```
import numpy as np
import pandas as pd

import plotly.express as px
import plotly.graph_objs as go
import matplotlib.pyplot as plt
import seaborn as sns
sns.set_style('ticks')
sns.set_palette("viridis")

from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split

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


import warnings
warnings.filterwarnings('ignore')


from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

Reading the File

```
flights=pd.read_csv('/content/drive/MyDrive/Clean_Dataset.csv')


flights.head(10)
```

	Unnamed: 0	airline	flight	source_city	departure_time	stops	arrival_time	destination_city	class	duration	days_
0	0	SpiceJet	SG-8709	Delhi	Evening	zero	Night	Mumbai	Economy	2.17	
1	1	SpiceJet	SG-8157	Delhi	Early_Morning	zero	Morning	Mumbai	Economy	2.33	
2	2	AirAsia	I5-764	Delhi	Early_Morning	zero	Early_Morning	Mumbai	Economy	2.17	
3	3	Vistara	UK-995	Delhi	Morning	zero	Afternoon	Mumbai	Economy	2.25	
4	4	Vistara	UK-963	Delhi	Morning	zero	Morning	Mumbai	Economy	2.33	
5	5	Vistara	UK-945	Delhi	Morning	zero	Afternoon	Mumbai	Economy	2.33	
6	6	Vistara	UK-027	Delhi	Morning	zero	Morning	Mumbai	Economy	2.08	

Exploratory data analysis

```
flights.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
```

```
# Checking for null values in every column
flights.isnull().sum()
```

```
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
```

The dataset doesn't contain any null values

```
# Removing the 'Unnamed' column
flights = flights.drop('Unnamed: 0', axis = 1)
```

```
flights.head()
```

	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

```
flights.describe()
```

	duration	days_left	price
count	300153.000000	300153.000000	300153.000000
mean	12.221021	26.004751	20889.660523
std	7.191997	13.561004	22697.767366
min	0.830000	1.000000	1105.000000
25%	6.830000	15.000000	4783.000000
50%	11.250000	26.000000	7425.000000
75%	16.170000	38.000000	42521.000000
max	49.830000	49.000000	123071.000000

Insights

1. The average duration of flights in the dataset is 11.2 and the maximum duration is 49.8 and the minimum duration is 0.8.
2. The average days left of flights in the dataset is 26 and the maximum days left are 49 and the minimum days left are 1.
3. The average price of flights in the dataset is 7425 and the maximum price are 123071 and the minimum price are 1105.

```
flights.duplicated().sum()
```

```
0
```

```
pd.DataFrame({"Missing values (%)" : round(flights.isnull().sum()/len(flights), 2)})
```

	Missing values (%)
airline	0.0
flight	0.0
source_city	0.0
departure_time	0.0
stops	0.0
arrival_time	0.0
destination_city	0.0
class	0.0
duration	0.0
days_left	0.0
price	0.0

```
for i in flights:
    print(i)
    print(flights[i].unique(), "\n")
```

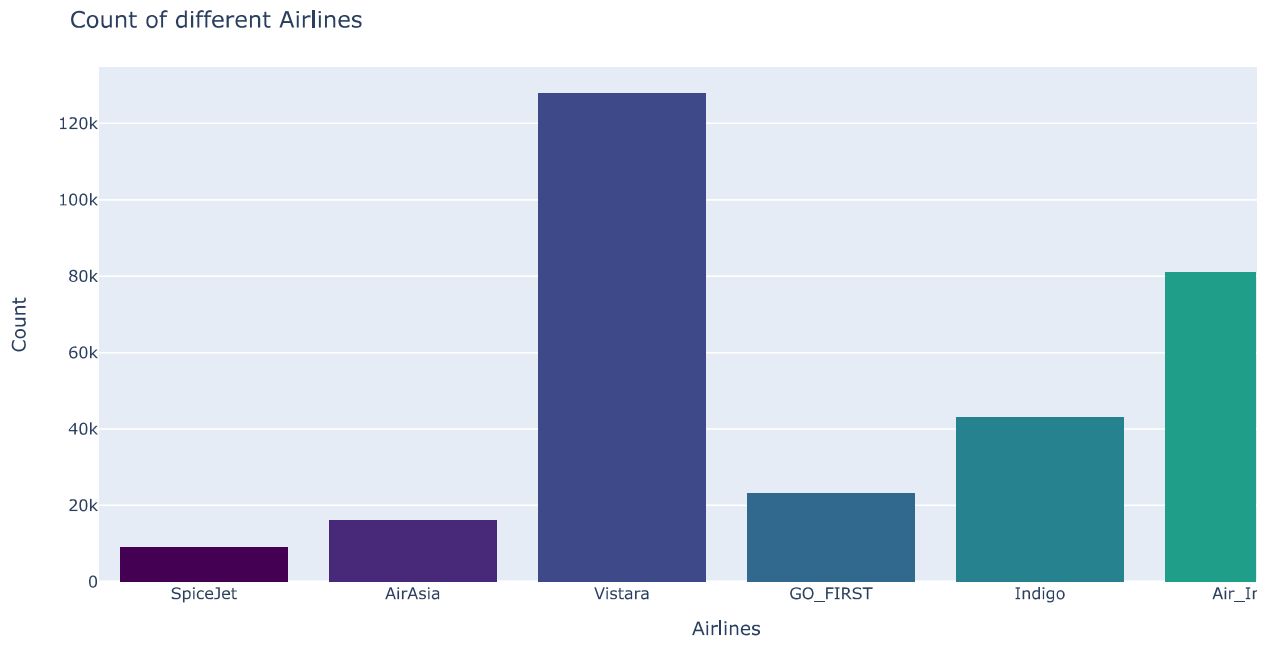
```
airline
['SpiceJet' 'AirAsia' 'Vistara' 'GO_FIRST' 'Indigo' 'Air_India']

flight
['SG-8709' 'SG-8157' 'I5-764' ... '6E-7127' '6E-7259' 'AI-433']

source_city
['Delhi' 'Mumbai' 'Bangalore' 'Kolkata' 'Hyderabad' 'Chennai']

departure_time
['Evening' 'Early_Morning' 'Morning' 'Afternoon' 'Night' 'Late_Night']

stops
['zero' 'one' 'two_or_more']

arrival_time
['Night' 'Morning' 'Early_Morning' 'Afternoon' 'Evening' 'Late_Night']

destination_city
['Mumbai' 'Bangalore' 'Kolkata' 'Hyderabad' 'Chennai' 'Delhi']

class
['Economy' 'Business']

duration
[ 2.17  2.33  2.25  2.08 12.25 16.33 11.75 14.5  15.67  3.75  2.5   5.83
  8.    6.   14.67 16.17 18.   23.17 24.17  8.83  4.5  15.25 11.   19.08
 22.83 26.42 17.75 19.58 26.67 15.17 20.83 11.42 22.25 26.   21.75  3.83
  4.42  7.67  8.33 10.42 23.75 19.5   6.5  12.42 21.08 28.17 28.25  9.25
 17.92  7.08 13.83  7.58 15.83 24.42  4.17  4.25  5.08 29.33 17.   27.17
 24.75  5.75 12.75 13.75 17.83  5.5  23.83  5.   26.5  12.83  8.92 11.17
 12.17 15.58 15.75  7.92 13.25 16.   22.75  6.33  7.25 30.08 18.25  6.08
  2.   12.33  3.5  10.25 14.17 25.58  4.08  9.75  6.67  9.67 10.08 12.58
  7.    8.25 15.5  10.17 23.5  25.75 11.5  21.42 14.25  7.75  5.33  5.67
  4.75 19.33  6.25 10.33  9.08 15.42  4.83 25.83  7.5  27.58 28.42  6.42
 24.58 16.5  11.33 24.83 14.92 26.08  5.42 28.5  27.33  8.67 20.42 20.08
  5.92 20.58 26.17 17.5  18.33 21.83 26.83 13.42 19.25 23.58 23.92 12.08
 14.42 25.92 18.58 13.08  4.58 21.58 10.5  11.67  8.58 24.67 14.75 17.17
  7.33  9.17 11.58 23.08 25.25 21.25  8.42 10.92 14.    4.92 25.08 12.67
 28.08 21.33 14.33 14.83 15.33 15.92  9.5  27.75 24.92 22.58  6.58  9.33
 13.   13.58 11.08  5.58 15.    6.75  9.83 11.25 11.83  9.58  8.5  14.08
 10.58 22.17 22.08  8.75 15.08 25.5  16.42 18.92  2.42  3.92 24.5  23.
 10.   19.   16.58  7.83  3.17 16.25 24.08  4.67  6.92 13.67 25.    8.08
 17.08  7.42 22.5  12.92  6.17 23.67  5.17 13.33 17.25 25.42 29.17  9.
  9.42  8.17 26.58 22.33 12.5  21.   24.25 27.5  18.17 16.67  2.83  2.75
  2.92  3.    7.17 25.67  2.67 10.83 18.5  21.17 23.33 24.   17.33 27.42
 26.25  4.33  9.92 19.67 26.33 16.92 25.33 27.25 27.92 27.   12.   10.67
 28.   21.67 21.92 13.92 13.5   2.58 22.42 24.33 18.08 29.58 29.67 31.25
 33.17 36.92 20.25 22.92  6.83  5.25 20.67 17.58 19.17 17.42 11.92 31.5
 19.83 27.08 26.92 21.5   6.07 16.83 20.5  18.67 10.75 18.42 28.58 19.92
 20.   22.   29.83 19.42 25.17  3.08 18.75 27.83 20.17 20.33 20.75 28.67
 35.83 34.83 23.25 26.75 19.75 27.67 16.08 30.25 28.75 30.58 18.83 13.17
 30.5  29.5  28.83 30.67 17.67 14.58 29.   28.33 39.67 16.75  1.92 37.42
 36.42 23.42 22.67 33.75 30.83 28.92  1.83  3.67  4.    1.58  1.67  1.75
```

```
20.92 33.5  30.33 35.25 31.83 40.   31.67 32.08 36.17 30.75 33.83 34.5
31.08 30.17 36.08 33.33 37.   34.08 37.08  1.42  1.33  1.5  37.83 29.25
34.25 36.83  3.42 31.58 33.58 34.58 32.17  1.17 35.17 37.67 37.25 39.75
31.   37.75 33.25 31.17 29.92 31.92 41.58 37.17 32.   35.5  39.   31.42
```

We have checked above if there are any duplicated or missing values in the dataset or not. The dataset is clean and the values inside the features are also fine. Let's move forward.

Univariate Analysis

```
fig=px.histogram(flights, x='airline', color='airline', color_discrete_sequence=px.colors.sequential.Viridis)
fig.update_layout(title_text='Count of different Airlines', xaxis_title='Airlines', yaxis_title='Count')
fig.show()
```



Most of the flights are from **'Vistara'** airlines

```
counts = flights['flight'].value_counts().nlargest(10)
top_flights = flights[flights['flight'].isin(counts.index)]
fig = px.histogram(top_flights, x='flight', color='flight', color_discrete_sequence=px.colors.sequential.Viridis)
fig.update_layout(title_text='Count of top 10 flights', xaxis_title='Flight', yaxis_title='Count')
fig.show()
```

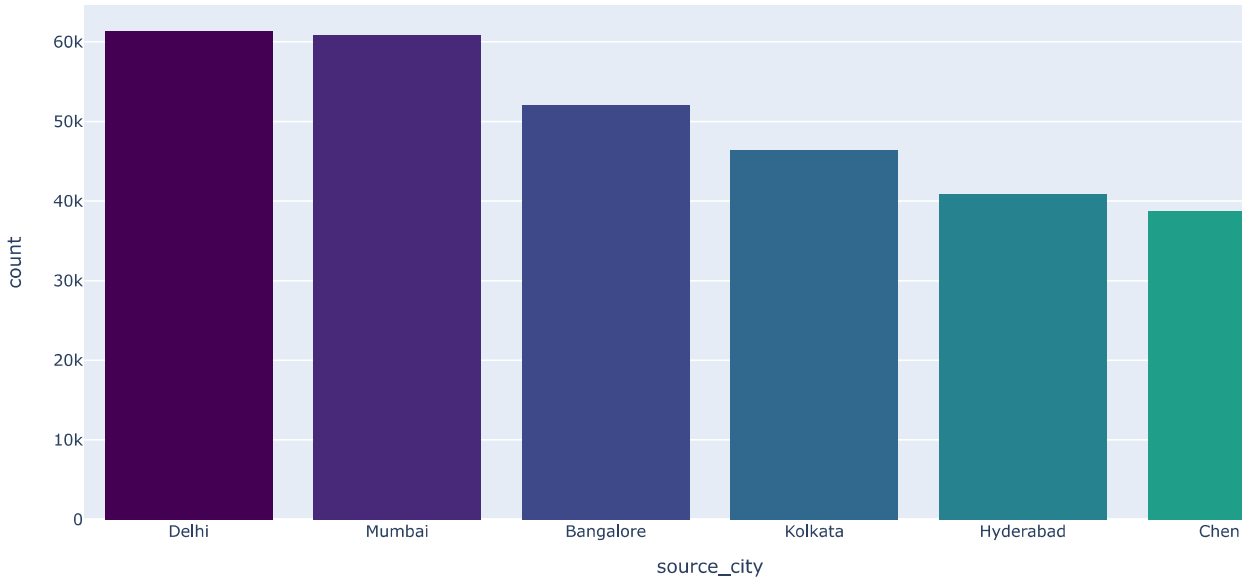

```
flights.head()
```

	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

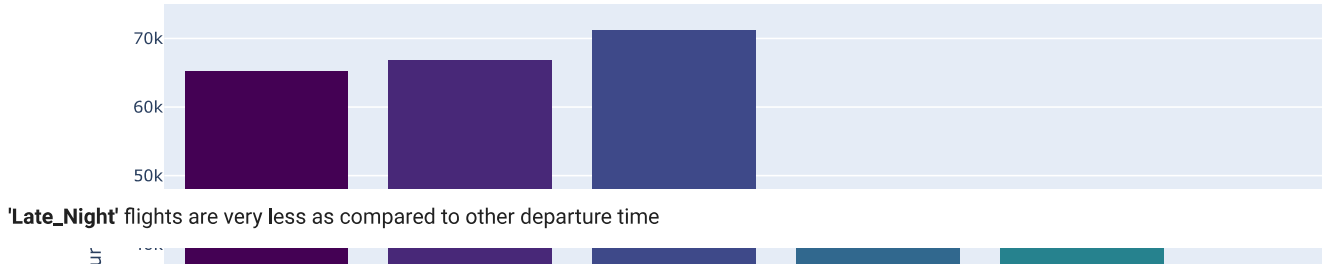
```
flights['source_city'].unique()
```

```
array(['Delhi', 'Mumbai', 'Bangalore', 'Kolkata', 'Hyderabad', 'Chennai'],
      dtype=object)
```

```
px.histogram(flights,x='source_city',color='source_city',color_discrete_sequence=px.colors.sequential.Viridis)
```

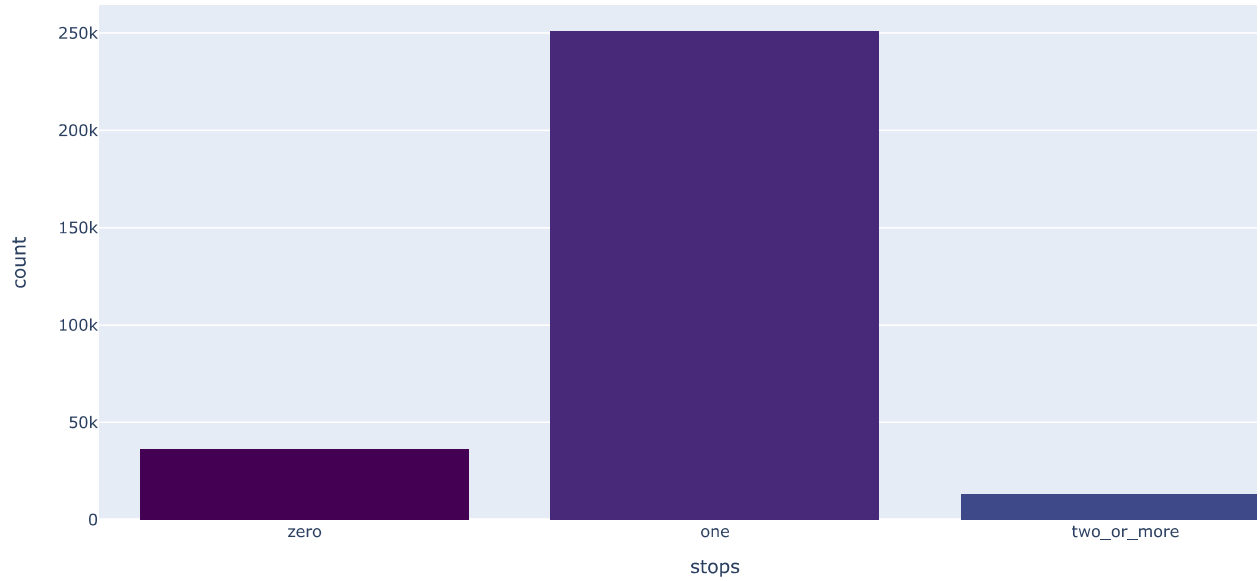



```
px.histogram(flights, x='departure_time', color='departure_time', color_discrete_sequence=px.colors.sequential.Viridis)
```



'**Late_Night**' flights are very less as compared to other departure time

```
px.histogram(flights, x='stops', color='stops', color_discrete_sequence=px.colors.sequential.Viridis)
```

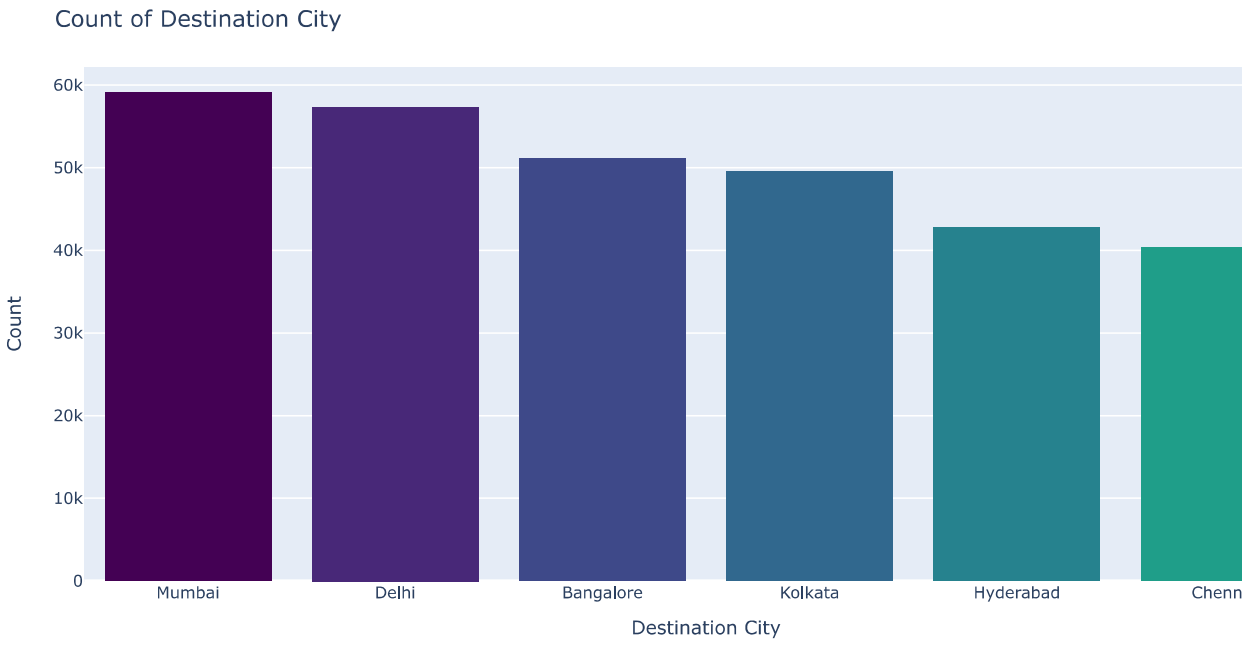


Most of the flights have **one stop**

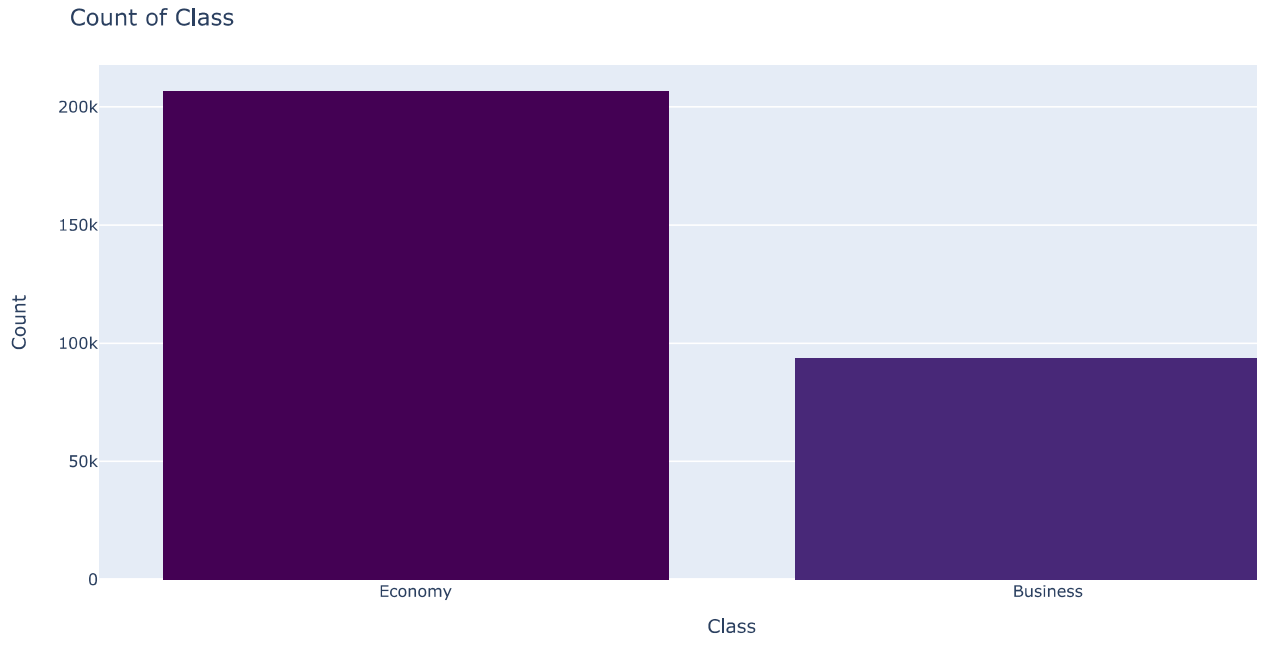
```
px.histogram(flights, x='arrival_time', color='arrival_time', color_discrete_sequence=px.colors.sequential.Viridis)
```

'Early_Morning' and 'Late_Night' arrivals are very less

```
counts = flights['destination_city'].value_counts()
sorted_destination_city = counts.sort_values(ascending = False).index.tolist()
fig = px.histogram(flights, x='destination_city', color='destination_city', color_discrete_sequence=px.colors.sequential.Viridis, category_or
fig.update_layout(title_text='Count of Destination City', xaxis_title='Destination City', yaxis_title='Count')
fig.show()
```




```
counts = flights['class'].value_counts()
sorted_class = counts.sort_values(ascending = False).index.tolist()
fig = px.histogram(flights, x='class', color='class', color_discrete_sequence=px.colors.sequential.Viridis, category_orders={"class": sorted_
fig.update_layout(title_text='Count of Class', xaxis_title='Class', yaxis_title='Count')
fig.show()
```



Economy class are 2x than Business class

Insights

1. Among all the airlines in the dataset, Vistara has the highest number of flights, while SpiceJet has the lowest.
2. UK-706 is the most frequently flown flight among the top 10 flights, whereas UK-860 is the least flown.
3. Delhi is the most popular source city for flights, while Chennai is the least popular.
4. Early morning is the most common departure time among people, with only a few choosing late night departures.
5. Majority of the flights have only one stop, while only a small number have 2 or more stops.
6. The most common arrival time for people is at night, with only a few arriving late night.
7. Mumbai is the most popular destination city for flights, while Chennai is the least popular.
8. Economy class is the most preferred travel class, with Business class being less popular.

Multivariate Analysis

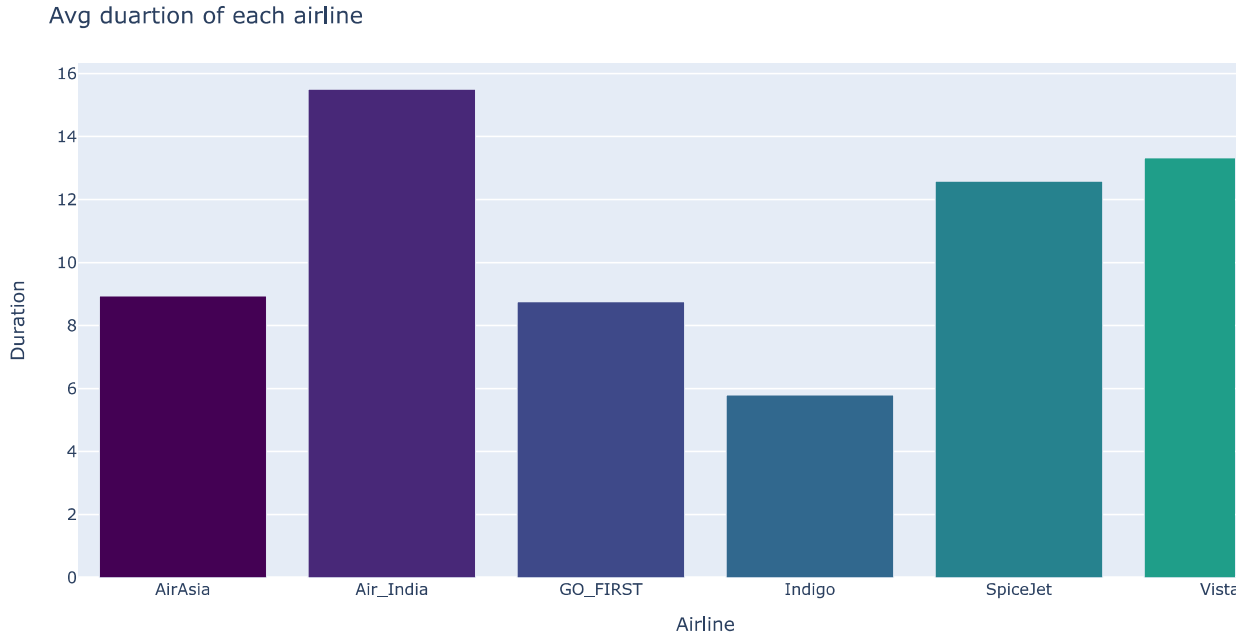
Analysis according to duration

Q1. What is the average duration of each airline?

```
airline_duration = flights.groupby('airline').mean()['duration']
airline_duration
```

```
airline
AirAsia       8.941714
Air_India    15.504235
GO_FIRST      8.755380
Indigo        5.795197
SpiceJet     12.579767
Vistara      13.326634
Name: duration, dtype: float64
```

```
airline_duration = flights.groupby('airline')['duration'].mean().reset_index()
fig = px.bar(airline_duration, x='airline', y='duration', color='airline', color_discrete_sequence=px.colors.sequential.Viridis)
fig.update_layout(title_text='Avg duartion of each airline', xaxis_title='Airline', yaxis_title='Duration')
fig.show()
```



What is the average duration of top 10 Flights?

```
flight_duration = flights.groupby('flight').mean()['duration']
flight_duration
```

```
flight
6E-102     3.208718
6E-105     7.500000
6E-113     5.660612
6E-121     5.645000
6E-123     6.750000
             ...
UK-988    16.382245
UK-993     8.829067
UK-994     9.325067
UK-995    10.471875
UK-996    18.559123
Name: duration, Length: 1561, dtype: float64
```

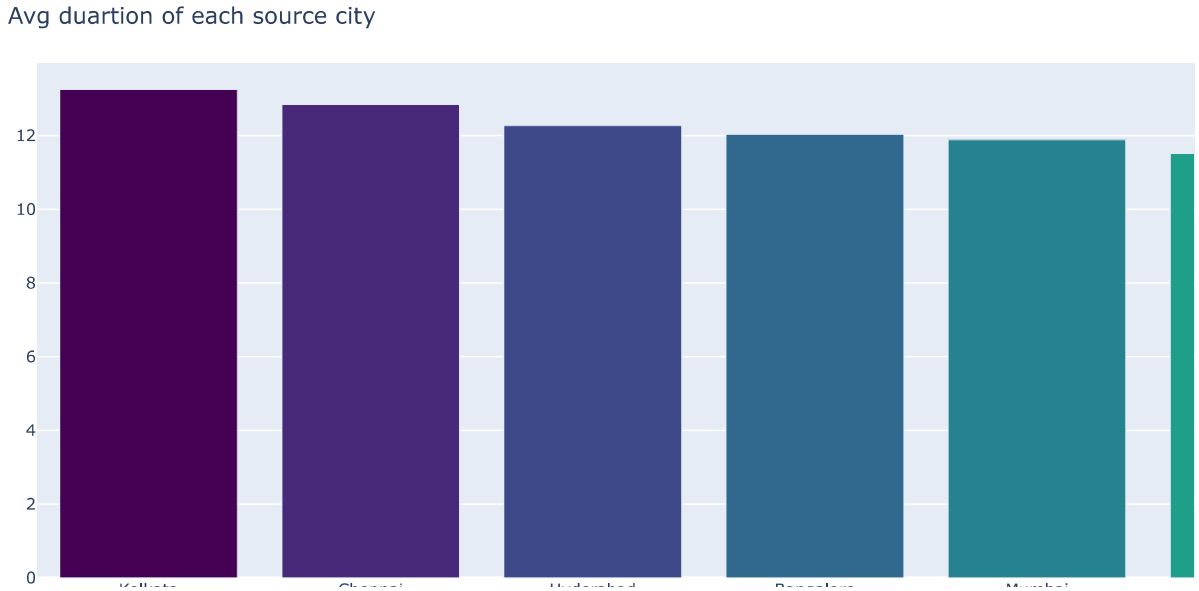
```
flight_duration = flights.groupby('flight').mean()['duration'].reset_index()
top10_flight_duration = flight_duration.nlargest(10, 'duration')
fig = px.bar(top10_flight_duration, x='flight', y='duration', color='flight', color_discrete_sequence=px.colors.sequential.Viridis)
fig.update_layout(title_text='Avg duration of top 10 flights', xaxis_title='Flight', yaxis_title='Duration')
fig.show()
```

What is the average duration of each source city?

```
flights.groupby('source_city').mean()['duration']
```

```
source_city
Bangalore    12.029203
Chennai      12.838901
Delhi        11.515499
Hyderabad    12.268075
Kolkata      13.249898
Mumbai       11.888448
Name: duration, dtype: float64
```

```
source_city_duration = flights.groupby('source_city')['duration'].mean().sort_values(ascending = False).reset_index()
fig = px.bar(source_city_duration, x = 'source_city', y = 'duration', color = 'source_city', color_discrete_sequence=px.colors.sequential.Vir
fig.update_layout(title_text='Avg duartion of each source city', xaxis_title='Source City', yaxis_title='Duration')
fig.show()
```

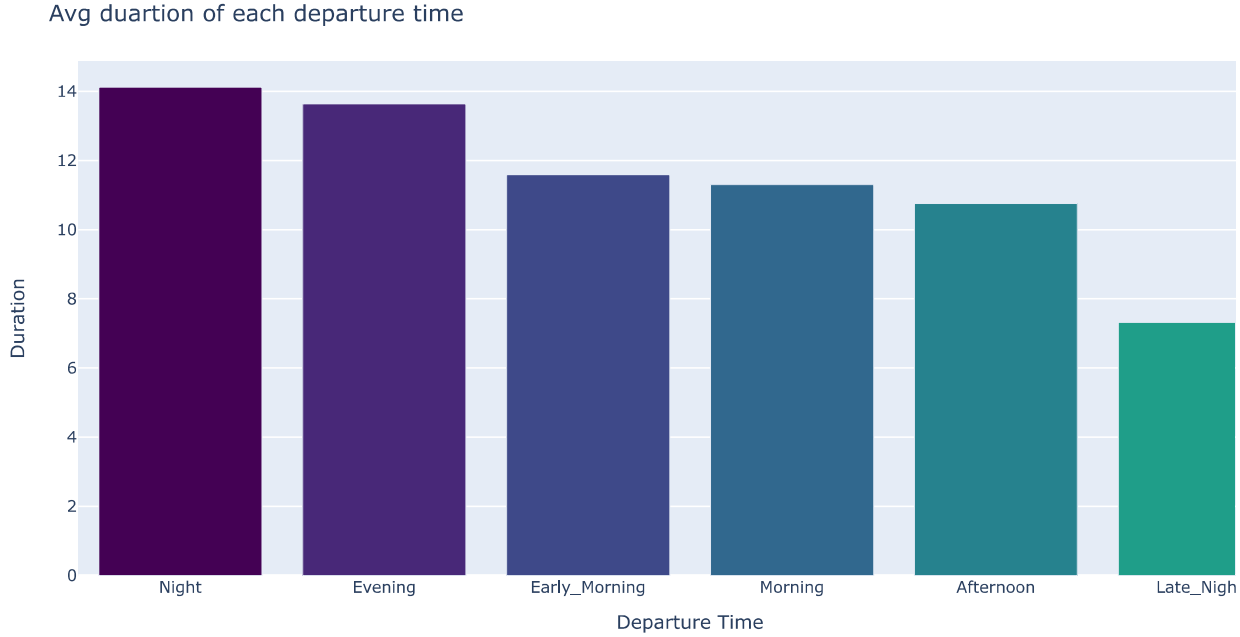


What is the average duration of each departure time?

```
flights.groupby('departure_time')['duration'].mean()
```

```
departure_time
Afternoon        10.757008
Early_Morning    11.587945
Evening          13.642493
Late_Night        7.320383
Morning          11.303808
Night            14.123963
Name: duration, dtype: float64
```

```
source_city_duration = flights.groupby('departure_time')['duration'].mean().sort_values(ascending = False).reset_index()
fig = px.bar(source_city_duration, x = 'departure_time', y = 'duration', color = 'departure_time', color_discrete_sequence=px.colors.sequenti
fig.update_layout(title_text='Avg duartion of each departure time', xaxis_title='Departure Time', yaxis_title='Duration')
fig.show()
```

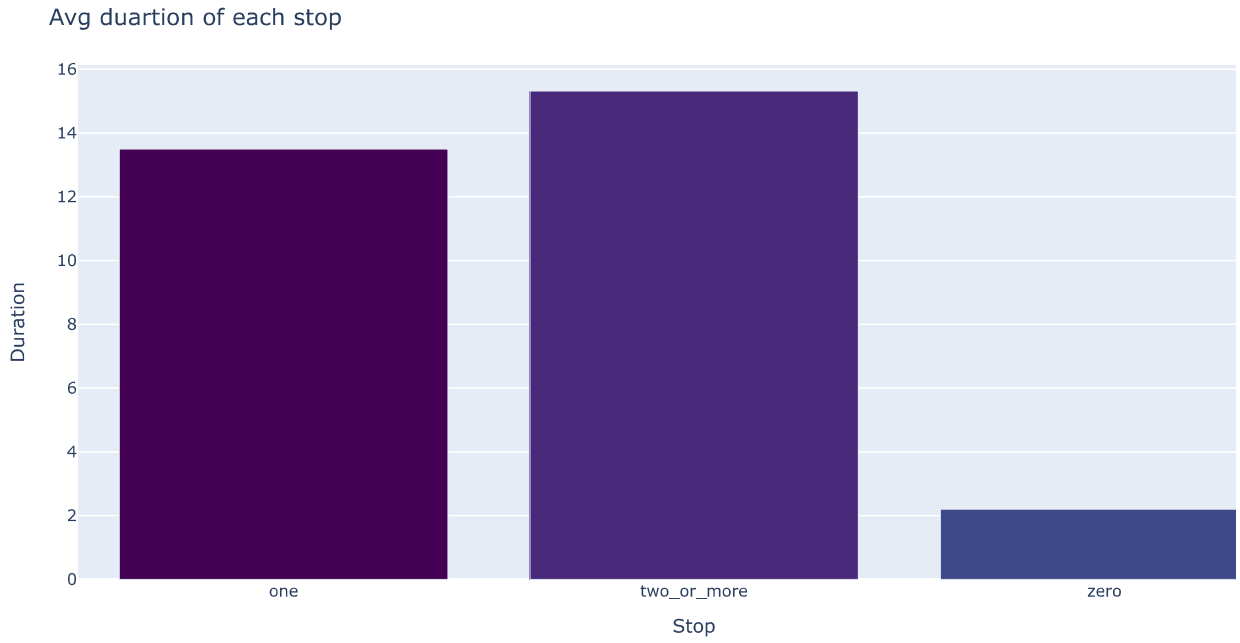


What is the average duration of each stop?

```
flights.groupby('stops')['duration'].mean()
```

```
stops
one            13.496514
two_or_more    15.317141
zero            2.191324
Name: duration, dtype: float64
```

```
source_city_duration = flights.groupby('stops')['duration'].mean().reset_index()
fig = px.bar(source_city_duration, x = 'stops', y = 'duration', color = 'stops', color_discrete_sequence=px.colors.sequential.Viridis)
fig.update_layout(title_text='Avg duartion of each stop', xaxis_title='Stop', yaxis_title='Duration')
fig.show()
```

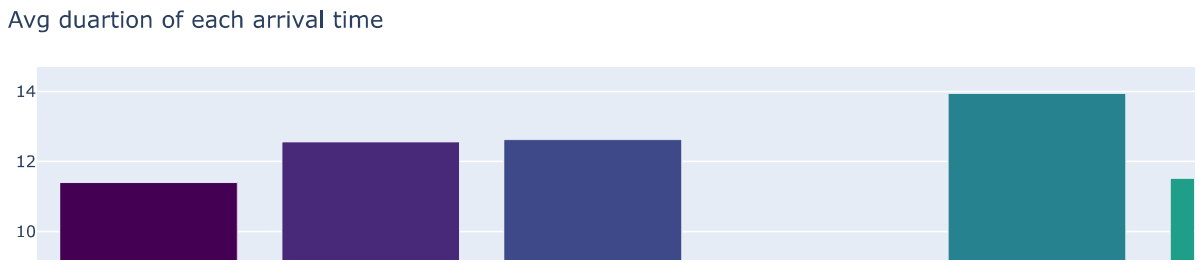


What is the average duration of each arrival time?

```
flights.groupby('arrival_time')['duration'].mean()
```

```
arrival_time
Afternoon        11.393490
Early_Morning    12.559448
Evening          12.621578
Late_Night        8.732261
Morning          13.940608
Night            11.521187
Name: duration, dtype: float64
```

```
source_city_duration = flights.groupby('arrival_time')['duration'].mean().reset_index()
fig = px.bar(source_city_duration, x = 'arrival_time', y = 'duration', color = 'arrival_time', color_discrete_sequence=px.colors.sequential.V
fig.update_layout(title_text='Avg duartion of each arrival time', xaxis_title='Arrival Time', yaxis_title='Duration')
fig.show()
```



What is the average duration of each destination city?

```
flights.groupby('destination_city')['duration'].mean()
```

```
destination_city
Bangalore    12.058039
Chennai      13.338900
Delhi        10.513310
Hyderabad    13.381945
Kolkata      13.214953
Mumbai       11.583355
Name: duration, dtype: float64
```

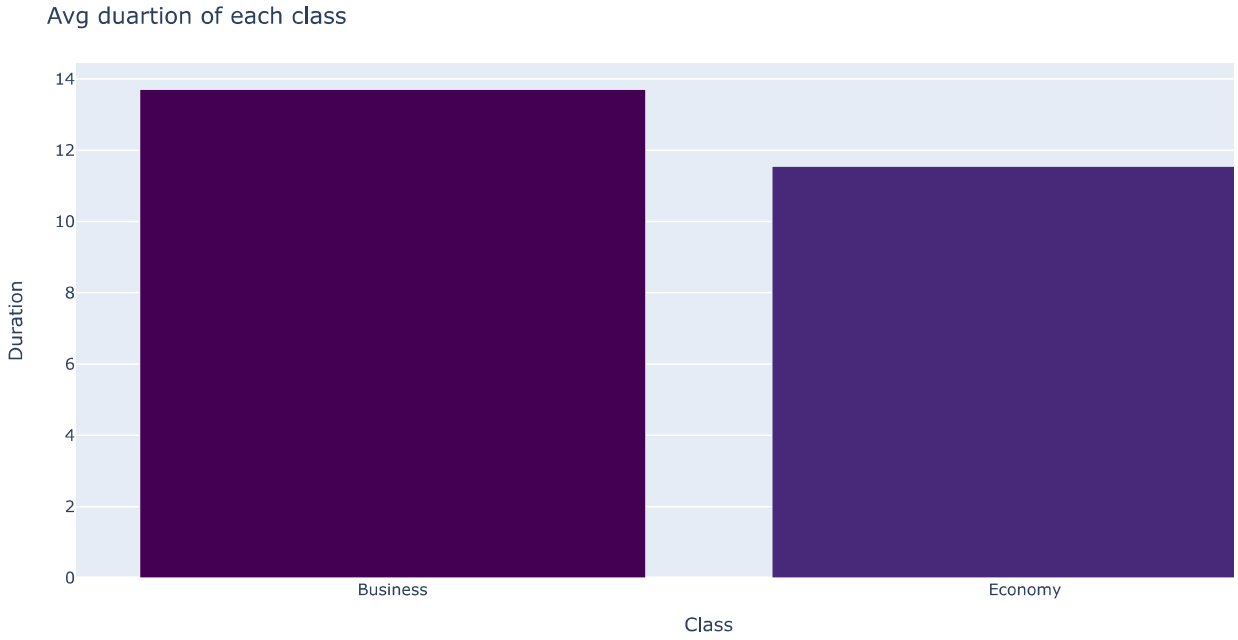
```
source_city_duration = flights.groupby('destination_city')['duration'].mean().reset_index()
fig = px.bar(source_city_duration, x = 'destination_city', y = 'duration', color = 'destination_city', color_discrete_sequence=px.colors.sequ
fig.update_layout(title_text='Avg duartion of each destination city', xaxis_title='Destination City', yaxis_title='Duration')
fig.show()
```

What is the average duration of each class?

```
flights.groupby('class')['duration'].mean()
```

```
class
Business    13.704274
Economy     11.550060
Name: duration, dtype: float64
```

```
source_city_duration = flights.groupby('class')['duration'].mean().sort_values(ascending = False).reset_index()
fig = px.bar(source_city_duration, x = 'class', y = 'duration', color = 'class', color_discrete_sequence=px.colors.sequential.Viridis)
fig.update_layout(title_text='Avg duartion of each class', xaxis_title='Class', yaxis_title='Duration')
fig.show()
```



Insights

Air India has the highest average flight duration among all airlines in the dataset, while **Indigo** has the lowest.

Among the **top 10** flights with the highest average duration, **UK-627** has the **highest** and **AI-721** has the **lowest** average duration.

Flights with **no stops** have the shortest average duration, while flights with **2 or more** stops have the longest average duration.

Morning arrival time has the **highest average flight duration**, while late night has the lowest. Night departure time has the highest average duration, while late night has the lowest.

Hyderabad has the longest average flight duration as a destination city, while **Kolkata** has the longest average flight duration as a source city.

Delhi has the shortest average duration for both destination and source cities.

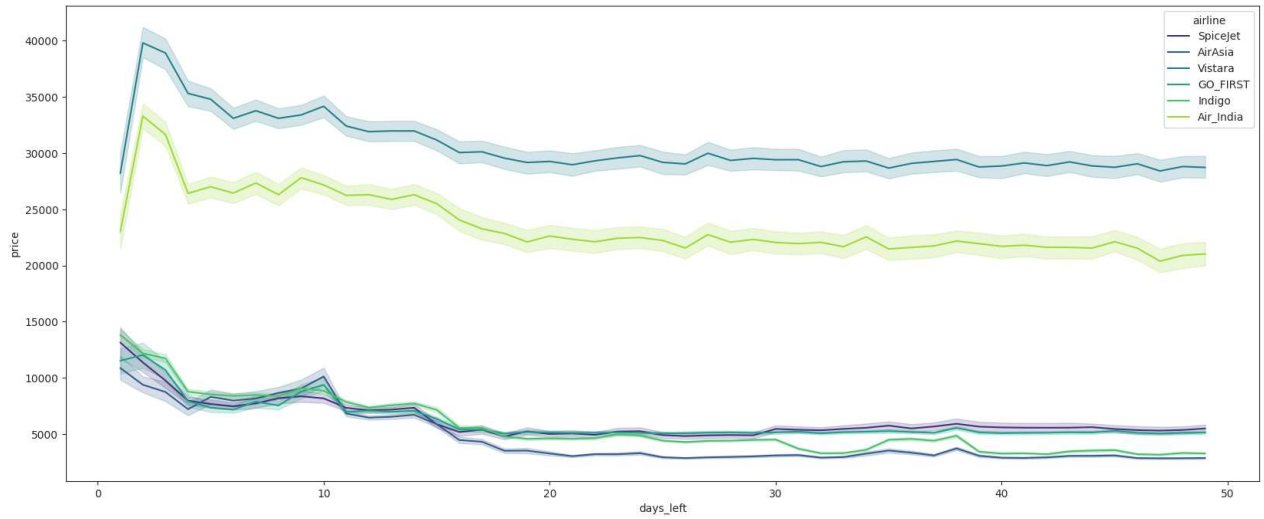
Economy class has a lower average flight duration than Business class.

Analysis according to Days Left

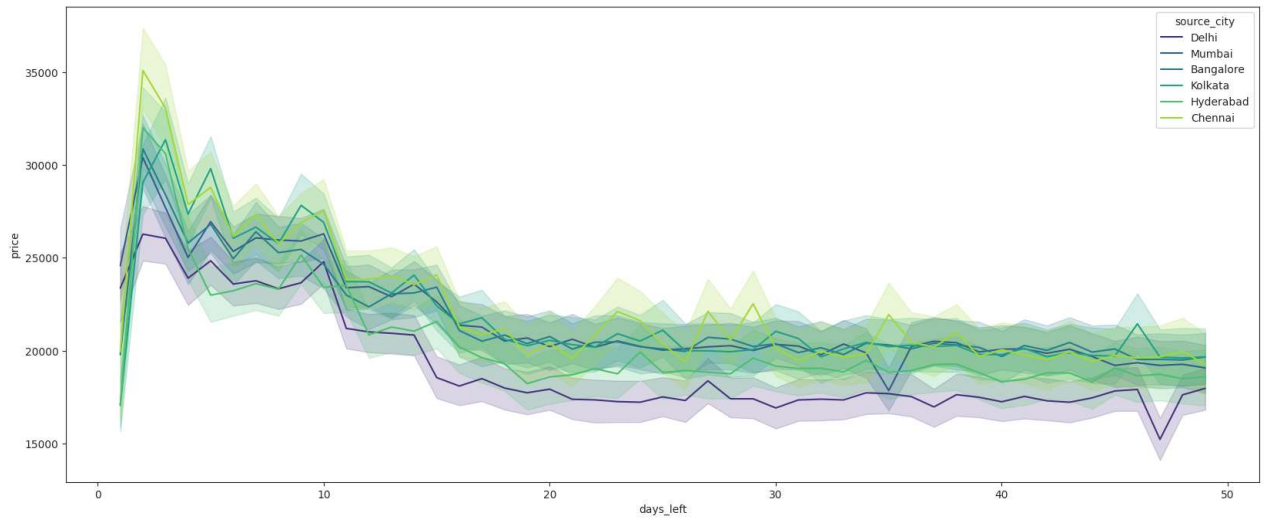
How does the price vary according to days left?

```
fig, ax = plt.subplots(figsize=(20, 8))
sns.lineplot(data=flights, x='days_left', y='price', ax=ax)
plt.show()
```

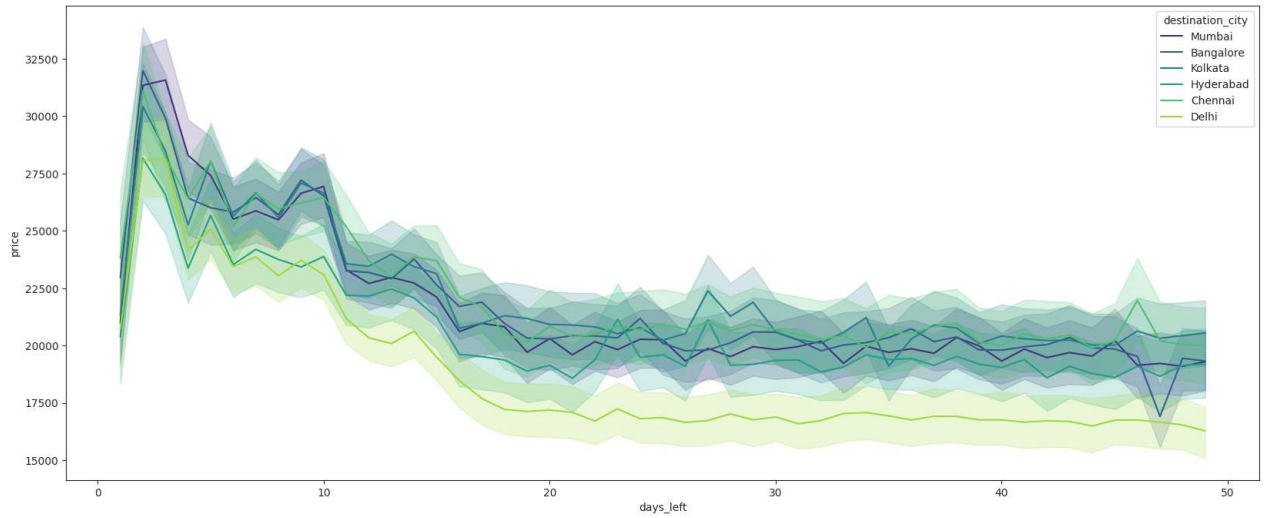

```
fig, ax = plt.subplots(figsize=(20, 8))
sns.lineplot(data=flights, x='days_left', y='price', hue = 'airline', ax=ax)

plt.show()
```



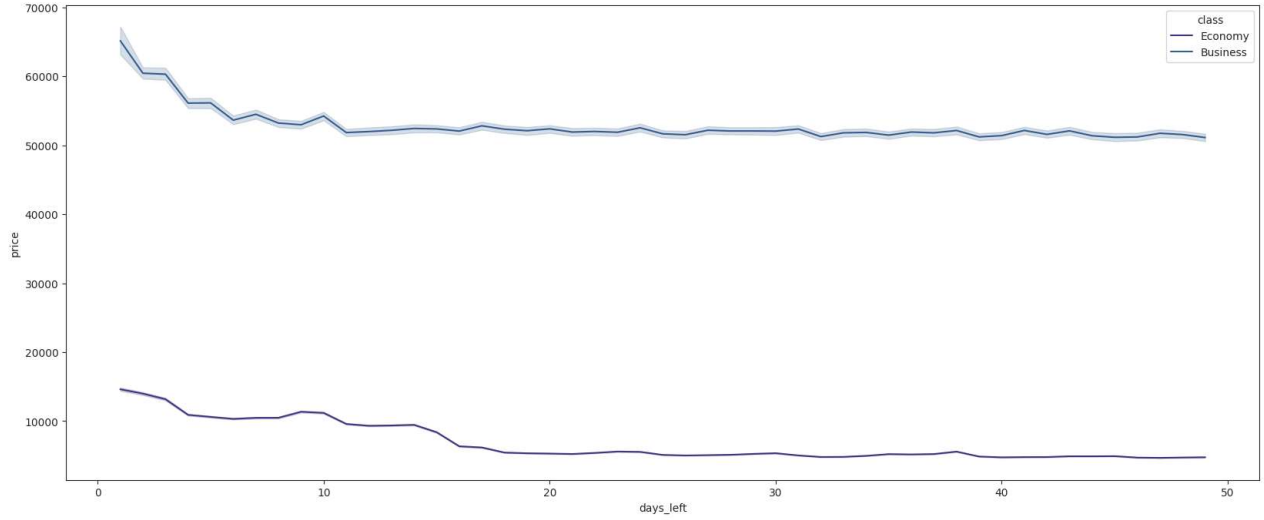

```
fig, ax = plt.subplots(figsize=(20, 8))
sns.lineplot(data=flights, x='days_left', y='price', hue = 'source_city', ax=ax)

plt.show()
```




```
fig, ax = plt.subplots(figsize=(20, 8))
sns.lineplot(data=flights, x='days_left', y='price', hue = 'destination_city', ax=ax)


plt.show()
```




```
fig, ax = plt.subplots(figsize=(20, 8))
sns.lineplot(data=flights, x='days_left', y='price', hue = 'class', ax=ax)

plt.show()
```



Insights

Prices for flights are highest when there are only 1-3 days left for departure, but decrease as the days left for departure increase.

Air India and Vistara are the most expensive airlines, and prices decrease as the days left for departure increase.

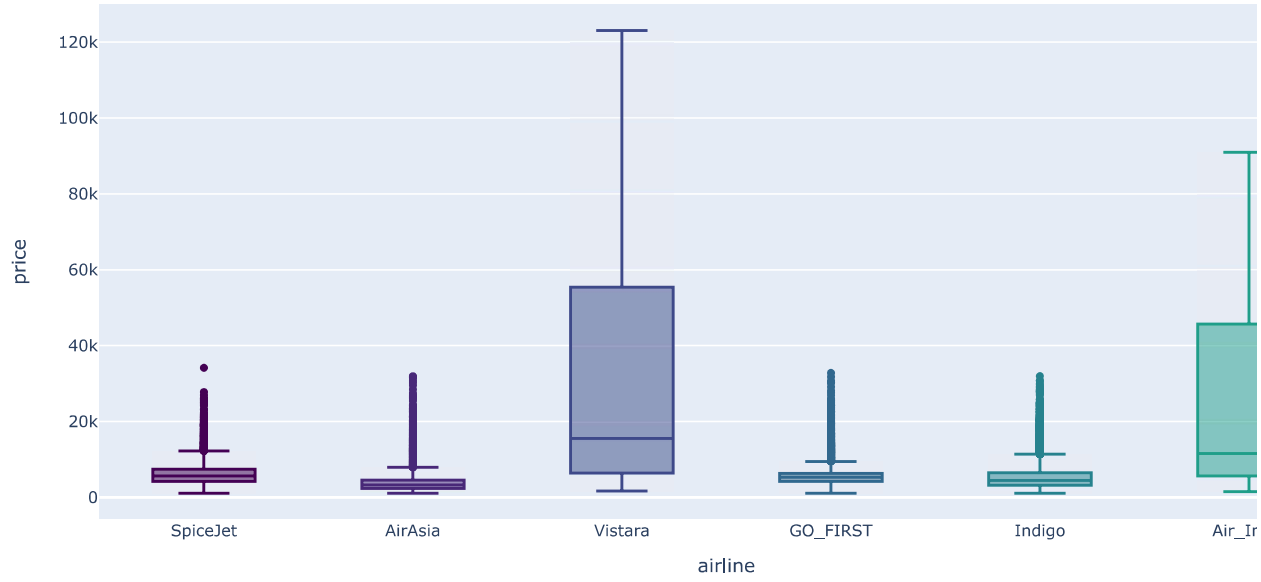
Late night departure times have lower prices compared to other departure times, but prices for late night arrival times are higher than evening arrivals.

Prices increase for various factors such as airline, source city, departure time, stops, arrival time, and class when there are only 1-2 days left for departure, but decrease as the days left for departure increase.

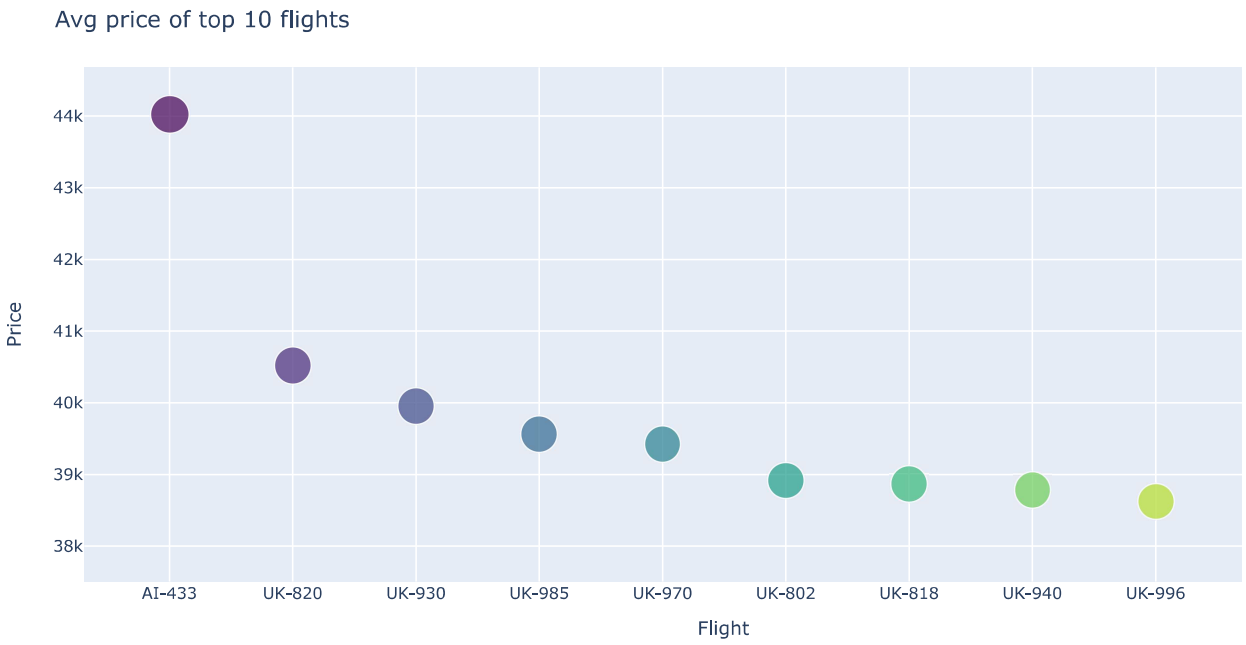
Analysis according to Price

```
fig = px.box(flights, x = 'airline', y = 'price', color = 'airline', color_discrete_sequence=px.colors.sequential.Viridis)
```

```
fig.show()
```

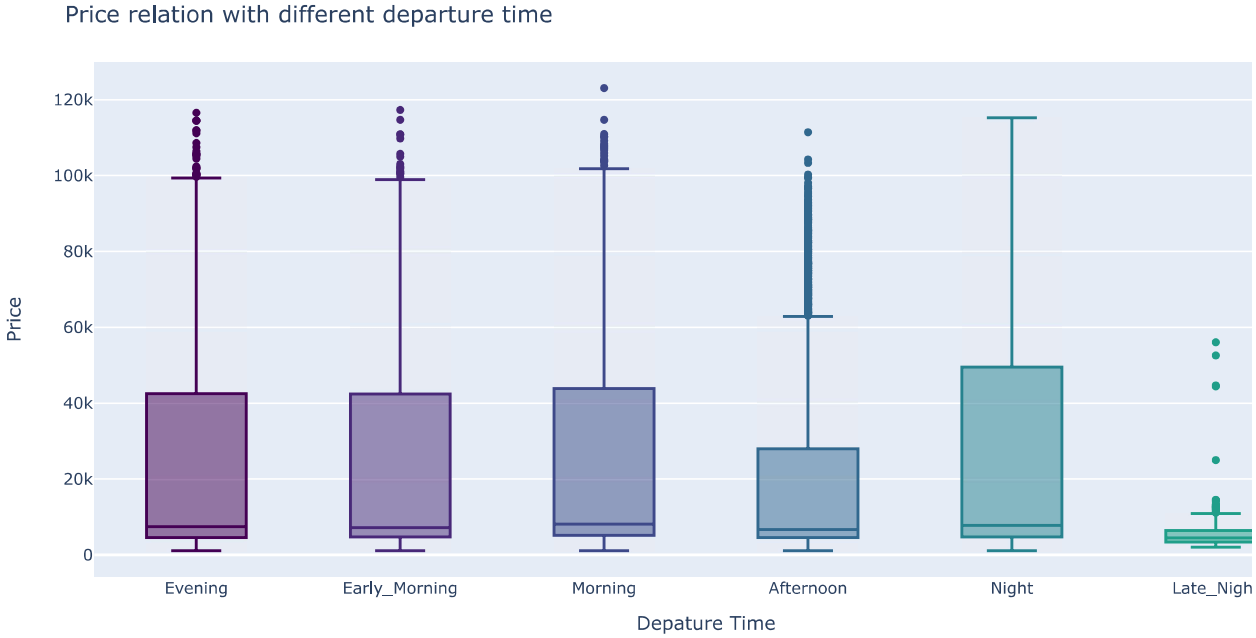



```
flight_price = flights.groupby('flight')['price'].mean().reset_index()
top10_flight_price = flight_price.nlargest(10, 'price')
fig = px.scatter(top10_flight_price, x='flight', y='price', color='flight', size = 'price', color_discrete_sequence=px.colors.sequential.Viri
fig.update_layout(title_text='Avg price of top 10 flights', xaxis_title='Flight', yaxis_title='Price')
fig.show()
```

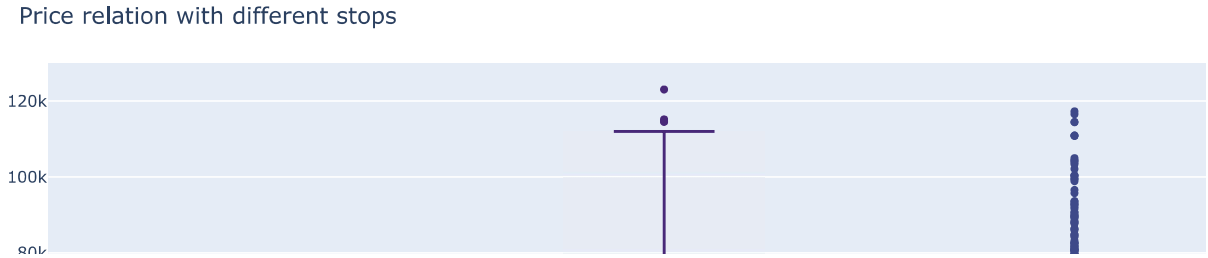



```
fig = px.box(flights, x = 'source_city', y = 'price', color = 'source_city', color_discrete_sequence=px.colors.sequential.Viridis)
fig.update_layout(title_text='Price relation with different source city', xaxis_title='Source City', yaxis_title='Price')
fig.show()
```

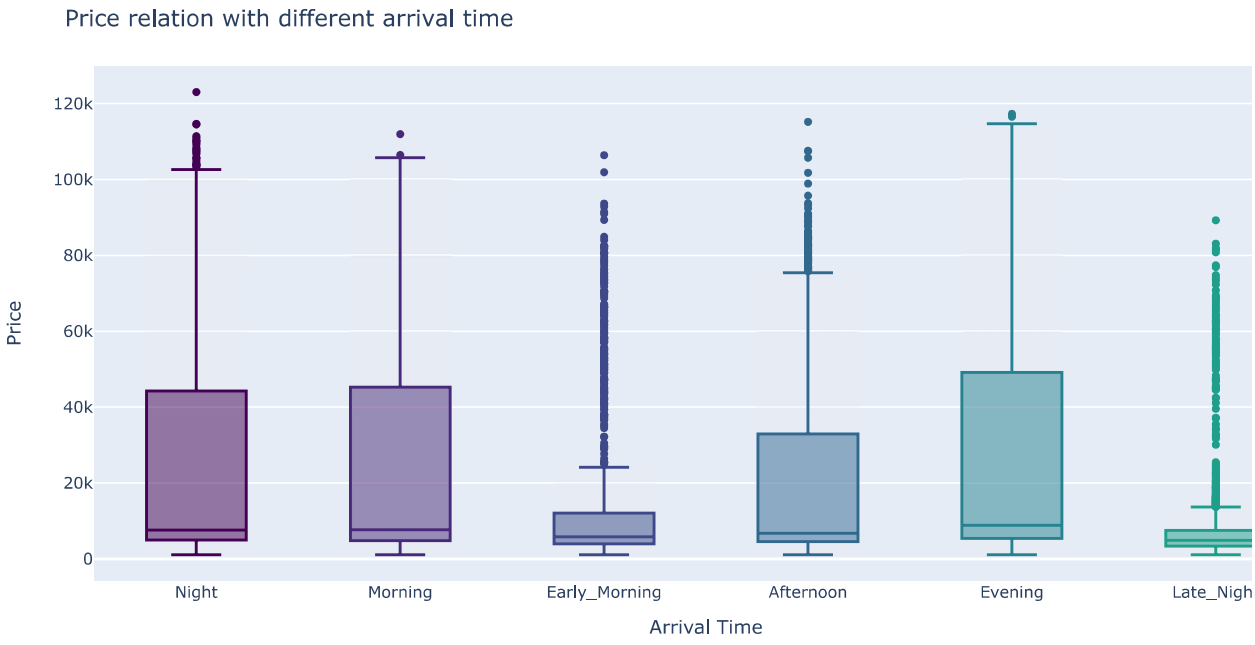

```
fig = px.box(flights, x = 'departure_time', y = 'price', color = 'departure_time', color_discrete_sequence=px.colors.sequential.Viridis)
fig.update_layout(title_text='Price relation with different departure time', xaxis_title='Depature Time', yaxis_title='Price')
fig.show()
```



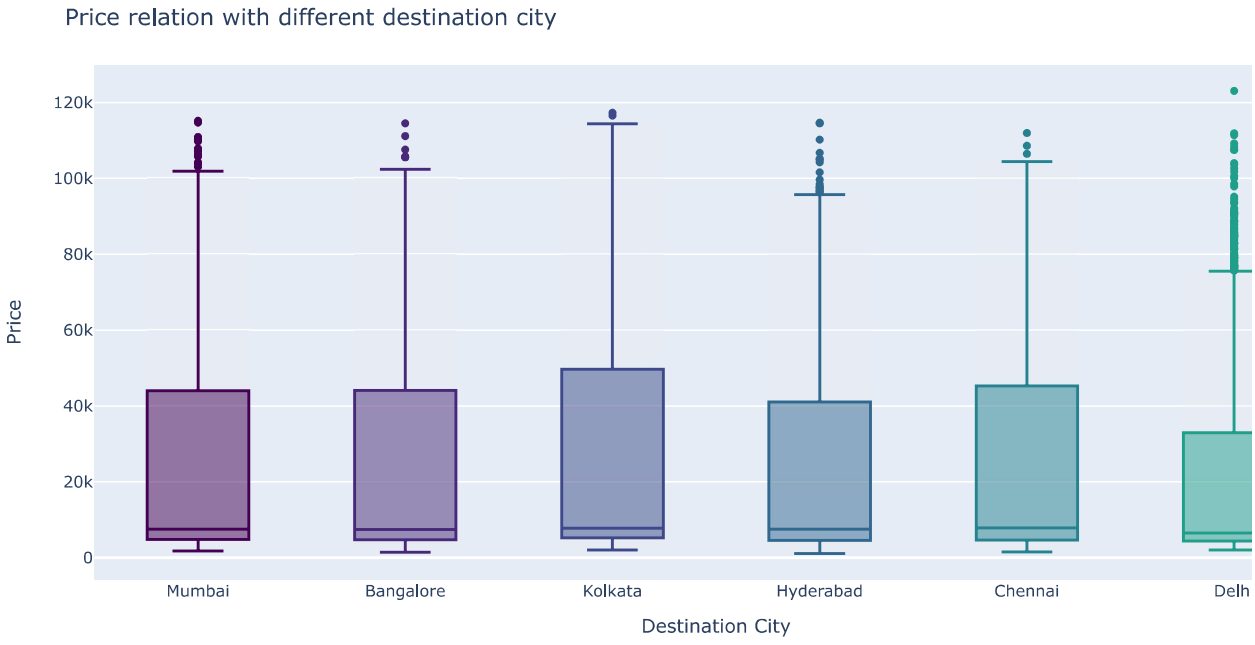

```
fig = px.box(flights, x = 'stops', y = 'price', color = 'stops', color_discrete_sequence=px.colors.sequential.Viridis)
fig.update_layout(title_text='Price relation with different stops', xaxis_title='Stops', yaxis_title='Price')
fig.show()
```



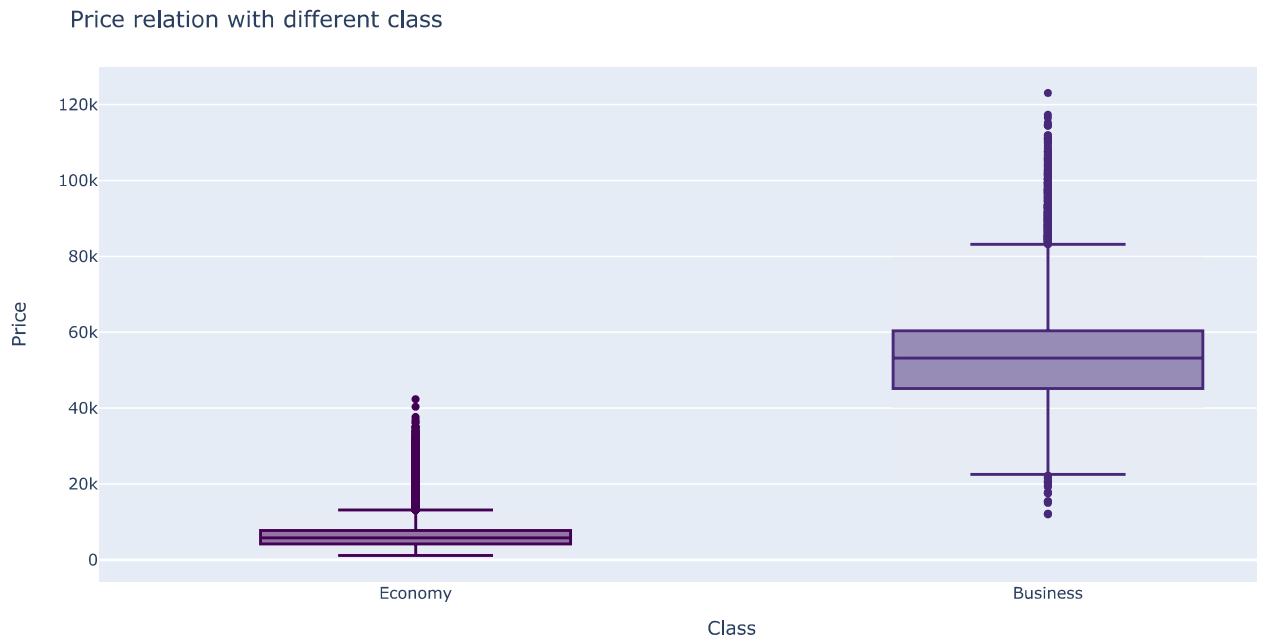

```
fig = px.box(flights, x = 'arrival_time', y = 'price', color = 'arrival_time', color_discrete_sequence=px.colors.sequential.Viridis)
fig.update_layout(title_text='Price relation with different arrival time', xaxis_title='Arrival Time', yaxis_title='Price')
fig.show()
```




```
fig = px.box(flights, x = 'destination_city', y = 'price', color = 'destination_city', color_discrete_sequence=px.colors.sequential.Viridis)
fig.update_layout(title_text='Price relation with different destination city', xaxis_title='Destination City', yaxis_title='Price')
fig.show()
```



```
fig = px.box(flights, x = 'class', y = 'price', color = 'class', color_discrete_sequence=px.colors.sequential.Viridis)
fig.update_layout(title_text='Price relation with different class', xaxis_title='Class', yaxis_title='Price')
fig.show()
```



Insights

Air India and Vistara are the most expensive airlines, while the prices for other airlines are similar.

AI-433 has the highest average price among the top 10 flights, while UK-814 has the lowest.

Kolkata is the source city with the highest average prices, while Delhi has the lowest.

Morning departure times have the highest average prices, while late night departure times have the lowest.

Flights with 2 or more stops have the highest average prices, while flights with no stops have the lowest.

Evening arrival times have the highest average prices, while late night arrival times have the lowest.

All destination cities have similar average prices, except for Kolkata which has the highest prices and Delhi which has the lowest.

Business class tickets have significantly higher average prices compared to economy class tickets.

Data Preprocessing

```
scaler = MinMaxScaler()
flights['duration'] = scaler.fit_transform(flights[['duration']])
```

Above i've used the min max scaler before encoding as this feature did not had a guassian distribution and is a contonous feature.

```
le = LabelEncoder()
flights = flights.apply(lambda x: le.fit_transform(x) if x.dtype == 'object' else x)
```

Above i've used the label encoding on all categorical features so that i can feed them to my machine leaning algorithms later

```
X = flights.drop(['price'], axis = 1)
y = flights['price']
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size = 0.7, random_state = 42)
```

Here i've splitted the data above intro training and testing part. I've kept 70% of my data for training purpose

```
scaler = MinMaxScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

Above i've used scaling technique once again for better performance of my machine learning algorithm.

Model Building

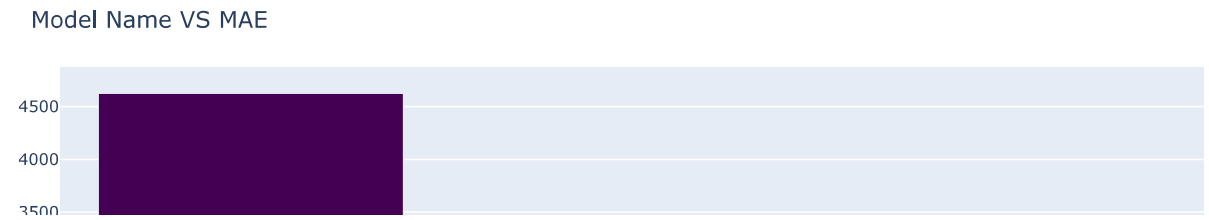
```
models = [LinearRegression(), DecisionTreeRegressor(), RandomForestRegressor()]

results = {'Model_Name': [], 'Mean_Absolute_Error_MAE': [], 'Mean_Absolute_Percentage_Error_MAPE': [],
           'Mean_Squared_Error_MSE': [], 'Root_Mean_Squared_Error_RMSE': [], 'R2_score': []}

for model in models:
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)

    mae = metrics.mean_absolute_error(y_test, y_pred)
    mape = np.mean(np.abs((y_test - y_pred) / y_test)) * 100
    mse = metrics.mean_squared_error(y_test, y_pred)
    rmse = np.sqrt(mse)
    r2 = metrics.r2_score(y_test, y_pred)

    results['Model_Name'].append(model.__class__.__name__)
    results['Mean_Absolute_Error_MAE'].append(mae)
    results['Mean_Absolute_Percentage_Error_MAPE'].append(mape)
    results['Mean_Squared_Error_MSE'].append(mse)
    results['Root_Mean_Squared_Error_RMSE'].append(rmse)
    results['R2_score'].append(r2)

Results = pd.DataFrame(results)
```

Model Evaluation

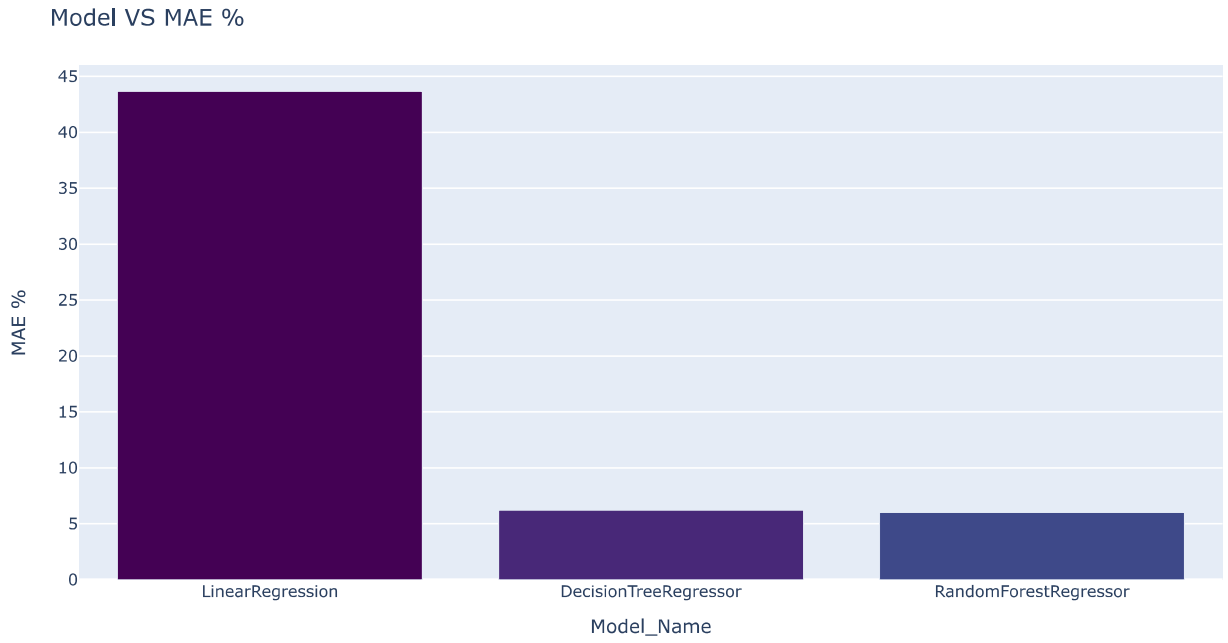
```
Results.head()
```

	Model_Name	Mean_Absolute_Error_MAE	Mean_Absolute_Percentage_Error_MAPE	Mean_Squared_Error_MSE	Root_Mean_Sc
0	LinearRegression	4623.408800	43.663631	4.906206e+07	
1	DecisionTreeRegressor	910.802373	6.211566	8.900364e+06	
2	RandomForestRegressor	893.335642	6.012623	5.647305e+06	

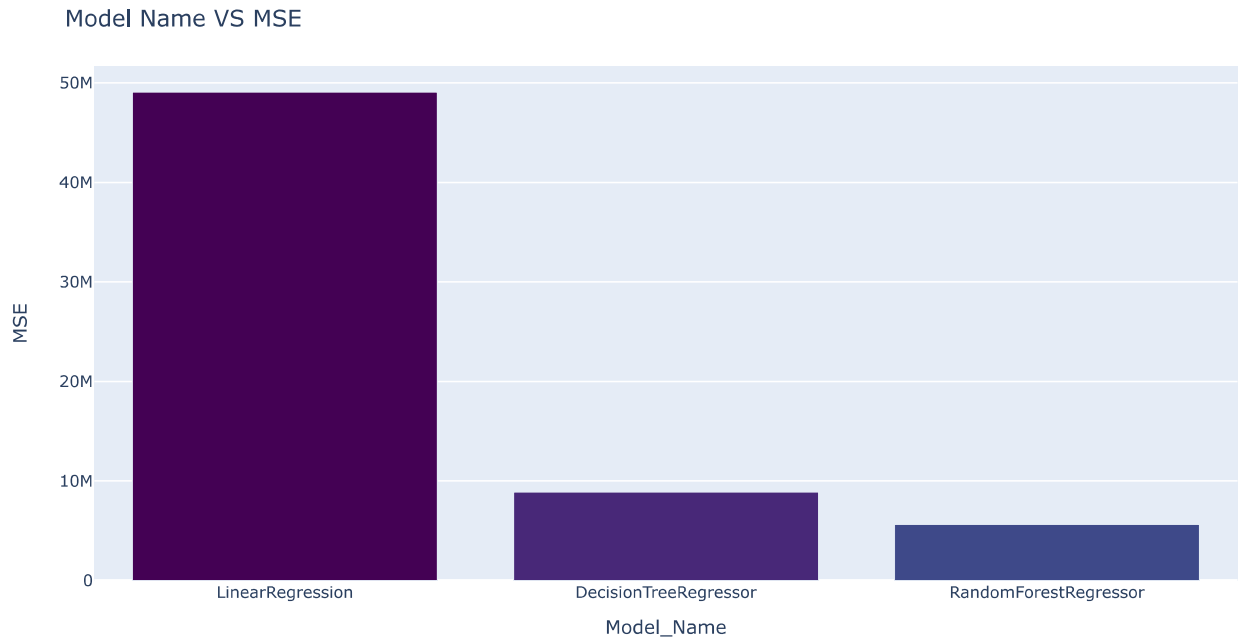
```
fig = px.bar(Results, x = 'Model_Name', y = 'Mean_Absolute_Error_MAE', color = 'Model_Name', color_discrete_sequence=px.colors.sequential.Vir
fig.update_layout(title_text='Model Name VS MAE', xaxis_title='Model_Name', yaxis_title='MAE')
fig.show()
```



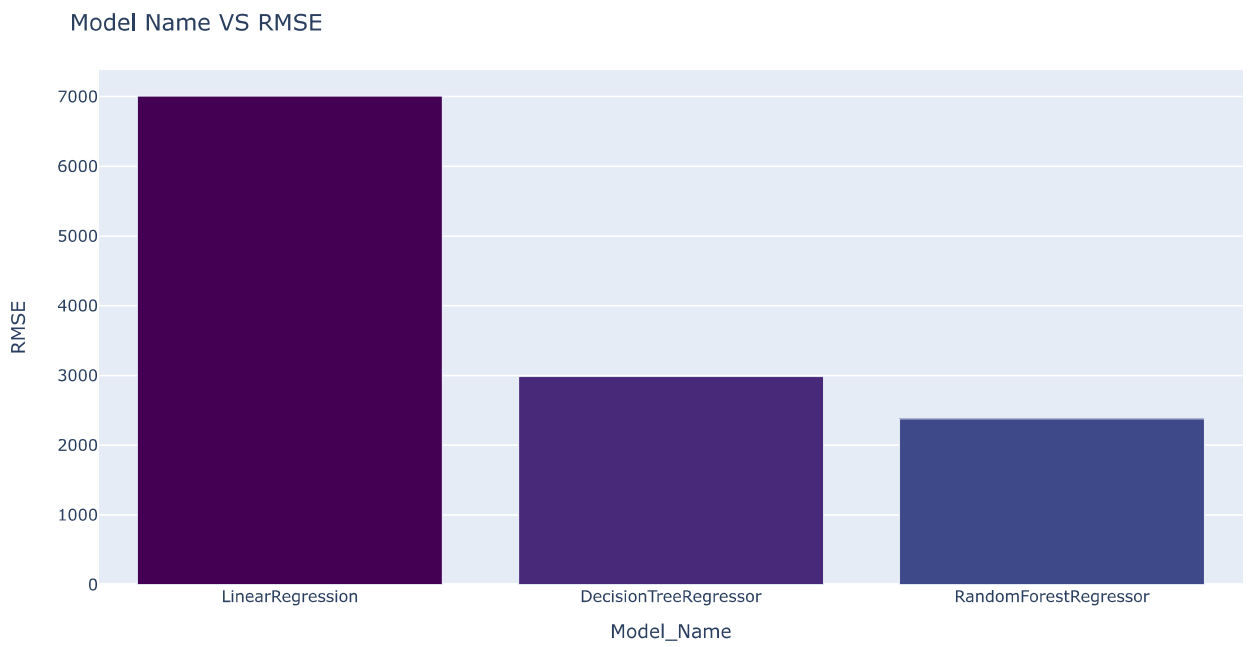

```
fig = px.bar(Results, x = 'Model_Name', y = 'Mean_Absolute_Percentage_Error_MAPE', color = 'Model_Name', color_discrete_sequence=px.colors.se
fig.update_layout(title_text='Model VS MAE %', xaxis_title='Model_Name', yaxis_title='MAE %')
fig.show()
```



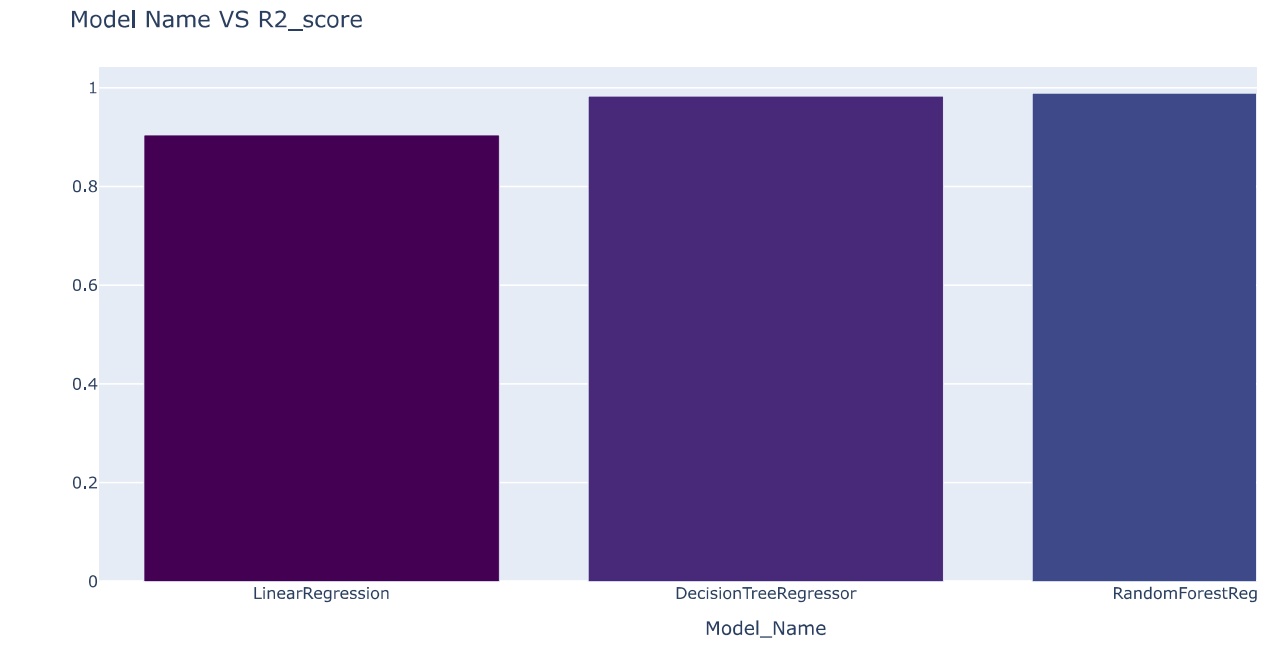

```
fig = px.bar(Results, x = 'Model_Name', y = 'Mean_Squared_Error_MSE', color = 'Model_Name', color_discrete_sequence=px.colors.sequential.Viri
fig.update_layout(title_text='Model Name VS MSE', xaxis_title='Model_Name', yaxis_title='MSE')
fig.show()
```



```
fig = px.bar(Results, x = 'Model_Name', y = 'Root_Mean_Squared_Error_RMSE', color = 'Model_Name', color_discrete_sequence=px.colors.sequentia
fig.update_layout(title_text='Model Name VS RMSE', xaxis_title='Model_Name', yaxis_title='RMSE')
fig.show()
```




```
fig = px.bar(Results, x = 'Model_Name', y = 'R2_score', color = 'Model_Name', color_discrete_sequence=px.colors.sequential.Viridis)
fig.update_layout(title_text='Model Name VS R2_score', xaxis_title='Model_Name', yaxis_title='R Squared')
fig.show()
```



Questions And their Answers from the Analysis:

a) Does price vary with Airlines?

Yes it does.

b) How is the price affected when tickets are bought in just 1 or 2 days before departure?

It is increased if bought just 1-2 days earlier than departure.

c) Does ticket price change based on the departure time and arrival time?

Yes only in morning, evenig and late night basis.

d) How the price changes with change in Source and Destination?

It doesn't change too much except for the city of Kolkata, Delhi and Hyderabad.

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

It varies too much as prices for business class are too high.

CONCLUSION

The dataset contains information on flight duration, days left for departure, price, airline, source city, departure and arrival times, number of stops, destination city, and travel class.

Among the airlines, Vistara has the highest number of flights, while SpiceJet has the lowest. Delhi is the most popular source city, and Mumbai is the most popular destination city. Economy class is the most preferred travel class, and Air India and Vistara are the most expensive airlines. Flights with no stops have the shortest average duration, while flights with 2 or more stops have the longest. Morning departure times and evening arrival times have the highest average prices, while late night departure and arrival times have the lowest. Business class tickets are more expensive than economy class tickets.

The Random Forest Regressor model performed the best with the least mean absolute error and highest r2 score.