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

In [2]:
```
df=pd.read_csv("/Users/tanya/Downloads/projects/Clean_Dataset.csv")
```

In [3]:
```
df.head()
```

Out[3]:

	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 [4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 300153 entries, 0 to 300152
Data columns (total 12 columns):
 #   Column            Non-Null Count   Dtype  
---  ------            --------------   -----  
 0   Unnamed: 0        300153 non-null  int64  
 1   airline           300153 non-null  object 
 2   flight            300153 non-null  object 
 3   source_city       300153 non-null  object 
 4   departure_time    300153 non-null  object 
 5   stops             300153 non-null  object 
 6   arrival_time      300153 non-null  object 
 7   destination_city  300153 non-null  object 
 8   class             300153 non-null  object 
 9   duration          300153 non-null  float64
 10  days_left         300153 non-null  int64  
 11  price             300153 non-null  int64  
dtypes: float64(1), int64(3), object(8)
memory usage: 27.5+ MB
```

In [5]:
```
df.shape
```

Out[5]:
```
(300153, 12)
```

In [6]:
```
#Dropping the first column since it isn't required.
df.drop("Unnamed: 0",axis=1,inplace=True)
```

In [7]:
```
df.isnull().sum()
```

Out[7]:
```
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
```

No null values

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

Out[8]:
```
0
```

In [9]:
```
df["stops"].value_counts()
```

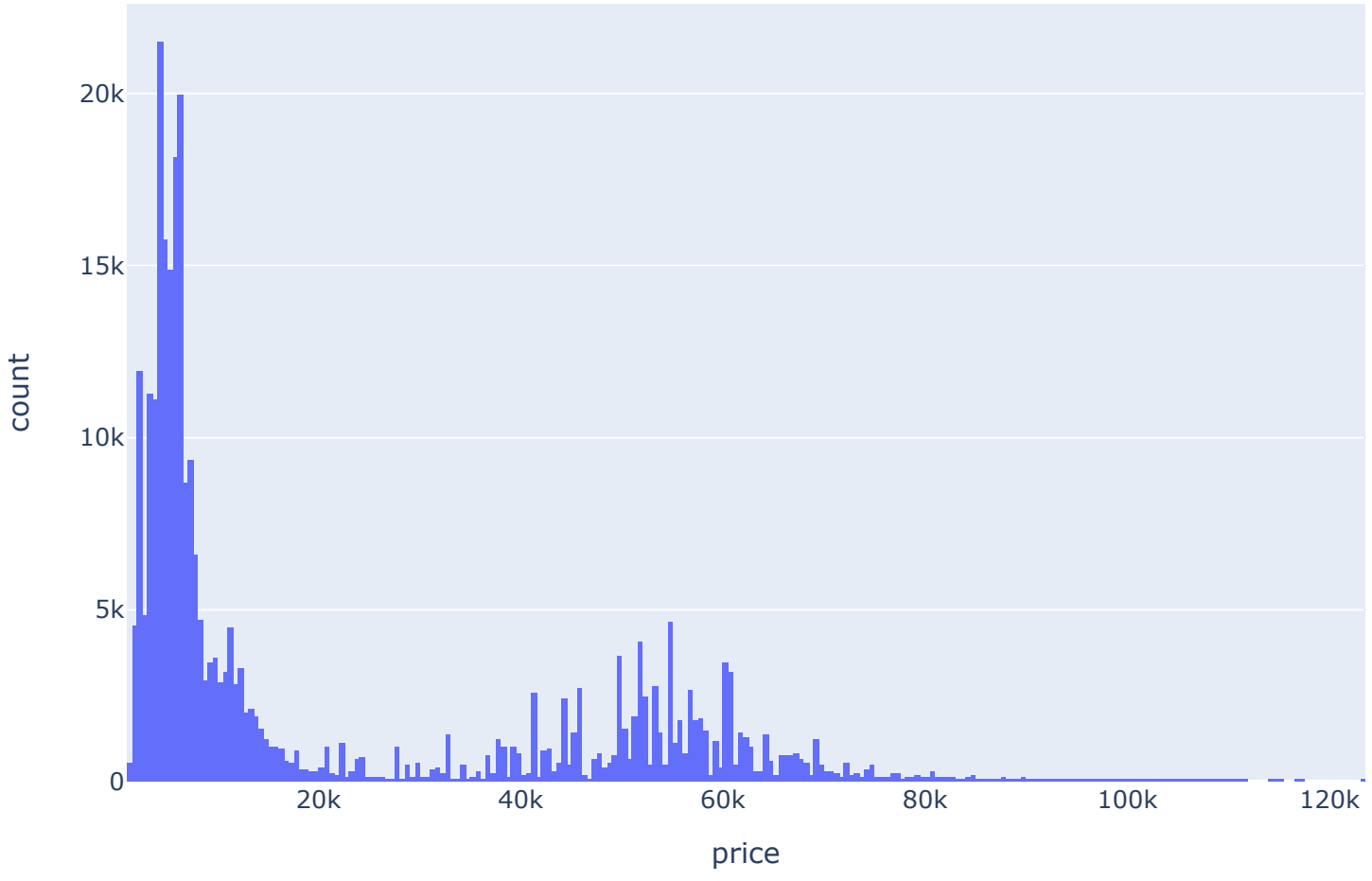
Out[9]:
```
one            250863
zero            36004
two_or_more     13286
Name: stops, dtype: int64
```

In [10]:
```
df["price"].agg(["min","max","median","mean","std"])
```

Out[10]:
```
min         1105.000000
max       123071.000000
median      7425.000000
mean       20889.660523
std        22697.767366
Name: price, dtype: float64
```

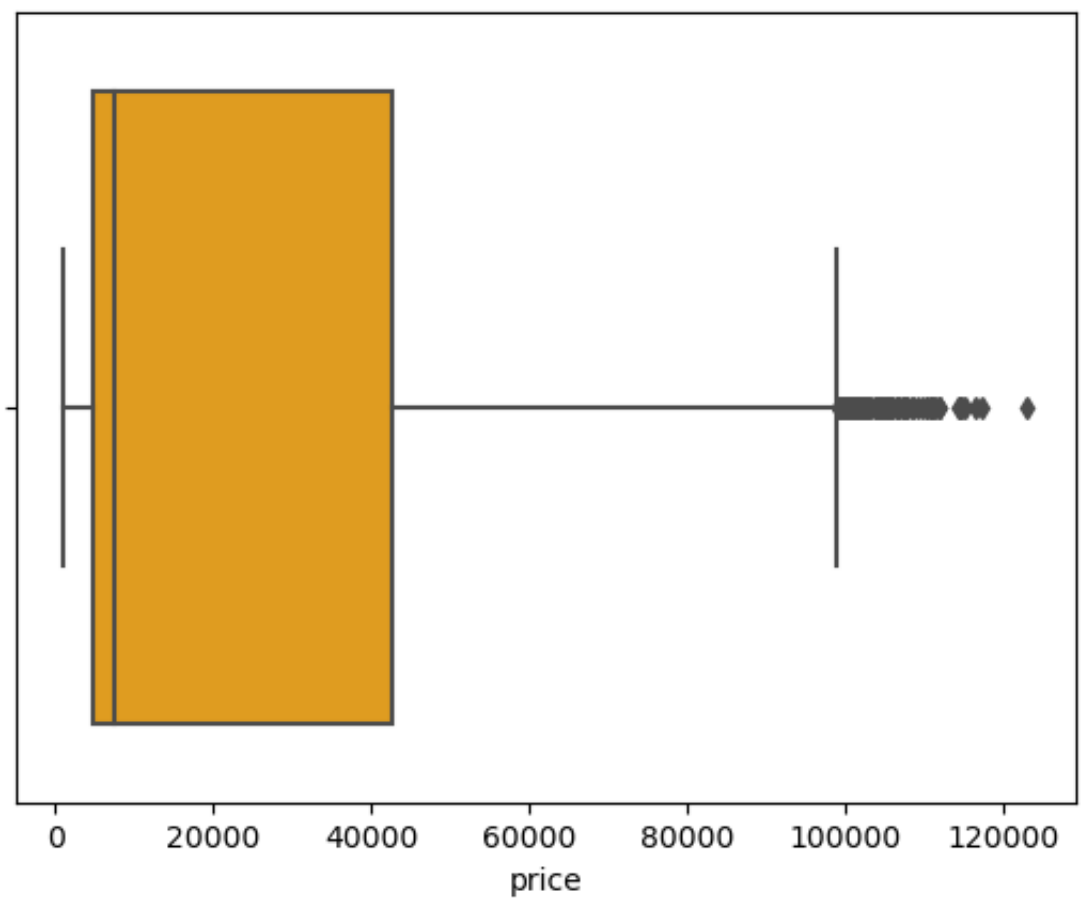
Since the median is smaller than mean, the distribution will be right skewed.

In [11]:
```
import plotly.express as px
fig = px.histogram(df, x="price")
fig.show()
```



In [12]:
```
#outlier analysis
sns.boxplot(data=df,x="price",color="orange")
```

Out[12]: <AxesSubplot:xlabel='price'>



There are outliers.

In [13]:
```
df.nunique()
```

Out[13]:
```
airline                6
flight              1561
source_city            6
departure_time         6
stops                  3
arrival_time           6
destination_city       6
class                  2
duration             476
days_left             49
price              12157
dtype: int64
```

In [14]:
```
for col in df.columns:
    if df[col].dtype == 'object':
        print(df[col].unique())
```

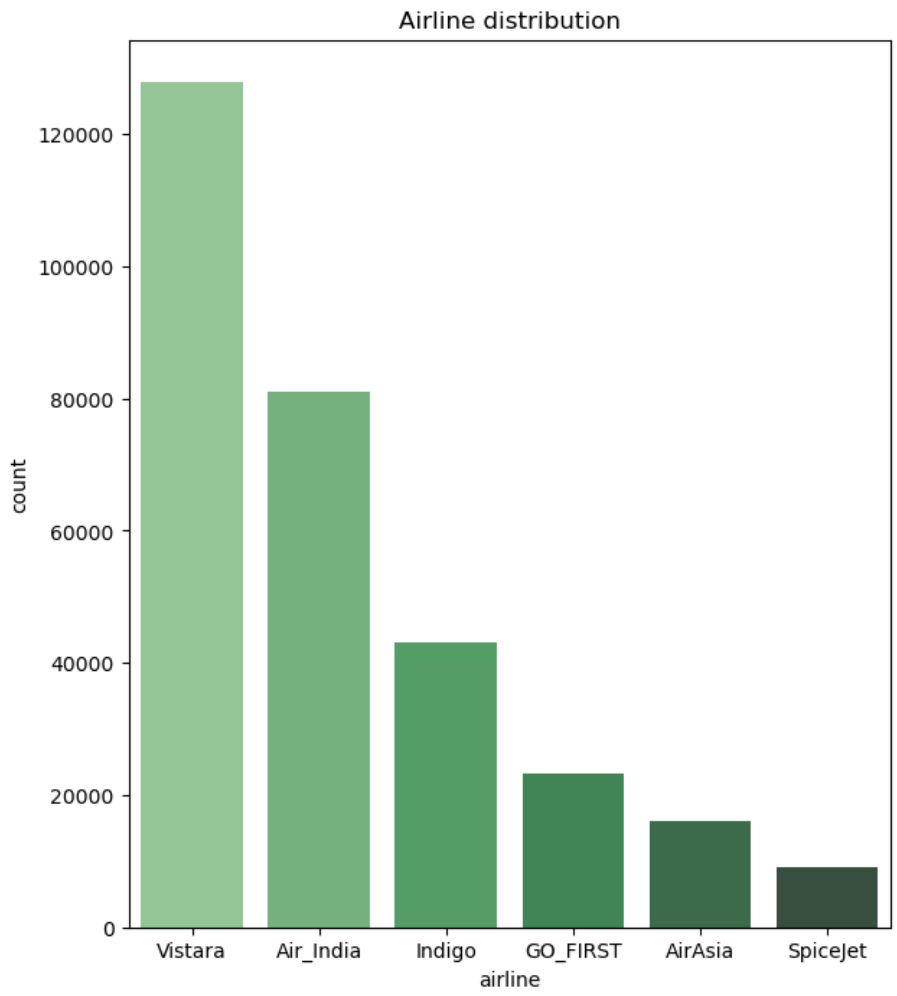
```
['SpiceJet' 'AirAsia' 'Vistara' 'GO_FIRST' 'Indigo' 'Air_India']
['SG-8709' 'SG-8157' 'I5-764' ... '6E-7127' '6E-7259' 'AI-433']
['Delhi' 'Mumbai' 'Bangalore' 'Kolkata' 'Hyderabad' 'Chennai']
['Evening' 'Early_Morning' 'Morning' 'Afternoon' 'Night' 'Late_Night']
['zero' 'one' 'two_or_more']
['Night' 'Morning' 'Early_Morning' 'Afternoon' 'Evening' 'Late_Night']
['Mumbai' 'Bangalore' 'Kolkata' 'Hyderabad' 'Chennai' 'Delhi']
['Economy' 'Business']
```

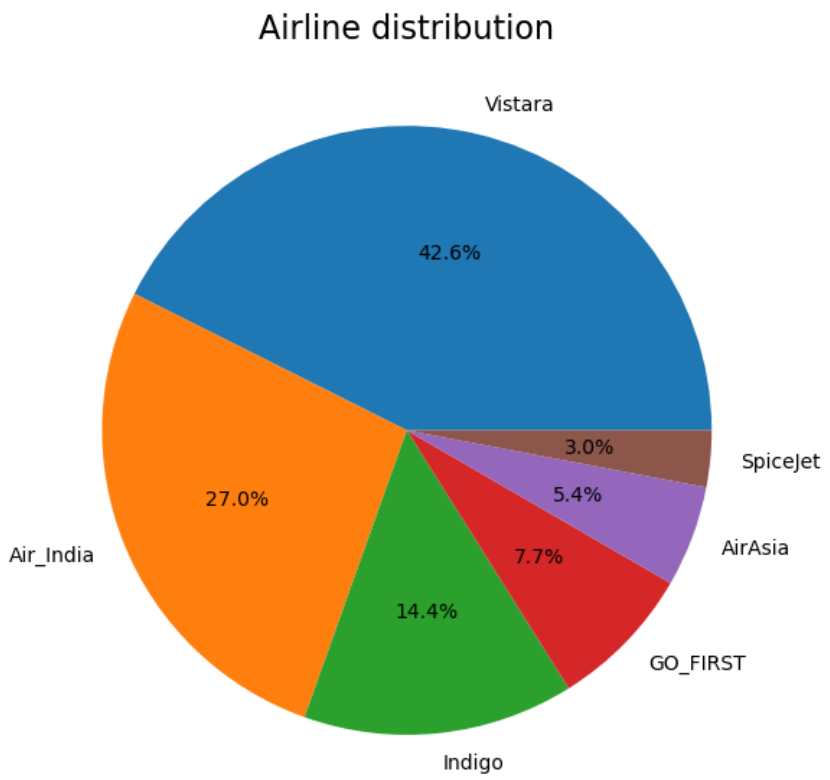
Ratio between the airlines.

In [15]:
```
df["airline"].value_counts()
```

Out[15]:
```
Vistara      127859
Air_India     80892
Indigo        43120
GO_FIRST      23173
AirAsia       16098
SpiceJet       9011
Name: airline, dtype: int64
```

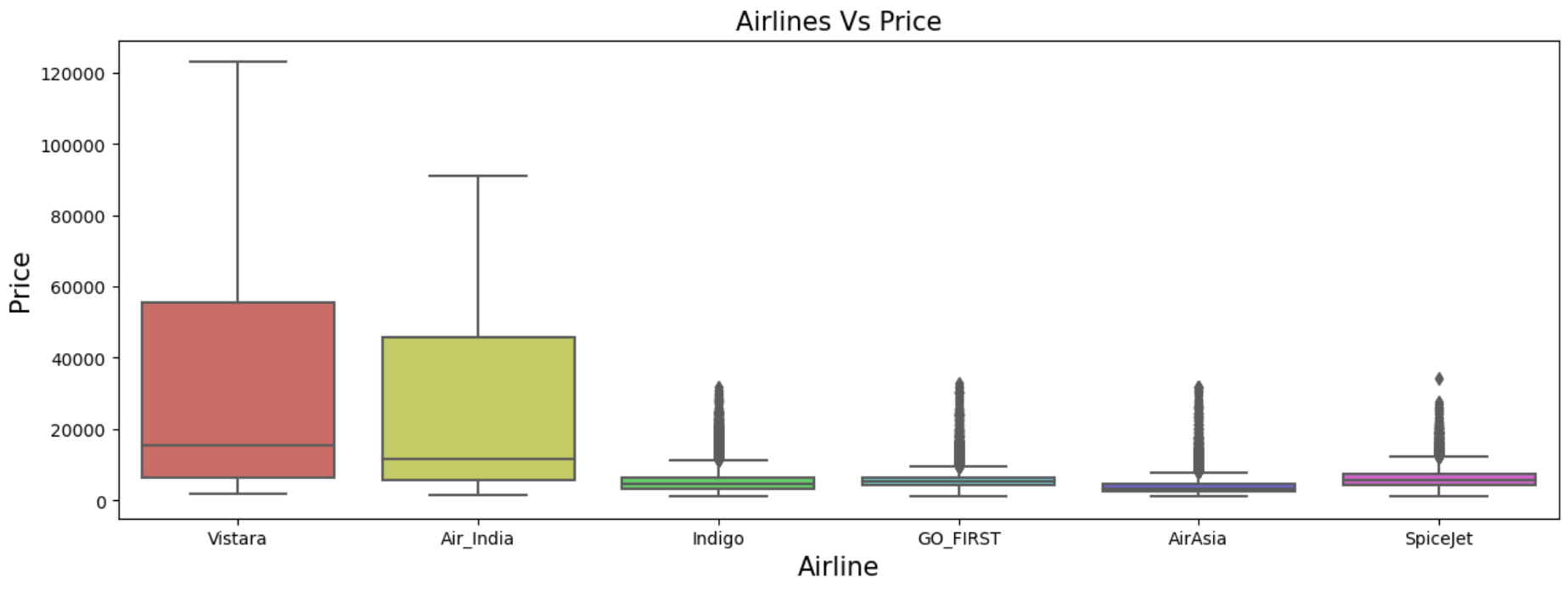
In [16]:
```
plt.figure(figsize=(15,8))
plt.subplot(1,2,1)
sns.countplot(x='airline',data=df,palette="Greens_d",order=df['airline'].value_counts().index)
plt.title("Airline distribution")
plt.subplot(1,2,2)
plt.title('Airline distribution', fontsize=16)
df['airline'].value_counts().plot(kind='pie', legend=None, ylabel='', autopct='%1.1f%%')
plt.show()
```





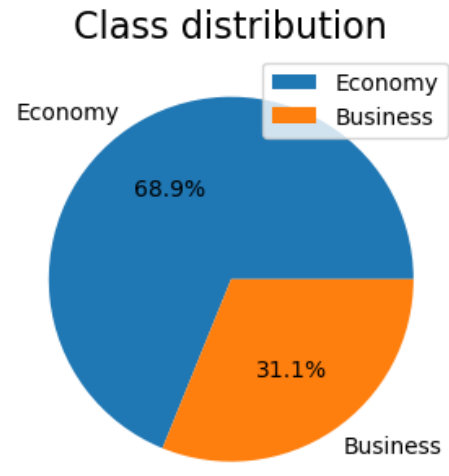
How is price from one airline to another?

In [17]:
```
plt.figure(figsize=(15,5))
sns.boxplot(x=df['airline'],y=df['price'],palette='hls',order=df['airline'].value_counts().index)
plt.title('Airlines Vs Price',fontsize=15)
plt.xlabel('Airline',fontsize=15)
plt.ylabel('Price',fontsize=15)
plt.show()
```



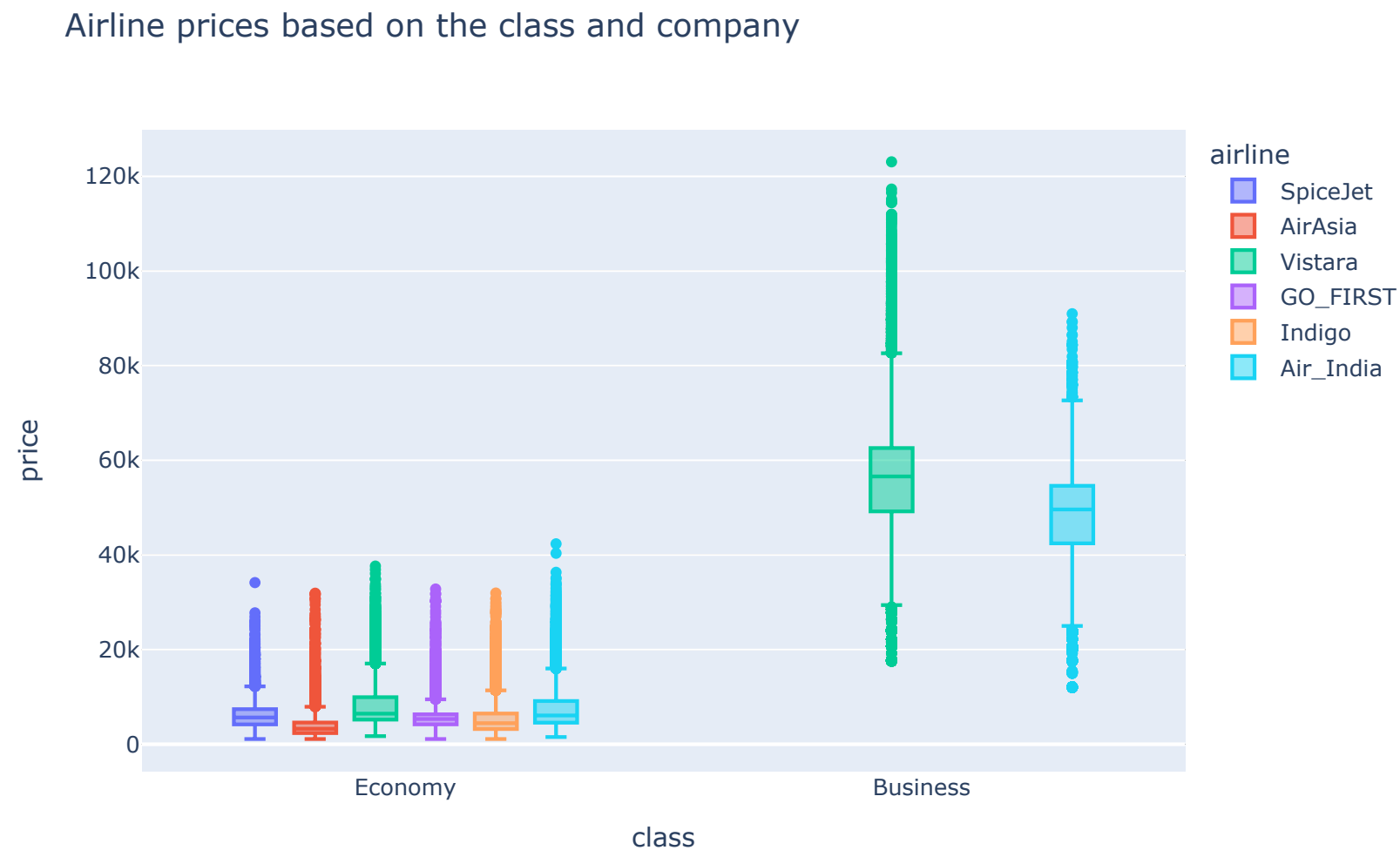
Ratio between the classes.

In [18]:
```
plt.figure(figsize=(15,8))
plt.subplot(2,2,1)
sns.countplot(x='class',data=df,palette="Greens_d",order=df['class'].value_counts().index)
plt.title("Class distribution")
plt.subplot(2,2,2)
plt.title('Class distribution', fontsize=16)
df['class'].value_counts().plot(kind='pie', legend=None, ylabel='', autopct='%1.1f%%')
plt.legend(['Economy','Business'])
plt.show()
```



How does the ticket price vary between Economy and Business class?

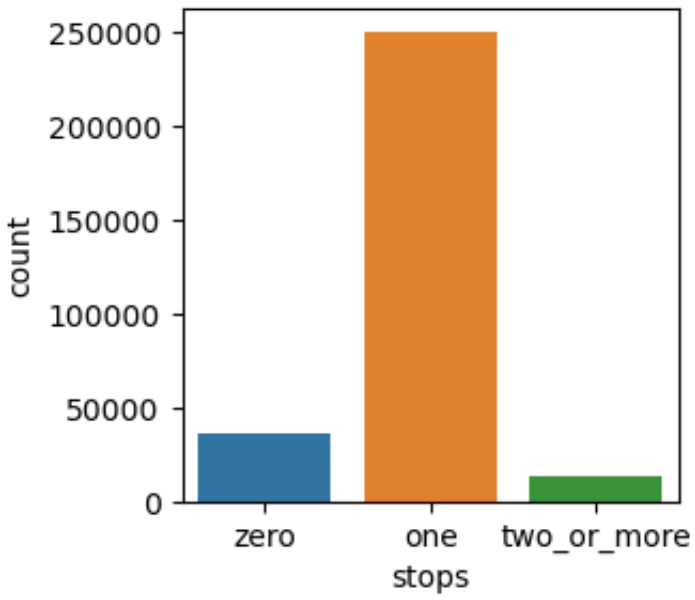
In [19]:
```
px.box(df,x='class',y='price',color='airline',title='Airline prices based on the class and company')
```



There are slight differences between each companies on this graph, AirAsia seems to have the cheapest flights when Air India and Vistara are more expensive. However it looks like Vistara's business tickets are a little more expensive than the Air India's ones.

Number of stops

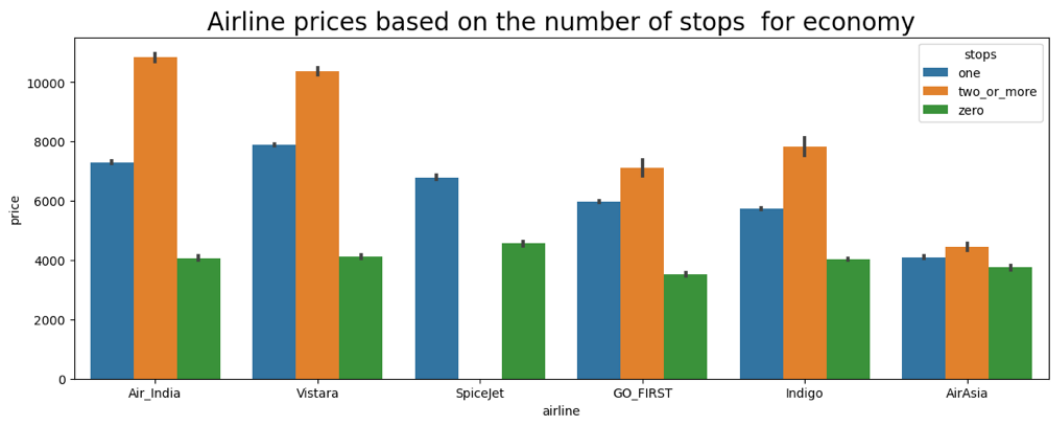
In [20]:
```
plt.figure(figsize=(3,3))
sns.countplot(x='stops',data=df)
plt.show()
```

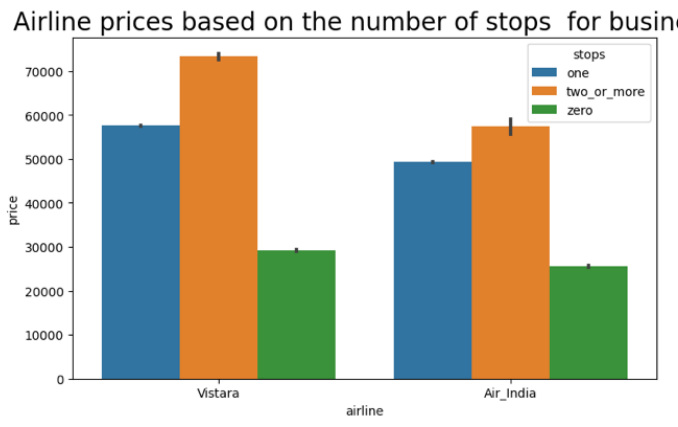


How Does the Ticket Price vary with the number of stops of a Flight?

In [21]:
```
fig, axs = plt.subplots (1, 2, gridspec_kw={'width_ratios': [5, 3]}, figsize=(25, 5))
sns.barplot(y = "price", x = "airline",hue="stops",data = df.loc[df["class"]=='Economy'].sort_values("pr
axs[0].set_title("Airline prices based on the number of stops  for economy",fontsize=20)
sns.barplot(y = "price", x = "airline",hue="stops",data = df.loc[df["class"]=='Business'].sort_values("p
axs[1].set_title("Airline prices based on the number of stops  for business",fontsize=20)
```

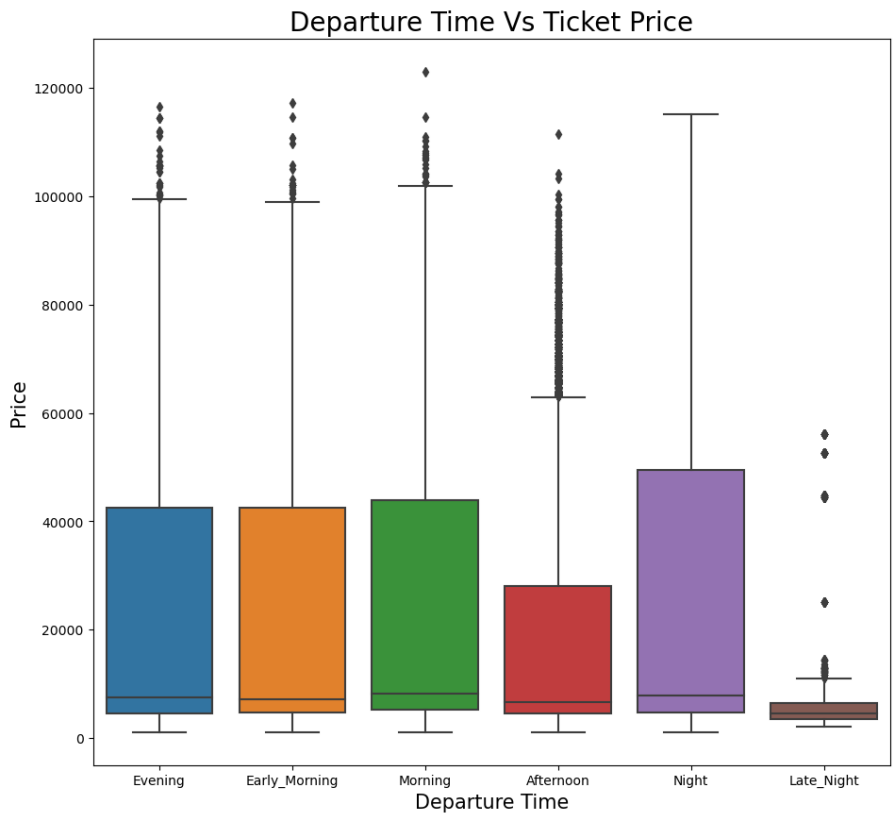
Out[21]:
```
Text(0.5, 1.0, 'Airline prices based on the number of stops  for business')
```

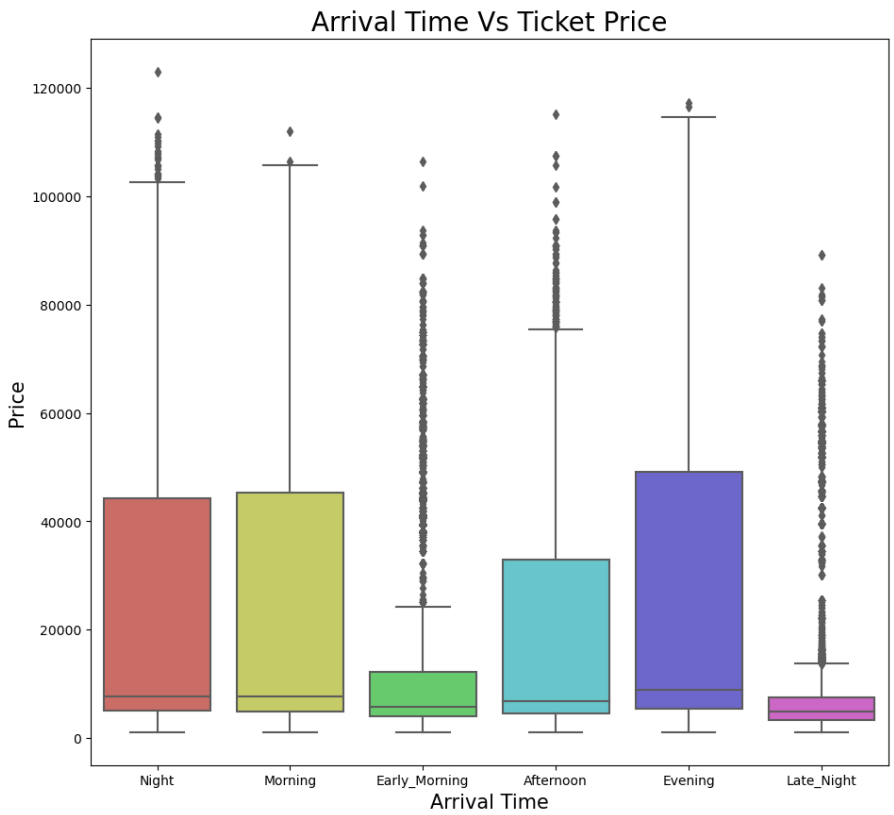




How the Ticket Price change based on the Departure Time and Arrival Time?

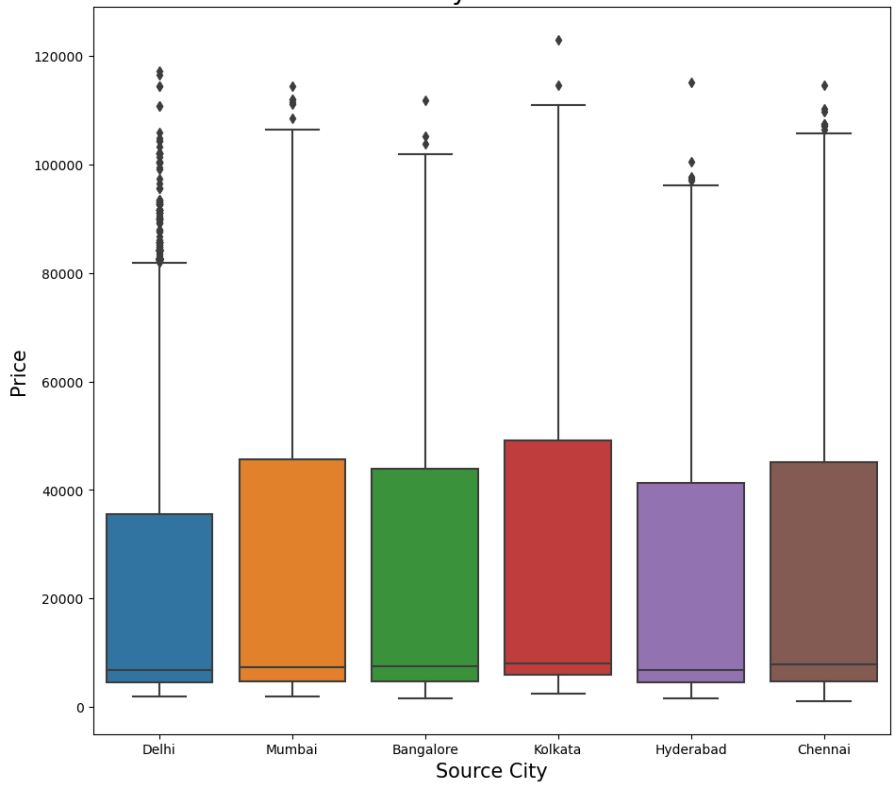
In [22]:
```
plt.figure(figsize=(24,10))
plt.subplot(1,2,1)
sns.boxplot(x='departure_time',y='price',data=df)
plt.title('Departure Time Vs Ticket Price',fontsize=20)
plt.xlabel('Departure Time',fontsize=15)
plt.ylabel('Price',fontsize=15)
plt.subplot(1,2,2)
sns.boxplot(x='arrival_time',y='price',data=df,palette='hls')
plt.title('Arrival Time Vs Ticket Price',fontsize=20)
plt.xlabel('Arrival Time',fontsize=15)
plt.ylabel('Price',fontsize=15)
plt.show()
```

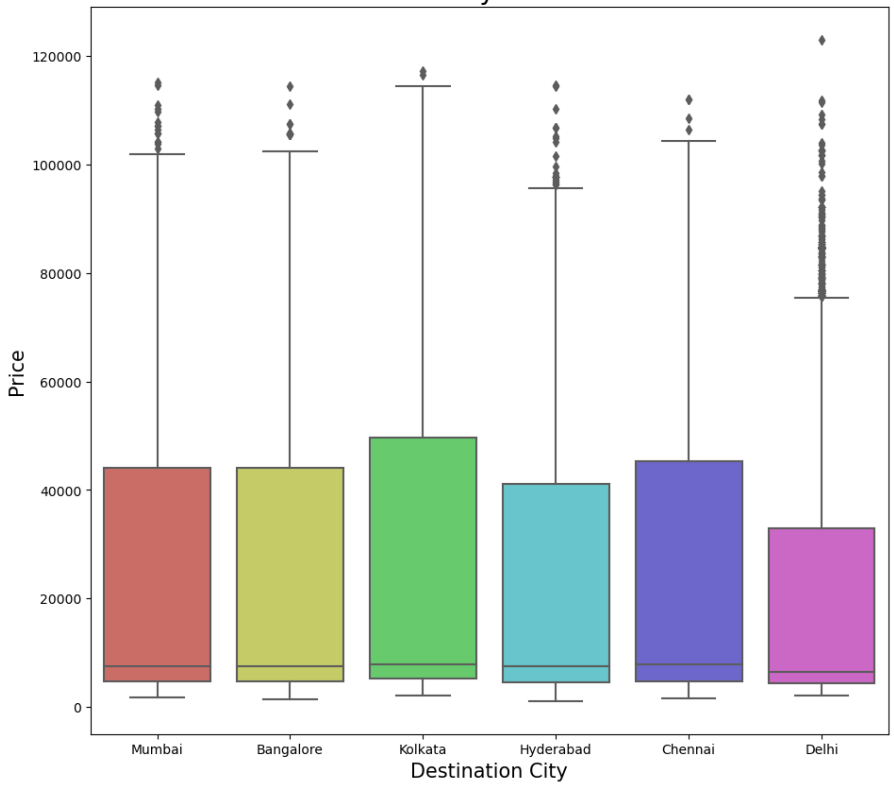




How the price changes with change in Source city and Destination city?

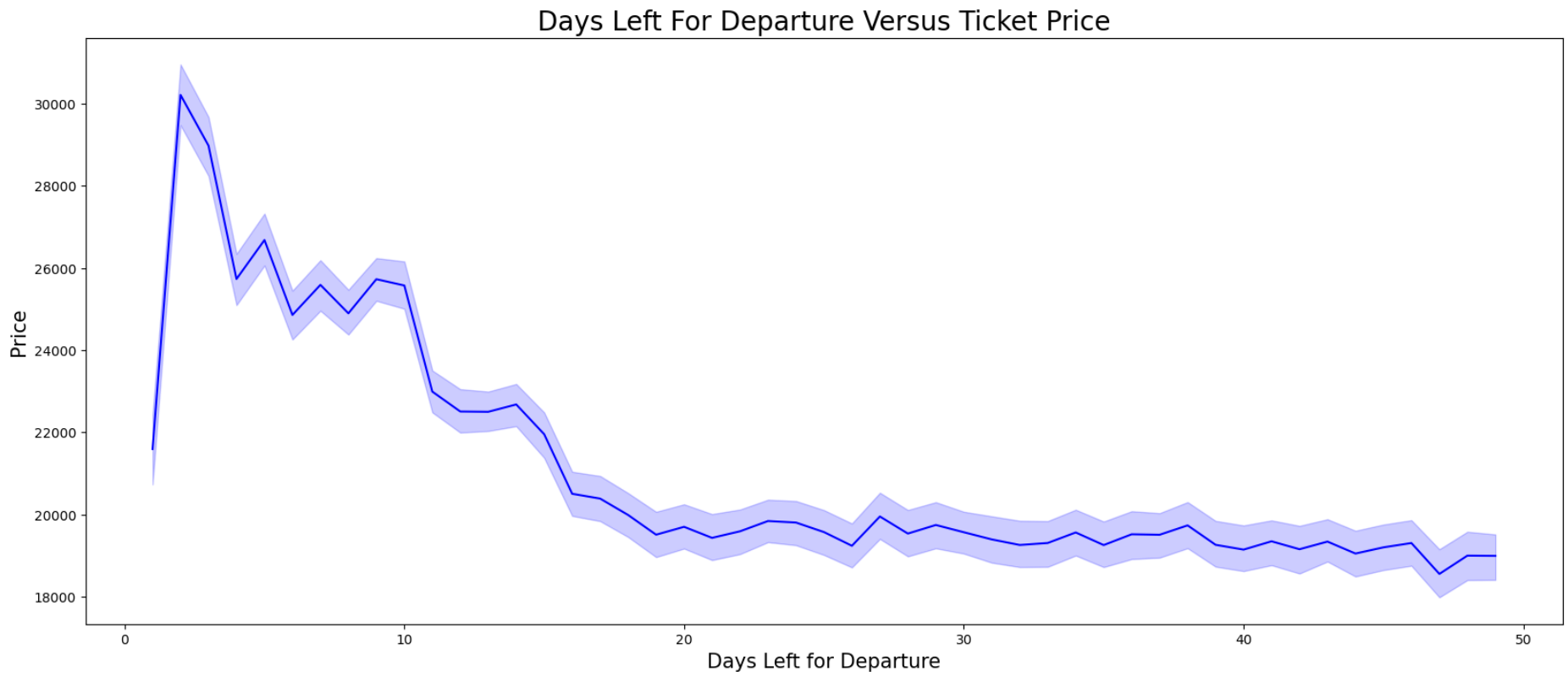
In [23]:
```
plt.figure(figsize=(24,10))
plt.subplot(1,2,1)
sns.boxplot(x='source_city',y='price',data=df)
plt.title('Source City Vs Ticket Price',fontsize=20)
plt.xlabel('Source City',fontsize=15)
plt.ylabel('Price',fontsize=15)
plt.subplot(1,2,2)
sns.boxplot(x='destination_city',y='price',data=df,palette='hls')
plt.title('Destination City Vs Ticket Price',fontsize=20)
plt.xlabel('Destination City',fontsize=15)
plt.ylabel('Price',fontsize=15)
plt.show()
```



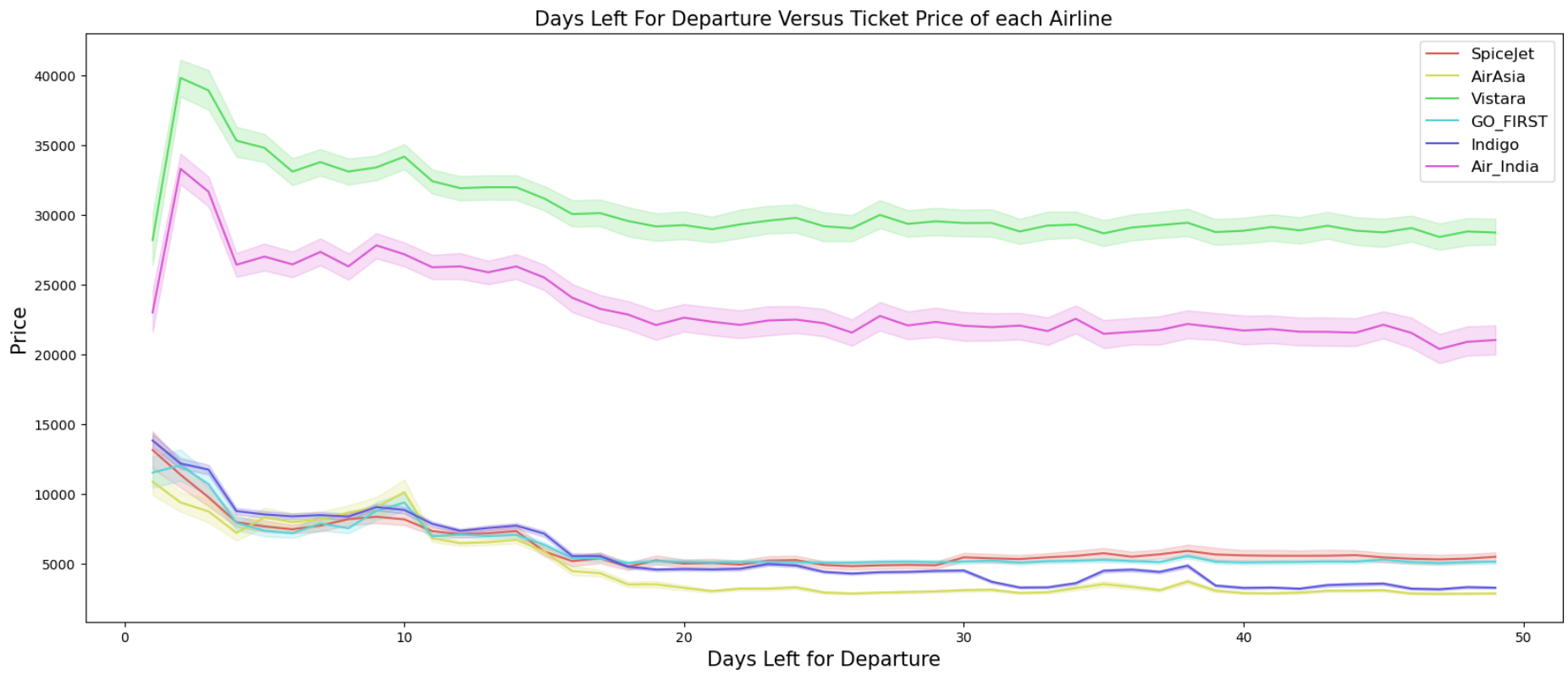


How does the price affected on the days left for Departure?

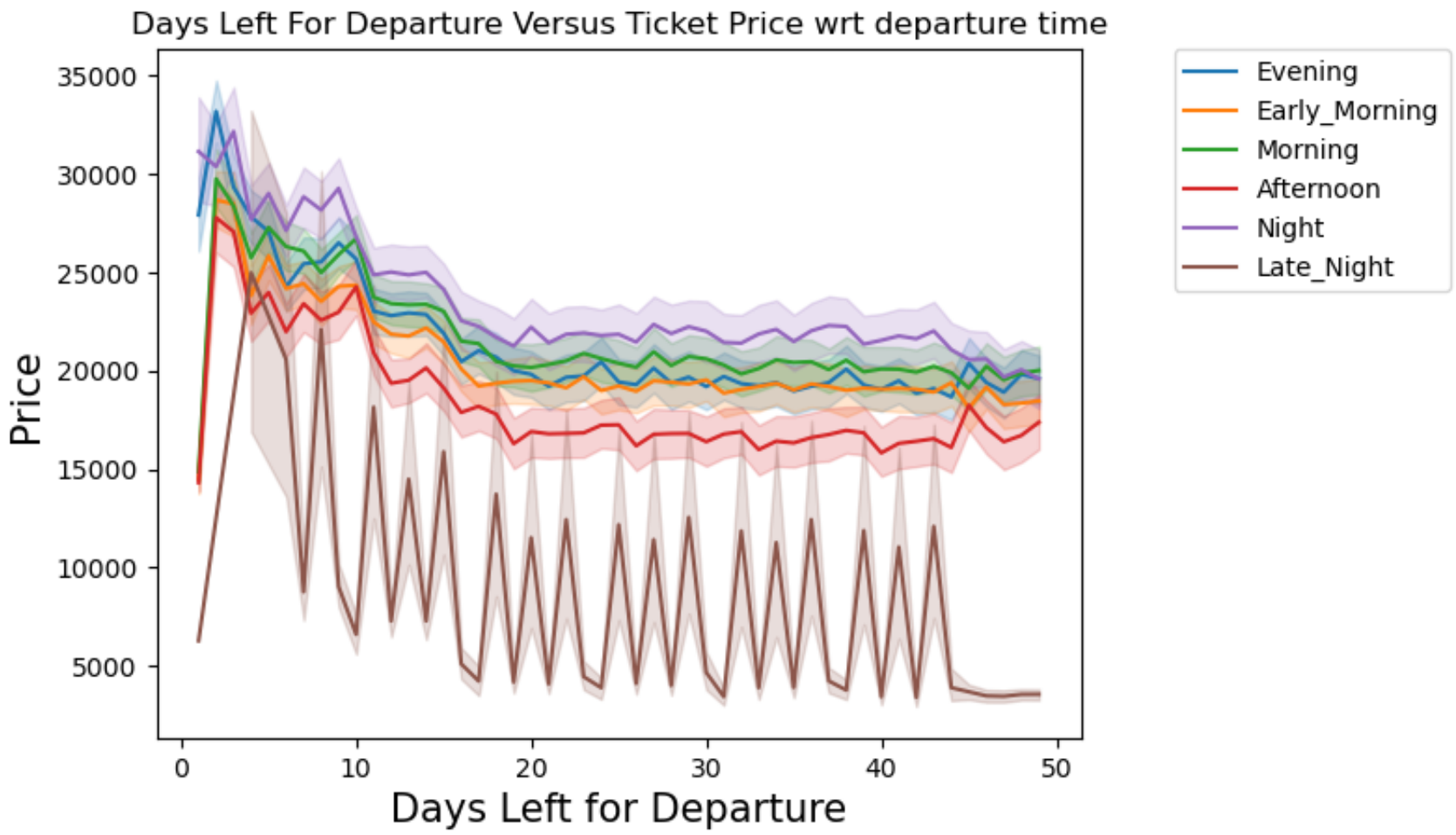
In [24]:
```
plt.figure(figsize=(20,8))
sns.lineplot(data=df,x='days_left',y='price',color='blue')
plt.title('Days Left For Departure Versus Ticket Price',fontsize=20)
plt.xlabel('Days Left for Departure',fontsize=15)
plt.ylabel('Price',fontsize=15)
plt.show()
```



In [25]:
```
plt.figure(figsize=(20,8))
sns.lineplot(data=df,x='days_left',y='price',color='blue',hue='airline',palette='hls')
plt.title('Days Left For Departure Versus Ticket Price of each Airline',fontsize=15)
plt.legend(fontsize=12)
plt.xlabel('Days Left for Departure',fontsize=15)
plt.ylabel('Price',fontsize=15)
plt.show()
```



In [26]:
```
sns.lineplot(data=df, x='days_left', y='price', hue = 'departure_time')
plt.title('Days Left For Departure Versus Ticket Price wrt departure time')
plt.xlabel('Days Left for Departure', fontsize=15)
plt.ylabel('Price', fontsize=15)
plt.legend(bbox_to_anchor=(1.4, 1), loc='best', borderaxespad=0)
plt.show()
```

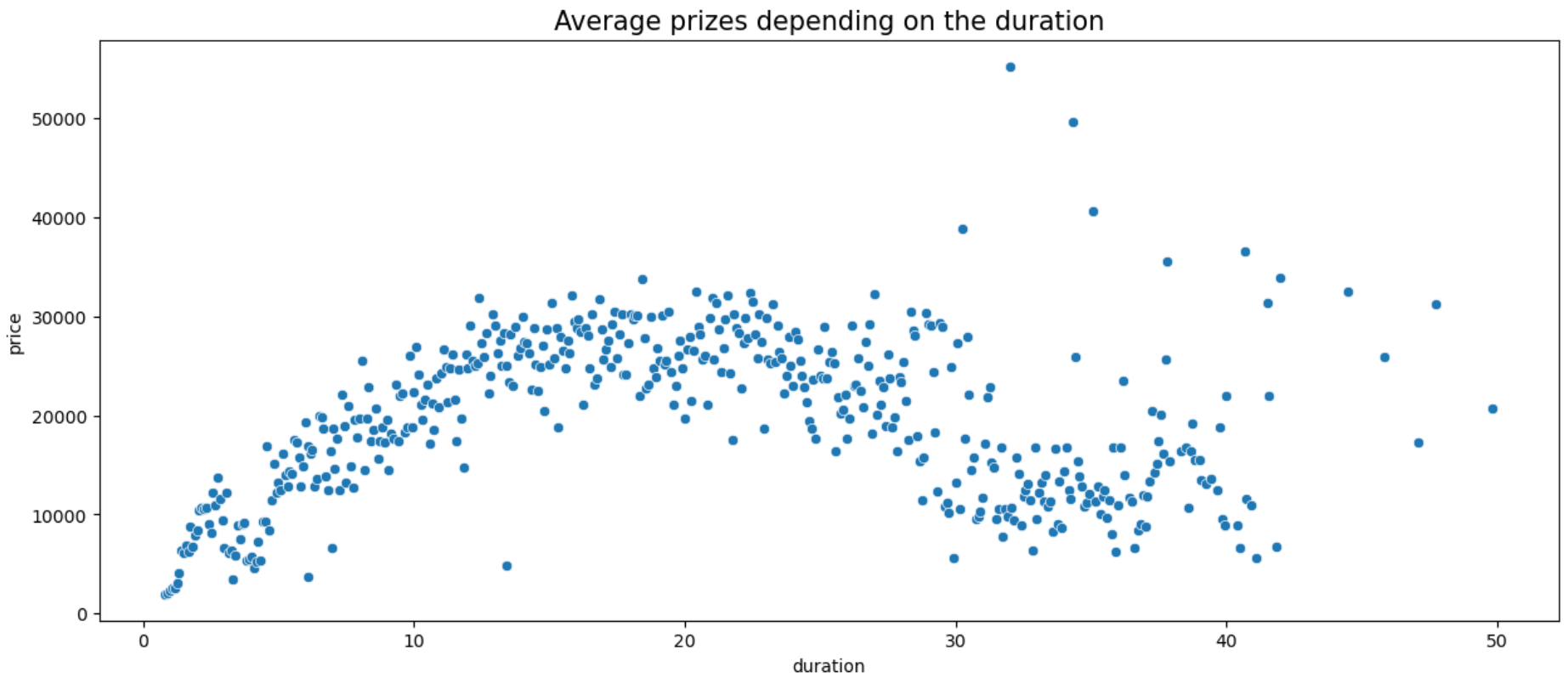


Prices for flights are highest when there are only 1-3 days left for departure, but decrease as the days left for departure increase. Air India and Vistara are the most expensive airlines, and prices decrease as the days left for departure increase. Late night departure times have lower prices compared to other departure times, but prices for late night arrival times are higher than evening arrivals.

Does the price change with the duration of the flight?

In [27]:
```
df_temp = df.groupby(['duration'])['price'].mean().reset_index()

plt.figure(figsize=(15,6))
ax = sns.scatterplot(x="duration", y="price", data=df_temp).set_title("Average prizes depending on the d
```



In [28]:
```
df_bk=df.copy()
```

In [29]:
```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
for col in df.columns:
    if df[col].dtype=='object':
        df[col]=le.fit_transform(df[col])
```

In [30]:
```
x=df.drop(['price'],axis=1)
y=df['price']
```

In [31]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.30,random_state=42)
x_train.shape,x_test.shape,y_train.shape,y_test.shape
```

Out[31]:
```
((210107, 10), (90046, 10), (210107,), (90046,))
```

In [32]:
```
from sklearn.preprocessing import MinMaxScaler
mmscaler=MinMaxScaler(feature_range=(0,1))
x_train=mmscaler.fit_transform(x_train)
x_test=mmscaler.fit_transform(x_test)
x_train=pd.DataFrame(x_train)
x_test=pd.DataFrame(x_test)
```

In [33]:
```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train, y_train)
y_pred = lr.predict(x_test)
```

In [34]:
```
from sklearn import metrics
print('Mean Absolute Error (MAE):', round(metrics.mean_absolute_error(y_test, y_pred),3))
print('Mean Squared Error (MSE):', round(metrics.mean_squared_error(y_test, y_pred),3))
print('Root Mean Squared Error (RMSE):', round(np.sqrt(metrics.mean_squared_error(y_test, y_pred)),3))
print('R2_score:', round(metrics.r2_score(y_test, y_pred),6))
```

```
Mean Absolute Error (MAE): 4630.296
Mean Squared Error (MSE): 49070241.265
Root Mean Squared Error (RMSE): 7005.015
R2_score: 0.904656
```

In [35]:
```
lr.fit(x_train, y_train)
y_pred = lr.predict(x_test)
```

In [36]:
```
out=pd.DataFrame({'Price_actual':y_test,'Price_pred':y_pred})
result=df_bk.merge(out,left_index=True,right_index=True)
```