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

In [2]:
```python
df=pd.read_csv('C:\Machine learning\Projects/Data_Train.xlsx - Sheet1.csv')
```

In [3]:
```python
df.head()
```

Out[3]:

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Duration	Total_Stops	Addi
0	IndiGo	24/03/2019	Banglore	New Delhi	BLR → DEL	22:20	01:10 22 Mar	2h 50m	non-stop	
1	Air India	1/05/2019	Kolkata	Banglore	CCU → IXR → BBI → BLR	05:50	13:15	7h 25m	2 stops	
2	Jet Airways	9/06/2019	Delhi	Cochin	DEL → LKO → BOM → COK	09:25	04:25 10 Jun	19h	2 stops	
3	IndiGo	12/05/2019	Kolkata	Banglore	CCU → NAG → BLR	18:05	23:30	5h 25m	1 stop	
4	IndiGo	01/03/2019	Banglore	New Delhi	BLR → NAG → DEL	16:50	21:35	4h 45m	1 stop	

In [4]:
```python
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 [5]:
```python
df.isnull().sum()
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

```
Out[5]: 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 [6]: df.shape
```

```
Out[6]: (10683, 11)
```

```
In [7]: df[df['Total_Stops'].isnull()]
```

Out[7]:

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

```
In [8]: df.dropna(inplace=True)
```

```
In [9]: df.isnull().sum()
```

```
Out[9]: 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
        dtype: int64
```

```
In [11]: d=df.copy()
```

```
In [12]: d.head()
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

Out[12]:

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Duration	Total_Stops	Addi
0	IndiGo	24/03/2019	Banglore	New Delhi	BLR → DEL	22:20	01:10 22 Mar	2h 50m	non-stop	
1	Air India	1/05/2019	Kolkata	Banglore	CCU → IXR → BBI → BLR	05:50	13:15	7h 25m	2 stops	
2	Jet Airways	9/06/2019	Delhi	Cochin	DEL → LKO → BOM → COK	09:25	04:25 10 Jun	19h	2 stops	
3	IndiGo	12/05/2019	Kolkata	Banglore	CCU → NAG → BLR	18:05	23:30	5h 25m	1 stop	
4	IndiGo	01/03/2019	Banglore	New Delhi	BLR → NAG → DEL	16:50	21:35	4h 45m	1 stop	

In [13]:
```
d.dtypes
```

Out[13]:
```
Airline            object
Date_of_Journey    object
Source             object
Destination        object
Route              object
Dep_Time           object
Arrival_Time       object
Duration           object
Total_Stops        object
Additional_Info    object
Price               int64
dtype: object
```

In [14]:
```
def change_into_datetime(col):
    d[col]=pd.to_datetime(d[col])
```

In [15]:
```
d.columns
```

Out[15]:
```
Index(['Airline', 'Date_of_Journey', 'Source', 'Destination', 'Route',
       'Dep_Time', 'Arrival_Time', 'Duration', 'Total_Stops',
       'Additional_Info', 'Price'],
      dtype='object')
```

In [16]:
```
for feature in ['Date_of_Journey','Dep_Time', 'Arrival_Time']:
    change_into_datetime(feature)
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

```
C:\Users\HP\anaconda3\lib\site-packages\pandas\core\tools\datetimes.py:1047: UserWarnin
g: Parsing '24/03/2019' in DD/MM/YYYY format. Provide format or specify infer_datetime_f
ormat=True for consistent parsing.
  cache_array = _maybe_cache(arg, format, cache, convert_listlike)
C:\Users\HP\anaconda3\lib\site-packages\pandas\core\tools\datetimes.py:1047: UserWarnin
g: Parsing '24/06/2019' in DD/MM/YYYY format. Provide format or specify infer_datetime_f
ormat=True for consistent parsing.
  cache_array = _maybe_cache(arg, format, cache, convert_listlike)
C:\Users\HP\anaconda3\lib\site-packages\pandas\core\tools\datetimes.py:1047: UserWarnin
g: Parsing '27/05/2019' in DD/MM/YYYY format. Provide format or specify infer_datetime_f
ormat=True for consistent parsing.
  cache_array = _maybe_cache(arg, format, cache, convert_listlike)
C:\Users\HP\anaconda3\lib\site-packages\pandas\core\tools\datetimes.py:1047: UserWarnin
g: Parsing '18/04/2019' in DD/MM/YYYY format. Provide format or specify infer_datetime_f
ormat=True for consistent parsing.
  cache_array = _maybe_cache(arg, format, cache, convert_listlike)
C:\Users\HP\anaconda3\lib\site-packages\pandas\core\tools\datetimes.py:1047: UserWarnin
g: Parsing '24/04/2019' in DD/MM/YYYY format. Provide format or specify infer_datetime_f
ormat=True for consistent parsing.
  cache_array = _maybe_cache(arg, format, cache, convert_listlike)
C:\Users\HP\anaconda3\lib\site-packages\pandas\core\tools\datetimes.py:1047: UserWarnin
g: Parsing '15/04/2019' in DD/MM/YYYY format. Provide format or specify infer_datetime_f
ormat=True for consistent parsing.
  cache_array = _maybe_cache(arg, format, cache, convert_listlike)
C:\Users\HP\anaconda3\lib\site-packages\pandas\core\tools\datetimes.py:1047: UserWarnin
g: Parsing '21/03/2019' in DD/MM/YYYY format. Provide format or specify infer_datetime_f
ormat=True for consistent parsing.
  cache_array = _maybe_cache(arg, format, cache, convert_listlike)
C:\Users\HP\anaconda3\lib\site-packages\pandas\core\tools\datetimes.py:1047: UserWarnin
g: Parsing '15/05/2019' in DD/MM/YYYY format. Provide format or specify infer_datetime_f
ormat=True for consistent parsing.
  cache_array = _maybe_cache(arg, format, cache, convert_listlike)
C:\Users\HP\anaconda3\lib\site-packages\pandas\core\tools\datetimes.py:1047: UserWarnin
g: Parsing '18/06/2019' in DD/MM/YYYY format. Provide format or specify infer_datetime_f
ormat=True for consistent parsing.
  cache_array = _maybe_cache(arg, format, cache, convert_listlike)
C:\Users\HP\anaconda3\lib\site-packages\pandas\core\tools\datetimes.py:1047: UserWarnin
g: Parsing '15/06/2019' in DD/MM/YYYY format. Provide format or specify infer_datetime_f
ormat=True for consistent parsing.
  cache_array = _maybe_cache(arg, format, cache, convert_listlike)
C:\Users\HP\anaconda3\lib\site-packages\pandas\core\tools\datetimes.py:1047: UserWarnin
g: Parsing '18/05/2019' in DD/MM/YYYY format. Provide format or specify infer_datetime_f
ormat=True for consistent parsing.
  cache_array = _maybe_cache(arg, format, cache, convert_listlike)
C:\Users\HP\anaconda3\lib\site-packages\pandas\core\tools\datetimes.py:1047: UserWarnin
g: Parsing '27/06/2019' in DD/MM/YYYY format. Provide format or specify infer_datetime_f
ormat=True for consistent parsing.
  cache_array = _maybe_cache(arg, format, cache, convert_listlike)
C:\Users\HP\anaconda3\lib\site-packages\pandas\core\tools\datetimes.py:1047: UserWarnin
g: Parsing '21/05/2019' in DD/MM/YYYY format. Provide format or specify infer_datetime_f
ormat=True for consistent parsing.
  cache_array = _maybe_cache(arg, format, cache, convert_listlike)
C:\Users\HP\anaconda3\lib\site-packages\pandas\core\tools\datetimes.py:1047: UserWarnin
g: Parsing '15/03/2019' in DD/MM/YYYY format. Provide format or specify infer_datetime_f
ormat=True for consistent parsing.
  cache_array = _maybe_cache(arg, format, cache, convert_listlike)
C:\Users\HP\anaconda3\lib\site-packages\pandas\core\tools\datetimes.py:1047: UserWarnin
g: Parsing '24/05/2019' in DD/MM/YYYY format. Provide format or specify infer_datetime_f
ormat=True for consistent parsing.
  cache_array = _maybe_cache(arg, format, cache, convert_listlike)
C:\Users\HP\anaconda3\lib\site-packages\pandas\core\tools\datetimes.py:1047: UserWarnin
g: Parsing '21/04/2019' in DD/MM/YYYY format. Provide format or specify infer_datetime_f
ormat=True for consistent parsing.
  cache_array = _maybe_cache(arg, format, cache, convert_listlike)
```
Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

```
C:\Users\HP\anaconda3\lib\site-packages\pandas\core\tools\datetimes.py:1047: UserWarnin
g: Parsing '21/06/2019' in DD/MM/YYYY format. Provide format or specify infer_datetime_f
ormat=True for consistent parsing.
  cache_array = _maybe_cache(arg, format, cache, convert_listlike)
C:\Users\HP\anaconda3\lib\site-packages\pandas\core\tools\datetimes.py:1047: UserWarnin
g: Parsing '27/03/2019' in DD/MM/YYYY format. Provide format or specify infer_datetime_f
ormat=True for consistent parsing.
  cache_array = _maybe_cache(arg, format, cache, convert_listlike)
C:\Users\HP\anaconda3\lib\site-packages\pandas\core\tools\datetimes.py:1047: UserWarnin
g: Parsing '18/03/2019' in DD/MM/YYYY format. Provide format or specify infer_datetime_f
ormat=True for consistent parsing.
  cache_array = _maybe_cache(arg, format, cache, convert_listlike)
C:\Users\HP\anaconda3\lib\site-packages\pandas\core\tools\datetimes.py:1047: UserWarnin
g: Parsing '27/04/2019' in DD/MM/YYYY format. Provide format or specify infer_datetime_f
ormat=True for consistent parsing.
  cache_array = _maybe_cache(arg, format, cache, convert_listlike)
```

In [17]:
```
d.dtypes
```

Out[17]:
```
Airline                    object
Date_of_Journey    datetime64[ns]
Source                     object
Destination                object
Route                      object
Dep_Time           datetime64[ns]
Arrival_Time       datetime64[ns]
Duration                   object
Total_Stops                object
Additional_Info            object
Price                       int64
dtype: object
```

In [18]:
```
d['Date_of_Journey'].min()
```

Out[18]:
```
Timestamp('2019-01-03 00:00:00')
```

In [19]:
```
d['Date_of_Journey'].max()
```

Out[19]:
```
Timestamp('2019-12-06 00:00:00')
```

In [20]:
```
d['journey_day']=d['Date_of_Journey'].dt.day
```

In [21]:
```
d['journey_month']=d['Date_of_Journey'].dt.month
```

In [22]:
```
d['journey_year']=d['Date_of_Journey'].dt.year
```

In [23]:
```
d.head(2)
```

Out[23]:

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Duration	Total_Stops	Additi
0	IndiGo	2019-03-24	Banglore	New Delhi	BLR → DEL	2023-05-26 22:20:00	2023-03-22 01:10:00	2h 50m	non-stop	
1	Air India	2019-01-05	Kolkata	Banglore	CCU → IXR → BBI → BLR	2023-05-26 05:50:00	2023-05-26 13:15:00	7h 25m	2 stops	

In [24]:
```
d.drop('Date_of_Journey',axis=1,inplace=True)
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

```
In [25]: d.head(2)
```

Out[25]:

	Airline	Source	Destination	Route	Dep_Time	Arrival_Time	Duration	Total_Stops	Additional_Info	Price	j
0	IndiGo	Banglore	New Delhi	BLR → DEL	2023-05-26 22:20:00	2023-03-22 01:10:00	2h 50m	non-stop	No info	3897	
1	Air India	Kolkata	Banglore	CCU → IXR → BBI → BLR	2023-05-26 05:50:00	2023-05-26 13:15:00	7h 25m	2 stops	No info	7662	

```
In [26]: def extract_hour_min(df,col):
             df[col+'_hour']=df[col].dt.hour
             df[col+'_minute']=df[col].dt.minute
             df.drop(col,axis=1,inplace=True)
             return df.head(2)
```

```
In [28]: extract_hour_min(d,'Dep_Time')
```

Out[28]:

	Airline	Source	Destination	Route	Arrival_Time	Duration	Total_Stops	Additional_Info	Price	journey_day
0	IndiGo	Banglore	New Delhi	BLR → DEL	2023-03-22 01:10:00	2h 50m	non-stop	No info	3897	24
1	Air India	Kolkata	Banglore	CCU → IXR → BBI → BLR	2023-05-26 13:15:00	7h 25m	2 stops	No info	7662	5

```
In [29]: extract_hour_min(d,'Arrival_Time')
```

Out[29]:

	Airline	Source	Destination	Route	Duration	Total_Stops	Additional_Info	Price	journey_day	journey_mor
0	IndiGo	Banglore	New Delhi	BLR → DEL	2h 50m	non-stop	No info	3897	24	
1	Air India	Kolkata	Banglore	CCU → IXR → BBI → BLR	7h 25m	2 stops	No info	7662	5	

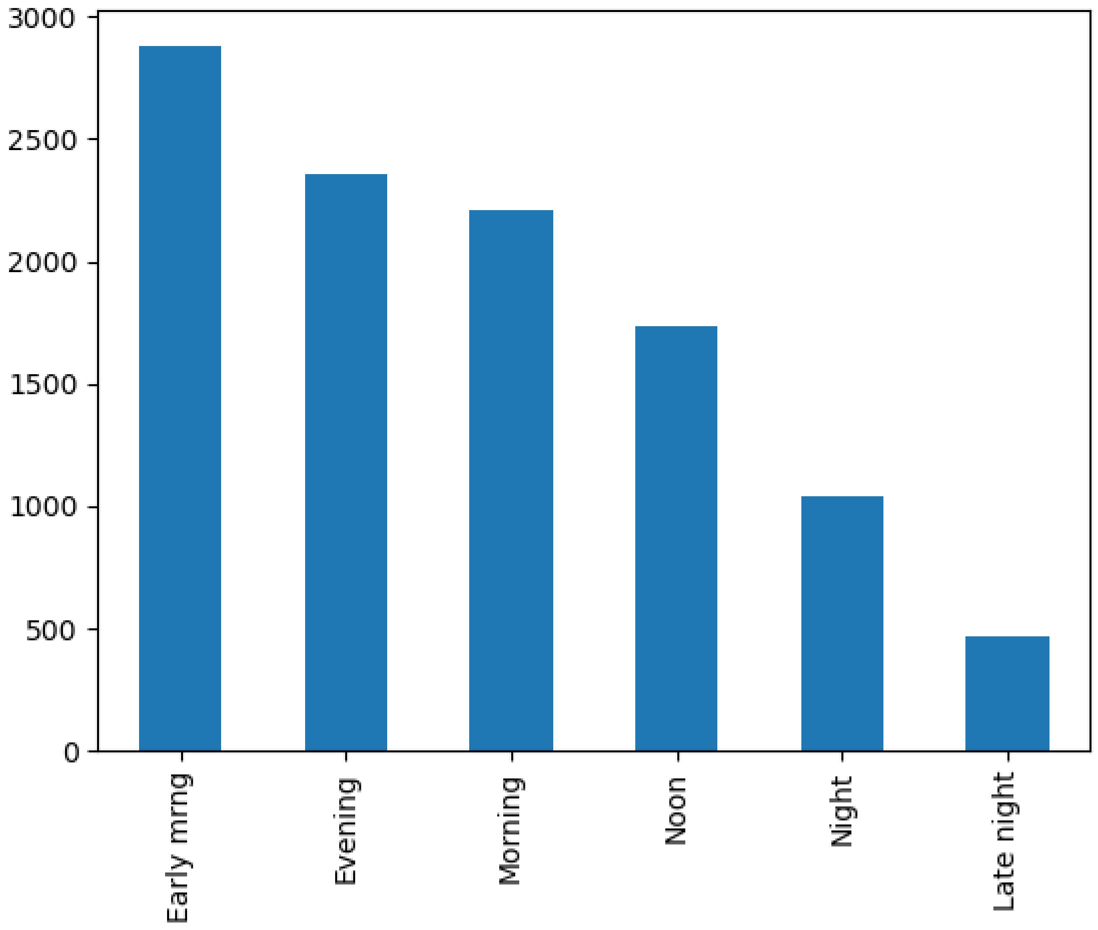
```
In [30]: def flight_dep_time(x):
             '''
             This function takes the flight Departure time
             and convert into appropriate format.
             '''
             if ( x> 4) and (x<=8 ):
                 return 'Early mrng'

             elif ( x>8 ) and (x<=12 ):
                 return 'Morning'
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

```
        return 'Noon'

    elif ( x>16 ) and (x<=20 ):
        return 'Evening'

    elif ( x>20 ) and (x<=24 ):
        return 'Night'
    else:
        return 'Late night'
```

In [32]:
```
d['Dep_Time_hour'].apply(flight_dep_time).value_counts().plot(kind='bar')
```

Out[32]:
```
<AxesSubplot:>
```

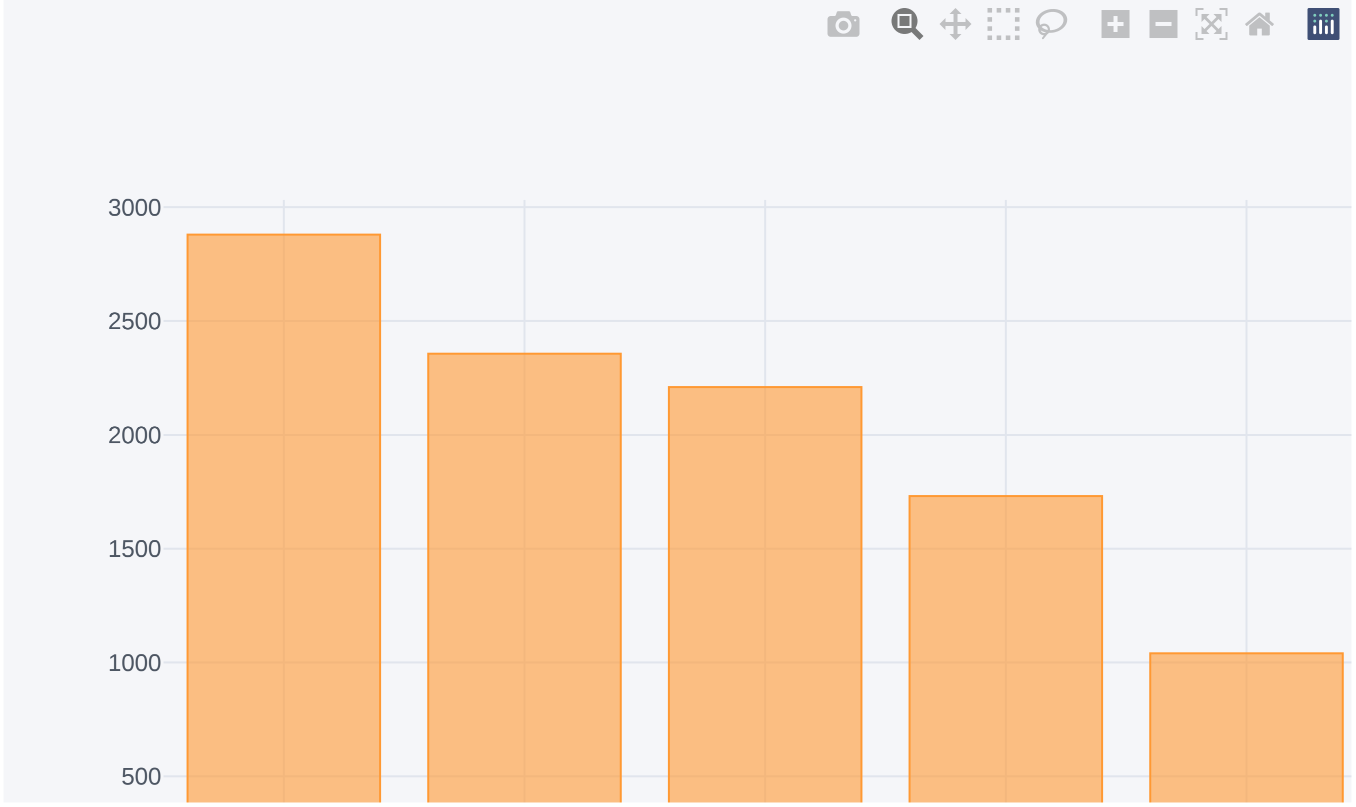


In [33]:
```
## Lets use Plotly interactive plots directly with Pandas dataframes, but First u need b

import plotly
import cufflinks as cf
from cufflinks.offline import go_offline
from plotly.offline import download_plotlyjs,init_notebook_mode,plot,iplot
```

In [34]:
```
cf.go_offline()
```

In [35]:
```
d['Dep_Time_hour'].apply(flight_dep_time).value_counts().iplot(kind='bar')
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js



```
In [36]: def preprocess_duration(x):
             if 'h' not in x:
                 x='0h '+x
             elif 'm' not in x:
                 x=x+' 0m'
             return x
```

```
In [37]: d['Duration']=d['Duration'].apply(preprocess_duration)
```

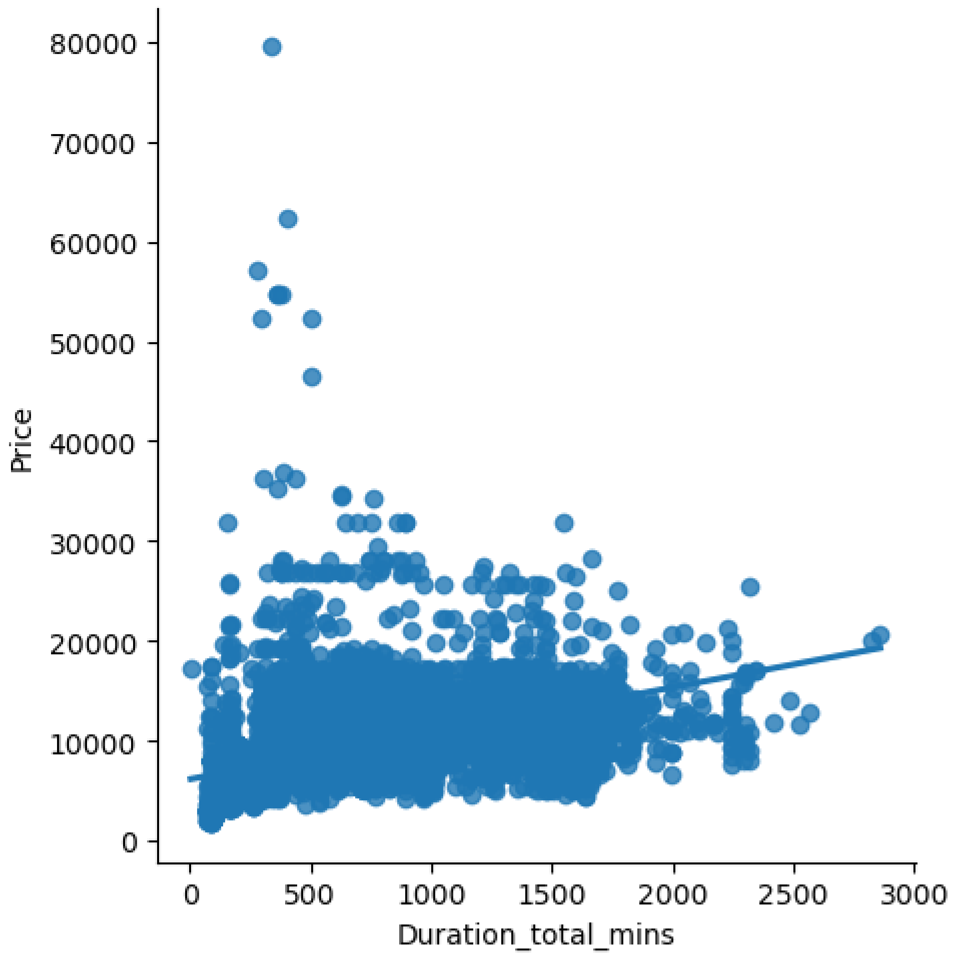
```
In [38]: d['Duration']
```

```
Out[38]: 0         2h 50m
         1         7h 25m
         2         19h 0m
         3         5h 25m
         4         4h 45m
                    ...
         10678     2h 30m
         10679     2h 35m
         10680      3h 0m
         10681     2h 40m
         10682     8h 20m
         Name: Duration, Length: 10682, dtype: object
```

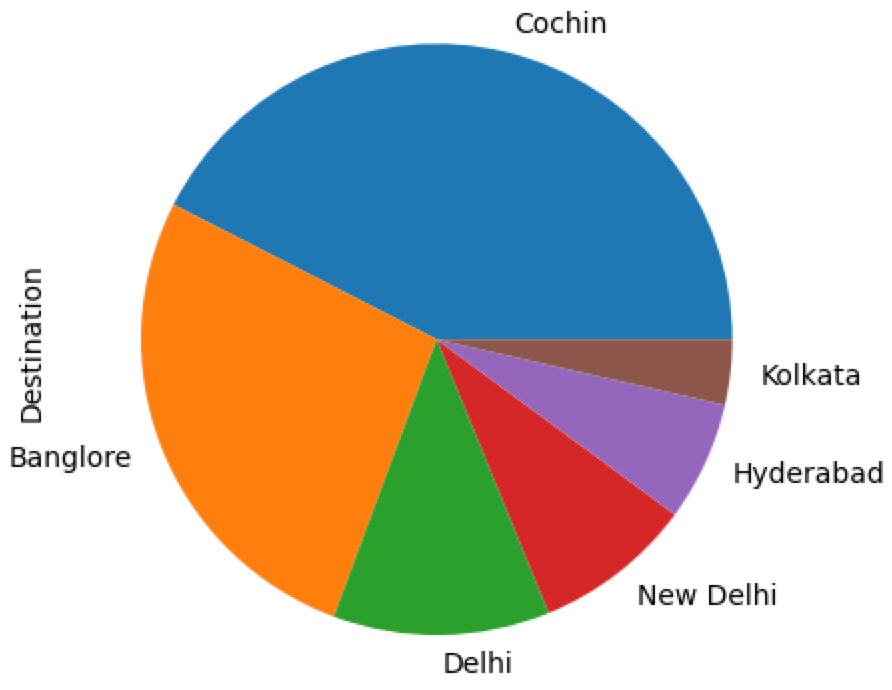
```
In [39]: d['Duration'][0].split(' ')[0]
```

```
Out[39]: '2h'
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

In [41]:
```
int(d['Duration'][0].split(' ')[0][0:-1])
```
Out[41]:
```
2
```
In [42]:
```
int(d['Duration'][0].split(' ')[1][0:-1])
```
Out[42]:
```
50
```
In [43]:
```
d['Duration_hours']=d['Duration'].apply(lambda x:int(x.split(' ')[0][0:-1]))
```
In [44]:
```
d['Duration_mins']=d['Duration'].apply(lambda x:int(x.split(' ')[1][0:-1]))
```
In [45]:
```
eval('2*60+50*1')
```
Out[45]:
```
170
```
In [46]:
```
d['Duration_total_mins']=d['Duration'].str.replace('h','*60').str.replace(' ','+').str.r
```
In [48]:
```
sns.lmplot(x='Duration_total_mins',y='Price',data=d)
```
Out[48]:
```
<seaborn.axisgrid.FacetGrid at 0x2dda3692e50>
```


In [49]:
```
d['Destination'].unique()
```
Out[49]:
```
array(['New Delhi', 'Banglore', 'Cochin', 'Kolkata', 'Delhi', 'Hyderabad'],
      dtype=object)
```
In [50]:
```
d['Destination'].value_counts().plot(kind='pie')
```
Out[50]:
```
<AxesSubplot:ylabel='Destination'>
```
Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js



```
In [ ]: '''
        Inference->>
        Final destination of majority of flights is Cochin. There are two values for Delhi desti

        '''
```

```
In [51]: d['Route']
```

```
Out[51]: 0                     BLR → DEL
         1         CCU → IXR → BBI → BLR
         2         DEL → LKO → BOM → COK
         3               CCU → NAG → BLR
         4               BLR → NAG → DEL
                              ...
         10678                 CCU → BLR
         10679                 CCU → BLR
         10680                 BLR → DEL
         10681                 BLR → DEL
         10682     DEL → GOI → BOM → COK
         Name: Route, Length: 10682, dtype: object
```

```
In [52]: d[d['Airline']=='Jet Airways'].groupby('Route').size().sort_values(ascending=False)
```

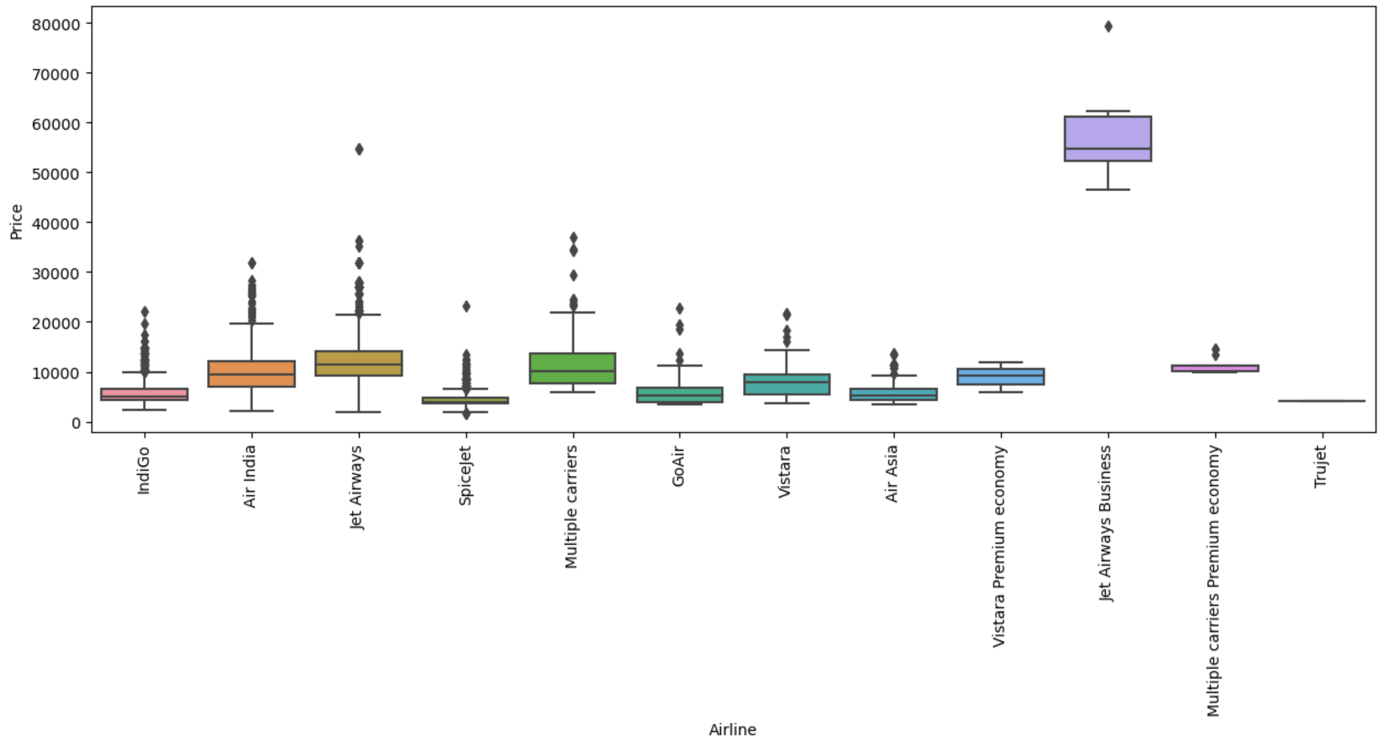
Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

```
Out[52]: Route
         CCU → BOM → BLR          930
         DEL → BOM → COK          875
         BLR → BOM → DEL          385
         BLR → DEL                382
         CCU → DEL → BLR          300
         BOM → HYD                207
         DEL → JAI → BOM → COK    207
         DEL → AMD → BOM → COK    141
         DEL → IDR → BOM → COK     86
         DEL → NAG → BOM → COK     61
         DEL → ATQ → BOM → COK     38
         DEL → COK                 34
         DEL → BHO → BOM → COK     29
         DEL → BDQ → BOM → COK     28
         DEL → LKO → BOM → COK     25
         DEL → JDH → BOM → COK     23
         CCU → GAU → BLR           22
         DEL → MAA → BOM → COK     16
         DEL → IXC → BOM → COK     13
         BLR → MAA → DEL           10
         BLR → BDQ → DEL            8
         DEL → UDR → BOM → COK      7
         BOM → DEL → HYD            5
         CCU → BOM → PNQ → BLR      4
         BLR → BOM → JDH → DEL      3
         DEL → DED → BOM → COK      2
         BOM → BDQ → DEL → HYD      2
         DEL → CCU → BOM → COK      1
         BOM → VNS → DEL → HYD      1
         BOM → UDR → DEL → HYD      1
         BOM → JDH → DEL → HYD      1
         BOM → IDR → DEL → HYD      1
         BOM → DED → DEL → HYD      1
         dtype: int64
```

```
In [53]: plt.figure(figsize=(15,5))
         sns.boxplot(y='Price',x='Airline',data=d)
         plt.xticks(rotation='vertical')
```

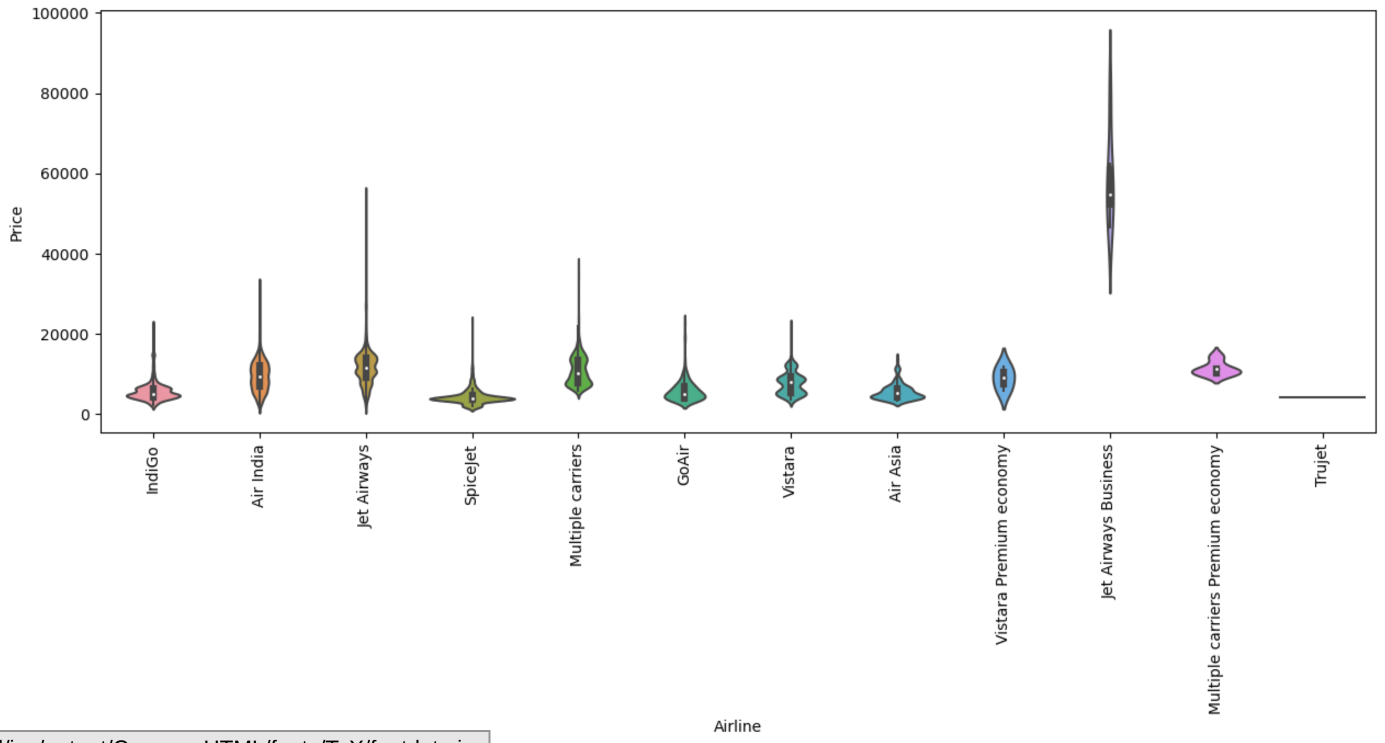
```
Out[53]: (array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11]),
          [Text(0, 0, 'IndiGo'),
           Text(1, 0, 'Air India'),
           Text(2, 0, 'Jet Airways'),
           Text(3, 0, 'SpiceJet'),
           Text(4, 0, 'Multiple carriers'),
           Text(5, 0, 'GoAir'),
           Text(6, 0, 'Vistara'),
           Text(7, 0, 'Air Asia'),
           Text(8, 0, 'Vistara Premium economy'),
           Text(9, 0, 'Jet Airways Business'),
           Text(10, 0, 'Multiple carriers Premium economy'),
           Text(11, 0, 'Trujet')])
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js



In [55]:
```
plt.figure(figsize=(15,5))
sns.violinplot(y='Price',x='Airline',data=d)
plt.xticks(rotation='vertical')
```

Out[55]:
```
(array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11]),
 [Text(0, 0, 'IndiGo'),
  Text(1, 0, 'Air India'),
  Text(2, 0, 'Jet Airways'),
  Text(3, 0, 'SpiceJet'),
  Text(4, 0, 'Multiple carriers'),
  Text(5, 0, 'GoAir'),
  Text(6, 0, 'Vistara'),
  Text(7, 0, 'Air Asia'),
  Text(8, 0, 'Vistara Premium economy'),
  Text(9, 0, 'Jet Airways Business'),
  Text(10, 0, 'Multiple carriers Premium economy'),
  Text(11, 0, 'Trujet')])
```



Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

```
In [56]: np.round(d['Additional_Info'].value_counts()/len(d)*100,2)

Out[56]: No info                         78.11
         In-flight meal not included     18.55
         No check-in baggage included     3.00
         1 Long layover                   0.18
         Change airports                  0.07
         Business class                   0.04
         No Info                          0.03
         1 Short layover                  0.01
         Red-eye flight                   0.01
         2 Long layover                   0.01
         Name: Additional_Info, dtype: float64
```

```
In [57]: d.drop(columns=['Additional_Info','Route','Duration_total_mins','journey_year'],axis=1,i
```

```
In [58]: d.columns

Out[58]: Index(['Airline', 'Source', 'Destination', 'Duration', 'Total_Stops', 'Price',
                'journey_day', 'journey_month', 'Dep_Time_hour', 'Dep_Time_minute',
                'Arrival_Time_hour', 'Arrival_Time_minute', 'Duration_hours',
                'Duration_mins'],
               dtype='object')
```

```
In [59]: d.head(4)
```

Out[59]:

	Airline	Source	Destination	Duration	Total_Stops	Price	journey_day	journey_month	Dep_Time_hour	D
0	IndiGo	Banglore	New Delhi	2h 50m	non-stop	3897	24	3	22	
1	Air India	Kolkata	Banglore	7h 25m	2 stops	7662	5	1	5	
2	Jet Airways	Delhi	Cochin	19h 0m	2 stops	13882	6	9	9	
3	IndiGo	Kolkata	Banglore	5h 25m	1 stop	6218	5	12	18	

```
In [60]: cat_col=[col for col in d.columns if d[col].dtype=='object']
```

```
In [61]: num_col=[col for col in d.columns if d[col].dtype!='object']
```

```
In [62]: cat_col

Out[62]: ['Airline', 'Source', 'Destination', 'Duration', 'Total_Stops']
```

```
In [ ]: ## Handling Categorical Data
            We are using 2 basic Encoding Techniques to convert Categorical data into some numer
            if data belongs to Nominal data (ie data is not in any order) -->> OneHotEncoder is
            if data belongs to Ordinal data (ie data is in order ) -->>       LabelEncoder is us
```

```
In [63]: d['Source'].unique()

Out[63]: array(['Banglore', 'Kolkata', 'Delhi', 'Chennai', 'Mumbai'], dtype=object)
```

```
In [64]: d['Source']
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

```
Out[64]: 0         Banglore
         1          Kolkata
         2            Delhi
         3          Kolkata
         4         Banglore
                     ...
         10678      Kolkata
         10679      Kolkata
         10680     Banglore
         10681     Banglore
         10682        Delhi
         Name: Source, Length: 10682, dtype: object
```

```
In [65]: d['Source'].apply(lambda x: 1 if x=='Banglore' else 0)
```

```
Out[65]: 0        1
         1        0
         2        0
         3        0
         4        1
                 ..
         10678    0
         10679    0
         10680    1
         10681    1
         10682    0
         Name: Source, Length: 10682, dtype: int64
```

```
In [66]: for category in d['Source'].unique():
             d['Source_'+category]=d['Source'].apply(lambda x: 1 if x==category else 0)
```

```
In [67]: d.head(3)
```

Out[67]:

	Airline	Source	Destination	Duration	Total_Stops	Price	journey_day	journey_month	Dep_Time_hour	D
0	IndiGo	Banglore	New Delhi	2h 50m	non-stop	3897	24	3	22	
1	Air India	Kolkata	Banglore	7h 25m	2 stops	7662	5	1	5	
2	Jet Airways	Delhi	Cochin	19h 0m	2 stops	13882	6	9	9	

```
In [68]: airlines=d.groupby(['Airline'])['Price'].mean().sort_values().index
```

```
In [69]: airlines
```

```
Out[69]: Index(['Trujet', 'SpiceJet', 'Air Asia', 'IndiGo', 'GoAir', 'Vistara',
                'Vistara Premium economy', 'Air India', 'Multiple carriers',
                'Multiple carriers Premium economy', 'Jet Airways',
                'Jet Airways Business'],
               dtype='object', name='Airline')
```

```
In [70]: dict1={key:index for index,key in enumerate(airlines,0)}
```

```
In [71]: dict1
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

```
Out[71]: {'Trujet': 0,
          'SpiceJet': 1,
          'Air Asia': 2,
          'IndiGo': 3,
          'GoAir': 4,
          'Vistara': 5,
          'Vistara Premium economy': 6,
          'Air India': 7,
          'Multiple carriers': 8,
          'Multiple carriers Premium economy': 9,
          'Jet Airways': 10,
          'Jet Airways Business': 11}
```

```
In [72]: d['Airline']=d['Airline'].map(dict1)
```

```
In [73]: d['Airline']
```

```
Out[73]: 0         3
         1         7
         2        10
         3         3
         4         3
                  ..
         10678     2
         10679     7
         10680    10
         10681     5
         10682     7
         Name: Airline, Length: 10682, dtype: int64
```

```
In [74]: d.head(2)
```

Out[74]:

	Airline	Source	Destination	Duration	Total_Stops	Price	journey_day	journey_month	Dep_Time_hour	De
0	3	Banglore	New Delhi	2h 50m	non-stop	3897	24	3	22	
1	7	Kolkata	Banglore	7h 25m	2 stops	7662	5	1	5	

```
In [75]: d['Destination'].unique()
```

```
Out[75]: array(['New Delhi', 'Banglore', 'Cochin', 'Kolkata', 'Delhi', 'Hyderabad'],
               dtype=object)
```

```
In [76]: d['Destination'].replace('New Delhi','Delhi',inplace=True)
```

```
In [77]: d['Destination'].unique()
```

```
Out[77]: array(['Delhi', 'Banglore', 'Cochin', 'Kolkata', 'Hyderabad'],
               dtype=object)
```

```
In [78]: dest=d.groupby(['Destination'])['Price'].mean().sort_values().index
```

```
In [79]: dest
```

```
Out[79]: Index(['Kolkata', 'Hyderabad', 'Delhi', 'Banglore', 'Cochin'], dtype='object', name='Des
         tination')
```

```
In [81]: dict2={key:index for index,key in enumerate(dest,0)}
```

```
In [82]: dict2
```

```
Out[82]: {'Kolkata': 0, 'Hyderabad': 1, 'Delhi': 2, 'Banglore': 3, 'Cochin': 4}
```

In [83]:
```
d['Destination']=d['Destination'].map(dict2)
```

In [84]:
```
d['Destination']
```

Out[84]:
```
0        2
1        3
2        4
3        3
4        2
        ..
10678    3
10679    3
10680    2
10681    2
10682    4
Name: Destination, Length: 10682, dtype: int64
```

In [86]:
```
d.head(2)
```

Out[86]:

	Airline	Source	Destination	Duration	Total_Stops	Price	journey_day	journey_month	Dep_Time_hour	De
0	3	Banglore	2	2h 50m	non-stop	3897	24	3	22	
1	7	Kolkata	3	7h 25m	2 stops	7662	5	1	5	

In [87]:
```
d['Total_Stops'].unique()
```

Out[87]:
```
array(['non-stop', '2 stops', '1 stop', '3 stops', '4 stops'],
      dtype=object)
```

In [88]:
```
stops={'non-stop':0, '2 stops':2, '1 stop':1, '3 stops':3, '4 stops':4}
```

In [90]:
```
d['Total_Stops']=d['Total_Stops'].map(stops)
```

In [91]:
```
d['Total_Stops']
```

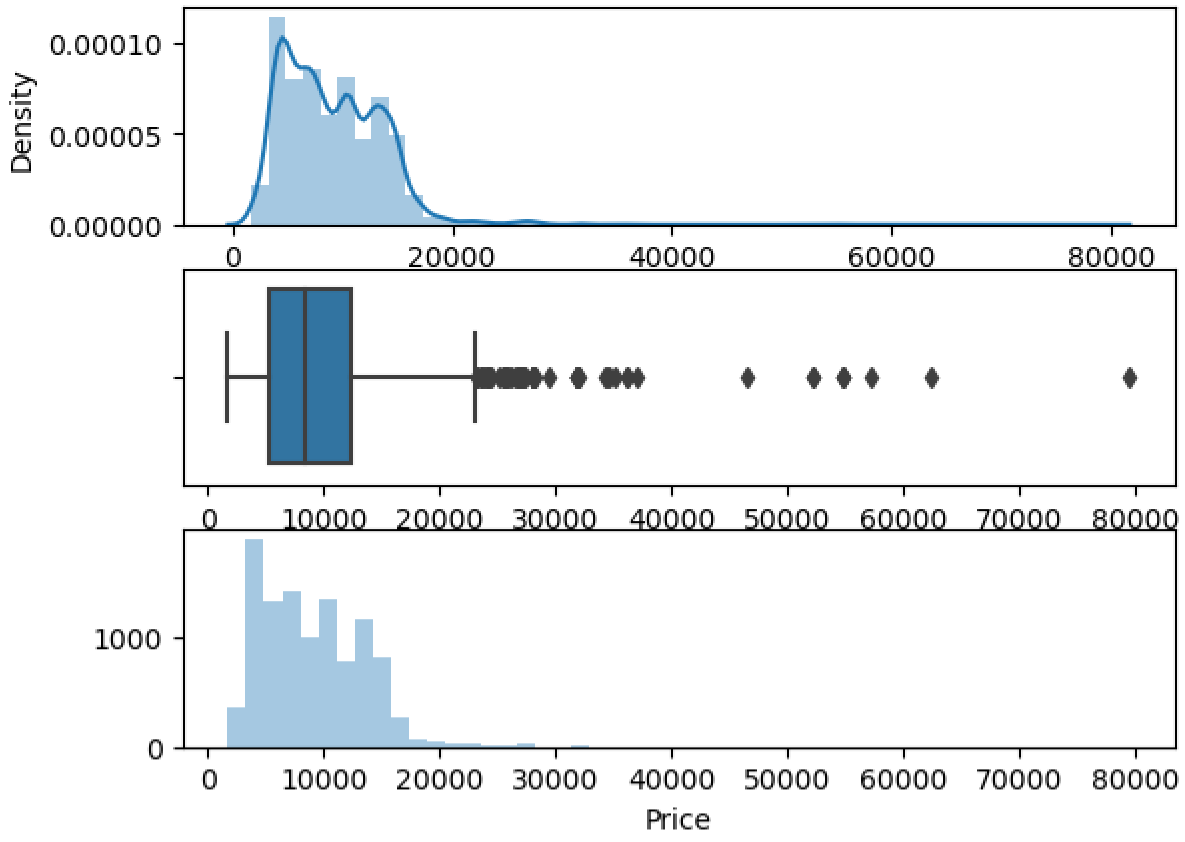
Out[91]:
```
0        0
1        2
2        2
3        1
4        1
        ..
10678    0
10679    0
10680    0
10681    0
10682    2
Name: Total_Stops, Length: 10682, dtype: int64
```

In [92]:
```
def plot(df,col):
    fig,(ax1,ax2,ax3)=plt.subplots(3,1)
    sns.distplot(df[col],ax=ax1)
    sns.boxplot(df[col],ax=ax2)
    sns.distplot(df[col],ax=ax3,kde=False)
```

In [93]:
```
plot(d,'Price')
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

```
C:\Users\HP\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning:

`distplot` is a deprecated function and will be removed in a future version. Please adap
t your code to use either `displot` (a figure-level function with similar flexibility) o
r `histplot` (an axes-level function for histograms).

C:\Users\HP\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning:

Pass the following variable as a keyword arg: x. From version 0.12, the only valid posit
ional argument will be `data`, and passing other arguments without an explicit keyword w
ill result in an error or misinterpretation.

C:\Users\HP\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning:

`distplot` is a deprecated function and will be removed in a future version. Please adap
t your code to use either `displot` (a figure-level function with similar flexibility) o
r `histplot` (an axes-level function for histograms).
```

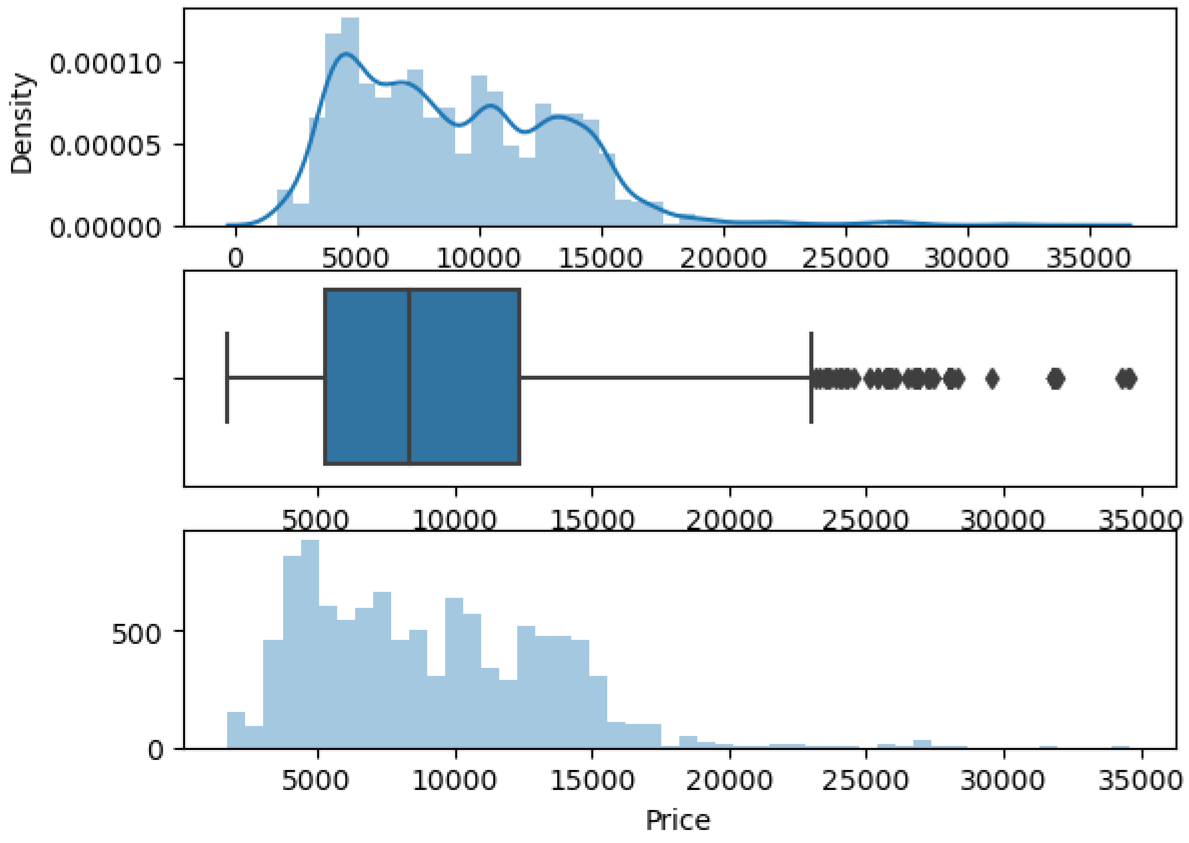


In [94]:
```python
d['Price']=np.where(d['Price']>=35000,d['Price'].median(),d['Price'])
```

In [95]:
```python
plot(d,'Price')
```

```
C:\Users\HP\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning:

Pass the following variable as a keyword arg: x. From version 0.12, the only valid posit
ional argument will be `data`, and passing other arguments without an explicit keyword w
ill result in an error or misinterpretation.

C:\Users\HP\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning:

`distplot` is a deprecated function and will be removed in a future version. Please adap
t your code to use either `displot` (a figure-level function with similar flexibility) o
r `histplot` (an axes-level function for histograms).
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js



```
In [96]: d.head(2)
```

Out[96]:

	Airline	Source	Destination	Duration	Total_Stops	Price	journey_day	journey_month	Dep_Time_hour	D
0	3	Banglore	2	2h 50m	0	3897.0	24	3	22	
1	7	Kolkata	3	7h 25m	2	7662.0	5	1	5	

```
In [97]: d.drop(columns=['Source','Duration'],axis=1,inplace=True)
```

```
In [98]: d.head(2)
```

Out[98]:

	Airline	Destination	Total_Stops	Price	journey_day	journey_month	Dep_Time_hour	Dep_Time_minute	Ar
0	3	2	0	3897.0	24	3	22	20	
1	7	3	2	7662.0	5	1	5	50	

```
In [99]: d.dtypes
```

```
Out[99]:  Airline                      int64
          Destination                  int64
          Total_Stops                  int64
          Price                      float64
          journey_day                  int64
          journey_month                int64
          Dep_Time_hour                int64
          Dep_Time_minute              int64
          Arrival_Time_hour            int64
          Arrival_Time_minute          int64
          Duration_hours               int64
          Duration_mins                int64
          Source_Banglore              int64
          Source_Kolkata               int64
          Source_Delhi                 int64
          Source_Chennai               int64
          Source_Mumbai                int64
          dtype: object
```

```
In [100…  from sklearn.feature_selection import mutual_info_regression
```

```
In [102…  X=d.drop(['Price'],axis=1)
```

```
In [103…  y=d['Price']
```

```
In [104…  X.dtypes
```

```
Out[104]:  Airline                   int64
           Destination               int64
           Total_Stops               int64
           journey_day               int64
           journey_month             int64
           Dep_Time_hour             int64
           Dep_Time_minute           int64
           Arrival_Time_hour         int64
           Arrival_Time_minute       int64
           Duration_hours            int64
           Duration_mins             int64
           Source_Banglore           int64
           Source_Kolkata            int64
           Source_Delhi              int64
           Source_Chennai            int64
           Source_Mumbai             int64
           dtype: object
```

```
In [105…  mutual_info_regression(X,y)
```

```
Out[105]:  array([0.97905922, 1.00278875, 0.79234432, 0.20019407, 0.24215526,
                  0.33400973, 0.25991151, 0.39654958, 0.34848813, 0.46649106,
                  0.34770539, 0.39249321, 0.44713114, 0.51939751, 0.13817632,
                  0.20435607])
```

```
In [106…  imp=pd.DataFrame(mutual_info_regression(X,y),index=X.columns)
          imp.columns=['importance']
```

```
In [107…  imp.sort_values(by='importance',ascending=False)
```

Out[107]:

	importance
Destination	1.009520
Airline	0.974462
Total_Stops	0.787646
Source_Delhi	0.520584
Duration_hours	0.462498
Source_Kolkata	0.458599
Arrival_Time_hour	0.403324
Source_Banglore	0.388890
Arrival_Time_minute	0.353946
Duration_mins	0.348627
Dep_Time_hour	0.340820
Dep_Time_minute	0.260207
journey_month	0.237165
journey_day	0.195206
Source_Mumbai	0.194308
Source_Chennai	0.140033

In [108…
```
from sklearn.model_selection import train_test_split
```

In [109…
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=4
```

In [110…
```
from sklearn.ensemble import RandomForestRegressor
```

In [111…
```
ml_model=RandomForestRegressor()
```

In [112…
```
model=ml_model.fit(X_train,y_train)
```

In [113…
```
y_pred=model.predict(X_test)
```

In [114…
```
y_pred
```

Out[114]:
```
array([16753.76,  6399.25,  8814.25, ...,  3517.64,  6416.24,  6856.26])
```

In [115…
```
y_pred.shape
```

Out[115]:
```
(2671,)
```

In [116…
```
len(X_test)
```

Out[116]:
```
2671
```

In [120…
```
import pickle
```

In [123…
```
file=open(r'C:\Machine learning\Projects/rf_random.pkl','wb')
```

In [124…
```
pickle.dump(model,file)
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

In [125…
```
model=open(r'C:\Machine learning\Projects/rf_random.pkl','rb')
```

In [126…
```
forest=pickle.load(model)
```

In [127…
```
forest.predict(X_test)
```

Out[127]:
```
array([16753.76,  6399.25,  8814.25, ...,  3517.64,  6416.24,  6856.26])
```

In [128…
```
def mape(y_true,y_pred):
    y_true,y_pred=np.array(y_true),np.array(y_pred)

    return np.mean(np.abs((y_true-y_pred)/y_true))*100
```

In [129…
```
mape(y_test,forest.predict(X_test))
```

Out[129]:
```
13.283570657580091
```

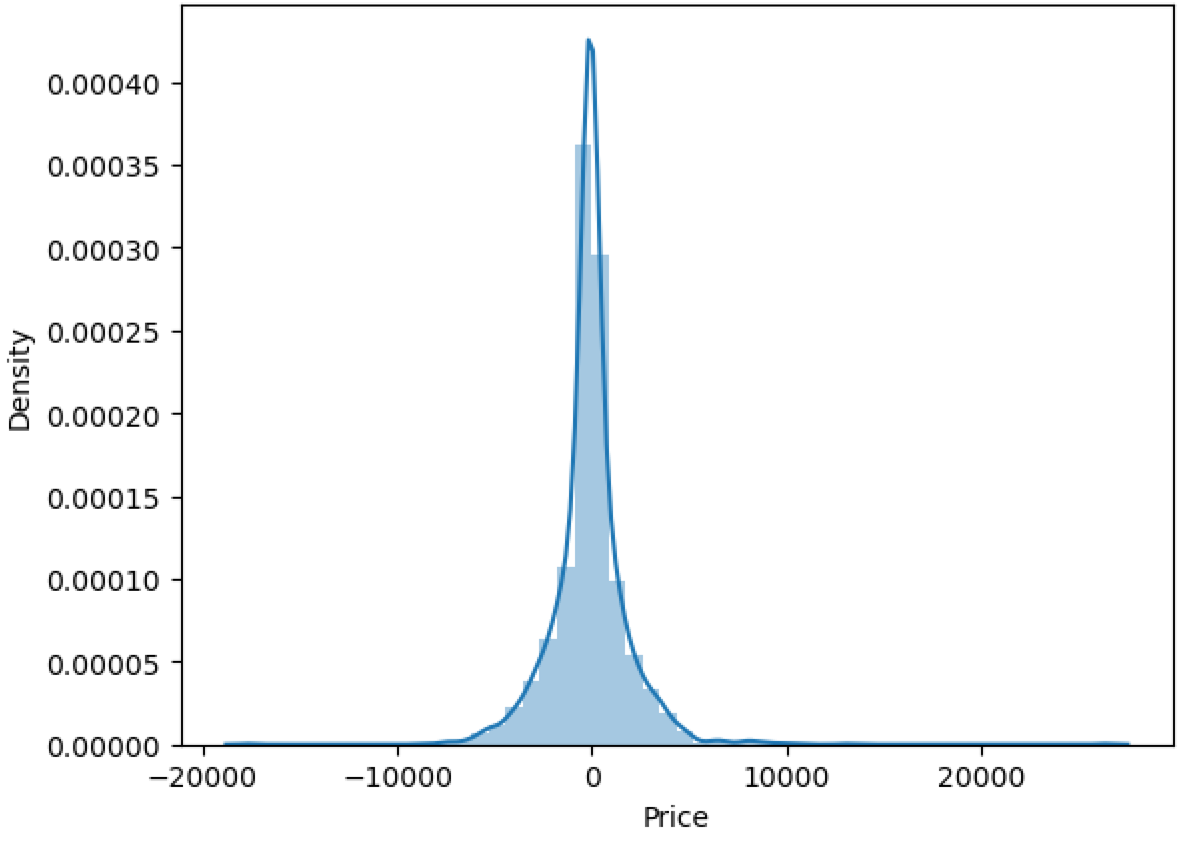
In [130…
```
def predict(ml_model):

    model=ml_model.fit(X_train,y_train)
    print('Training_score: {}'.format(model.score(X_train,y_train)))
    y_prediction=model.predict(X_test)
    print('Predictions are : {}'.format(y_prediction))
    print('\n')

    from sklearn import metrics
    r2_score=metrics.r2_score(y_test,y_prediction)
    print('r2_score: {}'.format(r2_score))
    print('MSE : ', metrics.mean_squared_error(y_test,y_prediction))
    print('MAE : ', metrics.mean_absolute_error(y_test,y_prediction))
    print('RMSE : ', np.sqrt(metrics.mean_squared_error(y_test,y_prediction)))
    print('MAPE : ', mape(y_test,y_prediction))
    sns.distplot(y_test-y_prediction)
```

In [131…
```
predict(RandomForestRegressor())
```

```
Training_score: 0.9520133421401826
Predictions are : [16766.78  6312.32  8885.19 ...  3505.9   6310.88  7056.28]


r2_score: 0.8075058190036632
MSE :  3747399.653561535
MAE :  1181.7847406895073
RMSE :  1935.820150107322
MAPE :  13.249204388003669
```

```
C:\Users\HP\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning:

`distplot` is a deprecated function and will be removed in a future version. Please adap
t your code to use either `displot` (a figure-level function with similar flexibility) o
r `histplot` (an axes-level function for histograms).
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js



In [132…
```
from sklearn.model_selection import RandomizedSearchCV
```

In [133…
```
reg_rf=RandomForestRegressor()
```

In [134…
```
np.linspace(start=1000,stop=1200,num=6)
```

Out[134]:
```
array([1000., 1040., 1080., 1120., 1160., 1200.])
```

In [135…
```
# Number of trees in random forest
n_estimators=[int(x) for x in np.linspace(start=1000,stop=1200,num=6)]

# Number of features to consider at every split
max_features=["auto", "sqrt"]

# Maximum number of levels in tree
max_depth=[int(x) for x in np.linspace(start=5,stop=30,num=4)]

# Minimum number of samples required to split a node
min_samples_split=[5,10,15,100]
```

In [138…
```
# Create the grid or hyper-parameter space
random_grid={
    'n_estimators':n_estimators,
    'max_features':max_features,
    'max_depth':max_depth,
    'min_samples_split':min_samples_split

}
```

In [137…
```
random_grid
```

Out[137]:
```
{'n_estimators': [1000, 1040, 1080, 1120, 1160, 1200],
 'max_features': ['auto', 'sqrt'],
 'max_depth': [5, 13, 21, 30],
```
Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js
```
15, 100]}
```

In [139…

```
rf_Random=RandomizedSearchCV(reg_rf,param_distributions=random_grid,cv=3,verbose=2,n_job
```

In [140…

```
rf_Random.fit(X_train,y_train)
```

```
Fitting 3 folds for each of 10 candidates, totalling 30 fits
```

Out[140]:

```
RandomizedSearchCV(cv=3, estimator=RandomForestRegressor(), n_jobs=-1,
                   param_distributions={'max_depth': [5, 13, 21, 30],
                                        'max_features': ['auto', 'sqrt'],
                                        'min_samples_split': [5, 10, 15, 100],
                                        'n_estimators': [1000, 1040, 1080, 1120,
                                                         1160, 1200]},
                   verbose=2)
```

In [141…

```
rf_Random.best_params_
```

Out[141]:

```
{'n_estimators': 1200,
 'min_samples_split': 10,
 'max_features': 'auto',
 'max_depth': 13}
```

In [142…

```
pred2=rf_Random.predict(X_test)
```

In [143…

```
from sklearn import metrics
metrics.r2_score(y_test,pred2)
```

Out[143]:

```
0.829656210782366
```

In [ ]:

In [ ]:

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js