# project no 1 - flight booking

In [1]:
```
# import all nessasary libraries

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

In [4]:
```
# import dataset in workbook by using pandas

df=pd.read_csv("C:/Users/Prince/Desktop/DATA EINSTEIN/industrial projects/project 1 filght booking/Flight_booking.csv")
df.head()
```

Out[4]:

| | Unnamed: 0 | airline | flight | source_city | departure_time | stops | arrival_time | destination_city | class | duration | days_left | price |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 0 | SpiceJet | SG-8709 | Delhi | Evening | zero | Night | Mumbai | Economy | 2.17 | 1 | 5953 |
| **1** | 1 | SpiceJet | SG-8157 | Delhi | Early_Morning | zero | Morning | Mumbai | Economy | 2.33 | 1 | 5953 |
| **2** | 2 | AirAsia | I5-764 | Delhi | Early_Morning | zero | Early_Morning | Mumbai | Economy | 2.17 | 1 | 5956 |
| **3** | 3 | Vistara | UK-995 | Delhi | Morning | zero | Afternoon | Mumbai | Economy | 2.25 | 1 | 5955 |
| **4** | 4 | Vistara | UK-963 | Delhi | Morning | zero | Morning | Mumbai | Economy | 2.33 | 1 | 5955 |

In [5]:
```
# shape of data (rows, columns)

df.shape
```

Out[5]: (300153, 12)

In [6]:
```
# remove unwanted column from dataset

df.drop(["Unnamed: 0"], axis=1, inplace=True)
df
```

Out[6]:

| | airline | flight | source_city | departure_time | stops | arrival_time | destination_city | class | duration | days_left | price |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | SpiceJet | SG-8709 | Delhi | Evening | zero | Night | Mumbai | Economy | 2.17 | 1 | 5953 |
| **1** | SpiceJet | SG-8157 | Delhi | Early_Morning | zero | Morning | Mumbai | Economy | 2.33 | 1 | 5953 |
| **2** | AirAsia | I5-764 | Delhi | Early_Morning | zero | Early_Morning | Mumbai | Economy | 2.17 | 1 | 5956 |
| **3** | Vistara | UK-995 | Delhi | Morning | zero | Afternoon | Mumbai | Economy | 2.25 | 1 | 5955 |
| **4** | Vistara | UK-963 | Delhi | Morning | zero | Morning | Mumbai | Economy | 2.33 | 1 | 5955 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **300148** | Vistara | UK-822 | Chennai | Morning | one | Evening | Hyderabad | Business | 10.08 | 49 | 69265 |
| **300149** | Vistara | UK-826 | Chennai | Afternoon | one | Night | Hyderabad | Business | 10.42 | 49 | 77105 |
| **300150** | Vistara | UK-832 | Chennai | Early_Morning | one | Night | Hyderabad | Business | 13.83 | 49 | 79099 |
| **300151** | Vistara | UK-828 | Chennai | Early_Morning | one | Evening | Hyderabad | Business | 10.00 | 49 | 81585 |
| **300152** | Vistara | UK-822 | Chennai | Morning | one | Evening | Hyderabad | Business | 10.08 | 49 | 81585 |

300153 rows × 11 columns

In [7]:
```
# checking shape size again to verify

df.shape
```

Out[7]: (300153, 11)

In [8]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 300153 entries, 0 to 300152
Data columns (total 11 columns):
 #   Column            Non-Null Count   Dtype  
---  ------            --------------   -----  
 0   airline           300153 non-null  object 
 1   flight            300153 non-null  object 
 2   source_city       300153 non-null  object 
 3   departure_time    300153 non-null  object 
 4   stops             300153 non-null  object 
 5   arrival_time      300153 non-null  object 
 6   destination_city  300153 non-null  object 
 7   class             300153 non-null  object 
 8   duration          300153 non-null  float64
 9   days_left         300153 non-null  int64  
 10  price             300153 non-null  int64  
dtypes: float64(1), int64(2), object(8)
memory usage: 25.2+ MB
```

In [9]:
```
df.describe()
```

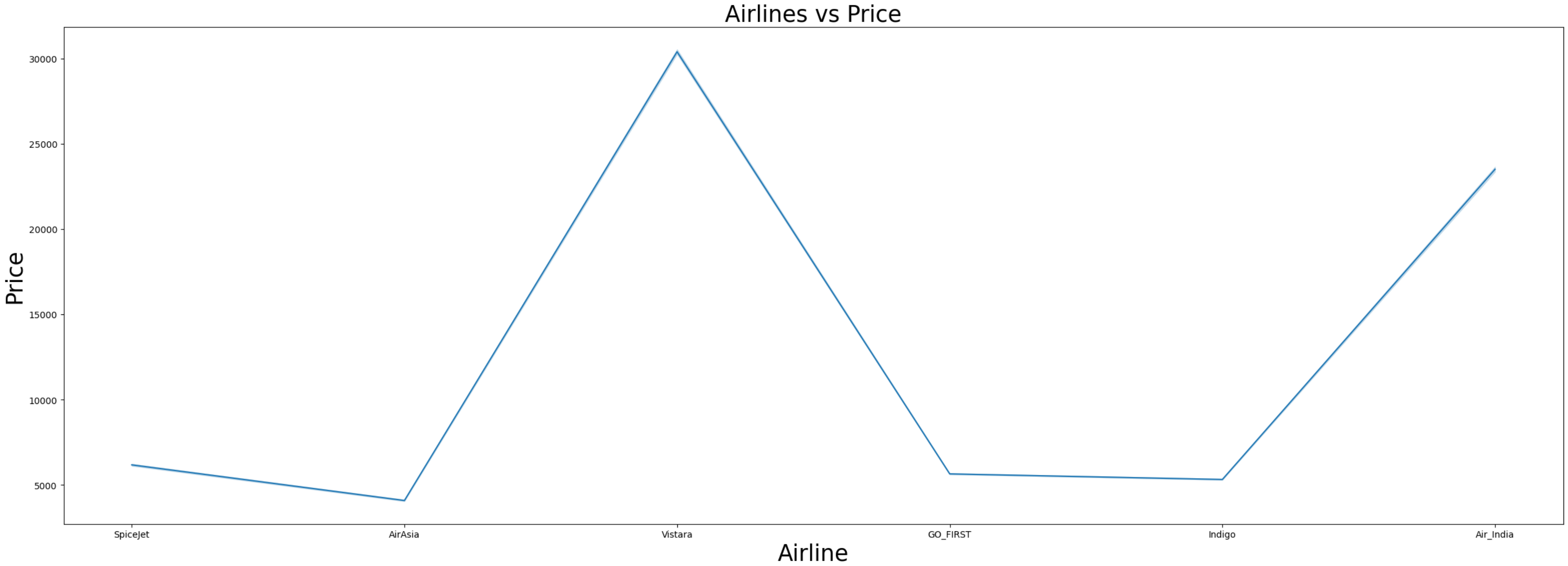
Out[9]:

| | duration | days_left | price |
|---|---|---|---|
| **count** | 300153.000000 | 300153.000000 | 300153.000000 |
| **mean** | 12.221021 | 26.004751 | 20889.660523 |
| **std** | 7.191997 | 13.561004 | 22697.767366 |
| **min** | 0.830000 | 1.000000 | 1105.000000 |
| **25%** | 6.830000 | 15.000000 | 4783.000000 |
| **50%** | 11.250000 | 26.000000 | 7425.000000 |
| **75%** | 16.170000 | 38.000000 | 42521.000000 |
| **max** | 49.830000 | 49.000000 | 123071.000000 |

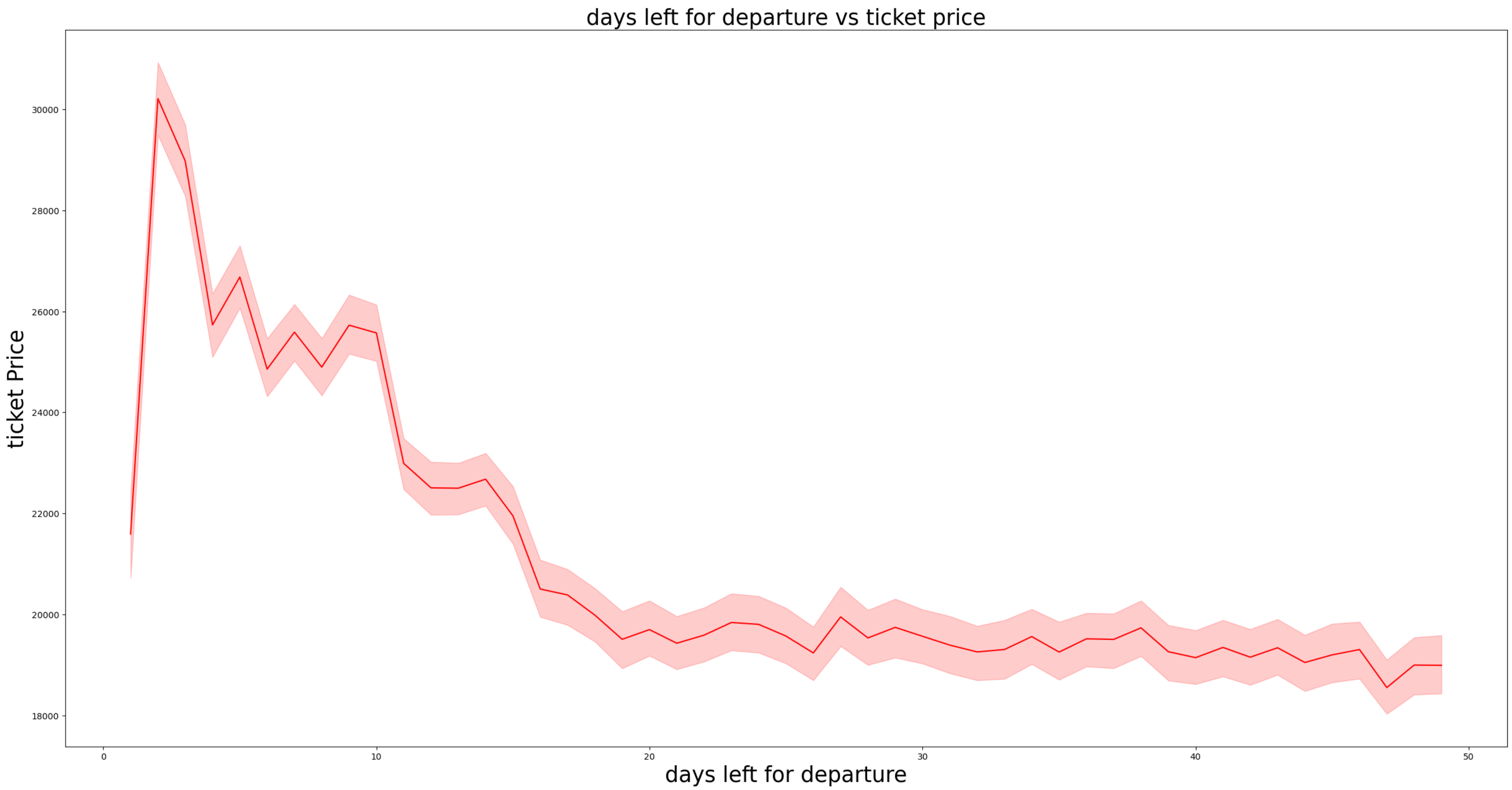
In [10]:
```
# checking null values

df.isnull().sum()
```

Out[10]:
```
airline             0
flight              0
source_city         0
departure_time      0
stops               0
arrival_time        0
destination_city    0
class               0
duration            0
days_left           0
price               0
dtype: int64
```

In [11]:
```
plt.figure(figsize=(30,10))
sns.lineplot(x=df['airline'],y=df['price'])
plt.title('Airlines vs Price',fontsize=25)
plt.xlabel('Airline', fontsize=25)
plt.ylabel('Price', fontsize=25)
plt.show()
```



In [12]:
```
plt.figure(figsize=(30,15))
sns.lineplot(data=df, x='days_left',y='price', color='red')
plt.title('days left for departure vs ticket price',fontsize=25)
plt.xlabel('days left for departure', fontsize=25)
plt.ylabel('ticket Price', fontsize=25)
plt.show()
```

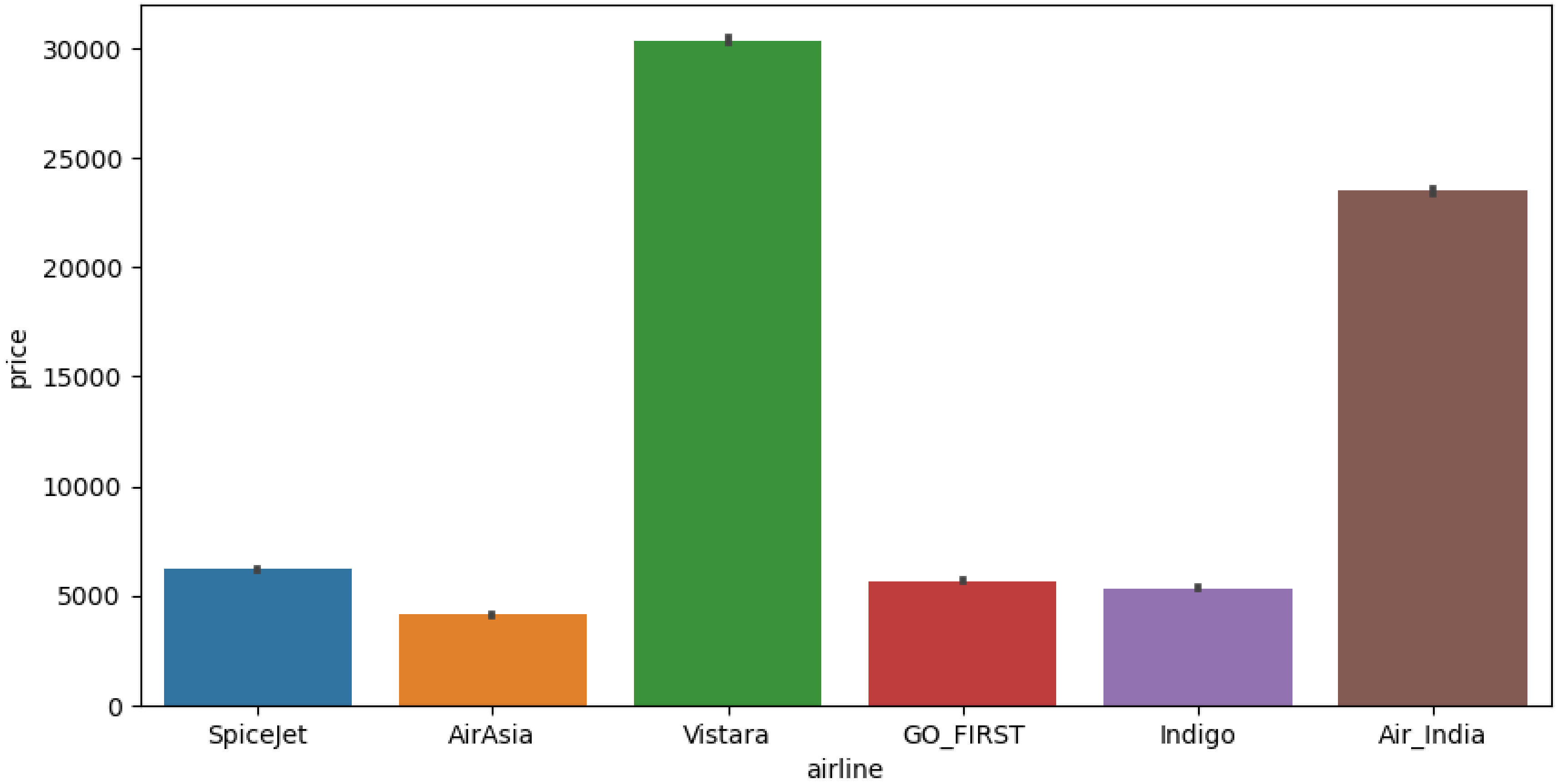


In [13]:
```
df["airline"].value_counts()
```

Out[13]:
```
Vistara      127859
Air_India     80892
Indigo        43120
GO_FIRST      23173
AirAsia       16098
SpiceJet       9011
Name: airline, dtype: int64
```

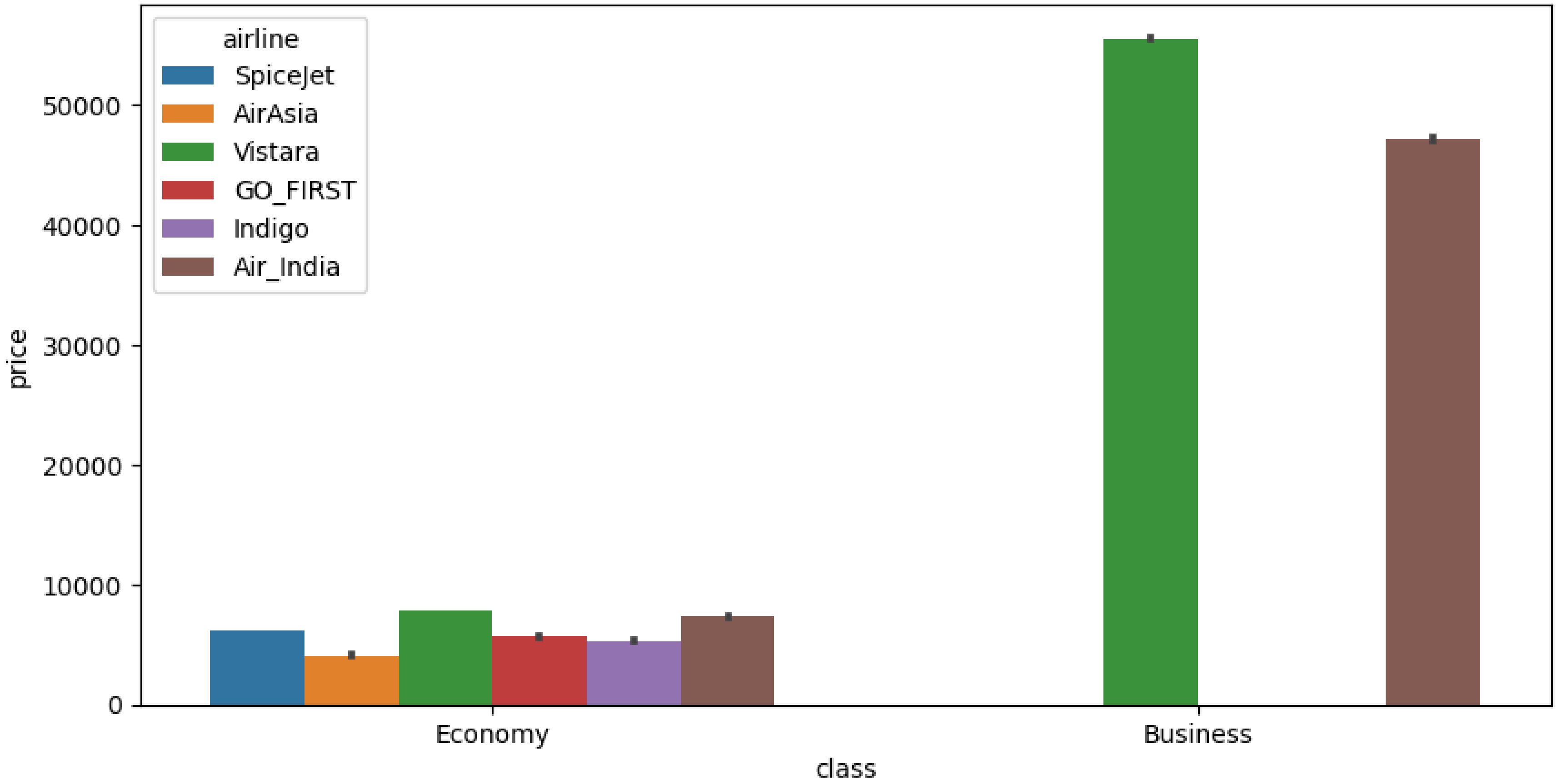
In [14]:
```
plt.figure(figsize=(10,5))
sns.barplot(x='airline', y='price', data=df)
```

Out[14]:
```
<AxesSubplot:xlabel='airline', ylabel='price'>
```

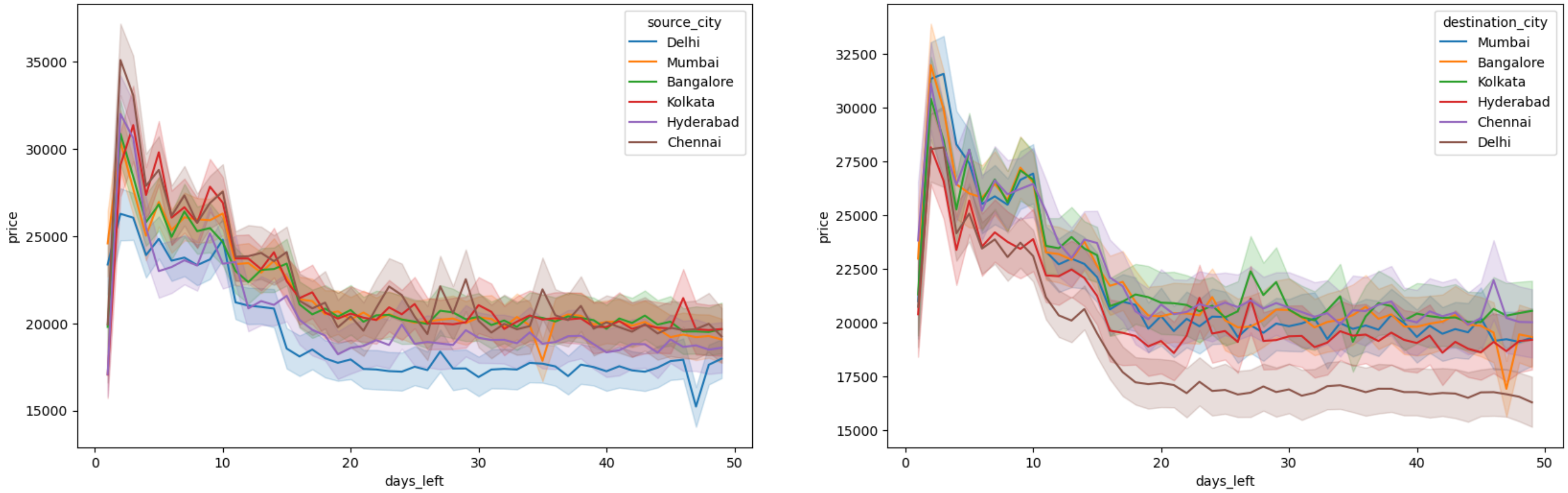


In [15]:
```
plt.figure(figsize=(10,5))
sns.barplot(x='class', y='price', hue='airline', data=df)
```

Out[15]: <AxesSubplot:xlabel='class', ylabel='price'>

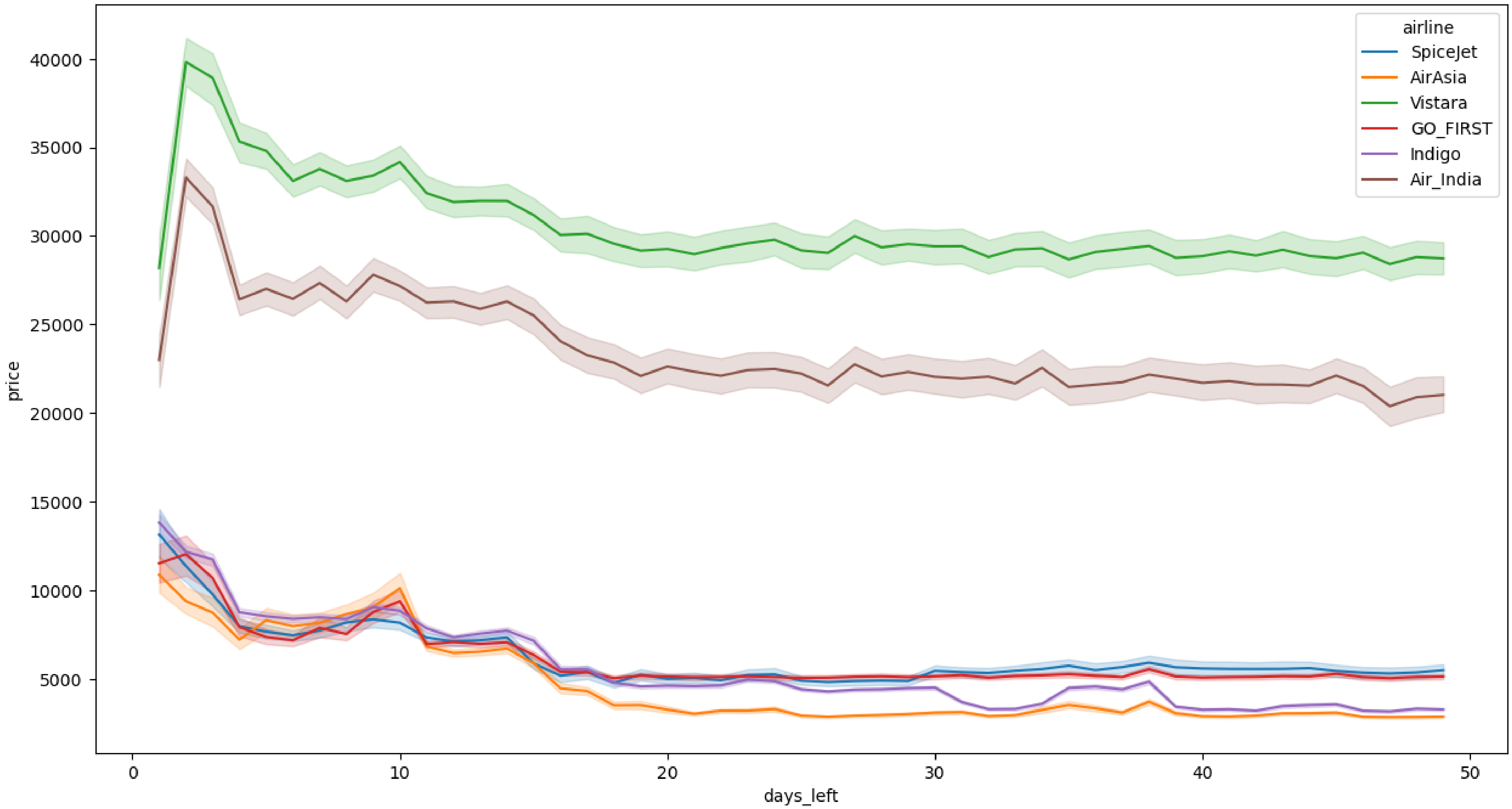


In [16]:
```
fig,ax=plt.subplots(1,2,figsize=(20,6))
sns.lineplot(data=df, x='days_left',y='price',hue='source_city', ax=ax[0])
sns.lineplot(data=df, x='days_left',y='price',hue='destination_city', ax=ax[1])
plt.show()
```



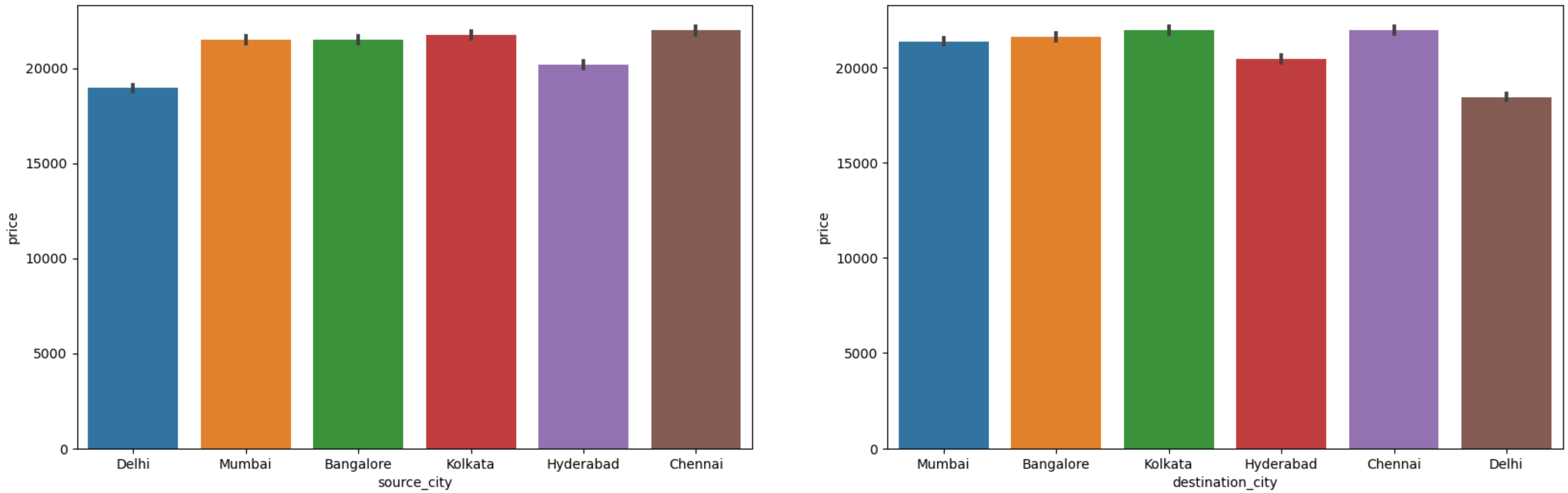
In [17]:
```
plt.figure(figsize=(15,8))
sns.lineplot(x='days_left', y='price', data=df, hue='airline')
```

Out[17]:
```
<AxesSubplot:xlabel='days_left', ylabel='price'>
```

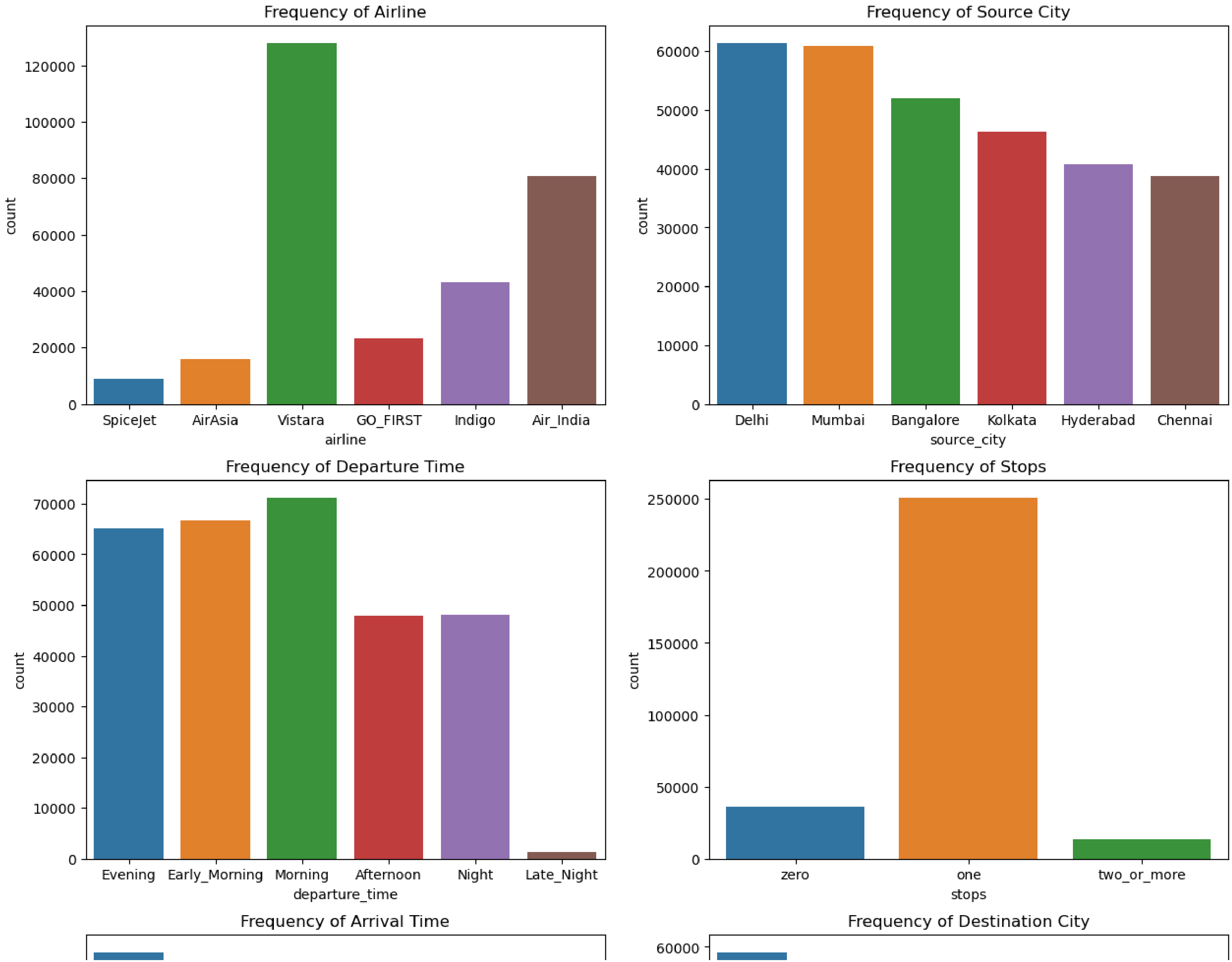


```
In [19]: fig,ax=plt.subplots(1,2,figsize=(20,6))
         sns.barplot(data=df, x='source_city',y='price', ax=ax[0])
         sns.barplot(data=df, x='destination_city',y='price', ax=ax[1])
```

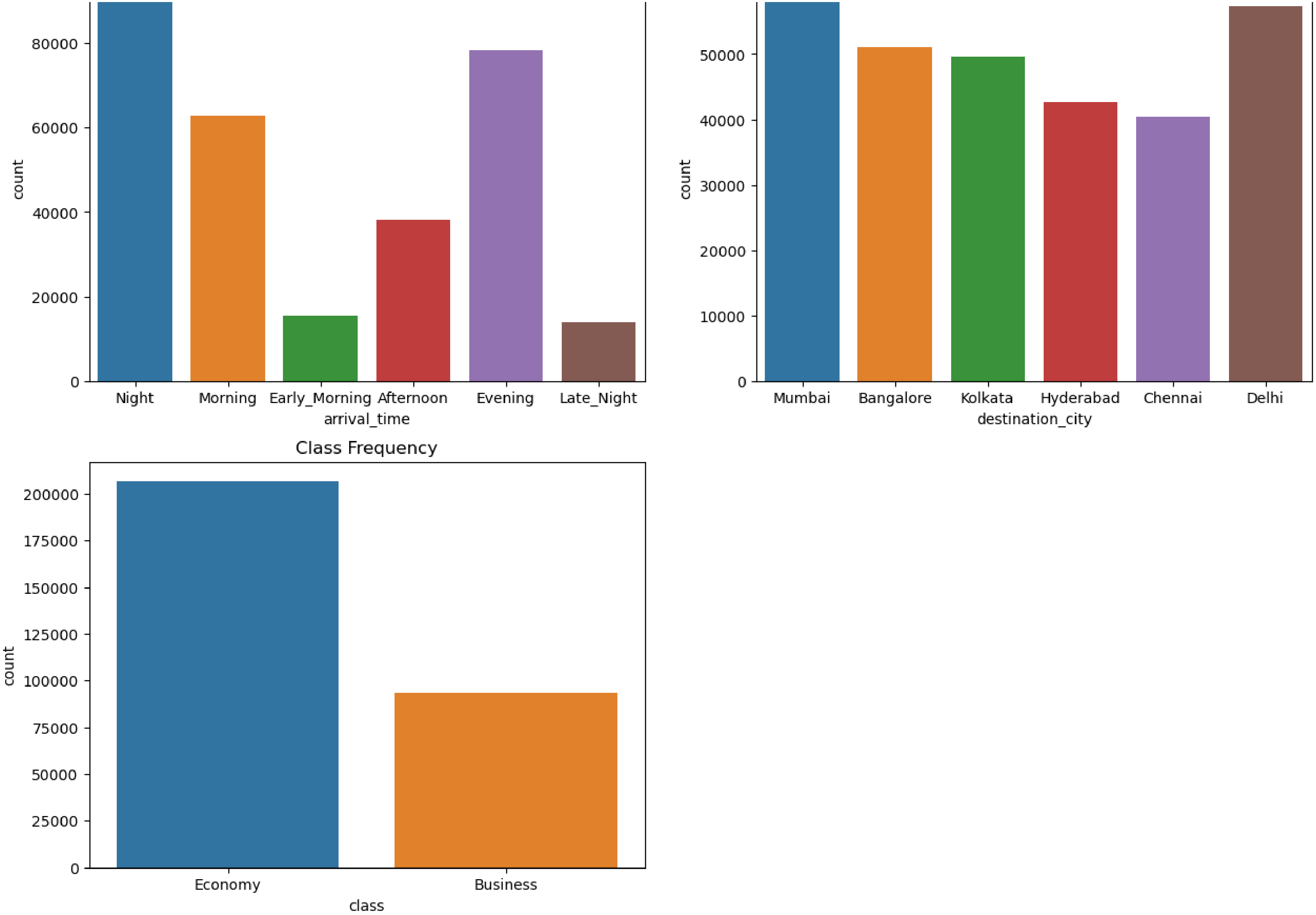
Out[19]: <AxesSubplot:xlabel='destination_city', ylabel='price'>



In [21]:
```python
# visualization of categoric features with countplot
plt.figure(figsize=(15,23))

plt.subplot(4, 2, 1)
sns.countplot(x=df["airline"], data=df)
plt.title("Frequency of Airline")

plt.subplot(4, 2, 2)
sns.countplot(x=df["source_city"], data=df)
plt.title("Frequency of Source City")

plt.subplot(4, 2, 3)
sns.countplot(x=df["departure_time"], data=df)
plt.title("Frequency of Departure Time")

plt.subplot(4, 2, 4)
sns.countplot(x=df["stops"], data=df)
plt.title("Frequency of Stops")

plt.subplot(4, 2, 5)
sns.countplot(x=df["arrival_time"], data=df)
plt.title("Frequency of Arrival Time")

plt.subplot(4, 2, 6)
sns.countplot(x=df["destination_city"], data=df)
plt.title("Frequency of Destination City")

plt.subplot(4, 2, 7)
sns.countplot(x=df["class"], data=df)
plt.title("Class Frequency")

plt.show()
```



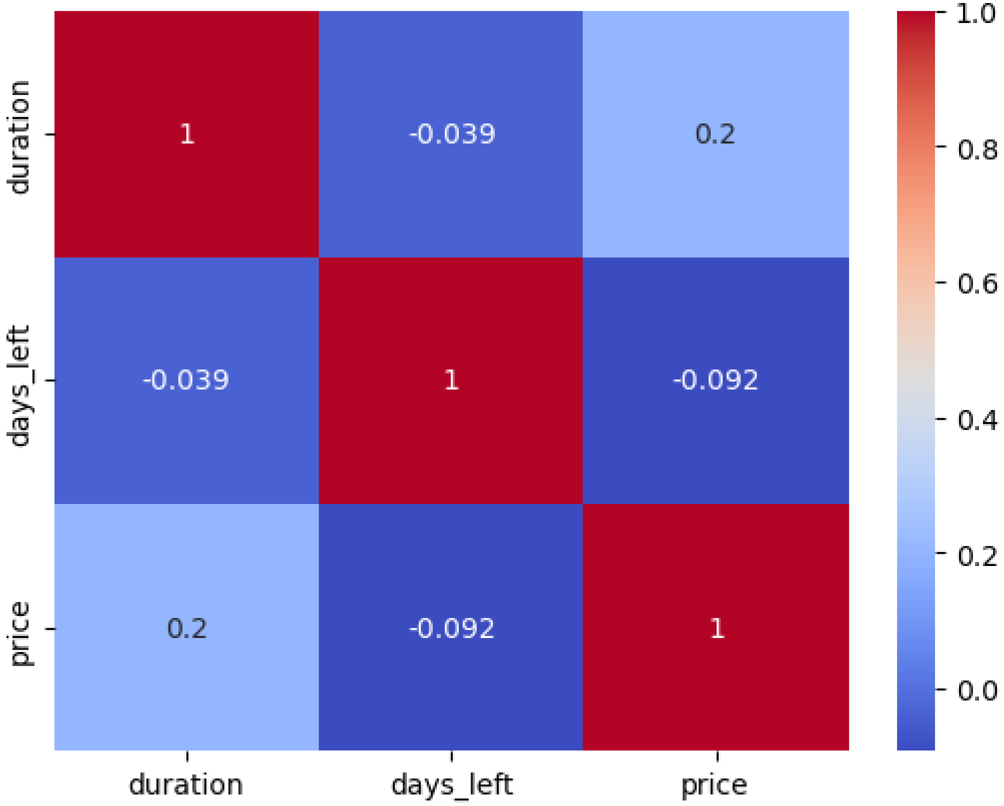
Frequency of Airline
count
120000
100000
80000
60000
40000
20000
0
SpiceJet
AirAsia
Vistara
GO_FIRST
Indigo
Air_India
airline
Frequency of Source City
count
60000
50000
40000
30000
20000
10000
0
Delhi
Mumbai
Bangalore
Kolkata
Hyderabad
Chennai
source_city
Frequency of Departure Time
count
70000
60000
50000
40000
30000
20000
10000
0
Evening
Early_Morning
Morning
Afternoon
Night
Late_Night
departure_time
Frequency of Stops
count
250000
200000
150000
100000
50000
0
zero
one
two_or_more
stops
Frequency of Arrival Time
Frequency of Destination City
60000



80000
60000
40000
20000
0
count
Night
Morning
Early_Morning
Afternoon
Evening
Late_Night
arrival_time
50000
40000
30000
20000
10000
0
count
Mumbai
Bangalore
Kolkata
Hyderabad
Chennai
Delhi
destination_city
Class Frequency
200000
175000
150000
125000
100000
75000
50000
25000
0
count
Economy
Business
class

In [22]:
```
sns.heatmap(df.corr(),annot=True,cmap="coolwarm")
```

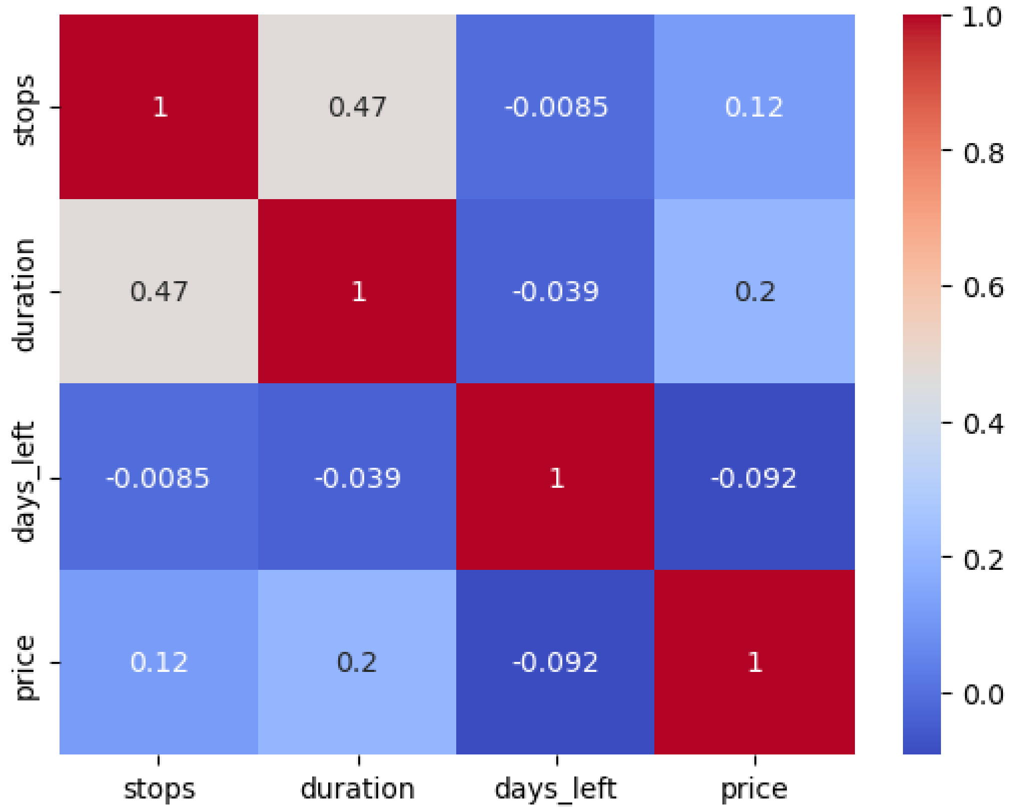
Out[22]: <AxesSubplot:>




In [24]:
```
df["stops"]=df["stops"].replace(["zero","one","two_or_more"],[0,1,2])
```

In [25]:
```
sns.heatmap(df.corr(),annot=True,cmap="coolwarm")
```

Out[25]: <AxesSubplot:>




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

Out[26]:

| | stops | duration | days_left | price |
|---|---|---|---|---|
| **stops** | 1.000000 | 0.468059 | -0.008540 | 0.119648 |
| **duration** | 0.468059 | 1.000000 | -0.039157 | 0.204222 |
| **days_left** | -0.008540 | -0.039157 | 1.000000 | -0.091949 |
| **price** | 0.119648 | 0.204222 | -0.091949 | 1.000000 |

In [27]:
```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
df["airline"]=le.fit_transform(df['airline'])
df["source_city"]=le.fit_transform(df["source_city"])
df["departure_time"]=le.fit_transform(df["departure_time"])
df["arrival_time"]=le.fit_transform(df["arrival_time"])
df["destination_city"]=le.fit_transform(df["destination_city"])
df["class"]=le.fit_transform(df["class"])
```

In [28]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 300153 entries, 0 to 300152
Data columns (total 11 columns):
 #   Column            Non-Null Count   Dtype  
---  ------            --------------   -----  
 0   airline           300153 non-null  int32  
 1   flight            300153 non-null  object 
 2   source_city       300153 non-null  int32  
 3   departure_time    300153 non-null  int32  
 4   stops             300153 non-null  int64  
 5   arrival_time      300153 non-null  int32  
 6   destination_city  300153 non-null  int32  
 7   class             300153 non-null  int32  
 8   duration          300153 non-null  float64
 9   days_left         300153 non-null  int64  
 10  price             300153 non-null  int64  
dtypes: float64(1), int32(6), int64(3), object(1)
memory usage: 18.3+ MB
```

In [29]:
```
df=df.drop(columns=("flight"))
```

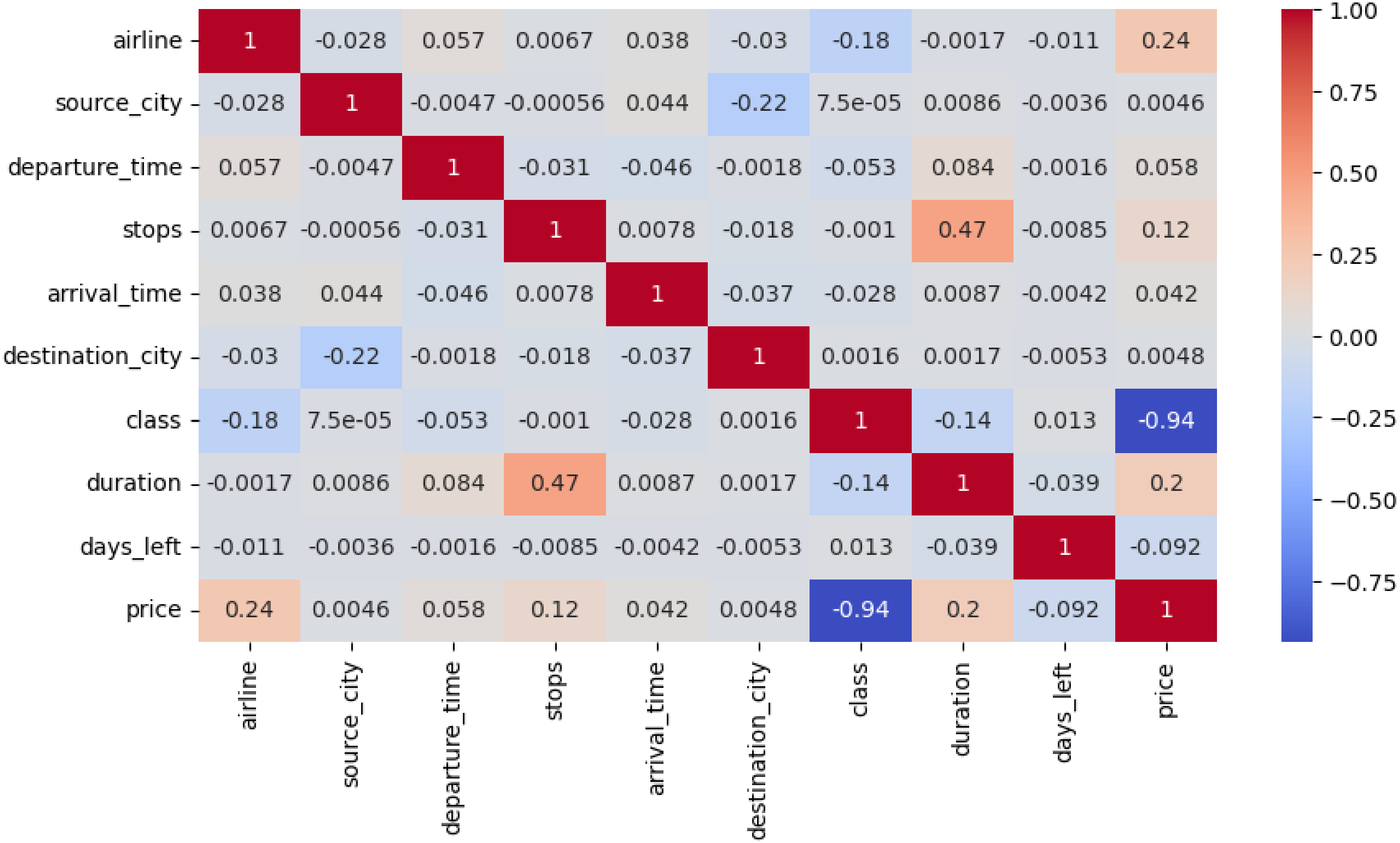
In [30]:
```
df
```

Out[30]:

| | airline | source_city | departure_time | stops | arrival_time | destination_city | class | duration | days_left | price |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 4 | 2 | 2 | 0 | 5 | 5 | 1 | 2.17 | 1 | 5953 |
| **1** | 4 | 2 | 1 | 0 | 4 | 5 | 1 | 2.33 | 1 | 5953 |
| **2** | 0 | 2 | 1 | 0 | 1 | 5 | 1 | 2.17 | 1 | 5956 |
| **3** | 5 | 2 | 4 | 0 | 0 | 5 | 1 | 2.25 | 1 | 5955 |
| **4** | 5 | 2 | 4 | 0 | 4 | 5 | 1 | 2.33 | 1 | 5955 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **300148** | 5 | 1 | 4 | 1 | 2 | 3 | 0 | 10.08 | 49 | 69265 |
| **300149** | 5 | 1 | 0 | 1 | 5 | 3 | 0 | 10.42 | 49 | 77105 |
| **300150** | 5 | 1 | 1 | 1 | 5 | 3 | 0 | 13.83 | 49 | 79099 |
| **300151** | 5 | 1 | 1 | 1 | 2 | 3 | 0 | 10.00 | 49 | 81585 |
| **300152** | 5 | 1 | 4 | 1 | 2 | 3 | 0 | 10.08 | 49 | 81585 |

300153 rows × 10 columns

In [31]:
```
plt.figure(figsize=(10,5))
sns.heatmap(df.corr(),annot=True,cmap="coolwarm")
plt.show()
```




| | airline | source_city | departure_time | stops | arrival_time | destination_city | class | duration | days_left | price |
|---|---|---|---|---|---|---|---|---|---|---|
| airline | 1 | -0.028 | 0.057 | 0.0067 | 0.038 | -0.03 | -0.18 | -0.0017 | -0.011 | 0.24 |
| source_city | -0.028 | 1 | -0.0047 | -0.00056 | 0.044 | -0.22 | 7.5e-05 | 0.0086 | -0.0036 | 0.0046 |
| departure_time | 0.057 | -0.0047 | 1 | -0.031 | -0.046 | -0.0018 | -0.053 | 0.084 | -0.0016 | 0.058 |
| stops | 0.0067 | -0.00056 | -0.031 | 1 | 0.0078 | -0.018 | -0.001 | 0.47 | -0.0085 | 0.12 |
| arrival_time | 0.038 | 0.044 | -0.046 | 0.0078 | 1 | -0.037 | -0.028 | 0.0087 | -0.0042 | 0.042 |
| destination_city | -0.03 | -0.22 | -0.0018 | -0.018 | -0.037 | 1 | 0.0016 | 0.0017 | -0.0053 | 0.0048 |
| class | -0.18 | 7.5e-05 | -0.053 | -0.001 | -0.028 | 0.0016 | 1 | -0.14 | 0.013 | -0.94 |
| duration | -0.0017 | 0.0086 | 0.084 | 0.47 | 0.0087 | 0.0017 | -0.14 | 1 | -0.039 | 0.2 |
| days_left | -0.011 | -0.0036 | -0.0016 | -0.0085 | -0.0042 | -0.0053 | 0.013 | -0.039 | 1 | -0.092 |
| price | 0.24 | 0.0046 | 0.058 | 0.12 | 0.042 | 0.0048 | -0.94 | 0.2 | -0.092 | 1 |

In [32]:
```
from statsmodels.stats.outliers_influence import variance_inflation_factor
col_list = []
for col in df.columns:
    if ((df[col].dtype != 'object') & (col != 'price') ):
        col_list.append(col)

X = df[col_list]
vif_data = pd.DataFrame()
vif_data["feature"] = X.columns
vif_data["VIF"] = [variance_inflation_factor(X.values, i)
                          for i in range(len(X.columns))]
print(vif_data)
```

```
            feature       VIF
0           airline  3.461766
1       source_city  2.933064
2    departure_time  2.746367
3             stops  7.464236
4      arrival_time  3.684695
5  destination_city  2.893218
6             class  2.917521
7          duration  5.037943
8         days_left  4.035735
```

In [33]:
```
df=df.drop(columns=["stops"])
```

In [34]:
```
from statsmodels.stats.outliers_influence import variance_inflation_factor
col_list = []
for col in df.columns:
    if ((df[col].dtype != 'object') & (col != 'price') ):
        col_list.append(col)

X = df[col_list]
vif_data = pd.DataFrame()
vif_data["feature"] = X.columns
vif_data["VIF"] = [variance_inflation_factor(X.values, i)
                          for i in range(len(X.columns))]
print(vif_data)
```

```
            feature       VIF
0           airline  3.370020
1       source_city  2.895803
2    departure_time  2.746255
3      arrival_time  3.632792
4  destination_city  2.857808
5             class  2.776721
6          duration  3.429344
7         days_left  3.950132
```

In [35]:
```
X = df.drop(columns=["price"])
y = df['price']
```

In [36]:
```
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)
```

LINEAR REGRESSION

In [37]:
```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
```

In [39]:
```
from sklearn.preprocessing import StandardScaler
```

In [40]:
```
sc=StandardScaler()
x_train=sc.fit_transform(x_train)
x_test=sc.transform(x_test)
```

In [41]:
```
lr.fit(x_train,y_train)
```

Out[41]:
```
LinearRegression()
```

In [42]:
```
y_pred=lr.predict(x_test)
```

In [43]:
```
from sklearn.metrics import r2_score
r2result= r2_score(y_test,y_pred)
r2_score(y_test,y_pred)
```

Out[43]:
```
0.897752737512321
```

In [44]:
```
from sklearn import metrics
mean_abs_error= metrics.mean_absolute_error(y_test,y_pred)
mean_abs_error
```

Out[44]:
```
4468.426673542101
```

In [45]:
```
from sklearn.metrics import mean_absolute_percentage_error
mean_absolute_percentage_error(y_test, y_pred)
```

Out[45]:
```
0.3476580461068153
```

In [66]:
```
mean_sq_error=metrics.mean_squared_error(y_test,y_pred)
mean_sq_error
```

Out[66]:
```
7996852.607987438
```

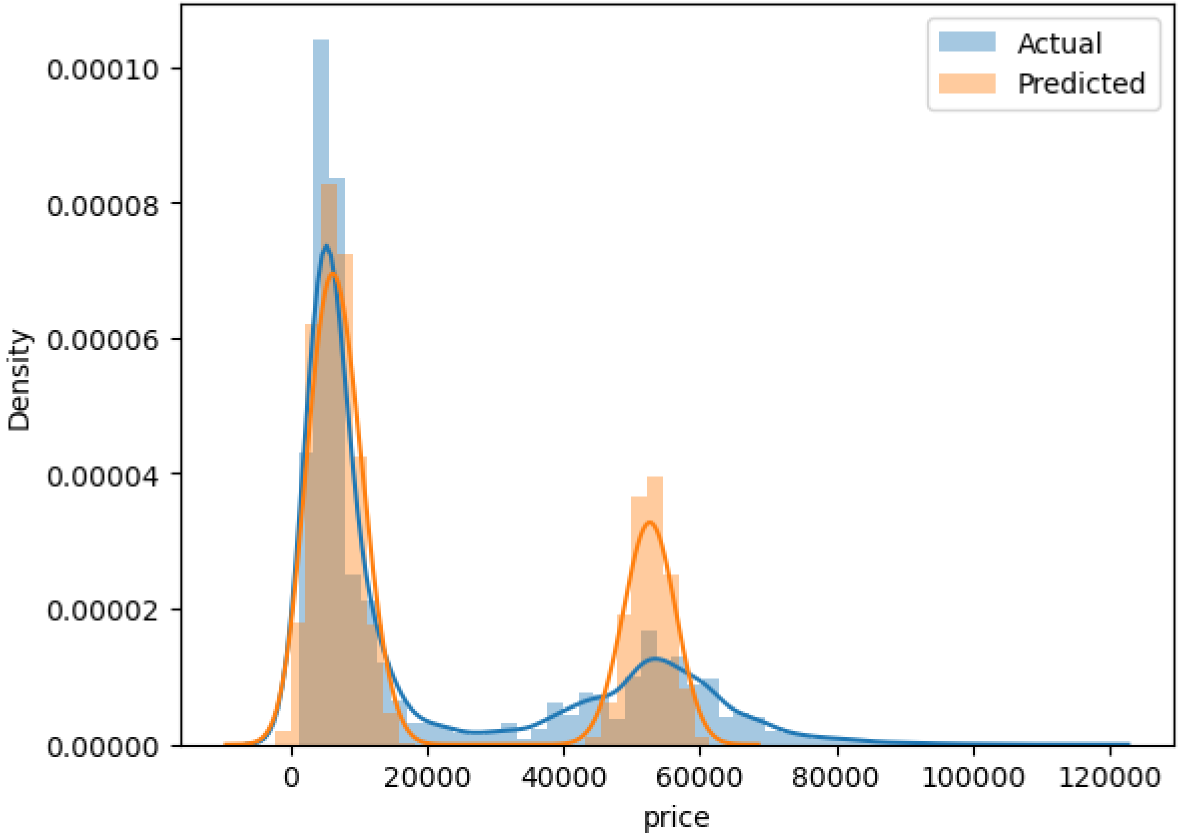
In [46]:
```
root_mean_sq_error = np.sqrt(metrics.mean_squared_error(y_test,y_pred))
root_mean_sq_error
```

Out[46]:
```
7259.934664536732
```

In [47]:
```
sns.distplot(y_test,label="Actual")
sns.distplot(y_pred,label="Predicted")
plt.legend()
```

```
C:\Users\Prince\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
C:\Users\Prince\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[47]: <matplotlib.legend.Legend at 0x189e36ed8e0>



DESCISION TREE REGRESSION MODEL

In [48]:
```
lr.predict([[5,5,5,4,4,0,11.67,28]])
```

Out[48]:
```
array([4373.12535091])
```

In [49]:
```
import numpy as np

# Given array
array = np.array([4373.12535091])

# Extract the element
normal_value = array[0]

print(normal_value)
```

```
4373.12535091
```

In [50]:
```
from sklearn.tree import DecisionTreeRegressor
dt=DecisionTreeRegressor()
dt.fit(x_train,y_train)
y_pred=dt.predict(x_test)
r2_score(y_test,y_pred)
```

Out[50]:
```
0.974595341374098
```

In [51]:
```
mean_abs_error= metrics.mean_absolute_error(y_test,y_pred)
mean_abs_error
```

Out[51]:
```
1220.179515583615
```

In [52]:
```
from sklearn.metrics import mean_absolute_percentage_error
mean_absolute_percentage_error(y_test, y_pred)
```

Out[52]:
```
0.07752387054568081
```

In [53]:
```
mean_sq_error=metrics.mean_squared_error(y_test,y_pred)
mean_sq_error
```

Out[53]:
```
13095651.187721064
```

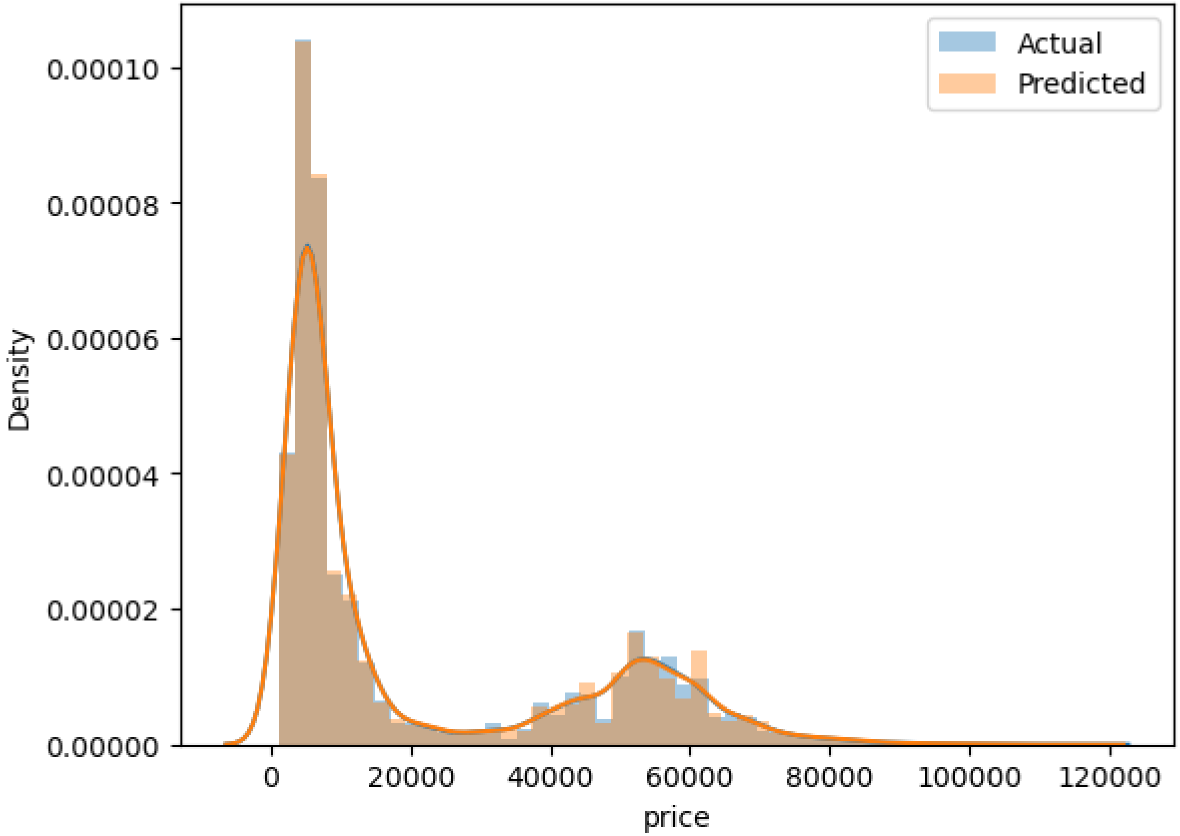
In [54]:
```
root_mean_sq_error = np.sqrt(metrics.mean_squared_error(y_test,y_pred))
root_mean_sq_error
```

Out[54]:
```
3618.7913987574725
```

In [55]:
```
sns.distplot(y_test,label="Actual")
sns.distplot(y_pred,label="Predicted")
plt.legend()
```

```
C:\Users\Prince\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function
and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar
flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
C:\Users\Prince\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function
and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar
flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[55]: <matplotlib.legend.Legend at 0x189e40976a0>



RANDOM FOREST REGRESSION

In [56]:
```
from sklearn.ensemble import RandomForestRegressor
rfr=RandomForestRegressor()
rfr.fit(x_train,y_train)
y_pred=rfr.predict(x_test)
r2_score(y_test,y_pred)
```

Out[56]: 0.984486658381825

In [57]:
```
from sklearn import metrics
mean_abs_error= metrics.mean_absolute_error(y_test,y_pred)
mean_abs_error
```

Out[57]: 1124.924115315751

In [58]:
```
from sklearn.metrics import mean_absolute_percentage_error
mean_absolute_percentage_error(y_test, y_pred)
```

Out[58]: 0.07345596497152751

In [59]:
```
mean_sq_error=metrics.mean_squared_error(y_test,y_pred)
mean_sq_error
```

Out[59]: 7996852.607987438

In [60]:
```
import numpy as np
root_mean_sq_error = np.sqrt(metrics.mean_squared_error(y_test,y_pred))
root_mean_sq_error
```

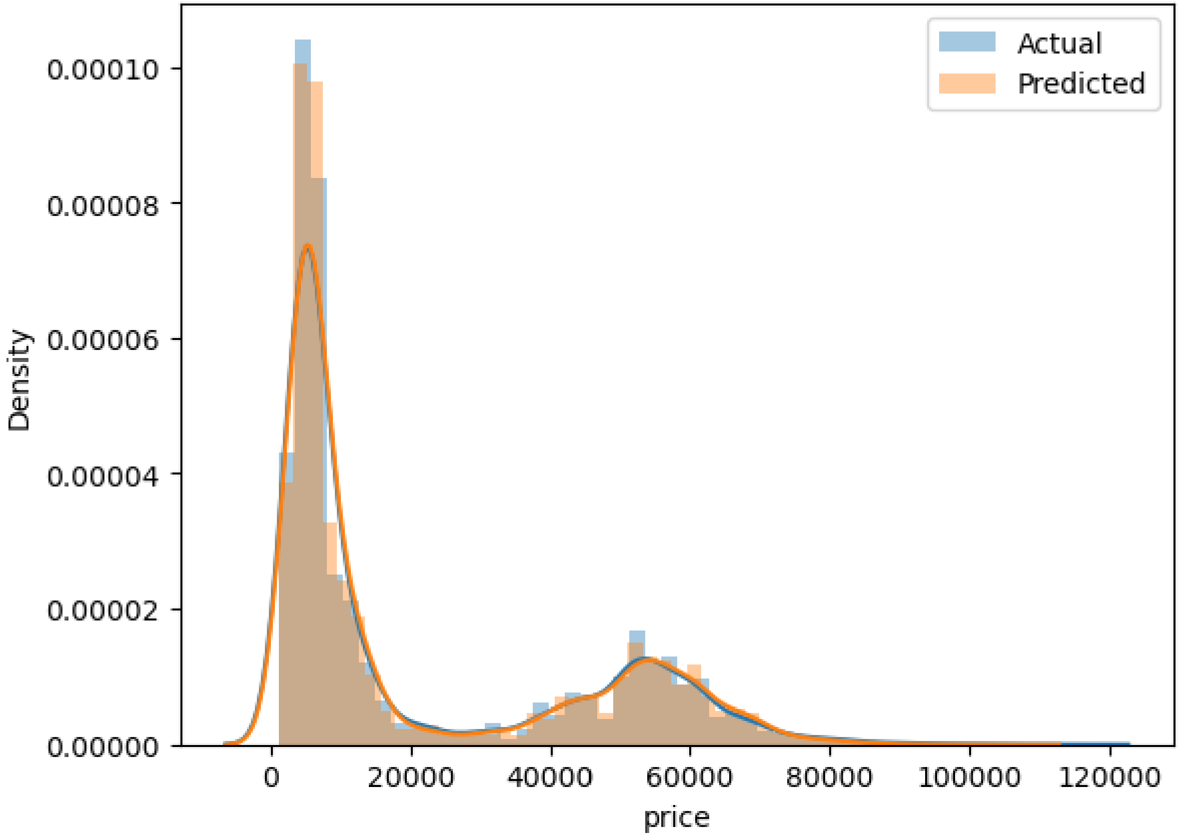
Out[60]: 2827.8706844527806

In [61]:

```
sns.distplot(y_test,label="Actual")
sns.distplot(y_pred,label="Predicted")
plt.legend()
```

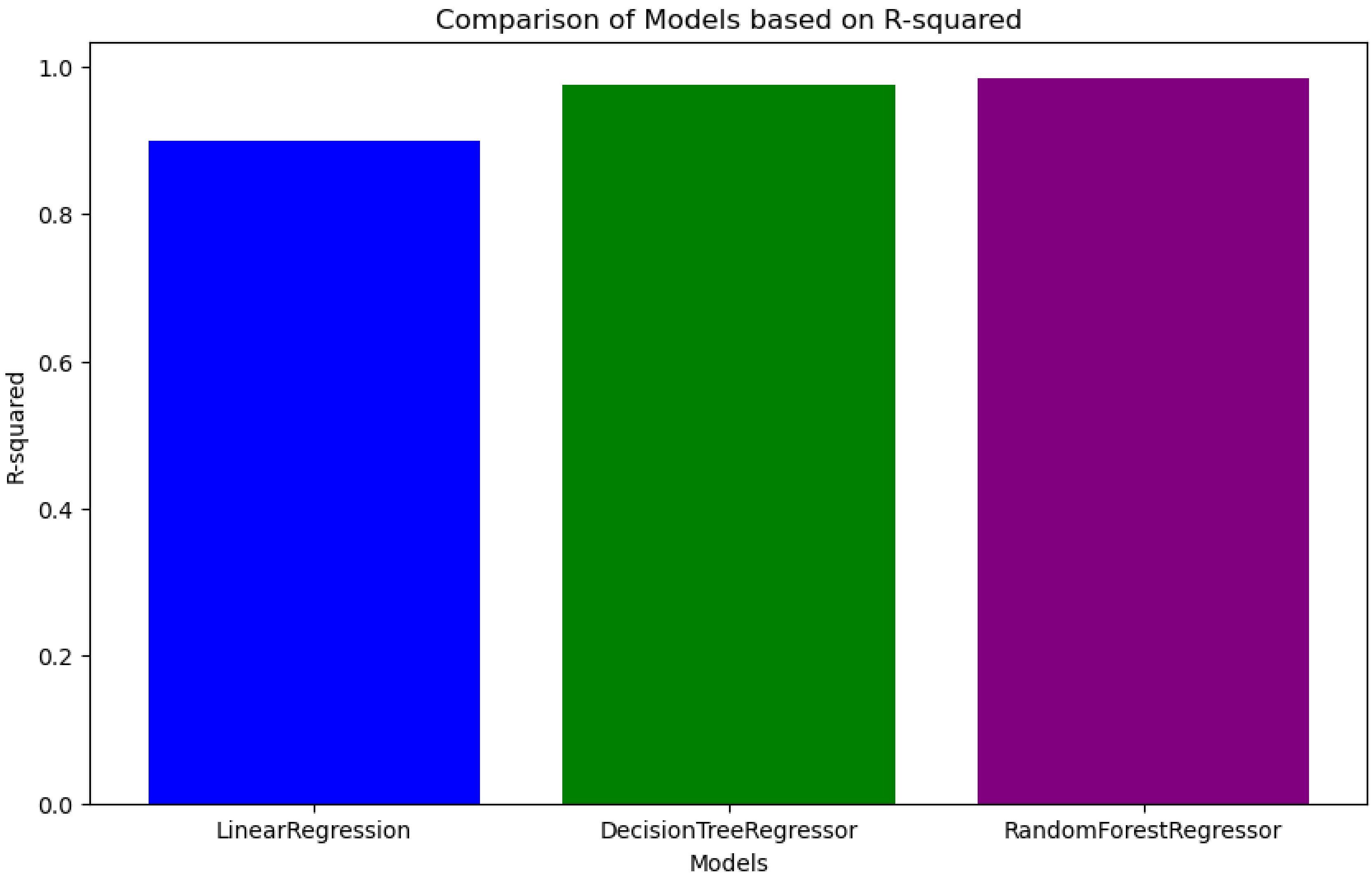
```
C:\Users\Prince\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
C:\Users\Prince\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[61]: <matplotlib.legend.Legend at 0x189e2e62610>



In [62]:

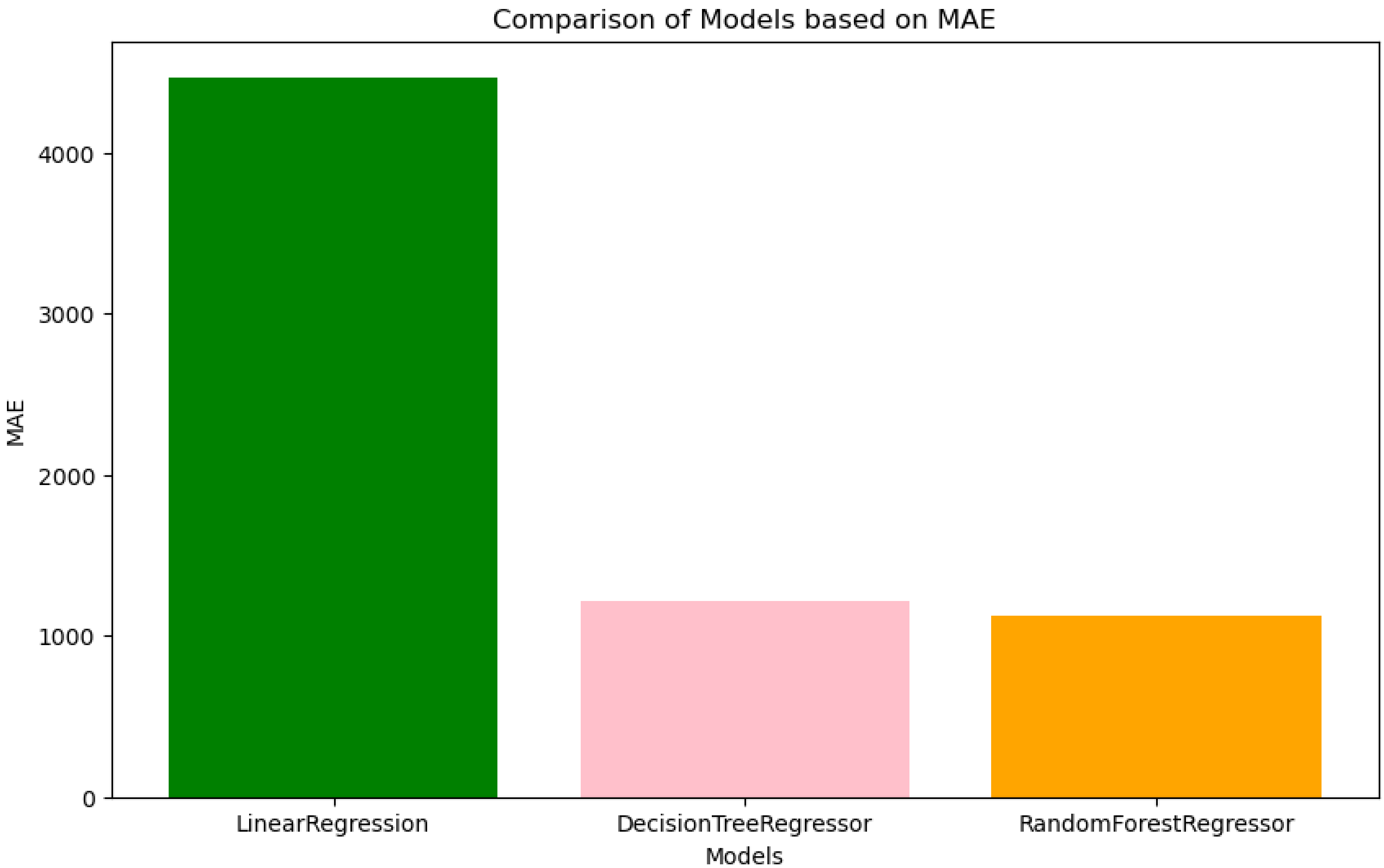
```
# R-squared values for different models
models = ['LinearRegression', 'DecisionTreeRegressor', 'RandomForestRegressor' ]
r_squared = [0.897752737512321 , 0.974595341374098 ,0.984486658381825 ]

# Set the figure size
plt.figure(figsize=(10, 6))
colors = ['blue', 'green', 'purple']

# Create a bar plot
plt.bar(models, r_squared, color=colors)

# Add labels and title
plt.xlabel('Models')
plt.ylabel('R-squared')
plt.title('Comparison of Models based on R-squared')

# Show the plot
plt.show()
```




```
COMPARISION OF DIFFERENT MODEL AS PER MEAN ABSOLUTE ERROR
```

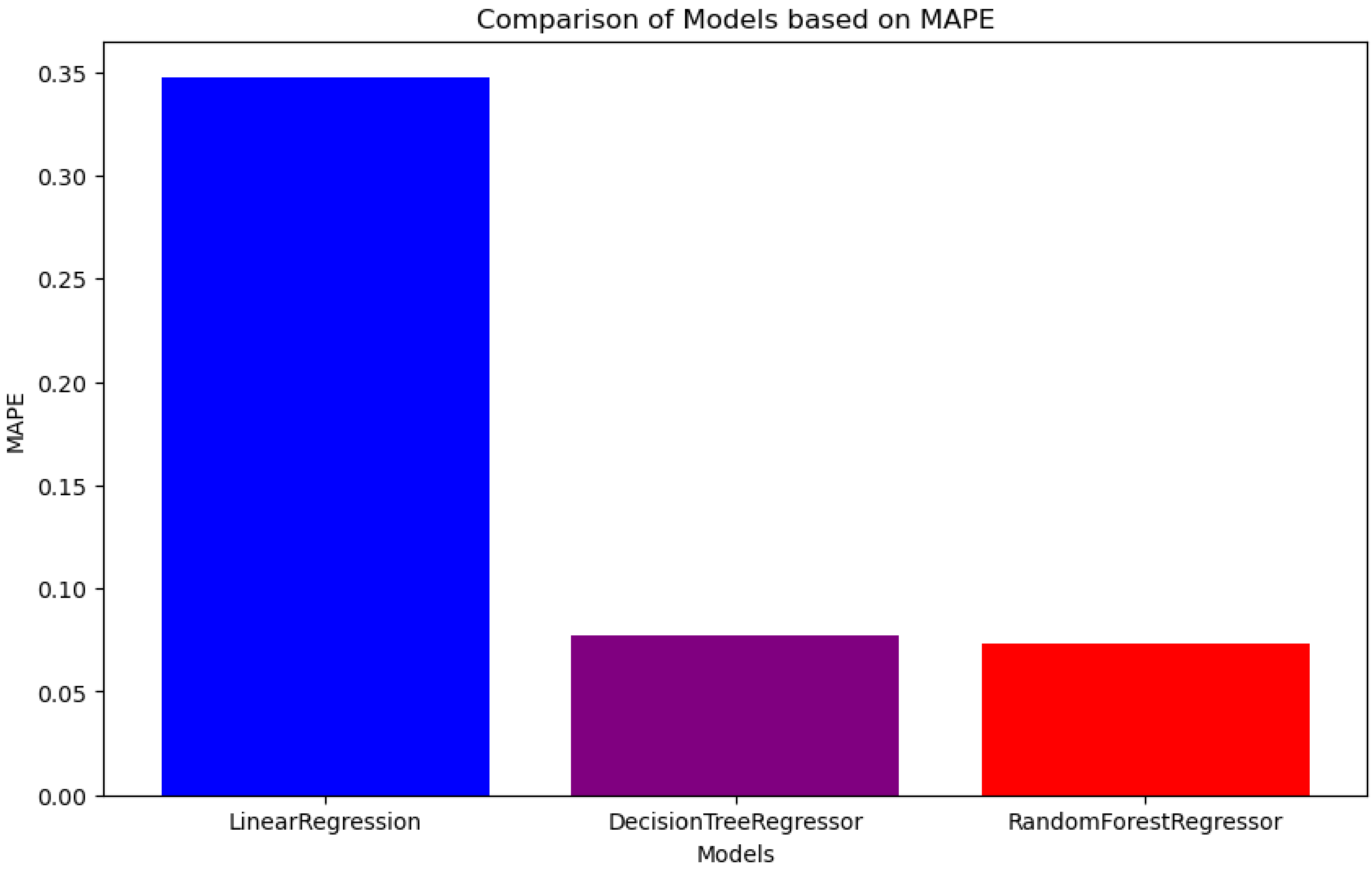
In [63]:

```
# MAE values for different models
models = ['LinearRegression', 'DecisionTreeRegressor', 'RandomForestRegressor' ]
MAE = [4468.426673542101, 1220.179515583615,1124.924115315751]

# Set the figure size
plt.figure(figsize=(10, 6))
colors = ['green', 'pink', 'orange']

# Create a bar plot
plt.bar(models, MAE, color=colors)

# Add labels and title
plt.xlabel('Models')
plt.ylabel('MAE')
plt.title('Comparison of Models based on MAE')

# Show the plot
plt.show()
```




COMPARISION OF DIFFERENT MODEL AS PER MEAN PERCENTAGE ABSOLUTE ERROR

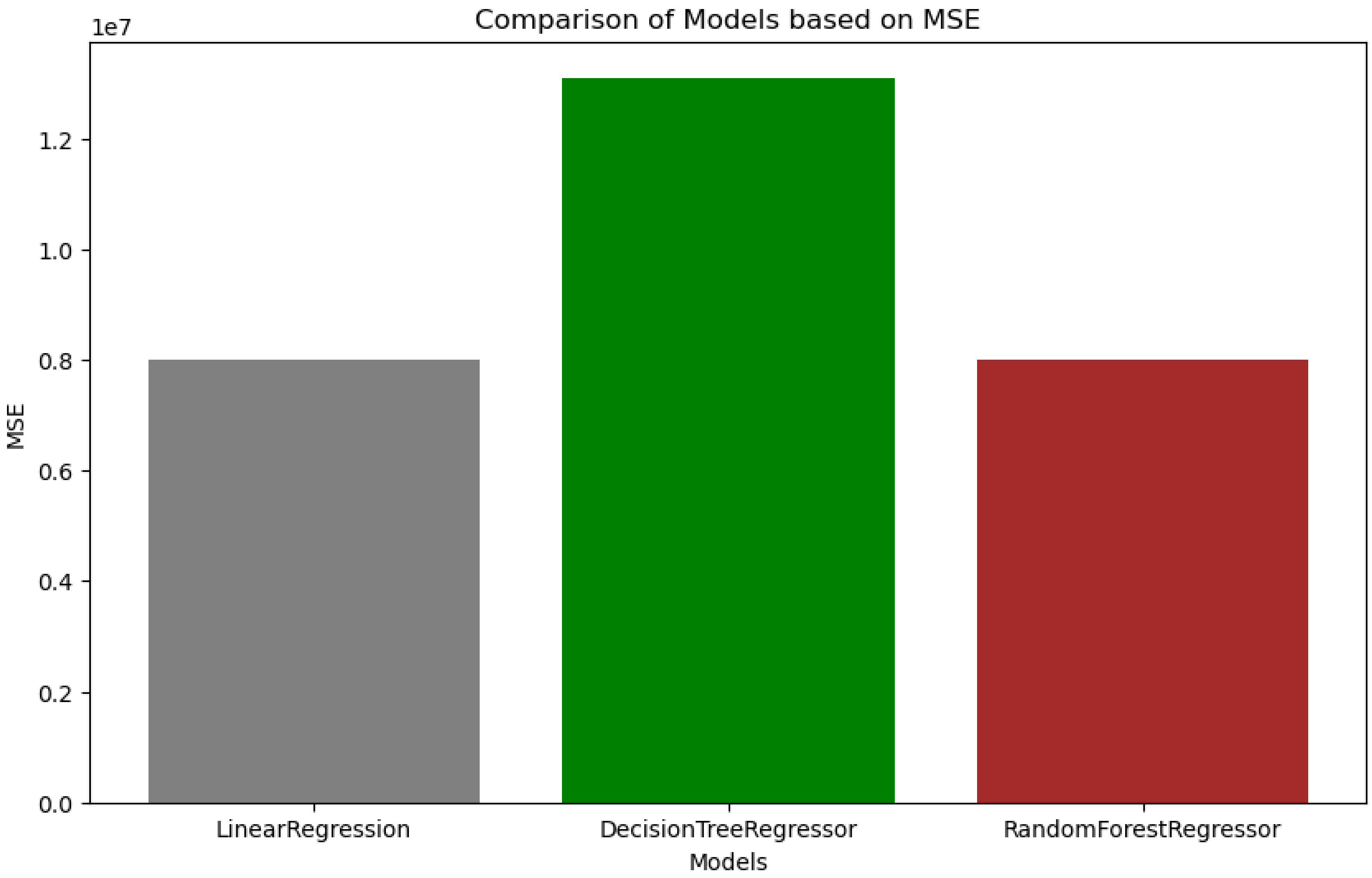
In [64]:
```
# MAPE values for different models

models = ['LinearRegression', 'DecisionTreeRegressor', 'RandomForestRegressor' ]
MAPE = [0.3476580461068153, 0.07752387054568081,0.07345596497152751]

# Set the figure size
plt.figure(figsize=(10, 6))
colors = ['blue', 'purple', 'red']

# Create a bar plot
plt.bar(models, MAPE, color=colors)

# Add labels and title
plt.xlabel('Models')
plt.ylabel('MAPE')
plt.title('Comparison of Models based on MAPE')

# Show the plot
plt.show()
```




```
COMPARISION OF DIFFERENT MODEL AS PER MEAN SQUARE ERROR
```

In [67]:

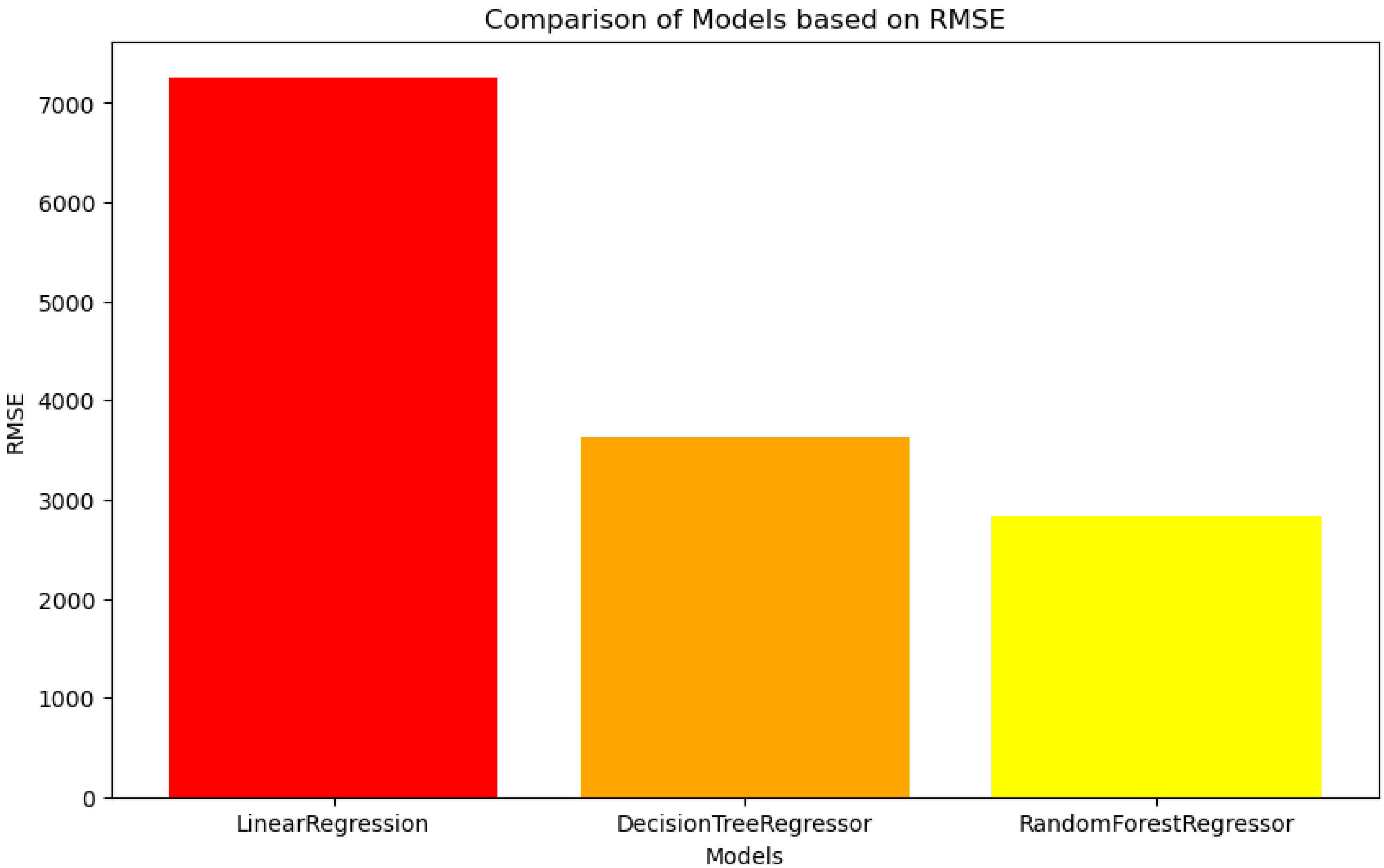
```
# MSE values for different models
models = ['LinearRegression', 'DecisionTreeRegressor', 'RandomForestRegressor' ]
MSE = [7996852.607987438,13095651.187721064,7996852.607987438]
# Set the figure size
plt.figure(figsize=(10, 6))
colors = ['grey', 'green', 'brown']
# Create a bar plot
plt.bar(models, MSE, color=colors)

# Add labels and title
plt.xlabel('Models')
plt.ylabel('MSE')
plt.title('Comparison of Models based on MSE')

# Show the plot
plt.show()
```




```
COMPARISION OF DIFFERENT MODEL AS PER ROOT MEAN SQUARE ERROR
```

In [68]:

```
# RMSE values for different models
models = ['LinearRegression', 'DecisionTreeRegressor', 'RandomForestRegressor' ]
RMSE = [7259.934664536732,3618.7913987574725,2827.8706844527806]
# Set the figure size
plt.figure(figsize=(10, 6))
colors = ['red', 'orange', 'yellow']
# Create a bar plot
plt.bar(models, RMSE, color=colors)

# Add labels and title
plt.xlabel('Models')
plt.ylabel('RMSE')
plt.title('Comparison of Models based on RMSE')

# Show the plot
plt.show()
```


Comparison of Models based on RMSE
RMSE
7000
6000
5000
4000
3000
2000
1000
0
LinearRegression
DecisionTreeRegressor
RandomForestRegressor
Models

In [ ]: