

Zomato Dataset Practical EDA Implementation

Submitted by : Ambarish Singh

In []:

```
#comment
#observations
```

In [98]:

```
## Importing Basic Libraries:-
```

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline      ## so that the images or vizualization are display in notebook
import warnings
warnings.filterwarnings('ignore')
```

UsageError: unrecognized arguments: # so that the images or vizualization are display in notebook itself.

In [6]:

```
## Loading Zomato Dataset in Dataframe
df = pd.read_csv('zomato.csv', encoding = 'latin-1')
```

In [9]:

```
## Head() used to Display top 5 Rows of a dataset.
df.head()
```

Out[9]:

	Restaurant ID	Restaurant Name	Country Code	City	Address	Locality	Locality Verbose	Longitude
0	6317637	Le Petit Souffle	162	Makati City	Third Floor, Century City Mall, Kalayaan Avenue...	Century City Mall, Poblacion, Makati City	Century City Mall, Poblacion, Makati City, Mak...	121.027535 1
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 1

	Restaurant ID	Restaurant Name	Country Code	City	Address	Locality	Locality Verbose	Longitude
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 1
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 1
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 1

5 rows × 21 columns



```
In [10]: ## Checking All Column Available in a Dataset
df.columns
```

```
Out[10]: 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 [11]: ## one more way to understand a dataset is by df.info()
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   LocalityVerbose   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
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 [12]:

```
## Describes() is used to check all Details of Statistics Method Related with ALL Numerical Variables
df.describe()
```

Out[12]:

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

In [14]:

```
## To check all Rows And Columns available in Dataset
df.shape
```

Out[14]:

(9551, 21)

In Data Analysis Whats All Things We Do:-

1. Missing Values
2. Explore about the Numerical Variables
3. Explore About the categorical Variables
4. Finding Relationship Between Features.

In [15]:

```
## In Order to find the sum of Missing Value in respect to all Columns
df.isnull().sum()
```

```
Out[15]: Restaurant ID      0  
          Restaurant Name    0  
          Country Code       0  
          City                0  
          Address              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 [18]: ## Another ways to find the Missing values by using List Comprehension.  
[features for features in df.columns if df[features].isnull().sum() >0]
```

```
Out[18]: ['Cuisines']
```

```
In [101...]: ## Another way of Finding Missing value is by using Heatmap.  
sns.heatmap(df.isnull(),yticklabels=False, cbar= False, cmap= 'viridis')
```

```
Out[101...]: <AxesSubplot:>
```

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	-

In [24]:

```
## Loading Another dataset in a dataframe:-  
df_contry= pd.read_excel('Country-Code.xlsx')
```

In [25]:

```
## For checking top 5 Row from a dataset  
df_contry.head()
```

Out[25]:

	Country Code	Country
0	1	India
1	14	Australia
2	30	Brazil
3	37	Canada
4	94	Indonesia

	Country Code	Country
0	1	India
1	14	Australia
2	30	Brazil
3	37	Canada
4	94	Indonesia

In [26]:

```
## For checking all columns Names Available in a dataset.  
df.columns
```

Out[26]:

```
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 [33]:

```
## pd.merge() is used to Merge Two Dataframe in one
final_df = pd.merge(df, df_contry, on = 'Country Code', how ='left')
```

In [34]:

```
## for Checking top 3 rows from a dataset
final_df.head(3)
```

Out[34]:

	Restaurant ID	Restaurant Name	Country Code	City	Address	Locality	Locality Verbose	Longitude	L
0	6317637	Le Petit Souffle	162	Makati City	Third Floor, Century City Mall, Kalayaan Avenue...	Century City Mall, Poblacion, Makati City	Century City Mall, Poblacion, Makati City, Mak...	121.027535	14
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
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

3 rows × 22 columns

In [35]:

```
## Another way to check a data types.
final_df.dtypes
```

Out[35]:

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 [36]: *## For Checking All Column Name Available in Dataset.*
`final_df.columns`

Out[36]: `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 [37]: *## With Respect to Country columns ,it is used to find totol value counts.*
`final_df.Country.value_counts()`

Out[37]:

India	8652
United States	434
United Kingdom	80
Brazil	60
UAE	60
South Africa	60
New Zealand	40
Turkey	34
Australia	24
Phillipines	22
Indonesia	21
Singapore	20
Qatar	20
Sri Lanka	20
Canada	4

Name: Country, dtype: int64

Observation :-

- Zamoto is mostly Available in India.

In [38]: *## In this Country Value Counted with the help of Indexing.*
`country_names = final_df.Country.value_counts().index`

In [39]: *## Displaying the Value Store in Contry_name variables.*
`country_names`

Out[39]: `Index(['India', 'United States', 'United Kingdom', 'Brazil', 'UAE',
 'South Africa', 'New Zealand', 'Turkey', 'Australia', 'Phillipines',
 'Indonesia', 'Singapore', 'Qatar', 'Sri Lanka', 'Canada'],
 dtype='object')`

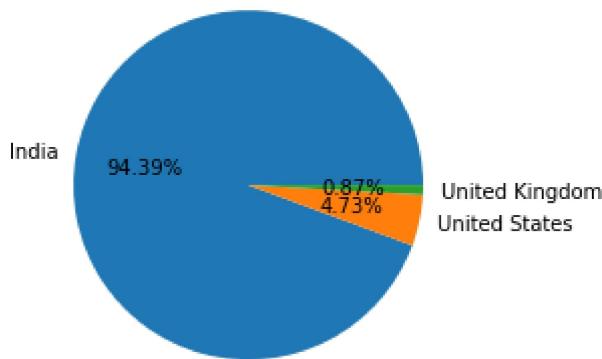
In [41]: *## In this Country Value Counted with the help of Value function.*
`country_val = final_df.Country.value_counts().values`

In [42]: *## Displaying the Value Store in Country_val variables.*
country_val

Out[42]: array([8652, 434, 80, 60, 60, 60, 40, 34, 24, 22, 21, 20, 20, 20, 4], dtype=int64)

In [47]: *## Plotting Pie chart for - Top 3 countries that use zomato.*
plt.pie(country_val[:3], labels = country_names[:3], autopct = '%1.2f%%')

Out[47]: ([



Observation :-

- Zomato maximum records or transaction are from India after that USA And then United Kingdoms

In [48]: *## Checking All the column name Available in a Dataset.*
final_df.columns

Out[48]: 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 [52]: *## We are Grouping the columns 'Aggregate rating', 'Rating color', 'Rating text' By using.*
final_df.groupby(['Aggregate rating', 'Rating color', 'Rating text']).size()

Out[52]:

Aggregate rating	Rating color	Rating text	Count
0.0	White	Not rated	2148
1.8	Red	Poor	1

```

1.9          Red      Poor       2
2.0          Red      Poor       7
2.1          Red      Poor      15
2.2          Red      Poor      27
2.3          Red      Poor      47
2.4          Red      Poor      87
2.5        Orange    Average   110
2.6        Orange    Average   191
2.7        Orange    Average   250
2.8        Orange    Average   315
2.9        Orange    Average   381
3.0        Orange    Average   468
3.1        Orange    Average   519
3.2        Orange    Average   522
3.3        Orange    Average   483
3.4        Orange    Average   498
3.5        Yellow    Good      480
3.6        Yellow    Good      458
3.7        Yellow    Good      427
3.8        Yellow    Good      400
3.9        Yellow    Good      335
4.0        Green     Very Good  266
4.1        Green     Very Good  274
4.2        Green     Very Good  221
4.3        Green     Very Good  174
4.4        Green     Very Good  144
4.5      Dark Green Excellent  95
4.6      Dark Green Excellent  78
4.7      Dark Green Excellent  42
4.8      Dark Green Excellent  25
4.9      Dark Green Excellent  61

```

dtype: int64

In [53]:

```
## Adding New column at Last as 'Rating count' by Renaming header 0 = 'Rating Count'
ratings = final_df.groupby(['Aggregate rating', 'Rating color', 'Rating text']).size().
```

In [54]:

```
## Displaying all the value which are store in rating variable.
ratings
```

Out[54]:

	Aggregate rating	Rating color	Rating text	Rating count
0	0.0	White	Not rated	2148
1	1.8	Red	Poor	1
2	1.9	Red	Poor	2
3	2.0	Red	Poor	7
4	2.1	Red	Poor	15
5	2.2	Red	Poor	27
6	2.3	Red	Poor	47
7	2.4	Red	Poor	87
8	2.5	Orange	Average	110
9	2.6	Orange	Average	191

	Aggregate rating	Rating color	Rating text	Rating count
10	2.7	Orange	Average	250
11	2.8	Orange	Average	315
12	2.9	Orange	Average	381
13	3.0	Orange	Average	468
14	3.1	Orange	Average	519
15	3.2	Orange	Average	522
16	3.3	Orange	Average	483
17	3.4	Orange	Average	498
18	3.5	Yellow	Good	480
19	3.6	Yellow	Good	458
20	3.7	Yellow	Good	427
21	3.8	Yellow	Good	400
22	3.9	Yellow	Good	335
23	4.0	Green	Very Good	266
24	4.1	Green	Very Good	274
25	4.2	Green	Very Good	221
26	4.3	Green	Very Good	174
27	4.4	Green	Very Good	144
28	4.5	Dark Green	Excellent	95
29	4.6	Dark Green	Excellent	78
30	4.7	Dark Green	Excellent	42
31	4.8	Dark Green	Excellent	25
32	4.9	Dark Green	Excellent	61

Observation

1. When Rating is between 4.5 to 4.9 ---> Excellent
2. When Rating is between 4.0 to 4.4 ---> Very Good
3. When Rating is between 3.5 to 3.9 ---> Good
4. When Rating is between 3.0 to 3.4 ---> Average
5. When Rating is between 2.5 to 2.9 ---> Average
6. When Rating is between 2.0 to 2.4 ---> Poor

In [55]:

```
## Showing Top 5 Rows of a dataset.
ratings.head()
```

Out[55]:

	Aggregate rating	Rating color	Rating text	Rating count
0	0.0	White	Not rated	2148
1	1.8	Red	Poor	1
2	1.9	Red	Poor	2
3	2.0	Red	Poor	7
4	2.1	Red	Poor	15

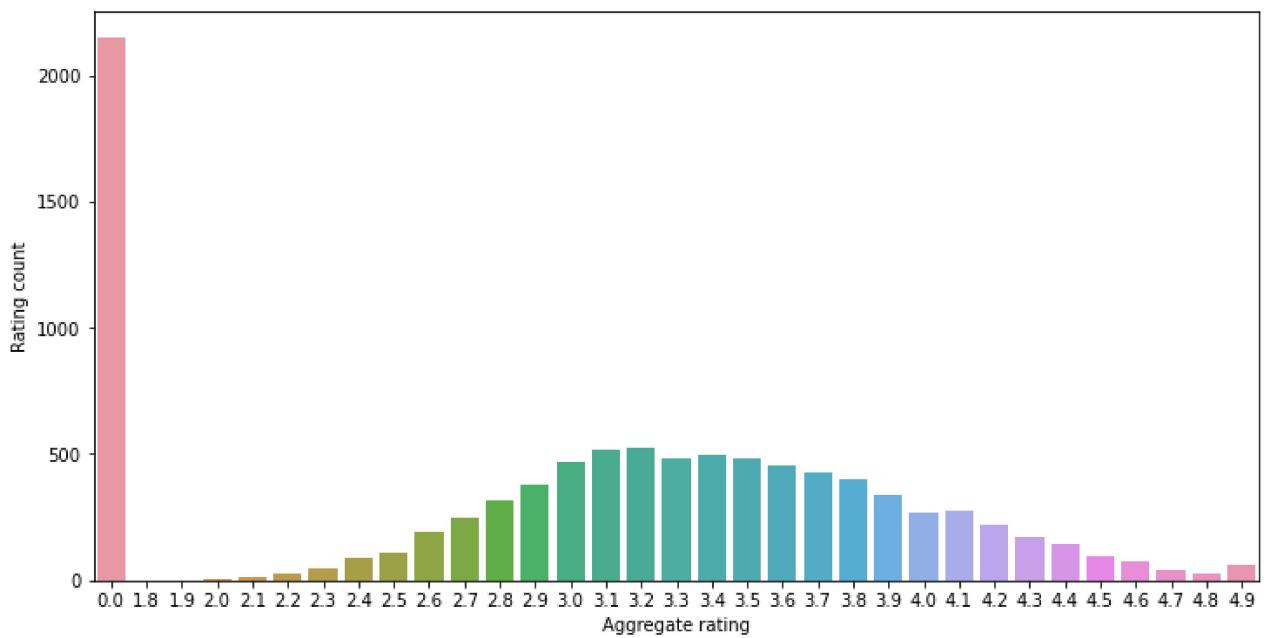
In [71]:

```
## Plotting Bar Plot Chart for x = 'Aggregate rating', y = 'Rating count' Values.

import matplotlib
matplotlib.rcParams['figure.figsize'] = (12,6)
sns.barplot(x = 'Aggregate rating', y = 'Rating count', data= ratings)
sns.barplot(x = 'Aggregate rating', y = 'Rating count', data= ratings)
```

Out[71]:

<AxesSubplot:xlabel='Aggregate rating', ylabel='Rating count'>

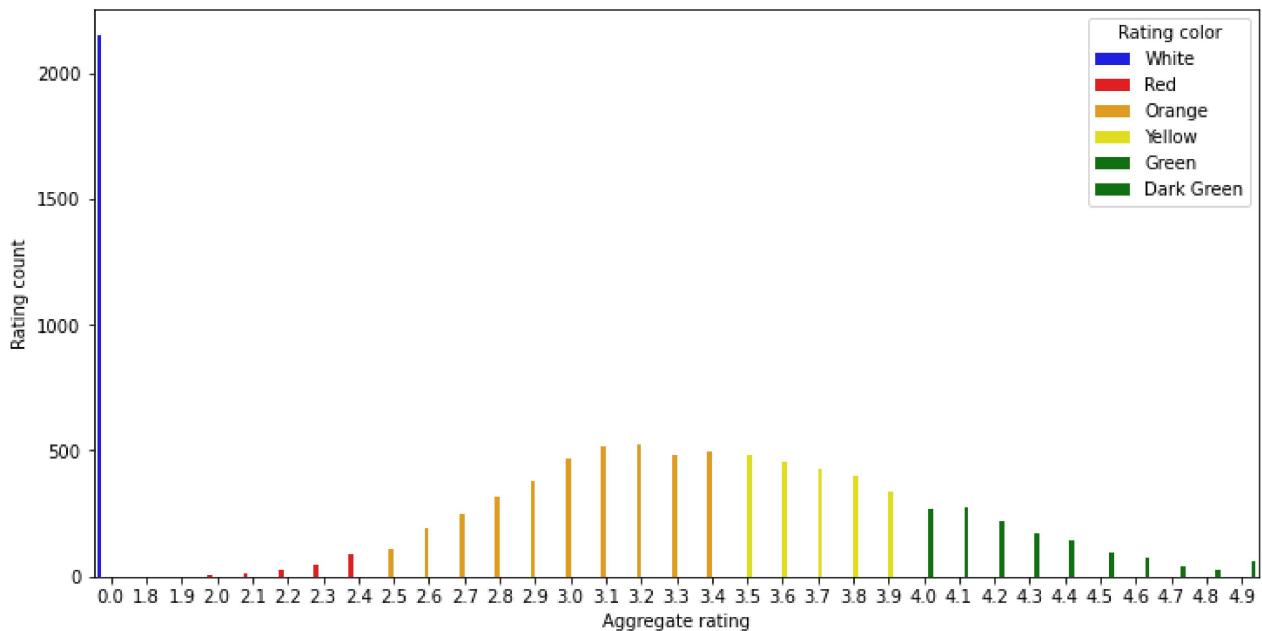


In [70]:

```
## Adjusting Bar Plot Layout with the help hue() & palette() properties.
sns.barplot(x = 'Aggregate rating', y = 'Rating count',hue ='Rating color' ,data= ratings)
```

Out[70]:

<AxesSubplot:xlabel='Aggregate rating', ylabel='Rating count'>



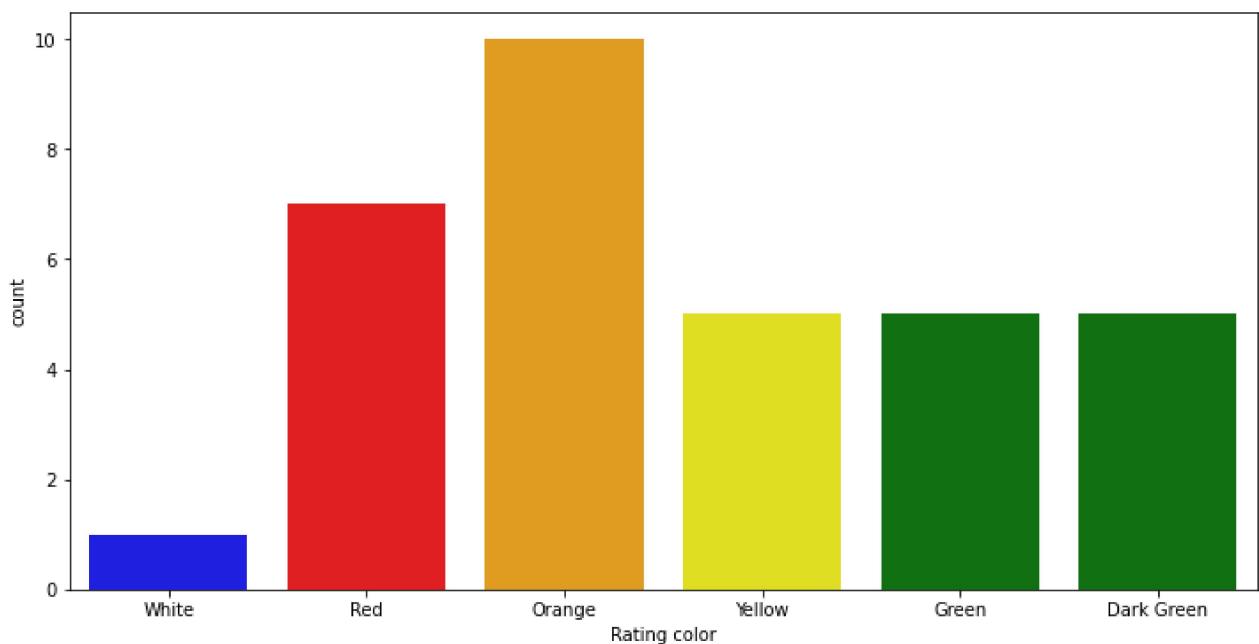
Observation:-

- Not Rated count is very High.
- Maximum Number of rating are between 2.5 to 3.4

In [66]:

```
## Plotting Count Plot Bar
sns.countplot(x='Rating color', data = ratings, palette =['blue', 'red','orange','yellow','green','darkgreen'])
```

Out[66]:



Find the Countries name that has given 0 rating ?

In [73]:

```
## Showing All the columns name Available in the dataset.
final_df.columns
```

```
Out[73]: 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 [80]: *## Solution for Above Questions :-*
`final_df.groupby(['Aggregate rating', 'Country']).size().reset_index().head(5)`

Out[80]:

	Aggregate rating	Country	0
0	0.0	Brazil	5
1	0.0	India	2139
2	0.0	United Kingdom	1
3	0.0	United States	3
4	1.8	India	1

Observations:-

- Maximum Number of 0 Ratings are from Indian customers

Find Out which currency is used by which country ?

In [81]: *## Showing All the columns name Available in the dataset.*
`final_df.columns`

Out[81]:

```
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 [85]: *## Finding Currency for Country by using Groupby()*
`final_df[['Country', 'Currency']].groupby(['Country', 'Currency']).size().reset_index()`

Out[85]:

	Country	Currency	0
0	Australia	Dollar(\$)	24
1	Brazil	Brazilian Real(R\$)	60
2	Canada	Dollar(\$)	4
3	India	Indian Rupees(Rs.)	8652
4	Indonesia	Indonesian Rupiah(IDR)	21
5	New Zealand	NewZealand(\$)	40
6	Phillipines	Botswana Pula(P)	22

	Country	Currency	0
7	Qatar	Qatari Rial(QR)	20
8	Singapore	Dollar(\$)	20
9	South Africa	Rand(R)	60
10	Sri Lanka	Sri Lankan Rupee(LKR)	20
11	Turkey	Turkish Lira(TL)	34
12	UAE	Emirati Diram(AED)	60
13	United Kingdom	Pounds(£)	80
14	United States	Dollar(\$)	434

Which Country Do have an online Deliveries option ?

In [86]:

```
## Country which has online delivery Available are Listed below:-  
final_df[final_df['Has Online delivery'] == "Yes"].Country.value_counts()
```

Out[86]:

```
India    2423  
UAE      28  
Name: Country, dtype: int64
```

In [87]:

```
## Country which has online delivery Available or Which has Not Available both list are  
final_df[['Has Online delivery','Country']].groupby(['Has Online delivery','Country']).
```

Out[87]:

	Has Online delivery	Country	0
0	No	Australia	24
1	No	Brazil	60
2	No	Canada	4
3	No	India	6229
4	No	Indonesia	21
5	No	New Zealand	40
6	No	Phillipines	22
7	No	Qatar	20
8	No	Singapore	20
9	No	South Africa	60
10	No	Sri Lanka	20
11	No	Turkey	34
12	No	UAE	32
13	No	United Kingdom	80
14	No	United States	434

Has Online delivery	Country	0
15	Yes	India 2423
16	Yes	UAE 28

Observation:-

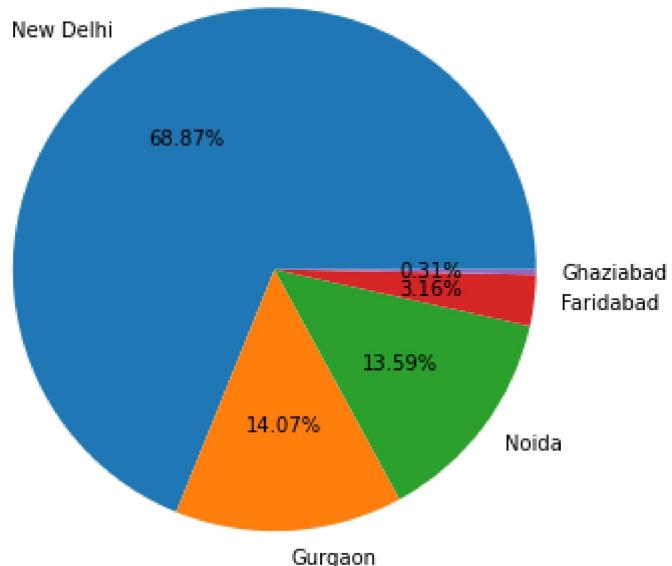
- Online Deliveries are available in India and UAE

Create a Pie chart for Cities Distribution

```
In [91]: ## Finding out all city Value Count and Value Label index.
city_values = final_df.City.value_counts().values
city_labels = final_df.City.value_counts().index
```

```
In [92]: ## Plotting the Pie Chart
plt.pie(city_values[:5], labels = city_labels[:5], autopct ='%1.2f%%')
```

```
Out[92]: ([<matplotlib.patches.Wedge at 0x29bf59e50a0>,
<matplotlib.patches.Wedge at 0x29bf59e5820>,
<matplotlib.patches.Wedge at 0x29bf59e5f40>,
<matplotlib.patches.Wedge at 0x29bf59f16a0>,
<matplotlib.patches.Wedge at 0x29bf59f1dc0>],
[Text(-0.6145352824185932, 0.9123301960708633, 'New Delhi'),
Text(0.0623675251198054, -1.0982305276263407, 'Gurgaon'),
Text(0.8789045225625368, -0.6614581167535246, 'Noida'),
Text(1.0922218418223437, -0.13058119407559224, 'Faridabad'),
Text(1.099946280005612, -0.010871113182029924, 'Ghaziabad')],
[Text(-0.3352010631374145, 0.497634652402289, '68.87%'),
Text(0.0340186500653484, -0.5990348332507311, '14.07%'),
Text(0.47940246685229276, -0.36079533641101336, '13.59%'),
Text(0.5957573682667329, -0.07122610585941394, '3.16%'),
Text(0.5999706981848791, -0.005929698099289049, '0.31%')])
```



Observations:-

- So, Number of transaction is mainly done by city 'New Delhi'.

Thank You