Source	Destination	Total_Stops	Additional_Info	is_train	Date	Month
0	IndiGo	Banglore	New Delhi	non-stop	No info	1	24	3
1	Air India	Kolkata	Banglore	2 stops	No info	1	1	5
2	Jet Airways	Delhi	Cochin	2 stops	No info	1	9	6
3	IndiGo	Kolkata	Banglore	1 stop	No info	1	12	5
4	IndiGo	Banglore	New Delhi	1 stop	No info	1	1	3

In [18]:
```
#accessing unique values in total_stops column
df['Total_Stops'].unique()
```

Out[18]:
```
array(['non-stop', '2 stops', '1 stop', '3 stops', nan, '4 stops'],
      dtype=object)
```

In [19]:
```
#converting text data to numerical values
df['Total_Stops'] = df['Total_Stops'].map({'non-stop':0, '2 stops':2, '1 stop':1
df.head()
```

Out[19]:

	Airline	Source	Destination	Total_Stops	Additional_Info	is_train	Date	Month
0	IndiGo	Banglore	New Delhi	0.0	No info	1	24	3
1	Air India	Kolkata	Banglore	2.0	No info	1	1	5
2	Jet Airways	Delhi	Cochin	2.0	No info	1	9	6
3	IndiGo	Kolkata	Banglore	1.0	No info	1	12	5
4	IndiGo	Banglore	New Delhi	1.0	No info	1	1	3

In [20]:
```
#filling null values with mode
df['Total_Stops'].fillna(df['Total_Stops'].mode()[0],inplace=True)
```

```
C:\Users\jishn\AppData\Local\Temp\ipykernel_26608\3100553634.py:2: FutureWarning:
A value is trying to be set on a copy of a DataFrame or Series through chained as
signment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work becau
se the intermediate object on which we are setting values always behaves as a cop
y.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.meth
od({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to pe
rform the operation inplace on the original object.


  df['Total_Stops'].fillna(df['Total_Stops'].mode()[0],inplace=True)
```

In [21]:
```
df.isnull().sum()
```

Out[21]:
```
Airline            0
Source             0
Destination        0
Total_Stops        0
Additional_Info    0
is_train           0
Date               0
Month              0
Year               0
Dep_hour           0
Dep_minute         0
Arrival_hour       0
Arrival_minute     0
Duration_minutes   0
dtype: int64
```

In [22]:
```
#printing unique values for the columns
print(df['Airline'].unique(),df['Source'].unique(),df['Destination'].unique())
```

```
['IndiGo' 'Air India' 'Jet Airways' 'SpiceJet' 'Multiple carriers' 'GoAir'
 'Vistara' 'Air Asia' 'Vistara Premium economy' 'Jet Airways Business'
 'Multiple carriers Premium economy' 'Trujet'] ['Banglore' 'Kolkata' 'Delhi' 'Che
nnai' 'Mumbai'] ['New Delhi' 'Banglore' 'Cochin' 'Kolkata' 'Delhi' 'Hyderabad']
```

In [23]:
```
#encoding categorical columns
%pip install scikit-learn
```

```
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import OneHotEncoder
import pandas as pd

le = LabelEncoder()
df['Additional_Info'] = le.fit_transform(df['Additional_Info'])

cols_to_encode = ['Airline', 'Source', 'Destination']

ohe = OneHotEncoder(sparse_output=False, drop=None)
encoded = ohe.fit_transform(df[cols_to_encode])
encoded_cols = ohe.get_feature_names_out(cols_to_encode)
encoded_df = pd.DataFrame(encoded, columns=encoded_cols, index=df.index)
df = pd.concat([df.drop(cols_to_encode, axis=1), encoded_df], axis=1)

df.head()
```

```
Requirement already satisfied: scikit-learn in c:\users\jishn\anaconda3\lib\site-
packages (1.5.1)
Requirement already satisfied: numpy>=1.19.5 in c:\users\jishn\anaconda3\lib\site
-packages (from scikit-learn) (1.26.4)
Requirement already satisfied: scipy>=1.6.0 in c:\users\jishn\anaconda3\lib\site-
packages (from scikit-learn) (1.13.1)
Requirement already satisfied: joblib>=1.2.0 in c:\users\jishn\anaconda3\lib\site
-packages (from scikit-learn) (1.4.2)
Requirement already satisfied: threadpoolctl>=3.1.0 in c:\users\jishn\anaconda3\l
ib\site-packages (from scikit-learn) (3.5.0)
Note: you may need to restart the kernel to use updated packages.
```

Out[23]:

	Total_Stops	Additional_Info	is_train	Date	Month	Year	Dep_hour	Dep_minute	A
0	0.0	0	1	24	3	2019	22	20	
1	2.0	0	1	1	5	2019	5	50	
2	2.0	0	1	9	6	2019	9	25	
3	1.0	0	1	12	5	2019	18	5	
4	1.0	0	1	1	3	2019	16	50	

5 rows × 34 columns

In [24]:

```
#scale numeric features to mean 0 and std dev 1
from sklearn.preprocessing import StandardScaler
import pandas as pd

#splitting dataset back to training and testing using flag
train_clean = df[df['is_train'] == 1].drop('is_train', axis=1)
test_clean = df[df['is_train'] == 0].drop('is_train', axis=1)

scaler = StandardScaler()
X_train = scaler.fit_transform(train_clean)
X_test = scaler.transform(test_clean)

X_train_df = pd.DataFrame(X_train, columns=train_clean.columns)
X_test_df = pd.DataFrame(X_test, columns=test_clean.columns)
```

In [25]:
```
X_train_df.head()
```

Out[25]:

	Total_Stops	Additional_Info	Date	Month	Year	Dep_hour	Dep_minute	Arri
0	-1.220744	-0.529309	1.237383	-1.467490	0.0	1.654259	-0.235050	-
1	1.741483	-0.529309	-1.475239	0.250276	0.0	-1.303095	1.363492	-
2	1.741483	-0.529309	-0.531719	1.109160	0.0	-0.607247	0.031373	-
3	0.260370	-0.529309	-0.177898	0.250276	0.0	0.958411	-1.034321	
4	0.260370	-0.529309	-1.475239	-1.467490	0.0	0.610487	1.363492	

5 rows × 33 columns

In [26]:
```
X_test_df.head()
```

Out[26]:

	Total_Stops	Additional_Info	Date	Month	Year	Dep_hour	Dep_minute	Arriv
0	0.260370	-0.529309	-0.885539	1.109160	0.0	0.784449	0.297797	-1
1	0.260370	-0.529309	-0.177898	0.250276	0.0	-1.129133	-0.235050	-0
2	0.260370	1.889256	0.883563	0.250276	0.0	1.132373	-0.501474	0
3	0.260370	-0.529309	0.883563	0.250276	0.0	-0.781209	-1.300745	1
4	-1.220744	-0.529309	1.237383	1.109160	0.0	1.828221	1.629915	-1

5 rows × 33 columns

In [27]:
```
X_train_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10683 entries, 0 to 10682
Data columns (total 33 columns):
 #   Column                                     Non-Null Count  Dtype
---  ------                                     --------------  -----
 0   Total_Stops                                10683 non-null  float64
 1   Additional_Info                            10683 non-null  float64
 2   Date                                       10683 non-null  float64
 3   Month                                      10683 non-null  float64
 4   Year                                       10683 non-null  float64
 5   Dep_hour                                   10683 non-null  float64
 6   Dep_minute                                 10683 non-null  float64
 7   Arrival_hour                               10683 non-null  float64
 8   Arrival_minute                             10683 non-null  float64
 9   Duration_minutes                           10683 non-null  float64
 10  Airline_Air Asia                           10683 non-null  float64
 11  Airline_Air India                          10683 non-null  float64
 12  Airline_GoAir                              10683 non-null  float64
 13  Airline_IndiGo                             10683 non-null  float64
 14  Airline_Jet Airways                        10683 non-null  float64
 15  Airline_Jet Airways Business               10683 non-null  float64
 16  Airline_Multiple carriers                  10683 non-null  float64
 17  Airline_Multiple carriers Premium economy  10683 non-null  float64
 18  Airline_SpiceJet                           10683 non-null  float64
 19  Airline_Trujet                             10683 non-null  float64
 20  Airline_Vistara                            10683 non-null  float64
 21  Airline_Vistara Premium economy            10683 non-null  float64
 22  Source_Banglore                            10683 non-null  float64
 23  Source_Chennai                             10683 non-null  float64
 24  Source_Delhi                               10683 non-null  float64
 25  Source_Kolkata                             10683 non-null  float64
 26  Source_Mumbai                              10683 non-null  float64
 27  Destination_Banglore                       10683 non-null  float64
 28  Destination_Cochin                         10683 non-null  float64
 29  Destination_Delhi                          10683 non-null  float64
 30  Destination_Hyderabad                      10683 non-null  float64
 31  Destination_Kolkata                        10683 non-null  float64
 32  Destination_New Delhi                      10683 non-null  float64
dtypes: float64(33)
memory usage: 2.7 MB
```

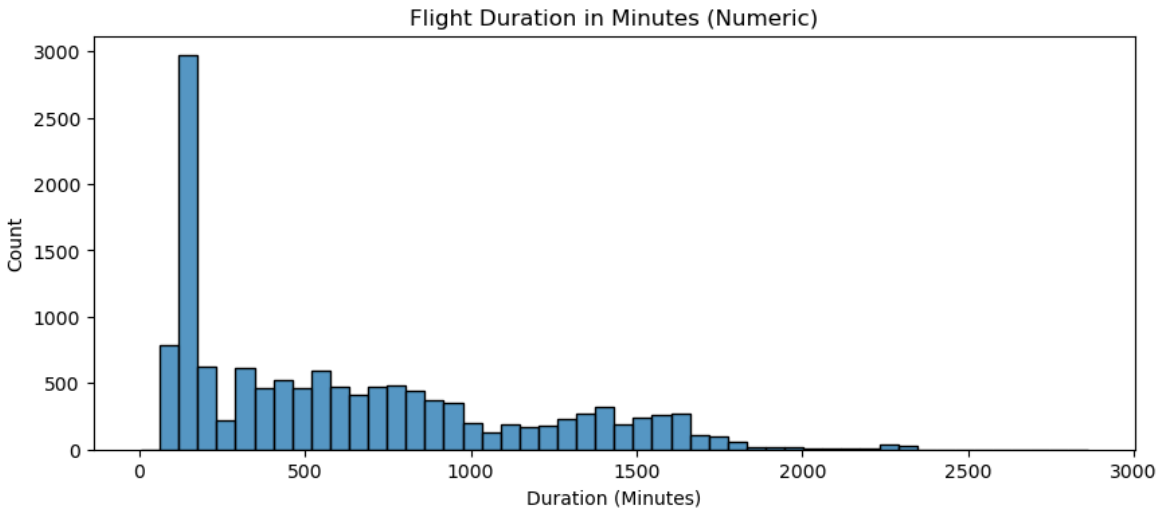
In [28]:
```
train_clean_scaled = pd.DataFrame(X_train, columns=train_clean.columns)
train_clean_scaled['Price'] = y.values

test_clean_scaled = pd.DataFrame(X_test, columns=test_clean.columns)

train_clean_scaled.to_excel("Cleaned_Train.xlsx", index=False)
test_clean_scaled.to_excel("Cleaned_Test.xlsx", index=False)

print("Cleaned and scaled train/test datasets saved successfully!!")
```

```
Cleaned and scaled train/test datasets saved successfully!!
```

Plot for Flight Distribution After Conversion to Minutes

In [32]:
```
plt.figure(figsize=(10,4))
sns.histplot(df['Duration_minutes'], bins=50)
plt.title("Flight Duration in Minutes (Numeric)")
plt.xlabel("Duration (Minutes)")
```

```
plt.ylabel("Count")
plt.show()
```



Plot for Total Stops Distribution after Encoding

In [30]:
```
plt.figure(figsize=(10,4))
sns.countplot(x='Total_Stops', data=df)
plt.title("Total Stops Distribution After Transformation")
plt.xlabel("Stops")
plt.ylabel("Count")
plt.show()
```