In [1]:
```
import pandas as pd
import numpy as np
import matplotlib as mpl
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import statsmodels.api as sm
sns.set_style("darkgrid")
mpl.rcParams['figure.figsize']=(20,5)
```

In [2]:
```
flight_predictions=pd.read_csv("Clean_Dataset.csv.zip")

flight_predictions
```

Out[2]:

	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
...	...	...	...	...	...	...	...	...	...	...	...	...
300148	300148	Vistara	UK-822	Chennai	Morning	one	Evening	Hyderabad	Business	10.08	49	69265
300149	300149	Vistara	UK-826	Chennai	Afternoon	one	Night	Hyderabad	Business	10.42	49	77105
300150	300150	Vistara	UK-832	Chennai	Early_Morning	one	Night	Hyderabad	Business	13.83	49	79099
300151	300151	Vistara	UK-828	Chennai	Early_Morning	one	Evening	Hyderabad	Business	10.00	49	81585
300152	300152	Vistara	UK-822	Chennai	Morning	one	Evening	Hyderabad	Business	10.08	49	81585

300153 rows × 12 columns

In [3]:
```
flight_predictions.describe()
```

Out[3]:

	Unnamed: 0	duration	days_left	price
count	300153.000000	300153.000000	300153.000000	300153.000000
mean	150076.000000	12.221021	26.004751	20889.660523
std	86646.852011	7.191997	13.561004	22697.767366
min	0.000000	0.830000	1.000000	1105.000000
25%	75038.000000	6.830000	15.000000	4783.000000
50%	150076.000000	11.250000	26.000000	7425.000000
75%	225114.000000	16.170000	38.000000	42521.000000
max	300152.000000	49.830000	49.000000	123071.000000

In [4]:
```
flight_predictions.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]:
```
flight_predictions.isnull().sum()
```

Out[5]:
```
Unnamed: 0          0
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 [6]:
```
flight_predictions.shape
```

Out[6]:
```
(300153, 12)
```

In [7]:
```
flight_predictions.drop("Unnamed: 0",axis="columns",inplace=True)
```

In [8]:
```
flight_predictions.rename(columns={"class":"seat_class","duration":"flight_duration","source_city":"origin_city"},inpla
```

In [9]:
```
flight_predictions.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   origin_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   seat_class        300153 non-null  object
 8   flight_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 [10]:
```
flight_predictions.isnull().sum()
```

Out[10]:
```
airline             0
flight              0
origin_city         0
departure_time      0
stops               0
arrival_time        0
destination_city    0
seat_class          0
flight_duration     0
days_left           0
price               0
dtype: int64
```

In [11]:
```
flight_predictions.dtypes
```

Out[11]:
```
airline              object
flight               object
origin_city          object
departure_time       object
stops                object
arrival_time         object
destination_city     object
seat_class           object
flight_duration     float64
days_left             int64
price                 int64
dtype: object
```

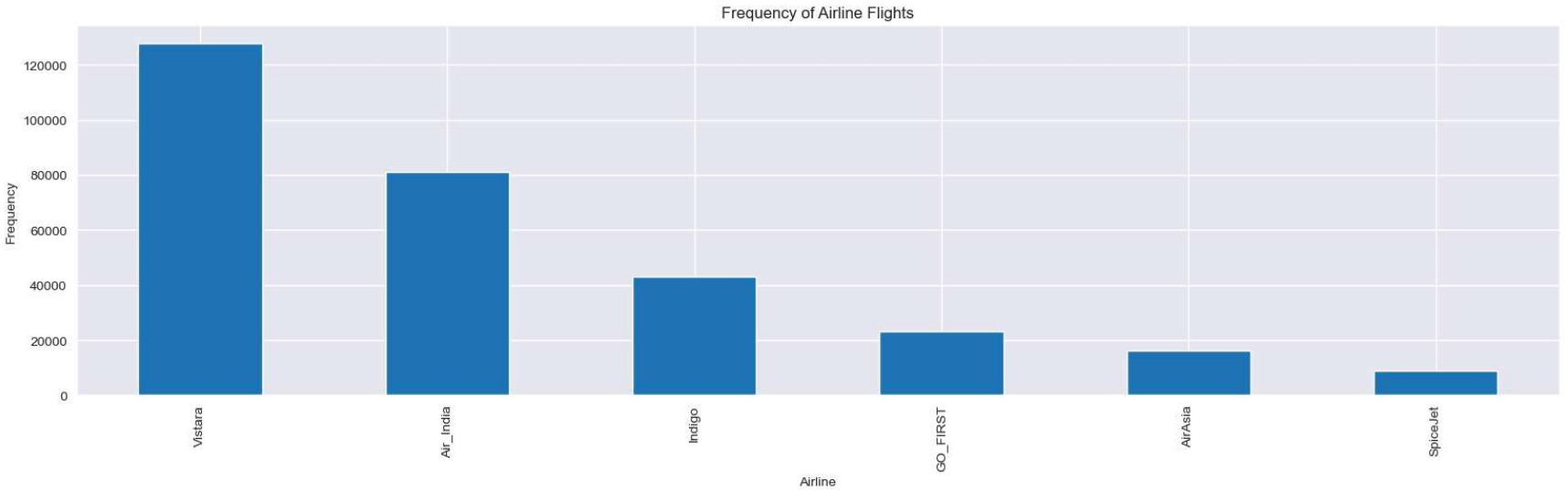
This dataset is clean with no missing data. I removed the "Unnamed:0" column because it did not contain any relevant information for this analysis. I also updated the following column names so that they would be easier to reference: class to seat_class; duration to flight_duration; source_city to origin_city.

In [12]:
```
flight_predictions["airline"].value_counts()
```

Out[12]:
```
airline
Vistara      127859
Air_India     80892
Indigo        43120
GO_FIRST      23173
AirAsia       16098
SpiceJet       9011
Name: count, dtype: int64
```

In [13]:
```
flight_predictions["airline"].value_counts().plot(kind="bar")
plt.xlabel("Airline")
```

```
plt.ylabel("Frequency")
plt.title("Frequency of Airline Flights")
plt.savefig("Airlineflightfrequency.jpg")
plt.show()
```



In [14]:
```
flight_predictions["airline"].value_counts(normalize=True).round(2)
```

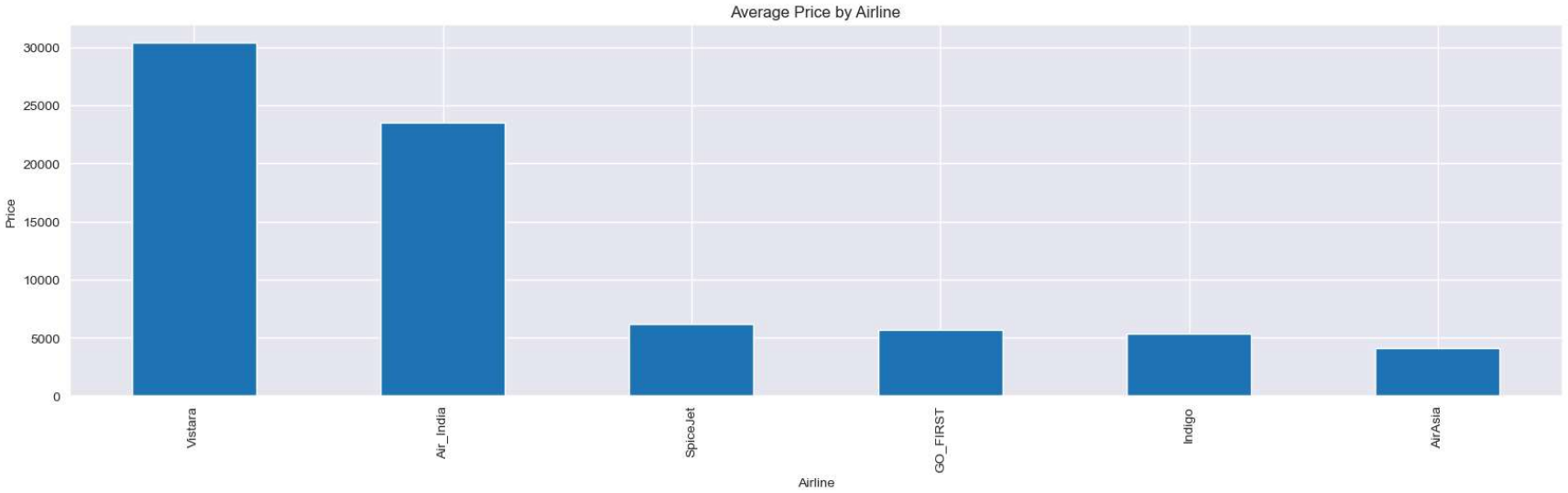
Out[14]:
```
airline
Vistara      0.43
Air_India    0.27
Indigo       0.14
GO_FIRST     0.08
AirAsia      0.05
SpiceJet     0.03
Name: proportion, dtype: float64
```

The three airlines with the highest frequency of flights are Vistara at 43% (127859 flights), followed by Air India at 27%(80892 flights), and lastly Indigo at 14% (43120 flights)

In [15]:
```
avg_price_by_airline=flight_predictions.groupby("airline")["price"].mean().sort_values(ascending=False).round(2)
avg_price_by_airline
```

Out[15]:
```
airline
Vistara      30396.54
Air_India    23507.02
SpiceJet      6179.28
GO_FIRST      5652.01
Indigo        5324.22
AirAsia       4091.07
Name: price, dtype: float64
```

In [16]:
```
avg_price_by_airline.plot(kind="bar")
plt.title("Average Price by Airline")
plt.xlabel("Airline")
plt.ylabel("Price")
plt.savefig("AveragePricebyairline.jpg")
plt.show()
```



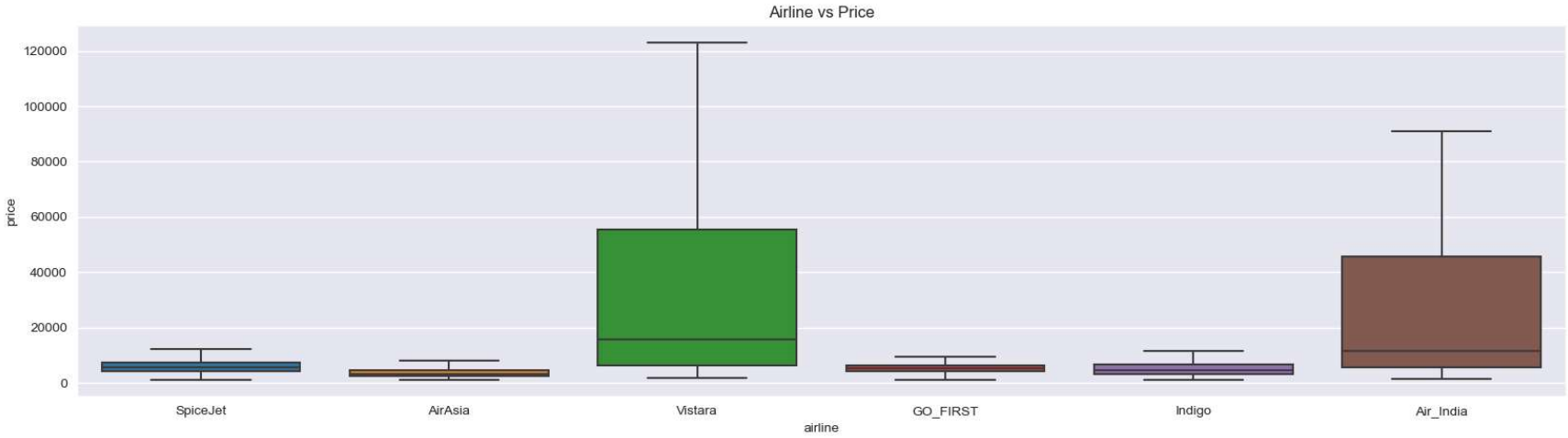
In [17]:
```
max_price_per_airline=flight_predictions.groupby("airline")["price"].max().sort_values(ascending=False)
max_price_per_airline
```

Out[17]:
```
airline
Vistara      123071
Air_India     90970
SpiceJet      34158
GO_FIRST      32803
Indigo        31952
AirAsia       31917
Name: price, dtype: int64
```

In [18]:
```
min_price_per_airline=flight_predictions.groupby("airline")["price"].min().sort_values(ascending=False)
min_price_per_airline
```

Out[18]:
```
airline
Vistara      1714
Air_India    1526
SpiceJet     1106
AirAsia      1105
GO_FIRST     1105
Indigo       1105
Name: price, dtype: int64
```

In [19]:
```
sns.boxplot(x="airline",y="price",data=flight_predictions,sym="")
plt.title("Airline vs Price" )
plt.savefig("Airlineboxplot.jpg")
plt.show()
```

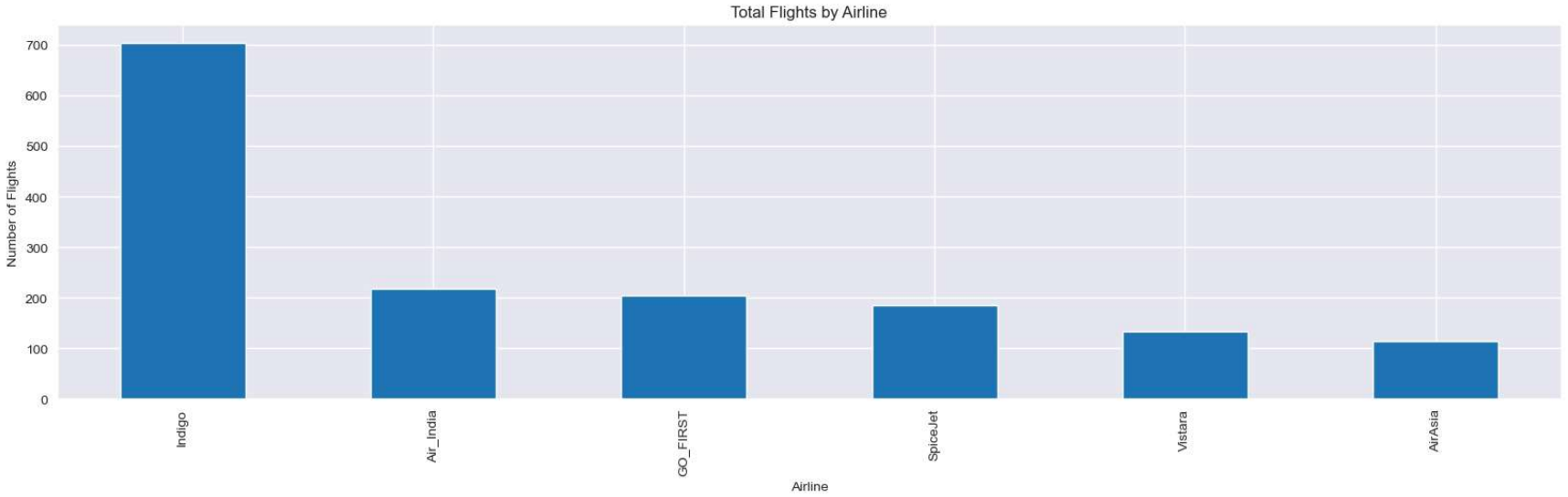


After reviewing the airlines by price, it was interesting to discover that although Indigo had the third highest frequency of flights,it had the second lowest average price by airline at 5324.22.

In [20]:
```
total_flights=flight_predictions.groupby(["airline","flight"], as_index=False).count()
total_flights_per_airline=total_flights.airline.value_counts(ascending=False)
total_flights_per_airline
```

Out[20]:
```
airline
Indigo       704
Air_India    218
GO_FIRST     205
SpiceJet     186
Vistara      133
AirAsia      115
Name: count, dtype: int64
```

In [21]:
```
total_flights_per_airline.plot(kind="bar")
plt.title("Total Flights by Airline")
plt.xlabel("Airline")
plt.ylabel("Number of Flights")
plt.savefig("TotalFlightsperairline.jpg")
plt.show()
```



Reviewing total flights per airline shows that the three airlines with the highest number of total flights are Indigo 704 flights, followed by Air_India with 218 flights, and GO_FIRST with 205 flights.

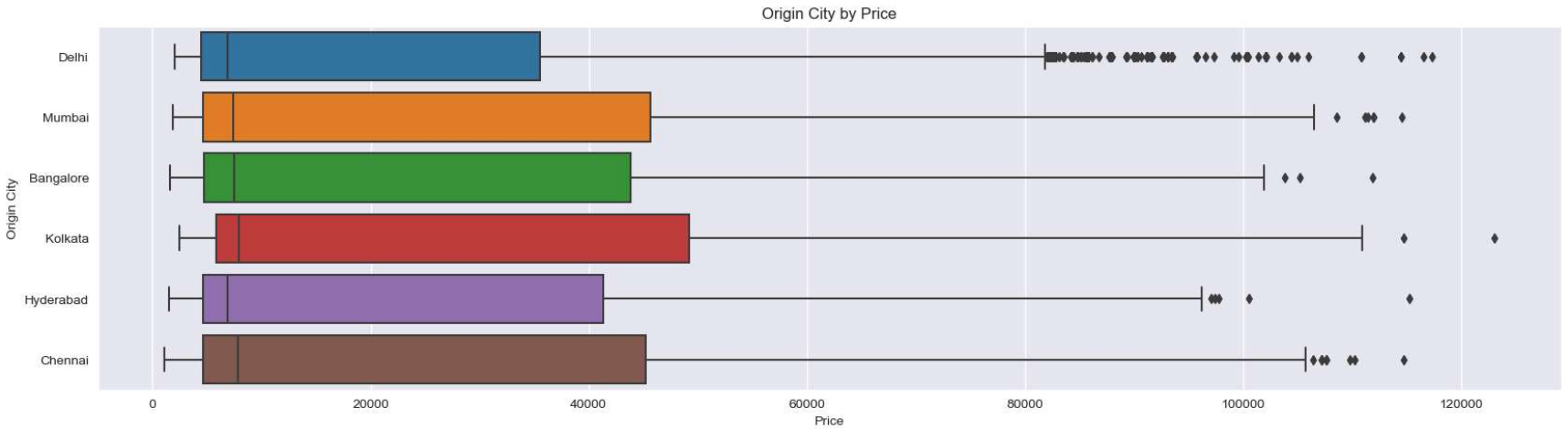
In [22]:
```
total_flights_by_origin_city=flight_predictions.groupby("origin_city")["flight"].count()
total_flights_by_origin_city
```

Out[22]:
```
origin_city
Bangalore    52061
Chennai      38700
Delhi        61343
Hyderabad    40806
Kolkata      46347
Mumbai       60896
Name: flight, dtype: int64
```

In [23]:
```
total_flights_by_destination_city=flight_predictions.groupby("destination_city")["flight"].count()
total_flights_by_destination_city
```

Out[23]:
```
destination_city
Bangalore    51068
Chennai      40368
Delhi        57360
Hyderabad    42726
Kolkata      49534
Mumbai       59097
Name: flight, dtype: int64
```

In [24]:
```
plt.subplots()
sns.boxplot(data=flight_predictions,y="origin_city",x="price")
plt.xlabel("Price")
plt.ylabel("Origin City")
plt.title("Origin City by Price")
plt.savefig("Origincitybyprice.jpg")
plt.show()
```



In [25]:
```
avg_price_by_origin_city=flight_predictions.groupby("origin_city")["price"].mean().round(2).sort_values()
avg_price_by_origin_city
```

Out[25]:
```
origin_city
Delhi        18951.33
Hyderabad    20155.62
Bangalore    21469.46
Mumbai       21483.82
Kolkata      21746.24
Chennai      21995.34
Name: price, dtype: float64
```

In [26]:
```
plt.subplots()
sns.boxplot(data=flight_predictions,y="destination_city",x="price")
plt.xlabel("Price")
plt.ylabel("Destination City")
plt.title("Destination City by Price")
plt.savefig("Desticitybyprice.jpg")
plt.show()
```

In [27]:
```
avg_price_by_destination_city=flight_predictions.groupby("destination_city")["price"].mean().round(2).sort_values()
avg_price_by_destination_city
```

Out[27]:
```
destination_city
Delhi        18436.77
Hyderabad    20427.66
Mumbai       21372.53
Bangalore    21593.96
Chennai      21953.32
Kolkata      21959.56
Name: price, dtype: float64
```

A comparision of origin city and destination city by price shows that Delhi is the most affordable to fly in/out of. Both boxplots also show that Delhi has the most outliers of all the cities.

In [28]:
```
total_stops=flight_predictions["stops"].value_counts().reset_index()
total_stops
```

Out[28]:

	stops	count
0	one	250863
1	zero	36004
2	two_or_more	13286

In [29]:
```
flight_predictions["stops"].value_counts(normalize=True).round(2)
```

Out[29]:
```
stops
one            0.84
zero           0.12
two_or_more    0.04
Name: proportion, dtype: float64
```

In [30]:
```
sns.boxplot(x="stops",y="price",data=flight_predictions,sym="",whis=[5,95])
plt.xlabel("Stops")
plt.ylabel("Price")
plt.title("Total Number of Stops vs Price")
plt.savefig("Stopsvsprice.jpg")
plt.show()
```

In [31]:
```
avg_price_per_stop= flight_predictions.groupby("stops")["price"].mean().round(2)
avg_price_per_stop
```

Out[31]:
```
stops
one            22900.99
two_or_more    14113.45
zero            9375.94
Name: price, dtype: float64
```

In [32]:
```
agg_price_per_stop=flight_predictions.groupby("stops")["price"].agg([min,max])
agg_price_per_stop
```

Out[32]:

stops	min	max
one	1105	123071
two_or_more	1966	117307
zero	1105	59573

Over 84% of the flights have one stop followed by no/zero stops with 12% and two or more stops at 4%. An indirect flight with one stop has the greatest spread in data whereas zero/no stops and two or more both have a smaller spread. I found it interesting to discover that a flight with no stops is more affordable on average than both one stop and two or more flights.

In [33]:
```
total_seat_class=flight_predictions["seat_class"].value_counts()
```

```
total_seat_class
```

Out[33]:
```
seat_class
Economy     206666
Business     93487
Name: count, dtype: int64
```

In [34]:
```
sns.countplot(x="seat_class",data=flight_predictions)
plt.xlabel("Seat Class")
plt.ylabel("Count")
plt.title("Number of Tickets Purchased by Seat Class")
plt.savefig("Ticketsbyseatclass.jpg")
plt.show()
```

In [35]:
```
total_seat_class_percentage=flight_predictions["seat_class"].value_counts(normalize=True).round(2)
total_seat_class_percentage
```

Out[35]:
```
seat_class
Economy     0.69
Business    0.31
Name: proportion, dtype: float64
```

In [36]:
```
sns.boxplot(x="seat_class",y="price",data=flight_predictions,sym="",whis=[5,95])
plt.xlabel("Seat Class")
plt.ylabel("Price")
plt.title("Seat Class vs Price")
plt.savefig("Seatclassvsprice.jpg")
plt.show()
```

In [37]:
```
avg_price_per_seat_class=flight_predictions.groupby("seat_class")["price"].mean().round(2)
avg_price_per_seat_class
```

Out[37]:
```
seat_class
Business    52540.08
Economy      6572.34
Name: price, dtype: float64
```

In [38]:
```
agg_price_per_seat_class=flight_predictions.groupby("seat_class")["price"].agg([min,max]).round(2)
agg_price_per_seat_class
```

Out[38]:

seat_class	min	max
Business	12000	123071
Economy	1105	42349

69% of passengers book flights in the Economy class and 31% book Business class seats. The average price of a Business class ticket(52540.08) is 8 times higher than an Economy class ticket(6572.34).

In [39]:
```
avg_price_departure_time=flight_predictions.groupby("departure_time")["price"].mean().round(2).reset_index()
avg_price_departure_time.sort_values(by="price")
```

Out[39]:

	departure_time	price
3	Late_Night	9295.30
0	Afternoon	18179.20
1	Early_Morning	20370.68
2	Evening	21232.36
4	Morning	21630.76
5	Night	23062.15

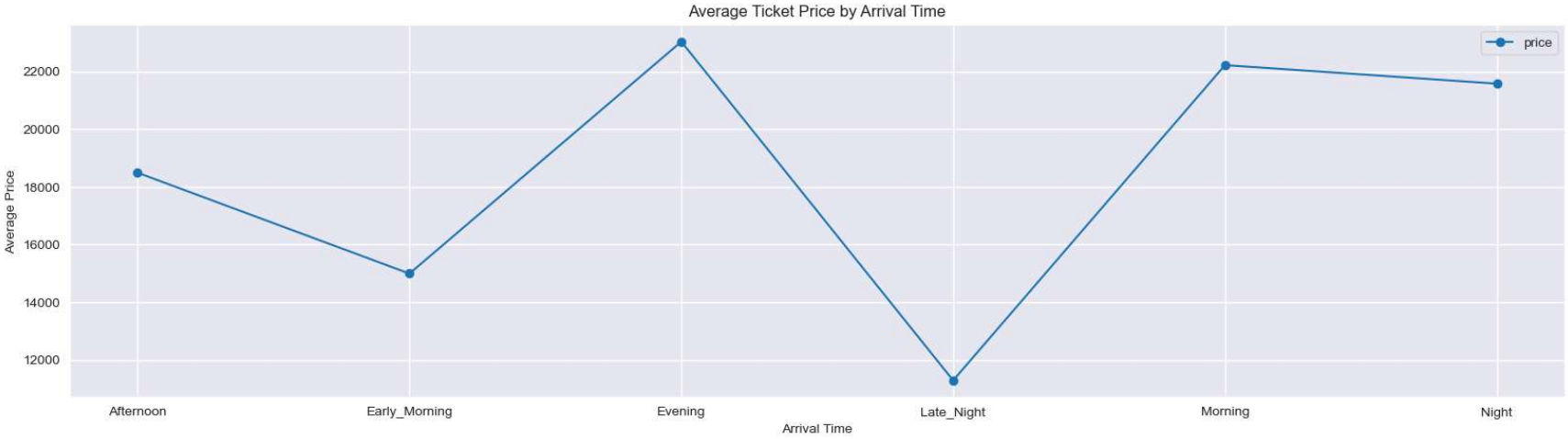
In [40]:
```
avg_price_departure_time.plot(x="departure_time",y="price",kind="line",marker="o")
plt.xlabel("Departure Time")
plt.ylabel("Average Price")
plt.title("Average Ticket Price by Departure Time")
plt.savefig("Avgpricebydepartureтime.jpg")
plt.show()
```

In [41]:
```
avg_price_arrival_time=flight_predictions.groupby("arrival_time")["price"].mean().round(2).reset_index()
avg_price_arrival_time.sort_values(by="price")
```

Out[41]:

	arrival_time	price
3	Late_Night	11284.91
1	Early_Morning	14993.14
0	Afternoon	18494.60
5	Night	21586.76
4	Morning	22231.08
2	Evening	23044.37

In [42]:
```
avg_price_arrival_time.plot(x="arrival_time",y="price",kind="line",marker="o")
plt.xlabel("Arrival Time")
plt.ylabel("Average Price")
plt.title("Average Ticket Price by Arrival Time")
plt.savefig("Avgpricebyarrivaltime.jpg")
plt.show()
```

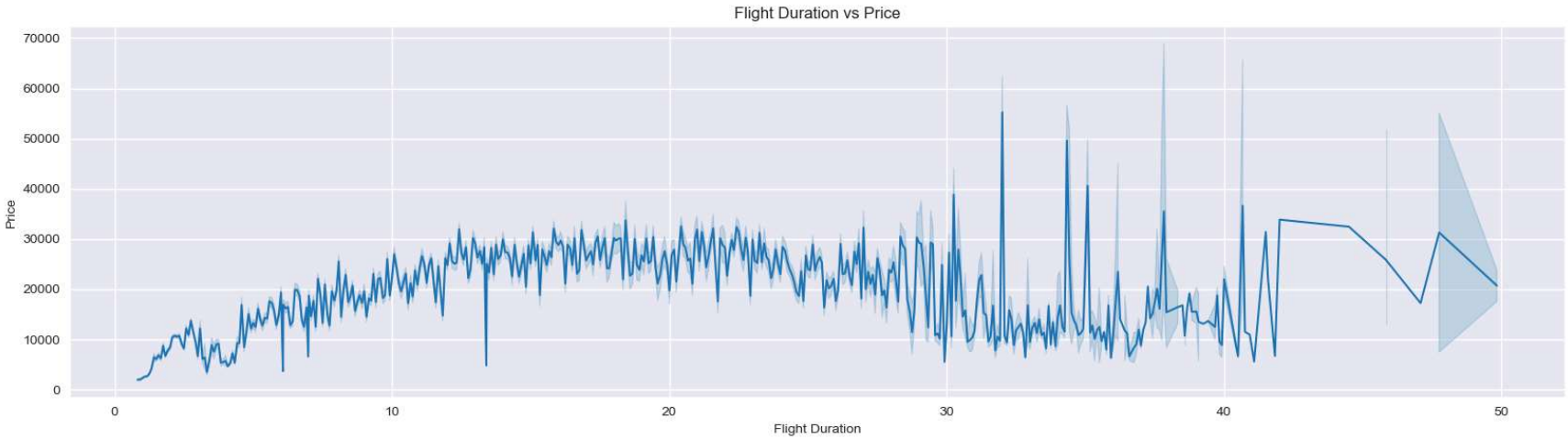


Late night departures and arrivals have the least expensive ticket prices out of all categories. The second most affordable are afternoon departures and early morning arrivals.

In [43]:
```
total_flight_duration=len(flight_predictions["flight_duration"].value_counts())
total_flight_duration
```

Out[43]:
476

In [44]:
```
sns.lineplot(data=flight_predictions,x="flight_duration",y="price")
plt.xlabel("Flight Duration")
plt.ylabel("Price")
plt.title("Flight Duration vs Price")
plt.show()
```

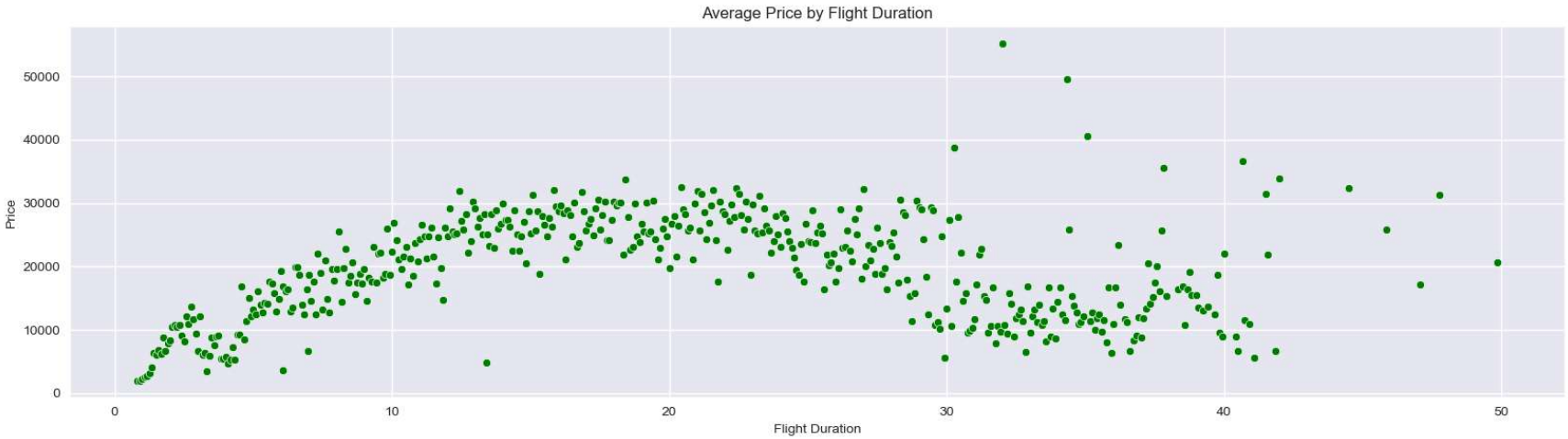


In [45]:
```
avg_duration_price=flight_predictions.groupby("flight_duration")["price"].mean().round(2).reset_index()
avg_duration_price.head(10)
```

Out[45]:

	flight_duration	price
0	0.83	1973.56
1	0.92	2003.54
2	1.00	2266.06
3	1.08	2589.31
4	1.17	2632.21
5	1.25	3137.86
6	1.33	4140.02
7	1.42	6433.74
8	1.50	6059.56
9	1.58	6823.26

In [56]:
```
sns.scatterplot(x="flight_duration",y="price",data=avg_duration_price,color="green")
plt.xlabel("Flight Duration")
plt.ylabel("Price")
plt.title("Average Price by Flight Duration")
plt.savefig("Average Price by Flight Duration.jpg")
plt.show()
```



The line graph shows that the ticket price increases the longer the duration of the flight. A scatter plot of average price verus flight duration shows that there does not seem to be a linear relationship between the data. There are also are outliers in the data. The average ticket prices reach the highest price at roughly a 20 hour flight duration before lowering.

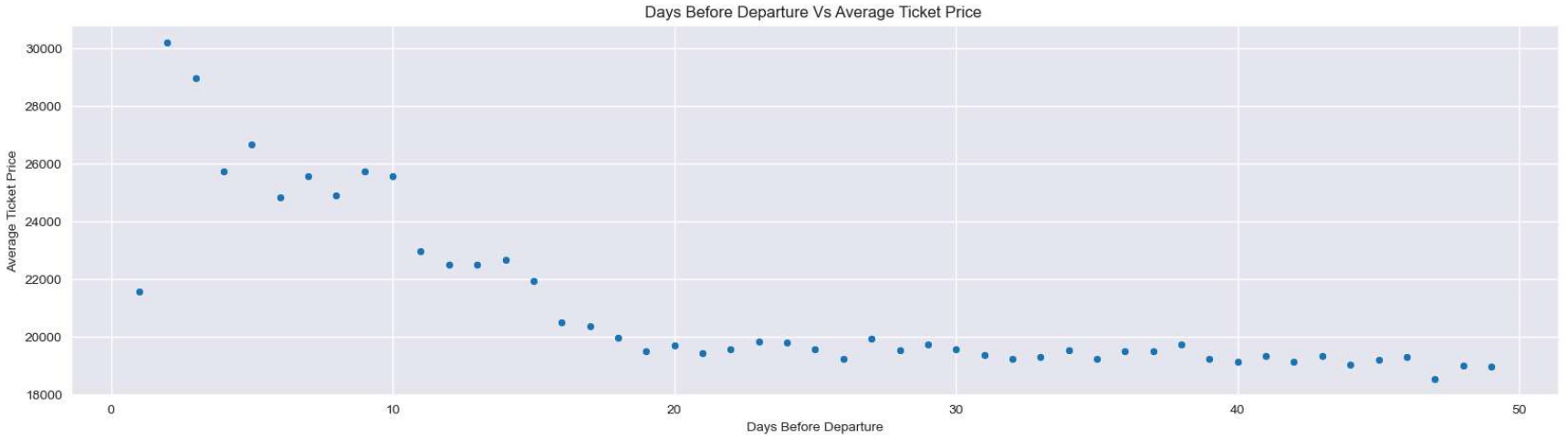
In [47]:
```
avg_price_days_left=flight_predictions.groupby("days_left")["price"].mean().round(2).reset_index().sort_values(by="days_
avg_price_days_left.head(10)
```

Out[47]:

	days_left	price
48	49	18992.97
47	48	18998.13
46	47	18553.27
45	46	19305.35
44	45	19199.88
43	44	19049.08
42	43	19340.53
41	42	19154.26
40	41	19347.44
39	40	19144.97

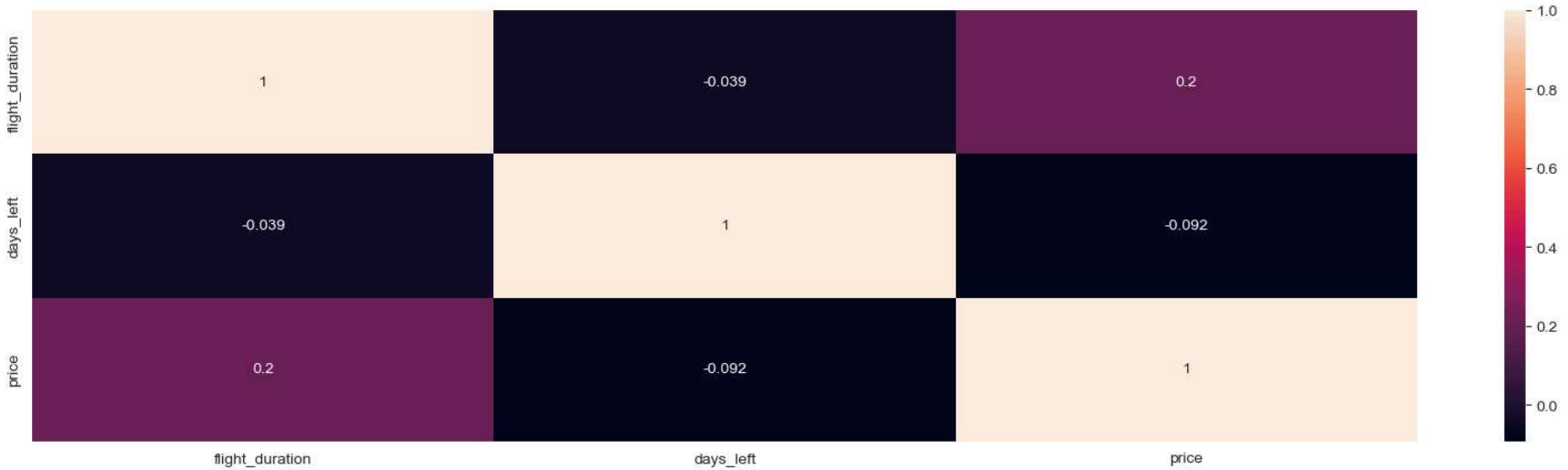
In [48]:

```
sns.scatterplot(x="days_left",y="price",data=avg_price_days_left)
plt.xlabel("Days Before Departure")
plt.ylabel("Average Ticket Price")
plt.title("Days Before Departure Vs Average Ticket Price ")
plt.savefig("Daystildepartprice.jpg")
plt.show()
```



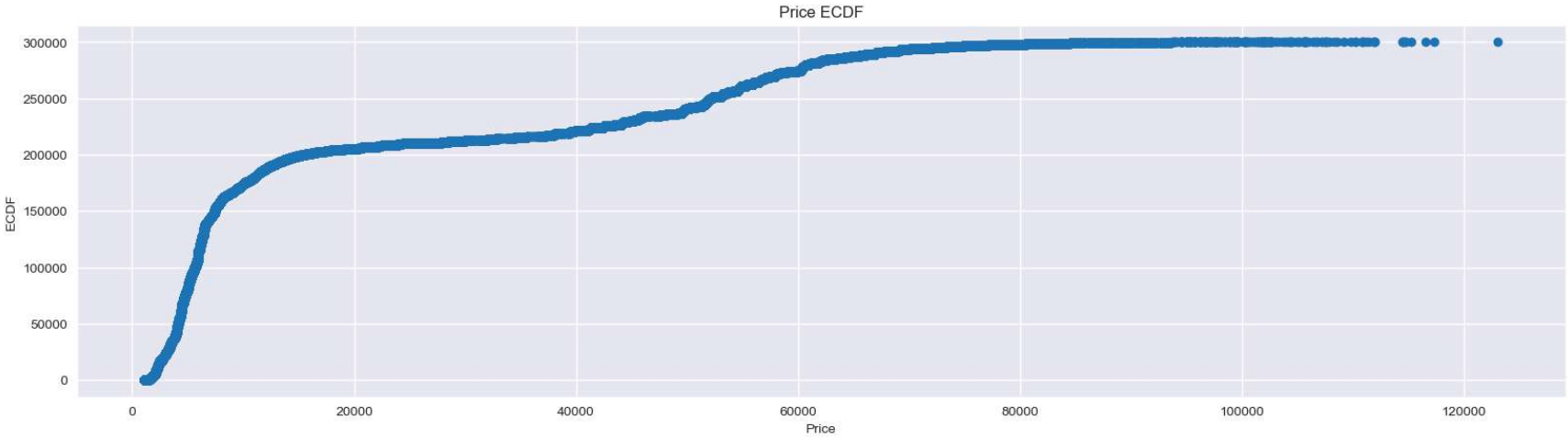
The data in the scatter plot above shows that days before flight departure and average ticket price have a negative linear correlation with minimal outliers. The closer that a ticket is purchased prior to departure, the more expensive it is.

In [49]:
```
correlated=flight_predictions[["flight_duration","days_left","price"]].corr()
sns.heatmap(correlated,annot=True)
plt.savefig("correlationheatmap.jpg")
plt.show()
```



There does not seem to be a correlation between flight_duration,days_left,and price.

In [50]:
```
x=np.sort(flight_predictions["price"])
y=np.arange(1,len(x)+1/len(x))
plt.plot(x,y,marker="o",linestyle="none")
plt.xlabel("Price")
plt.ylabel("ECDF")
plt.title("Price ECDF")
plt.show()
```



In [51]:
```
flight_predictions = pd.read_csv("Clean_Dataset.csv.zip")
independent_variables = flight_predictions[["days_left", "duration"]]
independent_variables['fp_airlines'] = flight_predictions['airline'].replace({"Vistara": 1,"Air_India": 2,"Indigo": 3,"
independent_variables['fp_stops'] = flight_predictions['stops'].replace({"one": 1,"zero": 2,"two_or_more": 3})
independent_variables = sm.add_constant(independent_variables)
dependent_variable = flight_predictions["price"]
regression_model = sm.OLS(dependent_variable, independent_variables).fit()
print(regression_model.summary())
```

```
C:\Users\sarah\AppData\Local\Temp\ipykernel_38892\1046952287.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a
-view-versus-a-copy
  independent_variables['fp_airlines'] = flight_predictions['airline'].replace({"Vistara": 1,"Air_India": 2,"Indigo":
3,"GO_FIRST": 4,"AirAsia": 5,"SpiceJet": 6})
C:\Users\sarah\AppData\Local\Temp\ipykernel_38892\1046952287.py:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a
-view-versus-a-copy
  independent_variables['fp_stops'] = flight_predictions['stops'].replace({"one": 1,"zero": 2,"two_or_more": 3})
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  price   R-squared:                       0.213
Model:                            OLS   Adj. R-squared:                  0.213
Method:                 Least Squares   F-statistic:                 2.036e+04
Date:                Wed, 24 Apr 2024   Prob (F-statistic):               0.00
Time:                        17:55:07   Log-Likelihood:            -3.4004e+06
No. Observations:              300153   AIC:                         6.801e+06
Df Residuals:                  300148   BIC:                         6.801e+06
Df Model:                           4
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const        4.273e+04    167.052    255.767      0.000    4.24e+04    4.31e+04
days_left    -139.4627      2.713    -51.410      0.000    -144.780    -134.146
duration      224.5483      5.442     41.265      0.000     213.883     235.214
fp_airlines -6699.1389     28.151   -237.973      0.000   -6754.314   -6643.964
fp_stops    -5403.8103     75.561    -71.516      0.000   -5551.908   -5255.713
==============================================================================
Omnibus:                    28688.179   Durbin-Watson:                   0.147
Prob(Omnibus):                  0.000   Jarque-Bera (JB):            25328.627
Skew:                           0.637   Prob(JB):                         0.00
Kurtosis:                       2.366   Cond. No.                         150.
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

The regression model shows that there is not a strong fit with the predictor variable, price, due to the R Squared Value of 21%.

In [52]:
```
regression_model.params.sort_values().plot(kind="bar")
plt.title("Regression Coefficients for Flight Predictions: Price")
plt.savefig("Regressionbarchart.jpg")
plt.show()
```

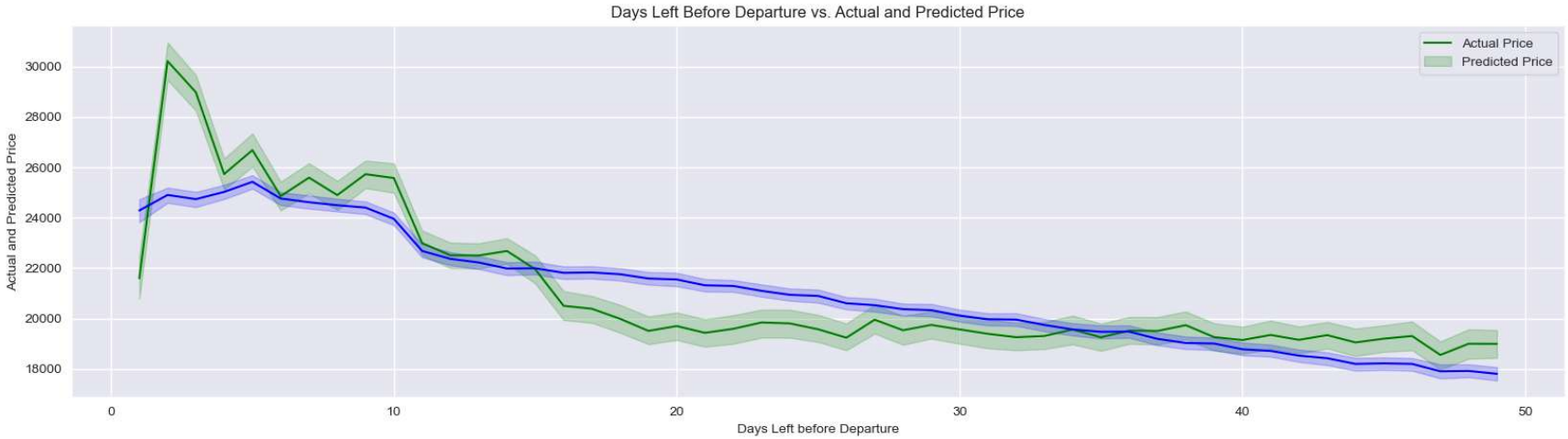
In [53]:
```
flight_predictions["prediction"]=pd.DataFrame(regression_model.predict(independent_variables))
flight_predictions.head(10)
```

Out[53]:

	Unnamed: 0	airline	flight	source_city	departure_time	stops	arrival_time	destination_city	class	duration	days_left	price	pre
0	0	SpiceJet	SG-8709	Delhi	Evening	zero	Night	Mumbai	Economy	2.17	1	5953	-7928
1	1	SpiceJet	SG-8157	Delhi	Early_Morning	zero	Morning	Mumbai	Economy	2.33	1	5953	-7892
2	2	AirAsia	I5-764	Delhi	Early_Morning	zero	Early_Morning	Mumbai	Economy	2.17	1	5956	-1229
3	3	Vistara	UK-995	Delhi	Morning	zero	Afternoon	Mumbai	Economy	2.25	1	5955	25585
4	4	Vistara	UK-963	Delhi	Morning	zero	Morning	Mumbai	Economy	2.33	1	5955	25603
5	5	Vistara	UK-945	Delhi	Morning	zero	Afternoon	Mumbai	Economy	2.33	1	5955	25603
6	6	Vistara	UK-927	Delhi	Morning	zero	Morning	Mumbai	Economy	2.08	1	6060	25547
7	7	Vistara	UK-951	Delhi	Afternoon	zero	Evening	Mumbai	Economy	2.17	1	6060	25567
8	8	GO_FIRST	G8-334	Delhi	Early_Morning	zero	Morning	Mumbai	Economy	2.17	1	5954	5469
9	9	GO_FIRST	G8-336	Delhi	Afternoon	zero	Evening	Mumbai	Economy	2.25	1	5954	5487

In [54]:

```
sns.lineplot(x="days_left",y="price",data=flight_predictions,color="green")
sns.lineplot(x="days_left",y="prediction",data=flight_predictions,color="blue")
plt.xlabel("Days Left before Departure")
plt.ylabel("Actual and Predicted Price")
plt.title("Days Left Before Departure vs. Actual and Predicted Price")
plt.legend(labels=["Actual Price","Predicted Price"])
plt.savefig("predictionmodel.jpg")
plt.show()
```



Due to the R Squared variable being so low, the prediction model above is not a success and should not be used. The recommendation is for Easemytrip to provide a another dataset that contains a longer timespan if they still want a prediction model for their customers.

In [ ]: