Exploratory Data Analysis

Analysis 1

In [1]:
```
#food delivery data (zomato)
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline

data = pd.read_csv("D:/zomato.csv",encoding='latin1')
data
```

Out[1]:

	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	Price range	Aggregate rating	Rating color	Rating text	Votes
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	3	4.8	Dark Green	Excellent	314
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	3	4.5	Dark Green	Excellent	591
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	4	4.4	Green	Very Good	270
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	...	Botswana Pula(P)	No	No	No	No	4	4.9	Dark Green	Excellent	365
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	4	4.8	Dark Green	Excellent	229
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
9546	5915730	NamlÛ± Gurme	208	ÛÁstanbul	KemankeÐõ Karamustafa PaÐóa Mahallesi, RÛ±htÛ±...	Karaki_y	Karaki_y, ÛÁstanbul	28.977392	41.022793	Turkish	...	Turkish Lira(TL)	No	No	No	No	3	4.1	Green	Very Good	788
9547	5908749	Ceviz AÛöacÛ±	208	ÛÁstanbul	KoÐóuyolu Mahallesi, Muhittin iist_ndaÛö Cadd...	KoÐóuyolu	KoÐóuyolu, ÛÁstanbul	29.041297	41.009847	World Cuisine, Patisserie, Cafe	...	Turkish Lira(TL)	No	No	No	No	3	4.2	Green	Very Good	1034
9548	5915807	Huqqa	208	ÛÁstanbul	Kurui_eÐóme Mahallesi, Muallim Naci Caddesi, N...	Kurui_eÐóme	Kurui_eÐóme, ÛÁstanbul	29.034640	41.055817	Italian, World Cuisine	...	Turkish Lira(TL)	No	No	No	No	4	3.7	Yellow	Good	661
9549	5916112	AÐõÐök Kahve	208	ÛÁstanbul	Kurui_eÐóme Mahallesi, Muallim Naci Caddesi, N...	Kurui_eÐóme	Kurui_eÐóme, ÛÁstanbul	29.036019	41.057979	Restaurant Cafe	...	Turkish Lira(TL)	No	No	No	No	4	4.0	Green	Very Good	901
9550	5927402	Walter's Coffee Roastery	208	ÛÁstanbul	CafeaÛöa Mahallesi, BademaltÛ± Sokak, No 21/B,...	Moda	Moda, ÛÁstanbul	29.026016	40.984776	Cafe	...	Turkish Lira(TL)	No	No	No	No	2	4.0	Green	Very Good	591

9551 rows × 21 columns

In [2]:
```
data.head()
```

Out[2]:

	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	Price range	Aggregate rating	Rating color	Rating text	Votes
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	3	4.8	Dark Green	Excellent	314
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	3	4.5	Dark Green	Excellent	591
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	4	4.4	Green	Very Good	270
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	...	Botswana Pula(P)	No	No	No	No	4	4.9	Dark Green	Excellent	365
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	4	4.8	Dark Green	Excellent	229

5 rows × 21 columns

In [3]:
```
#to get the columns name
data.columns
```

Out[3]:
```
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 [4]:
```
data.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 [5]:
```
data.describe()
#it gives only numerical values not categorical
```

Out[5]:

	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 [6]:
```
#shape
data.shape
```

Out[6]:
```
(9551, 21)
```

In [7]:
```
#missing values
#find numerical and categorical
```

In [ ]:
```
#find relationships between the features

#now finding the ,whether null value is there or not
data.isnull().sum()
```

Out[7]:
```
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
```

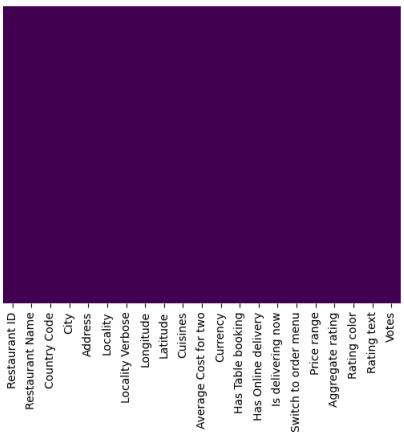
In [8]:
```
#finding null value using list comprehensions
[features for features in data.columns if data[features].isnull().sum()>1]
```

Out[8]:
```
['Cuisines']
```

In [9]:
```
#heatmap
sns.heatmap(data.isnull(),yticklabels=False,cbar=False,cmap="viridis")

#as there are so many values ,its showing in the very smalll values ,which i am not able to see
```

Out[9]:
```
<Axes: >
```



In [23]:
```
#importing country code data
d_country = pd.read_excel("D:/Country-Code.xlsx")
d_country.head()
```

Out[23]:

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

In [24]:
```
#combining the data frame as in the first dataset there is country code column and in this also we can find the country code
#so ,we can merge dataframe using merge
pd.merge(data,d_country,on='Country Code',how='left')
```

Out[24]:

	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	Aggregate rating	Rating color	Rating text	Votes	Country
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	4.8	Dark Green	Excellent	314	Phillipines
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	4.5	Dark Green	Excellent	591	Phillipines
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	4.4	Green	Very Good	270	Phillipines
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	4.9	Dark Green	Excellent	365	Phillipines
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	4.8	Dark Green	Excellent	229	Phillipines
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
9546	5915730	NamlÜ± Gurme	208	ÜÁstanbul	KemankeÐõ Karamustafa PaÐõa Mahallesi, RÜ±htÛ±...	Karaki_y	Karaki_y, ÜÁstanbul	28.977392	41.022793	Turkish	...	No	No	No	No	3	4.1	Green	Very Good	788	Turkey
9547	5908749	Ceviz AÜðacÜ±	208	ÜÁstanbul	KoÐõuyolu Mahallesi, Muhittin Üstü_ndaÜó Cadd...	KoÐõuyolu	KoÐõuyolu, ÜÁstanbul	29.041297	41.009847	World Cuisine, Patisserie, Cafe	...	No	No	No	No	3	4.2	Green	Very Good	1034	Turkey
9548	5915807	Huqqa	208	ÜÁstanbul	Kurui_eÐõme Mahallesi, Muallim Naci Caddesi, N...	Kurui_eÐõme	Kurui_eÐõme, ÜÁstanbul	29.034640	41.055817	Italian, World Cuisine	...	No	No	No	No	4	3.7	Yellow	Good	661	Turkey
9549	5916112	AÐõÐõk Kahve	208	ÜÁstanbul	Kurui_eÐõme Mahallesi, Muallim Naci Caddesi, N...	Kurui_eÐõme	Kurui_eÐõme, ÜÁstanbul	29.036019	41.057979	Restaurant Cafe	...	No	No	No	No	4	4.0	Green	Very Good	901	Turkey
9550	5927402	Walter's Coffee Roastery	208	ÜÁstanbul	CafeaÛða Mahallesi, BademaltÜ± Sokak, No 21/B,...	Moda	Moda, ÜÁstanbul	29.026016	40.984776	Cafe	...	No	No	No	No	2	4.0	Green	Very Good	591	Turkey

9551 rows × 22 columns

In [25]:
```
#saving the new data
new_data=pd.merge(data,d_country,on='Country Code',how='left')
new_data.head(2)
```

Out[25]:

	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	Aggregate rating	Rating color	Rating text	Votes	Country
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	4.8	Dark Green	Excellent	314	Phillipines
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	4.5	Dark Green	Excellent	591	Phillipines

2 rows × 22 columns

In [26]:
```
#to check the data types
new_data.dtypes
```

Out[26]:
```
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 [27]:
```
new_data.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', 'Country'],
      dtype='object')
```

In [28]:
```
new_data.Country.value_counts()
```

Out[28]:
```
Country
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: count, dtype: int64
```

In [29]:
```
county_value= new_data.Country.value_counts().values
county_value
```

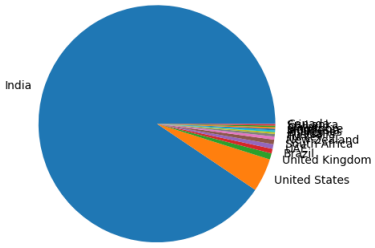
Out[29]:
```
array([8652,  434,   80,   60,   60,   60,   40,   34,   24,   22,   21,
         20,   20,   20,    4], dtype=int64)
```

In [30]:
```
country_name=new_data.Country.value_counts().index
country_name
```

Out[30]:
```
Index(['India', 'United States', 'United Kingdom', 'Brazil', 'UAE',
       'South Africa', 'New Zealand', 'Turkey', 'Australia', 'Phillipines',
       'Indonesia', 'Singapore', 'Qatar', 'Sri Lanka', 'Canada'],
      dtype='object', name='Country')
```

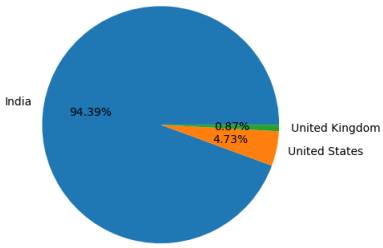
In [33]:
```
#ploting pie chart,to see which country has highest transactions
plt.pie(county_value,labels=country_name)
```

Out[33]:
```
([<matplotlib.patches.Wedge at 0x2155e46b890>,
  <matplotlib.patches.Wedge at 0x2155e474590>,
  <matplotlib.patches.Wedge at 0x2155e475590>,
  <matplotlib.patches.Wedge at 0x2155e476650>,
  <matplotlib.patches.Wedge at 0x2155e477410>,
  <matplotlib.patches.Wedge at 0x2155e480450>,
  <matplotlib.patches.Wedge at 0x2155e481550>,
  <matplotlib.patches.Wedge at 0x2155e4823d0>,
  <matplotlib.patches.Wedge at 0x2155e477510>,
  <matplotlib.patches.Wedge at 0x2155e4902d0>,
  <matplotlib.patches.Wedge at 0x2155e491110>,
  <matplotlib.patches.Wedge at 0x2155e492190>,
  <matplotlib.patches.Wedge at 0x2155e4931d0>,
  <matplotlib.patches.Wedge at 0x2155e4a0050>,
  <matplotlib.patches.Wedge at 0x2155e4a0890>],
 [Text(-1.052256163793291, 0.3205572737577906, 'India'),
  Text(0.9911329812843455, -0.47713249041523, 'United States'),
  Text(1.0572858296119743, -0.3035567072257165, 'United Kingdom'),
  Text(1.070138816916019, -0.2545641619112621, 'Brazil'),
  Text(1.0793506814479759, -0.21213699926648824, 'UAE'),
  Text(1.0868811472449 73, -0.16937937230799818, 'South Africa'),
  Text(1.0918635911832035, -0.13354361927294 86, 'New Zealand'),
  Text(1.0947903814016446, -0.10692998078388304, 'Turkey'),
  Text(1.0966310239 45382, -0.08602556201794338, 'Australia'),
  Text(1.0978070729776455, -0.06942355882735218, 'Phillipines'),
  Text(1.0986791544015209, -0.05388984768543213, 'Indonesia'),
  Text(1.0993059848742366, -0.03906855026341 3035, 'Singapore'),
  Text(1.0997248508282123, -0.02460187941736628, 'Qatar'),
  Text(1.0999533462179636, -0.01013094980271 6446, 'Sri Lanka'),
  Text(1.0999990477553414, -0.00144738983767076 38, 'Canada')])
```



In [32]:
```
#above plot is very bad why because ,in other counties zomato is very less ,now we can see only the top 5 transaction
#so
plt.pie(county_value[:3],labels=country_name[:3],autopct="%1.2f%%")
```

Out[32]:
```
([<matplotlib.patches.Wedge at 0x2155e416310>,
  <matplotlib.patches.Wedge at 0x2155e417350>,
  <matplotlib.patches.Wedge at 0x2155e430b50>],
 [Text(-1.0829742700952103, 0.19278674827836725, 'India'),
  Text(1.0772817158383 56, -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%')])
```



inference : So the top 3 countries is india,uk,us By this we can observe that zomato has max records and transaction are from india after that USA And UK

In [34]:
```
new_data.columns
```

Out[34]:
```
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 [39]:
```
new_data.groupby(['Aggregate rating', 'Rating color','Rating text'])
#now the data frame is formed
```

Out[39]:
```
<pandas.core.groupby.generic.DataFrameGroupBy object at 0x000002155E416590>
```

In [40]:
```
new_data.groupby(['Aggregate rating', 'Rating color','Rating text']).size()
```

Out[40]:
```
Aggregate rating  Rating color  Rating text
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 [41]:
```
ratings=new_data.groupby(['Aggregate rating', 'Rating color','Rating text']).size().reset_index().rename(columns={0:"Rating Count"}) #rename :changing 0 to new name
ratings
```

Out[41]:

	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

1) when rating is between 4.5 - 4.9 ---> excellent 2) when rating is between 4 - 3.4 --->very good 3) when rating is between 3.5 to 3.9 ----> good 4) when rating is between 3 to 3.4 and 2.5 to 2.9 ----->average 5) when rating is between2 to 3.4 ----> poor '''

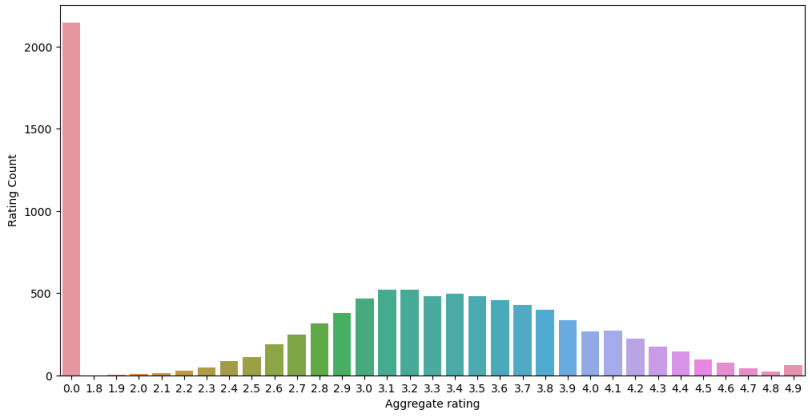
In [42]:
```
ratings.head()
```

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

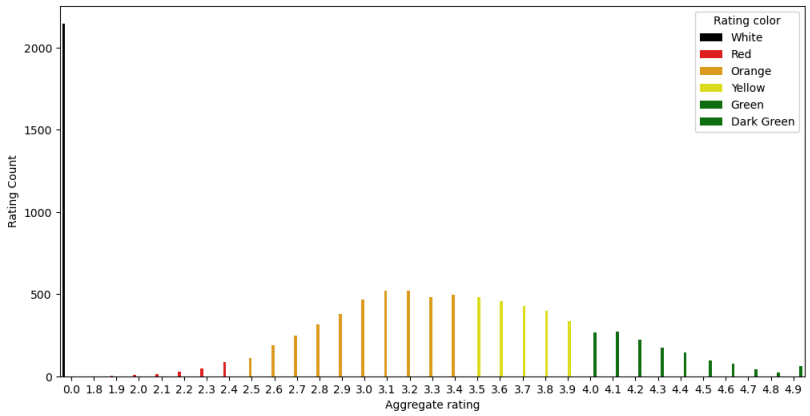
In [43]:
```
#visualization
#barplot
import matplotlib
matplotlib.rcParams['figure.figsize']=(12,6)
sns.barplot(x="Aggregate rating",y="Rating Count",data=ratings)
```

Out[43]:
```
<Axes: xlabel='Aggregate rating', ylabel='Rating Count'>
```



In [60]:
```
sns.barplot(x="Aggregate rating",y="Rating Count",data=ratings,hue="Rating color",palette=['Black','red','orange','yellow','green','green'])
```

Out[60]:
```
<Axes: xlabel='Aggregate rating', ylabel='Rating Count'>
```

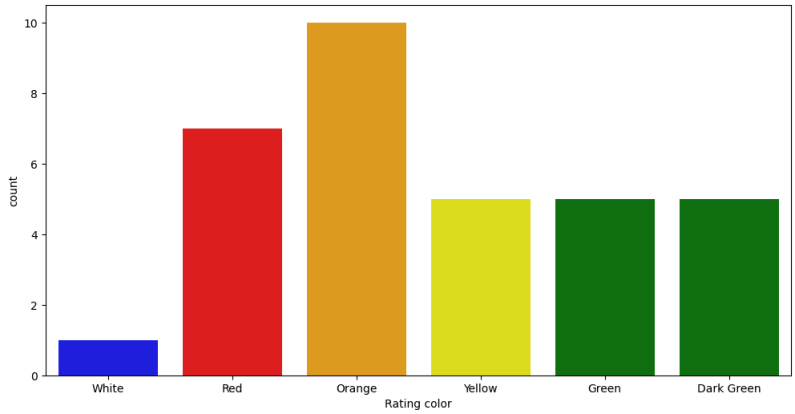


obseravtions

Not rated count is very high which is blue in colour (where there are no reviews) maximum number of rating are between 2.5 to 3.4

In [44]:
```
##countplot
sns.countplot(x="Rating color",data=ratings ,palette=['blue','red','orange','yellow','green','green'])
```

Out[44]:
```
<Axes: xlabel='Rating color', ylabel='count'>
```



In [61]:
```
#find the countries that has given 0 rating
new_data.columns
```

Out[61]:
```
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 [62]:
```
new_data[new_data["Rating color"]=='White'].groupby('Country')
```

Out[62]:
```
<pandas.core.groupby.generic.DataFrameGroupBy object at 0x000001349A950410>
```

In [63]:
```
new_data[new_data["Rating color"]=='White'].groupby('Country').size()
```

Out[63]:
```
Country
Brazil              5
India            2139
United Kingdom      1
United States       3
dtype: int64
```

In [64]:
```
new_data[new_data["Rating color"]=='White'].groupby('Country').size().reset_index()
```

Out[64]:

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

Observations from the above :
Maximum Number of 0 Ratings are from Indian customer

In [65]:
```
#finding currency used by different countries
new_data.columns
```

Out[65]:
```
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 [66]:
```
new_data[['Country','Currency']].groupby(['Country','Currency']).size().reset_index()
```

Out[66]:

	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
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

In [67]:
```
#country has online delivery
new_data[new_data['Has Online delivery']=='Yes'].Country.value_counts()
```

Out[67]:
```
Country
India    2423
UAE        28
Name: count, dtype: int64
```

In [333…
```
new_data[['Has Online delivery','Country']].groupby(['Has Online delivery','Country']).size().reset_index()
```

Out[333]:

	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

by the result we can observe that only india and UAE has online deliveries

In [73]:
```
new_data.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]:
```
#piechart for cities distribution
new_data['City'].value_counts()
```

Out[74]:
```
City
New Delhi           5473
Gurgaon             1118
Noida               1080
Faridabad            251
Ghaziabad             25
                    ... 
Panchkula              1
Mc Millan              1
Mayfield               1
Macedon                1
Vineland Station       1
Name: count, Length: 141, dtype: int64
```

In [75]:
```
city_val=new_data['City'].value_counts().values
city_val
```

Out[75]:
```
array([5473, 1118, 1080,  251,   25,   21,   21,   21,   21,   21,   20,
         20,   20,   20,   20,   20,   20,   20,   20,   20,   20,   20,
         20,   20,   20,   20,   20,   20,   20,   20,   20,   20,   20,
         20,   20,   20,   20,   20,   20,   20,   20,   20,   20,   20,
         20,   20,   20,   20,   20,   20,   20,   20,   20,   20,   20,
         20,   20,   20,   20,   20,   20,   20,   20,   20,   20,   20,
         20,   20,   20,   20,   20,   20,   20,   20,   20,   20,   20,
         18,   18,   16,   14,   11,    6,    4,    4,    3,    3,    2,
          2,    2,    2,    2,    2,    2,    2,    1,    1,    1,    1,
          1,    1,    1,    1,    1,    1,    1,    1,    1,    1,    1,
          1,    1,    1,    1,    1,    1,    1,    1,    1,    1,    1,
          1,    1,    1,    1,    1,    1,    1,    1,    1,    1,    1,
          1,    1,    1,    1,    1,    1,    1,    1,    1], dtype=int64)
```

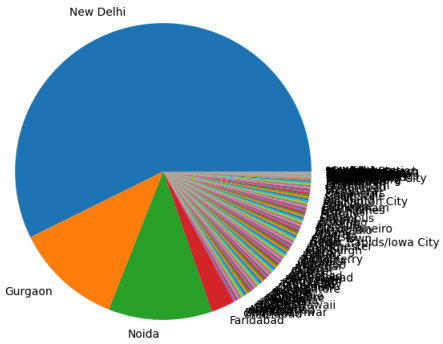
In [76]:
```
city_label = new_data['City'].value_counts().index
city_label
```

Out[76]:
```
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', name='City', length=141)
```

In [77]:
```
#ploting pie chart
plt.pie(city_val,labels=city_label)
```

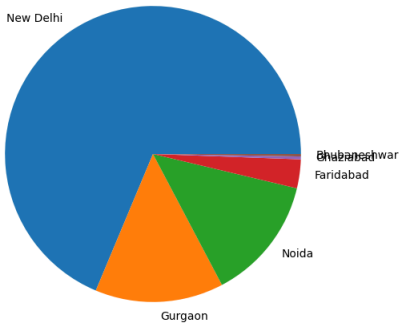
Out[77]: ([<matplotlib.patches.Wedge at 0x1349b9f6610>,
<matplotlib.patches.Wedge at 0x1349b9f4710>,
<matplotlib.patches.Wedge at 0x1349b9e1250>,
<matplotlib.patches.Wedge at 0x1349b9e3110>,
<matplotlib.patches.Wedge at 0x1349b9cbbd0>,
<matplotlib.patches.Wedge at 0x1349b9d0450>,
<matplotlib.patches.Wedge at 0x1349b9cc850>,
<matplotlib.patches.Wedge at 0x1349b9cea90>,
<matplotlib.patches.Wedge at 0x1349b9cb7d0>,
<matplotlib.patches.Wedge at 0x13497247d90>,
<matplotlib.patches.Wedge at 0x1349b9b44d0>,
<matplotlib.patches.Wedge at 0x1349b9b71d0>,
<matplotlib.patches.Wedge at 0x1349b9d5750>,
<matplotlib.patches.Wedge at 0x1349ba047d0>,
<matplotlib.patches.Wedge at 0x1349ba05c10>,
<matplotlib.patches.Wedge at 0x1349ba06e10>,
<matplotlib.patches.Wedge at 0x1349ba07b50>,
<matplotlib.patches.Wedge at 0x1349ba09090>,
<matplotlib.patches.Wedge at 0x1349ba099d0>,
<matplotlib.patches.Wedge at 0x1349ba0b490>,
<matplotlib.patches.Wedge at 0x1349ba5c490>,
<matplotlib.patches.Wedge at 0x1349ba5d350>,
<matplotlib.patches.Wedge at 0x1349ba5e250>,
<matplotlib.patches.Wedge at 0x1349ba5f150>,
<matplotlib.patches.Wedge at 0x1349ba6c050>,
<matplotlib.patches.Wedge at 0x1349ba6ca90>,
<matplotlib.patches.Wedge at 0x1349ba6df50>,
<matplotlib.patches.Wedge at 0x1349ba6eed0>,
<matplotlib.patches.Wedge at 0x1349ba6fb10>,
<matplotlib.patches.Wedge at 0x1349ba7cd90>,
<matplotlib.patches.Wedge at 0x1349ba7dd50>,
<matplotlib.patches.Wedge at 0x1349ba7ec50>,
<matplotlib.patches.Wedge at 0x1349ba7fb10>,
<matplotlib.patches.Wedge at 0x1349ba88b50>,
<matplotlib.patches.Wedge at 0x1349ba899d0>,
<matplotlib.patches.Wedge at 0x1349ba8a9d0>,
<matplotlib.patches.Wedge at 0x1349ba8b990>,
<matplotlib.patches.Wedge at 0x1349ba94910>,
<matplotlib.patches.Wedge at 0x1349ba95890>,
<matplotlib.patches.Wedge at 0x1349ba96810>,
<matplotlib.patches.Wedge at 0x1349ba97890>,
<matplotlib.patches.Wedge at 0x1349baa0850>,
<matplotlib.patches.Wedge at 0x1349baa1910>,
<matplotlib.patches.Wedge at 0x1349baa2950>,
<matplotlib.patches.Wedge at 0x1349baa3650>,
<matplotlib.patches.Wedge at 0x1349b6e4790>,
<matplotlib.patches.Wedge at 0x1349b6e5550>,
<matplotlib.patches.Wedge at 0x1349b6e65d0>,
<matplotlib.patches.Wedge at 0x1349b6e71d0>,
<matplotlib.patches.Wedge at 0x1349b6f4490>,
<matplotlib.patches.Wedge at 0x1349b6f5290>,
<matplotlib.patches.Wedge at 0x1349b6f61d0>,
<matplotlib.patches.Wedge at 0x1349b6f7090>,
<matplotlib.patches.Wedge at 0x1349b6f7f50>,
<matplotlib.patches.Wedge at 0x1349b704950>,
<matplotlib.patches.Wedge at 0x1349b705d10>,
<matplotlib.patches.Wedge at 0x1349b706c10>,
<matplotlib.patches.Wedge at 0x1349b7079d0>,
<matplotlib.patches.Wedge at 0x1349b714b90>,
<matplotlib.patches.Wedge at 0x1349b715ad0>,
<matplotlib.patches.Wedge at 0x1349b716a50>,
<matplotlib.patches.Wedge at 0x1349b717810>,
<matplotlib.patches.Wedge at 0x1349b724790>,
<matplotlib.patches.Wedge at 0x1349b725510>,
<matplotlib.patches.Wedge at 0x1349b7262d0>,
<matplotlib.patches.Wedge at 0x1349b727210>,
<matplotlib.patches.Wedge at 0x1349b730190>,
<matplotlib.patches.Wedge at 0x1349b7311d0>,
<matplotlib.patches.Wedge at 0x1349b731f90>,
<matplotlib.patches.Wedge at 0x1349b732e90>,
<matplotlib.patches.Wedge at 0x1349b733d50>,
<matplotlib.patches.Wedge at 0x1349b744c50>,
<matplotlib.patches.Wedge at 0x1349b745b50>,
<matplotlib.patches.Wedge at 0x1349b746a50>,
<matplotlib.patches.Wedge at 0x1349b747850>,
<matplotlib.patches.Wedge at 0x1349b754890>,
<matplotlib.patches.Wedge at 0x1349b755b10>,
<matplotlib.patches.Wedge at 0x1349b756dd0>,
<matplotlib.patches.Wedge at 0x1349b760090>,
<matplotlib.patches.Wedge at 0x1349b7613d0>,
<matplotlib.patches.Wedge at 0x1349b762790>,
<matplotlib.patches.Wedge at 0x1349b763a50>,
<matplotlib.patches.Wedge at 0x1349b76cd90>,
<matplotlib.patches.Wedge at 0x1349b76dfd0>,
<matplotlib.patches.Wedge at 0x1349b76f310>,
<matplotlib.patches.Wedge at 0x1349b77c610>,
<matplotlib.patches.Wedge at 0x1349b77d910>,
<matplotlib.patches.Wedge at 0x1349b77ec10>,
<matplotlib.patches.Wedge at 0x1349b77fb10>,
<matplotlib.patches.Wedge at 0x1349b78d150>,
<matplotlib.patches.Wedge at 0x1349b78e390>,
<matplotlib.patches.Wedge at 0x1349b78f750>,
<matplotlib.patches.Wedge at 0x1349b798990>,
<matplotlib.patches.Wedge at 0x1349b799cd0>,
<matplotlib.patches.Wedge at 0x1349b79b0d0>,
<matplotlib.patches.Wedge at 0x1349b7a0410>,
<matplotlib.patches.Wedge at 0x1349b7a1790>,
<matplotlib.patches.Wedge at 0x1349b7a2a10>,
<matplotlib.patches.Wedge at 0x1349b7a3cd0>,
<matplotlib.patches.Wedge at 0x1349b7b4f10>,
<matplotlib.patches.Wedge at 0x1349b7b6210>,
<matplotlib.patches.Wedge at 0x1349b7b7490>,
<matplotlib.patches.Wedge at 0x1349b7bc7d0>,
<matplotlib.patches.Wedge at 0x1349b7bda90>,
<matplotlib.patches.Wedge at 0x1349b7bed50>,
<matplotlib.patches.Wedge at 0x1349b7bff90>,
<matplotlib.patches.Wedge at 0x1349b7cd390>,
<matplotlib.patches.Wedge at 0x1349b7ce610>,
<matplotlib.patches.Wedge at 0x1349b7cf8d0>,
<matplotlib.patches.Wedge at 0x1349b7d4b10>,
<matplotlib.patches.Wedge at 0x1349b7d5dd0>,
<matplotlib.patches.Wedge at 0x1349b7d70d0>,
<matplotlib.patches.Wedge at 0x1349b7e0350>,
<matplotlib.patches.Wedge at 0x1349b7e1710>,
<matplotlib.patches.Wedge at 0x1349b7e2990>,
<matplotlib.patches.Wedge at 0x1349b7e3bd0>,
<matplotlib.patches.Wedge at 0x1349b7f4ed0>,
<matplotlib.patches.Wedge at 0x1349b7f60d0>,
<matplotlib.patches.Wedge at 0x1349b7f7410>,
<matplotlib.patches.Wedge at 0x1349b804650>,
<matplotlib.patches.Wedge at 0x1349b805910>,
<matplotlib.patches.Wedge at 0x1349b806c10>,
<matplotlib.patches.Wedge at 0x1349b807f10>,
<matplotlib.patches.Wedge at 0x1349b80d150>,
<matplotlib.patches.Wedge at 0x1349b80e350>,
<matplotlib.patches.Wedge at 0x1349b80f6d0>,
<matplotlib.patches.Wedge at 0x1349b81c910>,
<matplotlib.patches.Wedge at 0x1349b81db50>,
<matplotlib.patches.Wedge at 0x1349b81ed50>,
<matplotlib.patches.Wedge at 0x1349b828050>,
<matplotlib.patches.Wedge at 0x1349b8292d0>,
<matplotlib.patches.Wedge at 0x1349b82a5d0>,
<matplotlib.patches.Wedge at 0x1349b82b8d0>,
<matplotlib.patches.Wedge at 0x1349b838bd0>,
<matplotlib.patches.Wedge at 0x1349b839ed0>,
<matplotlib.patches.Wedge at 0x1349b83b1d0>,
<matplotlib.patches.Wedge at 0x1349b8405d0>,
<matplotlib.patches.Wedge at 0x1349b8418d0>,
<matplotlib.patches.Wedge at 0x1349b842bd0>,
<matplotlib.patches.Wedge at 0x1349b843f50>,
<matplotlib.patches.Wedge at 0x1349b84d110>],
[Text(-0.2501618674207579, 1.0711764747643402, 'New Delhi'),
Text(-0.7451219157193338, -0.8091930120278745, 'Gurgaon'),
Text(-0.023335860702208486, -1.0997524437823665, 'Noida'),
Text(0.4451062250173843, -1.0059226851258372, 'Faridabad'),
Text(0.5344697170236794, -0.9614271275477035, 'Ghaziabad'),
Text(0.5489550432957175, -0.9532304865247423, 'Bhubaneshwar'),
Text(0.5620711022307091, -0.9455559613461045, 'Amritsar'),
Text(0.5750798895062447, -0.9377009761568373, 'Ahmedabad'),
Text(0.5879789223862543, -0.9296670300863097, 'Lucknow'),
Text(0.6007657390813725, -0.9214556564186973, 'Guwahati'),
Text(0.6131375319471077, -0.9132701500201403, 'Ludhiana'),
Text(0.6251000125752387, -0.9051242084843107, 'Rest of Hawaii'),
Text(0.6369545100691332, -0.8968215832051494, 'Kochi'),
Text(0.6486986328151094, -0.8883637114284944, 'Kolkata'),
Text(0.6603304609951781, -0.8797520572763077, 'Pocatello'),
Text(0.6718479810532302, -0.8709881114887267, 'Pensacola'),
Text(0.6832491992207405, -0.8620733911705064, 'Mumbai'),
Text(0.6945321418619113, -0.853009439526144, 'Mangalore'),
Text(0.7056948558153163, -0.8437978255931927, 'Singapore'),
Text(0.7167354087320118, -0.8344401439706484, 'Orlando'),

```
Text(0.7276518894100411, -0.8249380145429095, 'Mysore'),
Text(0.7384424081252736, -0.8152930821993689, 'Nagpur'),
Text(0.749105096958536, -0.8055070165496653, 'Nashik'),
Text(0.7596381101189571, -0.7955815116346655, 'Coimbatore'),
Text(0.7700396242634912, -0.7855182856332128, 'Dehradun'),
Text(0.7803078388125538, -0.7753190805646936, 'Savannah'),
Text(0.7904409762617105, -0.7649856619874872, 'Tampa Bay'),
Text(0.8004372824893825, -0.7545198186933281, 'Sioux City'),
Text(0.810295027060492, -0.74392333623976578, 'Abu Dhabi'),
Text(0.8200125035260163, -0.7331981274260015, 'Bangalore'),
Text(0.8295880297183893, -0.7223459703964306, 'Aurangabad'),
Text(0.8390199480426914, -0.7113687698981728, 'Allahabad'),
Text(0.8483066257635978, -0.7002684261664089, 'Agra'),
Text(0.8574464552288012, -0.6890468607533333, 'Sharjah'),
Text(0.8664378544433541, -0.6777060161955162, 'Dubai'),
Text(0.8752792667514493, -0.666247855677634, 'Goa'),
Text(0.8839691616979604, -0.6546743626926333, 'Bhopal'),
Text(0.8925060349973365, -0.6429875406983664, 'Chennai'),
Text(0.9008884088532129, -0.6311894127707836, 'Waterloo'),
Text(0.9091148322142286, -0.6192820212537218, 'Indore'),
Text(0.9171838810252173, -0.6072674274053569, 'Valdosta'),
Text(0.9250945158473717, -0.5951477110413896, 'Puducherry'),
Text(0.9328442952317741, -0.5829249701750087, 'Kanpur'),
Text(0.9404329496929795, -0.5706013206537136, 'Jaipur'),
Text(0.9478588082047126, -0.5581788957930441, 'Patna'),
Text(0.9551205852955462, -0.5456598460072843, 'Macon'),
Text(0.9622170238977674, -0.53304633843721264, 'Edinburgh'),
Text(0.9691468955649892, -0.5203405565749647, 'London'),
Text(0.9759090006848006, -0.5075446998860239, 'Manchester'),
Text(0.9825021686864295, -0.4946609834285123, 'Pune'),
Text(0.9889252582433793, -0.48169163746697259, 'Cedar Rapids/Iowa City'),
Text(0.9951771574709969, -0.4686389071000686, 'Boise'),
Text(1.0012567841189528, -0.4555050518444039, 'Cape Town'),
Text(1.0071630857585825, -0.44229234527099196, 'Augusta'),
Text(1.0128950399650722, -0.4290030745975545, 'Athens'),
Text(1.0184516544944486, -0.41563954029600603, 'Albany'),
Text(1.0238319674553341, -0.40220405569378004, 'SiÉo Paulo'),
Text(1.0290350474754957, -0.3886989465731858, 'Rio de Janeiro'),
Text(1.0340599938629995, -0.37512655076927554, 'Brasí_lia'),
Text(1.0389059367621981, -0.36148921776487311, 'Pretoria'),
Text(1.0435720373042763, -0.34778930828391281, 'Colombo'),
Text(1.0480574877524684, -0.33402919388278?, 'Ankara'),
Text(1.0523615116418888, -0.32021125653980437, 'Columbus'),
Text(1.0564833631393936, -0.30633788824285857, 'Doha'),
Text(1.0604223310452774, -0.2924114905753542, 'Dalton'),
Text(1.0641777311713583, -0.27843447430047213, 'Varanasi'),
Text(1.0677489142044403, -0.26440925894385514, 'Des Moines'),
Text(1.0711352619461 37, -0.2503382723747622, 'Birmingham'),
Text(1.0743361881944267, -0.23622395038579308, 'Dubuque'),
Text(1.07735113884513, -0.2220687362712315, 'Ranchi'),
Text(1.0801795919878292, -0.20787508040408997, 'Wellington City'),
Text(1.0828210579962145, -0.19364543981193752, 'Auckland'),
Text(1.0852750796128416, -0.17938227775156726, 'Surat'),
Text(1.0875412320282876, -0.16508806328258516, 'Gainesville'),
Text(1.0896191229546865, -0.15076527084000446, 'Davenport'),
Text(1.0915083926936402, -0.13641637980858945, 'Vadodara'),
Text(1.0932087141984825, -0.12204387408018827, 'Vizag'),
Text(1.0946487378584324, -0.10837038666047613, 'Hyderabad'),
Text(1.0958552219281623, -0.0954009044652007, 'Chandigarh'),
Text(1.0968535914548911, -0.08313963502750614, 'Jakarta'),
Text(1.0976205848484757, -0.07231218235463886, 'ÜÁstanbul'),
Text(1.0981781050169444, -0.06328388152913558, 'Sandton'),
Text(1.0985148037575858, -0.05714215541465917, 'Johannesburg'),
Text(1.0986968174339933, -0.05352852847234547, 'Taguig City'),
Text(1.0988338696837698, -0.05063720801735?5, 'Mandaluyong City'),
Text(1.0989475488721216, -0.04810701433321760?6, 'Pasig City'),
Text(1.0990403511294076, -0.04593807341790467, 'Pasay City'),
Text(1.0991144163556001, -0.04413048559965186, 'Bogor'),
Text(1.0991715283611 4, -0.04268432914729057?, 'Inner City'),
Text(1.0992267369414956, -0.04123809880380096?, 'Secunderabad'),
Text(1.0992800429761727, -0.03979179707306476, 'Tangerang'),
Text(1.0993314460478671, -0.03834542645877398, 'Makati City'),
Text(1.0993809460675954, -0.03689898946473986?, 'San Juan City'),
Text(1.0994285429496677, -0.03545248859488858, 'Santa Rosa'),
Text(1.0994742366116894, -0.03400592635325686, 'Hepburn Springs'),
Text(1.0995072578223395, -0.03292096590319728?, 'Yorkton'),
Text(1.0995286771658292, -0.03219764106206654, 'Consort'),
Text(1.0995496206609192, -0.03147430228660530?, 'Inverloch'),
Text(1.0995700882985457, -0.03075094989856428, 'Huskisson'),
Text(1.0995900800698508, -0.03002758418486867, 'Bandung'),
Text(1.0996095959661822, -0.02930420548469653?, 'Forrest'),
Text(1.0996286359790945, -0.02858081410240017, 'Flaxton'),
Text(1.0996472001003474, -0.02785741035104519?, 'Fernley'),
Text(1.0996652883211907, -0.02713399454370257?, 'East Ballina'),
Text(1.0996829006359445, -0.02641056699344852, 'Dicky Beach'),
Text(1.0997000370348387, -0.0256871280133643, 'Cochrane'),
Text(1.0997166975111725, -0.02496367791653614, 'Lakeview'),
Text(1.0997328820577363, -0.02424021701605507?, 'Clatskanie'),
Text(1.0997485906675255, -0.02351674562501681, 'Chatham-Kent'),
Text(1.0997638233333742, -0.0227932640565216, 'Beechworth'),
Text(1.0997785800497932, -0.02206977262367410?, 'Balingup'),
Text(1.0997928608092 93, -0.02134627163958323?, 'Armidale'),
Text(1.0998066656060612, -0.02062276141736205?, 'Tagaytay City'),
Text(1.0998199944341234, -0.01989924227012762?, 'Randburg'),
Text(1.0998328472877106, -0.01917571451100085, 'Quezon City'),
Text(1.0998452241612613, -0.01845217845310638?, 'Lakes Entrance'),
Text(1.0998571250494182, -0.01772863440957246?, 'Lincoln'),
Text(1.0998685499470318, -0.01700508026935077?, 'Winchester Bay'),
Text(1.0998794988491571, -0.01628152361811633?, 'Lorn'),
Text(1.0998899717510557, -0.01555795749646733?, 'Weirton'),
Text(1.0998999686481954, -0.01483438464172503?, 'Victor Harbor'),
Text(1.0999094895362498, -0.01411080536703358?, 'Vernonia'),
Text(1.0999185344110984, -0.01338721998553993?, 'Trentham East'),
Text(1.099927103268827, -0.01266362881039364?, 'Tanunda'),
Text(1.0999351961057269, -0.01194003215474683, 'Princeton'),
Text(1.0999428129182958, -0.01121643033175393?, 'Potrero'),
Text(1.0999499537032376, -0.01049282365457166?, 'Phillip Island'),
Text(1.0999566184574614, -0.00976921243635881?, 'Penola'),
Text(1.0999628071780836, -0.00904559699027615?, 'Paynesville'),
Text(1.0999685198624252, -0.00832197762948627?, 'Mohali'),
Text(1.0999737565080143, -0.00759835466715345?, 'Palm Cove'),
Text(1.0999785171125844, -0.00687472841644354?, 'Ojo Caliente'),
Text(1.0999828016740756, -0.00615109919052382?, 'Montville'),
Text(1.0999866610190633, -0.00542746730256284, 'Monroe'),
Text(1.0999899426606609, -0.00470383306573030?, 'Miller'),
Text(1.0999927990082561, -0.00398019679319694?, 'Middleton Beach'),
Text(1.0999951794552532, -0.00325655879813435?, 'Panchkula'),
Text(1.0999970837776 55, -0.00253291933971490?5, 'Mc Millan'),
Text(1.0999985120489424, -0.00180927889311154?7, 'Mayfield'),
Text(1.0999994642684974, -0.00108563760949772?5, 'Macedon'),
Text(1.0999999404359078, -0.00036199585604719?7, 'Vineland Station')])
```



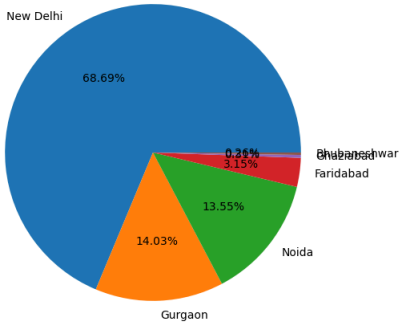
In [78]:
```
plt.pie(city_val[:6],labels=city_label[:6])
```

Out[78]:
```
([<matplotlib.patches.Wedge at 0x1349b8c6550>,
  <matplotlib.patches.Wedge at 0x1349b8c6a10>,
  <matplotlib.patches.Wedge at 0x1349b8c7950>,
  <matplotlib.patches.Wedge at 0x1349b8d8910>,
  <matplotlib.patches.Wedge at 0x1349b8d9790>,
  <matplotlib.patches.Wedge at 0x1349b8da790>],
 [Text(-0.6093229902525419, 0.9158195747797166, 'New Delhi'),
  Text(0.04855897468526975, -1.0989276709490554, 'Gurgaon'),
  Text(0.8689790405567204, -0.6744445322434767, 'Noida'),
  Text(1.08995637491287, -0.14830745357801672, 'Faridabad'),
  Text(1.099616212633222, -0.029054860422456693, 'Ghaziabad'),
  Text(1.0999622942144016, -0.0091077607890389 5, 'Bhubaneshwar')])
```



In [79]:
```
#auto percentages
plt.pie(city_val[:6],labels=city_label[:6],autopct="%1.2f%%")
```

Out[79]:
```
([<matplotlib.patches.Wedge at 0x1349bd06d50>,
  <matplotlib.patches.Wedge at 0x1349bd1c150>,
  <matplotlib.patches.Wedge at 0x1349bd1d650>,
  <matplotlib.patches.Wedge at 0x1349bd1efd0>,
  <matplotlib.patches.Wedge at 0x1349bd285d0>,
  <matplotlib.patches.Wedge at 0x1349bd29cd0>],
 [Text(-0.6093229902525419, 0.9158195747797166, 'New Delhi'),
  Text(0.04855897468526975, -1.0989276709490554, 'Gurgaon'),
  Text(0.8689790405567204, -0.6744445322434767, 'Noida'),
  Text(1.08995637491287, -0.14830745357801672, 'Faridabad'),
  Text(1.099616212633222, -0.029054860422456693, 'Ghaziabad'),
  Text(1.0999622942144016, -0.0091077607890389 5, 'Bhubaneshwar')],
 [Text(-0.3323579946832046, 0.4995379498798454, '68.69%'),
  Text(0.026486713464692 59, -0.5994150932449392, '14.03%'),
  Text(0.4739885675763928 5, -0.3678788357691 69, '13.55%'),
  Text(0.5945216590433835, -0.0808949746789182, '3.15%'),
  Text(0.5997906614363029, -0.0158481056849 76375, '0.31%'),
  Text(0.5999794332078554, -0.0049678695212939 72, '0.26%')])
```



so the maximum transactions from new delhi which is 68.69%

In [80]:
```
new_data.columns
```

Out[80]:
```
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]:
```
 #Remove duplicates
new_data.drop_duplicates(inplace=True)
#Finding top 10 cuisines
cuisine_counts =new_data['Cuisines'].value_counts()
cuisine_counts
```

Out[86]:
```
Cuisines
North Indian                                          936
North Indian, Chinese                                 511
Chinese                                               354
Fast Food                                             354
North Indian, Mughlai                                 334
                                                     ...
Bengali, Fast Food                                      1
North Indian, Rajasthani, Asian                         1
Chinese, Thai, Malaysian, Indonesian                    1
Bakery, Desserts, North Indian, Bengali, South Indian   1
Italian, World Cuisine                                  1
Name: count, Length: 1825, dtype: int64
```

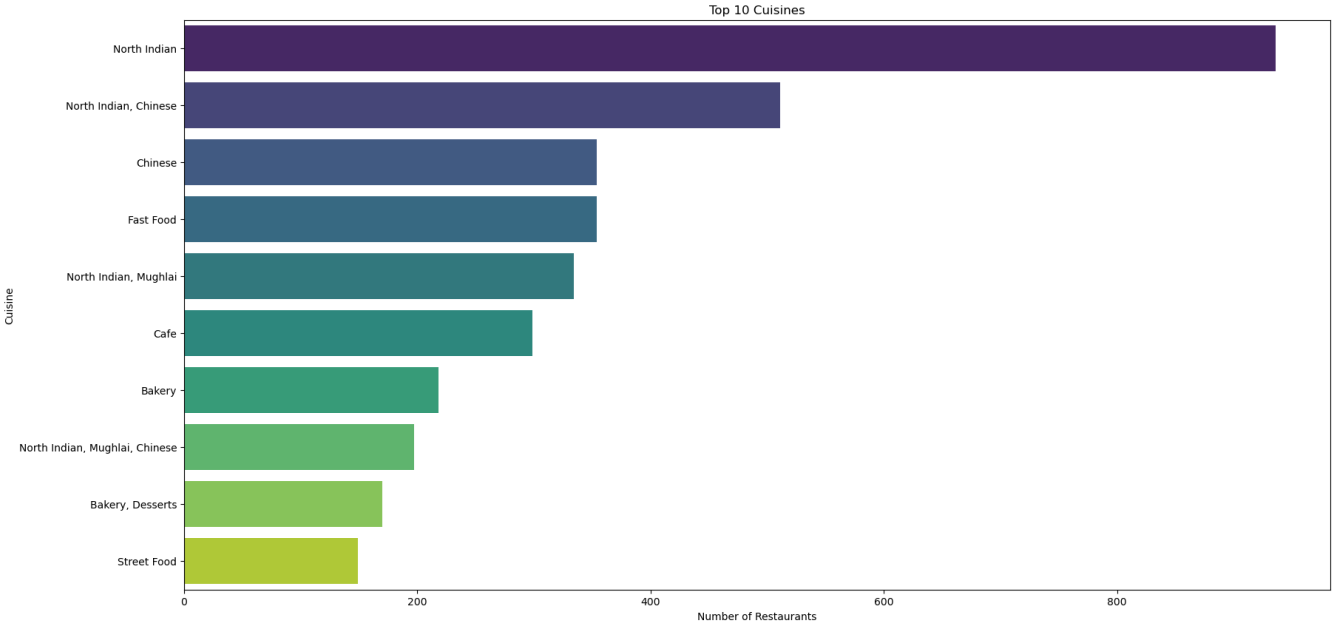
In [87]:
```
# Drop rows with mi
# Display the top 10 cuisines
print(cuisine_counts.head(10).reset_index())
```

```
                        Cuisines  count
0                   North Indian    936
1          North Indian, Chinese    511
2                        Chinese    354
3                      Fast Food    354
4          North Indian, Mughlai    334
5                           Cafe    299
6                         Bakery    218
7  North Indian, Mughlai, Chinese    197
8               Bakery, Desserts    170
9                    Street Food    149
```

In [88]:
```
import matplotlib.pyplot as plt
import seaborn as sns

# Plot the top 10 cuisines
top_cuisines = cuisine_counts.head(10)

plt.figure(figsize=(20, 10))
sns.barplot(x=top_cuisines.values, y=top_cuisines.index, palette='viridis')
plt.title('Top 10 Cuisines')
plt.xlabel('Number of Restaurants')
plt.ylabel('Cuisine')
plt.show()
```


Top 10 Cuisines
North Indian
North Indian, Chinese
Chinese
Fast Food
North Indian, Mughlai
Cafe
Bakery
North Indian, Mughlai, Chinese
Bakery, Desserts
Street Food
Cuisine
0
200
400
600
800
Number of Restaurants