1) Defining the problem statment In this project, we study the data which is in tabular format using various Python libraries like Pandas, Numpy, Matplotlib and Seaborn.

We study different columns of the table and try to co-relate them with others and find a relation between those two.

We try to find and analyze those key factors like class of travel, duration of fight, etc. which helps us understand the pricing of tickets to plan and schedule our air travel in efficient way

**About the dataset:**

1) Airline: The name of the airline company is stored in the airline column. It is a categorical feature having 6 different airlines.

2) Flight: Flight stores information regarding the plane's flight code. It is a categorical feature.

3) Source City: City from which the flight takes off. It is a categorical feature having 6 unique cities.

4) Departure Time: This is a derived categorical feature obtained created by grouping time periods into bins. It stores information about the departure time and have 6 unique time labels.

5) Stops: A categorical feature with 3 distinct values that stores the number of stops between the source and destination cities.

6) Arrival Time: This is a derived categorical feature created by grouping time intervals into bins. It has six distinct time labels and keeps information about the arrival time.

7) Destination City: City where the flight will land. It is a categorical feature having 6 unique cities.

8) Class: A categorical feature that contains information on seat class; it has two distinct values: Business and Economy.

9) Duration: A continuous feature that displays the overall amount of time it takes to travel between cities in **Hours.**

10)Days Left: This is a derived characteristic that is calculated by subtracting the trip date by the booking date.

11) Price: Target variable stores information of the ticket price.

In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import os
```

```
import warnings
warnings.filterwarnings("ignore")
```

In [2]:
```
df = pd.read_csv("C:\\Users\\Admin\\Desktop\\PythonFiles\\data_science\\Indian_airlines_analysis\\Indian_Airlines.csv")
df.head(20)
```

Out[2]:

| | airline | flight | source_city | departure_time | stops | arrival_time | destination_city | class | duration | days_left | price |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | SpiceJet | SG-8709 | Delhi | Evening | zero | Night | Mumbai | Economy | 2.17 | 1 | 5953 |
| 1 | SpiceJet | SG-8157 | Delhi | Early_Morning | zero | Morning | Mumbai | Economy | 2.33 | 1 | 5953 |
| 2 | AirAsia | I5-764 | Delhi | Early_Morning | zero | Early_Morning | Mumbai | Economy | 2.17 | 1 | 5956 |
| 3 | Vistara | UK-995 | Delhi | Morning | zero | Afternoon | Mumbai | Economy | 2.25 | 1 | 5955 |
| 4 | Vistara | UK-963 | Delhi | Morning | zero | Morning | Mumbai | Economy | 2.33 | 1 | 5955 |
| 5 | Vistara | UK-945 | Delhi | Morning | zero | Afternoon | Mumbai | Economy | 2.33 | 1 | 5955 |
| 6 | Vistara | UK-927 | Delhi | Morning | zero | Morning | Mumbai | Economy | 2.08 | 1 | 6060 |
| 7 | Vistara | UK-951 | Delhi | Afternoon | zero | Evening | Mumbai | Economy | 2.17 | 1 | 6060 |
| 8 | GO_FIRST | G8-334 | Delhi | Early_Morning | zero | Morning | Mumbai | Economy | 2.17 | 1 | 5954 |
| 9 | GO_FIRST | G8-336 | Delhi | Afternoon | zero | Evening | Mumbai | Economy | 2.25 | 1 | 5954 |
| 10 | GO_FIRST | G8-392 | Delhi | Afternoon | zero | Evening | Mumbai | Economy | 2.25 | 1 | 5954 |
| 11 | GO_FIRST | G8-338 | Delhi | Morning | zero | Afternoon | Mumbai | Economy | 2.33 | 1 | 5954 |
| 12 | Indigo | 6E-5001 | Delhi | Early_Morning | zero | Morning | Mumbai | Economy | 2.17 | 1 | 5955 |
| 13 | Indigo | 6E-6202 | Delhi | Morning | zero | Afternoon | Mumbai | Economy | 2.17 | 1 | 5955 |
| 14 | Indigo | 6E-549 | Delhi | Afternoon | zero | Evening | Mumbai | Economy | 2.25 | 1 | 5955 |
| 15 | Indigo | 6E-6278 | Delhi | Morning | zero | Morning | Mumbai | Economy | 2.33 | 1 | 5955 |
| 16 | Air_India | AI-887 | Delhi | Early_Morning | zero | Morning | Mumbai | Economy | 2.08 | 1 | 5955 |
| 17 | Air_India | AI-665 | Delhi | Early_Morning | zero | Morning | Mumbai | Economy | 2.17 | 1 | 5955 |
| 18 | AirAsia | I5-747 | Delhi | Evening | one | Early_Morning | Mumbai | Economy | 12.25 | 1 | 5949 |
| 19 | AirAsia | I5-747 | Delhi | Evening | one | Morning | Mumbai | Economy | 16.33 | 1 | 5949 |

In [3]:
```
df.shape
```

Out[3]: (300153, 11)

In [4]:
```
# for col in df.columns:
#     if df[col].dtype == "O":
#         print(col,":",df[col].unique())

for col in df.columns:
    print("'{}' field has {} unique number of values, Dtype: {}".format(col,df[col].nunique(), df[col].dtype))
    print("'{}' has unique values: {}".format(col, df[col].unique()))
    print("")
```

```
'airline' field has 6 unique number of values, Dtype: object
'airline' has unique values: ['SpiceJet' 'AirAsia' 'Vistara' 'GO_FIRST' 'Indigo' 'Air_India']

'flight' field has 1561 unique number of values, Dtype: object
'flight' has unique values: ['SG-8709' 'SG-8157' 'I5-764' ... '6E-7127' '6E-7259' 'AI-433']

'source_city' field has 6 unique number of values, Dtype: object
'source_city' has unique values: ['Delhi' 'Mumbai' 'Bangalore' 'Kolkata' 'Hyderabad' 'Chennai']

'departure_time' field has 6 unique number of values, Dtype: object
'departure_time' has unique values: ['Evening' 'Early_Morning' 'Morning' 'Afternoon' 'Night' 'Late_Night']

'stops' field has 3 unique number of values, Dtype: object
'stops' has unique values: ['zero' 'one' 'two_or_more']

'arrival_time' field has 6 unique number of values, Dtype: object
'arrival_time' has unique values: ['Night' 'Morning' 'Early_Morning' 'Afternoon' 'Evening' 'Late_Night']

'destination_city' field has 6 unique number of values, Dtype: object
'destination_city' has unique values: ['Mumbai' 'Bangalore' 'Kolkata' 'Hyderabad' 'Chennai' 'Delhi']

'class' field has 2 unique number of values, Dtype: object
'class' has unique values: ['Economy' 'Business']

'duration' field has 476 unique number of values, Dtype: float64
'duration' has unique values: [ 2.17  2.33  2.25  2.08 12.25 16.33 11.75 14.5  15.67  3.75  2.5   5.83
  8.    6.   14.67 16.17 18.   23.17 24.17  8.83  4.5  15.25 11.   19.08
 22.83 26.42 17.75 19.58 26.67 15.17 20.83 11.42 22.25 26.   21.75  3.83
  4.42  7.67  8.33 10.42 23.75 19.5   6.5  12.42 21.08 28.17 28.25  9.25
 17.92  7.08 13.83  7.58 15.83 24.42  4.17  4.25  5.08 29.33 17.   27.17
 24.75  5.75 12.75 13.75 17.83  5.5  23.83  5.   26.5  12.83  8.92 11.17
 12.17 15.58 15.75  7.92 13.25 16.   22.75  6.33  7.25 30.08 18.25  6.08
  2.   12.33  3.5  10.25 14.17 25.58  4.08  9.75  6.67  9.67 10.08 12.58
  7.    8.25 15.5  10.17 23.5  25.75 11.5  21.42 14.25  7.75  5.33  5.67
  4.75 19.33  6.25 10.33  9.08 15.42  4.83 25.83  7.5  27.58 28.42  6.42
 24.58 16.5  11.33 24.83 14.92 26.08  5.42 28.5  27.33  8.67 20.42 20.08
  5.92 20.58 26.17 17.5  18.33 21.83 26.83 13.42 19.25 23.58 23.92 12.08
 14.42 25.92 18.58 13.08  4.58 21.58 10.5  11.67  8.58 24.67 14.75 17.17
  7.33  9.17 11.58 23.08 25.25 21.25  8.42 10.92 14.    4.92 25.08 12.67
 28.08 21.33 14.33 14.83 15.33 15.92  9.5  27.75 24.92 22.58  6.58  9.33
 13.   13.58 11.08  5.58 15.    6.75  9.83 11.25 11.83  9.58  8.5  14.08
 10.58 22.17 22.08  8.75 15.08 25.5  16.42 18.92  2.42  3.92 24.5  23.
 10.   19.   16.58  7.83  3.17 16.25 24.08  4.67  6.92 13.67 25.    8.08
 17.08  7.42 22.5  12.92  6.17 23.67  5.17 13.33 17.25 25.42 29.17  9.
  9.42  8.17 26.58 22.33 12.5  21.   24.25 27.5  18.17 16.67  2.83  2.75
```

```
 2.92  3.    7.17 25.67  2.67 10.83 18.5  21.17 23.33 24.   17.33 27.42
26.25  4.33  9.92 19.67 26.33 16.92 25.33 27.25 27.92 27.   12.   10.67
28.   21.67 21.92 13.92 13.5   2.58 22.42 24.33 18.08 29.58 29.67 31.25
33.17 36.92 20.25 22.92  6.83  5.25 20.67 17.58 19.17 17.42 11.92 31.5
19.83 27.08 26.92 21.5   6.07 16.83 20.5  18.67 10.75 18.42 28.58 19.92
20.   22.   29.83 19.42 25.17  3.08 18.75 27.83 20.17 20.33 20.75 28.67
35.83 34.83 23.25 26.75 19.75 27.67 16.08 30.25 28.75 30.58 18.83 13.17
30.5  29.5  28.83 30.67 17.67 14.58 29.   28.33 39.67 16.75  1.92 37.42
36.42 23.42 22.67 33.75 30.83 28.92  1.83  3.67  4.    1.58  1.67  1.75
20.92 33.5  30.33 35.25 31.83 40.   31.67 32.08 36.17 30.75 33.83 34.5
31.08 30.17 36.08 33.33 37.   34.08 37.08  1.42  1.33  1.5  37.83 29.25
34.25 36.83  3.42 31.58 33.58 34.58 32.17  1.17 35.17 37.67 37.25 39.75
31.   37.75 33.25 31.17 29.92 31.92 41.58 37.17 32.   35.5  39.   31.42
34.33 38.58 33.67  3.33  3.58 34.17 37.33 35.42 30.   33.08 38.83 32.67
29.42 35.08 32.5  29.75  1.08  1.   32.42  1.25 34.92 35.75 31.75 33.
34.75 36.58 41.83 36.5  33.42 36.25 40.67  0.92 37.58  0.83 40.75 47.75
36.   40.42 33.92 30.42 40.5  29.08 38.5  34.67  6.98 30.92 35.58  3.25
39.83 37.92 37.5  39.08 38.75 34.42 40.92 13.4  31.33 42.   32.75 32.33
32.83 35.92 32.25 38.67 38.33 39.25 41.08 32.92 36.75 35.67 35.33 49.83
34.   47.08 45.83 39.42 39.92 32.58 44.5  41.5 ]

'days_left' field has 49 unique number of values, Dtype: int64
'days_left' has unique values: [ 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48
 49]

'price' field has 12157 unique number of values, Dtype: int64
'price' has unique values: [ 5953  5956  5955 ... 87051 74731 77105]
```

In [5]:
```
df.isna().sum()    #There are no Null values in the dataset
```

Out[5]:
```
airline             0
flight              0
source_city         0
departure_time      0
stops               0
arrival_time        0
destination_city    0
class               0
duration            0
days_left           0
price               0
dtype: int64
```

In [6]:
```
df[df["flight"] == "SG-8157"]
```

Out[6]:

| | airline | flight | source_city | departure_time | stops | arrival_time | destination_city | class | duration | days_left | price |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **1** | SpiceJet | SG-8157 | Delhi | Early_Morning | zero | Morning | Mumbai | Economy | 2.33 | 1 | 5953 |
| **138** | SpiceJet | SG-8157 | Delhi | Early_Morning | zero | Morning | Mumbai | Economy | 2.33 | 2 | 5955 |
| **252** | SpiceJet | SG-8157 | Delhi | Early_Morning | zero | Morning | Mumbai | Economy | 2.33 | 3 | 5953 |
| **401** | SpiceJet | SG-8157 | Delhi | Early_Morning | zero | Morning | Mumbai | Economy | 2.33 | 4 | 5953 |
| **568** | SpiceJet | SG-8157 | Delhi | Early_Morning | zero | Morning | Mumbai | Economy | 2.33 | 5 | 5953 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **41885** | SpiceJet | SG-8157 | Delhi | Early_Morning | one | Night | Chennai | Economy | 15.00 | 42 | 5107 |
| **41886** | SpiceJet | SG-8157 | Delhi | Early_Morning | one | Early_Morning | Chennai | Economy | 25.67 | 42 | 5107 |
| **42056** | SpiceJet | SG-8157 | Delhi | Early_Morning | one | Night | Chennai | Economy | 15.00 | 43 | 5107 |
| **42057** | SpiceJet | SG-8157 | Delhi | Early_Morning | one | Early_Morning | Chennai | Economy | 25.67 | 43 | 5107 |
| **42226** | SpiceJet | SG-8157 | Delhi | Early_Morning | one | Night | Chennai | Economy | 15.00 | 44 | 4942 |

131 rows × 11 columns

Question 1: What are the number of flights operated by each Airline???

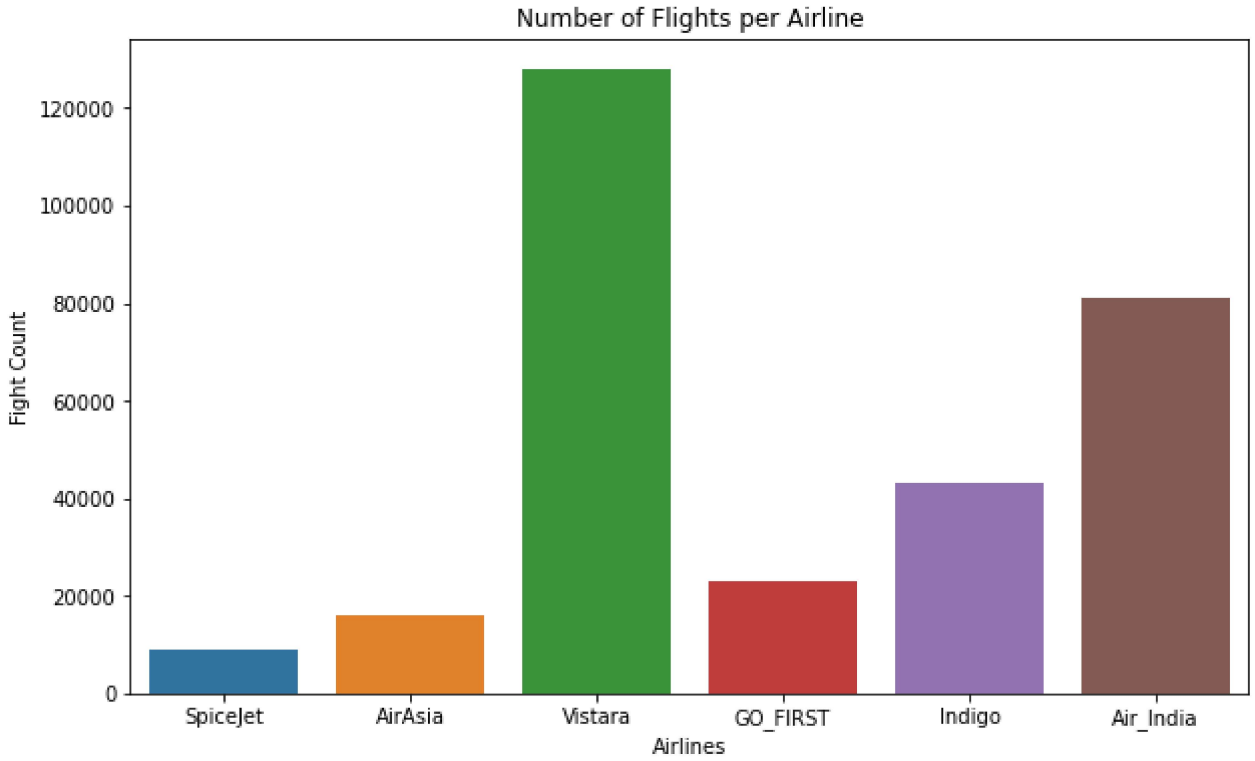
In [7]:
```
df.groupby("airline")["flight"].count()
```

Out[7]:
```
airline
AirAsia       16098
Air_India     80892
GO_FIRST      23173
Indigo        43120
SpiceJet       9011
Vistara      127859
Name: flight, dtype: int64
```

SOLUTION1 : The number of flights associated with each airline is as follows: AirAsia (16,098), Air_India (80,892), GO_FIRST (23,173), Indigo (43,120), SpiceJet (9,011), Vistara (127,859)

## The diagram below gives the representation of the solution above:

In [8]:
```
plt.figure(figsize=(10,6))
A = sns.countplot(x="airline",data=df)
A.set(xlabel = "Airlines" , ylabel = "Fight Count",title="Number of Flights per Airline")
plt.show()
```




Question 2: What is the price range according to the class of travel???

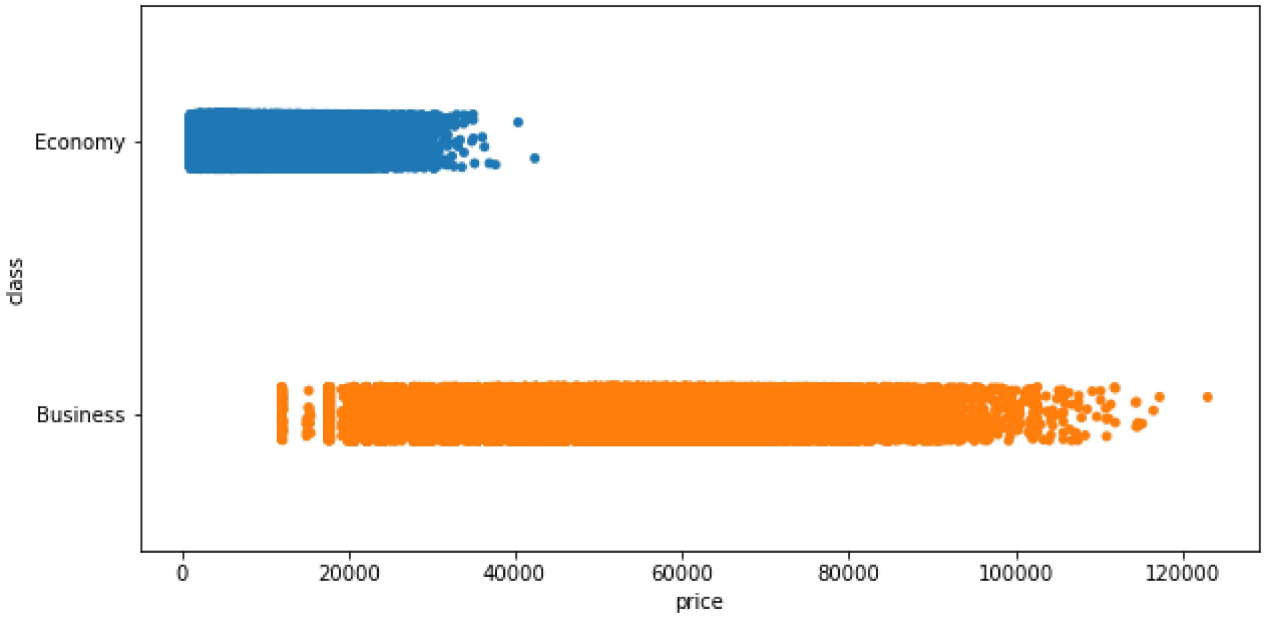
In [9]:
```
df[df["price"] == df["price"].min() ].head()
print("The minimum flight price is:", 1105)
```

```
The minimum flight price is: 1105
```

In [10]:
```
df[df["price"] == df["price"].max()]
print("The maximum flight price is:",123071," and it is by the 'Vistara' Airline")
```

```
The maximum flight price is: 123071  and it is by the 'Vistara' Airline
```

In [11]:
```
plt.figure(figsize=(10,5))
A = sns.stripplot(x = "price", y = "class" ,data = df)
# A.axvline(2500)
# A.axvline(22500)
# A.axvline(25000,color="r")
# A.axvline(95000,color="r")

plt.show()
```




## From the diagram above, we can conclude that the price range of `Business` class is between 20000 & 95000 while that of `Economy` class is between 2000 - 25000

Question 3: What is the availability of Tickets according to Class of Travel???

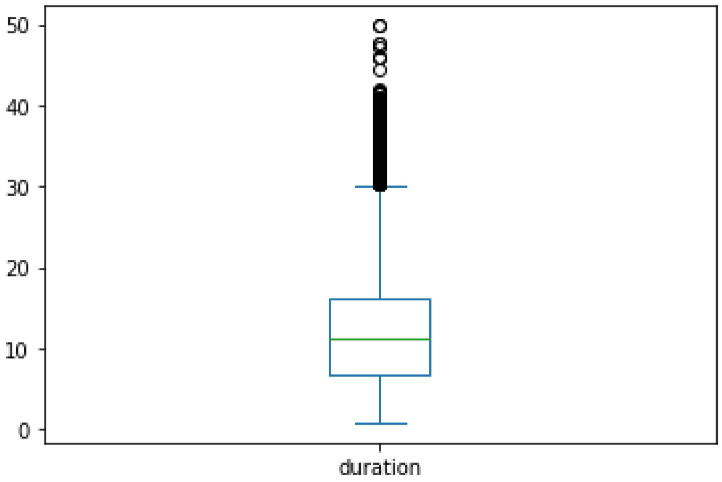
In [12]:
```
plt.figure(figsize=(10,5))

A = sns.countplot(x="class", data = df)
A.set(title="Availability of Tickets according to Class of Travel",xlabel = "Class of Travel",ylabel = "Ticket Availabi

# dir(sns)
plt.show()
```


Solution 3: From the diagram above, the `Economy` class had over 20,000 Tickets while the `Business` class had nearly 10,000 Tickets: meaning that the availability of `Economy` class was nearly as twice as the `Business` class

Question 4: What is the Price of Ticket for different Airlines based on the Duration of Flight???

In [13]:
```
df["duration"].plot.box()
```

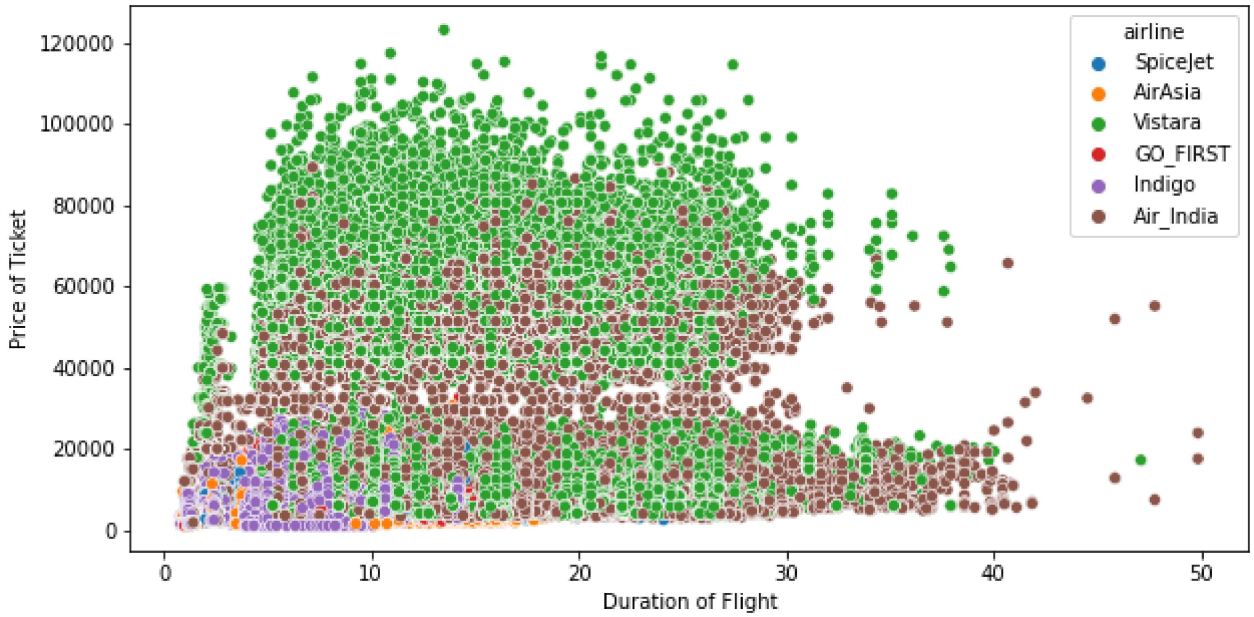
Out[13]:
```
<AxesSubplot:>
```




In [14]:
```
df.groupby(["airline","duration"]).mean()["price"]
```

Out[14]:
```
airline  duration
AirAsia  0.92         2137.000000
         1.00         2123.644444
         1.08         2581.423913
         1.17         2372.707143
         1.25         2181.877193
                          ...
Vistara  38.83       13840.000000
         39.00       20293.000000
         39.75       18746.500000
         40.00       19400.000000
         47.08       17223.000000
Name: price, Length: 1597, dtype: float64
```

In [15]:
```
plt.figure(figsize=(10,5))
A = sns.scatterplot( x = df['duration'],y = df['price'], hue= df["airline"])
A.set(xlabel="Duration of Flight",ylabel = "Price of Ticket")

plt.show()
```




Solution 4: By observing the figure above, we can see the distribution of ticket prices varying with the duration of Flight

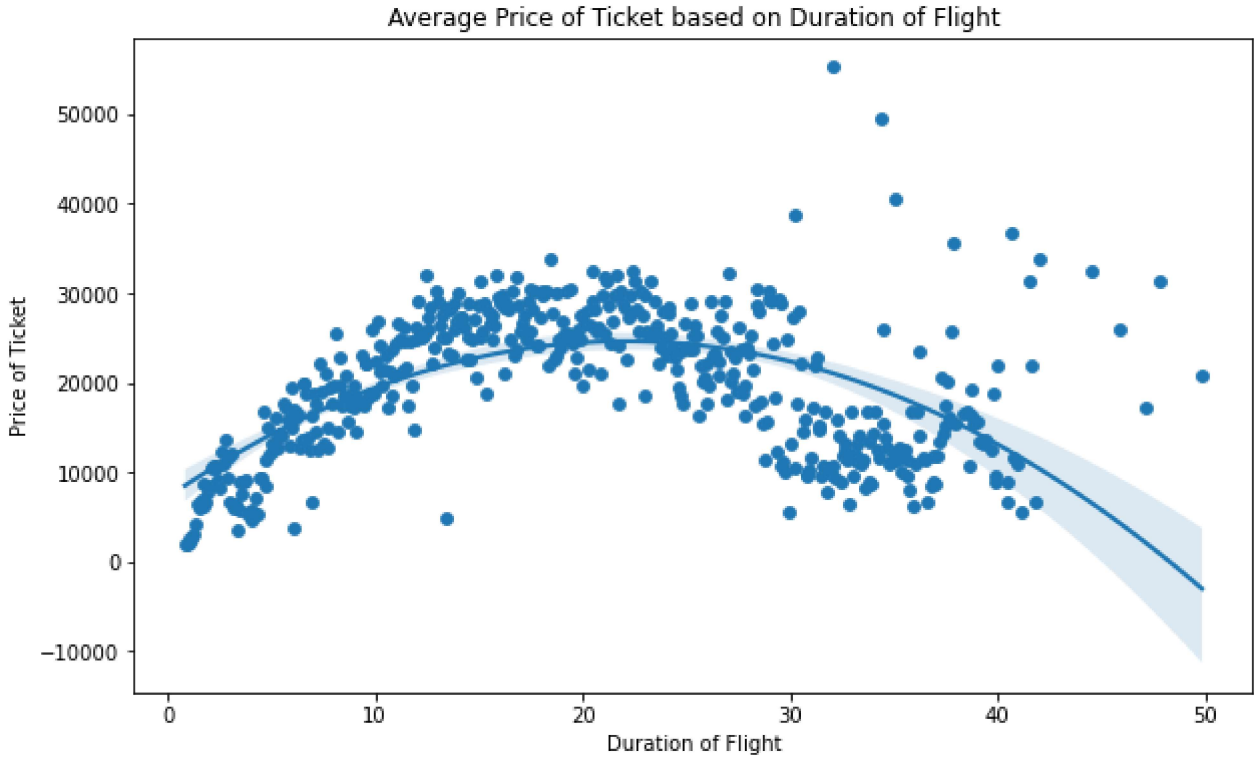
Question 5: How does the Price of Ticket vary with the Duration of Flight???

In [16]:
```
df.groupby("duration").mean()["price"]
```

Out[16]:
```
duration
0.83      1973.557692
0.92      2003.538462
1.00      2266.063660
1.08      2589.311031
1.17      2632.209375
              ...
44.50    32439.000000
45.83    25851.333333
47.08    17223.000000
47.75    31291.500000
49.83    20714.500000
Name: price, Length: 476, dtype: float64
```

In [17]:
```
plt.figure(figsize=(10,6))
tmp_df = df.groupby(["duration"])['price'].mean().reset_index()
A = sns.scatterplot(x = 'duration',y ="price", data= tmp_df, )
A = sns.regplot(x="duration",y="price",data = tmp_df, order= 2)
```

```
A.set(xlabel="Duration of Flight",ylabel="Price of Ticket",title="Average Price of Ticket based on Duration of Flight")
# A.axvline(20 , color ='r')


plt.show()
```




Solution 5: From the figure above, we see that there exists a non-linear relationship between the `Duration` of ticket and the average `Price` of Flight

Question 6: How do Ticket `Prices` vary according to `Departure` time & `Arrival` time???

In [18]:
```
df.groupby(["departure_time","arrival_time"])["price"].mean()
```

Out[18]:

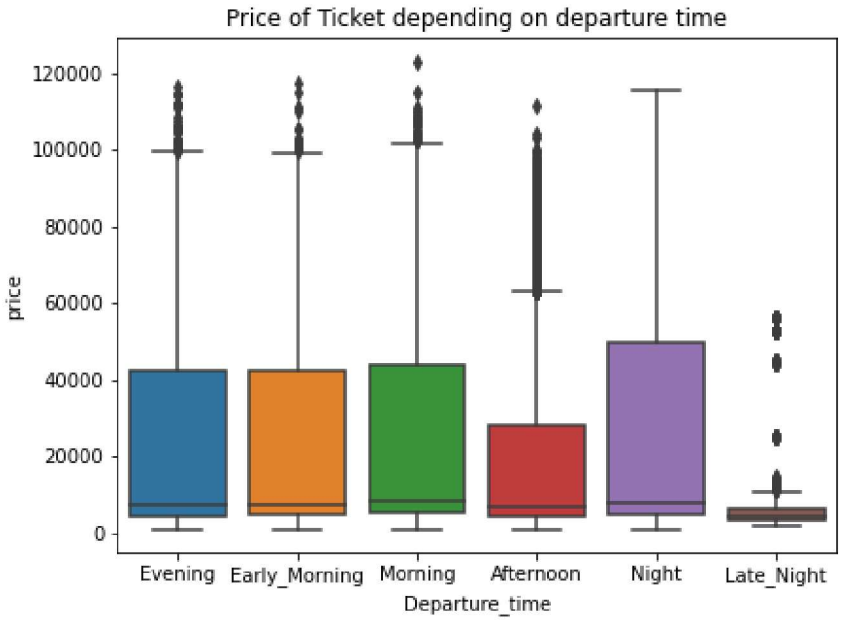
| departure_time | arrival_time | price |
|---|---|---|
| Afternoon | Afternoon | 14741.746011 |
| | Early_Morning | 21159.511401 |
| | Evening | 15548.836651 |
| | Late_Night | 10811.615662 |
| | Morning | 26056.580994 |
| | Night | 19097.798757 |
| Early_Morning | Afternoon | 16564.731082 |
| | Early_Morning | 9540.191523 |
| | Evening | 24804.077441 |
| | Late_Night | 29638.037500 |
| | Morning | 12299.838792 |
| | Night | 25792.007764 |
| Evening | Afternoon | 25569.672277 |
| | Early_Morning | 18175.929529 |
| | Evening | 24612.547643 |
| | Late_Night | 8547.029235 |
| | Morning | 28326.216482 |
| | Night | 15641.705242 |
| Late_Night | Afternoon | 18520.684932 |
| | Early_Morning | 4548.700599 |
| | Evening | 21217.106383 |
| | Late_Night | 4288.291971 |
| | Morning | 7104.374684 |
| | Night | 29439.811321 |
| Morning | Afternoon | 13271.433547 |
| | Early_Morning | 22625.786578 |
| | Evening | 21636.529952 |
| | Late_Night | 23624.505755 |
| | Morning | 17548.738563 |
| | Night | 26827.897551 |
| Night | Afternoon | 28115.362126 |
| | Early_Morning | 12075.785461 |
| | Evening | 31425.824194 |
| | Late_Night | 6590.831766 |
| | Morning | 25573.642137 |
| | Night | 17817.956009 |

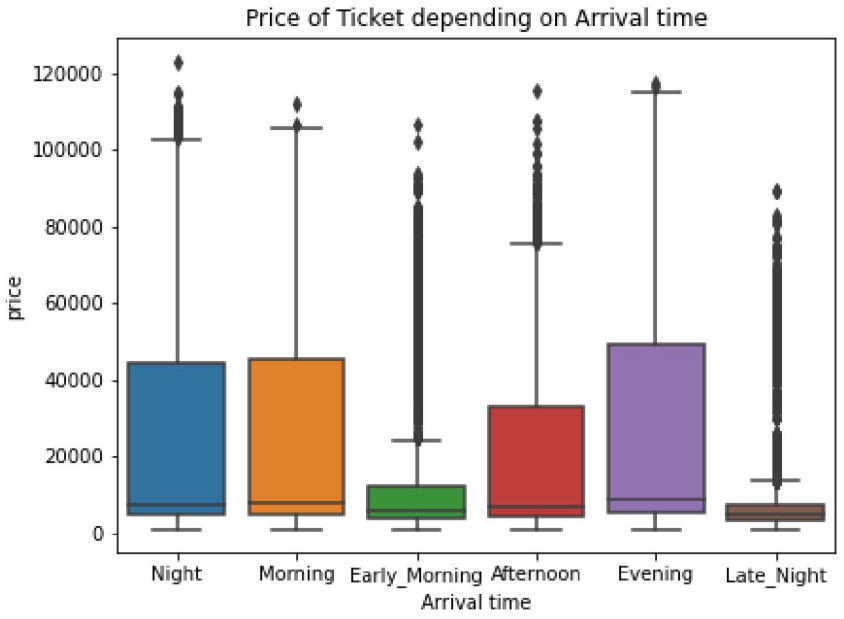
Name: price, dtype: float64

In [19]:

```
plt.figure(figsize=(15,5))
plt.subplot(1,2,1)
A = sns.boxplot(data=df, y= "price", x="departure_time")
A.set(xlabel="Departure_time", title="Price of Ticket depending on departure time")

plt.subplot(1,2,2)
```

```
B = sns.boxplot(data=df, y= "price", x="arrival_time")
B.set(xlabel="Arrival time",title="Price of Ticket depending on Arrival time")


plt.show()
```




Solution 6: We can see that flights departing at late night & also arriving at late night are usually the cheapest amongst others

Question 7: How do the prices vary based on the number of stops & Airline???

In [20]:
```
df.groupby("stops").mean()["price"].reset_index(name="price").sort_values("price",ascending=False)
```

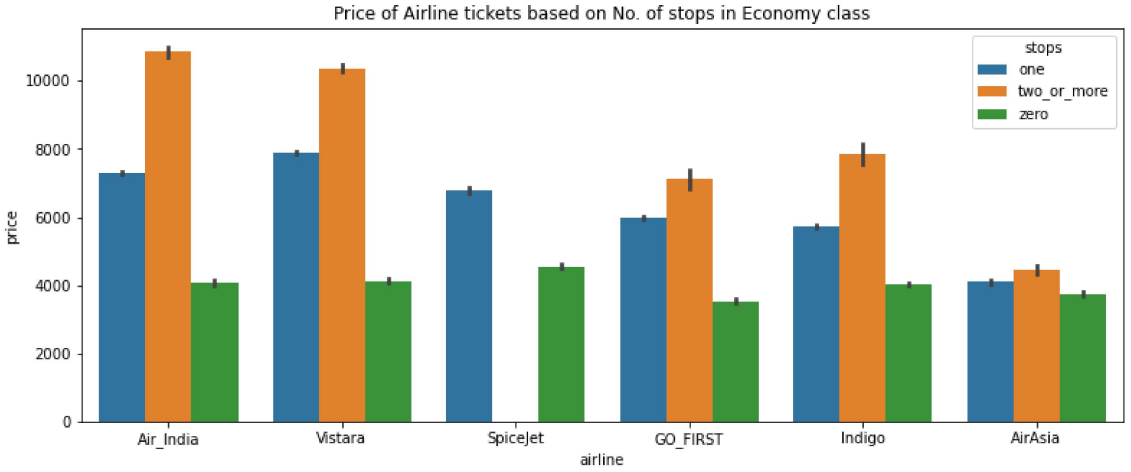
Out[20]:

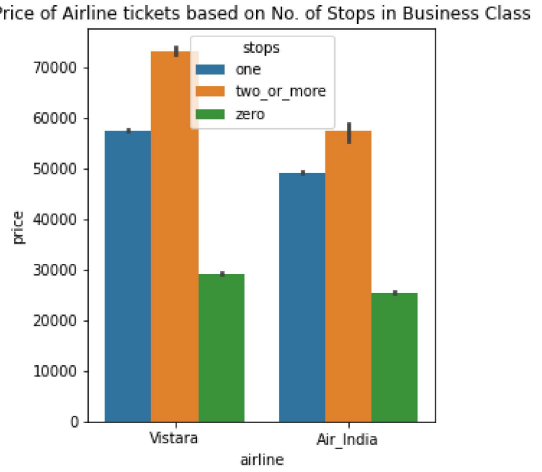
| | stops | price |
|---|---|---|
| **0** | one | 22900.992482 |
| **1** | two_or_more | 14113.450775 |
| **2** | zero | 9375.938535 |

In [21]:
```
plt.figure()
fig, axs = plt.subplots(1,2,gridspec_kw = {"width_ratios":[3,1]}, figsize=(19,5))
A = sns.barplot(y="price",x="airline", data=df.loc[df["class"] == "Economy"].sort_values("price" ,ascending=False),ax =
```

```
A.set(title="Price of Airline tickets based on No. of stops in Economy class")

B = sns.barplot(x="airline", y="price", data=df.loc[df["class"] == "Business"].sort_values("price" ,ascending=False),ax
B.set(title="Price of Airline tickets based on No. of Stops in Business Class")



plt.show()
```

```
<Figure size 432x288 with 0 Axes>
```



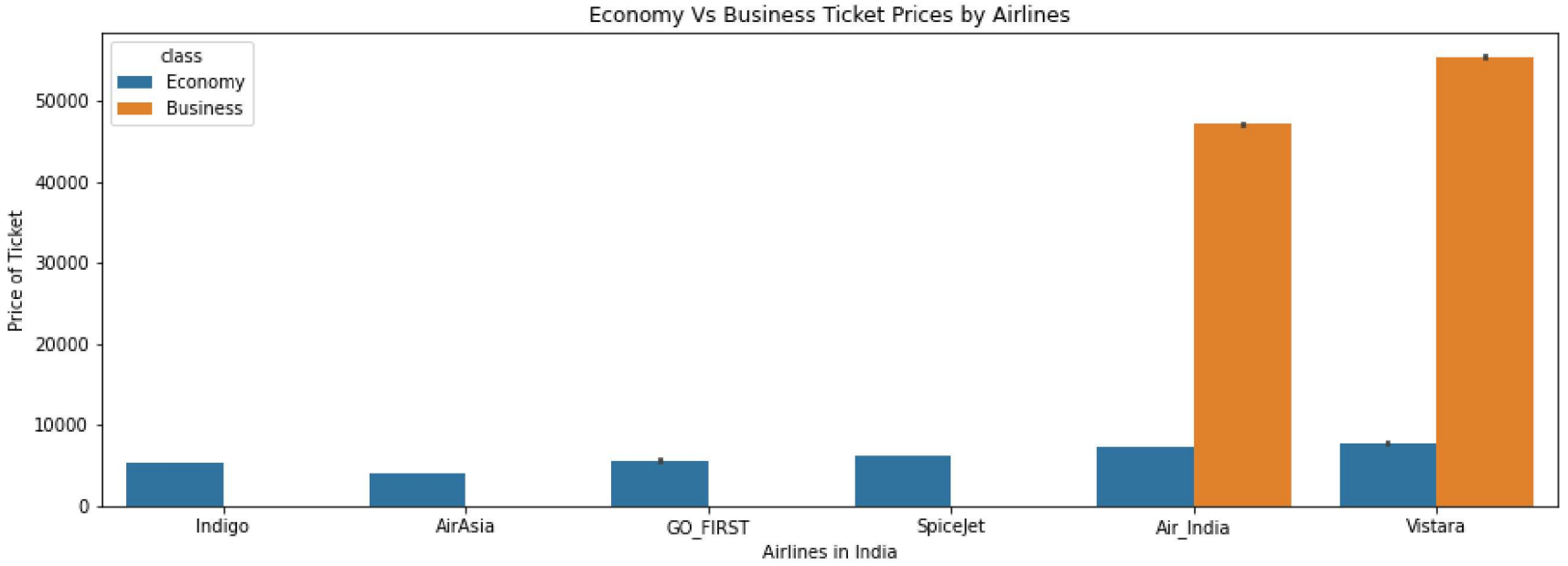

Solution 7: From the chart above, we can see that the price of AIrline tickets with zero stops are the cheapest while flights . One can also note that SpiceJet Airline do not have flights with two or more stops(2+ stops). AirAsia particularly seems to show little variation between its non-stop, one-stop and teo or more stops.

Question 8: How do prices vary across different airlines and class of Travel???

In [22]:
```
df.groupby(["airline","class"])["price"].mean()
```

Out[22]:
```
airline    class
AirAsia    Economy      4091.072742
Air_India  Business    47131.039212
           Economy      7313.682169
GO_FIRST   Economy      5652.007595
Indigo     Economy      5324.216303
SpiceJet   Economy      6179.278881
Vistara    Business    55477.027777
           Economy      7806.943645
Name: price, dtype: float64
```

In [23]:
```
plt.figure(figsize = (15,5))
A = sns.barplot(data=df.sort_values("price"), x = "airline" , y="price",hue="class")
A.set(title="Economy Vs Business Ticket Prices by Airlines",xlabel='Airlines in India', ylabel='Price of Ticket')


plt.show()
```




Solution 8: From the figure above, we can see AirAsia offers the cheapest price Tickets & that 'Business' class tickets for 'Vistara' cost much higher than 'Air India'

In [24]:
```
df[(df["airline"] == "Vistara") & (df["flight"] == "UK-996")]["source_city"].value_counts()
```

Out[24]:
```
Mumbai    1266
Name: source_city, dtype: int64
```

In [25]:
```
df[df["airline"] == "Vistara"]["flight"].unique()
```

Out[25]:
```
array(['UK-995', 'UK-963', 'UK-945', 'UK-927', 'UK-951', 'UK-933',
       'UK-813', 'UK-817', 'UK-819', 'UK-801', 'UK-815', 'UK-955',
       'UK-627', 'UK-899', 'UK-747', 'UK-809', 'UK-737', 'UK-871',
       'UK-977', 'UK-637', 'UK-835', 'UK-705', 'UK-707', 'UK-673',
       'UK-879', 'UK-837', 'UK-985', 'UK-953', 'UK-683', 'UK-859',
       'UK-829', 'UK-706', 'UK-839', 'UK-833', 'UK-847', 'UK-855',
       'UK-981', 'UK-975', 'UK-993', 'UK-943', 'UK-941', 'UK-811',
       'UK-807', 'UK-727', 'UK-812', 'UK-671', 'UK-741', 'UK-721',
       'UK-641', 'UK-994', 'UK-910', 'UK-958', 'UK-960', 'UK-940',
       'UK-944', 'UK-996', 'UK-988', 'UK-651', 'UK-863', 'UK-851',
       'UK-613', 'UK-875', 'UK-852', 'UK-853', 'UK-773', 'UK-771',
       'UK-825', 'UK-775', 'UK-970', 'UK-928', 'UK-823', 'UK-621',
       'UK-865', 'UK-841', 'UK-877', 'UK-655', 'UK-954', 'UK-930',
       'UK-986', 'UK-950', 'UK-902', 'UK-653', 'UK-873', 'UK-849',
       'UK-845', 'UK-857', 'UK-821', 'UK-827', 'UK-861', 'UK-820',
       'UK-802', 'UK-810', 'UK-808', 'UK-816', 'UK-818', 'UK-858',
       'UK-657', 'UK-850', 'UK-854', 'UK-755', 'UK-867', 'UK-814',
       'UK-897', 'UK-893', 'UK-864', 'UK-846', 'UK-866', 'UK-778',
       'UK-772', 'UK-776', 'UK-774', 'UK-738', 'UK-720', 'UK-708',
       'UK-880', 'UK-860', 'UK-890', 'UK-870', 'UK-830', 'UK-876',
       'UK-878', 'UK-874', 'UK-659', 'UK-898', 'UK-894', 'UK-838',
       'UK-836', 'UK-832', 'UK-834', 'UK-822', 'UK-824', 'UK-826',
       'UK-828'], dtype=object)
```

In [40]:
```
df["stops"] = df["stops"].replace({"zero":0 , "one":1,"two_or_more":2})
df[(df["airline"] == "Vistara") & (df["flight"] == "UK-996") & (df["days_left"] == 1)
  & (df["arrival_time"] == "Morning") & (df["departure_time"] == "Evening")].head(150)
```

Out[40]:

| | airline | flight | source_city | departure_time | stops | arrival_time | destination_city | class | duration | days_left | price |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **52888** | Vistara | UK-996 | Mumbai | Evening | 1 | Morning | Bangalore | Economy | 16.33 | 1 | 16653 |
| **52892** | Vistara | UK-996 | Mumbai | Evening | 1 | Morning | Bangalore | Economy | 17.33 | 1 | 17073 |
| **70363** | Vistara | UK-996 | Mumbai | Evening | 1 | Morning | Hyderabad | Economy | 15.17 | 1 | 16233 |
| **77353** | Vistara | UK-996 | Mumbai | Evening | 1 | Morning | Chennai | Economy | 15.42 | 1 | 16653 |
| **230011** | Vistara | UK-996 | Mumbai | Evening | 1 | Morning | Bangalore | Business | 16.33 | 1 | 78240 |
| **230012** | Vistara | UK-996 | Mumbai | Evening | 1 | Morning | Bangalore | Business | 17.33 | 1 | 78240 |

In [30]:
```
pd.options.display.max_rows
```

Out[30]:
```
150
```