```
## importing necessary packages !

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

```
train_data = pd.read_excel("Data_Train.xlsx")
```

```
train_data.head(4)
```

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Duration	Total_Stops	Additional_Info	Price
0	IndiGo	24/03/2019	Banglore	New Delhi	BLR → DEL	22:20	01:10 22 Mar	2h 50m	non-stop	No info	3897
1	Air India	1/05/2019	Kolkata	Banglore	CCU → IXR → BBI → BLR	05:50	13:15	7h 25m	2 stops	No info	7662

```
train_data.tail(4)
```

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Duration	Total_Stops	Additional_Info	Price
10679	Air India	27/04/2019	Kolkata	Banglore	CCU → BLR	20:45	23:20	2h 35m	non-stop	No info	4145
10680	Jet Airways	27/04/2019	Banglore	Delhi	BLR → DEL	08:20	11:20	3h	non-stop	No info	7229
10681	Vistara	01/03/2019	Banglore	New Delhi	BLR →	11:30	14:10	2h 40m	non-stop	No info	12648

```
train_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10683 entries, 0 to 10682
Data columns (total 11 columns):
 #   Column           Non-Null Count  Dtype 
---  ------           --------------  ----- 
 0   Airline          10683 non-null  object
 1   Date_of_Journey  10683 non-null  object
 2   Source           10683 non-null  object
 3   Destination      10683 non-null  object
 4   Route            10682 non-null  object
 5   Dep_Time         10683 non-null  object
 6   Arrival_Time     10683 non-null  object
 7   Duration         10683 non-null  object
 8   Total_Stops      10682 non-null  object
 9   Additional_Info  10683 non-null  object
 10  Price            10683 non-null  int64 
dtypes: int64(1), object(10)
memory usage: 918.2+ KB
```

```
## After loading it is important to check null/missing values in a column or a row
## Missing value :  values which occur when no data is recorded for an observation..

train_data.isnull().sum()

## train_data.isnull().sum(axis=0)
## by-default axis is 0 , ie it computes total missing values column-wise !
```

	0
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

```
train_data['Total_Stops'].isnull()
```

	Total_Stops
0	False
1	False
2	False
3	False
4	False
...	...
10678	False
10679	False
10680	False
10681	False
10682	False

10683 rows × 1 columns

dtype: bool

```
##getting all the rows where we have missing value

train_data[train_data['Total_Stops'].isnull()]
```

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Duration	Total_Stops	Additional_Info	Price
9039	Air India	6/05/2019	Delhi	Cochin	NaN	09:45	09:25 07 May	23h 40m	NaN	No info	7480

```
train_data.dropna(inplace=True)

train_data.isnull().sum()
```

	0
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

```
train_data.dtypes
```

	0
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 order to more accurate memory usage , u can leverage memory_usage="deep" in info()
train_data.info(memory_usage="deep")
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 10682 entries, 0 to 10682
Data columns (total 11 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   Airline          10682 non-null  object
 1   Date_of_Journey  10682 non-null  object
 2   Source           10682 non-null  object
 3   Destination      10682 non-null  object
 4   Route            10682 non-null  object
 5   Dep_Time         10682 non-null  object
 6   Arrival_Time     10682 non-null  object
 7   Duration         10682 non-null  object
 8   Total_Stops      10682 non-null  object
 9   Additional_Info  10682 non-null  object
 10  Price            10682 non-null  int64
dtypes: int64(1), object(10)
memory usage: 7.4 MB
```

```
data = train_data.copy()
```

```
data.columns
```

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

```
data.head(2)
```

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

```
data.dtypes
```

	0
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

```
def change_into_Datetime(col):
    data[col] = pd.to_datetime(data[col])
```

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

```
data.columns
```

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

```
for feature in ['Dep_Time', 'Arrival_Time' , 'Date_of_Journey']:
    change_into_Datetime(feature)
```

```
data.dtypes
```

	0
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

```
data["Journey_day"] = data['Date_of_Journey'].dt.day
data["Journey_month"] = data['Date_of_Journey'].dt.month
data["Journey_year"] = data['Date_of_Journey'].dt.year
```

```
data.head(3)
```

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Duration	Total_Stops	Additional_Info	Price	Jou
0	IndiGo	2019-03-24	Banglore	New Delhi	BLR → DEL	2025-02-17 22:20:00	2025-03-22 01:10:00	2h 50m	non-stop	No info	3897	
1	Air India	2019-05-01	Kolkata	Banglore	CCU → IXR → BBI → BLR	2025-02-17 05:50:00	2025-02-17 13:15:00	7h 25m	2 stops	No info	7662	
2	Jet Airways	2019-06-09	Delhi	Cochin	DEL → LKO → BOM → COK	2025-02-17 09:25:00	2025-06-10 04:25:00	19h	2 stops	No info	13882	

```
def extract_hour_min(df , col):
    df[col+"_hour"] = df[col].dt.hour
    df[col+"_minute"] = df[col].dt.minute
    return df.head(3)
```

```
data.columns
```

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

```
# Departure time is when a plane leaves the gate.

extract_hour_min(data , "Dep_Time")
```

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Duration	Total_Stops	Additional_Info	Price	Jou
0	IndiGo	2019-03-24	Banglore	New Delhi	BLR → DEL	2025-02-17 22:20:00	2025-03-22 01:10:00	2h 50m	non-stop	No info	3897	
1	Air India	2019-05-01	Kolkata	Banglore	CCU → IXR → BBI → BLR	2025-02-17 05:50:00	2025-02-17 13:15:00	7h 25m	2 stops	No info	7662	
2	Jet Airways	2019-06-09	Delhi	Cochin	DEL → LKO → BOM → COK	2025-02-17 09:25:00	2025-06-10 04:25:00	19h	2 stops	No info	13882	

```
extract_hour_min(data , "Arrival_Time")
```

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Duration	Total_Stops	Additional_Info	Price	Jou
0	IndiGo	2019-03-24	Banglore	New Delhi	BLR → DEL	2025-02-17 22:20:00	2025-03-22 01:10:00	2h 50m	non-stop	No info	3897	
1	Air India	2019-05-01	Kolkata	Banglore	CCU → IXR → BBI → BLR	2025-02-17 05:50:00	2025-02-17 13:15:00	7h 25m	2 stops	No info	7662	
2	Jet Airways	2019-06-09	Delhi	Cochin	DEL → LKO → BOM → COK	2025-02-17 09:25:00	2025-06-10 04:25:00	19h	2 stops	No info	13882	

```
## we have extracted derived attributes from ['Arrival_Time' , "Dep_Time"] , so lets drop both these features ..
cols_to_drop = ['Arrival_Time' , "Dep_Time"]

data.drop(cols_to_drop , axis=1 , inplace=True )
```

```
data.head(3)
```

	Airline	Date_of_Journey	Source	Destination	Route	Duration	Total_Stops	Additional_Info	Price	Journey_day	Journey_month
0	IndiGo	2019-03-24	Banglore	New Delhi	BLR → DEL	2h 50m	non-stop	No info	3897	24	3
1	Air India	2019-05-01	Kolkata	Banglore	CCU → IXR → BBI → BLR	7h 25m	2 stops	No info	7662	1	5
2	Jet Airways	2019-06-09	Delhi	Cochin	DEL → LKO → BOM → COK	19h	2 stops	No info	13882	9	6

```
data.shape
```

```
(10682, 16)
```

```
data.columns
```

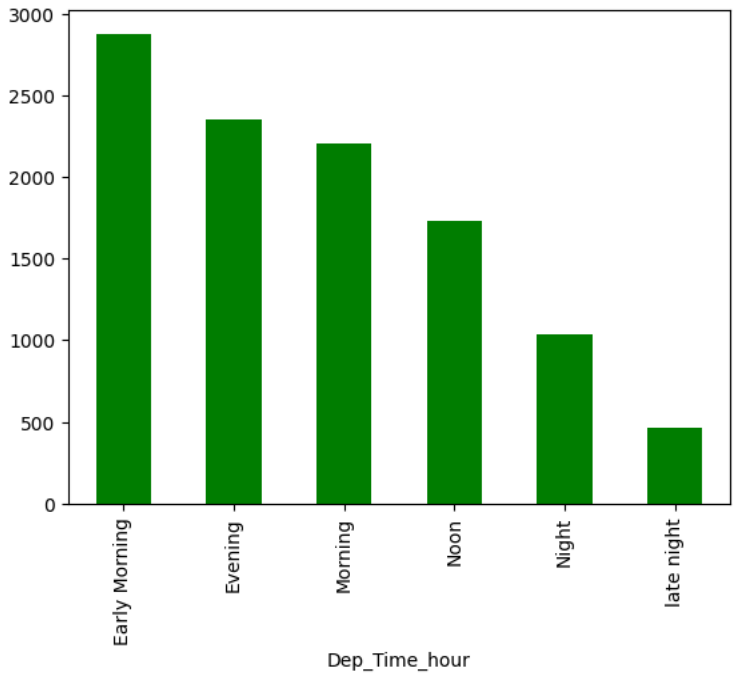
```
Index(['Airline', 'Date_of_Journey', 'Source', 'Destination', 'Route',
       'Duration', 'Total_Stops', 'Additional_Info', 'Price', 'Journey_day',
       'Journey_month', 'Journey_year', 'Dep_Time_hour', 'Dep_Time_minute',
       'Arrival_Time_hour', 'Arrival_Time_minute'],
      dtype='object')
```

```
#### Converting the flight Dep_Time into proper time i.e. mid_night, morning, afternoon and evening.

def flight_dep_time(x):
    if (x>4) and (x<=8):
        return "Early Morning"
    elif (x>8) and (x<=12):
        return "Morning"
    elif (x>12) and (x<=16):
        return "Noon"
    elif (x>16) and (x<=20):
        return "Evening"
    elif (x>20) and (x<=24):
        return "Night"
    else:
        return "late night"
```

```
data['Dep_Time_hour'].apply(flight_dep_time).value_counts().plot(kind="bar" , color="g")
```

```
<Axes: xlabel='Dep_Time_hour'>
```




```
!pip install plotly
!pip install chart_studio
```

```
Requirement already satisfied: plotly in /usr/local/lib/python3.11/dist-packages (5.24.1)
Requirement already satisfied: tenacity>=6.2.0 in /usr/local/lib/python3.11/dist-packages (from plotly) (9.0.0)
Requirement already satisfied: packaging in /usr/local/lib/python3.11/dist-packages (from plotly) (24.2)
Collecting chart_studio
  Downloading chart_studio-1.1.0-py3-none-any.whl.metadata (1.3 kB)
Requirement already satisfied: plotly in /usr/local/lib/python3.11/dist-packages (from chart_studio) (5.24.1)
Requirement already satisfied: requests in /usr/local/lib/python3.11/dist-packages (from chart_studio) (2.32.3)
Collecting retrying>=1.3.3 (from chart_studio)
  Downloading retrying-1.3.4-py3-none-any.whl.metadata (6.9 kB)
Requirement already satisfied: six in /usr/local/lib/python3.11/dist-packages (from chart_studio) (1.17.0)
Requirement already satisfied: tenacity>=6.2.0 in /usr/local/lib/python3.11/dist-packages (from plotly->chart_studio) (9.0.0)
Requirement already satisfied: packaging in /usr/local/lib/python3.11/dist-packages (from plotly->chart_studio) (24.2)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.11/dist-packages (from requests->chart_studio) (3
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-packages (from requests->chart_studio) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.11/dist-packages (from requests->chart_studio) (2.3.0)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.11/dist-packages (from requests->chart_studio) (2025.1.3
Downloading chart_studio-1.1.0-py3-none-any.whl (64 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 64.4/64.4 kB 1.6 MB/s eta 0:00:00
Downloading retrying-1.3.4-py3-none-any.whl (11 kB)
Installing collected packages: retrying, chart_studio
Successfully installed chart_studio-1.1.0 retrying-1.3.4
```

```
!pip install cufflinks
```

```
Requirement already satisfied: cufflinks in /usr/local/lib/python3.11/dist-packages (0.17.3)
Requirement already satisfied: numpy>=1.9.2 in /usr/local/lib/python3.11/dist-packages (from cufflinks) (1.26.4)
Requirement already satisfied: pandas>=0.19.2 in /usr/local/lib/python3.11/dist-packages (from cufflinks) (2.2.2)
Requirement already satisfied: plotly>=4.1.1 in /usr/local/lib/python3.11/dist-packages (from cufflinks) (5.24.1)
Requirement already satisfied: six>=1.9.0 in /usr/local/lib/python3.11/dist-packages (from cufflinks) (1.17.0)
Requirement already satisfied: colorlover>=0.2.1 in /usr/local/lib/python3.11/dist-packages (from cufflinks) (0.3.0)
Requirement already satisfied: setuptools>=34.4.1 in /usr/local/lib/python3.11/dist-packages (from cufflinks) (75.1.0)
Requirement already satisfied: ipython>=5.3.0 in /usr/local/lib/python3.11/dist-packages (from cufflinks) (7.34.0)
Requirement already satisfied: ipywidgets>=7.0.0 in /usr/local/lib/python3.11/dist-packages (from cufflinks) (7.7.1)
Collecting jedi>=0.16 (from ipython>=5.3.0->cufflinks)
  Downloading jedi-0.19.2-py2.py3-none-any.whl.metadata (22 kB)
Requirement already satisfied: decorator in /usr/local/lib/python3.11/dist-packages (from ipython>=5.3.0->cufflinks) (4.4.2)
Requirement already satisfied: pickleshare in /usr/local/lib/python3.11/dist-packages (from ipython>=5.3.0->cufflinks) (0.7.5)
Requirement already satisfied: traitlets>=4.2 in /usr/local/lib/python3.11/dist-packages (from ipython>=5.3.0->cufflinks) (5.7.1
Requirement already satisfied: prompt-toolkit!=3.0.0,!=3.0.1,<3.1.0,>=2.0.0 in /usr/local/lib/python3.11/dist-packages (from ipyt
Requirement already satisfied: pygments in /usr/local/lib/python3.11/dist-packages (from ipython>=5.3.0->cufflinks) (2.18.0)
Requirement already satisfied: backcall in /usr/local/lib/python3.11/dist-packages (from ipython>=5.3.0->cufflinks) (0.2.0)
Requirement already satisfied: matplotlib-inline in /usr/local/lib/python3.11/dist-packages (from ipython>=5.3.0->cufflinks) (0.1
Requirement already satisfied: pexpect>4.3 in /usr/local/lib/python3.11/dist-packages (from ipython>=5.3.0->cufflinks) (4.9.0)
Requirement already satisfied: ipykernel>=4.5.1 in /usr/local/lib/python3.11/dist-packages (from ipywidgets>=7.0.0->cufflinks) (5
Requirement already satisfied: ipython-genutils~=0.2.0 in /usr/local/lib/python3.11/dist-packages (from ipywidgets>=7.0.0->cuffli
Requirement already satisfied: widgetsnbextension~=3.6.0 in /usr/local/lib/python3.11/dist-packages (from ipywidgets>=7.0.0->cuff
Requirement already satisfied: jupyterlab-widgets>=1.0.0 in /usr/local/lib/python3.11/dist-packages (from ipywidgets>=7.0.0->cuff
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.11/dist-packages (from pandas>=0.19.2->cufflinks
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist-packages (from pandas>=0.19.2->cufflinks) (2025.1)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-packages (from pandas>=0.19.2->cufflinks) (2025.1
```

```
Requirement already satisfied: tenacity>=6.2.0 in /usr/local/lib/python3.11/dist-packages (from plotly>=4.1.1->cufflinks) (9.0.0
Requirement already satisfied: packaging in /usr/local/lib/python3.11/dist-packages (from plotly>=4.1.1->cufflinks) (24.2)
Requirement already satisfied: jupyter-client in /usr/local/lib/python3.11/dist-packages (from ipykernel>=4.5.1->ipywidgets>=7.0
Requirement already satisfied: tornado>=4.2 in /usr/local/lib/python3.11/dist-packages (from ipykernel>=4.5.1->ipywidgets>=7.0.0
Requirement already satisfied: parso<0.9.0,>=0.8.4 in /usr/local/lib/python3.11/dist-packages (from jedi>=0.16->ipython>=5.3.0->c
Requirement already satisfied: ptyprocess>=0.5 in /usr/local/lib/python3.11/dist-packages (from pexpect>4.3->ipython>=5.3.0->cuff
Requirement already satisfied: wcwidth in /usr/local/lib/python3.11/dist-packages (from prompt-toolkit!=3.0.0,!=3.0.1,<3.1.0,>=2
Requirement already satisfied: notebook>=4.4.1 in /usr/local/lib/python3.11/dist-packages (from widgetsnbextension~=3.6.0->ipywic
Requirement already satisfied: jinja2 in /usr/local/lib/python3.11/dist-packages (from notebook>=4.4.1->widgetsnbextension~=3.6.0
Requirement already satisfied: pyzmq<25,>=17 in /usr/local/lib/python3.11/dist-packages (from notebook>=4.4.1->widgetsnbextensior
Requirement already satisfied: argon2-cffi in /usr/local/lib/python3.11/dist-packages (from notebook>=4.4.1->widgetsnbextension~=
Requirement already satisfied: jupyter-core>=4.6.1 in /usr/local/lib/python3.11/dist-packages (from notebook>=4.4.1->widgetsnbext
Requirement already satisfied: nbformat in /usr/local/lib/python3.11/dist-packages (from notebook>=4.4.1->widgetsnbextension~=3.6
Requirement already satisfied: nbconvert>=5 in /usr/local/lib/python3.11/dist-packages (from notebook>=4.4.1->widgetsnbextension-
Requirement already satisfied: nest-asyncio>=1.5 in /usr/local/lib/python3.11/dist-packages (from notebook>=4.4.1->widgetsnbexter
Requirement already satisfied: Send2Trash>=1.8.0 in /usr/local/lib/python3.11/dist-packages (from notebook>=4.4.1->widgetsnbexter
Requirement already satisfied: terminado>=0.8.3 in /usr/local/lib/python3.11/dist-packages (from notebook>=4.4.1->widgetsnbextens
Requirement already satisfied: prometheus-client in /usr/local/lib/python3.11/dist-packages (from notebook>=4.4.1->widgetsnbexter
Requirement already satisfied: nbclassic>=0.4.7 in /usr/local/lib/python3.11/dist-packages (from notebook>=4.4.1->widgetsnbextens
Requirement already satisfied: platformdirs>=2.5 in /usr/local/lib/python3.11/dist-packages (from jupyter-core>=4.6.1->notebook>=
Requirement already satisfied: notebook-shim>=0.2.3 in /usr/local/lib/python3.11/dist-packages (from nbclassic>=0.4.7->notebook>=
Requirement already satisfied: beautifulsoup4 in /usr/local/lib/python3.11/dist-packages (from nbconvert>=5->notebook>=4.4.1->wic
Requirement already satisfied: bleach!=5.0.0 in /usr/local/lib/python3.11/dist-packages (from bleach[css]!=5.0.0->nbconvert>=5->n
Requirement already satisfied: defusedxml in /usr/local/lib/python3.11/dist-packages (from nbconvert>=5->notebook>=4.4.1->widgets
Requirement already satisfied: jupyterlab-pygments in /usr/local/lib/python3.11/dist-packages (from nbconvert>=5->notebook>=4.4.1
Requirement already satisfied: markupsafe>=2.0 in /usr/local/lib/python3.11/dist-packages (from nbconvert>=5->notebook>=4.4.1->wi
Requirement already satisfied: mistune<4,>=2.0.3 in /usr/local/lib/python3.11/dist-packages (from nbconvert>=5->notebook>=4.4.1->
Requirement already satisfied: nbclient>=0.5.0 in /usr/local/lib/python3.11/dist-packages (from nbconvert>=5->notebook>=4.4.1->wi
Requirement already satisfied: pandocfilters>=1.4.1 in /usr/local/lib/python3.11/dist-packages (from nbconvert>=5->notebook>=4.4
Requirement already satisfied: fastjsonschema>=2.15 in /usr/local/lib/python3.11/dist-packages (from nbformat->notebook>=4.4.1->w
Requirement already satisfied: jsonschema>=2.6 in /usr/local/lib/python3.11/dist-packages (from nbformat->notebook>=4.4.1->widget
```

```
import plotly
import cufflinks as cf
from cufflinks.offline import go_offline
from plotly.offline import plot , iplot , init_notebook_mode , download_plotlyjs
init_notebook_mode(connected=True)
cf.go_offline()
```

```
data['Dep_Time_hour'].apply(flight_dep_time).value_counts().iplot(kind="bar")
```

```
data.head(3)
```

	Airline	Date_of_Journey	Source	Destination	Route	Duration	Total_Stops	Additional_Info	Price	Journey_day	Journey_month
0	IndiGo	2019-03-24	Banglore	New Delhi	BLR → DEL	2h 50m	non-stop	No info	3897	24	3
1	Air India	2019-05-01	Kolkata	Banglore	CCU → IXR → BBI → BLR	7h 25m	2 stops	No info	7662	1	5
2	Jet Airways	2019-06-09	Delhi	Cochin	DEL → LKO → BOM → COK	19h	2 stops	No info	13882	9	6

```
def preprocess_duration(x):
    if 'h' not in x:
        x = '0h' + ' ' + x
    elif 'm' not in x:
        x = x + ' ' +'0m'

    return x
```

```
data['Duration'] = data['Duration'].apply(preprocess_duration)
```

```
data['Duration']
```

	Duration
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

10682 rows × 1 columns

dtype: object

```
data['Duration_hours'] = data['Duration'].apply(lambda x : int(x.split(' ')[0][0:-1]))
```

```
data['Duration_mins'] = data['Duration'].apply(lambda x : int(x.split(' ')[1][0:-1]))
```

```
data.head(2)
```

	Airline	Date_of_Journey	Source	Destination	Route	Duration	Total_Stops	Additional_Info	Price	Journey_day	Journey_month
0	IndiGo	2019-03-24	Banglore	New Delhi	BLR → DEL	2h 50m	non-stop	No info	3897	24	3
1	Air India	2019-05-01	Kolkata	Banglore	CCU → IXR → BBI → BLR	7h 25m	2 stops	No info	7662	1	5

```
pd.to_timedelta(data["Duration"]).dt.components.hours
```

	hours
0	2
1	7
2	19
3	5
4	4
...	...
10678	2
10679	2
10680	3
10681	2
10682	8

10682 rows × 1 columns

dtype: int64

```
data["Duration_hour"] = pd.to_timedelta(data["Duration"]).dt.components.hours

data["Duration_minute"] = pd.to_timedelta(data["Duration"]).dt.components.minutes

data['Duration'] ## converting duration into total minutes duration ..
```

	Duration
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

10682 rows × 1 columns

dtype: object

```
data['Duration_total_mins'] = data['Duration'].str.replace('h' ,"*60").str.replace(' ' , '+').str.replace('m' , "*1").apply(eval)

#data["Duration_in_minute"] = data["Duration_hour"]*60 + data["Duration_minute"]

data['Duration_total_mins']
```

	Duration_total_mins
0	170
1	445
2	1140
3	325
4	285
...	...
10678	150
10679	155
10680	180
10681	160
10682	500

10682 rows × 1 columns

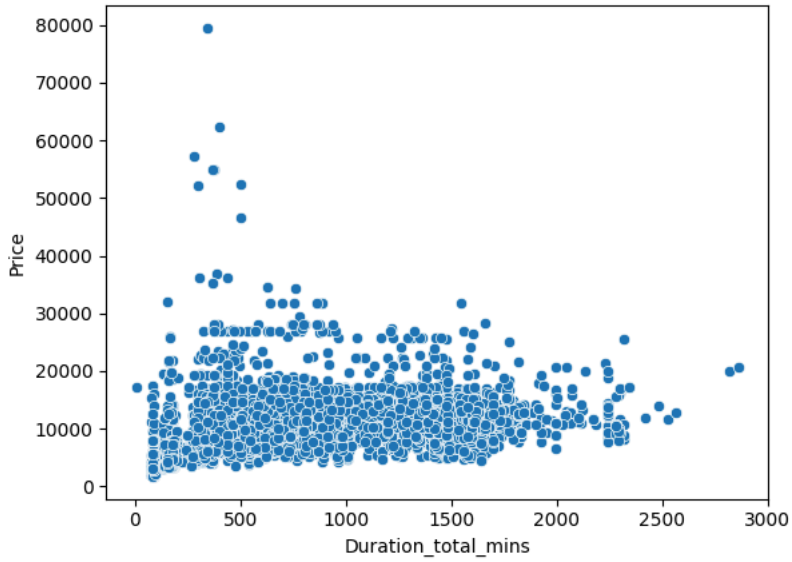
dtype: int64

```
data.columns
```

```
Index(['Airline', 'Date_of_Journey', 'Source', 'Destination', 'Route',
       'Duration', 'Total_Stops', 'Additional_Info', 'Price', 'Journey_day',
       'Journey_month', 'Journey_year', 'Dep_Time_hour', 'Dep_Time_minute',
       'Arrival_Time_hour', 'Arrival_Time_minute', 'Duration_hours',
       'Duration_mins', 'Duration_hour', 'Duration_minute',
       'Duration_total_mins'],
      dtype='object')
```

```
sns.scatterplot(x="Duration_total_mins" , y="Price" , data=data)
```

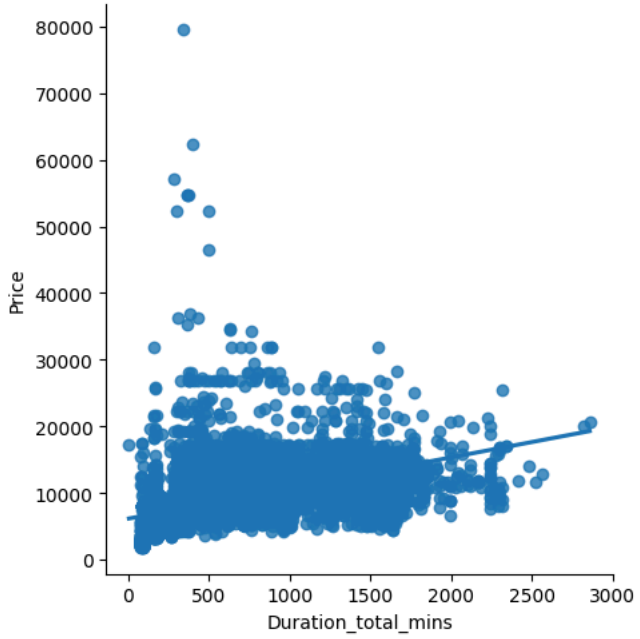
```
<Axes: xlabel='Duration_total_mins', ylabel='Price'>
```




```
sns.lmplot(x="Duration_total_mins" , y="Price" , data=data)

### pretty clear that As the duration of minutes increases Flight price also increases.
```

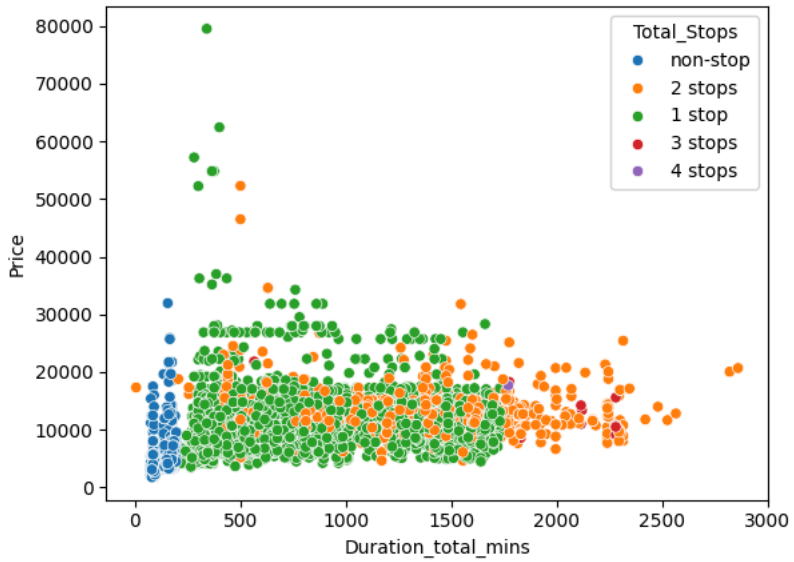
```
<seaborn.axisgrid.FacetGrid at 0x78de08295210>
```




```
### lets understand whether total stops affect price or not !
```

```
sns.scatterplot(x="Duration_total_mins" , y="Price" , hue="Total_Stops", data=data)
```

```
<Axes: xlabel='Duration_total_mins', ylabel='Price'>
```




```
data['Airline']=='Jet Airways'
```

	Airline
0	False
1	False
2	True
3	False
4	False
...	...
10678	False
10679	False
10680	True
10681	False
10682	False

10682 rows × 1 columns

dtype: bool

```
data[data['Airline']=='Jet Airways'].groupby('Route').size().sort_values(ascending=False)
```

Route	0
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

```
data.columns
```

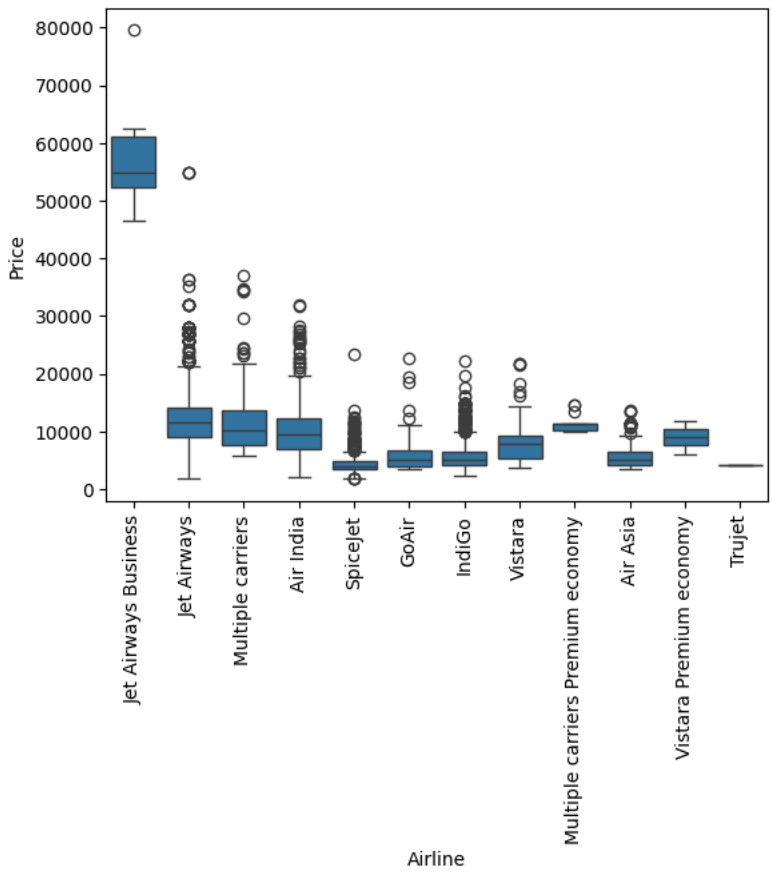
BOM → IDR → DEL → HYD 1

```
Index(['Airline', 'Date_of_Journey', 'Source', 'Destination', 'Route',
       'Duration', 'Total_Stops', 'Additional_Info', 'Price', 'Journey_day',
       'Journey_month', 'Journey_year', 'Dep_Time_hour', 'Dep_Time_minute',
       'Arrival_Time_hour', 'Arrival_Time_minute', 'Duration_hours',
       'Duration_mins', 'Duration_hour', 'Duration_minute',
       'Duration_total_mins'],
      dtype='object')
```

BOM → DED → DEL → HYD

dtype: int64

```
sns.boxplot(y='Price' , x='Airline' , data=data.sort_values('Price' , ascending=False))
plt.xticks(rotation="vertical")
plt.show()
```




```
data.head(2)
```

	Airline	Date_of_Journey	Source	Destination	Route	Duration	Total_Stops	Additional_Info	Price	Journey_day	...	Journey_y
0	IndiGo	2019-03-24	Banglore	New Delhi	BLR → DEL	2h 50m	non-stop	No info	3897	24	...	2(
1	Air India	2019-05-01	Kolkata	Banglore	CCU → IXR → BBI → BLR	7h 25m	2 stops	No info	7662	1	...	2(

2 rows × 21 columns

```
cat_col = [col for col in data.columns if data[col].dtype=="object"]
```

```
num_col = [col for col in data.columns if data[col].dtype!="object"]
```

```
cat_col
```

```
['Airline',
 'Source',
 'Destination',
 'Route',
 'Duration',
 'Total_Stops',
 'Additional_Info']
```

```
data['Source'].unique()
```

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

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

	Source
0	1
1	0
2	0
3	0
4	1
...	...
10678	0
10679	0
10680	1
10681	1
10682	0

10682 rows × 1 columns

dtype: int64

```
for sub_category in data['Source'].unique():
    data['Source_'+sub_category] = data['Source'].apply(lambda x : 1 if x==sub_category else 0)
```

```
data.head(3)
```

	Airline	Date_of_Journey	Source	Destination	Route	Duration	Total_Stops	Additional_Info	Price	Journey_day	...	Duration_
0	IndiGo	2019-03-24	Banglore	New Delhi	BLR → DEL	2h 50m	non-stop	No info	3897	24	...	
1	Air India	2019-05-01	Kolkata	Banglore	CCU → IXR → BBI → BLR	7h 25m	2 stops	No info	7662	1	...	
2	Jet Airways	2019-06-09	Delhi	Cochin	DEL → LKO → BOM → COK	19h 0m	2 stops	No info	13882	9	...	

3 rows × 26 columns

```
cat_col
```

```
['Airline',
 'Source',
 'Destination',
 'Route',
 'Duration',
 'Total_Stops',
 'Additional_Info']
```

```
data.head(2)
```

	Airline	Date_of_Journey	Source	Destination	Route	Duration	Total_Stops	Additional_Info	Price	Journey_day	...	Duration_l
0	IndiGo	2019-03-24	Banglore	New Delhi	BLR → DEL	2h 50m	non-stop	No info	3897	24	...	
1	Air India	2019-05-01	Kolkata	Banglore	CCU → IXR → BBI → BLR	7h 25m	2 stops	No info	7662	1	...	

2 rows × 26 columns

```
data['Airline'].nunique()
```

```
12
```

```
data.groupby(['Airline'])['Price'].mean().sort_values()
```

Airline	Price
Trujet	4140.000000
SpiceJet	4338.284841
Air Asia	5590.260188
IndiGo	5673.682903
GoAir	5861.056701
Vistara	7796.348643
Vistara Premium economy	8962.333333
Air India	9612.427756
Multiple carriers	10902.678094
Multiple carriers Premium economy	11418.846154
Jet Airways	11643.923357
Jet Airways Business	58358.666667

dtype: float64

```
airlines = data.groupby(['Airline'])['Price'].mean().sort_values().index
```

```
airlines
```

```
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')
```

```
dict_airlines = {key:index for index , key in enumerate(airlines , 0)}
```

```
dict_airlines
```

```
{'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}
```

```
data['Airline'] = data['Airline'].map(dict_airlines)
```

```
data['Airline']
```

	Airline
0	3
1	7
2	10
3	3
4	3
...	...
10678	2
10679	7
10680	10
10681	5
10682	7

10682 rows × 1 columns

dtype: int64

```
data.head(3)
```

	Airline	Date_of_Journey	Source	Destination	Route	Duration	Total_Stops	Additional_Info	Price	Journey_day	...	Duration_
0	3	2019-03-24	Banglore	New Delhi	BLR → DEL	2h 50m	non-stop	No info	3897	24	...	
1	7	2019-05-01	Kolkata	Banglore	CCU → IXR → BBI → BLR	7h 25m	2 stops	No info	7662	1	...	
2	10	2019-06-09	Delhi	Cochin	DEL → LKO → BOM → COK	19h 0m	2 stops	No info	13882	9	...	

3 rows × 26 columns

```
data['Destination'].unique()
```

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

```
data['Destination'].replace('New Delhi' , 'Delhi' , inplace=True)
```

```
data['Destination'].unique()
```

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

```
dest = data.groupby(['Destination'])['Price'].mean().sort_values().index
```

```
dest
```

```
Index(['Kolkata', 'Hyderabad', 'Delhi', 'Banglore', 'Cochin'], dtype='object', name='Destination')
```

```
dict_dest = {key:index for index , key in enumerate(dest , 0)}
```

```
dict_dest
```

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

```
data['Destination'] = data['Destination'].map(dict_dest)
```

```
data['Destination']
```

	Destination
0	2
1	3
2	4
3	3
4	2
...	...
10678	3
10679	3
10680	2
10681	2
10682	4

10682 rows × 1 columns

dtype: int64

```
data.head(3)
```

	Airline	Date_of_Journey	Source	Destination	Route	Duration	Total_Stops	Additional_Info	Price	Journey_day	...	Duration_
0	3	2019-03-24	Banglore	2	BLR → DEL	2h 50m	non-stop	No info	3897	24	...	
1	7	2019-05-01	Kolkata	3	CCU → IXR → BBI → BLR	7h 25m	2 stops	No info	7662	1	...	
2	10	2019-06-09	Delhi	4	DEL → LKO → BOM → COK	19h 0m	2 stops	No info	13882	9	...	

3 rows × 26 columns

```
data.head(3)
```


	Airline	Date_of_Journey	Source	Destination	Route	Duration	Total_Stops	Additional_Info	Price	Journey_day	...	Duration_
0	3	2019-03-24	Banglore	2	BLR → DEL	2h 50m	non-stop	No info	3897	24	...	
1	7	2019-05-01	Kolkata	3	CCU → IXR → BBI → BLR	7h 25m	2 stops	No info	7662	1	...	
2	10	2019-06-09	Delhi	4	DEL → LKO → BOM → COK	19h 0m	2 stops	No info	13882	9	...	

3 rows × 26 columns


```
data['Total_Stops']
```

	Total_Stops
0	non-stop
1	2 stops
2	2 stops
3	1 stop
4	1 stop
...	...
10678	non-stop
10679	non-stop
10680	non-stop
10681	non-stop
10682	2 stops

10682 rows × 1 columns

dtype: object

```
data['Total_Stops'].unique()
```

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

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

```
data['Total_Stops'] = data['Total_Stops'].map(stop)
```

```
data['Total_Stops']
```

	Total_Stops
0	0
1	2
2	2
3	1
4	1
...	...
10678	0
10679	0
10680	0
10681	0
10682	2

10682 rows × 1 columns

dtype: int64

```
data.head(1)
```

	Airline	Date_of_Journey	Source	Destination	Route	Duration	Total_Stops	Additional_Info	Price	Journey_day	...	Duration_l
0	3	2019-03-24	Banglore	2	BLR → DEL	2h 50m	0	No info	3897	24	...	

1 rows × 26 columns

```
data.columns
```

```
Index(['Airline', 'Date_of_Journey', 'Source', 'Destination', 'Route',
       'Duration', 'Total_Stops', 'Additional_Info', 'Price', 'Journey_day',
       'Journey_month', 'Journey_year', 'Dep_Time_hour', 'Dep_Time_minute',
       'Arrival_Time_hour', 'Arrival_Time_minute', 'Duration_hours',
       'Duration_mins', 'Duration_hour', 'Duration_minute',
       'Duration_total_mins', 'Source_Banglore', 'Source_Kolkata',
```

```
       'Source_Delhi', 'Source_Chennai', 'Source_Mumbai'],
      dtype='object')
```

```
data['Additional_Info'].value_counts()/len(data)*100

# Additional_Info contains almost 80% no_info,so we can drop this column
```

Additional_Info	count
No info	78.112713
In-flight meal not included	18.554578
No check-in baggage included	2.995694
1 Long layover	0.177869
Change airports	0.065531
Business class	0.037446
No Info	0.028085
1 Short layover	0.009362
Red-eye flight	0.009362
2 Long layover	0.009362

dtype: float64

```
data.head(4)
```

	Airline	Date_of_Journey	Source	Destination	Route	Duration	Total_Stops	Additional_Info	Price	Journey_day	...	Duration_
0	3	2019-03-24	Banglore	2	BLR → DEL	2h 50m	0	No info	3897	24	...	
1	7	2019-05-01	Kolkata	3	CCU → IXR → BBI → BLR	7h 25m	2	No info	7662	1	...	
2	10	2019-06-09	Delhi	4	DEL → LKO → BOM → COK	19h 0m	2	No info	13882	9	...	
3	3	2019-05-12	Kolkata	3	CCU → NAG → BLR	5h 25m	1	No info	6218	12	...	

4 rows × 26 columns

```
data.columns
```

```
Index(['Airline', 'Date_of_Journey', 'Source', 'Destination', 'Route',
       'Duration', 'Total_Stops', 'Additional_Info', 'Price', 'Journey_day',
       'Journey_month', 'Journey_year', 'Dep_Time_hour', 'Dep_Time_minute',
       'Arrival_Time_hour', 'Arrival_Time_minute', 'Duration_hours',
       'Duration_mins', 'Duration_hour', 'Duration_minute',
       'Duration_total_mins', 'Source_Banglore', 'Source_Kolkata',
       'Source_Delhi', 'Source_Chennai', 'Source_Mumbai'],
      dtype='object')
```

```
data['Journey_year'].unique()
```

```
array([2019], dtype=int32)
```

```
data.drop(columns=['Date_of_Journey' , 'Additional_Info' , 'Duration_total_mins' , 'Source' , 'Journey_year'] , axis=1 , inplace=True)
```

```
data.columns
```

```
Index(['Airline', 'Destination', 'Route', 'Duration', 'Total_Stops', 'Price',
       'Journey_day', 'Journey_month', 'Dep_Time_hour', 'Dep_Time_minute',
       'Arrival_Time_hour', 'Arrival_Time_minute', 'Duration_hours',
       'Duration_mins', 'Duration_hour', 'Duration_minute', 'Source_Banglore',
       'Source_Kolkata', 'Source_Delhi', 'Source_Chennai', 'Source_Mumbai'],
      dtype='object')
```

```
data.head(4)
```

	Airline	Destination	Route	Duration	Total_Stops	Price	Journey_day	Journey_month	Dep_Time_hour	Dep_Time_minute	...	Arriva
0	3	2	BLR → DEL	2h 50m	0	3897	24	3	22	20	...	
1	7	3	CCU → IXR → BBI → BLR	7h 25m	2	7662	1	5	5	50	...	
2	10	4	DEL → LKO → BOM → COK	19h 0m	2	13882	9	6	9	25	...	
3	3	3	CCU → NAG → BLR	5h 25m	1	6218	12	5	18	5	...	

4 rows × 21 columns

```
data.drop(columns=['Route'] , axis=1 , inplace=True)

## we can drop Route as well bcz Route is directly related to Total stops & considering 2 same features doesnt make sense while building
```

```
data.head(3)
```

	Airline	Destination	Duration	Total_Stops	Price	Journey_day	Journey_month	Dep_Time_hour	Dep_Time_minute	Arrival_Time_hour
0	3	2	2h 50m	0	3897	24	3	22	20	1
1	7	3	7h 25m	2	7662	1	5	5	50	13
2	10	4	19h 0m	2	13882	9	6	9	25	4

```
data.drop(columns=['Duration'] , axis=1 , inplace=True)

## we can drop "Duration" feature as we have extracted "Duration hour" & "Duration Minute"..
```

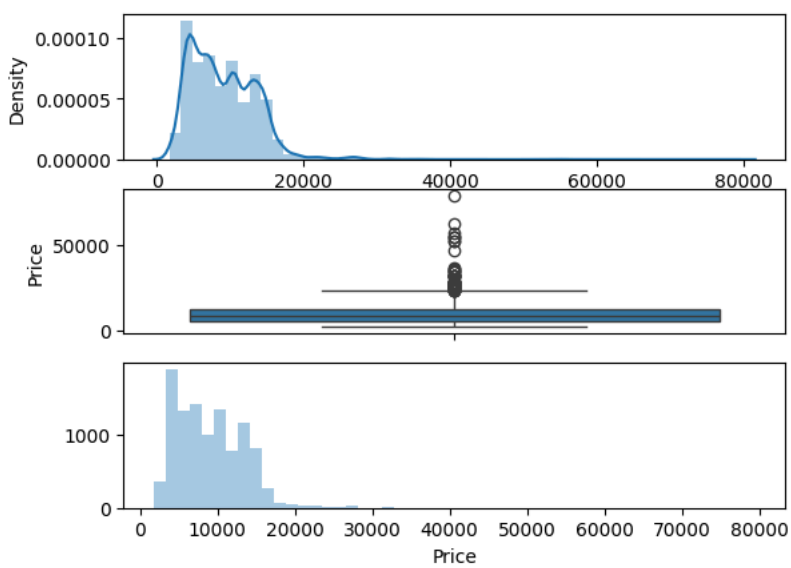
```
data.head(3)
```

	Airline	Destination	Total_Stops	Price	Journey_day	Journey_month	Dep_Time_hour	Dep_Time_minute	Arrival_Time_hour	Arrival_
0	3	2	0	3897	24	3	22	20	1	
1	7	3	2	7662	1	5	5	50	13	
2	10	4	2	13882	9	6	9	25	4	

```
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)
```

```
plot(data , 'Price')
```




```
q1 = data['Price'].quantile(0.25)
q3 = data['Price'].quantile(0.75)

iqr = q3- q1

maximum = q3 + 1.5*iqr
minimum = q1 - 1.5*iqr
```

```
print(maximum)
```

```
23017.0
```

```
print(minimum)
```

```
-5367.0
```

```
print([price for price in data['Price'] if price> maximum or price<minimum])
```

```
[27430, 36983, 26890, 26890, 25139, 27210, 52229, 26743, 26890, 25735, 27992, 26890, 26890, 23583, 26890, 23533, 24115, 25735, 54826
```

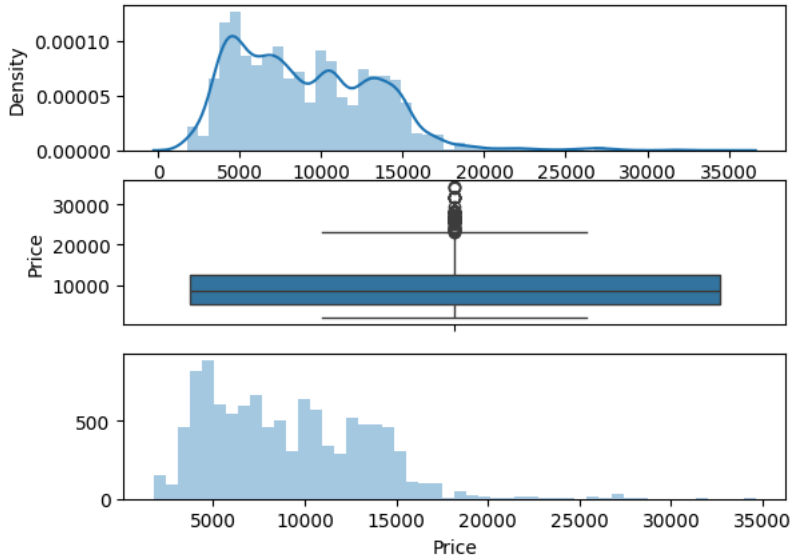
```
len([price for price in data['Price'] if price> maximum or price<minimum])
```

```
94
```

```
### wherever I have price >35K just replace replace it with median of Price

data['Price'] = np.where(data['Price']>=35000 , data['Price'].median() , data['Price'])

plot(data , 'Price')
```




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

```
y = data['Price']
```

```
from sklearn.feature_selection import mutual_info_regression
```

```
imp = mutual_info_regression(X , y)
```

```
imp
```

```
array([1.31866835, 1.06476081, 0.78987932, 0.37576399, 0.62323616,
       0.92060983, 0.75833521, 1.14140761, 0.89897128, 1.12075867,
       0.6783199 , 0.94768975, 0.67807822, 0.38309987, 0.45622937,
       0.52090786, 0.14027126, 0.19986067])
```

```
imp_df = pd.DataFrame(imp , index=X.columns)
```

```
imp_df.columns = ['importance']
```

```
imp_df
```

	importance
Airline	1.318668
Destination	1.064761
Total_Stops	0.789879
Journey_day	0.375764
Journey_month	0.623236
Dep_Time_hour	0.920610
Dep_Time_minute	0.758335
Arrival_Time_hour	1.141408
Arrival_Time_minute	0.898971
Duration_hours	1.120759
Duration_mins	0.678320
Duration_hour	0.947690
Duration_minute	0.678078
Source_Banglore	0.383100
Source_Kolkata	0.456229
Source_Delhi	0.520908
Source_Chennai	0.140271
Source_Mumbai	0.199861

```
imp_df.sort_values(by='importance' , ascending=False)
```

	importance
Airline	1.318668
Arrival_Time_hour	1.141408
Duration_hours	1.120759
Destination	1.064761
Duration_hour	0.947690
Dep_Time_hour	0.920610
Arrival_Time_minute	0.898971
Total_Stops	0.789879
Dep_Time_minute	0.758335
Duration_mins	0.678320
Duration_minute	0.678078
Journey_month	0.623236
Source_Delhi	0.520908
Source_Kolkata	0.456229
Source_Banglore	0.383100
Journey_day	0.375764
Source_Mumbai	0.199861
Source_Chennai	0.140271

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

```
X_train, X_test, y_train, y_test = train_test_split(
   X, y, test_size=0.25, random_state=42)
```

```
from sklearn.ensemble import RandomForestRegressor
```

```
ml_model = RandomForestRegressor()
```

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

```
▾ RandomForestRegressor
RandomForestRegressor()
```

```
y_pred = ml_model.predict(X_test)
```

```
y_pred
```

```
array([16828.03 ,  5403.81 ,  8902.5  , ...,  3536.27 ,  6486.074,
        6847.79 ])
```

```
from sklearn import metrics
```

```
metrics.r2_score(y_test , y_pred)
```

```
0.8108258730536901
```

```
!pip install pickle
```

```
ERROR: Could not find a version that satisfies the requirement pickle (from versions: none)
ERROR: No matching distribution found for pickle
```

```
import pickle
```

```
# open a file, where you want to store the data
file = open('rf_random.pkl' , 'wb')
```

```
# dump information to that file
pickle.dump(ml model , file)
```

```
model = open('rf_random.pkl' , 'rb')
```

```
forest = pickle.load(model)
```

```
y_pred2 = forest.predict(X_test)
```

```
metrics.r2_score(y_test , y_pred2)
```

```
0.8108258730536901
```

```
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
```

```
mape(y_test , y_pred)
```

```
13.170562309689032
```

```
from sklearn import metrics
```

```
def predict(ml_model):
    model = ml_model.fit(X_train , y_train)
    print('Training score : {}'.format(model.score(X_train , y_train)))
    y_predection = model.predict(X_test)
    print('predictions are : {}'.format(y_predection))
    print('\n')
    r2_score = metrics.r2_score(y_test , y_predection)
    print('r2 score : {}'.format(r2_score))
    print('MAE : {}'.format(metrics.mean_absolute_error(y_test , y_predection)))
    print('MSE : {}'.format(metrics.mean_squared_error(y_test , y_predection)))
    print('RMSE : {}'.format(np.sqrt(metrics.mean_squared_error(y_test , y_predection))))
    print('MAPE : {}'.format(mape(y_test , y_predection)))
    sns.distplot(y_test - y_predection)
```

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

```
Training score : 0.9514682503074033
predictions are : [16814.59  5414.02  8836.86 ...  3452.5   6242.7   6891.56]


r2 score : 0.8127625037347651
MAE : 1174.9250203089894
MSE : 3645064.619649064
```