[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import scipy.stats
from scipy.stats import shapiro, chi2, normaltest, kstest, zscore
from sklearn.model_selection import train_test_split
```

[2]:
```
# create a function for calculating statistical analysis of numerical column
from scipy.stats import skew, mode
def cal_stats(data, column_name):
    column = data[column_name]
    # Calculating statistics
    stats = {
        'Mean': column.mean(),
        'Median': column.median(),
        'Mode': mode(column)[0],
        'Min': column.min(),
        'Max': column.max(),
        'Range': column.max() - column.min(),
        'Variance': column.var(),
        'Standard Deviation': column.std(),
        'Skewness': skew(column),
    }
    df1=pd.DataFrame.from_dict(stats,orient='index',columns=['Statistics'])
    return df1
```

[3]:
```
# create a function for calculate analysis of categorical columns
def analyze_categorical(data, column):
    # Frequency distribution
    f = data[column].value_counts()
    p = data[column].value_counts(normalize=True) * 100

    # Combine frequency and percentage
    analysis = pd.DataFrame({"Frequency": f, "Percentage": p})
    print(f"Analysis of {column}:")
    print(analysis)

    # Bar plot
    f.plot(kind="bar")
    plt.title(f"Frequency Distribution of {column}")
    plt.xlabel(column)
    plt.show()

    # Pie chart
    p.plot(kind="pie", autopct="%.1f%%")
    plt.title(f"Percentage Distribution of {column}")
    plt.show()
```

[4]:
```
# data gathering
train_data = pd.read_csv('Data_Train.csv')
test_data = pd.read_csv('Test_set.csv')
```

[5]:
```
train_data.head()
```

[5]:

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Duration	Total_Stops	Additional_Info	Price
0	IndiGo	24/03/2019	Banglore	New Delhi	BLR ? DEL	22:20	01:10 22 Mar	2h 50m	non-stop	No info	3897
1	Air India	1/05/2019	Kolkata	Banglore	CCU ? IXR ? BBI ? BLR	05:50	13:15	7h 25m	2 stops	No info	7662
2	Jet Airways	9/06/2019	Delhi	Cochin	DEL ? LKO ? BOM ? COK	09:25	04:25 10 Jun	19h	2 stops	No info	13882
3	IndiGo	12/05/2019	Kolkata	Banglore	CCU ? NAG ? BLR	18:05	23:30	5h 25m	1 stop	No info	6218
4	IndiGo	01/03/2019	Banglore	New Delhi	BLR ? NAG ? DEL	16:50	21:35	4h 45m	1 stop	No info	13302

[6]:
```
test_data.head()
```

[6]:

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Duration	Total_Stops	Additional_Info
0	Jet Airways	6/06/2019	Delhi	Cochin	DEL ? BOM ? COK	17:30	04:25 07 Jun	10h 55m	1 stop	No info
1	IndiGo	12/05/2019	Kolkata	Banglore	CCU ? MAA ? BLR	06:20	10:20	4h	1 stop	No info
2	Jet Airways	21/05/2019	Delhi	Cochin	DEL ? BOM ? COK	19:15	19:00 22 May	23h 45m	1 stop	In-flight meal not included
3	Multiple carriers	21/05/2019	Delhi	Cochin	DEL ? BOM ? COK	08:00	21:00	13h	1 stop	No info
4	Air Asia	24/06/2019	Banglore	Delhi	BLR ? DEL	23:55	02:45 25 Jun	2h 50m	non-stop	No info

[7]:
```
train_data.columns
```

[7]:
```
Index(['Airline', 'Date_of_Journey', 'Source', 'Destination', 'Route',
       'Dep_Time', 'Arrival_Time', 'Duration', 'Total_Stops',
       'Additional_Info', 'Price'],
      dtype='object')
```

[8]:
```
test_data.columns
```

[8]:
```
Index(['Airline', 'Date_of_Journey', 'Source', 'Destination', 'Route',
       'Dep_Time', 'Arrival_Time', 'Duration', 'Total_Stops',
       'Additional_Info'],
      dtype='object')
```

[9]:
```
# There is a mismatch because the Price column exists only in the training dataset.
```

[10]:
```
print(train_data.shape)
test_data.shape
```

```
(10683, 11)
```

[10]:
```
(2671, 10)
```

[11]:
```
print(train_data.size)
test_data.size
```

```
117513
```

[11]:
```
26710
```

[12]:
```
print(train_data.info())
print('*'*80)
test_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10683 entries, 0 to 10682
Data columns (total 11 columns):
 #   Column           Non-Null Count  Dtype 
---  ------           --------------  ----- 
 0   Airline          10683 non-null  object
 1   Date_of_Journey  10683 non-null  object
 2   Source           10683 non-null  object
 3   Destination      10683 non-null  object
 4   Route            10682 non-null  object
 5   Dep_Time         10683 non-null  object
 6   Arrival_Time     10683 non-null  object
 7   Duration         10683 non-null  object
 8   Total_Stops      10682 non-null  object
 9   Additional_Info  10683 non-null  object
 10  Price            10683 non-null  int64 
dtypes: int64(1), object(10)
memory usage: 918.2+ KB
None
********************************************************************************
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2671 entries, 0 to 2670
Data columns (total 10 columns):
 #   Column           Non-Null Count  Dtype 
---  ------           --------------  ----- 
 0   Airline          2671 non-null   object
 1   Date_of_Journey  2671 non-null   object
 2   Source           2671 non-null   object
 3   Destination      2671 non-null   object
 4   Route            2671 non-null   object
 5   Dep_Time         2671 non-null   object
 6   Arrival_Time     2671 non-null   object
 7   Duration         2671 non-null   object
 8   Total_Stops      2671 non-null   object
 9   Additional_Info  2671 non-null   object
dtypes: object(10)
memory usage: 208.8+ KB
```

[13]:
```
print(train_data.describe())
test_data.describe()
```

```
              Price
count  10683.000000
mean    9087.064121
std     4611.359167
min     1759.000000
25%     5277.000000
50%     8372.000000
75%    12373.000000
max    79512.000000
```

[13]:

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Duration	Total_Stops	Additional_Info
count	2671	2671	2671	2671	2671	2671	2671	2671	2671	2671
unique	11	44	5	6	100	199	704	320	5	6
top	Jet Airways	9/05/2019	Delhi	Cochin	DEL ? BOM ? COK	10:00	19:00	2h 50m	1 stop	No info
freq	897	144	1145	1145	624	62	113	122	1431	2148

[ ]:
```

```

[14]:
```
print(train_data.dtypes)
print('*'*80)
test_data.dtypes
```

```
Airline            object
Date_of_Journey    object
Source             object
Destination        object
Route              object
Dep_Time           object
Arrival_Time       object
Duration           object
Total_Stops        object
Additional_Info    object
Price               int64
dtype: object
********************************************************************************
```

[14]:
```
Airline            object
Date_of_Journey    object
Source             object
Destination        object
Route              object
Dep_Time           object
Arrival_Time       object
Duration           object
Total_Stops        object
Additional_Info    object
dtype: object
```

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

```
Airline            0
Date_of_Journey    0
Source             0
Destination        0
Route              1
Dep_Time           0
Arrival_Time       0
Duration           0
Total_Stops        1
Additional_Info    0
Price              0
dtype: int64
```

[16]:
```
print(test_data.isnull().sum())
```

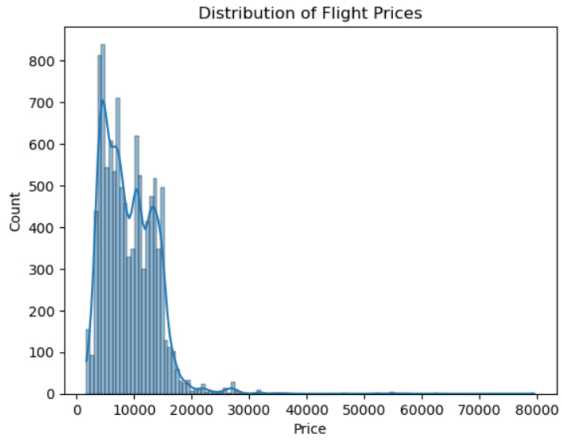
```
Airline           0
Date_of_Journey   0
Source            0
Destination       0
Route             0
Dep_Time          0
Arrival_Time      0
Duration          0
Total_Stops       0
Additional_Info   0
dtype: int64
```

[17]:
```
train_data=train_data.fillna(train_data['Route'].mode()[0])
```

[18]:
```
train_data=train_data.fillna(train_data['Total_Stops'].mode()[0])
```

Analysis of numerical columns (only one "Price " also this is Target column)

[20]:
```
# Visualize distribution of the target column in training data
sns.histplot(train_data['Price'], kde=True)
plt.title("Distribution of Flight Prices")
plt.show()
```



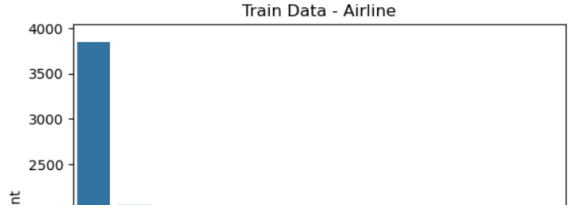
[21]:
```
cal_stats(train_data,'Price')
```

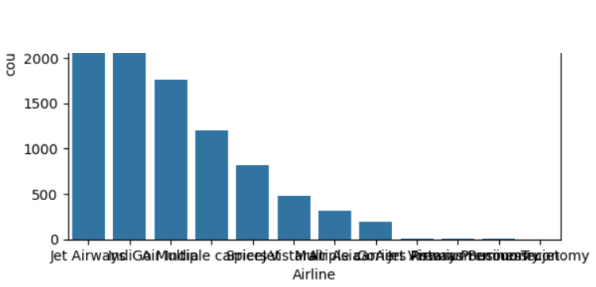
[21]:

	Statistics
Mean	9.087064e+03
Median	8.372000e+03
Mode	1.026200e+04
Min	1.759000e+03
Max	7.951200e+04
Range	7.775300e+04
Variance	2.126463e+07
Standard Deviation	4.611359e+03
Skewness	1.812298e+00

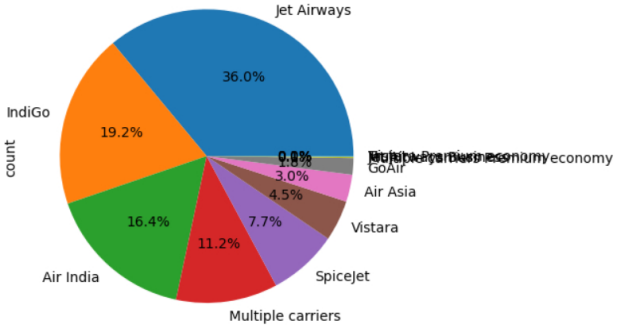
Analysis of categorical columns

[23]:
```
# visualized categorical columns

categorical_cols = ['Airline', 'Source', 'Destination', 'Route', 'Total_Stops', 'Additional_Info']

for col in categorical_cols:
    # Bar plot for train_data
    sns.countplot(data=train_data, x=col, order=train_data[col].value_counts().index)
    plt.title(f"Train Data - {col}")
    plt.show()

    # Pie chart for train_data
    train_data[col].value_counts().plot.pie(autopct='%1.1f%%')
    plt.title(f"Train Data - {col} (Pie Chart)")
    plt.show()

    # Bar plot for test_data
    sns.countplot(data=test_data, x=col, order=test_data[col].value_counts().index)
    plt.title(f"Test Data - {col}")
    plt.show()

    # Pie chart for test_data
    test_data[col].value_counts().plot.pie(autopct='%1.1f%%')
    plt.title(f"Test Data - {col} (Pie Chart)")
    plt.show()
```

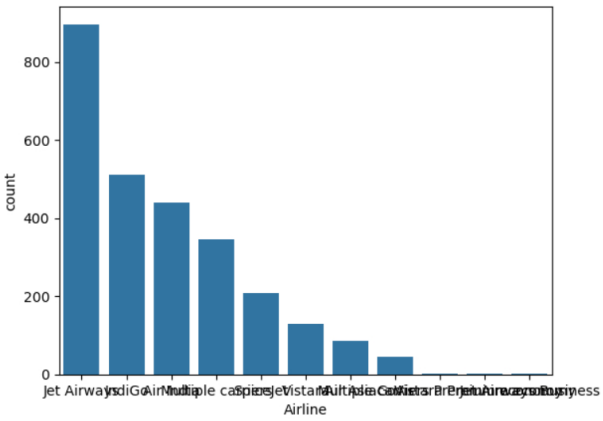




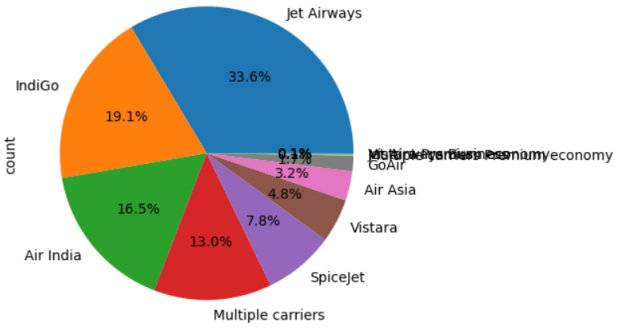
Train Data - Airline (Pie Chart)



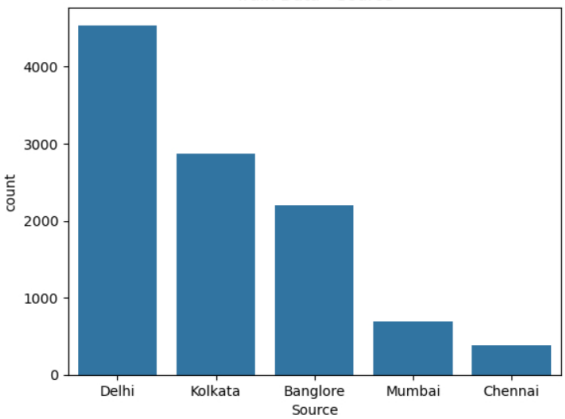
Test Data - Airline



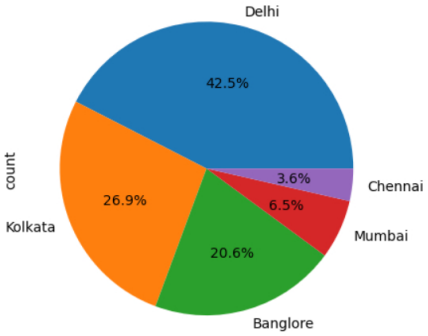
Test Data - Airline (Pie Chart)

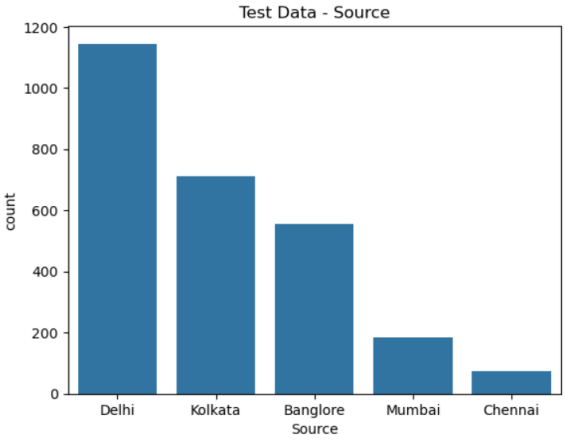


Train Data - Source

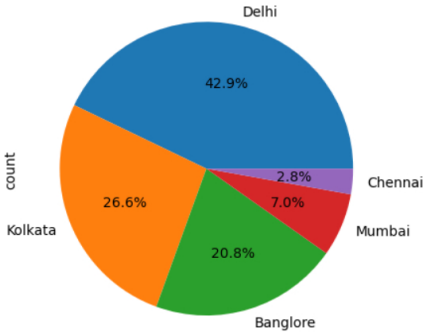


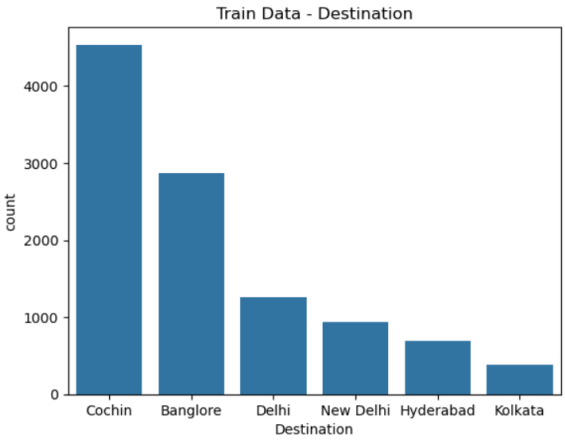
Train Data - Source (Pie Chart)



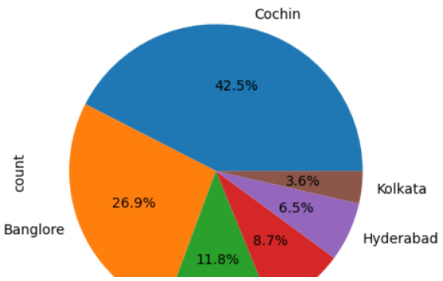


Test Data - Source (Pie Chart)

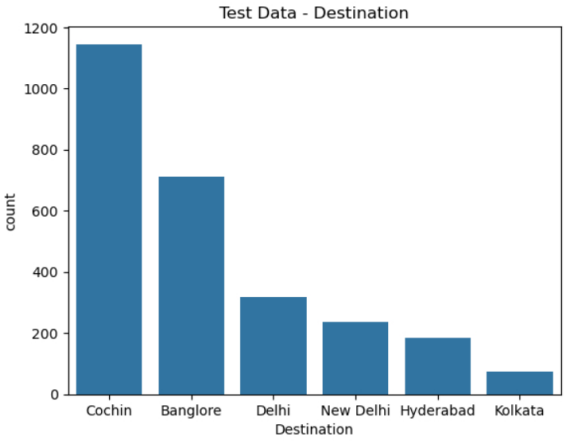


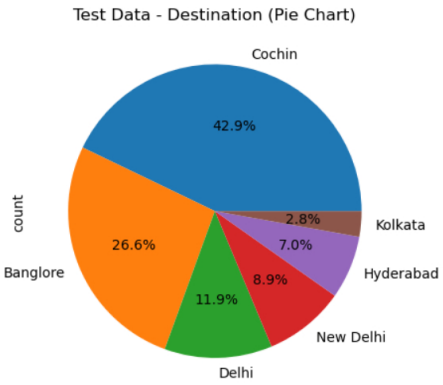


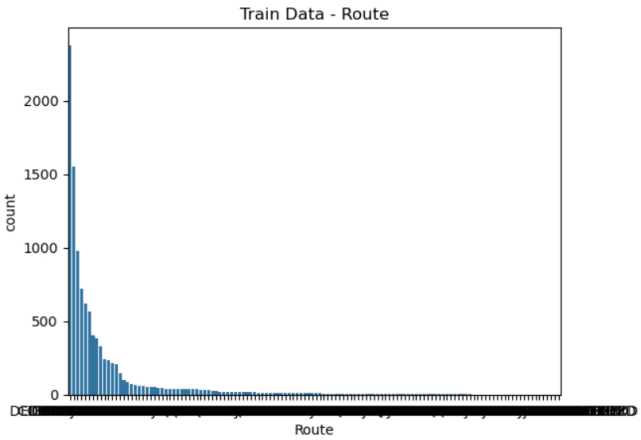
Train Data - Destination (Pie Chart)

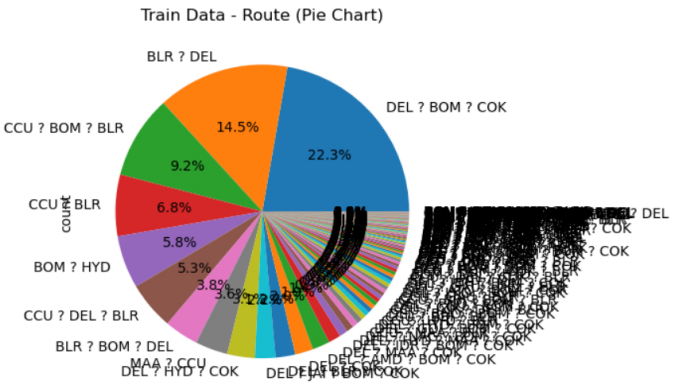


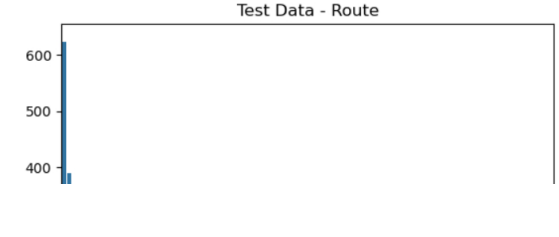

New Delhi
Delhi

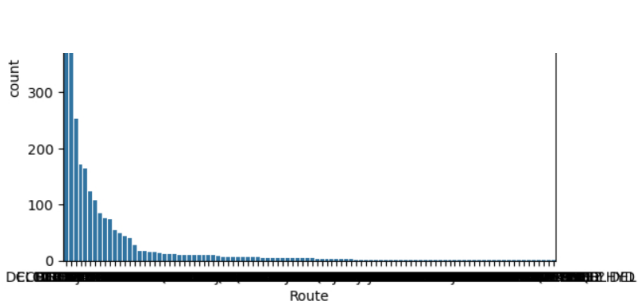

Test Data - Destination
count
Destination
Cochin
Banglore
Delhi
New Delhi
Hyderabad
Kolkata


Test Data - Destination (Pie Chart)
Cochin
42.9%
Kolkata
2.8%
Banglore
26.6%
Hyderabad
7.0%
New Delhi
8.9%
Delhi
11.9%

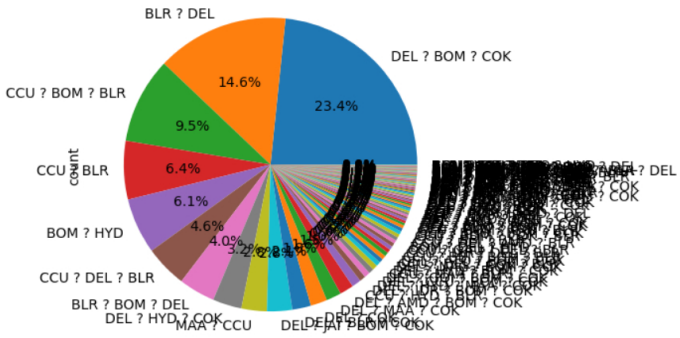

Train Data - Route
count
Route

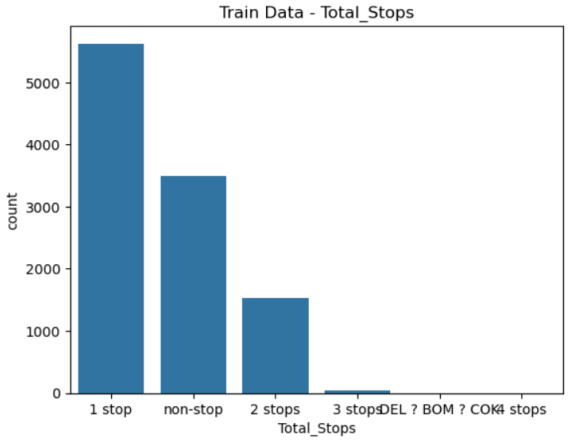

Train Data - Route (Pie Chart)
BLR ? DEL
DEL ? BOM ? COK
CCU ? BOM ? BLR
14.5%
22.3%
9.2%
CCU ? BLR
6.8%
5.8%
BOM ? HYD
5.3%
3.8%
CCU ? DEL ? BLR
BLR ? BOM ? DEL
DEL ? HYD ? COK


Test Data - Route

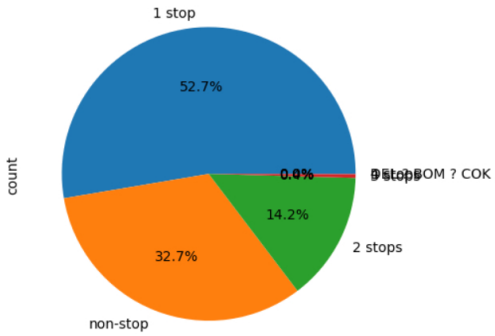


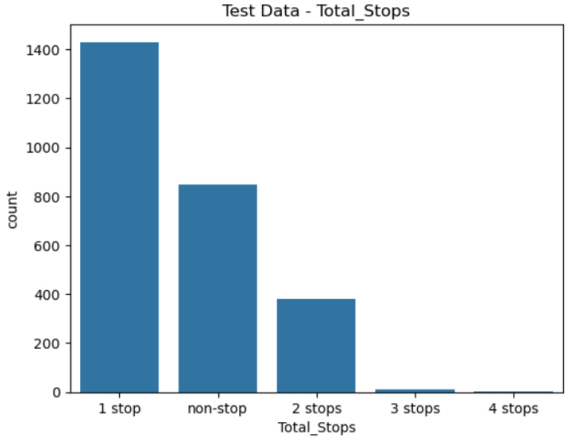
Test Data - Route (Pie Chart)



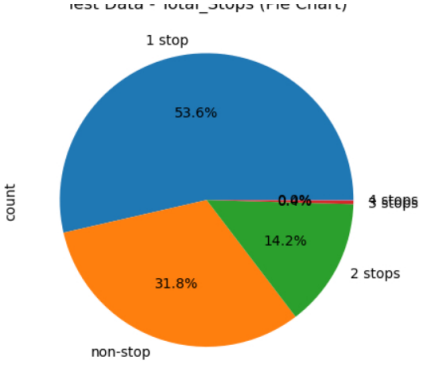


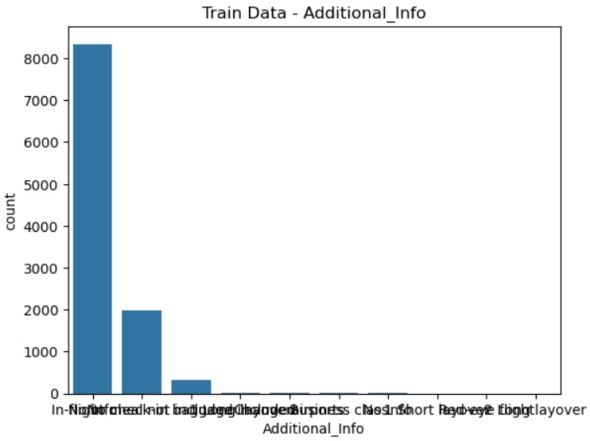
Train Data - Total_Stops (Pie Chart)



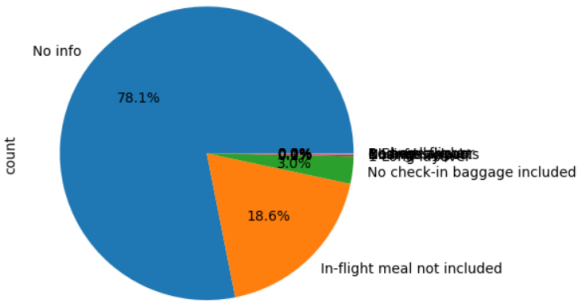


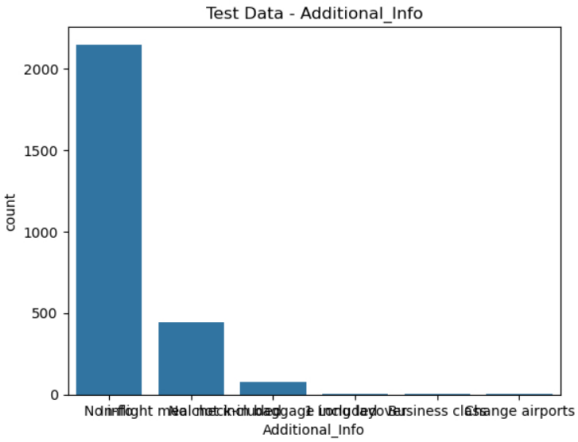
Test Data - Total_Stops (Pie Chart)



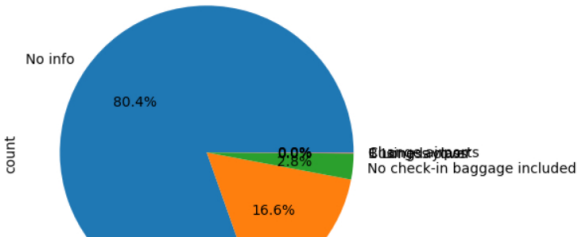


Train Data - Additional_Info (Pie Chart)



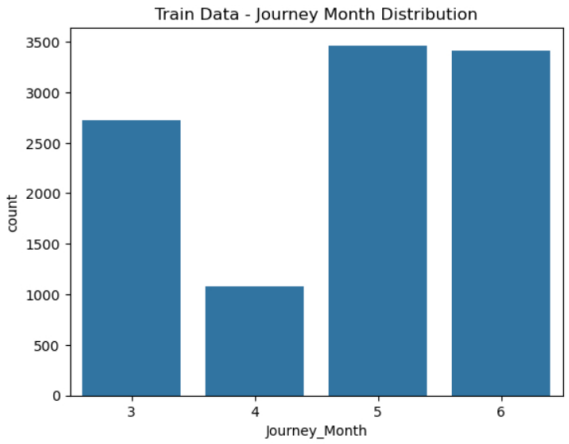


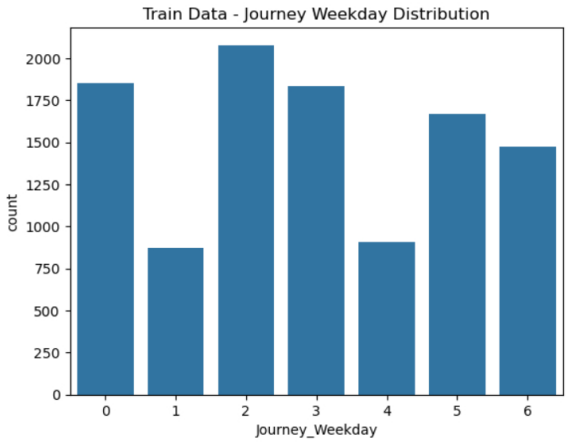
Test Data - Additional_Info (Pie Chart)



Analysis of Date_of_Journey columns

[25]:
```
# Convert to datetime
train_data['Date_of_Journey'] = pd.to_datetime(train_data['Date_of_Journey'], format='%d/%m/%Y')
test_data['Date_of_Journey'] = pd.to_datetime(test_data['Date_of_Journey'], format='%d/%m/%Y')

# Extract day, month, and weekday
train_data['Journey_Day'] = train_data['Date_of_Journey'].dt.day
train_data['Journey_Month'] = train_data['Date_of_Journey'].dt.month
train_data['Journey_Weekday'] = train_data['Date_of_Journey'].dt.weekday

test_data['Journey_Day'] = test_data['Date_of_Journey'].dt.day
test_data['Journey_Month'] = test_data['Date_of_Journey'].dt.month
test_data['Journey_Weekday'] = test_data['Date_of_Journey'].dt.weekday

# Plot distributions
sns.countplot(x='Journey_Month', data=train_data)
plt.title('Train Data - Journey Month Distribution')
plt.show()

sns.countplot(x='Journey_Weekday', data=train_data)
plt.title('Train Data - Journey Weekday Distribution')
plt.show()
```



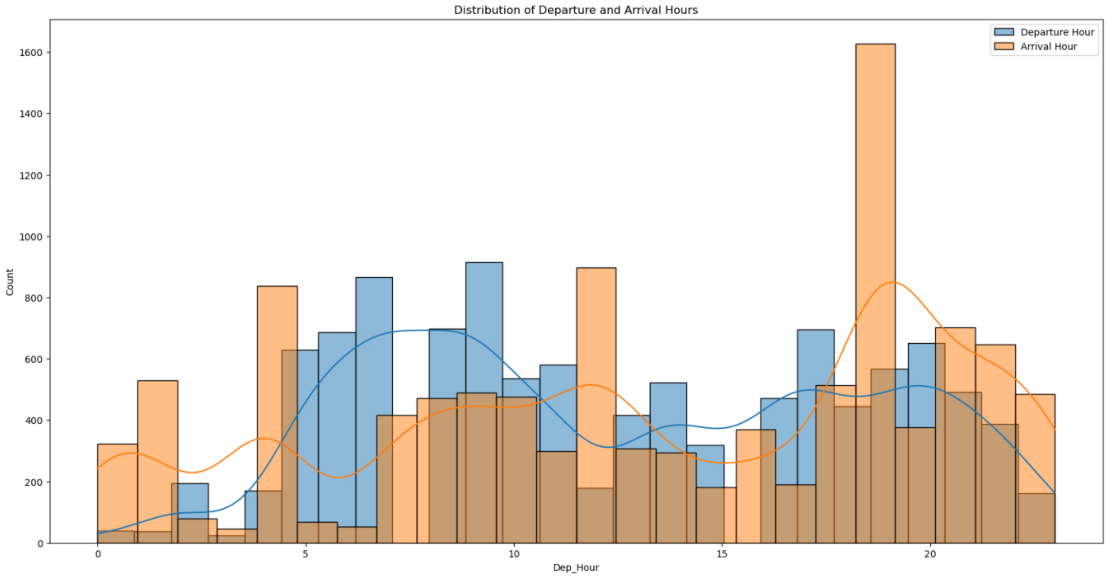


Analysis of Dep_Time and Arrival_Time columns

[27]:
```
# Convert to datetime
train_data['Dep_Hour'] = pd.to_datetime(train_data['Dep_Time']).dt.hour
train_data['Dep_Minute'] = pd.to_datetime(train_data['Dep_Time']).dt.minute
train_data['Arrival_Hour'] = pd.to_datetime(train_data['Arrival_Time']).dt.hour
train_data['Arrival_Minute'] = pd.to_datetime(train_data['Arrival_Time']).dt.minute

test_data['Dep_Hour'] = pd.to_datetime(test_data['Dep_Time']).dt.hour
test_data['Dep_Minute'] = pd.to_datetime(test_data['Dep_Time']).dt.minute
test_data['Arrival_Hour'] = pd.to_datetime(test_data['Arrival_Time']).dt.hour
test_data['Arrival_Minute'] = pd.to_datetime(test_data['Arrival_Time']).dt.minute

# Plot distribution of departure and arrival hours
plt.figure(figsize=(20, 10))
sns.histplot(train_data['Dep_Hour'],  kde=True, label='Departure Hour')
sns.histplot(train_data['Arrival_Hour'],  kde=True, label='Arrival Hour')
plt.legend()
plt.title('Distribution of Departure and Arrival Hours')
plt.show()
```

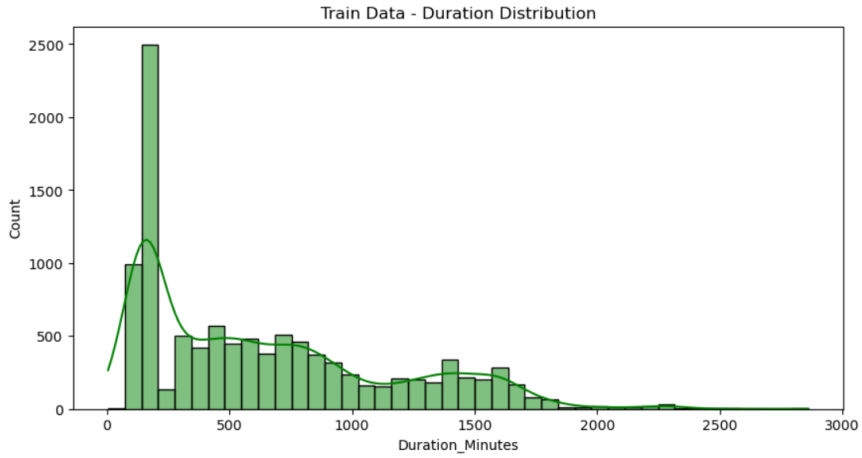
```
C:\Users\icon\AppData\Local\Temp\ipykernel_9800\1520994945.py:2: UserWarning: Could not infer format, so each element will be parsed individually, falling
back to `dateutil`. To ensure parsing is consistent and as-expected, please specify a format.
  train_data['Dep_Hour'] = pd.to_datetime(train_data['Dep_Time']).dt.hour
```

```
C:\Users\icon\AppData\Local\Temp\ipykernel_9800\1520994945.py:3: UserWarning: Could not infer format, so each element will be parsed individually, falling back to `dateutil`. To ensure parsing is consistent and as-expected, please specify a format.
  train_data['Dep_Minute'] = pd.to_datetime(train_data['Dep_Time']).dt.minute
C:\Users\icon\AppData\Local\Temp\ipykernel_9800\1520994945.py:4: UserWarning: Could not infer format, so each element will be parsed individually, falling back to `dateutil`. To ensure parsing is consistent and as-expected, please specify a format.
  train_data['Arrival_Hour'] = pd.to_datetime(train_data['Arrival_Time']).dt.hour
C:\Users\icon\AppData\Local\Temp\ipykernel_9800\1520994945.py:5: UserWarning: Could not infer format, so each element will be parsed individually, falling back to `dateutil`. To ensure parsing is consistent and as-expected, please specify a format.
  train_data['Arrival_Minute'] = pd.to_datetime(train_data['Arrival_Time']).dt.minute
C:\Users\icon\AppData\Local\Temp\ipykernel_9800\1520994945.py:7: UserWarning: Could not infer format, so each element will be parsed individually, falling back to `dateutil`. To ensure parsing is consistent and as-expected, please specify a format.
  test_data['Dep_Hour'] = pd.to_datetime(test_data['Dep_Time']).dt.hour
C:\Users\icon\AppData\Local\Temp\ipykernel_9800\1520994945.py:8: UserWarning: Could not infer format, so each element will be parsed individually, falling back to `dateutil`. To ensure parsing is consistent and as-expected, please specify a format.
  test_data['Dep_Minute'] = pd.to_datetime(test_data['Dep_Time']).dt.minute
C:\Users\icon\AppData\Local\Temp\ipykernel_9800\1520994945.py:9: UserWarning: Could not infer format, so each element will be parsed individually, falling back to `dateutil`. To ensure parsing is consistent and as-expected, please specify a format.
  test_data['Arrival_Hour'] = pd.to_datetime(test_data['Arrival_Time']).dt.hour
C:\Users\icon\AppData\Local\Temp\ipykernel_9800\1520994945.py:10: UserWarning: Could not infer format, so each element will be parsed individually, falling back to `dateutil`. To ensure parsing is consistent and as-expected, please specify a format.
  test_data['Arrival_Minute'] = pd.to_datetime(test_data['Arrival_Time']).dt.minute
```



Analysis of Duration column

[29]:
```python
# Convert duration to minutes
def duration_to_minutes(duration):
    hours = 0
    minutes = 0
    if 'h' in duration:
        hours = int(duration.split('h')[0])
    if 'm' in duration:
        minutes = int(duration.split('h')[-1].replace('m', '').strip())
    return hours * 60 + minutes

train_data['Duration_Minutes'] = train_data['Duration'].apply(duration_to_minutes)
test_data['Duration_Minutes'] = test_data['Duration'].apply(duration_to_minutes)

# Plot duration distribution
plt.figure(figsize=(10, 5))
sns.histplot(train_data['Duration_Minutes'], kde=True, color='green')
plt.title('Train Data - Duration Distribution')
plt.show()
```



[30]:
```python
train_data
```

[30]:

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Duration	Total_Stops	Additional_Info	Price	Journey_Day	Journey_
0	IndiGo	2019-03-24	Banglore	New Delhi	BLR ? DEL	22:20	01:10 22 Mar	2h 50m	non-stop	No info	3897	24	
	Air				CCU ? IXR ?								

1	India	2019-05-01	Kolkata	Banglore	BBI ? BLR	05:50	13:15	7h 25m	2 stops	No info	7662	1
2	Jet Airways	2019-06-09	Delhi	Cochin	DEL ? LKO ? BOM ? COK	09:25	04:25 10 Jun	19h	2 stops	No info	13882	9
3	IndiGo	2019-05-12	Kolkata	Banglore	CCU ? NAG ? BLR	18:05	23:30	5h 25m	1 stop	No info	6218	12
4	IndiGo	2019-03-01	Banglore	New Delhi	BLR ? NAG ? DEL	16:50	21:35	4h 45m	1 stop	No info	13302	1
...	...	...	...	...	...	...	...	...	...	...	...	...
10678	Air Asia	2019-04-09	Kolkata	Banglore	CCU ? BLR	19:55	22:25	2h 30m	non-stop	No info	4107	9
10679	Air India	2019-04-27	Kolkata	Banglore	CCU ? BLR	20:45	23:20	2h 35m	non-stop	No info	4145	27
10680	Jet Airways	2019-04-27	Banglore	Delhi	BLR ? DEL	08:20	11:20	3h	non-stop	No info	7229	27
10681	Vistara	2019-03-01	Banglore	New Delhi	BLR ? DEL	11:30	14:10	2h 40m	non-stop	No info	12648	1
10682	Air India	2019-05-09	Delhi	Cochin	DEL ? GOI ? BOM ? COK	10:55	19:15	8h 20m	2 stops	No info	11753	9

10683 rows × 19 columns

Encoding

Airline column

[33]:
```
train_data['Airline'].unique()
```

[33]:
```
array(['IndiGo', 'Air India', 'Jet Airways', 'SpiceJet',
       'Multiple carriers', 'GoAir', 'Vistara', 'Air Asia',
       'Vistara Premium economy', 'Jet Airways Business',
       'Multiple carriers Premium economy', 'Trujet'], dtype=object)
```

[34]:
```
train_data['Airline']=train_data['Airline'].replace({'IndiGo':0, 'Air India':1, 'Jet Airways':2, 'SpiceJet':3,
       'Multiple carriers':4, 'GoAir':5, 'Vistara':6, 'Air Asia':7,
       'Vistara Premium economy':8, 'Jet Airways Business':9,
       'Multiple carriers Premium economy':10, 'Trujet':11})
```

```
C:\Users\icon\AppData\Local\Temp\ipykernel_9800\137208105.py:1: FutureWarning: Downcasting behavior in `replace` is deprecated and will be removed in a fut
ure version. To retain the old behavior, explicitly call `result.infer_objects(copy=False)`. To opt-in to the future behavior, set `pd.set_option('future.n
o_silent_downcasting', True)`
  train_data['Airline']=train_data['Airline'].replace({'IndiGo':0, 'Air India':1, 'Jet Airways':2, 'SpiceJet':3,
```

[35]:
```
test_data['Airline']=test_data['Airline'].replace({'IndiGo':0, 'Air India':1, 'Jet Airways':2, 'SpiceJet':3,
       'Multiple carriers':4, 'GoAir':5, 'Vistara':6, 'Air Asia':7,
       'Vistara Premium economy':8, 'Jet Airways Business':9,
       'Multiple carriers Premium economy':10, 'Trujet':11})
```

```
C:\Users\icon\AppData\Local\Temp\ipykernel_9800\1349144151.py:1: FutureWarning: Downcasting behavior in `replace` is deprecated and will be removed in a fu
ture version. To retain the old behavior, explicitly call `result.infer_objects(copy=False)`. To opt-in to the future behavior, set `pd.set_option('future.
no_silent_downcasting', True)`
  test_data['Airline']=test_data['Airline'].replace({'IndiGo':0, 'Air India':1, 'Jet Airways':2, 'SpiceJet':3,
```

[36]:
```
train_data.drop(columns=['Date_of_Journey'], inplace=True)
test_data.drop(columns=['Date_of_Journey'], inplace=True)
```

Source column

[38]:
```
train_data['Source'].unique()
```

[38]:
```
array(['Banglore', 'Kolkata', 'Delhi', 'Chennai', 'Mumbai'], dtype=object)
```

[39]:
```
train_data['Source']=train_data['Source'].replace({'Banglore':0, 'Kolkata':1, 'Delhi':2, 'Chennai':3, 'Mumbai':4})
```

```
C:\Users\icon\AppData\Local\Temp\ipykernel_9800\3032934907.py:1: FutureWarning: Downcasting behavior in `replace` is deprecated and will be removed in a fu
ture version. To retain the old behavior, explicitly call `result.infer_objects(copy=False)`. To opt-in to the future behavior, set `pd.set_option('future.
no_silent_downcasting', True)`
  train_data['Source']=train_data['Source'].replace({'Banglore':0, 'Kolkata':1, 'Delhi':2, 'Chennai':3, 'Mumbai':4})
```

[40]:
```
test_data['Source']=test_data['Source'].replace({'Banglore':0, 'Kolkata':1, 'Delhi':2, 'Chennai':3, 'Mumbai':4})
```

```
C:\Users\icon\AppData\Local\Temp\ipykernel_9800\1931568196.py:1: FutureWarning: Downcasting behavior in `replace` is deprecated and will be removed in a fu
ture version. To retain the old behavior, explicitly call `result.infer_objects(copy=False)`. To opt-in to the future behavior, set `pd.set_option('future.
no_silent_downcasting', True)`
  test_data['Source']=test_data['Source'].replace({'Banglore':0, 'Kolkata':1, 'Delhi':2, 'Chennai':3, 'Mumbai':4})
```

Destination column

[42]:
```
train_data['Destination'].unique()
```

[42]:
```
array(['New Delhi', 'Banglore', 'Cochin', 'Kolkata', 'Delhi', 'Hyderabad'],
      dtype=object)
```

[43]:
```
train_data['Destination']=train_data['Destination'].replace({'New Delhi':0, 'Banglore':1, 'Cochin':2, 'Kolkata':3, 'Delhi':4, 'Hyderabad':5})
```

```
C:\Users\icon\AppData\Local\Temp\ipykernel_9800\1360004538.py:1: FutureWarning: Downcasting behavior in `replace` is deprecated and will be removed in a fu
ture version. To retain the old behavior, explicitly call `result.infer_objects(copy=False)`. To opt-in to the future behavior, set `pd.set_option('future.
no_silent_downcasting', True)`
  train_data['Destination']=train_data['Destination'].replace({'New Delhi':0, 'Banglore':1, 'Cochin':2, 'Kolkata':3, 'Delhi':4, 'Hyderabad':5})
```

[44]:
```
test_data['Destination']=test_data['Destination'].replace({'New Delhi':0, 'Banglore':1, 'Cochin':2, 'Kolkata':3, 'Delhi':4, 'Hyderabad':5})
```

```
C:\Users\icon\AppData\Local\Temp\ipykernel_9800\1203504125.py:1: FutureWarning: Downcasting behavior in `replace` is deprecated and will be removed in a fu
ture version. To retain the old behavior, explicitly call `result.infer_objects(copy=False)`. To opt-in to the future behavior, set `pd.set_option('future.
no_silent_downcasting', True)`
  test_data['Destination']=test_data['Destination'].replace({'New Delhi':0, 'Banglore':1, 'Cochin':2, 'Kolkata':3, 'Delhi':4, 'Hyderabad':5})
```

Route column

[46]:
```
test_data['Route'].unique()
```

[46]:
```
array(['DEL ? BOM ? COK', 'CCU ? MAA ? BLR', 'BLR ? DEL',
       'BLR ? TRV ? DEL', 'CCU ? HYD ? BLR', 'CCU ? BLR',
       'CCU ? BOM ? BLR', 'DEL ? MAA ? BOM ? COK', 'DEL ? HYD ? COK',
       'BLR ? BOM ? DEL', 'CCU ? DEL ? BLR', 'DEL ? JAI ? BOM ? COK',
       'DEL ? BHO ? BOM ? COK', 'BOM ? HYD', 'CCU ? PNQ ? BLR',
       'MAA ? CCU', 'CCU ? BOM ? COK ? BLR', 'DEL ? BLR ? COK',
       'DEL ? HYD ? MAA ? COK', 'DEL ? MAA ? COK',
       'CCU ? IXR ? BBI ? BOM ? BLR', 'CCU ? DEL ? AMD ? BLR',
       'DEL ? COK', 'CCU ? JAI ? BOM ? BLR', 'DEL ? LKO ? COK',
       'CCU ? IXB ? BLR', 'DEL ? CCU ? BOM ? COK',
       'CCU ? GAU ? DEL ? BLR', 'BLR ? COK ? DEL',
       'BLR ? BOM ? NAG ? DEL', 'CCU ? BBI ? BOM ? BLR',
       'BLR ? AMD ? DEL', 'BOM ? DEL ? HYD', 'DEL ? GOI ? BOM ? COK',
       'DEL ? IXU ? BOM ? COK', 'DEL ? HYD ? BOM ? COK',
       'CCU ? DEL ? COK ? BLR', 'BLR ? GOI ? DEL',
       'BLR ? BOM ? BHO ? DEL', 'DEL ? IDR ? BOM ? COK',
       'DEL ? AMD ? BOM ? COK', 'DEL ? RPR ? NAG ? BOM ? COK',
       'DEL ? NAG ? BOM ? COK', 'CCU ? PAT ? BLR',
       'BLR ? HYD ? VGA ? DEL', 'BLR ? BOM ? IXC ? DEL',
       'BLR ? BOM ? IDR ? DEL', 'CCU ? VNS ? DEL ? BLR',
       'CCU ? BBI ? BLR', 'BLR ? MAA ? DEL', 'CCU ? IXR ? DEL ? BLR',
       'CCU ? BOM ? GOI ? BLR', 'DEL ? ATQ ? BOM ? COK',
       'BLR ? VGA ? DEL', 'DEL ? BDQ ? BOM ? COK', 'BLR ? HYD ? DEL',
       'BLR ? BOM ? AMD ? DEL', 'DEL ? DED ? BOM ? COK',
       'DEL ? AMD ? COK', 'CCU ? BOM ? HBX ? BLR', 'BLR ? STV ? DEL',
       'CCU ? KNU ? BLR', 'DEL ? TRV ? COK', 'BLR ? CCU ? GAU ? DEL',
       'DEL ? UDR ? BOM ? COK', 'BOM ? BLR ? HYD',
       'CCU ? BOM ? AMD ? BLR', 'DEL ? JDH ? BOM ? COK',
       'DEL ? PNQ ? COK', 'DEL ? GWL ? IDR ? BOM ? COK',
       'BLR ? BOM ? IDR ? GWL ? DEL', 'BLR ? BDQ ? DEL',
       'BOM ? JDH ? DEL ? HYD', 'CCU ? GAU ? BLR',
       'BLR ? BOM ? UDR ? DEL', 'BLR ? BBI ? DEL',
       'BOM ? VGA ? TIR ? HYD', 'CCU ? DEL ? VGA ? BLR',
       'DEL ? LKO ? BOM ? COK', 'CCU ? JAI ? DEL ? BLR',
       'CCU ? NAG ? BLR', 'BOM ? JAI ? DEL ? HYD', 'BLR ? CCU ? DEL',
       'BLR ? VGA ? VTZ ? DEL', 'CCU ? BOM ? TRV ? BLR',
       'BOM ? GOI ? HYD', 'BOM ? IXC ? DEL ? HYD',
       'CCU ? DEL ? COK ? TRV ? BLR', 'BLR ? GAU ? DEL',
       'BLR ? BOM ? JDH ? DEL', 'DEL ? IXC ? BOM ? COK',
       'CCU ? BBI ? HYD ? BLR', 'BLR ? VGA ? HYD ? DEL',
       'BLR ? NAG ? DEL', 'CCU ? AMD ? BLR', 'BOM ? AMD ? ISK ? HYD',
       'BLR ? CCU ? BBI ? HYD ? VGA ? DEL', 'CCU ? BBI ? IXR ? DEL ? BLR',
       'CCU ? IXR ? BBI ? BLR', 'BOM ? GOI ? PNQ ? HYD'], dtype=object)
```

[47]:
```
# Frequency encoding for 'Route'
route_counts = train_data['Route'].value_counts()
train_data['Route_Encoded'] = train_data['Route'].map(route_counts)
test_data['Route_Encoded'] = test_data['Route'].map(route_counts)

print(train_data[['Route', 'Route_Encoded']].head())
```

```
                  Route  Route_Encoded
0             BLR ? DEL           1552
1  CCU ? IXR ? BBI ? BLR              6
2  DEL ? LKO ? BOM ? COK             41
3        CCU ? NAG ? BLR              9
4        BLR ? NAG ? DEL              3
```

[48]:
```
train_data.drop(columns=['Route'], inplace=True)
test_data.drop(columns=['Route'], inplace=True)
```

Dep_Time, Arrival_Time, Duration columns

[50]:
```
train_data.drop(columns=['Dep_Time','Arrival_Time', 'Duration'], inplace=True)
test_data.drop(columns=['Dep_Time','Arrival_Time', 'Duration'], inplace=True)
```

Total_Stops column

[52]:
```
train_data['Total_Stops'].unique()
```

[52]:
```
array(['non-stop', '2 stops', '1 stop', '3 stops', 'DEL ? BOM ? COK',
       '4 stops'], dtype=object)
```

[53]:
```
train_data['Total_Stops']=train_data['Total_Stops'].replace({'non-stop':0, '2 stops':2, '1 stop':1, '3 stops':3, 'DEL ? BOM ? COK':5,'4 stops':4})
test_data['Total_Stops']=test_data['Total_Stops'].replace({'non-stop':0, '2 stops':2, '1 stop':1, '3 stops':3, 'DEL ? BOM ? COK':5,'4 stops':4})
```

```
C:\Users\icon\AppData\Local\Temp\ipykernel_9800\2853076606.py:1: FutureWarning: Downcasting behavior in `replace` is deprecated and will be removed in a fu
ture version. To retain the old behavior, explicitly call `result.infer_objects(copy=False)`. To opt-in to the future behavior, set `pd.set_option('future.
no_silent_downcasting', True)`
  train_data['Total_Stops']=train_data['Total_Stops'].replace({'non-stop':0, '2 stops':2, '1 stop':1, '3 stops':3, 'DEL ? BOM ? COK':5,'4 stops':4})
C:\Users\icon\AppData\Local\Temp\ipykernel_9800\2853076606.py:2: FutureWarning: Downcasting behavior in `replace` is deprecated and will be removed in a fu
ture version. To retain the old behavior, explicitly call `result.infer_objects(copy=False)`. To opt-in to the future behavior, set `pd.set_option('future.
no_silent_downcasting', True)`
  test_data['Total_Stops']=test_data['Total_Stops'].replace({'non-stop':0, '2 stops':2, '1 stop':1, '3 stops':3, 'DEL ? BOM ? COK':5,'4 stops':4})
```

Additional_Info column

[55]:
```
train_data['Additional_Info'].value_counts()
```

[55]:
```
Additional_Info
No info                         8345
In-flight meal not included     1982
No check-in baggage included     320
1 Long layover                    19
Change airports                    7
Business class                     4
No Info                            3
1 Short layover                    1
Red-eye flight                     1
2 Long layover                     1
Name: count, dtype: int64
```

[56]:
```
# Standardize 'No Info' to 'No info'
train_data['Additional_Info'] = train_data['Additional_Info'].replace({'No Info': 'No info'})
test_data['Additional_Info'] = test_data['Additional_Info'].replace({'No Info': 'No info'})
```

[57]:
```
train_data['Additional_Info'].unique()
```

[57]:
```
array(['No info', 'In-flight meal not included',
       'No check-in baggage included', '1 Short layover',
       '1 Long layover', 'Change airports', 'Business class',
       'Red-eye flight', '2 Long layover'], dtype=object)
```

[58]:
```
train_data['Additional_Info']=train_data['Additional_Info'].replace({'No info':0, 'In-flight meal not included':1,'No check-in baggage included':2, '1 Short
    '1 Long layover':4, 'Change airports':5, 'Business class':6,'Red-eye flight':7, '2 Long layover':8})

test_data['Additional_Info']=test_data['Additional_Info'].replace({'No info':0, 'In-flight meal not included':1,'No check-in baggage included':2, '1 Short
    '1 Long layover':4, 'Change airports':5, 'Business class':6,'Red-eye flight':7, '2 Long layover':8})
```

```
C:\Users\icon\AppData\Local\Temp\ipykernel_9800\1614609736.py:1: FutureWarning: Downcasting behavior in `replace` is deprecated and will be removed in a fu
ture version. To retain the old behavior, explicitly call `result.infer_objects(copy=False)`. To opt-in to the future behavior, set `pd.set_option('future.
no_silent_downcasting', True)`
  train_data['Additional_Info']=train_data['Additional_Info'].replace({'No info':0, 'In-flight meal not included':1,'No check-in baggage included':2, '1 Sh
ort layover':3,
C:\Users\icon\AppData\Local\Temp\ipykernel_9800\1614609736.py:4: FutureWarning: Downcasting behavior in `replace` is deprecated and will be removed in a fu
ture version. To retain the old behavior, explicitly call `result.infer_objects(copy=False)`. To opt-in to the future behavior, set `pd.set_option('future.
no_silent_downcasting', True)`
  test_data['Additional_Info']=test_data['Additional_Info'].replace({'No info':0, 'In-flight meal not included':1,'No check-in baggage included':2, '1 Shor
t layover':3,
```

[59]:
```
train_data
```

[59]:

	Airline	Source	Destination	Total_Stops	Additional_Info	Price	Journey_Day	Journey_Month	Journey_Weekday	Dep_Hour	Dep_Minute	Arrival_Hou
0	0	0	0	0	0	3897	24	3	6	22	20	
1	1	1	1	2	0	7662	1	5	2	5	50	1
2	2	2	2	2	0	13882	9	6	6	9	25	
3	0	1	1	1	0	6218	12	5	6	18	5	2
4	0	0	0	1	0	13302	1	3	4	16	50	2
...	...	...	...	...	...	...	...	...	...	...	...	
10678	7	1	1	0	0	4107	9	4	1	19	55	2
10679	1	1	1	0	0	4145	27	4	5	20	45	2
10680	2	0	4	0	0	7229	27	4	5	8	20	1
10681	6	0	0	0	0	12648	1	3	4	11	30	1
10682	1	2	2	2	0	11753	9	5	3	10	55	1

10683 rows × 15 columns

[60]:
```
test_data
```

[60]:

	Airline	Source	Destination	Total_Stops	Additional_Info	Journey_Day	Journey_Month	Journey_Weekday	Dep_Hour	Dep_Minute	Arrival_Hour	Arriva
0	2	2	2	1	0	6	6	3	17	30	4	
1	0	1	1	1	0	12	5	6	6	20	10	
2	2	2	2	1	1	21	5	1	19	15	19	
3	4	2	2	1	0	21	5	1	8	0	21	
4	7	0	4	0	0	24	6	0	23	55	2	
...	...	...	...	...	...	...	...	...	...	...	...	
2666	1	1	1	1	0	6	6	3	20	30	20	
2667	0	1	1	0	0	27	3	2	14	20	16	
2668	2	2	2	1	0	6	3	2	21	50	4	
2669	1	2	2	1	0	6	3	2	4	0	19	
2670	4	2	2	1	0	15	6	5	4	55	19	

2671 rows × 14 columns

[61]:
```
# Move 'Price' to the last position
train_data = train_data.loc[:, train_data.columns.difference(['Price']).tolist() + ['Price']]
```

[62]:
```
# Dependent column (target) in train data
y_train = train_data.iloc[:,14]

# Independent columns in train_data
X_train = train_data.iloc[:,:14]

# For the test dataset
X_test = train_data.iloc[:,:14]
```

[63]:
```
train_data
```

[63]:

	Additional_Info	Airline	Arrival_Hour	Arrival_Minute	Dep_Hour	Dep_Minute	Destination	Duration_Minutes	Journey_Day	Journey_Month	Journey_V
0	0	0	1	10	22	20	0	170	24	3	
1	0	1	13	15	5	50	1	445	1	5	
2	0	2	4	25	9	25	2	1140	9	6	
3	0	0	23	30	18	5	1	325	12	5	
4	0	0	21	35	16	50	0	285	1	3	
...	...	...	...	...	...	...	...	...	...	...	
10678	0	7	22	25	19	55	1	150	9	4	
10679	0	1	23	20	20	45	1	155	27	4	
10680	0	2	11	20	8	20	4	180	27	4	
10681	0	6	14	10	11	30	0	160	1	3	

10682	0	1	19	15	10	55	2	500	9	5

10683 rows × 15 columns

[64]:
```
y_train.shape
```

[64]:
```
(10683,)
```

[65]:
```
X_train.shape
```

[65]:
```
(10683, 14)
```

[66]:
```
X_test.shape
```

[66]:
```
(10683, 14)
```

[ ]: