In [114]:
```
import pandas as pd
import warnings
warnings.filterwarnings('ignore')
import sqlite3
```

In [115]:
```
import csv

file_path =r'D:\Masters\SEM 1\Python\Data_Train.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 rows
    print(row)
```

```
{'ID': '1', 'Airline': 'IndiGo', 'Date_of_Journey': '24-03-2019', 'Source': 'Banglore', 'Desti
nation': 'New Delhi', 'Route': 'BLR → DEL', 'Dep_Time': '22:20', 'Arrival_Time': '22-03-2023 0
1:10', 'Duration': '2h 50m', 'Total_Stops': 'non-stop', 'Additional_Info': 'No info', 'Price':
'3897'}
{'ID': '2', 'Airline': 'Air India', 'Date_of_Journey': '01-05-2019', 'Source': 'Kolkata', 'Des
tination': 'Banglore', 'Route': 'CCU → IXR → BBI → BLR', 'Dep_Time': '05:50', 'Arrival_Time':
'13:15', 'Duration': '7h 25m', 'Total_Stops': '2 stops', 'Additional_Info': 'No info', 'Pric
e': '7662'}
{'ID': '3', 'Airline': 'Jet Airways', 'Date_of_Journey': '09-06-2019', 'Source': 'Delhi', 'Des
tination': 'Cochin', 'Route': 'DEL → LKO → BOM → COK', 'Dep_Time': '09:25', 'Arrival_Time': '1
0-06-2023 04:25', 'Duration': '19h', 'Total_Stops': '2 stops', 'Additional_Info': 'No info',
'Price': '13882'}
{'ID': '4', 'Airline': 'IndiGo', 'Date_of_Journey': '12-05-2019', 'Source': 'Kolkata', 'Destin
ation': 'Banglore', 'Route': 'CCU → NAG → BLR', 'Dep_Time': '18:05', 'Arrival_Time': '23:30',
'Duration': '5h 25m', 'Total_Stops': '1 stop', 'Additional_Info': 'No info', 'Price': '6218'}
{'ID': '5', 'Airline': 'IndiGo', 'Date_of_Journey': '01-03-2019', 'Source': 'Banglore', 'Desti
nation': 'New Delhi', 'Route': 'BLR → NAG → DEL', 'Dep_Time': '16:50', 'Arrival_Time': '21:3
5', 'Duration': '4h 45m', 'Total_Stops': '1 stop', 'Additional_Info': 'No info', 'Price': '133
02'}
{'ID': '6', 'Airline': 'SpiceJet', 'Date_of_Journey': '24-06-2019', 'Source': 'Kolkata', 'Dest
ination': 'Banglore', 'Route': 'CCU → BLR', 'Dep_Time': '09:00', 'Arrival_Time': '11:25', 'Dur
ation': '2h 25m', 'Total_Stops': 'non-stop', 'Additional_Info': 'No info', 'Price': '3873'}
{'ID': '7', 'Airline': 'Jet Airways', 'Date_of_Journey': '12-03-2019', 'Source': '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-fli
ght meal not included', 'Price': '11087'}
{'ID': '8', 'Airline': 'Jet Airways', 'Date_of_Journey': '01-03-2019', 'Source': '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 inf
o', 'Price': '22270'}
{'ID': '9', 'Airline': 'Jet Airways', 'Date_of_Journey': '12-03-2019', 'Source': '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-fli
ght meal not included', 'Price': '11087'}
{'ID': '10', 'Airline': 'Multiple carriers', 'Date_of_Journey': '27-05-2019', 'Source': 'Delh
i', 'Destination': 'Cochin', 'Route': 'DEL → BOM → COK', 'Dep_Time': '11:25', 'Arrival_Time':
'19:15', 'Duration': '7h 50m', 'Total_Stops': '1 stop', 'Additional_Info': 'No info', 'Price':
'8625'}
```

In [116]:

```
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, Total_Stops) 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['Airline'], fd['Source'], fd['Destination'], fd['Route'],
        insert_flight(conn_normalized, insert_tuple)
```

In [117]:
```
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_Time, Duration) 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['Date_of_Journey'], fd['Dep_Time'], fd['Arrival_Time'], fd
        insert_schedule(conn_normalized, insert_tuple)
```

In [118]:
```
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 [119]:
```
import pandas as pd

conn_normalized = create_connection('normalized3.db')

sql_statement = """SELECT Flight.ID, Airline, Source, Destination, Route, Total_Stops,
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', 'Route', 'Total_Stop

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 [120]:
```
df.head()
```

Out[120]:

	ID	Airline	Source	Destination	Route	Total_Stops	Date_of_Journey	Dep_Time	Arrival_Time	Duration	Additio
0	1	IndiGo	Banglore	New Delhi	BLR → DEL	non-stop	24-03-2019	22:20	22-03-2023 01:10	2h 50m	
1	2	Air India	Kolkata	Banglore	CCU → IXR → BBI → BLR	2 stops	01-05-2019	05:50	13:15	7h 25m	
2	3	Jet Airways	Delhi	Cochin	DEL → LKO → BOM → COK	2 stops	09-06-2019	09:25	10-06-2023 04:25	19h	
3	4	IndiGo	Kolkata	Banglore	CCU → NAG → BLR	1 stop	12-05-2019	18:05	23:30	5h 25m	
4	5	IndiGo	Banglore	New Delhi	BLR → NAG → DEL	1 stop	01-03-2019	16:50	21:35	4h 45m	

In [121]:
```
#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 [122]:
```
#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 [123]:
```
import pandas as pd

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

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

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 [124]:
```
df1.head()
```

Out[124]:

	ID	Airline	Source	Destination	Route	Total_Stops	Date_of_Journey	Dep_Time	Arrival_Time	Duration	Additio
0	1	IndiGo	Banglore	New Delhi	BLR → DEL	non-stop	24-03-2019	22:20	22-03-2023 01:10	2h 50m	
1	2	Air India	Kolkata	Banglore	CCU → IXR → BBI → BLR	2 stops	01-05-2019	05:50	13:15	7h 25m	
2	3	Jet Airways	Delhi	Cochin	DEL → LKO → BOM → COK	2 stops	09-06-2019	09:25	10-06-2023 04:25	19h	
3	4	IndiGo	Kolkata	Banglore	CCU → NAG → BLR	1 stop	12-05-2019	18:05	23:30	5h 25m	
4	5	IndiGo	Banglore	New Delhi	BLR → NAG → DEL	1 stop	01-03-2019	16:50	21:35	4h 45m	

In [125]:
```
df1.Arrival_Time.value_counts()
```

Out[125]:
```
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 [126]:
```
bins = [0, 4, 8, 12, 16,20,24]
labels = ['Late_Night', 'Early_Morning','Morning', 'Afternoon', 'Evening','Night']

df1['Arrival_Time_Category'] = pd.cut(pd.to_datetime(df1['Arrival_Time']).dt.hour, bins=bins, l
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 [127]:
```
df1.head()
```

Out[127]:

	ID	Airline	Source	Destination	Route	Total_Stops	Date_of_Journey	Dep_Time	Arrival_Time	Duration	Additio
0	1	IndiGo	Banglore	New Delhi	BLR → DEL	non-stop	24-03-2019	22:20	22-03-2023 01:10	2h 50m	
1	2	Air India	Kolkata	Banglore	CCU → IXR → BBI → BLR	2 stops	01-05-2019	05:50	13:15	7h 25m	
2	3	Jet Airways	Delhi	Cochin	DEL → LKO → BOM → COK	2 stops	09-06-2019	09:25	10-06-2023 04:25	19h	
3	4	IndiGo	Kolkata	Banglore	CCU → NAG → BLR	1 stop	12-05-2019	18:05	23:30	5h 25m	
4	5	IndiGo	Banglore	New Delhi	BLR → NAG → DEL	1 stop	01-03-2019	16:50	21:35	4h 45m	

In [128]:
```
df1.Total_Stops.value_counts()
```

Out[128]:
```
1 stop      5625
non-stop    3475
2 stops     1318
3 stops       43
                1
4 stops         1
Name: Total_Stops, dtype: int64
```

In [129]:
```
#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[129]:
```
1.0    5625
0.0    3475
2.0    1318
3.0      43
4.0       1
Name: Total_Stops, dtype: int64
```

In [130]:
```
df1.head()
```

Out[130]:

	ID	Airline	Source	Destination	Route	Total_Stops	Date_of_Journey	Dep_Time	Arrival_Time	Duration	Additio
0	1	IndiGo	Banglore	New Delhi	BLR → DEL	0.0	24-03-2019	22:20	22-03-2023 01:10	2h 50m	
1	2	Air India	Kolkata	Banglore	CCU → IXR → BBI → BLR	2.0	01-05-2019	05:50	13:15	7h 25m	
2	3	Jet Airways	Delhi	Cochin	DEL → LKO → BOM → COK	2.0	09-06-2019	09:25	10-06-2023 04:25	19h	
3	4	IndiGo	Kolkata	Banglore	CCU → NAG → BLR	1.0	12-05-2019	18:05	23:30	5h 25m	
4	5	IndiGo	Banglore	New Delhi	BLR → NAG → DEL	1.0	01-03-2019	16:50	21:35	4h 45m	

In [131]:
```
drop = ['Arrival_Time', 'Dep_Time','Route']

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

In [132]:
```
df2.head()
```

Out[132]:

	ID	Airline	Source	Destination	Total_Stops	Date_of_Journey	Duration	Additional_Info	Price	Dep_Time_Categ
0	1	IndiGo	Banglore	New Delhi	0.0	24-03-2019	2h 50m	No info	3897	N
1	2	Air India	Kolkata	Banglore	2.0	01-05-2019	7h 25m	No info	7662	Early_Morr
2	3	Jet Airways	Delhi	Cochin	2.0	09-06-2019	19h	No info	13882	Morr
3	4	IndiGo	Kolkata	Banglore	1.0	12-05-2019	5h 25m	No info	6218	Ever
4	5	IndiGo	Banglore	New Delhi	1.0	01-03-2019	4h 45m	No info	13302	Ever

In [133]:
```
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 [134]:
```
df2.head()
```

Out[134]:

	ID	Airline	Source	Destination	Total_Stops	Date_of_Journey	Duration	Additional_Info	Price	Dep_Time_Categ
0	1	IndiGo	Banglore	New Delhi	0.0	24-03-2019	0 days 02:50:00	No info	3897	N
1	2	Air India	Kolkata	Banglore	2.0	01-05-2019	0 days 07:25:00	No info	7662	Early_Morr
2	3	Jet Airways	Delhi	Cochin	2.0	09-06-2019	0 days 19:00:00	No info	13882	Morr
3	4	IndiGo	Kolkata	Banglore	1.0	12-05-2019	0 days 05:25:00	No info	6218	Ever
4	5	IndiGo	Banglore	New Delhi	1.0	01-03-2019	0 days 04:45:00	No info	13302	Ever

In [135]:
```
df2['Date_of_Journey'] = pd.to_datetime(df2['Date_of_Journey'], format='%d-%m-%Y')
df2['Weekday_of_Journey'] = df2['Date_of_Journey'].dt.weekday.astype('object')
```

In [136]:
```
df2['Month_of_Journey'] = df2['Date_of_Journey'].dt.month.astype('object')
```

In [137]:
```
df2.Month_of_Journey.value_counts()
```

Out[137]:
```
5    3396
6    3311
3    2678
4    1078
Name: Month_of_Journey, dtype: int64
```

In [138]:
```
drop = ['Duration', 'Date_of_Journey']

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

In [139]:
```
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 [140]:
```
df3.Additional_Info.value_counts()
```

Out[140]:
```
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 [141]:
```
# dropping this column as there is no info in it

drop = ['Additional_Info']

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

In [142]:
```
df4.head()
```

Out[142]:

	ID	Airline	Source	Destination	Total_Stops	Price	Dep_Time_Category	Arrival_Time_Category	Total_Duration_H
0	1	IndiGo	Banglore	New Delhi	0.0	3897	Night	Late_Night	2.83
1	2	Air India	Kolkata	Banglore	2.0	7662	Early_Morning	Afternoon	7.41
2	3	Jet Airways	Delhi	Cochin	2.0	13882	Morning	Early_Morning	19.00
3	4	IndiGo	Kolkata	Banglore	1.0	6218	Evening	Night	5.41
4	5	IndiGo	Banglore	New Delhi	1.0	13302	Evening	Night	4.75

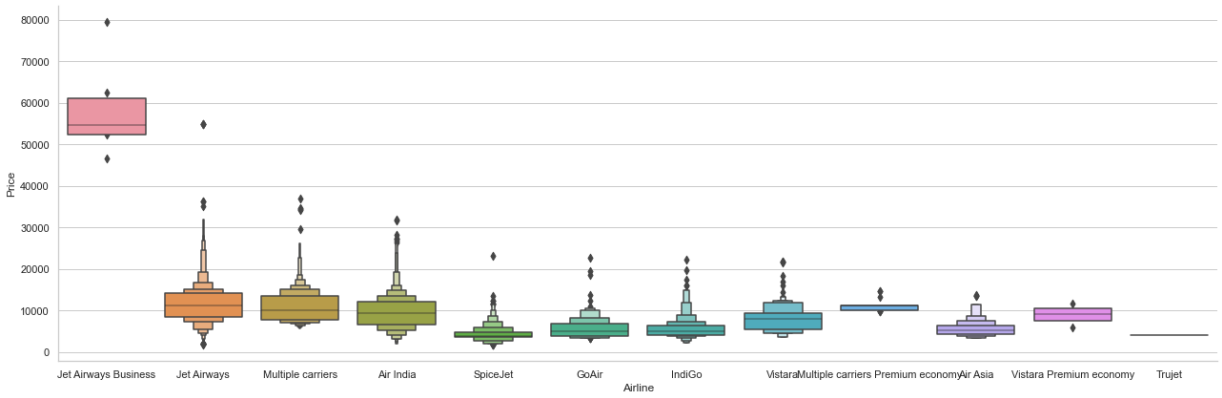
In [143]:
```
df4.info()
```

```
<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
```

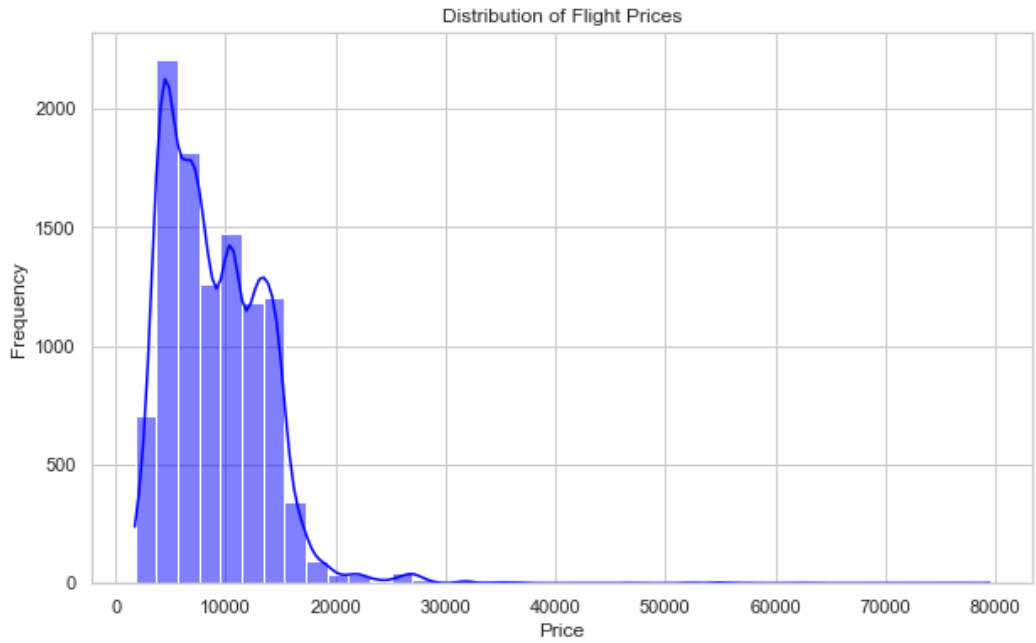
In [144]:
```
df4.dropna(inplace=True)
```

In [145]:
```
df4['Dep_Time_Category'] = df4['Dep_Time_Category'].astype('object')
df4['Arrival_Time_Category'] = df4['Arrival_Time_Category'].astype('object')
```

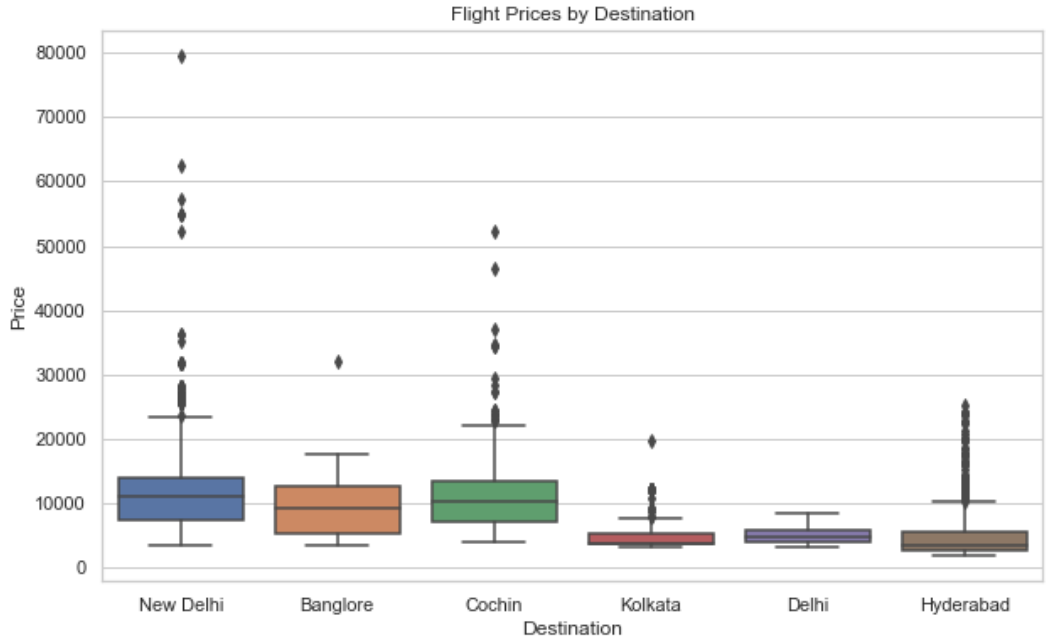
In [146]:
```
# From graph we can see that Jet Airways Business have the highest Price.
# Apart from the first Airline almost all are having similar median
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

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



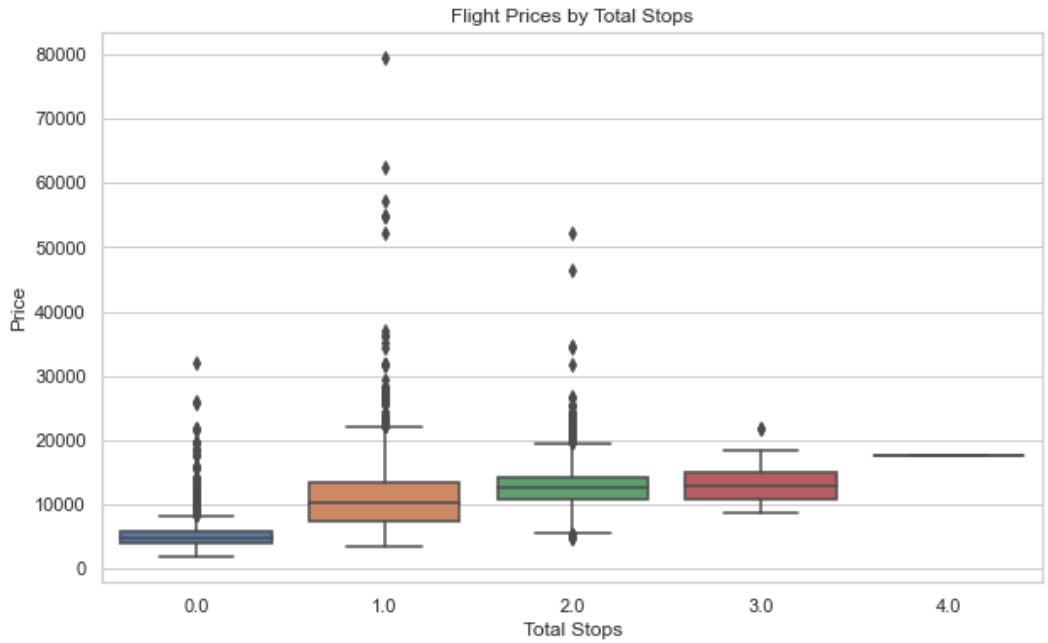
In [147]:
```
import matplotlib.pyplot as plt
import seaborn as sns

# 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 [148]:
```
# 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 [149]:
```
# 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 [150]:
```
## 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 [151]:
```
# 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].skew()) > 0.5]

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

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

In [152]:
```
# 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 [153]:
```
df_log = df1.copy()
df_log[skewed_vars] = np.log(df_log[skewed_vars])
df_log.head()
```

Out[153]:

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

In [154]:
```
airlines = df_log.groupby(["Airline"])["Price"].mean().sort_values().index
airlines
```

Out[154]:
```
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 [155]:
```
dict_airlines = {key:index for index , key in enumerate(airlines , 0)}
dict_airlines
```

Out[155]:
```
{'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 [156]:
```
df_log['Airline'] = df_log['Airline'].map(dict_airlines)
df_log.head()
```

Out[156]:

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

In [157]:
```
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), columns=numeric_co

# 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[157]:

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

In [158]:
```
df_s.Month_of_Journey.value_counts()
```

Out[158]:
```
5    3371
6    3282
3    2544
4    1076
Name: Month_of_Journey, dtype: int64
```

In [159]:
```
table = pd.crosstab(df_s['Source'], df_s['Destination'], margins=True, margins_name='Total')
table
```

Out[159]:

Destination	Banglore	Cochin	Delhi	Hyderabad	Kolkata	New Delhi	Total
Source							
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 [160]:
```
df_s['Destination'] = df_s['Destination'].replace({'New Delhi': 'Delhi'})
```

In [161]:
```
df_s.head()
```

Out[161]:

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

In [162]:
```
df_s.Month_of_Journey.value_counts()
```

Out[162]:
```
5    3371
6    3282
3    2544
4    1076
Name: Month_of_Journey, dtype: int64
```

In [163]:
```
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 [164]:
```
# 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=categorical_vars, drop_firs

# 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 [165]:
```
df3.head()
```

Out[165]:

	ID	Airline	Total_Stops	Price	Total_Duration_Hours	Source_Chennai	Source_Delhi	Source_Kolkata
0	-1.732302	-1.011394	-1.219149	-1.390365	-1.034602	0	0	0
1	-1.731971	-0.005191	1.882081	-0.047202	0.034476	0	0	1
2	-1.731640	1.001012	1.882081	1.133554	1.079594	0	1	0
3	-1.731308	-1.011394	0.331466	-0.462081	-0.314651	0	0	1
4	-1.730977	-1.011394	0.331466	1.048762	-0.460564	0	0	0

5 rows × 32 columns

In [166]:
```
# no variable with near zero variance

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

Out[166]:
```
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 [167]:
```
target = ['Price']
t1 = df3[target]
t1.head()
```

Out[167]:

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

In [168]:
```
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 [169]:
```
p1 = df3.drop(df3.columns[3], axis=1)
p1.head()
```

Out[169]:

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

5 rows × 31 columns

In [170]:
```
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 [171]:
```
# Finds correlation between Independent and dependent attributes

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

plt.show()
```

In [172]:
```
all_independent_vars = p1.columns.difference(['Destination_Cochin', 'Destination_Hyderabad', 'D

# 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(bool))
upper.head()

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

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

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

Out[172]:
```
['Total_Stops']
```

In [173]:
```
corr_matrix
```

Out[173]:

	Airline	Arrival_Time_Category_Early_Morning	Arrival_Time_Category_Evening
Airline	1.000000	0.102452	0.175589
Arrival_Time_Category_Early_Morning	0.102452	1.000000	0.216430
Arrival_Time_Category_Evening	0.175589	0.216430	1.000000
Arrival_Time_Category_Late_Night	0.198290	0.120386	0.184043
Arrival_Time_Category_Morning	0.043499	0.167669	0.256327
Arrival_Time_Category_Night	0.046465	0.194485	0.297323
Dep_Time_Category_Early_Morning	0.081995	0.053071	0.058802
Dep_Time_Category_Evening	0.066122	0.028356	0.013477
Dep_Time_Category_Late_Night	0.004210	0.230057	0.041732
Dep_Time_Category_Morning	0.045295	0.087384	0.044801
Dep_Time_Category_Night	0.049277	0.010613	0.109845
Destination_Delhi	0.091539	0.088121	0.146815
ID	0.006704	0.020471	0.001364
Month_of_Journey_4	0.153708	0.052505	0.034941
Month_of_Journey_5	0.057910	0.005694	0.005312
Month_of_Journey_6	0.050779	0.031027	0.045430
Source_Chennai	0.236327	0.036335	0.081902
Source_Delhi	0.228714	0.101861	0.193524
Source_Kolkata	0.017516	0.067563	0.003873
Source_Mumbai	0.093331	0.089826	0.089813
Total_Duration_Hours	0.524143	0.078367	0.155646
Total_Stops	0.432444	0.143035	0.138629
Weekday_of_Journey_1	0.020230	0.016602	0.010946
Weekday_of_Journey_2	0.019621	0.010213	0.005499
Weekday_of_Journey_3	0.052061	0.021670	0.038424
Weekday_of_Journey_4	0.095075	0.025214	0.056184
Weekday_of_Journey_5	0.012388	0.017439	0.019199
Weekday_of_Journey_6	0.018054	0.011518	0.027889

28 rows × 28 columns

In [174]:
```
drop = ['Destination_Cochin', 'Destination_Hyderabad', 'Destination_Kolkata', 'ID','Source_Delh

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

In [175]:
```
p2.isnull().sum()
```

Out[175]:
```
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 [176]:
```
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, random_state=0)
```

In [177]:
```
## 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, positive=positive)
```

In [178]:
```
# 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[178]:
```
LinearRegression
LinearRegression()
```

In [179]:
```
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 0.7320217084569263
```

In [180]:
```
y_test
```

Out[180]:

	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 [181]:
```
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split
```

In [182]:
```
#GetParams
from sklearn.ensemble import RandomForestRegressor
estimator = RandomForestRegressor()
estimator.get_params()
```

Out[182]:
```
{'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 [183]:
```
# 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[183]:
```
RandomForestRegressor
RandomForestRegressor(random_state=0)
```

In [184]:
```
# Predicting the Test set results
y_pred = regressor.predict(X_test)
```

In [185]:
```
# Evaluating the Model Performance
from sklearn.metrics import r2_score
r2_score(y_test, y_pred)
#0.7271872568702012
```

Out[185]: 0.7876107785095255

In [ ]: