```
** Title:- Airlines Flight data analysis

#First we need to import the necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# import the CSV Dataset
data = pd.read_csv("C:\\Users\\david\\Downloads\\
airlines_flights_data.csv")
data

         index   airline   flight source_city departure_time stops  \
0            0  SpiceJet  SG-8709       Delhi        Evening  zero
1            1  SpiceJet  SG-8157       Delhi  Early_Morning  zero
2            2   AirAsia   I5-764       Delhi  Early_Morning  zero
3            3   Vistara   UK-995       Delhi        Morning  zero
4            4   Vistara   UK-963       Delhi        Morning  zero
...        ...       ...      ...         ...            ...   ...
300148  300148   Vistara   UK-822     Chennai        Morning   one
300149  300149   Vistara   UK-826     Chennai      Afternoon   one
300150  300150   Vistara   UK-832     Chennai  Early_Morning   one
300151  300151   Vistara   UK-828     Chennai  Early_Morning   one
300152  300152   Vistara   UK-822     Chennai        Morning   one

         arrival_time destination_city     class  duration  days_left
price
0               Night           Mumbai   Economy      2.17          1
5953
1             Morning           Mumbai   Economy      2.33          1
5953
2       Early_Morning           Mumbai   Economy      2.17          1
5956
3           Afternoon           Mumbai   Economy      2.25          1
5955
4             Morning           Mumbai   Economy      2.33          1
5955
...               ...              ...       ...       ...        ...
...
300148        Evening        Hyderabad  Business     10.08         49
69265
300149          Night        Hyderabad  Business     10.42         49
77105
300150          Night        Hyderabad  Business     13.83         49
79099
300151        Evening        Hyderabad  Business     10.00         49
81585
300152        Evening        Hyderabad  Business     10.08         49
81585
```

```
[300153 rows x 12 columns]
```

```
pip install mysql-connector-python
```

```
Requirement already satisfied: mysql-connector-python in c:\users\
david\anaconda3\lib\site-packages (9.4.0)
Note: you may need to restart the kernel to use updated packages.
```

```
import mysql.connector

try:
    connection = mysql.connector.connect(
        host='localhost',       # Your MySQL host, e.g., localhost
        user='root',    # Your MySQL username
        password='Jyothi@1110', # Your MySQL password
           # Your database name)
    )
    if connection.is_connected():
        print("Successfully connected to MySQL database")
except mysql.connector.Error as err:
    print(f"Error: {err}")
```

```
Successfully connected to MySQL database
```

```
data.head()
```

```
   index   airline   flight source_city departure_time stops
arrival_time  \
0      0  SpiceJet  SG-8709       Delhi        Evening  zero
Night
1      1  SpiceJet  SG-8157       Delhi  Early_Morning  zero
Morning
2      2   AirAsia   I5-764       Delhi  Early_Morning  zero
Early_Morning
3      3   Vistara   UK-995       Delhi        Morning  zero
Afternoon
4      4   Vistara   UK-963       Delhi        Morning  zero
Morning

  destination_city    class  duration  days_left  price
0           Mumbai  Economy      2.17          1   5953
1           Mumbai  Economy      2.33          1   5953
2           Mumbai  Economy      2.17          1   5956
3           Mumbai  Economy      2.25          1   5955
4           Mumbai  Economy      2.33          1   5955
```

```
data.tail()
```

```
         index  airline  flight source_city departure_time stops
arrival_time  \
300148  300148  Vistara  UK-822     Chennai        Morning   one
Evening
300149  300149  Vistara  UK-826     Chennai      Afternoon   one
Night
300150  300150  Vistara  UK-832     Chennai  Early_Morning   one
Night
300151  300151  Vistara  UK-828     Chennai  Early_Morning   one
Evening
300152  300152  Vistara  UK-822     Chennai        Morning   one
Evening

       destination_city     class  duration  days_left  price
300148        Hyderabad  Business     10.08         49  69265
300149        Hyderabad  Business     10.42         49  77105
300150        Hyderabad  Business     13.83         49  79099
300151        Hyderabad  Business     10.00         49  81585
300152        Hyderabad  Business     10.08         49  81585
```

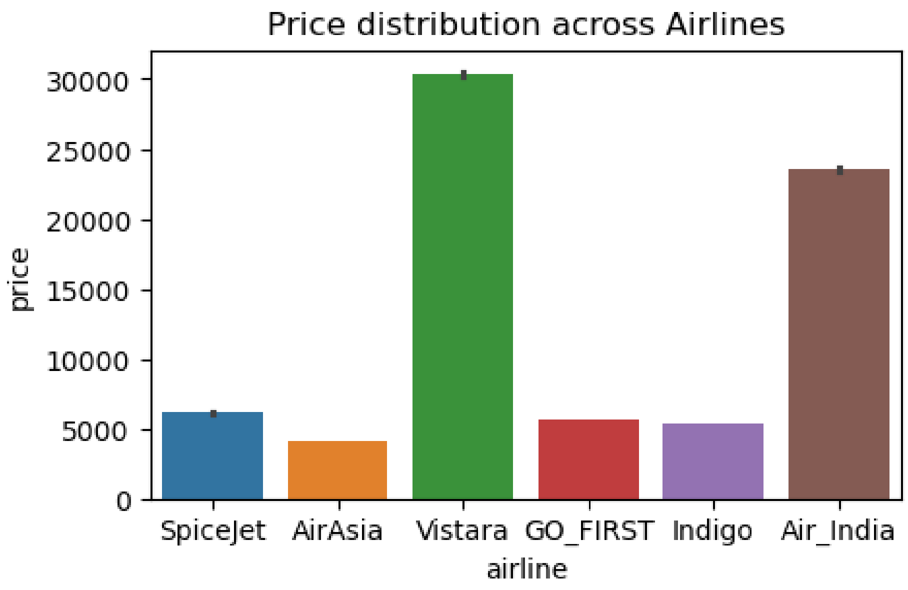
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 300153 entries, 0 to 300152
Data columns (total 12 columns):
 #   Column            Non-Null Count   Dtype  
---  ------            --------------   -----  
 0   index             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
```

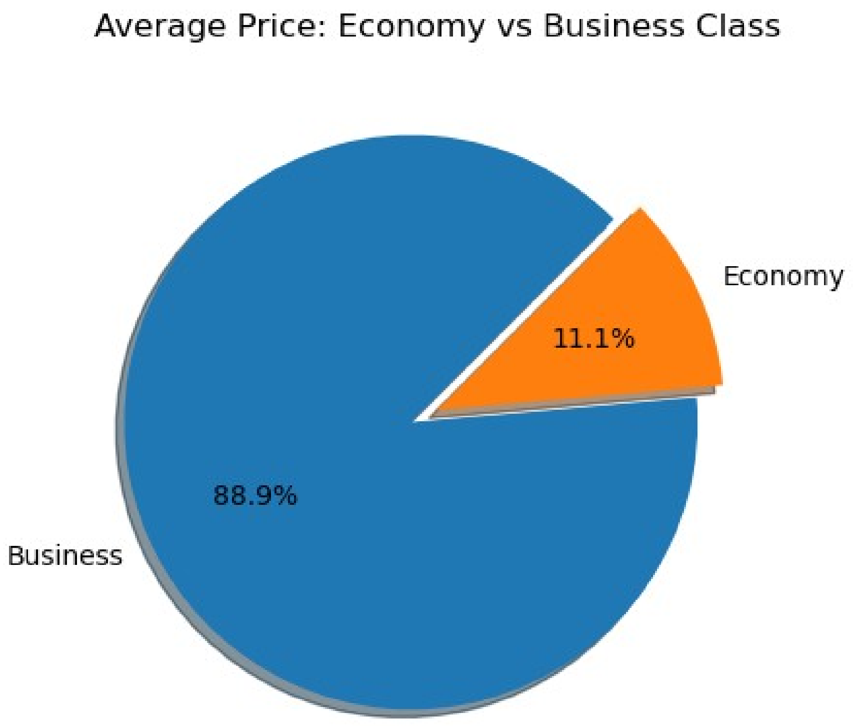
```
data.describe()
```

```
               index       duration      days_left          price
count  300153.000000  300153.000000  300153.000000  300153.000000
mean   150076.000000      12.221021      26.004751   20889.660523
std     86646.852011       7.191997      13.561004   22697.767366
min         0.000000       0.830000       1.000000    1105.000000
25%     75038.000000       6.830000      15.000000    4783.000000
50%    150076.000000      11.250000      26.000000    7425.000000
```

```
75%    225114.000000      16.170000      38.000000   42521.000000
max    300152.000000      49.830000      49.000000  123071.000000

data.isnull().sum()

index               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

data[data.duplicated()]

Empty DataFrame
Columns: [index, airline, flight, source_city, departure_time, stops,
arrival_time, destination_city, class, duration, days_left, price]
Index: []

# 1)price distribution across Airlines
plt.figure(figsize=(5, 3))
sns.barplot(data=data,x="airline",y="price",hue="airline")
plt.title("Price distribution across Airlines")
plt.show()
```

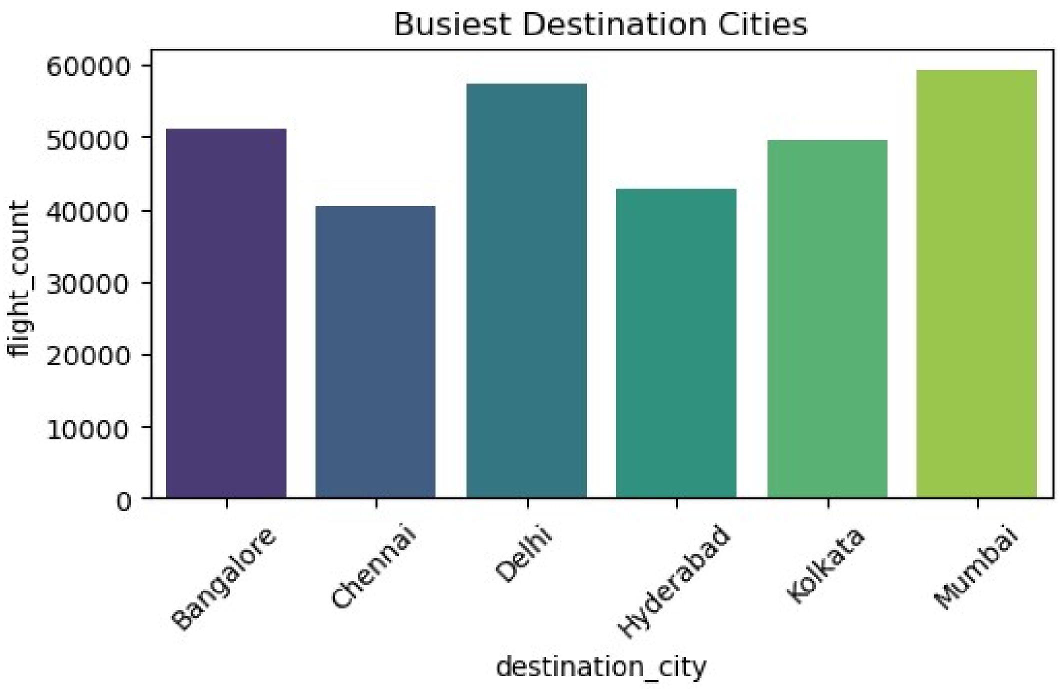


```python
# 2)Average Price: Economy vs. Business Class
Average_price = data.groupby('class')['price'].mean().reset_index()
values = Average_price['price']
labels = Average_price['class']
explode = [0.1, 0]

plt.pie(values,labels=labels,autopct='%1.1f%
%',startangle=45,shadow=True,explode=explode)
plt.title("Average Price: Economy vs Business Class")
plt.show()
```

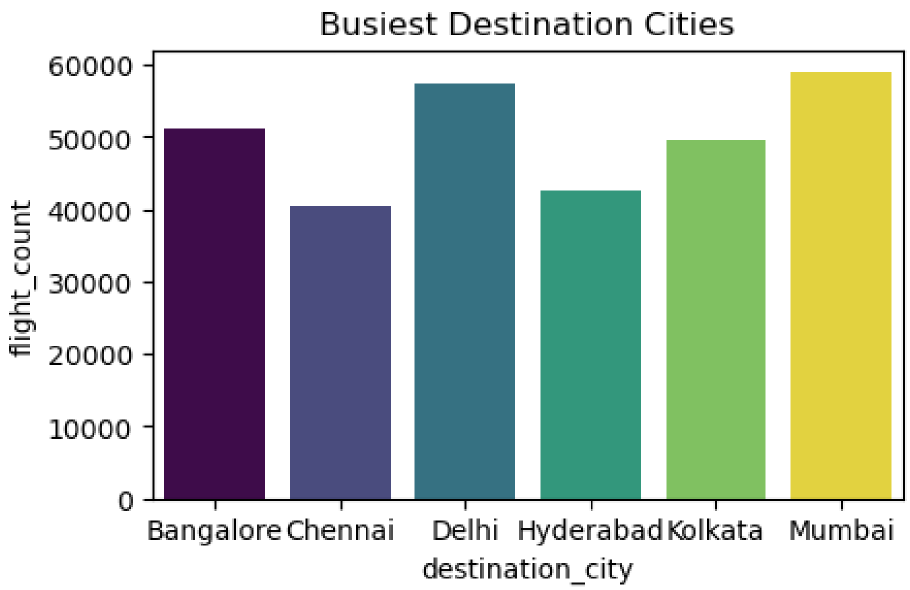


```python
# 3) Busiest Source & Destination Cities
#A)Top Source Cities

source_counts = data.groupby("destination_city")
["flight"].count().reset_index()
source_counts.rename(columns={"flight": "flight_count"}, inplace=True)

plt.figure(figsize=(6,3))
sns.barplot(x="destination_city", y="flight_count",
data=source_counts, palette="viridis")
plt.xticks(rotation=45)
plt.title("Busiest Destination Cities")
plt.show()
```

```
C:\Users\david\AppData\Local\Temp\ipykernel_7160\2846909472.py:8:
FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `x` variable to `hue` and set
`legend=False` for the same effect.

  sns.barplot(x="destination_city", y="flight_count",
data=source_counts, palette="viridis")
```

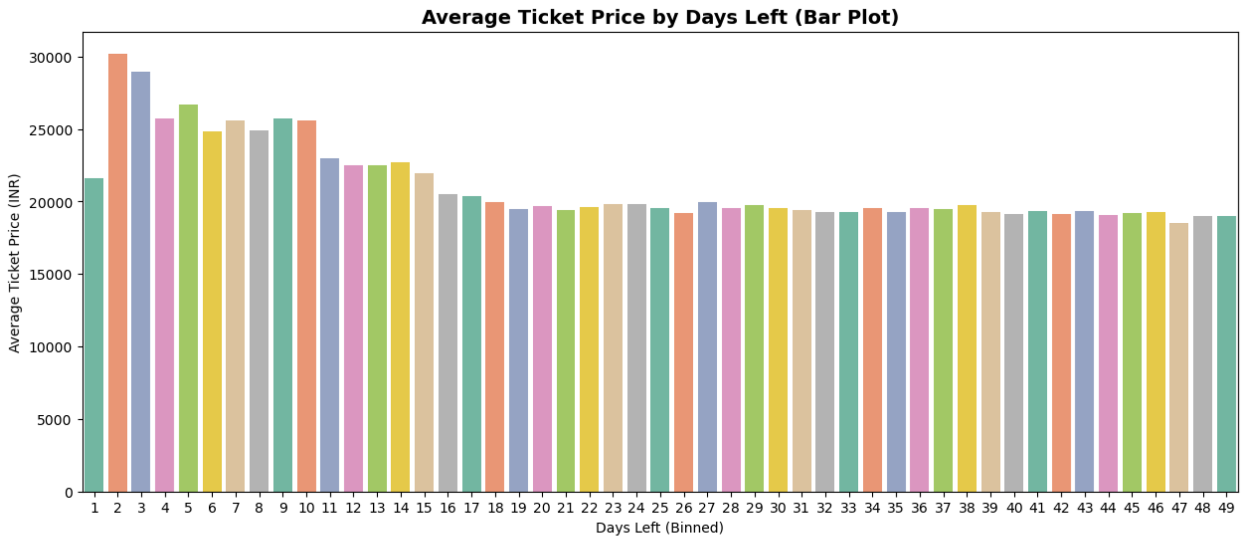


```
#3) Busiest Source & Destination Cities
#B)Top Destination Cities

dest_counts = data.groupby("destination_city")
["flight"].count().reset_index()
dest_counts.rename(columns={"flight": "flight_count"}, inplace=True)
print(dest_counts.columns)

Index(['destination_city', 'flight_count'], dtype='object')

plt.figure(figsize=(5,3))
sns.barplot(x="destination_city",y="flight_count",hue=dest_counts.inde
x, legend=False, data=dest_counts,palette="viridis")
plt.title("Busiest Destination Cities")
plt.show()
```

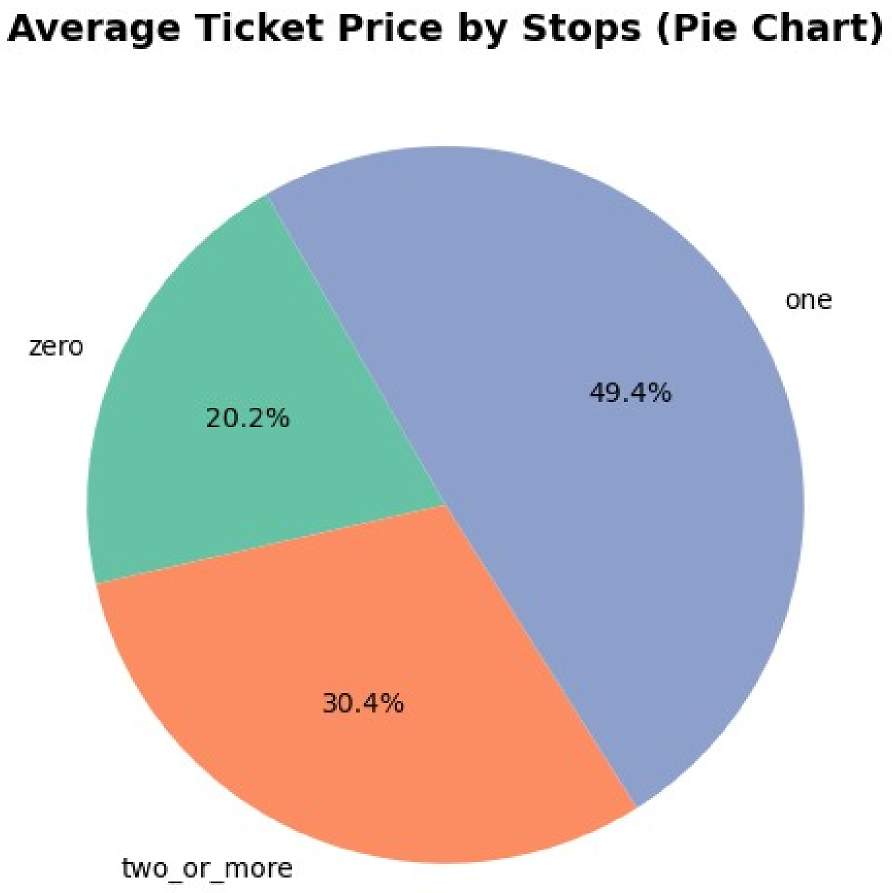


```
# 4)Days Left vs. Ticket Price Trend
avg_price_by_days = data.groupby("days_left")["price"].mean()

# --- Bar Plot ---
plt.figure(figsize=(15,6))
sns.barplot(x=avg_price_by_days.index,
y=avg_price_by_days.values,dodge=False, palette="Set2")
plt.title("Average Ticket Price by Days Left (Bar Plot)", fontsize=14,
fontweight="bold")
plt.xlabel("Days Left (Binned)")
plt.ylabel("Average Ticket Price (INR)")
plt.show()

print("Average Ticket Price by Days Left Bin:\n")
#print(avg_price_by_days)
```

```
C:\Users\david\AppData\Local\Temp\ipykernel_25960\2776151662.py:5:
FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `x` variable to `hue` and set
`legend=False` for the same effect.

  sns.barplot(x=avg_price_by_days.index,
y=avg_price_by_days.values,dodge=False, palette="Set2")
```



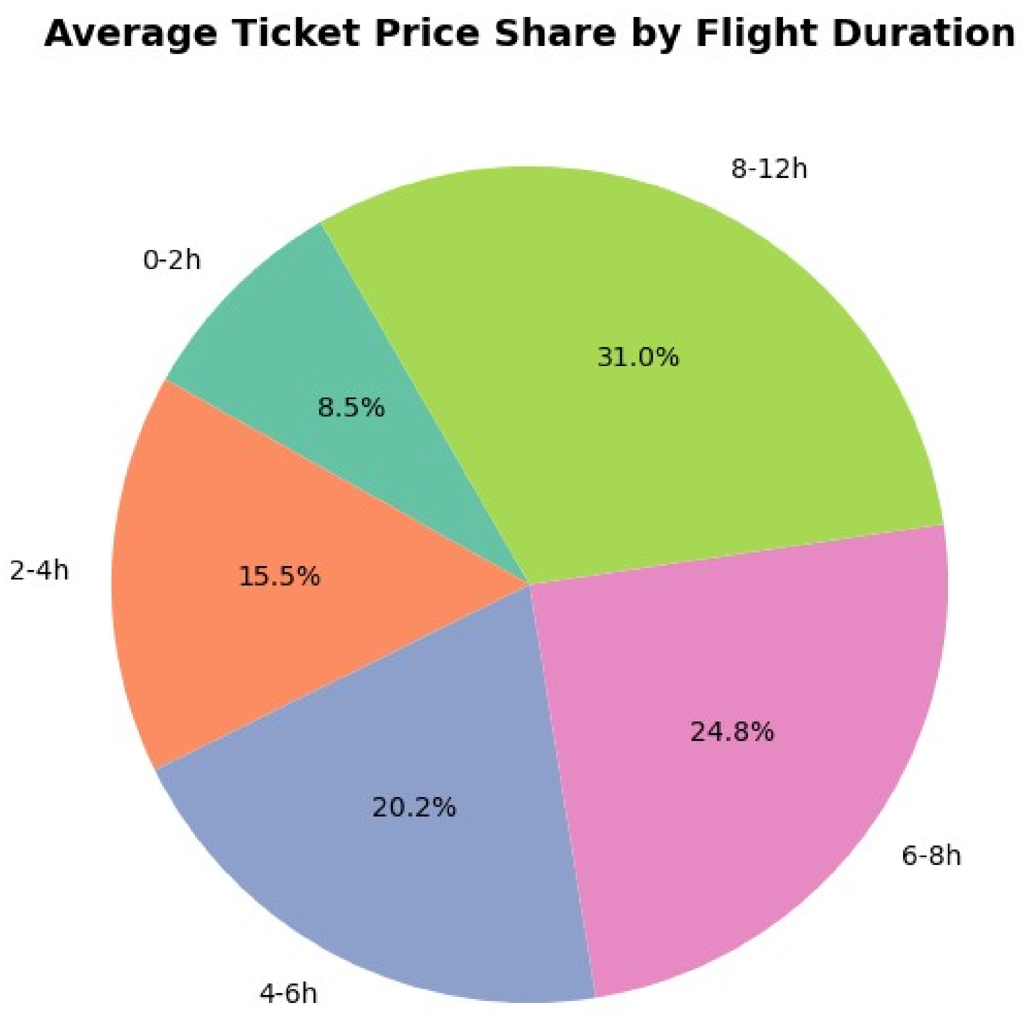
```
Average Ticket Price by Days Left Bin:


# 5)Stops vs. Price Comparison
plt.figure(figsize=(6,6))
plt.pie(
    avg_price_by_stops.values,
    labels=avg_price_by_stops.index,
    autopct="%1.1f%%",
    startangle=120,
    colors=sns.color_palette("Set2")
)
plt.title("Average Ticket Price by Stops (Pie Chart)", fontsize=14,
fontweight="bold")
plt.show()

print("Average Price by Stops:\n")
print(avg_price_by_stops)
```



```
Average Price by Stops:

stops
zero            9375.938535
two_or_more    14113.450775
one            22900.992482
Name: price, dtype: float64

# 6)Duration Impact on Ticket Pricing

# 1. Create bins for flight durations
bins = [0, 2, 4, 6, 8, 12]  # adjust according to your dataset
labels = ["0-2h", "2-4h", "4-6h", "6-8h", "8-12h"]

# Create the duration_bin column
data["duration_bin"] = pd.cut(data["duration"], bins=bins,
labels=labels)

avg_price_by_duration = data.groupby("duration_bin")
["price"].mean().dropna()
```

```
# --- Pie Plot ---
plt.figure(figsize=(7,7))
plt.pie(
    avg_price_by_duration.values,
    labels=avg_price_by_duration.index,
    autopct="%1.1f%%",
    startangle=120,
    colors=sns.color_palette("Set2")
)
plt.title("Average Ticket Price Share by Flight Duration",
fontsize=14, fontweight="bold")
plt.show()
```

```
C:\Users\david\AppData\Local\Temp\ipykernel_25816\1675222407.py:10:
FutureWarning: The default of observed=False is deprecated and will be
changed to True in a future version of pandas. Pass observed=False to
retain current behavior or observed=True to adopt the future default
and silence this warning.
  avg_price_by_duration = data.groupby("duration_bin")
["price"].mean().dropna()
```



```
# SQL Queries

!pip install pandasql

Requirement already satisfied: pandasql in c:\users\david\anaconda3\
lib\site-packages (0.7.3)
Requirement already satisfied: numpy in c:\users\david\anaconda3\lib\
site-packages (from pandasql) (2.1.3)
Requirement already satisfied: pandas in c:\users\david\anaconda3\lib\
site-packages (from pandasql) (2.2.3)
Requirement already satisfied: sqlalchemy in c:\users\david\anaconda3\
lib\site-packages (from pandasql) (2.0.39)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\
david\anaconda3\lib\site-packages (from pandas->pandasql)
(2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in c:\users\david\
anaconda3\lib\site-packages (from pandas->pandasql) (2024.1)
Requirement already satisfied: tzdata>=2022.7 in c:\users\david\
anaconda3\lib\site-packages (from pandas->pandasql) (2025.2)
Requirement already satisfied: six>=1.5 in c:\users\david\anaconda3\
lib\site-packages (from python-dateutil>=2.8.2->pandas->pandasql)
(1.17.0)
Requirement already satisfied: greenlet!=0.4.17 in c:\users\david\
anaconda3\lib\site-packages (from sqlalchemy->pandasql) (3.1.1)
Requirement already satisfied: typing-extensions>=4.6.0 in c:\users\
david\anaconda3\lib\site-packages (from sqlalchemy->pandasql) (4.12.2)

import pandas as pd
import pandasql as ps

# Load CSV into Pandas DataFrame
df = pd.read_csv(r"C:\Users\david\Downloads\
airlines_flights_data.csv")
df

         index   airline   flight source_city departure_time stops  \
0            0  SpiceJet  SG-8709       Delhi        Evening  zero
1            1  SpiceJet  SG-8157       Delhi  Early_Morning  zero
2            2   AirAsia   I5-764       Delhi  Early_Morning  zero
3            3   Vistara   UK-995       Delhi        Morning  zero
4            4   Vistara   UK-963       Delhi        Morning  zero
...        ...       ...      ...         ...            ...   ...
300148  300148   Vistara   UK-822     Chennai        Morning   one
300149  300149   Vistara   UK-826     Chennai      Afternoon   one
300150  300150   Vistara   UK-832     Chennai  Early_Morning   one
300151  300151   Vistara   UK-828     Chennai  Early_Morning   one
300152  300152   Vistara   UK-822     Chennai        Morning   one

         arrival_time destination_city     class  duration  days_left
price
0               Night           Mumbai   Economy      2.17          1
5953
1             Morning           Mumbai   Economy      2.33          1
5953
2       Early_Morning           Mumbai   Economy      2.17          1
5956
3           Afternoon           Mumbai   Economy      2.25          1
5955
4             Morning           Mumbai   Economy      2.33          1
5955
...               ...              ...       ...       ...        ...
...
300148        Evening        Hyderabad  Business     10.08         49
69265
300149          Night        Hyderabad  Business     10.42         49
77105
```

```
300150           Night        Hyderabad  Business     13.83         49
79099
300151         Evening        Hyderabad  Business     10.00         49
81585
300152         Evening        Hyderabad  Business     10.08         49
81585

[300153 rows x 12 columns]
```

```
#-- 1.  Find the average ticket price per airline.
import pandasql as ps

query = """
SELECT airline, AVG(price) as avg_price
FROM df
WHERE source_city = 'Delhi'
GROUP BY airline
ORDER BY avg_price DESC
LIMIT 5;
"""

result = ps.sqldf(query, locals())
print(result)
```

```
     airline     avg_price
0    Vistara  28938.500342
1  Air_India  21899.890758
2   SpiceJet   6084.649762
3   GO_FIRST   5742.409679
4     Indigo   5386.570134
```

```
#-- 2. List the top 5 most expensive routes (source to destination).

import pandasql as ps
query= """
SELECT
    source_city,
    destination_city,
    AVG(price) AS Average_price
FROM
    df
GROUP BY
    source_city, destination_city
ORDER BY
    Average_price DESC
LIMIT 5;
"""
result = ps.sqldf(query, locals())
print(result)
```

```
  source_city destination_city  Average_price
0     Chennai        Bangalore   25081.850454
1     Kolkata          Chennai   23660.361040
2   Bangalore          Kolkata   23500.061229
3   Bangalore          Chennai   23321.850078
4      Mumbai        Bangalore   23147.873807
```

```
#3.Which airline offers the lowest average ticket price for Business
class?

import pandas as pd
import pandasql as ps
query = """
SELECT airline, AVG(price) AS avg_Business_price
FROM df
WHERE class = 'Business'
GROUP BY airline
ORDER BY avg_Business_price
LIMIT 1;
"""
result = ps.sqldf(query, locals())
print(result)
```

```
     airline  avg_Business_price
0  Air_India        47131.039212
```

```
#4.Find routes with more than 50 flights and their average ticket
prices.

import pandasql as ps
query = """
SELECT source_city, destination_city, COUNT(*) as
flight_count,AVG(price) as avg_price
FROM df
GROUP BY source_city, destination_city
HAVING COUNT(*)> 50;
"""
result = ps.sqldf(query, locals())
print(result)
```

```
   source_city destination_city  flight_count     avg_price
0    Bangalore          Chennai          6410  23321.850078
1    Bangalore            Delhi         13756  17723.313972
2    Bangalore        Hyderabad          8928  21226.121192
3    Bangalore          Kolkata         10028  23500.061229
4    Bangalore           Mumbai         12939  23128.618672
5      Chennai        Bangalore          6493  25081.850454
6      Chennai            Delhi          9783  18981.863948
7      Chennai        Hyderabad          6103  21591.345404
8      Chennai          Kolkata          6983  22669.932407
```

```
9       Chennai           Mumbai       9338  22765.849647
10        Delhi        Bangalore      14012  17880.216315
11        Delhi          Chennai      10780  19369.881354
12        Delhi        Hyderabad       9328  17347.288379
13        Delhi          Kolkata      11934  20566.409418
14        Delhi           Mumbai      15289  19355.829812
15    Hyderabad        Bangalore       7854  21347.177998
16    Hyderabad          Chennai       6395  21848.065989
17    Hyderabad            Delhi       8506  17243.945685
18    Hyderabad          Kolkata       7987  20823.893201
19    Hyderabad           Mumbai      10064  20080.865759
20      Kolkata        Bangalore       9824  22744.808428
21      Kolkata          Chennai       6653  23660.361040
22      Kolkata            Delhi      10506  19422.354559
23      Kolkata        Hyderabad       7897  21500.011397
24      Kolkata           Mumbai      11467  22078.883579
25       Mumbai        Bangalore      12885  23147.873807
26       Mumbai          Chennai      10130  22781.899112
27       Mumbai            Delhi      14809  18725.320008
28       Mumbai        Hyderabad      10470  21004.046705
29       Mumbai          Kolkata      12602  22379.146723
```

```
#5. Compare average price difference between Economy and Business
class per airline.

import pandasql as ps
query = """
SELECT
    airline,
    MAX(CASE WHEN class = 'Business' THEN avg_price END) -
    MAX(CASE WHEN class = 'Economy' THEN avg_price END) AS price_diff
FROM (
    SELECT
        airline,
        class,
        AVG(price) AS avg_price
    FROM df
    WHERE class IN ('Economy', 'Business')
    GROUP BY airline, class
) t
GROUP BY airline
ORDER BY price_diff DESC;

"""
result = ps.sqldf(query, locals())
print(result)
```

```
     airline    price_diff
0    Vistara  47670.084132
1  Air_India  39817.357044
```

```
2   SpiceJet           NaN
3     Indigo           NaN
4   GO_FIRST           NaN
5    AirAsia           NaN
```

```
#6. Which cities have the most incoming flights?

import pandasql as ps
query = """
SELECT destination_city as citty, COUNT(*) as incoming_flights
FROM df
GROUP BY destination_city
ORDER BY incoming_flights DESC;
"""
result = ps.sqldf(query, locals())
print(result)
```

```
       citty  incoming_flights
0     Mumbai             59097
1      Delhi             57360
2  Bangalore             51068
3    Kolkata             49534
4  Hyderabad             42726
5    Chennai             40368
```

```
#7.Which cities have the most outgoing flights?

import pandasql as ps
query = """
SELECT source_city as city, COUNT(*) as total_outgoing_flights
FROM df
GROUP BY source_city
ORDER BY total_outgoing_flights DESC;
"""
resul = ps.sqldf(query, locals())
print(result)
```

```
       citty  incoming_flights
0     Mumbai             59097
1      Delhi             57360
2  Bangalore             51068
3    Kolkata             49534
4  Hyderabad             42726
5    Chennai             40368
```

```
#8.Find the busiest route (most number of flights).

import pandasql as ps
query = """
SELECT source_city,destination_city, COUNT(*) as total_flights
FROM df
```

```
GROUP BY source_city,destination_city
ORDER BY total_flights DESC
LIMIT 1;
"""
result = ps.sqldf(query, locals())
print(result)
```

```
  source_city destination_city  total_flights
0       Delhi           Mumbai          15289
```

```
#9. List the top 3 airlines with cheapest average prices for each
route.

import pandas as pd

# Make route column
df["route"] = df["source_city"] + " → " + df["destination_city"]

# Step 1: Average price per airline per route
avg_prices = (
    df.groupby(["route", "airline"], as_index=False)["price"]
    .mean()
    .rename(columns={"price": "avg_price"})
)

# Step 2: Rank airlines within each route
avg_prices["rank"] = avg_prices.groupby("route")
["avg_price"].rank(method="first")

# Step 3: Keep only top 3 cheapest per route
top3 = avg_prices[avg_prices["rank"] <= 3].sort_values(["route",
"avg_price"])

top3
```

```
                   route   airline    avg_price  rank
0    Bangalore → Chennai   AirAsia  2073.043478   1.0
3    Bangalore → Chennai    Indigo  2363.326241   2.0
4    Bangalore → Chennai  SpiceJet  2613.310345   3.0
6      Bangalore → Delhi   AirAsia  4807.092426   1.0
8      Bangalore → Delhi  GO_FIRST  5524.702628   2.0
..                   ...       ...          ...   ...
168   Mumbai → Hyderabad  GO_FIRST  4603.866889   2.0
169   Mumbai → Hyderabad    Indigo  5870.954610   3.0
171     Mumbai → Kolkata   AirAsia  3977.937365   1.0
173     Mumbai → Kolkata  GO_FIRST  6106.502609   2.0
175     Mumbai → Kolkata  SpiceJet  7065.210689   3.0

[90 rows x 4 columns]
```

```
#10.Find flights with duration more than 5 hours but priced below
average.

import pandasql as ps
query ="""
SELECT *FROM df
WHERE duration >5
AND price < (SELECT AVG(price) FROM df);
"""
result = ps.sqldf(query,locals())
print(result)
```

```
         index    airline   flight source_city departure_time
stops  \
0           18    AirAsia  I5-747       Delhi        Evening
one
1           19    AirAsia  I5-747       Delhi        Evening
one
2           20   GO_FIRST  G8-266       Delhi  Early_Morning
one
3           21   GO_FIRST  G8-101       Delhi  Early_Morning
one
4           22   GO_FIRST  G8-103       Delhi        Evening
one
...        ...        ...      ...         ...            ...
...
169086  206663    Vistara  UK-826     Chennai      Afternoon
one
169087  206664    Vistara  UK-822     Chennai        Morning
one
169088  206665    Vistara  UK-824     Chennai          Night
one
169089  258681  Air_India  AI-776   Bangalore  Early_Morning
two_or_more
169090  258806  Air_India  AI-776   Bangalore  Early_Morning
two_or_more

         arrival_time destination_city    class  duration  days_left
price  \
0       Early_Morning           Mumbai  Economy     12.25          1
5949
1             Morning           Mumbai  Economy     16.33          1
5949
2             Evening           Mumbai  Economy     11.75          1
5954
3               Night           Mumbai  Economy     14.50          1
5954
4             Morning           Mumbai  Economy     15.67          1
5954
...               ...              ...      ...       ...        ...
```

```
...
169086         Morning        Hyderabad   Economy     20.58        49
8640
169087         Morning        Hyderabad   Economy     23.33        49
8640
169088           Night        Hyderabad   Economy     24.42        49
8640
169089       Afternoon        Hyderabad  Business      6.67        45
12000
169090       Afternoon        Hyderabad  Business      6.67        47
12000

                          route
0              Delhi → Mumbai
1              Delhi → Mumbai
2              Delhi → Mumbai
3              Delhi → Mumbai
4              Delhi → Mumbai
...                         ...
169086    Chennai → Hyderabad
169087    Chennai → Hyderabad
169088    Chennai → Hyderabad
169089  Bangalore → Hyderabad
169090  Bangalore → Hyderabad

[169091 rows x 13 columns]

#11. Which flight class shows the highest average price variation (std
deviation)?

import pandasql as ps
query = """
SELECT
    class, AVG(price) AS price_std_dev
FROM df
GROUP BY class
ORDER BY price_std_dev DESC;
"""
result = ps.sqldf(query,locals())
print(result)

      class  price_std_dev
0  Business   52540.081124
1   Economy    6572.342383

#12.Find how ticket prices vary with number of days left before
departure.

import pandasql as ps
query="""
```

```
SELECT days_left, AVG(price) as avg_price
FROM df
GROUP BY days_left
ORDER BY avg_price DESC;
"""
result = ps.sqldf(query, locals())
print(result)

    days_left     avg_price
0           2  30211.299801
1           3  28976.083569
2           5  26679.773368
3           4  25730.905653
4           9  25726.246072
5           7  25588.367351
6          10  25572.819134
7           8  24895.883995
8           6  24856.493902
9          11  22990.656070
10         14  22678.002363
11         12  22505.803322
12         13  22498.885384
13         15  21952.540852
14          1  21591.867151
15         16  20503.546237
16         17  20386.353949
17         18  19987.445168
18         27  19950.866195
19         23  19840.913451
20         24  19803.908896
21         29  19744.653119
22         38  19734.912316
23         20  19699.983390
24         22  19590.667385
25         25  19571.641791
26         30  19567.580834
27         34  19562.008266
28         28  19534.986047
29         36  19517.688444
30         19  19507.677375
31         37  19506.306516
32         21  19430.494058
33         31  19392.706612
34         41  19347.440460
35         43  19340.528894
36         33  19306.271739
37         46  19305.351623
38         39  19262.095556
39         32  19258.135308
40         35  19255.652996
```

```
41          26  19238.290278
42          45  19199.876307
43          42  19154.261659
44          40  19144.972439
45          44  19049.080174
46          48  18998.126851
47          49  18992.971888
48          47  18553.272038
```

```
#13. Which time of day (Morning, Evening, etc.) has the highest
average ticket price?

import pandasql as ps
query = """
SELECT arrival_time,departure_time, AVG(price) as avg_price
FROM df
GROUP BY arrival_time,departure_time
ORDER BY avg_price DESC
LIMIT 1;
"""
result = ps.sqldf(query, locals())
print(result)
```

```
  arrival_time departure_time     avg_price
0      Evening          Night  31425.824194
```

```
#14. Find Price trends by airline and travel class

import pandasql as ps
query= """
SELECT airline, class, AVG(price) as avg_price
FROM df
GROUP BY airline,class
ORDER BY airline,avg_price DESC;
"""
result = ps.sqldf(query, locals())
print(result)
```

```
     airline     class     avg_price
0    AirAsia   Economy   4091.072742
1  Air_India  Business  47131.039212
2  Air_India   Economy   7313.682169
3   GO_FIRST   Economy   5652.007595
4     Indigo   Economy   5324.216303
5   SpiceJet   Economy   6179.278881
6    Vistara  Business  55477.027777
7    Vistara   Economy   7806.943645
```

```
#15.Compare the airlines average prices for non-stop vs. 1-stop vs. 2+
stops flights.
```

```python
import pandasql as ps
query="""
SELECT airline, stops as stop_type, AVG(price) as avg_price
FROM df
GROUP BY airline, stops
ORDER BY airline, avg_price DESC;
"""
result = ps.sqldf(query, locals())
print(result)
```

```
      airline    stop_type     avg_price
0     AirAsia  two_or_more   4432.956367
1     AirAsia          one   4096.963741
2     AirAsia         zero   3747.960970
3   Air_India          one  24805.782885
4   Air_India         zero  14403.191918
5   Air_India  two_or_more  13771.177062
6    GO_FIRST  two_or_more   7107.708642
7    GO_FIRST          one   5972.272755
8    GO_FIRST         zero   3526.924915
9      Indigo  two_or_more   7834.838753
10     Indigo          one   5733.028878
11     Indigo         zero   4023.049037
12   SpiceJet          one   6789.364636
13   SpiceJet         zero   4556.430950
14    Vistara          one  32353.149720
15    Vistara  two_or_more  18850.767996
16    Vistara         zero  16416.273587
```