Importing Libraries

In [1]:
```
#We start off this project by importing all the necessary libraries that will be required for the process.
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
```

In [2]:
```
#Loading the data and removing unnecessary column from the dataframe
df = pd.read_csv('Flight_Booking.csv')
```

In [7]:
```
#df = df.drop(columns=["Unnamed: 0"])  (already done if you do it again get error.So, kept in comment)
```

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

Out[8]:

	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

In [10]:
```
#Checking the shape of a dataframe and datatypes of  all columns along with calculating the statistical data.
df.shape
```

Out[10]:
```
(300153, 11)
```

In [11]:
```
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 [12]:
```
df.describe()
```

Out[12]:

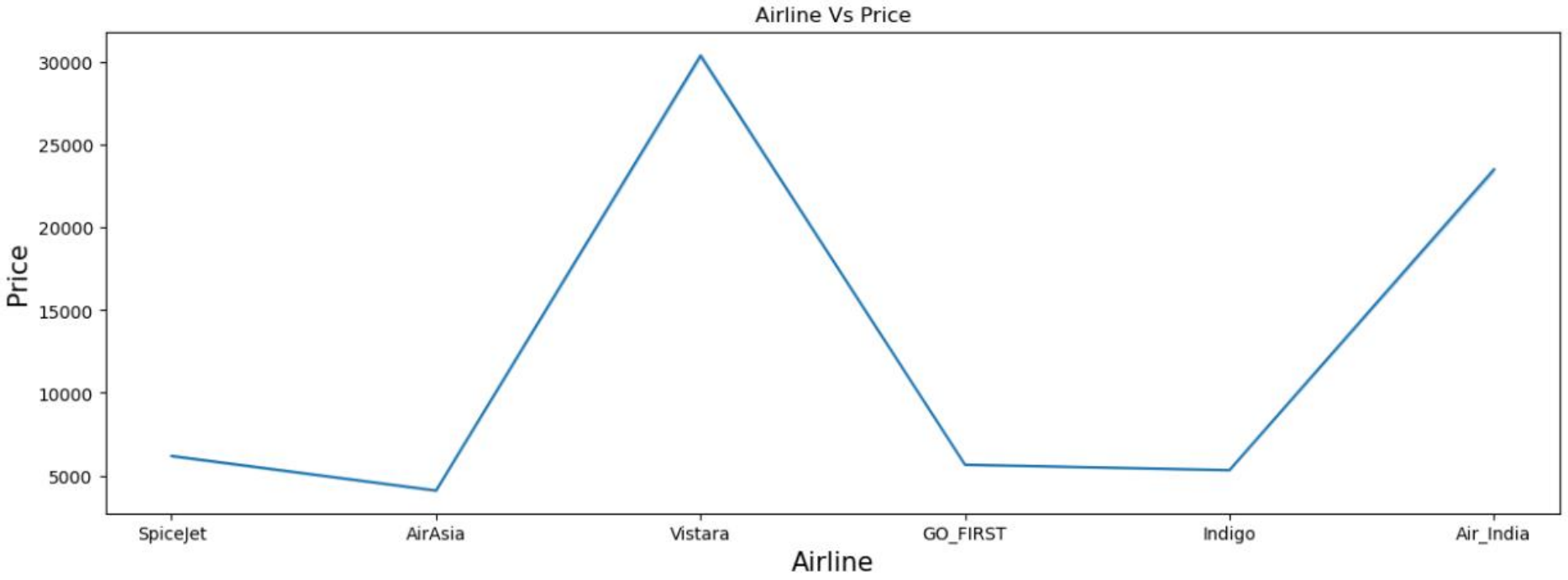
	duration	days_left	price
count	300153.000000	300153.000000	300153.000000
mean	12.221021	26.004751	20889.660523
std	7.191997	13.561004	22697.767366
min	0.830000	1.000000	1105.000000
25%	6.830000	15.000000	4783.000000
50%	11.250000	26.000000	7425.000000
75%	16.170000	38.000000	42521.000000
max	49.830000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

In [13]:
```
#Checking out the missing values in a dataframe.
df.isnull().sum()
```

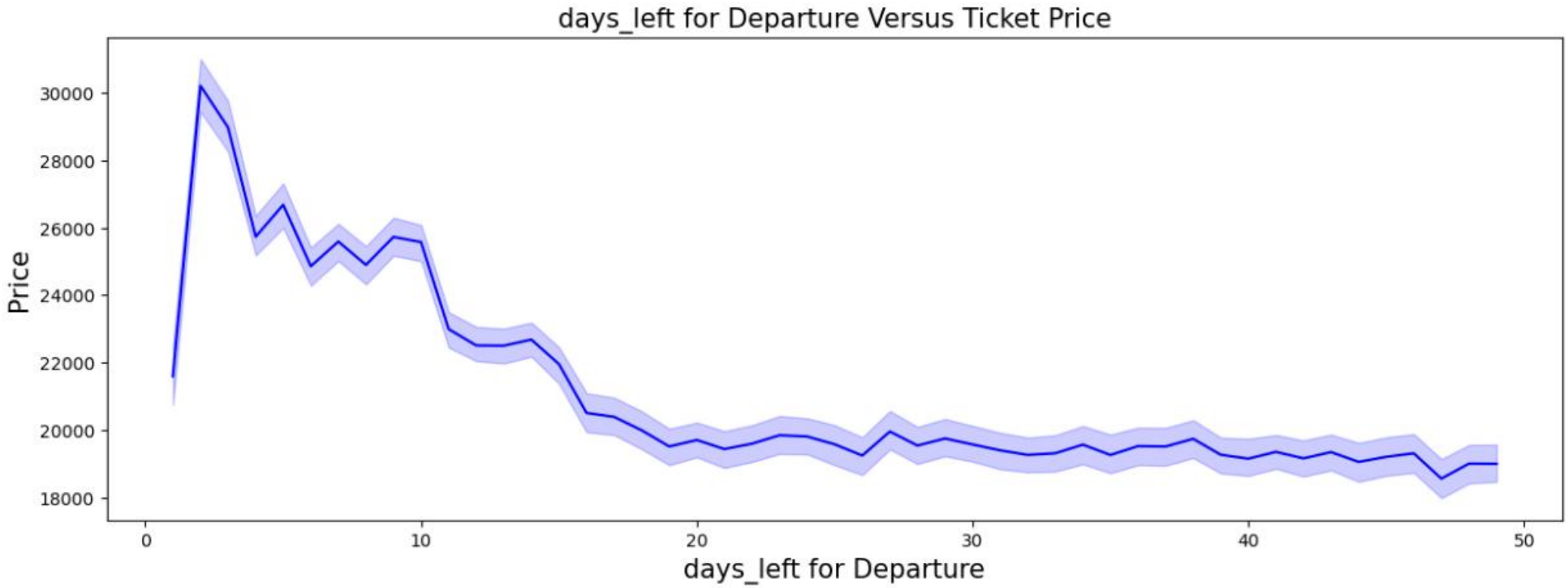
Out[13]:
```
airline             0
flight              0
source_city         0
departure_time      0
stops               0
arrival_time        0
destination_city    0
class               0
duration            0
days_left           0
price               0
dtype: int64
```

Data visualization.

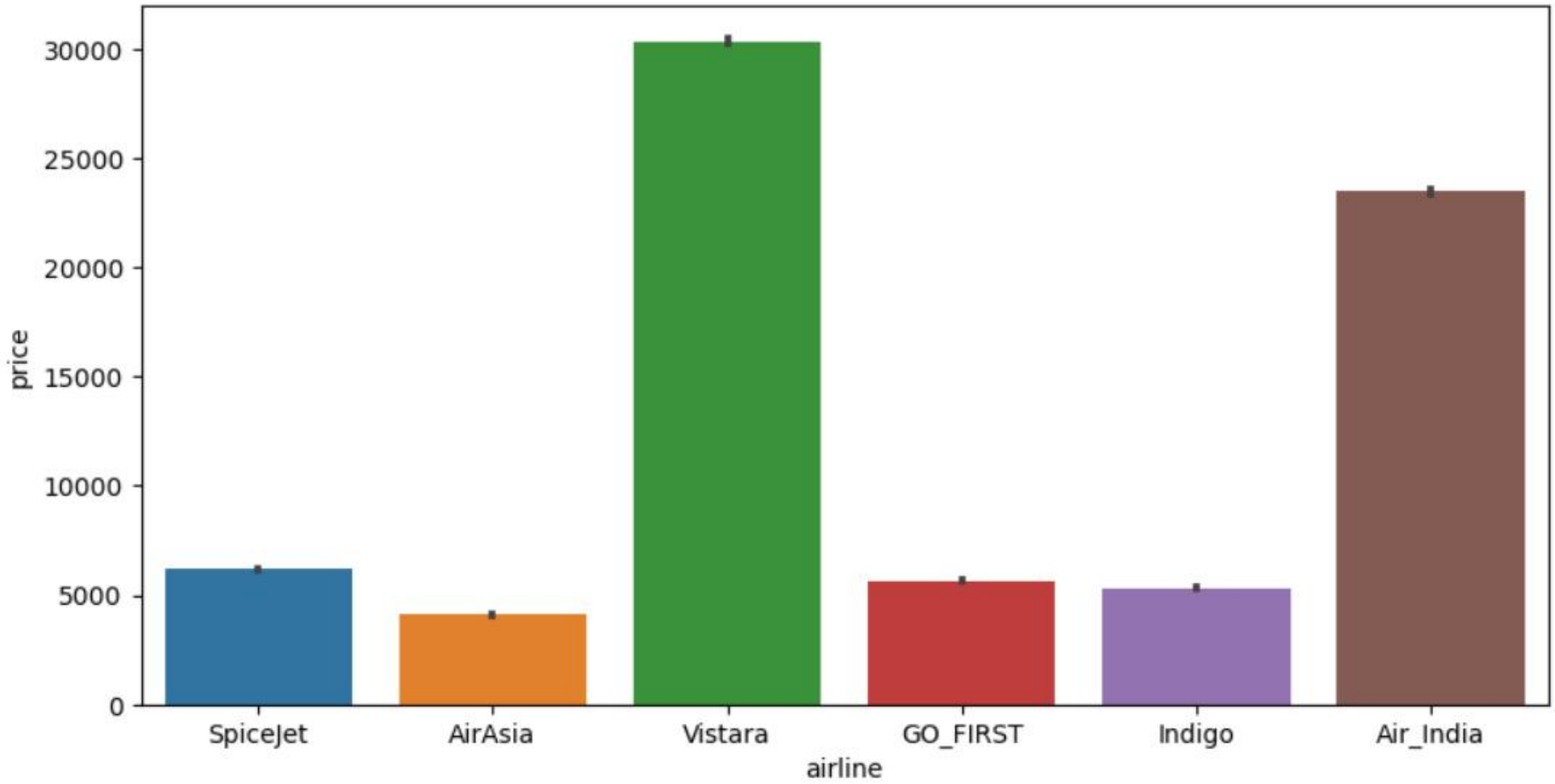
In [15]:
```
#Data visualization.
#There is a variation in price with different airlines.
plt.figure(figsize=(15,5))
sns.lineplot(x=df['airline'],y=df['price'])
plt.title('Airline Vs Price')
plt.xlabel('Airline', fontsize=15)
plt.ylabel('Price', fontsize=15)
plt.show()
```



In [16]:
```
#The price of the ticket increases as the days left for departure decreases
plt.figure(figsize=(15,5))
sns.lineplot(data=df,x='days_left',y='price',color='blue')
plt.title('days_left for Departure Versus Ticket Price',fontsize=15)
plt.xlabel('days_left for Departure', fontsize=15)
plt.ylabel('Price', fontsize=15)
plt.show()
```

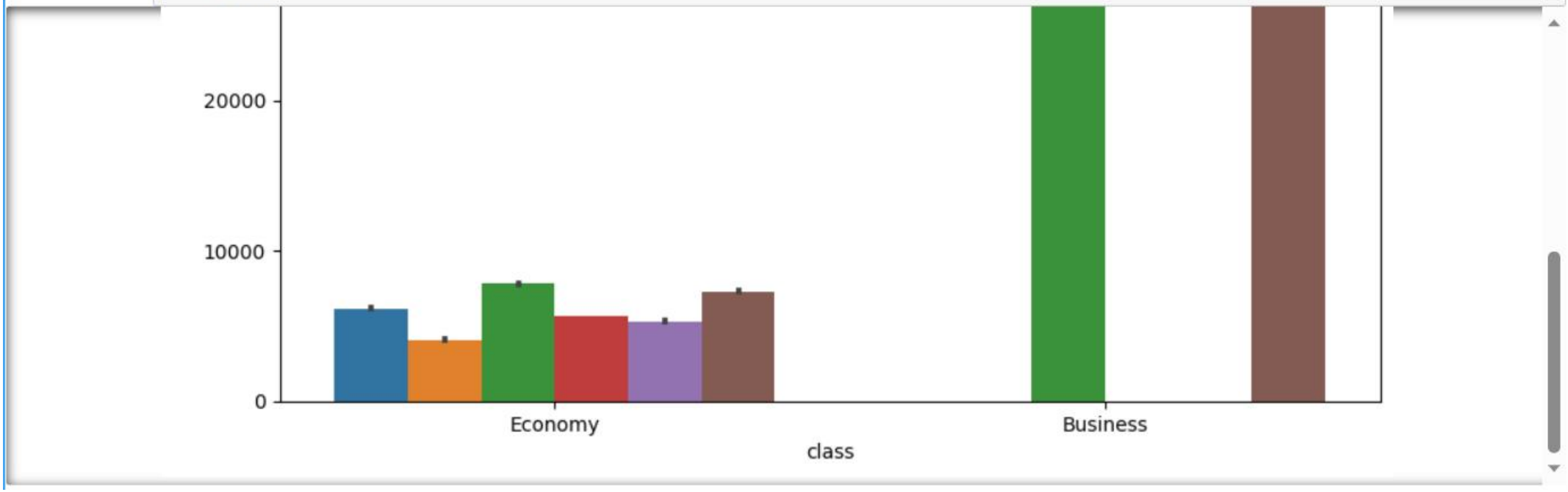


In [20]:
```
#Price range of all the flights
plt.figure(figsize=(10,5))
sns.barplot(x='airline',y='price',data=df)
plt.show()
```

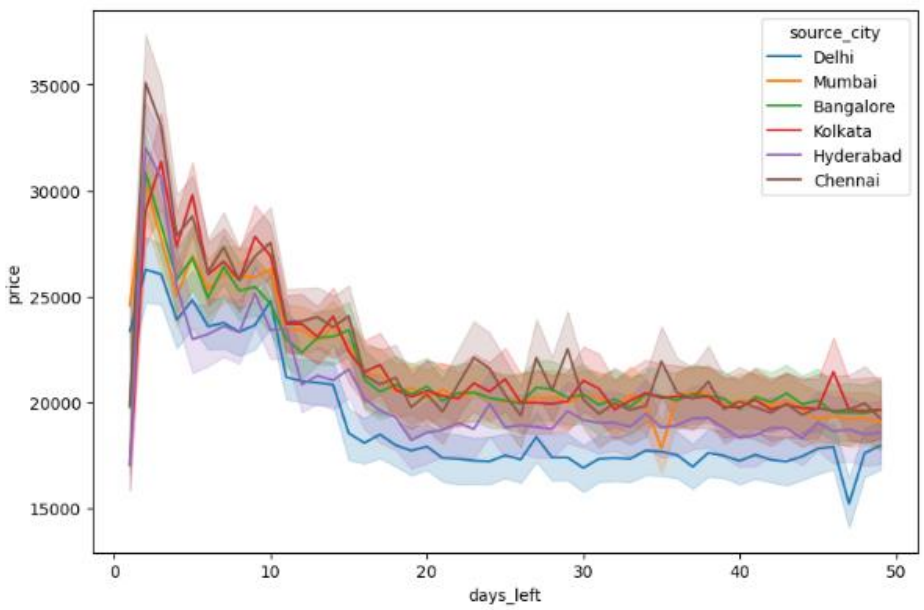


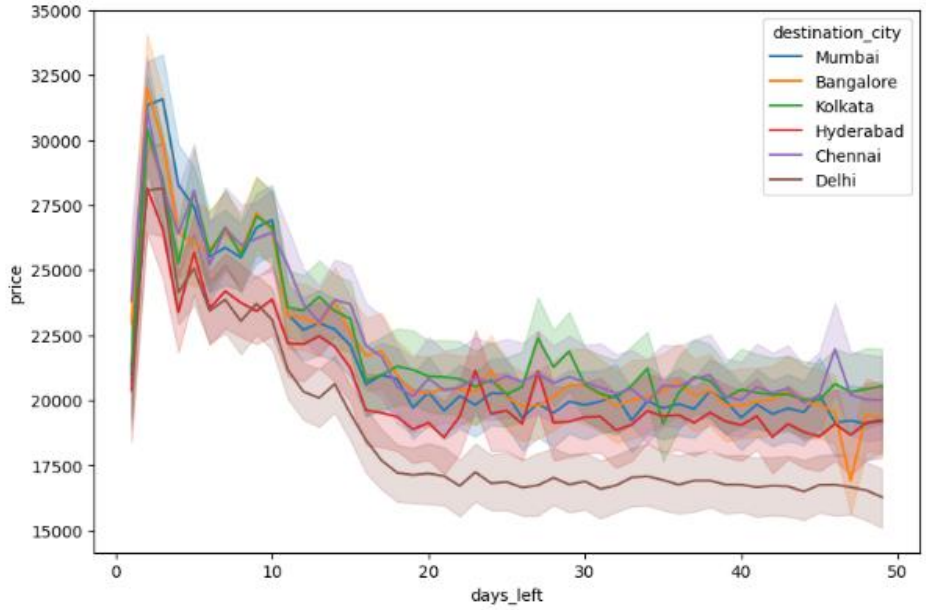
In [73]:
```
#Range of price of all the flights of Economy and Business class
plt.figure(figsize=(10,8))
sns.barplot(x='class',y='price',data=df,hue='airline')
plt.show()
```

In [73]:
```
#Range of price of all the flights of Economy and Business class
plt.figure(figsize=(10,8))
sns.barplot(x='class',y='price',data=df,hue='airline')
plt.show()
```

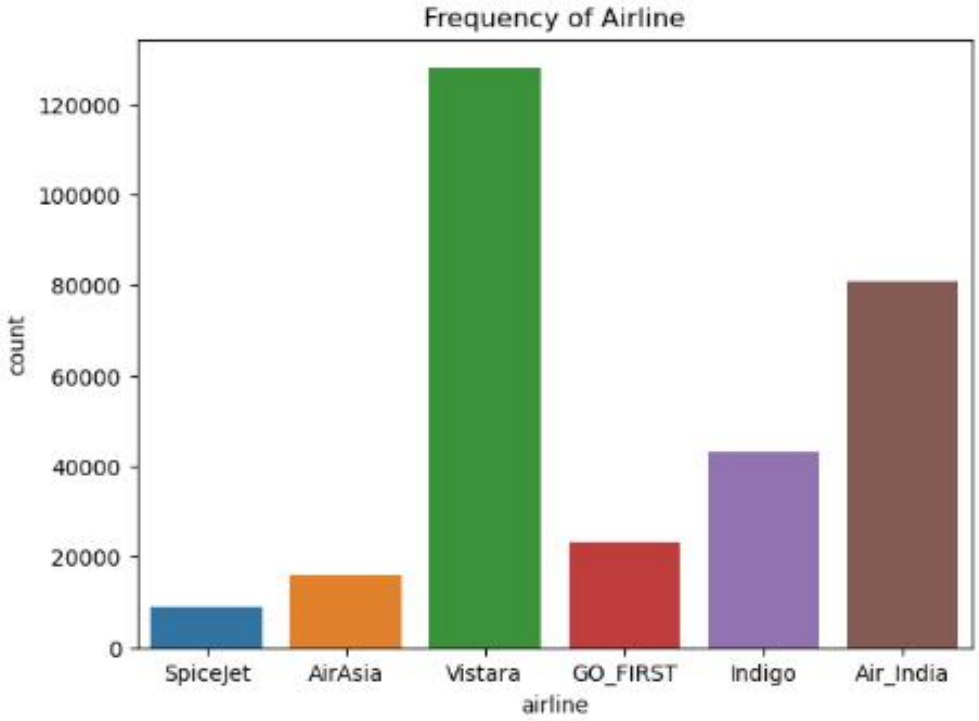


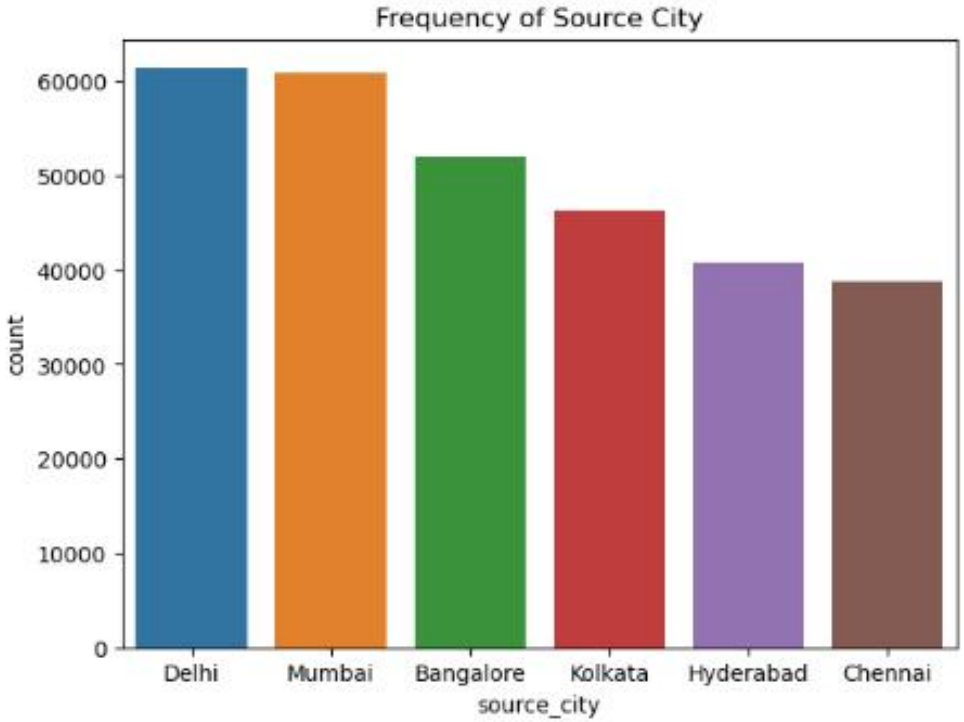
In [31]:
```
#Range of price of flights with source and destination city according to the days left
fig,ax=plt.subplots(1,2,figsize=(20,6))
sns.lineplot(x='days_left',y='price',data=df,hue='source_city',ax=ax[0])
sns.lineplot(x='days_left',y='price',data=df,hue='destination_city',ax=ax[1])
plt.show()
```

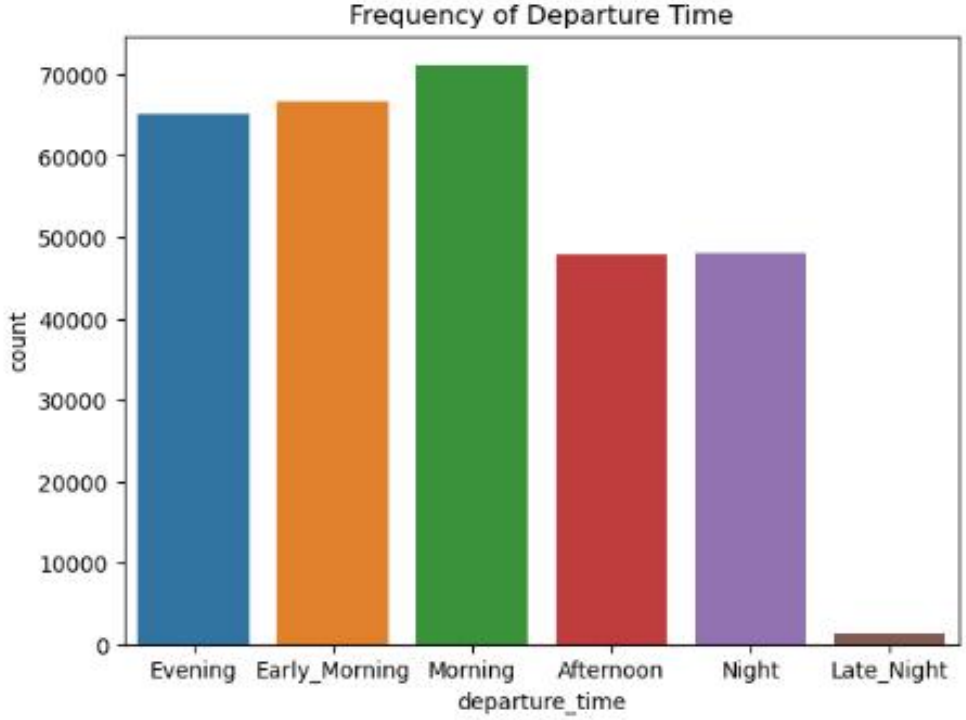


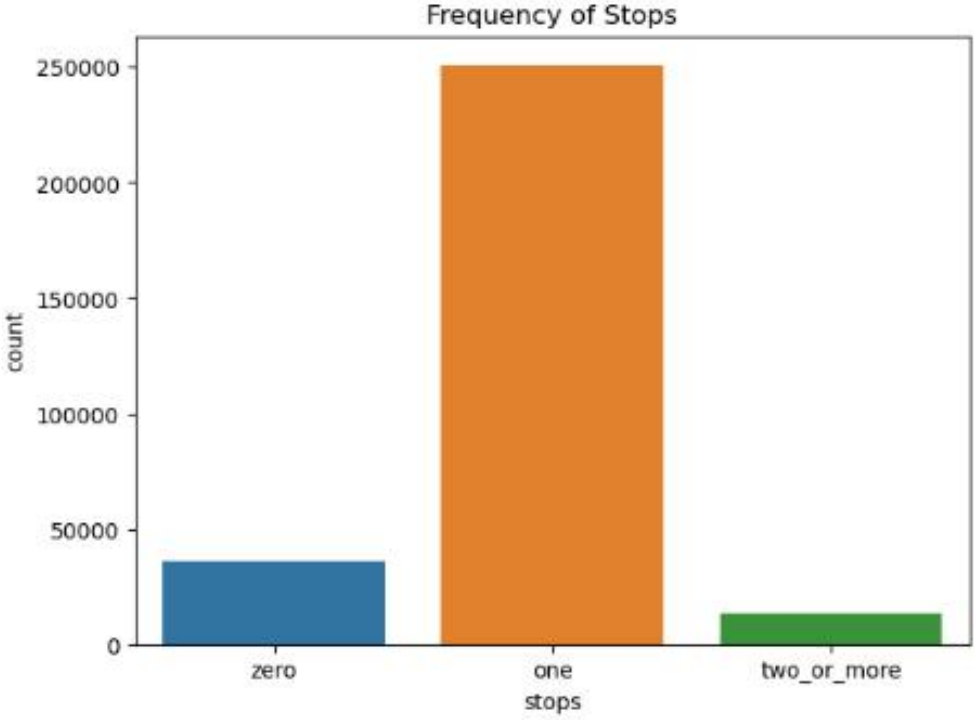


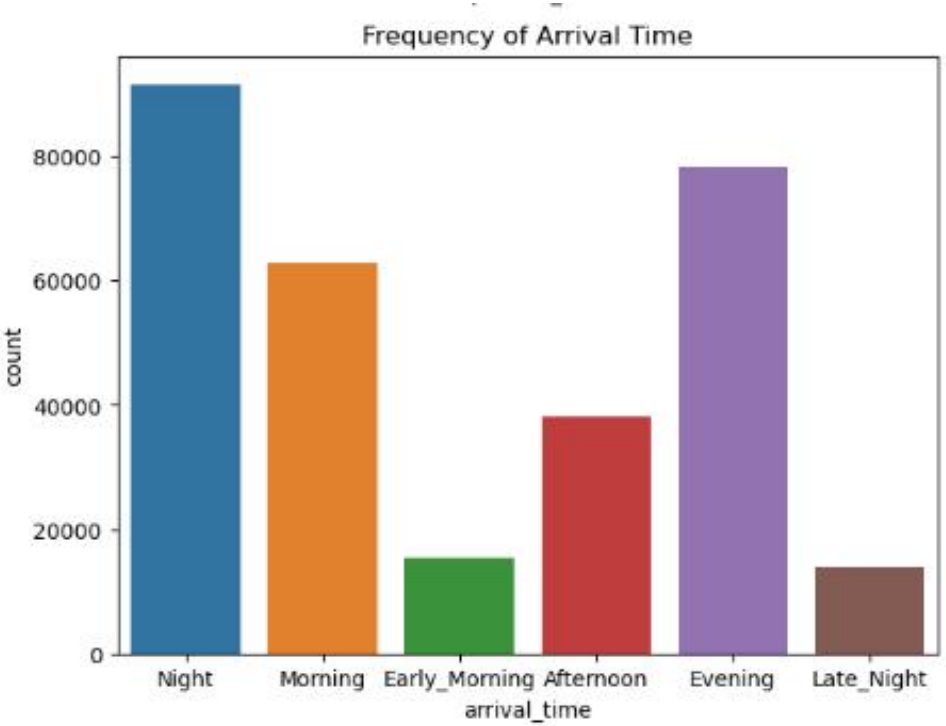
In [32]:
```
#Visualization of categorical features with countplot
plt.figure(figsize=(15,23))

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()
```

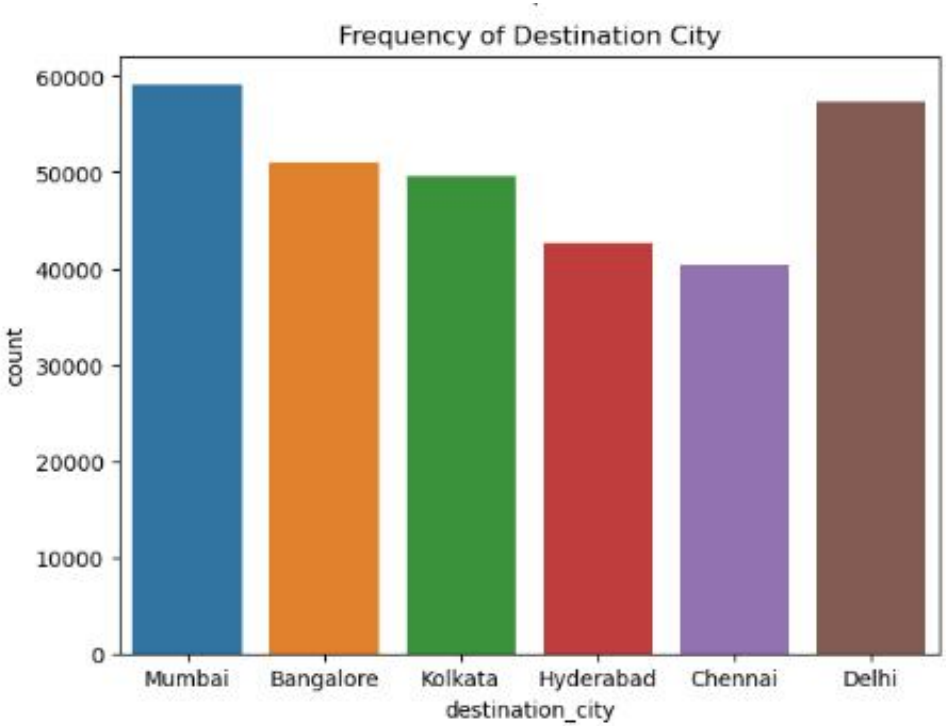

Frequency of Airline
count
airline
SpiceJet
AirAsia
Vistara
GO_FIRST
Indigo
Air_India
0
20000
40000
60000
80000
100000
120000

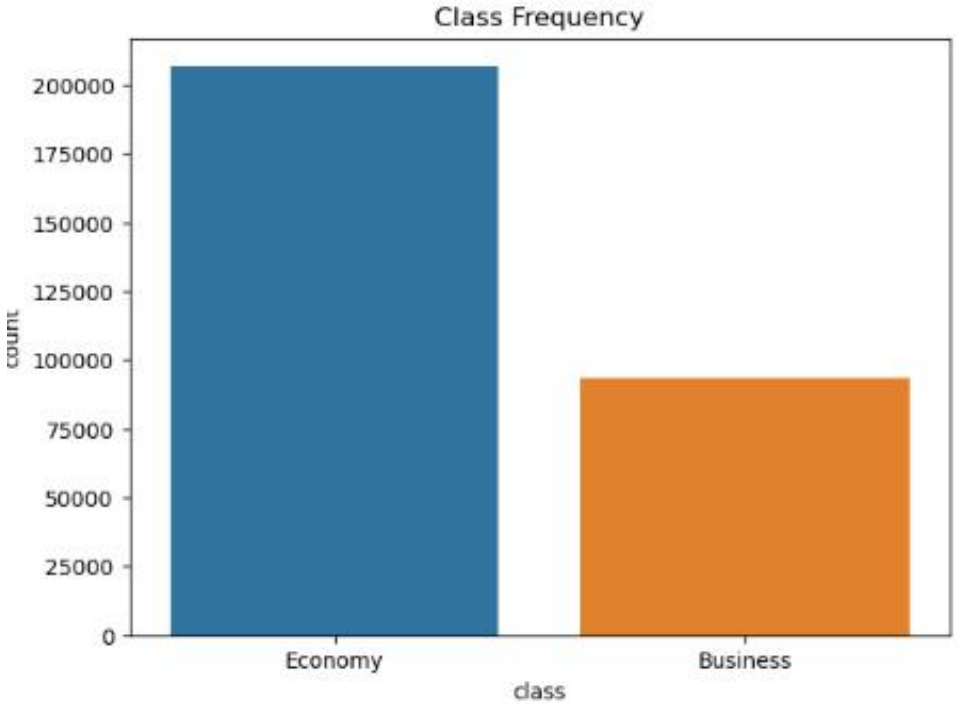

Frequency of Source City
count
source_city
Delhi
Mumbai
Bangalore
Kolkata
Hyderabad
Chennai
0
10000
20000
30000
40000
50000
60000


Frequency of Departure Time
count
departure_time
Evening
Early_Morning
Morning
Afternoon
Night
Late_Night
0
10000
20000
30000
40000
50000
60000
70000


Frequency of Stops
count
stops
zero
one
two_or_more
0
50000
100000
150000
200000
250000


Frequency of Arrival Time
count
0
20000
40000
60000
80000
Night
Morning
Early_Morning
Afternoon
Evening
Late_Night
arrival_time


Frequency of Destination City
count
0
10000
20000
30000
40000
50000
60000
Mumbai
Bangalore
Kolkata
Hyderabad
Chennai
Delhi
destination_city


Class Frequency
count
0
25000
50000
75000
100000
125000
150000
175000
200000
Economy
Business
class

Label Encoding ✓

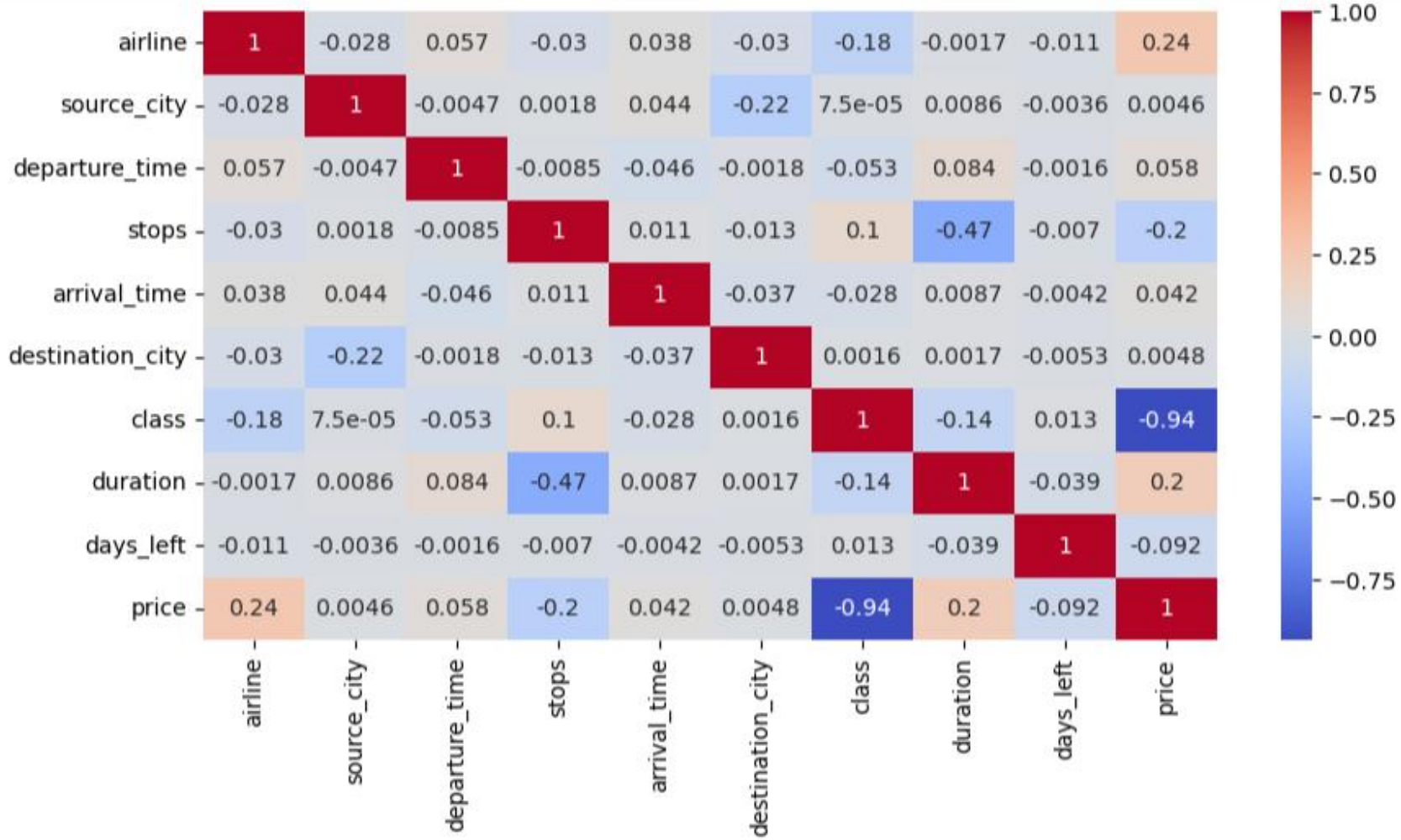
In [34]:
```
#Performing One Hot Encoding for categorical features of a dataframe
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
df['airline']=le.fit_transform(df['airline'])
df['source_city']=le.fit_transform(df['source_city'])
df['departure_time']=le.fit_transform(df['departure_time'])
df['stops']=le.fit_transform(df['stops'])
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'])
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  int64
 1   flight            300153 non-null  object
 2   source_city       300153 non-null  int64
 3   departure_time    300153 non-null  int64
 4   stops             300153 non-null  int64
 5   arrival_time      300153 non-null  int64
 6   destination_city  300153 non-null  int64
 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(1), int64(8), object(1)
memory usage: 24.0+ MB
```

Feature Selection

In [35]:
```
#Plotting the correlation graph to see the correlation between features and dependent variable.
plt.figure(figsize=(10,5))
sns.heatmap(df.corr(),annot=True,cmap="coolwarm")
plt.show()
```

```
C:\Users\user\AppData\Local\Temp\ipykernel_27112\824039507.py:3: FutureWarning: The default value of numeric_only in DataFrame.
corr is deprecated. In a future version, it will default to False. Select only valid columns or specify the value of numeric_on
ly to silence this warning.
  sns.heatmap(df.corr(),annot=True,cmap="coolwarm")
```



In [36]:
```
#Selecting the features using VIF. VIF should be less than 5. So drop the stops feature.
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  3.393124
1      source_city  2.927766
2    departure_time  2.779427
3             stops  1.426614
4      arrival_time  3.684550
5  destination_city  2.885337
6             class  2.849370
7          duration  4.113876
8         days_left  3.976790
```

In [37]:
```
#Dropping the stops column. All features are having VIF less than 5.
df = df.drop(columns=["stops"])

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

Linear Regression

In [50]:
```
#Applying standardization and implementing Linear Regression Model to predict the price of a flight.
#df = df.drop(columns=["flight"])

X = df.drop(columns=["price"])
y = df['price']

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)

from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)

from sklearn.linear_model import LinearRegression
lr = LinearRegression()
lr.fit(X_train,y_train)

y_pred = lr.predict(X_test)

difference = pd.DataFrame(np.c_[y_test,y_pred],columns=["Actual_Value","Predicted_Value"])
difference
```

Out[50]:

	Actual_Value	Predicted_Value
0	7366.0	4673.755319
1	64831.0	51713.744720
2	6195.0	6610.897658
3	60160.0	55489.844234
4	6578.0	5120.342596
...	...	...
60026	5026.0	4960.777767
60027	3001.0	4693.865426
60028	6734.0	4974.962678
60029	5082.0	2729.650066
60030	66465.0	59638.748598

60031 rows × 2 columns

In [52]:

```
from sklearn.metrics import r2_score
print(r2_score(y_test,y_pred))

from sklearn import metrics
mean_abs_error = metrics.mean_absolute_error(y_test,y_pred)
print(mean_abs_error)

from sklearn.metrics import mean_absolute_percentage_error
print(mean_absolute_percentage_error(y_test,y_pred))

mean_sq_error = metrics.mean_squared_error(y_test,y_pred)
print(mean_sq_error)

root_mean_sq_error = np.sqrt(metrics.mean_squared_error(y_test,y_pred))
print(root_mean_sq_error)
```

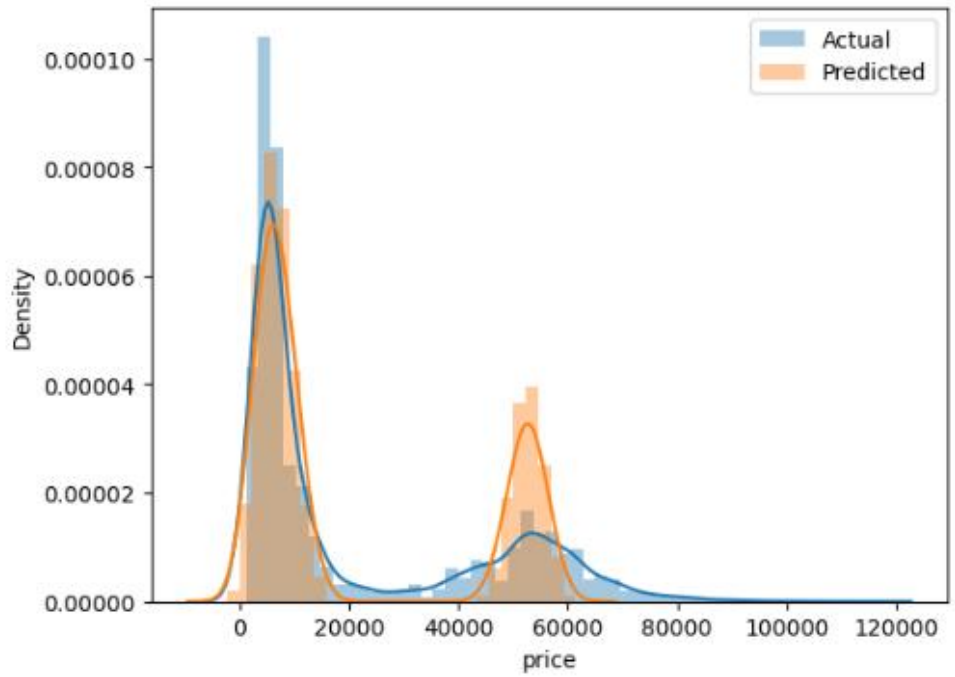
```
0.897752737512321
4468.426673542099
0.34765804610681533
52706651.333342075
7259.934664536732
```

In [60]:

```
#Calculating r2 score,MAE, MAPE, MSE, RMSE. Root Mean square error(RMSE) of the Linear regression model is 7259.93 and
#Mean absolute percentage error(MAPE) is 34 percent. Lower the RMSE and MAPE better the model.
```

In [53]:
```
#Plotting the graph of actual and predicted price of flight
sns.distplot(y_test,label="Actual")
sns.distplot(y_pred,label="Predicted")
plt.legend()
plt.show()
```

```
C:\Users\user\AppData\Local\Temp\ipykernel_27112\2659814403.py:2: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(y_test,label="Actual")
C:\Users\user\AppData\Local\Temp\ipykernel_27112\2659814403.py:3: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(y_pred,label="Predicted")
```



Decision Tree Regressor

In [54]:

```
from sklearn.tree import DecisionTreeRegressor
dt=DecisionTreeRegressor()
dt.fit(X_train,y_train)

y_pred = dt.predict(X_test)

print(r2_score(y_test,y_pred))

mean_abs_error = metrics.mean_absolute_error(y_test,y_pred)
print(mean_abs_error)

from sklearn.metrics import mean_absolute_percentage_error
print(mean_absolute_percentage_error(y_test,y_pred))

mean_sq_error = metrics.mean_squared_error(y_test,y_pred)
print(mean_sq_error)

root_mean_sq_error = np.sqrt(metrics.mean_squared_error(y_test,y_pred))
print(root_mean_sq_error)
```

```
0.9746155826561153
1219.8505161222256
0.07751052777094827
13085217.165646823
3617.349466895177
```

In [59]:

```
#Mean absolute percentage error is 7.7 percent and RMSE is 3617 which is less than the linear regression model
```

Random Forest Regressor

In [57]:
```
from sklearn.ensemble import RandomForestRegressor
rfr = RandomForestRegressor()
rfr.fit(X_train,y_train)

y_pred = rfr.predict(X_test)

print(r2_score(y_test,y_pred))

mean_abs_error = metrics.mean_absolute_error(y_test,y_pred)
print(mean_abs_error)

from sklearn.metrics import mean_absolute_percentage_error
print(mean_absolute_percentage_error(y_test,y_pred))

mean_sq_error = metrics.mean_squared_error(y_test,y_pred)
print(mean_sq_error)

root_mean_sq_error = np.sqrt(metrics.mean_squared_error(y_test,y_pred))
print(root_mean_sq_error)
```

```
0.9845756673709423
1121.8688138116072
0.07329615483635711
7950970.051909215
2819.746451706113
```

In [61]:
```
#Mean absolute percentage error is 7.3 percent and RMSE is 2819 which is less than the linear regression and decision tree model
```

In [58]:
```
sns.distplot(y_test,label="Actual")
sns.distplot(y_pred,label="Predicted")
plt.legend()
plt.show()
```

```
C:\Users\user\AppData\Local\Temp\ipykernel_27112\326679581.py:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(y_test,label="Actual")
C:\Users\user\AppData\Local\Temp\ipykernel_27112\326679581.py:2: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(y_pred,label="Predicted")
```