In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.simplefilter('ignore')
```

In [2]:

```
df = pd.read_excel('Data_Train.xlsx')
```

In [3]:

```
df.head()
```

Out[3]:

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

In [4]:

```
df.shape
```

Out[4]:

```
(10683, 11)
```

In [5]:
```
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
```

Missing values

In [6]:
```
df.isna().sum()
```

Out[6]:
```
Airline            0
Date_of_Journey    0
Source             0
Destination        0
Route              1
Dep_Time           0
Arrival_Time       0
Duration           0
Total_Stops        1
Additional_Info    0
Price              0
dtype: int64
```

In [7]:
```
a = df.isna().any()
na_col = a[a].index
na_col
```

Out[7]:
```
Index(['Route', 'Total_Stops'], dtype='object')
```

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

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

Out[9]:
```
220
```

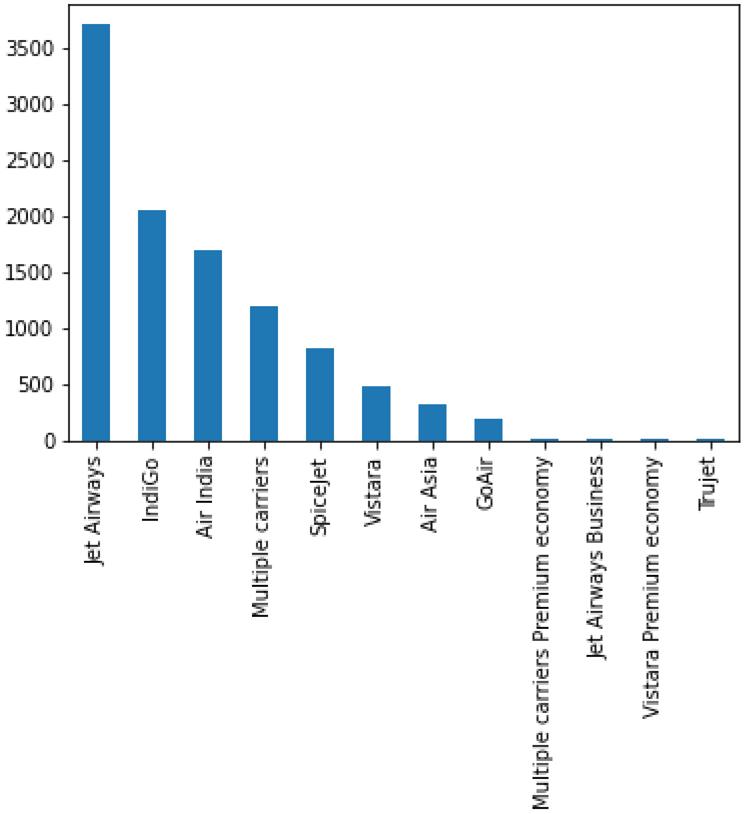
There are 220 rows which are duplicates.

In [10]:
```
df.drop_duplicates(inplace=True)
```

In [11]:
```
print(df['Airline'].value_counts())
df['Airline'].value_counts().plot(kind = 'bar')
```

```
Jet Airways                          3700
IndiGo                               2043
Air India                            1694
Multiple carriers                    1196
SpiceJet                              815
Vistara                               478
Air Asia                              319
GoAir                                 194
Multiple carriers Premium economy      13
Jet Airways Business                    6
Vistara Premium economy                 3
Trujet                                  1
Name: Airline, dtype: int64
```

Out[11]:
```
<AxesSubplot:>
```



`jet Airways` , `indiGO` , `Air Inida` are the top airlines which passengers prefer.

In [12]:

```
airlines_with_very_less_data= (df['Airline'].value_counts() < 14)
l = airlines_with_very_less_data[airlines_with_very_less_data].index
l
```

Out[12]:

```
Index(['Multiple carriers Premium economy', 'Jet Airways Business',
       'Vistara Premium economy', 'Trujet'],
      dtype='object')
```

these are the airlines which data are very less compare to other airlines. `Multiple carriers Premium economy` , `Jet Airways Business` , `Vistara Premium economy` , `Trujet`

In [13]:

```
#converting string dtype to datetime
df['Date_of_Journey'] = pd.to_datetime(df['Date_of_Journey'])
```

In [14]:

```
df['Date_of_Journey'].dt.year.value_counts()
```

Out[14]:

```
2019    10462
Name: Date_of_Journey, dtype: int64
```

we have only `2019` data

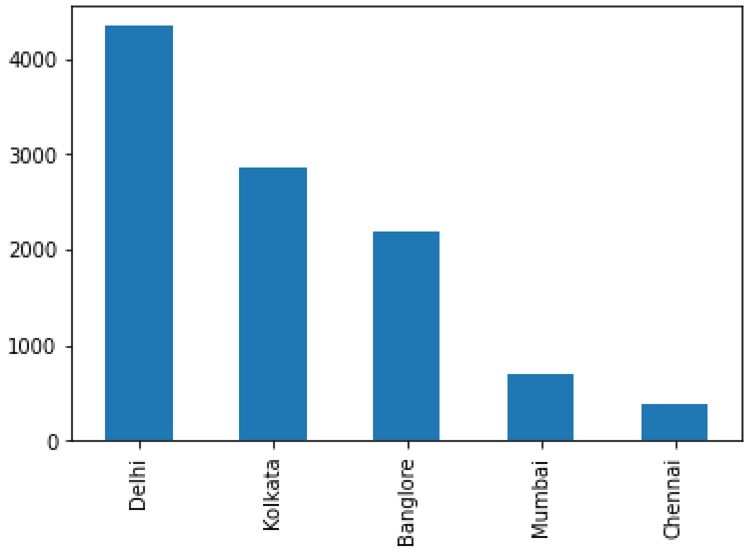
In [15]:

```
print(df['Source'] .value_counts())
df['Source'].value_counts().plot(kind = 'bar')
```

```
Delhi       4345
Kolkata     2860
Banglore    2179
Mumbai       697
Chennai      381
Name: Source, dtype: int64
```

Out[15]:

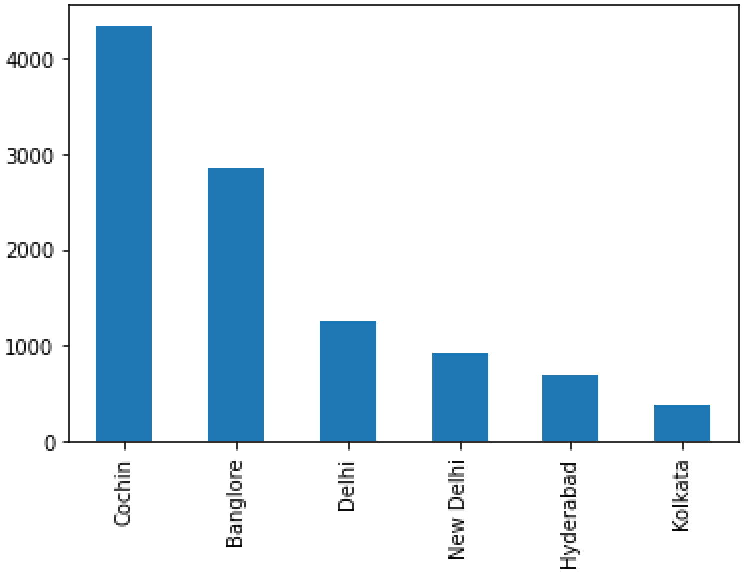
```
<AxesSubplot:>
```



In [16]:
```
print(df['Destination'] .value_counts())
df['Destination'].value_counts().plot(kind = 'bar')
```

```
Cochin        4345
Banglore      2860
Delhi         1265
New Delhi      914
Hyderabad      697
Kolkata        381
Name: Destination, dtype: int64
```

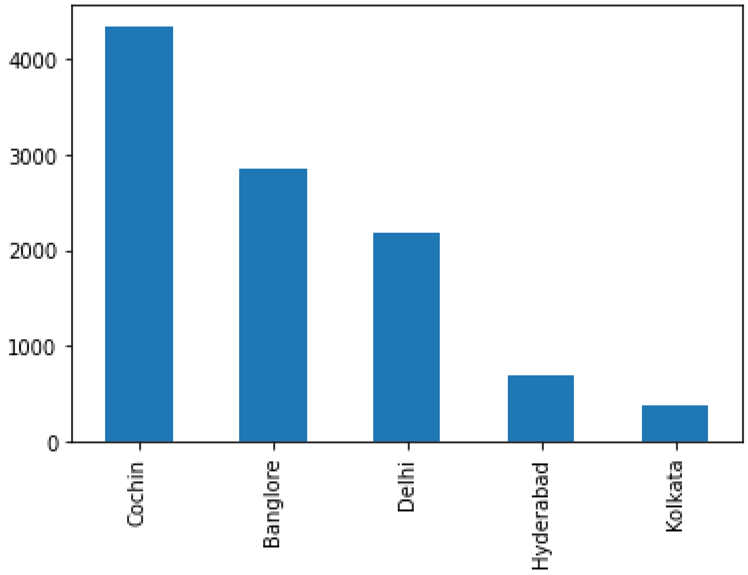
Out[16]:
```
<AxesSubplot:>
```



From the above bar graph we have to destination delhi , new delhi. but in source only delhi is there so we will replace NewDelhi to Delhi.

In [17]:
```
df['Destination'] = df['Destination'].replace('New Delhi' , 'Delhi')
df['Destination'].value_counts().plot(kind = 'bar')
```

Out[17]:
```
<AxesSubplot:>
```



In [18]:
```
df['Route'].value_counts()
```

Out[18]:
```
DEL → BOM → COK          2376
BLR → DEL                1536
CCU → BOM → BLR           979
CCU → BLR                 724
BOM → HYD                 621
                         ...
CCU → VTZ → BLR             1
CCU → IXZ → MAA → BLR       1
BOM → COK → MAA → HYD       1
BOM → CCU → HYD             1
BOM → BBI → HYD             1
Name: Route, Length: 128, dtype: int64
```

There are total 128 different routes.

In [19]:
```
print(df['Total_Stops'].value_counts())
```

```
1 stop      5625
non-stop    3475
2 stops     1318
3 stops       43
4 stops        1
Name: Total_Stops, dtype: int64
```

there are total 5 kind of stops but we have only one data for `4 stops` so remove that.

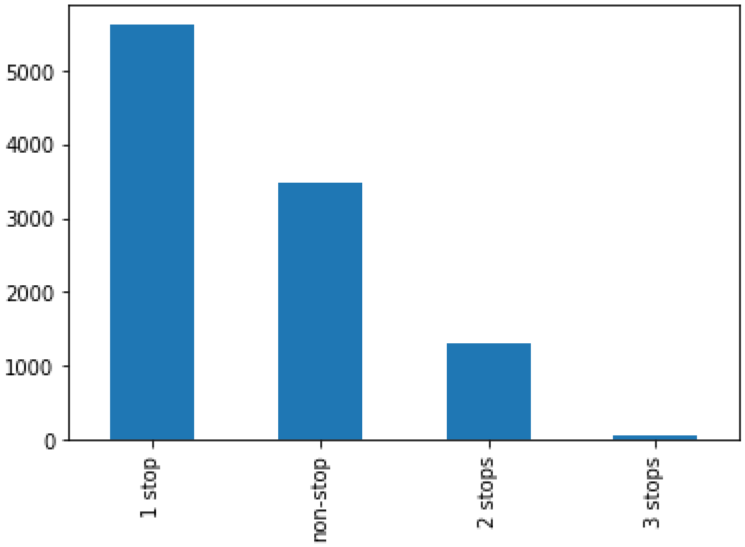
In [20]:
```
df = df[df['Total_Stops']!='4 stops']
```

In [21]:

