```
#
```

In [1]:
```
import pandas as pd
import warnings
warnings.filterwarnings('ignore')
import sqlite3
```

In [2]:
```
import csv

file_path = r'C:\Users\snigd\Downloads\DataTrain.csv'

flight_data = []

with open(file_path, mode='r', encoding='utf-8') as csvfile:

    reader = csv.DictReader(csvfile)

    for row in reader:
        flight_data.append(row)

for row in flight_data[:10]:  # Adjust the number to view more or fewer row
    print(row)
```


```
{'ID': '1', 'Airline': 'IndiGo', 'Date_of_Journey': '24-03-2019', 'Sourc
e': 'Banglore', 'Destination': 'New Delhi', 'Route': 'BLR → DEL', 'Dep_Tim
e': '22:20', 'Arrival_Time': '22-03-2023 01:10', 'Duration': '2h 50m', 'To
tal_Stops': 'non-stop', 'Additional_Info': 'No info', 'Price': '3897'}
{'ID': '2', 'Airline': 'Air India', 'Date_of_Journey': '01-05-2019', 'Sour
ce': 'Kolkata', 'Destination': 'Banglore', 'Route': 'CCU → IXR → BBI → BL
R', 'Dep_Time': '05:50', 'Arrival_Time': '13:15', 'Duration': '7h 25m', 'T
otal_Stops': '2 stops', 'Additional_Info': 'No info', 'Price': '7662'}
{'ID': '3', 'Airline': 'Jet Airways', 'Date_of_Journey': '09-06-2019', 'So
urce': 'Delhi', 'Destination': 'Cochin', 'Route': 'DEL → LKO → BOM → COK',
'Dep_Time': '09:25', 'Arrival_Time': '10-06-2023 04:25', 'Duration': '19
h', 'Total_Stops': '2 stops', 'Additional_Info': 'No info', 'Price': '1388
2'}
{'ID': '4', 'Airline': 'IndiGo', 'Date_of_Journey': '12-05-2019', 'Sourc
e': 'Kolkata', 'Destination': 'Banglore', 'Route': 'CCU → NAG → BLR', 'Dep
_Time': '18:05', 'Arrival_Time': '23:30', 'Duration': '5h 25m', 'Total_Sto
ps': '1 stop', 'Additional_Info': 'No info', 'Price': '6218'}
{'ID': '5', 'Airline': 'IndiGo', 'Date_of_Journey': '01-03-2019', 'Sourc
e': 'Banglore', 'Destination': 'New Delhi', 'Route': 'BLR → NAG → DEL', 'D
ep_Time': '16:50', 'Arrival_Time': '21:35', 'Duration': '4h 45m', 'Total_S
tops': '1 stop', 'Additional_Info': 'No info', 'Price': '13302'}
{'ID': '6', 'Airline': 'SpiceJet', 'Date_of_Journey': '24-06-2019', 'Sourc
e': 'Kolkata', 'Destination': 'Banglore', 'Route': 'CCU → BLR', 'Dep_Tim
e': '09:00', 'Arrival_Time': '11:25', 'Duration': '2h 25m', 'Total_Stops':
'non-stop', 'Additional_Info': 'No info', 'Price': '3873'}
{'ID': '7', 'Airline': 'Jet Airways', 'Date_of_Journey': '12-03-2019', 'So
urce': 'Banglore', 'Destination': 'New Delhi', 'Route': 'BLR → BOM → DEL',
'Dep_Time': '18:55', 'Arrival_Time': '13-03-2023 10:25', 'Duration': '15h
30m', 'Total_Stops': '1 stop', 'Additional_Info': 'In-flight meal not incl
uded', 'Price': '11087'}
{'ID': '8', 'Airline': 'Jet Airways', 'Date_of_Journey': '01-03-2019', 'So
urce': 'Banglore', 'Destination': 'New Delhi', 'Route': 'BLR → BOM → DEL',
'Dep_Time': '08:00', 'Arrival_Time': '02-03-2023 05:05', 'Duration': '21h
5m', 'Total_Stops': '1 stop', 'Additional_Info': 'No info', 'Price': '2227
0'}
{'ID': '9', 'Airline': 'Jet Airways', 'Date_of_Journey': '12-03-2019', 'So
urce': 'Banglore', 'Destination': 'New Delhi', 'Route': 'BLR → BOM → DEL',
'Dep_Time': '08:55', 'Arrival_Time': '13-03-2023 10:25', 'Duration': '25h
30m', 'Total_Stops': '1 stop', 'Additional_Info': 'In-flight meal not incl
uded', 'Price': '11087'}
{'ID': '10', 'Airline': 'Multiple carriers', 'Date_of_Journey': '27-05-201
9', 'Source': 'Delhi', 'Destination': 'Cochin', 'Route': 'DEL → BOM → CO
K', 'Dep_Time': '11:25', 'Arrival_Time': '19:15', 'Duration': '7h 50m', 'T
otal_Stops': '1 stop', 'Additional_Info': 'No info', 'Price': '8625'}
```

In [3]:
```
def create_connection(db_file, delete_db=False):
    import os
    if delete_db and os.path.exists(db_file):
        os.remove(db_file)

    conn = None
    try:
        conn = sqlite3.connect(db_file)
        conn.execute("PRAGMA foreign_keys = 1")
    except Error as e:
        print(e)

    return conn


def create_table(conn, create_table_sql):
    try:
        c = conn.cursor()
        c.execute(create_table_sql)
    except Error as e:
        print(e)

def execute_sql_statement(sql_statement, conn):
    cur = conn.cursor()
    cur.execute(sql_statement)

    rows = cur.fetchall()

    return rows

create_table_sql = """CREATE TABLE IF NOT EXISTS [Flight] (
[ID] INTEGER NOT NULL PRIMARY KEY,
[Airline] TEXT NOT NULL,
[Source] TEXT NOT NULL,
[Destination] TEXT NOT NULL,
[Route] TEXT NOT NULL,
[Total_Stops] TEXT NOT NULL
);"""
conn_normalized = create_connection('normalized3.db', delete_db = True)

def insert_flight(conn_normalized, values):
    sql = ''' INSERT INTO Flight(ID, Airline, Source, Destination, Route, T
    cur = conn_normalized.cursor()
    cur.execute(sql, values)
    return cur.lastrowid

with conn_normalized:
    create_table(conn_normalized, create_table_sql)
    for fd in flight_data:
        insert_tuple = (fd['ID'], fd['Airline'], fd['Source'], fd['Destinat
        insert_flight(conn_normalized, insert_tuple)
```

In [4]:
```
create_table_sql = """CREATE TABLE IF NOT EXISTS [Schedule] (
[ID] INTEGER NOT NULL PRIMARY KEY,
[Date_of_Journey] TEXT NOT NULL,
[Dep_Time] TEXT NOT NULL,
[Arrival_Time] TEXT NOT NULL,
[Duration] TEXT NOT NULL,
FOREIGN KEY(ID) REFERENCES Flight(ID)
);"""
conn_normalized = create_connection('normalized3.db')

def insert_schedule(conn_normalized, values):
    sql = ''' INSERT INTO Schedule(ID, Date_of_Journey, Dep_Time, Arrival_T
    cur = conn_normalized.cursor()
    cur.execute(sql, values)
    return cur.lastrowid

with conn_normalized:
    create_table(conn_normalized, create_table_sql)
    for fd in flight_data:
        insert_tuple = (fd['ID'], fd['Date_of_Journey'], fd['Dep_Time'], fd
        insert_schedule(conn_normalized, insert_tuple)
```

In [5]:
```
create_table_sql = """CREATE TABLE IF NOT EXISTS [Pricing] (
[ID] INTEGER NOT NULL PRIMARY KEY,
[Additional_Info] TEXT NOT NULL,
[Price] INTEGER NOT NULL,
FOREIGN KEY(ID) REFERENCES Flight(ID)
);"""
conn_normalized = create_connection('normalized3.db')

def insert_pricing(conn_normalized, values):
    sql = ''' INSERT INTO Pricing(ID, Additional_Info, Price) VALUES(?, ?,
    cur = conn_normalized.cursor()
    cur.execute(sql, values)
    return cur.lastrowid

with conn_normalized:
    create_table(conn_normalized, create_table_sql)
    for fd in flight_data:
        insert_tuple = (fd['ID'], fd['Additional_Info'], fd['Price'])
        insert_pricing(conn_normalized, insert_tuple)
```

In [6]:
```
import pandas as pd

conn_normalized = create_connection('normalized3.db')

sql_statement = """SELECT Flight.ID, Airline, Source, Destination, Route, T
Date_of_Journey, Dep_Time, Arrival_Time, Duration,
Additional_Info, Price
FROM Flight JOIN Schedule ON Flight.ID = Schedule.ID
JOIN Pricing ON Pricing.ID = Flight.ID"""

dats = execute_sql_statement(sql_statement, conn_normalized)

df = pd.DataFrame(dats, columns=['ID', 'Airline', 'Source', 'Destination',

print(df)


conn_normalized.commit()
conn_normalized.close()
```

```
          ID      Airline    Source Destination                  Route  \
0          1       IndiGo  Banglore   New Delhi              BLR → DEL
1          2    Air India   Kolkata    Banglore  CCU → IXR → BBI → BLR
2          3  Jet Airways     Delhi      Cochin  DEL → LKO → BOM → COK
3          4       IndiGo   Kolkata    Banglore        CCU → NAG → BLR
4          5       IndiGo  Banglore   New Delhi        BLR → NAG → DEL
...      ...          ...       ...         ...                    ...
10458  10459     Air Asia   Kolkata    Banglore              CCU → BLR
10459  10460    Air India   Kolkata    Banglore              CCU → BLR
10460  10461  Jet Airways  Banglore       Delhi              BLR → DEL
10461  10462      Vistara  Banglore   New Delhi              BLR → DEL
10462  10463    Air India     Delhi      Cochin  DEL → GOI → BOM → COK

      Total_Stops Date_of_Journey Dep_Time      Arrival_Time Duration  \
0        non-stop      24-03-2019    22:20  22-03-2023 01:10   2h 50m
1         2 stops      01-05-2019    05:50             13:15   7h 25m
2         2 stops      09-06-2019    09:25  10-06-2023 04:25      19h
3          1 stop      12-05-2019    18:05             23:30   5h 25m
4          1 stop      01-03-2019    16:50             21:35   4h 45m
...           ...             ...      ...               ...      ...
10458    non-stop      09-04-2019    19:55             22:25   2h 30m
10459    non-stop      27-04-2019    20:45             23:20   2h 35m
10460    non-stop      27-04-2019    08:20             11:20       3h
10461    non-stop      01-03-2019    11:30             14:10   2h 40m
10462     2 stops      09-05-2019    10:55             19:15   8h 20m

      Additional_Info  Price
0             No info   3897
1             No info   7662
2             No info  13882
3             No info   6218
4             No info  13302
...               ...    ...
10458         No info   4107
10459         No info   4145
10460         No info   7229
10461         No info  12648
10462         No info  11753

[10463 rows x 12 columns]
```

In [7]:
```
df.head()
```

Out[7]:

	ID	Airline	Source	Destination	Route	Total_Stops	Date_of_Journey	Dep_Time	Arriva
0	1	IndiGo	Banglore	New Delhi	BLR → DEL	non-stop	24-03-2019	22:20	22-0
1	2	Air India	Kolkata	Banglore	CCU → IXR → BBI → BLR	2 stops	01-05-2019	05:50	
2	3	Jet Airways	Delhi	Cochin	DEL → LKO → BOM → COK	2 stops	09-06-2019	09:25	10-0
3	4	IndiGo	Kolkata	Banglore	CCU → NAG → BLR	1 stop	12-05-2019	18:05	
4	5	IndiGo	Banglore	New Delhi	BLR → NAG → DEL	1 stop	01-03-2019	16:50	

In [8]:
```
#Checking if there are any null/na values

df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10463 entries, 0 to 10462
Data columns (total 12 columns):
 #   Column           Non-Null Count  Dtype 
---  ------           --------------  ----- 
 0   ID               10463 non-null  int64 
 1   Airline          10463 non-null  object
 2   Source           10463 non-null  object
 3   Destination      10463 non-null  object
 4   Route            10463 non-null  object
 5   Total_Stops      10463 non-null  object
 6   Date_of_Journey  10463 non-null  object
 7   Dep_Time         10463 non-null  object
 8   Arrival_Time     10463 non-null  object
 9   Duration         10463 non-null  object
 10  Additional_Info  10463 non-null  object
 11  Price            10463 non-null  int64 
dtypes: int64(2), object(10)
memory usage: 981.0+ KB
```

In [9]:
```
#Deleting the duplicate rows


import pandas as pd

df1 = df.drop_duplicates().reset_index(drop=True)

df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10463 entries, 0 to 10462
Data columns (total 12 columns):
 #   Column           Non-Null Count  Dtype 
---  ------           --------------  ----- 
 0   ID               10463 non-null  int64 
 1   Airline          10463 non-null  object
 2   Source           10463 non-null  object
 3   Destination      10463 non-null  object
 4   Route            10463 non-null  object
 5   Total_Stops      10463 non-null  object
 6   Date_of_Journey  10463 non-null  object
 7   Dep_Time         10463 non-null  object
 8   Arrival_Time     10463 non-null  object
 9   Duration         10463 non-null  object
 10  Additional_Info  10463 non-null  object
 11  Price            10463 non-null  int64 
dtypes: int64(2), object(10)
memory usage: 981.0+ KB
```

In [10]:
```
import pandas as pd

bins = [0, 4, 8, 12, 16,20,24]
labels = ['Late_Night', 'Early_Morning','Morning', 'Afternoon', 'Evening','

df1['Dep_Time_Category'] = pd.cut(pd.to_datetime(df1['Dep_Time']).dt.hour,

print(df1['Dep_Time_Category'].value_counts())
```

```
Morning          2687
Early_Morning    2289
Evening          2135
Night            1644
Afternoon        1413
Late_Night        295
Name: Dep_Time_Category, dtype: int64
```

In [11]:
```
df1.head()
```

Out[11]:

	ID	Airline	Source	Destination	Route	Total_Stops	Date_of_Journey	Dep_Time	Arriva
0	1	IndiGo	Banglore	New Delhi	BLR → DEL	non-stop	24-03-2019	22:20	22-0
1	2	Air India	Kolkata	Banglore	CCU → IXR → BBI → BLR	2 stops	01-05-2019	05:50	
2	3	Jet Airways	Delhi	Cochin	DEL → LKO → BOM → COK	2 stops	09-06-2019	09:25	10-0
3	4	IndiGo	Kolkata	Banglore	CCU → NAG → BLR	1 stop	12-05-2019	18:05	
4	5	IndiGo	Banglore	New Delhi	BLR → NAG → DEL	1 stop	01-03-2019	16:50	

In [12]:
```
df1.Arrival_Time.value_counts()
```

Out[12]:
```
19:00                412
21:00                360
19:15                333
16:10                154
12:35                122
                    ...
04-05-2023 00:50       1
02-06-2023 00:50       1
02-06-2023 00:25       1
13-03-2023 08:55       1
13-03-2023 21:20       1
Name: Arrival_Time, Length: 1343, dtype: int64
```

In [13]:
```
bins = [0, 4, 8, 12, 16,20,24]
labels = ['Late_Night', 'Early_Morning','Morning', 'Afternoon', 'Evening','

df1['Arrival_Time_Category'] = pd.cut(pd.to_datetime(df1['Arrival_Time']).d
print(df1['Arrival_Time_Category'].value_counts())
```

```
Evening          2624
Night            2205
Morning          1729
Afternoon        1640
Early_Morning    1292
Late_Night        973
Name: Arrival_Time_Category, dtype: int64
```

In [14]:
```
df1.head()
```

Out[14]:

	ID	Airline	Source	Destination	Route	Total_Stops	Date_of_Journey	Dep_Time	Arriva
0	1	IndiGo	Banglore	New Delhi	BLR → DEL	non-stop	24-03-2019	22:20	22-0
1	2	Air India	Kolkata	Banglore	CCU → IXR → BBI → BLR	2 stops	01-05-2019	05:50	
2	3	Jet Airways	Delhi	Cochin	DEL → LKO → BOM → COK	2 stops	09-06-2019	09:25	10-0
3	4	IndiGo	Kolkata	Banglore	CCU → NAG → BLR	1 stop	12-05-2019	18:05	
4	5	IndiGo	Banglore	New Delhi	BLR → NAG → DEL	1 stop	01-03-2019	16:50	

In [15]:
```
df1.Total_Stops.value_counts()
```

Out[15]:
```
1 stop      5625
non-stop    3475
2 stops     1318
3 stops       43
                1
4 stops         1
Name: Total_Stops, dtype: int64
```

In [16]:
```
#map string values in the total stops column into integer values
dictionary={'non-stop':0,
    '1 stop':1,
    '2 stops':2,
    '3 stops':3,
    '4 stops':4}

df1['Total_Stops']=df1['Total_Stops'].map(dictionary)
df1['Total_Stops'].value_counts()
```

Out[16]:
```
1.0    5625
0.0    3475
2.0    1318
3.0      43
4.0       1
Name: Total_Stops, dtype: int64
```

In [17]:
```
df1.head()
```

Out[17]:

	ID	Airline	Source	Destination	Route	Total_Stops	Date_of_Journey	Dep_Time	Arriva
0	1	IndiGo	Banglore	New Delhi	BLR → DEL	0.0	24-03-2019	22:20	22-0
1	2	Air India	Kolkata	Banglore	CCU → IXR → BBI → BLR	2.0	01-05-2019	05:50	
2	3	Jet Airways	Delhi	Cochin	DEL → LKO → BOM → COK	2.0	09-06-2019	09:25	10-0
3	4	IndiGo	Kolkata	Banglore	CCU → NAG → BLR	1.0	12-05-2019	18:05	
4	5	IndiGo	Banglore	New Delhi	BLR → NAG → DEL	1.0	01-03-2019	16:50	

In [18]:
```
drop = ['Arrival_Time', 'Dep_Time','Route']

df2= df1.drop(columns=drop)
```

In [19]:
```
df2.head()
```

Out[19]:

	ID	Airline	Source	Destination	Total_Stops	Date_of_Journey	Duration	Additional_Info
0	1	IndiGo	Banglore	New Delhi	0.0	24-03-2019	2h 50m	No info
1	2	Air India	Kolkata	Banglore	2.0	01-05-2019	7h 25m	No info
2	3	Jet Airways	Delhi	Cochin	2.0	09-06-2019	19h	No info
3	4	IndiGo	Kolkata	Banglore	1.0	12-05-2019	5h 25m	No info
4	5	IndiGo	Banglore	New Delhi	1.0	01-03-2019	4h 45m	No info

In [20]:
```
import pandas as pd
df2['Duration'] = pd.to_timedelta(df2['Duration'])

df2['Total_Duration_Hours'] = df2['Duration'].dt.total_seconds() / 3600.0

print(df2[['Duration', 'Total_Duration_Hours']])
```

```
             Duration  Total_Duration_Hours
0     0 days 02:50:00              2.833333
1     0 days 07:25:00              7.416667
2     0 days 19:00:00             19.000000
3     0 days 05:25:00              5.416667
4     0 days 04:45:00              4.750000
...               ...                   ...
10458 0 days 02:30:00              2.500000
10459 0 days 02:35:00              2.583333
10460 0 days 03:00:00              3.000000
10461 0 days 02:40:00              2.666667
10462 0 days 08:20:00              8.333333

[10463 rows x 2 columns]
```

In [21]:
```
df2.head()
```

Out[21]:

	ID	Airline	Source	Destination	Total_Stops	Date_of_Journey	Duration	Additional_Info
0	1	IndiGo	Banglore	New Delhi	0.0	24-03-2019	0 days 02:50:00	No info
1	2	Air India	Kolkata	Banglore	2.0	01-05-2019	0 days 07:25:00	No info
2	3	Jet Airways	Delhi	Cochin	2.0	09-06-2019	0 days 19:00:00	No info
3	4	IndiGo	Kolkata	Banglore	1.0	12-05-2019	0 days 05:25:00	No info
4	5	IndiGo	Banglore	New Delhi	1.0	01-03-2019	0 days 04:45:00	No info

In [22]:
```
df2['Date_of_Journey'] = pd.to_datetime(df2['Date_of_Journey'], format='%d-
df2['Weekday_of_Journey'] = df2['Date_of_Journey'].dt.weekday.astype('objec
```

In [23]:
```
df2['Month_of_Journey'] = df2['Date_of_Journey'].dt.month.astype('object')
```

In [24]:
```
df2.Month_of_Journey.value_counts()
```

Out[24]:
```
5    3396
6    3311
3    2678
4    1078
Name: Month_of_Journey, dtype: int64
```

In [25]:
```
drop = ['Duration', 'Date_of_Journey']

df3 = df2.drop(columns=drop)
```

In [26]:
```
df3.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10463 entries, 0 to 10462
Data columns (total 12 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   ID                     10463 non-null  int64
 1   Airline                10463 non-null  object
 2   Source                 10463 non-null  object
 3   Destination            10463 non-null  object
 4   Total_Stops            10462 non-null  float64
 5   Additional_Info        10463 non-null  object
 6   Price                  10463 non-null  int64
 7   Dep_Time_Category      10463 non-null  category
 8   Arrival_Time_Category  10463 non-null  category
 9   Total_Duration_Hours   10463 non-null  float64
 10  Weekday_of_Journey     10463 non-null  object
 11  Month_of_Journey       10463 non-null  object
dtypes: category(2), float64(2), int64(2), object(6)
memory usage: 838.4+ KB
```

In [27]:
```
df3.Additional_Info.value_counts()
```

Out[27]:
```
No info                         8183
In-flight meal not included     1926
No check-in baggage included     318
1 Long layover                    19
Change airports                    7
Business class                     4
No Info                            3
1 Short layover                    1
Red-eye flight                     1
2 Long layover                     1
Name: Additional_Info, dtype: int64
```

In [28]:
```
# dropping this column as there is no info in it

drop = ['Additional_Info']

df4 = df3.drop(columns=drop)
```

In [29]:
```
df4.head()
```

Out[29]:

	ID	Airline	Source	Destination	Total_Stops	Price	Dep_Time_Category	Arrival_Time_C
0	1	IndiGo	Banglore	New Delhi	0.0	3897	Night	La
1	2	Air India	Kolkata	Banglore	2.0	7662	Early_Morning	A
2	3	Jet Airways	Delhi	Cochin	2.0	13882	Morning	Early_
3	4	IndiGo	Kolkata	Banglore	1.0	6218	Evening	
4	5	IndiGo	Banglore	New Delhi	1.0	13302	Evening	

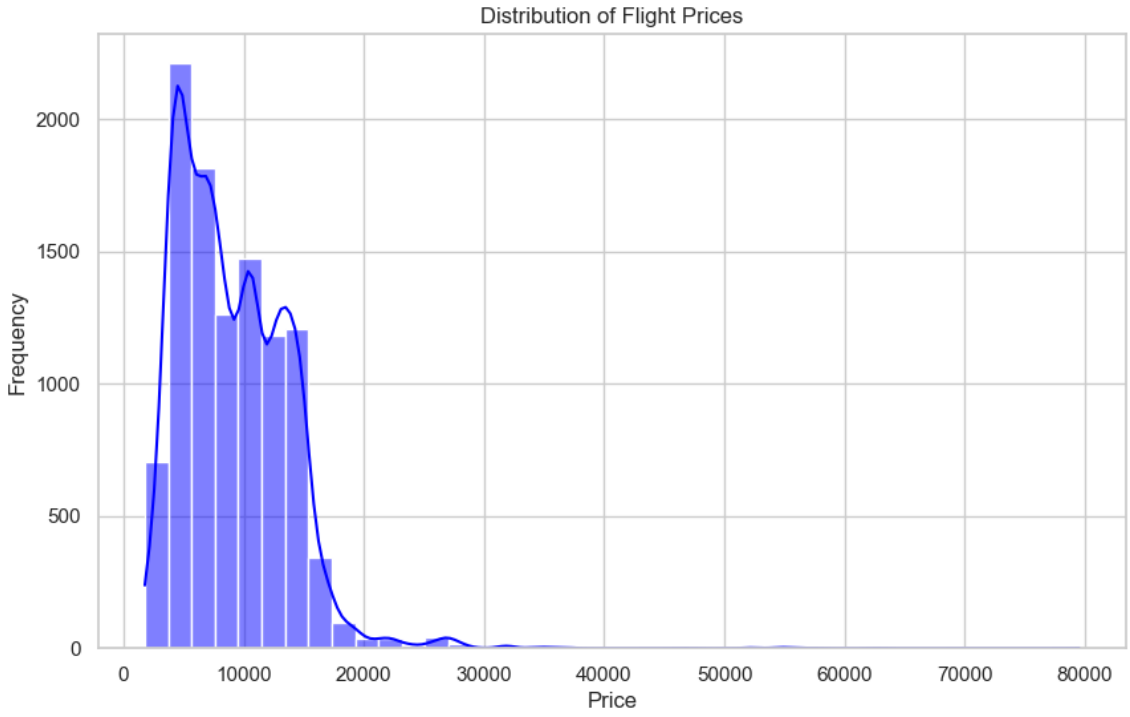
In [30]:
```
df4.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10463 entries, 0 to 10462
Data columns (total 11 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   ID                     10463 non-null  int64
 1   Airline                10463 non-null  object
 2   Source                 10463 non-null  object
 3   Destination            10463 non-null  object
 4   Total_Stops            10462 non-null  float64
 5   Price                  10463 non-null  int64
 6   Dep_Time_Category      10463 non-null  category
 7   Arrival_Time_Category  10463 non-null  category
 8   Total_Duration_Hours   10463 non-null  float64
 9   Weekday_of_Journey     10463 non-null  object
 10  Month_of_Journey       10463 non-null  object
dtypes: category(2), float64(2), int64(2), object(5)
memory usage: 756.7+ KB
```

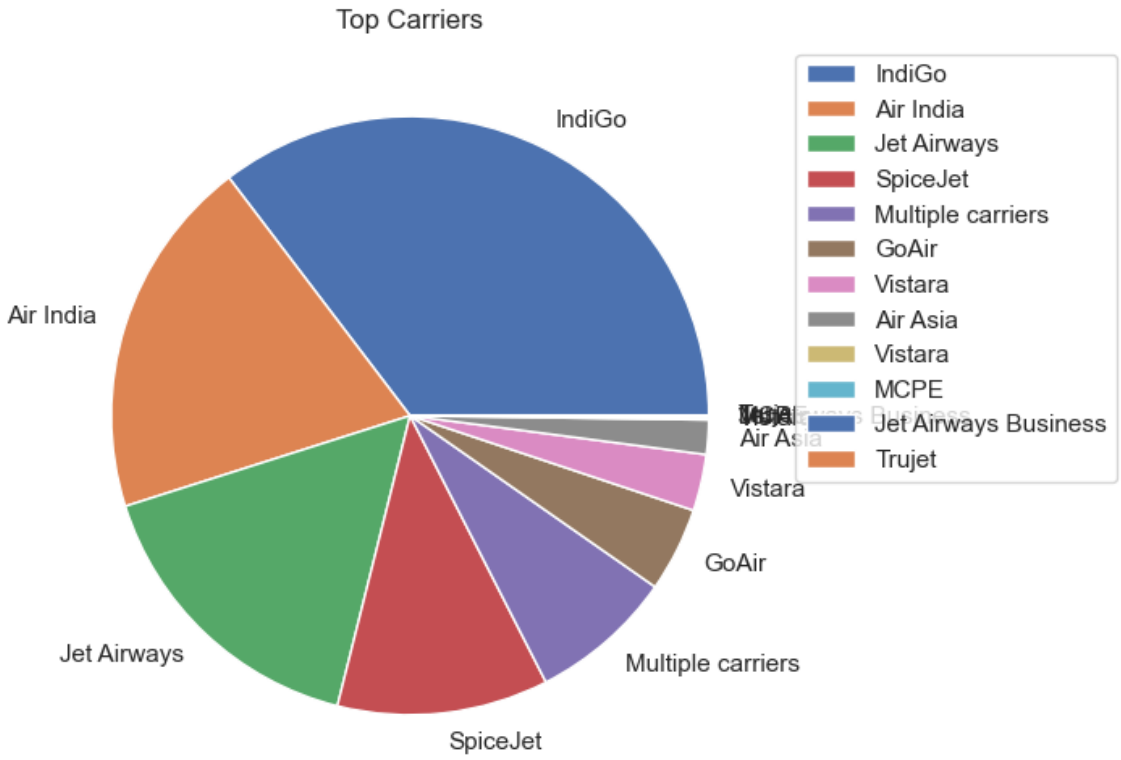
In [31]:
```
df4.dropna(inplace=True)
```

In [32]:
```
df4['Dep_Time_Category'] = df4['Dep_Time_Category'].astype('object')
df4['Arrival_Time_Category'] = df4['Arrival_Time_Category'].astype('object'
```

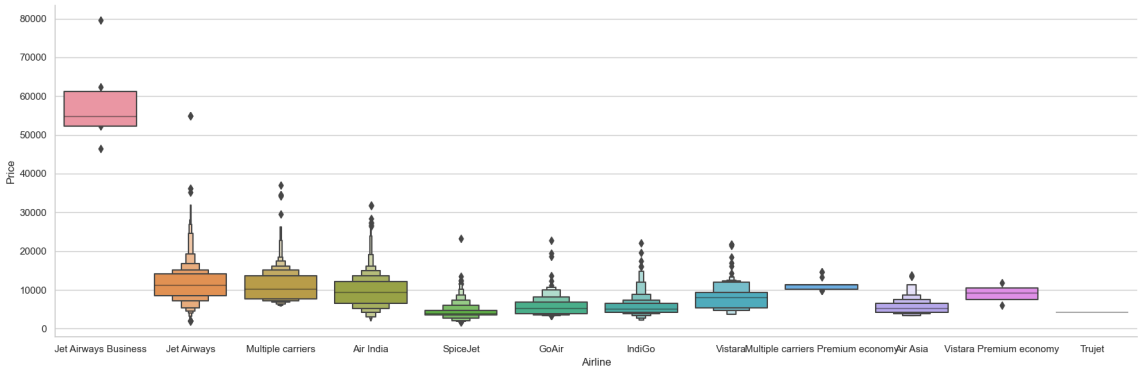
In [33]:
```
import matplotlib.pyplot as plt
import seaborn as sns

# Setting the style for the plots
sns.set(style="whitegrid")

# Plotting the distribution of the 'Price' variable
plt.figure(figsize=(10, 6))
sns.histplot(df4['Price'], kde=True, bins=40, color='blue')
plt.title('Distribution of Flight Prices')
plt.xlabel('Price')
plt.ylabel('Frequency')
plt.show()
```



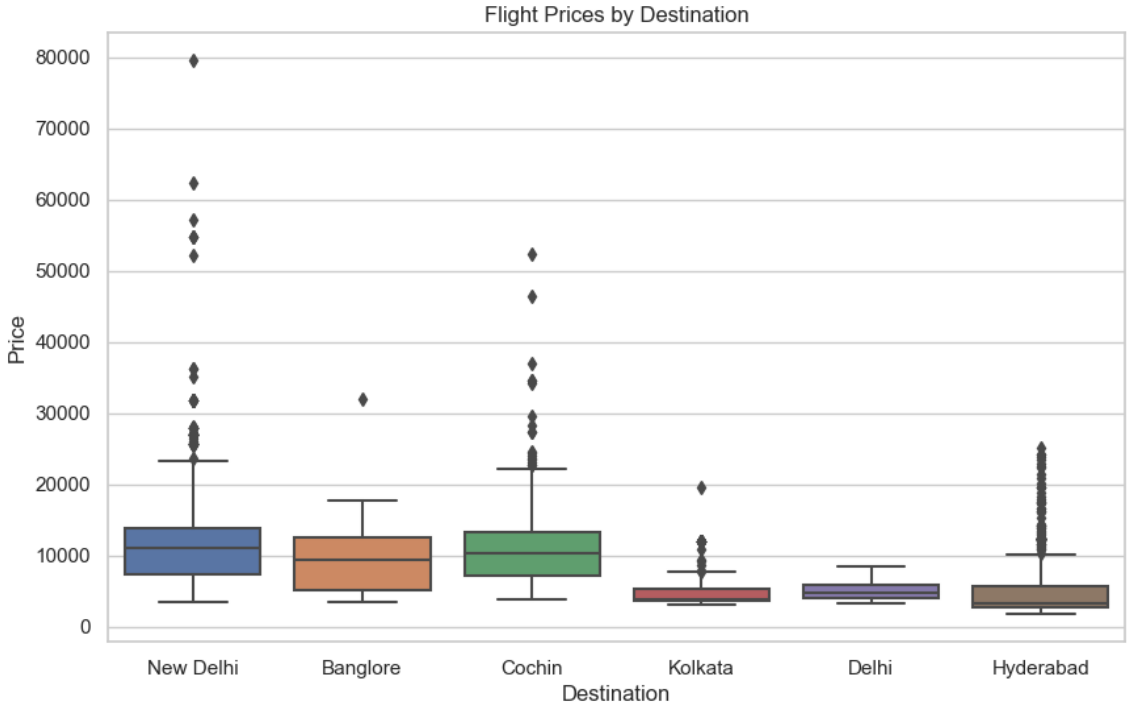
In [37]:
```
import numpy as np
top_carriers = np.array(df4['Airline'].value_counts(sort=True))


labels = ['IndiGo','Air India','Jet Airways','SpiceJet','Multiple carriers'

plt.figure(figsize=(8,6))
plt.pie(top_carriers, labels=labels,shadow=False)
plt.legend(labels=labels,
           bbox_to_anchor=(1,1))
plt.title('Top Carriers')
plt.show()
```



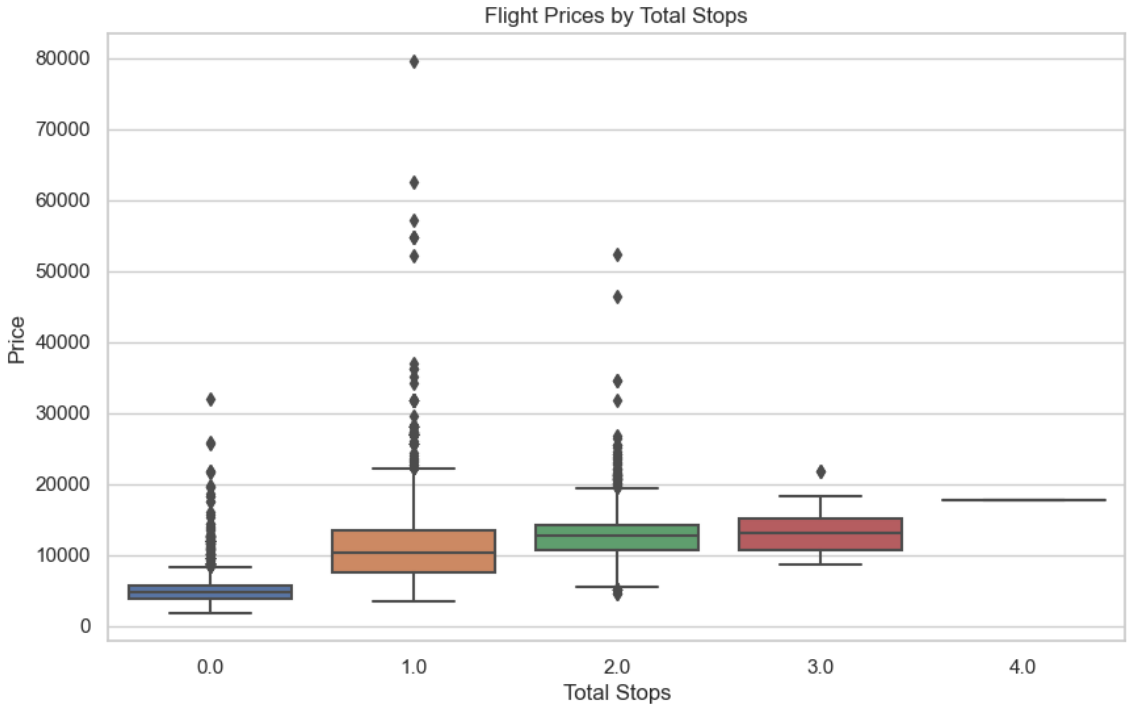
In [38]:
```
# From graph we can see that Jet Airways Business have the highest Price.
# Apart from the first Airline almost all are having similar median
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Airline vs Price
sns.catplot(y = "Price", x = "Airline", data = df4.sort_values("Price", asc
plt.show()
```



In [39]:
```
# Box plot to see how price varies with 'Destination'
plt.figure(figsize=(10, 6))
sns.boxplot(x='Destination', y='Price', data=df4)
plt.title('Flight Prices by Destination')
plt.xlabel('Destination')
plt.ylabel('Price')
plt.show()
```



In [40]:
```
# Box plot to see how price varies with 'Total_Stops'
plt.figure(figsize=(10, 6))
sns.boxplot(x='Total_Stops', y='Price', data=df4)
plt.title('Flight Prices by Total Stops')
plt.xlabel('Total Stops')
plt.ylabel('Price')
plt.show()
```



In [41]:
```
## Outlier analysis

import numpy as np
df=df4.copy()
cols = df.columns
all_outliers = []

for col in cols:
    if np.issubdtype(df[col].dtype, np.number):
        mean_val = df[col].mean()
        sd_val = df[col].std()
        z_scores = (df[col] - mean_val) / sd_val
        outliers = np.where((z_scores < -3) | (z_scores > 3))[0]
        all_outliers.extend(outliers)

# Get unique indices of all outliers
all_outliers = np.unique(all_outliers)

# Remove rows with outliers
df1 = df.drop(index=all_outliers).reset_index(drop=True)

# Display the cleaned DataFrame
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10273 entries, 0 to 10272
Data columns (total 11 columns):
 #   Column                 Non-Null Count  Dtype  
---  ------                 --------------  -----  
 0   ID                     10273 non-null  int64  
 1   Airline                10273 non-null  object 
 2   Source                 10273 non-null  object 
 3   Destination            10273 non-null  object 
 4   Total_Stops            10273 non-null  float64
 5   Price                  10273 non-null  int64  
 6   Dep_Time_Category      10273 non-null  object 
 7   Arrival_Time_Category  10273 non-null  object 
 8   Total_Duration_Hours   10273 non-null  float64
 9   Weekday_of_Journey     10273 non-null  object 
 10  Month_of_Journey       10273 non-null  object 
dtypes: float64(2), int64(2), object(7)
memory usage: 883.0+ KB
```

In [42]:
```
# Observing if there are any right skewed varibales in my dataset

import matplotlib.pyplot as plt
from scipy.stats import skew
from sklearn.preprocessing import FunctionTransformer

continuous_subset = df1.select_dtypes(include=np.number)

# Identify right-skewed variables
skewed_vars = [col for col in continuous_subset.columns if np.abs(df1[col].

print("Skewed Variables:", skewed_vars)
```

```
Skewed Variables: ['Price', 'Total_Duration_Hours']
```

In [43]:
```
# to see the minimum of the observation

min_values = continuous_subset[skewed_vars].min()
print("Minimum Values:", min_values)
```

```
Minimum Values: Price                   1759.000000
Total_Duration_Hours       0.083333
dtype: float64
```

In [44]:
```
df_log = df1.copy()
df_log[skewed_vars] = np.log(df_log[skewed_vars])
df_log.head()
```

Out[44]:

	ID	Airline	Source	Destination	Total_Stops	Price	Dep_Time_Category	Arrival_Time
0	1	IndiGo	Banglore	New Delhi	0.0	8.267962	Night	
1	2	Air India	Kolkata	Banglore	2.0	8.944028	Early_Morning	
2	3	Jet Airways	Delhi	Cochin	2.0	9.538348	Morning	Ea
3	4	IndiGo	Kolkata	Banglore	1.0	8.735204	Evening	
4	5	IndiGo	Banglore	New Delhi	1.0	9.495670	Evening	

In [45]:
```
airlines = df_log.groupby(["Airline"])["Price"].mean().sort_values().index
airlines
```

Out[45]:
```
Index(['SpiceJet', 'Trujet', 'Air Asia', 'IndiGo', 'GoAir', 'Vistara',
       'Air India', 'Vistara Premium economy', 'Multiple carriers',
       'Jet Airways', 'Multiple carriers Premium economy',
       'Jet Airways Business'],
      dtype='object', name='Airline')
```

In [46]:
```
dict_airlines = {key:index for index , key in enumerate(airlines , 0)}
dict_airlines
```

Out[46]:
```
{'SpiceJet': 0,
 'Trujet': 1,
 'Air Asia': 2,
 'IndiGo': 3,
 'GoAir': 4,
 'Vistara': 5,
 'Air India': 6,
 'Vistara Premium economy': 7,
 'Multiple carriers': 8,
 'Jet Airways': 9,
 'Multiple carriers Premium economy': 10,
 'Jet Airways Business': 11}
```

In [47]:
```
df_log['Airline'] = df_log['Airline'].map(dict_airlines)
df_log.head()
```

Out[47]:

	ID	Airline	Source	Destination	Total_Stops	Price	Dep_Time_Category	Arrival_Time
0	1	3	Banglore	New Delhi	0.0	8.267962	Night	
1	2	6	Kolkata	Banglore	2.0	8.944028	Early_Morning	
2	3	9	Delhi	Cochin	2.0	9.538348	Morning	Ear
3	4	3	Kolkata	Banglore	1.0	8.735204	Evening	
4	5	3	Banglore	New Delhi	1.0	9.495670	Evening	

In [48]:
```
from sklearn.preprocessing import StandardScaler

# Separate numeric and non-numeric columns
numeric_columns = df_log.select_dtypes(include='number')
non_numeric_columns = df_log.select_dtypes(exclude='number')

# Scale numeric columns using StandardScaler from scikit-learn
scaler = StandardScaler()
scaled_numeric_columns = pd.DataFrame(scaler.fit_transform(numeric_columns)

# Combine scaled numeric columns and non-numeric columns
df_s = pd.concat([scaled_numeric_columns, non_numeric_columns], axis=1)

# Display the resulting DataFrame
df_s.head()
```

Out[48]:

	ID	Airline	Total_Stops	Price	Total_Duration_Hours	Source	Destination
0	-1.732302	-1.011394	-1.219149	-1.390365	-1.034602	Banglore	New Delhi
1	-1.731971	-0.005191	1.882081	-0.047202	0.034476	Kolkata	Banglore
2	-1.731640	1.001012	1.882081	1.133554	1.079594	Delhi	Cochin
3	-1.731308	-1.011394	0.331466	-0.462081	-0.314651	Kolkata	Banglore
4	-1.730977	-1.011394	0.331466	1.048762	-0.460564	Banglore	New Delhi

In [49]:
```
df_s.Month_of_Journey.value_counts()
```

Out[49]:
```
5    3371
6    3282
3    2544
4    1076
Name: Month_of_Journey, dtype: int64
```

In [50]:
```
table = pd.crosstab(df_s['Source'], df_s['Destination'], margins=True, marg
table
```

Out[50]:

Source \ Destination	Banglore	Cochin	Delhi	Hyderabad	Kolkata	New Delhi	Total
Banglore	0	0	1262	0	0	837	2099
Chennai	0	0	0	0	381	0	381
Delhi	0	4266	0	0	0	0	4266
Kolkata	2841	0	0	0	0	0	2841
Mumbai	0	0	0	686	0	0	686
Total	2841	4266	1262	686	381	837	10273

In [51]:
```
df_s['Destination'] = df_s['Destination'].replace({'New Delhi': 'Delhi'})
```

In [52]:
```
df_s.head()
```

Out[52]:

	ID	Airline	Total_Stops	Price	Total_Duration_Hours	Source	Destination
0	-1.732302	-1.011394	-1.219149	-1.390365	-1.034602	Banglore	Delhi
1	-1.731971	-0.005191	1.882081	-0.047202	0.034476	Kolkata	Banglore
2	-1.731640	1.001012	1.882081	1.133554	1.079594	Delhi	Cochin
3	-1.731308	-1.011394	0.331466	-0.462081	-0.314651	Kolkata	Banglore
4	-1.730977	-1.011394	0.331466	1.048762	-0.460564	Banglore	Delhi

In [53]:
```
df_s.Month_of_Journey.value_counts()
```

Out[53]:
```
5    3371
6    3282
3    2544
4    1076
Name: Month_of_Journey, dtype: int64
```

In [54]:
```
df_s.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10273 entries, 0 to 10272
Data columns (total 11 columns):
 #   Column                 Non-Null Count  Dtype  
---  ------                 --------------  -----  
 0   ID                     10273 non-null  float64
 1   Airline                10273 non-null  float64
 2   Total_Stops            10273 non-null  float64
 3   Price                  10273 non-null  float64
 4   Total_Duration_Hours   10273 non-null  float64
 5   Source                 10273 non-null  object 
 6   Destination            10273 non-null  object 
 7   Dep_Time_Category      10273 non-null  object 
 8   Arrival_Time_Category  10273 non-null  object 
 9   Weekday_of_Journey     10273 non-null  object 
 10  Month_of_Journey       10273 non-null  object 
dtypes: float64(5), object(6)
memory usage: 883.0+ KB
```

In [55]:
```
# Identify categorical variables

categorical_vars = df_s.select_dtypes(include='object').columns

# One-hot encode categorical variables using get_dummies
encoded_categorical = pd.get_dummies(df_s[categorical_vars], prefix=categor

# Select numeric variables
numerical_data = df_s.select_dtypes(exclude='object')

# Combine numerical and encoded categorical data
df3 = pd.concat([numerical_data, encoded_categorical], axis=1)

df3.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10273 entries, 0 to 10272
Data columns (total 32 columns):
 #   Column                                 Non-Null Count  Dtype  
---  ------                                 --------------  -----  
 0   ID                                     10273 non-null  float64
 1   Airline                                10273 non-null  float64
 2   Total_Stops                            10273 non-null  float64
 3   Price                                  10273 non-null  float64
 4   Total_Duration_Hours                   10273 non-null  float64
 5   Source_Chennai                         10273 non-null  uint8  
 6   Source_Delhi                           10273 non-null  uint8  
 7   Source_Kolkata                         10273 non-null  uint8  
 8   Source_Mumbai                          10273 non-null  uint8  
 9   Destination_Cochin                     10273 non-null  uint8  
 10  Destination_Delhi                      10273 non-null  uint8  
 11  Destination_Hyderabad                  10273 non-null  uint8  
 12  Destination_Kolkata                    10273 non-null  uint8  
 13  Dep_Time_Category_Early_Morning        10273 non-null  uint8  
 14  Dep_Time_Category_Evening              10273 non-null  uint8  
 15  Dep_Time_Category_Late_Night           10273 non-null  uint8  
 16  Dep_Time_Category_Morning              10273 non-null  uint8  
 17  Dep_Time_Category_Night                10273 non-null  uint8  
 18  Arrival_Time_Category_Early_Morning    10273 non-null  uint8  
 19  Arrival_Time_Category_Evening          10273 non-null  uint8  
 20  Arrival_Time_Category_Late_Night       10273 non-null  uint8  
 21  Arrival_Time_Category_Morning          10273 non-null  uint8  
 22  Arrival_Time_Category_Night            10273 non-null  uint8  
 23  Weekday_of_Journey_1                   10273 non-null  uint8  
 24  Weekday_of_Journey_2                   10273 non-null  uint8  
 25  Weekday_of_Journey_3                   10273 non-null  uint8  
 26  Weekday_of_Journey_4                   10273 non-null  uint8  
 27  Weekday_of_Journey_5                   10273 non-null  uint8  
 28  Weekday_of_Journey_6                   10273 non-null  uint8  
 29  Month_of_Journey_4                     10273 non-null  uint8  
 30  Month_of_Journey_5                     10273 non-null  uint8  
 31  Month_of_Journey_6                     10273 non-null  uint8  
dtypes: float64(5), uint8(27)
memory usage: 672.3 KB
```

In [56]:
```
df3.head()
```

Out[56]:

ıta	Source_Mumbai	Destination_Cochin	...	Arrival_Time_Category_Night	Weekday_of_Journey_
0	0	0	...	0	
1	0	0	...	0	
0	0	1	...	0	
1	0	0	...	1	
0	0	0	...	1	

In [57]:
```
# no variable with near zero variance

nzv = df3.apply(lambda x: x.nunique() <= 1)
nzv
#d = d.loc[:, ~nzv]
```

Out[57]:

```
ID                                    False
Airline                               False
Total_Stops                           False
Price                                 False
Total_Duration_Hours                  False
Source_Chennai                        False
Source_Delhi                          False
Source_Kolkata                        False
Source_Mumbai                         False
Destination_Cochin                    False
Destination_Delhi                     False
Destination_Hyderabad                 False
Destination_Kolkata                   False
Dep_Time_Category_Early_Morning       False
Dep_Time_Category_Evening             False
Dep_Time_Category_Late_Night          False
Dep_Time_Category_Morning             False
Dep_Time_Category_Night               False
Arrival_Time_Category_Early_Morning   False
Arrival_Time_Category_Evening         False
Arrival_Time_Category_Late_Night      False
Arrival_Time_Category_Morning         False
Arrival_Time_Category_Night           False
Weekday_of_Journey_1                  False
Weekday_of_Journey_2                  False
Weekday_of_Journey_3                  False
Weekday_of_Journey_4                  False
Weekday_of_Journey_5                  False
Weekday_of_Journey_6                  False
Month_of_Journey_4                    False
Month_of_Journey_5                    False
Month_of_Journey_6                    False
dtype: bool
```

In [58]:
```
target = ['Price']
t1 = df3[target]
t1.head()
```

Out[58]:

	Price
0	-1.390365
1	-0.047202
2	1.133554
3	-0.462081
4	1.048762

In [59]:
```
df3.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10273 entries, 0 to 10272
Data columns (total 32 columns):
 #   Column                                 Non-Null Count  Dtype  
---  ------                                 --------------  -----  
 0   ID                                     10273 non-null  float64
 1   Airline                                10273 non-null  float64
 2   Total_Stops                            10273 non-null  float64
 3   Price                                  10273 non-null  float64
 4   Total_Duration_Hours                   10273 non-null  float64
 5   Source_Chennai                         10273 non-null  uint8  
 6   Source_Delhi                           10273 non-null  uint8  
 7   Source_Kolkata                         10273 non-null  uint8  
 8   Source_Mumbai                          10273 non-null  uint8  
 9   Destination_Cochin                     10273 non-null  uint8  
 10  Destination_Delhi                      10273 non-null  uint8  
 11  Destination_Hyderabad                  10273 non-null  uint8  
 12  Destination_Kolkata                    10273 non-null  uint8  
 13  Dep_Time_Category_Early_Morning        10273 non-null  uint8  
 14  Dep_Time_Category_Evening              10273 non-null  uint8  
 15  Dep_Time_Category_Late_Night           10273 non-null  uint8  
 16  Dep_Time_Category_Morning              10273 non-null  uint8  
 17  Dep_Time_Category_Night                10273 non-null  uint8  
 18  Arrival_Time_Category_Early_Morning    10273 non-null  uint8  
 19  Arrival_Time_Category_Evening          10273 non-null  uint8  
 20  Arrival_Time_Category_Late_Night       10273 non-null  uint8  
 21  Arrival_Time_Category_Morning          10273 non-null  uint8  
 22  Arrival_Time_Category_Night            10273 non-null  uint8  
 23  Weekday_of_Journey_1                   10273 non-null  uint8  
 24  Weekday_of_Journey_2                   10273 non-null  uint8  
 25  Weekday_of_Journey_3                   10273 non-null  uint8  
 26  Weekday_of_Journey_4                   10273 non-null  uint8  
 27  Weekday_of_Journey_5                   10273 non-null  uint8  
 28  Weekday_of_Journey_6                   10273 non-null  uint8  
 29  Month_of_Journey_4                     10273 non-null  uint8  
 30  Month_of_Journey_5                     10273 non-null  uint8  
 31  Month_of_Journey_6                     10273 non-null  uint8  
dtypes: float64(5), uint8(27)
memory usage: 672.3 KB
```

In [60]:
```
p1 = df3.drop(df3.columns[3], axis=1)
p1.head()
```

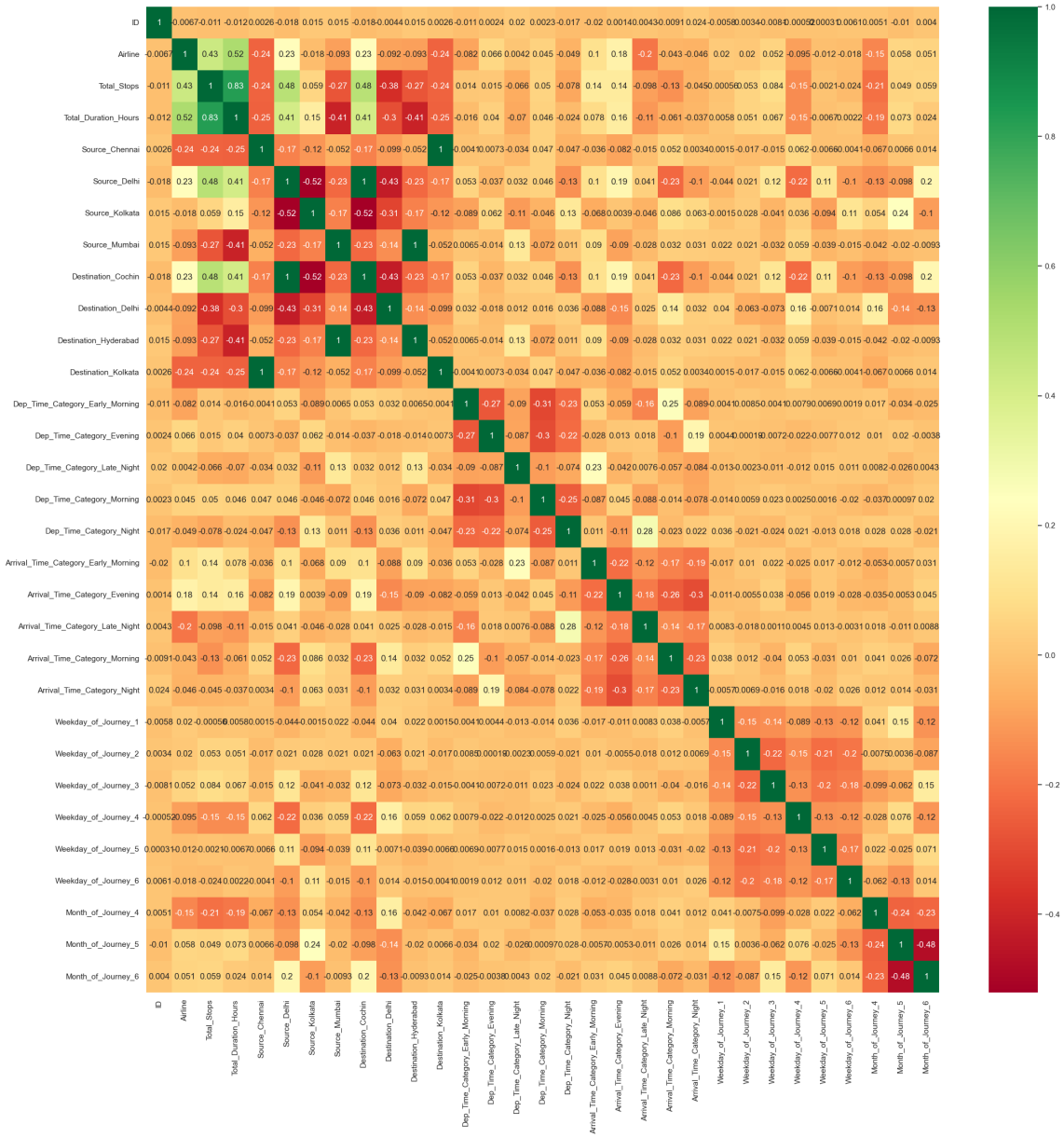
Out[60]:

	ID	Airline	Total_Stops	Total_Duration_Hours	Source_Chennai	Source_Delhi	S
0	-1.732302	-1.011394	-1.219149	-1.034602	0	0	
1	-1.731971	-0.005191	1.882081	0.034476	0	0	
2	-1.731640	1.001012	1.882081	1.079594	0	1	
3	-1.731308	-1.011394	0.331466	-0.314651	0	0	
4	-1.730977	-1.011394	0.331466	-0.460564	0	0	

5 rows × 31 columns

In [61]:
```
from sklearn.decomposition import PCA  # to apply PCA
import seaborn as sns
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
```

In [62]:
```
# Finds correlation between Independent and dependent attributes

plt.figure(figsize = (25,25))
sns.heatmap(p1.corr(), annot = True, cmap = "RdYlGn")

plt.show()
```



In [63]:
```
all_independent_vars = p1.columns.difference(['Destination_Cochin', 'Destin

# Select independent variables excluding those to be excluded
X = p1[all_independent_vars]


threshold = 0.8

# Absolute value correlation matrix
corr_matrix = X.corr().abs()
corr_matrix.head()

# Upper triangle of correlations
upper = corr_matrix.where(np.triu(np.ones(corr_matrix.shape), k=1).astype(b
upper.head()

# Select columns with correlations above threshold
to_drop = [column for column in upper.columns if any(upper[column] > thresh

print('There are %d columns to remove :' % (len(to_drop)))
to_drop
```

```
There are 1 columns to remove :
```

Out[63]:
```
['Total_Stops']
```

In [64]:
```
corr_matrix
```

Out[64]:

ning	Dep_Time_Category_Late_Night	Dep_Time_Category_Morning	...	Source_Kolkata	Source
6122	0.004210	0.045295	...	0.017516	
8356	0.230057	0.087384	...	0.067563	
3477	0.041732	0.044801	...	0.003873	
8230	0.007637	0.088329	...	0.046360	
9623	0.057479	0.013916	...	0.086228	
4845	0.084265	0.077787	...	0.062951	
6866	0.089912	0.308526	...	0.088536	
0000	0.087139	0.299013	...	0.062048	
7139	1.000000	0.100742	...	0.105938	
9013	0.100742	1.000000	...	0.045733	
0049	0.074138	0.254400	...	0.130167	
8118	0.011797	0.016463	...	0.313308	
2391	0.019919	0.002319	...	0.014716	
0143	0.008233	0.037404	...	0.054319	
9647	0.026336	0.000965	...	0.239973	
3833	0.004255	0.020019	...	0.102979	
7277	0.033627	0.047307	...	0.121340	
6971	0.032431	0.046171	...	0.521032	
2048	0.105938	0.045733	...	1.000000	
4443	0.132202	0.071599	...	0.165388	
0286	0.069723	0.046473	...	0.150737	
4665	0.065621	0.049790	...	0.059157	
4371	0.012823	0.014050	...	0.001472	
0195	0.002290	0.005945	...	0.028441	
7188	0.010878	0.023498	...	0.040953	
2217	0.012336	0.002459	...	0.035585	
7711	0.014657	0.001636	...	0.093991	
1568	0.010858	0.019703	...	0.110544	

In [65]:
```
drop = ['Destination_Cochin', 'Destination_Hyderabad', 'Destination_Kolkata

p2 = p1.drop(columns=drop)
```

In [66]:
```
p2.isnull().sum()
```

Out[66]:
```
Airline                                0
Total_Stops                            0
Total_Duration_Hours                   0
Source_Chennai                         0
Source_Kolkata                         0
Source_Mumbai                          0
Destination_Delhi                      0
Dep_Time_Category_Early_Morning        0
Dep_Time_Category_Evening              0
Dep_Time_Category_Late_Night           0
Dep_Time_Category_Morning              0
Dep_Time_Category_Night                0
Arrival_Time_Category_Early_Morning    0
Arrival_Time_Category_Evening          0
Arrival_Time_Category_Late_Night       0
Arrival_Time_Category_Morning          0
Arrival_Time_Category_Night            0
Weekday_of_Journey_1                   0
Weekday_of_Journey_2                   0
Weekday_of_Journey_3                   0
Weekday_of_Journey_4                   0
Weekday_of_Journey_5                   0
Weekday_of_Journey_6                   0
Month_of_Journey_4                     0
Month_of_Journey_5                     0
Month_of_Journey_6                     0
dtype: int64
```

In [67]:
```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(p2, t1, test_size=0.3,
```

In [68]:
```
## Linear reg
#GetParams
from sklearn.linear_model import LinearRegression
estimator = LinearRegression()
estimator.get_params()

#GridSearchCV
from sklearn.model_selection import GridSearchCV
copy_X=[True, False]
fit_intercept=[True,False]
n_jobs=[None,-1,-2]
positive=[False,True]
param_grid = dict(copy_X=copy_X, fit_intercept=fit_intercept, n_jobs=n_jobs
```

In [69]:
```
# Training the Multiple Linear Regression model on the Training set
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_train, y_train)
```

Out[69]:
```
LinearRegression
LinearRegression()
```

In [70]:
```
y_pred = regressor.predict(X_test)

from sklearn.metrics import r2_score
r2 = r2_score(y_test, y_pred)
print('R2 score is', r2)
```

```
R2 score is 0.7320217084569262
```

In [71]:
```
y_test
```

Out[71]:

	Price
5266	1.249194
3043	0.184553
334	0.376425
9418	1.425446
2869	0.127438
...	...
2159	-1.775664
585	-0.485867
4907	-1.419121
2481	1.007564
1045	-0.740102

3082 rows × 1 columns

In [72]:
```
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split
```

In [73]:
```
#GetParams
from sklearn.ensemble import RandomForestRegressor
estimator = RandomForestRegressor()
estimator.get_params()
```

Out[73]:
```
{'bootstrap': True,
 'ccp_alpha': 0.0,
 'criterion': 'squared_error',
 'max_depth': None,
 'max_features': 1.0,
 'max_leaf_nodes': None,
 'max_samples': None,
 'min_impurity_decrease': 0.0,
 'min_samples_leaf': 1,
 'min_samples_split': 2,
 'min_weight_fraction_leaf': 0.0,
 'n_estimators': 100,
 'n_jobs': None,
 'oob_score': False,
 'random_state': None,
 'verbose': 0,
 'warm_start': False}
```

In [74]:
```
# Training the Random Forest Regression model on the whole dataset
from sklearn.ensemble import RandomForestRegressor
regressor = RandomForestRegressor(random_state = 0)
regressor.fit(X_train, y_train)
```

Out[74]:
```
RandomForestRegressor
RandomForestRegressor(random_state=0)
```

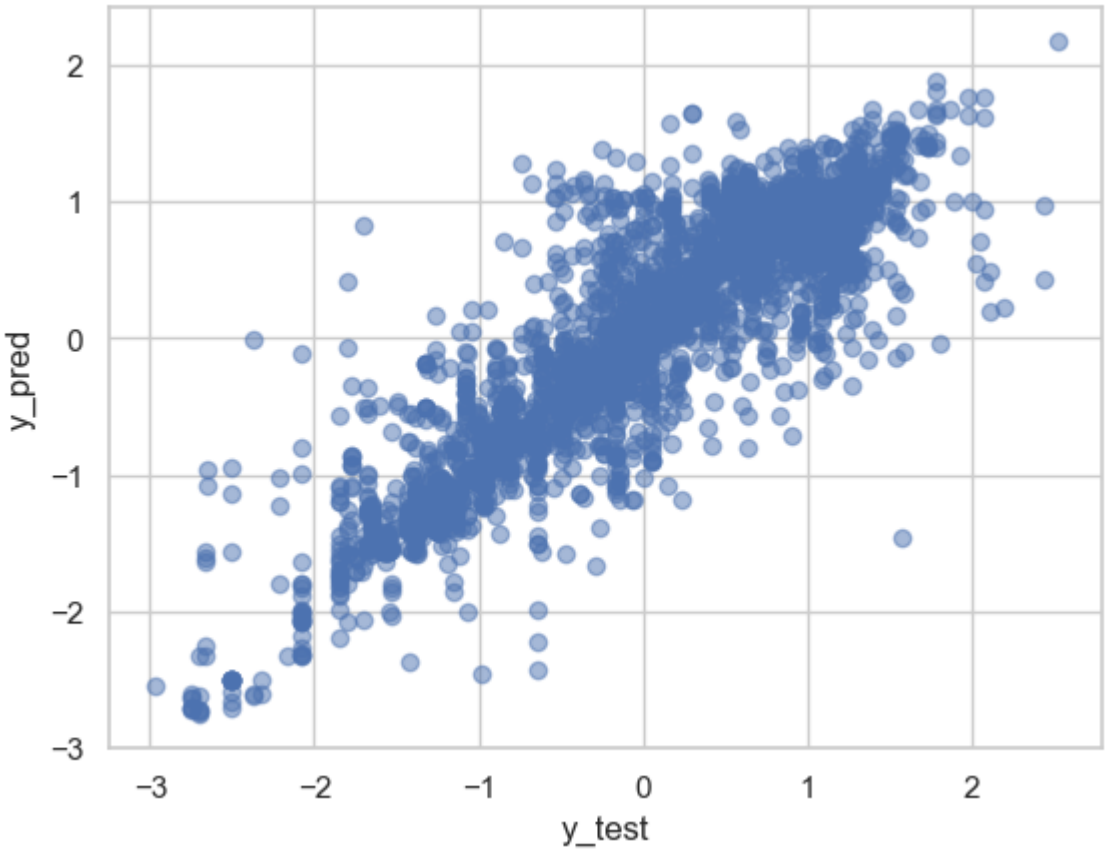
In [75]:
```
# Predicting the Test set results
y_pred = regressor.predict(X_test)
```

In [76]:
```
# Evaluating the Model Performance
from sklearn.metrics import r2_score
r2_score(y_test, y_pred)
```

Out[76]:
```
0.7876107785095255
```

In [77]:
```
plt.scatter(y_test, y_pred, alpha = 0.5)
plt.xlabel("y_test")
plt.ylabel("y_pred")
plt.show
```

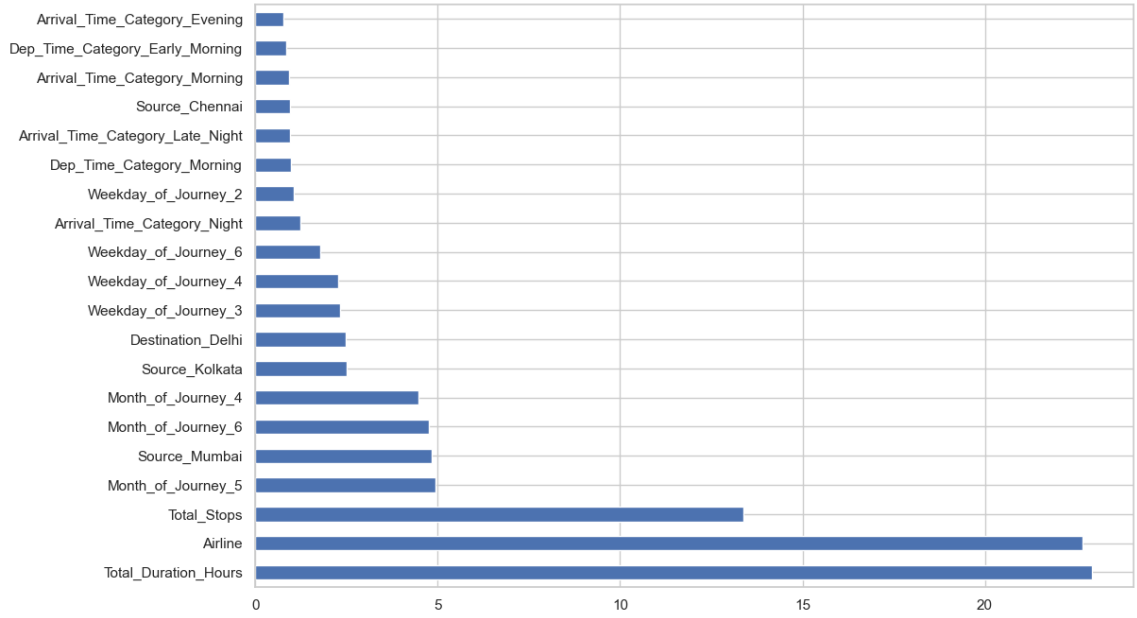
Out[77]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```



In [78]:
```
from catboost import CatBoostRegressor

# Create a CatBoostRegressor model
model = CatBoostRegressor()

# Train the model
model.fit(X_train, y_train)

# Feature importances
feat_importances = pd.Series(model.get_feature_importance(), index=X_train.

# Plot the top 20 features
plt.figure(figsize=(12, 8))
feat_importances.nlargest(20).plot(kind='barh')
plt.show()
```

```
152:    learn: 0.4338940        total: 608ms    remaining: 3.36s
153:    learn: 0.4335734        total: 611ms    remaining: 3.35s
154:    learn: 0.4332219        total: 614ms    remaining: 3.35s
155:    learn: 0.4328167        total: 617ms    remaining: 3.34s
156:    learn: 0.4327498        total: 620ms    remaining: 3.33s
157:    learn: 0.4323814        total: 623ms    remaining: 3.32s
158:    learn: 0.4321846        total: 626ms    remaining: 3.31s
159:    learn: 0.4320277        total: 629ms    remaining: 3.3s
160:    learn: 0.4319352        total: 632ms    remaining: 3.29s
161:    learn: 0.4316187        total: 635ms    remaining: 3.28s
162:    learn: 0.4314299        total: 638ms    remaining: 3.27s
163:    learn: 0.4312192        total: 641ms    remaining: 3.27s
164:    learn: 0.4309020        total: 644ms    remaining: 3.26s
165:    learn: 0.4307279        total: 647ms    remaining: 3.25s
166:    learn: 0.4303259        total: 650ms    remaining: 3.24s
167:    learn: 0.4299760        total: 653ms    remaining: 3.23s
168:    learn: 0.4297789        total: 655ms    remaining: 3.22s
169:    learn: 0.4294994        total: 659ms    remaining: 3.21s
170:    learn: 0.4293463        total: 661ms    remaining: 3.21s
171:    learn: 0.4290270        total: 665ms    remaining: 3.2s
```

In [79]:
```
# Plot the top 20 features
plt.figure(figsize=(12, 8))
feat_importances.nlargest(20).plot(kind='barh')
plt.show()
```



In [80]:
```
y_pred = model.predict(X_test)
r2 = r2_score(y_test, y_pred)
print("R2 Score on Test Data:", r2)
```

```
R2 Score on Test Data: 0.8349613167542072
```