Predicting Flight Ticket Prices

```
# Author : Shaik Mastanvali
#Email:mastanvalishaik661@gmail.com
# The Model is concerned with the flight price prediction
```

Import Libraries and Data

```
# Importing necessary libraries
import numpy as np
import pandas as pd
import os
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')

# Import The Data Set
df= pd.read_excel('Data_Train.xlsx')

# Displaying the initial rows of the dataset
print("Initial few rows of the dataset: ")
df.head(20)
```

```
Initial few rows of the dataset:

              Airline Date_of_Journey    Source Destination  \
0              IndiGo      24/03/2019  Banglore   New Delhi
1           Air India       1/05/2019   Kolkata    Banglore
2         Jet Airways       9/06/2019     Delhi      Cochin
3              IndiGo      12/05/2019   Kolkata    Banglore
4              IndiGo      01/03/2019  Banglore   New Delhi
5            SpiceJet      24/06/2019   Kolkata    Banglore
6         Jet Airways      12/03/2019  Banglore   New Delhi
7         Jet Airways      01/03/2019  Banglore   New Delhi
8         Jet Airways      12/03/2019  Banglore   New Delhi
9   Multiple carriers      27/05/2019     Delhi      Cochin
10          Air India       1/06/2019     Delhi      Cochin
11             IndiGo      18/04/2019   Kolkata    Banglore
12          Air India      24/06/2019   Chennai     Kolkata
13        Jet Airways       9/05/2019   Kolkata    Banglore
14             IndiGo      24/04/2019   Kolkata    Banglore
15          Air India       3/03/2019     Delhi      Cochin
16           SpiceJet      15/04/2019     Delhi      Cochin
17        Jet Airways      12/06/2019     Delhi      Cochin
18          Air India      12/06/2019     Delhi      Cochin
19        Jet Airways      27/05/2019     Delhi      Cochin

                    Route Dep_Time  Arrival_Time Duration Total_Stops
\
```

0	BLR → DEL	22:20	01:10 22 Mar	2h 50m	non-stop
1	CCU → IXR → BBI → BLR	05:50	13:15	7h 25m	2 stops
2	DEL → LKO → BOM → COK	09:25	04:25 10 Jun	19h	2 stops
3	CCU → NAG → BLR	18:05	23:30	5h 25m	1 stop
4	BLR → NAG → DEL	16:50	21:35	4h 45m	1 stop
5	CCU → BLR	09:00	11:25	2h 25m	non-stop
6	BLR → BOM → DEL	18:55	10:25 13 Mar	15h 30m	1 stop
7	BLR → BOM → DEL	08:00	05:05 02 Mar	21h 5m	1 stop
8	BLR → BOM → DEL	08:55	10:25 13 Mar	25h 30m	1 stop
9	DEL → BOM → COK	11:25	19:15	7h 50m	1 stop
10	DEL → BLR → COK	09:45	23:00	13h 15m	1 stop
11	CCU → BLR	20:20	22:55	2h 35m	non-stop
12	MAA → CCU	11:40	13:55	2h 15m	non-stop
13	CCU → BOM → BLR	21:10	09:20 10 May	12h 10m	1 stop
14	CCU → BLR	17:15	19:50	2h 35m	non-stop
15	DEL → AMD → BOM → COK	16:40	19:15 04 Mar	26h 35m	2 stops
16	DEL → PNQ → COK	08:45	13:15	4h 30m	1 stop
17	DEL → BOM → COK	14:00	12:35 13 Jun	22h 35m	1 stop
18	DEL → CCU → BOM → COK	20:15	19:15 13 Jun	23h	2 stops
19	DEL → BOM → COK	16:00	12:35 28 May	20h 35m	1 stop

	Additional_Info	Price
0	Null	3897
1	Null	7662
2	Null	13882
3	Null	6218
4	Null	13302
5	Null	3873
6	In-flight meal not included	11087
7	Null	22270
8	In-flight meal not included	11087

```
9                         Null   8625
10                        Null   8907
11                        Null   4174
12                        Null   4667
13  In-flight meal not included   9663
14                        Null   4804
15                        Null  14011
16                        Null   5830
17  In-flight meal not included  10262
18                        Null  13381
19  In-flight meal not included  12898
```

```
# Getting an overview of total no of rows and column in the dataset
print("\nOverview of the total no of rows and column:")
df.shape
```

```
Overview of the total no of rows and column:

(10683, 11)
```

```
# Getting an overview of the features and their types in the dataset
print("\nOverview of the features and their types:")
df.info()
```

```
Overview of the features and their types:
<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
```

```
# Getting an overview of the dataset
print("\nOverview of the dataset:")
df.describe()
```

```
Overview of the dataset:
```

	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

```python
# Getting an overview of the dataset including all
print("\nOverview of the dataset:")
df.describe(include='all').T
```

```
Overview of the dataset:
```

	count	unique	top	freq	mean \
Airline	10683	12	Jet Airways	3849	NaN
Date_of_Journey	10683	44	18/05/2019	504	NaN
Source	10683	5	Delhi	4537	NaN
Destination	10683	6	Cochin	4537	NaN
Route	10682	128	DEL → BOM → COK	2376	NaN
Dep_Time	10683	222	18:55	233	NaN
Arrival_Time	10683	1343	19:00	423	NaN
Duration	10683	368	2h 50m	550	NaN
Total_Stops	10682	5	1 stop	5625	NaN
Additional_Info	10683	10	Null	8347	NaN
Price	10683.0	NaN	NaN	NaN	9087.064121

	std	min	25%	50%	75%	max
Airline	NaN	NaN	NaN	NaN	NaN	NaN
Date_of_Journey	NaN	NaN	NaN	NaN	NaN	NaN
Source	NaN	NaN	NaN	NaN	NaN	NaN
Destination	NaN	NaN	NaN	NaN	NaN	NaN
Route	NaN	NaN	NaN	NaN	NaN	NaN
Dep_Time	NaN	NaN	NaN	NaN	NaN	NaN
Arrival_Time	NaN	NaN	NaN	NaN	NaN	NaN
Duration	NaN	NaN	NaN	NaN	NaN	NaN

```
Total_Stops                NaN      NaN      NaN      NaN       NaN       NaN

Additional_Info            NaN      NaN      NaN      NaN       NaN       NaN

Price              4611.359167   1759.0   5277.0   8372.0   12373.0   79512.0
```

```
# Getting an overview of the dataset including Object Type
print("\nOverview of the dataset:")
df.describe(include='O').T
```

```
Overview of the dataset:
```

	count	unique	top	freq
Airline	10683	12	Jet Airways	3849
Date_of_Journey	10683	44	18/05/2019	504
Source	10683	5	Delhi	4537
Destination	10683	6	Cochin	4537
Route	10682	128	DEL → BOM → COK	2376
Dep_Time	10683	222	18:55	233
Arrival_Time	10683	1343	19:00	423
Duration	10683	368	2h 50m	550
Total_Stops	10682	5	1 stop	5625
Additional_Info	10683	10	Null	8347

```
df.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
```

TO FIND UNIQUE VALUES IN EACH COLUMN

```
for i in df.columns:
    print(f"The Unique Values in feature {i} is",df[i].unique(),sep='\
n')

print("************************************************************")
```

```
The Unique Values in feature Airline is
['IndiGo' 'Air India' 'Jet Airways' 'SpiceJet' 'Multiple carriers'
```

```
'GoAir'
 'Vistara' 'Air Asia' 'Vistara Premium economy' 'Jet Airways Business'
 'Multiple carriers Premium economy' 'Trujet']
*********************************************************
The Unique Values in feature Date_of_Journey is
['24/03/2019' '1/05/2019' '9/06/2019' '12/05/2019' '01/03/2019'
 '24/06/2019' '12/03/2019' '27/05/2019' '1/06/2019' '18/04/2019'
 '9/05/2019' '24/04/2019' '3/03/2019' '15/04/2019' '12/06/2019'
 '6/03/2019' '21/03/2019' '3/04/2019' '6/05/2019' '15/05/2019'
 '18/06/2019' '15/06/2019' '6/04/2019' '18/05/2019' '27/06/2019'
 '21/05/2019' '06/03/2019' '3/06/2019' '15/03/2019' '3/05/2019'
 '9/03/2019' '6/06/2019' '24/05/2019' '09/03/2019' '1/04/2019'
 '21/04/2019' '21/06/2019' '27/03/2019' '18/03/2019' '12/04/2019'
 '9/04/2019' '1/03/2019' '03/03/2019' '27/04/2019']
*********************************************************
The Unique Values in feature Source is
['Banglore' 'Kolkata' 'Delhi' 'Chennai' 'Mumbai']
*********************************************************
The Unique Values in feature Destination is
['New Delhi' 'Banglore' 'Cochin' 'Kolkata' 'Delhi' 'Hyderabad']
*********************************************************
The Unique Values in feature Route is
['BLR → DEL' 'CCU → IXR → BBI → BLR' 'DEL → LKO → BOM → COK'
 'CCU → NAG → BLR' 'BLR → NAG → DEL' 'CCU → BLR' 'BLR → BOM → DEL'
 'DEL → BOM → COK' 'DEL → BLR → COK' 'MAA → CCU' 'CCU → BOM → BLR'
 'DEL → AMD → BOM → COK' 'DEL → PNQ → COK' 'DEL → CCU → BOM → COK'
 'BLR → COK → DEL' 'DEL → IDR → BOM → COK' 'DEL → LKO → COK'
 'CCU → GAU → DEL → BLR' 'DEL → NAG → BOM → COK' 'CCU → MAA → BLR'
 'DEL → HYD → COK' 'CCU → HYD → BLR' 'DEL → COK' 'CCU → DEL → BLR'
 'BLR → BOM → AMD → DEL' 'BOM → DEL → HYD' 'DEL → MAA → COK' 'BOM →
HYD'
 'DEL → BHO → BOM → COK' 'DEL → JAI → BOM → COK' 'DEL → ATQ → BOM →
COK'
 'DEL → JDH → BOM → COK' 'CCU → BBI → BOM → BLR' 'BLR → MAA → DEL'
 'DEL → GOI → BOM → COK' 'DEL → BDQ → BOM → COK' 'CCU → JAI → BOM →
BLR'
 'CCU → BBI → BLR' 'BLR → HYD → DEL' 'DEL → TRV → COK'
 'CCU → IXR → DEL → BLR' 'DEL → IXU → BOM → COK' 'CCU → IXB → BLR'
 'BLR → BOM → JDH → DEL' 'DEL → UDR → BOM → COK' 'DEL → HYD → MAA →
COK'
 'CCU → BOM → COK → BLR' 'BLR → CCU → DEL' 'CCU → BOM → GOI → BLR'
 'DEL → RPR → NAG → BOM → COK' 'DEL → HYD → BOM → COK'
 'CCU → DEL → AMD → BLR' 'CCU → PNQ → BLR' 'BLR → CCU → GAU → DEL'
 'CCU → DEL → COK → BLR' 'BLR → PNQ → DEL' 'BOM → JDH → DEL → HYD'
 'BLR → BOM → BHO → DEL' 'DEL → AMD → COK' 'BLR → LKO → DEL'
 'CCU → GAU → BLR' 'BOM → GOI → HYD' 'CCU → BOM → AMD → BLR'
 'CCU → BBI → IXR → DEL → BLR' 'DEL → DED → BOM → COK'
 'DEL → MAA → BOM → COK' 'BLR → AMD → DEL' 'BLR → VGA → DEL'
 'CCU → JAI → DEL → BLR' 'CCU → AMD → BLR' 'CCU → VNS → DEL → BLR'
```

```
 'BLR → BOM → IDR → DEL' 'BLR → BBI → DEL' 'BLR → GOI → DEL'
 'BOM → AMD → ISK → HYD' 'BOM → DED → DEL → HYD' 'DEL → IXC → BOM →
COK'
 'CCU → PAT → BLR' 'BLR → CCU → BBI → DEL' 'CCU → BBI → HYD → BLR'
 'BLR → BOM → NAG → DEL' 'BLR → CCU → BBI → HYD → DEL' 'BLR → GAU →
DEL'
 'BOM → BHO → DEL → HYD' 'BOM → JLR → HYD' 'BLR → HYD → VGA → DEL'
 'CCU → KNU → BLR' 'CCU → BOM → PNQ → BLR' 'DEL → BBI → COK'
 'BLR → VGA → HYD → DEL' 'BOM → JDH → JAI → DEL → HYD'
 'DEL → GWL → IDR → BOM → COK' 'CCU → RPR → HYD → BLR' 'CCU → VTZ →
BLR'
 'CCU → DEL → VGA → BLR' 'BLR → BOM → IDR → GWL → DEL'
 'CCU → DEL → COK → TRV → BLR' 'BOM → COK → MAA → HYD' 'BOM → NDC →
HYD'
 'BLR → BDQ → DEL' 'CCU → BOM → TRV → BLR' 'CCU → BOM → HBX → BLR'
 'BOM → BDQ → DEL → HYD' 'BOM → CCU → HYD' 'BLR → TRV → COK → DEL'
 'BLR → IDR → DEL' 'CCU → IXZ → MAA → BLR' 'CCU → GAU → IMF → DEL →
BLR'
 'BOM → GOI → PNQ → HYD' 'BOM → BLR → CCU → BBI → HYD' 'BOM → MAA →
HYD'
 'BLR → BOM → UDR → DEL' 'BOM → UDR → DEL → HYD' 'BLR → VGA → VTZ →
DEL'
 'BLR → HBX → BOM → BHO → DEL' 'CCU → IXA → BLR' 'BOM → RPR → VTZ →
HYD'
 'BLR → HBX → BOM → AMD → DEL' 'BOM → IDR → DEL → HYD' 'BOM → BLR →
HYD'
 'BLR → STV → DEL' 'CCU → IXB → DEL → BLR' 'BOM → JAI → DEL → HYD'
 'BOM → VNS → DEL → HYD' 'BLR → HBX → BOM → NAG → DEL' nan
 'BLR → BOM → IXC → DEL' 'BLR → CCU → BBI → HYD → VGA → DEL'
 'BOM → BBI → HYD']
*********************************************************
The Unique Values in feature Dep_Time is
['22:20' '05:50' '09:25' '18:05' '16:50' '09:00' '18:55' '08:00'
'08:55'
 '11:25' '09:45' '20:20' '11:40' '21:10' '17:15' '16:40' '08:45'
'14:00'
 '20:15' '16:00' '14:10' '22:00' '04:00' '21:25' '21:50' '07:00'
'07:05'
 '09:50' '14:35' '10:35' '15:05' '14:15' '06:45' '20:55' '11:10'
'05:45'
 '19:00' '23:05' '11:00' '09:35' '21:15' '23:55' '19:45' '08:50'
'15:40'
 '06:05' '15:00' '13:55' '05:55' '13:20' '05:05' '06:25' '17:30'
'08:20'
 '19:55' '06:30' '14:05' '02:00' '09:40' '08:25' '20:25' '13:15'
'02:15'
 '16:55' '20:45' '05:15' '19:50' '20:00' '06:10' '19:30' '04:45'
'12:55'
 '18:15' '17:20' '15:25' '23:00' '12:00' '14:45' '11:50' '11:30'
```

```
'14:40'
 '19:10' '06:00' '23:30' '07:35' '13:05' '12:30' '15:10' '12:50'
'18:25'
 '16:30' '00:40' '06:50' '13:00' '19:15' '01:30' '17:00' '10:00'
'19:35'
 '15:30' '12:10' '16:10' '20:35' '22:25' '21:05' '05:35' '05:10'
'06:40'
 '15:15' '00:30' '08:30' '07:10' '05:30' '14:25' '05:25' '10:20'
'17:45'
 '13:10' '22:10' '04:55' '17:50' '21:20' '06:20' '15:55' '20:30'
'17:25'
 '09:30' '07:30' '02:35' '10:55' '17:10' '09:10' '18:45' '15:20'
'22:50'
 '14:55' '14:20' '13:25' '22:15' '11:05' '16:15' '20:10' '06:55'
'19:05'
 '07:55' '07:45' '10:10' '08:15' '11:35' '21:00' '17:55' '16:45'
'18:20'
 '03:50' '08:35' '19:20' '20:05' '17:40' '04:40' '17:35' '09:55'
'05:00'
 '18:00' '02:55' '20:40' '22:55' '22:40' '21:30' '08:10' '17:05'
'07:25'
 '15:45' '09:15' '15:50' '11:45' '22:05' '18:35' '00:25' '19:40'
'20:50'
 '22:45' '10:30' '23:25' '11:55' '10:45' '11:15' '12:20' '14:30'
'07:15'
 '01:35' '18:40' '09:20' '21:55' '13:50' '01:40' '00:20' '04:15'
'13:45'
 '18:30' '06:15' '02:05' '12:15' '13:30' '06:35' '10:05' '08:40'
'03:05'
 '21:35' '16:35' '02:30' '16:25' '05:40' '15:35' '13:40' '07:20'
'04:50'
 '12:45' '10:25' '12:05' '11:20' '21:40' '03:00']
*********************************************************
The Unique Values in feature Arrival_Time is
['01:10 22 Mar' '13:15' '04:25 10 Jun' ... '06:50 10 Mar' '00:05 19
Mar'
 '21:20 13 Mar']
*********************************************************
The Unique Values in feature Duration is
['2h 50m' '7h 25m' '19h' '5h 25m' '4h 45m' '2h 25m' '15h 30m' '21h 5m'
 '25h 30m' '7h 50m' '13h 15m' '2h 35m' '2h 15m' '12h 10m' '26h 35m'
 '4h 30m' '22h 35m' '23h' '20h 35m' '5h 10m' '15h 20m' '2h 55m' '13h
20m'
 '15h 10m' '5h 45m' '5h 55m' '13h 25m' '22h' '5h 30m' '10h 25m' '5h
15m'
 '2h 30m' '6h 15m' '11h 55m' '11h 5m' '8h 30m' '22h 5m' '2h 45m' '12h'
 '16h 5m' '19h 55m' '3h 15m' '25h 20m' '3h' '16h 15m' '15h 5m' '6h
30m'
 '25h 5m' '12h 25m' '27h 20m' '10h 15m' '10h 30m' '1h 30m' '1h 25m'
```

```
 '26h 30m' '7h 20m' '13h 30m' '5h' '19h 5m' '14h 50m' '2h 40m' '22h
10m'
 '9h 35m' '10h' '21h 20m' '18h 45m' '12h 20m' '18h' '9h 15m' '17h 30m'
 '16h 35m' '12h 15m' '7h 30m' '24h' '8h 55m' '7h 10m' '14h 30m' '30h
20m'
 '15h' '12h 45m' '10h 10m' '15h 25m' '14h 5m' '20h 15m' '23h 10m'
 '18h 10m' '16h' '2h 20m' '8h' '16h 55m' '3h 10m' '14h' '23h 50m'
 '21h 40m' '21h 15m' '10h 50m' '8h 15m' '8h 35m' '11h 50m' '27h 35m'
 '8h 25m' '20h 55m' '4h 50m' '8h 10m' '24h 25m' '23h 35m' '25h 45m'
 '26h 10m' '28h 50m' '25h 15m' '9h 20m' '9h 10m' '3h 5m' '11h 30m'
 '9h 30m' '17h 35m' '5h 5m' '25h 50m' '20h' '13h' '18h 25m' '24h 10m'
 '4h 55m' '25h 35m' '6h 20m' '18h 40m' '19h 25m' '29h 20m' '9h 5m'
 '10h 45m' '11h 40m' '22h 55m' '37h 25m' '25h 40m' '13h 55m' '8h 40m'
 '23h 30m' '12h 35m' '24h 15m' '1h 20m' '11h' '11h 15m' '14h 35m'
 '12h 55m' '9h' '7h 40m' '11h 45m' '24h 55m' '17h 5m' '29h 55m' '22h
15m'
 '14h 40m' '7h 15m' '20h 10m' '20h 45m' '27h' '24h 30m' '20h 25m' '5h
35m'
 '14h 45m' '5h 40m' '4h 5m' '15h 55m' '7h 45m' '28h 20m' '4h 20m' '3h
40m'
 '8h 50m' '23h 45m' '24h 45m' '21h 35m' '8h 5m' '6h 25m' '15h 50m'
 '26h 25m' '24h 50m' '26h' '23h 5m' '7h 55m' '26h 20m' '23h 15m' '5h
20m'
 '4h' '9h 45m' '8h 20m' '17h 25m' '7h 5m' '34h 5m' '6h 5m' '5h 50m'
'7h'
 '4h 25m' '13h 45m' '19h 15m' '22h 30m' '16h 25m' '13h 50m' '27h 5m'
 '28h 10m' '4h 40m' '15h 40m' '4h 35m' '18h 30m' '38h 15m' '6h 35m'
 '12h 30m' '11h 20m' '7h 35m' '29h 35m' '26h 55m' '23h 40m' '12h 50m'
 '9h 50m' '21h 55m' '10h 55m' '21h 10m' '20h 40m' '30h' '13h 10m' '8h
45m'
 '6h 10m' '17h 45m' '21h 45m' '3h 55m' '17h 20m' '30h 30m' '21h 25m'
 '12h 40m' '24h 35m' '19h 10m' '22h 40m' '14h 55m' '21h' '6h 45m'
 '28h 40m' '9h 40m' '16h 40m' '16h 20m' '16h 45m' '1h 15m' '6h 55m'
 '11h 25m' '14h 20m' '12h 5m' '24h 5m' '28h 15m' '17h 50m' '20h 20m'
 '28h 5m' '10h 20m' '14h 15m' '35h 15m' '35h 35m' '26h 40m' '28h'
 '14h 25m' '13h 5m' '37h 20m' '36h 10m' '25h 55m' '35h 5m' '19h 45m'
 '27h 55m' '47h' '10h 35m' '1h 35m' '16h 10m' '38h 20m' '6h' '16h 50m'
 '14h 10m' '23h 20m' '17h 40m' '11h 35m' '18h 20m' '6h 40m' '30h 55m'
 '24h 40m' '29h 50m' '28h 25m' '17h 15m' '22h 45m' '25h 25m' '21h 50m'
 '33h 15m' '30h 15m' '3h 35m' '27h 40m' '30h 25m' '18h 50m' '27h 45m'
 '15h 15m' '10h 40m' '26h 15m' '36h 25m' '26h 50m' '15h 45m' '19h 40m'
 '22h 25m' '19h 35m' '25h' '26h 45m' '38h' '4h 15m' '25h 10m' '18h
15m'
 '6h 50m' '23h 55m' '17h 55m' '23h 25m' '17h 10m' '24h 20m' '28h 30m'
 '27h 10m' '19h 20m' '15h 35m' '9h 25m' '21h 30m' '34h 25m' '18h 35m'
 '29h 40m' '26h 5m' '29h 5m' '27h 25m' '16h 30m' '11h 10m' '28h 55m'
 '29h 10m' '34h' '30h 40m' '30h 45m' '32h 55m' '10h 5m' '35h 20m' '32h
5m'
 '31h 40m' '19h 50m' '33h 45m' '30h 10m' '13h 40m' '19h 30m' '31h 30m'
```

```
 '34h 30m' '27h 50m' '38h 35m' '42h 5m' '4h 10m' '39h 5m' '3h 50m'
'5m'
 '32h 30m' '31h 55m' '33h 20m' '27h 30m' '18h 55m' '9h 55m' '41h 20m'
 '20h 5m' '31h 50m' '42h 45m' '3h 25m' '37h 10m' '29h 30m' '32h 20m'
 '20h 50m' '40h 20m' '13h 35m' '47h 40m']
*********************************************************
The Unique Values in feature Total_Stops is
['non-stop' '2 stops' '1 stop' '3 stops' nan '4 stops']
*********************************************************
The Unique Values in feature Additional_Info is
['Null ' 'Null' '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']
*********************************************************
The Unique Values in feature Price is
[ 3897  7662 13882 ...  9790 12352 12648]
*********************************************************
```

