In [685…

```
import pandas as pd
import numpy as np
import seaborn as sns
import re
import matplotlib.pyplot as plt
import warnings


# Ignore all warnings
warnings.filterwarnings('ignore')
```

In [686…

```
filepath='MPC95.csv'
df=pd.read_csv(filepath)
df
```

Out[686]:

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	D
0	IndiGo	24-03-2019	Banglore	New Delhi	BLR ? DEL	22:20	22-03-2024 01:10	
1	Air India	01-05-2019	Kolkata	Banglore	CCU ? IXR ? BBI ? BLR	05:50	13:15	
2	Jet Airways	09-06-2019	Delhi	Cochin	DEL ? LKO ? BOM ? COK	09:25	10-06-2024 04:25	
3	IndiGo	12-05-2019	Kolkata	Banglore	CCU ? NAG ? BLR	18:05	23:30	
4	IndiGo	01-03-2019	Banglore	New Delhi	BLR ? NAG ? DEL	16:50	21:35	
...	...	...	...	...	...	...	...	
10678	Air Asia	09-04-2019	Kolkata	Banglore	CCU ? BLR	19:55	22:25	
10679	Air India	27-04-2019	Kolkata	Banglore	CCU ? BLR	20:45	23:20	
10680	Jet Airways	27-04-2019	Banglore	Delhi	BLR ? DEL	08:20	11:20	
10681	Vistara	01-03-2019	Banglore	New Delhi	BLR ? DEL	11:30	14:10	
10682	Air India	09-05-2019	Delhi	Cochin	DEL ? GOI ? BOM ? COK	10:55	19:15	

10683 rows × 11 columns

In [687…

```
(df.head())
```

Out[687]:

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Durati
0	IndiGo	24-03-2019	Banglore	New Delhi	BLR ? DEL	22:20	22-03-2024 01:10	2h 50
1	Air India	01-05-2019	Kolkata	Banglore	CCU ? IXR ? BBI ? BLR	05:50	13:15	7h 25
2	Jet Airways	09-06-2019	Delhi	Cochin	DEL ? LKO ? BOM ? COK	09:25	10-06-2024 04:25	1
3	IndiGo	12-05-2019	Kolkata	Banglore	CCU ? NAG ? BLR	18:05	23:30	5h 25
4	IndiGo	01-03-2019	Banglore	New Delhi	BLR ? NAG ? DEL	16:50	21:35	4h 45

In [688…

```
df.tail()
```

Out[688]:

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	D
10678	Air Asia	09-04-2019	Kolkata	Banglore	CCU ? BLR	19:55	22:25	
10679	Air India	27-04-2019	Kolkata	Banglore	CCU ? BLR	20:45	23:20	
10680	Jet Airways	27-04-2019	Banglore	Delhi	BLR ? DEL	08:20	11:20	
10681	Vistara	01-03-2019	Banglore	New Delhi	BLR ? DEL	11:30	14:10	
10682	Air India	09-05-2019	Delhi	Cochin	DEL ? GOI ? BOM ? COK	10:55	19:15	

In [689…

```
df.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
```

In [690…
```
df['Price'].describe()
```

Out[690]:
```
count    10683.000000
mean      9087.064121
std       4611.359167
min       1759.000000
25%       5277.000000
50%       8372.000000
75%      12373.000000
max      79512.000000
Name: Price, dtype: float64
```

In [691…
```
df.nunique()
```

Out[691]:
```
Airline             12
Date_of_Journey     40
Source               5
Destination          6
Route              128
Dep_Time           222
Arrival_Time      1343
Duration           368
Total_Stops          5
Additional_Info     10
Price             1870
dtype: int64
```

In [692…
```
columns = df.columns

# Create a list to store unique counts for each column
unique_counts = [df[col].value_counts() for col in columns]

# Create a DataFrame with column names and unique counts
result_df = pd.DataFrame({"Unique Elements": columns, "Counts": unique_counts})

# Print the result DataFrame
print(result_df.to_string())
```

```
    Unique Elements
Counts
0           Airline
Airline
Jet Airways                          3849
IndiGo                               2053
Air India                            1752
Multiple carriers                    1196
SpiceJet                              818
Vistara                               479
Air Asia                              319
GoAir                                 194
Multiple carriers Premium economy      13
Jet Airways Business                    6
Vistara Premium economy                 3
Trujet                                  1
Name: count, dtype: int64
1   Date_of_Journey  Date_of_Journey
18-05-2019    504
06-06-2019    503
21-05-2019    497
09-06-2019    495
12-06-2019    493
09-05-2019    484
21-03-2019    423
15-05-2019    405
06-03-2019    403
27-05-2019    382
27-06-2019    355
24-06-2019    351
01-06-2019    342
03-06-2019    333
15-06-2019    328
24-03-2019    323
03-03-2019    315
09-03-2019    302
27-03-2019    299
24-05-2019    286
06-05-2019    282
01-05-2019    277
12-05-2019    259
01-04-2019    257
01-03-2019    199
15-03-2019    162
18-03-2019    156
12-03-2019    142
09-04-2019    125
03-04-2019    110
21-06-2019    109
18-06-2019    105
06-04-2019    100
27-04-2019     94
24-04-2019     92
03-05-2019     90
15-04-2019     89
21-04-2019     82
```

```
18-04-2019     67
12-04-2019     63
Name: count, dtype: int64
2            Source
Source
Delhi       4537
Kolkata     2871
Banglore    2197
Mumbai       697
Chennai      381
Name: count, dtype: int64
3       Destination
Destination
Cochin       4537
Banglore     2871
Delhi        1265
New Delhi     932
Hyderabad     697
Kolkata       381
Name: count, dtype: int64
4             Route
Route
DEL ? BOM ? COK          2376
BLR ? DEL                1552
CCU ? BOM ? BLR           979
CCU ? BLR                 724
BOM ? HYD                 621
                         ...
CCU ? VTZ ? BLR             1
CCU ? IXZ ? MAA ? BLR       1
BOM ? COK ? MAA ? HYD       1
BOM ? CCU ? HYD             1
BOM ? BBI ? HYD             1
Name: count, Length: 128, dtype: int64
5          Dep_Time
Dep_Time
18:55    233
17:00    227
07:05    205
10:00    203
07:10    202
        ...
16:25      1
01:35      1
21:35      1
04:15      1
03:00      1
Name: count, Length: 222, dtype: int64
6      Arrival_Time
Arrival_Time
19:00               423
21:00               360
19:15               333
16:10               154
12:35               122
                   ...
```

```
02-06-2024 00:25      1
13-03-2024 08:55      1
19-05-2024 11:05      1
22-05-2024 12:30      1
13-03-2024 21:20      1
Name: count, Length: 1343, dtype: int64
7          Duration
Duration
2h 50m     550
1h 30m     386
2h 45m     337
2h 55m     337
2h 35m     329
           ...
31h 30m      1
30h 25m      1
42h 5m       1
4h 10m       1
47h 40m      1
Name: count, Length: 368, dtype: int64
8       Total_Stops
Total_Stops
1 stop      5625
non-stop    3491
2 stops     1520
3 stops       45
4 stops        1
Name: count, dtype: int64
9   Additional_Info
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
10            Price
Price
10262    258
10844    212
7229     162
4804     160
4823     131
         ...
14153      1
8488       1
7826       1
6315       1
12648      1
Name: count, Length: 1870, dtype: int64
```

In [693…
```
df.isnull().sum()
```

Out[693]:
```
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
```

In [694…
```
df1=df.dropna()
```

In [695…
```
def convert2min(duration):
    h=m=0
    for i in duration.split():
        if 'h' in i:
            h=int(i.replace('h',''))
        elif 'm' in i:
            m=int(i.replace('m',''))
    total=(float(h*60+m))
    return (total)

df1.loc[:,'DurationinMin'] = df1['Duration'].apply(convert2min)


def stopinNumeric(total_stops):
    a={"1 stop":1,"non-stop":0,"2 stops":2,"3 stops":3,"4 stops":1}
    result=(a[total_stops])
    return result

df1.loc[:, 'TotalStopNumeric'] = df1['Total_Stops'].apply(stopinNumeric)

df1.head()
```

Out[695]:

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Durati
0	IndiGo	24-03-2019	Banglore	New Delhi	BLR ? DEL	22:20	22-03-2024 01:10	2h 50
1	Air India	01-05-2019	Kolkata	Banglore	CCU ? IXR ? BBI ? BLR	05:50	13:15	7h 25
2	Jet Airways	09-06-2019	Delhi	Cochin	DEL ? LKO ? BOM ? COK	09:25	10-06-2024 04:25	1
3	IndiGo	12-05-2019	Kolkata	Banglore	CCU ? NAG ? BLR	18:05	23:30	5h 25
4	IndiGo	01-03-2019	Banglore	New Delhi	BLR ? NAG ? DEL	16:50	21:35	4h 45

In [696…

```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 10682 entries, 0 to 10682
Data columns (total 13 columns):
 #   Column            Non-Null Count  Dtype 
---  ------            --------------  ----- 
 0   Airline           10682 non-null  object
 1   Date_of_Journey   10682 non-null  object
 2   Source            10682 non-null  object
 3   Destination       10682 non-null  object
 4   Route             10682 non-null  object
 5   Dep_Time          10682 non-null  object
 6   Arrival_Time      10682 non-null  object
 7   Duration          10682 non-null  object
 8   Total_Stops       10682 non-null  object
 9   Additional_Info   10682 non-null  object
 10  Price             10682 non-null  int64 
 11  DurationinMin     10682 non-null  float64
 12  TotalStopNumeric  10682 non-null  int64 
dtypes: float64(1), int64(2), object(10)
memory usage: 1.1+ MB
```

In [697…

```
df1.isnull().sum()
```

Out[697]:
```
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
Price               0
DurationinMin       0
TotalStopNumeric    0
dtype: int64
```

In [698…
```
from datetime import datetime, timedelta
df2=df1.drop(columns=['Total_Stops','Duration'])
df2['Dep_Day']=df2['Date_of_Journey'].str.split('-').str[0].astype(int)
df2['Dep_Month']=df2['Date_of_Journey'].str.split('-').str[1].astype(int)
df2['Dep_Hour']=pd.to_datetime(df2['Dep_Time']).dt.hour
df2['Dep_min']=pd.to_datetime(df2['Dep_Time']).dt.minute

def extract_components(row):
    journey_date = row['Date_of_Journey']
    arrival_time = row['Arrival_Time']

    # Check if the arrival time includes a date
    if ' ' in arrival_time:
        arrival_datetime = datetime.strptime(arrival_time, '%d-%m-%Y %H:%M')
    else:
        arrival_datetime = datetime.strptime(f"{journey_date} {arrival_time}", '%d-
        # Check if time is on the next day
        if arrival_datetime.time() < datetime.strptime('12:00', '%H:%M').time():  #
            arrival_datetime += timedelta(days=1)

    return pd.Series([arrival_datetime.day, arrival_datetime.month, arrival_datetim

df2[['Arr_Day', 'Arr_Month', 'Arr_Hour', 'Arr_minute']] = df2.apply(extract_compone



df2.drop(['Dep_Time','Date_of_Journey','Arrival_Time'],axis=1,inplace=True)
df2
```

Out[698]:

	Airline	Source	Destination	Route	Additional_Info	Price	DurationinMin	Total
0	IndiGo	Banglore	New Delhi	BLR ? DEL	No info	3897	170.0	
1	Air India	Kolkata	Banglore	CCU ? IXR ? BBI ? BLR	No info	7662	445.0	
2	Jet Airways	Delhi	Cochin	DEL ? LKO ? BOM ? COK	No info	13882	1140.0	
3	IndiGo	Kolkata	Banglore	CCU ? NAG ? BLR	No info	6218	325.0	
4	IndiGo	Banglore	New Delhi	BLR ? NAG ? DEL	No info	13302	285.0	
...	...	...	...	...	...	...	...	
10678	Air Asia	Kolkata	Banglore	CCU ? BLR	No info	4107	150.0	
10679	Air India	Kolkata	Banglore	CCU ? BLR	No info	4145	155.0	
10680	Jet Airways	Banglore	Delhi	BLR ? DEL	No info	7229	180.0	
10681	Vistara	Banglore	New Delhi	BLR ? DEL	No info	12648	160.0	
10682	Air India	Delhi	Cochin	DEL ? GOI ? BOM ? COK	No info	11753	500.0	

10682 rows × 16 columns

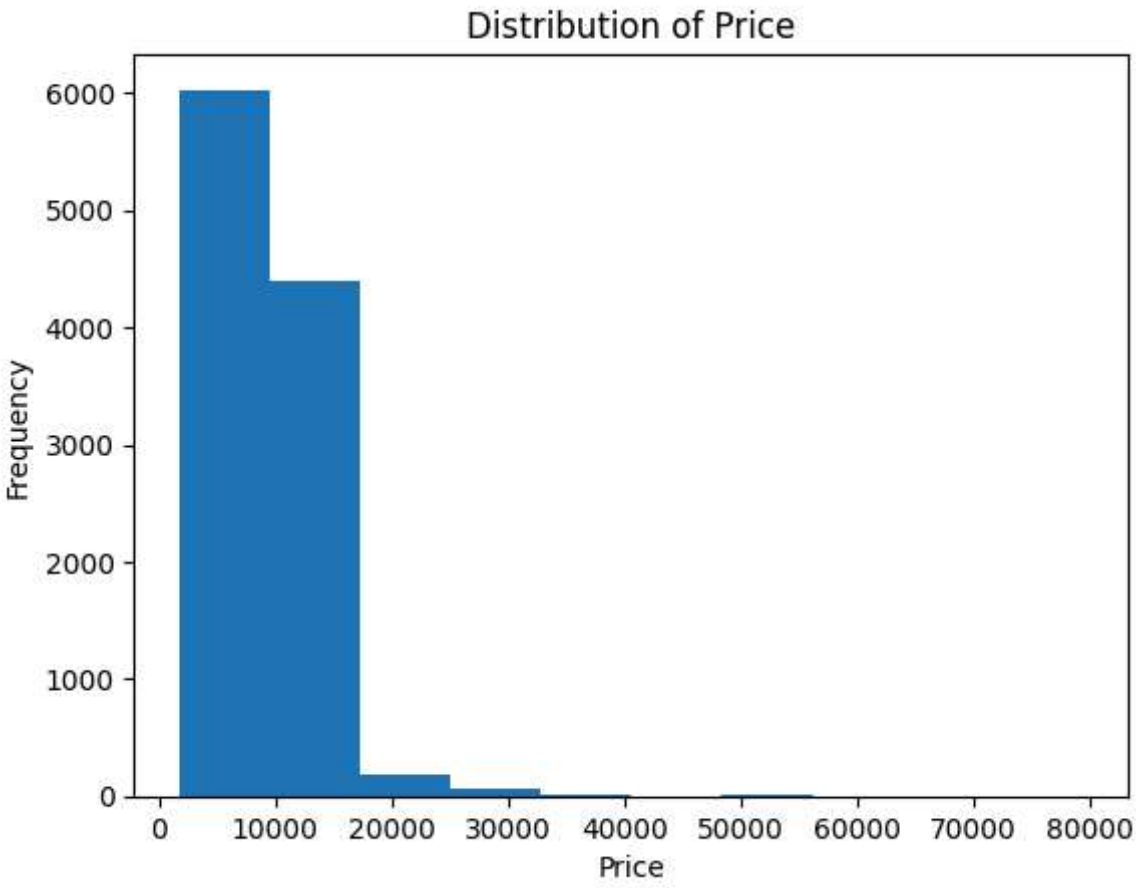
In [699…

```
df2['Additional_Info'].value_counts()
```

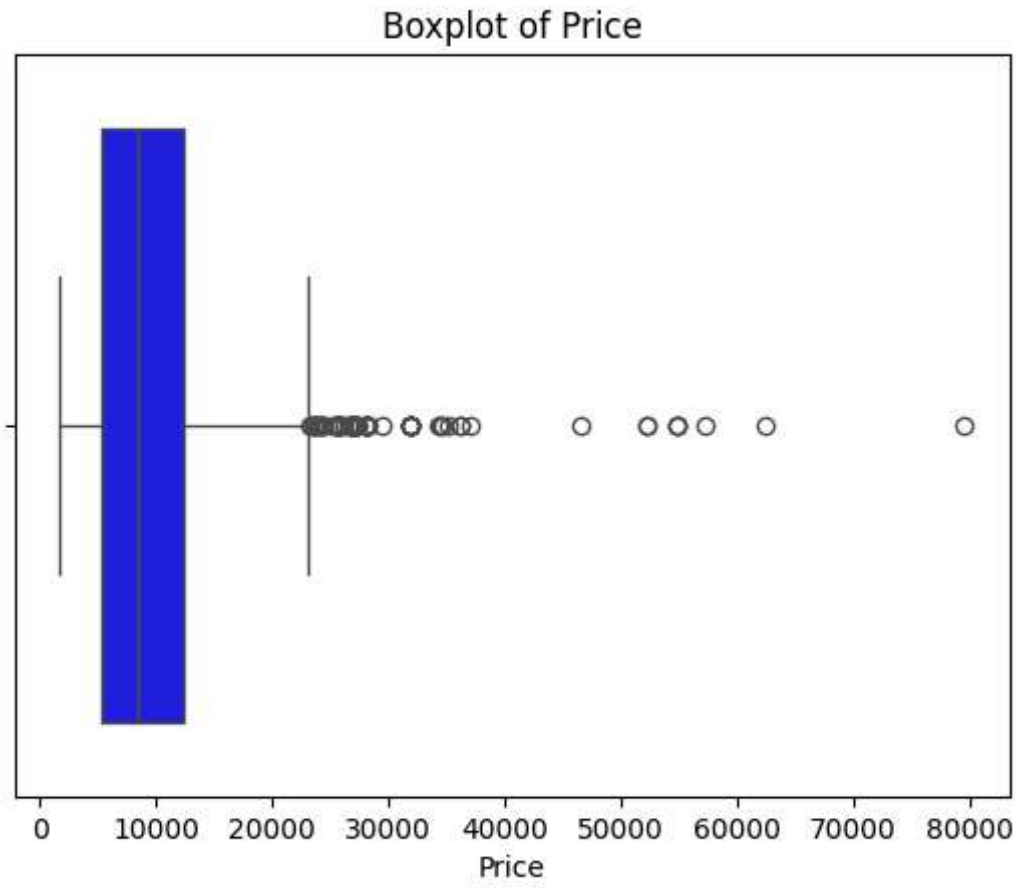
Out[699]:
```
Additional_Info
No info                         8344
In-flight meal not included     1982
No check-in baggage included     320
1 Long layover                    19
Change airports                    7
Business class                     4
No Info                            3
1 Short layover                    1
Red-eye flight                     1
2 Long layover                     1
Name: count, dtype: int64
```

Univariate analysis

In [700…
```
#sns.kdeplot(df['Price'], fill=True)
df2['Price'].hist(grid=False)
plt.xlabel('Price')
plt.ylabel("Frequency")
plt.title("Distribution of Price")
plt.show()
```

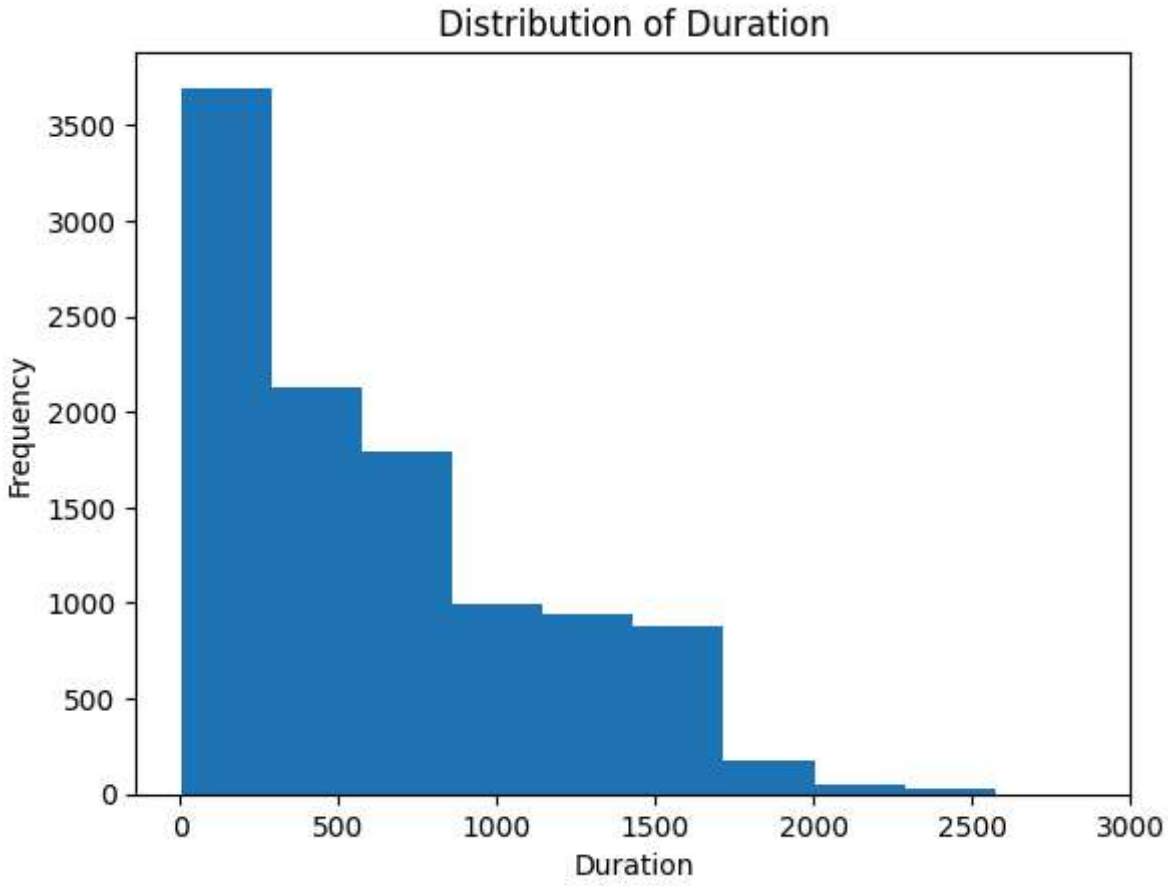


In [701…
```
sns.boxplot(x=df['Price'],color='blue')
plt.title("Boxplot of Price")
plt.xlabel('Price')
plt.show()
```



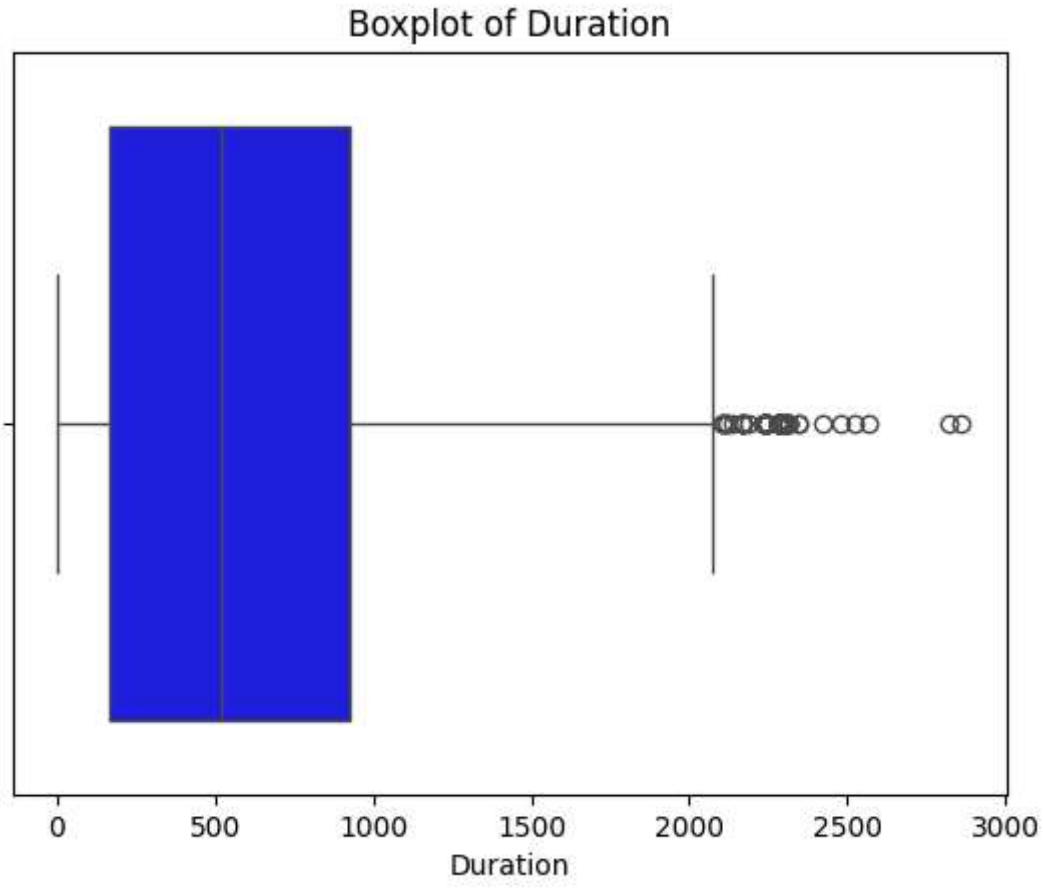
In [702…

```
#sns.kdeplot(df2['DurationinHours'], fill=True)
df2['DurationinMin'].hist(grid=False)
plt.xlabel('Duration')
plt.ylabel("Frequency")
plt.title("Distribution of Duration")
plt.show()
```



In [703…

```
sns.boxplot(x=df2['DurationinMin'],color='blue')
plt.title("Boxplot of Duration")
plt.xlabel('Duration')
plt.show()
```



In [704…
```
df2.nunique()
```

Out[704]:
```
Airline              12
Source                5
Destination           6
Route               128
Additional_Info      10
Price              1870
DurationinMin       368
TotalStopNumeric      4
Dep_Day              10
Dep_Month             4
Dep_Hour             24
Dep_min              12
Arr_Day              28
Arr_Month             4
Arr_Hour             24
Arr_minute           12
dtype: int64
```

In [705…
```
df2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 10682 entries, 0 to 10682
Data columns (total 16 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   Airline           10682 non-null  object
 1   Source            10682 non-null  object
 2   Destination       10682 non-null  object
 3   Route             10682 non-null  object
 4   Additional_Info   10682 non-null  object
 5   Price             10682 non-null  int64
 6   DurationinMin     10682 non-null  float64
 7   TotalStopNumeric  10682 non-null  int64
 8   Dep_Day           10682 non-null  int32
 9   Dep_Month         10682 non-null  int32
 10  Dep_Hour          10682 non-null  int32
 11  Dep_min           10682 non-null  int32
 12  Arr_Day           10682 non-null  int64
 13  Arr_Month         10682 non-null  int64
 14  Arr_Hour          10682 non-null  int64
 15  Arr_minute        10682 non-null  int64
dtypes: float64(1), int32(4), int64(6), object(5)
memory usage: 1.2+ MB
```

In [706…

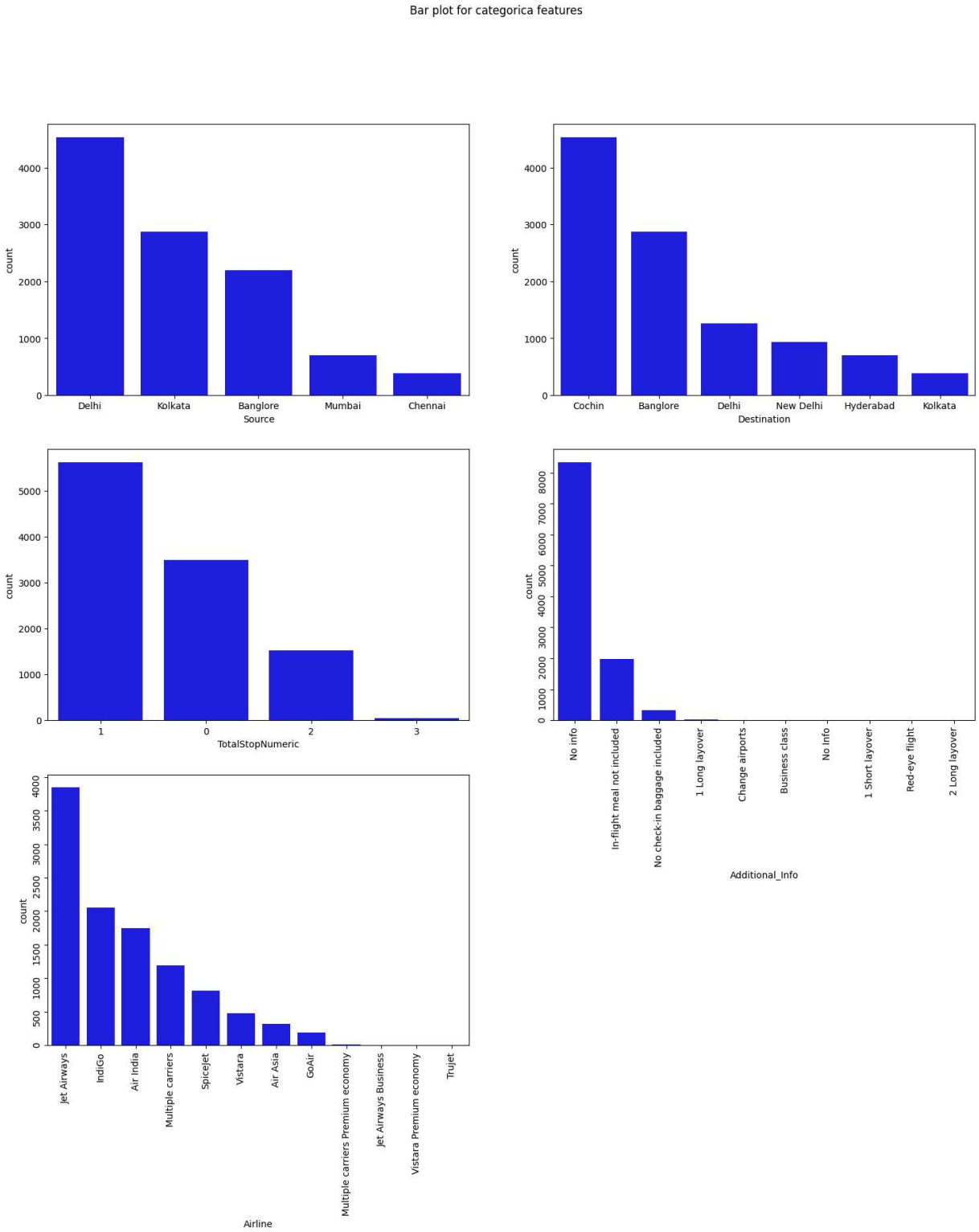
```
df2['DurationinMin'].value_counts()
```

Out[706]:

```
DurationinMin
170.0     550
90.0      386
165.0     337
175.0     337
155.0     329
          ...
1890.0      1
1825.0      1
2525.0      1
250.0       1
2860.0      1
Name: count, Length: 368, dtype: int64
```

In [707…

```
fig, axes=plt.subplots(3,2,figsize=(18,18))
fig.suptitle('Bar plot for categorica features')
sns.countplot(ax=axes[0,0],x='Source',data=df2,color='blue',order=df2['Source'].val
sns.countplot(ax=axes[0,1],x='Destination',data=df2,color='blue',order=df2['Destina
sns.countplot(ax=axes[2,0],x='Airline',data=df2,color='blue',order=df2['Airline'].v
sns.countplot(ax=axes[1,1],x='Additional_Info',data=df2,color='blue',order=df2['Add
sns.countplot(ax=axes[1,0],x='TotalStopNumeric',data=df2,color='blue',order=df2['To
axes[2,1].axis('off')
axes[2,0].tick_params(labelrotation=90)
axes[1,1].tick_params(labelrotation=90)
```



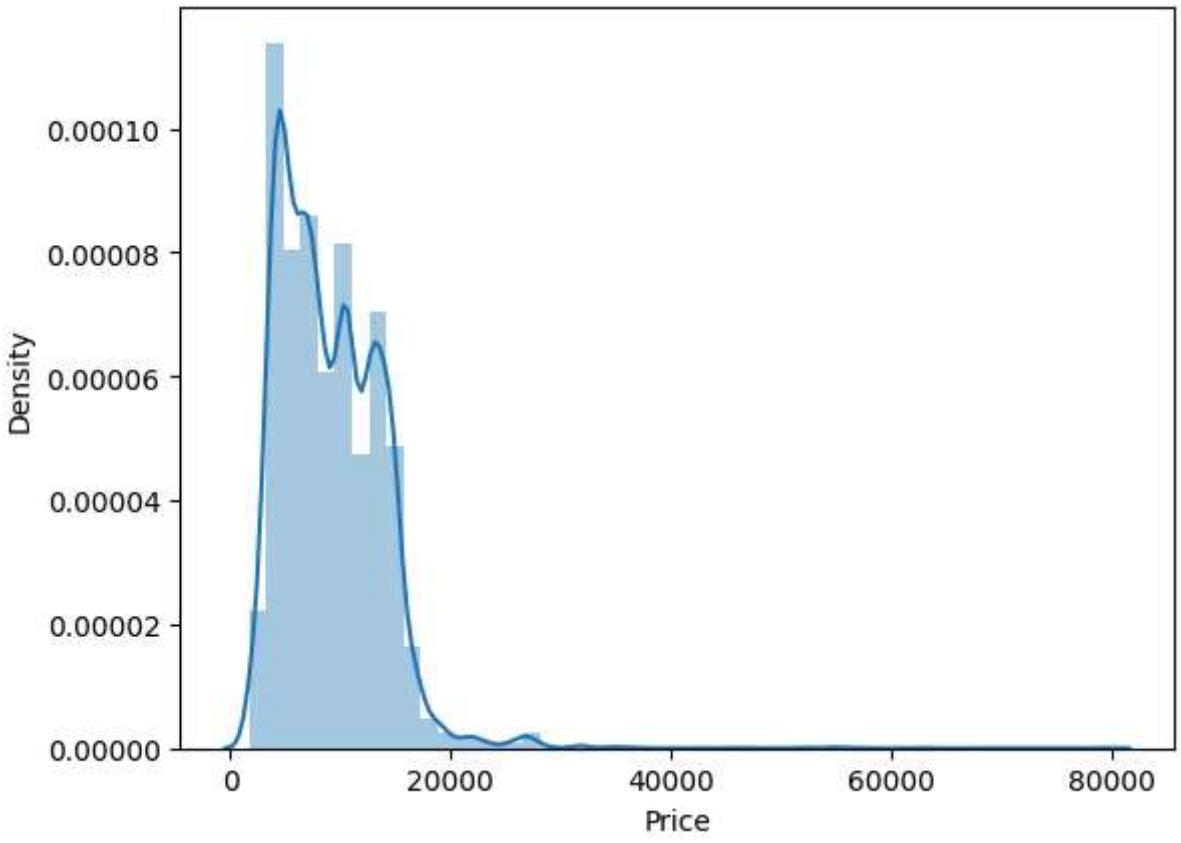
In [708…

```
def log_transform(data,col):
    for i in col:
        if (data[i]==1.0).all():
            data[i + '_log']=np.log(data[i]+1)
        else:
            data[i + '_log']=np.log(data[i])
    data.info()
```

```
log_transform(df2,['Price','DurationinMin'])
sns.distplot(df2['Price'],axlabel="Price")
```

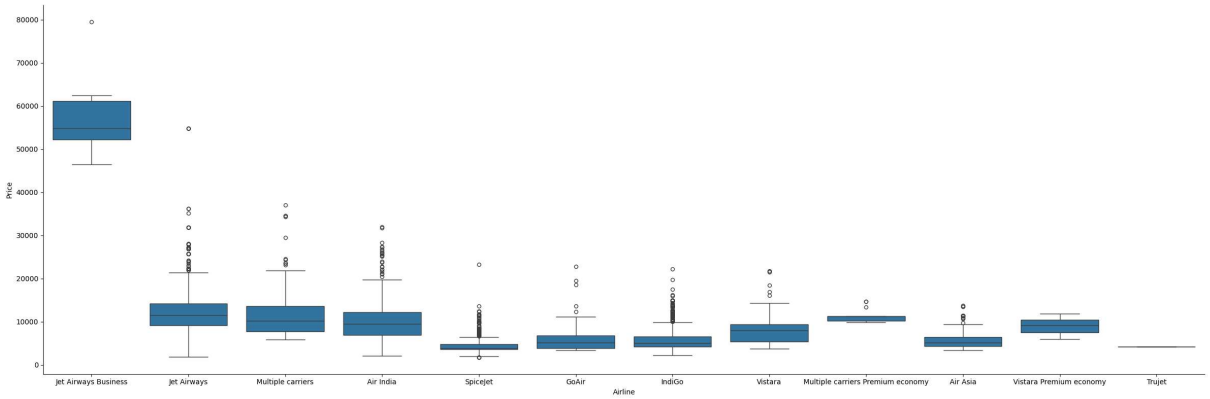
```
<class 'pandas.core.frame.DataFrame'>
Index: 10682 entries, 0 to 10682
Data columns (total 18 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Airline            10682 non-null  object 
 1   Source             10682 non-null  object 
 2   Destination        10682 non-null  object 
 3   Route              10682 non-null  object 
 4   Additional_Info    10682 non-null  object 
 5   Price              10682 non-null  int64  
 6   DurationinMin      10682 non-null  float64
 7   TotalStopNumeric   10682 non-null  int64  
 8   Dep_Day            10682 non-null  int32  
 9   Dep_Month          10682 non-null  int32  
 10  Dep_Hour           10682 non-null  int32  
 11  Dep_min            10682 non-null  int32  
 12  Arr_Day            10682 non-null  int64  
 13  Arr_Month          10682 non-null  int64  
 14  Arr_Hour           10682 non-null  int64  
 15  Arr_minute         10682 non-null  int64  
 16  Price_log          10682 non-null  float64
 17  DurationinMin_log  10682 non-null  float64
dtypes: float64(3), int32(4), int64(6), object(5)
memory usage: 1.4+ MB
```

Out[708]: <Axes: xlabel='Price', ylabel='Density'>



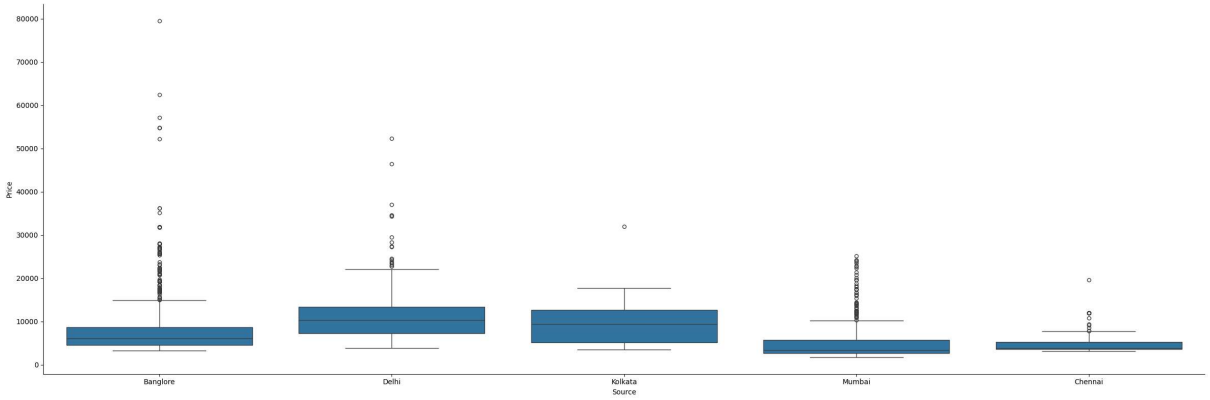
In [709…
```
sns.catplot(y='Price',x='Airline',data=df2.sort_values('Price',ascending=False),kin
```

Out[709]:
```
<seaborn.axisgrid.FacetGrid at 0x26b59386790>
```



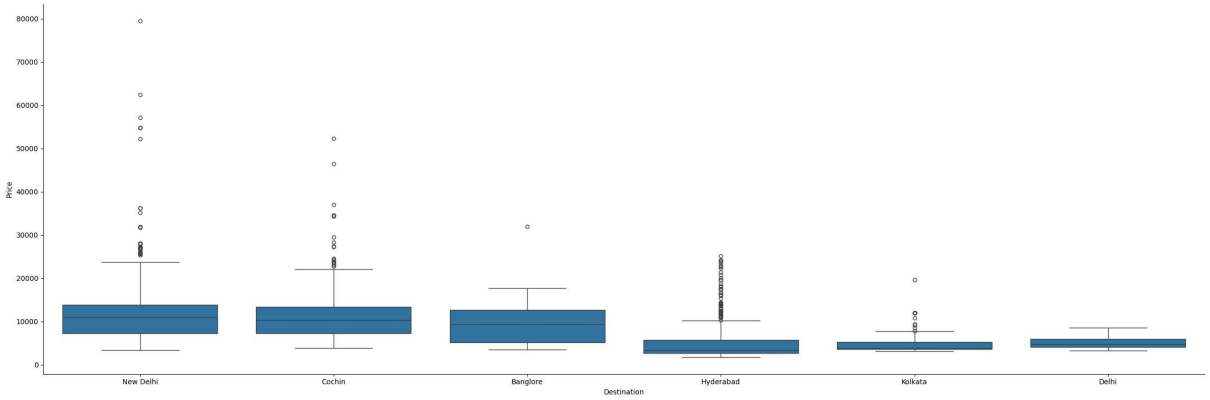
In [710…
```
sns.catplot(y='Price',x='Source',data=df2.sort_values('Price',ascending=False),kind
```

Out[710]:
```
<seaborn.axisgrid.FacetGrid at 0x26b7cb41650>
```



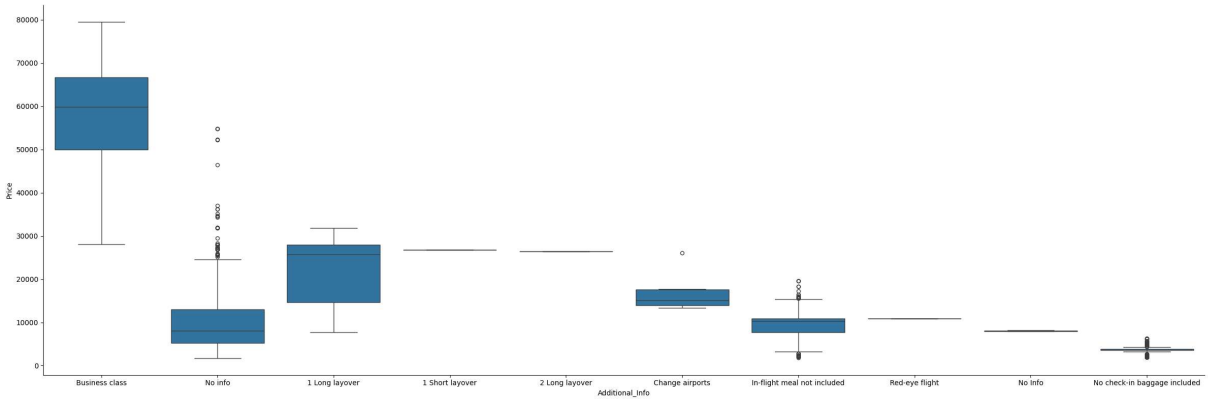
In [711…
```
sns.catplot(y='Price',x='Destination',data=df2.sort_values('Price',ascending=False)
```

Out[711]:
```
<seaborn.axisgrid.FacetGrid at 0x26b7cbd1c50>
```



In [712…
```
sns.catplot(y='Price',x='Additional_Info',data=df2.sort_values('Price',ascending=Fa
```

Out[712]:
```
<seaborn.axisgrid.FacetGrid at 0x26b944514d0>
```

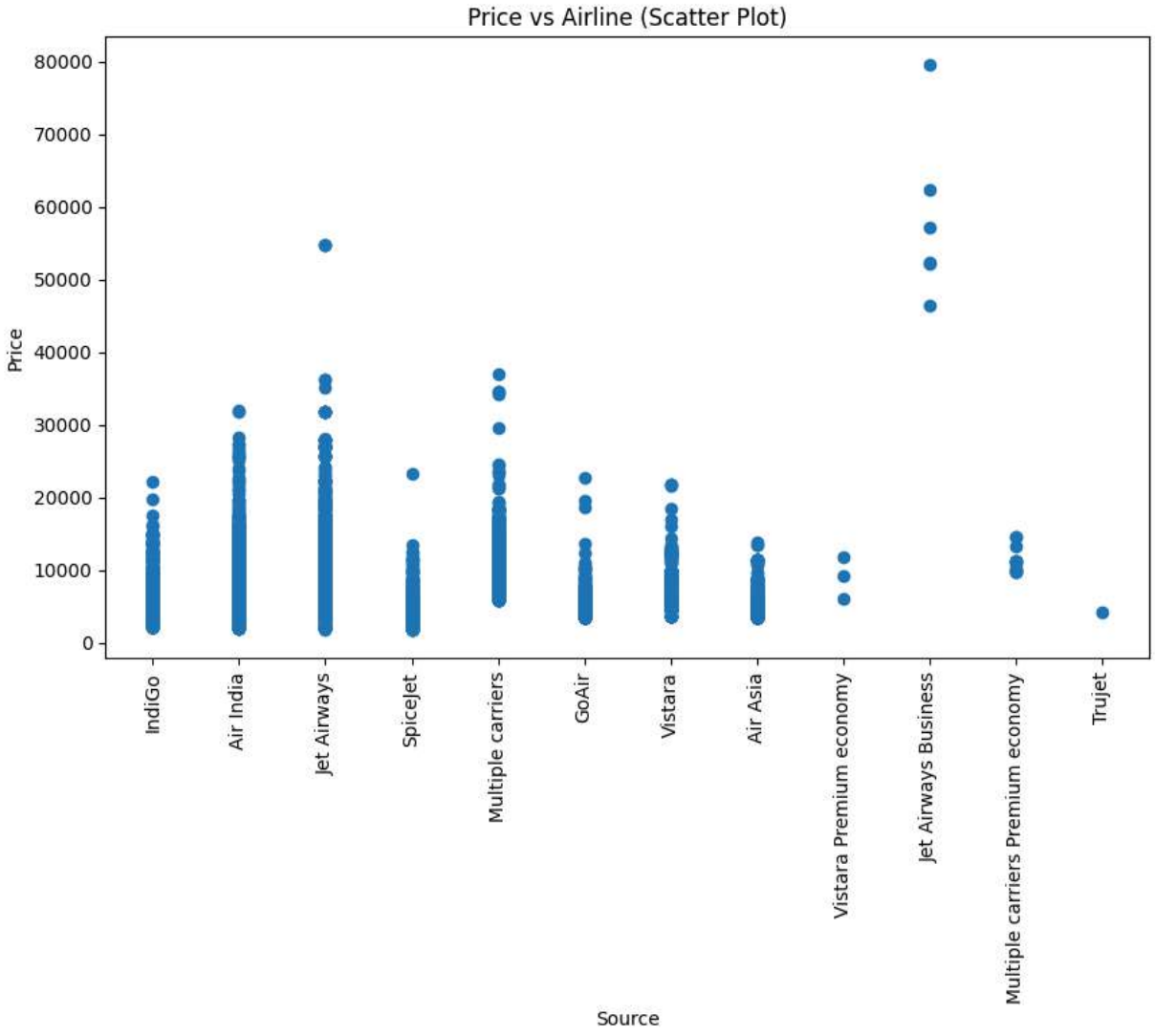


In [713…
```
df2.head()
```

Out[713]:

	Airline	Source	Destination	Route	Additional_Info	Price	DurationinMin	TotalStop
0	IndiGo	Banglore	New Delhi	BLR ? DEL	No info	3897	170.0	
1	Air India	Kolkata	Banglore	CCU ? IXR ? BBI ? BLR	No info	7662	445.0	
2	Jet Airways	Delhi	Cochin	DEL ? LKO ? BOM ? COK	No info	13882	1140.0	
3	IndiGo	Kolkata	Banglore	CCU ? NAG ? BLR	No info	6218	325.0	
4	IndiGo	Banglore	New Delhi	BLR ? NAG ? DEL	No info	13302	285.0	

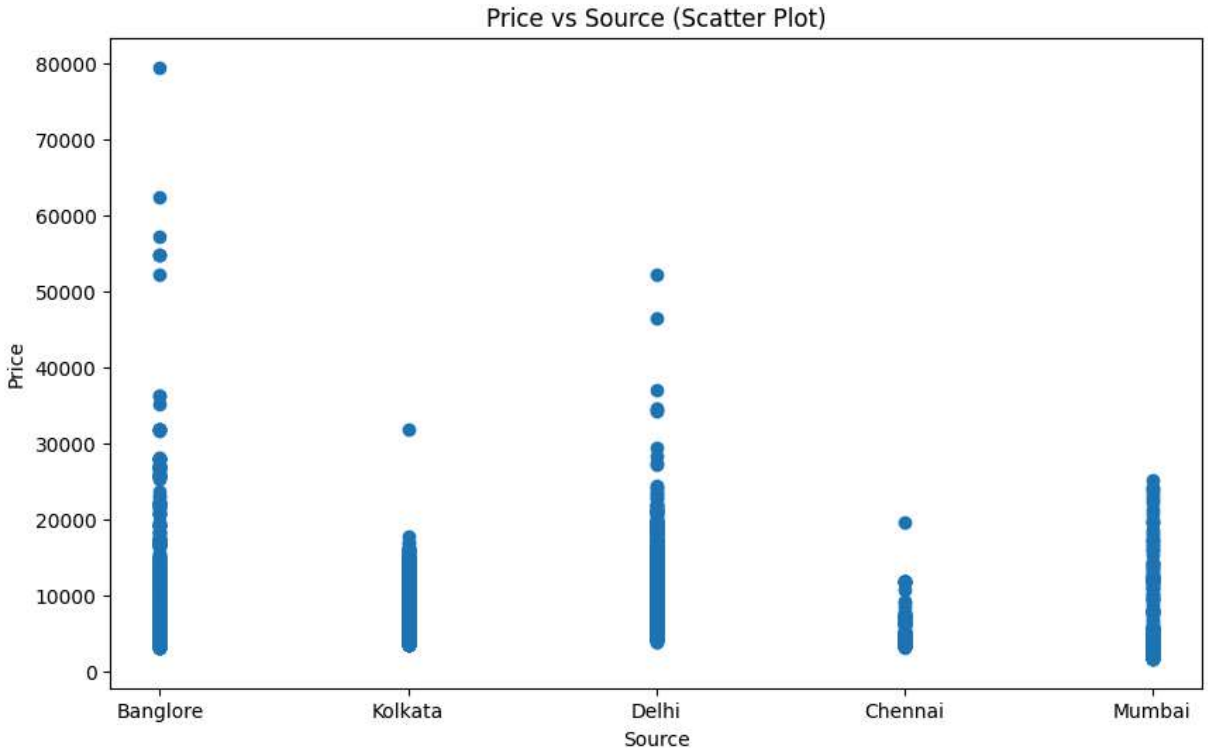
In [714…
```
plt.figure(figsize=(10, 6))
plt.scatter(df2["Airline"], df2["Price"])
plt.xlabel("Source")
plt.ylabel("Price")
plt.title("Price vs Airline (Scatter Plot)")
plt.xticks(rotation=90)
plt.show()
```



In [715…

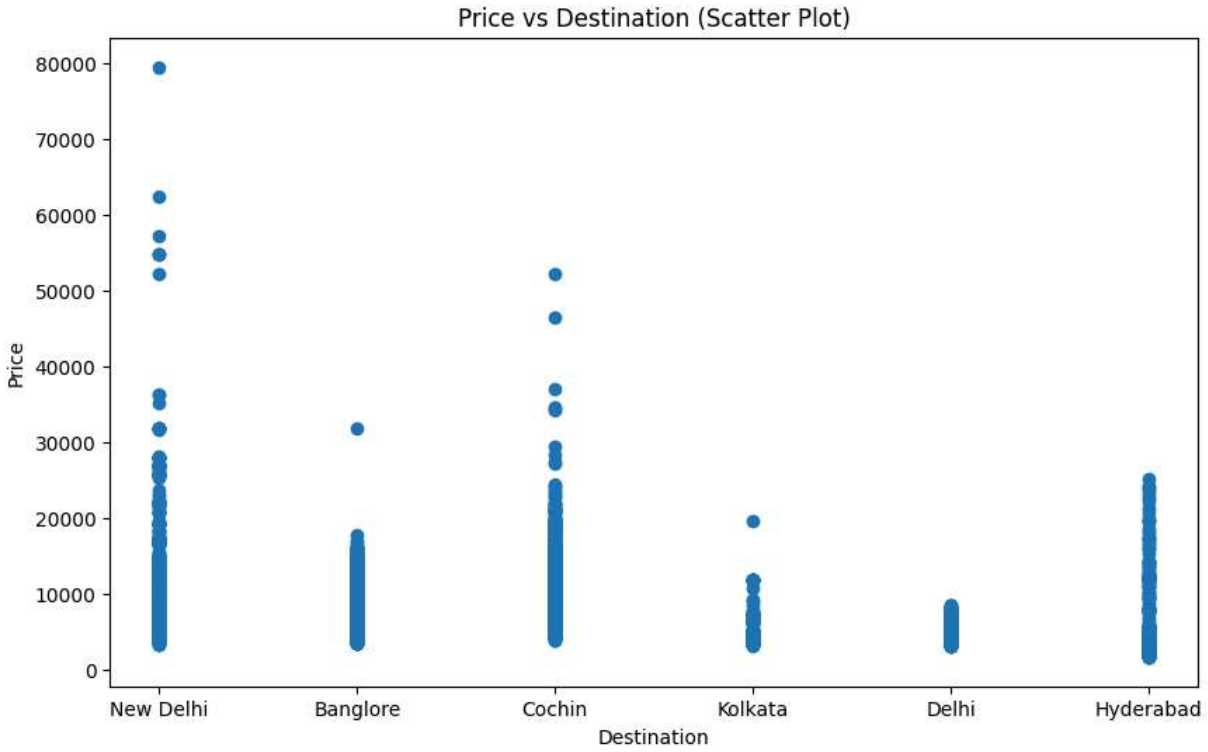
```
plt.figure(figsize=(10, 6))
plt.scatter(df1["Source"], df1["Price"])
plt.xlabel("Source")
plt.ylabel("Price")
plt.title("Price vs Source (Scatter Plot)")

plt.show()
```



In [716…

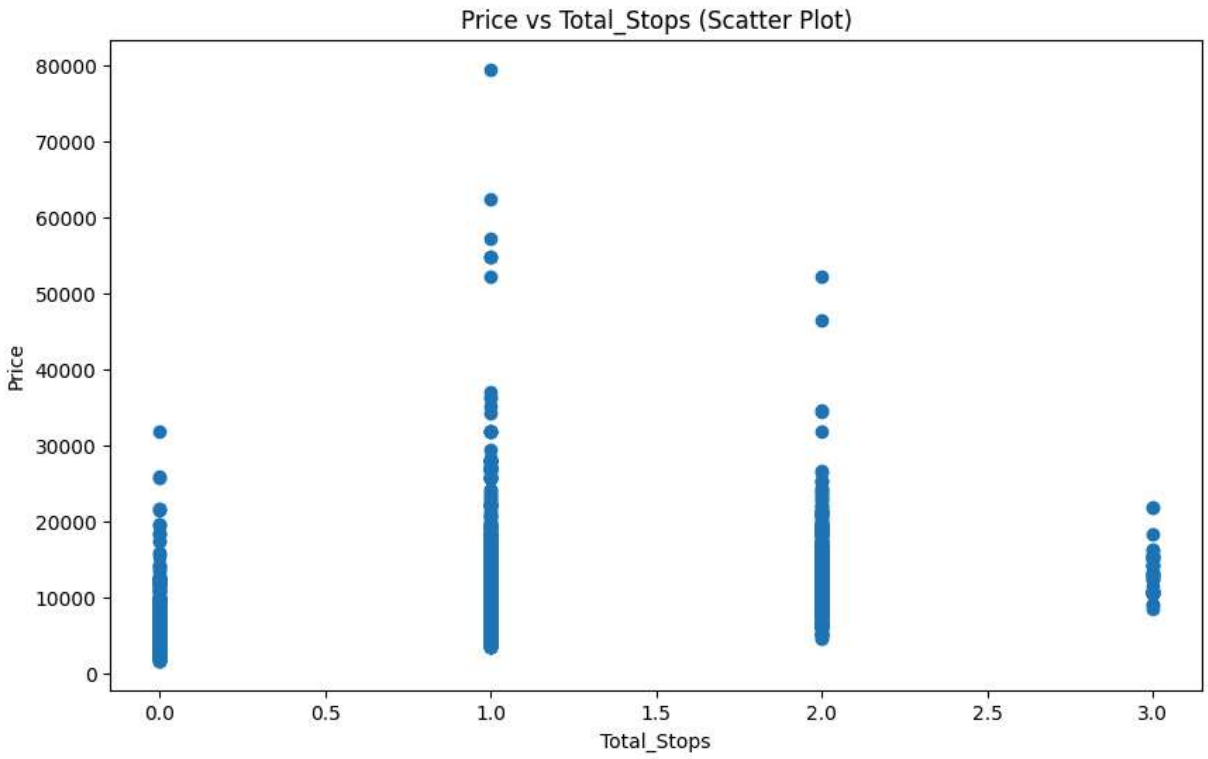
```
plt.figure(figsize=(10, 6))
plt.scatter(df1["Destination"], df1["Price"])
plt.xlabel("Destination")
plt.ylabel("Price")
plt.title("Price vs Destination (Scatter Plot)")
plt.show()
```



In [717…

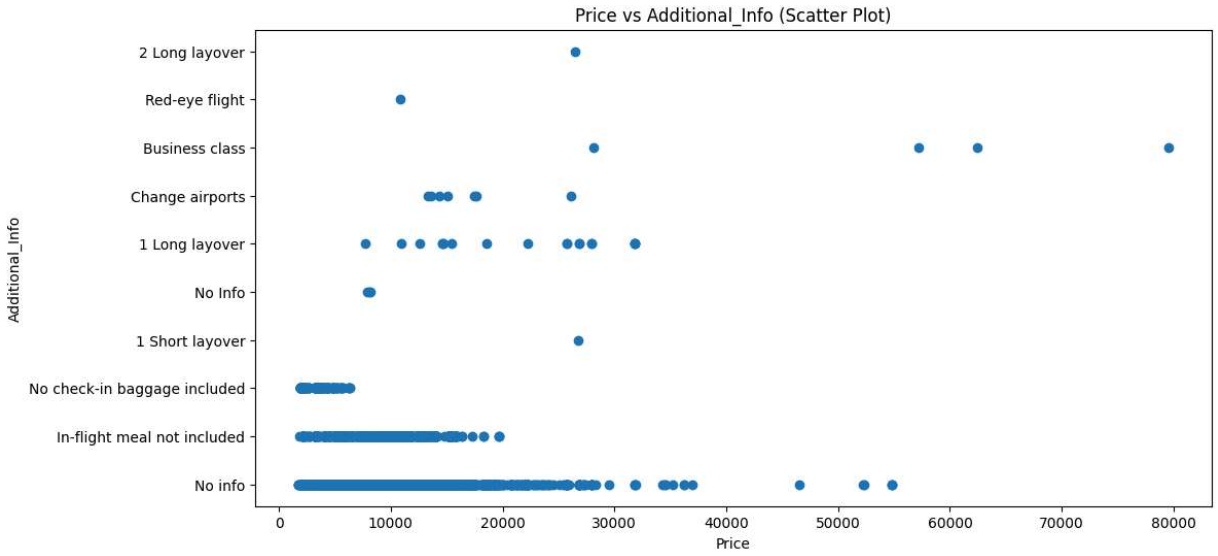
```
plt.figure(figsize=(10, 6))
plt.scatter(df1["TotalStopNumeric"], df1["Price"])
```

```
plt.xlabel("Total_Stops")
plt.ylabel("Price")
plt.title("Price vs Total_Stops (Scatter Plot)")
plt.show()
```



In [718…

```
plt.figure(figsize=(12, 6))
plt.scatter( df1["Price"],df1["Additional_Info"])
plt.ylabel("Additional_Info")
plt.xlabel("Price")
plt.title("Price vs Additional_Info (Scatter Plot)")
plt.show()
```



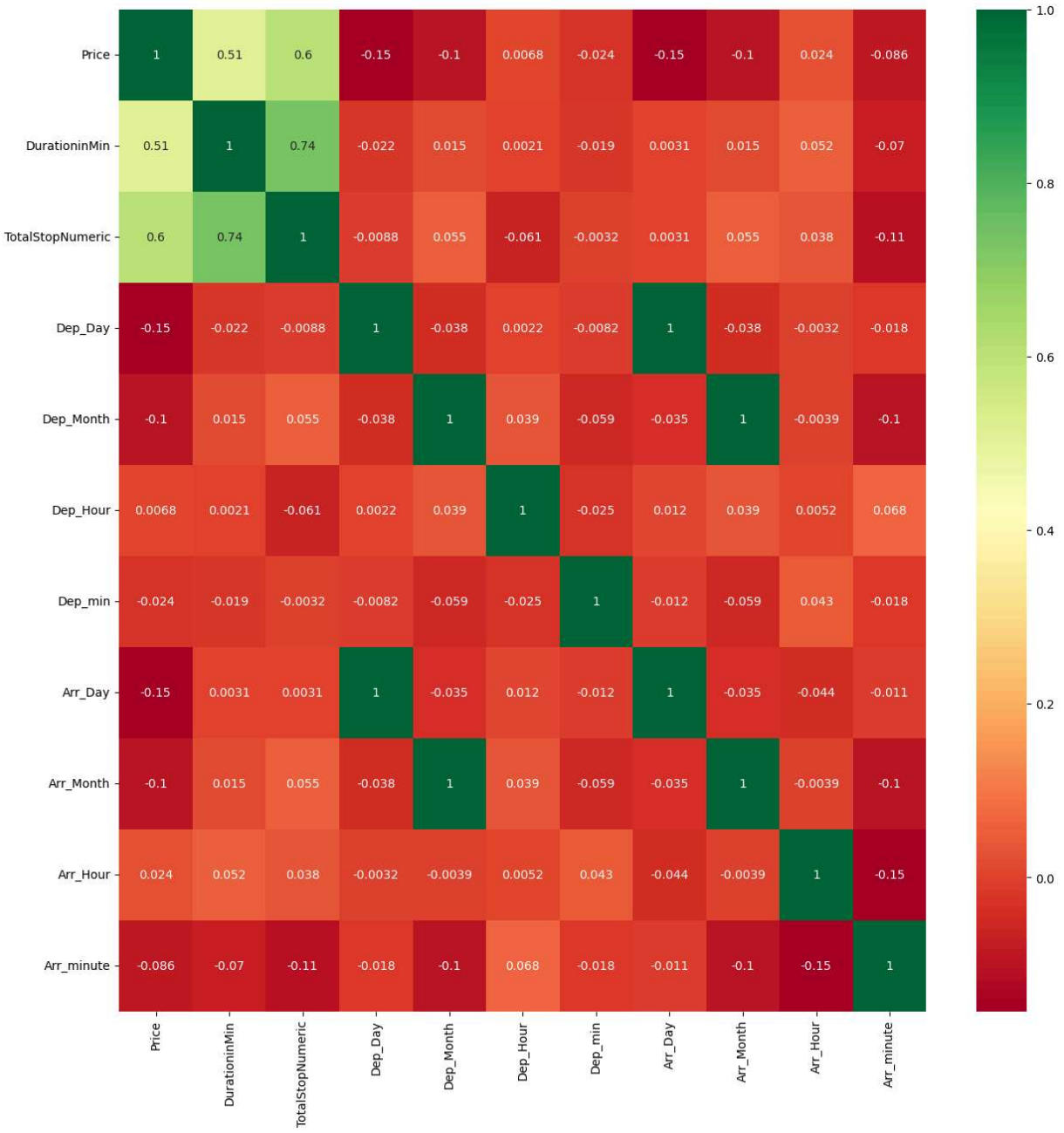
In [719…

```
df2.columns.tolist()
```

Out[719]:
```
['Airline',
 'Source',
 'Destination',
 'Route',
 'Additional_Info',
 'Price',
 'DurationinMin',
 'TotalStopNumeric',
 'Dep_Day',
 'Dep_Month',
 'Dep_Hour',
 'Dep_min',
 'Arr_Day',
 'Arr_Month',
 'Arr_Hour',
 'Arr_minute',
 'Price_log',
 'DurationinMin_log']
```

In [720…
```
Num_columns=['Price',
 'DurationinMin',
 'TotalStopNumeric',
 'Dep_Day',
 'Dep_Month',
 'Dep_Hour',
 'Dep_min',
 'Arr_Day',
 'Arr_Month',
 'Arr_Hour',
 'Arr_minute']
```

In [721…
```
plt.figure(figsize=(15,15))
sns.heatmap(df2[Num_columns].corr(),annot=True,cmap="RdYlGn")
plt.show()
```



In [722…

```
df3=df2
df3
```

Out[722]:

	Airline	Source	Destination	Route	Additional_Info	Price	DurationinMin	Total
0	IndiGo	Banglore	New Delhi	BLR ? DEL	No info	3897	170.0	
1	Air India	Kolkata	Banglore	CCU ? IXR ? BBI ? BLR	No info	7662	445.0	
2	Jet Airways	Delhi	Cochin	DEL ? LKO ? BOM ? COK	No info	13882	1140.0	
3	IndiGo	Kolkata	Banglore	CCU ? NAG ? BLR	No info	6218	325.0	
4	IndiGo	Banglore	New Delhi	BLR ? NAG ? DEL	No info	13302	285.0	
...	...	...	...	...	...	...	...	
10678	Air Asia	Kolkata	Banglore	CCU ? BLR	No info	4107	150.0	
10679	Air India	Kolkata	Banglore	CCU ? BLR	No info	4145	155.0	
10680	Jet Airways	Banglore	Delhi	BLR ? DEL	No info	7229	180.0	
10681	Vistara	Banglore	New Delhi	BLR ? DEL	No info	12648	160.0	
10682	Air India	Delhi	Cochin	DEL ? GOI ? BOM ? COK	No info	11753	500.0	

10682 rows × 18 columns

In [723…

```
df3.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 10682 entries, 0 to 10682
Data columns (total 18 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Airline            10682 non-null  object 
 1   Source             10682 non-null  object 
 2   Destination        10682 non-null  object 
 3   Route              10682 non-null  object 
 4   Additional_Info    10682 non-null  object 
 5   Price              10682 non-null  int64  
 6   DurationinMin      10682 non-null  float64
 7   TotalStopNumeric   10682 non-null  int64  
 8   Dep_Day            10682 non-null  int32  
 9   Dep_Month          10682 non-null  int32  
 10  Dep_Hour           10682 non-null  int32  
 11  Dep_min            10682 non-null  int32  
 12  Arr_Day            10682 non-null  int64  
 13  Arr_Month          10682 non-null  int64  
 14  Arr_Hour           10682 non-null  int64  
 15  Arr_minute         10682 non-null  int64  
 16  Price_log          10682 non-null  float64
 17  DurationinMin_log  10682 non-null  float64
dtypes: float64(3), int32(4), int64(6), object(5)
memory usage: 1.4+ MB
```

In [724…
```
df3.isnull().sum()
```

Out[724]:
```
Airline             0
Source              0
Destination         0
Route               0
Additional_Info     0
Price               0
DurationinMin       0
TotalStopNumeric    0
Dep_Day             0
Dep_Month           0
Dep_Hour            0
Dep_min             0
Arr_Day             0
Arr_Month           0
Arr_Hour            0
Arr_minute          0
Price_log           0
DurationinMin_log   0
dtype: int64
```

In [725…
```
df3_categorical=df3.select_dtypes(exclude=['int64','int32','float64'])
df3_numerical=df3.select_dtypes(include=['int64','int32','float64'])
```

In [726…
```
df3_categorical
```

Out[726]:

	Airline	Source	Destination	Route	Additional_Info
0	IndiGo	Banglore	New Delhi	BLR ? DEL	No info
1	Air India	Kolkata	Banglore	CCU ? IXR ? BBI ? BLR	No info
2	Jet Airways	Delhi	Cochin	DEL ? LKO ? BOM ? COK	No info
3	IndiGo	Kolkata	Banglore	CCU ? NAG ? BLR	No info
4	IndiGo	Banglore	New Delhi	BLR ? NAG ? DEL	No info
...	...	...	...	...	...
10678	Air Asia	Kolkata	Banglore	CCU ? BLR	No info
10679	Air India	Kolkata	Banglore	CCU ? BLR	No info
10680	Jet Airways	Banglore	Delhi	BLR ? DEL	No info
10681	Vistara	Banglore	New Delhi	BLR ? DEL	No info
10682	Air India	Delhi	Cochin	DEL ? GOI ? BOM ? COK	No info

10682 rows × 5 columns

In [727…

```
df3_numerical
```

Out[727]:

	Price	DurationinMin	TotalStopNumeric	Dep_Day	Dep_Month	Dep_Hour	Dep_m
0	3897	170.0	0	24	3	22	
1	7662	445.0	2	1	5	5	
2	13882	1140.0	2	9	6	9	
3	6218	325.0	1	12	5	18	
4	13302	285.0	1	1	3	16	
...	...	...	...	...	...	...	
10678	4107	150.0	0	9	4	19	
10679	4145	155.0	0	27	4	20	
10680	7229	180.0	0	27	4	8	
10681	12648	160.0	0	1	3	11	
10682	11753	500.0	2	9	5	10	

10682 rows × 13 columns

In [728…

```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
```

```
df3_categorical=df3_categorical.apply(le.fit_transform)
df3_categorical.head()
```

Out[728]:

	Airline	Source	Destination	Route	Additional_Info
0	3	0	5	18	8
1	1	3	0	84	8
2	4	2	1	118	8
3	3	3	0	91	8
4	3	0	5	29	8

In [729…

```
df4=pd.concat([df3_numerical,df3_categorical],axis=1)
df4.head()
```

Out[729]:

	Price	DurationinMin	TotalStopNumeric	Dep_Day	Dep_Month	Dep_Hour	Dep_min	A
0	3897	170.0	0	24	3	22	20	
1	7662	445.0	2	1	5	5	50	
2	13882	1140.0	2	9	6	9	25	
3	6218	325.0	1	12	5	18	5	
4	13302	285.0	1	1	3	16	50	

In [730…

```
from sklearn.model_selection import train_test_split


X = df4.drop('Price', axis=1)
y = df4['Price']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_sta
```

In [731…

```
X.head()
```

Out[731]:

	DurationinMin	TotalStopNumeric	Dep_Day	Dep_Month	Dep_Hour	Dep_min	Arr_Day
0	170.0	0	24	3	22	20	22
1	445.0	2	1	5	5	50	1
2	1140.0	2	9	6	9	25	10
3	325.0	1	12	5	18	5	12
4	285.0	1	1	3	16	50	1

In [732…

```
y.head()
```

Out[732]:
```
0     3897
1     7662
2    13882
3     6218
4    13302
Name: Price, dtype: int64
```

In [733…
```
print(f"Size of X_train: {X_train.shape}")
print(f"Size of X_test: {X_test.shape}")
print(f"Size of y_train: {y_train.shape}")
print(f"Size of y_test: {y_test.shape}")
```

```
Size of X_train: (8545, 17)
Size of X_test: (2137, 17)
Size of y_train: (8545,)
Size of y_test: (2137,)
```

In [734…
```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error, mean_squared_error
model = LinearRegression()

# Train the model
model.fit(X_train, y_train)

# Predict on the test set
y_pred = model.predict(X_test)

# Evaluate the model
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
rmse = mean_squared_error(y_test, y_pred, squared=False)
mape = round(np.mean(np.abs((y_test - y_pred) / y_test)) * 100,4)
print(f'MAPE: {mape}%')

print(f'MAE: {mae}')
print(f'MSE: {mse}')
print(f'RMSE: {rmse}')
```

```
MAPE: 12.6833%
MAE: 783.2090460622619
MSE: 1915705.0382900124
RMSE: 1384.0899675563046
```

In [735…
```
from sklearn.linear_model import Ridge

# Initialize the model
model = Ridge(alpha=1.0)

# Train the model
model.fit(X_train, y_train)

# Predict on the test set
y_pred = model.predict(X_test)

# Evaluate the model
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
```

```
rmse = mean_squared_error(y_test, y_pred, squared=False)
mape = round(np.mean(np.abs((y_test - y_pred) / y_test)) * 100,4)
print(f'MAPE: {mape}%')
print(f'MAE: {mae}')
print(f'MSE: {mse}')
print(f'RMSE: {rmse}')
```

```
MAPE: 12.6661%
MAE: 783.0090176914506
MSE: 1915441.649515739
RMSE: 1383.9948155667848
```

In [736…

```
from sklearn.linear_model import Lasso

# Initialize the model
model = Lasso(alpha=1.0)

# Train the model
model.fit(X_train, y_train)

# Predict on the test set
y_pred = model.predict(X_test)

# Evaluate the model
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
rmse = mean_squared_error(y_test, y_pred, squared=False)
mape = round(np.mean(np.abs((y_test - y_pred) / y_test)) * 100,4)
print(f'MAPE: {mape}%')
print(f'MAE: {mae}')
print(f'MSE: {mse}')
print(f'RMSE: {rmse}')
```

```
MAPE: 12.6753%
MAE: 783.1595673861253
MSE: 1915729.7139075012
RMSE: 1384.098881549834
```

In [737…

```
from sklearn.tree import DecisionTreeRegressor

# Initialize the model
model = DecisionTreeRegressor(random_state=42)

# Train the model
model.fit(X_train, y_train)

# Predict on the test set
y_pred = model.predict(X_test)

# Evaluate the model
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
rmse = mean_squared_error(y_test, y_pred, squared=False)
mape = round(np.mean(np.abs((y_test - y_pred) / y_test)) * 100,4)
print(f'MAPE: {mape}%')

print(f'MAE: {mae}')
```

```
print(f'MSE: {mse}')
print(f'RMSE: {rmse}')
```

```
MAPE: 0.0226%
MAE: 5.098736546560599
MSE: 12553.337388862892
RMSE: 112.04167701736212
```

In [738…

```
from sklearn.ensemble import RandomForestRegressor

# Initialize the model
model = RandomForestRegressor(n_estimators=100, random_state=42)

# Train the model
model.fit(X_train, y_train)

# Predict on the test set
y_pred = model.predict(X_test)

# Evaluate the model
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
rmse = mean_squared_error(y_test, y_pred, squared=False)

mape = round(np.mean(np.abs((y_test - y_pred) / y_test)) * 100,4)
print(f'MAPE: {mape}%')

print(f'MAE: {mae}')
print(f'MSE: {mse}')
print(f'RMSE: {rmse}')
```

```
MAPE: 0.0219%
MAE: 4.684365933551718
MSE: 11194.563041226027
RMSE: 105.80436210868636
```

In [739…

```
from sklearn.svm import SVR

# Initialize the model
model = SVR(kernel='rbf')

# Train the model
model.fit(X_train, y_train)

# Predict on the test set
y_pred = model.predict(X_test)

# Evaluate the model
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
rmse = mean_squared_error(y_test, y_pred, squared=False)
mape = round(np.mean(np.abs((y_test - y_pred) / y_test)) * 100,4)
print(f'MAPE: {mape}%')

print(f'MAE: {mae}')
```

```
print(f'MSE: {mse}')
print(f'RMSE: {rmse}')
```

```
MAPE: 40.4657%
MAE: 3081.2626118685057
MSE: 18193187.47472896
RMSE: 4265.347286532359
```

In [740…

```
import numpy as np
from keras.models import Sequential
from keras.layers import Dense

# Initialize the model
model = Sequential()
model.add(Dense(64, input_dim=X_train.shape[1], activation='relu'))
model.add(Dense(32, activation='relu'))
model.add(Dense(1))

# Compile the model
model.compile(optimizer='adam', loss='mean_squared_error')

# Train the model
model.fit(X_train, y_train, epochs=50, batch_size=10, verbose=1, validation_split=0

# Predict on the test set
y_pred = model.predict(X_test)

# Evaluate the model
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
rmse = mean_squared_error(y_test, y_pred, squared=False)


print(f'MAE: {mae}')
print(f'MSE: {mse}')
print(f'RMSE: {rmse}')
```

```
Epoch 1/50
684/684 [==============================] - 3s 2ms/step - loss: 30535182.0000 - val_l
oss: 15541286.0000
Epoch 2/50
684/684 [==============================] - 1s 2ms/step - loss: 17548858.0000 - val_l
oss: 14235487.0000
Epoch 3/50
684/684 [==============================] - 1s 2ms/step - loss: 16104950.0000 - val_l
oss: 12900492.0000
Epoch 4/50
684/684 [==============================] - 1s 2ms/step - loss: 14793209.0000 - val_l
oss: 11770262.0000
Epoch 5/50
684/684 [==============================] - 1s 2ms/step - loss: 13928226.0000 - val_l
oss: 11137416.0000
Epoch 6/50
684/684 [==============================] - 1s 2ms/step - loss: 13351492.0000 - val_l
oss: 10982194.0000
Epoch 7/50
684/684 [==============================] - 1s 2ms/step - loss: 12987088.0000 - val_l
oss: 10496225.0000
Epoch 8/50
684/684 [==============================] - 1s 2ms/step - loss: 12656712.0000 - val_l
oss: 10200371.0000
Epoch 9/50
684/684 [==============================] - 1s 2ms/step - loss: 12327094.0000 - val_l
oss: 9949919.0000
Epoch 10/50
684/684 [==============================] - 1s 2ms/step - loss: 12103552.0000 - val_l
oss: 9688356.0000
Epoch 11/50
684/684 [==============================] - 1s 2ms/step - loss: 11832748.0000 - val_l
oss: 9806750.0000
Epoch 12/50
684/684 [==============================] - 1s 2ms/step - loss: 11689089.0000 - val_l
oss: 9393645.0000
Epoch 13/50
684/684 [==============================] - 1s 2ms/step - loss: 11492373.0000 - val_l
oss: 9216104.0000
Epoch 14/50
684/684 [==============================] - 1s 2ms/step - loss: 11281938.0000 - val_l
oss: 9119416.0000
Epoch 15/50
684/684 [==============================] - 1s 2ms/step - loss: 11162610.0000 - val_l
oss: 9340015.0000
Epoch 16/50
684/684 [==============================] - 1s 2ms/step - loss: 11061648.0000 - val_l
oss: 8813823.0000
Epoch 17/50
684/684 [==============================] - 1s 2ms/step - loss: 10866134.0000 - val_l
oss: 8714253.0000
Epoch 18/50
684/684 [==============================] - 1s 2ms/step - loss: 10751480.0000 - val_l
oss: 8716580.0000
Epoch 19/50
684/684 [==============================] - 1s 2ms/step - loss: 10695619.0000 - val_l
```

```
oss: 8588970.0000
Epoch 20/50
684/684 [==============================] - 1s 2ms/step - loss: 10583574.0000 - val_l
oss: 9664596.0000
Epoch 21/50
684/684 [==============================] - 1s 2ms/step - loss: 10511421.0000 - val_l
oss: 8466673.0000
Epoch 22/50
684/684 [==============================] - 1s 2ms/step - loss: 10373984.0000 - val_l
oss: 8412221.0000
Epoch 23/50
684/684 [==============================] - 1s 2ms/step - loss: 10289447.0000 - val_l
oss: 8192175.0000
Epoch 24/50
684/684 [==============================] - 1s 2ms/step - loss: 10203809.0000 - val_l
oss: 8211860.5000
Epoch 25/50
684/684 [==============================] - 1s 2ms/step - loss: 10099496.0000 - val_l
oss: 8216289.5000
Epoch 26/50
684/684 [==============================] - 1s 2ms/step - loss: 10057754.0000 - val_l
oss: 8096488.0000
Epoch 27/50
684/684 [==============================] - 1s 2ms/step - loss: 9982343.0000 - val_lo
ss: 7921939.0000
Epoch 28/50
684/684 [==============================] - 1s 2ms/step - loss: 9889281.0000 - val_lo
ss: 7932483.5000
Epoch 29/50
684/684 [==============================] - 1s 2ms/step - loss: 9845152.0000 - val_lo
ss: 7824013.0000
Epoch 30/50
684/684 [==============================] - 1s 2ms/step - loss: 9772753.0000 - val_lo
ss: 8037556.0000
Epoch 31/50
684/684 [==============================] - 1s 2ms/step - loss: 9705394.0000 - val_lo
ss: 8343362.5000
Epoch 32/50
684/684 [==============================] - 1s 2ms/step - loss: 9639931.0000 - val_lo
ss: 7707582.0000
Epoch 33/50
684/684 [==============================] - 1s 2ms/step - loss: 9550010.0000 - val_lo
ss: 7595823.0000
Epoch 34/50
684/684 [==============================] - 1s 2ms/step - loss: 9443407.0000 - val_lo
ss: 7591621.0000
Epoch 35/50
684/684 [==============================] - 1s 2ms/step - loss: 9382306.0000 - val_lo
ss: 7511890.0000
Epoch 36/50
684/684 [==============================] - 1s 2ms/step - loss: 9381212.0000 - val_lo
ss: 7358943.5000
Epoch 37/50
684/684 [==============================] - 1s 2ms/step - loss: 9270569.0000 - val_lo
ss: 7326869.0000
Epoch 38/50
```

```
684/684 [==============================] - 1s 2ms/step - loss: 9132798.0000 - val_lo
ss: 7648306.5000
Epoch 39/50
684/684 [==============================] - 1s 2ms/step - loss: 9056798.0000 - val_lo
ss: 7248926.0000
Epoch 40/50
684/684 [==============================] - 1s 2ms/step - loss: 8972745.0000 - val_lo
ss: 7267108.0000
Epoch 41/50
684/684 [==============================] - 1s 2ms/step - loss: 8965803.0000 - val_lo
ss: 7273823.5000
Epoch 42/50
684/684 [==============================] - 1s 2ms/step - loss: 8804771.0000 - val_lo
ss: 7166815.0000
Epoch 43/50
684/684 [==============================] - 1s 2ms/step - loss: 8816127.0000 - val_lo
ss: 7309501.5000
Epoch 44/50
684/684 [==============================] - 1s 2ms/step - loss: 8679548.0000 - val_lo
ss: 6923695.0000
Epoch 45/50
684/684 [==============================] - 1s 2ms/step - loss: 8566074.0000 - val_lo
ss: 6808769.0000
Epoch 46/50
684/684 [==============================] - 1s 2ms/step - loss: 8639710.0000 - val_lo
ss: 6795716.5000
Epoch 47/50
684/684 [==============================] - 1s 2ms/step - loss: 8485186.0000 - val_lo
ss: 6719898.0000
Epoch 48/50
684/684 [==============================] - 1s 2ms/step - loss: 8385021.5000 - val_lo
ss: 6932751.5000
Epoch 49/50
684/684 [==============================] - 1s 2ms/step - loss: 8300499.5000 - val_lo
ss: 7062167.5000
Epoch 50/50
684/684 [==============================] - 1s 2ms/step - loss: 8149290.0000 - val_lo
ss: 6735060.5000
67/67 [==============================] - 0s 1ms/step
MAE: 1934.8634500463377
MSE: 7438783.998030022
RMSE: 2727.4134263125607
```

In [741…

```python
import numpy as np
import xgboost as xgb
from keras.models import Sequential
from keras.layers import Dense, Dropout
from keras.regularizers import l2
from sklearn.linear_model import LinearRegression, Ridge, Lasso
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.svm import SVR
from sklearn.metrics import mean_absolute_error, mean_squared_error
from sklearn.model_selection import train_test_split

# Assuming df is your DataFrame and 'price' is your target column
```

```
X = df4.drop('Price', axis=1)
y = df4['Price']

# Split the data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_sta

# Define models
models = {
    'Linear Regression': LinearRegression(),
    'Ridge Regression': Ridge(alpha=1.0),
    'Lasso Regression': Lasso(alpha=1.0),
    'Decision Tree Regression': DecisionTreeRegressor(random_state=42),
    'Random Forest Regression': RandomForestRegressor(n_estimators=100, random_stat
    'XGBoost Regression': xgb.XGBRegressor(objective='reg:squarederror', n_estimato
    'Support Vector Regression': SVR(kernel='rbf'),
    'Neural Network': Sequential([
        Dense(64, input_dim=X_train.shape[1], activation='relu', kernel_regularizer
        Dropout(0.2),
        Dense(32, activation='relu', kernel_regularizer=l2(0.01)),
        Dense(1)
    ])
}

results = {}

for name, model in models.items():
    if name == 'Neural Network':
        model.compile(optimizer='adam', loss='mean_squared_error')
        history = model.fit(X_train, y_train, epochs=50, batch_size=10, verbose=0,
        y_pred = model.predict(X_test).flatten()
    else:
        model.fit(X_train, y_train)
        y_pred = model.predict(X_test)

    # Calculate metrics
    mae = mean_absolute_error(y_test, y_pred)
    mse = mean_squared_error(y_test, y_pred)
    rmse = mean_squared_error(y_test, y_pred, squared=False)
    mape = np.mean(np.abs((y_test - y_pred) / y_test)) * 100

    # Store results
    results[name] = {'MAE': mae, 'MSE': mse, 'RMSE': rmse, 'MAPE': mape}

# Display results
for name, metrics in results.items():
    print(f"{name}:")
    for metric_name, value in metrics.items():
        print(f"  {metric_name}: {value:.4f}")
    print()

# Find the best model based on MAPE
best_model_name = min(results, key=lambda x: results[x]['MAPE'])
best_model_metrics = results[best_model_name]

print(f"Best Model: {best_model_name}")
```

```
for metric_name, value in best_model_metrics.items():
    print(f"{metric_name}: {value:.4f}")
```

```
67/67 [==============================] - 0s 1ms/step
Linear Regression:
  MAE: 783.2090
  MSE: 1915705.0383
  RMSE: 1384.0900
  MAPE: 12.6833

Ridge Regression:
  MAE: 783.0090
  MSE: 1915441.6495
  RMSE: 1383.9948
  MAPE: 12.6661

Lasso Regression:
  MAE: 783.1596
  MSE: 1915729.7139
  RMSE: 1384.0989
  MAPE: 12.6753

Decision Tree Regression:
  MAE: 5.0987
  MSE: 12553.3374
  RMSE: 112.0417
  MAPE: 0.0226

Random Forest Regression:
  MAE: 4.6844
  MSE: 11194.5630
  RMSE: 105.8044
  MAPE: 0.0219

XGBoost Regression:
  MAE: 62.1597
  MSE: 418748.7924
  RMSE: 647.1080
  MAPE: 0.3509

Support Vector Regression:
  MAE: 3081.2626
  MSE: 18193187.4747
  RMSE: 4265.3473
  MAPE: 40.4657

Neural Network:
  MAE: 2019.0243
  MSE: 8045137.0742
  RMSE: 2836.3951
  MAPE: 25.2309

Best Model: Random Forest Regression
MAE: 4.6844
MSE: 11194.5630
RMSE: 105.8044
MAPE: 0.0219
```

In [742…

```
metric_names = ['MAE', 'MSE', 'RMSE', 'MAPE']
metric_values = {metric: [results[model][metric] for model in models.keys()] for me

fig, axs = plt.subplots(2, 2, figsize=(15, 12))
axs[0, 0].bar(models.keys(), metric_values['MAE'], color='skyblue')
axs[0, 0].set_title('Mean Absolute Error (MAE)')
axs[0, 0].set_xticklabels(models.keys(), rotation=45, ha='right')

axs[0, 1].bar(models.keys(), metric_values['MSE'], color='lightgreen')
axs[0, 1].set_title('Mean Squared Error (MSE)')
axs[0, 1].set_xticklabels(models.keys(), rotation=45, ha='right')

axs[1, 0].bar(models.keys(), metric_values['RMSE'], color='salmon')
axs[1, 0].set_title('Root Mean Squared Error (RMSE)')
axs[1, 0].set_xticklabels(models.keys(), rotation=45, ha='right')

axs[1, 1].bar(models.keys(), metric_values['MAPE'], color='orange')
axs[1, 1].set_title('Mean Absolute Percentage Error (MAPE)')
axs[1, 1].set_xticklabels(models.keys(), rotation=45, ha='right')

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