Airfare ML : Predicting Flight Fares

Context :

> An airline is a company that provides air transport services for traveling passengers and freight. Airlines use aircraft to supply these services and may form partnerships or alliances with other airlines for codeshare agreements, in which they both offer and operate the same flight. Generally, airline companies are recognized with an air operating certificate or license issued by a governmental aviation body. Airlines may be scheduled or charter operators.

> Airlines assign prices to their services in an attempt to maximize profitability. The pricing of airline tickets has become increasingly complicated over the years and is now largely determined by computerized yield management systems.

> The price of an Airline Ticket is affected by a number of factors, such as flight duration, days left for departure, arrival time and departure time etc. Airline organizations may diminish the cost at the time they need to build the market and at the time when the tickets are less accessible. They may maximize the costs. The price may rely upon different factors. Each factor has its own proprietary rules and algorithms to set the price accordingly. Recent advances in Artificial Intelligence (AI) and Machine Learning (ML) makes it possible to infer such rules and model the price variation.

Sources :

> Data collected using Python script with Beautiful Soup and Selenium libraries. Script collected data on various flight details such as Date of booking, Date of travel, Airline and class, Departure time and source, Arrival time and destination, Duration, Total stops, Price. The scraping process was designed to collect data for flights departing from a specific set of airports (Top 7 busiest airports in India). Note that the Departure Time feature also includes the Source airport, and the Arrival Time feature also includes the Destination airport. Which is later extracted in Cleaned_dataset. Also both cleaned and scraped datasets have provided so that one can use dataset as per their requirement and convenience. Inspiration:

> Dataset created to provide users with valuable resource for analyzing flight fares in India. Detailed information on flight fares over time can be used to develop more accurate pricing models and inform users about best times to book tickets. Data can also be used

Library

In [59]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
colors = ['#97C1A9','#FFFFFF']

import warnings
warnings.filterwarnings("ignore")

import imblearn
from collections import Counter
from imblearn.over_sampling import SMOTE
from xgboost import XGBRegressor
from lightgbm import LGBMRegressor
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.model_selection import train_test_split

import pickle
import flaml
```

Data

In [3]:
```
data=pd.read_csv('Cleaned_dataset.csv')
```

In [4]:
```
data.shape
```

Out[4]:
```
(452088, 13)
```

In [5]:
```
data.describe()
```

Out[5]:

	Duration_in_hours	Days_left	Fare
count	452088.000000	452088.000000	452088.000000
mean	12.349222	25.627902	22840.100890
std	7.431478	14.300846	20307.963002
min	0.750000	1.000000	1307.000000
25%	6.583300	13.000000	8762.750000
50%	11.333300	26.000000	13407.000000
75%	16.500000	38.000000	35587.000000
max	43.583300	50.000000	143019.000000

In [6]:
```
data.isnull().sum()
```

Out[6]:
```
Date_of_journey      0
Journey_day          0
Airline              0
Flight_code          0
Class                0
Source               0
Departure            0
Total_stops          0
Arrival              0
Destination          0
Duration_in_hours    0
Days_left            0
Fare                 0
dtype: int64
```

In [7]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 452088 entries, 0 to 452087
Data columns (total 13 columns):
 #   Column             Non-Null Count   Dtype  
---  ------             --------------   -----  
 0   Date_of_journey    452088 non-null  object 
 1   Journey_day        452088 non-null  object 
 2   Airline            452088 non-null  object 
 3   Flight_code        452088 non-null  object 
 4   Class              452088 non-null  object 
 5   Source             452088 non-null  object 
 6   Departure          452088 non-null  object 
 7   Total_stops        452088 non-null  object 
 8   Arrival            452088 non-null  object 
 9   Destination        452088 non-null  object 
 10  Duration_in_hours  452088 non-null  float64
 11  Days_left          452088 non-null  int64  
 12  Fare               452088 non-null  int64  
dtypes: float64(1), int64(2), object(10)
memory usage: 44.8+ MB
```

In [8]:
```python
data = data.dropna()
data.drop_duplicates( keep=False, inplace=True)
data = data.reset_index(drop = True)
data.shape
```

Out[8]:
```
(440087, 13)
```

EDA

In [9]:
```python
df=data.copy()
```

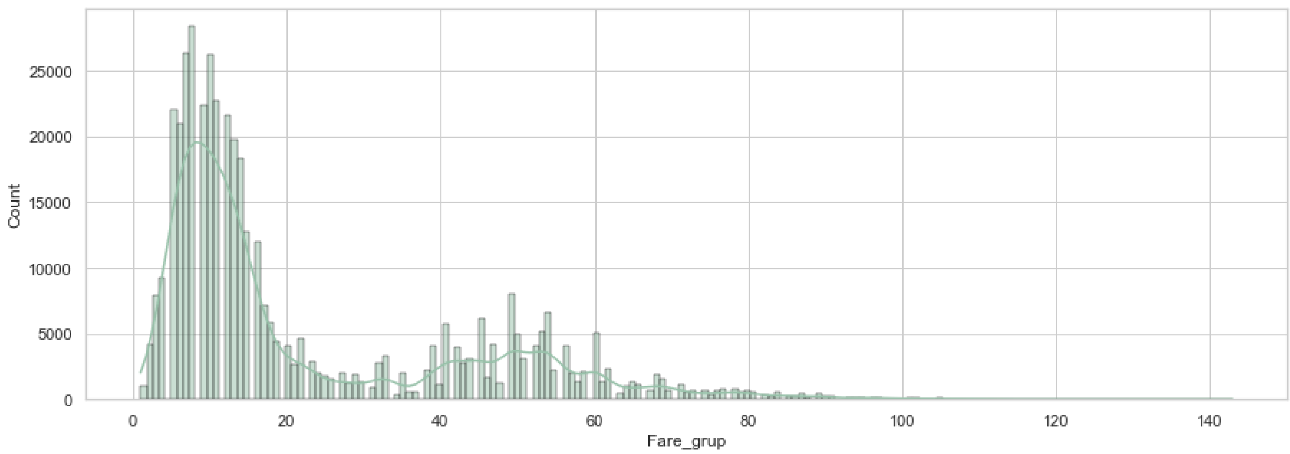
In [10]:
```python
col = list(data.columns)
categorical_features = []
numerical_features = []
for i in col:
    if len(data[i].unique()) > 10:
        numerical_features.append(i)
    else:
        categorical_features.append(i)

print('Categorical Features :',*categorical_features)
print('Numerical Features :',*numerical_features)
```

```
Categorical Features : Journey_day Airline Class Source Departure Total_stops Arriva
l Destination
Numerical Features : Date_of_journey Flight_code Duration_in_hours Days_left Fare
```

In [ ]:
```python
df.head
```

Fare

In [12]:
```python
plt.figure(figsize=(15,5))
sns.set(style='whitegrid')

df['Fare_grup'] = [ int(i / 1000) for i in df['Fare']]
ax=sns.histplot(df['Fare_grup'],kde=True,color=colors[0],edgecolor = 'k');
# ax.bar_label(ax.containers[0])
```



In [13]:
```python
#skewness and kurtosis
print("Skewness: %f" % df['Fare'].skew())
print("Kurtosis: %f" % df['Fare'].kurt())
```

```
Skewness: 1.305353
Kurtosis: 0.796830
```

Categorical

In [14]:
```
df[categorical_features].head()
```

Out[14]:

	Journey_day	Airline	Class	Source	Departure	Total_stops	Arrival	Destination
0	Monday	SpiceJet	Economy	Delhi	After 6 PM	non-stop	After 6 PM	Mumbai
1	Monday	Indigo	Economy	Delhi	After 6 PM	non-stop	Before 6 AM	Mumbai
2	Monday	GO FIRST	Economy	Delhi	After 6 PM	non-stop	Before 6 AM	Mumbai
3	Monday	SpiceJet	Economy	Delhi	After 6 PM	non-stop	After 6 PM	Mumbai
4	Monday	Air India	Economy	Delhi	After 6 PM	non-stop	After 6 PM	Mumbai

In [15]:
```
categorical_features
```

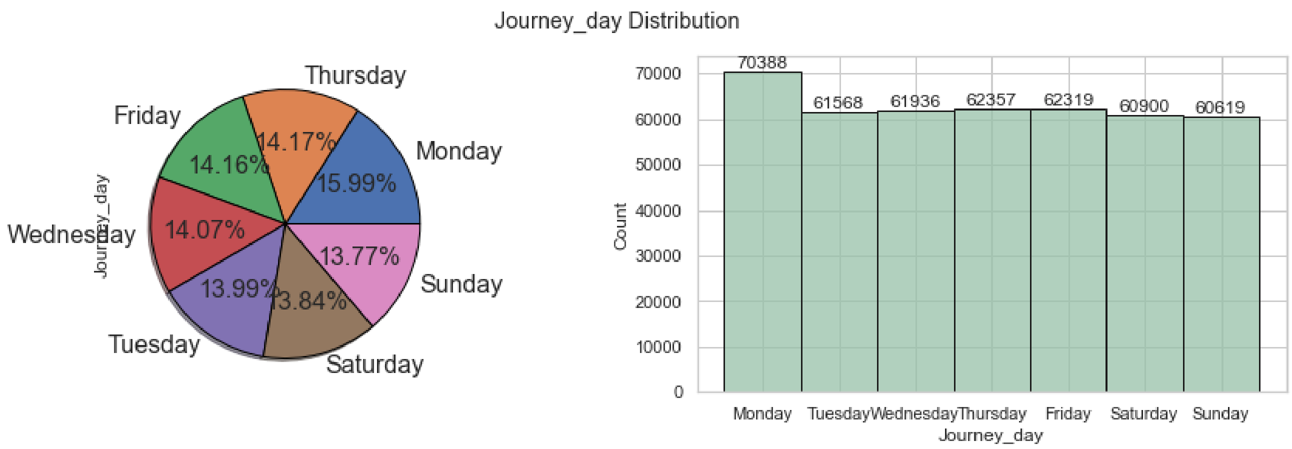
Out[15]:
```
['Journey_day',
 'Airline',
 'Class',
 'Source',
 'Departure',
 'Total_stops',
 'Arrival',
 'Destination']
```

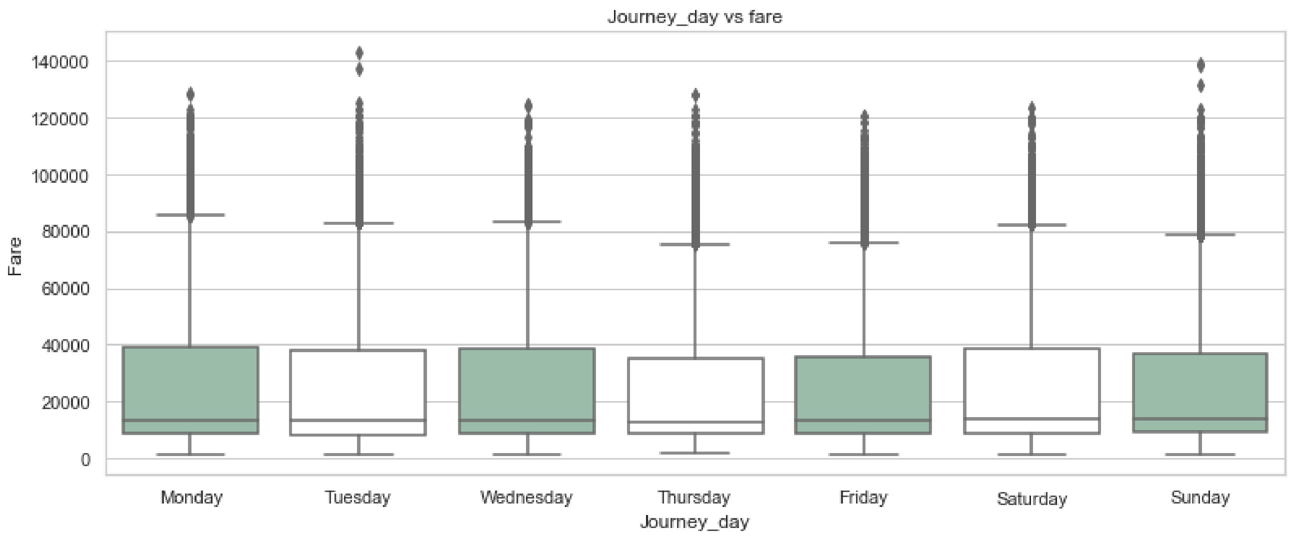
In [16]:
```
def catplot(x):
    sns.set(style='whitegrid')
    fig = plt.subplots(1,2,figsize = (15,4))
    plt.subplot(1,2,1)
    df[x].value_counts().plot.pie(autopct='%1.2f%%', shadow=True, textprops={'fontsiz

    plt.subplot(1,2,2)
    ax=sns.histplot(data=df,x=x,color=colors[0],edgecolor = 'k')
    ax.bar_label(ax.containers[0])

    tit = x + ' Distribution'
    plt.suptitle(tit)

    fig = plt.subplots(1,1,figsize = (13,5))
    plt.subplot(1,1,1)
    ax=sns.boxplot(x = x ,y = 'Fare',data = df,palette = colors);
    tit2=x + ' vs fare'
    plt.title(tit2)
```

In [ ]:

Journey_day

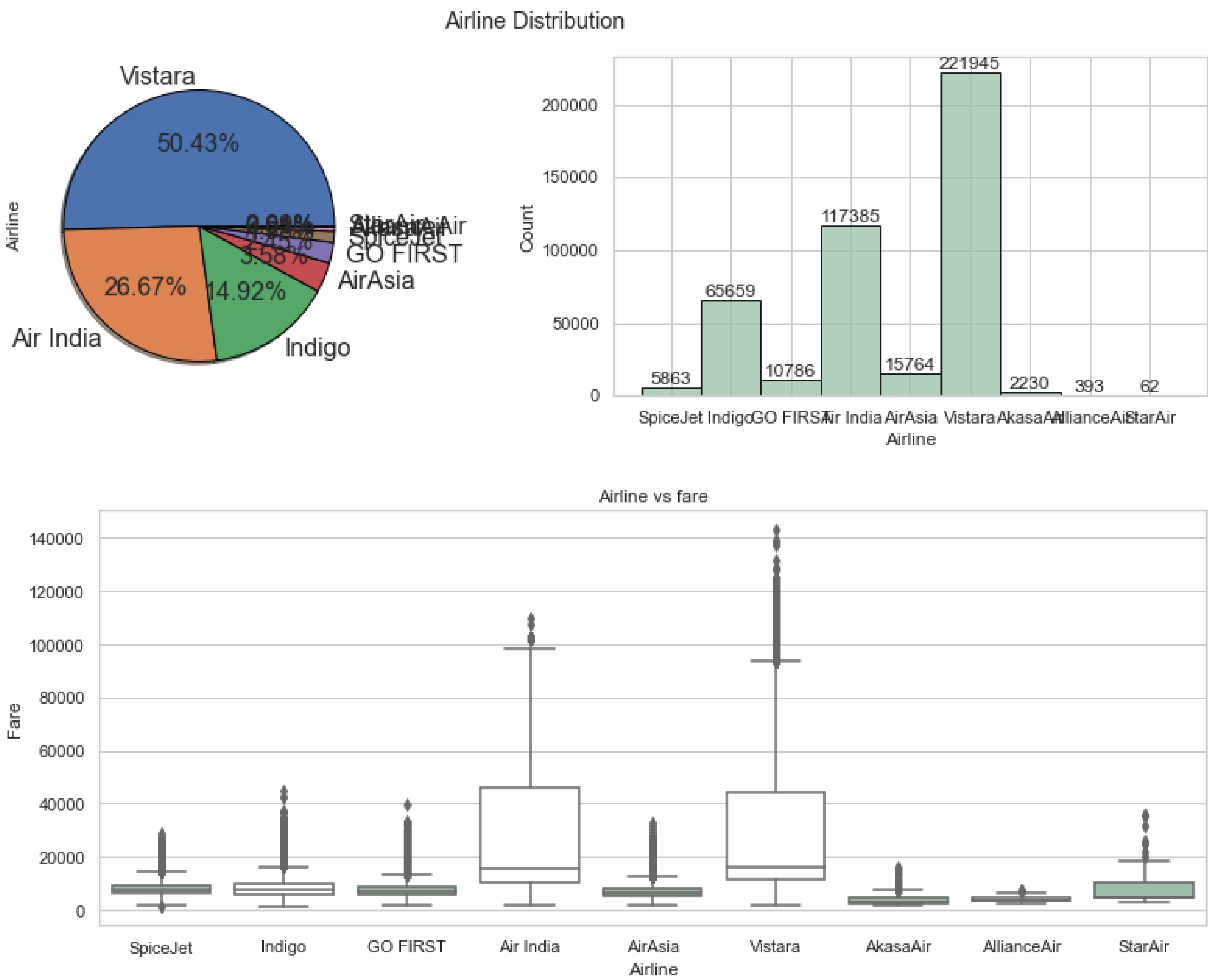
In [17]:
```
catplot('Journey_day')
```





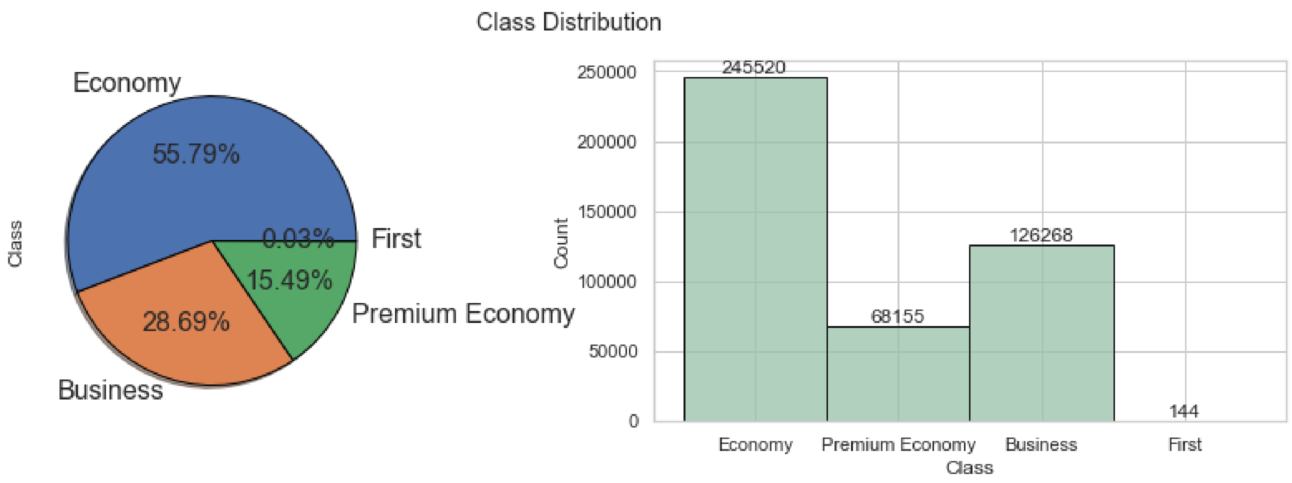
Airline

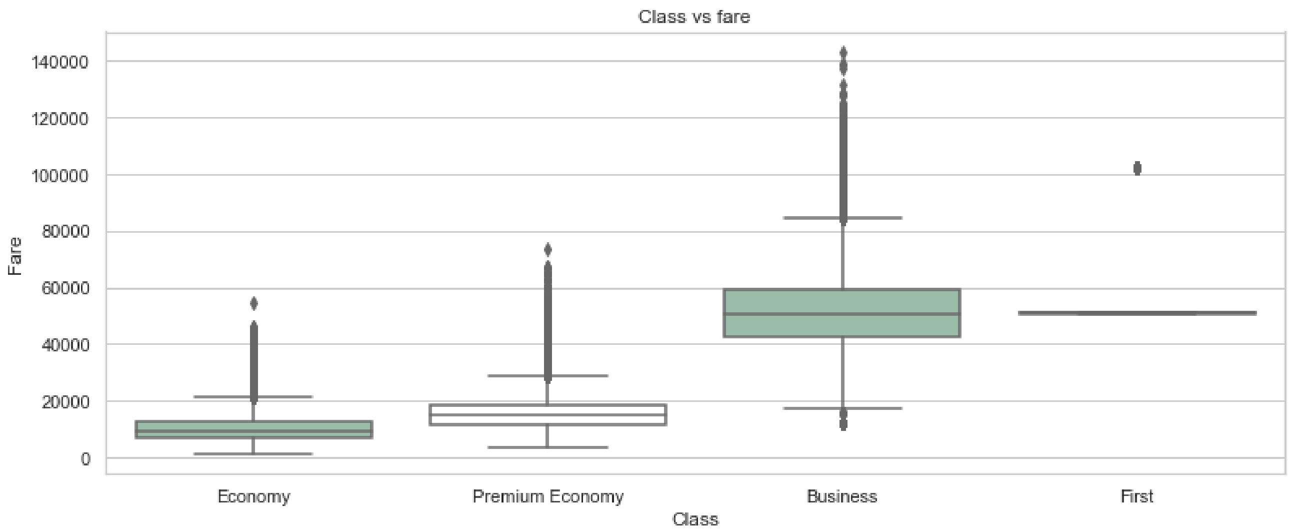
In [18]:
```
catplot('Airline')
```



Class

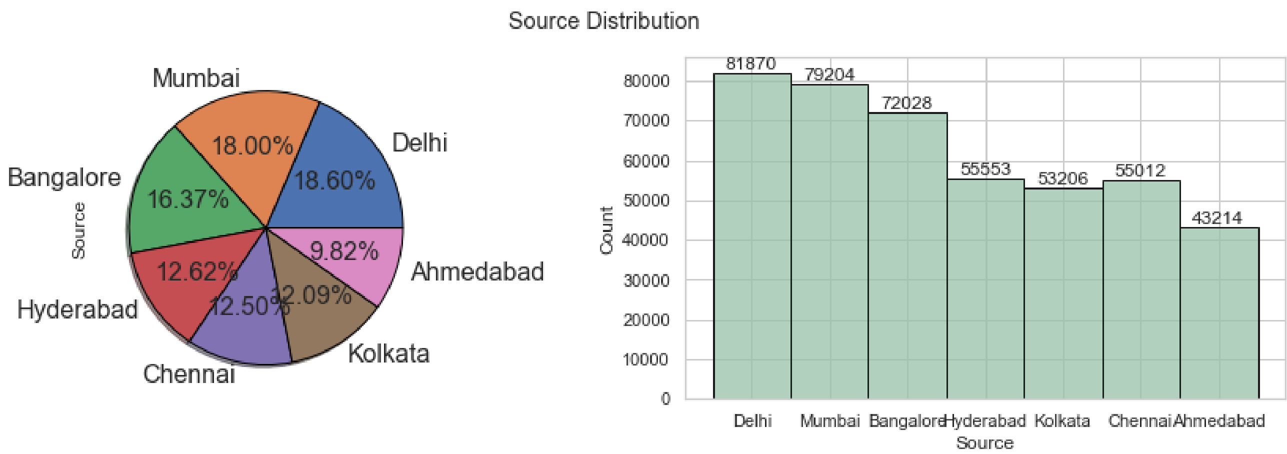
In [19]:
```
catplot('Class')
```

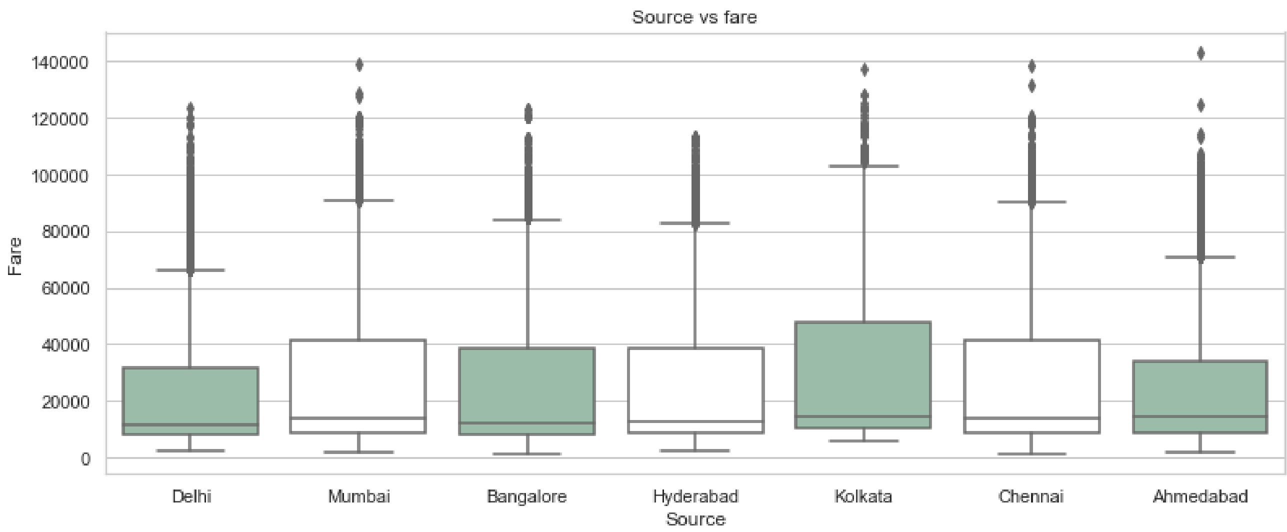




Source

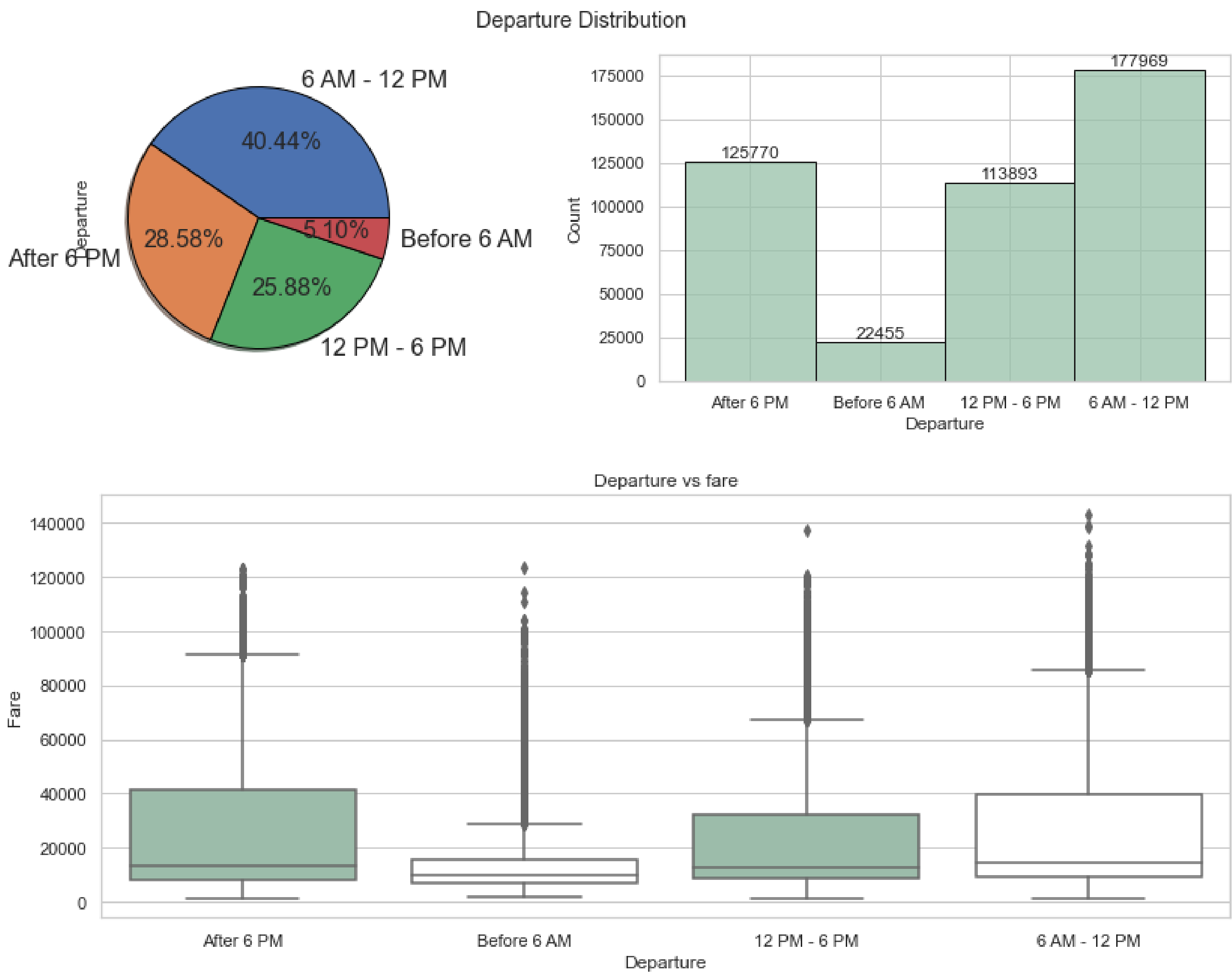
```
In [20]: catplot('Source')
```





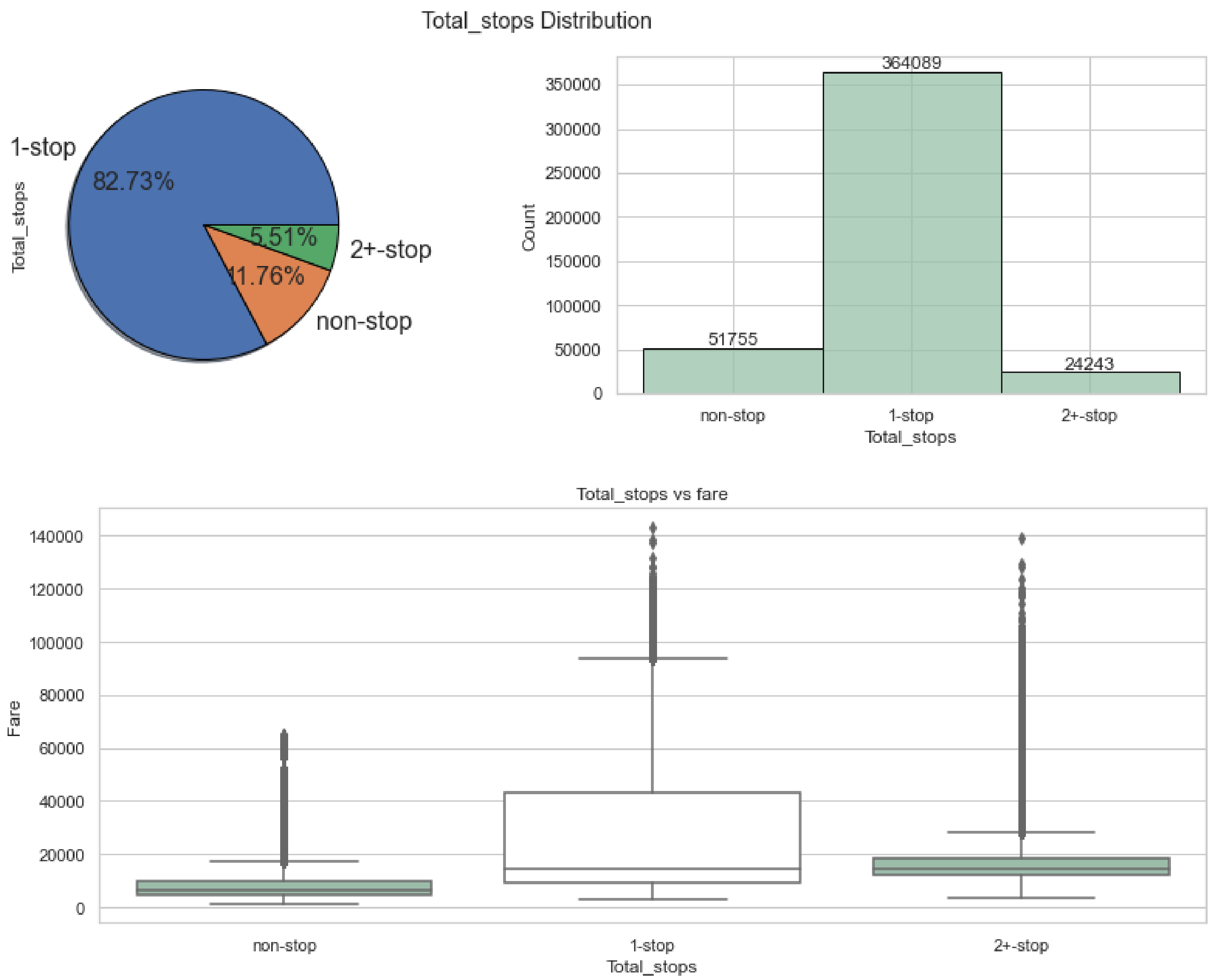
Departure

In [21]:
```
catplot('Departure')
```



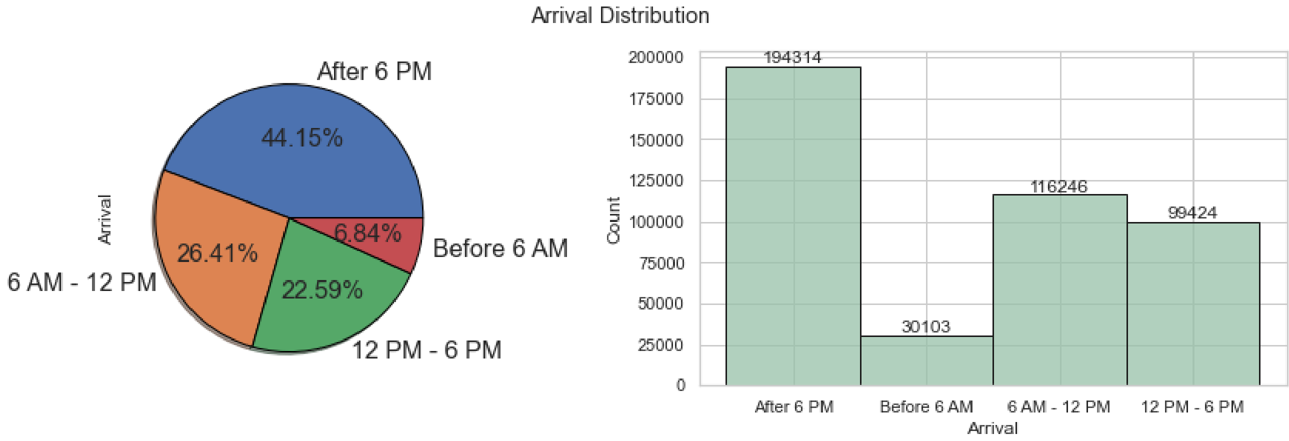
Total_stops

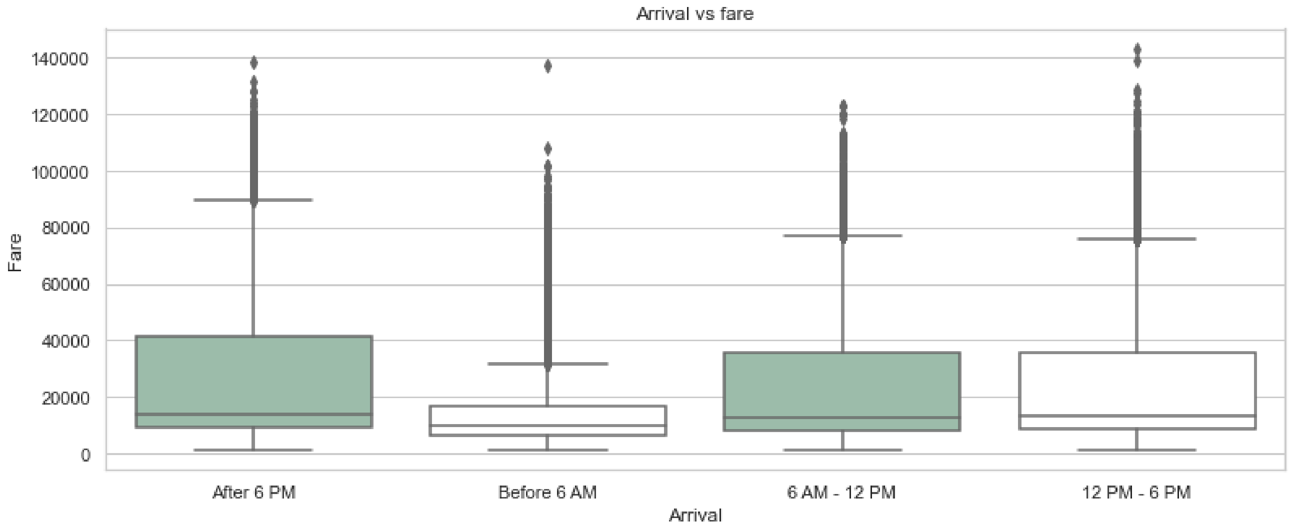
In [22]:
```python
catplot('Total_stops')
```



Arrival

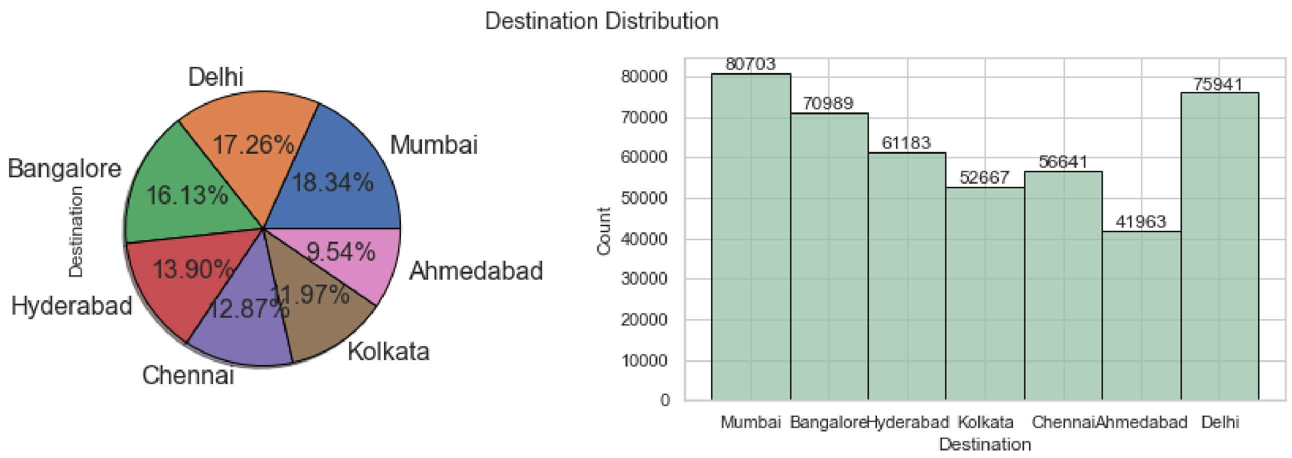
In [23]:
```
catplot('Arrival')
```

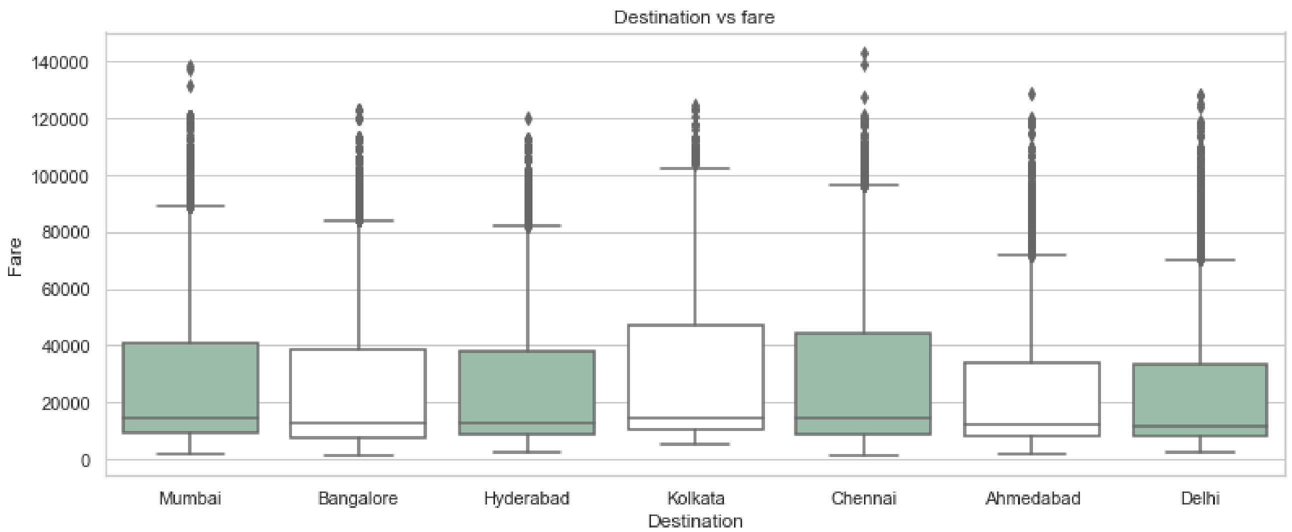




Destination

In [24]:
```
catplot('Destination')
```



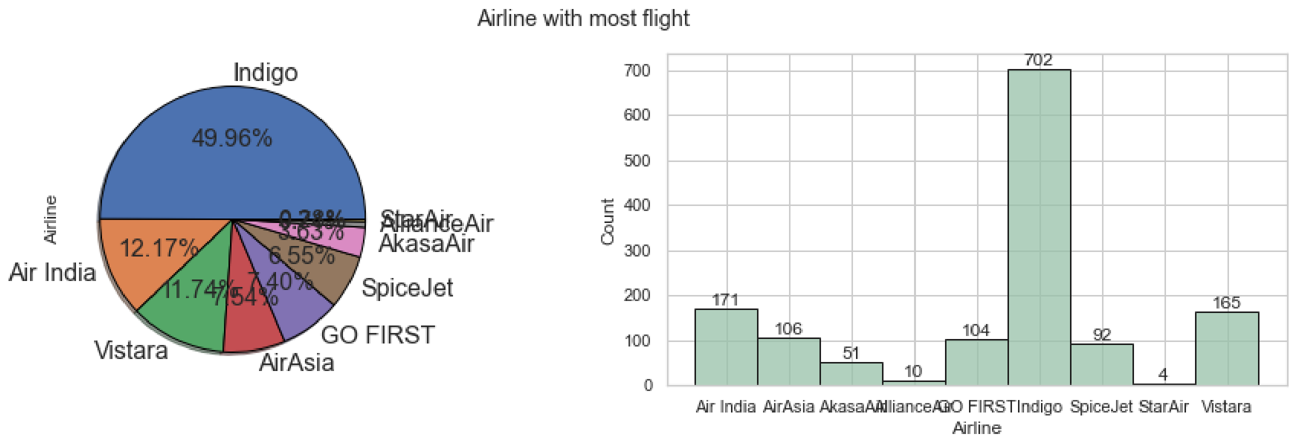


Airline with most flights

In [25]:
```
df1=df.groupby(['Airline','Flight_code'],as_index=False).count()
```

In [26]:
```
sns.set(style='whitegrid')
fig = plt.subplots(1,2,figsize = (16,4))
plt.subplot(1,2,1)
df1['Airline'].value_counts().plot.pie(autopct='%1.2f%%', shadow=True, textprops=

plt.subplot(1,2,2)
ax=sns.histplot(data=df1,x='Airline',color=colors[0],edgecolor = 'k')
ax.bar_label(ax.containers[0])

plt.suptitle('Airline with most flight')
```

Out[26]:
```
Text(0.5, 0.98, 'Airline with most flight')
```



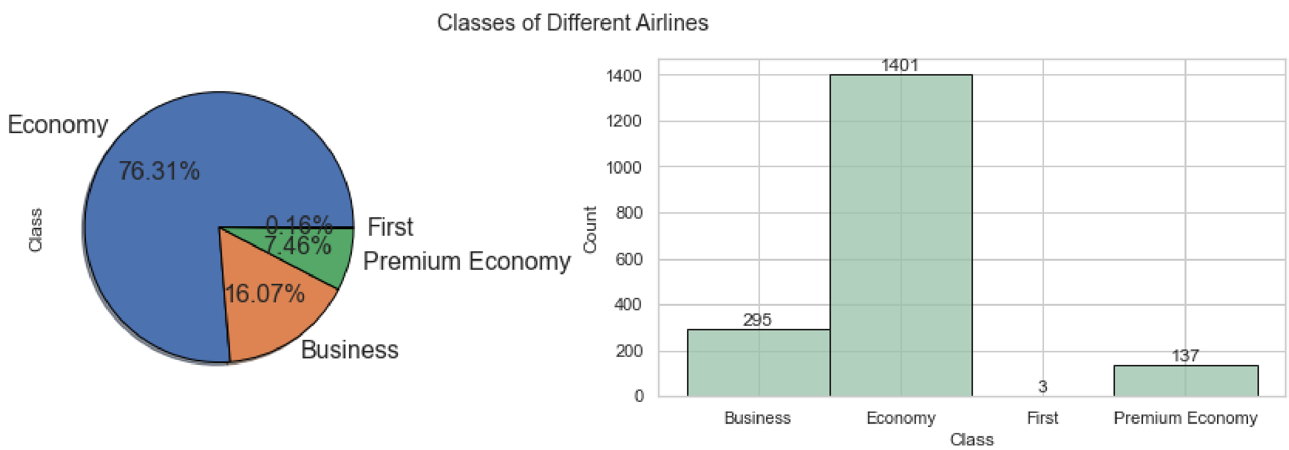
In [ ]:

Classes of Different Airlines

In [27]:
```
df2= df.groupby(['Airline','Flight_code','Class'],as_index=False).count()
```

In [28]:
```
sns.set(style='whitegrid')
fig = plt.subplots(1,2,figsize = (16,4))
plt.subplot(1,2,1)
df2['Class'].value_counts().plot.pie(autopct='%1.2f%%', shadow=True, textprops={'

plt.subplot(1,2,2)
ax=sns.histplot(data=df2,x='Class',color=colors[0],edgecolor = 'k')
ax.bar_label(ax.containers[0])

plt.suptitle('Classes of Different Airlines')
```

Out[28]:
```
Text(0.5, 0.98, 'Classes of Different Airlines')
```

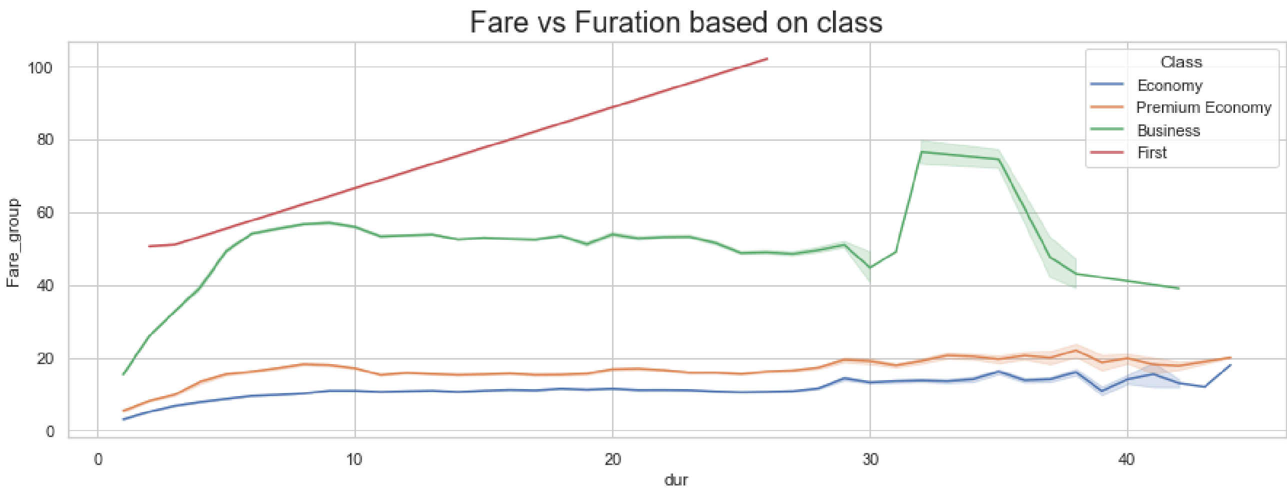


Fare vs Duration based on class

In [29]:
```
df['dur']=df['Duration_in_hours'].round(0)
df['Fare_group'] = [ int(i / 1000) for i in df['Fare']]
```

In [30]:
```
plt.figure(figsize=(15,5))

sns.lineplot(data = df,x = 'dur',y = 'Fare_group',hue = 'Class')
plt.title('Fare vs Furation based on class',fontsize=20)
```

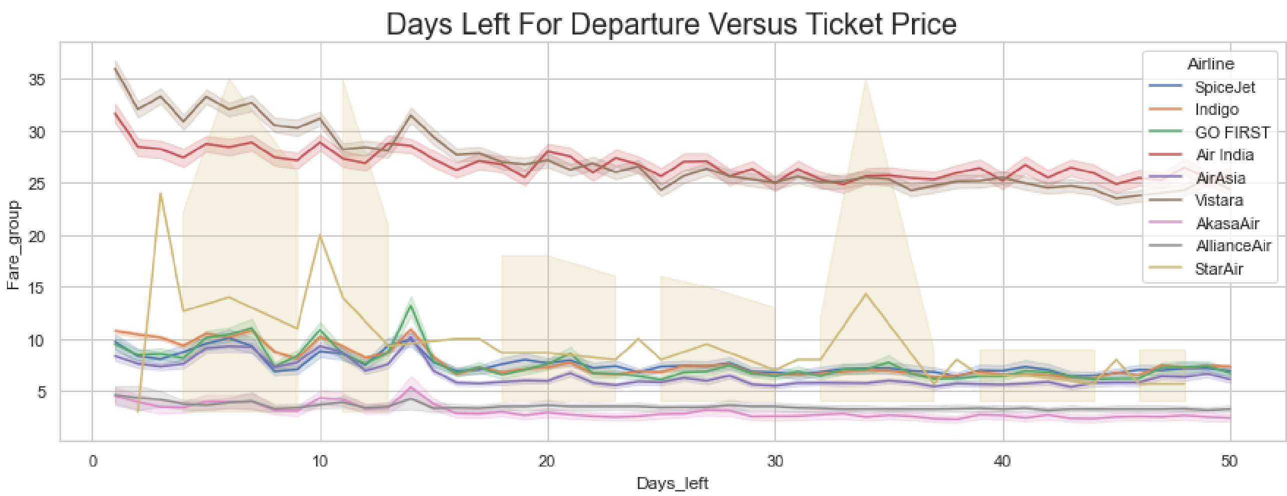
Out[30]: Text(0.5, 1.0, 'Fare vs Furation based on class')



Days Left For Departure Versus Ticket Price of each Airline

In [31]:
```
plt.figure(figsize=(15,5))

sns.lineplot(data=df,x='Days_left',y='Fare_group',hue='Airline')
plt.title('Days Left For Departure Versus Ticket Price',fontsize=20)
```

Out[31]: Text(0.5, 1.0, 'Days Left For Departure Versus Ticket Price')



Total number of Flights from one city to another

In [32]:
```
df.groupby(['Flight_code','Source','Destination','Airline','Class'],as_index=False).c
```

Out[32]:

	Source	Destination	Flight_code
0	Delhi	Mumbai	361
1	Mumbai	Delhi	312
2	Delhi	Bangalore	281
3	Delhi	Chennai	250
4	Delhi	Kolkata	224
5	Delhi	Hyderabad	222
6	Mumbai	Bangalore	221
7	Mumbai	Kolkata	215
8	Mumbai	Chennai	195
9	Mumbai	Hyderabad	192

Average Price of different Airlnes from Source city to Destination city

In [33]:
```
df.groupby(['Airline','Source','Destination'],as_index=False)['Fare'].mean().sort_valu
```

Out[33]:

	Airline	Source	Destination	Fare
0	Vistara	Kolkata	Mumbai	35827.490454
1	Vistara	Kolkata	Delhi	35236.037466
2	Vistara	Delhi	Kolkata	34471.920570
3	Vistara	Kolkata	Bangalore	33903.469274
4	Vistara	Bangalore	Kolkata	33137.275827
5	Vistara	Mumbai	Kolkata	32766.361448
6	Air India	Bangalore	Kolkata	32429.328096
7	Air India	Kolkata	Bangalore	32326.384379
8	Air India	Ahmedabad	Chennai	31986.554209
9	Vistara	Kolkata	Chennai	31906.405215

Feature Engineering

Encoding

In [34]:
```
FE=data.copy()
```

In [35]:
```python
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
for col in FE.columns:
    if FE[col].dtype=='object':
        FE[col]=le.fit_transform(FE[col])
```

Scaling

In [36]:
```python
from sklearn.preprocessing import MinMaxScaler,StandardScaler
```

In [37]:
```python
FE['Duration_in_hours']=FE['Duration_in_hours'].round(1)
```

In [38]:
```python
FE.head()
```

Out[38]:

	Date_of_journey	Journey_day	Airline	Flight_code	Class	Source	Departure	Total_stops	Arrival	De
0	0	1	6	1209	1	3	2	2	2	
1	0	1	5	164	1	3	2	2	3	
2	0	1	4	942	1	3	2	2	3	
3	0	1	6	1224	1	3	2	2	2	
4	0	1	0	852	1	3	2	2	2	

In [39]:
```python
FE.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 440087 entries, 0 to 440086
Data columns (total 13 columns):
 #   Column             Non-Null Count   Dtype
---  ------             --------------   -----
 0   Date_of_journey    440087 non-null  int32
 1   Journey_day        440087 non-null  int32
 2   Airline            440087 non-null  int32
 3   Flight_code        440087 non-null  int32
 4   Class              440087 non-null  int32
 5   Source             440087 non-null  int32
 6   Departure          440087 non-null  int32
 7   Total_stops        440087 non-null  int32
 8   Arrival            440087 non-null  int32
 9   Destination        440087 non-null  int32
 10  Duration_in_hours  440087 non-null  float64
 11  Days_left          440087 non-null  int64
 12  Fare               440087 non-null  int64
dtypes: float64(1), int32(10), int64(2)
memory usage: 26.9 MB
```

In [40]:
```python
mms = MinMaxScaler() # Normalization

# Normalization
FE['Duration_in_hours'] = mms.fit_transform(FE[['Duration_in_hours']])
```

In [41]:
```
FE
```
Out[41]:

	Date_of_journey	Journey_day	Airline	Flight_code	Class	Source	Departure	Total_stops	Arriva
0	0	1	6	1209	1	3	2	2	[illegible]
1	0	1	5	164	1	3	2	2	[illegible]
2	0	1	4	942	1	3	2	2	[illegible]
3	0	1	6	1224	1	3	2	2	[illegible]
4	0	1	0	852	1	3	2	2	[illegible]
...	...	...	...	...	...	...	...	...	..
440082	49	1	8	1386	0	0	2	0	[illegible]
440083	49	1	8	1367	0	0	1	0	[illegible]
440084	49	1	8	1358	0	0	3	0	[illegible]
440085	49	1	8	1374	0	0	1	0	[illegible]
440086	49	1	8	1360	0	0	1	0	[illegible]

440087 rows × 13 columns

Correlation

In [42]:
```
corr = FE.corrwith(data['Fare']).sort_values(ascending = False).to_frame()
corr.columns = ['Fare']

plt.subplots(figsize = (5,5))
sns.heatmap(corr,annot = True,cmap = colors,linewidths = 0.4,linecolor = 'black');

plt.title('Fare Correlation');
```

We will create 2 Datasets :

- **1st** : Based on the statistical and EDA , we will drop the following features : 'Date_of_journey', 'Journey_day', 'Airline','Source', 'Departure','Arrival', 'Destination', 'Days_left'
- **2nd** : We will use all features

In [43]:
```
df1=FE.copy()
df2=FE.copy()

# Dataset for model based on Statistical Test :
df1 = df1.drop(columns = ['Date_of_journey', 'Journey_day', 'Airline','Source', 'Depa
```

Model

In [44]:
```
f1 = df1.iloc[:,:4].values
t1 = df1.iloc[:,4].values
f2 = df2.iloc[:,:12].values
t2 = df2.iloc[:,12].values
```

In [45]:
```
x_train1, x_test1, y_train1, y_test1 = train_test_split(f1, t1, test_size = 0.15, ran
x_train2, x_test2, y_train2, y_test2 = train_test_split(f2, t2, test_size = 0.15, ran
```

In [46]:
```
def model(regression,x_train,y_train,x_test,y_test):
    regression.fit(x_train,y_train)
    prediction = regression.predict(x_test)
    print("MAPE Score : ", (np.mean(np.abs((y_test - prediction)/y_test))*100))
```

XGB

In [47]:
```
Regressor_xgb = XGBRegressor(random_state=1)

model(Regressor_xgb,x_train1,y_train1,x_test1,y_test1)
```

```
MAPE Score :  20.642650288800915
```

In [48]:
```
model(Regressor_xgb,x_train2,y_train2,x_test2,y_test2)
```

```
MAPE Score :  14.001368031928349
```

LGBM

In [49]:
```
Regressor_LGBM = LGBMRegressor(random_state=1)

model(Regressor_LGBM,x_train1,y_train1,x_test1,y_test1)
```

```
MAPE Score :  23.195415291350354
```

In [50]:
```
model(Regressor_LGBM,x_train2,y_train2,x_test2,y_test2)
```

```
MAPE Score :  17.328064339817942
```

Gradient Booster

In [51]:
```
Regressor_grad = GradientBoostingRegressor(random_state=1)

model(Regressor_grad,x_train1,y_train1,x_test1,y_test1)
```

```
MAPE Score :  25.210125029384766
```

In [52]:
```
model(Regressor_grad,x_train2,y_train2,x_test2,y_test2)
```

```
MAPE Score :  21.847673452188594
```

Hyperparameter Tuning

From these results it is found that Dataset 2 shows better results and XGB is the best model

In [58]:
```
pip install flaml
```

```
Collecting flaml
  Downloading FLAML-1.2.3-py3-none-any.whl (256 kB)
Requirement already satisfied: xgboost>=0.90 in d:\app\anaconda\lib\site-packages (f
rom flaml) (1.7.5)
Requirement already satisfied: scikit-learn>=0.24 in d:\app\anaconda\lib\site-packag
es (from flaml) (1.2.2)
Requirement already satisfied: lightgbm>=2.3.1 in d:\app\anaconda\lib\site-packages
(from flaml) (3.3.5)
Requirement already satisfied: scipy>=1.4.1 in d:\app\anaconda\lib\site-packages (fr
om flaml) (1.8.0)
Requirement already satisfied: NumPy>=1.17.0rc1 in d:\app\anaconda\lib\site-packages
(from flaml) (1.24.3)
Requirement already satisfied: pandas>=1.1.4 in d:\app\anaconda\lib\site-packages (f
rom flaml) (1.3.4)
Requirement already satisfied: wheel in d:\app\anaconda\lib\site-packages (from ligh
tgbm>=2.3.1->flaml) (0.37.0)
Requirement already satisfied: pytz>=2017.3 in d:\app\anaconda\lib\site-packages (fr
om pandas>=1.1.4->flaml) (2021.3)
Requirement already satisfied: python-dateutil>=2.7.3 in d:\app\anaconda\lib\site-pa
ckages (from pandas>=1.1.4->flaml) (2.8.2)
Requirement already satisfied: six>=1.5 in d:\app\anaconda\lib\site-packages (from p
ython-dateutil>=2.7.3->pandas>=1.1.4->flaml) (1.16.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in d:\app\anaconda\lib\site-pack
ages (from scikit-learn>=0.24->flaml) (2.2.0)
Requirement already satisfied: joblib>=1.1.1 in d:\app\anaconda\lib\site-packages (f
rom scikit-learn>=0.24->flaml) (1.2.0)
Installing collected packages: flaml
Successfully installed flaml-1.2.3
Note: you may need to restart the kernel to use updated packages.
```

In [54]:
```
# from flaml import AutoML

# automl = AutoML()
# settings = {
#     "time_budget": 120,  # total running time in seconds
#     "metric": 'mape',  # primary metrics for regression can be chosen from: ['mae',
#     "estimator_list": ['xgboost'],  # list of ML learners; we tune lightgbm in this
#     "task": 'regression',  # task type
#     "log_file_name": '/content/drive/MyDrive/projek/fare pred/xg4.log',  # flaml lo
#     "seed": 1,    # random seed
# }
# automl.fit(X_train=x_train2, y_train=y_train2, **settings)
```

After training for **2 minutes the best results** were obtained and we saved them as files

In [55]:
```
# ''' pickle and save the automl object '''

# with open('/content/drive/MyDrive/projek/fare pred/fare_pred.pkl', 'wb') as f:
#     pickle.dump(automl, f, pickle.HIGHEST_PROTOCOL)
```

In [56]:
```
def MAPE(y_true, y_pred):
    y_true, y_pred = np.array(y_true), np.array(y_pred)
    return np.mean((np.abs((y_true - y_pred) / y_true)) * 100).round(2)
```

In [60]:
```
tuned_model = pd.read_pickle(r'fare_pred.pkl')
```

In [61]:
```
XGB_Tuned = tuned_model.predict(x_test2)
print("MAPE Score : ", (np.mean(np.abs((y_test2 - XGB_Tuned)/y_test2))*100))
```

```
MAPE Score :  8.779263207342018
```