1. Importing Libraries

We import the essential Python libraries for data analysis and visualization:

pandas, numpy ⟶ data manipulation and analysis

matplotlib, seaborn ⟶ static plots for EDA

plotly ⟶ interactive visualizations (scatter, bar, box, sunburst, etc.)bold text

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from plotly.subplots import make_subplots
import plotly.graph_objects as go
import plotly.express as px
```

2. Loading the Dataset

The dataset is loaded into a DataFrame named df using pd.read_csv(). Displaying it ensures that the data is correctly imported.

```
df=pd.read_csv(r"/content/airlines_flights_data.csv")
df
```

| | index | 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

3. Dataset Shape

We check the number of rows and columns using df.shape to understand dataset size.

```
df.shape
```

```
(300153, 12)
```

4. Checking Missing Values

We calculate the percentage of missing values in each column. This helps decide whether to drop, fill, or leave them.

```
df.isnull().sum()/len(df.index)
```

| | 0 |
|---|---|
| **index** | 0.0 |
| **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 |

**dtype:** float64

5. Dropping Unwanted Columns

The redundant "index" column is removed since it adds no analytical value.

```
df=df.drop(["index"],axis=1)
df
```

| | airline | flight | source_city | departure_time | stops | arrival_time | destination_city | class | duration | days_left | price |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | SpiceJet | SG-8709 | Delhi | Evening | zero | Night | Mumbai | Economy | 2.17 | 1 | 5953 |
| **1** | SpiceJet | SG-8157 | Delhi | Early_Morning | zero | Morning | Mumbai | Economy | 2.33 | 1 | 5953 |
| **2** | AirAsia | I5-764 | Delhi | Early_Morning | zero | Early_Morning | Mumbai | Economy | 2.17 | 1 | 5956 |
| **3** | Vistara | UK-995 | Delhi | Morning | zero | Afternoon | Mumbai | Economy | 2.25 | 1 | 5955 |
| **4** | Vistara | UK-963 | Delhi | Morning | zero | Morning | Mumbai | Economy | 2.33 | 1 | 5955 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **300148** | Vistara | UK-822 | Chennai | Morning | one | Evening | Hyderabad | Business | 10.08 | 49 | 69265 |
| **300149** | Vistara | UK-826 | Chennai | Afternoon | one | Night | Hyderabad | Business | 10.42 | 49 | 77105 |
| **300150** | Vistara | UK-832 | Chennai | Early_Morning | one | Night | Hyderabad | Business | 13.83 | 49 | 79099 |
| **300151** | Vistara | UK-828 | Chennai | Early_Morning | one | Evening | Hyderabad | Business | 10.00 | 49 | 81585 |
| **300152** | Vistara | UK-822 | Chennai | Morning | one | Evening | Hyderabad | Business | 10.08 | 49 | 81585 |

300153 rows × 11 columns

6. Dataset Information

Using df.info(), we get column names, data types, and non-null counts. This reveals which features are categorical and which are numerical.

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 300153 entries, 0 to 300152
Data columns (total 11 columns):
 #   Column            Non-Null Count   Dtype  
---  ------            --------------   -----  
 0   airline           300153 non-null  object 
 1   flight            300153 non-null  object 
 2   source_city       300153 non-null  object 
 3   departure_time    300153 non-null  object 
 4   stops             300153 non-null  object 
 5   arrival_time      300153 non-null  object 
 6   destination_city  300153 non-null  object 
 7   class             300153 non-null  object 
 8   duration          300153 non-null  float64
 9   days_left         300153 non-null  int64  
 10  price             300153 non-null  int64  
dtypes: float64(1), int64(2), object(8)
memory usage: 25.2+ MB
```

7. Summary Statistics

The df.describe() function provides mean, median, standard deviation, and quartiles of numerical columns. This helps spot outliers and data ranges.

```
df.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 |

8. First and Last Rows

We preview the first (head()) and last (tail()) few rows of the dataset to understand its structure and sample values.

```
df.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 |
| **4** | Vistara | UK-963 | Delhi | Morning | zero | Morning | Mumbai | Economy | 2.33 | 1 | 5955 |

```
df.tail()
```

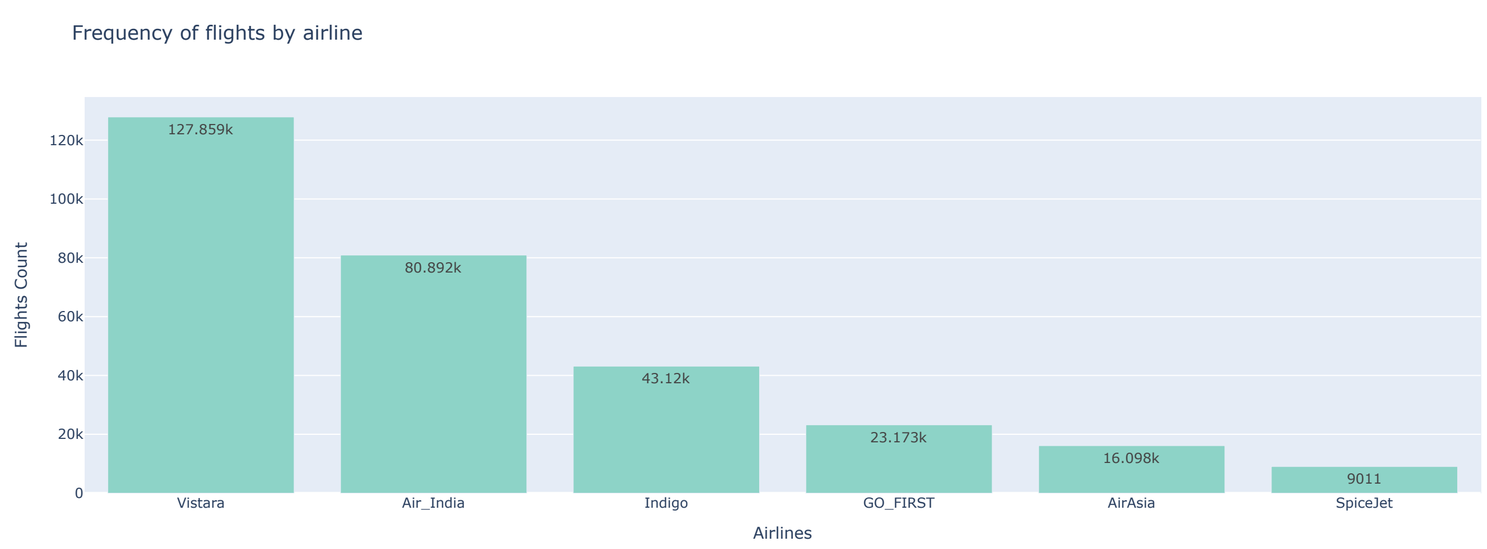
| | airline | flight | source_city | departure_time | stops | arrival_time | destination_city | class | duration | days_left | price |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **300148** | Vistara | UK-822 | Chennai | Morning | one | Evening | Hyderabad | Business | 10.08 | 49 | 69265 |
| **300149** | Vistara | UK-826 | Chennai | Afternoon | one | Night | Hyderabad | Business | 10.42 | 49 | 77105 |
| **300150** | Vistara | UK-832 | Chennai | Early_Morning | one | Night | Hyderabad | Business | 13.83 | 49 | 79099 |
| **300151** | Vistara | UK-828 | Chennai | Early_Morning | one | Evening | Hyderabad | Business | 10.00 | 49 | 81585 |
| **300152** | Vistara | UK-822 | Chennai | Morning | one | Evening | Hyderabad | Business | 10.08 | 49 | 81585 |

VISUALIZATIONS:-

9. Frequency of Flights by Airline

We visualize the number of flights per airline. This highlights which airlines dominate in terms of flight count.

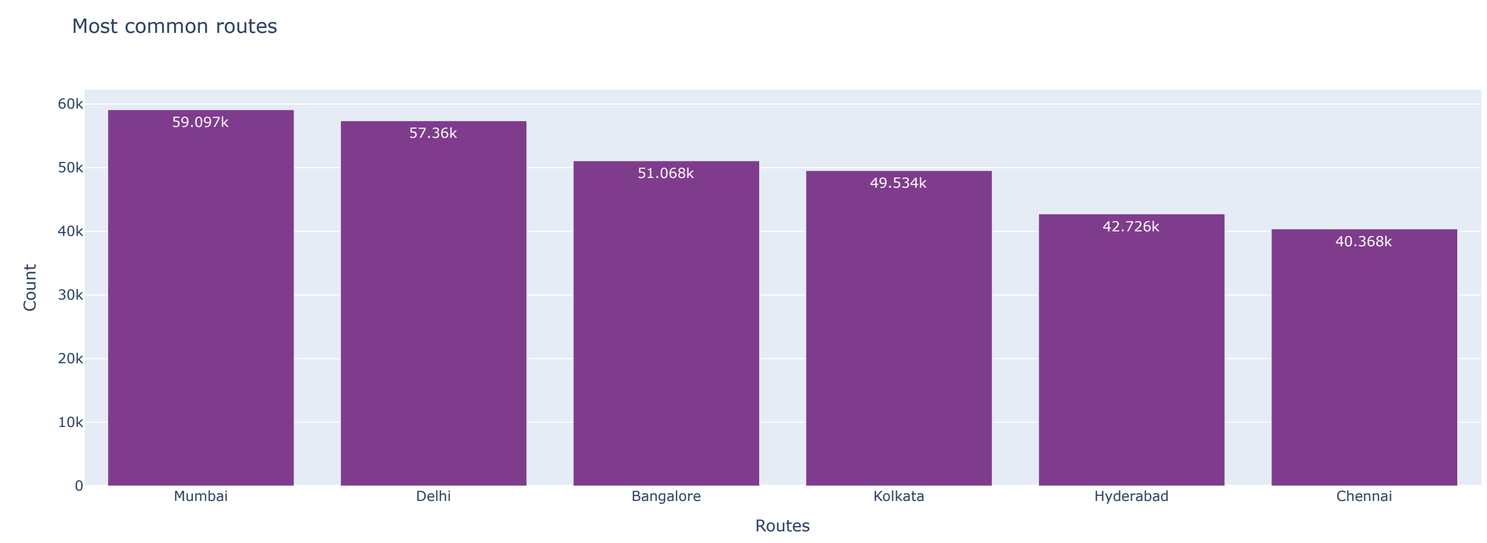
```
#Frequency of flights by airline.
airlines=df["airline"].value_counts().sort_values(ascending=False)
airlines=px.bar(x=airlines.index,y=airlines.values,text_auto=True,color_discrete_sequence=px.colors.qualitative.Set3 ,
                title="Frequency of flights by airline",labels={"x":"Airlines","y":"Flights Count"})
airlines.show()
```




10. Most Common Routes

We check the most popular destination cities. This reveals which routes are in highest demand.

```
#Most common routes
routes=df["destination_city"].value_counts().sort_values(ascending=False)
routes=px.bar(x=routes.index,y=routes.values,text_auto=True,color_discrete_sequence=px.colors.qualitative.Bold,title="Most common routes",labels={"x":"Routes","y":"Count"})
routes.show()
```




11. Class-Wise Airline Price Distribution

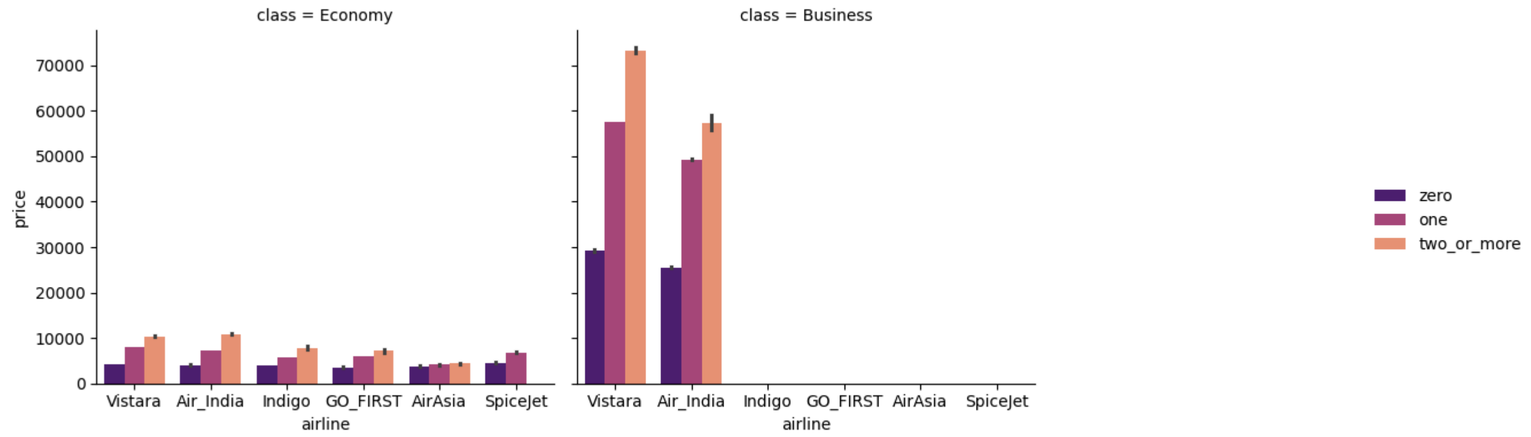
We use bar plots (faceted by class) to compare how ticket prices vary across airlines and stops.

Non-stop flights are generally more expensive.

Business class tickets cost more than economy.

```
# class wise airline & price
plt.figure(figsize=(12,6))
departure_times=sns.FacetGrid(df,col="class",col_wrap=3,height=4)
departure_times.map(sns.barplot,"airline","price",hue="stops", order=df["airline"].value_counts().index,data=df,palette='magma')
departure_times.add_legend()
plt.tight_layout()
plt.show()
```

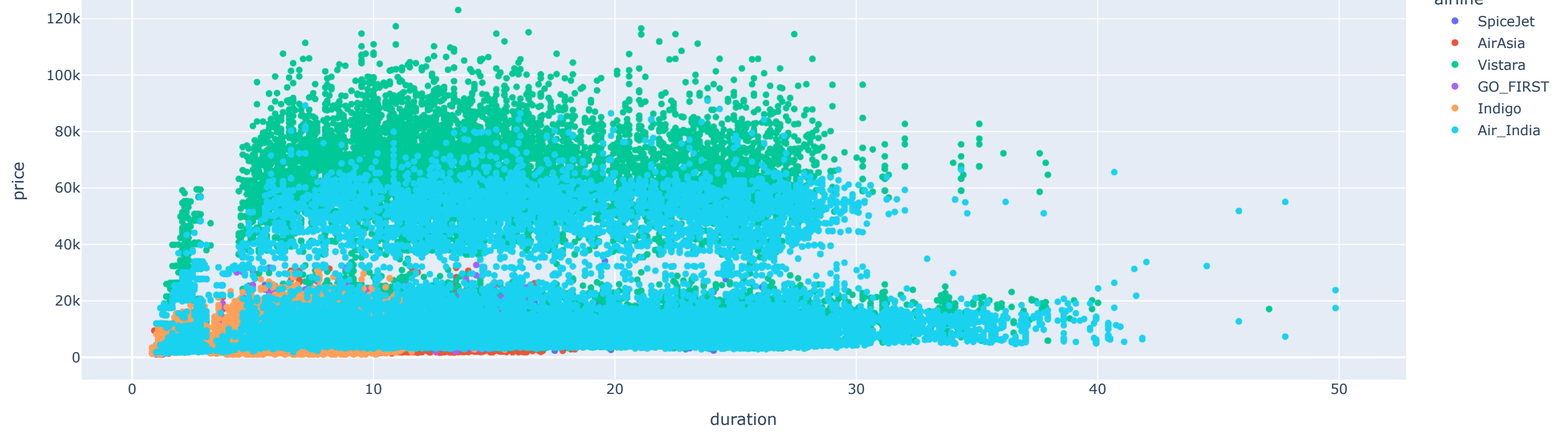
```
<Figure size 1200x600 with 0 Axes>
```




12. Duration vs. Price

A scatter plot is created to explore the relationship between flight duration and ticket price. Longer flights often cost more, but some airlines show exceptions

```
# duration wise price by airlines
duration=px.scatter(df,x="duration",y="price",color="airline",title="Duration wise price by airlines")
duration.show()
```




13. Flight Price vs. Days Left

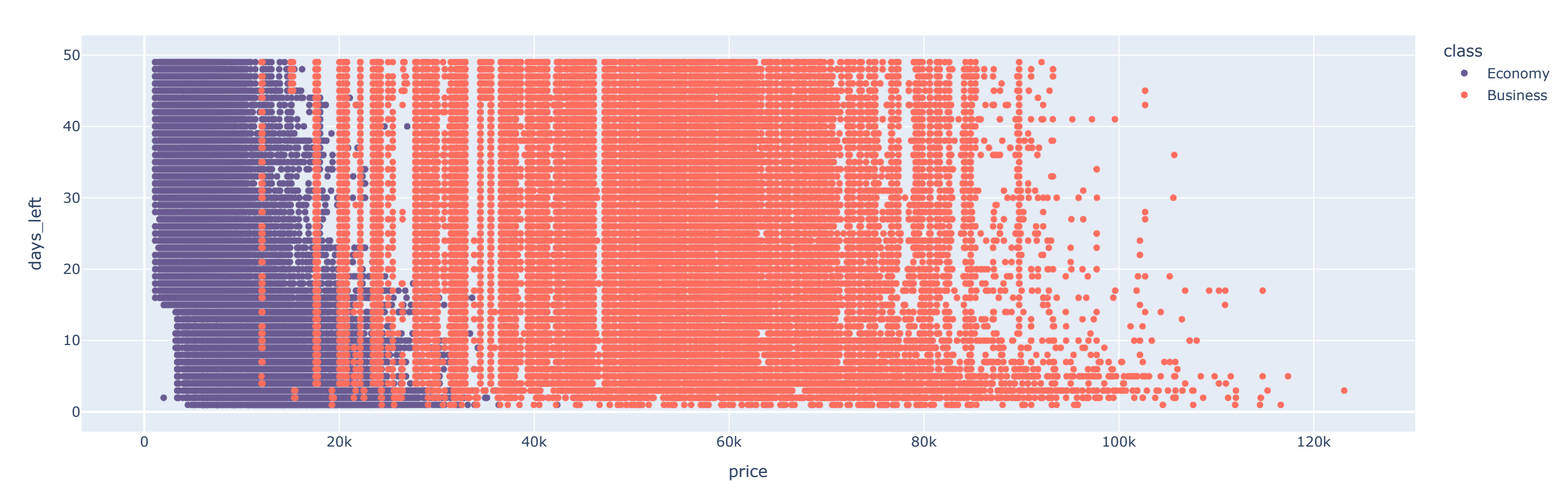
We examine how prices change with the number of days left before departure.

Prices generally increase closer to departure (airline pricing strategy).

Economy vs. Business classes follow different trends.

```
# Flight price vs. days_left by Class
import plotly.express as px

fig = px.scatter(df, x="price", y="days_left", color="class",color_discrete_sequence=["#6B5B95", "#FF6F61"],
                 title="Flight price vs. days_left by Class")
fig.show()
```



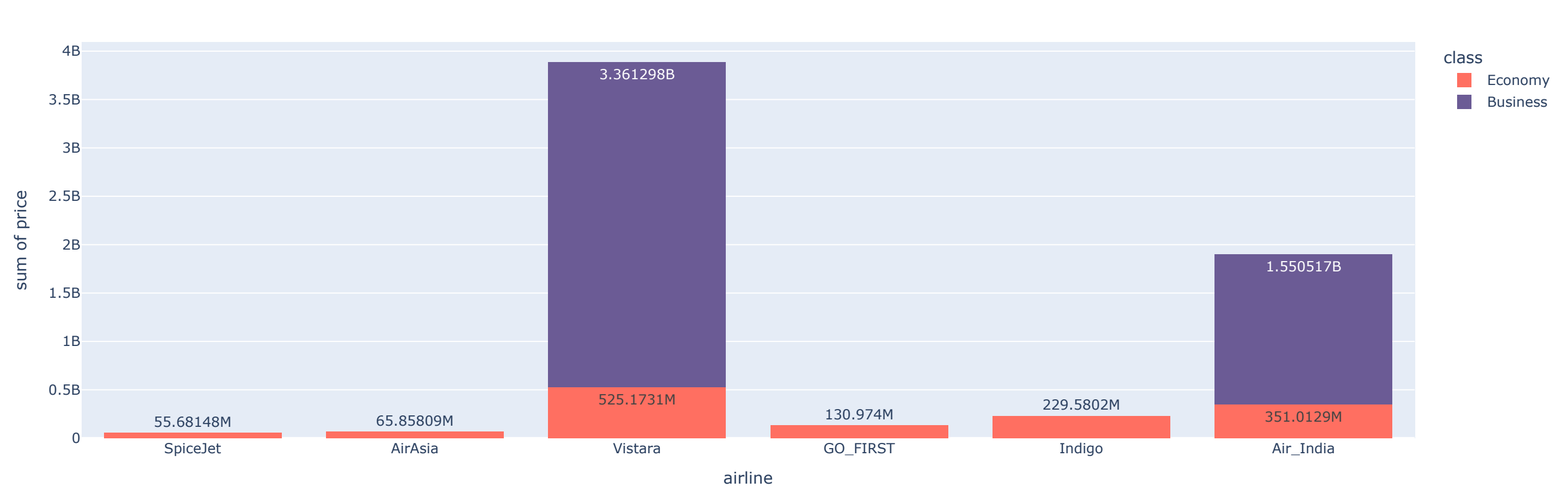

14. Price Distribution Among Airlines

A histogram shows how ticket prices are distributed for each airline across classes.

Some airlines consistently charge higher prices (premium carriers).

Low-cost carriers cluster in lower price ranges.

```
# distribution of price among airlines
price=px.histogram(df,x="airline",y="price",color="class",text_auto=True,color_discrete_sequence=["#FF6F61", "#6B5B95", "#045D75", "#F4B400", "#0F9D58"],
                   title="Distribution of price among airlines by class",labels={"x":"Airlines","y":"Sum of price"})
price.show()
```



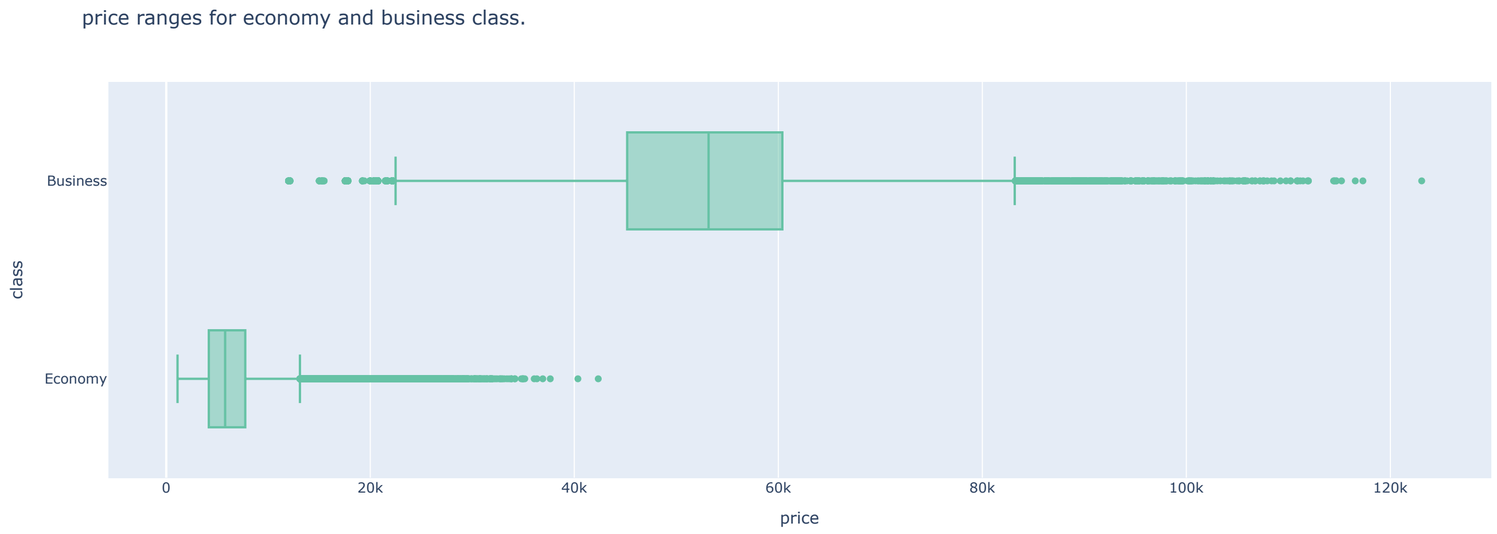

15. Price Ranges by Class

A boxplot compares Economy vs. Business ticket prices.

Business class has higher median values and wider spread.

Economy class is more tightly distributed.

```
# price ranges for economy and business class
fig=px.box(df,x="price",y="class",    color_discrete_sequence=px.colors.qualitative.Set2
,title="price ranges for economy and business class.")
fig.show()
```



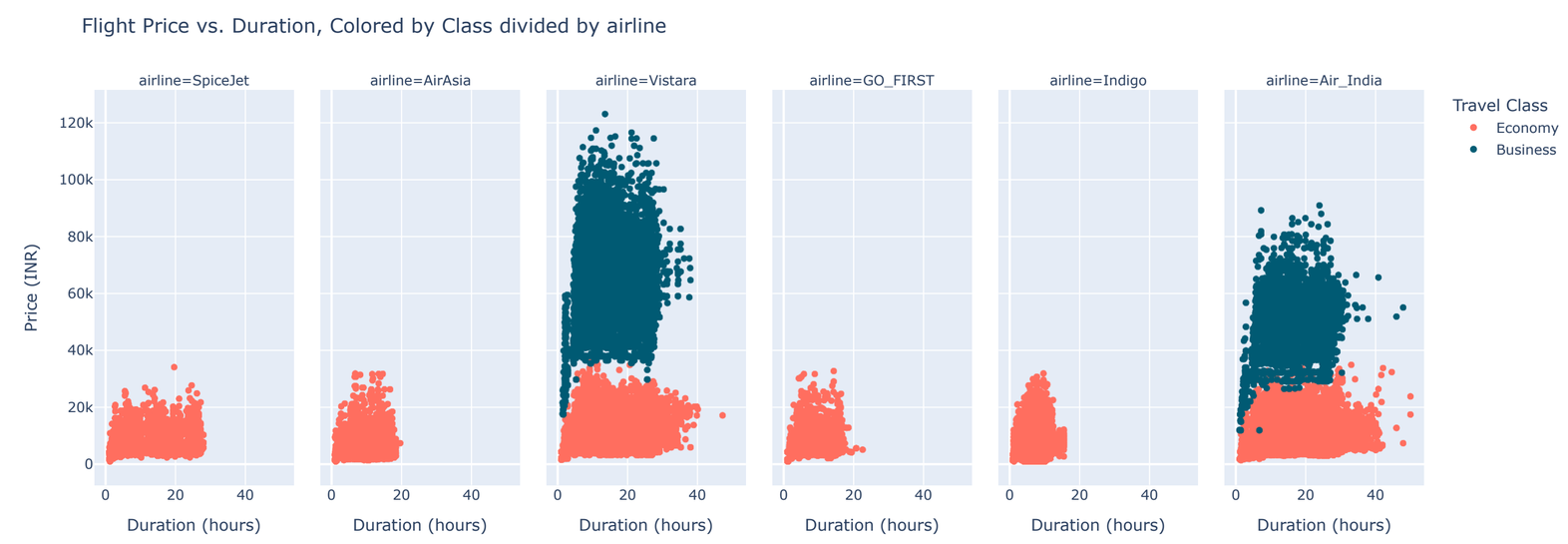

16. Price vs. Duration by Airline and Class

Scatter plots (separated by airline) show how price relates to duration, colored by travel class.

Helps compare patterns across airlines.

Reveals if some airlines overcharge for shorter flights.

```
# Flight Price vs. Duration, Colored by Class divided by airline
fig1 = px.scatter(df,x='duration',y='price',color='class',
                  facet_col="airline",color_discrete_sequence=["#FF6F61", "#045D75", "#F4B400"] ,
                  title='Flight Price vs. Duration, Colored by Class divided by airline',
                  labels={ "duration": "Duration (hours)", "price": "Price (INR)", "class": "Travel Class"})
fig1.show()
```




17. Pairplot of Features by Airline

> Add blockquote

The pairplot is a great way to visualize the relationship between multiple numerical features in the dataset.

Each scatter plot shows the relationship between two features (e.g., duration vs. price, days_left vs. price, etc.).

The diagonal plots display the distribution (histogram) of each individual feature.

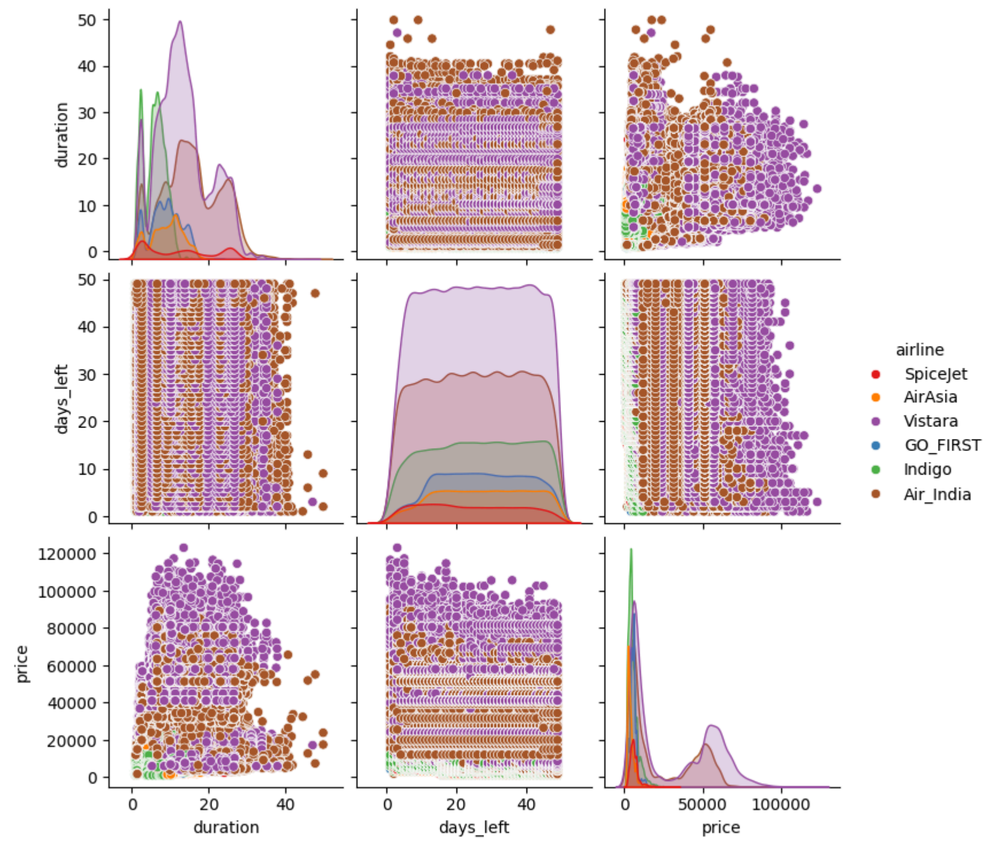
The hue="airline" parameter colors the points by airline, which helps compare how different airlines behave in terms of flight duration, pricing, and advance booking.

This makes it easier to spot clusters, overlaps, and trends — for example, some airlines may consistently have shorter durations or higher prices compared to others.

👉 Overall, the pairplot provides a comprehensive overview of feature interactions and highlights differences between airlines.

```
plt.figure(figsize=(12,10))
airline_palette = {"SpiceJet": "#E41A1C", "AirAsia": "#FF7F00","Vistara": "#984EA3","GO_FIRST": "#377EB8","Indigo": "#4DAF4A","Air_India": "#A65628"}
sns.pairplot(df, hue="airline", palette=airline_palette)
```

```
<seaborn.axisgrid.PairGrid at 0x7d7260ca68a0>
<Figure size 1200x1000 with 0 Axes>
```

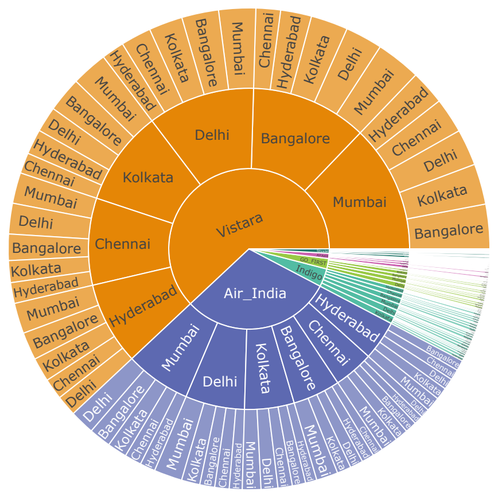



18. Source–Destination Airline Sunburst Chart

A sunburst chart displays a hierarchical view of: Airline → Source City → Destination City, with flight prices.

Helps visualize route networks of airlines.

```
# Create the sunburst chart
sunburst = px.sunburst(df, path=["airline", "source_city", "destination_city"],values="price",color_discrete_sequence=px.colors.qualitative.Vivid,title="Sunburst Chart: Airline Route Revenue")
sunburst.update_layout(width=600,height=600)
sunburst.show()
```



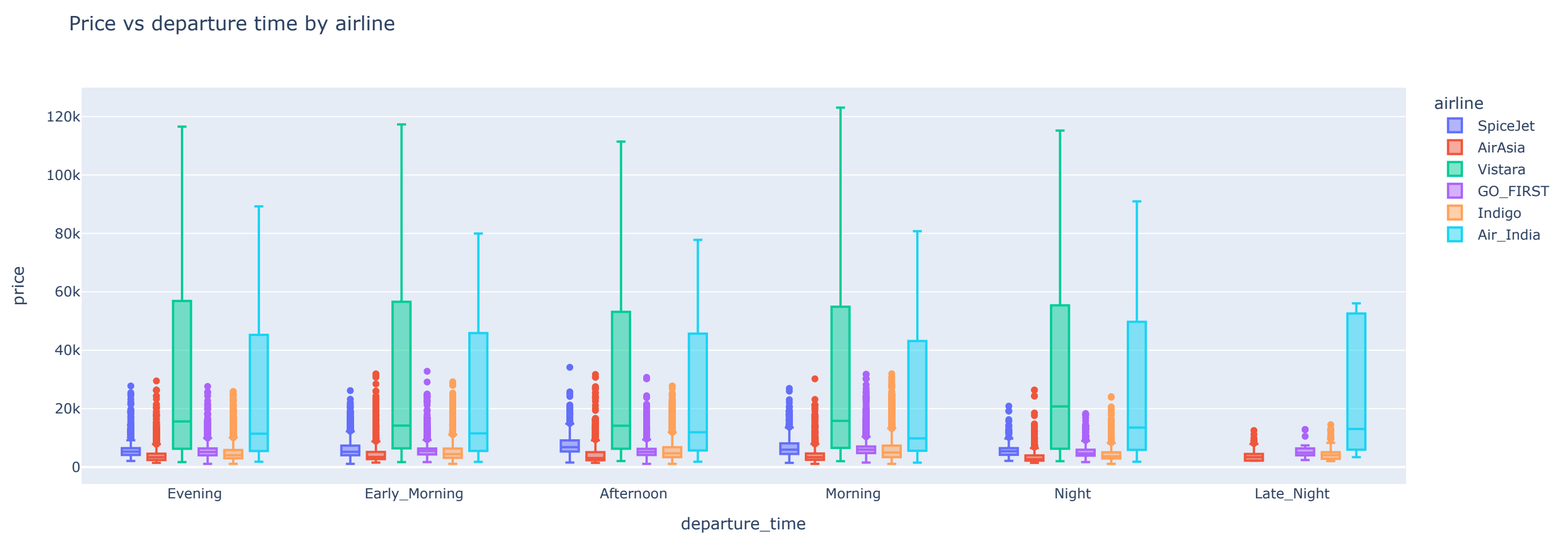

19.Price vs. Departure Time

Boxplots compare prices across different departure times for each airline.

Morning/evening slots may be costlier depending on airline.

Some carriers show cheaper night flights.

```
# price vs departure time
departure=px.box(df,x="departure_time",color="airline",y="price",title="Price vs departure time by airline")
departure.show()
```



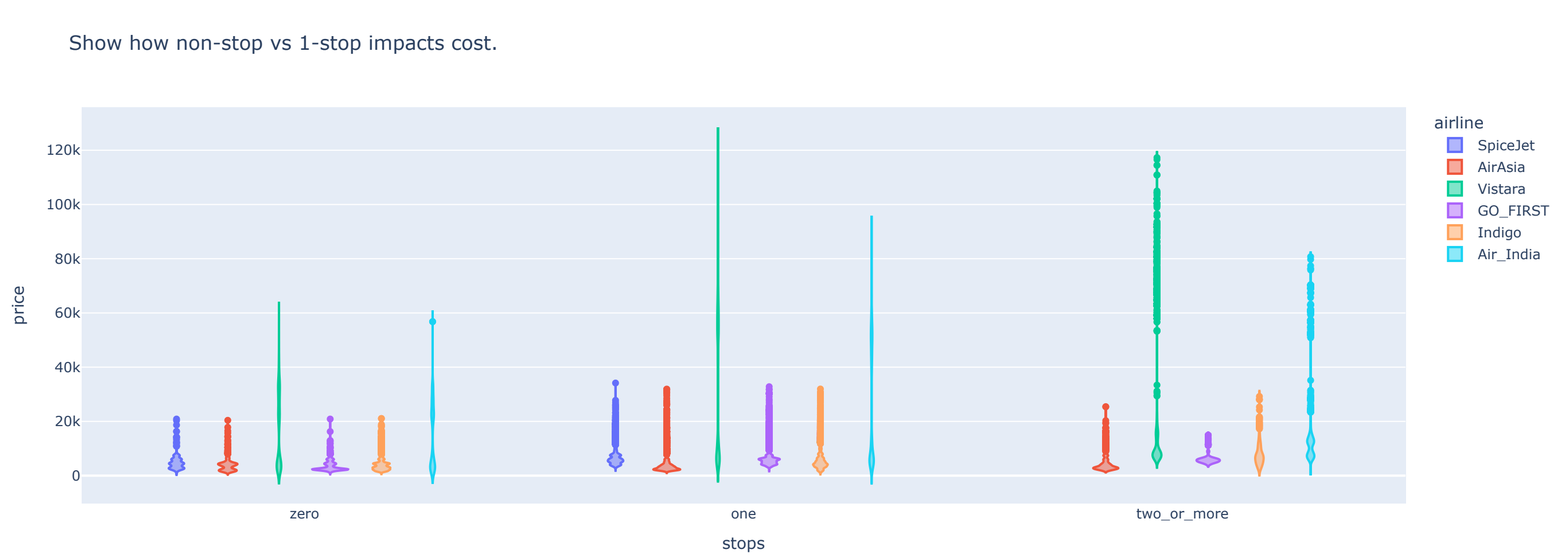

20. Impact of Stops on Flight Prices

Non-stop flights are the cheapest on average, making them the preferred option for budget-conscious travelers.

Adding stops significantly increases ticket prices, especially for full-service carriers like Air India and Vistara.

This pattern suggests that travelers willing to pay more often choose stopover flights with premium airlines, likely due to longer routes, international travel, or additional services.

```
# Show how non-stop vs 1-stop impacts cost.
stop_vs_price=px.violin(df,x="stops",y="price",color="airline",title="Show how non-stop vs 1-stop impacts cost.")
stop_vs_price.show()
```



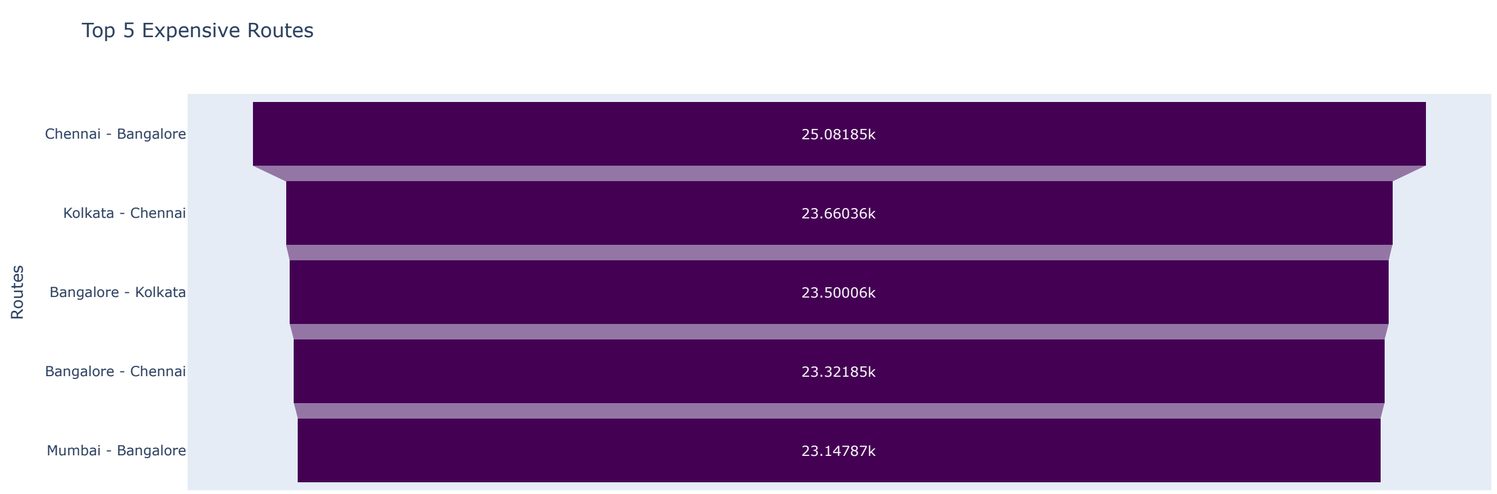

21.Top 5 Expensive Routes

Chennai ⟶ Bangalore is the most expensive route, with an average price of around ₹25,081.

Other costly routes include Kolkata ⟶ Chennai, Bangalore ⟶ Kolkata, Bangalore ⟶ Chennai, and Mumbai ⟶ Bangalore, with average prices ranging between ₹23,100 – ₹23,600.

👉 This insight shows that short-distance but high-demand routes (like Chennai–Bangalore) can have higher ticket prices due to demand–supply imbalance, limited availability, or premium flight schedules.

```
# Top 5 Expensive Routes
exp_routes=df.groupby(["source_city","destination_city"])["price"].mean().sort_values(ascending=False)[:5]
exp_routes_1=px.funnel(x=exp_routes.values,y=[f"{source} - {destination}" for source, destination in exp_routes.index],
                       color_discrete_sequence=px.colors.sequential.Viridis,
                       title="Top 5 Expensive Routes",labels={"x":"Average Price","y":"Routes"})
exp_routes_1.show()
```




22.Airline Market Share by Class

Vistara dominates the Business Class market, contributing the largest share, and also has a noticeable share in Economy Class.

Air India follows a similar pattern with a strong presence in Business Class, reflecting its focus on premium travelers.

IndiGo, GO FIRST, AirAsia, and SpiceJet are concentrated mainly in the Economy Class, consistent with their budget airline positioning.

The color scale indicates that Business Class tickets (yellow zones) are priced significantly higher compared to Economy Class (purple zones), which aligns with expectations.

👉 This analysis highlights how full-service carriers (Vistara, Air India) dominate the premium Business segment, while low-cost carriers (IndiGo, GO FIRST, AirAsia, SpiceJet) compete heavily in the Economy segment.

```
# Airline Market Share by Class
fig = px.treemap(df,
                 path=["airline", "class"],
                 values="price",
                 color="price",
                 color_continuous_scale=px.colors.sequential.Viridis,
                 title="Airline Market Share by Class",
                 )
fig.show()
```


23.Average Flight Price by Route

Bangalore–Chennai and Chennai–Mumbai routes show higher average prices, which could be due to heavy business travel demand and limited direct low-cost options.

Delhi–Hyderabad and Kolkata–Hyderabad routes are relatively cheaper, possibly due to strong competition among low-cost carriers.

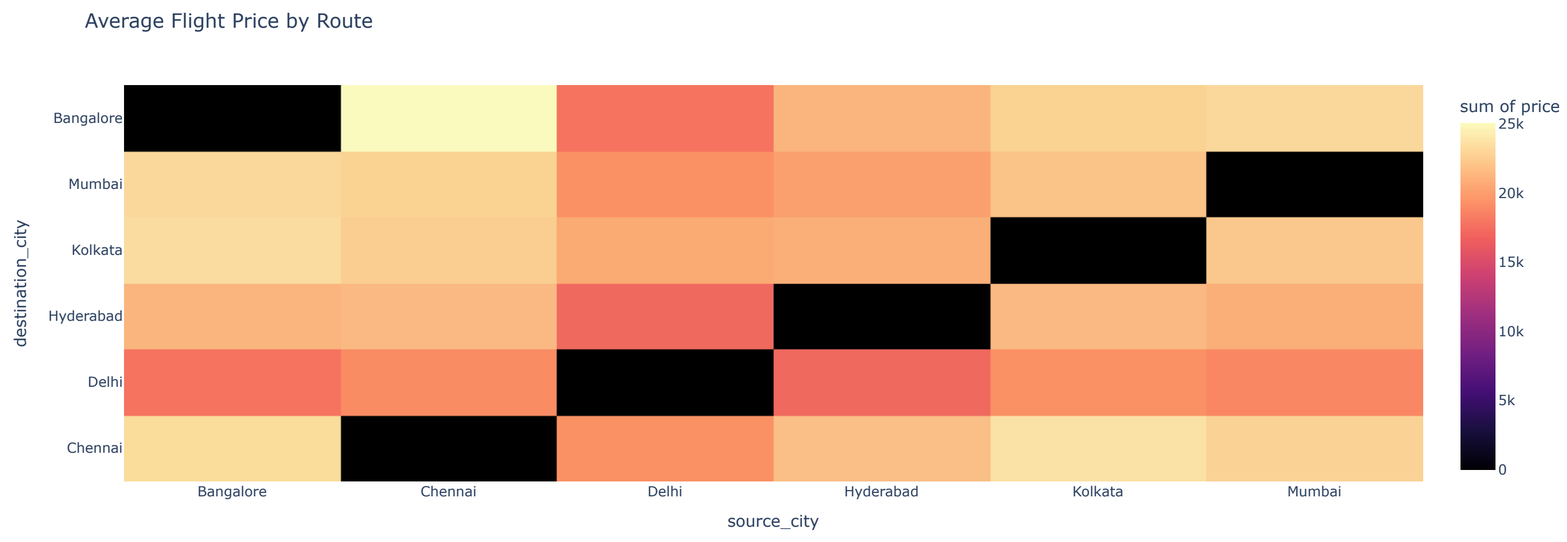
Major metro connections like Delhi–Mumbai and Bangalore–Mumbai fall in the mid-range pricing zone, reflecting a balance between demand and availability of multiple carriers.

Some city-pairs (dark blue squares) don't have direct flights, which explains missing or near-zero values.

👉 This heatmap helps identify premium routes (high average fares) and competitive routes (low average fares), which is crucial for both travelers (to find cheaper options) and airlines (to optimize pricing strategies).

```
# Average Flight Price by Route
route_data = df.groupby(["source_city", "destination_city"])["price"].mean().reset_index()

fig = px.density_heatmap(route_data,
                         x="source_city",
                         y="destination_city",
                         z="price",
                         color_continuous_scale=px.colors.sequential.Magma,
                         title="Average Flight Price by Route")
fig.show()
```




24.Correlation between Numeric Features

1. List item
2. List item

The heatmap shows how the numeric variables (duration, days_left, and price) are correlated with each other.

Key findings:

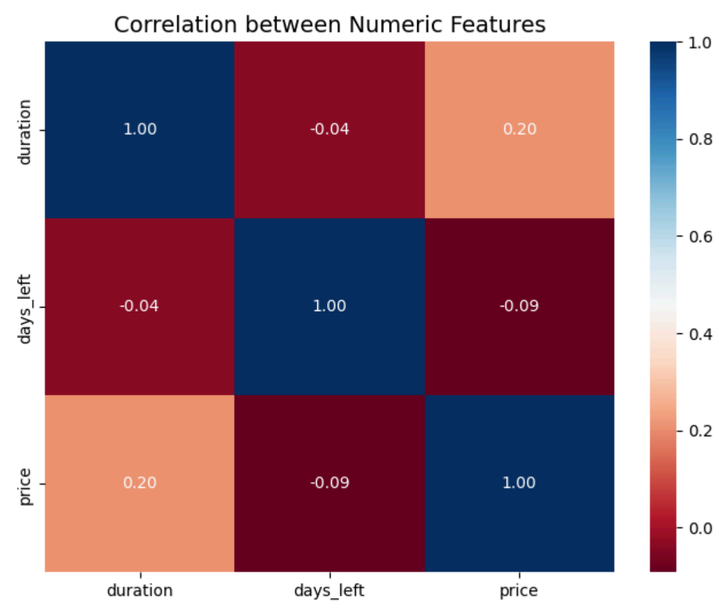
Duration vs Price (0.20) ⟶ There is a weak positive correlation, meaning longer flights tend to have slightly higher prices, but the relationship is not very strong.

Days Left vs Price (-0.09) ⟶ A very weak negative correlation, indicating that booking earlier does not always guarantee a lower price. Other factors (airline, route, demand) influence pricing more strongly than the days left.

Duration vs Days Left (-0.04) ⟶ Almost no correlation, as the length of a flight has nothing to do with how early a ticket is booked.

👉 Overall, this heatmap reveals that flight prices are not heavily influenced by basic numeric features like duration or days left. Instead, categorical factors (like airline, route, class, stops) play a much more important role in pricing.

```
plt.figure(figsize=(8,6))
sns.heatmap(df.corr(numeric_only=True), annot=True, cmap="RdBu", fmt=".2f")
plt.title("Correlation between Numeric Features", fontsize=14)
plt.show()
```




25.Parallel Coordinates Plot

The parallel coordinates plot is used to visualize multi-dimensional data. Each line in the chart represents a single flight, and the vertical axes represent different features.

By coloring the lines according to price, we can observe how different factors (like airline, source city, destination, stops, duration, etc.) collectively influence the ticket price.

High-priced flights usually appear as distinct colored lines, allowing us to trace which feature combinations (e.g., certain airlines, long duration, premium classes) are driving the cost up.

On the other hand, lower-priced flights cluster around economy classes, shorter durations, and non-stop routes.

👉 This visualization is especially helpful to see how multiple categorical and numerical variables interact together and contribute to flight pricing, rather than looking at them individually.

```
#parallel coordinate plot
plot=px.parallel_coordinates(df,color="price",title="parallel coordinate plot")
plot.show()
```