Central Function to Prepare the Process data & Model data

```
def preprocess(data):
    """
    Function to Process data and get the process data & Modeling data
    """
    df.dropna(inplace = True)
    df.drop_duplicates(inplace = True)

    df['Date_of_Journey'] = pd.to_datetime(df['Date_of_Journey'])
    df['day'] = pd.DatetimeIndex(df['Date_of_Journey']).day
    df['month'] = pd.DatetimeIndex(df['Date_of_Journey']).month
    df['weekday'] = pd.DatetimeIndex(df['Date_of_Journey']).weekday

    df['Total_Stops'] = df['Total_Stops'].replace('non-stop', '0')
    df['Total_Stops'] = df['Total_Stops'].replace('1 stop', '1')
    df['Total_Stops'] = df['Total_Stops'].replace('2 stops', '2')
    df['Total_Stops'] = df['Total_Stops'].replace('3 stops', '3')
    df['Total_Stops'] = df['Total_Stops'].replace('4 stops', '4')

    df['Destination'] = np.where(df['Destination'] == 'New Delhi',
'Delhi', df['Destination'])
    df['Airline'] = np.where(df['Airline'] == 'Jet Airways
Business','Jet Airways',df['Airline'])
    df['Airline'] = np.where(df['Airline'] == 'Vistara Premium
economy','Vistara',df['Airline'])
    df['Airline'] = np.where(df['Airline'] == 'Multiple carriers
Premium economy','Multiple carriers',df['Airline'])


    arrival_time = []
    for i in data["Arrival_Time"]:
```

```
        arrival_time.append(i[:5])
    df['Arrival_Time'] = arrival_time
    df['Arrival_Time_hour'] =
pd.DatetimeIndex(df['Arrival_Time']).hour
    df['Arrival_Time_minutes'] =
pd.DatetimeIndex(df['Arrival_Time']).minute

    df['Duration_Total_Hour'] =
df['Duration'].str.replace('h','*1').str.replace('
','+').str.replace('m','/60').apply(eval)

    data1 = pd.get_dummies(data,
prefix=['Airline','Source','Destination'],columns =
['Airline','Source','Destination'], drop_first = True)

data1.drop(['Date_of_Journey','Dep_Time','Arrival_Time','Additional_In
fo','Route'], axis =1, inplace = True)
    return data, data1

### Get The EDA & Model Data
data_eda, data_model = preprocess(df)

data_eda
```

```
            Airline Date_of_Journey    Source Destination  \
0            IndiGo      2019-03-24  Banglore       Delhi
1         Air India      2019-01-05   Kolkata    Banglore
2       Jet Airways      2019-09-06     Delhi      Cochin
3            IndiGo      2019-12-05   Kolkata    Banglore
4            IndiGo      2019-01-03  Banglore       Delhi
...             ...             ...       ...         ...
10678      Air Asia      2019-09-04   Kolkata    Banglore
10679     Air India      2019-04-27   Kolkata    Banglore
10680   Jet Airways      2019-04-27  Banglore       Delhi
10681       Vistara      2019-01-03  Banglore       Delhi
10682     Air India      2019-09-05     Delhi      Cochin

                       Route Dep_Time Arrival_Time Duration
Total_Stops  \
0                  BLR → DEL    22:20        01:10   2h 50m
0
1      CCU → IXR → BBI → BLR    05:50        13:15   7h 25m
2
2      DEL → LKO → BOM → COK    09:25        04:25      19h
2
3            CCU → NAG → BLR    18:05        23:30   5h 25m
1
4            BLR → NAG → DEL    16:50        21:35   4h 45m
1
...                      ...      ...          ...      ...          ..
```

```
.
10678              CCU → BLR    19:55        22:25   2h 30m
0
10679              CCU → BLR    20:45        23:20   2h 35m
0
10680              BLR → DEL    08:20        11:20       3h
0
10681              BLR → DEL    11:30        14:10   2h 40m
0
10682   DEL → GOI → BOM → COK    10:55        19:15   8h 20m
2

      Additional_Info  Price  day  month  weekday
Arrival_Time_hour  \
0                Null   3897   24      3        6                    1

1                 Null   7662    5      1        5                   13

2                 Null  13882    6      9        4                    4

3                 Null   6218    5     12        3                   23

4                 Null  13302    3      1        3                   21

...                ...    ...  ...    ...      ...                 ...

10678             Null   4107    4      9        2                   22

10679             Null   4145   27      4        5                   23

10680             Null   7229   27      4        5                   11

10681             Null  12648    3      1        3                   14

10682             Null  11753    5      9        3                   19


       Arrival_Time_minutes  Duration_Total_Hour
0                        10             2.833333
1                        15             7.416667
2                        25            19.000000
3                        30             5.416667
4                        35             4.750000
...                     ...                  ...
10678                    25             2.500000
10679                    20             2.583333
10680                    20             3.000000
10681                    10             2.666667
10682                    15             8.333333

[10462 rows x 17 columns]
```

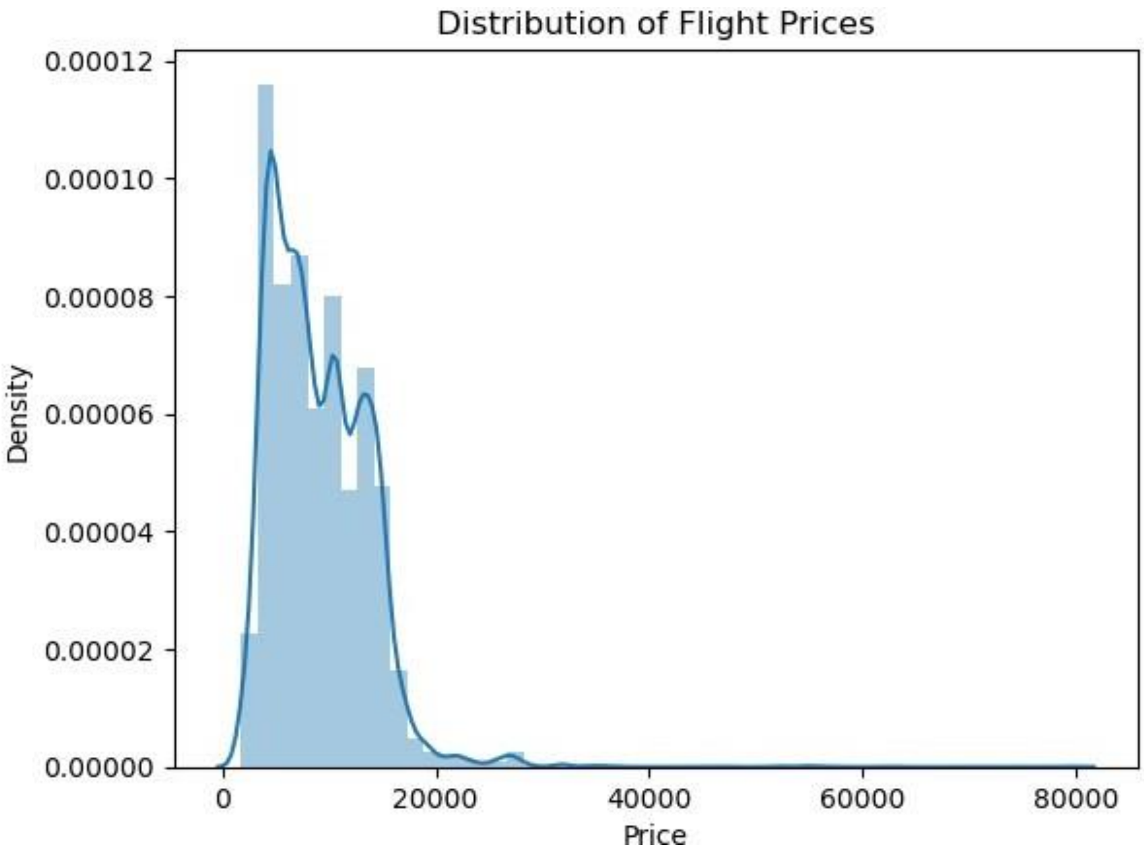
```
data_model

      Duration Total_Stops  Price  day  month  weekday  
Arrival_Time_hour  \
0        2h 50m           0   3897   24      3        6   
1   
1        7h 25m           2   7662    5      1        5   
13   
2           19h           2  13882    6      9        4   
4   
3        5h 25m           1   6218    5     12        3   
23   
4        4h 45m           1  13302    3      1        3   
21   
...         ...         ...    ...  ...    ...      ...                
...   
10678    2h 30m           0   4107    4      9        2   
22   
10679    2h 35m           0   4145   27      4        5   
23   
10680        3h           0   7229   27      4        5   
11   
10681    2h 40m           0  12648    3      1        3   
14   
10682    8h 20m           2  11753    5      9        3   
19   

       Arrival_Time_minutes                                          
India  ...  \
0                        10                                          
0  ...   
1                        15         7.416667                       
1  ...   
2                        25        19.000000                       
0  ...   
3                        30         5.416667                       
0  ...   
4                        35         4.750000                       
0  ...   
...                     ...              ...                  ...  .
..   
10678                    25         2.500000                       
0  ...   
10679                    20         2.583333                       
1  ...   
10680                    20         3.000000                       
0  ...   
10681                    10         2.666667                       
0  ...   
10682                    15         8.333333                       
```

```
1  ...

       Airline_Trujet  Airline_Vistara  Source_Chennai
Source_Delhi  \
0                   0                0               0               0

1                   0                0               0               0

2                   0                0               0               1

3                   0                0               0               0

4                   0                0               0               0

...               ...              ...             ...             ...

10678               0                0               0               0

10679               0                0               0               0

10680               0                0               0               0

10681               0                1               0               0

10682               0                0               0               1


       Source_Kolkata  Source_Mumbai  Destination_Cochin
Destination_Delhi  \
0                   0              0                   0
1
1                   1              0                   0
0
2                   0              0                   1
0
3                   1              0                   0
0
4                   0              0                   0
1
...               ...            ...                 ...
...
10678               1              0                   0
0
10679               1              0                   0
0
10680               0              0                   0
1
10681               0              0                   0
1
10682               0              0                   1
```

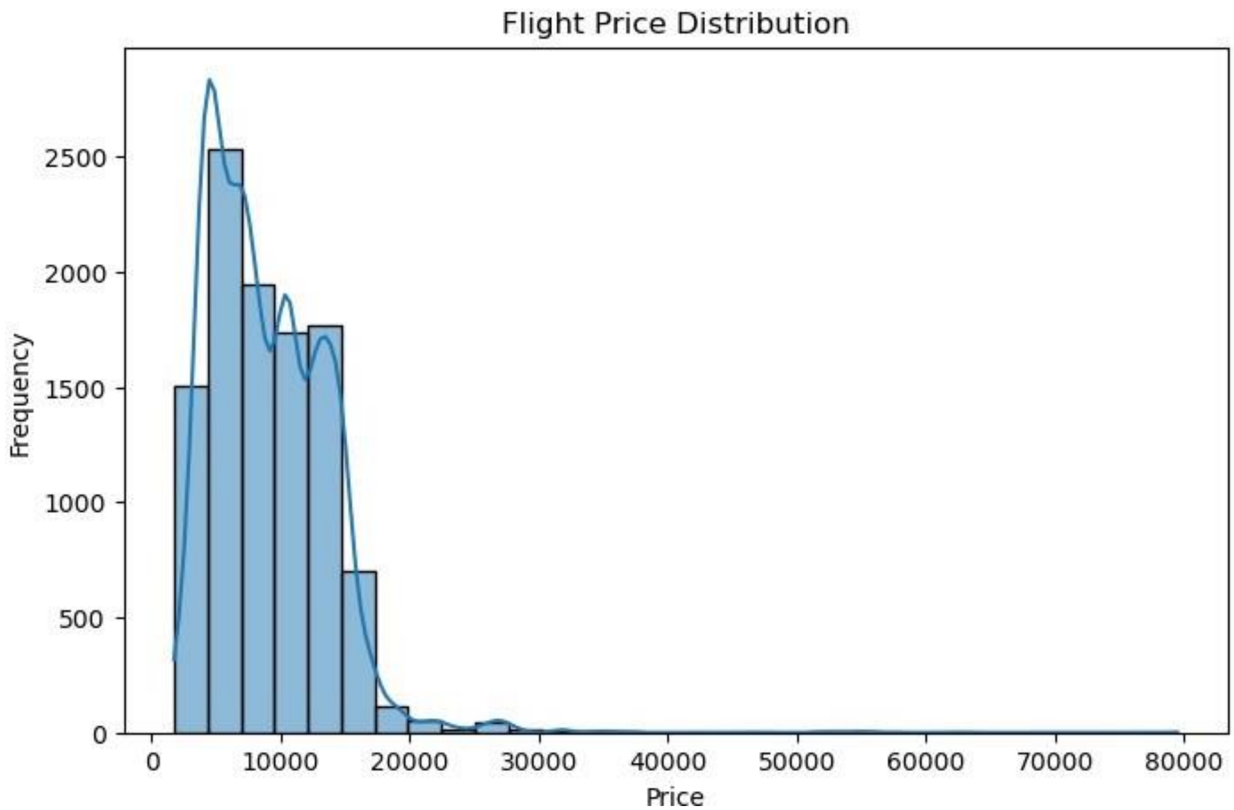
```
0

       Destination_Hyderabad  Destination_Kolkata
0                          0                    0
1                          0                    0
2                          0                    0
3                          0                    0
4                          0                    0
...                      ...                  ...
10678                      0                    0
10679                      0                    0
10680                      0                    0
10681                      0                    0
10682                      0                    0

[10462 rows x 25 columns]
```

Univariate Exploratory Data Analysis

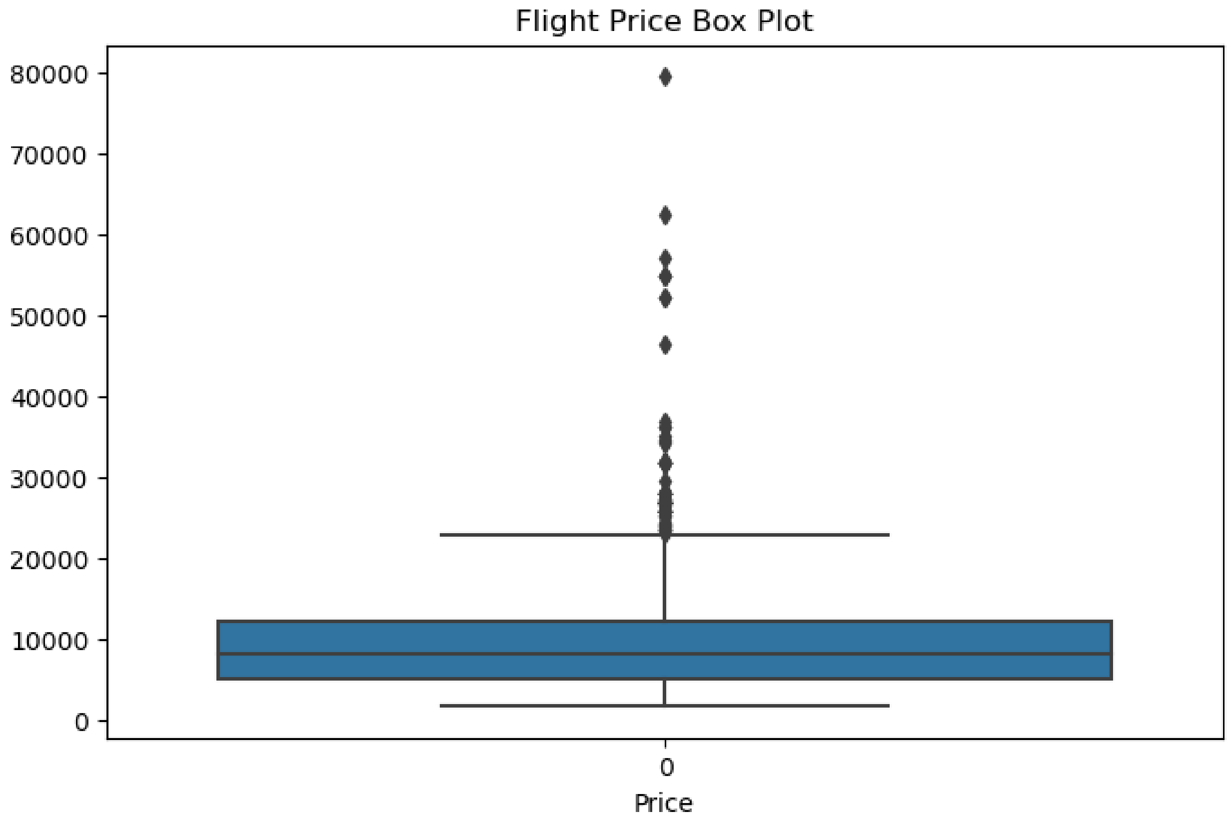
```
# Histogram for 'price'
sns.distplot(data_eda['Price'])
plt.title('Distribution of Flight Prices')
plt.xlabel('Price')
plt.ylabel('Density')
plt.show()
```



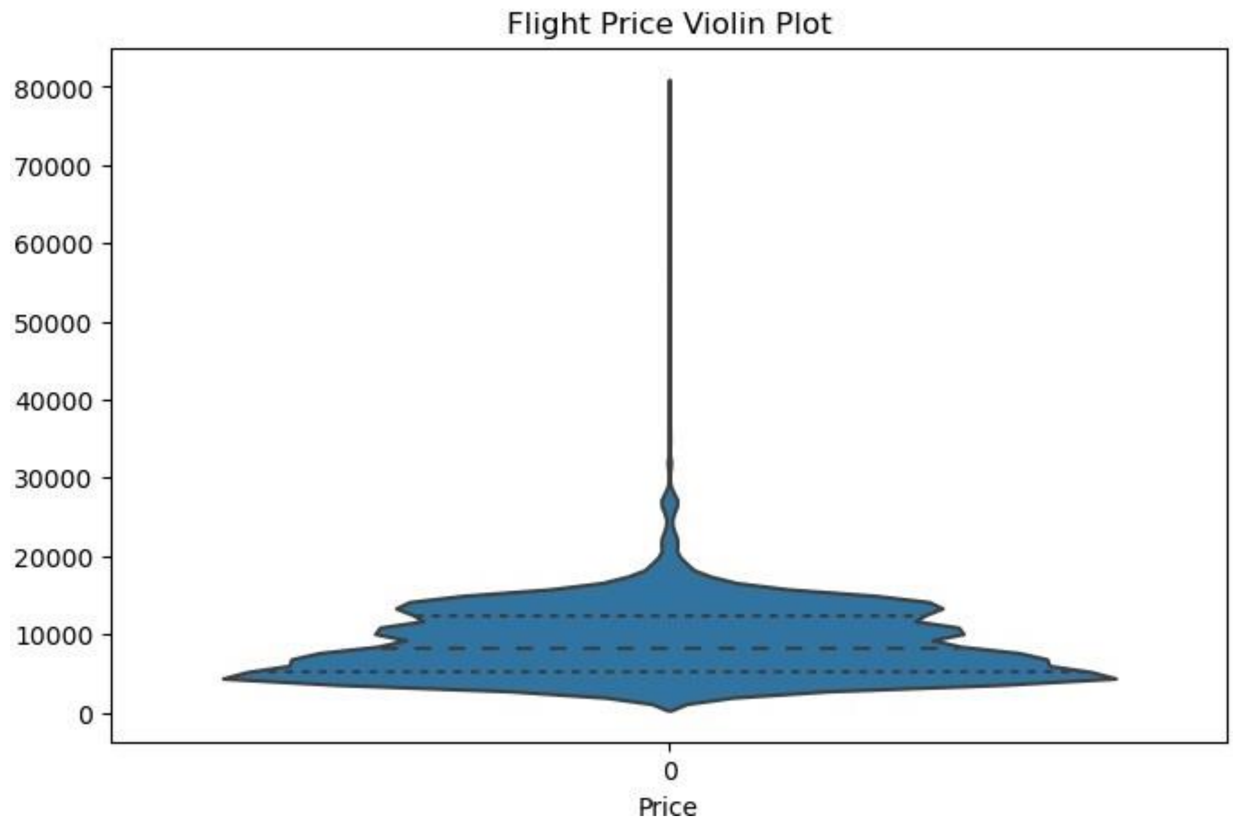

```python
# Create a histogram to visualize the distribution of flight prices
plt.figure(figsize=(8, 5))
sns.histplot(data_eda['Price'], kde=True, bins=30)
plt.title("Flight Price Distribution")
plt.xlabel("Price")
plt.ylabel("Frequency")
plt.show()
```



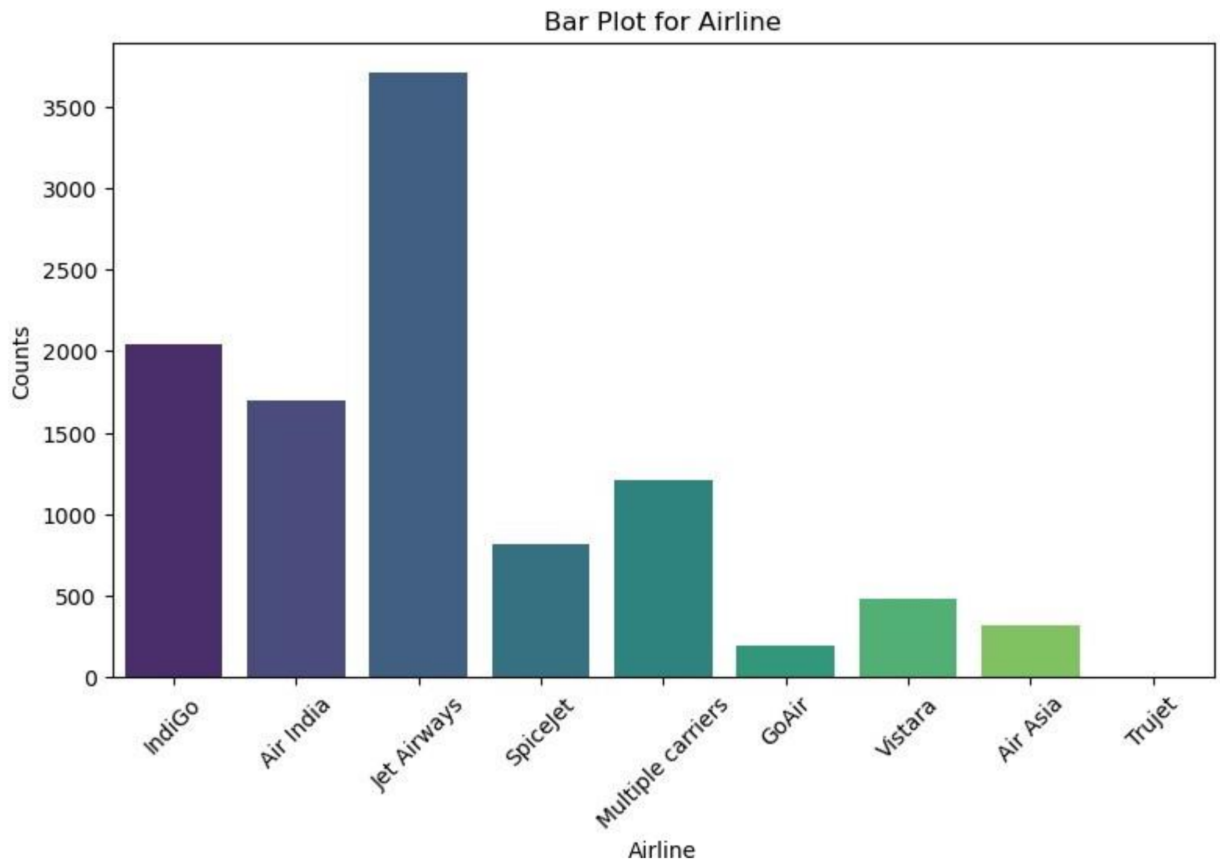

```python
# Create a box plot to identify outliers
plt.figure(figsize=(8, 5))
sns.boxplot(data_eda['Price'])
plt.title("Flight Price Box Plot")
plt.xlabel("Price")
plt.show()
```

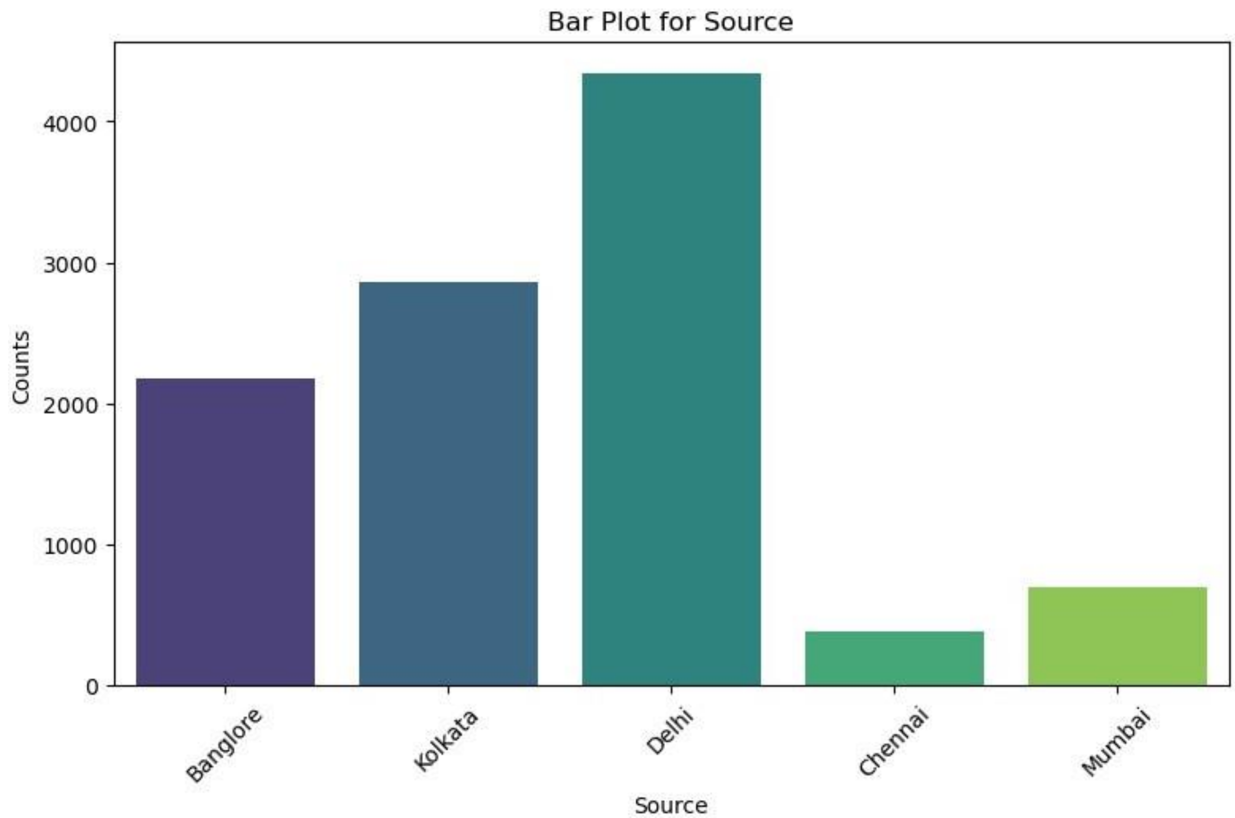


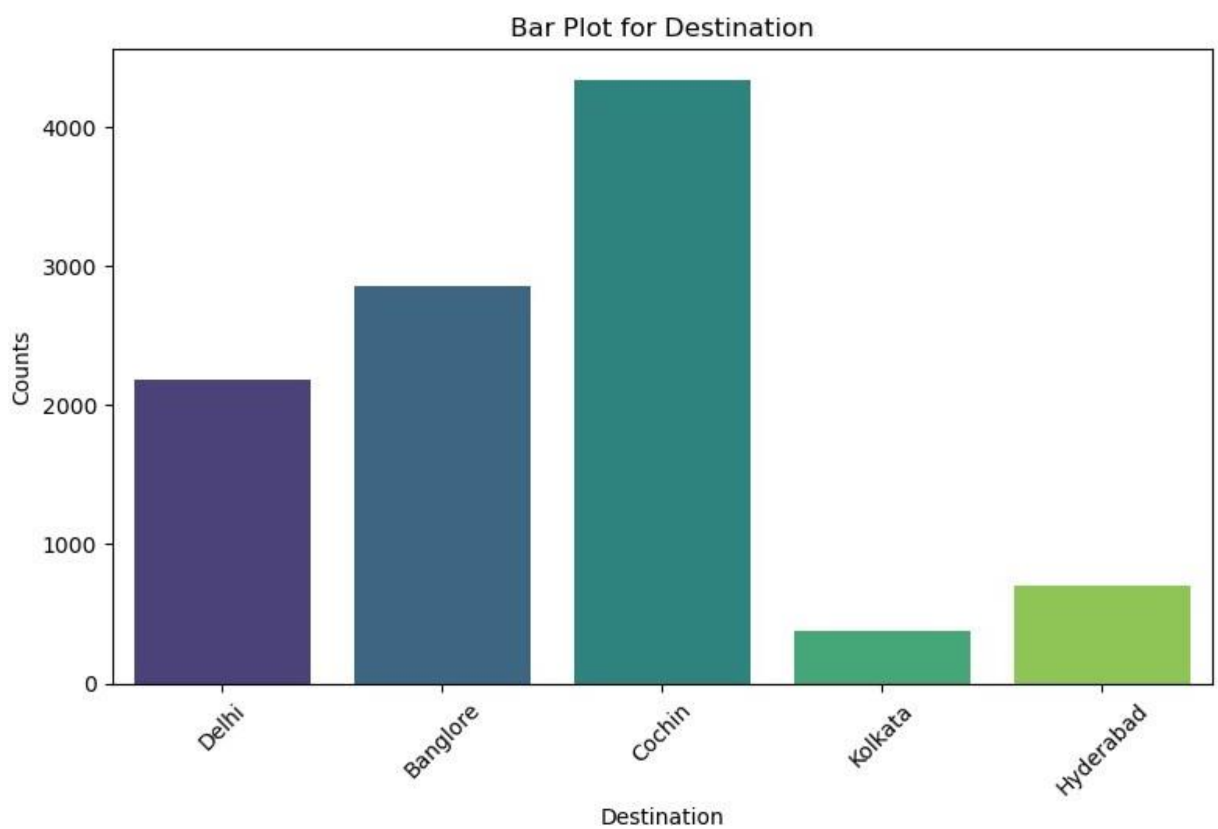

```
# Create a violin plot for a more detailed distribution view
plt.figure(figsize=(8,  5))
sns.violinplot(data_eda["Price"], inner="quartile")
plt.title("Flight Price Violin Plot")
plt.xlabel("Price")
plt.show()
```

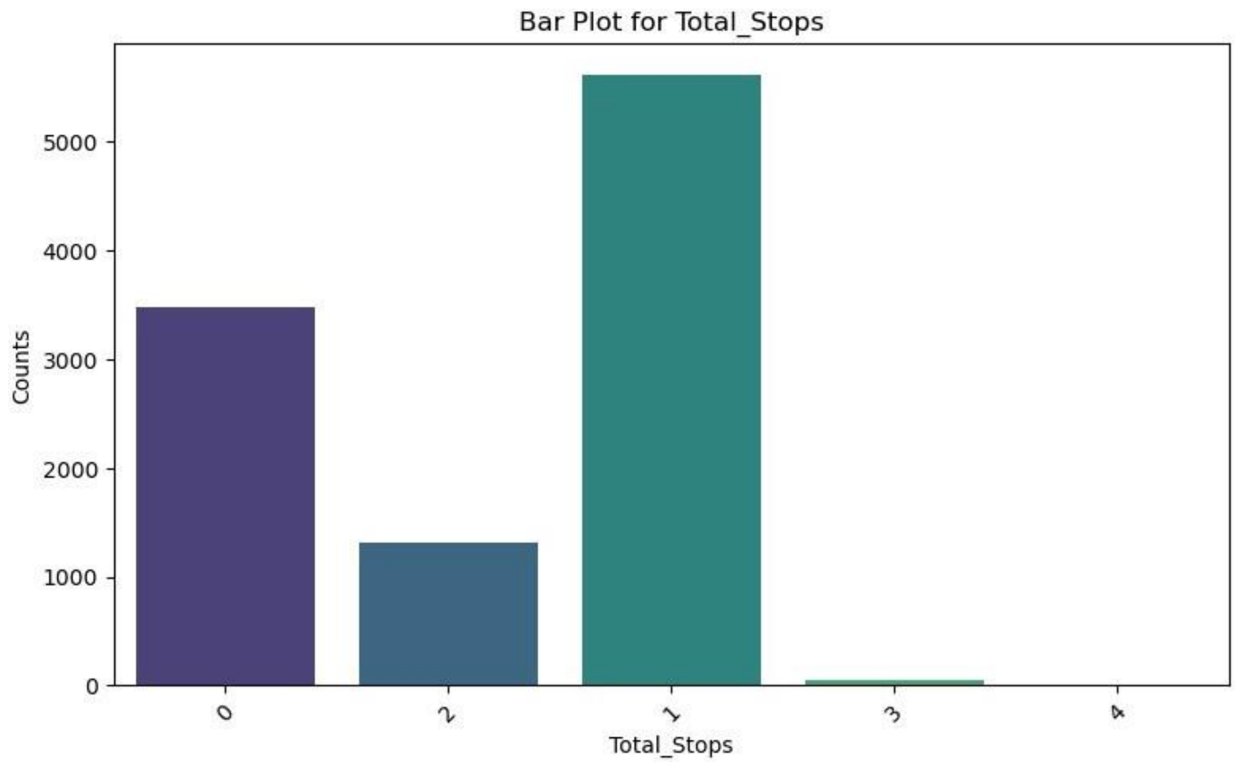


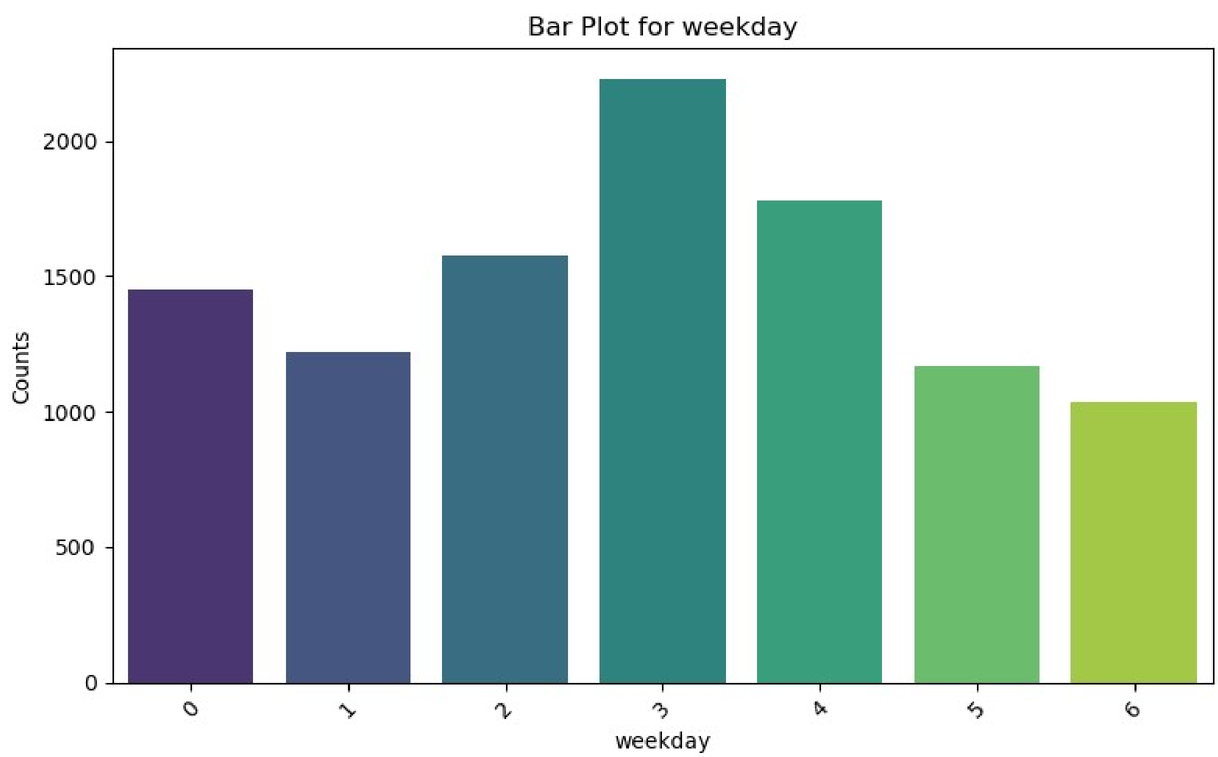

```
#Airline    Date_of_Journey Source      Destination      Route Dep_Time
      Arrival_Time     Duration    Total_Stops      Additional_Info Price
      day   month weekday     Arrival_Time_hour      Arrival_Time_minutes
      Duration_Total_Hour
seg_data =
['Airline','Source','Destination','Total_Stops','weekday','month']
for edcol in seg_data:
    plt.figure(figsize=(8, 5))
    # Creating a count plot using seaborn
    sns.countplot(x=data_eda[edcol], palette="viridis")

    plt.xlabel(edcol)
    plt.ylabel('Counts')
    plt.title(f'Bar Plot for {edcol}')

    # Adjust layout for better visualization
    plt.tight_layout()
    plt.xticks(rotation=45)
    # Display the plot
    plt.show()
    #print(data_eda[edcol].value_counts())
    #print(data_eda[edcol].index)
    #print(data_eda[edcol].values)
```

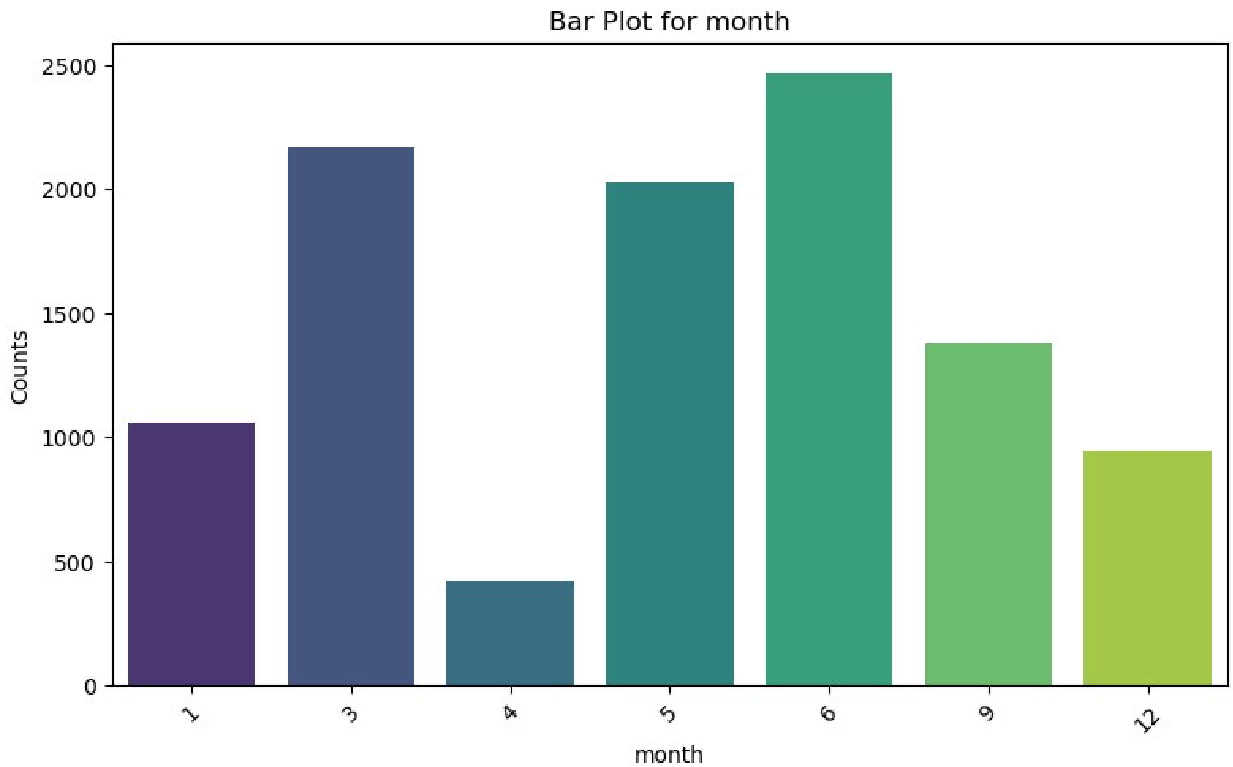

Bar Plot for Airline
Counts
0
500
1000
1500
2000
2500
3000
3500
IndiGo
Air India
Jet Airways
SpiceJet
Multiple carriers
GoAir
Vistara
Air Asia
Trujet
Airline


Bar Plot for Source
Counts
4000
3000
2000
1000
0
Banglore
Kolkata
Delhi
Chennai
Mumbai
Source

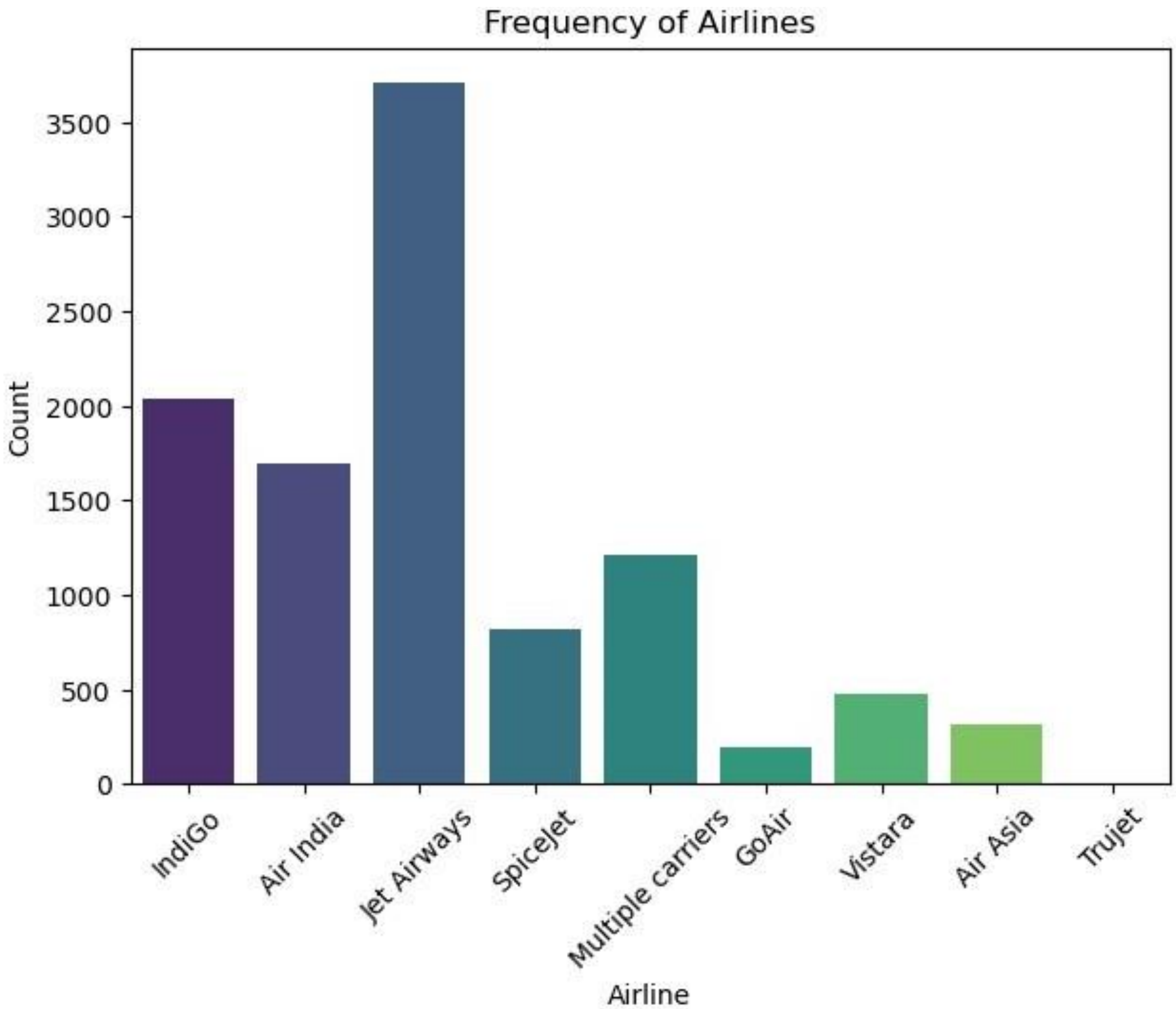

Bar Plot for Destination
Counts
4000
3000
2000
1000
0
Delhi
Banglore
Cochin
Kolkata
Hyderabad
Destination


Bar Plot for Total_Stops
Counts
0
1000
2000
3000
4000
5000
0
2
1
3
4
Total_Stops

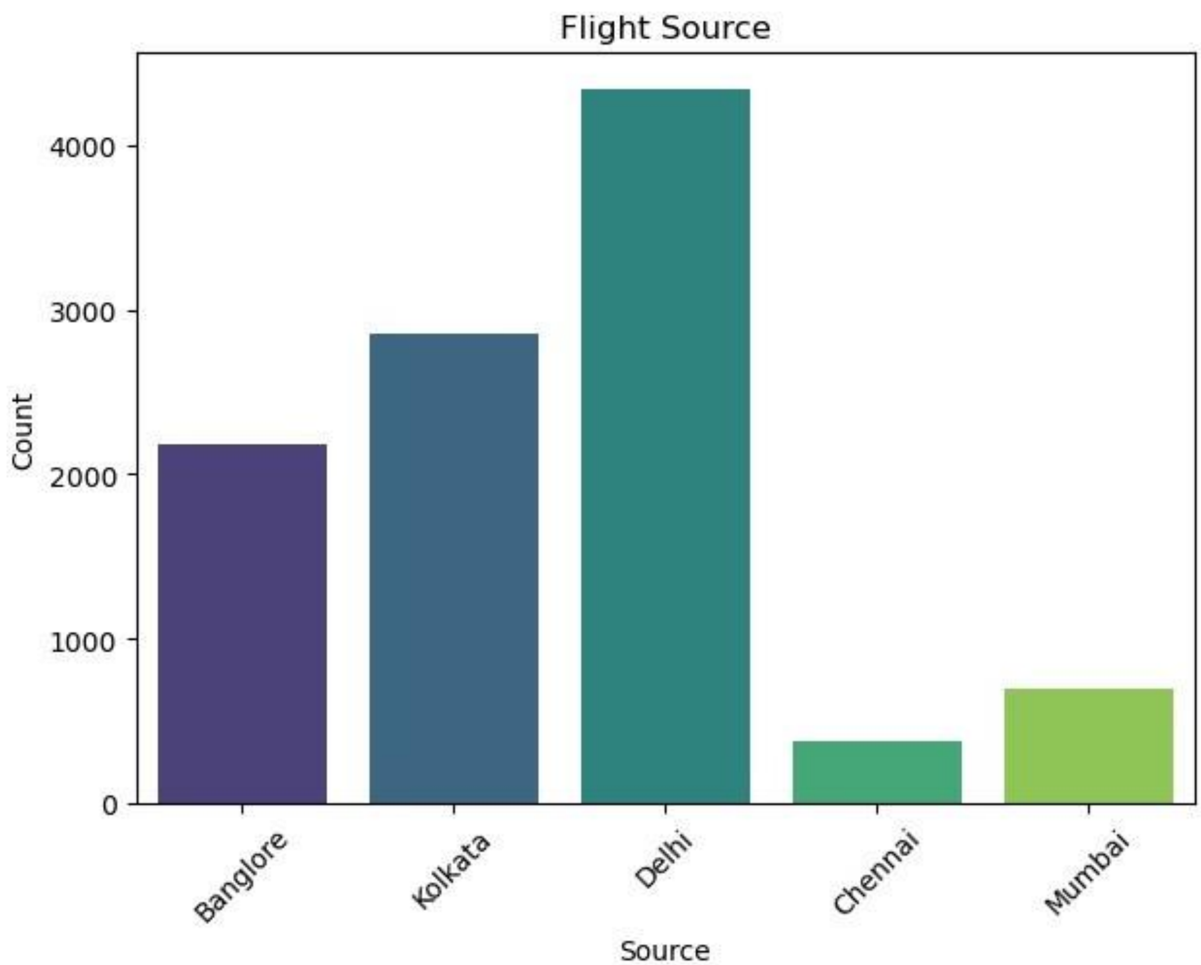

Bar Plot for weekday
Counts
0
500
1000
1500
2000
0
1
2
3
4
5
6
weekday



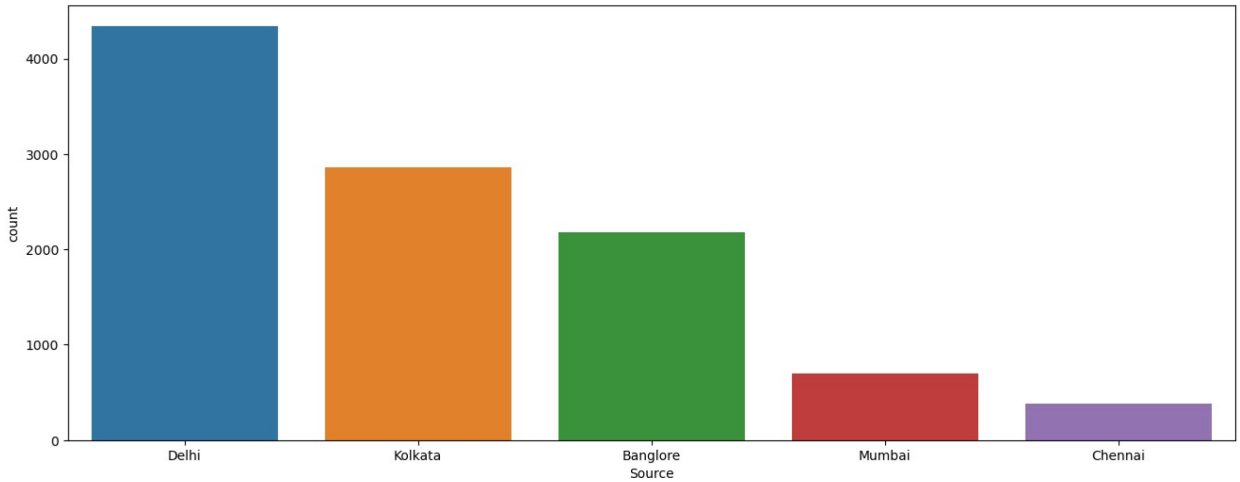

```
# Generate a count plot to visualize the frequency of unique flight
prices

# Univariate EDA for categorical variables

# Count plot for 'airline'
sns.countplot(x=data_eda['Airline'], palette="viridis")
plt.title('Frequency of Airlines')
plt.xlabel('Airline')
plt.ylabel('Count')
plt.tight_layout()
plt.xticks(rotation=45)
plt.show()
```



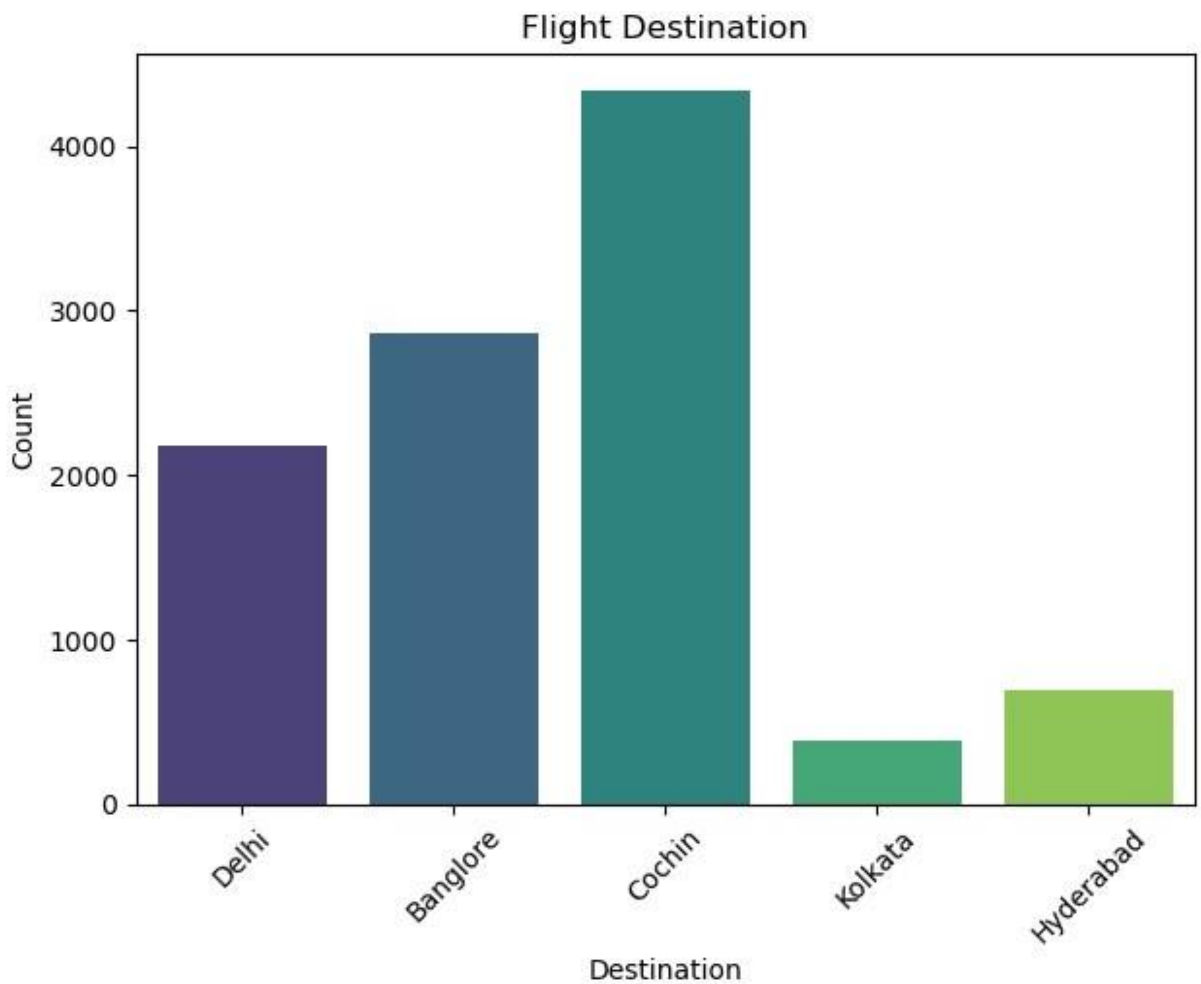

```
# Univariate EDA for categorical variables

# Count plot for 'Source'
sns.countplot(x=data_eda['Source'], palette="viridis")
plt.title('Flight Source')
plt.xlabel('Source')
plt.ylabel('Count')
plt.tight_layout()
plt.xticks(rotation=45)
plt.show()
```



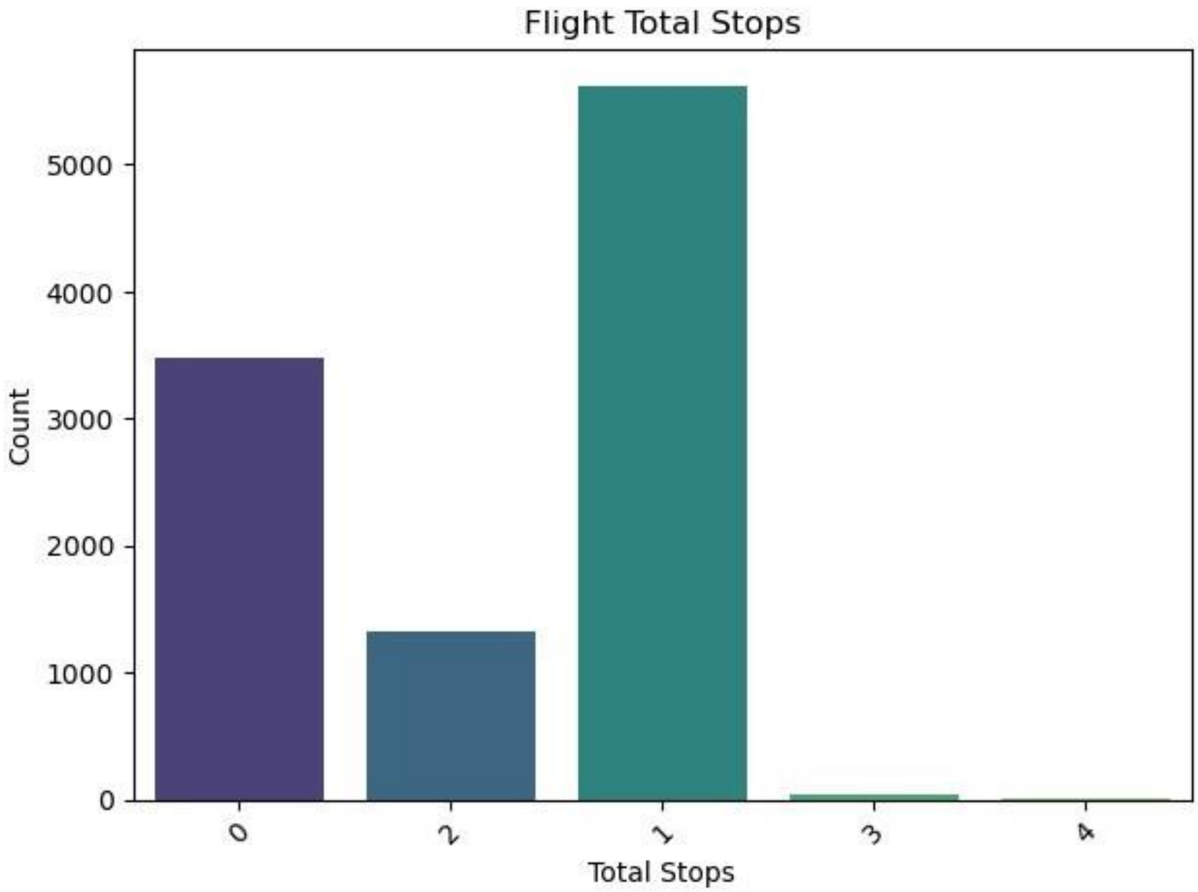

```
plt.figure(figsize=(16,6))
print(data_eda['Source'].value_counts)
sns.countplot(x="Source",data=data_eda,order=data_eda['Source'].value_
counts().index)

<bound method IndexOpsMixin.value_counts of 0        Banglore
1         Kolkata
2           Delhi
3         Kolkata
4        Banglore
           ...
10678     Kolkata
10679     Kolkata
10680    Banglore
10681    Banglore
10682       Delhi
Name: Source, Length: 10462, dtype: object>

<Axes: xlabel='Source', ylabel='count'>
```




```python
# Univariate EDA for categorical variables

# Count plot for 'Source'
sns.countplot(x=data_eda['Destination'], palette="viridis")
plt.title('Flight Destination')
plt.xlabel('Destination')
plt.ylabel('Count')
plt.tight_layout()
plt.xticks(rotation=45)
plt.show()
```

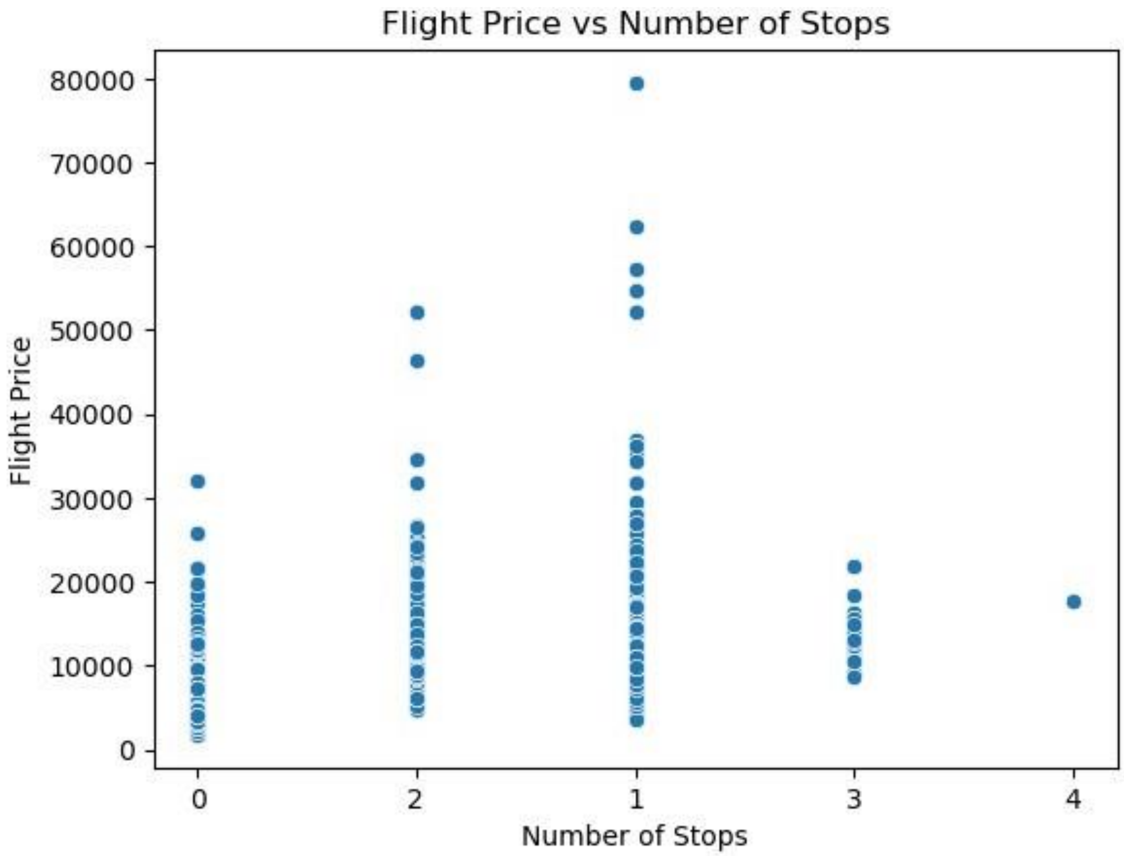



```
# Univariate EDA for categorical variables

# Count plot for 'Source'
sns.countplot(x=data_eda['Total_Stops'], palette="viridis")
plt.title('Flight Total Stops')
plt.xlabel('Total Stops')
plt.ylabel('Count')
plt.tight_layout()
plt.xticks(rotation=45)
plt.show()
```



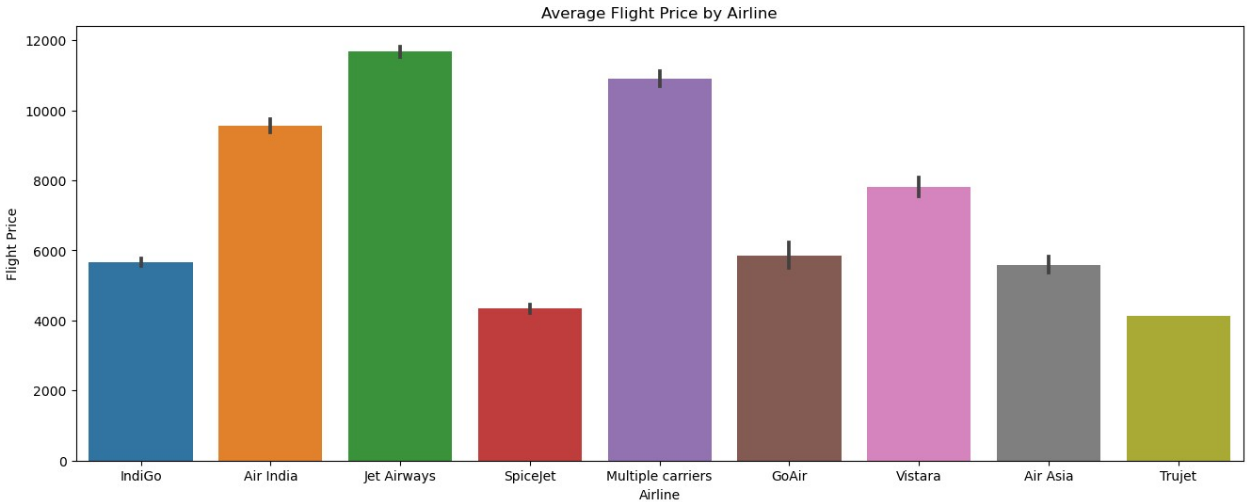
Bivariate Analysis

```
#relationship between the flight price and the number of stops.
sns.scatterplot(x='Total_Stops', y='Price', data=data_eda)
plt.xlabel('Number of Stops')
plt.ylabel('Flight Price')
plt.title('Flight Price vs Number of Stops')

# Display the plot
plt.show()
```

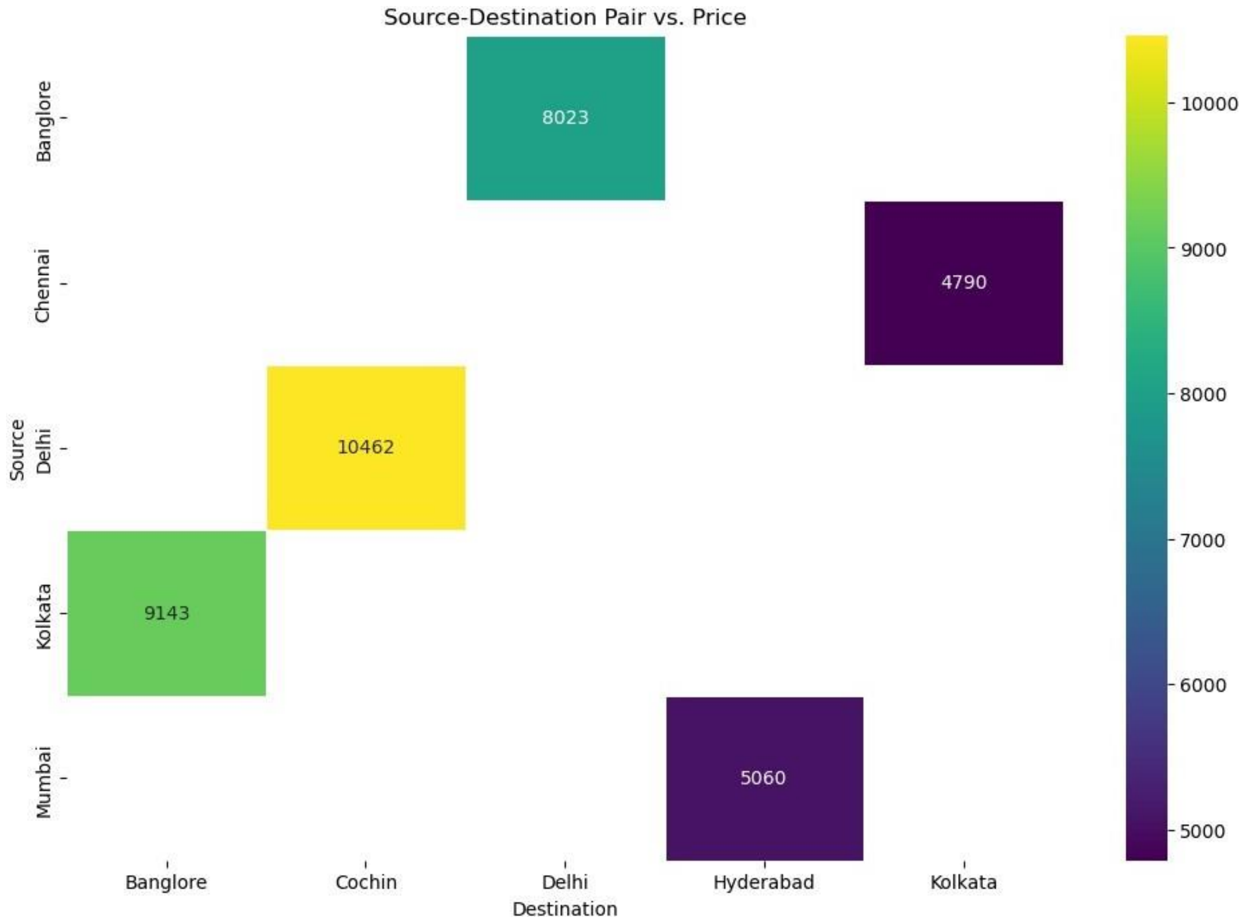



```
# Average Price Duistrubution as per airline
plt.figure(figsize=(16,6))
sns.barplot(x="Airline",y='Price',data=data_eda)
plt.xlabel('Airline')
plt.ylabel('Flight Price')
plt.title('Average Flight Price by Airline')
```

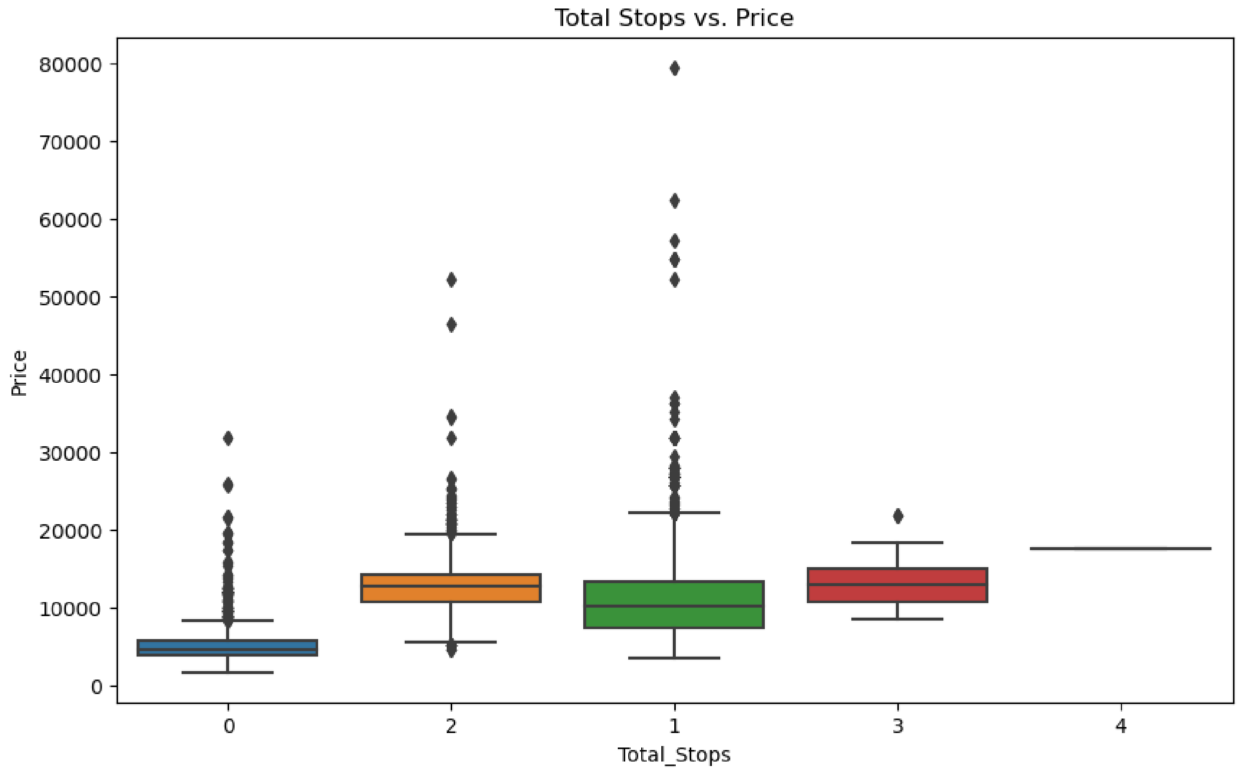
```
Text(0.5, 1.0, 'Average Flight Price by Airline')
```



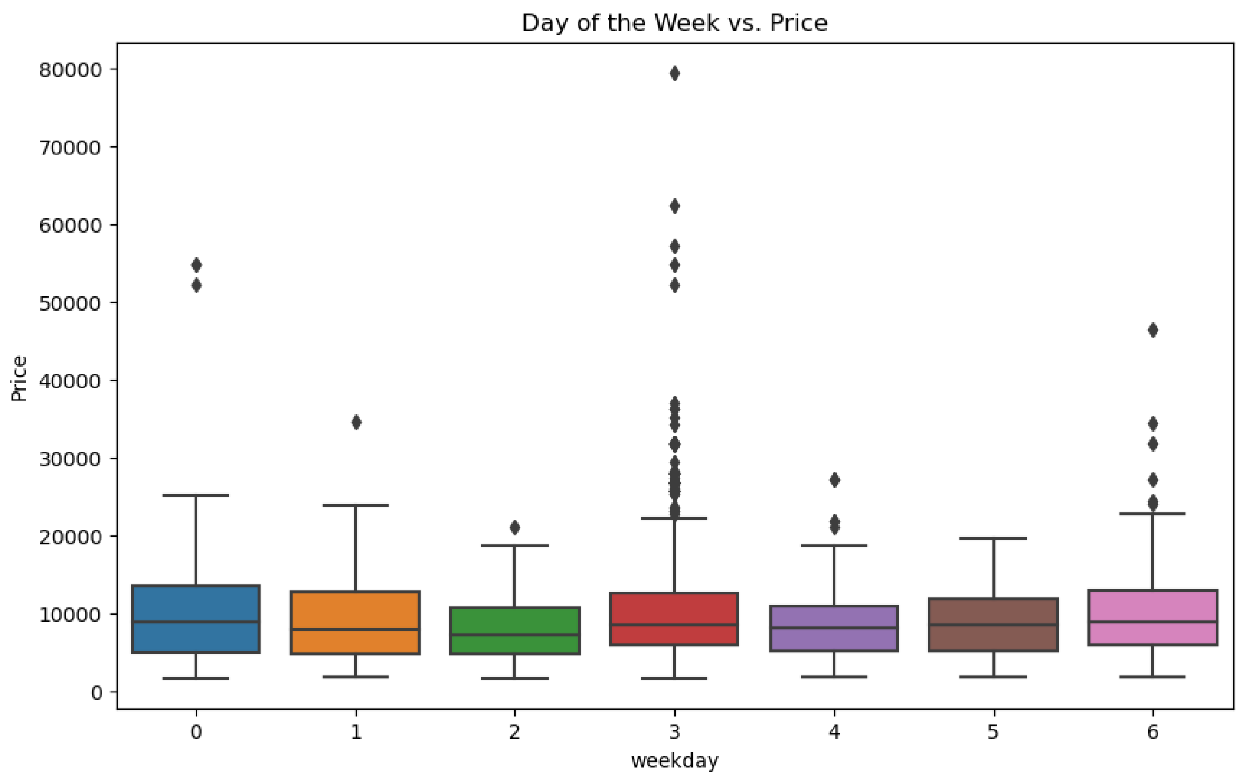

```
## Example 2: Source-Destination Pair vs. Price
plt.figure(figsize=(12, 8))
heatmap_data = data_eda.pivot_table(index='Source',
columns='Destination', values='Price', aggfunc='mean')
sns.heatmap(heatmap_data, cmap='viridis', annot=True, fmt=".0f",
linewidths=.5)
plt.title('Source-Destination Pair vs. Price')
plt.show()
```



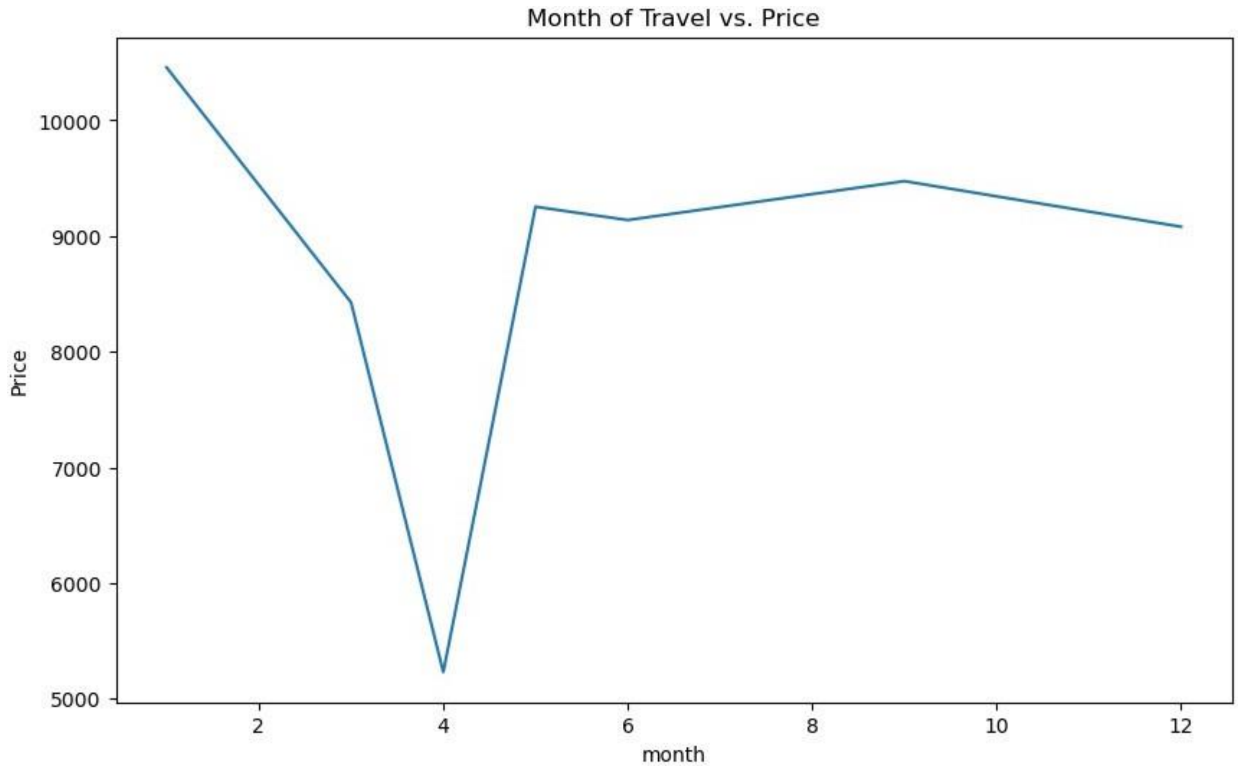

```
# Example 3: Total Stops vs. Price
plt.figure(figsize=(10, 6))
sns.boxplot(x='Total_Stops', y='Price', data=data_eda)
plt.title('Total Stops vs. Price')
plt.show()
```




```
# Example 4: Day of the Week vs. Price
plt.figure(figsize=(10, 6))
sns.boxplot(x='weekday', y='Price', data=data_eda)
plt.title('Day of the Week vs. Price')
plt.show()
```

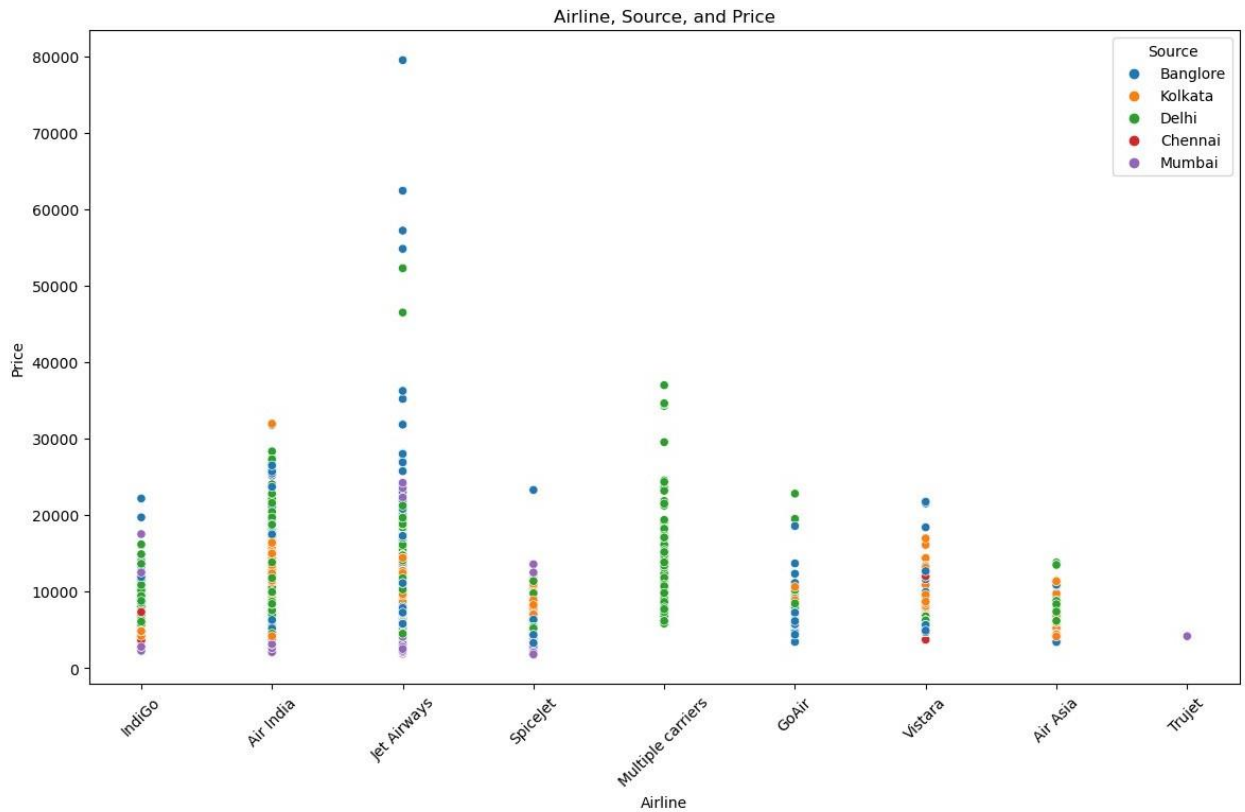



```
# Example 5: Month of Travel vs. Price
plt.figure(figsize=(10, 6))
sns.lineplot(x='month', y='Price', data=data_eda, estimator='mean',
ci=None)
plt.title('Month of Travel vs. Price')
plt.show()
```

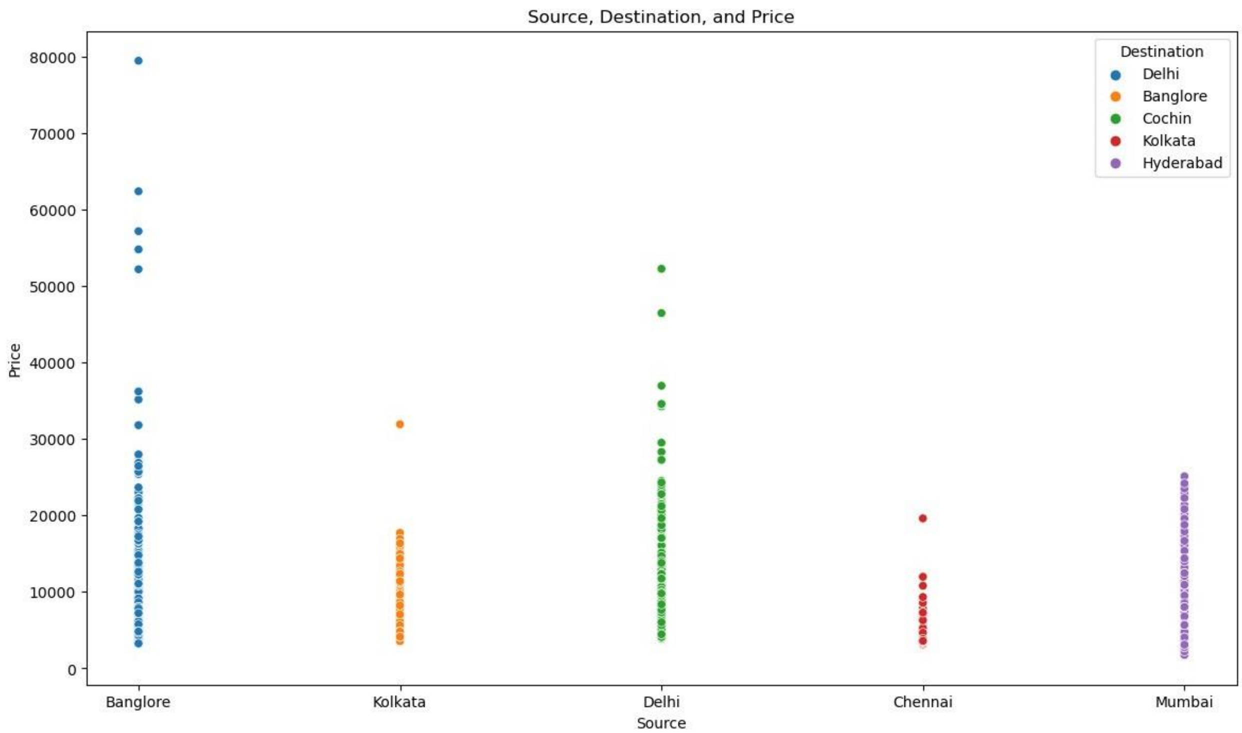


Multivariate Analysis

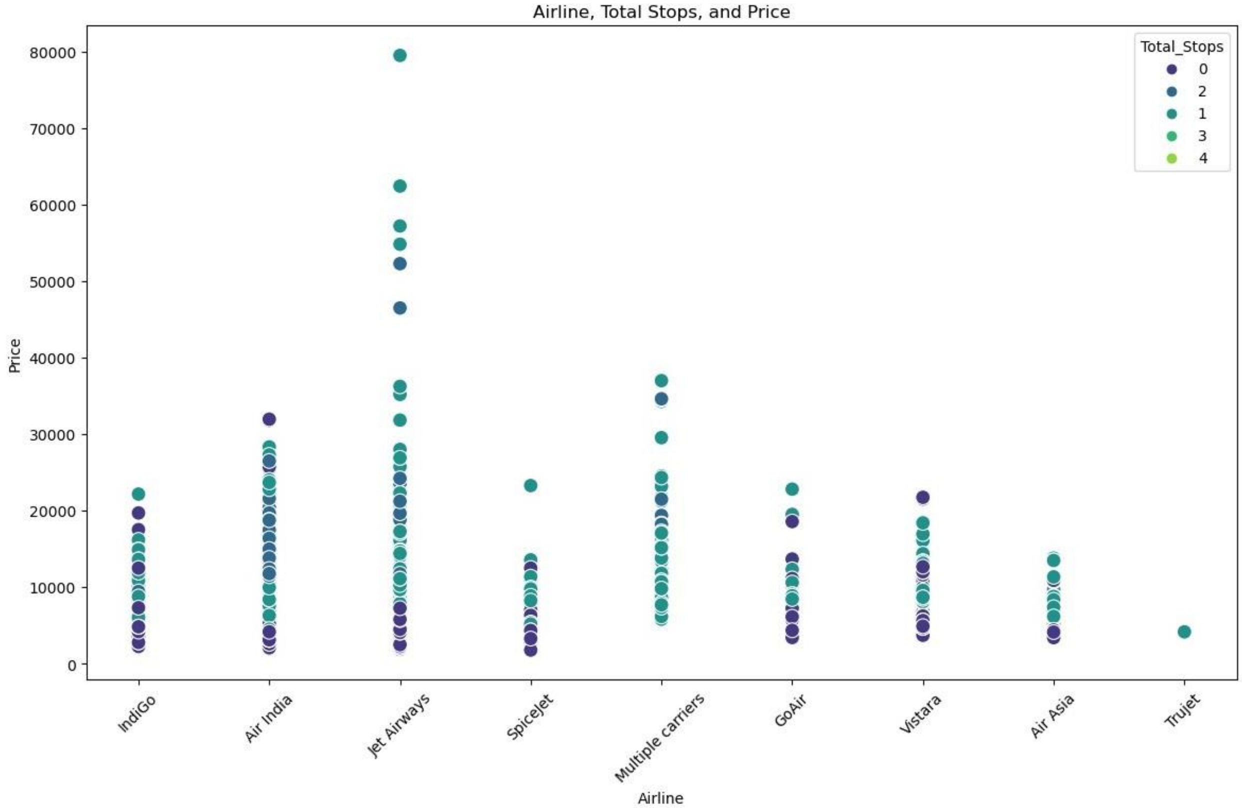
```
# Airline, Source, and Price
plt.figure(figsize=(14, 8))
sns.scatterplot(x='Airline', y='Price', hue='Source', data=data_eda)
plt.xticks(rotation=45)
plt.title('Airline, Source, and Price')
plt.show()
```



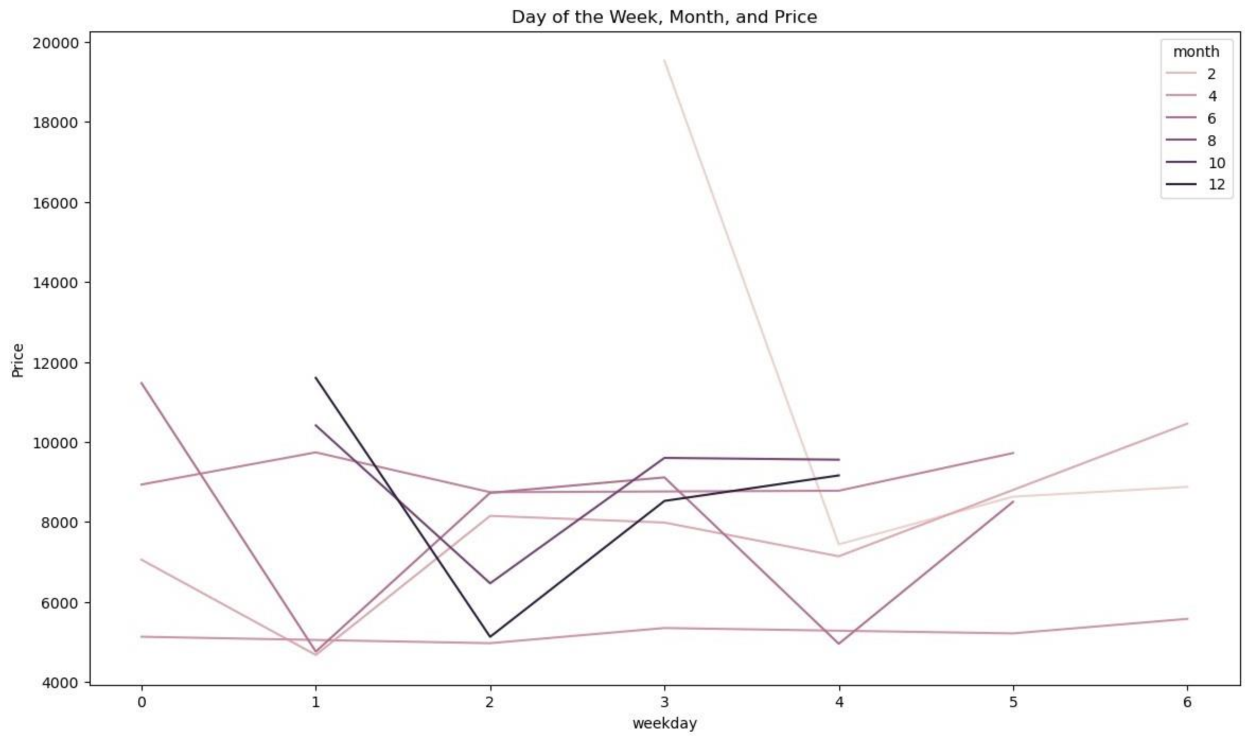

```
#Source, Destination, and Price
plt.figure(figsize=(14, 8))
sns.scatterplot(x='Source', y='Price', hue='Destination',
data=data_eda)
plt.title('Source, Destination, and Price')
plt.show()
```



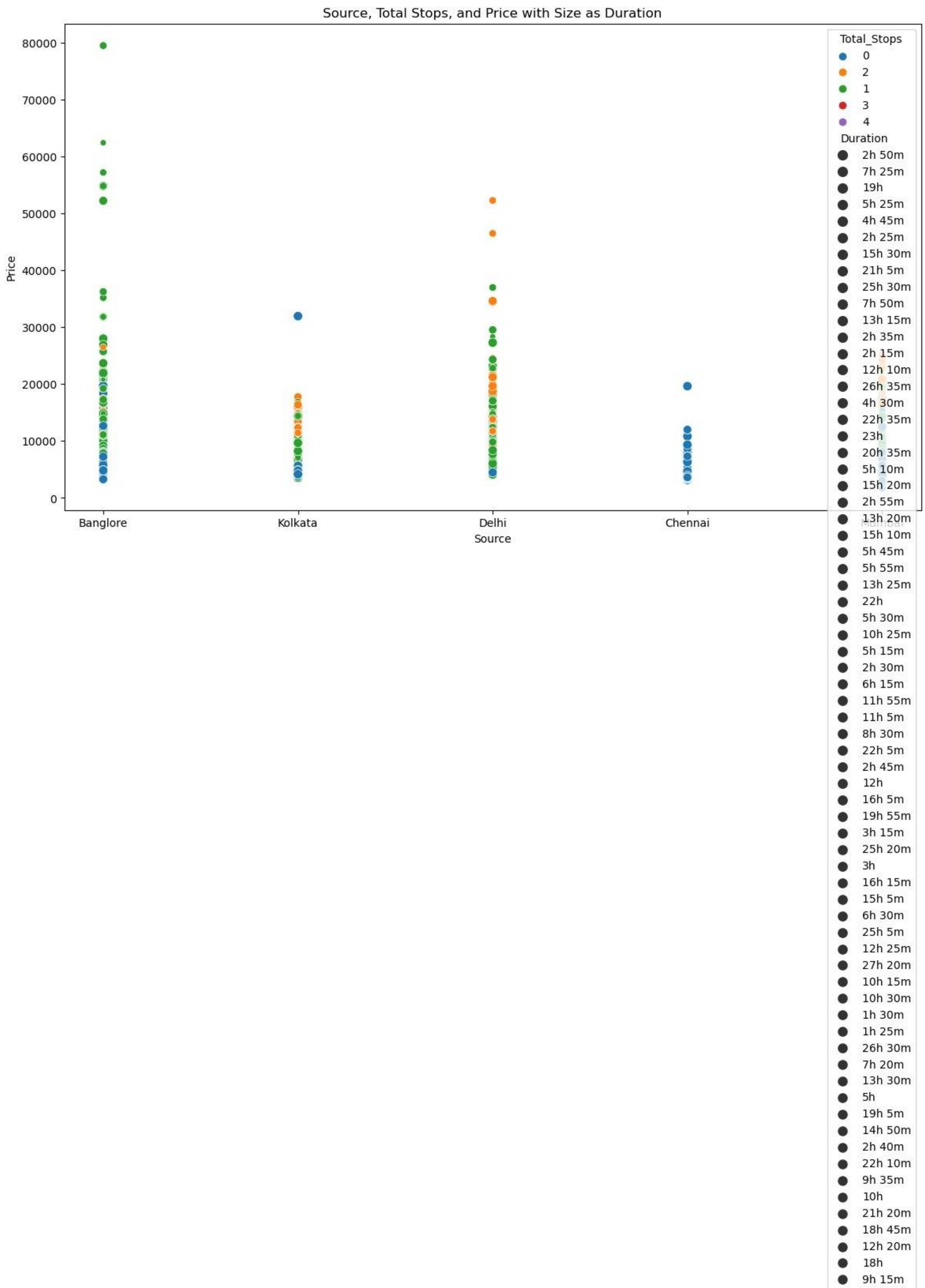

```
#  Airline, Total Stops, and Price
plt.figure(figsize=(14, 8))
sns.scatterplot(x='Airline', y='Price', hue='Total_Stops',
data=data_eda, palette='viridis', s=100)
plt.xticks(rotation=45)
plt.title('Airline, Total Stops, and Price')
plt.show()
```




```
#  Day of the Week, Month, and Price
plt.figure(figsize=(14, 8))
sns.lineplot(x='weekday', y='Price', hue='month', data=data_eda,
estimator='mean', ci=None)
plt.title('Day of the Week, Month, and Price')
plt.show()
```

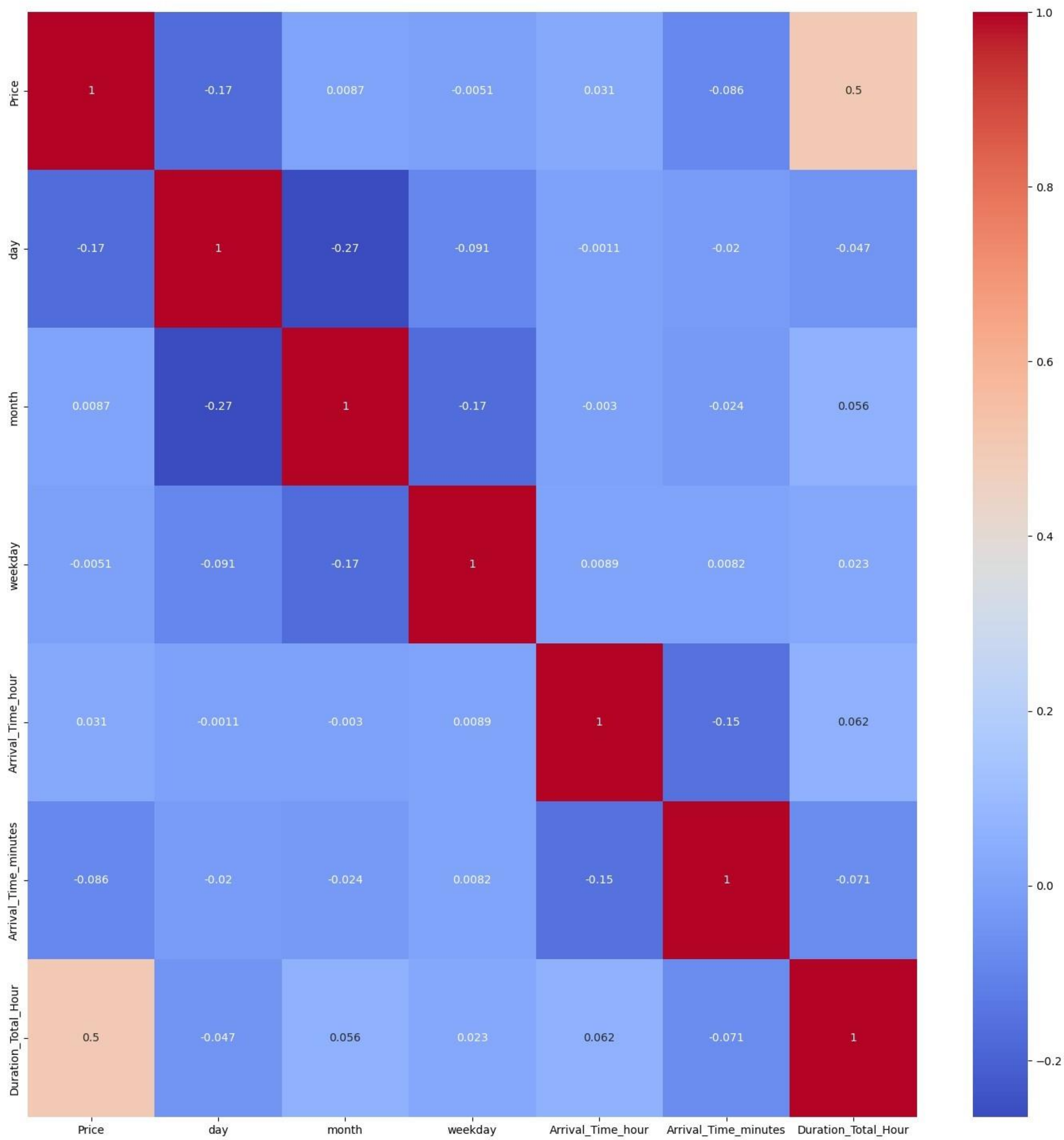



```
#  Source, Total Stops, and Price with Size as Duration
plt.figure(figsize=(14, 8))
sns.scatterplot(x='Source', y='Price', hue='Total_Stops',
size='Duration', data=data_eda)
plt.title('Source, Total Stops, and Price with Size as Duration')
plt.show()
```


Source, Total Stops, and Price with Size as Duration
Price
80000
70000
60000
50000
40000
30000
20000
10000
0
Banglore
Kolkata
Delhi
Chennai
Source
Total_Stops
0
2
1
3
4
Duration
2h 50m
7h 25m
19h
5h 25m
4h 45m
2h 25m
15h 30m
21h 5m
25h 30m
7h 50m
13h 15m
2h 35m
2h 15m
12h 10m
26h 35m
4h 30m
22h 35m
23h
20h 35m
5h 10m
15h 20m
2h 55m
13h 20m
15h 10m
5h 45m
5h 55m
13h 25m
22h
5h 30m
10h 25m
5h 15m
2h 30m
6h 15m
11h 55m
11h 5m
8h 30m
22h 5m
2h 45m
12h
16h 5m
19h 55m
3h 15m
25h 20m
3h
16h 15m
15h 5m
6h 30m
25h 5m
12h 25m
27h 20m
10h 15m
10h 30m
1h 30m
1h 25m
26h 30m
7h 20m
13h 30m
5h
19h 5m
14h 50m
2h 40m
22h 10m
9h 35m
10h
21h 20m
18h 45m
12h 20m
18h
9h 15m

Features:

```
data_eda.columns

Index(['Airline', 'Date of Journey', 'Source', 'Destination', 'Route',
       'Dep_Time', 'Arrival_Time', 'Duration', 'Total_Stops',
       'Additional_Info', 'Price', 'day', 'month', 'weekday',
       'Arrival_Time_hour', 'Arrival_Time_minutes',
'Duration_Total_Hour'],
      dtype='object')

plt.figure(figsize = (18,18))

sns.heatmap(data_eda.corr(),annot= True, cmap = "coolwarm")

plt.show()
```




```
data model

      Duration Total_Stops  Price  day  month  weekday  
Arrival_Time_hour  \
0      2h 50m            0   3897   24      3        6
1
1      7h 25m            2   7662    5      1        5
13
2         19h            2  13882    6      9        4
4
```

```
3        5h 25m            1   6218    5     12         3
23
4        4h 45m            1  13302    3      1         3
21
...         ...          ...     ...  ...     ...       ...
...
10678    2h 30m            0   4107    4      9         2
22
10679    2h 35m            0   4145   27      4         5
23
10680        3h            0   7229   27      4         5
11
10681    2h 40m            0  12648    3      1         3
14
10682    8h 20m            2  11753    5      9         3
19

       Arrival_Time_minutes  Duration_Total_Hour  Airline_Air
India  ...  \
0                        10             2.833333
0  ...
1                        15             7.416667

2                        25            19.000000

3                        30             5.416667

4                        35             4.750000
0  ...
...                     ...                  ...                ...  .
..
10678                    25             2.500000
0  ...
10679                    20             2.583333
1  ...
10680                    20             3.000000
0  ...
10681                    10             2.666667
0  ...
10682                    15             8.333333
1  ...

       Airline_Trujet  Airline_Vistara  Source_Chennai
Source Delhi  \
0                   0                0               0             0

1                   0                0               0             0

2                   0                0               0             1
```

```
3                       0                   0                  0                0

4                       0                   0                  0                0

...                   ...                 ...                ...              ...

10678                   0                   0                  0                0

10679                   0                   0                  0                0

10680                   0                   0                  0                0

10681                   0                   1                  0                0

10682                   0                   0                  0                1


        Source_Kolkata  Source_Mumbai  Destination_Cochin
Destination_Delhi  \
0                    0              0                   0
1
1                    1              0                   0
0
2                    0              0                   1
0
3                    1              0                   0
0
4                    0              0                   0
1
...                ...            ...                 ...
...
10678                1              0                   0
0
10679                1              0                   0
0
10680                0              0                   0
1
10681                0              0                   0
1
10682                0              0                   1
0

         Destination_Hyderabad  Destination_Kolkata
0                            0                    0
1                            0                    0
2                            0                    0
3                            0                    0
4                            0                    0
...                        ...                  ...
10678                        0                    0
```

```
10679                          0                      0
10680                          0                      0
10681                          0                      0
10682                          0                      0

[10462 rows x 25 columns]
```

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

Modeling:

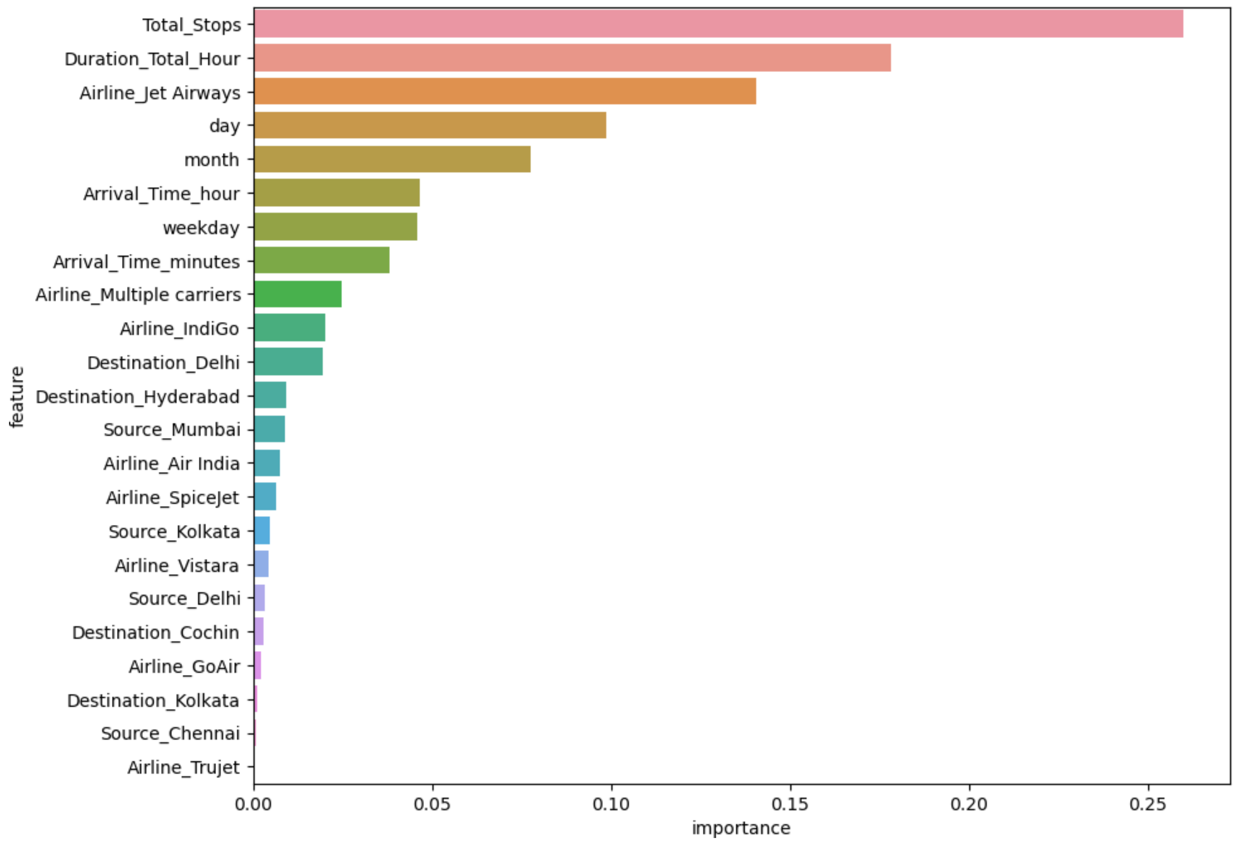
```
from sklearn.model_selection import train_test_split
```

Splitting the data

```
# 60% Train - 20% Val - 20% Test
X_train,X_test,y_train,y_test =
train_test_split(X,y,test_size=0.2,random_state = 42)
```

Feature Selection

```
from sklearn.ensemble import ExtraTreesRegressor
extractor = ExtraTreesRegressor()

extractor.fit(X_train,y_train)

ExtraTreesRegressor()

x_columns = X_train.columns
feature_rank = pd.DataFrame({'feature':x_columns,
'importance':extractor.feature_importances_})

feature_rank = feature_rank.sort_values('importance',ascending =
False)

plt.figure(figsize=(10,8))
sns.barplot(x='importance',y='feature',data=feature_rank)

<Axes: xlabel='importance', ylabel='feature'>
```




```
feature_rank['cumsum'] = feature_rank['importance'].cumsum()*100

feature_rank.head(15)
```

	feature	importance	cumsum
0	Total_Stops	0.259787	25.978693
6	Duration_Total_Hour	0.178139	43.792551
10	Airline_Jet Airways	0.140639	57.856429
1	day	0.098762	67.732653
2	month	0.077389	75.471525
4	Arrival_Time_hour	0.046616	80.133125
3	weekday	0.045805	84.713609
5	Arrival_Time_minutes	0.037964	88.510040
11	Airline_Multiple carriers	0.024545	90.964538
9	Airline_IndiGo	0.020016	92.966111
20	Destination_Delhi	0.019444	94.910548
21	Destination_Hyderabad	0.009054	95.815959
18	Source_Mumbai	0.009008	96.716795
7	Airline_Air India	0.007338	97.450575
12	Airline_SpiceJet	0.006395	98.090109

Model Building

```
from sklearn.metrics import r2_score,
mean_absolute_error,mean_squared_error
```

Defining a function to get metrics for val set

```
def predict(ml_model):
    print('Model name is: {}'.format(ml_model))
    model = ml_model.fit(X_train,y_train)
    print("Training Score: {}".format(model.score(X_train,y_train)))

    predictions = model.predict(X_test)

    r2score = r2_score (y_test,predictions)
    print('R2 Score is: {}'.format(r2score))


    print('MAE: {}'.format(mean_absolute_error(y_test,predictions)))
    print('MSE: {}'.format(mean_squared_error(y_test,predictions)))
    print('RMSE:
{}'.format(np.sqrt(mean_squared_error(y_test,predictions))))
```

```
predict(LinearRegression())
```

```
Model name is: LinearRegression()
Training Score: 0.5436238457678474
R2 Score is: 0.5470202809630111
MAE: 2112.4202978707826
MSE: 9444752.3355083
RMSE: 3073.2315785681203
```

```
predict(DecisionTreeRegressor())
```

```
Model name is: DecisionTreeRegressor()
Training Score: 0.969740512514991
R2 Score is: 0.684604526388076
MAE: 1411.542403248925
MSE: 6576082.793149785
RMSE: 2564.3874108936398
```

```
predict(RandomForestRegressor())
```

```
Model name is: RandomForestRegressor()
Training Score: 0.9493580477495638
R2 Score is: 0.7837484950241442
MAE: 1247.3299823523612
MSE: 4508903.645885126
RMSE: 2123.417915975356
```

```
from sklearn.model_selection import RandomizedSearchCV
params = {'n_estimators':[100,200,300,400,500], 'max_features' :
['auto','sqrt'], 'max_depth' : [5,10,15,20]}
rf = RandomForestRegressor()
rf_cv = RandomizedSearchCV(rf,params,cv=10,verbose = True, n_jobs=-1)
rf_cv.fit(X_train,y_train)
```

```
Fitting 10 folds for each of 10 candidates, totalling 100 fits
```

```
RandomizedSearchCV(cv=10, estimator=RandomForestRegressor(), n_jobs=-
1,
                   param_distributions={'max_depth': [5, 10, 15, 20],
                                        'max_features': ['auto',
'sqrt'],
                                        'n_estimators': [100, 200,
300, 400,
                                                         500]},
                   verbose=True)
```

```
rf_cv.best_params_
```

```
{'n_estimators': 100, 'max_features': 'sqrt', 'max_depth': 15}
```

```
predict(RandomForestRegressor(n_estimators = 400, max_features =
'sqrt',max_depth = 15))
```

```
Model name is: RandomForestRegressor(max_depth=15,
max_features='sqrt', n_estimators=400)
Training Score: 0.9158773724333831
R2 Score is: 0.7944075667998258
MAE: 1295.805965448281
MSE: 4286659.053430207
RMSE: 2070.4248485347657
```