Airfare Price Prediction

Statement of Issue It can be hard to guess airline ticket rates, we might see a fare today, find out the price of the same flight tomorrow, it's going to be a different story. We may have heard travelers sometimes complain that the costs of airline fares are too volatile. As data scientists, we can show that something can be expected provided the correct data.

FEATURES: Airline: The name of the airline.

Date_of_Journey: The date of the journey

Source: The source from which the service begins.

Destination: The destination where the service ends.

Route: The route taken by the flight to reach the destination.

Dep_Time: The time when the journey starts from the source.

Arrival_Time: Time of arrival at the destination.

Duration: Total duration of the flight.

Total_Stops: Total stops between the source and destination.

Additional_Info: Additional information about the flight

Price: The price of the ticket

In [ ]:

In [ ]:

In [ ]:

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
sns.set()
```

Importing dataset

In [2]:
```
df = pd.read_excel("Case_study_Flight_Fare_.xlsx")
```

set max coulmns to None so we can see all columns from dataset

In [4]:
```
pd.set_option('display.max_columns', None)
```

show the first five rows

In [5]:
```
df.head()
```

Out[5]:

	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
2	Jet Airways	9/06/2019	Delhi	Cochin	DEL → LKO → BOM → COK	09:25	04:25 10 Jun	19h	2 stops	No info	13882
3	IndiGo	12/05/2019	Kolkata	Banglore	CCU → NAG →	18:05	23:30	5h 25m	1 stop	No info	6218

					BLR						
4	IndiGo	01/03/2019	Banglore	New Delhi	BLR → NAG → DEL	16:50	21:35	4h 45m	1 stop	No info	13302

Chech basic information of dataset

In [6]:
```
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
```

Check the value counts of Duration column

In [7]:
```
df["Duration"].value_counts()
```

Out[7]:
```
2h 50m     550
1h 30m     386
2h 55m     337
2h 45m     337
2h 35m     329
          ...
29h 40m      1
35h 35m      1
27h 30m      1
13h 35m      1
27h 55m      1
Name: Duration, Length: 368, dtype: int64
```

check the count of null values in dataset column

In [8]:
```
df.isnull().sum()
```

Out[8]:
```
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
```

check the unique values in Route counts

In [9]:
```
df['Route'].unique()
```

Out[9]:
```
array(['BLR → DEL', 'CCU → IXR → BBI → BLR', 'DEL → LKO → BOM → COK',
       'CCU → NAG → BLR', 'BLR → NAG → DEL', 'CCU → BLR',
       'BLR → BOM → DEL', 'DEL → BOM → COK', 'DEL → BLR → COK',
       'MAA → CCU', 'CCU → BOM → BLR', 'DEL → AMD → BOM → COK',
       'DEL → PNQ → COK', 'DEL → CCU → BOM → COK', 'BLR → COK → DEL',
       'DEL → IDR → BOM → COK', 'DEL → LKO → COK',
       'CCU → GAU → DEL → BLR', 'DEL → NAG → BOM → COK',
       'CCU → MAA → BLR', 'DEL → HYD → COK', 'CCU → HYD → BLR',
       'DEL → COK', 'CCU → DEL → BLR', 'BLR → BOM → AMD → DEL',
       'BOM → DEL → HYD', 'DEL → MAA → COK', 'BOM → HYD',
       'DEL → BHO → BOM → COK', 'DEL → JAI → BOM → COK',
       'DEL → ATQ → BOM → COK', 'DEL → JDH → BOM → COK',
       'CCU → BBI → BOM → BLR', 'BLR → MAA → DEL',
       'DEL → GOI → BOM → COK', 'DEL → BDQ → BOM → COK',
       'CCU → JAI → BOM → BLR', 'CCU → BBI → BLR', 'BLR → HYD → DEL',
       'DEL → TRV → COK', 'CCU → IXR → DEL → BLR',
       'DEL → IXU → BOM → COK', 'CCU → IXB → BLR',
       'BLR → BOM → JDH → DEL', 'DEL → UDR → BOM → COK',
       'DEL → HYD → MAA → COK', 'CCU → BOM → COK → BLR',
       'BLR → CCU → DEL', 'CCU → BOM → GOI → BLR',
       'DEL → RPR → NAG → BOM → COK', 'DEL → HYD → BOM → COK',
       'CCU → DEL → AMD → BLR', 'CCU → PNQ → BLR',
       'BLR → CCU → GAU → DEL', 'CCU → DEL → COK → BLR',
       'BLR → PNQ → DEL', 'BOM → JDH → DEL → HYD',
       'BLR → BOM → BHO → DEL', 'DEL → AMD → COK', 'BLR → LKO → DEL',
       'CCU → GAU → BLR', 'BOM → GOI → HYD', 'CCU → BOM → AMD → BLR',
       'CCU → BBI → IXR → DEL → BLR', 'DEL → DED → BOM → COK',
       'DEL → MAA → BOM → COK', 'BLR → AMD → DEL', 'BLR → VGA → DEL',
       'CCU → JAI → DEL → BLR', 'CCU → AMD → BLR',
       'CCU → VNS → DEL → BLR', 'BLR → BOM → IDR → DEL',
       'BLR → BBI → DEL', 'BLR → GOI → DEL', 'BOM → AMD → ISK → HYD',
       'BOM → DED → DEL → HYD', 'DEL → IXC → BOM → COK',
       'CCU → PAT → BLR', 'BLR → CCU → BBI → DEL',
       'CCU → BBI → HYD → BLR', 'BLR → BOM → NAG → DEL',
       'BLR → CCU → BBI → HYD → DEL', 'BLR → GAU → DEL',
       'BOM → BHO → DEL → HYD', 'BOM → JLR → HYD',
       'BLR → HYD → VGA → DEL', 'CCU → KNU → BLR',
       'CCU → BOM → PNQ → BLR', 'DEL → BBI → COK',
       'BLR → VGA → HYD → DEL', 'BOM → JDH → JAI → DEL → HYD',
       'DEL → GWL → IDR → BOM → COK', 'CCU → RPR → HYD → BLR',
       'CCU → VTZ → BLR', 'CCU → DEL → VGA → BLR',
       'BLR → BOM → IDR → GWL → DEL', 'CCU → DEL → COK → TRV → BLR',
       'BOM → COK → MAA → HYD', 'BOM → NDC → HYD', 'BLR → BDQ → DEL',
       'CCU → BOM → TRV → BLR', 'CCU → BOM → HBX → BLR',
       'BOM → BDQ → DEL → HYD', 'BOM → CCU → HYD',
       'BLR → TRV → COK → DEL', 'BLR → IDR → DEL',
       'CCU → IXZ → MAA → BLR', 'CCU → GAU → IMF → DEL → BLR',
       'BOM → GOI → PNQ → HYD', 'BOM → BLR → CCU → BBI → HYD',
       'BOM → MAA → HYD', 'BLR → BOM → UDR → DEL',
       'BOM → UDR → DEL → HYD', 'BLR → VGA → VTZ → DEL',
       'BLR → HBX → BOM → BHO → DEL', 'CCU → IXA → BLR',
       'BOM → RPR → VTZ → HYD', 'BLR → HBX → BOM → AMD → DEL',
       'BOM → IDR → DEL → HYD', 'BOM → BLR → HYD', 'BLR → STV → DEL',
       'CCU → IXB → DEL → BLR', 'BOM → JAI → DEL → HYD',
       'BOM → VNS → DEL → HYD', 'BLR → HBX → BOM → NAG → DEL', nan,
       'BLR → BOM → IXC → DEL', 'BLR → CCU → BBI → HYD → VGA → DEL',
       'BOM → BBI → HYD'], dtype=object)
```

check unique value in Total_Stops

In [10]:
```
df['Total_Stops'].unique()
```

Out[10]:
```
array(['non-stop', '2 stops', '1 stop', '3 stops', nan, '4 stops'],
      dtype=object)
```

There is only one value in Total_Stops & Route so we can drop null value from dataset

In [11]:
```
df.dropna(inplace = True)
```

In [12]:
```
df.isnull().sum()
```

Out[12]:
```
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 [13]:
```
df.head()
```

Out[13]:

	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
2	Jet Airways	9/06/2019	Delhi	Cochin	DEL → LKO → BOM → COK	09:25	04:25 10 Jun	19h	2 stops	No info	13882
3	IndiGo	12/05/2019	Kolkata	Banglore	CCU → NAG → BLR	18:05	23:30	5h 25m	1 stop	No info	6218
4	IndiGo	01/03/2019	Banglore	New Delhi	BLR → NAG → DEL	16:50	21:35	4h 45m	1 stop	No info	13302

Now we extract day values and month values from Date_of_Journey and create two new columns Journey_day & Journey_month

In [ ]:
```
# the Date_of_Journey given above is in categorical format as it is given in dd/mm/yyyy format, so it is not cons
# as date, so first we will convert it to date using pd.to_datetime. If it was already given in yyyy/mm/dd format
# wouldn't require to write format, we could directly use as written in bracket to extract day, month. In format
# write as the way it is given (percent d slash percent m slash percent Y). here .dt is used to convert to datet
```

In [14]:
```
# for explanation
pd.to_datetime(df.Date_of_Journey, format="%d/%m/%Y").dt.day
```

Out[14]:
```
0        24
1         1
2         9
3        12
4         1
         ..
10678     9
10679    27
10680    27
10681     1
10682     9
Name: Date_of_Journey, Length: 10682, dtype: int64
```

In [23]:
```
# for explanation
pd.to_datetime(df.Date_of_Journey, format="%d/%m/%Y").dt.month
```

Out[23]:
```
0        3
1        5
2        6
3        5
4        3
        ..
10678    4
10679    4
10680    4
```

```
10681    3
10682    5
Name: Date_of_Journey, Length: 10682, dtype: int64
```

In [19]:
```
#for explanation
pd.to_datetime(df.Date_of_Journey, format="%d/%m/%Y").dt.hour.unique() #no hour in our datetime so it will give 0
```

Out[19]:
```
array([0], dtype=int64)
```

In [18]:
```
# for explanation
pd.to_datetime(df.Date_of_Journey, format="%d/%m/%Y").dt.year.unique()
```

Out[18]:
```
array([2019], dtype=int64)
```

In [24]:
```
df["Journey_day"] = pd.to_datetime(df.Date_of_Journey, format="%d/%m/%Y").dt.day
```

In [25]:
```
df["Journey_month"] = pd.to_datetime(df["Date_of_Journey"], format = "%d/%m/%Y").dt.month
```

In [26]:
```
df.head()
```

Out[26]:

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Duration	Total_Stops	Additional_Info	Price	Journey_day
0	IndiGo	24/03/2019	Banglore	New Delhi	BLR → DEL	22:20	01:10 22 Mar	2h 50m	non-stop	No info	3897	24
1	Air India	1/05/2019	Kolkata	Banglore	CCU → IXR → BBI → BLR	05:50	13:15	7h 25m	2 stops	No info	7662	1
2	Jet Airways	9/06/2019	Delhi	Cochin	DEL → LKO → BOM → COK	09:25	04:25 10 Jun	19h	2 stops	No info	13882	9
3	IndiGo	12/05/2019	Kolkata	Banglore	CCU → NAG → BLR	18:05	23:30	5h 25m	1 stop	No info	6218	12
4	IndiGo	01/03/2019	Banglore	New Delhi	BLR → NAG → DEL	16:50	21:35	4h 45m	1 stop	No info	13302	1

so after we create two new column from date_of_journey , now we drop Date_of_Journey column from dataset

In [27]:
```
df.drop(["Date_of_Journey"], axis = 1, inplace = True)
```

same things we have do with Dep_time column , we create two new column Dep_hour and Dep_min from extract hour and min from Dep_Time

In [28]:
```
df["Dep_hour"] = pd.to_datetime(df["Dep_Time"]).dt.hour

df["Dep_min"] = pd.to_datetime(df["Dep_Time"]).dt.minute
```

```
df.drop(["Dep_Time"], axis = 1, inplace = True)
```

In [29]:
```
df.head()
```

Out[29]:

	Airline	Source	Destination	Route	Arrival_Time	Duration	Total_Stops	Additional_Info	Price	Journey_day	Journey_month	Dep_hour	I
0	IndiGo	Banglore	New Delhi	BLR → DEL	01:10 22 Mar	2h 50m	non-stop	No info	3897	24	3	22	
1	Air India	Kolkata	Banglore	CCU → IXR → BBI → BLR	13:15	7h 25m	2 stops	No info	7662	1	5	5	
2	Jet Airways	Delhi	Cochin	DEL → LKO → BOM → COK	04:25 10 Jun	19h	2 stops	No info	13882	9	6	9	
3	IndiGo	Kolkata	Banglore	CCU → NAG → BLR	23:30	5h 25m	1 stop	No info	6218	12	5	18	
4	IndiGo	Banglore	New Delhi	BLR → NAG → DEL	21:35	4h 45m	1 stop	No info	13302	1	3	16	

Similar to Date_of_Journey we can extract values from Arrival_Time

In [30]:
```
# Extracting Hours
df["Arrival_hour"] = pd.to_datetime(df.Arrival_Time).dt.hour

# Extracting Minutes
df["Arrival_min"] = pd.to_datetime(df.Arrival_Time).dt.minute

# Now we can drop Arrival_Time as it is of no use
df.drop(["Arrival_Time"], axis = 1, inplace = True)
```

In [31]:
```
df.head()
```

Out[31]:

	Airline	Source	Destination	Route	Duration	Total_Stops	Additional_Info	Price	Journey_day	Journey_month	Dep_hour	Dep_min	Arri
0	IndiGo	Banglore	New Delhi	BLR → DEL	2h 50m	non-stop	No info	3897	24	3	22	20	
1	Air India	Kolkata	Banglore	CCU → IXR → BBI → BLR	7h 25m	2 stops	No info	7662	1	5	5	50	
2	Jet Airways	Delhi	Cochin	DEL → LKO → BOM → COK	19h	2 stops	No info	13882	9	6	9	25	
3	IndiGo	Kolkata	Banglore	CCU → NAG → BLR	5h 25m	1 stop	No info	6218	12	5	18	5	
4	IndiGo	Banglore	New Delhi	BLR → NAG → DEL	4h 45m	1 stop	No info	13302	1	3	16	50	

check the all the values in Duration

In [32]:
```
duration = list(df["Duration"])
duration
```

Out[32]:
```
['2h 50m',
 '7h 25m',
 '19h',
 '5h 25m',
 '4h 45m',
 '2h 25m',
 '15h 30m',
 '21h 5m',
 '25h 30m',
 '7h 50m',
 '13h 15m',
 '2h 35m',
 '2h 15m',
 '12h 10m',
 '2h 35m',
 '26h 35m',
 '4h 30m',
 '22h 35m',
 '23h',
 '20h 35m',
 '5h 10m',
 '15h 20m',
 '2h 50m',
 '2h 55m',
 '13h 20m',
 '15h 10m',
 '5h 45m',
 '5h 55m',
 '2h 50m',
 '2h 15m',
 '2h 15m',
 '13h 25m',
 '2h 50m',
 '22h',
 '5h 30m',
 '10h 25m',
 '5h 15m',
 '2h 30m',
 '6h 15m',
 '11h 55m',
 '11h 5m',
 '8h 30m',
 '22h 5m',
 '2h 45m',
 '12h',
 '2h 50m',
 '2h 50m',
 '2h 15m',
 '16h 5m',
 '19h 55m',
 '3h 15m',
 '25h 20m',
 '2h 50m',
 '3h',
 '2h 50m',
 '16h 15m',
 '15h 5m',
 '2h 15m',
 '6h 30m',
 '25h 5m',
 '12h 25m',
 '27h 20m',
 '10h 15m',
 '10h 30m',
 '2h 15m',
 '10h 25m',
 '2h 50m',
 '1h 30m',
 '13h 20m',
 '2h 15m',
 '1h 25m',
 '26h 30m',
 '7h 20m',
 '13h 30m',
 '5h',
 '2h 45m',
```

```
'2h 50m',
'1h 30m',
'19h 5m',
'2h 15m',
'14h 50m',
'2h 40m',
'22h 10m',
'9h 35m',
'10h',
'21h 20m',
'5h 25m',
'18h 45m',
'12h 20m',
'18h',
'9h 15m',
'11h 5m',
'17h 30m',
'13h 20m',
'2h 35m',
'2h 25m',
'2h 40m',
'3h',
'1h 25m',
'16h 35m',
'12h 20m',
'12h 15m',
'7h 30m',
'24h',
'2h 45m',
'8h 55m',
'7h 10m',
'14h 30m',
'30h 20m',
'19h 55m',
'15h',
'2h 55m',
'12h 45m',
'10h 10m',
'5h 55m',
'15h 25m',
'16h 5m',
'10h 10m',
'2h 40m',
'11h 55m',
'2h 40m',
'1h 25m',
'14h 5m',
'20h 15m',
'2h 25m',
'23h 10m',
'18h 10m',
'16h',
'2h 50m',
'2h 20m',
'8h',
'16h 55m',
'3h 10m',
'2h 45m',
'14h',
'23h 50m',
'21h 40m',
'21h 15m',
'2h 45m',
'10h 50m',
'8h',
'8h 15m',
'5h 25m',
'8h 35m',
'11h 50m',
'27h 35m',
'8h 25m',
'20h 55m',
'4h 50m',
'8h 10m',
'2h 20m',
'24h 25m',
'2h 50m',
'12h 20m',
'12h 15m',
'23h 35m',
'16h 35m',
'2h 50m',
'25h 45m',
```

```
'26h 10m',
'28h 50m',
'2h 45m',
'25h 15m',
'9h 20m',
'4h 30m',
'15h 25m',
'1h 30m',
'2h 40m',
'9h 10m',
'2h 45m',
'22h 35m',
'3h 5m',
'11h 30m',
'9h 30m',
'12h 10m',
'17h 35m',
'5h 5m',
'25h 50m',
'20h',
'13h',
'18h 25m',
'1h 30m',
'9h 30m',
'16h',
'2h 20m',
'4h 30m',
'5h 45m',
'24h 10m',
'2h 35m',
'2h 45m',
'4h 55m',
'25h 35m',
'6h 20m',
'3h 5m',
'18h 40m',
'2h 15m',
'12h 10m',
'19h 25m',
'2h 50m',
'8h 25m',
'9h 15m',
'2h 50m',
'3h 15m',
'2h 50m',
'12h 20m',
'2h 55m',
'9h 35m',
'2h 35m',
'9h 30m',
'29h 20m',
'2h 55m',
'5h 15m',
'9h 5m',
'10h 45m',
'4h 55m',
'1h 30m',
'11h 40m',
'2h 45m',
'2h 55m',
'10h',
'2h 45m',
'10h 15m',
'1h 25m',
'9h 15m',
'22h 55m',
'28h 50m',
'37h 25m',
'2h 50m',
'1h 30m',
'1h 30m',
'25h 40m',
'2h 35m',
'6h 15m',
'17h 30m',
'5h 45m',
'2h 35m',
'25h 30m',
'13h 55m',
'2h 55m',
'10h 15m',
'8h 40m',
'5h 30m',
```

```
'12h',
'2h 15m',
'2h 15m',
'2h 50m',
'9h 10m',
'10h 25m',
'2h 20m',
'4h 30m',
'3h 10m',
'23h 30m',
'27h 20m',
'12h 35m',
'24h 15m',
'1h 20m',
'2h 45m',
'3h',
'11h',
'13h 30m',
'2h 50m',
'2h 55m',
'2h 45m',
'11h 15m',
'14h 35m',
'13h 55m',
'14h 5m',
'15h 5m',
'2h 15m',
'15h',
'12h 55m',
'9h',
'12h 15m',
'7h 40m',
'27h 20m',
'1h 25m',
'1h 20m',
'5h 30m',
'11h 45m',
'3h 15m',
'24h 55m',
'27h 20m',
'2h 55m',
'17h 5m',
'2h 40m',
'8h',
'3h',
'27h 35m',
'7h 30m',
'22h 55m',
'8h 35m',
'29h 55m',
'22h 15m',
'1h 20m',
'2h 40m',
'3h',
'12h 10m',
'2h 35m',
'11h 45m',
'2h 35m',
'1h 30m',
'2h 40m',
'2h 35m',
'22h 55m',
'14h 40m',
'15h 25m',
'7h 15m',
'20h 10m',
'20h 45m',
'2h 30m',
'23h 50m',
'27h',
'5h 30m',
'11h 50m',
'2h 15m',
'24h 30m',
'2h 20m',
'6h 30m',
'20h 25m',
'2h 35m',
'2h 55m',
'5h 35m',
'2h 30m',
'14h 45m',
'12h 20m',
```

```
'25h 30m',
'5h 35m',
'11h 45m',
'2h 55m',
'5h 40m',
'4h 5m',
'8h',
'2h 45m',
'2h 40m',
'2h 55m',
'15h 55m',
'23h 10m',
'7h 20m',
'2h 50m',
'7h 30m',
'3h',
'7h 45m',
'25h 20m',
'10h 25m',
'30h 20m',
'2h 35m',
'9h',
'15h 10m',
'28h 20m',
'3h',
'2h 45m',
'4h 55m',
'2h 40m',
'22h 10m',
'10h 10m',
'2h 50m',
'4h 20m',
'11h',
'3h',
'22h 35m',
'3h 5m',
'2h 40m',
'2h 50m',
'21h 5m',
'11h 50m',
'25h 35m',
'15h 25m',
'2h 25m',
'3h 40m',
'2h 55m',
'3h 10m',
'11h 50m',
'2h 50m',
'2h 50m',
'1h 30m',
'11h 30m',
'8h 50m',
'20h 15m',
'20h 10m',
'16h 5m',
'12h 35m',
'17h 5m',
'3h',
'15h 30m',
'11h 30m',
'2h 20m',
'23h 45m',
'1h 30m',
'2h 35m',
'24h 45m',
'21h 35m',
'5h 30m',
'12h 15m',
'8h 5m',
'5h',
'2h 50m',
'6h 25m',
'2h 20m',
'2h 45m',
'12h 45m',
'15h 50m',
'27h',
'26h 25m',
'2h 55m',
'24h 50m',
'8h 40m',
'25h 45m',
'2h 50m',
```

```
'6h 15m',
'10h 30m',
'26h',
'17h 5m',
'21h 20m',
'5h 35m',
'2h 50m',
'25h 50m',
'12h 10m',
'17h 5m',
'8h 30m',
'23h 5m',
'9h 20m',
'7h 55m',
'1h 30m',
'2h 50m',
'2h 15m',
'10h',
'2h 35m',
'7h 45m',
'2h 55m',
'26h 20m',
'2h 40m',
'11h',
'19h 25m',
'2h 55m',
'2h 45m',
'23h 15m',
'9h',
'3h 10m',
'6h 25m',
'8h 40m',
'2h 50m',
'12h 20m',
'5h 20m',
'4h',
'9h 45m',
'2h 55m',
'26h 20m',
'2h 15m',
'9h 45m',
'24h 55m',
'11h 55m',
'8h 20m',
'17h 25m',
'2h 50m',
'7h 5m',
'21h 15m',
'5h',
'2h 25m',
'5h 15m',
'15h 10m',
'11h 45m',
'3h',
'4h 55m',
'2h 45m',
'2h 30m',
'9h 45m',
'2h 40m',
'6h 30m',
'5h 30m',
'2h 40m',
'34h 5m',
'15h 25m',
'9h 30m',
'6h 5m',
'2h 40m',
'5h 50m',
'2h 45m',
'27h 20m',
'3h',
'5h 15m',
'16h',
'7h',
'8h 40m',
'9h',
'2h 20m',
'2h 20m',
'6h 20m',
'4h 25m',
'2h 20m',
'11h 5m',
'16h 55m',
```

```
'13h 45m',
'25h 45m',
'13h 55m',
'5h 15m',
'19h 15m',
'7h 30m',
'9h',
'24h',
'18h 10m',
'2h 55m',
'1h 30m',
'21h 20m',
'6h 15m',
'14h 30m',
'1h 20m',
'12h 55m',
'2h 25m',
'22h 30m',
'22h 55m',
'5h 25m',
'1h 25m',
'7h 15m',
'12h 20m',
'20h 10m',
'21h 15m',
'2h 15m',
'10h 30m',
'14h 5m',
'2h 50m',
'16h 25m',
'5h 15m',
'13h 50m',
'2h 15m',
'7h 15m',
'27h 5m',
'27h 35m',
'5h',
'2h 50m',
'2h 50m',
'15h 20m',
'5h 20m',
'10h 10m',
'27h 35m',
'9h 35m',
'28h 10m',
'14h',
'2h 35m',
'2h 55m',
'2h 35m',
'4h 40m',
'2h 30m',
'11h 15m',
'7h 55m',
'8h 50m',
'1h 25m',
'15h 40m',
'2h 35m',
'2h 55m',
'13h 25m',
'4h 35m',
'18h 30m',
'2h 15m',
'38h 15m',
'3h 5m',
'25h 50m',
'3h 10m',
'2h 50m',
'11h 30m',
'6h 35m',
'27h 35m',
'2h 35m',
'7h 30m',
'3h 15m',
'9h 35m',
'12h 30m',
'11h 20m',
'3h',
'2h 35m',
'1h 30m',
'26h 10m',
'4h 55m',
'2h 45m',
'8h 40m',
```

```
'7h 5m',
'2h 45m',
'24h 50m',
'2h 20m',
'11h',
'9h',
'2h 55m',
'3h 5m',
'2h 35m',
'13h',
'1h 25m',
'7h 35m',
'2h 40m',
'12h 55m',
'29h 20m',
'5h',
'8h 40m',
'3h',
'3h',
'2h 45m',
'12h 10m',
'2h 20m',
'7h 55m',
'29h 35m',
'26h 55m',
'12h 45m',
'7h 35m',
'2h 30m',
'11h 45m',
'23h 40m',
'13h 50m',
'12h 50m',
'9h 50m',
'2h 30m',
'21h 55m',
'26h 10m',
'2h 55m',
'20h 15m',
'20h 10m',
'10h 55m',
'2h 30m',
'21h 10m',
'2h 35m',
'20h 40m',
'2h 50m',
'6h 30m',
'30h',
'13h 10m',
'2h 55m',
'18h',
'8h',
'8h 45m',
'12h 30m',
'13h 20m',
'6h 10m',
'7h 30m',
'19h 15m',
'2h 35m',
'12h',
'3h',
'2h 50m',
'1h 25m',
'23h 5m',
'2h 50m',
'22h 5m',
'2h 15m',
'5h 45m',
'2h 50m',
'9h 30m',
'8h 55m',
'17h 45m',
'2h 50m',
'21h 45m',
'12h 35m',
'1h 30m',
'2h 15m',
'3h 55m',
'2h 45m',
'7h 30m',
'17h 20m',
'1h 30m',
'1h 30m',
'30h 30m',
```

```
'5h',
'11h 15m',
'2h 50m',
'19h 5m',
'6h 35m',
'7h 30m',
'2h 15m',
'2h 45m',
'12h 45m',
'2h 50m',
'26h 35m',
'12h 55m',
'1h 30m',
'8h 40m',
'14h',
'23h 30m',
'29h 55m',
'22h 10m',
'4h 50m',
'2h 45m',
'13h 30m',
'13h 20m',
'2h 20m',
'11h 55m',
'5h 10m',
'11h 45m',
'13h 50m',
'3h',
'6h 10m',
'2h 20m',
'10h 45m',
'2h 15m',
'2h 55m',
'1h 30m',
'7h 35m',
'26h 35m',
'4h 45m',
'21h 25m',
'8h 35m',
'8h 30m',
'2h 15m',
'12h 40m',
'8h 25m',
'25h 40m',
'2h 45m',
'29h 20m',
'2h 40m',
'2h 50m',
'2h 20m',
'7h 15m',
'20h 15m',
'11h 45m',
'2h 35m',
'13h 30m',
'1h 30m',
'24h 35m',
'20h 35m',
'2h 50m',
'4h 35m',
'2h 20m',
'11h 45m',
'1h 30m',
'12h 50m',
'24h 30m',
'11h 30m',
'4h',
'11h',
'5h 25m',
'2h 35m',
'5h 10m',
'10h 10m',
'26h 55m',
'7h 15m',
'8h 10m',
'2h 55m',
'5h 15m',
'2h 20m',
'1h 25m',
'4h 50m',
'2h 20m',
'2h 20m',
'13h 15m',
'19h 10m',
```

```
'22h 40m',
'14h 50m',
'15h 10m',
'7h 5m',
'1h 25m',
'2h 55m',
'8h',
'2h 45m',
'14h 55m',
'1h 30m',
'16h 15m',
'2h 50m',
'5h',
'22h 15m',
'5h 30m',
'8h 35m',
'15h 25m',
'21h',
'13h 30m',
'2h 50m',
'6h 45m',
'1h 30m',
'8h 40m',
'1h 30m',
'2h 50m',
'2h 35m',
'16h 5m',
'23h 50m',
'27h 5m',
'16h 15m',
'15h 25m',
'2h 50m',
'8h 35m',
'13h 50m',
'2h 40m',
'2h 25m',
'3h 10m',
'2h 55m',
'28h 40m',
'2h 50m',
'2h 35m',
'9h 40m',
'16h 40m',
'6h 20m',
'27h',
'14h 35m',
'12h 10m',
'7h 35m',
'2h 20m',
'19h 10m',
'20h',
'3h',
'5h 10m',
'2h 40m',
'29h 35m',
'4h 55m',
'3h',
'12h 30m',
'12h 35m',
'26h 20m',
'13h 15m',
'2h 50m',
'12h 45m',
'16h 20m',
'2h 40m',
'8h 15m',
'12h',
'2h 50m',
'11h',
'2h 20m',
'13h 50m',
'8h 10m',
'12h 10m',
'2h 40m',
'2h 45m',
'2h 50m',
'24h',
'3h',
'12h 35m',
'25h 50m',
'2h 55m',
'17h 5m',
'29h 20m',
```

```
'8h 55m',
'3h 5m',
'9h',
'10h 25m',
'16h 45m',
'2h 45m',
'26h 20m',
'13h 15m',
'10h 30m',
'7h 35m',
'11h',
'2h 50m',
'25h 45m',
'6h 25m',
'1h 25m',
'20h 10m',
'11h 55m',
'2h 50m',
'5h',
'27h',
'2h 50m',
'1h 15m',
'10h 15m',
'24h 50m',
'2h 50m',
'20h',
'5h 30m',
'2h 45m',
'6h 55m',
'2h 45m',
'5h 15m',
'6h 15m',
'2h 50m',
'19h 25m',
'2h 55m',
'11h 25m',
'14h 5m',
'2h 45m',
'3h 15m',
'14h 20m',
'12h 30m',
'5h 20m',
'26h 20m',
'3h 5m',
'1h 30m',
'7h 5m',
'6h 10m',
'8h',
'12h 5m',
'12h 55m',
'9h 35m',
'15h 10m',
'11h 40m',
'2h 55m',
'4h 30m',
'2h 40m',
'2h 30m',
'24h 10m',
'28h 20m',
'1h 25m',
'10h 10m',
'11h 30m',
'2h 50m',
'13h 45m',
'6h 25m',
'12h 55m',
'4h 25m',
'13h 30m',
'24h 5m',
'16h 5m',
'8h 20m',
'3h 40m',
'9h 30m',
'28h 15m',
'13h 20m',
'17h 50m',
'20h 20m',
'28h 5m',
'15h 25m',
'12h 5m',
'10h 20m',
'12h 25m',
'7h 35m',
```

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

```
'8h 40m',
'10h 25m',
'35h 5m',
'14h 15m',
'3h',
'25h 50m',
'2h 30m',
'8h 40m',
'10h',
'11h 30m',
'7h 30m',
...]
```

now create loop for check duration contains only hour min and if yes add min or hour in it

In [50]:
```
for i in range(len(duration)):
    if len(duration[i].split()) != 2:    # Check if duration contains only hour or mins
        if "h" in duration[i]:
            duration[i] = duration[i].strip() + " 0m"   # Adds 0 minute
        else:
            duration[i] = "0h " + duration[i]           # Adds 0 hour
```

In [36]:
```
# for explanation
len(duration)
```

Out[36]: 10682

In [37]:
```
# for explanation
range(len(duration))
```

Out[37]: range(0, 10682)

In [38]:
```
'Akash Govil'.split() # if we dont give any separator in split then by default it separates at space.
```

Out[38]: ['Akash', 'Govil']

In [39]:
```
'Akash_Govil'.split('_')
```

Out[39]: ['Akash', 'Govil']

In [41]:
```
'Akash_Govil'.split() # here no space and also we didnt gave any separator so it kep as it is
```

Out[41]: ['Akash_Govil']

In [40]:
```
'Akash Govil'.split()[0]
```

Out[40]: 'Akash'

In [42]:
```
duration[1] # i is from 0 to 10682
```

Out[42]: '7h 25m'

In [43]:
```
duration[1].split()
```

Out[43]: ['7h', '25m']

In [44]:
```
len(duration[1].split())
```

Out[44]: 2

In [45]:
```
len(duration[2].split()) # so here something of hour or minute is missing since it gives 1 i.e !=2 (explnation of
```

Out[45]: 1

In [46]:
```
'h' in duration[2] # explantion of line 3
```

Out[46]: True

In [47]:
```
'h '.strip() #explanation of line 4. removes unwanted white space. now we add ' 0m'(space before 0)
```

Out[47]: 'h'

In [48]:
```
# e.g.
'Akash'+'Govil'
```

Out[48]: 'AkashGovil'

In [49]:
```
'Akash'+' Govil'
```

Out[49]: 'Akash Govil'

In [51]:
```
duration
```

Out[51]:
```
['2h 50m',
 '7h 25m',
 '19h 0m',
 '5h 25m',
 '4h 45m',
 '2h 25m',
 '15h 30m',
 '21h 5m',
 '25h 30m',
 '7h 50m',
 '13h 15m',
 '2h 35m',
 '2h 15m',
 '12h 10m',
 '2h 35m',
 '26h 35m',
 '4h 30m',
 '22h 35m',
 '23h 0m',
 '20h 35m',
 '5h 10m',
 '15h 20m',
 '2h 50m',
 '2h 55m',
 '13h 20m',
 '15h 10m',
 '5h 45m',
 '5h 55m',
 '2h 50m',
 '2h 15m',
```

```
'2h 15m',
'13h 25m',
'2h 50m',
'22h 0m',
'5h 30m',
'10h 25m',
'5h 15m',
'2h 30m',
'6h 15m',
'11h 55m',
'11h 5m',
'8h 30m',
'22h 5m',
'2h 45m',
'12h 0m',
'2h 50m',
'2h 50m',
'2h 15m',
'16h 5m',
'19h 55m',
'3h 15m',
'25h 20m',
'2h 50m',
'3h 0m',
'2h 50m',
'16h 15m',
'15h 5m',
'2h 15m',
'6h 30m',
'25h 5m',
'12h 25m',
'27h 20m',
'10h 15m',
'10h 30m',
'2h 15m',
'10h 25m',
'2h 50m',
'1h 30m',
'13h 20m',
'2h 15m',
'1h 25m',
'26h 30m',
'7h 20m',
'13h 30m',
'5h 0m',
'2h 45m',
'2h 50m',
'1h 30m',
'19h 5m',
'2h 15m',
'14h 50m',
'2h 40m',
'22h 10m',
'9h 35m',
'10h 0m',
'21h 20m',
'5h 25m',
'18h 45m',
'12h 20m',
'18h 0m',
'9h 15m',
'11h 5m',
'17h 30m',
'13h 20m',
'2h 35m',
'2h 25m',
'2h 40m',
'3h 0m',
'1h 25m',
'16h 35m',
'12h 20m',
'12h 15m',
'7h 30m',
'24h 0m',
'2h 45m',
'8h 55m',
'7h 10m',
'14h 30m',
'30h 20m',
'19h 55m',
'15h 0m',
'2h 55m',
'12h 45m',
```

```
'10h 10m',
'5h 55m',
'15h 25m',
'16h 5m',
'10h 10m',
'2h 40m',
'11h 55m',
'2h 40m',
'1h 25m',
'14h 5m',
'20h 15m',
'2h 25m',
'23h 10m',
'18h 10m',
'16h 0m',
'2h 50m',
'2h 20m',
'8h 0m',
'16h 55m',
'3h 10m',
'2h 45m',
'14h 0m',
'23h 50m',
'21h 40m',
'21h 15m',
'2h 45m',
'10h 50m',
'8h 0m',
'8h 15m',
'5h 25m',
'8h 35m',
'11h 50m',
'27h 35m',
'8h 25m',
'20h 55m',
'4h 50m',
'8h 10m',
'2h 20m',
'24h 25m',
'2h 50m',
'12h 20m',
'12h 15m',
'23h 35m',
'16h 35m',
'2h 50m',
'25h 45m',
'26h 10m',
'28h 50m',
'2h 45m',
'25h 15m',
'9h 20m',
'4h 30m',
'15h 25m',
'1h 30m',
'2h 40m',
'9h 10m',
'2h 45m',
'22h 35m',
'3h 5m',
'11h 30m',
'9h 30m',
'12h 10m',
'17h 35m',
'5h 5m',
'25h 50m',
'20h 0m',
'13h 0m',
'18h 25m',
'1h 30m',
'9h 30m',
'16h 0m',
'2h 20m',
'4h 30m',
'5h 45m',
'24h 10m',
'2h 35m',
'2h 45m',
'4h 55m',
'25h 35m',
'6h 20m',
'3h 5m',
'18h 40m',
'2h 15m',
```

```
'12h 10m',
'19h 25m',
'2h 50m',
'8h 25m',
'9h 15m',
'2h 50m',
'3h 15m',
'2h 50m',
'12h 20m',
'2h 55m',
'9h 35m',
'2h 35m',
'9h 30m',
'29h 20m',
'2h 55m',
'5h 15m',
'9h 5m',
'10h 45m',
'4h 55m',
'1h 30m',
'11h 40m',
'2h 45m',
'2h 55m',
'10h 0m',
'2h 45m',
'10h 15m',
'1h 25m',
'9h 15m',
'22h 55m',
'28h 50m',
'37h 25m',
'2h 50m',
'1h 30m',
'1h 30m',
'25h 40m',
'2h 35m',
'6h 15m',
'17h 30m',
'5h 45m',
'2h 35m',
'25h 30m',
'13h 55m',
'2h 55m',
'10h 15m',
'8h 40m',
'5h 30m',
'12h 0m',
'2h 15m',
'2h 15m',
'2h 50m',
'9h 10m',
'10h 25m',
'2h 20m',
'4h 30m',
'3h 10m',
'23h 30m',
'27h 20m',
'12h 35m',
'24h 15m',
'1h 20m',
'2h 45m',
'3h 0m',
'11h 0m',
'13h 30m',
'2h 50m',
'2h 55m',
'2h 45m',
'11h 15m',
'14h 35m',
'13h 55m',
'14h 5m',
'15h 5m',
'2h 15m',
'15h 0m',
'12h 55m',
'9h 0m',
'12h 15m',
'7h 40m',
'27h 20m',
'1h 25m',
'1h 20m',
'5h 30m',
'11h 45m',
```

```
'3h 15m',
'24h 55m',
'27h 20m',
'2h 55m',
'17h 5m',
'2h 40m',
'8h 0m',
'3h 0m',
'27h 35m',
'7h 30m',
'22h 55m',
'8h 35m',
'29h 55m',
'22h 15m',
'1h 20m',
'2h 40m',
'3h 0m',
'12h 10m',
'2h 35m',
'11h 45m',
'2h 35m',
'1h 30m',
'2h 40m',
'2h 35m',
'22h 55m',
'14h 40m',
'15h 25m',
'7h 15m',
'20h 10m',
'20h 45m',
'2h 30m',
'23h 50m',
'27h 0m',
'5h 30m',
'11h 50m',
'2h 15m',
'24h 30m',
'2h 20m',
'6h 30m',
'20h 25m',
'2h 35m',
'2h 55m',
'5h 35m',
'2h 30m',
'14h 45m',
'12h 20m',
'25h 30m',
'5h 35m',
'11h 45m',
'2h 55m',
'5h 40m',
'4h 5m',
'8h 0m',
'2h 45m',
'2h 40m',
'2h 55m',
'15h 55m',
'23h 10m',
'7h 20m',
'2h 50m',
'7h 30m',
'3h 0m',
'7h 45m',
'25h 20m',
'10h 25m',
'30h 20m',
'2h 35m',
'9h 0m',
'15h 10m',
'28h 20m',
'3h 0m',
'2h 45m',
'4h 55m',
'2h 40m',
'22h 10m',
'10h 10m',
'2h 50m',
'4h 20m',
'11h 0m',
'3h 0m',
'22h 35m',
'3h 5m',
'2h 40m',
```

```
'2h 50m',
'21h 5m',
'11h 50m',
'25h 35m',
'15h 25m',
'2h 25m',
'3h 40m',
'2h 55m',
'3h 10m',
'11h 50m',
'2h 50m',
'2h 50m',
'1h 30m',
'11h 30m',
'8h 50m',
'20h 15m',
'20h 10m',
'16h 5m',
'12h 35m',
'17h 5m',
'3h 0m',
'15h 30m',
'11h 30m',
'2h 20m',
'23h 45m',
'1h 30m',
'2h 35m',
'24h 45m',
'21h 35m',
'5h 30m',
'12h 15m',
'8h 5m',
'5h 0m',
'2h 50m',
'6h 25m',
'2h 20m',
'2h 45m',
'12h 45m',
'15h 50m',
'27h 0m',
'26h 25m',
'2h 55m',
'24h 50m',
'8h 40m',
'25h 45m',
'2h 50m',
'6h 15m',
'10h 30m',
'26h 0m',
'17h 5m',
'21h 20m',
'5h 35m',
'2h 50m',
'25h 50m',
'12h 10m',
'17h 5m',
'8h 30m',
'23h 5m',
'9h 20m',
'7h 55m',
'1h 30m',
'2h 50m',
'2h 15m',
'10h 0m',
'2h 35m',
'7h 45m',
'2h 55m',
'26h 20m',
'2h 40m',
'11h 0m',
'19h 25m',
'2h 55m',
'2h 45m',
'23h 15m',
'9h 0m',
'3h 10m',
'6h 25m',
'8h 40m',
'2h 50m',
'12h 20m',
'5h 20m',
'4h 0m',
'9h 45m',
```

```
'2h 55m',
'26h 20m',
'2h 15m',
'9h 45m',
'24h 55m',
'11h 55m',
'8h 20m',
'17h 25m',
'2h 50m',
'7h 5m',
'21h 15m',
'5h 0m',
'2h 25m',
'5h 15m',
'15h 10m',
'11h 45m',
'3h 0m',
'4h 55m',
'2h 45m',
'2h 30m',
'9h 45m',
'2h 40m',
'6h 30m',
'5h 30m',
'2h 40m',
'34h 5m',
'15h 25m',
'9h 30m',
'6h 5m',
'2h 40m',
'5h 50m',
'2h 45m',
'27h 20m',
'3h 0m',
'5h 15m',
'16h 0m',
'7h 0m',
'8h 40m',
'9h 0m',
'2h 20m',
'2h 20m',
'6h 20m',
'4h 25m',
'2h 20m',
'11h 5m',
'16h 55m',
'13h 45m',
'25h 45m',
'13h 55m',
'5h 15m',
'19h 15m',
'7h 30m',
'9h 0m',
'24h 0m',
'18h 10m',
'2h 55m',
'1h 30m',
'21h 20m',
'6h 15m',
'14h 30m',
'1h 20m',
'12h 55m',
'2h 25m',
'22h 30m',
'22h 55m',
'5h 25m',
'1h 25m',
'7h 15m',
'12h 20m',
'20h 10m',
'21h 15m',
'2h 15m',
'10h 30m',
'14h 5m',
'2h 50m',
'16h 25m',
'5h 15m',
'13h 50m',
'2h 15m',
'7h 15m',
'27h 5m',
'27h 35m',
'5h 0m',
```

```
'2h 50m',
'2h 50m',
'15h 20m',
'5h 20m',
'10h 10m',
'27h 35m',
'9h 35m',
'28h 10m',
'14h 0m',
'2h 35m',
'2h 55m',
'2h 35m',
'4h 40m',
'2h 30m',
'11h 15m',
'7h 55m',
'8h 50m',
'1h 25m',
'15h 40m',
'2h 35m',
'2h 55m',
'13h 25m',
'4h 35m',
'18h 30m',
'2h 15m',
'38h 15m',
'3h 5m',
'25h 50m',
'3h 10m',
'2h 50m',
'11h 30m',
'6h 35m',
'27h 35m',
'2h 35m',
'7h 30m',
'3h 15m',
'9h 35m',
'12h 30m',
'11h 20m',
'3h 0m',
'2h 35m',
'1h 30m',
'26h 10m',
'4h 55m',
'2h 45m',
'8h 40m',
'7h 5m',
'2h 45m',
'24h 50m',
'2h 20m',
'11h 0m',
'9h 0m',
'2h 55m',
'3h 5m',
'2h 35m',
'13h 0m',
'1h 25m',
'7h 35m',
'2h 40m',
'12h 55m',
'29h 20m',
'5h 0m',
'8h 40m',
'3h 0m',
'3h 0m',
'2h 45m',
'12h 10m',
'2h 20m',
'7h 55m',
'29h 35m',
'26h 55m',
'12h 45m',
'7h 35m',
'2h 30m',
'11h 45m',
'23h 40m',
'13h 50m',
'12h 50m',
'9h 50m',
'2h 30m',
'21h 55m',
'26h 10m',
'2h 55m',
```

```
'20h 15m',
'20h 10m',
'10h 55m',
'2h 30m',
'21h 10m',
'2h 35m',
'20h 40m',
'2h 50m',
'6h 30m',
'30h 0m',
'13h 10m',
'2h 55m',
'18h 0m',
'8h 0m',
'8h 45m',
'12h 30m',
'13h 20m',
'6h 10m',
'7h 30m',
'19h 15m',
'2h 35m',
'12h 0m',
'3h 0m',
'2h 50m',
'1h 25m',
'23h 5m',
'2h 50m',
'22h 5m',
'2h 15m',
'5h 45m',
'2h 50m',
'9h 30m',
'8h 55m',
'17h 45m',
'2h 50m',
'21h 45m',
'12h 35m',
'1h 30m',
'2h 15m',
'3h 55m',
'2h 45m',
'7h 30m',
'17h 20m',
'1h 30m',
'1h 30m',
'30h 30m',
'5h 0m',
'11h 15m',
'2h 50m',
'19h 5m',
'6h 35m',
'7h 30m',
'2h 15m',
'2h 45m',
'12h 45m',
'2h 50m',
'26h 35m',
'12h 55m',
'1h 30m',
'8h 40m',
'14h 0m',
'23h 30m',
'29h 55m',
'22h 10m',
'4h 50m',
'2h 45m',
'13h 30m',
'13h 20m',
'2h 20m',
'11h 55m',
'5h 10m',
'11h 45m',
'13h 50m',
'3h 0m',
'6h 10m',
'2h 20m',
'10h 45m',
'2h 15m',
'2h 55m',
'1h 30m',
'7h 35m',
'26h 35m',
'4h 45m',
```

```
'21h 25m',
'8h 35m',
'8h 30m',
'2h 15m',
'12h 40m',
'8h 25m',
'25h 40m',
'2h 45m',
'29h 20m',
'2h 40m',
'2h 50m',
'2h 20m',
'7h 15m',
'20h 15m',
'11h 45m',
'2h 35m',
'13h 30m',
'1h 30m',
'24h 35m',
'20h 35m',
'2h 50m',
'4h 35m',
'2h 20m',
'11h 45m',
'1h 30m',
'12h 50m',
'24h 30m',
'11h 30m',
'4h 0m',
'11h 0m',
'5h 25m',
'2h 35m',
'5h 10m',
'10h 10m',
'26h 55m',
'7h 15m',
'8h 10m',
'2h 55m',
'5h 15m',
'2h 20m',
'1h 25m',
'4h 50m',
'2h 20m',
'2h 20m',
'13h 15m',
'19h 10m',
'22h 40m',
'14h 50m',
'15h 10m',
'7h 5m',
'1h 25m',
'2h 55m',
'8h 0m',
'2h 45m',
'14h 55m',
'1h 30m',
'16h 15m',
'2h 50m',
'5h 0m',
'22h 15m',
'5h 30m',
'8h 35m',
'15h 25m',
'21h 0m',
'13h 30m',
'2h 50m',
'6h 45m',
'1h 30m',
'8h 40m',
'1h 30m',
'2h 50m',
'2h 35m',
'16h 5m',
'23h 50m',
'27h 5m',
'16h 15m',
'15h 25m',
'2h 50m',
'8h 35m',
'13h 50m',
'2h 40m',
'2h 25m',
'3h 10m',
```

```
'2h 55m',
'28h 40m',
'2h 50m',
'2h 35m',
'9h 40m',
'16h 40m',
'6h 20m',
'27h 0m',
'14h 35m',
'12h 10m',
'7h 35m',
'2h 20m',
'19h 10m',
'20h 0m',
'3h 0m',
'5h 10m',
'2h 40m',
'29h 35m',
'4h 55m',
'3h 0m',
'12h 30m',
'12h 35m',
'26h 20m',
'13h 15m',
'2h 50m',
'12h 45m',
'16h 20m',
'2h 40m',
'8h 15m',
'12h 0m',
'2h 50m',
'11h 0m',
'2h 20m',
'13h 50m',
'8h 10m',
'12h 10m',
'2h 40m',
'2h 45m',
'2h 50m',
'24h 0m',
'3h 0m',
'12h 35m',
'25h 50m',
'2h 55m',
'17h 5m',
'29h 20m',
'8h 55m',
'3h 5m',
'9h 0m',
'10h 25m',
'16h 45m',
'2h 45m',
'26h 20m',
'13h 15m',
'10h 30m',
'7h 35m',
'11h 0m',
'2h 50m',
'25h 45m',
'6h 25m',
'1h 25m',
'20h 10m',
'11h 55m',
'2h 50m',
'5h 0m',
'27h 0m',
'2h 50m',
'1h 15m',
'10h 15m',
'24h 50m',
'2h 50m',
'20h 0m',
'5h 30m',
'2h 45m',
'6h 55m',
'2h 45m',
'5h 15m',
'6h 15m',
'2h 50m',
'19h 25m',
'2h 55m',
'11h 25m',
'14h 5m',
```

```
'2h 45m',
'3h 15m',
'14h 20m',
'12h 30m',
'5h 20m',
'26h 20m',
'3h 5m',
'1h 30m',
'7h 5m',
'6h 10m',
'8h 0m',
'12h 5m',
'12h 55m',
'9h 35m',
'15h 10m',
'11h 40m',
'2h 55m',
'4h 30m',
'2h 40m',
'2h 30m',
'24h 10m',
'28h 20m',
'1h 25m',
'10h 10m',
'11h 30m',
'2h 50m',
'13h 45m',
'6h 25m',
'12h 55m',
'4h 25m',
'13h 30m',
'24h 5m',
'16h 5m',
'8h 20m',
'3h 40m',
'9h 30m',
'28h 15m',
'13h 20m',
'17h 50m',
'20h 20m',
'28h 5m',
'15h 25m',
'12h 5m',
'10h 20m',
'12h 25m',
'7h 35m',
'2h 50m',
'4h 35m',
'11h 15m',
'2h 35m',
'26h 0m',
'2h 25m',
'2h 35m',
'14h 15m',
'2h 35m',
'22h 5m',
'6h 35m',
'8h 30m',
'18h 45m',
'35h 15m',
'2h 15m',
'2h 30m',
'2h 20m',
'2h 30m',
'7h 35m',
'35h 35m',
'26h 30m',
'11h 30m',
'26h 40m',
'3h 0m',
'1h 30m',
'2h 55m',
'14h 15m',
'16h 55m',
'12h 20m',
'29h 35m',
'20h 20m',
'8h 25m',
'13h 50m',
'10h 0m',
'19h 25m',
'28h 0m',
'2h 50m',
```

```
'17h 20m',
'2h 45m',
'2h 55m',
'13h 25m',
'12h 55m',
'9h 45m',
'21h 25m',
'15h 40m',
'2h 20m',
'14h 25m',
'2h 50m',
'6h 35m',
'22h 10m',
'14h 25m',
'2h 30m',
'23h 35m',
'13h 5m',
'1h 30m',
'5h 5m',
'1h 20m',
'22h 55m',
'1h 30m',
'2h 35m',
'37h 20m',
'2h 50m',
'2h 50m',
'36h 10m',
'9h 15m',
'1h 25m',
'5h 5m',
'2h 15m',
'15h 10m',
'25h 55m',
'5h 40m',
'7h 30m',
'17h 45m',
'2h 20m',
'8h 15m',
'3h 40m',
'5h 55m',
'2h 30m',
'17h 30m',
'13h 50m',
'2h 15m',
'21h 35m',
'9h 30m',
'8h 40m',
'10h 25m',
'35h 5m',
'14h 15m',
'3h 0m',
'25h 50m',
'2h 30m',
'8h 40m',
'10h 0m',
'11h 30m',
'7h 30m',
...]
```

now Extract hour and min from duration column and create two new column Duration_hours & Duration_mins

In [63]:
```
duration_hours = []
duration_mins = []
for i in range(len(duration)):
    duration_hours.append(int(duration[i].split("h")[0]))    # Extract value of hours from duration
    duration_mins.append(int(duration[i].split("m")[0].split()[-1]))   # Extracts only value of minutes from dura
```

In [52]:
```
# for explanation of above cell
duration[0].split('h')
```

Out[52]: ['2', ' 50m']

In [ ]:

In [53]:
```
# for explanation
duration[0].split('h')[0] # then int is used ahead of this to convert the value into integer.
```

Out[53]: '2'

In [54]:
```
int(duration[0].split('h')[0])
```

Out[54]: 2

In [62]:
```
duration[0].split('m')
```

Out[62]: ['2h 50', '']

In [56]:
```
duration[0].split('m')[0]
```

Out[56]: '2h 50'

In [57]:
```
duration[0].split('m')[0].split()[-1] # or duration[0].split('m')[0].split()[1]
```

Out[57]: '50'

In [58]:
```
import re # regular expression
```

In [59]:
```
re.sub(' ','-','Akash Govil')
```

Out[59]: 'Akash-Govil'

In [60]:
```
re.sub('h',' ',duration[0])
```

Out[60]: '2 50m'

In [61]:
```
re.sub('h',' ',duration[0]).split()[0]
```

Out[61]: '2'

In [64]:
```
duration_hours
```

Out[64]:
```
[2,
 7,
 19,
 5,
 4,
 2,
 15,
 21,
 25,
 7,
 13,
 2,
 2,
 12,
 2,
 26,
 4,
 22,
 23,
```

```
20,
5,
15,
2,
2,
13,
15,
5,
5,
2,
2,
2,
13,
2,
22,
5,
10,
5,
2,
6,
11,
11,
8,
22,
2,
12,
2,
2,
2,
16,
19,
3,
25,
2,
3,
2,
16,
15,
2,
6,
25,
12,
27,
10,
10,
2,
10,
2,
1,
13,
2,
1,
26,
7,
13,
5,
2,
2,
1,
19,
2,
14,
2,
22,
9,
10,
21,
5,
18,
12,
18,
9,
11,
17,
13,
2,
2,
2,
3,
1,
16,
12,
12,
```

7,
24,
2,
8,
7,
14,
30,
19,
15,
2,
12,
10,
5,
15,
16,
10,
2,
11,
2,
1,
14,
20,
2,
23,
18,
16,
2,
2,
8,
16,
3,
2,
14,
23,
21,
21,
2,
10,
8,
8,
5,
8,
11,
27,
8,
20,
4,
8,
2,
24,
2,
12,
12,
23,
16,
2,
25,
26,
28,
2,
25,
9,
4,
15,
1,
2,
9,
2,
22,
3,
11,
9,
12,
17,
5,
25,
20,
13,
18,
1,
9,
16,
2,

4,
5,
24,
2,
2,
4,
25,
6,
3,
18,
2,
12,
19,
2,
8,
9,
2,
3,
2,
12,
2,
9,
2,
9,
29,
2,
5,
9,
10,
4,
1,
11,
2,
2,
10,
2,
10,
1,
9,
22,
28,
37,
2,
1,
1,
25,
2,
6,
17,
5,
2,
25,
13,
2,
10,
8,
5,
12,
2,
2,
2,
9,
10,
2,
4,
3,
23,
27,
12,
24,
1,
2,
3,
11,
13,
2,
2,
2,
11,
14,
13,
14,
15,

2,
15,
12,
9,
12,
7,
27,
1,
1,
5,
11,
3,
24,
27,
2,
17,
2,
8,
3,
27,
7,
22,
8,
29,
22,
1,
2,
3,
12,
2,
11,
2,
1,
2,
2,
22,
14,
15,
7,
20,
20,
2,
23,
27,
5,
11,
2,
24,
2,
6,
20,
2,
2,
5,
2,
14,
12,
25,
5,
11,
2,
5,
4,
8,
2,
2,
2,
15,
23,
7,
2,
7,
3,
7,
25,
10,
30,
2,
9,
15,
28,
3,
2,

4,
2,
22,
10,
2,
4,
11,
3,
22,
3,
2,
2,
21,
11,
25,
15,
2,
3,
2,
3,
11,
2,
2,
1,
11,
8,
20,
20,
16,
12,
17,
3,
15,
11,
2,
23,
1,
2,
24,
21,
5,
12,
8,
5,
2,
6,
2,
2,
12,
15,
27,
26,
2,
24,
8,
25,
2,
6,
10,
26,
17,
21,
5,
2,
25,
12,
17,
8,
23,
9,
7,
1,
2,
2,
10,
2,
7,
2,
26,
2,
11,
19,
2,

2,
23,
9,
3,
6,
8,
2,
12,
5,
4,
9,
2,
26,
2,
9,
24,
11,
8,
17,
2,
7,
21,
5,
2,
5,
15,
11,
3,
4,
2,
2,
9,
2,
6,
5,
2,
34,
15,
9,
6,
2,
5,
2,
27,
3,
5,
16,
7,
8,
9,
2,
2,
6,
4,
2,
11,
16,
13,
25,
13,
5,
19,
7,
9,
24,
18,
2,
1,
21,
6,
14,
1,
12,
2,
22,
22,
5,
1,
7,
12,
20,
21,
2,

10,
14,
2,
16,
5,
13,
2,
7,
27,
27,
5,
2,
2,
15,
5,
10,
27,
9,
28,
14,
2,
2,
2,
4,
2,
11,
7,
8,
1,
15,
2,
2,
13,
4,
18,
2,
38,
3,
25,
3,
2,
11,
6,
27,
2,
7,
3,
9,
12,
11,
3,
2,
1,
26,
4,
2,
8,
7,
2,
24,
2,
11,
9,
2,
3,
2,
13,
1,
7,
2,
12,
29,
5,
8,
3,
3,
2,
12,
2,
7,
29,
26,
12,

7,
2,
11,
23,
13,
12,
9,
2,
21,
26,
2,
20,
20,
10,
2,
21,
2,
20,
2,
6,
30,
13,
2,
18,
8,
8,
12,
13,
6,
7,
19,
2,
12,
3,
2,
1,
23,
2,
22,
2,
5,
2,
9,
8,
17,
2,
21,
12,
1,
2,
3,
2,
7,
17,
1,
1,
30,
5,
11,
2,
19,
6,
7,
2,
2,
12,
2,
26,
12,
1,
8,
14,
23,
29,
22,
4,
2,
13,
13,
2,
11,
5,
11,

13,
3,
6,
2,
10,
2,
2,
1,
7,
26,
4,
21,
8,
8,
2,
12,
8,
25,
2,
29,
2,
2,
2,
7,
20,
11,
2,
13,
1,
24,
20,
2,
4,
2,
11,
1,
12,
24,
11,
4,
11,
5,
2,
5,
10,
26,
7,
8,
2,
5,
2,
1,
4,
2,
2,
13,
19,
22,
14,
15,
7,
1,
2,
8,
2,
14,
1,
16,
2,
5,
22,
5,
8,
15,
21,
13,
2,
6,
1,
8,
1,
2,
2,

```
16,
23,
27,
16,
15,
2,
8,
13,
2,
2,
3,
2,
28,
2,
2,
9,
16,
6,
27,
14,
12,
7,
2,
19,
20,
3,
5,
2,
29,
4,
3,
12,
12,
26,
13,
2,
12,
16,
2,
8,
12,
2,
11,
2,
13,
8,
12,
2,
2,
2,
24,
3,
12,
25,
2,
17,
29,
8,
3,
9,
10,
16,
2,
26,
13,
10,
7,
11,
2,
25,
6,
1,
20,
11,
2,
5,
27,
2,
1,
10,
24,
2,
20,
```

5,
2,
6,
2,
5,
6,
2,
19,
2,
11,
14,
2,
3,
14,
12,
5,
26,
3,
1,
7,
6,
8,
12,
12,
9,
15,
11,
2,
4,
2,
2,
24,
28,
1,
10,
11,
2,
13,
6,
12,
4,
13,
24,
16,
8,
3,
9,
28,
13,
17,
20,
28,
15,
12,
10,
12,
7,
2,
4,
11,
2,
26,
2,
2,
14,
2,
22,
6,
8,
18,
35,
2,
2,
2,
2,
7,
35,
26,
11,
26,
3,
1,
2,

```
14,
16,
12,
29,
20,
8,
13,
10,
19,
28,
2,
17,
2,
2,
13,
12,
9,
21,
15,
2,
14,
2,
6,
22,
14,
2,
23,
13,
1,
5,
1,
22,
1,
2,
37,
2,
2,
36,
9,
1,
5,
2,
15,
25,
5,
7,
17,
2,
8,
3,
5,
2,
17,
13,
2,
21,
9,
8,
10,
35,
14,
3,
25,
2,
8,
10,
11,
7,
...]
```

In [65]:
```
duration_mins
```

Out[65]:
```
[50,
25,
0,
25,
45,
25,
30,
5,
```

30,
50,
15,
35,
15,
10,
35,
35,
30,
35,
0,
35,
10,
20,
50,
55,
20,
10,
45,
55,
50,
15,
15,
25,
50,
0,
30,
25,
15,
30,
15,
55,
5,
30,
5,
45,
0,
50,
50,
15,
5,
55,
15,
20,
50,
0,
50,
15,
5,
15,
30,
5,
25,
20,
15,
30,
15,
25,
50,
30,
20,
15,
25,
30,
20,
30,
0,
45,
50,
30,
5,
15,
50,
40,
10,
35,
0,
20,
25,
45,
20,
0,
15,

5,
30,
20,
35,
25,
40,
0,
25,
35,
20,
15,
30,
0,
45,
55,
10,
30,
20,
55,
0,
55,
45,
10,
55,
25,
5,
10,
40,
55,
40,
25,
5,
15,
25,
10,
10,
0,
50,
20,
0,
55,
10,
45,
0,
50,
40,
15,
45,
50,
0,
15,
25,
35,
50,
35,
25,
55,
50,
10,
20,
25,
50,
20,
15,
35,
35,
50,
45,
10,
50,
45,
15,
20,
30,
25,
30,
40,
10,
45,
35,
5,
30,
30,

10,
35,
5,
50,
0,
0,
25,
30,
30,
0,
20,
30,
45,
10,
35,
45,
55,
35,
20,
5,
40,
15,
10,
25,
50,
25,
15,
50,
15,
50,
20,
55,
35,
35,
30,
20,
55,
15,
5,
45,
55,
30,
40,
45,
55,
0,
45,
15,
25,
15,
55,
50,
25,
50,
30,
30,
40,
35,
15,
30,
45,
35,
30,
55,
55,
15,
40,
30,
0,
15,
15,
50,
10,
25,
20,
30,
10,
30,
20,
35,
15,
20,
45,

0,
0,
30,
50,
55,
45,
15,
35,
55,
5,
5,
15,
0,
55,
0,
15,
40,
20,
25,
20,
30,
45,
15,
55,
20,
55,
5,
40,
0,
0,
35,
30,
55,
35,
55,
15,
20,
40,
0,
10,
35,
45,
35,
30,
40,
35,
55,
40,
25,
15,
10,
45,
30,
50,
0,
30,
50,
15,
30,
20,
30,
25,
35,
55,
35,
30,
45,
20,
30,
35,
45,
55,
40,
5,
0,
45,
40,
55,
55,
10,
20,
50,
30,

0,
45,
20,
25,
20,
35,
0,
10,
20,
0,
45,
55,
40,
10,
10,
50,
20,
0,
0,
35,
5,
40,
50,
5,
50,
35,
25,
25,
40,
55,
10,
50,
50,
50,
30,
30,
50,
15,
10,
5,
35,
5,
0,
30,
30,
20,
45,
30,
35,
45,
35,
30,
15,
5,
0,
50,
25,
20,
45,
45,
50,
0,
25,
55,
50,
40,
45,
50,
15,
30,
0,
5,
20,
35,
50,
50,
10,
5,
30,
5,
20,
55,
30,

50,
15,
0,
35,
45,
55,
20,
40,
0,
25,
55,
45,
15,
0,
10,
25,
40,
50,
20,
20,
0,
45,
55,
20,
15,
45,
55,
55,
20,
25,
50,
5,
15,
0,
25,
15,
10,
45,
0,
55,
45,
30,
45,
40,
30,
30,
40,
5,
25,
30,
5,
40,
50,
45,
20,
0,
15,
0,
0,
40,
0,
20,
20,
20,
25,
20,
5,
55,
45,
45,
55,
15,
15,
30,
0,
0,
10,
55,
30,
20,
15,
30,
20,

55,
25,
30,
55,
25,
25,
15,
20,
10,
15,
15,
30,
5,
50,
25,
15,
50,
15,
15,
5,
35,
0,
50,
50,
20,
20,
10,
35,
35,
10,
0,
35,
55,
35,
40,
30,
15,
55,
50,
25,
40,
35,
55,
25,
35,
30,
15,
15,
5,
50,
10,
50,
30,
35,
35,
35,
30,
15,
35,
30,
20,
0,
35,
30,
10,
55,
45,
40,
5,
45,
50,
20,
0,
0,
55,
5,
35,
0,
25,
35,
40,
55,
20,

0,
40,
0,
0,
45,
10,
20,
55,
35,
55,
45,
35,
30,
45,
40,
50,
50,
50,
30,
55,
10,
55,
15,
10,
55,
30,
10,
35,
40,
50,
30,
0,
10,
55,
0,
0,
45,
30,
20,
10,
30,
15,
35,
0,
0,
50,
25,
5,
50,
5,
15,
45,
50,
30,
55,
45,
50,
45,
35,
30,
15,
55,
45,
30,
20,
30,
30,
30,
0,
15,
50,
5,
35,
30,
15,
45,
45,
50,
35,
55,
30,
40,
0,

30,
55,
10,
50,
45,
30,
20,
20,
55,
10,
45,
50,
0,
10,
20,
45,
15,
55,
30,
35,
35,
45,
25,
35,
30,
15,
40,
25,
40,
45,
20,
40,
50,
20,
15,
15,
45,
35,
30,
30,
35,
35,
50,
35,
20,
45,
30,
50,
30,
30,
0,
0,
25,
35,
10,
10,
55,
15,
10,
55,
15,
20,
25,
50,
20,
20,
15,
10,
40,
50,
10,
5,
25,
55,
0,
45,
55,
30,
15,
50,
0,
15,
30,

35,
25,
0,
30,
50,
45,
30,
40,
30,
50,
35,
5,
50,
5,
15,
25,
50,
35,
50,
40,
25,
10,
55,
40,
50,
35,
40,
40,
20,
0,
35,
10,
35,
20,
10,
0,
0,
10,
40,
35,
55,
0,
30,
35,
20,
15,
50,
45,
20,
40,
15,
0,
50,
0,
20,
50,
10,
10,
40,
45,
50,
0,
0,
35,
50,
55,
5,
20,
55,
5,
0,
25,
45,
45,
20,
15,
30,
35,
0,
50,
45,
25,
25,

10,
55,
50,
0,
0,
50,
15,
15,
50,
50,
0,
30,
45,
55,
45,
15,
15,
50,
25,
55,
25,
5,
45,
15,
20,
30,
20,
20,
5,
30,
5,
10,
0,
5,
55,
35,
10,
40,
55,
30,
40,
30,
10,
20,
25,
10,
30,
50,
45,
25,
55,
25,
30,
5,
5,
20,
40,
30,
15,
20,
50,
20,
5,
25,
5,
20,
25,
35,
50,
35,
15,
35,
0,
25,
35,
15,
35,
5,
35,
30,
45,
15,
15,

```
30,
20,
30,
35,
35,
30,
30,
40,
0,
30,
55,
15,
55,
20,
35,
20,
25,
50,
0,
25,
0,
50,
20,
45,
55,
25,
55,
45,
25,
40,
20,
25,
50,
35,
10,
25,
30,
35,
5,
30,
5,
20,
55,
30,
35,
20,
50,
50,
10,
15,
25,
5,
15,
10,
55,
40,
30,
45,
20,
15,
40,
55,
30,
30,
50,
15,
35,
30,
40,
25,
5,
15,
0,
50,
30,
40,
0,
30,
30,
...]
```

In [66]:

In [66]:
```
# Adding duration_hours and duration_mins list to data dataframe

df["Duration_hours"] = duration_hours
df["Duration_mins"] = duration_mins
```

Now drop Duration column

In [67]:
```
df.drop(["Duration"], axis = 1, inplace = True)
```

In [68]:
```
df.head()
```

Out[68]:

	Airline	Source	Destination	Route	Total_Stops	Additional_Info	Price	Journey_day	Journey_month	Dep_hour	Dep_min	Arrival_hour
0	IndiGo	Banglore	New Delhi	BLR → DEL	non-stop	No info	3897	24	3	22	20	1
1	Air India	Kolkata	Banglore	CCU → IXR → BBI → BLR	2 stops	No info	7662	1	5	5	50	13
2	Jet Airways	Delhi	Cochin	DEL → LKO → BOM → COK	2 stops	No info	13882	9	6	9	25	4
3	IndiGo	Kolkata	Banglore	CCU → NAG → BLR	1 stop	No info	6218	12	5	18	5	23
4	IndiGo	Banglore	New Delhi	BLR → NAG → DEL	1 stop	No info	13302	1	3	16	50	21

Handling Categorical Data

check the value counts in Airline

In [69]:
```
df["Airline"].value_counts()
```

Out[69]:
```
Jet Airways                          3849
IndiGo                               2053
Air India                            1751
Multiple carriers                    1196
SpiceJet                              818
Vistara                               479
Air Asia                              319
GoAir                                 194
Multiple carriers Premium economy      13
Jet Airways Business                    6
Vistara Premium economy                 3
Trujet                                  1
Name: Airline, dtype: int64
```

Display price according to Airline

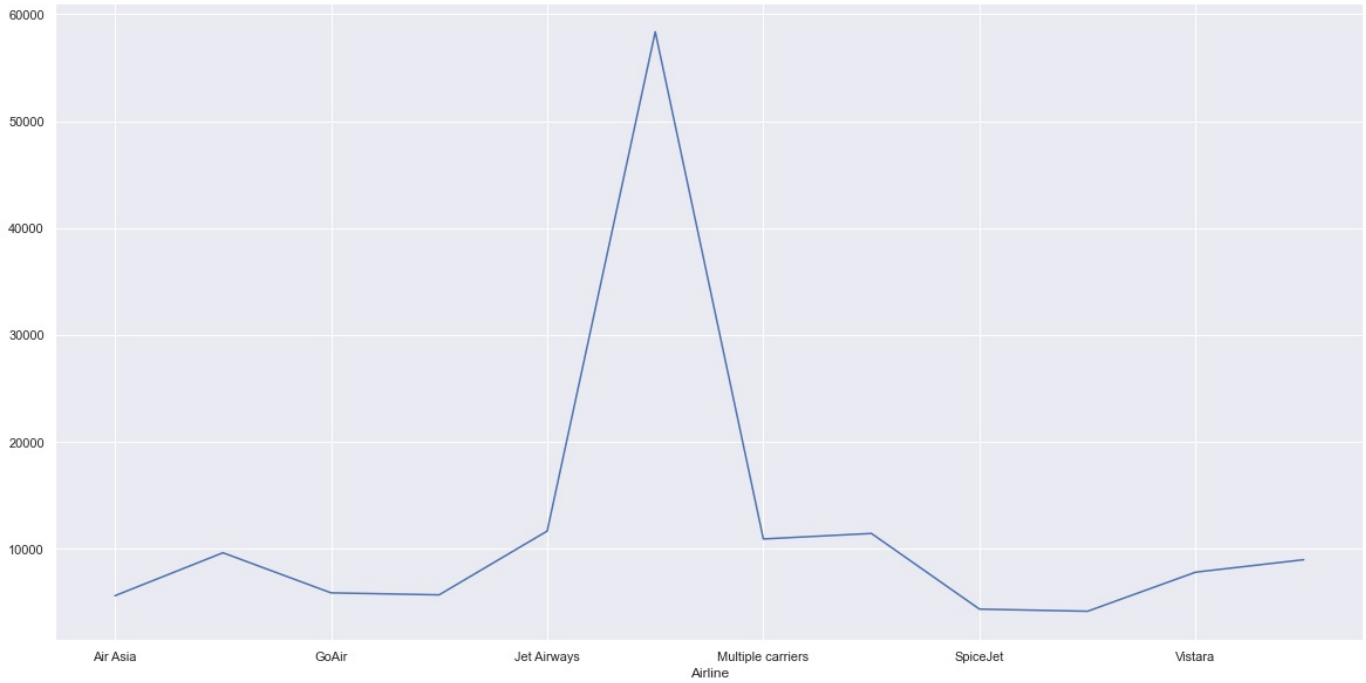
In [70]:
```
df.groupby('Airline')['Price'].mean()
```

Out[70]:
```
Airline
```

```
Air Asia                          5590.260188
Air India                         9612.427756
GoAir                             5861.056701
IndiGo                            5673.682903
Jet Airways                      11643.923357
Jet Airways Business             58358.666667
Multiple carriers                10902.678094
Multiple carriers Premium economy    11418.846154
SpiceJet                          4338.284841
Trujet                            4140.000000
Vistara                           7796.348643
Vistara Premium economy           8962.333333
Name: Price, dtype: float64
```

In [71]:
```
df.groupby('Airline')['Price'].mean().plot(figsize=(20,10)) #plot shows there is relation between airline and pri
```

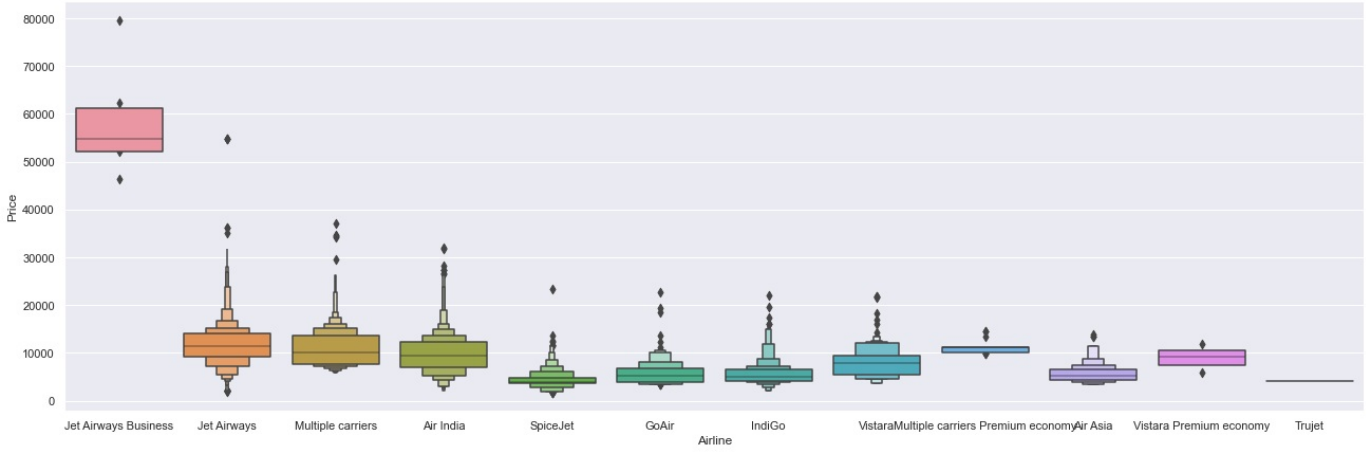
Out[71]: <AxesSubplot:xlabel='Airline'>



here we can say that jet airways price higher than any onther airline

check the average price according to Airline

In [72]:
```
sns.catplot(x = "Airline", y = "Price", data = df.sort_values("Price", ascending = False), kind="boxen", height =
plt.show()
```



now we hace to create dummy data for Categorical

Airline is Nominal Categorical data we will perform OneHotEncoding

In [73]:
```
Airline = df[["Airline"]]

Airline = pd.get_dummies(Airline, drop_first= True)

Airline.head()
```

Out[73]:

	Airline_Air India	Airline_GoAir	Airline_IndiGo	Airline_Jet Airways	Airline_Jet Airways Business	Airline_Multiple carriers	Airline_Multiple carriers Premium economy	Airline_SpiceJet	Airline_Trujet	Airline_Vist
0	0	0	1	0	0	0	0	0	0	
1	1	0	0	0	0	0	0	0	0	
2	0	0	0	1	0	0	0	0	0	
3	0	0	1	0	0	0	0	0	0	
4	0	0	1	0	0	0	0	0	0	

check the Source values counts

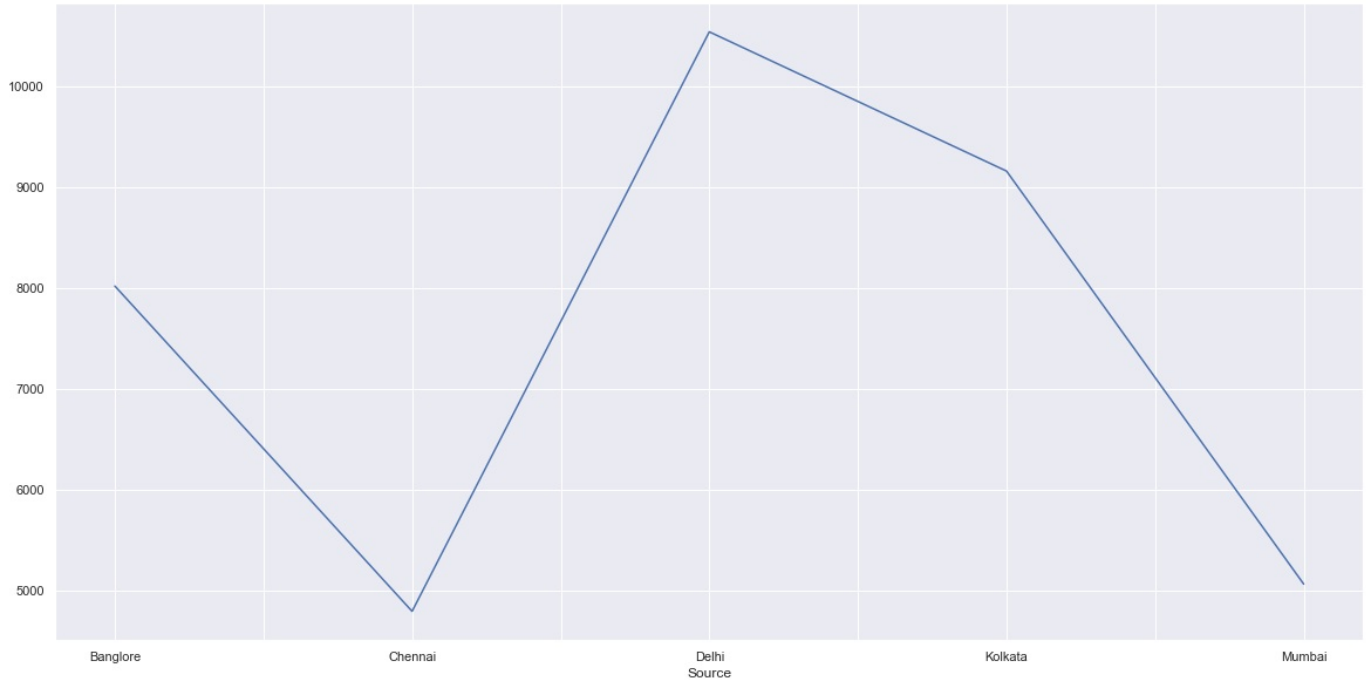
In [74]:
```
df["Source"].value_counts()
```

Out[74]:
```
Delhi       4536
Kolkata     2871
Banglore    2197
Mumbai       697
Chennai      381
Name: Source, dtype: int64
```

display average price according to source

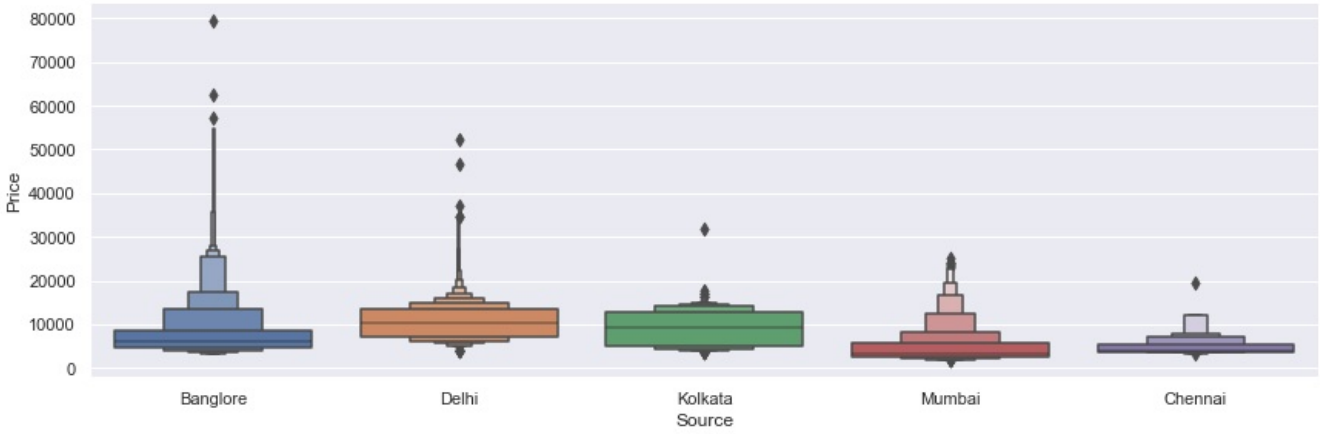
In [75]:
```
df.groupby('Source')['Price'].mean().plot(figsize=(20,10))
```

Out[75]:
```
<AxesSubplot:xlabel='Source'>
```



In [76]:
```
# Source vs Price

sns.catplot(y = "Price", x = "Source", data = df.sort_values("Price", ascending = False), kind="boxen", height =
plt.show()
```



Source is Nominal Categorical data we will perform OneHotEncoding

In [77]:
```
Source = df[["Source"]]

Source = pd.get_dummies(Source, drop_first= True)

Source.head()
```

Out[77]:

	Source_Chennai	Source_Delhi	Source_Kolkata	Source_Mumbai
0	0	0	0	0
1	0	0	1	0
2	0	1	0	0
3	0	0	1	0
4	0	0	0	0

check Destination value counts

In [78]:
```
df["Destination"].value_counts()
```

Out[78]:
```
Cochin       4536
Banglore     2871
Delhi        1265
New Delhi     932
Hyderabad     697
Kolkata       381
Name: Destination, dtype: int64
```

Destination is Nominal Categorical data we will perform OneHotEncoding

In [79]:
```
Destination = df[["Destination"]]

Destination = pd.get_dummies(Destination, drop_first = True)

Destination.head()
```

Out[79]:

	Destination_Cochin	Destination_Delhi	Destination_Hyderabad	Destination_Kolkata	Destination_New Delhi
0	0	0	0	0	1

1	0	0	0	0	0
2	1	0	0	0	0
3	0	0	0	0	0
4	0	0	0	0	1

check Route column

In [80]:
```
df["Route"]
```

Out[80]:
```
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 [81]:
```
df['Additional_Info']
```

Out[81]:
```
0         No info
1         No info
2         No info
3         No info
4         No info
           ...
10678     No info
10679     No info
10680     No info
10681     No info
10682     No info
Name: Additional_Info, Length: 10682, dtype: object
```

Drop Route and Additional_Info columns because its unrelevant

In [82]:
```
df.drop(["Route", "Additional_Info"], axis = 1, inplace = True)
```

check the Total_Stops values counts

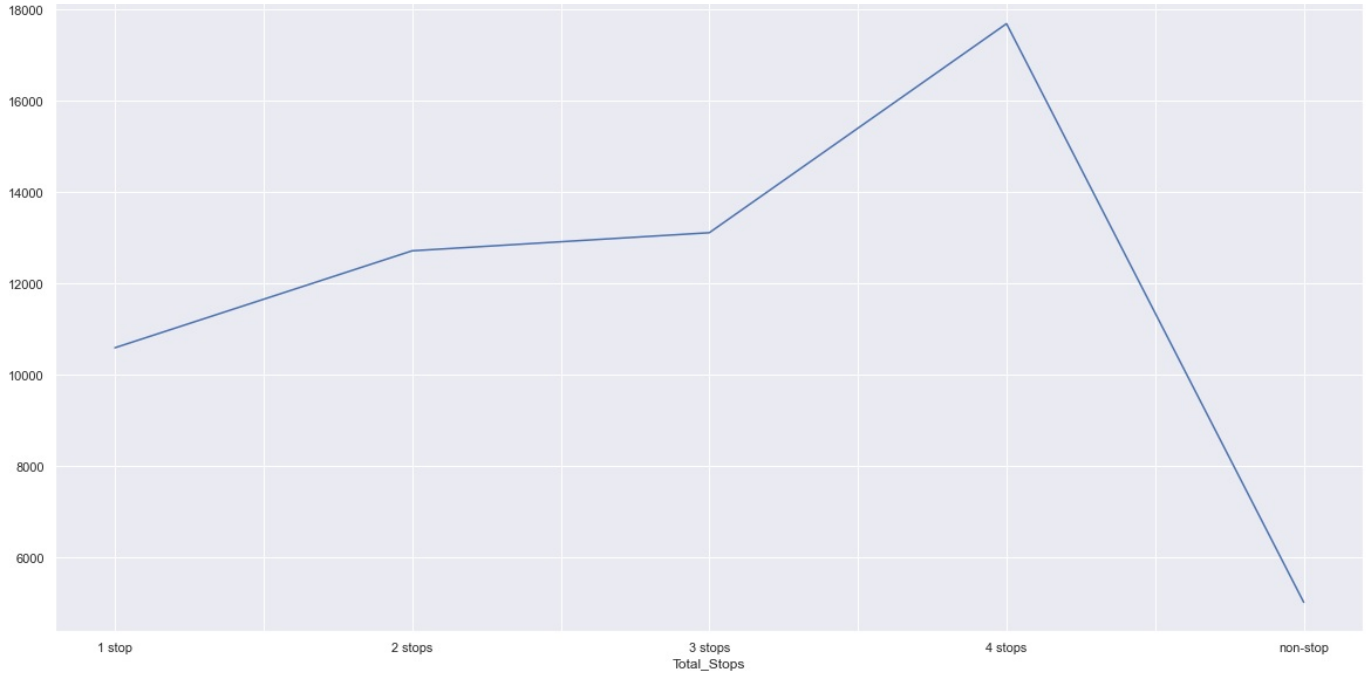
In [84]:
```
df["Total_Stops"].value_counts()
```

Out[84]:
```
1 stop      5625
non-stop    3491
2 stops     1520
3 stops       45
4 stops        1
Name: Total_Stops, dtype: int64
```

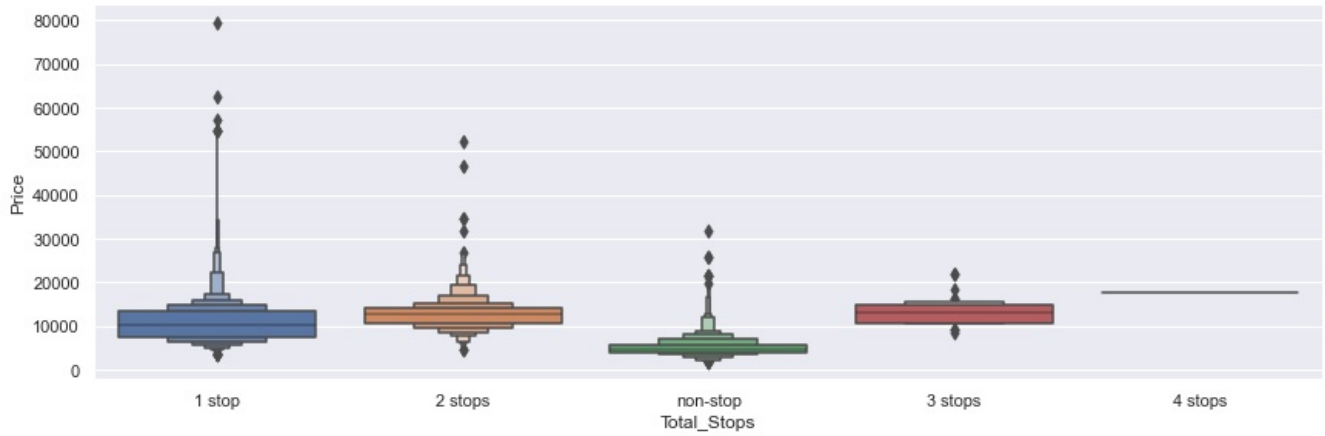
Display Average price of all Total_Stops

In [85]:
```
df.groupby('Total_Stops')['Price'].mean().plot(figsize=(20,10))
```

Out[85]:
```
<AxesSubplot:xlabel='Total_Stops'>
```



In [86]:
```
sns.catplot(y = "Price", x = "Total_Stops", data = df.sort_values("Price", ascending = False), kind="boxen", heig
plt.show()
```



Now replace categorical value in Total_stop with numeric value by manually

In [87]:
```
df.replace({"non-stop": 0, "1 stop": 1, "2 stops": 2, "3 stops": 3, "4 stops": 4}, inplace = True)
```

In [88]:
```
df.head()
```

Out[88]:

	Airline	Source	Destination	Total_Stops	Price	Journey_day	Journey_month	Dep_hour	Dep_min	Arrival_hour	Arrival_min	Duration_ho
0	IndiGo	Banglore	New Delhi	0	3897	24	3	22	20	1	10	
1	Air India	Kolkata	Banglore	2	7662	1	5	5	50	13	15	
2	Jet Airways	Delhi	Cochin	2	13882	9	6	9	25	4	25	
3	IndiGo	Kolkata	Banglore	1	6218	12	5	18	5	23	30	
4	IndiGo	Banglore	New Delhi	1	13302	1	3	16	50	21	35	

Now concatenate all dummy data which we created with our orignal dataset

In [89]:
```
# Concatenate dataframe --> data + Airline + Source + Destination

df = pd.concat([df, Airline, Source, Destination], axis = 1)
```

In [90]:
```
df.head()
```

Out[90]:

	Airline	Source	Destination	Total_Stops	Price	Journey_day	Journey_month	Dep_hour	Dep_min	Arrival_hour	Arrival_min	Duration_ho
0	IndiGo	Banglore	New Delhi	0	3897	24	3	22	20	1	10	
1	Air India	Kolkata	Banglore	2	7662	1	5	5	50	13	15	
2	Jet Airways	Delhi	Cochin	2	13882	9	6	9	25	4	25	
3	IndiGo	Kolkata	Banglore	1	6218	12	5	18	5	23	30	
4	IndiGo	Banglore	New Delhi	1	13302	1	3	16	50	21	35	

Drop Categorical columns from dataset

In [91]:
```
df.drop(["Airline", "Source", "Destination"], axis = 1, inplace = True)
```

In [92]:
```
df.head()
```

Out[92]:

	Total_Stops	Price	Journey_day	Journey_month	Dep_hour	Dep_min	Arrival_hour	Arrival_min	Duration_hours	Duration_mins	Airline_Air India
0	0	3897	24	3	22	20	1	10	2	50	0
1	2	7662	1	5	5	50	13	15	7	25	1
2	2	13882	9	6	9	25	4	25	19	0	0
3	1	6218	12	5	18	5	23	30	5	25	0
4	1	13302	1	3	16	50	21	35	4	45	0

In [93]:
```
df.shape
```

Out[93]: (10682, 30)

Feature Selection

In [51]:
```
df.shape
```

Out[51]: (10682, 30)

Check all columns from dataset

In [94]:
```
df.columns
```

Out[94]:
```
Index(['Total_Stops', 'Price', 'Journey_day', 'Journey_month', 'Dep_hour',
       'Dep_min', 'Arrival_hour', 'Arrival_min', 'Duration_hours',
       'Duration_mins', 'Airline_Air India', 'Airline_GoAir', 'Airline_IndiGo',
       'Airline_Jet Airways', 'Airline_Jet Airways Business',
       'Airline_Multiple carriers',
       'Airline_Multiple carriers Premium economy', 'Airline_SpiceJet',
       'Airline_Trujet', 'Airline_Vistara', 'Airline_Vistara Premium economy',
       'Source_Chennai', 'Source_Delhi', 'Source_Kolkata', 'Source_Mumbai',
```

```
       'Destination_Cochin', 'Destination_Delhi', 'Destination_Hyderabad',
       'Destination_Kolkata', 'Destination_New Delhi'],
      dtype='object')
```

Create target and features set

In [95]:
```
X = df.drop('Price',axis=1)
y = df.Price
```

In [96]:
```
X.head()
```

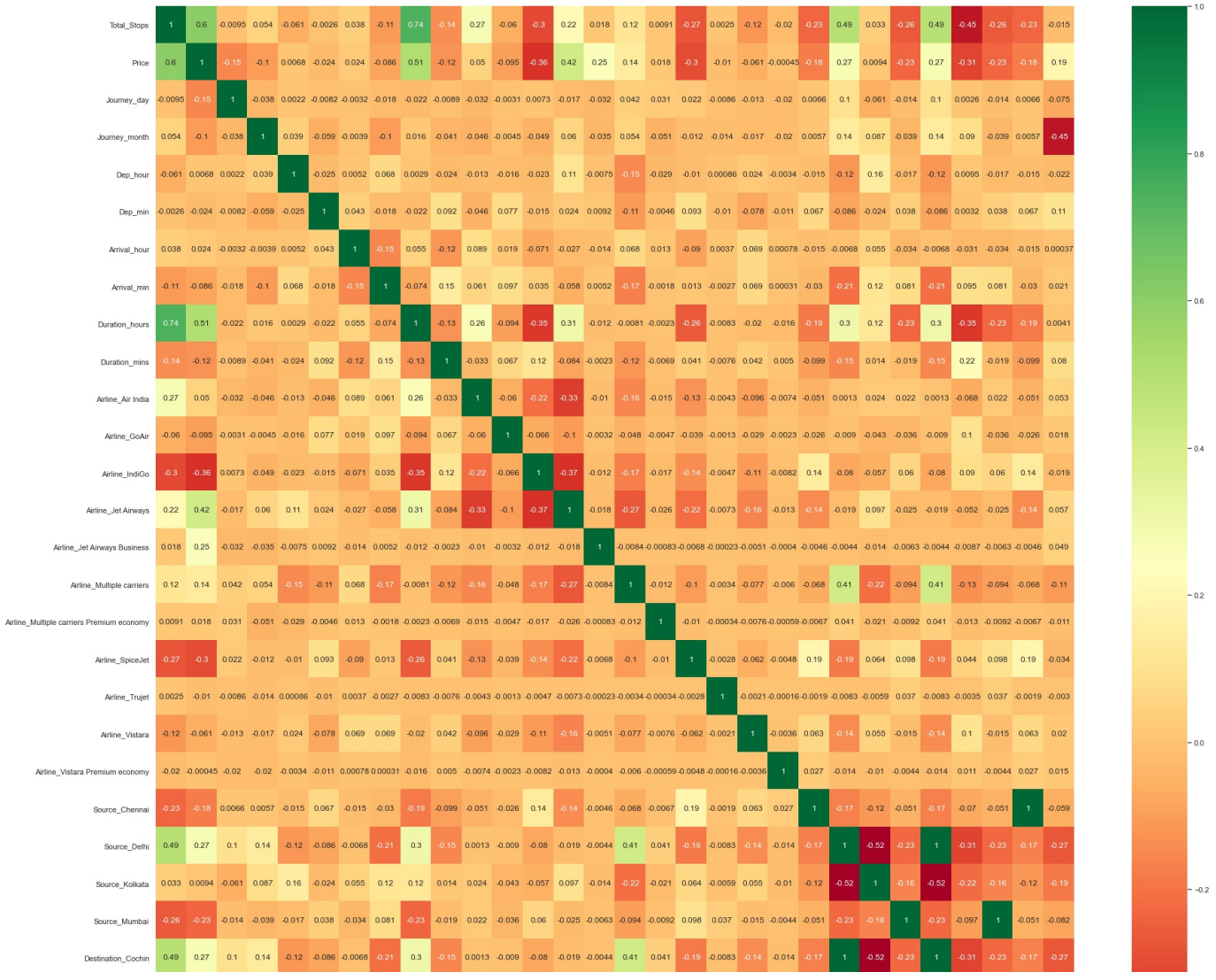
Out[96]:

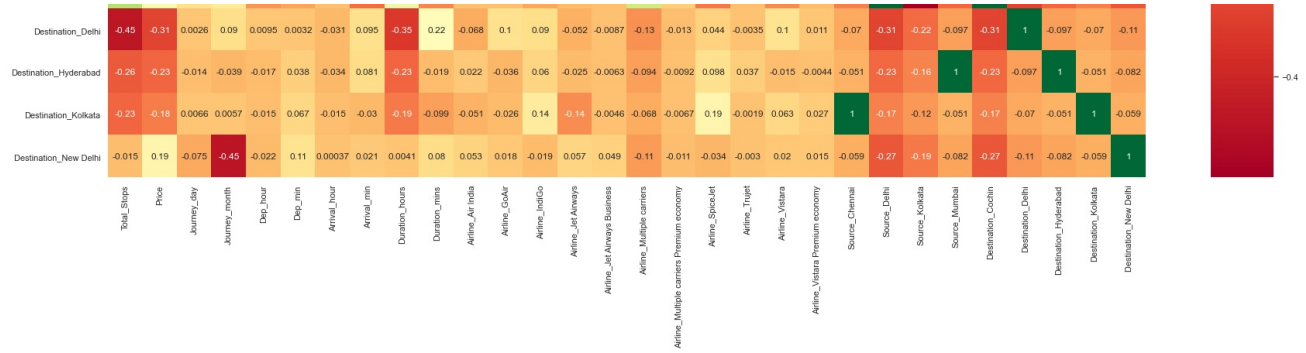
	Total_Stops	Journey_day	Journey_month	Dep_hour	Dep_min	Arrival_hour	Arrival_min	Duration_hours	Duration_mins	Airline_Air India	Airline
0	0	24	3	22	20	1	10	2	50	0	
1	2	1	5	5	50	13	15	7	25	1	
2	2	9	6	9	25	4	25	19	0	0	
3	1	12	5	18	5	23	30	5	25	0	
4	1	1	3	16	50	21	35	4	45	0	

Finds correlation between Independent and dependent attributes

In [97]:
```
plt.figure(figsize = (30,30))
sns.heatmap(df.corr(), annot = True, cmap = "RdYlGn")

plt.show()
```





Find the important Featuresn using ExtraTreesRegressor

In [98]:
```
from sklearn.ensemble import ExtraTreesRegressor
selection = ExtraTreesRegressor()
selection.fit(X, y)
```

Out[98]:
```
ExtraTreesRegressor()
```

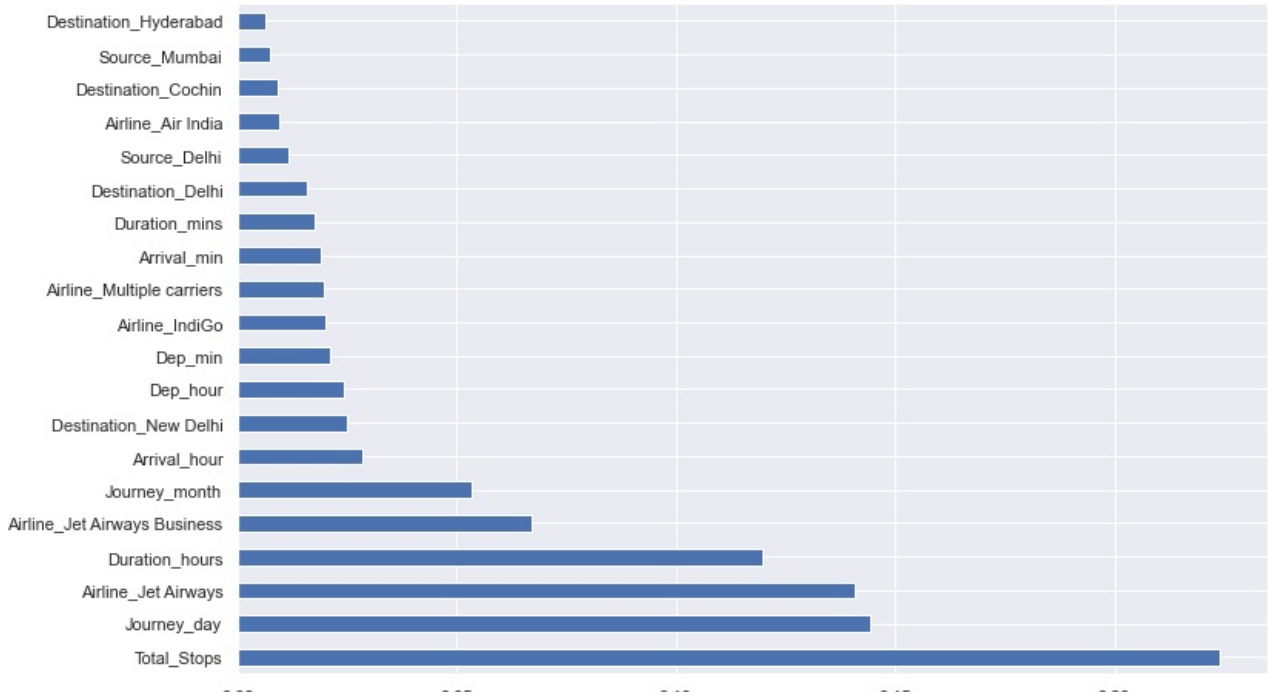
print all features importances

In [99]:
```
print(selection.feature_importances_)
```

```
[2.23838640e-01 1.44267766e-01 5.34722485e-02 2.41573711e-02
 2.09762177e-02 2.85246298e-02 1.91140167e-02 1.19667886e-01
 1.74560464e-02 9.49473788e-03 1.95932134e-03 1.99846220e-02
 1.40689753e-01 6.71526548e-02 1.96808705e-02 8.46167994e-04
 3.48327739e-03 1.07780705e-04 4.97244044e-03 8.41045032e-05
 5.77434103e-04 1.17974821e-02 3.26591463e-03 7.59878820e-03
 9.10351828e-03 1.59492902e-02 6.30718884e-03 5.78988160e-04
 2.48908424e-02]
```

plot graph of feature importances for better visualization

In [100…
```
plt.figure(figsize = (12,8))
feat_importances = pd.Series(selection.feature_importances_, index=X.columns)
feat_importances.nlargest(20).plot(kind='barh')
plt.show()
```



create training and testing data

In [101…
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 42)
```

Apply Linear regression on training dataset

In [102…
```
from sklearn.linear_model import LinearRegression
```

In [103…
```
model_li = LinearRegression()
model_li.fit(X_train,y_train)
```

Out[103… LinearRegression()

print trainging and testing score

In [104…
```
model_li.score(X_train,y_train)
```

Out[104… 0.6240840020468166

In [105…
```
model_li.score(X_test,y_test)
```

Out[105… 0.6195943729070101

Now try all different regression algorithm and find the testing score

In [106…
```
from sklearn.tree import DecisionTreeRegressor
from sklearn.neighbors import KNeighborsRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import AdaBoostRegressor
from sklearn.svm import SVR
```

In [107…
```
model = [DecisionTreeRegressor,SVR,RandomForestRegressor,KNeighborsRegressor,AdaBoostRegressor]

for mod in model:
    reg = mod()
    reg = reg.fit(X_train,y_train)
    print(mod , 'accuracy',reg.score(X_test,y_test))
```

```
<class 'sklearn.tree._classes.DecisionTreeRegressor'> accuracy 0.7271700570293645
<class 'sklearn.svm._classes.SVR'> accuracy -0.00041646312498344606
<class 'sklearn.ensemble._forest.RandomForestRegressor'> accuracy 0.7972916123593482
<class 'sklearn.neighbors._regression.KNeighborsRegressor'> accuracy 0.5743709506218349
<class 'sklearn.ensemble._weight_boosting.AdaBoostRegressor'> accuracy 0.5011076957929421
```

Now apply KFold and cross validation technique

In [108…
```
from sklearn.model_selection import KFold,cross_val_score
```

In [109…
```
models = []
models.append(('KNN', KNeighborsRegressor()))
```

```
models.append(('CART', DecisionTreeRegressor()))
models.append(('RF', RandomForestRegressor()))
models.append(('SVM', SVR()))
models.append(('AdaBoost', AdaBoostRegressor()))


results = []
names = []
for name,model in models:
    kfold = KFold(n_splits=10)
    cv_result =cross_val_score(model,X_train,y_train,cv=kfold)
    names.append(name)
    results.append(cv_result)
for i in range(len(names)):
    print(names[i],results[i].mean())
```

```
KNN 0.5654993361648268
CART 0.6912889148141191
RF 0.804145462843865
SVM -0.00016124772787321496
AdaBoost 0.42066613141881753
```

Here we see RandomForestRegressor gives us best score so we can use RandomForest Regressor algorithm

In [110…

```
from sklearn.ensemble import RandomForestRegressor
reg_rf = RandomForestRegressor()
reg_rf.fit(X_train, y_train)
```

Out[110… RandomForestRegressor()

In [111…

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

In [112…

```
reg_rf.score(X_train, y_train)
```

Out[112… 0.9534394156782514

In [113…

```
reg_rf.score(X_test, y_test)
```

Out[113… 0.7961678810136219

Perform Hyper-parameter tuning using RandomizedSearchCV

In [114…

```
from sklearn.model_selection import RandomizedSearchCV
```

create list for all possible parameter

In [115…

```
n_estimators = [int(x) for x in np.linspace(start = 100, stop = 1200, num = 12)]
max_features = ['auto', 'sqrt']
max_depth = [int(x) for x in np.linspace(5, 30, num = 6)]
min_samples_split = [2, 5, 10, 15, 100]
min_samples_leaf = [1, 2, 5, 10]
```

In [116…

```
random_grid = {'n_estimators': n_estimators,
               'max_features': max_features,
               'max_depth': max_depth,
               'min_samples_split': min_samples_split,
               'min_samples_leaf': min_samples_leaf}
```

In [117…

```
random_grid
```

Out[117…
```
{'n_estimators': [100,
  200,
  300,
  400,
  500,
  600,
  700,
  800,
  900,
  1000,
  1100,
  1200],
 'max_features': ['auto', 'sqrt'],
 'max_depth': [5, 10, 15, 20, 25, 30],
 'min_samples_split': [2, 5, 10, 15, 100],
 'min_samples_leaf': [1, 2, 5, 10]}
```

Random search of parameters, using 5 fold cross validation and search across 100 different combinations

In [75]:
```
rf_random = RandomizedSearchCV(estimator = reg_rf,
                               param_distributions = random_grid,
                               scoring='neg_mean_squared_error',
                               n_iter = 10, cv = 5,
                               verbose=2,
                               random_state=42, n_jobs = 1)
```

In [76]:
```
rf_random.fit(X_train,y_train)
```

```
Fitting 5 folds for each of 10 candidates, totalling 50 fits
[CV] n_estimators=900, min_samples_split=5, min_samples_leaf=5, max_features=sqrt, max_depth=10

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.


[CV]  n_estimators=900, min_samples_split=5, min_samples_leaf=5, max_features=sqrt, max_depth=10, total=   5.6s
[CV] n_estimators=900, min_samples_split=5, min_samples_leaf=5, max_features=sqrt, max_depth=10
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    5.5s remaining:    0.0s


[CV]  n_estimators=900, min_samples_split=5, min_samples_leaf=5, max_features=sqrt, max_depth=10, total=   5.2s
[CV] n_estimators=900, min_samples_split=5, min_samples_leaf=5, max_features=sqrt, max_depth=10
[CV]  n_estimators=900, min_samples_split=5, min_samples_leaf=5, max_features=sqrt, max_depth=10, total=   4.9s
[CV] n_estimators=900, min_samples_split=5, min_samples_leaf=5, max_features=sqrt, max_depth=10
[CV]  n_estimators=900, min_samples_split=5, min_samples_leaf=5, max_features=sqrt, max_depth=10, total=   4.0s
[CV] n_estimators=900, min_samples_split=5, min_samples_leaf=5, max_features=sqrt, max_depth=10
[CV]  n_estimators=900, min_samples_split=5, min_samples_leaf=5, max_features=sqrt, max_depth=10, total=   6.8s
[CV] n_estimators=1100, min_samples_split=10, min_samples_leaf=2, max_features=sqrt, max_depth=15
[CV]  n_estimators=1100, min_samples_split=10, min_samples_leaf=2, max_features=sqrt, max_depth=15, total=   8.1s
[CV] n_estimators=1100, min_samples_split=10, min_samples_leaf=2, max_features=sqrt, max_depth=15
[CV]  n_estimators=1100, min_samples_split=10, min_samples_leaf=2, max_features=sqrt, max_depth=15, total=   7.1s
[CV] n_estimators=1100, min_samples_split=10, min_samples_leaf=2, max_features=sqrt, max_depth=15
[CV]  n_estimators=1100, min_samples_split=10, min_samples_leaf=2, max_features=sqrt, max_depth=15, total=   8.7s
[CV] n_estimators=1100, min_samples_split=10, min_samples_leaf=2, max_features=sqrt, max_depth=15
[CV]  n_estimators=1100, min_samples_split=10, min_samples_leaf=2, max_features=sqrt, max_depth=15, total=   8.4s
[CV] n_estimators=1100, min_samples_split=10, min_samples_leaf=2, max_features=sqrt, max_depth=15
[CV]  n_estimators=1100, min_samples_split=10, min_samples_leaf=2, max_features=sqrt, max_depth=15, total=   6.0s
[CV] n_estimators=300, min_samples_split=100, min_samples_leaf=5, max_features=auto, max_depth=15
[CV]  n_estimators=300, min_samples_split=100, min_samples_leaf=5, max_features=auto, max_depth=15, total=   3.2s
[CV] n_estimators=300, min_samples_split=100, min_samples_leaf=5, max_features=auto, max_depth=15
[CV]  n_estimators=300, min_samples_split=100, min_samples_leaf=5, max_features=auto, max_depth=15, total=   3.3s
[CV] n_estimators=300, min_samples_split=100, min_samples_leaf=5, max_features=auto, max_depth=15
[CV]  n_estimators=300, min_samples_split=100, min_samples_leaf=5, max_features=auto, max_depth=15, total=   3.2s
[CV] n_estimators=300, min_samples_split=100, min_samples_leaf=5, max_features=auto, max_depth=15
[CV]  n_estimators=300, min_samples_split=100, min_samples_leaf=5, max_features=auto, max_depth=15, total=   3.1s
[CV] n_estimators=300, min_samples_split=100, min_samples_leaf=5, max_features=auto, max_depth=15
[CV]  n_estimators=300, min_samples_split=100, min_samples_leaf=5, max_features=auto, max_depth=15, total=   3.3s
[CV] n_estimators=400, min_samples_split=5, min_samples_leaf=5, max_features=auto, max_depth=15
[CV]  n_estimators=400, min_samples_split=5, min_samples_leaf=5, max_features=auto, max_depth=15, total=   6.8s
[CV] n_estimators=400, min_samples_split=5, min_samples_leaf=5, max_features=auto, max_depth=15
[CV]  n_estimators=400, min_samples_split=5, min_samples_leaf=5, max_features=auto, max_depth=15, total=   8.2s
[CV] n_estimators=400, min_samples_split=5, min_samples_leaf=5, max_features=auto, max_depth=15
[CV]  n_estimators=400, min_samples_split=5, min_samples_leaf=5, max_features=auto, max_depth=15, total=   7.3s
[CV] n_estimators=400, min_samples_split=5, min_samples_leaf=5, max_features=auto, max_depth=15
[CV]  n_estimators=400, min_samples_split=5, min_samples_leaf=5, max_features=auto, max_depth=15, total=   6.2s
[CV] n_estimators=400, min_samples_split=5, min_samples_leaf=5, max_features=auto, max_depth=15
```

```
[CV]  n_estimators=400, min_samples_split=5, min_samples_leaf=5, max_features=auto, max_depth=15, total=   5.9s
[CV] n_estimators=700, min_samples_split=5, min_samples_leaf=10, max_features=auto, max_depth=20
[CV]  n_estimators=700, min_samples_split=5, min_samples_leaf=10, max_features=auto, max_depth=20, total=   9.3s
[CV] n_estimators=700, min_samples_split=5, min_samples_leaf=10, max_features=auto, max_depth=20
[CV]  n_estimators=700, min_samples_split=5, min_samples_leaf=10, max_features=auto, max_depth=20, total=   9.1s
[CV] n_estimators=700, min_samples_split=5, min_samples_leaf=10, max_features=auto, max_depth=20
[CV]  n_estimators=700, min_samples_split=5, min_samples_leaf=10, max_features=auto, max_depth=20, total=   8.9s
[CV] n_estimators=700, min_samples_split=5, min_samples_leaf=10, max_features=auto, max_depth=20
[CV]  n_estimators=700, min_samples_split=5, min_samples_leaf=10, max_features=auto, max_depth=20, total=   9.0s
[CV] n_estimators=700, min_samples_split=5, min_samples_leaf=10, max_features=auto, max_depth=20
[CV]  n_estimators=700, min_samples_split=5, min_samples_leaf=10, max_features=auto, max_depth=20, total=   9.1s
[CV] n_estimators=1000, min_samples_split=2, min_samples_leaf=1, max_features=sqrt, max_depth=25
[CV]  n_estimators=1000, min_samples_split=2, min_samples_leaf=1, max_features=sqrt, max_depth=25, total=  10.0s
[CV] n_estimators=1000, min_samples_split=2, min_samples_leaf=1, max_features=sqrt, max_depth=25
[CV]  n_estimators=1000, min_samples_split=2, min_samples_leaf=1, max_features=sqrt, max_depth=25, total=   9.7s
[CV] n_estimators=1000, min_samples_split=2, min_samples_leaf=1, max_features=sqrt, max_depth=25
[CV]  n_estimators=1000, min_samples_split=2, min_samples_leaf=1, max_features=sqrt, max_depth=25, total=   9.3s
[CV] n_estimators=1000, min_samples_split=2, min_samples_leaf=1, max_features=sqrt, max_depth=25
[CV]  n_estimators=1000, min_samples_split=2, min_samples_leaf=1, max_features=sqrt, max_depth=25, total=   9.3s
[CV] n_estimators=1000, min_samples_split=2, min_samples_leaf=1, max_features=sqrt, max_depth=25
[CV]  n_estimators=1000, min_samples_split=2, min_samples_leaf=1, max_features=sqrt, max_depth=25, total=   9.2s
[CV] n_estimators=1100, min_samples_split=15, min_samples_leaf=10, max_features=sqrt, max_depth=5
[CV]  n_estimators=1100, min_samples_split=15, min_samples_leaf=10, max_features=sqrt, max_depth=5, total=   2.9s
[CV] n_estimators=1100, min_samples_split=15, min_samples_leaf=10, max_features=sqrt, max_depth=5
[CV]  n_estimators=1100, min_samples_split=15, min_samples_leaf=10, max_features=sqrt, max_depth=5, total=   3.2s
[CV] n_estimators=1100, min_samples_split=15, min_samples_leaf=10, max_features=sqrt, max_depth=5
[CV]  n_estimators=1100, min_samples_split=15, min_samples_leaf=10, max_features=sqrt, max_depth=5, total=   3.0s
[CV] n_estimators=1100, min_samples_split=15, min_samples_leaf=10, max_features=sqrt, max_depth=5
[CV]  n_estimators=1100, min_samples_split=15, min_samples_leaf=10, max_features=sqrt, max_depth=5, total=   3.1s
[CV] n_estimators=1100, min_samples_split=15, min_samples_leaf=10, max_features=sqrt, max_depth=5
[CV]  n_estimators=1100, min_samples_split=15, min_samples_leaf=10, max_features=sqrt, max_depth=5, total=   3.3s
[CV] n_estimators=300, min_samples_split=15, min_samples_leaf=1, max_features=sqrt, max_depth=15
[CV]  n_estimators=300, min_samples_split=15, min_samples_leaf=1, max_features=sqrt, max_depth=15, total=   1.5s
[CV] n_estimators=300, min_samples_split=15, min_samples_leaf=1, max_features=sqrt, max_depth=15
[CV]  n_estimators=300, min_samples_split=15, min_samples_leaf=1, max_features=sqrt, max_depth=15, total=   1.4s
[CV] n_estimators=300, min_samples_split=15, min_samples_leaf=1, max_features=sqrt, max_depth=15
[CV]  n_estimators=300, min_samples_split=15, min_samples_leaf=1, max_features=sqrt, max_depth=15, total=   1.4s
[CV] n_estimators=300, min_samples_split=15, min_samples_leaf=1, max_features=sqrt, max_depth=15
[CV]  n_estimators=300, min_samples_split=15, min_samples_leaf=1, max_features=sqrt, max_depth=15, total=   1.6s
[CV] n_estimators=300, min_samples_split=15, min_samples_leaf=1, max_features=sqrt, max_depth=15
[CV]  n_estimators=300, min_samples_split=15, min_samples_leaf=1, max_features=sqrt, max_depth=15, total=   1.5s
[CV] n_estimators=700, min_samples_split=10, min_samples_leaf=2, max_features=sqrt, max_depth=5
[CV]  n_estimators=700, min_samples_split=10, min_samples_leaf=2, max_features=sqrt, max_depth=5, total=   2.0s
[CV] n_estimators=700, min_samples_split=10, min_samples_leaf=2, max_features=sqrt, max_depth=5
[CV]  n_estimators=700, min_samples_split=10, min_samples_leaf=2, max_features=sqrt, max_depth=5, total=   2.1s
[CV] n_estimators=700, min_samples_split=10, min_samples_leaf=2, max_features=sqrt, max_depth=5
[CV]  n_estimators=700, min_samples_split=10, min_samples_leaf=2, max_features=sqrt, max_depth=5, total=   2.0s
[CV] n_estimators=700, min_samples_split=10, min_samples_leaf=2, max_features=sqrt, max_depth=5
[CV]  n_estimators=700, min_samples_split=10, min_samples_leaf=2, max_features=sqrt, max_depth=5, total=   2.1s
[CV] n_estimators=700, min_samples_split=10, min_samples_leaf=2, max_features=sqrt, max_depth=5
[CV]  n_estimators=700, min_samples_split=10, min_samples_leaf=2, max_features=sqrt, max_depth=5, total=   1.9s
[CV] n_estimators=700, min_samples_split=15, min_samples_leaf=1, max_features=auto, max_depth=20
[CV]  n_estimators=700, min_samples_split=15, min_samples_leaf=1, max_features=auto, max_depth=20, total=  10.9s
[CV] n_estimators=700, min_samples_split=15, min_samples_leaf=1, max_features=auto, max_depth=20
[CV]  n_estimators=700, min_samples_split=15, min_samples_leaf=1, max_features=auto, max_depth=20, total=  10.9s
[CV] n_estimators=700, min_samples_split=15, min_samples_leaf=1, max_features=auto, max_depth=20
[CV]  n_estimators=700, min_samples_split=15, min_samples_leaf=1, max_features=auto, max_depth=20, total=  11.1s
[CV] n_estimators=700, min_samples_split=15, min_samples_leaf=1, max_features=auto, max_depth=20
[CV]  n_estimators=700, min_samples_split=15, min_samples_leaf=1, max_features=auto, max_depth=20, total=  11.0s
[CV] n_estimators=700, min_samples_split=15, min_samples_leaf=1, max_features=auto, max_depth=20
[CV]  n_estimators=700, min_samples_split=15, min_samples_leaf=1, max_features=auto, max_depth=20, total=  11.1s

[Parallel(n_jobs=1)]: Done  50 out of  50 | elapsed:  4.9min finished
```

Out[76]:
```
RandomizedSearchCV(cv=5, error_score=nan,
                   estimator=RandomForestRegressor(bootstrap=True,
                                                   ccp_alpha=0.0,
                                                   criterion='mse',
                                                   max_depth=None,
                                                   max_features='auto',
                                                   max_leaf_nodes=None,
                                                   max_samples=None,
                                                   min_impurity_decrease=0.0,
                                                   min_impurity_split=None,
                                                   min_samples_leaf=1,
                                                   min_samples_split=2,
                                                   min_weight_fraction_leaf=0.0,
                                                   n_estimators=100,
                                                   n_jobs=None, oob_score=Fals...
                   iid='deprecated', n_iter=10, n_jobs=1,
                   param_distributions={'max_depth': [5, 10, 15, 20, 25, 30],
                                        'max_features': ['auto', 'sqrt'],
```

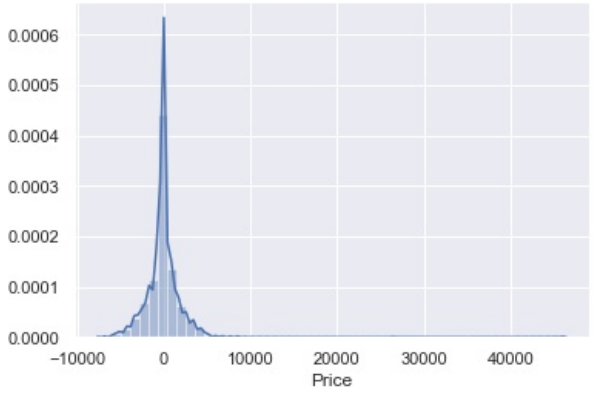
```
                                'min_samples_leaf': [1, 2, 5, 10],
                                'min_samples_split': [2, 5, 10, 15,
                                                      100],
                                'n_estimators': [100, 200, 300, 400,
                                                 500, 600, 700, 800,
                                                 900, 1000, 1100,
                                                 1200]},
                   pre_dispatch='2*n_jobs', random_state=42, refit=True,
                   return_train_score=False, scoring='neg_mean_squared_error',
                   verbose=2)
```

In [77]:
```
rf_random.best_params_
```

Out[77]:
```
{'n_estimators': 700,
 'min_samples_split': 15,
 'min_samples_leaf': 1,
 'max_features': 'auto',
 'max_depth': 20}
```

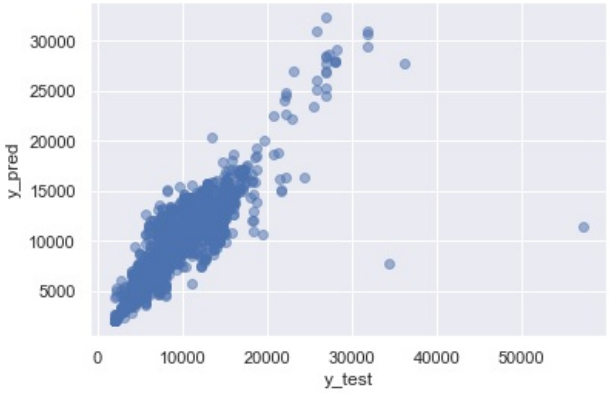
compare y_test and y_pred value using distplot

In [78]:
```
sns.distplot(y_test-y_pred)
plt.show()
```



And scatter plot

In [79]:
```
plt.scatter(y_test, y_pred, alpha = 0.5)
plt.xlabel("y_test")
plt.ylabel("y_pred")
plt.show()
```



Model Evalution

In [80]:
```
from sklearn.metrics import mean_absolute_error,mean_squared_error,r2_score
```

check mean_absolute_error

In [81]:
```
mean_absolute_error(y_test, y_pred)
```

Out[81]: 1179.9788104872175

check mean_squared_error

In [82]:
```
mean_squared_error(y_test, y_pred)
```

Out[82]: 4349400.741053828

check r2_score

In [83]:
```
r2_score(y_test, y_pred)
```

Out[83]: 0.798284510731937

In [ ]:
```

```

In [ ]:
```

```