Project: EDA on Flight Fare and predict Flight Fare Using Machine Learning

Submitted by,

JINIYA XAVIER

Under the guidance of,

SHALINI KUMARI

Enrol no: EBEON1222624771

Abstract

Flight fare refers to the price that passengers pay for air travel. The price of airline ticket changes frequently nowadays and there is plenty of difference. Price change keeps happening within few hours for the identical flight. It is a dynamic and fluctuating aspect of the aviation industry, influenced by various factors. These factors include the distance between the origin and destination cities, seasonal variations, supply and demand dynamics, airline competition, cabin class preferences, booking timing, and additional fees and taxes. The fare for a flight can vary significantly depending on these factors.

Airlines utilize sophisticated pricing systems and revenue management techniques to optimize fares and maximize profitability. Passengers often seek to find the best deals by comparing fares, considering different airlines, travel dates, and booking in advance.

Understanding the complexities of flight fare can help travellers make informed decisions and plan their trips effectively while considering their budget and preferences. Booking timing can impact fares, with early bookings often resulting in lower prices. Market competition between airlines on specific routes can lead to fare fluctuations.

So I am going to analyze the features that affect the flight fare and predict the price of flight using different machine learning algorithms.

CONTENTS

TABLE OF CONTENTS

CHAPTER- 1

1.1 INTRODUCTION

In the present scenario, flight fare prediction has become increasingly important in the travel industry. Understanding the factors behind flight fare variations is crucial for both travellers and industry professionals aiming to optimize pricing strategies.

Travelers are actively seeking reliable fare predictions to plan their trips effectively and optimize their budgets. Airlines and travel agencies are also eager to utilize advanced analytical models and machine learning algorithms to forecast fares and adjust pricing strategies accordingly.

Overall, in the current dynamic landscape of air travel, accurate and reliable flight fare prediction is of utmost importance for both travelers and industry stakeholders. It enables informed decision-making, helps optimize travel budgets, and supports airlines in effectively managing revenue and capacity.

1.2 PROBLEM STATEMENT

The basic idea of analyzing flight fare data is to gain insights into the factors influencing the pricing of flights for different routes and airlines. With the continuous growth of the aviation industry and increasing demand for air travel, it has become essential to understand the dynamics of flight fares and provide accurate predictions for travellers.

In today's fast-paced world, air travel has become a vital mode of transportation. However, with the multitude of airlines, routes, and travel options available, it has become challenging for passengers to determine the most cost-effective flights for their desired destinations. The aim of this analysis is to develop a model that can accurately predict flight fares based on various factors, enabling travelers to make informed decisions when booking flights.

STUDY OF EXISTING SYSTEM

1. EDA on Flight Data

   - Author: **RAHUL KRISHNAN M** -In this project, he did the analysis part in 2 ways that is univariate analysis and bivariate analysis.

2. Flight Fare Prediction | EDA | Linear Regression

   - Author: **TUNAHAN ULUSOY** -In this project, he did the analysis part and applied machine learning algorithm to predict the price of a airlines.

IDENTIFICATION GAPS IN EXISTING SYSTEMS

1 EDA on Flight Data

   - Author-**RAHUL KRISHNAN M** - Analysis Part should be more in detail.

2. Flight Price Prediction | EDA | Linear Regression

   - Author-**TUNAHAN ULUSOY** - In this project, he applied only one machine learning algorithms to predict the price of a airlines.

DISCUSSED ON PROPOSED SOLUTION

1. EDA on Flight Data

   - Author-**RAHUL KRISHNAN M-** Analysis Part should be more in detail like which include the variation in price when tickets are bought in just few days before departure, variation in price with departure time and arrival time in both business and economy class, Average ticket price in each route and so on.

2. Flight Price Prediction | EDA | Linear Regression

    - Author-**TUNAHAN ULUSOY-** We can apply additional three more Machine learning algorithms Random Forest Regressor, Decision tree Regressor and Extra Tree Regressor and compare the best model by using the accuracy of each model.

Tools/Technology used to implement proposed solution:

- Python
- Numpy
- Pandas
- Matplotlib
- Seaborn
- Sklearn
- Jupyter Notebook

Chapter 2

Features & Predictor

- **AIRLINE** – The name of the airline company is stored in the airline column. It is a categorical feature having 6 different airlines.
- **FLIGHT:** Flight stores information regarding the plane's flight code. It is a categorical feature.
- **SOURCE_CITY**: City from which the flight takes off. It is a categorical feature having 6 unique cities.
- **DEPARTURE_TIME**: This is a derived categorical feature obtained created by grouping time periods into bins. It stores information about the departure time and have 6 unique time labels.
- **STOPS**: A categorical feature with 3 distinct values that stores the number of stops between the source and destination cities.
- **ARRIVAL TIME**: This is a derived categorical feature created by grouping time intervals into bins. It has six distinct time labels and keeps information about the arrival time.
- **DESTINATION CITY**: City where the flight will land. It is a categorical feature having 6 unique cities.
- **CLASS**: A categorical feature that contains information on seat class; it has two distinct values: Business and Economy.
- **DURATION**: A continuous feature that displays the overall amount of time it takes to travel between cities in hours.
- **DAYS LEFT**: This is a derived characteristic that is calculated by subtracting the trip date by the booking date.
- **PRICE**: Target variable stores information of the ticket price.

NOTE:

- There are 300153 rows and 12 columns
- Numerical – 4 columns, which is quantitative data that can be measured.
- String – 8 columns which is Categorical data that has an order to it.

Chapter 3

Methodology

The dataset used in this project was sourced from Kaggle, a well-known platform hosting a diverse range of datasets. Kaggle datasets are typically of high quality, but it is essential to verify their cleanliness before conducting any analysis. The datasets which were collected from Kaggle website was clean data. In this project, performed several data pre-processing steps to prepare the dataset for analysis. Converted integer value to float to ensure compatibility with our analytical techniques. And transformed the all column heading to uppercase for consistency and ease of analysis. Next, checked for duplicates within the dataset. Additionally, created a new column based on specific criteria to provide additional insights for our analysis. And dropped unwanted column that were deemed irrelevant for our research objectives. These pre-processing steps were crucial in ensuring data quality and usability, setting a strong foundation for our subsequent analysis and findings.

Machine Learning Algorithms

Linear Regression:

Linear Regression is a supervised machine learning algorithm that is primarily used for regression tasks. It aims to model the relationship between a dependent variable and independent variables by predicting a target value. Regression models are widely employed for analyzing the association between variables and making forecasts. The choice of regression model depends on factors such as the nature of the relationship between the dependent and independent variables under consideration and the number of independent variables utilized in the analysis. The dependent variable in a regression is often referred to as the outcome variable, criterion variable, endogenous variable, or regressand, while the independent variables can be referred to as exogenous variables, predictor variables, or regressors.

Decision Tree Regressor:

Decision Trees are a type of non-parametric supervised learning technique employed for both classification and regression tasks. The objective of a decision tree is to construct a model that can predict the value of a target variable by learning straightforward decision rules derived from the features of the data. Each decision tree can be visualized as a collection of simple decision rules that approximate the target variable with piecewise constant values, allowing for intuitive interpretation of the model's behaviour.

Random Forest Regressor:

A random forest is a meta-estimator algorithm that builds multiple decision trees for classification. It improves predictive accuracy and mitigates overfitting by fitting these decision trees on different subsets of the dataset and then averaging their predictions. This ensemble approach of combining multiple decision trees enhances the overall performance and robustness of the random forest algorithm.

Extra Tree Regressor:

The Extra-Trees Regressor is a meta-estimator algorithm that utilizes randomized decision trees, also known as extra-trees, to improve predictive accuracy and combat overfitting. It fits multiple extra-trees on different subsets of the dataset and employs averaging to enhance the overall performance. By introducing randomness in the tree-building process, the Extra-Trees Regressor reduces the risk of overfitting and provides more robust predictions. The algorithm’s ability to combine the predictions from multiple randomized decision trees makes it a valuable tool for regression tasks.

Implementation Steps:

The project is implemented using the Python programming language, specifically in Anaconda Navigator's Jupyter Notebook. Jupyter Notebook offers faster execution compared others when implementing machine learning algorithms. One of the main advantages of Jupyter Notebook is its capability for data visualization and plotting various graphs. The implementation steps can be summarized as follows: a) Dataset collection,

b) Importing Libraries: NumPy, Pandas, Matplotlib, Seaborn, and Sklearn,

c) Exploratory data analysis to gain insights about the data,

d) Data cleaning and pre-processing, where we checked for null values.

Additionally, performed feature engineering on the dataset by converting categorical variables into numerical variables using functions provided by the Pandas library.

LIBRARIES:

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

from sklearn.preprocessing  import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score

from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import ExtraTreesRegressor
from sklearn.tree import DecisionTreeRegressor

import warnings warnings.filterwarnings('ignore')
```

Reading the Dataset and assigning that to the variable df:

df= pd.read_csv(“flight.csv”)

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
...	...	...	...	...	...	...	...	...	...	...	...	...
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

Find out how many rows and columns present in the dataset

```
[4]: df.shape
t[4]: (300153, 12)
```

Showing all the columns headings in the dataset

```
In [5]: df.columns

Out[5]: Index(['Unnamed: 0', 'airline', 'flight', 'source_city', 'departure_time',
               'stops', 'arrival_time', 'destination_city', 'class', 'duration',
               'days_left', 'price'],
              dtype='object')
```

Showing all the features(columns) with its corresponding datatypes

In [6]:
```
df.dtypes
```

Out[6]:
```
Unnamed: 0            int64
airline              object
flight               object
source_city          object
departure_time       object
stops                object
arrival_time         object
destination_city     object
class                object
duration            float64
days_left             int64
price                 int64
dtype: object
```

Showing the information of all the features

In [7]:
```
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
```

To check the Null Values for each column

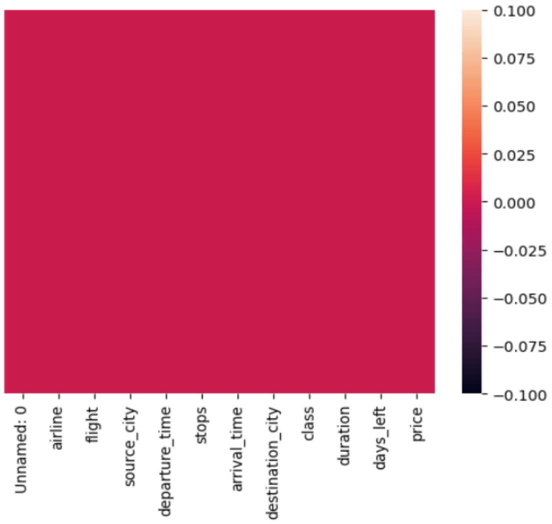
```
In [8]: df.isnull().values.any()

Out[8]: False

In [9]: df.isnull().sum()

Out[9]: 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 [10]: sns.heatmap(df.isnull(),yticklabels=False)
         plt.show()
```



Combining columns and creating a new column

```
In [11]: df["Route"] = df.source_city + '-' + df.destination_city

In [12]: df.shape

Out[12]: (300153, 13)
```

Dropping the unwanted feature

```
In [13]: df.drop(['Unnamed: 0'],axis=1,inplace=True)

In [14]: df.shape

Out[14]: (300153, 12)
```

Converting column heading to upper case

```
In [15]: df.columns=df.columns.str.upper()

In [16]: df.columns

Out[16]: Index(['AIRLINE', 'FLIGHT', 'SOURCE_CITY', 'DEPARTURE_TIME', 'STOPS',
                'ARRIVAL_TIME', 'DESTINATION_CITY', 'CLASS', 'DURATION', 'DAYS_LEFT',
                'PRICE', 'ROUTE'],
               dtype='object')
```

- All column heading are converted into uppercase by using str.upper function.

Checking for the length of duplicate

```
In [17]: len(df[df.duplicated()])

Out[17]: 0
```

Converting the datatype

```
In [18]: df[['PRICE']] = df[['PRICE']].astype('float64')

In [19]: df.dtypes

Out[19]: AIRLINE              object
         FLIGHT               object
         SOURCE_CITY          object
         DEPARTURE_TIME       object
         STOPS                object
         ARRIVAL_TIME         object
         DESTINATION_CITY     object
         CLASS                object
         DURATION            float64
         DAYS_LEFT             int64
         PRICE               float64
         ROUTE                object
         dtype: object
```

Finding unique value in the features

```
In [20]: df.nunique()

Out[20]: 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
         ROUTE                  30
         dtype: int64
```

Getting first 5 rows

In [22]:
```
df.head()
```

Out[22]:

	AIRLINE	FLIGHT	SOURCE_CITY	DEPARTURE_TIME	STOPS	ARRIVAL_TIME	DESTINATION_CITY	CLASS	DURATION	DAYS_LEFT	PRICE	ROUTE
0	SpiceJet	SG-8709	Delhi	Evening	zero	Night	Mumbai	Economy	2.17	1	5953.0	Delhi-Mumbai
1	SpiceJet	SG-8157	Delhi	Early_Morning	zero	Morning	Mumbai	Economy	2.33	1	5953.0	Delhi-Mumbai
2	AirAsia	I5-764	Delhi	Early_Morning	zero	Early_Morning	Mumbai	Economy	2.17	1	5956.0	Delhi-Mumbai
3	Vistara	UK-995	Delhi	Morning	zero	Afternoon	Mumbai	Economy	2.25	1	5955.0	Delhi-Mumbai
4	Vistara	UK-963	Delhi	Morning	zero	Morning	Mumbai	Economy	2.33	1	5955.0	Delhi-Mumbai

Getting last 5 rows

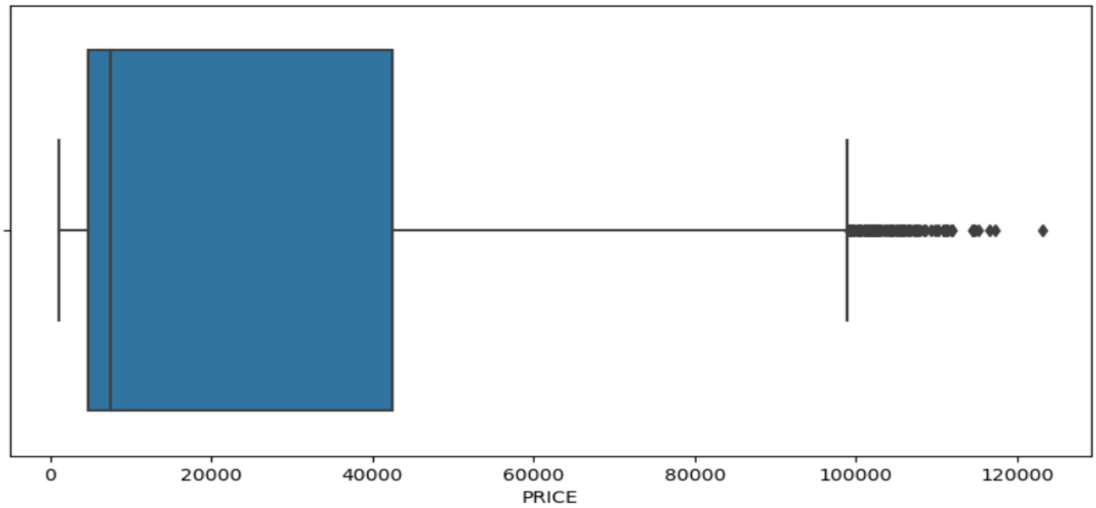
In [23]:
```
df.tail()
```

Out[23]:

	AIRLINE	FLIGHT	SOURCE_CITY	DEPARTURE_TIME	STOPS	ARRIVAL_TIME	DESTINATION_CITY	CLASS	DURATION	DAYS_LEFT	PRICE	ROUT
300148	Vistara	UK-822	Chennai	Morning	one	Evening	Hyderabad	Business	10.08	49	69265.0	Chenna Hyderaba
300149	Vistara	UK-826	Chennai	Afternoon	one	Night	Hyderabad	Business	10.42	49	77105.0	Chenna Hyderaba
300150	Vistara	UK-832	Chennai	Early_Morning	one	Night	Hyderabad	Business	13.83	49	79099.0	Chenna Hyderaba
300151	Vistara	UK-828	Chennai	Early_Morning	one	Evening	Hyderabad	Business	10.00	49	81585.0	Chenna Hyderaba
300152	Vistara	UK-822	Chennai	Morning	one	Evening	Hyderabad	Business	10.08	49	81585.0	Chenna Hyderaba

Treating outliers

In [24]:
```
plt.figure(figsize=(10,5))
sns.boxplot(df['PRICE'])
plt.show()
```



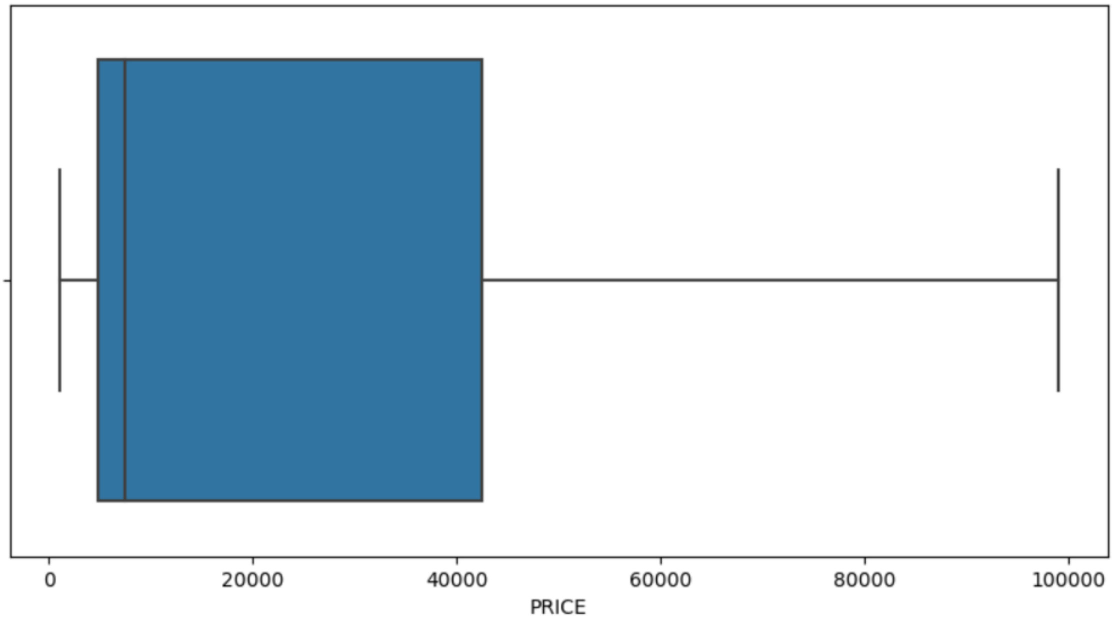
In [25]:
```
df=df[~(df['PRICE']>99100)]
```

- The code filters out any prices greater than 99100 from the original dataframe 'df'.

In [26]:
```
df.shape
```

Out[26]:
```
(300030, 12)
```

In [27]:
```
plt.figure(figsize=(10,5))
sns.boxplot(df['PRICE'])
plt.show()
```



Exploratory data analysis

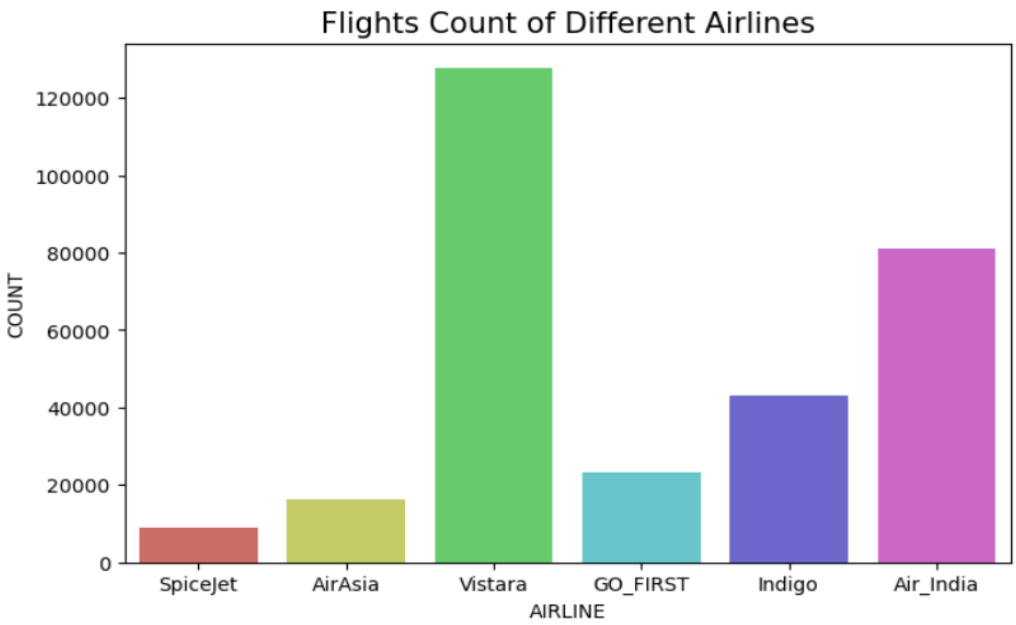
Analysing Airlines feature

In [28]:
```
df.AIRLINE.value_counts()
```

Out[28]:
```
Vistara      127736
Air_India     80892
Indigo        43120
GO_FIRST      23173
AirAsia       16098
SpiceJet       9011
Name: AIRLINE, dtype: int64
```

- In airline column there are 6 unique airlines

In [29]:
```
plt.figure(figsize=(8,5))
sns.countplot(df['AIRLINE'],palette='hls')
plt.title('Flights Count of Different Airlines',fontsize=15)
plt.xlabel('AIRLINE')
plt.ylabel('COUNT')
plt.show()
```



Observation:

- There 6 airlines details are mentioned in the dataset.
- Vistara becoming the most popular airline among others
- Second popular airline is Air India
- Spicejet is the least popular.

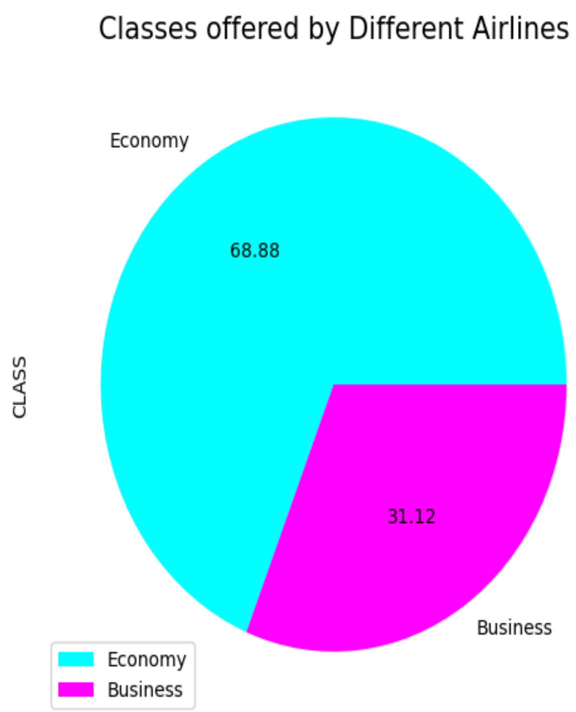
Analysing Class Feature

In [30]:
```
df.CLASS.value_counts()
```

Out[30]:
```
Economy     206666
Business     93364
Name: CLASS, dtype: int64
```

- In class column there are economic and business classes

In [31]:
```
plt.figure(figsize=(8,6))
df['CLASS'].value_counts().plot(kind='pie',textprops={'color':'black'},autopct='%.2f',cmap='cool')
plt.title('Classes offered by Different Airlines',fontsize=15)
plt.legend(['Economy','Business'])
plt.show()
```



Observation:

- Economy Class is the most common class among the six airlines.
- Economy class has a percentage of 68.88%
- Whereas business class has a percentage of 31.12%

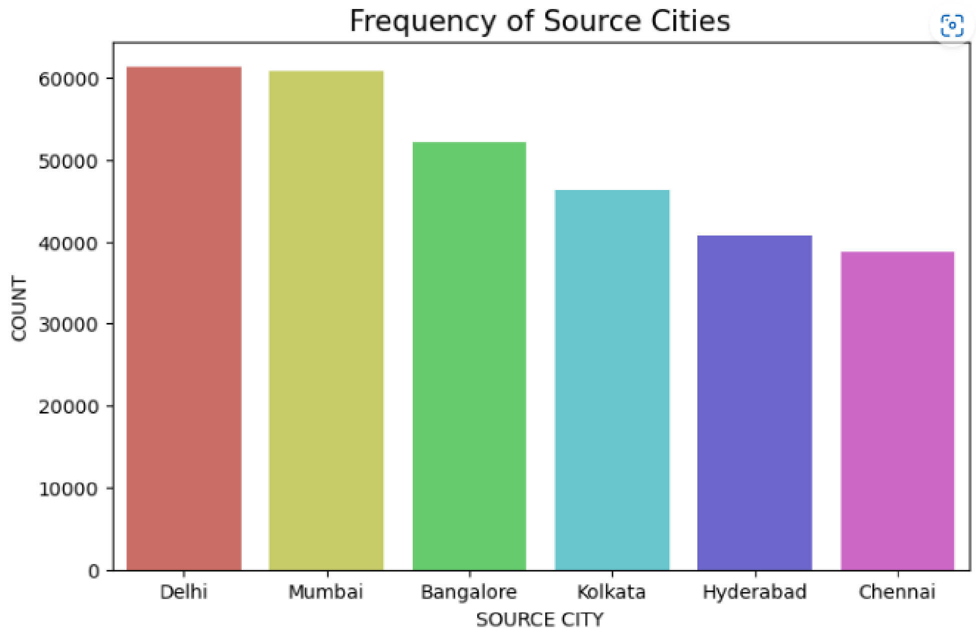
Analysing source city

```
In [32]: df.SOURCE_CITY.value_counts()
```

```
Out[32]: Delhi        61316
         Mumbai       60870
         Bangalore    52055
         Kolkata      46307
         Hyderabad    40804
         Chennai      38678
         Name: SOURCE_CITY, dtype: int64
```

- In the source city column there are 6 cities.

```
In [33]: plt.figure(figsize=(8,5))
         sns.countplot(df['SOURCE_CITY'],palette='hls')
         plt.title('Frequency of Source Cities',fontsize=15)
         plt.xlabel('SOURCE CITY')
         plt.ylabel('COUNT')
         plt.show()
```



Observation:

- Delhi appears as the source city in the dataset 61343 times. And the least appeared is Chennai
- After Delhi, Mumbai is the leading one

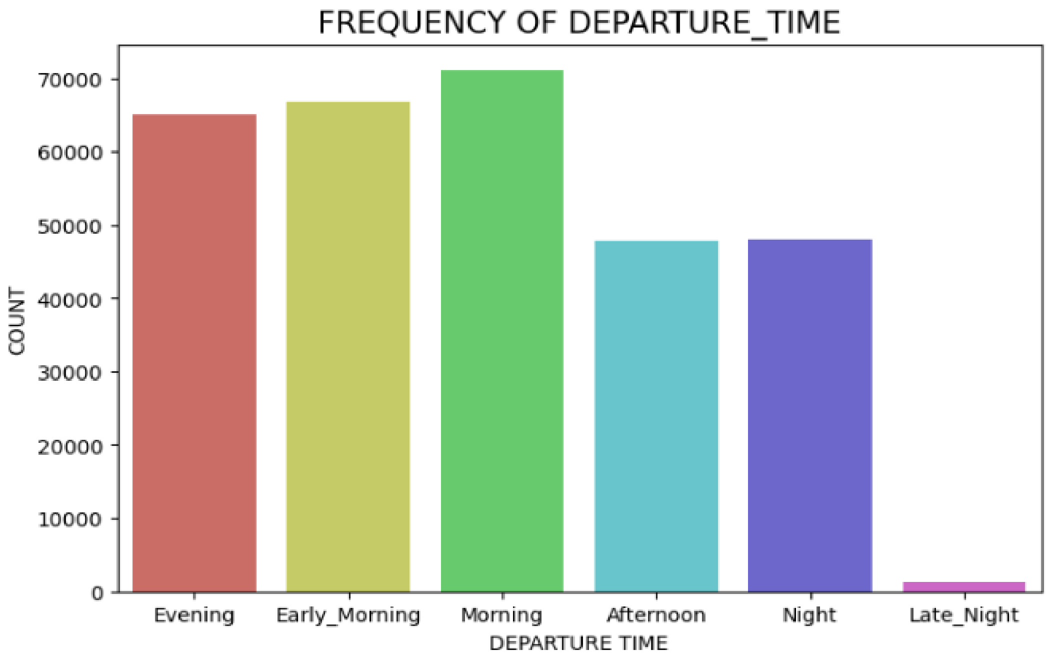
Analysing Departure Time

In [34]:
```
df.DEPARTURE_TIME.value_counts()
```

Out[34]:
```
Morning          71102
Early_Morning    66767
Evening          65067
Night            48000
Afternoon        47788
Late_Night        1306
Name: DEPARTURE_TIME, dtype: int64
```

- In the Departure time column, there are 6 timings mentioned.

In [35]:
```
plt.figure(figsize=(8,5))
sns.countplot(df['DEPARTURE_TIME'],palette='hls')
plt.title('FREQUENCY OF DEPARTURE_TIME',fontsize=15)
plt.xlabel('DEPARTURE TIME')
plt.ylabel('COUNT')
plt.show()
```



Observation:

- The above graph shows that, 71146 flights in the dataset that depart in the morning.
- It helps in understanding the patterns and preferences of flight departures based on the time of day.
- Only 1306 flights depart in the Late night

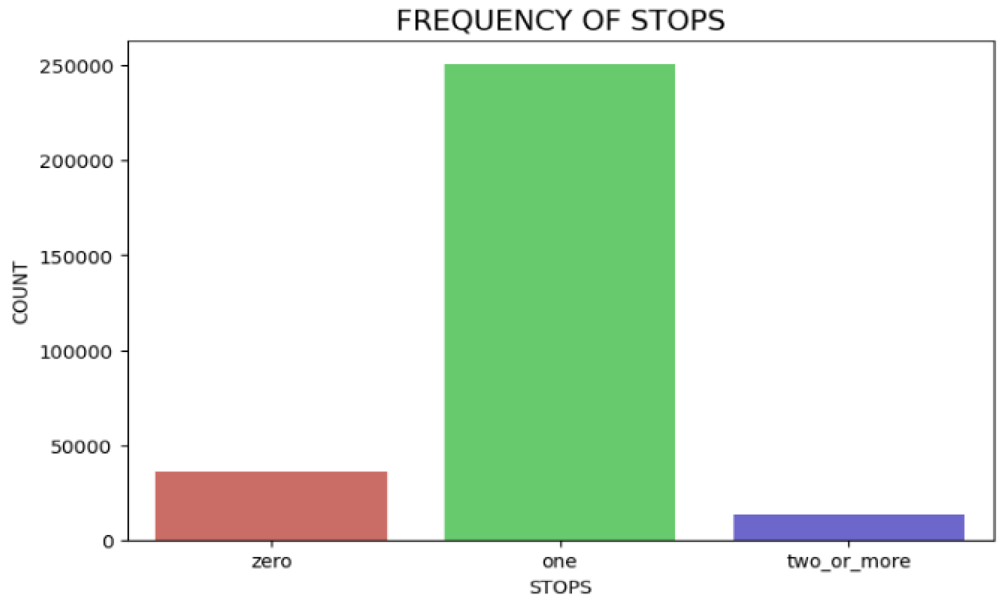
Analysing Stops

```
In [36]: df.STOPS.value_counts()

Out[36]: one            250768
         zero            36004
         two_or_more     13258
         Name: STOPS, dtype: int64
```

- In the Stops column, there are 3 values can be seen.

```
In [37]: plt.figure(figsize=(8,5))
         sns.countplot(df['STOPS'],palette='hls')
         plt.title('FREQUENCY OF STOPS',fontsize=15)
         plt.xlabel('STOPS')
         plt.ylabel('COUNT')
         plt.show()
```



Observation:

- Among the flights in the dataset, the most preferred option appears to be those with one stop. With a count of 250,863, flights with a single stop seem to be the preferred choice for travellers.
- The least preferred is the two or more stop category

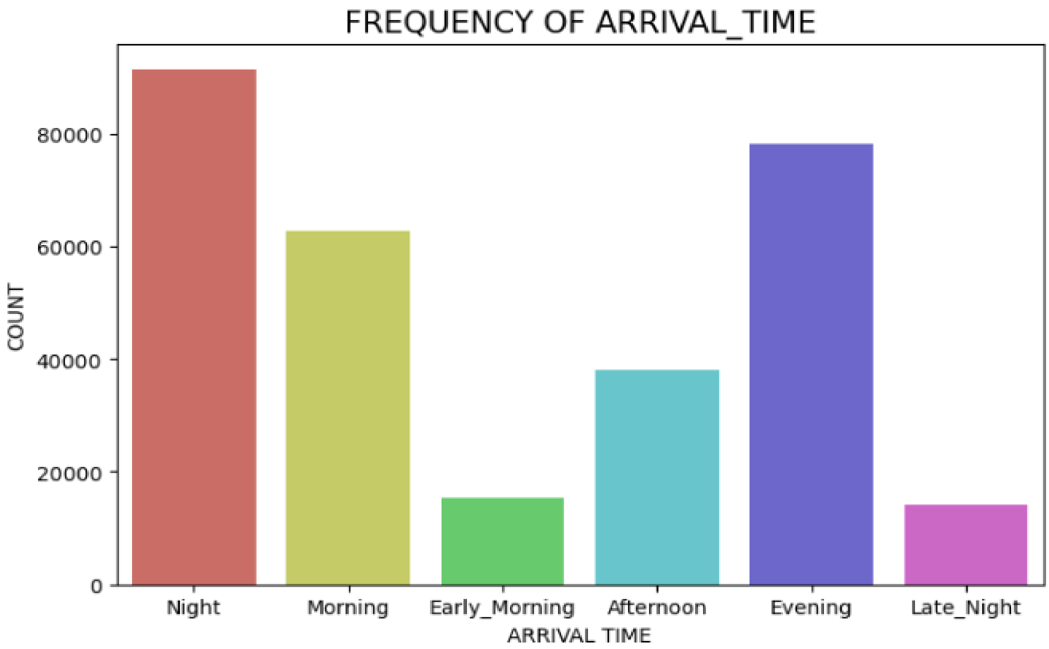
Analysing Arrival Time

```
In [38]: df.ARRIVAL_TIME.value_counts()

Out[38]: Night            91488
         Evening          78279
         Morning          62713
         Afternoon        38134
         Early_Morning    15415
         Late_Night       14001
         Name: ARRIVAL_TIME, dtype: int64
```

- In the arrival column, there are 6 unique values are avalaible.

```
In [39]: plt.figure(figsize=(8,5))
         sns.countplot(df['ARRIVAL_TIME'],palette='hls')
         plt.title('FREQUENCY OF ARRIVAL_TIME',fontsize=15)
         plt.xlabel('ARRIVAL TIME')
         plt.ylabel('COUNT')
         plt.show()
```



Observation:

- The arrival time distribution indicates that a significant number of flights arrive during the night, with a count of 91,538.
- Smaller number of flights arriving during early morning and late night.

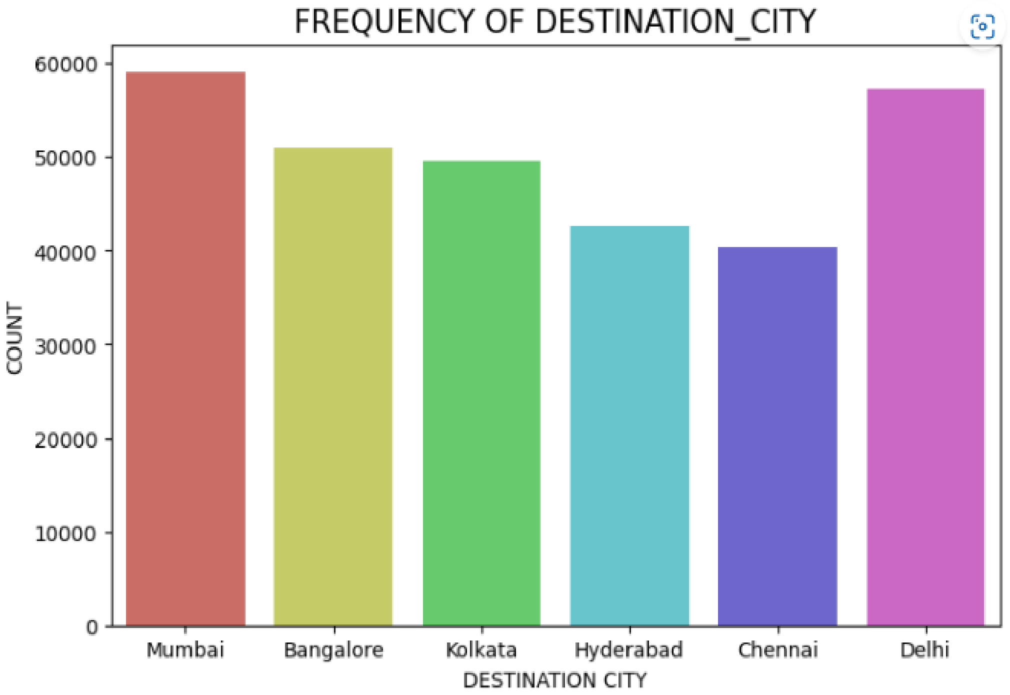
Analysing Destination City

```
In [40]: df.DESTINATION_CITY.value_counts()
```

```
Out[40]: Mumbai       59067
         Delhi        57339
         Bangalore    51042
         Kolkata      49511
         Hyderabad    42716
         Chennai      40355
         Name: DESTINATION_CITY, dtype: int64
```

- In the destination city also there are 6 unique cities

```
In [41]: plt.figure(figsize=(8,5))
         sns.countplot(df['DESTINATION_CITY'],palette='hls')
         plt.title('FREQUENCY OF DESTINATION_CITY',fontsize=15)
         plt.xlabel('DESTINATION CITY')
         plt.ylabel('COUNT')
         plt.show()
```



Observation:

- The data shows that Mumbai has the highest count of flights with a value of 59,097, indicating that it is a popular destination among the flights in the dataset.
- Delhi closely follows with a count of 57,360.
- Overall, these counts provide insights into the distribution of flights across different destination cities, indicating the relative popularity and demand for travel to these specific locations.

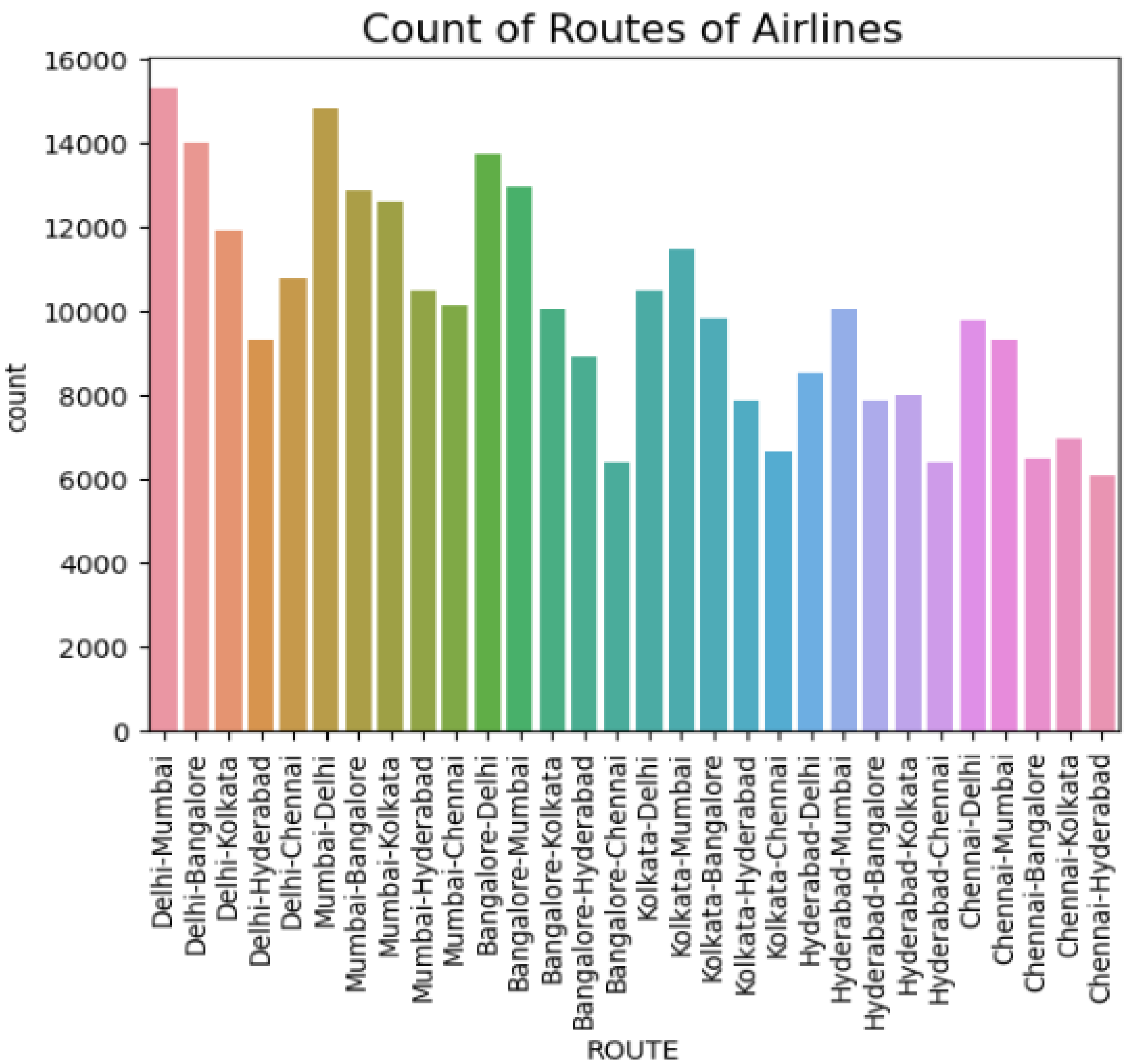
Analysing Routes

```
In [42]: df['ROUTE'].value_counts()
```

```
Out[42]: Delhi-Mumbai          15289
         Mumbai-Delhi          14808
         Delhi-Bangalore       14012
         Bangalore-Delhi       13755
         Bangalore-Mumbai      12935
         Mumbai-Bangalore      12867
         Mumbai-Kolkata        12601
         Delhi-Kolkata         11917
         Kolkata-Mumbai        11458
         Delhi-Chennai         10772
         Kolkata-Delhi         10488
         Mumbai-Hyderabad      10469
         Mumbai-Chennai        10125
         Hyderabad-Mumbai      10062
         Bangalore-Kolkata     10027
         Kolkata-Bangalore      9818
         Chennai-Delhi          9782
         Delhi-Hyderabad        9326
         Chennai-Mumbai         9323
         Bangalore-Hyderabad    8928
         Hyderabad-Delhi        8506
         Hyderabad-Kolkata      7987
         Kolkata-Hyderabad      7890
         Hyderabad-Bangalore    7854
         Chennai-Kolkata        6979
         Kolkata-Chennai        6653
         Chennai-Bangalore      6491
         Bangalore-Chennai      6410
         Hyderabad-Chennai      6395
         Chennai-Hyderabad      6103
         Name: ROUTE, dtype: int64
```

- These are the different routes mentioned in the dataset.

```python
sns.countplot(x="ROUTE", data=df)
plt.xticks(rotation=90)
plt.title("Count of Routes of Airlines",fontsize=15)
plt.show()
```



Observation:

- Based on the count plot, it can be observed that the Delhi-Mumbai route has the highest frequency of flights compared to other routes. This indicates that there is a higher demand for flights between Delhi and Mumbai.
- Conversely, the Mumbai-Delhi route also has a high frequency of flights, indicating a significant travel demand in the opposite direction.

Analysing variation of price with different features

Analysing variation in price with different Airlines

```
In [44]: df.groupby("AIRLINE")["PRICE"].describe()
```

Out[44]:

AIRLINE	count	mean	std	min	25%	50%	75%	max
AirAsia	16098.0	4091.072742	2824.055172	1105.0	2361.0	3276.0	4589.0	31917.0
Air_India	80892.0	23507.019112	20905.116909	1526.0	5623.0	11520.0	45693.0	90970.0
GO_FIRST	23173.0	5652.007595	2513.865560	1105.0	4205.0	5336.0	6324.0	32803.0
Indigo	43120.0	5324.216303	3268.894831	1105.0	3219.0	4453.0	6489.0	31952.0
SpiceJet	9011.0	6179.278881	2999.630406	1106.0	4197.0	5654.0	7412.0	34158.0
Vistara	127736.0	30325.046361	25545.309562	1714.0	6412.0	15512.0	55377.0	98972.0

- Vistara and Air India airlines have maximum price when compared to rest. Other airlines have somewhat equal price range but Air Asia maintains low price.So Vistara is the most expensive expensive among this.

Taking the statistical analysis of individual airlines

In [46]:
```
Vistara_stats = df.loc[df['AIRLINE'] == 'Vistara'].describe()
Vistara_stats
```

Out[46]:

	DURATION	DAYS_LEFT	PRICE
count	127736.000000	127736.000000	127736.000000
mean	13.325598	25.911474	30325.046361
std	6.778365	13.631443	25545.309562
min	1.000000	1.000000	1714.000000
25%	8.500000	14.000000	6412.000000
50%	12.500000	26.000000	15512.000000
75%	17.000000	38.000000	55377.000000
max	47.080000	49.000000	98972.000000

In [47]:
```
AirIndia_stats = df.loc[df['AIRLINE'] == 'Air_India'].describe()
AirIndia_stats
```

Out[47]:

	DURATION	DAYS_LEFT	PRICE
count	80892.000000	80892.000000	80892.000000
mean	15.504235	25.497466	23507.019112
std	7.613365	13.725776	20905.116909
min	1.000000	1.000000	1526.000000
25%	10.080000	14.000000	5623.000000
50%	15.000000	26.000000	11520.000000
75%	21.670000	37.000000	45693.000000
max	49.830000	49.000000	90970.000000

In [48]:
```
spicejet_stats = df.loc[df['AIRLINE'] == 'SpiceJet'].describe()
spicejet_stats
```

Out[48]:

	DURATION	DAYS_LEFT	PRICE
count	9011.000000	9011.000000	9011.000000
mean	12.579767	24.122850	6179.278881
std	8.927157	13.658816	2999.630406
min	1.000000	1.000000	1106.000000
25%	2.830000	12.000000	4197.000000
50%	12.000000	23.000000	5654.000000
75%	21.080000	36.000000	7412.000000
max	27.920000	49.000000	34158.000000

In [49]:
```python
GoFirst_stats = df.loc[df['AIRLINE'] == 'GO_FIRST'].describe()
GoFirst_stats
```

Out[49]:

	DURATION	DAYS_LEFT	PRICE
count	23173.000000	23173.000000	23173.000000
mean	8.755380	27.430415	5652.007595
std	4.015146	12.385957	2513.865560
min	1.000000	1.000000	1105.000000
25%	6.000000	17.000000	4205.000000
50%	8.830000	27.000000	5336.000000
75%	11.750000	38.000000	6324.000000
max	22.500000	49.000000	32803.000000

In [50]:
```python
Indigo_stats = df.loc[df['AIRLINE'] == 'Indigo'].describe()
Indigo_stats
```

Out[50]:

	DURATION	DAYS_LEFT	PRICE
count	43120.000000	43120.000000	43120.000000
mean	5.795197	26.264309	5324.216303
std	2.769322	13.717115	3268.894831
min	0.830000	1.000000	1105.000000
25%	2.920000	15.000000	3219.000000
50%	6.000000	27.000000	4453.000000
75%	7.750000	38.000000	6489.000000
max	15.420000	49.000000	31952.000000

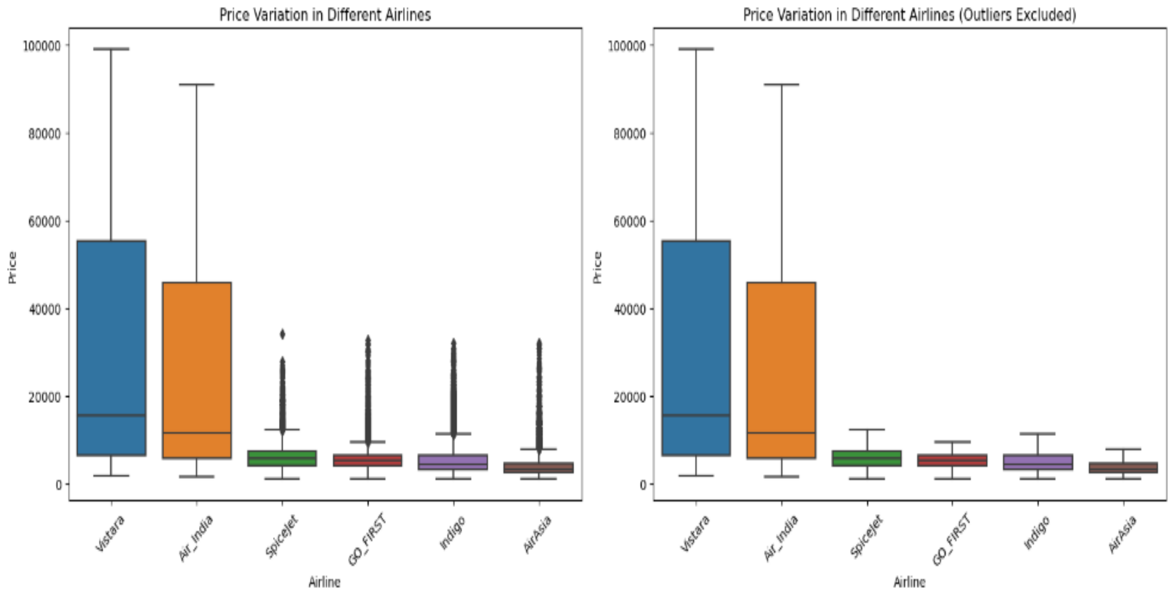
In [51]:
```python
AirAsia_stats = df.loc[df['AIRLINE'] == 'AirAsia'].describe()
AirAsia_stats
```

Out[51]:

	DURATION	DAYS_LEFT	PRICE
count	16098.000000	16098.000000	16098.000000
mean	8.941714	27.735184	4091.072742
std	4.173152	12.889223	2824.055172
min	0.920000	1.000000	1105.000000
25%	5.920000	17.000000	2361.000000
50%	9.330000	28.000000	3276.000000
75%	11.830000	39.000000	4589.000000
max	19.580000	49.000000	31917.000000

```
In [52]: fig, axs = plt.subplots(1, 2, figsize=(16, 6))
sns.boxplot(data=df.sort_values('PRICE',ascending=False), x='AIRLINE', y='PRICE', ax=axs[0])
axs[0].set_xlabel('Airline')
axs[0].set_ylabel('Price')
axs[0].set_title('Price Variation in Different Airlines')
axs[0].tick_params(axis='x', rotation=45)

sns.boxplot(data=df.sort_values('PRICE',ascending=False), x='AIRLINE', y='PRICE', showfliers=False, ax=axs[1])
axs[1].set_xlabel('Airline')
axs[1].set_ylabel('Price')
axs[1].set_title('Price Variation in Different Airlines (Outliers Excluded)')
axs[1].tick_params(axis='x', rotation=45)
plt.tight_layout()
plt.show()
```



Observation:

- As we can see Vistara has Maximum Price range.
- Vistara and Air India airlines have maximum price when compared to rest
- SpiceJet , AirAsia , GO_First and Indigo has some what equal prices.

Analysing variation in price with different class

```
In [53]: df.groupby("CLASS")["PRICE"].describe()
```

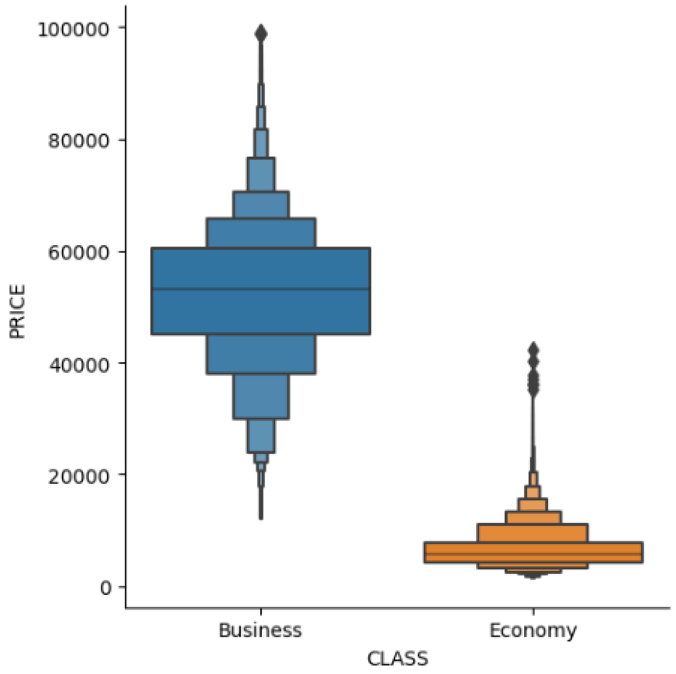
Out[53]:

CLASS	count	mean	std	min	25%	50%	75%	max
Business	93364.0	52471.444572	12838.029183	12000.0	45185.0	53164.0	60396.0	98972.0
Economy	206666.0	6572.342383	3743.519517	1105.0	4173.0	5772.0	7746.0	42349.0

```
In [54]: plt.figure(figsize=(15,15))
         sns.catplot(x='CLASS',y='PRICE',data=df.sort_values('PRICE',ascending=False),kind='boxen')
```

```
Out[54]: <seaborn.axisgrid.FacetGrid at 0x1bd8097e700>

<Figure size 1500x1500 with 0 Axes>
```



Observation:

- Ticket Price is Maximum for Business Class When compared to Economy Class
- Most of the passengers prefer business class. In business class most of passengers are prefer 40000-60000 range flight price.
- Economy class very less passengers prefer 40000-60000 price rate. Economy class most of passengers are prefer <20000 price range.

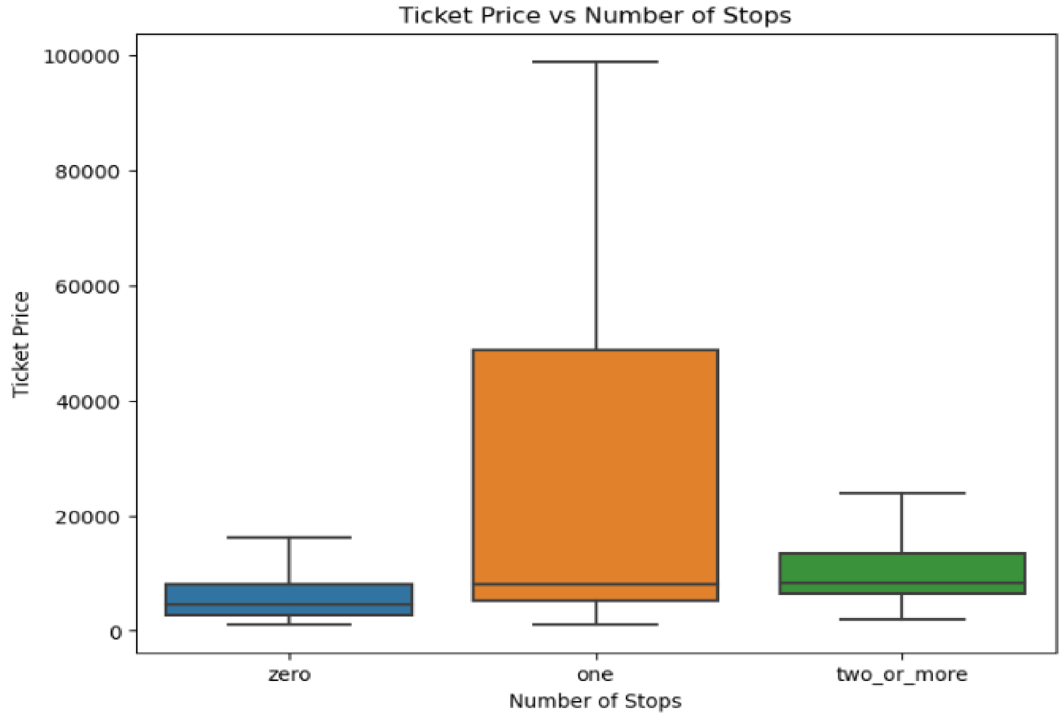
Analysing variation in Ticket price varies with no of stops

In [55]:
```
df.groupby("STOPS")["PRICE"].describe()
```

Out[55]:

STOPS	count	mean	std	min	25%	50%	75%	max
one	250768.0	22870.062037	23576.865371	1105.0	5136.0	7958.0	48851.0	98972.0
two_or_more	13258.0	13921.606125	17180.273151	1966.0	6432.0	8280.0	13410.0	98904.0
zero	36004.0	9375.938535	10623.008293	1105.0	2586.0	4499.0	8064.0	59573.0

In [56]:
```
plt.figure(figsize=(8, 6))
sns.boxplot(data=df, x='STOPS', y='PRICE',showfliers=False)
plt.xlabel('Number of Stops')
plt.ylabel('Ticket Price')
plt.title('Ticket Price vs Number of Stops')
plt.show()
```



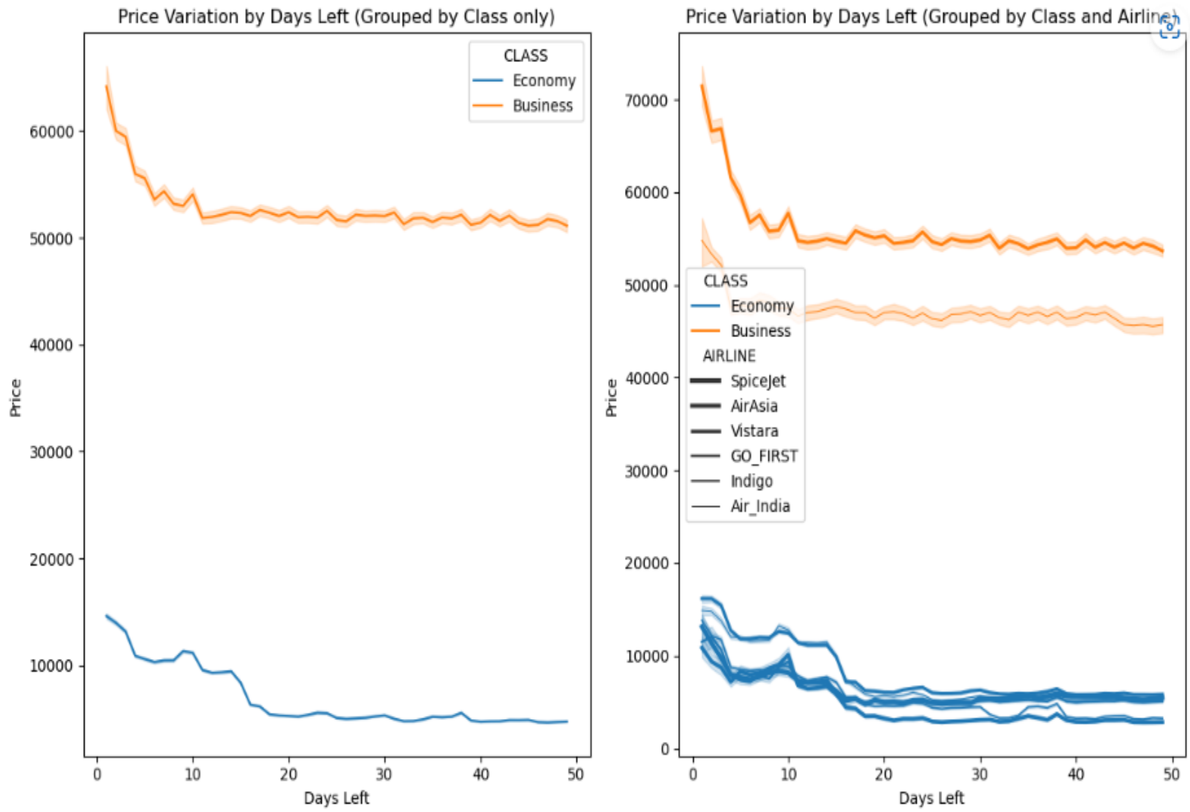
Observation:

- Flights having one stop has maximum ticket price

Analysing variation in price when tickets are bought in just few days before departure

In [57]:
```
fig, axs = plt.subplots(1, 2, figsize=(12, 7))
sns.lineplot(data=df, x='DAYS_LEFT', y='PRICE', hue='CLASS', ax=axs[0])
axs[0].set_xlabel('Days Left')
axs[0].set_ylabel('Price')
axs[0].set_title('Price Variation by Days Left (Grouped by Class only)')

sns.lineplot(data=df, x='DAYS_LEFT', y='PRICE', hue='CLASS', size='AIRLINE', ax=axs[1])
axs[1].set_xlabel('Days Left')
axs[1].set_ylabel('Price')
axs[1].set_title('Price Variation by Days Left (Grouped by Class and Airline)')
plt.tight_layout()
plt.show()
```



Observation:

- According to the analysis, it is clear that there is a significant increase in ticket prices when purchased only 1-2 days before the scheduled departure.
- Additionally, Business class tickets experience even higher price hikes compared to Economy class tickets. This highlights the importance of planning and booking flights well in advance to secure more affordable prices.
- Prices are low when a ticket is booked many days before the departure and price increases as the days left get reduced

Analysing variation in price with departure time and arrival time in both business and economy class

```
In [58]: business_df = df[df['CLASS'] == 'Business']
         economy_df = df[df['CLASS'] == 'Economy']
```

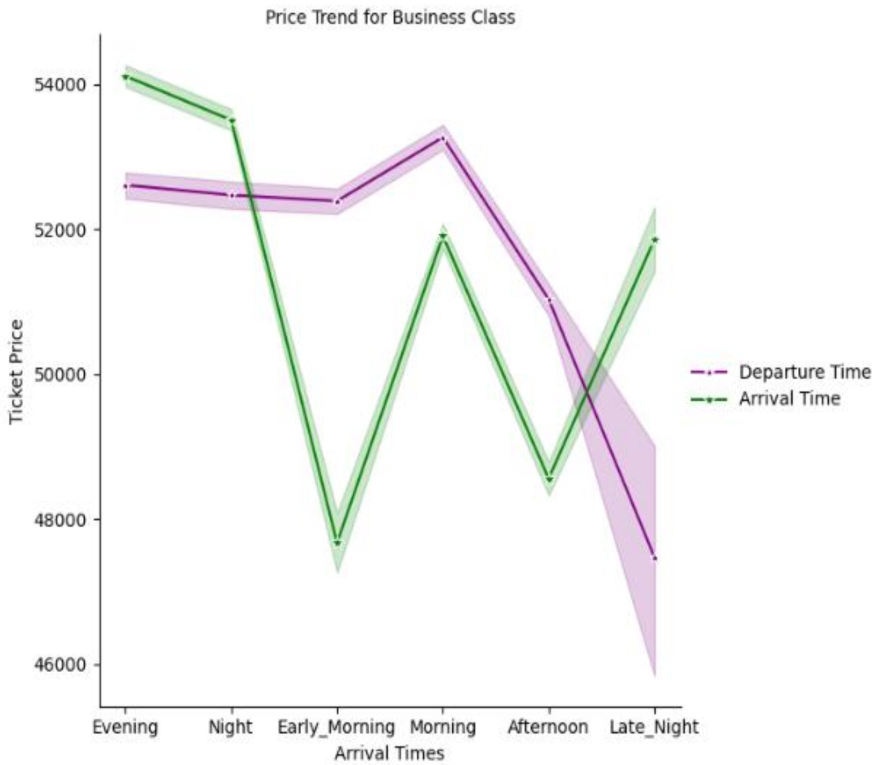
```
In [59]: graph = sns.FacetGrid(economy_df, col="CLASS", height=6)
         graph.map(sns.lineplot, 'DEPARTURE_TIME', 'PRICE', color='purple', label='Departure Time')
         graph.map(sns.lineplot, 'ARRIVAL_TIME', 'PRICE', color='green', label='Arrival Time')
         graph.set_axis_labels("Departure Times/Arrival Times", "Ticket Price")
         graph.add_legend()
         graph.set_titles("Price Trend for Economy Class")
         plt.show()
```

Observation:

- The line plot for departure time is displayed in purple, while the line plot for arrival time is shown in green.
- The x-axis represents the departure or arrival times, while the y-axis represents the corresponding ticket prices.
- From the visualization, it is apparent that in the Economy class, ticket prices tend to be higher when the departure time is in the morning, gradually decreasing throughout the day.
- Similarly, when considering arrival time, prices are generally higher in the evening and lower during the early morning and late night periods.

- This information can be valuable for travellers looking to optimize their budget by selecting departure and arrival times that align with lower-priced tickets.

```
In [60]: graph1 = sns.FacetGrid(business_df, col="CLASS", height=6)
         graph1.map(sns.lineplot, 'DEPARTURE_TIME', 'PRICE', color='purple', label='Departure Time',marker='*')
         graph1.map(sns.lineplot, 'ARRIVAL_TIME', 'PRICE', color='green', label='Arrival Time',marker='*',markersize=7)
         graph1.set_axis_labels("Arrival Times", "Ticket Price")
         graph1.add_legend()
         graph1.set_titles("Price Trend for Business Class")
         plt.show()
```



Observation:

- The arrival time line plot is shown in green, and the departure time line plot is shown in purple.
- The trend for the Business class is slightly different than that of the Economy class. In business class, ticket prices tend to be higher when the departure time is in the morning, gradually it decreased.
- In case of arrival time, price higher in the evening and lower in early morning.

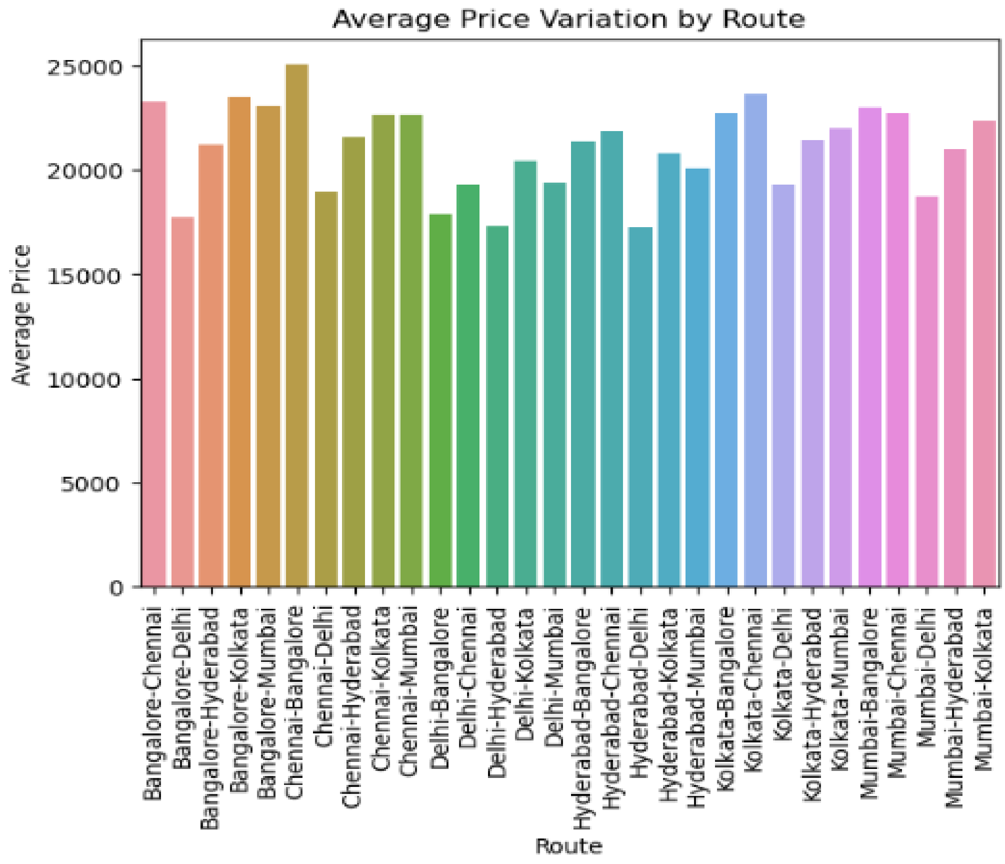
Analysing Average ticket price in each route

In [61]:
```
route_prices = df.groupby('ROUTE')['PRICE'].mean().reset_index()
route_prices
```

Out[61]:

	ROUTE	PRICE
0	Bangalore-Chennai	23321.850078
1	Bangalore-Delhi	17716.468484
2	Bangalore-Hyderabad	21226.121192
3	Bangalore-Kolkata	23491.916426
4	Bangalore-Mumbai	23104.470584
5	Chennai-Bangalore	25056.425974
6	Chennai-Delhi	18973.205071
7	Chennai-Hyderabad	21591.345404
8	Chennai-Kolkata	22623.859292
9	Chennai-Mumbai	22633.566663
10	Delhi-Bangalore	17880.216315
11	Delhi-Chennai	19308.759283
12	Delhi-Hyderabad	17327.472872
13	Delhi-Kolkata	20445.224385
14	Delhi-Mumbai	19355.829812
15	Hyderabad-Bangalore	21347.177998
16	Hyderabad-Chennai	21848.065989
17	Hyderabad-Delhi	17243.945685
18	Hyderabad-Kolkata	20823.893201
19	Hyderabad-Mumbai	20063.415126
20	Kolkata-Bangalore	22694.805154
21	Kolkata-Chennai	23660.361040
22	Kolkata-Delhi	19276.015637
23	Kolkata-Hyderabad	21424.094043
24	Kolkata-Mumbai	22015.584308
25	Mumbai-Bangalore	23035.396596
26	Mumbai-Chennai	22739.893926
27	Mumbai-Delhi	18719.059090
28	Mumbai-Hyderabad	20996.531856
29	Mumbai-Kolkata	22372.914689

```
In [62]: sns.barplot(data=route_prices, x='ROUTE', y='PRICE')
         plt.xticks(rotation=90)
         plt.title("Average Price Variation by Route")
         plt.xlabel("Route")
         plt.ylabel("Average Price")
         plt.xticks(rotation=90)
         plt.show()
```



Observation:

- The above data provides average ticket prices for various routes between different cities in India.
- The prices for different routes vary significantly.
- The highest average ticket price is observed for the Chennai-Bangalore route (25,056.43), while the lowest is for the Delhi-Hyderabad route (17,243.95).
- These average ticket prices can be useful for travellers to estimate the cost of air travel between different city pairs in India.

Analysing average ticket price by each route on the basis of class

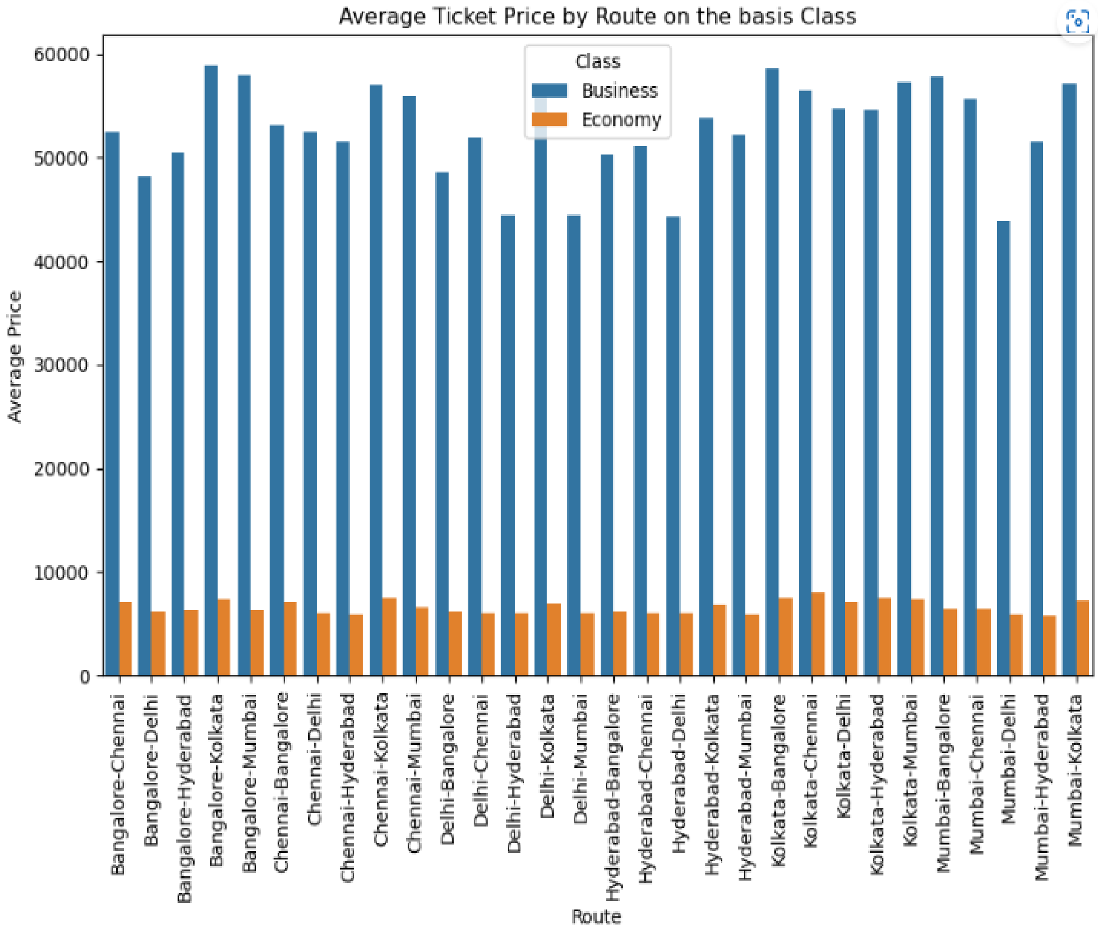
In [63]:
```
route_class_prices = df.groupby(['ROUTE','CLASS'])['PRICE'].mean().reset_index()
route_class_prices
```

Out[63]:

	ROUTE	CLASS	PRICE
0	Bangalore-Chennai	Business	52436.915395
1	Bangalore-Chennai	Economy	7105.953850
2	Bangalore-Delhi	Business	48127.546101
3	Bangalore-Delhi	Economy	6124.897982
4	Bangalore-Hyderabad	Business	50395.796948
5	Bangalore-Hyderabad	Economy	6360.141698
6	Bangalore-Kolkata	Business	58839.943631
7	Bangalore-Kolkata	Economy	7375.638594
8	Bangalore-Mumbai	Business	57983.403626
9	Bangalore-Mumbai	Economy	6381.093332
10	Chennai-Bangalore	Business	53069.921313
11	Chennai-Bangalore	Economy	7175.020192
12	Chennai-Delhi	Business	52424.542983
13	Chennai-Delhi	Economy	6075.961190
14	Chennai-Hyderabad	Business	51559.874283
15	Chennai-Hyderabad	Economy	5960.788831
16	Chennai-Kolkata	Business	56992.472744
17	Chennai-Kolkata	Economy	7547.295815
18	Chennai-Mumbai	Business	55982.927536
19	Chennai-Mumbai	Economy	6529.119453
20	Delhi-Bangalore	Business	48576.027921
21	Delhi-Bangalore	Economy	6175.622535
22	Delhi-Chennai	Business	51903.924984
23	Delhi-Chennai	Economy	6102.317245
24	Delhi-Hyderabad	Business	44409.806922
25	Delhi-Hyderabad	Economy	6031.164261
26	Delhi-Kolkata	Business	55983.122587
27	Delhi-Kolkata	Economy	7045.621678
28	Delhi-Mumbai	Business	44364.442811
29	Delhi-Mumbai	Economy	6059.826087

30	Hyderabad-Bangalore	Business	50358.290706
31	Hyderabad-Bangalore	Economy	6234.882649
32	Hyderabad-Chennai	Business	51132.155288
33	Hyderabad-Chennai	Economy	6049.884930
34	Hyderabad-Delhi	Business	44250.700281
35	Hyderabad-Delhi	Economy	6072.296659
36	Hyderabad-Kolkata	Business	53729.157762
37	Hyderabad-Kolkata	Economy	6881.680392
38	Hyderabad-Mumbai	Business	52148.155975
39	Hyderabad-Mumbai	Economy	5969.259906
40	Kolkata-Bangalore	Business	58586.947674
41	Kolkata-Bangalore	Economy	7471.621990
42	Kolkata-Chennai	Business	56502.775035
43	Kolkata-Chennai	Economy	8011.745229
44	Kolkata-Delhi	Business	54713.079716
45	Kolkata-Delhi	Economy	7161.400077
46	Kolkata-Hyderabad	Business	54575.548180
47	Kolkata-Hyderabad	Economy	7489.144374
48	Kolkata-Mumbai	Business	57301.183607
49	Kolkata-Mumbai	Economy	7405.787239
50	Mumbai-Bangalore	Business	57773.371545
51	Mumbai-Bangalore	Economy	6432.511946
52	Mumbai-Chennai	Business	55625.692079
53	Mumbai-Chennai	Economy	6420.917984
54	Mumbai-Delhi	Business	43832.830038
55	Mumbai-Delhi	Economy	5889.281400
56	Mumbai-Hyderabad	Business	51579.822650
57	Mumbai-Hyderabad	Economy	5774.891130
58	Mumbai-Kolkata	Business	57095.080742
59	Mumbai-Kolkata	Economy	7227.971735

```python
In [64]: route_class_prices = df.groupby(['ROUTE', 'CLASS'])['PRICE'].mean().reset_index()
         plt.figure(figsize=(10, 6))
         sns.barplot(data=route_class_prices, x='ROUTE', y='PRICE', hue='CLASS')
         plt.xticks(rotation=90)
         plt.title("Average Ticket Price by Route on the basis Class")
         plt.xlabel("Route")
         plt.ylabel("Average Price")
         plt.legend(title="Class")
         plt.show()
```



Observation:

- The data indicates that both business class and economy class are available for each route.
- The graph clearly shows that the average ticket price for business class is significantly higher than that of economy class.

Analysis of average price by destination city and source city

In [65]:
```
average_price_by_destination = df.groupby('DESTINATION_CITY')['PRICE'].mean().reset_index()
average_price_by_destination
```

Out[65]:

	DESTINATION_CITY	PRICE
0	Bangalore	21551.930430
1	Chennai	21926.879048
2	Delhi	18404.952947
3	Hyderabad	20407.425719
4	Kolkata	21921.040415
5	Mumbai	21330.573738

In [66]:
```
average_price_by_Source = df.groupby('SOURCE_CITY')['PRICE'].mean().reset_index()
average_price_by_Source
```

Out[66]:

	SOURCE_CITY	PRICE
0	Bangalore	21459.987821
1	Chennai	21948.233880
2	Delhi	18913.571971
3	Hyderabad	20151.324331
4	Kolkata	21674.638780
5	Mumbai	21448.389864

```
In [67]: fig, axs = plt.subplots(1, 2, figsize=(12, 6))
df.groupby('DESTINATION_CITY').mean()['PRICE'].plot(kind='bar', color='hotpink', ax=axs[0])
axs[0].set_xlabel('Destination City')
axs[0].set_ylabel('Average Price')
axs[0].set_xticklabels(axs[0].get_xticklabels(), rotation=45)
axs[0].set_title('Average Price by Destination City')

df.groupby('SOURCE_CITY').mean()['PRICE'].plot(kind='bar', color='deeppink', ax=axs[1])
axs[1].set_xlabel('Source City')
axs[1].set_ylabel('Average Price')
axs[1].set_xticklabels(axs[1].get_xticklabels(), rotation=45)
axs[1].set_title('Average Price by Source City')
plt.tight_layout()
plt.show()
```

Observation:

* In both graphs, it is evident that the average ticket prices are generally lower for Delhi compared to Chennai and Kolkata.

* Chennai and Kolkata are the leading ones.

Class wise analysis of each airline

In [68]:
```
df_pivot=df.copy()
```

In [69]:
```
business_df = df[df['CLASS'] == 'Business']
df_pivot = pd.pivot_table(economy_df, index='AIRLINE', values='CLASS', aggfunc='count')
df_pivot.columns = ['Flight Count']
df_pivot
```

Out[69]:

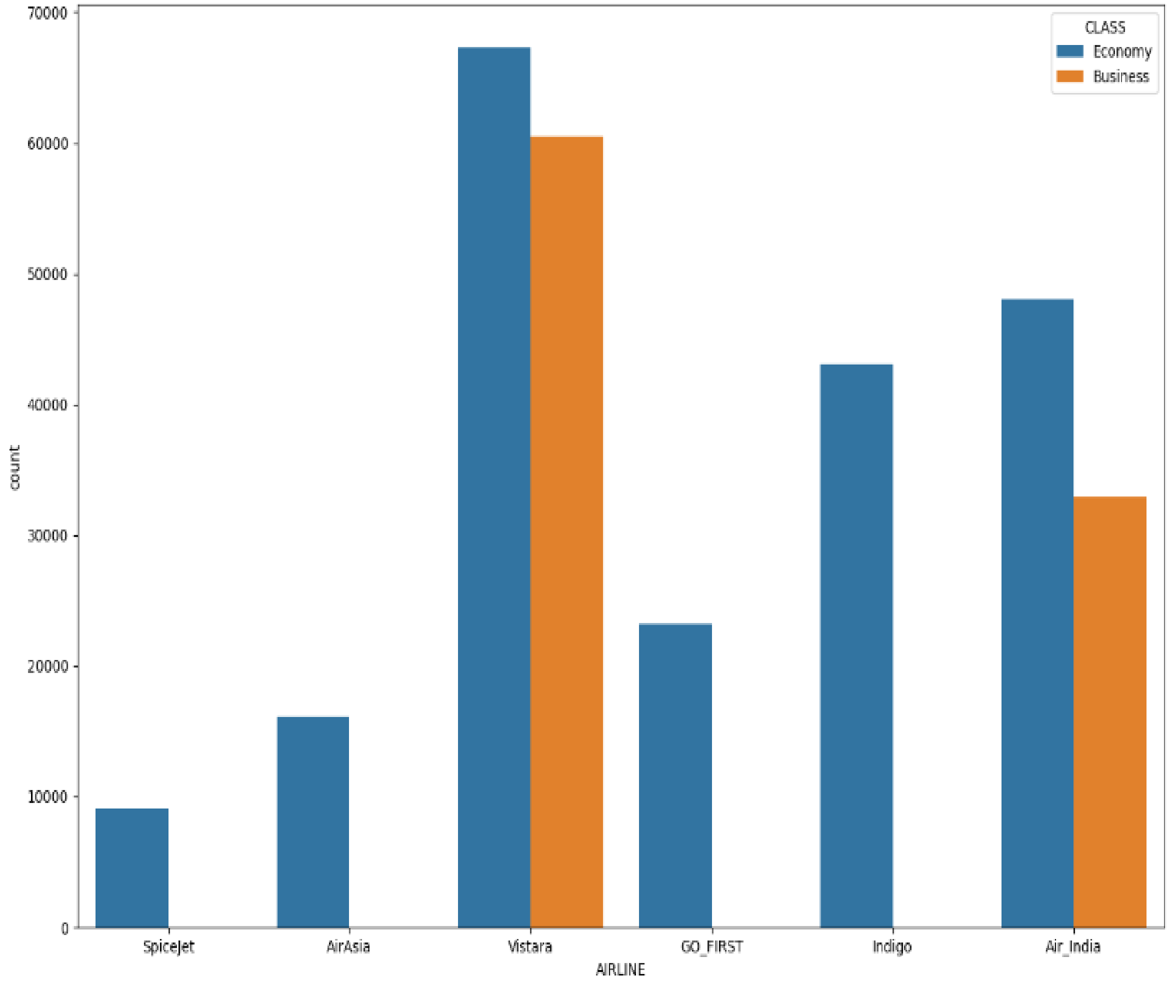
AIRLINE	Flight Count
Air_India	32898
Vistara	60466

In [70]:
```
economy_df = df[df['CLASS'] == 'Economy']
df_pivot = pd.pivot_table(economy_df, index='AIRLINE', values='CLASS', aggfunc='count')
df_pivot.columns = ['Flight Count']
df_pivot
```

Out[70]:

AIRLINE	Flight Count
AirAsia	16098
Air_India	47994
GO_FIRST	23173
Indigo	43120
SpiceJet	9011
Vistara	67270

```
[71]: plt.figure(figsize = (15,10))
sns.countplot(data = df, x = "AIRLINE" ,hue = "CLASS")
plt.show()
```



Observation:

- Based on the observation that the number of flights in the economy class varies among different airlines.
- SpiceJet has the lowest count with 9,011 flights, while Vistara has the highest count with 67,270 flights. Air India, GO_FIRST, Indigo, and AirAsia also have significant numbers of flights in the economy class.
- But there are only two airlines available for the business class namely Vistara and air India.

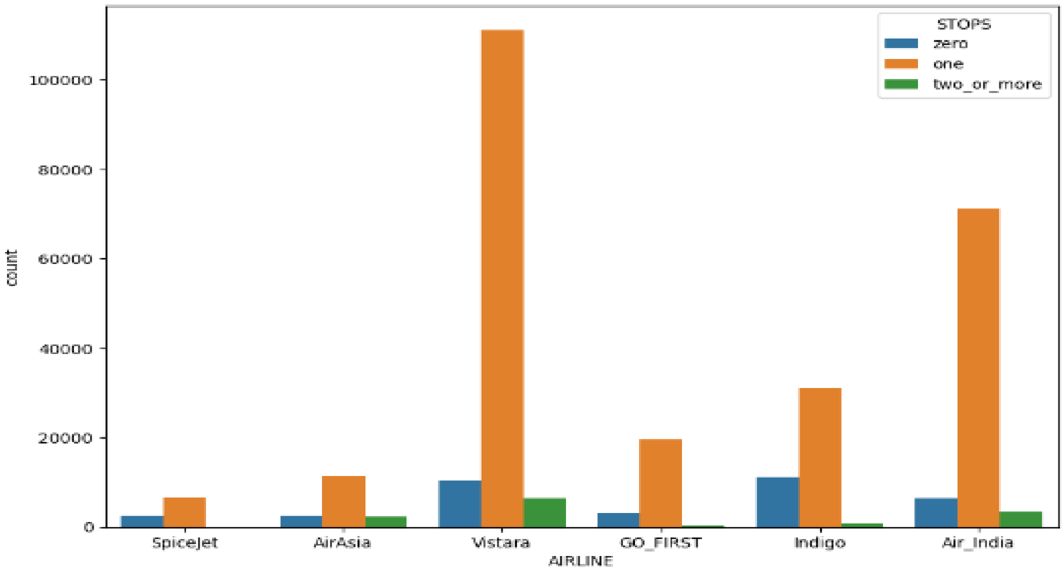
Analysis of Stops in each Airlines

In [72]:
```
zero_stops_df = df[df['STOPS'] == 'zero']
one_stop_df = df[df['STOPS'] == 'one']
two_or_more_stops_df = df[df['STOPS'] == 'two_or_more']
pivot_zero = pd.pivot_table(zero_stops_df, index='AIRLINE', values='STOPS', aggfunc='count')
pivot_one = pd.pivot_table(one_stop_df, index='AIRLINE', values='STOPS', aggfunc='count')
pivot_two_or_more = pd.pivot_table(two_or_more_stops_df, index='AIRLINE', values='STOPS', aggfunc='count')
df_combined = pd.concat([pivot_zero, pivot_one, pivot_two_or_more], axis=1)
df_combined.columns = ['Zero Stops', 'One Stop', 'Two or More Stops']
df_combined
```

Out[72]:

AIRLINE	Zero Stops	One Stop	Two or More Stops
AirAsia	2434	11418	2246.0
Air_India	6409	71004	3479.0
GO_FIRST	3223	19545	405.0
Indigo	11216	31166	738.0
SpiceJet	2462	6549	NaN
Vistara	10260	111086	6390.0

In [73]:
```
plt.figure(figsize = (10,7))
sns.countplot(data = df, x = "AIRLINE", hue = "STOPS")
plt.show()
```



Observation:

- Based on the given data, it can be inferred that the majority of travellers prefer flights with one stop in all airlines.

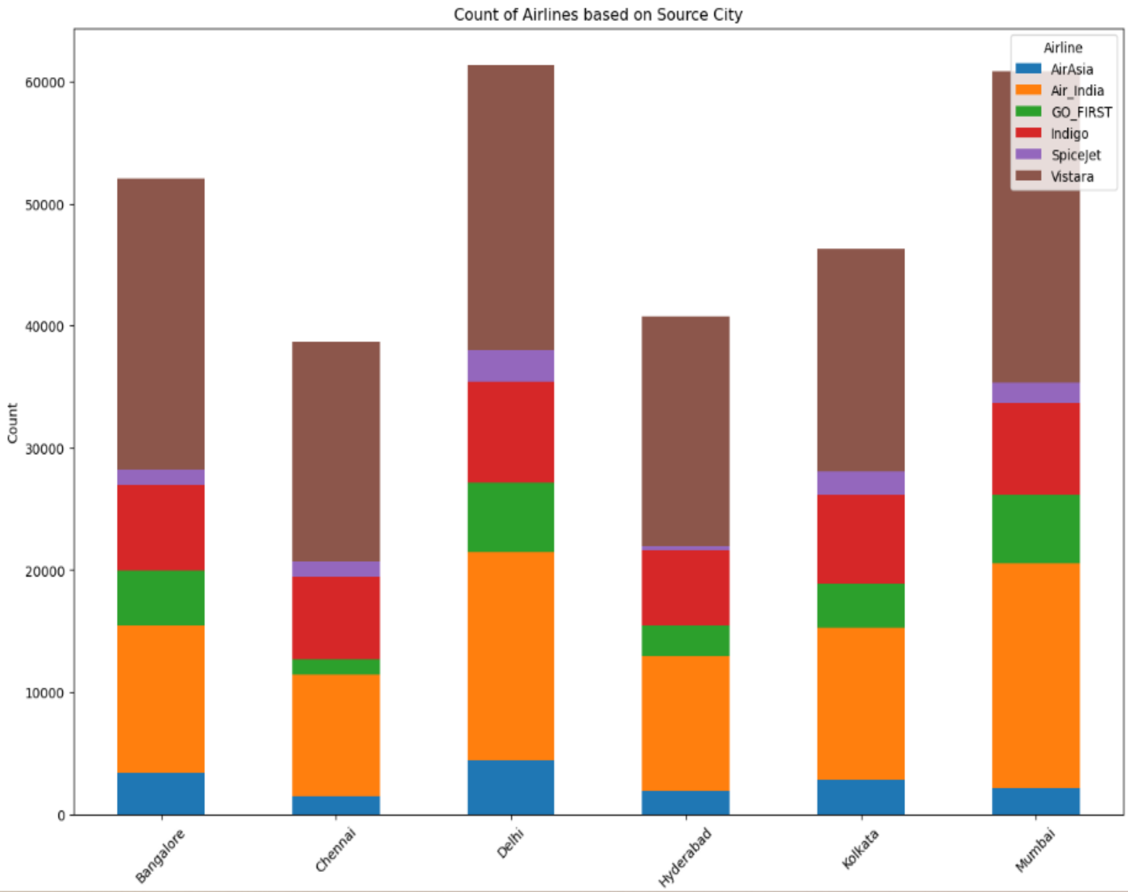
Analysis of source city in each airline.

In [74]:
```
df_analysis = df.groupby(['SOURCE_CITY', 'AIRLINE']).size().reset_index(name='Count')
df_analysis
```

Out[74]:

	SOURCE_CITY	AIRLINE	Count
0	Bangalore	AirAsia	3364
1	Bangalore	Air_India	12052
2	Bangalore	GO_FIRST	4498
3	Bangalore	Indigo	7080
4	Bangalore	SpiceJet	1255
5	Bangalore	Vistara	23806
6	Chennai	AirAsia	1498
7	Chennai	Air_India	9912
8	Chennai	GO_FIRST	1289
9	Chennai	Indigo	6746
10	Chennai	SpiceJet	1219
11	Chennai	Vistara	18014
12	Delhi	AirAsia	4387
13	Delhi	Air_India	17063
14	Delhi	GO_FIRST	5724
15	Delhi	Indigo	8277
16	Delhi	SpiceJet	2524
17	Delhi	Vistara	23341
18	Hyderabad	AirAsia	1844
19	Hyderabad	Air_India	11088
20	Hyderabad	GO_FIRST	2504
21	Hyderabad	Indigo	6215
22	Hyderabad	SpiceJet	332
23	Hyderabad	Vistara	18821
24	Kolkata	AirAsia	2829
25	Kolkata	Air_India	12400
26	Kolkata	GO_FIRST	3590
27	Kolkata	Indigo	7296
28	Kolkata	SpiceJet	1947
29	Kolkata	Vistara	18245
30	Mumbai	AirAsia	2176
31	Mumbai	Air_India	18377
32	Mumbai	GO_FIRST	5568
33	Mumbai	Indigo	7506
34	Mumbai	SpiceJet	1734
35	Mumbai	Vistara	25509

In [75]:
```python
grouped_data = df.groupby(['SOURCE_CITY', 'AIRLINE']).size().unstack()
grouped_data.plot(kind='bar', stacked=True, figsize=(15, 10))
plt.xlabel('Source City')
plt.ylabel('Count')
plt.title('Count of Airlines based on Source City')
plt.xticks(rotation=45)
plt.legend(title='Airline')
plt.show()
```



Observation:

- The data reveals the distribution of flights originating from various source cities for different airlines.
- Vistara emerges as the dominant airline in terms of flights from most cities, including Bangalore, Chennai, Delhi, Hyderabad, Kolkata, and Mumbai.
- These findings shed light on the airline preferences and availability of flight options in different source cities, indicating Vistara and Air India as prominent choices for travellers across multiple locations.

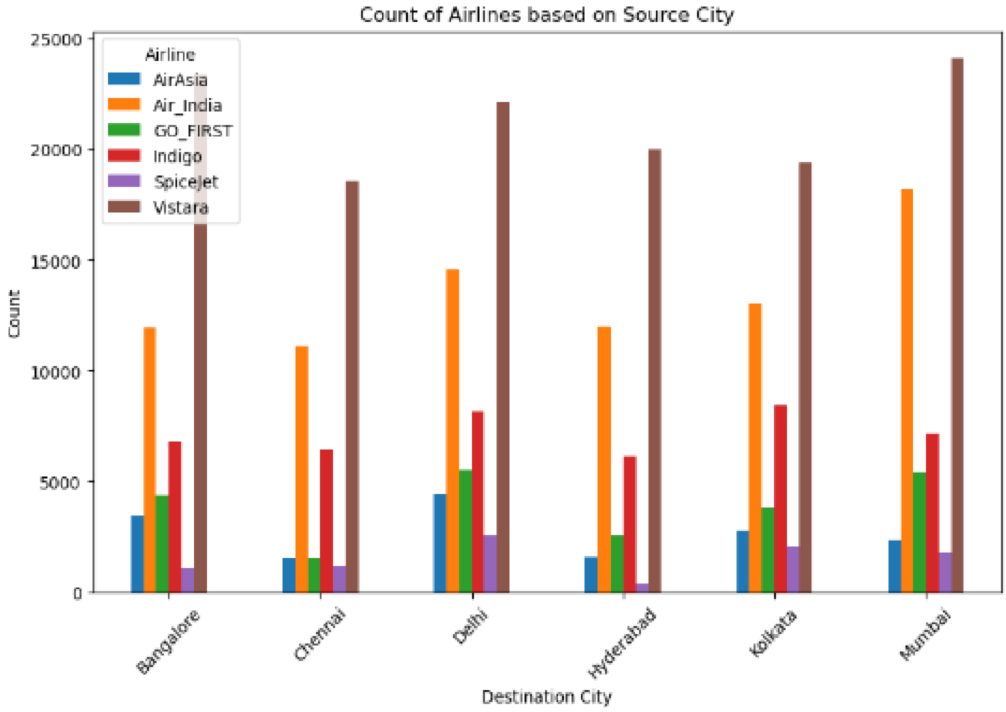
Analysis of destination city in each airline

In [76]:
```
df_analysis1 = df.groupby(['DESTINATION_CITY', 'AIRLINE']).size().reset_index(name='Count')
df_analysis1
```

Out[76]:

	DESTINATION_CITY	AIRLINE	Count
0	Bangalore	AirAsia	3437
1	Bangalore	Air_India	11959
2	Bangalore	GO_FIRST	4386
3	Bangalore	Indigo	6772
4	Bangalore	SpiceJet	1088
5	Bangalore	Vistara	23400
6	Chennai	AirAsia	1516
7	Chennai	Air_India	11141
8	Chennai	GO_FIRST	1488
9	Chennai	Indigo	6449
10	Chennai	SpiceJet	1172
11	Chennai	Vistara	18589
12	Delhi	AirAsia	4433
13	Delhi	Air_India	14550
14	Delhi	GO_FIRST	5509
15	Delhi	Indigo	8133
16	Delhi	SpiceJet	2541
17	Delhi	Vistara	22173
18	Hyderabad	AirAsia	1560
19	Hyderabad	Air_India	12022
20	Hyderabad	GO_FIRST	2576
21	Hyderabad	Indigo	6147
22	Hyderabad	SpiceJet	383
23	Hyderabad	Vistara	20028
24	Kolkata	AirAsia	2789
25	Kolkata	Air_India	13043
26	Kolkata	GO_FIRST	3794
27	Kolkata	Indigo	8437
28	Kolkata	SpiceJet	2054
29	Kolkata	Vistara	19394
30	Mumbai	AirAsia	2363
31	Mumbai	Air_India	18177
32	Mumbai	GO_FIRST	5420
33	Mumbai	Indigo	7182
34	Mumbai	SpiceJet	1773
35	Mumbai	Vistara	24152

```
In [77]: grouped_data1 = df.groupby(['DESTINATION_CITY', 'AIRLINE']).size().unstack()
         grouped_data1.plot(kind='bar', figsize=(10, 6))
         plt.xlabel('Destination City')
         plt.ylabel('Count')
         plt.title('Count of Airlines based on Source City')
         plt.xticks(rotation=45)
         plt.legend(title='Airline')
         plt.show()
```



Observation:

- The data highlights the distribution of flights to various destination cities across different airlines. Vistara emerges as the leading airline same as the case of source city

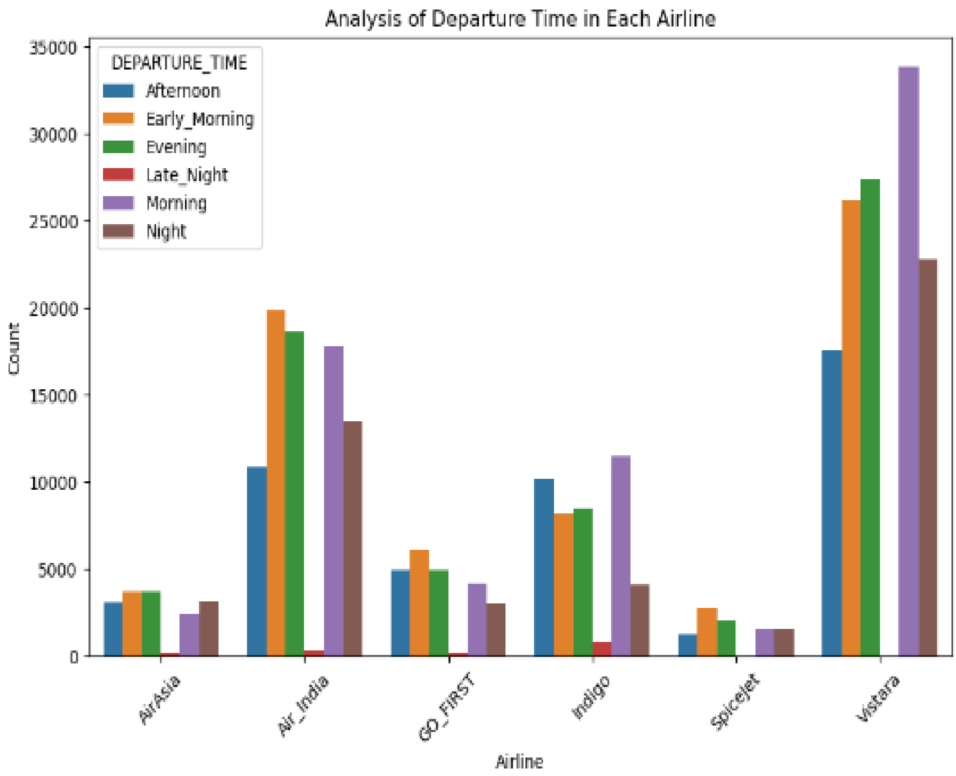
Analysis of departure time in each airline.

In [78]:
```
df_analysis1 = df.groupby(['DEPARTURE_TIME', 'AIRLINE']).size().reset_index(name='Count')
df_analysis1
```

Out[78]:

	DEPARTURE_TIME	AIRLINE	Count
0	Afternoon	AirAsia	3078
1	Afternoon	Air_India	10876
2	Afternoon	GO_FIRST	4942
3	Afternoon	Indigo	10155
4	Afternoon	SpiceJet	1193
5	Afternoon	Vistara	17544
6	Early_Morning	AirAsia	3892
7	Early_Morning	Air_India	19867
8	Early_Morning	GO_FIRST	6103
9	Early_Morning	Indigo	8184
10	Early_Morning	SpiceJet	2728
11	Early_Morning	Vistara	26193
12	Evening	AirAsia	3657
13	Evening	Air_India	18626
14	Evening	GO_FIRST	4904
15	Evening	Indigo	8460
16	Evening	SpiceJet	2031
17	Evening	Vistara	27389
18	Late_Night	AirAsia	143
19	Late_Night	Air_India	291
20	Late_Night	GO_FIRST	146
21	Late_Night	Indigo	726
22	Morning	AirAsia	2348
23	Morning	Air_India	17768
24	Morning	GO_FIRST	4116
25	Morning	Indigo	11491
26	Morning	SpiceJet	1519
27	Morning	Vistara	33860
28	Night	AirAsia	3180
29	Night	Air_India	13484
30	Night	GO_FIRST	2962
31	Night	Indigo	4104
32	Night	SpiceJet	1540
33	Night	Vistara	22750

```
In [79]: plt.figure(figsize=(10, 6))
grouped_data = df.groupby(['AIRLINE', 'DEPARTURE_TIME']).size().reset_index(name='Count')
sns.barplot(data=grouped_data, x='AIRLINE', y='Count', hue='DEPARTURE_TIME')
plt.xlabel('Airline')
plt.ylabel('Count')
plt.title('Analysis of Departure Time in Each Airline')
plt.xticks(rotation=45)
plt.show()
```



Observation:

- The data provides insights into the distribution of flights based on the departure time. Vistara consistently appears to have the highest number of flights, indicating its popularity among travellers.
- In the afternoon and evening, Vistara continues to lead with significant flight counts, followed by Air_India and Indigo.
- Early morning departures also see Vistara as the dominant airline.

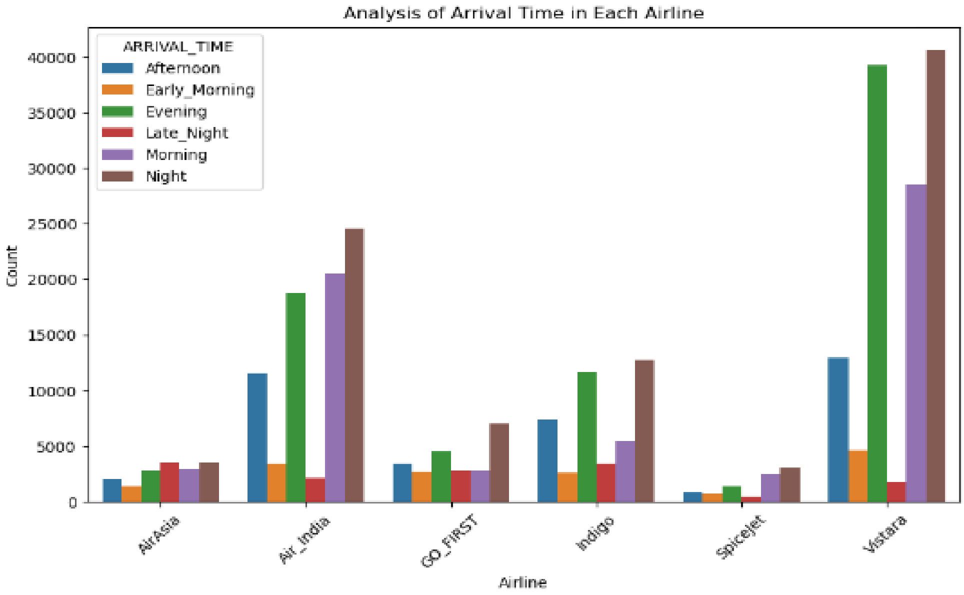
Analysis of Arrival time in each airline

In [80]:
```
df_analysis1 = df.groupby(['ARRIVAL_TIME', 'AIRLINE']).size().reset_index(name='Count')
df_analysis1
```

Out[80]:

	ARRIVAL_TIME	AIRLINE	Count
0	Afternoon	AirAsia	2052
1	Afternoon	Air_India	11576
2	Afternoon	GO_FIRST	3373
3	Afternoon	Indigo	7367
4	Afternoon	SpiceJet	844
5	Afternoon	Vistara	12922
6	Early_Morning	AirAsia	1407
7	Early_Morning	Air_India	3405
8	Early_Morning	GO_FIRST	2705
9	Early_Morning	Indigo	2537
10	Early_Morning	SpiceJet	721
11	Early_Morning	Vistara	4640
12	Evening	AirAsia	2762
13	Evening	Air_India	18748
14	Evening	GO_FIRST	4503
15	Evening	Indigo	11591
16	Evening	SpiceJet	1403
17	Evening	Vistara	39272
18	Late_Night	AirAsia	3491
19	Late_Night	Air_India	2090
20	Late_Night	GO_FIRST	2778
21	Late_Night	Indigo	3455
22	Late_Night	SpiceJet	456
23	Late_Night	Vistara	1731
24	Morning	AirAsia	2909
25	Morning	Air_India	20521
26	Morning	GO_FIRST	2761
27	Morning	Indigo	5469
28	Morning	SpiceJet	2525
29	Morning	Vistara	28528
30	Night	AirAsia	3477
31	Night	Air_India	24552
32	Night	GO_FIRST	7053
33	Night	Indigo	12701
34	Night	SpiceJet	3062
35	Night	Vistara	40643

In [81]:
```
plt.figure(figsize=(10, 6))
grouped_data = df.groupby(['AIRLINE', 'ARRIVAL_TIME']).size().reset_index(name='Count')
sns.barplot(data=grouped_data, x='AIRLINE', y='Count', hue='ARRIVAL_TIME')
plt.xlabel('Airline')
plt.ylabel('Count')
plt.title('Analysis of Arrival Time in Each Airline')
plt.xticks(rotation=45)
plt.show()
```



Observation:

- The data presents the distribution of flights based on the arrival time and airline. Vistara stands out as the dominant airline across different arrival times

Creation of a Model:

Convert the Object Datatypes to int using label encoder

```
In [82]: from sklearn.preprocessing import LabelEncoder
         columns_to_encode = ['AIRLINE', 'SOURCE_CITY','DEPARTURE_TIME','STOPS','ARRIVAL_TIME','DESTINATION_CITY','CLASS']
         le = LabelEncoder()
         for column in columns_to_encode:
             df[column] = le.fit_transform(df[column])
```

Taking x and y for the model

```
n [83]: x=df.drop(['PRICE','ROUTE','FLIGHT'],axis=1)
         y=df['PRICE']
```

Spliting the dataset into train and test

```
[85]: x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.30,random_state=42)
```

Checking for the shape of train and test for both independent and dependant variable.

```
In [86]: x_train.shape,x_test.shape,y_train.shape,y_test.shape
Out[86]: ((210021, 9), (90009, 9), (210021,), (90009,))
```

<u>Model 1- Linear Regression Model</u>

Creation of the linear regression model

```
modelmlr.fit(x_train, y_train)
LinearRegression()
```

```
In [92]: scoretrain = round(modelmlr.score(x_train, y_train) * 100, 2)
         scoretest=round(modelmlr.score(x_test,y_test)*100,2)
         r2=round(r2_score(y_test,y_pred)*100,2)
```

```
In [93]: print('The score of train is',scoretrain)
         print('The score of test is',scoretest)
         print('The r2 score is',r2)

         The score of train is 90.59
         The score of test is 90.62
         The r2 score is 90.62
```

Observation:

- 90.62 % of Accuracy in Linear Regression Model.

<u>Model 2- Decision Tree Regressor</u>

Creation of the Decision Tree regression model

```
In [96]: modeldcr.fit(x_train,y_train)
Out[96]: DecisionTreeRegressor()
```

```
In [98]: scoretrain = round(modeldcr.score(x_train, y_train) * 100, 2)
         scoretest=round(modeldcr.score(x_test,y_test)*100,2)
         r2D=round(r2_score(y_test,y_pred)*100,2)
```

```
In [99]: print('The score of train is',scoretrain)
         print('The score of test is',scoretest)
         print('The r2 score is',r2D)

         The score of train is 99.94
         The score of test is 97.71
         The r2 score is 97.71
```

Observation:

- 97.71 % of Accuracy in Decision Tree Regression Model

Model 3- Random Forest Regressor

Creation of the Random Forest regression model

```
In [102]: modelrfr.fit(x_train, y_train)
Out[102]: RandomForestRegressor()
```

```
In [104]: scoretrain = round(modelrfr.score(x_train, y_train) * 100, 2)
          scoretest=round(modelrfr.score(x_test,y_test)*100,2)
          r2R=round(r2_score(y_test,y_pred)*100,2)
```

```
In [105]: print('The score of train is',scoretrain)
          print('The score of test is',scoretest)
          print('The r2 score is',r2R)

          The score of train is 99.76
          The score of test is 98.56
          The r2 score is 98.56
```

Observation:

- 98.56 % of Accuracy in Random Forest Regression

Model 4- Extra Tree Regressor

Creation of the linear regression model

```
In [108]: modeletr.fit(x_train,y_train)
Out[108]: ExtraTreesRegressor()
```

```
In [110]: scoretrain = round(modeletr.score(x_train, y_train) * 100, 2)
          scoretest=round(modeletr.score(x_test,y_test)*100,2)
          r2E=round(r2_score(y_test,y_pred)*100,2)
```

```
In [111]: print('The score of train is',scoretrain)
          print('The score of test is',scoretest)
          print('The r2 score is',r2E)

          The score of train is 99.94
          The score of test is 98.36
          The r2 score is 98.36
```

Observation:

- 98.36 % of Accuracy in Extra tree regression model.

Chapter 4

Analysis of the Result

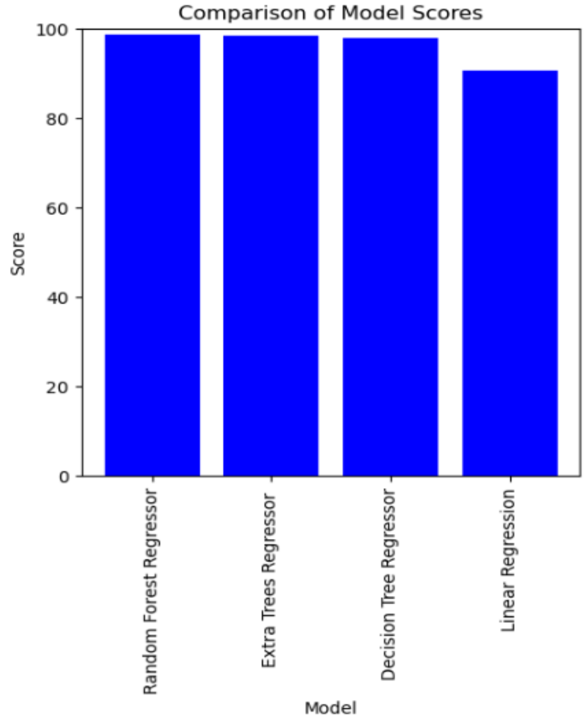
Finding the best model:

In [112]:
```
results = pd.DataFrame({'Model': [ 'Linear Regression','Decision Tree Regressor', 'Random Forest Regressor',
                                   'Extra Trees Regressor '],'Score': [ r2L,r2D, r2R,r2E]})
output_df = results.sort_values(by='Score', ascending=False)
output_df = output_df.reset_index(drop=True)
output_df
```

Out[112]:

	Model	Score
0	Random Forest Regressor	98.56
1	Extra Trees Regressor	98.36
2	Decision Tree Regressor	97.71
3	Linear Regression	90.62

In [113]:
```
plt.figure(figsize=(5,5))
plt.bar(output_df['Model'], output_df['Score'], color='blue')
plt.xlabel('Model')
plt.ylabel('Score')
plt.title('Comparison of Model Scores')
plt.ylim(0, 100)
plt.xticks(rotation=90)
plt.show()
```



Observation:

- Random Forest Regressor Model having highest Accuracy Score 98.56%
- Extra Tree Regressor Model having the second highest Accuracy Score 98.35%
- Decision Tree Regressor Model having the Third Highest Accuracy Score 97.70%
- Linear Regression Model Provide the lowest Accuracy Score 90.62%

Chapter 5

Conclusion

- Based on the provided accuracy scores, the Random Forest model emerges as the superior choice for predicting flight fares compared to other models. The Random Forest model achieved an impressive training score of 99.76, indicating its ability to accurately predict flight fares when trained on the available data. Furthermore, when tested on new, unseen data, the model obtained a score of 98.56%, highlighting its capability to generalize well and make reliable predictions. The high R2 score of 98.56 further confirms the strong correlation between the predicted and actual flight fares.

- Our machine learning algorithm will able to predict the restaurant rating.

- There are 6 unique airlines. Vistara becoming the most popular airline and Spicejet is the least popular
- Economy Class is the most common class among the airlines

- flights with a single stop seem to be the preferred choice for travellers.
- It can be observed that the Delhi-Mumbai route has the highest frequency of flights compared to other routes.
- Vistara has Maximum Price range
- Flights having one stop has maximum ticket price
- There is a significant increase in ticket prices when purchased only 1-2 days before the scheduled departure

References

1.Scikit Learn official documentation- [scikit-learn: machine learning in Python — scikit-learn 1.2.2 documentation]

2.Javapoint -[Tutorials List - Javatpoint]

3. Flight Price Prediction | EDA | Linear Regression- [Flight Price Prediction | EDA | Linear Regression | Kaggle]

4. EDA on Flight Data-[EDA on Flight Data | Kaggle]