```
df['Total_Stops'].value_counts().plot(kind = 'bar')
```

Out[21]:

```
<AxesSubplot:>
```



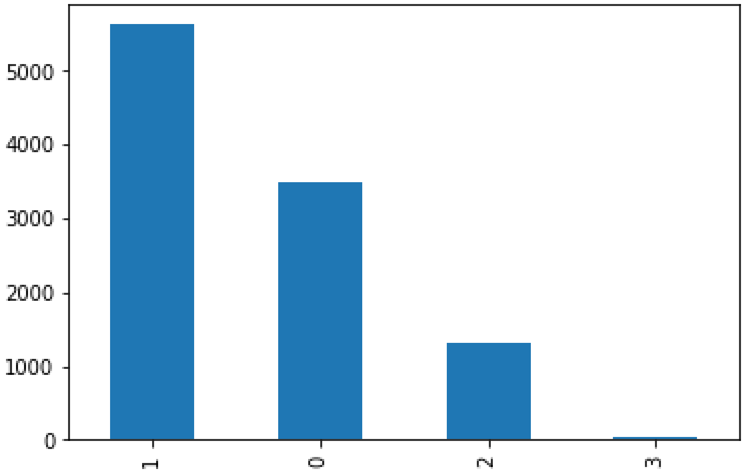
Replacing 1stops = 1 , 2stops= 2 , 3stops = 3 , non-stops = 0

In [22]:

```
df['Total_Stops'] = df['Total_Stops'].str[0].replace('n' , 0).astype('int32')
df['Total_Stops'].value_counts().plot(kind = 'bar')
```

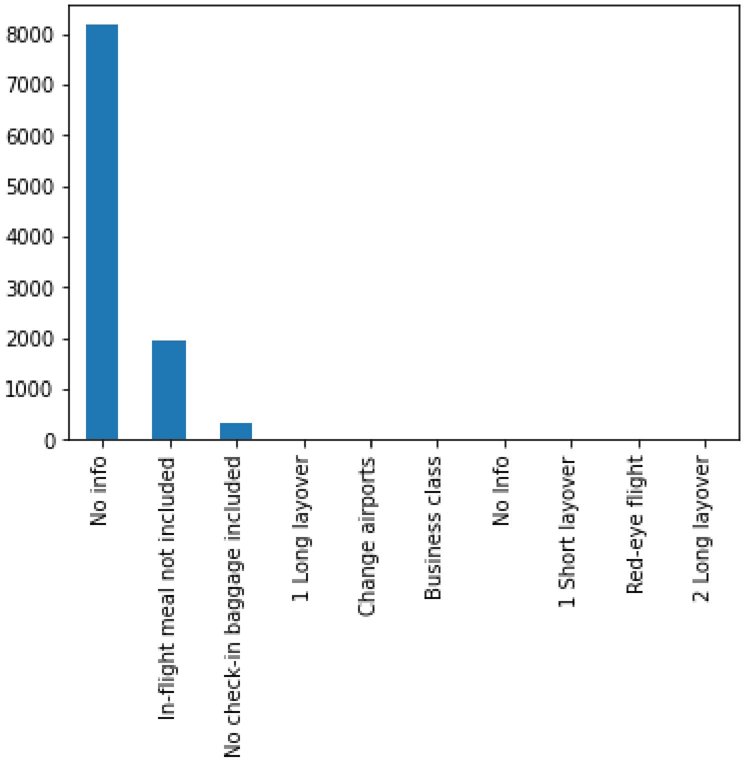
Out[22]:

```
<AxesSubplot:>
```



In [23]:
```
df['Additional_Info'].value_counts().plot(kind = 'bar')
```

Out[23]:
```
<AxesSubplot:>
```



In [24]:
```
df['Dep_Time' ][0:5]
```

Out[24]:
```
0    22:20
1    05:50
2    09:25
3    18:05
4    16:50
Name: Dep_Time, dtype: object
```

In [25]:
```
# Extracting departure hours and departure minutes
df['dep_hour' ] = df['Dep_Time'].str.split(':' , expand = True)[0].astype('int32'
df['dep_min']  =  df['Dep_Time'].str.split(':' , expand = True)[1].astype('int32'
```

In [26]:
```
df['Arrival_Time'][0:5]
```

Out[26]:
```
0    01:10 22 Mar
1           13:15
2    04:25 10 Jun
3           23:30
4           21:35
Name: Arrival_Time, dtype: object
```

In [27]:
```
# Extracting arrival hours and arrival minutes

df['arrival_hour'] = df['Arrival_Time'].str[0:5].str.split(':' , expand = True)[0
df['arrival_min']  = df['Arrival_Time'].str[0:5].str.split(':' , expand = True)[1
```

In [28]:
```
# Extracting Duration hours and duration minutes

df['Duration_hour' ] = df['Duration'].str.replace('h','').str.replace('m' , '').s
df['Duration_min' ] = df['Duration'].str.replace('h','').str.replace('m' , '').st
```

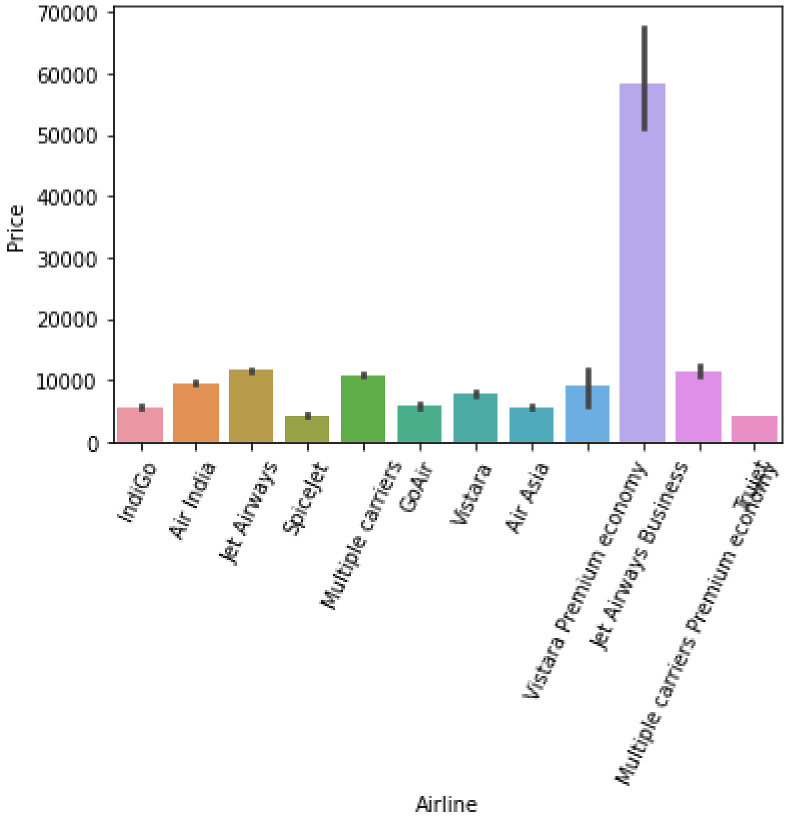
In [29]:
```
#calculation total duration hours
df['total_hours'] = df['Duration_hour'] + df['Duration_min'] / 60
```

EDA

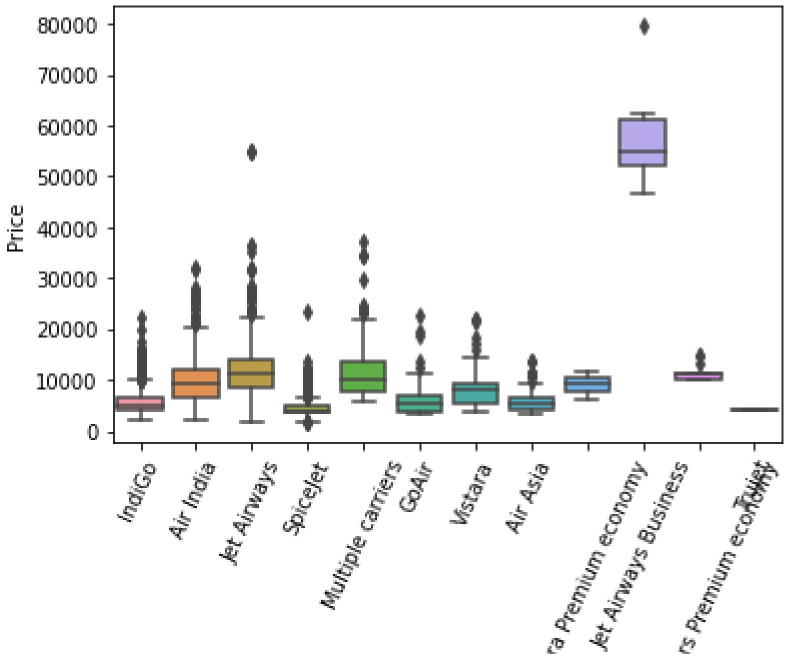
In [30]:
```
df.columns
```

Out[30]:
```
Index(['Airline', 'Date_of_Journey', 'Source', 'Destination', 'Route',
       'Dep_Time', 'Arrival_Time', 'Duration', 'Total_Stops',
       'Additional_Info', 'Price', 'dep_hour', 'dep_min', 'arrival_hour',
       'arrival_min', 'Duration_hour', 'Duration_min', 'total_hours'],
      dtype='object')
```

In [31]:
```
sns.barplot(x = 'Airline' , y = 'Price'  , data = df )
plt.xticks(rotation = 65)
plt.show()
```

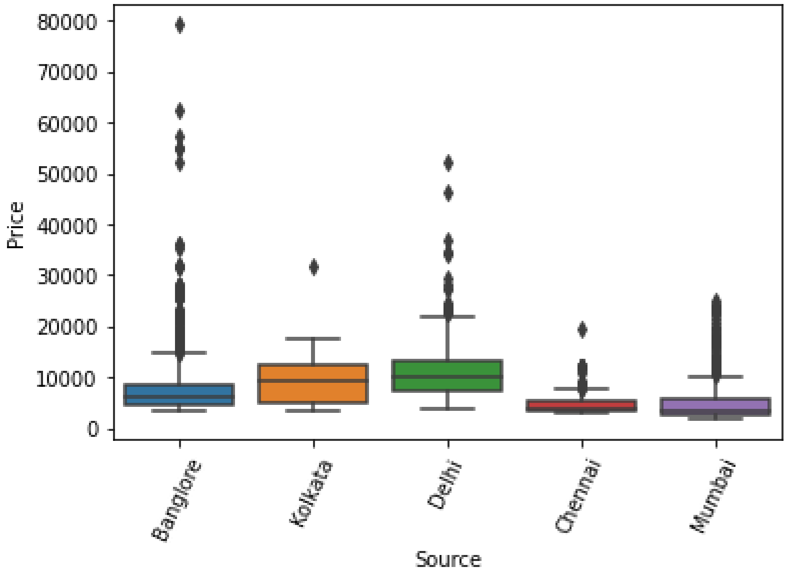


In [32]:
```
sns.boxplot(x = 'Airline' , y = 'Price' , data = df )
plt.xticks(rotation = 65)
plt.show()
```

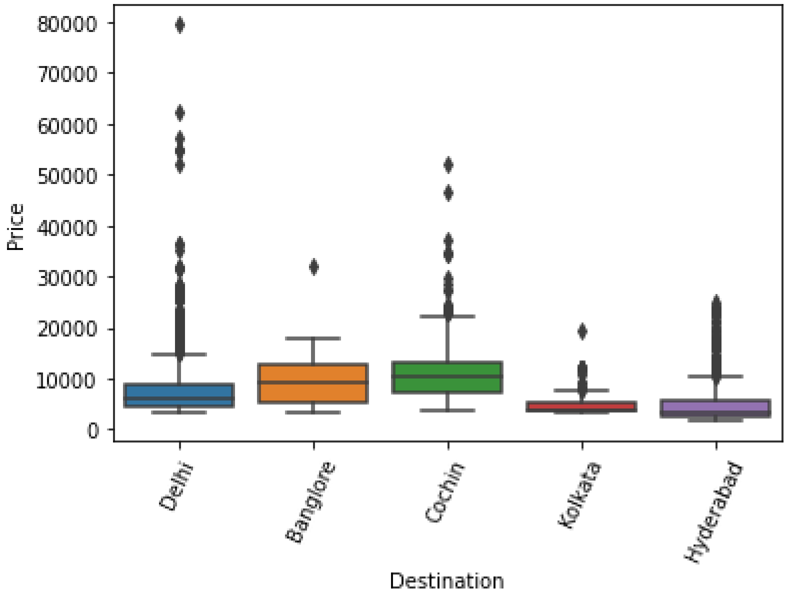


From graph we can see that Jet Airways Business have the highest Price., Apart from the Jet Airways Business almost all are having similar median

In [33]:
```
sns.boxplot(x = 'Source' , y = 'Price' , data = df )
plt.xticks(rotation = 65)
plt.show()
```



In [34]:
```
sns.boxplot(x = 'Destination' , y = 'Price' , data = df )
plt.xticks(rotation = 65)
plt.show()
```



In [35]:
```
df_blr_del = df[df['Route']=='BLR → DEL']
```

Lets analyse flight fare price for banglore to delhi with different airline.

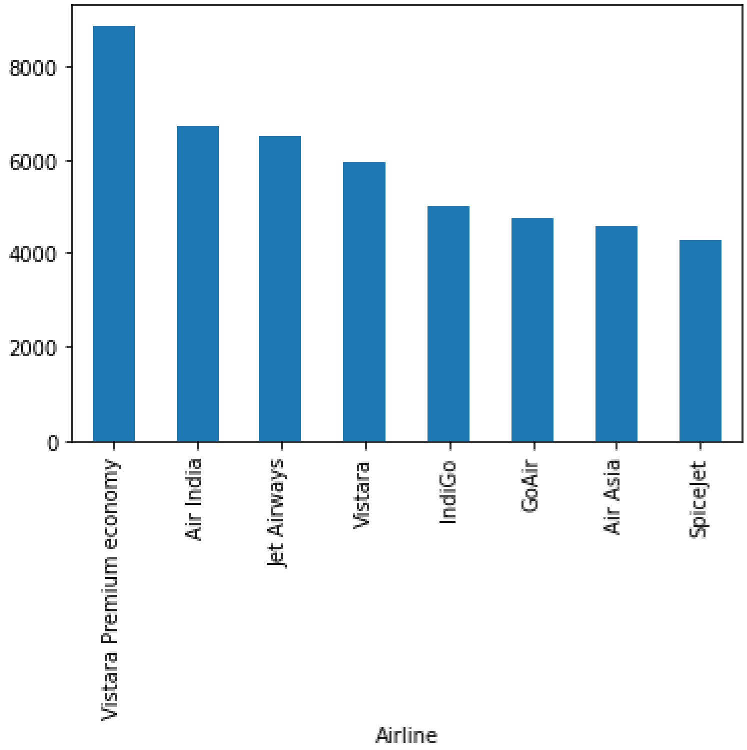
In [36]:

```
print(df_blr_del.groupby('Airline')['Price'].mean().sort_values(ascending = False
df_blr_del.groupby('Airline')['Price'].mean().sort_values(ascending = False).plot
```

```
Airline
Vistara Premium economy    8881.000000
Air India                  6716.757962
Jet Airways                6498.803150
Vistara                    5960.674286
IndiGo                     5023.526427
GoAir                      4767.033708
Air Asia                   4574.280899
SpiceJet                   4289.847059
Name: Price, dtype: float64
```

Out[36]:

```
<AxesSubplot:xlabel='Airline'>
```

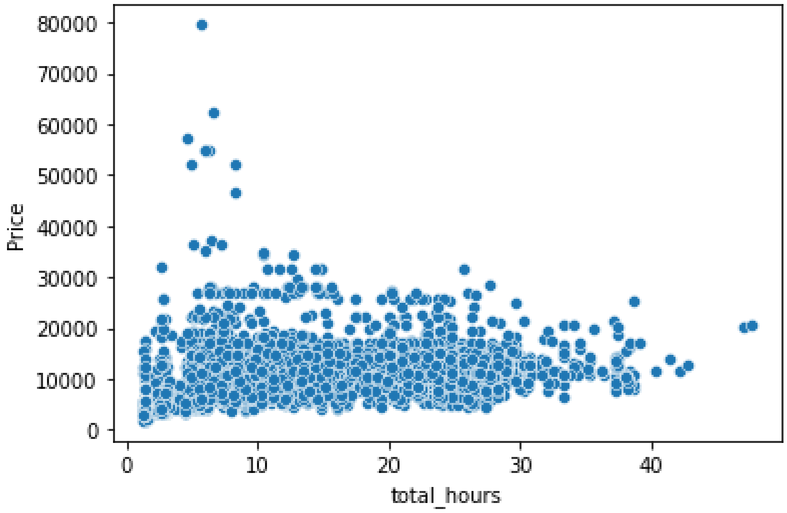


`Vistara premium economy` airline fare price is quite high comparing with other airlines. SpiceJet fare is very cheap.

BLG->DEL average fare is almost 6500+

```
In [37]: sns.scatterplot(x = 'total_hours' , y = 'Price' , data = df)
```

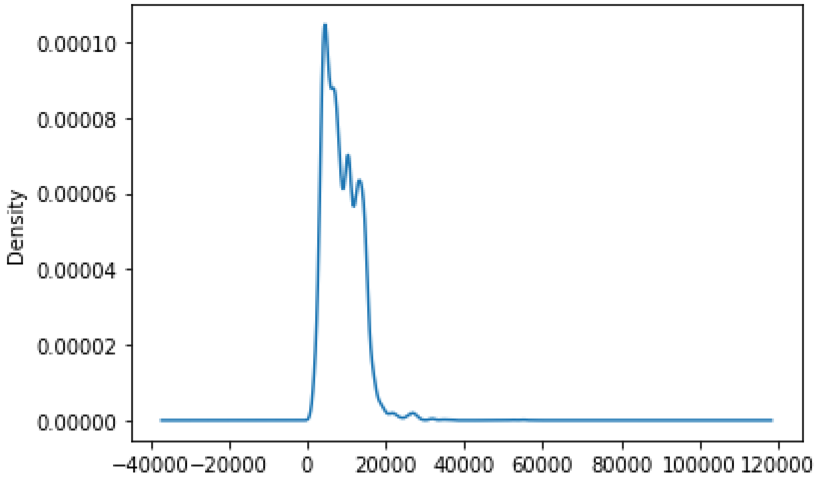
```
Out[37]: <AxesSubplot:xlabel='total_hours', ylabel='Price'>
```



Outliers

```
In [38]: df['Price'].plot(kind = 'kde')
```

```
Out[38]: <AxesSubplot:ylabel='Density'>
```

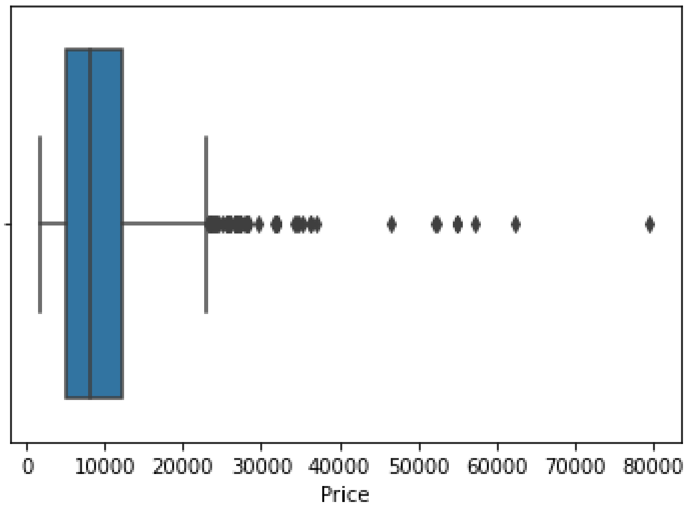


In [39]:

```
sns.boxplot(x = 'Price' , data = df)
```

Out[39]:

```
<AxesSubplot:xlabel='Price'>
```



In [40]:

```
df.describe()
```

Out[40]:

	Total_Stops	Price	dep_hour	dep_min	arrival_hour	arrival_min	Durati
count	10461.000000	10461.000000	10461.000000	10461.000000	10461.000000	10461.000000	1046
mean	0.802027	9025.962527	12.479208	24.402543	13.387917	24.720390	1
std	0.659900	4624.295514	5.727034	18.814954	6.855835	16.571178	
min	0.000000	1759.000000	0.000000	0.000000	0.000000	0.000000	
25%	0.000000	5224.000000	8.000000	5.000000	8.000000	10.000000	
50%	1.000000	8266.000000	11.000000	25.000000	14.000000	25.000000	
75%	1.000000	12341.000000	18.000000	40.000000	19.000000	35.000000	1
max	3.000000	79512.000000	23.000000	55.000000	23.000000	55.000000	4

In [41]:

```
df['Price'] = np.where(df['Price']>40000 , df['Price'].median() , df['Price'])
```

In [42]:

```
df['month'] = df['Date_of_Journey'].dt.month
df['date']  =df['Date_of_Journey'].dt.day
```

In [43]:
```
cols = ['Date_of_Journey' , 'Route' , 'Dep_Time' , 'Arrival_Time' , 'Duration' ,
final_df = df.drop(columns=cols , axis = True )
final_df.head()
```

Out[43]:

	Airline	Source	Destination	Total_Stops	Price	dep_hour	dep_min	arrival_hour	arrival_m
0	IndiGo	Banglore	Delhi	0	3897.0	22	20	1	1
1	Air India	Kolkata	Banglore	2	7662.0	5	50	13	1
2	Jet Airways	Delhi	Cochin	2	13882.0	9	25	4	2
3	IndiGo	Kolkata	Banglore	1	6218.0	18	5	23	3
4	IndiGo	Banglore	Delhi	1	13302.0	16	50	21	3

In [44]:
```
final_df = pd.get_dummies(final_df , drop_first=True)
```

In [45]:
```
final_df.head()
```

Out[45]:

	Total_Stops	Price	dep_hour	dep_min	arrival_hour	arrival_min	total_hours	month	date	A
0	0	3897.0	22	20	1	10	2.833333	3	24	
1	2	7662.0	5	50	13	15	7.416667	1	5	
2	2	13882.0	9	25	4	25	19.000000	9	6	
3	1	6218.0	18	5	23	30	5.416667	12	5	
4	1	13302.0	16	50	21	35	4.750000	1	3	

5 rows × 28 columns

Correlation

In [46]:

```
final_df.head()
```

Out[46]:

	Total_Stops	Price	dep_hour	dep_min	arrival_hour	arrival_min	total_hours	month	date	A
0	0	3897.0	22	20	1	10	2.833333	3	24	
1	2	7662.0	5	50	13	15	7.416667	1	5	
2	2	13882.0	9	25	4	25	19.000000	9	6	
3	1	6218.0	18	5	23	30	5.416667	12	5	
4	1	13302.0	16	50	21	35	4.750000	1	3	

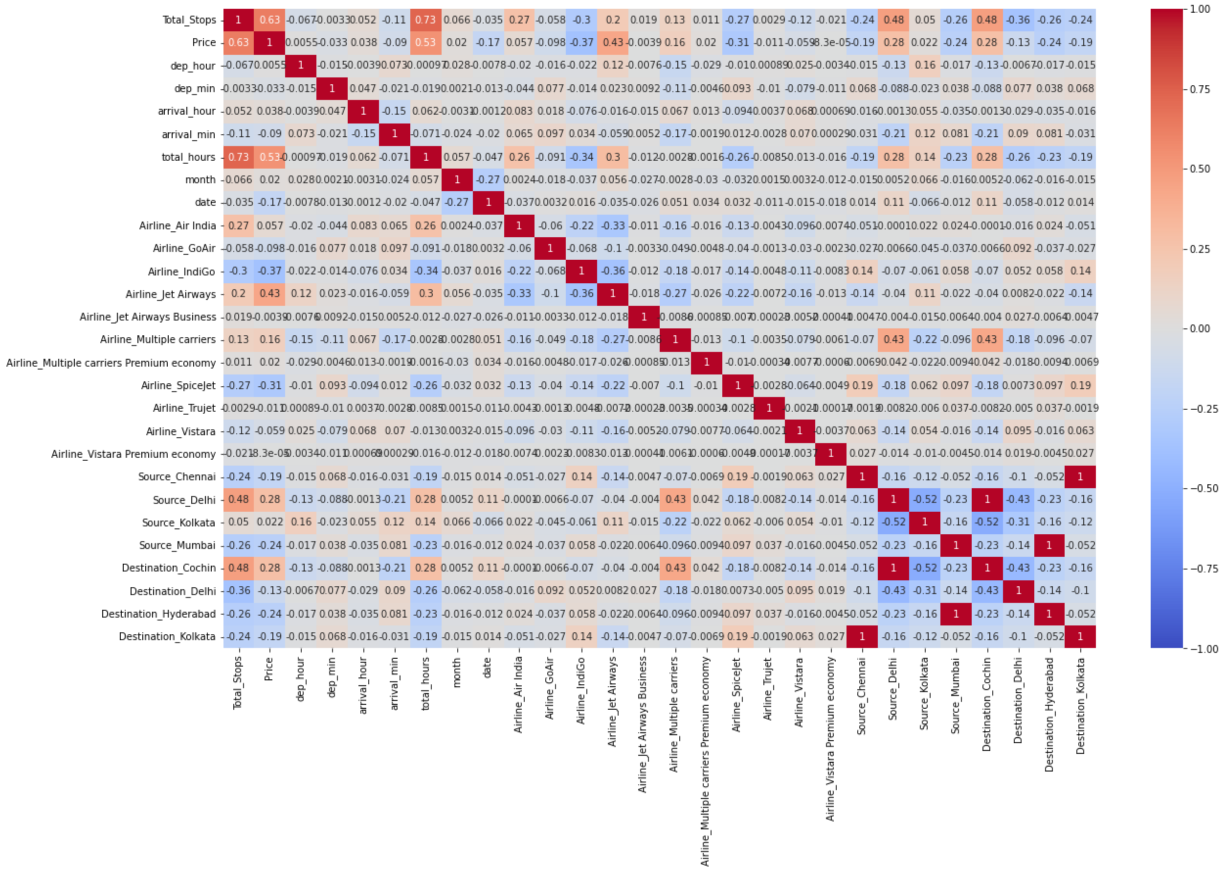
5 rows × 28 columns

In [47]:

```
plt.figure(figsize= (20,12))
sns.heatmap(final_df.corr() , annot = True , cmap = 'coolwarm' , vmin = -1 , vma
```

Out[47]:

```
<AxesSubplot:>
```



In [48]:
```
final_df.corr()['Price'].sort_values(ascending = False )[1:]
```

Out[48]:
```
Total_Stops                                   0.627674
total_hours                                   0.532665
Airline_Jet Airways                           0.433802
Destination_Cochin                            0.279750
Source_Delhi                                  0.279750
Airline_Multiple carriers                     0.156851
Airline_Air India                             0.056779
arrival_hour                                  0.038111
Source_Kolkata                                0.022219
month                                         0.020392
Airline_Multiple carriers Premium economy     0.019542
dep_hour                                      0.005497
Airline_Vistara Premium economy              -0.000083
Airline_Jet Airways Business                 -0.003913
Airline_Trujet                               -0.010776
dep_min                                      -0.032893
Airline_Vistara                              -0.058872
arrival_min                                  -0.090399
Airline_GoAir                                -0.097666
Destination_Delhi                            -0.131812
date                                         -0.172852
Source_Chennai                               -0.185513
Destination_Kolkata                          -0.185513
Source_Mumbai                                -0.238543
Destination_Hyderabad                        -0.238543
Airline_SpiceJet                             -0.307391
Airline_IndiGo                               -0.371606
Name: Price, dtype: float64
```

In [ ]:
```

```

In [49]:
```
# dropping price columns
X = final_df.drop('Price' , 1 )
# standarization of price columns
y = np.log1p(final_df['Price'])
```

In [50]:
```
#train test split with 33% test values
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_
```

In [51]:
```
X_train.shape , X_test.shape
```

Out[51]:
```
((7008, 27), (3453, 27))
```

In [52]:
```
from sklearn.linear_model import LinearRegression,Ridge,Lasso
from sklearn.neighbors import KNeighborsRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor,GradientBoostingRegressor,AdaB
from sklearn.metrics import r2_score , mean_absolute_error , mean_absolute_percen
```

In [53]:

```
def model_feature(model):
    model.fit(X_train , y_train)
    y_pred = model.predict(X_test)
    print(str(model)[0 : -2] + ' ' 'Model')
    print('r2_score:{}'.format(round(r2_score(y_test , y_pred) , 2)))
    print('MAE',round(mean_absolute_error(y_test , y_pred) , 2))
    print('MAPE' ,  round(mean_absolute_percentage_error(y_test , y_pred) , 2))
    print('MSE' , round(mean_squared_error(y_test , y_pred) , 2))
```

LinearRegression

In [54]:

```
model_feature(LinearRegression())
```

```
LinearRegression Model
r2_score:0.69
MAE 0.21
MAPE 0.02
MSE 0.08
```

Lasso

In [55]:

```
model_feature(Lasso())
```

```
Lasso Model
r2_score:0.31
MAE 0.34
MAPE 0.04
MSE 0.18
```

Ridge

In [56]:

```
model_feature(Ridge())
```

```
Ridge Model
r2_score:0.69
MAE 0.21
MAPE 0.02
MSE 0.08
```

KNeighborsRegressor

In [57]:
```
model_feature(KNeighborsRegressor())
```

```
KNeighborsRegressor Model
r2_score:0.7
MAE 0.21
MAPE 0.02
MSE 0.08
```

RandomForestRegressor

In [58]:
```
model_feature(RandomForestRegressor())
```

```
RandomForestRegressor Model
r2_score:0.86
MAE 0.13
MAPE 0.01
MSE 0.04
```

GradientBoostingRegressor

In [59]:
```
model_feature(GradientBoostingRegressor())
```

```
GradientBoostingRegressor Model
r2_score:0.83
MAE 0.16
MAPE 0.02
MSE 0.04
```

AdaBoostRegressor

In [60]:
```
model_feature(AdaBoostRegressor())
```

```
AdaBoostRegressor Model
r2_score:0.71
MAE 0.22
MAPE 0.03
MSE 0.07
```

ExtraTreesRegressor

In [61]:
```
model_feature(ExtraTreesRegressor())
```

```
ExtraTreesRegressor Model
r2_score:0.82
MAE 0.14
MAPE 0.02
MSE 0.05
```

VotingRegressor

In [62]:
```
from sklearn.ensemble import VotingRegressor,StackingRegressor


rf = RandomForestRegressor(n_estimators=350,random_state=3,max_samples=0.5,max_fe
gbdt = GradientBoostingRegressor(n_estimators=100,max_features=0.5)
# xgb = XGBRegressor(n_estimators=25,learning_rate=0.3,max_depth=5)
et = ExtraTreesRegressor(n_estimators=100,random_state=3,max_samples=0.5,max_feat

model_vr = VotingRegressor([('rf', rf), ('gbdt', gbdt), ('et',et)],weights=[5,1,

model_vr.fit(X_train,y_train)

y_pred = model_vr.predict(X_test)

print('R2 score',r2_score(y_test,y_pred))
print('MAE',mean_absolute_error(y_test,y_pred))
```

```
R2 score 0.8695471971376612
MAE 0.13300083528164577
```

StackingRegressor

In [63]:
```
from sklearn.ensemble import VotingRegressor,StackingRegressor

estimators = [
    ('rf', RandomForestRegressor(n_estimators=350,random_state=3,max_samples=0.5,
    ('gbdt',GradientBoostingRegressor(n_estimators=100,max_features=0.5)),
]

pipe = StackingRegressor(estimators=estimators, final_estimator=Ridge(alpha=100))

pipe.fit(X_train,y_train)

y_pred = pipe.predict(X_test)

print('R2 score',r2_score(y_test,y_pred))
print('MAE',mean_absolute_error(y_test,y_pred))
```

```
R2 score 0.866720440181483
MAE 0.13688271524270923
```

RandomForestRegressor

In [64]:
```
model_rfr = RandomForestRegressor()
model_rfr.fit(X_train , y_train)
model_rfr.predict(X_test)
r2_score(y_test,y_pred)
```

Out[64]:
```
0.866720440181483
```

In [65]:
```
model_list = [LinearRegression() , Ridge() , Lasso() , KNeighborsRegressor() , De
model_list1 = []
R2_score = []
mae = []
mape = []
mse = []

for model in model_list:
    model_list1.append(str(model)[0:-2])
    model.fit(X_train , y_train)
    y_pred = model.predict(X_test)
    R2_score.append(round(r2_score(y_test , y_pred) , 2))
    mae.append(round(mean_absolute_error(y_test , y_pred) , 2))
    mape.append(round(mean_absolute_percentage_error(y_test , y_pred) , 2))
    mse.append(round(mean_squared_error(y_test , y_pred) , 2))
```

In [66]:
```
dict = {'Model':model_list1, 'R2_score':R2_score , 'MAPE':mape , 'MAE':mae , 'MSE
model_df = pd.DataFrame(dict).sort_values(ascending = False , by = 'R2_score')
model_df
```

Out[66]:

	Model	R2_score	MAPE	MAE	MSE
5	RandomForestRegressor	0.86	0.01	0.13	0.04
6	GradientBoostingRegressor	0.83	0.02	0.16	0.04
8	ExtraTreesRegressor	0.82	0.02	0.14	0.05
4	DecisionTreeRegressor	0.77	0.02	0.15	0.06
7	AdaBoostRegressor	0.71	0.03	0.23	0.08
3	KNeighborsRegressor	0.70	0.02	0.21	0.08
0	LinearRegression	0.69	0.02	0.21	0.08
1	Ridge	0.69	0.02	0.21	0.08
2	Lasso	0.31	0.04	0.34	0.18

we are getting Best R2_score with RandomForestRegressor

Exporting the Model

In [67]:

```
import pickle

pickle.dump(final_df,open('final_df.pkl','wb'))
pickle.dump(model_rfr,open('model.pkl','wb'))
```

In [ ]:

```

```