In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

In [3]:

```
df = pd.read_csv('zomato.csv',encoding ='latin-1')

df.head()
```

Out[3]:

	Restaurant ID	Restaurant Name	Country Code	City	Address	Locality	Locality Verbose	Longitude	Latitude	Cuisines	...	Currency	Has Table booking	Has Online delivery	Is delivering now	Switch to order menu	Pric rang
0	6317637	Le Petit Souffle	162	Makati City	Third Floor, Century City Mall, Kalayaan Avenu...	Century City Mall, Poblacion, Makati City	Century City Mall, Poblacion, Makati City, Mak...	121.027535	14.565443	French, Japanese, Desserts	...	Botswana Pula(P)	Yes	No	No	No	
1	6304287	Izakaya Kikufuji	162	Makati City	Little Tokyo, 2277 Chino Roces Avenue, Legaspi...	Little Tokyo, Legaspi Village, Makati City	Little Tokyo, Legaspi Village, Makati City, Ma...	121.014101	14.553708	Japanese	...	Botswana Pula(P)	Yes	No	No	No	
2	6300002	Heat - Edsa Shangri-La	162	Mandaluyong City	Edsa Shangri-La, 1 Garden Way, Ortigas, Mandal...	Edsa Shangri-La, Ortigas, Mandaluyong City	Edsa Shangri-La, Ortigas, Mandaluyong City, Ma...	121.056831	14.581404	Seafood, Asian, Filipino, Indian	...	Botswana Pula(P)	Yes	No	No	No	
3	6318506	Ooma	162	Mandaluyong City	Third Floor, Mega Fashion Hall, SM Megamall,	SM Megamall, Ortigas, Mandaluyong City	SM Megamall, Ortigas, Mandaluyong City, Mandal...	121.056475	14.585318	Japanese, Sushi	...	Botswana Pula(P)	No	No	No	No	

	Restaurant ID	Restaurant Name	Country Code	City	Address	Locality	Locality Verbose	Longitude	Latitude	Cuisines	...	Currency	Has Table booking	Has Online delivery	Is delivering now	Switch to order menu	Pric rang
4	6314302	Sambo Kojin	162	Mandaluyong City	Third Floor, Mega Atrium, SM Megamall, Ortigas...	SM Megamall, Ortigas, Mandaluyong City	SM Megamall, Ortigas, Mandaluyong City, Mandal...	121.057508	14.584450	Japanese, Korean	...	Botswana Pula(P)	Yes	No	No	No	

5 rows × 21 columns

In [4]:

```
df.columns
```

Out[4]:

```
Index(['Restaurant ID', 'Restaurant Name', 'Country Code', 'City', 'Address',
       'Locality', 'Locality Verbose', 'Longitude', 'Latitude', 'Cuisines',
       'Average Cost for two', 'Currency', 'Has Table booking',
       'Has Online delivery', 'Is delivering now', 'Switch to order menu',
       'Price range', 'Aggregate rating', 'Rating color', 'Rating text',
       'Votes'],
      dtype='object')
```

In [6]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9551 entries, 0 to 9550
Data columns (total 21 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   Restaurant ID         9551 non-null   int64  
 1   Restaurant Name       9551 non-null   object 
 2   Country Code          9551 non-null   int64  
 3   City                  9551 non-null   object 
 4   Address               9551 non-null   object 
 5   Locality              9551 non-null   object 
 6   Locality Verbose      9551 non-null   object 
 7   Longitude             9551 non-null   float64
 8   Latitude              9551 non-null   float64
 9   Cuisines              9542 non-null   object 
 10  Average Cost for two  9551 non-null   int64  
 11  Currency              9551 non-null   object 
 12  Has Table booking     9551 non-null   object 
 13  Has Online delivery   9551 non-null   object 
 14  Is delivering now     9551 non-null   object 
 15  Switch to order menu  9551 non-null   object 
```

```
 15  Switch to order menu   9551 non-null   object
 16  Price range            9551 non-null   int64
 17  Aggregate rating       9551 non-null   float64
 18  Rating color           9551 non-null   object
 19  Rating text            9551 non-null   object
 20  Votes                  9551 non-null   int64
dtypes: float64(3), int64(5), object(13)
memory usage: 1.5+ MB
```

In [7]:

```
df.describe()
```

Out[7]:

	Restaurant ID	Country Code	Longitude	Latitude	Average Cost for two	Price range	Aggregate rating	Votes
count	9.551000e+03	9551.000000	9551.000000	9551.000000	9551.000000	9551.000000	9551.000000	9551.000000
mean	9.051128e+06	18.365616	64.126574	25.854381	1199.210763	1.804837	2.666370	156.909748
std	8.791521e+06	56.750546	41.467058	11.007935	16121.183073	0.905609	1.516378	430.169145
min	5.300000e+01	1.000000	-157.948486	-41.330428	0.000000	1.000000	0.000000	0.000000
25%	3.019625e+05	1.000000	77.081343	28.478713	250.000000	1.000000	2.500000	5.000000
50%	6.004089e+06	1.000000	77.191964	28.570469	400.000000	2.000000	3.200000	31.000000
75%	1.835229e+07	1.000000	77.282006	28.642758	700.000000	2.000000	3.700000	131.000000
max	1.850065e+07	216.000000	174.832089	55.976980	800000.000000	4.000000	4.900000	10934.000000

In [8]:

```
### in data analysis we check
#1. Missing values
#2. explore numerical variables
#3. explore categorical variables
#4. finding relationships between features
#5.
```

In [9]:

```
df.isnull().sum()
```

Out[9]:

```
Restaurant ID           0
Restaurant Name         0
Country Code            0
City                    0
Address                 0
Locality                0
```

```
Locality                  0
Locality Verbose          0
Longitude                 0
Latitude                  0
Cuisines                  9
Average Cost for two      0
Currency                  0
Has Table booking         0
Has Online delivery       0
Is delivering now         0
Switch to order menu      0
Price range               0
Aggregate rating          0
Rating color              0
Rating text               0
Votes                     0
dtype: int64
```

In [11]:

```
[feature for feature in df.columns if df[feature].isnull().sum()>0]
```

Out[11]:

```
['Cuisines']
```

In [19]:

```
[feature for feature in df.columns if df[feature].isnull().sum()>0]
```

Out[19]:

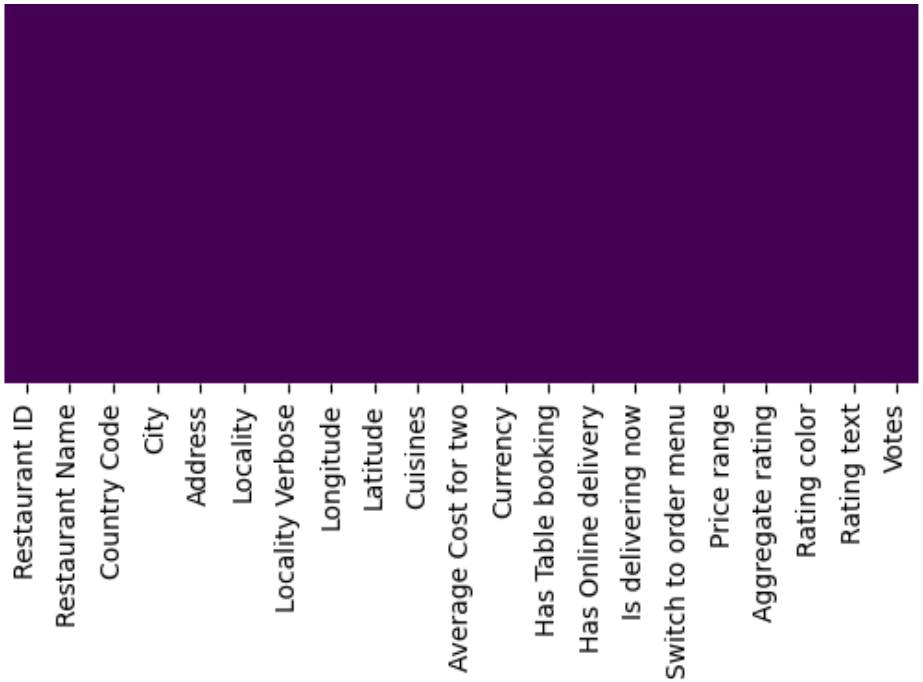
```
['Cuisines']
```

In [20]:

```
sns.heatmap(df.isnull(), yticklabels=False, cbar=False, cmap='viridis')
```

Out[20]:

```
<AxesSubplot:>
```



In [21]:

```
df_country =pd.read_excel('Country-code.xlsx')
```

In [24]:

```
df_country.columns
```

Out[24]:

```
Index(['Country Code', 'Country'], dtype='object')
```

In [23]:

```
df.columns
```

Out[23]:

```
Index(['Restaurant ID', 'Restaurant Name', 'Country Code', 'City', 'Address',
       'Locality', 'Locality Verbose', 'Longitude', 'Latitude', 'Cuisines',
       'Average Cost for two', 'Currency', 'Has Table booking',
       'Has Online delivery', 'Is delivering now', 'Switch to order menu',
       'Price range', 'Aggregate rating', 'Rating color', 'Rating text',
       'Votes'],
      dtype='object')
```

In [26]:

```
df=pd.merge(df, df_country,on= 'Country Code', how='left' )
```

In [27]:

```
df.head()
```

Out[27]:

	Restaurant ID	Restaurant Name	Country Code	City	Address	Locality	Locality Verbose	Longitude	Latitude	Cuisines	...	Has Table booking	Has Online delivery	Is delivering now	Switch to order menu	Price range	Aggregat ratin
0	6317637	Le Petit Souffle	162	Makati City	Third Floor, Century City Mall, Kalayaan Avenu...	Century City Mall, Poblacion, Makati City	Century City Mall, Poblacion, Makati City, Mak...	121.027535	14.565443	French, Japanese, Desserts	...	Yes	No	No	No	3	4
1	6304287	Izakaya Kikufuji	162	Makati City	Little Tokyo, 2277 Chino Roces Avenue, Legaspi...	Little Tokyo, Legaspi Village, Makati City	Little Tokyo, Legaspi Village, Makati City, Ma...	121.014101	14.553708	Japanese	...	Yes	No	No	No	3	4
2	6300002	Heat - Edsa Shangri-La	162	Mandaluyong City	Edsa Shangri-La, 1 Garden Way, Ortigas, Mandal...	Edsa Shangri-La, Ortigas, Mandaluyong City	Edsa Shangri-La, Ortigas, Mandaluyong City, Ma...	121.056831	14.581404	Seafood, Asian, Filipino, Indian	...	Yes	No	No	No	4	4
3	6318506	Ooma	162	Mandaluyong City	Third Floor, Mega Fashion Hall, SM Megamall, O...	SM Megamall, Ortigas, Mandaluyong City	SM Megamall, Ortigas, Mandaluyong City, Mandal...	121.056475	14.585318	Japanese, Sushi	...	No	No	No	No	4	4
4	6314302	Sambo Kojin	162	Mandaluyong City	Third Floor, Mega Atrium, SM Megamall, Ortigas...	SM Megamall, Ortigas, Mandaluyong City	SM Megamall, Ortigas, Mandaluyong City, Mandal...	121.057508	14.584450	Japanese, Korean	...	Yes	No	No	No	4	4

5 rows × 22 columns

In [28]:

```
df.dtypes
```

Out[28]:

```
Restaurant ID             int64
Restaurant Name          object
Country Code              int64
City                     object
Address                  object
Locality                 object
Locality Verbose         object
Longitude               float64
Latitude                float64
Cuisines                 object
Average Cost for two      int64
Currency                 object
Has Table booking        object
Has Online delivery      object
Is delivering now        object
Switch to order menu     object
Price range               int64
Aggregate rating        float64
Rating color             object
Rating text              object
Votes                     int64
Country                  object
dtype: object
```

In [29]:

```
df.columns
```

Out[29]:

```
Index(['Restaurant ID', 'Restaurant Name', 'Country Code', 'City', 'Address',
       'Locality', 'Locality Verbose', 'Longitude', 'Latitude', 'Cuisines',
       'Average Cost for two', 'Currency', 'Has Table booking',
       'Has Online delivery', 'Is delivering now', 'Switch to order menu',
       'Price range', 'Aggregate rating', 'Rating color', 'Rating text',
       'Votes', 'Country'],
      dtype='object')
```

In [33]:

```
country_names=df.Country.value_counts().index
```

In [34]:

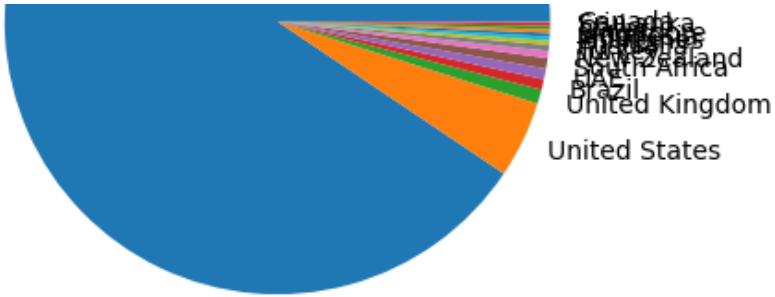
```
country_count=df.Country.value_counts().values
```

In [37]:

```
plt.pie(country_count,labels=country_names)
```

Out[37]:

```
([<matplotlib.patches.Wedge at 0x22994615670>,
  <matplotlib.patches.Wedge at 0x22994615bb0>,
  <matplotlib.patches.Wedge at 0x229946240d0>,
  <matplotlib.patches.Wedge at 0x229946245b0>,
  <matplotlib.patches.Wedge at 0x22994624a90>,
  <matplotlib.patches.Wedge at 0x22994624f70>,
  <matplotlib.patches.Wedge at 0x22994c62490>,
  <matplotlib.patches.Wedge at 0x22994c62970>,
  <matplotlib.patches.Wedge at 0x22994c62e50>,
  <matplotlib.patches.Wedge at 0x22994c6f370>,
  <matplotlib.patches.Wedge at 0x22994615640>,
  <matplotlib.patches.Wedge at 0x22994c6fd00>,
  <matplotlib.patches.Wedge at 0x22994c7d220>,
  <matplotlib.patches.Wedge at 0x22994c7d700>,
  <matplotlib.patches.Wedge at 0x22994c7dbe0>],
 [Text(-1.052256163793291, 0.3205572737577906, 'India'),
  Text(0.9911329812843455, -0.477132490415823, 'United States'),
  Text(1.0572858296119743, -0.3035567072257165, 'United Kingdom'),
  Text(1.070138816916019, -0.2545641619112621, 'Brazil'),
  Text(1.0793506814479759, -0.21213699926648824, 'UAE'),
  Text(1.086881147244973, -0.16937937230799818, 'South Africa'),
  Text(1.0918635911832035, -0.1335436192729486, 'New Zealand'),
  Text(1.0947903814016446, -0.10692998078388304, 'Turkey'),
  Text(1.096631023945382, -0.08602556201794338, 'Australia'),
  Text(1.0978070729776455, -0.06942355882735218, 'Phillipines'),
  Text(1.0986791544015209, -0.05388984768543213, 'Indonesia'),
  Text(1.0993059848742366, -0.039068550263413035, 'Singapore'),
  Text(1.0997248508282123, -0.02460187941736628, 'Qatar'),
  Text(1.0999533462179636, -0.010130949802716446, 'Sri Lanka'),
  Text(1.0999990477553414, -0.0014473898376707638, 'Canada')])
```

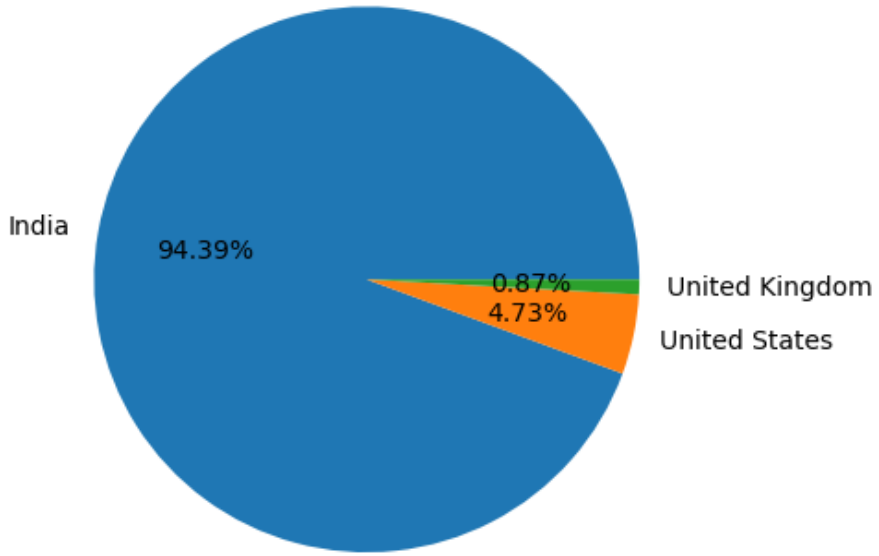


In [41]:

```
#top 3 countries
plt.pie(country_count[:3],labels=country_names[:3], autopct='%1.2f%%')
```

Out[41]:

```
([<matplotlib.patches.Wedge at 0x22994d85280>,
  <matplotlib.patches.Wedge at 0x22994d859a0>,
  <matplotlib.patches.Wedge at 0x22994d92100>],
 [Text(-1.0829742700952103, 0.19278674827836725, 'India'),
  Text(1.077281715838356, -0.22240527134123297, 'United States'),
  Text(1.0995865153823035, -0.03015783794312073, 'United Kingdom')],
 [Text(-0.590713238233751, 0.10515640815183668, '94.39%'),
  Text(0.5876082086391032, -0.12131196618612707, '4.73%'),
  Text(0.5997744629358018, -0.01644972978715676, '0.87%')])
```



In [42]:

```
df.columns
```

Out[42]:

```
Index(['Restaurant ID', 'Restaurant Name', 'Country Code', 'City', 'Address',
       'Locality', 'Locality Verbose', 'Longitude', 'Latitude', 'Cuisines',
       'Average Cost for two', 'Currency', 'Has Table booking',
       'Has Online delivery', 'Is delivering now', 'Switch to order menu',
       'Price range', 'Aggregate rating', 'Rating color', 'Rating text',
       'Votes', 'Country'],
      dtype='object')
```

In [47]:

```
ratings=df.groupby(['Aggregate rating', 'Rating color', 'Rating text']).size().reset_index().rename(columns={0:'Rating_Count'})
```

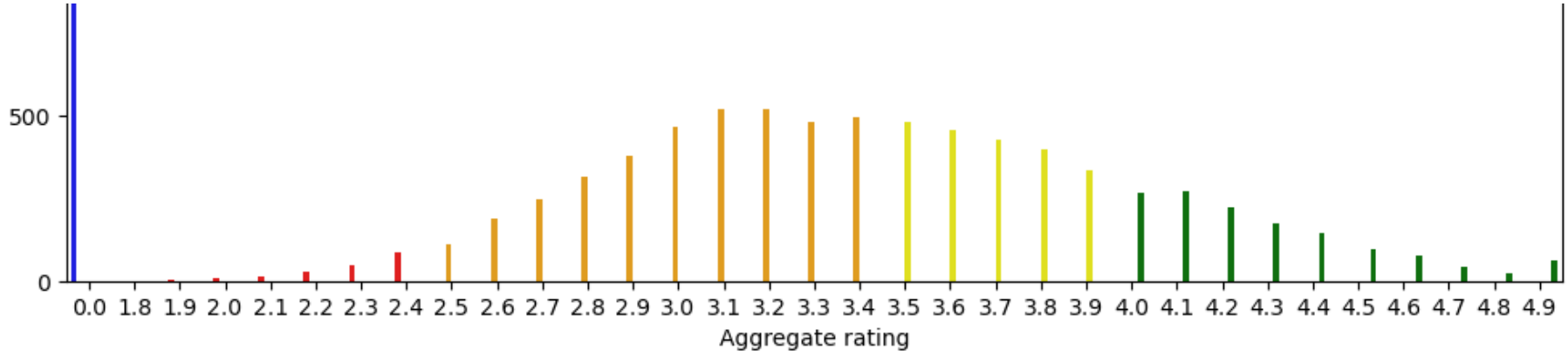
In [54]:

```
import matplotlib

matplotlib.rcParams['figure.figsize'] =(12,6)
sns.barplot(x='Aggregate rating', y='Rating_Count', data=ratings,hue='Rating color',palette=['blue','red','orange','yellow','green
','green'])
```

Out[54]:

```
<AxesSubplot:xlabel='Aggregate rating', ylabel='Rating_Count'>
```

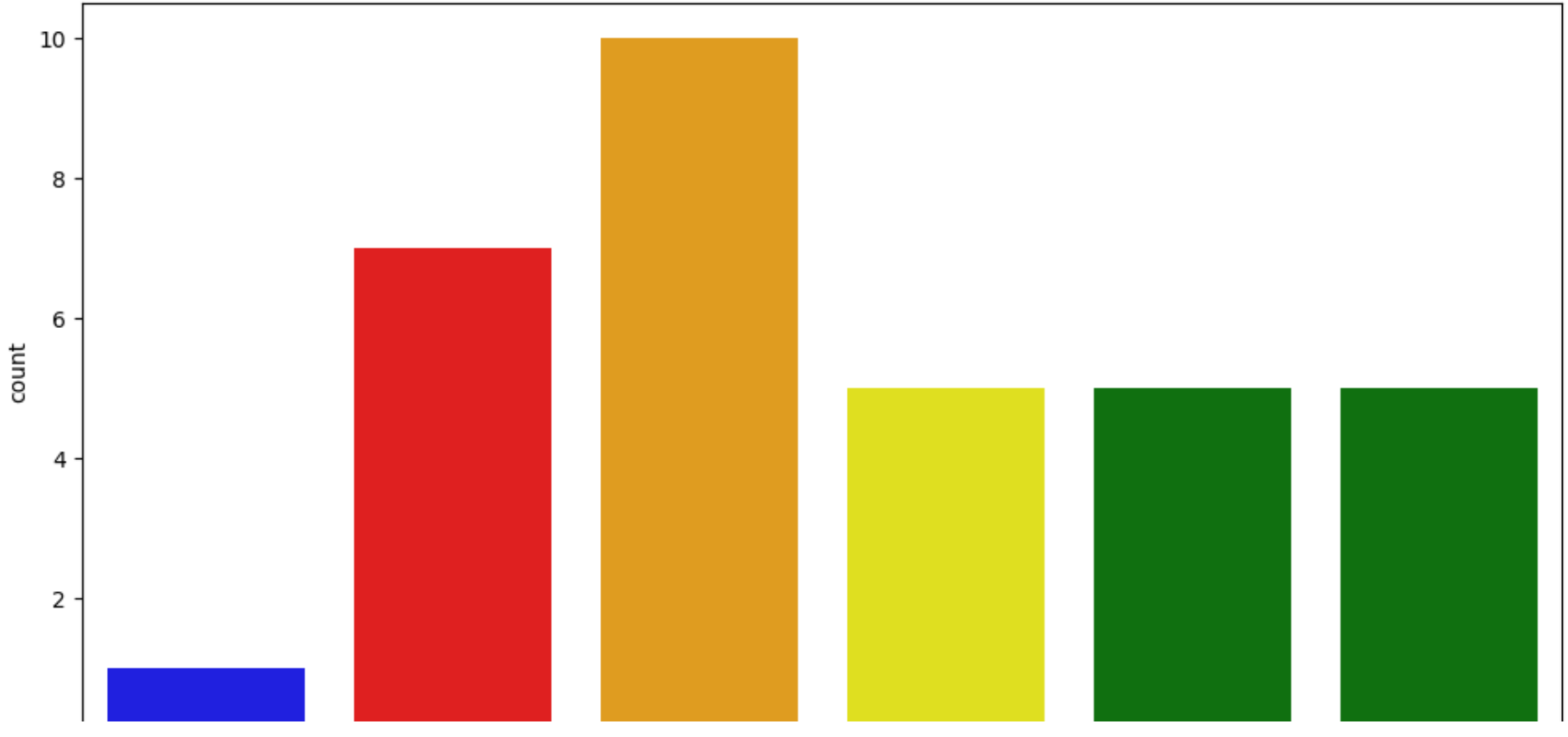


In [56]:

```
#count_plot
sns.countplot(x="Rating color",data=ratings,palette=['blue','red','orange','yellow','green','green'])
```

Out[56]:

```
<AxesSubplot:xlabel='Rating color', ylabel='count'>
```



In [61]:

```
##find the countries name that has given zero rating

df[df['Aggregate rating']==0.0]['Country'].value_counts().reset_index()
```

Out[61]:

	index	Country
0	India	2139
1	Brazil	5
2	United States	3
3	United Kingdom	1

In [62]:

```
df.columns
```

Out[62]:

```
Index(['Restaurant ID', 'Restaurant Name', 'Country Code', 'City', 'Address',
       'Locality', 'Locality Verbose', 'Longitude', 'Latitude', 'Cuisines',
       'Average Cost for two', 'Currency', 'Has Table booking',
       'Has Online delivery', 'Is delivering now', 'Switch to order menu',
       'Price range', 'Aggregate rating', 'Rating color', 'Rating text',
       'Votes', 'Country'],
      dtype='object')
```

In [68]:

```
df[ ['Country','Has Online delivery']].groupby(['Country','Has Online delivery']).size().reset_index()
```

Out[68]:

	Country	Has Online delivery	0
0	Australia	No	24
1	Brazil	No	60
2	Canada	No	4
3	India	No	6229

4	India	Yes	2423
	Country	Has Online delivery	0
5	Indonesia	No	21
6	New Zealand	No	40
7	Phillipines	No	22
8	Qatar	No	20
9	Singapore	No	20
10	South Africa	No	60
11	Sri Lanka	No	20
12	Turkey	No	34
13	UAE	No	32
14	UAE	Yes	28
15	United Kingdom	No	80
16	United States	No	434

In [69]:

```
##which countries do have online deliveries

df.columns
```

Out[69]:

```
Index(['Restaurant ID', 'Restaurant Name', 'Country Code', 'City', 'Address',
       'Locality', 'Locality Verbose', 'Longitude', 'Latitude', 'Cuisines',
       'Average Cost for two', 'Currency', 'Has Table booking',
       'Has Online delivery', 'Is delivering now', 'Switch to order menu',
       'Price range', 'Aggregate rating', 'Rating color', 'Rating text',
       'Votes', 'Country'],
      dtype='object')
```

In [74]:

```

```

Out[74]:

```
Index(['New Delhi', 'Gurgaon', 'Noida', 'Faridabad', 'Ghaziabad',
       'Bhubaneshwar', 'Amritsar', 'Ahmedabad', 'Lucknow', 'Guwahati',
       ...
       'Ojo Caliente', 'Montville', 'Monroe', 'Miller', 'Middleton Beach',
       'Panchkula', 'Mc Millan', 'Mayfield', 'Macedon', 'Vineland Station'],
      dtype='object', length=141)
```

In [84]:

In [84]:

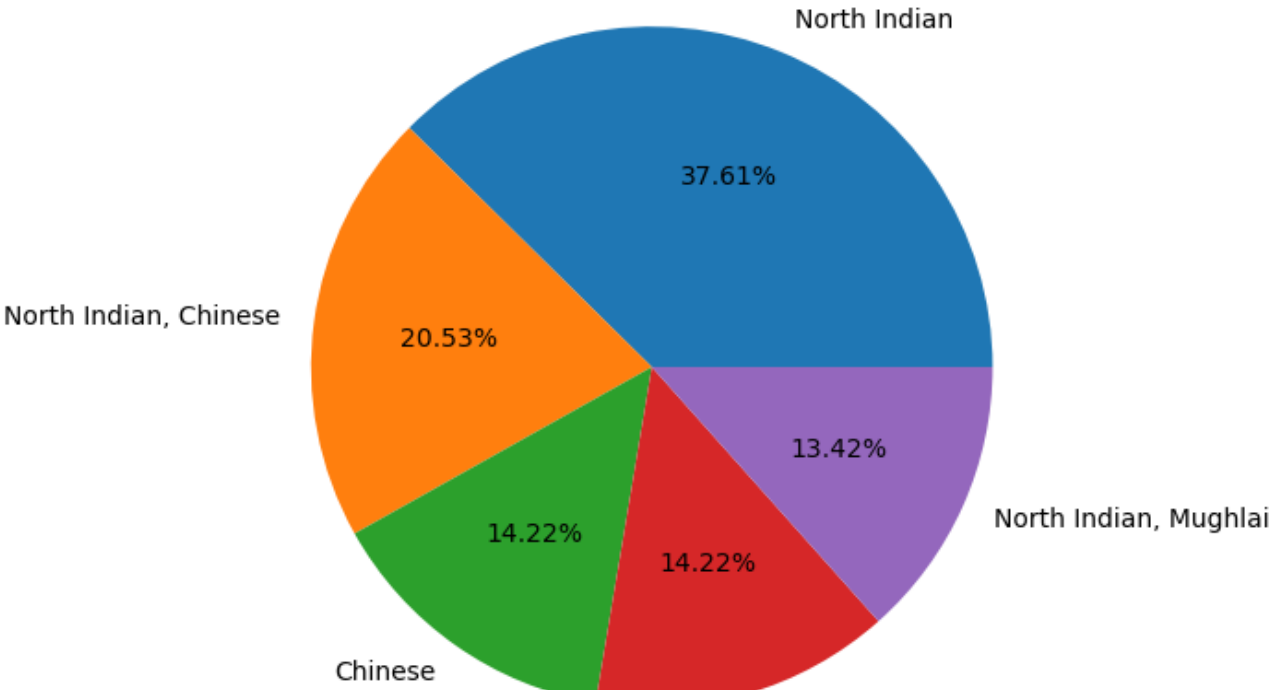
```
Cuisines_name= df['Cuisines'].value_counts().index
Cuisines_count = df['Cuisines'].value_counts().values
```

In [85]:

```
plt.pie(Cuisines_count[:5],labels=Cuisines_name[:5], autopct='%1.2f%%')
```

Out[85]:

```
([<matplotlib.patches.Wedge at 0x22999ad09a0>,
  <matplotlib.patches.Wedge at 0x22999adf100>,
  <matplotlib.patches.Wedge at 0x22999adf820>,
  <matplotlib.patches.Wedge at 0x22999adff40>,
  <matplotlib.patches.Wedge at 0x22999aed6d0>],
 [Text(0.4175823090707363, 1.0176566292965188, 'North Indian'),
  Text(-1.090169487983684, 0.14673270756512372, 'North Indian, Chinese'),
  Text(-0.6326704341218391, -0.8998489438722946, 'Chinese'),
  Text(0.3048764124205039, -1.0569060379946758, 'Fast Food'),
  Text(1.0036916287969155, -0.45011455684413826, 'North Indian, Mughlai')],
 [Text(0.22777216858403795, 0.5550854341617375, '37.61%'),
  Text(-0.5946379025365549, 0.0800360223082493, '20.53%'),
  Text(-0.3450929640664576, -0.4908266966576152, '14.22%'),
  Text(0.16629622495663846, -0.5764942025425503, '14.22%'),
  Text(0.5474681611619538, -0.24551703100589356, '13.42%')])
```



In [83]:

```
##find the top 10 cuisins
df.columns
```

Out[83]:

```
Index(['Restaurant ID', 'Restaurant Name', 'Country Code', 'City', 'Address',
       'Locality', 'Locality Verbose', 'Longitude', 'Latitude', 'Cuisines',
       'Average Cost for two', 'Currency', 'Has Table booking',
       'Has Online delivery', 'Is delivering now', 'Switch to order menu',
       'Price range', 'Aggregate rating', 'Rating color', 'Rating text',
       'Votes', 'Country'],
      dtype='object')
```

In [86]:

```
df.columns
```

Out[86]:

```
Index(['Restaurant ID', 'Restaurant Name', 'Country Code', 'City', 'Address',
       'Locality', 'Locality Verbose', 'Longitude', 'Latitude', 'Cuisines',
       'Average Cost for two', 'Currency', 'Has Table booking',
       'Has Online delivery', 'Is delivering now', 'Switch to order menu',
       'Price range', 'Aggregate rating', 'Rating color', 'Rating text',
       'Votes', 'Country'],
      dtype='object')
```

Black firday dataset

In [ ]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplt as plt
import seaborn as sns
%matplotlib inline
```

In [89]:

```
##importing the dataset
df_train = pd.read_csv('train.csv')
df_train.head()
```

```
df_test = pd.read_csv('test.csv')
df_test.head()
```

Out[89]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Category_1	Product_Category_2	Product_Category_3
0	1000004	P00128942	M	46-50	7	B	2	1	1	11.0	NaN
1	1000009	P00113442	M	26-35	17	C	0	0	3	5.0	NaN
2	1000010	P00288442	F	36-45	1	B	4+	1	5	14.0	NaN
3	1000010	P00145342	F	36-45	1	B	4+	1	4	9.0	NaN
4	1000011	P00053842	F	26-35	1	C	1	0	4	5.0	12.0

In [91]:

```
#append both train and test data

df = df_train.append(df_test)
```

```
C:\Users\gourav mehta\AppData\Local\Temp\ipykernel_13452\3903015598.py:3: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a future version. Use pandas.concat instead.
  df = df_train.append(df_test)
```

In [92]:

```
df.head()
```

Out[92]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Category_1	Product_Category_2	Product_Category_3	Purchase
0	1000001	P00069042	F	0-17	10	A	2	0	3	NaN	NaN	8370.0
1	1000001	P00248942	F	0-17	10	A	2	0	1	6.0	14.0	15200.0
2	1000001	P00087842	F	0-17	10	A	2	0	12	NaN	NaN	1422.0
3	1000001	P00085442	F	0-17	10	A	2	0	12	14.0	NaN	1057.0
4	1000002	P00285442	M	55+	16	C	4+	0	8	NaN	NaN	7969.0

In [93]:

```
##basic code
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 783667 entries, 0 to 233598
Data columns (total 12 columns):
 #   Column                      Non-Null Count   Dtype
---  ------                      --------------   -----
 0   User_ID                     783667 non-null  int64
 1   Product_ID                  783667 non-null  object
 2   Gender                      783667 non-null  object
 3   Age                         783667 non-null  object
 4   Occupation                  783667 non-null  int64
 5   City_Category               783667 non-null  object
 6   Stay_In_Current_City_Years  783667 non-null  object
 7   Marital_Status              783667 non-null  int64
 8   Product_Category_1          783667 non-null  int64
 9   Product_Category_2          537685 non-null  float64
 10  Product_Category_3          237858 non-null  float64
 11  Purchase                    550068 non-null  float64
dtypes: float64(3), int64(4), object(5)
memory usage: 77.7+ MB
```

In [94]:

```
df.describe()
```

Out[94]:

	User_ID	Occupation	Marital_Status	Product_Category_1	Product_Category_2	Product_Category_3	Purchase
count	7.836670e+05	783667.000000	783667.000000	783667.000000	537685.000000	237858.000000	550068.000000
mean	1.003029e+06	8.079300	0.409777	5.366196	9.844506	12.668605	9263.968713
std	1.727267e+03	6.522206	0.491793	3.878160	5.089093	4.125510	5023.065394
min	1.000001e+06	0.000000	0.000000	1.000000	2.000000	3.000000	12.000000
25%	1.001519e+06	2.000000	0.000000	1.000000	5.000000	9.000000	5823.000000
50%	1.003075e+06	7.000000	0.000000	5.000000	9.000000	14.000000	8047.000000
75%	1.004478e+06	14.000000	1.000000	8.000000	15.000000	16.000000	12054.000000
max	1.006040e+06	20.000000	1.000000	20.000000	18.000000	18.000000	23961.000000

In [95]:

```
df.drop(["User_ID"],axis =1, inplace =True)
```

In [96]:

```
df.head()
```

```
df.head()
```

Out[96]:

	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Category_1	Product_Category_2	Product_Category_3	Purchase
0	P00069042	F	0-17	10	A	2	0	3	NaN	NaN	8370.0
1	P00248942	F	0-17	10	A	2	0	1	6.0	14.0	15200.0
2	P00087842	F	0-17	10	A	2	0	12	NaN	NaN	1422.0
3	P00085442	F	0-17	10	A	2	0	12	14.0	NaN	1057.0
4	P00285442	M	55+	16	C	4+	0	8	NaN	NaN	7969.0

In [97]:

```
##convert age categorical into numerical feature
pd.get_dummies(df['Gender'])
```

Out[97]:

	F	M
0	1	0
1	1	0
2	1	0
3	1	0
4	0	1
...	...	...
233594	1	0
233595	1	0
233596	1	0
233597	1	0
233598	1	0

783667 rows × 2 columns

In [98]:

```
df['Gender'] = df['Gender'].map({'F':0,'M':1})
```

In [99]:

```
df.head()
```

```
df.head()
```

Out[99]:

	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Category_1	Product_Category_2	Product_Category_3	Purchase
0	P00069042	0	0-17	10	A	2	0	3	NaN	NaN	8370.0
1	P00248942	0	0-17	10	A	2	0	1	6.0	14.0	15200.0
2	P00087842	0	0-17	10	A	2	0	12	NaN	NaN	1422.0
3	P00085442	0	0-17	10	A	2	0	12	14.0	NaN	1057.0
4	P00285442	1	55+	16	C	4+	0	8	NaN	NaN	7969.0

In [100]:

```
#handle categorical feature age

df.Age.value_counts()
```

Out[100]:

```
26-35    313015
36-45    156724
18-25    141953
46-50     65278
51-55     54784
55+       30579
0-17      21334
Name: Age, dtype: int64
```

In [102]:

```
df.Age.unique()
```

Out[102]:

```
array(['0-17', '55+', '26-35', '46-50', '51-55', '36-45', '18-25'],
      dtype=object)
```

In [104]:

```
pd.get_dummies(df['Age'], drop_first=True)
```

Out[104]:

	18-25	26-35	36-45	46-50	51-55	55+
0	0	0	0	0	0	0
1	0	0	0	0	0	0

2	18-25	26-35	36-45	46-50	51-55	55+
3	0	0	0	0	0	0
4	0	0	0	0	0	1
...	...	...	...	...	...	...
233594	0	1	0	0	0	0
233595	0	1	0	0	0	0
233596	0	1	0	0	0	0
233597	0	0	0	1	0	0
233598	0	0	0	1	0	0

783667 rows × 6 columns

In [105]:

```
df['Age'] =df['Age'].map({'0-17':1, '18-25':2,'26-35':3,'36-45':4,'46-50':5,'51-55':6,'55+':7})
```

In [106]:

```
df.head()
```

Out[106]:

	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Category_1	Product_Category_2	Product_Category_3	Purchase
0	P00069042	0	1	10	A	2	0	3	NaN	NaN	8370.0
1	P00248942	0	1	10	A	2	0	1	6.0	14.0	15200.0
2	P00087842	0	1	10	A	2	0	12	NaN	NaN	1422.0
3	P00085442	0	1	10	A	2	0	12	14.0	NaN	1057.0
4	P00285442	1	7	16	C	4+	0	8	NaN	NaN	7969.0

In [109]:

```
df_city=pd.get_dummies(df.City_Category,drop_first=True)
```

In [110]:

```
df_city.head()
```

Out[110]:

B	C

0	B	C
1	0	0
2	0	0
3	0	0
4	0	1

In [111]:

```
df= pd.concat([df,df_city],axis =1)
```

In [112]:

```
df.head()
```

Out[112]:

	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Category_1	Product_Category_2	Product_Category_3	Purchase	B	C
0	P00069042	0	1	10	A	2	0	3	NaN	NaN	8370.0	0	0
1	P00248942	0	1	10	A	2	0	1	6.0	14.0	15200.0	0	0
2	P00087842	0	1	10	A	2	0	12	NaN	NaN	1422.0	0	0
3	P00085442	0	1	10	A	2	0	12	14.0	NaN	1057.0	0	0
4	P00285442	1	7	16	C	4+	0	8	NaN	NaN	7969.0	0	1

In [114]:

```
##drop city category
df.drop('City_Category',inplace =True,axis=1)
```

In [115]:

```
df.head()
```

Out[115]:

	Product_ID	Gender	Age	Occupation	Stay_In_Current_City_Years	Marital_Status	Product_Category_1	Product_Category_2	Product_Category_3	Purchase	B	C
0	P00069042	0	1	10	2	0	3	NaN	NaN	8370.0	0	0
1	P00248942	0	1	10	2	0	1	6.0	14.0	15200.0	0	0
2	P00087842	0	1	10	2	0	12	NaN	NaN	1422.0	0	0
3	P00085442	0	1	10	2	0	12	14.0	NaN	1057.0	0	0
4	P00285442	1	7	16	4+	0	8	NaN	NaN	7969.0	0	1

Product_ID Gender Age Occupation Stay_In_Current_City_Years Marital_Status Product_Category_1 Product_Category_2 Product_Category_3 Purchase B C

In [116]:

```
df.isnull().sum()
```

Out[116]:

```
Product_ID                        0
Gender                            0
Age                               0
Occupation                        0
Stay_In_Current_City_Years        0
Marital_Status                    0
Product_Category_1                0
Product_Category_2           245982
Product_Category_3           545809
Purchase                     233599
B                                 0
C                                 0
dtype: int64
```

In [119]:

```
#focus on replacing missing values
df.Product_Category_2.value_counts()
```

Out[119]:

```
8.0     91317
14.0    78834
2.0     70498
16.0    61687
15.0    54114
5.0     37165
4.0     36705
6.0     23575
11.0    20230
17.0    19104
13.0    15054
9.0      8177
12.0     7801
10.0     4420
3.0      4123
18.0     4027
7.0       854
Name: Product_Category_2, dtype: int64
```

In [127]:

```
df['Product_Category_2']=df['Product_Category_2'].fillna(df['Product_Category_2'].mode()[0])
```

In [132]:

```
df.isnull().sum()
```

Out[132]:

```
Product_ID                        0
Gender                            0
Age                               0
Occupation                        0
Stay_In_Current_City_Years        0
Marital_Status                    0
Product_Category_1                0
Product_Category_2                0
Product_Category_3                0
Purchase                     233599
B                                 0
C                                 0
dtype: int64
```

In [129]:

```
df.Product_Category_3.value_counts()
```

Out[129]:

```
16.0    46469
15.0    39968
14.0    26283
17.0    23818
5.0     23799
8.0     17861
9.0     16532
12.0    13115
13.0     7849
6.0      6888
18.0     6621
4.0      2691
11.0     2585
10.0     2501
3.0       878
Name: Product_Category_3, dtype: int64
```

In [130]:

```
df['Product_Category_3']=df['Product_Category_3'].fillna(df['Product_Category_3'].mode()[0])
```

In [131]:

```
df.head()
```

Out[131]:

	Product_ID	Gender	Age	Occupation	Stay_In_Current_City_Years	Marital_Status	Product_Category_1	Product_Category_2	Product_Category_3	Purchase	B	C
0	P00069042	0	1	10	2	0	3	8.0	16.0	8370.0	0	0
1	P00248942	0	1	10	2	0	1	6.0	14.0	15200.0	0	0
2	P00087842	0	1	10	2	0	12	8.0	16.0	1422.0	0	0
3	P00085442	0	1	10	2	0	12	14.0	16.0	1057.0	0	0
4	P00285442	1	7	16	4+	0	8	8.0	16.0	7969.0	0	1

In [133]:

```
df.head()
```

Out[133]:

	Product_ID	Gender	Age	Occupation	Stay_In_Current_City_Years	Marital_Status	Product_Category_1	Product_Category_2	Product_Category_3	Purchase	B	C
0	P00069042	0	1	10	2	0	3	8.0	16.0	8370.0	0	0
1	P00248942	0	1	10	2	0	1	6.0	14.0	15200.0	0	0
2	P00087842	0	1	10	2	0	12	8.0	16.0	1422.0	0	0
3	P00085442	0	1	10	2	0	12	14.0	16.0	1057.0	0	0
4	P00285442	1	7	16	4+	0	8	8.0	16.0	7969.0	0	1

In [134]:

```
df.shape
```

Out[134]:

```
(783667, 12)
```

In [135]:

```
df['Stay_In_Current_City_Years'].unique()
```

Out[135]:

```
array(['2', '4+', '3', '1', '0'], dtype=object)
```

In [136]:

```
df['Stay_In_Current_City_Years']=df['Stay_In_Current_City_Years'].str.replace('+','')
```

```
C:\Users\gourav mehta\AppData\Local\Temp\ipykernel_13452\2063355665.py:1: FutureWarning: The default value of regex will change fr
```

```
om True to False in a future version. In addition, single character regular expressions will *not* be treated as literal strings wh
en regex=True.
  df['Stay_In_Current_City_Years']=df['Stay_In_Current_City_Years'].str.replace('+','')
```

In [139]:

```
df['Stay_In_Current_City_Years']= df['Stay_In_Current_City_Years'].astype(int)
```

In [140]:

```
df['Stay_In_Current_City_Years']
```

Out[140]:

```
0         2
1         2
2         2
3         2
4         4
         ..
233594    4
233595    4
233596    4
233597    4
233598    4
Name: Stay_In_Current_City_Years, Length: 783667, dtype: int32
```

In [141]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 783667 entries, 0 to 233598
Data columns (total 12 columns):
 #   Column                      Non-Null Count   Dtype
---  ------                      --------------   -----
 0   Product_ID                  783667 non-null  object
 1   Gender                      783667 non-null  int64
 2   Age                         783667 non-null  int64
 3   Occupation                  783667 non-null  int64
 4   Stay_In_Current_City_Years  783667 non-null  int32
 5   Marital_Status              783667 non-null  int64
 6   Product_Category_1          783667 non-null  int64
 7   Product_Category_2          783667 non-null  float64
 8   Product_Category_3          783667 non-null  float64
 9   Purchase                    550068 non-null  float64
 10  B                           783667 non-null  uint8
 11  C                           783667 non-null  uint8
dtypes: float64(3), int32(1), int64(5), object(1), uint8(2)
memory usage: 80.4+ MB
```

In [146]:

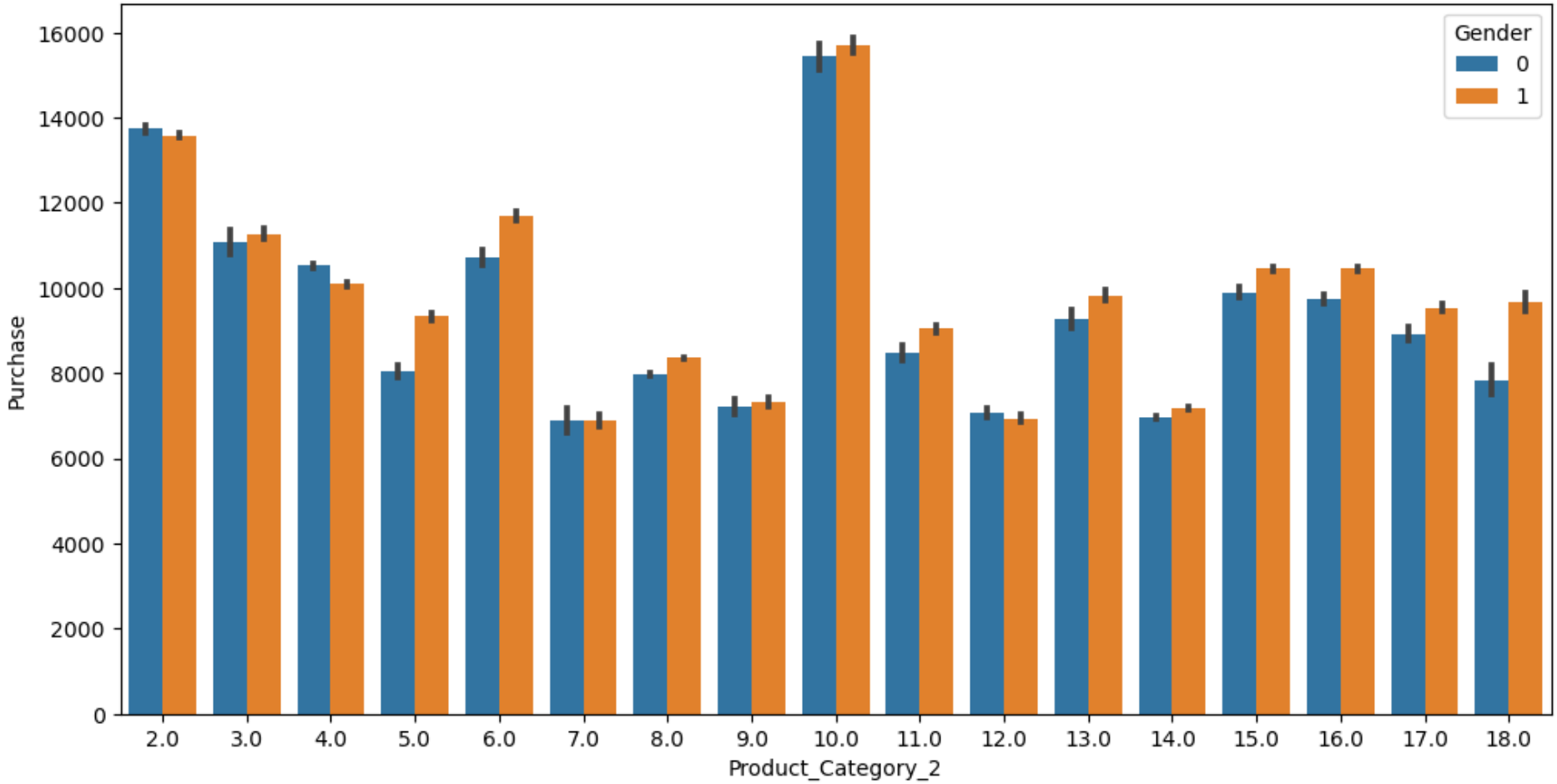
```
##visualizatio

sns.barplot('Product_Category_2','Purchase',hue='Gender',data=df)
```

```
C:\Users\gourav mehta\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variables as keywor
d args: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit
keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[146]:

```
<AxesSubplot:xlabel='Product_Category_2', ylabel='Purchase'>
```



In [148]:

```
#feature scaling
```

```
df_test = df[df['Purchase'].isnull()]
```

In [149]:

```
df_train = df[~df['Purchase'].isnull()]
```

In [150]:

```
df_test.shape
```

Out[150]:

```
(233599, 12)
```

In [151]:

```
df_train.shape
```

Out[151]:

```
(550068, 12)
```

In [152]:

```
df.shape
```

Out[152]:

```
(783667, 12)
```

In [158]:

```
X.shape
```

Out[158]:

```
(550067, 12)
```

In [155]:

```
y.shape
```

Out[155]:

```
(550068,)
```

In [159]:

```
X=df_train.drop('Purchase',axis=1)
y=df_train['Purchase']
```

In [ ]:

In [160]:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.33, random_state=42)
```

In [162]:

```
X_train.drop('Product_ID',axis=1,inplace=True)
X_test.drop('Product_ID',axis=1,inplace=True)
```

In [163]:

```
from sklearn.preprocessing import StandardScaler
sc= StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

In [ ]:

In [164]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

In [165]:

```
train_df = pd.read_excel('Data_train.xlsx')
```

In [166]:

```
train_df.head()
```

Out[166]:

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Duration	Total_Stops	Additional_Info	Price
0	IndiGo	24/03/2019	Banglore	New Delhi	BLR → DEL	22:20	01:10 22 Mar	2h 50m	non-stop	No info	3897
1	Air India	1/05/2019	Kolkata	Banglore	CCU → IXR → BBI → BLR	05:50	13:15	7h 25m	2 stops	No info	7662

2	Jet Airways	9/06/2019	Delhi	Cochin	DEL → LKO → BOM → COK	09:25	04:25 10 Jun	19h	2 stops	No info	13882
3	IndiGo	12/05/2019	Kolkata	Banglore	CCU → NAG → BLR	18:05	23:30	5h 25m	1 stop	No info	6218
4	IndiGo	01/03/2019	Banglore	New Delhi	BLR → NAG → DEL	16:50	21:35	4h 45m	1 stop	No info	13302

In [167]:

```
test_df = pd.read_excel('Test_set.xlsx')
```

In [168]:

```
test_df.head()
```

Out[168]:

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Duration	Total_Stops	Additional_Info
0	Jet Airways	6/06/2019	Delhi	Cochin	DEL → BOM → COK	17:30	04:25 07 Jun	10h 55m	1 stop	No info
1	IndiGo	12/05/2019	Kolkata	Banglore	CCU → MAA → BLR	06:20	10:20	4h	1 stop	No info
2	Jet Airways	21/05/2019	Delhi	Cochin	DEL → BOM → COK	19:15	19:00 22 May	23h 45m	1 stop	In-flight meal not included
3	Multiple carriers	21/05/2019	Delhi	Cochin	DEL → BOM → COK	08:00	21:00	13h	1 stop	No info
4	Air Asia	24/06/2019	Banglore	Delhi	BLR → DEL	23:55	02:45 25 Jun	2h 50m	non-stop	No info

In [209]:

```
final_df = train_df.append(test_df)
```

```
C:\Users\gourav mehta\AppData\Local\Temp\ipykernel_13452\714863615.py:1: FutureWarning: The frame.append method is deprecated and
will be removed from pandas in a future version. Use pandas.concat instead.
  final_df = train_df.append(test_df)
```

In [170]:

```
final_df.head()
```

Out[170]:

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Duration	Total_Stops	Additional_Info	Price
0	IndiGo	24/03/2019	Banglore	New Delhi	BLR → DEL	22:20	01:10 22 Mar	2h 50m	non-stop	No info	3897.0
1	Air India	1/05/2019	Kolkata	Banglore	CCU → IXR → BBI → BLR	05:50	13:15	7h 25m	2 stops	No info	7662.0
2	Jet Airways	9/06/2019	Delhi	Cochin	DEL → LKO → BOM → COK	09:25	04:25 10 Jun	19h	2 stops	No info	13882.0
3	IndiGo	12/05/2019	Kolkata	Banglore	CCU → NAG → BLR	18:05	23:30	5h 25m	1 stop	No info	6218.0

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Duration	Total_Stops	Additional_Info	Price
4	IndiGo	01/03/2019	Banglore	New Delhi	BLR → NAG → DEL	16:50	21:35	4h 45m	1 stop	No info	13302.0

In [171]:

```
final_df.shape
```

Out[171]:

```
(13354, 11)
```

In [172]:

```
test_df.shape
```

Out[172]:

```
(2671, 10)
```

In [173]:

```
train_df.shape
```

Out[173]:

```
(10683, 11)
```

In [177]:

```
final_df.head()
```

Out[177]:

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Duration	Total_Stops	Additional_Info	Price
0	IndiGo	24/03/2019	Banglore	New Delhi	BLR → DEL	22:20	01:10 22 Mar	2h 50m	non-stop	No info	3897.0
1	Air India	1/05/2019	Kolkata	Banglore	CCU → IXR → BBI → BLR	05:50	13:15	7h 25m	2 stops	No info	7662.0
2	Jet Airways	9/06/2019	Delhi	Cochin	DEL → LKO → BOM → COK	09:25	04:25 10 Jun	19h	2 stops	No info	13882.0
3	IndiGo	12/05/2019	Kolkata	Banglore	CCU → NAG → BLR	18:05	23:30	5h 25m	1 stop	No info	6218.0
4	IndiGo	01/03/2019	Banglore	New Delhi	BLR → NAG → DEL	16:50	21:35	4h 45m	1 stop	No info	13302.0

In [178]:

```
final_df.columns
```

Out[178]:

```
Index(['Airline', 'Date_of_Journey', 'Source', 'Destination', 'Route',
       'Dep_Time', 'Arrival_Time', 'Duration', 'Total_Stops',
       'Additional_Info', 'Price'],
```

```
      dtype='object')
```

In [183]:

```
final_df['Date_of_Journey'].str.split('/').str[0]
```

Out[183]:

```
0       24
1        1
2        9
3       12
4       01
        ..
2666     6
2667    27
2668     6
2669     6
2670    15
Name: Date_of_Journey, Length: 13354, dtype: object
```

In [210]:

```
final_df['Date']= final_df['Date_of_Journey'].str.split('/').str[0].astype(int)
final_df['Month']=final_df['Date_of_Journey'].str.split('/').str[1].astype(int)
final_df['Year']=final_df['Date_of_Journey'].str.split('/').str[2].astype(int)
```

In [187]:

```
final_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 13354 entries, 0 to 2670
Data columns (total 14 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   Airline          13354 non-null  object
 1   Date_of_Journey  13354 non-null  object
 2   Source           13354 non-null  object
 3   Destination      13354 non-null  object
 4   Route            13353 non-null  object
 5   Dep_Time         13354 non-null  object
 6   Arrival_Time     13354 non-null  object
 7   Duration         13354 non-null  object
 8   Total_Stops      13353 non-null  object
 9   Additional_Info  13354 non-null  object
 10  Price            10683 non-null  float64
 11  Date             13354 non-null  int32
 12  Month            13354 non-null  int32
 13  Year             13354 non-null  int32
dtypes: float64(1), int32(3), object(10)
```

```
dtypes: float64(1), int32(3), object(10)
memory usage: 1.9+ MB
```

In [211]:

```
final_df.drop('Date_of_Journey',inplace =True,axis =1)
```

In [189]:

```
final_df.head()
```

Out[189]:

	Airline	Source	Destination	Route	Dep_Time	Arrival_Time	Duration	Total_Stops	Additional_Info	Price	Date	Month	Year
0	IndiGo	Banglore	New Delhi	BLR → DEL	22:20	01:10 22 Mar	2h 50m	non-stop	No info	3897.0	24	3	2019
1	Air India	Kolkata	Banglore	CCU → IXR → BBI → BLR	05:50	13:15	7h 25m	2 stops	No info	7662.0	1	5	2019
2	Jet Airways	Delhi	Cochin	DEL → LKO → BOM → COK	09:25	04:25 10 Jun	19h	2 stops	No info	13882.0	9	6	2019
3	IndiGo	Kolkata	Banglore	CCU → NAG → BLR	18:05	23:30	5h 25m	1 stop	No info	6218.0	12	5	2019
4	IndiGo	Banglore	New Delhi	BLR → NAG → DEL	16:50	21:35	4h 45m	1 stop	No info	13302.0	1	3	2019

In [ ]:

In [212]:

```
final_df['Arrival_Time'] = final_df['Arrival_Time'].str.split(' ').str[0]
```

In [192]:

```
final_df.columns
```

Out[192]:

```
Index(['Airline', 'Source', 'Destination', 'Route', 'Dep_Time', 'Arrival_Time',
       'Duration', 'Total_Stops', 'Additional_Info', 'Price', 'Date', 'Month',
       'Year'],
      dtype='object')
```

In [213]:

```
final_df['Arrival_hour'] = final_df['Arrival_Time'].str.split(':').str[0].astype(int)
final_df['Arrival_min'] = final_df['Arrival_Time'].str.split(':').str[1].astype(int)
```

In [214]:

```
final_df.drop('Arrival_Time',axis=1, inplace=True)
```

```
final_df.drop('Arrival_Time',axis=1, inplace =True)
```

In [215]:

```
final_df['Dep_hour'] = final_df['Dep_Time'].str.split(':').str[0].astype(int)
final_df['Dep_min'] = final_df['Dep_Time'].str.split(':').str[1].astype(int)

final_df.drop('Dep_Time',axis=1, inplace =True)
```

In [199]:

```
final_df.head()
```

Out[199]:

	Airline	Source	Destination	Route	Duration	Total_Stops	Additional_Info	Price	Date	Month	Year	Arrival_hour	Arrival_min	Dep_hour	Dep_min
0	IndiGo	Banglore	New Delhi	BLR → DEL	2h 50m	non-stop	No info	3897.0	24	3	2019	01	10	22	20
1	Air India	Kolkata	Banglore	CCU → IXR → BBI → BLR	7h 25m	2 stops	No info	7662.0	1	5	2019	13	15	5	50
2	Jet Airways	Delhi	Cochin	DEL → LKO → BOM → COK	19h	2 stops	No info	13882.0	9	6	2019	04	25	9	25
3	IndiGo	Kolkata	Banglore	CCU → NAG → BLR	5h 25m	1 stop	No info	6218.0	12	5	2019	23	30	18	5
4	IndiGo	Banglore	New Delhi	BLR → NAG → DEL	4h 45m	1 stop	No info	13302.0	1	3	2019	21	35	16	50

In [200]:

```
final_df.Total_Stops.unique()
```

Out[200]:

```
array(['non-stop', '2 stops', '1 stop', '3 stops', nan, '4 stops'],
      dtype=object)
```

In [201]:

```
final_df.Total_Stops.value_counts()
```

Out[201]:

```
1 stop      7056
non-stop    4340
2 stops     1899
3 stops       56
4 stops        2
Name: Total_Stops, dtype: int64
```

In [216]:

```
final_df.Total_Stops = final_df.Total_Stops.map({'non-stop':0,
```

```
                                                    '1 stop':1,
                                                    '2 stops':2,
                                                    '3 stops':3,
                                                    '4 stops':4,
                                                    'nan':1
                                                   })
```

In [217]:

```
final_df.head()
```

Out[217]:

	Airline	Source	Destination	Route	Duration	Total_Stops	Additional_Info	Price	Date	Month	Year	Arrival_hour	Arrival_min	Dep_hour	Dep_min
0	IndiGo	Banglore	New Delhi	BLR → DEL	2h 50m	0.0	No info	3897.0	24	3	2019	1	10	22	20
1	Air India	Kolkata	Banglore	CCU → IXR → BBI → BLR	7h 25m	2.0	No info	7662.0	1	5	2019	13	15	5	50
2	Jet Airways	Delhi	Cochin	DEL → LKO → BOM → COK	19h	2.0	No info	13882.0	9	6	2019	4	25	9	25
3	IndiGo	Kolkata	Banglore	CCU → NAG → BLR	5h 25m	1.0	No info	6218.0	12	5	2019	23	30	18	5
4	IndiGo	Banglore	New Delhi	BLR → NAG → DEL	4h 45m	1.0	No info	13302.0	1	3	2019	21	35	16	50

In [219]:

```
final_df.drop('Route', axis=1, inplace =True)
```

In [220]:

```
final_df.head()
```

Out[220]:

	Airline	Source	Destination	Duration	Total_Stops	Additional_Info	Price	Date	Month	Year	Arrival_hour	Arrival_min	Dep_hour	Dep_min
0	IndiGo	Banglore	New Delhi	2h 50m	0.0	No info	3897.0	24	3	2019	1	10	22	20
1	Air India	Kolkata	Banglore	7h 25m	2.0	No info	7662.0	1	5	2019	13	15	5	50
2	Jet Airways	Delhi	Cochin	19h	2.0	No info	13882.0	9	6	2019	4	25	9	25
3	IndiGo	Kolkata	Banglore	5h 25m	1.0	No info	6218.0	12	5	2019	23	30	18	5
4	IndiGo	Banglore	New Delhi	4h 45m	1.0	No info	13302.0	1	3	2019	21	35	16	50

In [221]:

```
final_df.Additional_Info.value_counts()
```

Out[221]:

```
No info                         10493
In-flight meal not included      2426
No check-in baggage included      396
1 Long layover                     20
Change airports                     8
Business class                      5
No Info                             3
1 Short layover                     1
Red-eye flight                      1
2 Long layover                      1
Name: Additional_Info, dtype: int64
```

In [243]:

```

```

Out[243]:

```
Int64Index([6474, 2660], dtype='int64')
```

In [244]:

```
final_df.drop(index=final_df[final_df['Duration']=='5m'].index,axis =0,inplace =True)
```

In [237]:

```
final_df['Duration'].str.split(' ').str[1].str.split('m').str[0].fillna(0).astype(int)
```

Out[237]:

```
0       50
1       25
2        0
3       25
4       45
        ..
2666    55
2667    35
2668    35
2669    15
2670    20
Name: Duration, Length: 13354, dtype: int32
```

In [245]:

```
#convert duration in minutes
final_df['Duration_mins'] = final_df['Duration'].str.split(' ').str[0].str.split('h').str[0].astype(int)*60+final_df['Duration'].s
tr.split(' ').str[1].str.split('m').str[0].fillna(0).astype(int)
```

In [246]:

```
final_df.head()
```

Out[246]:

	Airline	Source	Destination	Duration	Total_Stops	Additional_Info	Price	Date	Month	Year	Arrival_hour	Arrival_min	Dep_hour	Dep_min	Duration_mins
0	IndiGo	Banglore	New Delhi	2h 50m	0.0	No info	3897.0	24	3	2019	1	10	22	20	170
1	Air India	Kolkata	Banglore	7h 25m	2.0	No info	7662.0	1	5	2019	13	15	5	50	445
2	Jet Airways	Delhi	Cochin	19h	2.0	No info	13882.0	9	6	2019	4	25	9	25	1140
3	IndiGo	Kolkata	Banglore	5h 25m	1.0	No info	6218.0	12	5	2019	23	30	18	5	325
4	IndiGo	Banglore	New Delhi	4h 45m	1.0	No info	13302.0	1	3	2019	21	35	16	50	285

In [247]:

```
final_df.drop('Duration',axis=1, inplace =True)
```

In [235]:

```
final_df['Duration'].str.split(' ').str[1].str.split('m').str[0].fillna(0).astype(int)
```

Out[235]:

```
0       50
1       25
2        0
3       25
4       45
        ..
2666    55
2667    35
2668    35
2669    15
2670    20
Name: Duration, Length: 13354, dtype: int32
```

In [248]:

```
final_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 13351 entries, 0 to 2670
Data columns (total 14 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   Airline          13351 non-null  object
 1   Source           13351 non-null  object
 2   Destination      13351 non-null  object
 3   Total_Stops      13350 non-null  float64
```

```
 3   Total_Stops      13350 non-null  float64
 4   Additional_Info  13351 non-null  object
 5   Price            10681 non-null  float64
 6   Date             13351 non-null  int32
 7   Month            13351 non-null  int32
 8   Year             13351 non-null  int32
 9   Arrival_hour     13351 non-null  int32
 10  Arrival_min      13351 non-null  int32
 11  Dep_hour         13351 non-null  int32
 12  Dep_min          13351 non-null  int32
 13  Duration_mins    13351 non-null  int32
dtypes: float64(2), int32(8), object(4)
memory usage: 1.1+ MB
```

In [249]:

```
final_df['Airline'].unique()
```

Out[249]:

```
array(['IndiGo', 'Air India', 'Jet Airways', 'SpiceJet',
       'Multiple carriers', 'GoAir', 'Vistara', 'Air Asia',
       'Vistara Premium economy', 'Jet Airways Business',
       'Multiple carriers Premium economy', 'Trujet'], dtype=object)
```

In [250]:

```
from sklearn.preprocessing import LabelEncoder
labelencoder= LabelEncoder()
```

In [251]:

```
final_df['Airline'] = labelencoder.fit_transform(final_df['Airline'] )
```

In [252]:

```
final_df.head()
```

Out[252]:

	Airline	Source	Destination	Total_Stops	Additional_Info	Price	Date	Month	Year	Arrival_hour	Arrival_min	Dep_hour	Dep_min	Duration_mins
0	3	Banglore	New Delhi	0.0	No info	3897.0	24	3	2019	1	10	22	20	170
1	1	Kolkata	Banglore	2.0	No info	7662.0	1	5	2019	13	15	5	50	445
2	4	Delhi	Cochin	2.0	No info	13882.0	9	6	2019	4	25	9	25	1140
3	3	Kolkata	Banglore	1.0	No info	6218.0	12	5	2019	23	30	18	5	325
4	3	Banglore	New Delhi	1.0	No info	13302.0	1	3	2019	21	35	16	50	285

In [254]:

```
final_df['Source'] = labelencoder.fit_transform(final_df['Source'] )
final_df['Destination'] = labelencoder.fit_transform(final_df['Destination'] )
final_df['Additional_Info'] = labelencoder.fit_transform(final_df['Additional_Info'] )
```

In [255]:

```
final_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 13351 entries, 0 to 2670
Data columns (total 14 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   Airline          13351 non-null  int32
 1   Source           13351 non-null  int32
 2   Destination      13351 non-null  int32
 3   Total_Stops      13350 non-null  float64
 4   Additional_Info  13351 non-null  int32
 5   Price            10681 non-null  float64
 6   Date             13351 non-null  int32
 7   Month            13351 non-null  int32
 8   Year             13351 non-null  int32
 9   Arrival_hour     13351 non-null  int32
 10  Arrival_min      13351 non-null  int32
 11  Dep_hour         13351 non-null  int32
 12  Dep_min          13351 non-null  int32
 13  Duration_mins    13351 non-null  int32
dtypes: float64(2), int32(12)
memory usage: 938.7 KB
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: