SOURCE CODE

Importing all the Libraries

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

Importing Data for ploting

[6]:
```
data = pd.read_csv(r"D:\FIFTHFORCE\Filtered_Flight_Data.csv")
data.head(10)
```

```
C:\Users\ashmi\AppData\Local\Temp\ipykernel_11732\2894534669.py:1: DtypeWarning: Columns (10) have mixed types. Specify dtype option on import or set lo
w_memory=False.
  data = pd.read_csv(r"D:\FIFTHFORCE\Filtered_Flight_Data.csv")
```

[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
5	Vistara	UK-945	Delhi	Morning	zero	Afternoon	Mumbai	Economy	2.33	1	5955
6	Vistara	UK-927	Delhi	Morning	zero	Morning	Mumbai	Economy	2.08	1	6060
7	Vistara	UK-951	Delhi	Afternoon	zero	Evening	Mumbai	Economy	2.17	1	6060
8	GO_FIRST	G8-334	Delhi	Early_Morning	zero	Morning	Mumbai	Economy	2.17	1	5954
9	GO_FIRST	G8-336	Delhi	Afternoon	zero	Evening	Mumbai	Economy	2.25	1	5954

Data Copy

[8]:
```
dataCopy = data
dataCopy.head(10)
```

[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
5	Vistara	UK-945	Delhi	Morning	zero	Afternoon	Mumbai	Economy	2.33	1	5955
6	Vistara	UK-927	Delhi	Morning	zero	Morning	Mumbai	Economy	2.08	1	6060
7	Vistara	UK-951	Delhi	Afternoon	zero	Evening	Mumbai	Economy	2.17	1	6060
8	GO_FIRST	G8-334	Delhi	Early_Morning	zero	Morning	Mumbai	Economy	2.17	1	5954
9	GO_FIRST	G8-336	Delhi	Afternoon	zero	Evening	Mumbai	Economy	2.25	1	5954

Analysis I

Ploting Data with sub-data and charts

Ploting Data with sub-data and charts

[337]:
```
# Percentage of Flights available vs airline...

flight_count = dataCopy.groupby(['airline'], as_index = False)['flight'].count()
flight_count.rename(columns = {'flight':'Count_Flights', 'airline':'Airline'}, inplace = True)
flight_count

# Total Flights...
total_flights = flight_count['Count_Flights'].sum()
flight_count['Percentage'] = ((flight_count['Count_Flights']/total_flights) * 100).round(2)
flight_count.sort_values(by = 'Count_Flights', ascending = False, inplace = True)
flight_count
```

[337]:

	Airline	Count_Flights	Percentage
5	Vistara	128727	42.74
1	Air_India	81060	26.91
3	Indigo	43128	14.32
2	GO_FIRST	23176	7.69
0	AirAsia	16100	5.35
4	SpiceJet	9015	2.99

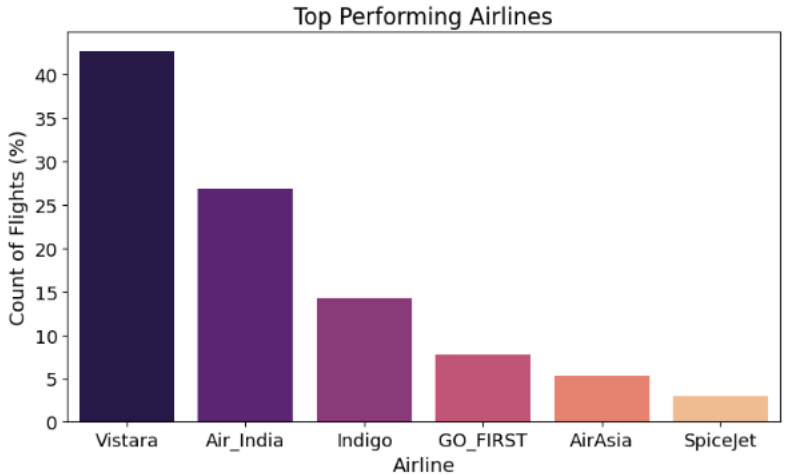
[338]:
```
# Percentage of Flights available vs City...

flight_count_city = dataCopy.groupby(['source_city'], as_index = False)['flight'].count()
flight_count_city.rename(columns = {'source_city':'City', 'flight':'Count_Flights'}, inplace = True)
flight_count_city

# Total Flights...
total_flights_city = flight_count_city['Count_Flights'].sum()
flight_count_city['Percentage'] = ((flight_count_city['Count_Flights']/total_flights_city) * 100).round(2)
flight_count_city.sort_values(by = 'Count_Flights', ascending = False, inplace = True)
flight_count_city
```

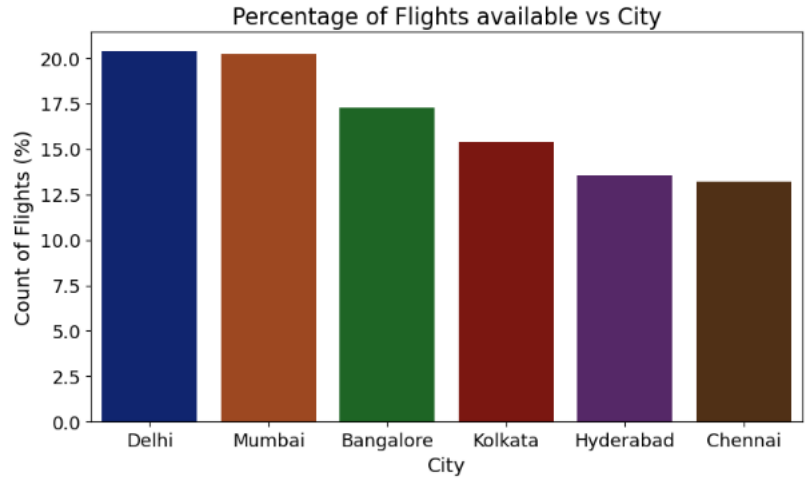
[338]:

	City	Count_Flights	Percentage
2	Delhi	61394	20.38
5	Mumbai	60896	20.22
0	Bangalore	52061	17.28
4	Kolkata	46347	15.39
3	Hyderabad	40806	13.55
1	Chennai	39702	13.18

[339]:

```
# Plotting the above data...
plot, axis = plt.subplots(nrows = 1, ncols = 2, figsize = (20, 5))

# Plotting barplot...
sns.barplot(x = "Airline", y = "Percentage", data = flight_count, ax = axis[0],
            palette = 'magma')
axis[0].set_xlabel('Airline', fontsize=14)
axis[0].set_ylabel('Count of Flights (%)', fontsize=14)
axis[0].tick_params(axis='both', labelsize=13)
# axis[0].set_xticklabels(axis[0].get_xticklabels(), rotation=90, fontsize=13)
axis[0].set_title('Top Performing Airlines', fontsize = 16)

sns.barplot(x = "City", y = "Percentage", data = flight_count_city, ax = axis[1],
            palette = 'dark')
axis[1].set_xlabel('City', fontsize=14)
axis[1].set_ylabel('Count of Flights (%)', fontsize=14)
axis[1].tick_params(axis='both', labelsize=13)
# axis[1].set_xticklabels(axis[1].get_xticklabels(), rotation=90, fontsize=13)
axis[1].set_title('Percentage of Flights available vs City', fontsize = 16)

plt.show()
```





Conclusion

- Vistara Flights are more in number than other Flights.
- Availability of Flights of Delhi and Mumbai are more in number.

[64]:
```
dataCopy.head(10)
```

[64]:

	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
5	Vistara	UK-945	Delhi	Morning	zero	Afternoon	Mumbai	Economy	2.33	1	5955

Analysis II

[257]:
```
# Departure_time vs source City vs Flight_Count...

flight_count_city = dataCopy.groupby(['source_city', 'departure_time'], as_index = False)['flight'].count()
flight_count_city.rename(columns = {'source_city':'City', 'departure_time':'Departure_time', 'flight':'Count_Flights'}, inplace = True)
flight_count_city
```

[257]:

	City	Departure_time	Count_Flights
0	Bangalore	Afternoon	5183
1	Bangalore	Early_Morning	13611
2	Bangalore	Evening	14243
3	Bangalore	Late_Night	457
4	Bangalore	Morning	12323
5	Bangalore	Night	6244
6	Chennai	Afternoon	5905
7	Chennai	Early_Morning	9568
8	Chennai	Evening	5546

[258]:
```
# For better analysis, Replace Early_Morning -> Morning, Late_Night -> Night... (Departure)
flight_count_city['Departure_time'].replace({'Early_Morning':'Morning', 'Late_Night':'Night'}, inplace = True)
flight_count_city
```

[258]:

	City	Departure_time	Count_Flights
0	Bangalore	Afternoon	5183
1	Bangalore	Morning	13611
2	Bangalore	Evening	14243
3	Bangalore	Night	457

[265]:
```
data_FC_Departure = flight_count_city.groupby(['City','Departure_time'], as_index = False)['Count_Flights'].sum()
data_FC_Departure
```

[265]:

	City	Departure_time	Count_Flights
0	Bangalore	Afternoon	5183
1	Bangalore	Evening	14243
2	Bangalore	Morning	25934
3	Bangalore	Night	6701
4	Chennai	Afternoon	5905
5	Chennai	Evening	5546
6	Chennai	Morning	20409
7	Chennai	Night	7842
8	Delhi	Afternoon	11246

[260]:
```
# Arival_time vs Destination City vs Flight_Count...
flight_count_Arrival_city = dataCopy.groupby(['destination_city', 'arrival_time'], as_index = False)['flight'].count()
flight_count_Arrival_city.rename(columns = {'destination_city':'Destination_City', 'arrival_time':'Arrival_Time', 'flight':'Count_Flights'}, inplace = T
flight_count_Arrival_city
```

[260]:

	Destination_City	Arrival_Time	Count_Flights
0	Bangalore	Afternoon	4827
1	Bangalore	Early_Morning	1823
2	Bangalore	Evening	13937
3	Bangalore	Late_Night	3176
4	Bangalore	Morning	11246
5	Bangalore	Night	16059
6	Chennai	Afternoon	2731

[261]:
```
flight_count_Arrival_city['Arrival_Time'].replace({'Early_Morning':'Morning', 'Late_Night':'Night'}, inplace = True)
flight_count_Arrival_city
```

[261]:

	Destination_City	Arrival_Time	Count_Flights
0	Bangalore	Afternoon	4827
1	Bangalore	Morning	1823
2	Bangalore	Evening	13937
3	Bangalore	Night	3176
4	Bangalore	Morning	11246
5	Bangalore	Night	16059
6	Chennai	Afternoon	2731
7	Chennai	Morning	3481
8	Chennai	Evening	9318

[318]:
```
data_FC_Departure1 = flight_count_Arrival_city.groupby(['Destination_City','Arrival_Time'], as_index = False)['Count_Flights'].sum()
data_FC_Departure1 = data_FC_Departure1.sort_values(by=['Destination_City', 'Count_Flights'], ascending = False)
data_FC_Departure1
```

[318]:

	Destination_City	Arrival_Time	Count_Flights
23	Mumbai	Night	23527
22	Mumbai	Morning	14373
21	Mumbai	Evening	12717
20	Mumbai	Afternoon	8531
17	Kolkata	Evening	17917
18	Kolkata	Morning	14359
19	Kolkata	Night	11718
16	Kolkata	Afternoon	5540

[379]:

```
# Plotting the above data...
plot, axis = plt.subplots(nrows = 2, ncols = 1, figsize = (22, 12))

# Plotting barplot...
sns.barplot(x = "City", y = "Count_Flights", data = data_FC_Departure, ax = axis[0], hue = 'Departure_time',
            palette='viridis')
axis[0].set_xlabel('Source City', fontsize=14)
axis[0].set_ylabel('Count_Flights', fontsize=14)
axis[0].tick_params(axis='both', labelsize=13)
axis[0].set_title('Departure_time vs Source City vs Flight_Count', fontsize = 18)

sns.barplot(x = "Destination_City", y = "Count_Flights", data = data_FC_Departure1, ax = axis[1], hue = 'Arrival_Time',
            palette='BuPu')
axis[1].set_xlabel('Destination_City', fontsize=14)
axis[1].set_ylabel('Count_Flights', fontsize=14)
axis[1].tick_params(axis='both', labelsize=13)
axis[1].set_title('Arrival_time vs Destination_City vs Flight_Count', fontsize = 18)

plt.show()
```


Departure_time vs Source City vs Flight_Count
Departure_time
Afternoon
Evening
Morning
Night
Count_Flights
0
5000
10000
15000
20000
25000
Bangalore
Chennai
Delhi
Hyderabad
Kolkata
Mumbai
Source City
Arrival_time vs Destination_City vs Flight_Count
Arrival_Time
Night
Morning
Evening
Afternoon
Count_Flights
0
5000
10000
15000
20000
Mumbai
Kolkata
Hyderabad
Delhi
Chennai
Bangalore
Destination_City

Conclusion

- People prefer to take Morning Flight from there respective Source_City.

[100]:
```
dataCopy.head(10)
```

[100]:

	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
5	Vistara	UK-945	Delhi	Morning	zero	Afternoon	Mumbai	Economy	2.33	1	5955
6	Vistara	UK-927	Delhi	Morning	zero	Morning	Mumbai	Economy	2.08	1	6060
7	Vistara	UK-951	Delhi	Afternoon	zero	Evening	Mumbai	Economy	2.17	1	6060
8	GO_FIRST	G8-334	Delhi	Early_Morning	zero	Morning	Mumbai	Economy	2.17	1	5954
9	GO_FIRST	G8-336	Delhi	Afternoon	zero	Evening	Mumbai	Economy	2.25	1	5954

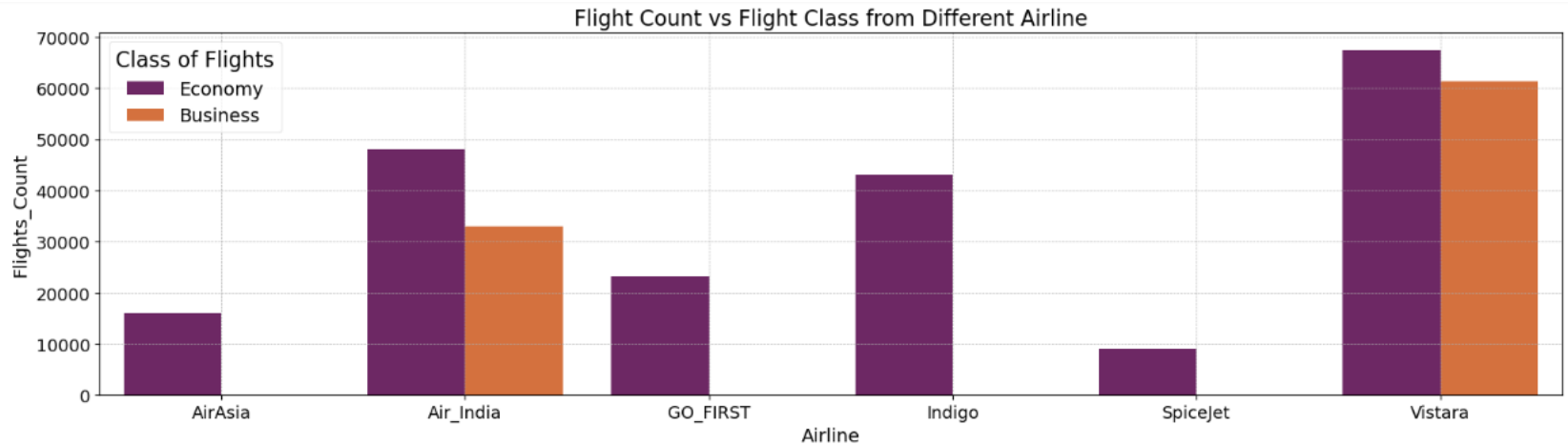
Analysis III

[104]:
```
# Flight Count vs Flight Class in Different Airline
data_Airline_class = dataCopy.groupby(['airline', 'class'], as_index = False)['flight'].count()
data_Airline_class
```

[104]:

	airline	class	flight
0	AirAsia	Economy	16100
1	Air_India	Business	32996
2	Air_India	Economy	48064
3	GO_FIRST	Economy	23176
4	Indigo	Economy	43128
5	SpiceJet	Economy	9015
6	Vistara	Business	61293
7	Vistara	Economy	67434

[385]:
```
# Plotting the above data...
plot, axis = plt.subplots(nrows = 1, ncols = 1, figsize = (20, 5))

# Plotting barplot...
sns.barplot(x = "airline", y = "flight", data = data_Airline_class, ax = axis, hue = 'class',
            palette = 'inferno')
axis.set_xlabel('Airline', fontsize=14)
axis.set_ylabel('Flights_Count', fontsize=14)
axis.tick_params(axis='both', labelsize=13)
axis.grid(visible = True, which='both', linestyle='--', linewidth=0.5)
axis.set_title('Flight Count vs Flight Class from Different Airline', fontsize = 16)
# Change the value of legends...
legend = axis.legend(title='Class of Flights', title_fontsize='16', fontsize='14')
legend.get_frame().set_linewidth(0.5)

plt.show()
```



Conclusion

- Passengers prefer to choose Business Class exclusively from Air India and Vistara.
- Among these, Vistara is the most preferred airline for Business Class
- Similarly, Vistara also holds the highest preference for Economy Class among travelers.

[242]:
```
dataCopy.head(10)
```

[242]:

	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	5953.0
1	SpiceJet	SG-8157	Delhi	Early_Morning	zero	Morning	Mumbai	Economy	2.33	1	5953.0
2	AirAsia	I5-764	Delhi	Early_Morning	zero	Early_Morning	Mumbai	Economy	2.17	1	5956.0
3	Vistara	UK-995	Delhi	Morning	zero	Afternoon	Mumbai	Economy	2.25	1	5955.0
4	Vistara	UK-963	Delhi	Morning	zero	Morning	Mumbai	Economy	2.33	1	5955.0
5	Vistara	UK-945	Delhi	Morning	zero	Afternoon	Mumbai	Economy	2.33	1	5955.0

Analysis IV

[243]:
```
dataCopy.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 301206 entries, 0 to 301205
Data columns (total 11 columns):
 #   Column            Non-Null Count   Dtype  
---  ------            --------------   -----  
 0   airline           301206 non-null  object 
 1   flight            301206 non-null  object 
 2   source_city       301206 non-null  object 
 3   departure_time    301206 non-null  object 
 4   stops             301206 non-null  object 
 5   arrival_time      301206 non-null  object 
 6   destination_city  301206 non-null  object 
 7   class             301206 non-null  object 
 8   duration          301206 non-null  float64
 9   days_left         301206 non-null  int64  
 10  price             301206 non-null  float64
dtypes: float64(2), int64(1), object(8)
memory usage: 25.3+ MB
```

[244]:
```
# Changing dataType of price...
dataCopy['price'].value_counts()
```

[244]:
```
price
54608.0      1547
2339.0       1442
54684.0      1390
60978.0      1383
60508.0      1230
49725.0      1205
51707.0      1205
5949.0       1196
49613.0      1150
5955.0       1138
56588.0      1111
55983.0      1108
60260.0      1107
6489.0       1082
```

[245]:
```
# Some non mumeric values are present so we have to convert it to null...
dataCopy['price'] = pd.to_numeric(dataCopy['price'], errors='coerce')

# Total null value set...
dataCopy['price'].isnull().sum()
```

[245]:
```
0
```

[246]:
```
# Changing null values to Mode values..
Mod_price = dataCopy['price'].mode()[0]
Mod_price
```

[246]:
```
54608.0
```

[247]:
```
# Setting Null values to Mod_Price...
dataCopy['price'] = dataCopy['price'].fillna(Mod_price)
```

[202]:
```
# Total null value have changed so no null values...
dataCopy['price'].isnull().sum()
```

[202]:
```
0
```

[248]:
```
# DataType Changed of Price column...
dataCopy.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 301206 entries, 0 to 301205
Data columns (total 11 columns):
 #   Column            Non-Null Count   Dtype
---  ------            --------------   -----
 0   airline           301206 non-null  object
 1   flight            301206 non-null  object
 2   source_city       301206 non-null  object
 3   departure_time    301206 non-null  object
 4   stops             301206 non-null  object
 5   arrival_time      301206 non-null  object
 6   destination_city  301206 non-null  object
 7   class             301206 non-null  object
 8   duration          301206 non-null  float64
 9   days_left         301206 non-null  int64
 10  price             301206 non-null  float64
dtypes: float64(2), int64(1), object(8)
```

[249]:
```
# Top 5 flights that gets travel more...
flight_travel = dataCopy['flight'].value_counts().reset_index()
flight_travel.columns = ['Flight', 'Flight_Count']
flight_travel = flight_travel.head(5)
flight_travel
```

[249]:

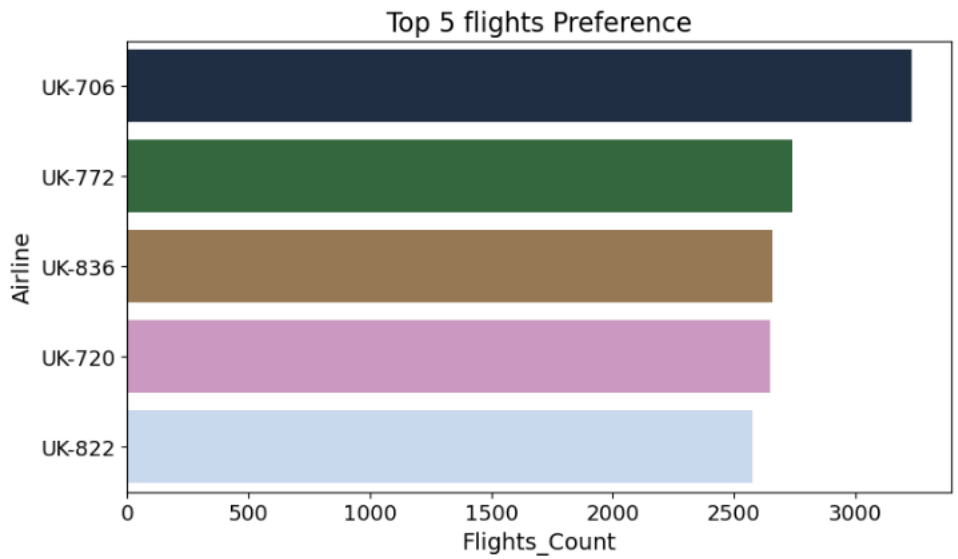
	Flight	Flight_Count
0	UK-706	3235
1	UK-772	2741
2	UK-836	2657
3	UK-720	2650
4	UK-822	2575

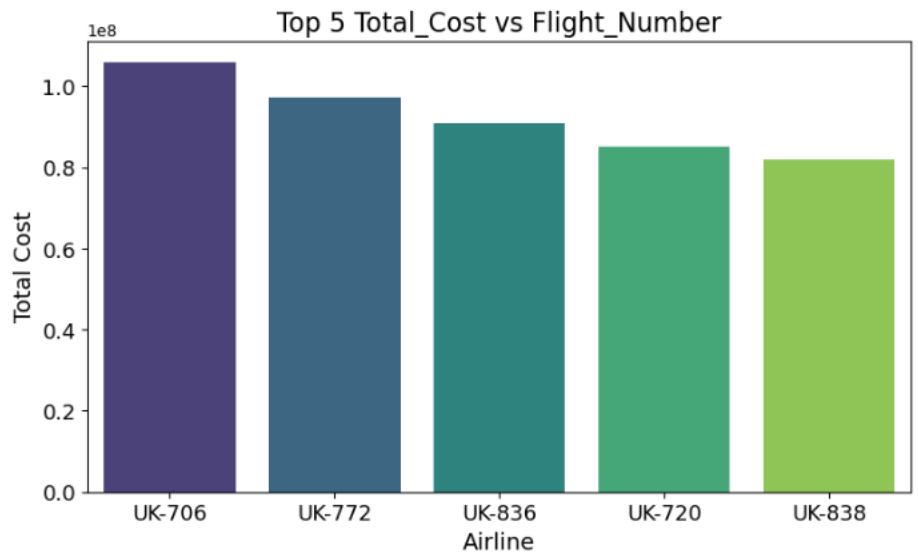
[207]:
```
# Top 5 Flights total revinue vs flight Number...
data_F_Price = dataCopy.groupby(['flight'], as_index = False)['price'].sum()
data_F_Price = data_F_Price.sort_values(by = ['price'], ascending = False)
top5_data_F_Price = data_F_Price.head(5)
top5_data_F_Price
```

[207]:

	flight	price
1442	UK-706	105871560.0
1454	UK-772	97212901.0
1490	UK-836	91016350.0
1445	UK-720	85182167.0
1492	UK-838	82050784.0

[211]:
```
plot, axis = plt.subplots(nrows = 1, ncols = 2, figsize = (20, 5))
sns.barplot(x = 'Flight_Count', y = 'Flight', data = flight_travel, ax = axis[0], orient = 'h', palette='cubehelix')
axis[0].set_xlabel('Flights_Count', fontsize=14)
axis[0].set_ylabel('Airline', fontsize=14)
axis[0].tick_params(axis='both', labelsize=13)
axis[0].set_title('Top 5 flights Preference', fontsize = 16)

sns.barplot(x = 'flight', y = 'price', data = top5_data_F_Price, ax = axis[1], palette='viridis')
axis[1].set_xlabel('Airline', fontsize=14)
axis[1].set_ylabel('Total Cost', fontsize=14)
axis[1].tick_params(axis='both', labelsize=13)
axis[1].set_title('Top 5 Total_Cost vs Flight_Number', fontsize = 16)
plt.show()
```





Conclusion

- The airline UK-706 operates a greater number of flights.
- UK-822 has a higher number of flights compared to UK-838, yet UK-838 generates more revenue than UK-822.
- UK-706 and UK-772 exhibit a dramatic variation in the number of flights. However, the total cost remains relatively stable between the two.

[177]:
```
dataCopy.size
```

[177]:
```
3313266
```

Analysis V

[182]:
```
len = dataCopy.shape[0]
pd.options.display.max_rows = len
```

[183]:
```
dataCopy['flight'].value_counts()
```

[183]:
```
flight
UK-706    3235
UK-772    2741
UK-836    2657
UK-720    2650
UK-822    2575
UK-828    2524
UK-832    2494
UK-874    2423
UK-826    2404
UK-838    2361
UK-860    2329
UK-876    2307
UK-878    2285
UK-824    2219
UK-830    2204
UK-870    2199
UK-774    2145
```

[217]:
```
dataCopy.head(10)
```

[304]:
```
# Common destination cities from each source city....
Common_Destination = dataCopy['destination_city'].value_counts().reset_index()
Common_Destination.rename(columns={'destination_city': 'Destination_City', 'count':'Flight_Count'}, inplace=True)
Common_Destination.sort_values
Common_Destination
```

[304]:

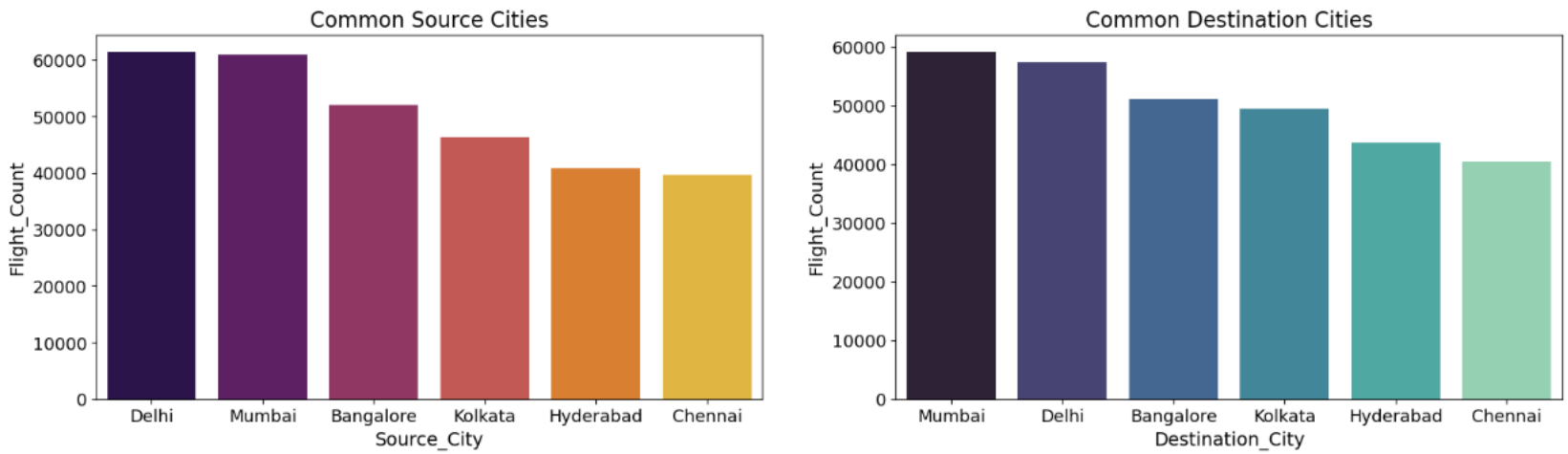
	Destination_City	Flight_Count
0	Mumbai	59148
1	Delhi	57360
2	Bangalore	51068
3	Kolkata	49534
4	Hyderabad	43728
5	Chennai	40368

[306]:
```
# Common Source cities....
Common_Source = dataCopy['source_city'].value_counts().reset_index()
Common_Source.rename(columns={'source_city': 'Source_City', 'count':'Flight_Count'}, inplace=True)
Common_Source.sort_values
Common_Source
```

[306]:

	Source_City	Flight_Count
0	Delhi	61394
1	Mumbai	60896
2	Bangalore	52061
3	Kolkata	46347
4	Hyderabad	40806
5	Chennai	39702

[310]:
```
plot, axis = plt.subplots(nrows = 1, ncols = 2, figsize = (20, 5))
sns.barplot(x = 'Source_City', y = 'Flight_Count', data = Common_Source, ax = axis[0], palette='inferno')
axis[0].set_xlabel('Source_City', fontsize=14)
axis[0].set_ylabel('Flight_Count', fontsize=14)
axis[0].tick_params(axis='both', labelsize=13)
axis[0].set_title('Common Source Cities', fontsize = 16)

sns.barplot(x = 'Destination_City', y = 'Flight_Count', data = Common_Destination, ax = axis[1], palette='mako')
axis[1].set_xlabel('Destination_City', fontsize=14)
axis[1].set_ylabel('Flight_Count', fontsize=14)
axis[1].tick_params(axis='both', labelsize=13)
axis[1].set_title('Common Destination Cities', fontsize = 16)
plt.show()
```



Conclusion

- The most frequent source city is Delhi.
- The most common destination city is Mumbai.

[312]:
```
dataCopy.head(10)
```

[312]:

	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	5953.0
1	SpiceJet	SG-8157	Delhi	Early_Morning	zero	Morning	Mumbai	Economy	2.33	1	5953.0
2	AirAsia	I5-764	Delhi	Early_Morning	zero	Early_Morning	Mumbai	Economy	2.17	1	5956.0
3	Vistara	UK-995	Delhi	Morning	zero	Afternoon	Mumbai	Economy	2.25	1	5955.0
4	Vistara	UK-963	Delhi	Morning	zero	Morning	Mumbai	Economy	2.33	1	5955.0
5	Vistara	UK-945	Delhi	Morning	zero	Afternoon	Mumbai	Economy	2.33	1	5955.0
6	Vistara	UK-927	Delhi	Morning	zero	Morning	Mumbai	Economy	2.08	1	6060.0
7	Vistara	UK-951	Delhi	Afternoon	zero	Evening	Mumbai	Economy	2.17	1	6060.0
8	GO_FIRST	G8-334	Delhi	Early_Morning	zero	Morning	Mumbai	Economy	2.17	1	5954.0
9	GO_FIRST	G8-336	Delhi	Afternoon	zero	Evening	Mumbai	Economy	2.25	1	5954.0

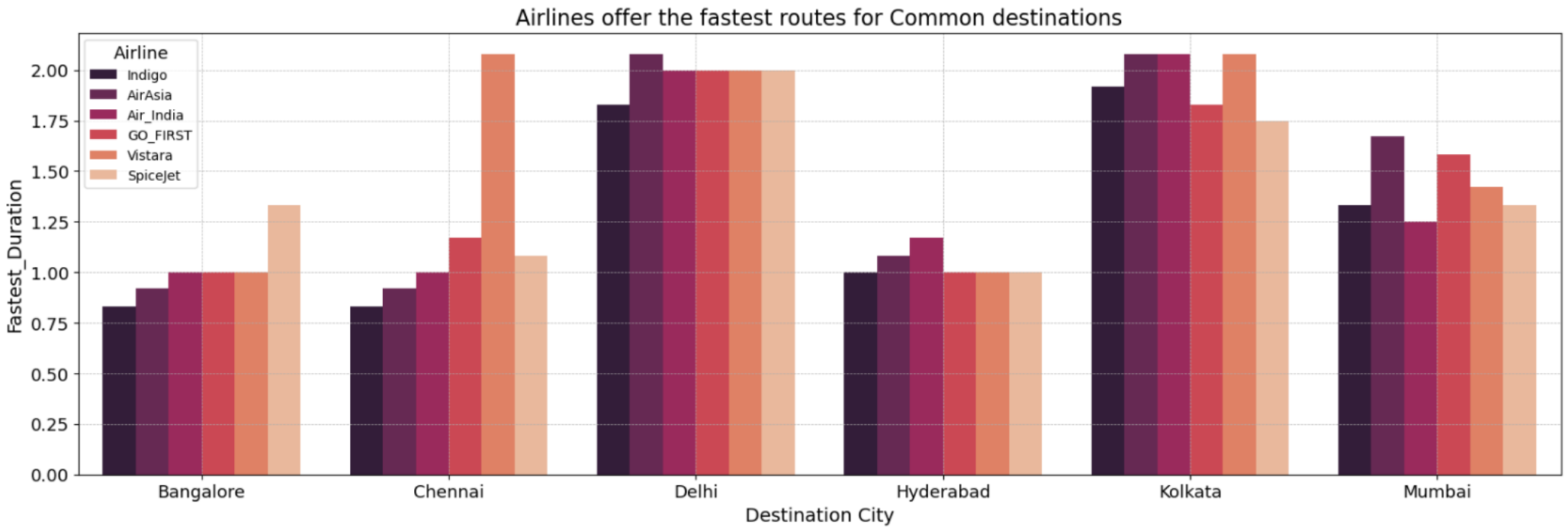
Analysis VI

[316]:
```
# Airlines offer the fastest routes for common destinations
fastest_routes = dataCopy.groupby(['destination_city', 'airline'], as_index=False)['duration'].min()
fastest_routes.rename(columns={'duration': 'Fastest_Duration'}, inplace=True)
fastest_routes = fastest_routes.sort_values(by=['destination_city', 'Fastest_Duration'])
fastest_routes
```

[316]:

	destination_city	airline	Fastest_Duration
3	Bangalore	Indigo	0.83
0	Bangalore	AirAsia	0.92
1	Bangalore	Air_India	1.00
2	Bangalore	GO_FIRST	1.00

[334]:
```
# Plotting the above data...
plot, axis = plt.subplots(nrows = 1, ncols = 1, figsize = (20, 6))

# Plotting barplot...
sns.barplot(x = "destination_city", y = "Fastest_Duration", data = fastest_routes, ax = axis, hue = 'airline',
            palette='rocket')
axis.set_xlabel('Destination City', fontsize=14)
axis.set_ylabel('Fastest_Duration', fontsize=14)
axis.tick_params(axis='both', labelsize=13)
axis.legend(loc='upper left', title = 'Airline', title_fontsize = '13', fontsize = '10')
axis.grid(visible = True, which='both', linestyle='--', linewidth=0.5, zorder=1)
axis.set_title('Airlines offer the fastest routes for Common destinations', fontsize = 16)

plt.show()
```



Conclusion

- For Bangalore, the fastest flight is offered by SpiceJet.
- For Chennai, Vistara provides the fastest flight.
- For Delhi, AirAsia has the quickest flight.
- For Hyderabad, Air India offers the fastest flight.
- For Kolkata, the fastest flights are available with AirAsia, Air India, and Vistara.
- For Mumbai, AirAsia provides the fastest flight.

[340]:
```
dataCopy.head(10)
```

[340]:

	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	5953.0
1	SpiceJet	SG-8157	Delhi	Early_Morning	zero	Morning	Mumbai	Economy	2.33	1	5953.0
2	AirAsia	I5-764	Delhi	Early_Morning	zero	Early_Morning	Mumbai	Economy	2.17	1	5956.0
3	Vistara	UK-995	Delhi	Morning	zero	Afternoon	Mumbai	Economy	2.25	1	5955.0
4	Vistara	UK-963	Delhi	Morning	zero	Morning	Mumbai	Economy	2.33	1	5955.0
5	Vistara	UK-945	Delhi	Morning	zero	Afternoon	Mumbai	Economy	2.33	1	5955.0
6	Vistara	UK-927	Delhi	Morning	zero	Morning	Mumbai	Economy	2.08	1	6060.0
7	Vistara	UK-951	Delhi	Afternoon	zero	Evening	Mumbai	Economy	2.17	1	6060.0
8	GO_FIRST	G8-334	Delhi	Early_Morning	zero	Morning	Mumbai	Economy	2.17	1	5954.0
9	GO_FIRST	G8-336	Delhi	Afternoon	zero	Evening	Mumbai	Economy	2.25	1	5954.0

Analysis VII

[341]:
```
dataCopy['stops'].unique()
```

[341]:
```
array(['zero', 'one', 'two_or_more'], dtype=object)
```

[342]:
```
conditions = [
    (dataCopy['stops'] == 'zero'),
    (dataCopy['stops'] == 'one'),
    (dataCopy['stops'] == 'two_or_more')
]
choices = [0, 1, 2]
```

[350]:
```
# Adding an extra column for stops...
dataCopy['Total_stops'] = np.select(conditions, choices, default = 0)
dataCopy.head(10)
```

[350]:

	airline	flight	source_city	departure_time	stops	arrival_time	destination_city	class	duration	days_left	price	Total_stops
0	SpiceJet	SG-8709	Delhi	Evening	zero	Night	Mumbai	Economy	2.17	1	5953.0	0
1	SpiceJet	SG-8157	Delhi	Early_Morning	zero	Morning	Mumbai	Economy	2.33	1	5953.0	0
2	AirAsia	I5-764	Delhi	Early_Morning	zero	Early_Morning	Mumbai	Economy	2.17	1	5956.0	0
3	Vistara	UK-995	Delhi	Morning	zero	Afternoon	Mumbai	Economy	2.25	1	5955.0	0
4	Vistara	UK-963	Delhi	Morning	zero	Morning	Mumbai	Economy	2.33	1	5955.0	0
5	Vistara	UK-945	Delhi	Morning	zero	Afternoon	Mumbai	Economy	2.33	1	5955.0	0
6	Vistara	UK-927	Delhi	Morning	zero	Morning	Mumbai	Economy	2.08	1	6060.0	0
7	Vistara	UK-951	Delhi	Afternoon	zero	Evening	Mumbai	Economy	2.17	1	6060.0	0
8	GO_FIRST	G8-334	Delhi	Early_Morning	zero	Morning	Mumbai	Economy	2.17	1	5954.0	0
9	GO_FIRST	G8-336	Delhi	Afternoon	zero	Evening	Mumbai	Economy	2.25	1	5954.0	0

[352]:
```
dataCopy['Total_stops'].unique()
```

0	SpiceJet	SG-8709	Delhi	Evening	zero	Night	Mumbai	Economy	2.17	1	5953.0	0
1	SpiceJet	SG-8157	Delhi	Early_Morning	zero	Morning	Mumbai	Economy	2.33	1	5953.0	0
2	AirAsia	I5-764	Delhi	Early_Morning	zero	Early_Morning	Mumbai	Economy	2.17	1	5956.0	0
3	Vistara	UK-995	Delhi	Morning	zero	Afternoon	Mumbai	Economy	2.25	1	5955.0	0
4	Vistara	UK-963	Delhi	Morning	zero	Morning	Mumbai	Economy	2.33	1	5955.0	0
5	Vistara	UK-945	Delhi	Morning	zero	Afternoon	Mumbai	Economy	2.33	1	5955.0	0
6	Vistara	UK-927	Delhi	Morning	zero	Morning	Mumbai	Economy	2.08	1	6060.0	0
7	Vistara	UK-951	Delhi	Afternoon	zero	Evening	Mumbai	Economy	2.17	1	6060.0	0
8	GO_FIRST	G8-334	Delhi	Early_Morning	zero	Morning	Mumbai	Economy	2.17	1	5954.0	0
9	GO_FIRST	G8-336	Delhi	Afternoon	zero	Evening	Mumbai	Economy	2.25	1	5954.0	0

[352]:
```
dataCopy['Total_stops'].unique()
```

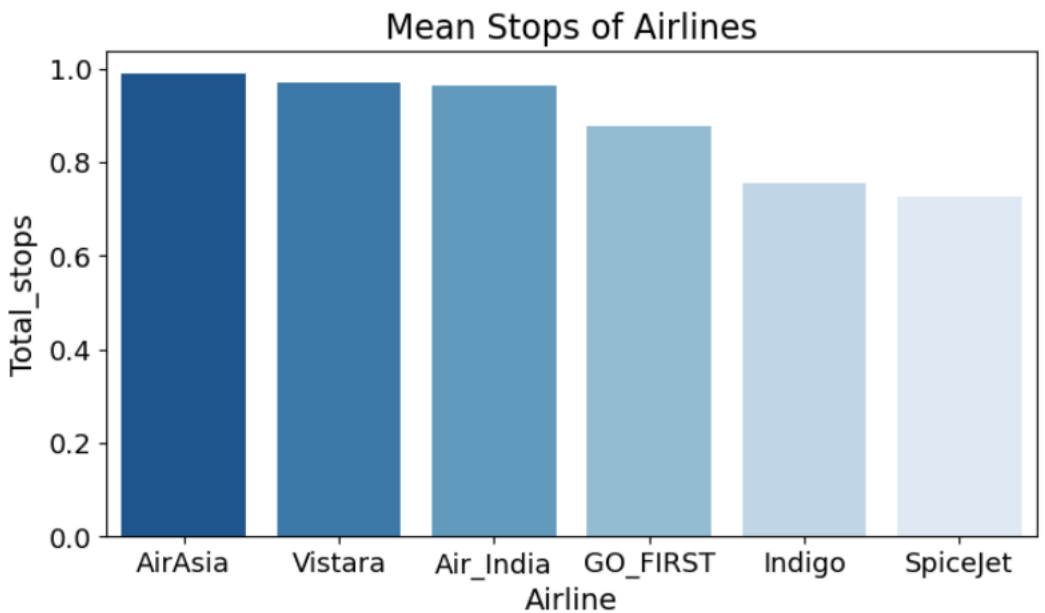
[352]:
```
array([0, 1, 2])
```

[361]:
```
stops = dataCopy.groupby(['airline'], as_index=False)['Total_stops'].mean()
stops = stops.sort_values(by=['Total_stops'], ascending = False)
stops
```

[361]:

	airline	Total_stops
0	AirAsia	0.988323
5	Vistara	0.970107
1	Air_India	0.963854
2	GO_FIRST	0.878409
3	Indigo	0.757026
4	SpiceJet	0.726789

[387]:
```
# Plotting the above data...
plot, axis = plt.subplots(nrows = 1, ncols = 1, figsize = (8, 4.2))

# Plotting barplot...
original_palette = sns.color_palette('Blues')
# Reverse the palette
reversed_palette = original_palette[::-1]
sns.barplot(x = "airline", y = "Total_stops", data = stops, ax = axis,
            palette = reversed_palette)
axis.set_xlabel('Airline', fontsize=14)
axis.set_ylabel('Total_stops', fontsize=14)
axis.tick_params(axis='both', labelsize=13)
axis.set_title('Mean Stops of Airlines', fontsize = 16)
plt.show()
```



[12]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats
```

[53]:
```
data = pd.read_csv(r"D:\FIFTHFORCE\Filter_Flight_Data2.csv")
data.head(10)
```

[53]:

	airline	flight	source_city	departure_time	arrival_time	destination_city	class	duration	days_left	price	Time_duration	Total_stops
0	SpiceJet	SG-8709	Delhi	Evening	Night	Mumbai	Economy	2.17	1	5953	0 days 02:16:00	0
1	SpiceJet	SG-8157	Delhi	Morning	Morning	Mumbai	Economy	2.33	1	5953	0 days 02:33:00	0
2	AirAsia	I5-764	Delhi	Morning	Morning	Mumbai	Economy	2.17	1	5956	0 days 02:16:00	0
3	Vistara	UK-995	Delhi	Morning	Afternoon	Mumbai	Economy	2.25	1	5955	0 days 02:25:00	0
4	Vistara	UK-963	Delhi	Morning	Morning	Mumbai	Economy	2.33	1	5955	0 days 02:33:00	0
5	Vistara	UK-945	Delhi	Morning	Afternoon	Mumbai	Economy	2.33	1	5955	0 days 02:33:00	0
6	Vistara	UK-927	Delhi	Morning	Morning	Mumbai	Economy	2.08	1	6060	0 days 02:08:00	0
7	Vistara	UK-951	Delhi	Afternoon	Evening	Mumbai	Economy	2.17	1	6060	0 days 02:16:00	0
8	GO_FIRST	G8-334	Delhi	Morning	Morning	Mumbai	Economy	2.17	1	5954	0 days 02:16:00	0
9	GO_FIRST	G8-336	Delhi	Afternoon	Evening	Mumbai	Economy	2.25	1	5954	0 days 02:25:00	0

[54]:
```
data.describe()
```

[54]:

	duration	days_left	price	Total_stops
count	301206.000000	301206.000000	3.012060e+05	301206.000000
mean	12.202056	26.042277	2.111804e+04	0.924550
std	7.135976	13.564973	2.544767e+04	0.397471
min	0.830000	1.000000	1.105000e+03	0.000000
25%	6.830000	15.000000	4.788000e+03	1.000000
50%	11.250000	26.000000	7.455000e+03	1.000000
75%	16.170000	38.000000	4.252100e+04	1.000000

[55]:
```
dataCopy = data
```

[59]:
```
# Function to convert duration in minutes.seconds to total seconds
def convert_to_seconds(duration):
    minutes = int(duration)
    seconds = (duration - minutes) * 100
    total_seconds = (minutes * 60) + seconds
    return total_seconds

# Apply the function to the 'Time_duration' column and create a new column
dataCopy['Time_duration_seconds'] = dataCopy['duration'].apply(convert_to_seconds)
dataCopy.head(10)
```

[59]:

	airline	flight	source_city	departure_time	arrival_time	destination_city	class	duration	days_left	price	Time_duration	Total_stops	Time_duration_seconds
0	SpiceJet	SG-8709	Delhi	Evening	Night	Mumbai	Economy	2.17	1	5953	0 days 02:16:00	0	137.0
1	SpiceJet	SG-8157	Delhi	Morning	Morning	Mumbai	Economy	2.33	1	5953	0 days 02:33:00	0	153.0
2	AirAsia	I5-764	Delhi	Morning	Morning	Mumbai	Economy	2.17	1	5956	0 days 02:16:00	0	137.0
3	Vistara	UK-995	Delhi	Morning	Afternoon	Mumbai	Economy	2.25	1	5955	0 days 02:25:00	0	145.0
4	Vistara	UK-963	Delhi	Morning	Morning	Mumbai	Economy	2.33	1	5955	0 days 02:33:00	0	153.0
5	Vistara	UK-945	Delhi	Morning	Afternoon	Mumbai	Economy	2.33	1	5955	0 days 02:33:00	0	153.0
6	Vistara	UK-927	Delhi	Morning	Morning	Mumbai	Economy	2.08	1	6060	0 days 02:08:00	0	128.0
7	Vistara	UK-951	Delhi	Afternoon	Evening	Mumbai	Economy	2.17	1	6060	0 days 02:16:00	0	137.0
8	GO_FIRST	G8-334	Delhi	Morning	Morning	Mumbai	Economy	2.17	1	5954	0 days 02:16:00	0	137.0
9	GO_FIRST	G8-336	Delhi	Afternoon	Evening	Mumbai	Economy	2.25	1	5954	0 days 02:25:00	0	145.0

[61]:
```
dataCopy.info()
```

[62]:
```
dataCopy1 = dataCopy
dataCopy1 = dataCopy1[dataCopy1['class'] == 'Economy']
```

[63]:
```
# Group by 'days_left' and calculate the mean price
# For Economy...
grouped_data = dataCopy1.groupby(['days_left'], as_index = False)['price'].mean().round(2)
# Sort by 'price' in descending order and get the top 10
grouped_data.sort_values(by='price', ascending=False, inplace=True)
grouped_data1 = grouped_data.head(10)
grouped_data1
```

0	1	36732.63
1	2	13980.83
2	3	13174.05
8	9	11352.92
9	10	11187.50
3	4	10901.39
4	5	10605.92
7	8	10479.49
6	7	10471.87
5	6	10319.68

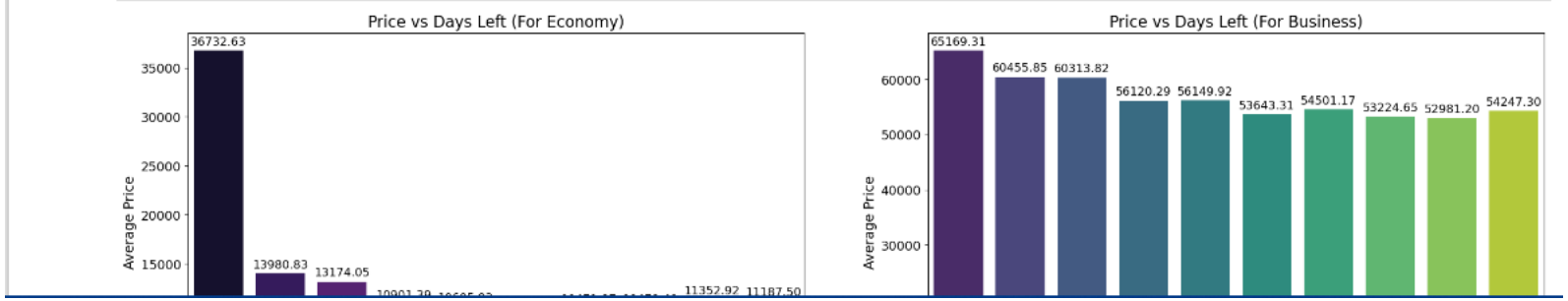
[64]:
```
# For Business class.....
dataCopy2 = dataCopy
dataCopy2 = dataCopy2[dataCopy2['class'] == 'Business']
```

[65]:
```
# Group by 'days_left' and calculate the mean price
# For Business...
grouped_data = dataCopy2.groupby(['days_left'], as_index = False)['price'].mean().round(2)
# Sort by 'price' in descending order and get the top 10
grouped_data.sort_values(by='price', ascending=False, inplace=True)
grouped_data2 = grouped_data.head(10)
grouped_data2
```

0	1	65169.31
1	2	60455.85
2	3	60313.82

[66]:

```
sorted_data = grouped_data1.sort_values(by='days_left')
# Plotting the sorted data
plot, axis = plt.subplots(nrows=1, ncols=2, figsize=(25, 7))

# Plotting barplot1
sns.barplot(x="days_left", y="price", data=grouped_data1, ax=axis[0], palette='magma')
axis[0].set_xlabel('Days Left', fontsize=14)
axis[0].set_ylabel('Average Price', fontsize=14)
axis[0].tick_params(axis='both', labelsize=13)
axis[0].set_title('Price vs Days Left (For Economy)', fontsize=16)
# Annotating each bar with its height
for p in axis[0].patches:
    axis[0].annotate(format(p.get_height(), '.2f'),
                     (p.get_x() + p.get_width() / 2., p.get_height()),
                     ha = 'center', va = 'center',
                     xytext = (0, 9),
                     textcoords = 'offset points',
                     fontsize=12, color='black')

# Plotting barplot2
sns.barplot(x="days_left", y="price", data=grouped_data2, ax=axis[1], palette='viridis')
axis[1].set_xlabel('Days Left', fontsize=14)
axis[1].set_ylabel('Average Price', fontsize=14)
axis[1].tick_params(axis='both', labelsize=13)
axis[1].set_title('Price vs Days Left (For Business)', fontsize=16)
# Annotating each bar with its height
for p in axis[1].patches:
    axis[1].annotate(format(p.get_height(), '.2f'),
                     (p.get_x() + p.get_width() / 2., p.get_height()),
                     ha = 'center', va = 'center',
                     xytext = (0, 9),
                     textcoords = 'offset points',
                     fontsize=12, color='black')
plt.show()
```



9	GO_FIRST	336	Delhi	Afternoon	Evening	Mumbai	Economy	2.25	1	5954	0 days 02:25:00	0	145.0

[68]:
```
dataCopy23 = dataCopy
```

[69]:
```
# For Economy...
dataCopy23 = dataCopy23[dataCopy23['class'] == 'Economy']
grouped_data2 = dataCopy23.groupby(['airline', 'Total_stops'], as_index = False)['price'].mean().round(2)
grouped_data2
```

	airline	Total_stops	price
7	GO_FIRST	1	6009.16
8	GO_FIRST	2	7107.71
9	Indigo	0	4077.11
10	Indigo	1	5920.72
11	Indigo	2	7834.84
12	SpiceJet	0	4924.15
13	SpiceJet	1	7158.12
14	Vistara	0	4605.13
15	Vistara	1	8234.66
16	Vistara	2	10371.42

[70]:
```
dataCopy24 = dataCopy
```

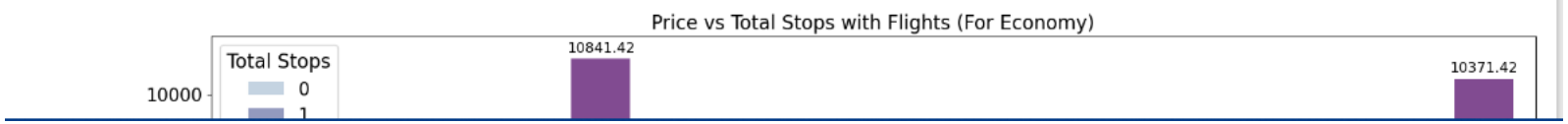
[71]:
```
# For Business...
dataCopy24 = dataCopy24[dataCopy24['class'] == 'Business']
grouped_data3 = dataCopy24.groupby(['airline', 'Total_stops'], as_index = False)['price'].mean().round(2)
grouped_data3
```

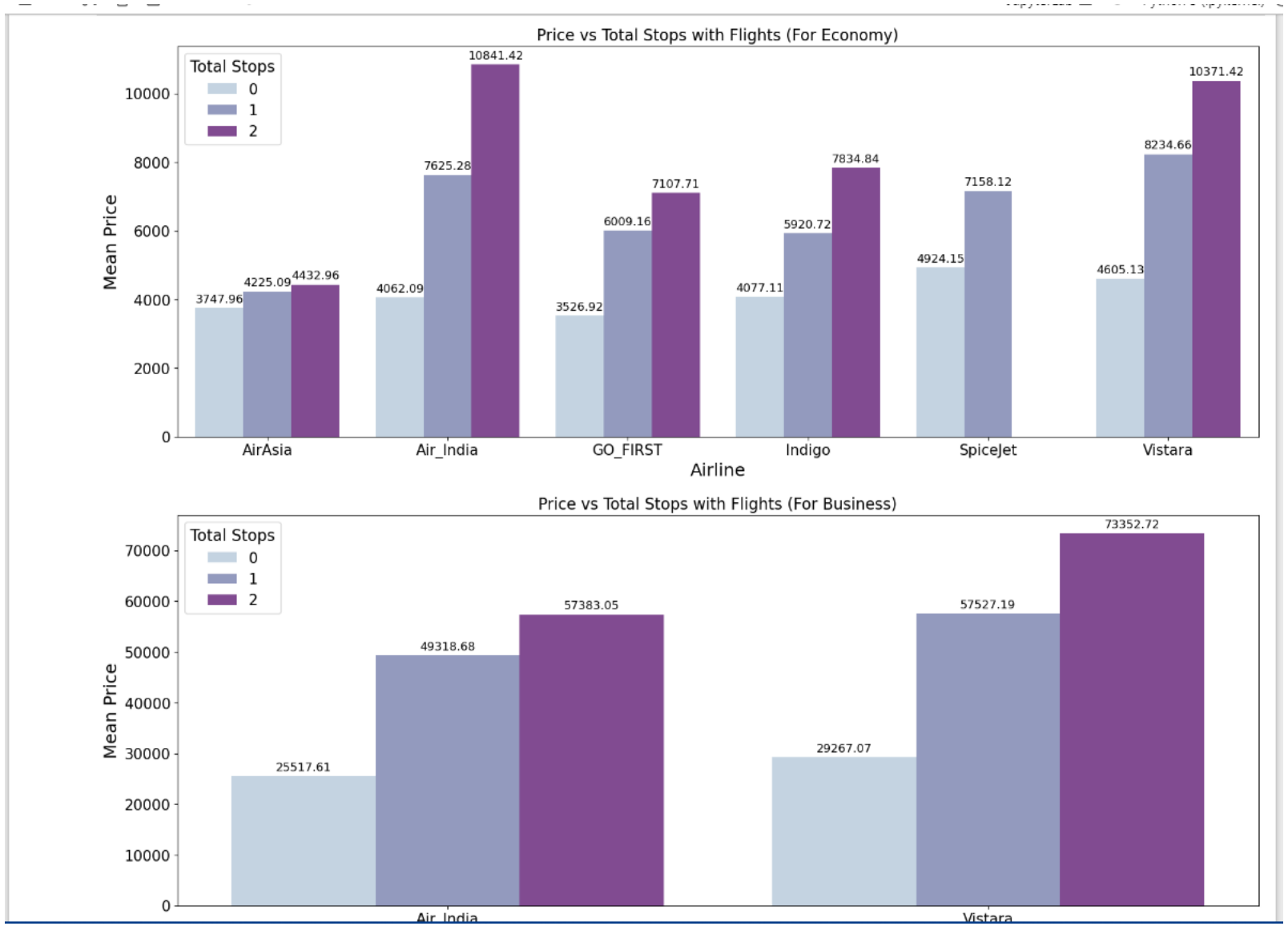
[71]:

	airline	Total_stops	price
0	Air_India	0	25517.61
1	Air_India	1	49318.68
2	Air_India	2	57383.05
3	Vistara	0	29267.07
4	Vistara	1	57527.19
5	Vistara	2	73352.72

[101]:

```
grouped_data2['Total_stops'] = grouped_data2['Total_stops'].astype('str')

# Create the plot
plot, axis = plt.subplots(nrows=2, ncols=1, figsize=(20, 16))
# Plotting the barplot
sns.barplot(x="airline", y="price", data=grouped_data2, hue='Total_stops', ax=axis[0], palette='BuPu')
# Set labels and title
axis[0].set_xlabel('Airline', fontsize=18)
axis[0].set_ylabel('Mean Price', fontsize=18)
axis[0].tick_params(axis='both', labelsize=15)
axis[0].set_title('Price vs Total Stops with Flights (For Economy)', fontsize=16)
axis[0].legend(title='Total Stops', title_fontsize='16', fontsize='15', loc='upper left')
# Annotating each bar with its height
for p in axis[0].patches:
    axis[0].annotate(format(p.get_height(), '.2f'),
                     (p.get_x() + p.get_width() / 2., p.get_height()),
                     ha = 'center', va = 'center',
                     xytext = (0, 9),
                     textcoords = 'offset points',
                     fontsize=12, color='black')


grouped_data3['Total_stops'] = grouped_data3['Total_stops'].astype('str')
# Plotting the barplot
sns.barplot(x="airline", y="price", data=grouped_data3, hue='Total_stops', ax=axis[1], palette='BuPu')
# Set labels and title
axis[1].set_xlabel('Airline', fontsize=18)
axis[1].set_ylabel('Mean Price', fontsize=18)
axis[1].tick_params(axis='both', labelsize=15)
axis[1].set_title('Price vs Total Stops with Flights (For Business)', fontsize=16)
axis[1].legend(title='Total Stops', title_fontsize='16', fontsize='15', loc='upper left')
for p in axis[1].patches:
    axis[1].annotate(format(p.get_height(), '.2f'),
                     (p.get_x() + p.get_width() / 2., p.get_height()),
                     ha = 'center', va = 'center',
                     xytext = (0, 9),
                     textcoords = 'offset points',
                     fontsize=12, color='black')
# Show the plot
plt.show()
```




Price vs Total Stops with Flights (For Economy)
Total Stops
0
1
2
Mean Price
Airline
AirAsia
Air_India
GO_FIRST
Indigo
SpiceJet
Vistara
3747.96
4225.09
4432.96
4062.09
7625.28
10841.42
3526.92
6009.16
7107.71
4077.11
5920.72
7834.84
4924.15
7158.12
4605.13
8234.66
10371.42
Price vs Total Stops with Flights (For Business)
Total Stops
0
1
2
Mean Price
Air_India
Vistara
25517.61
49318.68
57383.05
29267.07
57527.19
73352.72

[91]:
```
# Function to convert 'X.Y' to seconds
def convert_float_to_seconds(duration_float):
    # Extract hours and minutes from the float
    hours = int(duration_float)
    minutes = int((duration_float - hours) * 100)
    total_seconds = hours * 3600 + minutes * 60
    # Fractional part is seconds
    return total_seconds

# Apply the function to the DataFrame
df['Time_duration_seconds'] = df['duration'].apply(convert_float_to_seconds)
df.head(10)
```

[91]:

	airline	flight	source_city	departure_time	arrival_time	destination_city	class	duration	days_left	price	Time_duration	Total_stops	Time_duration_seconds
0	SpiceJet	SG-8709	Delhi	Evening	Night	Mumbai	Economy	2.17	1	5953	0 days 02:16:00	0	8160
1	SpiceJet	SG-8157	Delhi	Morning	Morning	Mumbai	Economy	2.33	1	5953	0 days 02:33:00	0	9180
2	AirAsia	I5-764	Delhi	Morning	Morning	Mumbai	Economy	2.17	1	5956	0 days 02:16:00	0	8160
3	Vistara	UK-995	Delhi	Morning	Afternoon	Mumbai	Economy	2.25	1	5955	0 days 02:25:00	0	8700
4	Vistara	UK-963	Delhi	Morning	Morning	Mumbai	Economy	2.33	1	5955	0 days 02:33:00	0	9180
5	Vistara	UK-945	Delhi	Morning	Afternoon	Mumbai	Economy	2.33	1	5955	0 days 02:33:00	0	9180
6	Vistara	UK-927	Delhi	Morning	Morning	Mumbai	Economy	2.08	1	6060	0 days 02:08:00	0	7680
7	Vistara	UK-951	Delhi	Afternoon	Evening	Mumbai	Economy	2.17	1	6060	0 days 02:16:00	0	8160
8	GO_FIRST	G8-334	Delhi	Morning	Morning	Mumbai	Economy	2.17	1	5954	0 days 02:16:00	0	8160
9	GO_FIRST	G8-336	Delhi	Afternoon	Evening	Mumbai	Economy	2.25	1	5954	0 days 02:25:00	0	8700

[92]:
```
# TimeDuration = df.groupby(['airline', 'Total_stops'], as_index = False)['price'].mean().round(2)
```

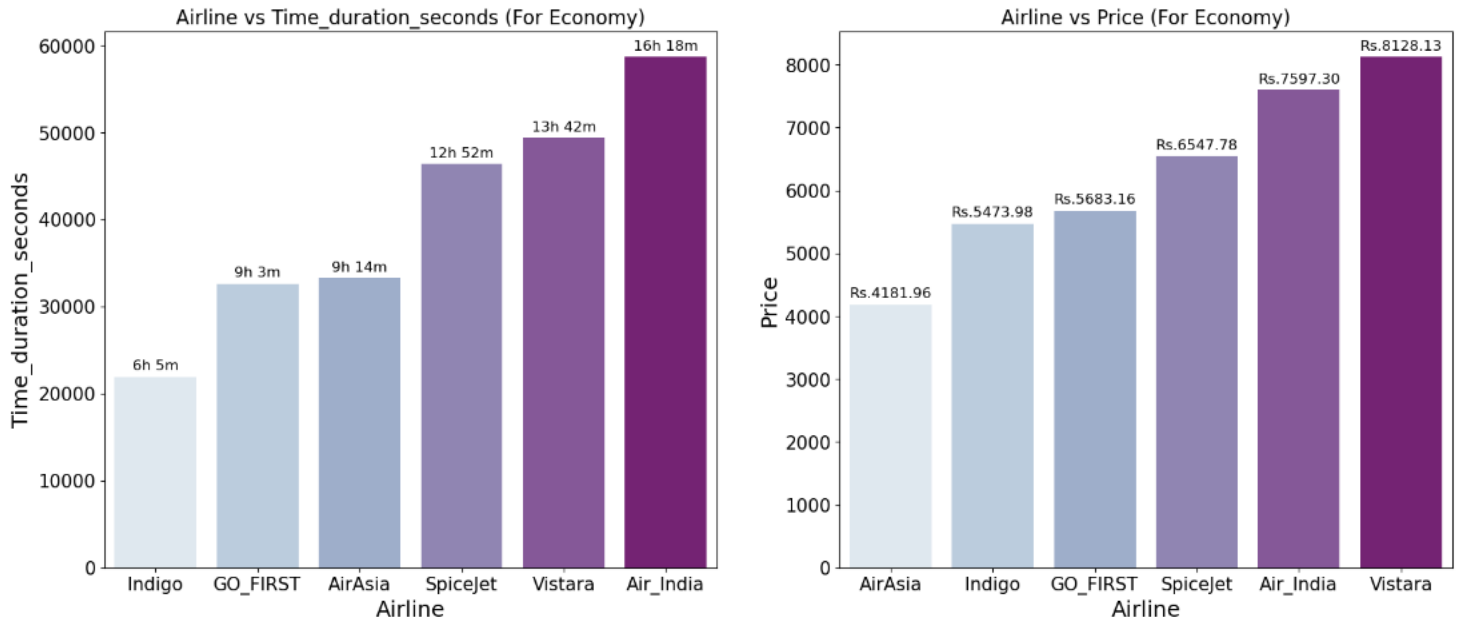
[95]:

```
# Create the plot
plot, axis = plt.subplots(nrows=1, ncols=2, figsize=(20, 8))
# Plotting the barplot
sns.barplot(x="airline", y="Time_duration_seconds", data=Time_duration_seconds_chart, ax=axis[0], palette='BuPu')
# Set labels and title
axis[0].set_xlabel('Airline', fontsize=18)
axis[0].set_ylabel('Time_duration_seconds', fontsize=18)
axis[0].tick_params(axis='both', labelsize=15)
axis[0].set_title('Airline vs Time_duration_seconds (For Economy)', fontsize=16)
# axis[0].legend(title='Total Stops', title_fontsize='16', fontsize='15', loc='upper left')

# Annotating each bar with its height
def seconds_to_hours_minutes(seconds):
    hours = int(seconds // 3600)
    minutes = int((seconds % 3600) // 60)
    return f'{hours}h {minutes}m'

for p in axis[0].patches:
    height = p.get_height()
    axis[0].annotate(seconds_to_hours_minutes(height),
                     (p.get_x() + p.get_width() / 2., p.get_height()),
                     ha = 'center', va = 'center',
                     xytext = (0, 9),
                     textcoords = 'offset points',
                     fontsize=12, color='black')


# Plotting the barplot
sns.barplot(x="airline", y="price", data=price_chart, ax=axis[1], palette='BuPu')
# Set labels and title
axis[1].set_xlabel('Airline', fontsize=18)
axis[1].set_ylabel('Price', fontsize=18)
axis[1].tick_params(axis='both', labelsize=15)
axis[1].set_title('Airline vs Price (For Economy)', fontsize=16)
# axis[1].legend(title='Total Stops', title_fontsize='16', fontsize='15', loc='upper left')
for p in axis[1].patches:
    axis[1].annotate('Rs.{:.2f}'.format(p.get_height()),
                     (p.get_x() + p.get_width() / 2., p.get_height()),
                     ha = 'center', va = 'center',
                     xytext = (0, 9),
                     textcoords = 'offset points',
                     fontsize=12, color='black')
# Show the plot
plt.show()
```

Airline vs Time_duration_seconds (For Economy)

Airline vs Price (For Economy)

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



[96]:
```
dataCopy['class'].unique()
```

[96]:
```
array(['Economy', 'Business'], dtype=object)
```

[97]:
```
df22 = dataCopy
```

[98]:
```
# For Business....
df12 = df22[df22['class'] == 'Business']
mean_duration_price_df = df12.groupby('airline').agg({
    'Time_duration_seconds': 'mean',
    'price': 'mean'
}).reset_index().round(2)
mean_duration_price_df
```

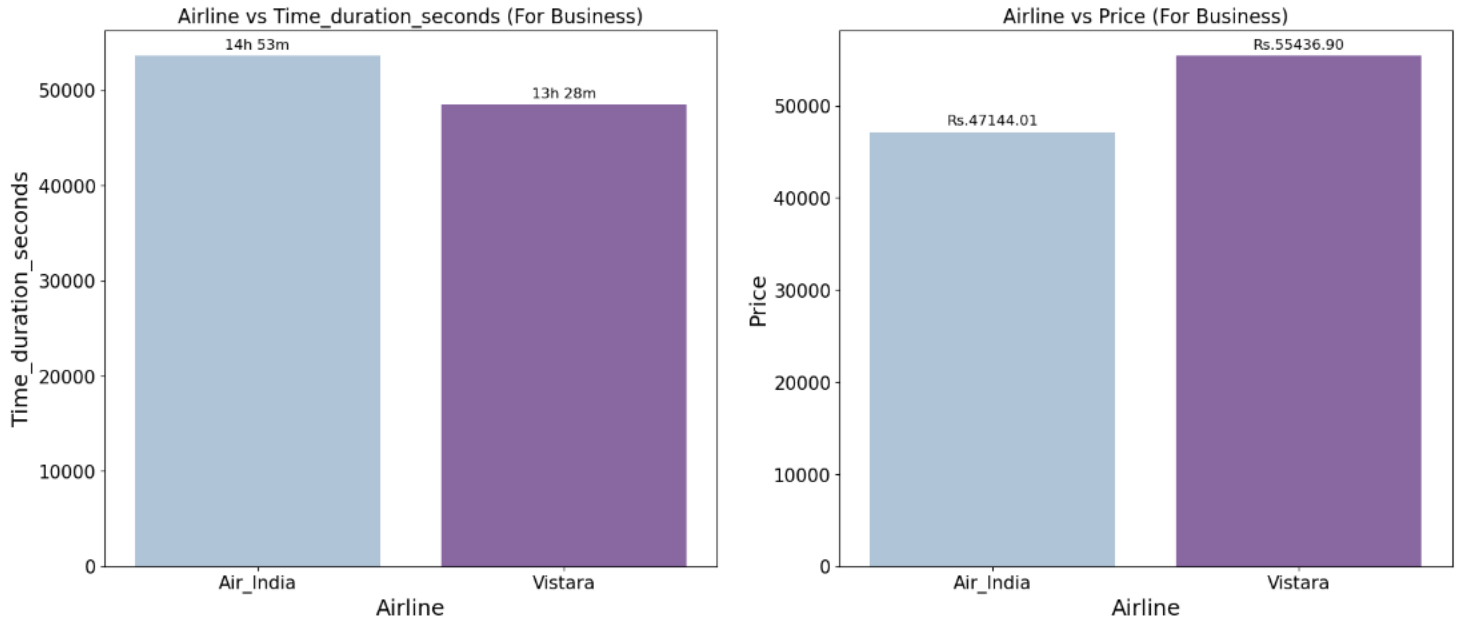
[98]:

	airline	Time_duration_seconds	price
0	Air_India	53626.26	47144.01

[100]:

```
# Create the plot
plot, axis = plt.subplots(nrows=1, ncols=2, figsize=(20, 8))
# Plotting the barplot
sns.barplot(x="airline", y="Time_duration_seconds", data=mean_duration_price_df, ax=axis[0], palette='BuPu')
# Set labels and title
axis[0].set_xlabel('Airline', fontsize=18)
axis[0].set_ylabel('Time_duration_seconds', fontsize=18)
axis[0].tick_params(axis='both', labelsize=15)
axis[0].set_title('Airline vs Time_duration_seconds (For Business)', fontsize=16)
# axis[0].legend(title='Total Stops', title_fontsize='16', fontsize='15', loc='upper left')
# Annotating each bar with its height
def seconds_to_hours_minutes(seconds):
    hours = int(seconds // 3600)
    minutes = int((seconds % 3600) // 60)
    return f'{hours}h {minutes}m'

for p in axis[0].patches:
    height = p.get_height()
    axis[0].annotate(seconds_to_hours_minutes(height),
                     (p.get_x() + p.get_width() / 2., p.get_height()),
                     ha = 'center', va = 'center',
                     xytext = (0, 9),
                     textcoords = 'offset points',
                     fontsize=12, color='black')


# Plotting the barplot
sns.barplot(x="airline", y="price", data=mean_duration_price_df, ax=axis[1], palette='BuPu')
# Set labels and title
axis[1].set_xlabel('Airline', fontsize=18)
axis[1].set_ylabel('Price', fontsize=18)
axis[1].tick_params(axis='both', labelsize=15)
axis[1].set_title('Airline vs Price (For Business)', fontsize=16)
# axis[1].legend(title='Total Stops', title_fontsize='16', fontsize='15', loc='upper left')
for p in axis[1].patches:
    axis[1].annotate('Rs.{:.2f}'.format(p.get_height()),
                     (p.get_x() + p.get_width() / 2., p.get_height()),
                     ha = 'center', va = 'center',
                     xytext = (0, 9),
                     textcoords = 'offset points',
                     fontsize=12, color='black')
# Show the plot
plt.show()
```

```
                    fontsize=12, color='black')
# Show the plot
plt.show()
```



[89]:
```
stops = dataCopy.groupby(['airline'], as_index=False)['Total_stops'].mean()
stops = stops.sort_values(by=['Total_stops'], ascending = False)
stops
```

[89]:

	airline	Total_stops
0	AirAsia	0.988323
5	Vistara	0.970107
1	Air_India	0.963854
2	GO_FIRST	0.878409
3	Indigo	0.757026
4	SpiceJet	0.726789

[107]:
```
# Group by 'Total_stops' and 'class', then sum the 'price'
grouped_data123 = dataCopy.groupby(['Total_stops', 'class'], as_index=False)['price'].sum()
grouped_data123
```

[107]:

	Total_stops	class	price
0	0	Business	225686039
1	0	Economy	116009482
2	1	Business	4651754576
3	1	Economy	1179848760
4	2	Business	76016995
5	2	Economy	111563051

[119]:
```
# Plotting the above data...
plot, axis = plt.subplots(nrows = 1, ncols = 1, figsize = (10, 5))

# Plotting barplot...
original_palette = sns.color_palette('BuPu')
# Reverse the palette
reversed_palette = original_palette[::-1]
sns.barplot(x = "Total_stops", y = "price", data = grouped_data123, hue = 'class', ax = axis,
            palette = reversed_palette)
axis.set_xlabel('Stops', fontsize=14)
axis.set_ylabel('Price', fontsize=14)
axis.tick_params(axis='both', labelsize=13)
axis.legend(title='Class', title_fontsize='16', fontsize='15', loc='upper right')
axis.set_title('Mean Stops of Airlines', fontsize = 16)
plt.show()
```

```
axis.set_title( Mean Stops of Airlines , fontsize = 16)
plt.show()
```