Zomato Dataset Exploratory Data Analysis

In [1]:
```
## import libraries

import pandas as pd                    ## Data Manipulation
import numpy as np                     ## Mathematical Calculations
import matplotlib.pyplot as plt        ## Data Visualization
import seaborn as sns                  ## Advanced Data Visualization
import warnings
warnings.filterwarnings('ignore')
```

In [98]:
```
##  upload the data file

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

df.head()
```

Out[98]:

	Restaurant ID	Restaurant Name	Country Code	City	Address	Locality	Locality Verbose	Longitude	La
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	14.5
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	14.5
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	14.5
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	14.5
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	14.5

5 rows × 21 columns

In [5]:
```
## now we will see the shape of the data set

df.shape
```
Out[5]:
```
(9551, 21)
```
In [6]:
```
## we see that there are 9551 rows and 21 columns in our data set

## let see the first 5 rows data

df.head()
```
Out[6]:

	Restaurant ID	Restaurant Name	Country Code	City	Address	Locality	Locality Verbose	Longitude	La
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.!
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	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.!
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.!

5 rows × 21 columns

In [8]:
```
## lets see the list of columns names

df.columns
```

Out[8]:
```
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 [9]:
```
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 
 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 [10]:
```
df.describe()
```

Out[10]:

	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	95
mean	9.051128e+06	18.365616	64.126574	25.854381	1199.210763	1.804837	2.666370	1
std	8.791521e+06	56.750546	41.467058	11.007935	16121.183073	0.905609	1.516378	4
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	1
max	1.850065e+07	216.000000	174.832089	55.976980	800000.000000	4.000000	4.900000	109

In [18]:
```
## check if there are any null values in data set

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

Out[18]:

```
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
```

Alternative way to we can check if there are any null or missing values

In [19]:

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

Out[19]:

```
['Cuisines']
```

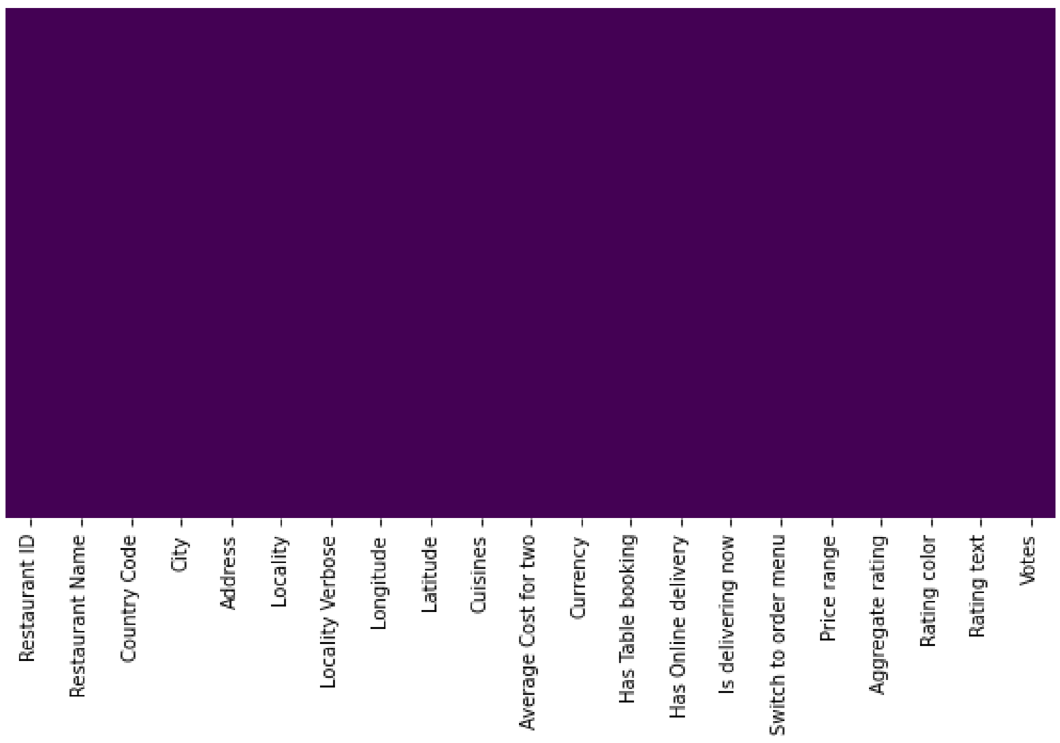
now we will see the heat-map for Cuisines

In [97]:

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

Out[97]:

```
<AxesSubplot: >
```



now we will import Country Code data set

In [24]:

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

df_country.head()
```

Out[24]:

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

In [25]:

```
df_country.shape
```

Out[25]:

```
(15, 2)
```

In [26]:

```
df_country.columns
```

Out[26]:

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

In [27]:

```
df.columns
```

Out[27]:

```
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')
```

We see that Country Code is common in both the data set now lets us merge them by using "Merge" function

In [30]:

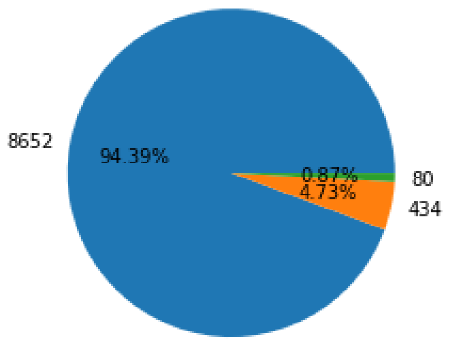
```
final_df = pd.merge(df, df_country, on='Country Code', how='left')
final_df.head(2)
```

Out[30]:

	Restaurant ID	Restaurant Name	Country Code	City	Address	Locality	Locality Verbose	Longitude	Latitude	Cuisi
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	Fren Japane Desse
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	Japan

2 rows × 22 columns

In [31]:
```
## now lets see the shape of final_df shape

final_df.shape
```
Out[31]:
```
(9551, 22)
```
In [32]:
```
final_df.dtypes
```
Out[32]:
```
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 [33]:
```
final_df.columns
```
Out[33]:
```
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 [34]:
```
country_names = final_df.Country.value_counts().index
```
In [35]:
```
country_val = final_df.Country.value_counts().values
```
In [37]:
```
country_names = final_df.Country.value_counts()
```
```
  Input In [37]
    country_names
    ^
SyntaxError: invalid syntax
```
In [38]:
```
## Pie Chart- Top 3 countries that uses zomato

plt.pie(country_val[:3], labels = country_names[:3], autopct = '%1.2f%%')
```
Out[38]:
```
([<matplotlib.patches.Wedge at 0x176a7025310>,
  <matplotlib.patches.Wedge at 0x176a70252e0>,
  <matplotlib.patches.Wedge at 0x176a7078220>],
 [Text(-1.0829742700952103, 0.19278674827836725, '8652'),
  Text(1.077281715838356, -0.22240527134123297, '434'),
  Text(1.0995865153823035, -0.03015783794312073, '80')],
 [Text(-0.590713238233751, 0.10515640815183668, '94.39%'),
  Text(0.5876082086391032, -0.12131196618612707, '4.73%'),
  Text(0.5997744629358018, -0.01644972978715676, '0.87%')])
```



Observation:Zomato maximum records or transaction are from India After that USA and then United Kingdoms

```
In [40]: final_df.columns
```

```
Out[40]: 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 [41]: ratings = final_df.groupby(['Aggregate rating', 'Rating color', 'Rating text']).size().re
```

```
In [42]: ratings
```

Out[42]:

	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
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

Observations

When Rating is between 4.5 to 4.9---> Excellent

When Rating are between 4.0 to 3.4--->very good when Rating is between 3.5 to 3.9----> good

when Rating is between 3.0 to 3.4----> average

when Rating is between 2.5 to 2.9----> average

when Rating is between 2.0 to 2.4----> Poor

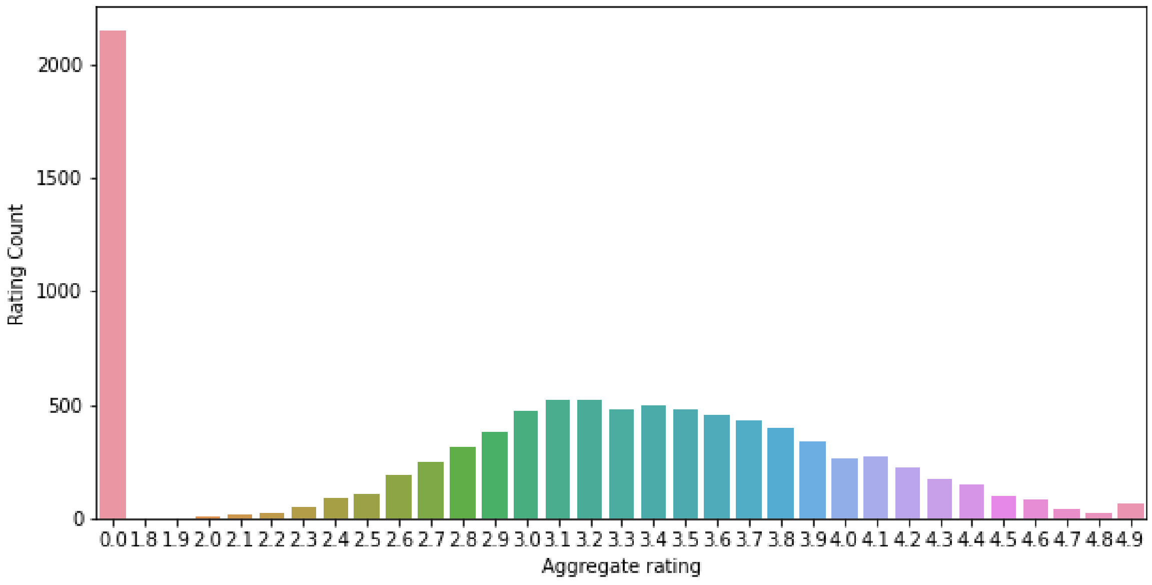
In [43]:
```
ratings.head()
```

Out[43]:

	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

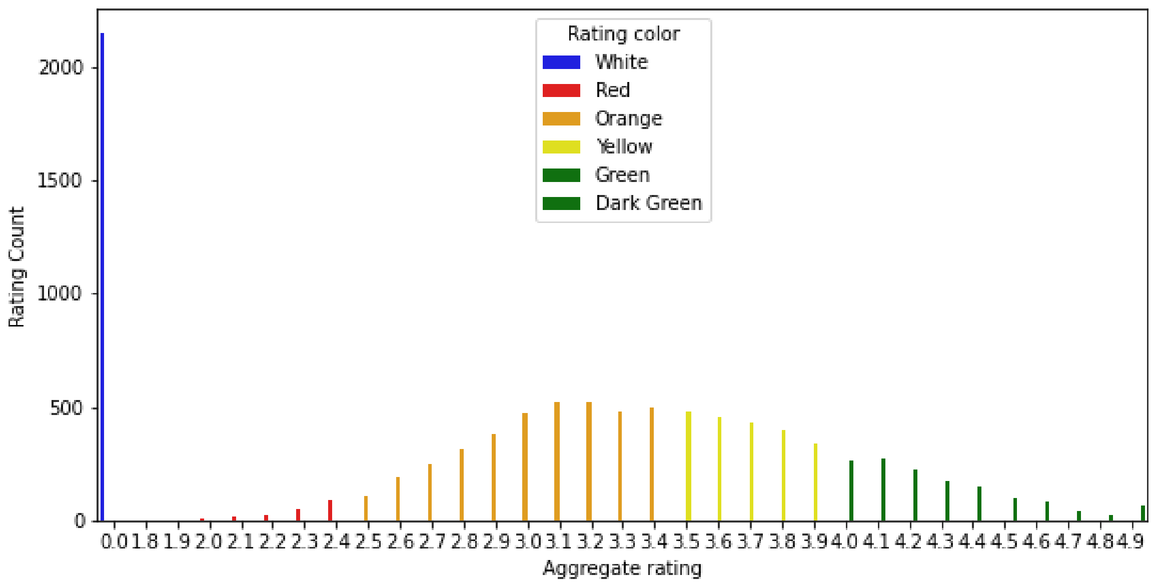
In [52]:
```
import matplotlib

matplotlib.rcParams['figure.figsize'] = (10, 5)

sns.barplot(x = "Aggregate rating", y = "Rating Count", data = ratings)
```

Out[52]:
```
<AxesSubplot: xlabel='Aggregate rating', ylabel='Rating Count'>
```



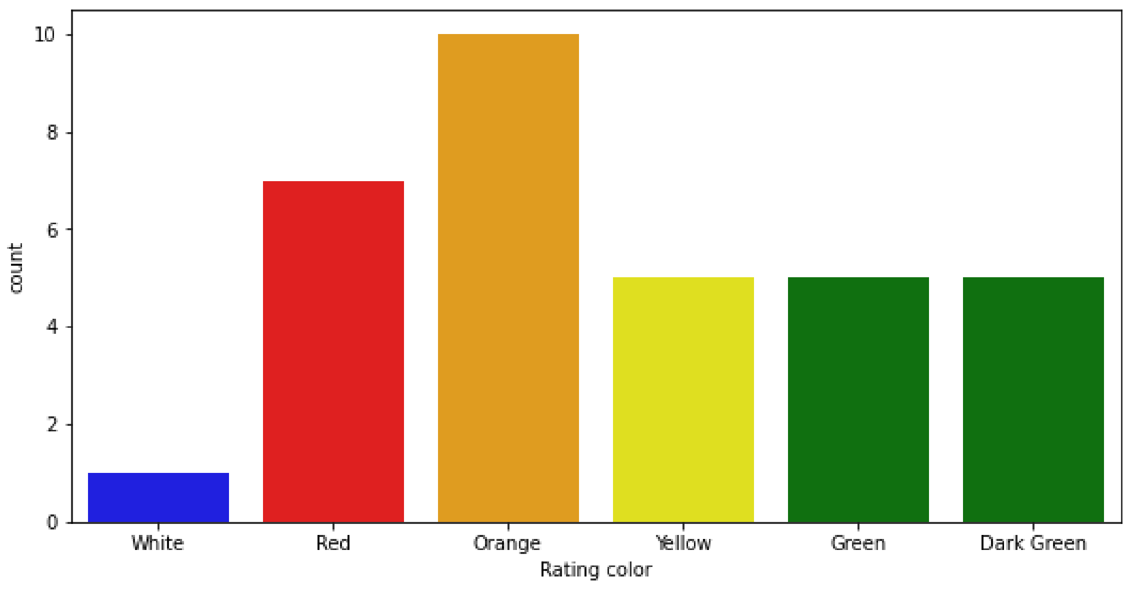
In [53]:
```
sns.barplot(x = "Aggregate rating", y = "Rating Count", hue = 'Rating color', data = rati
```

Out[53]:
```
<AxesSubplot: xlabel='Aggregate rating', ylabel='Rating Count'>
```



```
In [54]: ## Count plot

         sns.countplot(x = "Rating color", data = ratings, palette = ['blue','red','orange','yello
```

```
Out[54]: <AxesSubplot: xlabel='Rating color', ylabel='count'>
```



```
In [55]: ratings
```

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
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
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

In [56]:

```python
### Find the countries name that has given 0 rating

final_df[final_df['Rating color']=='White'].groupby('Country').size()
```

Out[56]:
```
Country
Brazil               5
India             2139
United Kingdom       1
United States        3
dtype: int64
```

In [57]:
```
final_df[final_df['Rating color']=='White'].groupby('Country')
```

Out[57]:
```
<pandas.core.groupby.generic.DataFrameGroupBy object at 0x00000176A94420A0>
```

In [58]:
```
final_df[final_df['Rating color']=='White'].groupby('Country').size().reset_index()
```

Out[58]:

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

In [59]:
```
final_df.groupby(['Aggregate rating','Country'])
```

Out[59]:
```
<pandas.core.groupby.generic.DataFrameGroupBy object at 0x00000176AB3E5820>
```

In [60]:
```
final_df.groupby(['Aggregate rating', 'Country']).size()
```

Out[60]:
```
Aggregate rating  Country
0.0               Brazil               5
                  India             2139
                  United Kingdom       1
                  United States        3
1.8               India                1
                                    ...
4.9               Sri Lanka            1
                  Turkey               3
                  UAE                  4
                  United Kingdom       4
                  United States       14
Length: 222, dtype: int64
```

In [61]:
```
final_df.groupby(['Aggregate rating', 'Country']).size().reset_index().head(5)
```

Out[61]:

	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

In [62]:
```
##find out which currency is used by which country?

final_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 [63]:
```
final_df[['Country','Currency']].groupby(['Country','Currency']).size().reset_index().hea
```

Out[63]:

	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

In [64]:
```
## Which Countries do have online deliveries option

final_df[final_df['Has Online delivery'] =="Yes"].Country.value_counts()
```

Out[64]:
```
India    2423
UAE        28
Name: Country, dtype: int64
```

In [65]:
```
final_df[['Has Online delivery','Country']].groupby(['Has Online delivery','Country']).si
```

Out[65]:

	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
15	Yes	India	2423
16	Yes	UAE	28

In [66]:
```
## Create a pie chart for top 5 cities distribution

final_df.City.value_counts()
```

```
Out[66]: New Delhi           5473
         Gurgaon             1118
         Noida               1080
         Faridabad            251
         Ghaziabad             25
                             ...
         Panchkula              1
         Mc Millan              1
         Mayfield               1
         Macedon                1
         Vineland Station       1
         Name: City, Length: 141, dtype: int64
```

```
In [68]: final_df.City.value_counts().index
```

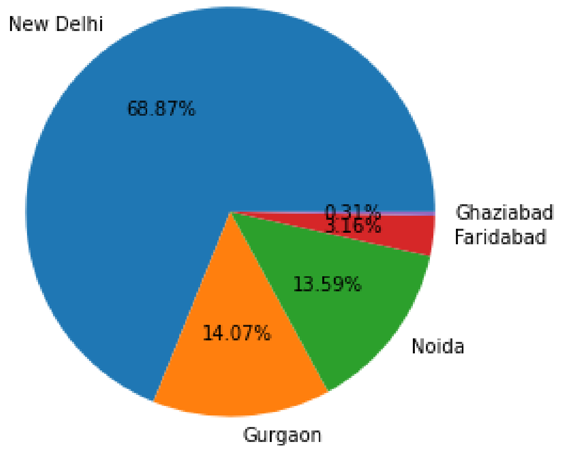
```
Out[68]: Index(['New Delhi', 'Gurgaon', 'Noida', 'Faridabad', 'Ghaziabad',
                'Bhubaneshwar', 'Amritsar', 'Ahmedabad', 'Lucknow', 'Guwahati',
                ...
                'Ojo Caliente', 'Montville', 'Monroe', 'Miller', 'Middleton Beach',
                'Panchkula', 'Mc Millan', 'Mayfield', 'Macedon', 'Vineland Station'],
               dtype='object', length=141)
```

```
In [69]: city_values=final_df.City.value_counts().values
```

```
In [70]: city_labels=final_df.City.value_counts().index
```

```
In [71]: plt.pie(city_values[:5] , labels = city_labels[:5] , autopct='%1.2f%%')
```

```
Out[71]: ([<matplotlib.patches.Wedge at 0x176ab3e9880>,
           <matplotlib.patches.Wedge at 0x176ab3e9610>,
           <matplotlib.patches.Wedge at 0x176ab37a850>,
           <matplotlib.patches.Wedge at 0x176ab37a760>,
           <matplotlib.patches.Wedge at 0x176abfa4490>],
          [Text(-0.6145352824185932, 0.912330196070863З, '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%')])
```



```
In [73]: final_df.columns
```

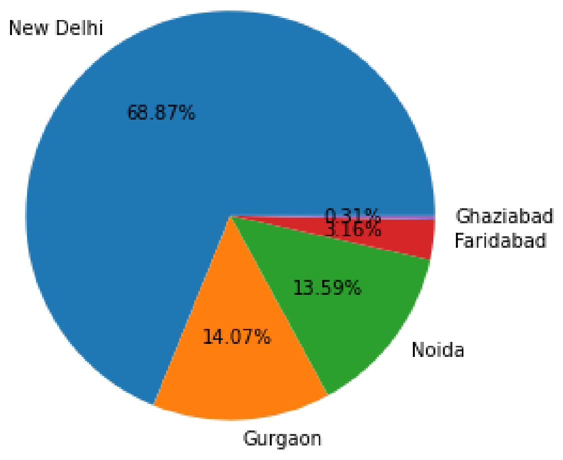
```
Out[73]: Index(['Restaurant ID', 'Restaurant Name', 'Country Code', 'City', 'Address',
               'Locality', 'Locality Verbose', 'Longitude', 'Latitude', 'Cuisines',
               'Average Cost for two', 'Currency', 'Has Table booking',
               'Has Online delivery', 'Is delivering now', 'Switch to order menu',
               'Price range', 'Aggregate rating', 'Rating color', 'Rating text',
               'Votes', 'Country'],
              dtype='object')
```

```
In [74]: final_df.Cuisines.value_counts().index
```

```
Out[74]: Index(['North Indian', 'North Indian, Chinese', 'Chinese', 'Fast Food',
               'North Indian, Mughlai', 'Cafe', 'Bakery',
               'North Indian, Mughlai, Chinese', 'Bakery, Desserts', 'Street Food',
               ...
               'Cafe, Pizza, Burger',
               'Healthy Food, Continental, Juices, Beverages, Italian, Salad, Lebanese',
               'Goan, American, Portuguese', 'South Indian, Desserts, Beverages',
               'Healthy Food, North Indian, Italian, Salad', 'Bengali, Fast Food',
               'North Indian, Rajasthani, Asian',
               'Chinese, Thai, Malaysian, Indonesian',
               'Bakery, Desserts, North Indian, Bengali, South Indian',
               'Italian, World Cuisine'],
              dtype='object', length=1825)
```

```
In [95]: plt.pie(city_values[:5] , labels = city_labels[:5] , autopct='%1.2f%%')
```

```
Out[95]: ([<matplotlib.patches.Wedge at 0x1769a01e0d0>,
           <matplotlib.patches.Wedge at 0x1769a01e790>,
           <matplotlib.patches.Wedge at 0x1769a01ee20>,
           <matplotlib.patches.Wedge at 0x1769a02a4f0>,
           <matplotlib.patches.Wedge at 0x1769a02ab80>],
          [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%')])
```



```
In [ ]:
```