**ZOMATO RESTURENT**

**Introduction**

**• 1.1 Data Description**

**• 1.2 Feature Description**

**• 1.3 Objective of this project**

**2. Fetching Data**

**• 2.1 Import necessary Libraries**

**• 2.2 Import data**

**3. EDA**

**• 3.1 Checking null values**

**• 3.2 Univariate Analysis**

**• 3.3 Bivariate Analysis**

**• 3.4 Multivariate Analysis**

**4. Encoding Dataframe**

**5. Describe the dataset**

**6. Correlation of the features with target column**

**7. Visualizing correlation of feature columns with label column.**

**8. Checking Distribution of data**

**9. Checking Outliers**

**• 9.1 Removing Outliers**

**• 9.2 Percentage of data loss**

**10. Checking Multicollinearity**

**• 10.1 Droping columns whose have multicollinearity**

**11. Slecting Kbest Features**

**• 11.1 Selecting best features based on ANOVA scores**

**• 11.2 Data skewness Before removing**

**• 11.3 Data Skewness After Removing Skewness**

**12. Building a model for predicting rainfalls (Regression)**

**• 12.1 Checking Multicollinearity**

**• 12.2 Linear Regression**

**• 12.3 Random Forest Regressor**

**• 12.4 Gradient Boosting Regressor**

**• 12.5 KNN Regressor**

**• 12.6 XGBRegressor**

**13. Selecting The Best Model**

**14. Hyperparameter Tuning**

**15. Plot Actual VS Predicting Rainfall**

**16. Documenting the Findings**

**17. Model Save**

**• 17.1 Loading the model**

**• 17.2 Predicting the rainfall**

**18. Building Model for Predicting the Price range**

**• 18.1 Checking multicollinearity**

**• 18.2 Balancing the Dataset**

**• 18.3 LogisticRegression**

**• 18.4 Random Forest Classifier**

**• 18.5 Gradient Boosting Classifier**

**• 18.6 XGBOOST**

**• 18.7 SVC**

**• 18.8 KNN**

**19. Selecting Best Model**

**20. Hyperparameter Tuning**

**21. ROC\_AUC Curve**

**22. Model Save**

**• 22.1 Loading the model**

**• 22.2 Predicting the price range**

**1. Introduction**

**Zomato Data Analysis is one of the most useful analysis for foodies who want to taste the best cuisines of every part of the world which lies in their budget. This analysis is also for those who want to find the value for money restaurants in various parts of the country for the cuisines. Additionally, this analysis caters the needs of people who are striving to get the best cuisine of the country and which locality of that country serves that cuisines with maximum number of restaurants.**

**1.1 Data Description**

**Zomato Data Analysis is one of the most useful analysis for foodies who want to taste the best cuisines of every part of the world which lies in their budget. This analysis is also for those who want to find the value for money restaurants in various parts of the country for the cuisines. Additionally, this analysis caters the needs of people who are striving to get the best cuisine of the country and which locality of that country serves that cuisines with maximum number of restaurants.**

**Data Storage:**

**This problem statement contains two datasets- Zomato.csv and country\_code.csv. Country\_code.csv contains two variables: • Country code • Country name The collected data has been stored in the Comma Separated Value file Zomato.csv. Each restaurant in the dataset is uniquely identified by its Restaurant Id. Every Restaurant contains the following variables:**

**1.2 Features Discription**

**This problem statement contains two datasets- Zomato.csv and country\_code.csv. Country\_code.csv contains two variables: • Country code • Country name The collected data has been stored in the Comma Separated Value file Zomato.csv. Each restaurant in the dataset is uniquely identified by its Restaurant Id. Every Restaurant contains the following variables:**

**• Restaurant Id: Unique id of every restaurant across various cities of the world**

**• Restaurant Name: Name of the restaurant**

**• Country Code: Country in which restaurant is located**

**• City: City in which restaurant is located**

**• Address: Address of the restaurant**

**• Locality: Location in the city**

**• Locality Verbose: Detailed description of the locality**

**• Longitude: Longitude coordinate of the restaurant's location**

**• Latitude: Latitude coordinate of the restaurant's location**

**• Cuisines: Cuisines offered by the restaurant**

**• Average Cost for two: Cost for two people in different currencies**

**• Currency: Currency of the country**

**• Has Table booking: yes/no**

**• Has Online delivery: yes/ no**

**• Is delivering: yes/ no**

**• Switch to order menu: yes/no**

**• Price range: range of price of food**

**• Aggregate Rating: Average rating out of 5**

**• Rating color: depending upon the average rating color**

**• Rating text: text on the basis of rating of rating**

**• Votes: Number of ratings casted by people**

**Problem statement : In this dataset predict 2 things –**

**•**

**1. Average Cost for two**

**•**

**2. Price range**

**1.3 Objective of this project**

**The goal of this machine learning project is to predict the:**

**• a) Average Cost for Two.**

**• b) Price Range.**

**import** pandas **as** pd

**import** numpy **as** np

**import** matplotlib.pyplot **as** plt

**import** seaborn **as** sns

In [2]:

pd**.**set\_option('display.max\_columns', **None**)

In [3]:

df1**=**pd**.**read\_excel('C:\\Users\\santu\\OneDrive\\Desktop\\proj ml\\zomatoo.xlsx')

In [4]:

df1**.**shape

Out[4]:

(9551, 21)

In [5]:

df\_country **=** pd**.**read\_excel('C:\\Users\\santu\\OneDrive\\Desktop\\proj ml\\Country-Code.xlsx')

df\_country**.**head()

Out[5]:

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

In [6]:

data**=**pd**.**merge(df1,df\_country,on**=**'Country Code',how**=**'left')

In [7]:

data**.**head()

Out[7]:

|  | **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** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **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 | 1100 | Botswana Pula(P) | 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 | 1200 | Botswana Pula(P) | 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 | 4000 | Botswana Pula(P) | 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 | 1500 | Botswana Pula(P) | 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 | 1500 | Botswana Pula(P) | Yes | No | No | No | 4 | 4.8 | Dark Green | Excellent | 229 | Phillipines |

In [8]:

data**.**shape

Out[8]:

(9551, 22)

In [9]:

data**.**drop\_duplicates()

data**.**shape

Out[9]:

(9551, 22)

In [10]:

data**.**nunique()

Out[10]:

Restaurant ID 9551

Restaurant Name 7446

Country Code 15

City 141

Address 8918

Locality 1208

Locality Verbose 1265

Longitude 8120

Latitude 8677

Cuisines 1825

Average Cost for two 140

Currency 12

Has Table booking 2

Has Online delivery 2

Is delivering now 2

Switch to order menu 1

Price range 4

Aggregate rating 33

Rating color 6

Rating text 6

Votes 1012

Country 15

dtype: int64

In [11]:

data**.**isnull()**.**sum()

Out[11]:

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

Country 0

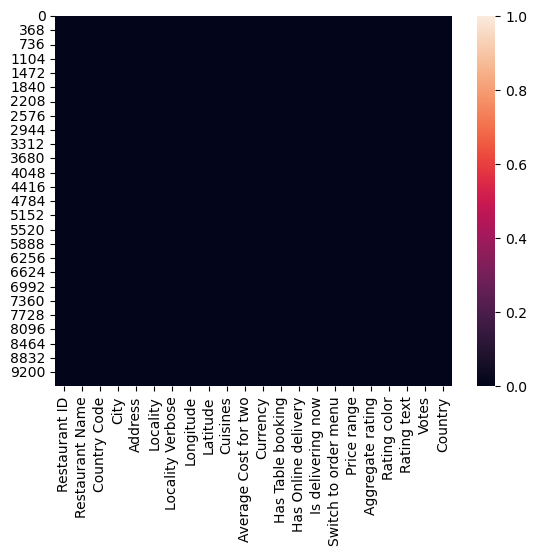
dtype: int64

In [12]:

sns**.**heatmap(data**.**isnull())

Out[12]:

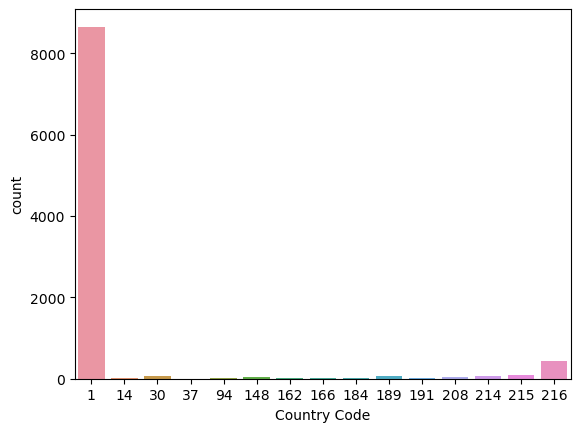
<Axes: >



In [13]:

sns**.**countplot(x**=**'Country Code',data**=**data)

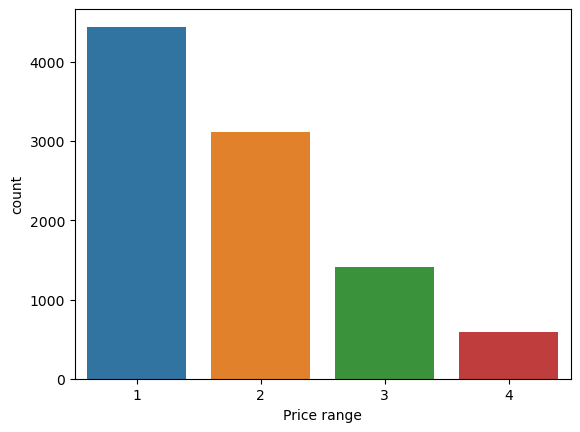
plt**.**show()



In [14]:

sns**.**countplot(x**=**'Price range',data**=**data)

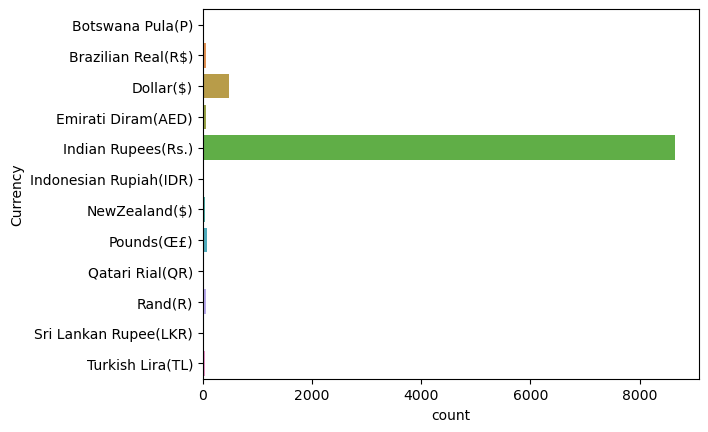
plt**.**show()



In [15]:

sns**.**countplot(y**=**'Currency',data**=**data)

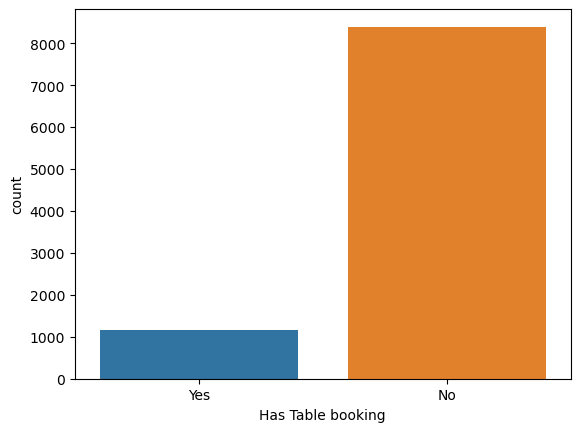
plt**.**show()



In [16]:

sns**.**countplot(x**=**'Has Table booking',data**=**data)

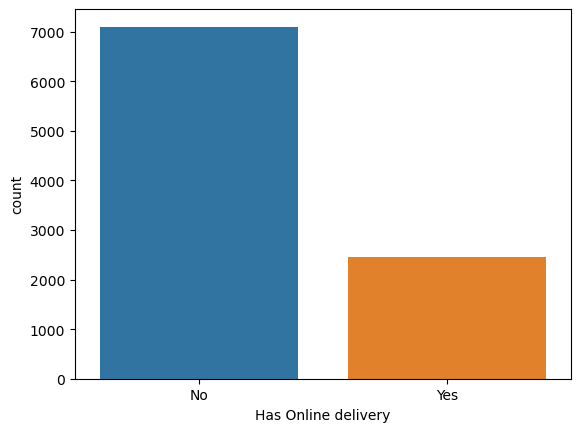
plt**.**show()



In [17]:

sns**.**countplot(x**=**'Has Online delivery',data**=**data)

plt**.**show()



In [18]:

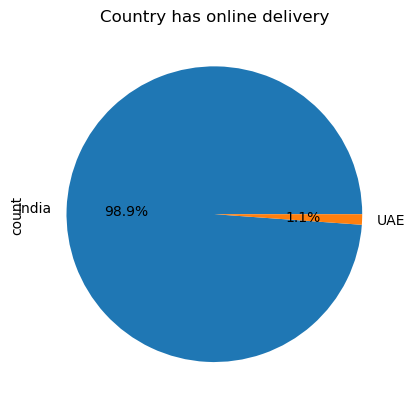
a**=**data[data['Has Online delivery']**==**'Yes']**.**Country**.**value\_counts()

In [19]:

a**.**plot**.**pie(autopct **=** '%1.1f%%')

plt**.**title('Country has online delivery')

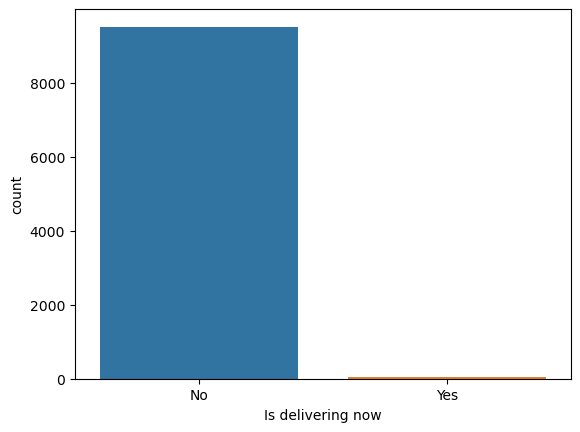
plt**.**show()



In [20]:

sns**.**countplot(x**=**'Is delivering now',data**=**data)

plt**.**show()

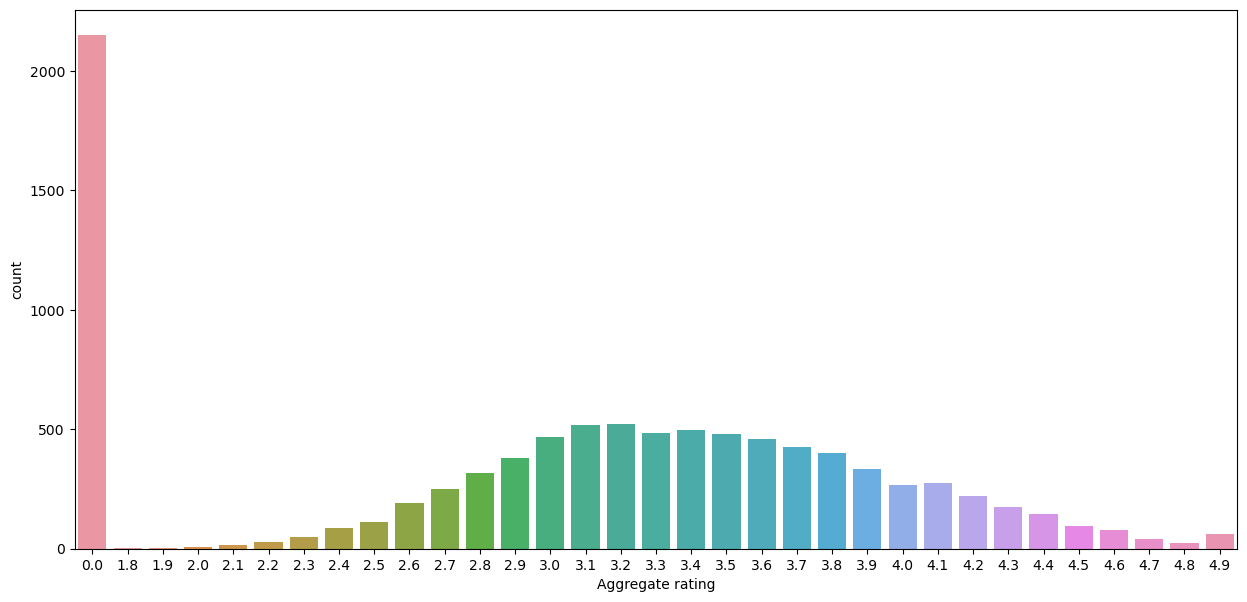


In [21]:

plt**.**figure(figsize**=**(15,7))

sns**.**countplot(x**=**'Aggregate rating',data**=**data)

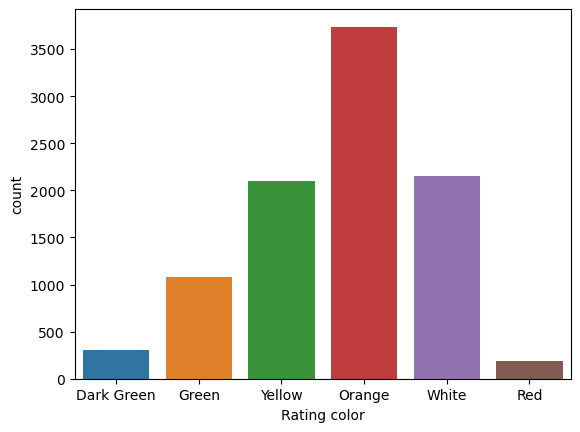
plt**.**show()



In [22]:

sns**.**countplot(x**=**'Rating color',data**=**data,)

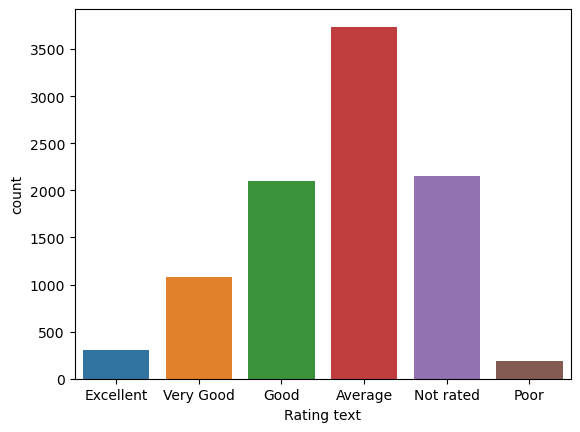
plt**.**show()



In [23]:

sns**.**countplot(x**=**'Rating text',data**=**data,)

plt**.**show()

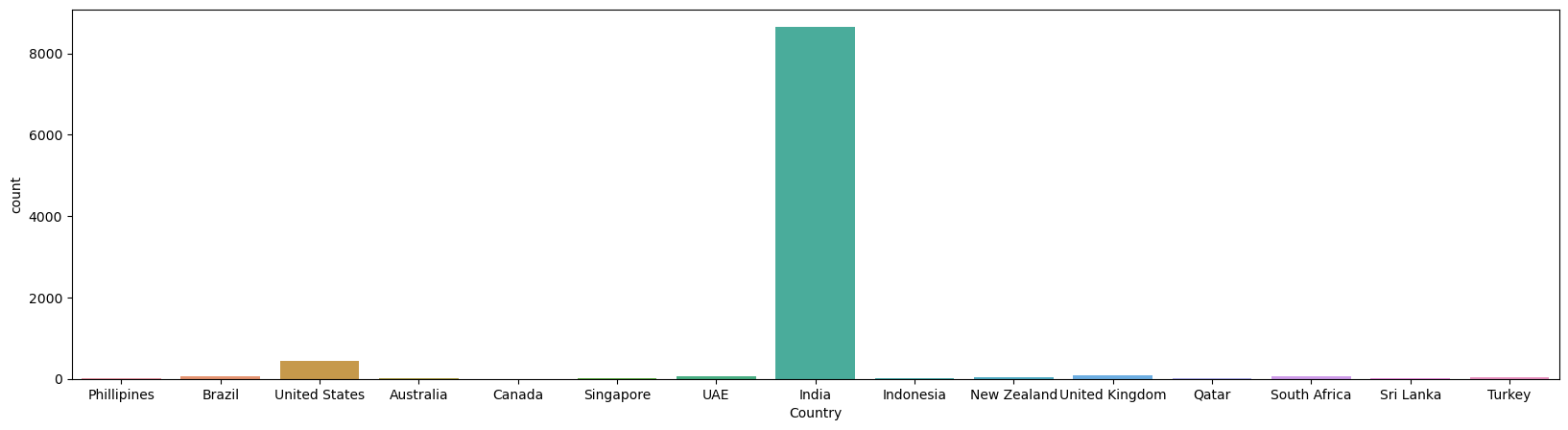


In [24]:

plt**.**figure(figsize**=**(20,5))

sns**.**countplot(x**=**'Country',data**=**data)

plt**.**show()



In [25]:

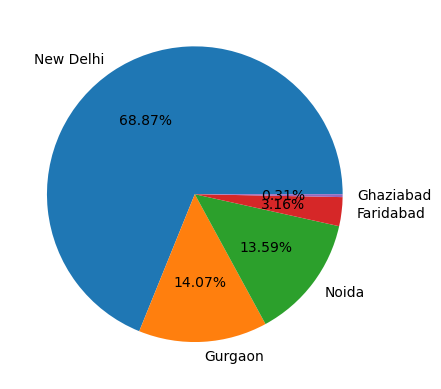
city\_values **=** data**.**City**.**value\_counts()**.**values

city\_labels **=** data**.**City**.**value\_counts()**.**index

In [26]:

plt**.**pie(city\_values[:5],labels**=**city\_labels[:5],autopct**=**'%1.2f%%')

plt**.**show()



In [27]:

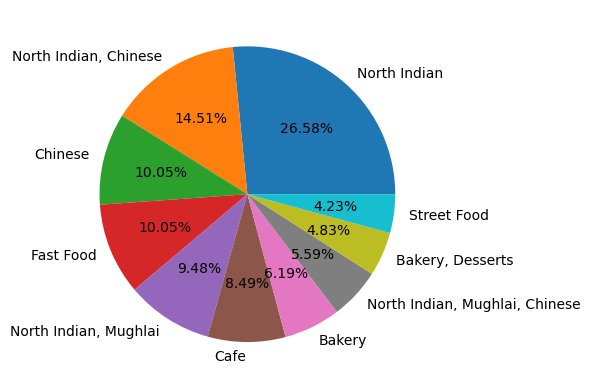
cuisin\_val**=**data**.**Cuisines**.**value\_counts()**.**values

cuisin\_label **=** data**.**Cuisines**.**value\_counts()**.**index

In [28]:

plt**.**pie(cuisin\_val[:10],labels**=**cuisin\_label[:10],autopct**=**'%1.2f%%')

plt**.**show()



In [29]:

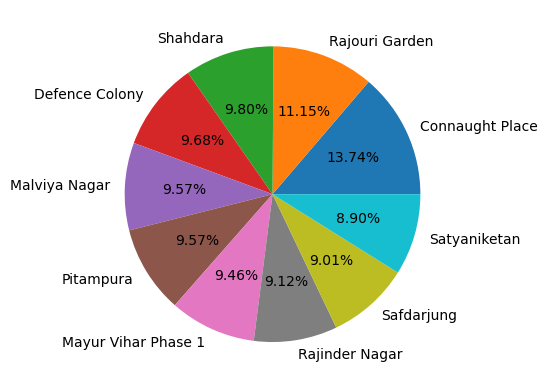
val**=**data**.**Locality**.**value\_counts()**.**values

label **=** data**.**Locality**.**value\_counts()**.**index

In [30]:

plt**.**pie(val[:10],labels**=**label[:10],autopct**=**'%1.2f%%')

plt**.**show()



In [31]:

sns**.**distplot(data['Average Cost for two'])

plt**.**show()

C:\Users\santu\AppData\Local\Temp\ipykernel\_10164\3546603495.py:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

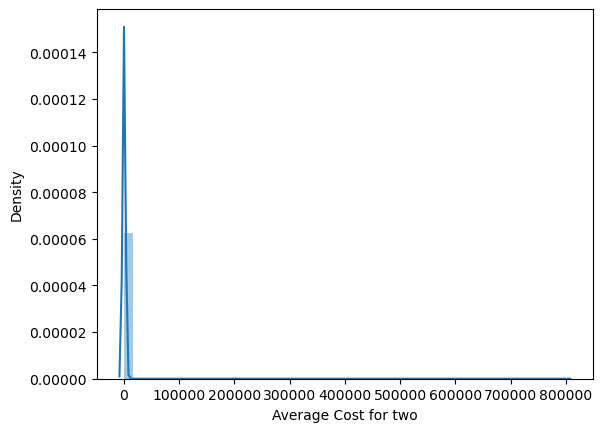
Please adapt your code to use either `displot` (a figure-level function with

similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see

https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

sns.distplot(data['Average Cost for two'])



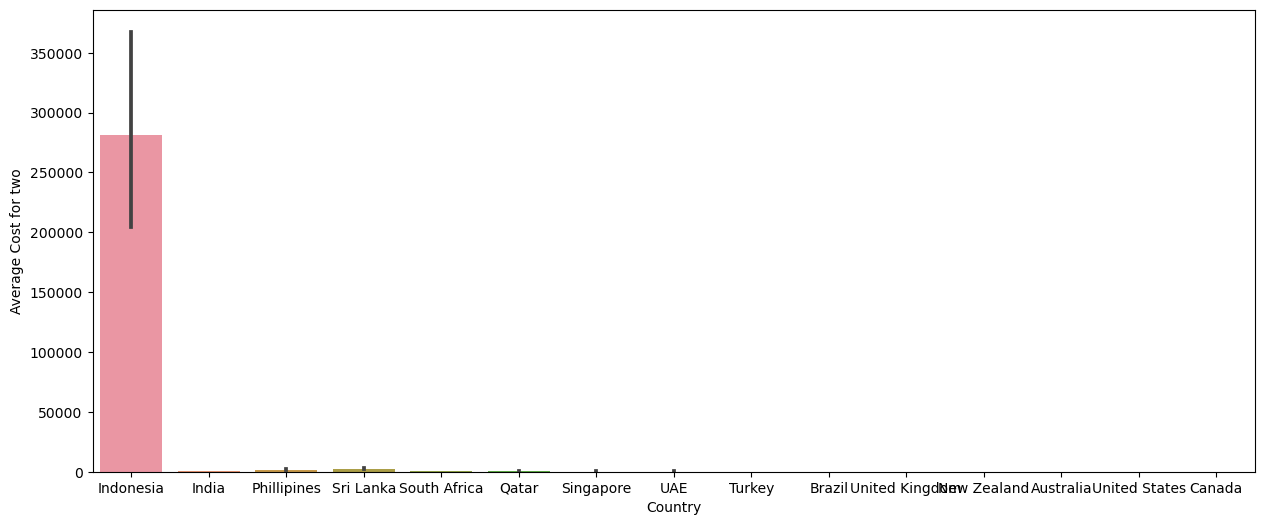
In [32]:

plt**.**figure(figsize**=**(15,6))

df\_good **=** data**.**sort\_values(by**=**"Average Cost for two",ascending**=False**)

sns**.**barplot(x**=**"Country",y**=**"Average Cost for two",data**=**df\_good)

plt**.**show()



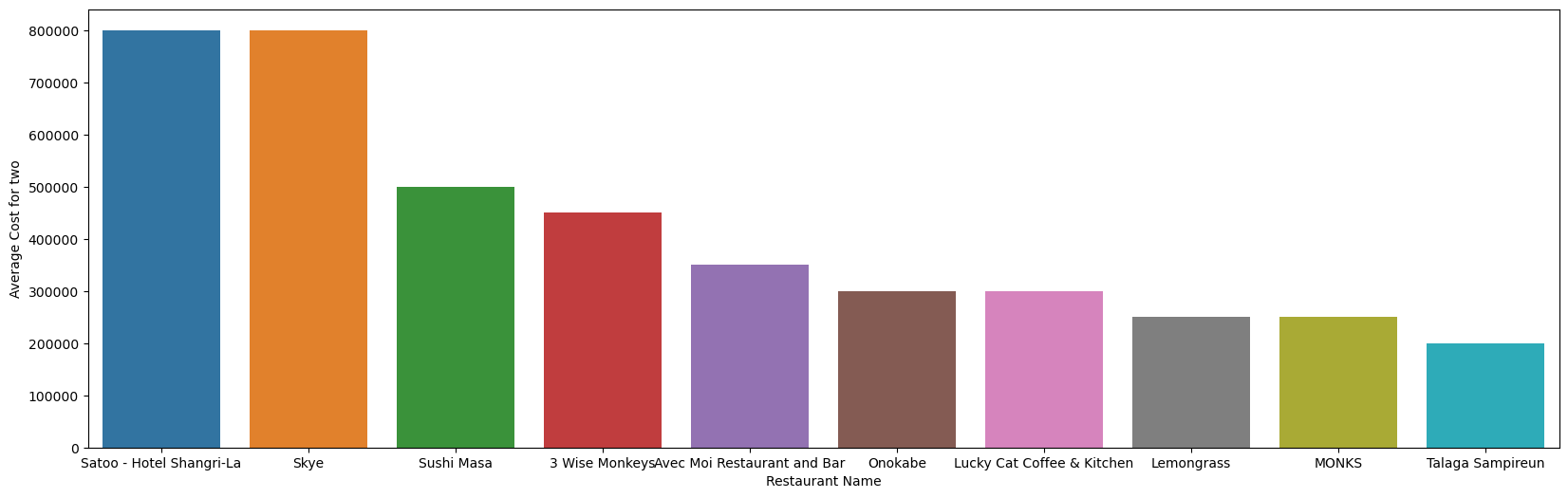
In [33]:

plt**.**figure(figsize**=**(20,6))

df\_good **=** data**.**sort\_values(by**=**"Average Cost for two",ascending**=False**)**.**iloc[0:10]

sns**.**barplot(x**=**"Restaurant Name",y**=**"Average Cost for two",data**=**df\_good)

plt**.**show()

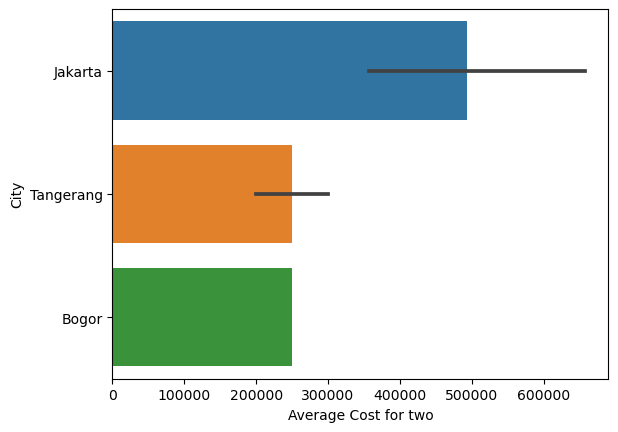


In [34]:

df\_good **=** data**.**sort\_values(by**=**"Average Cost for two",ascending**=False**)**.**iloc[0:10]

sns**.**barplot(y**=**"City",x**=**"Average Cost for two",data**=**df\_good)

plt**.**show()

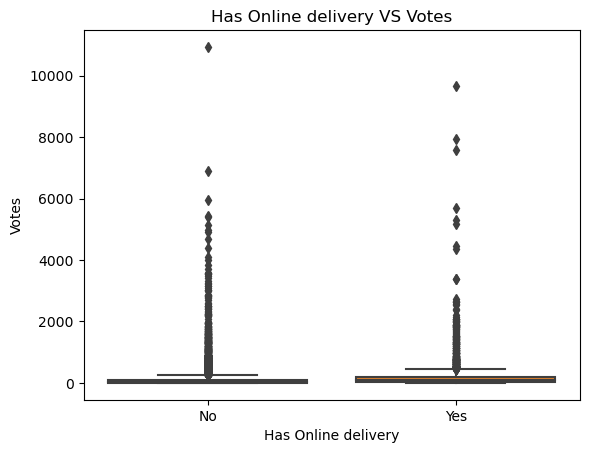


In [35]:

sns**.**boxplot(x**=**'Has Online delivery',y**=**'Votes',data**=**data)

plt**.**title('Has Online delivery VS Votes')

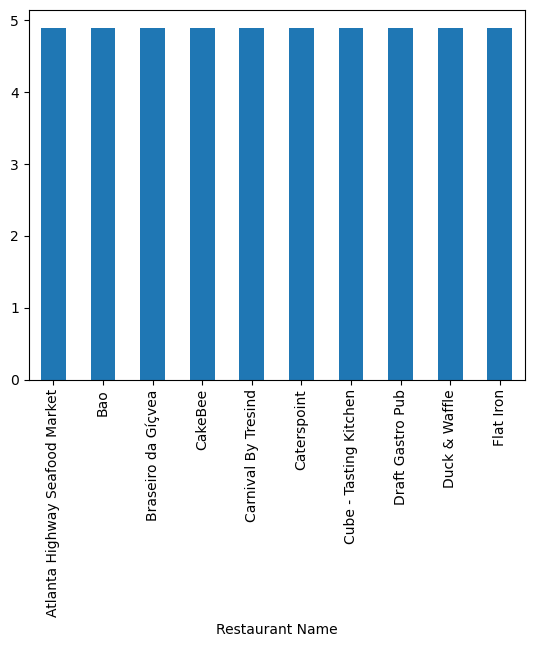
plt**.**show()



In [36]:

data**.**groupby('Restaurant Name')['Aggregate rating']**.**mean()**.**nlargest(10)**.**plot**.**bar()

plt**.**show()



In [37]:

sns**.**boxplot(x**=**'Has Online delivery',y**=**'Average Cost for two',data**=**data)

plt**.**title('Has Online delivery VS Average Cost for two')

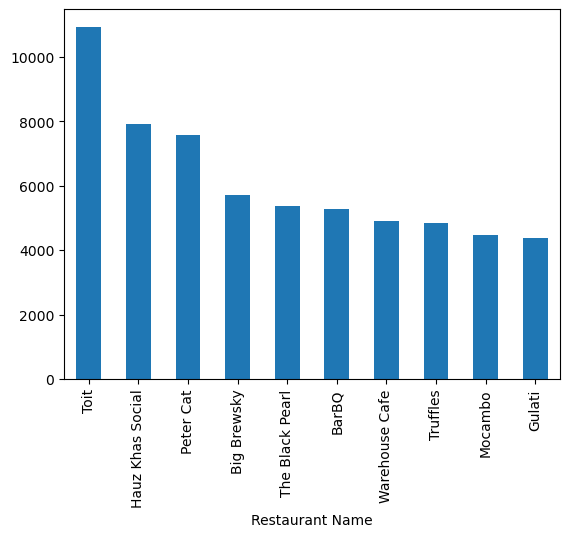
plt**.**show()



In [38]:

data**.**groupby('Restaurant Name')['Votes']**.**mean()**.**nlargest(10)**.**plot**.**bar()

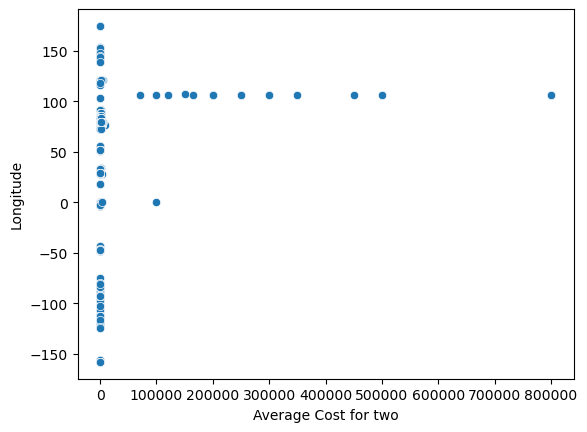
plt**.**show()



In [39]:

sns**.**scatterplot(x**=**'Average Cost for two',y**=**'Longitude',data**=**data)

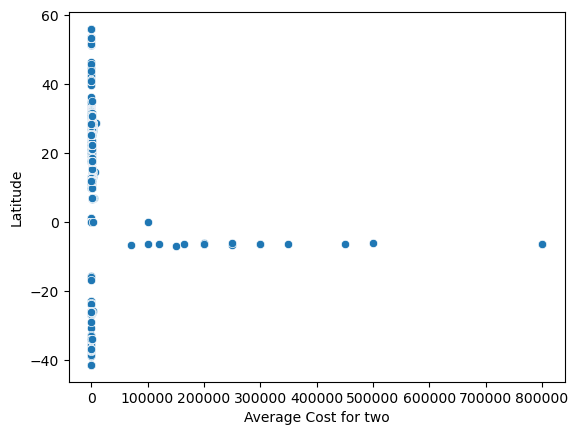
plt**.**show()



In [40]:

sns**.**scatterplot(x**=**'Average Cost for two',y**=**'Latitude',data**=**data)

plt**.**show()



In [41]:

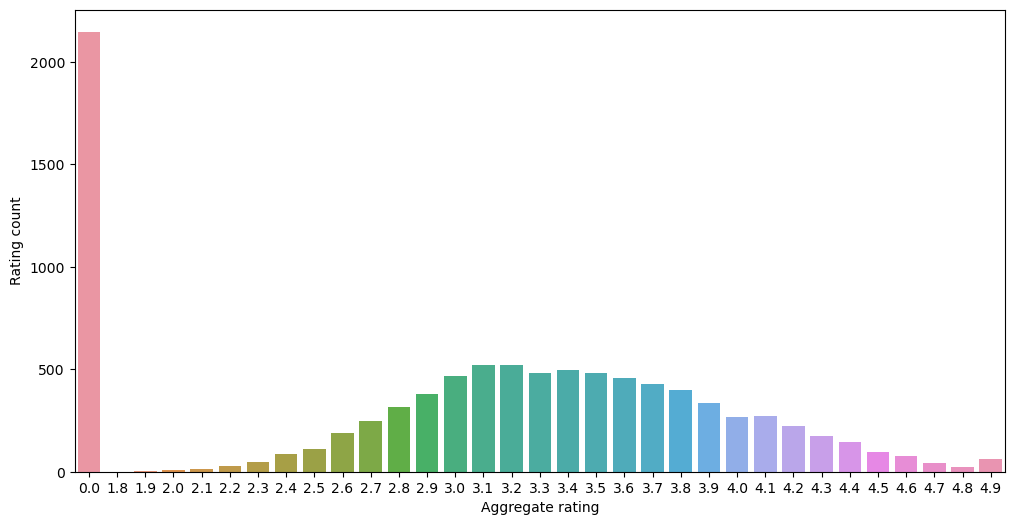
rating**=**data**.**groupby(['Aggregate rating','Rating color','Rating text'])**.**size()**.**reset\_index()**.**rename(columns**=**{0:'Rating count'})

In [42]:

plt**.**figure(figsize**=**(12,6))

sns**.**barplot(x**=**'Aggregate rating',y**=**'Rating count',data**=**rating)

plt**.**show()



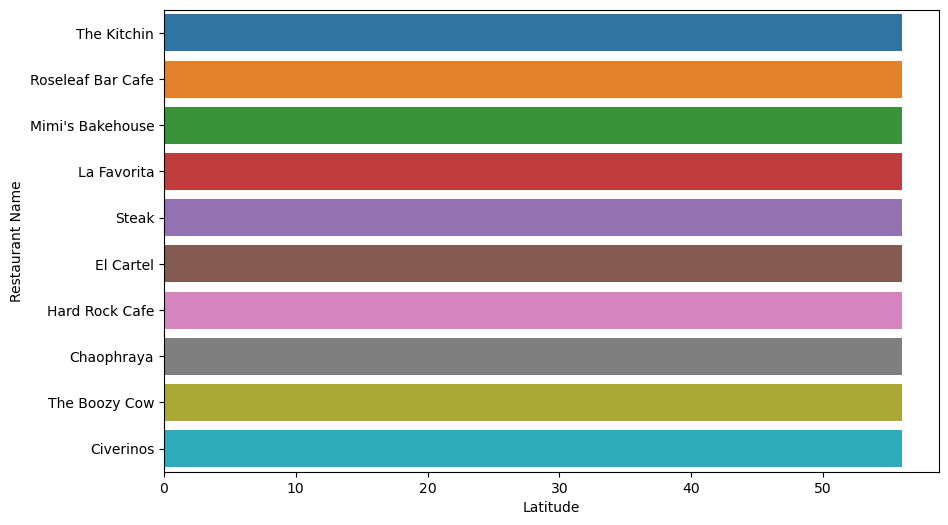
In [43]:

plt**.**figure(figsize**=**(10,6))

df\_good **=** data**.**sort\_values(by**=**"Latitude",ascending**=False**)**.**iloc[0:10,:]

sns**.**barplot(y**=**"Restaurant Name",x**=**"Latitude",data**=**df\_good)

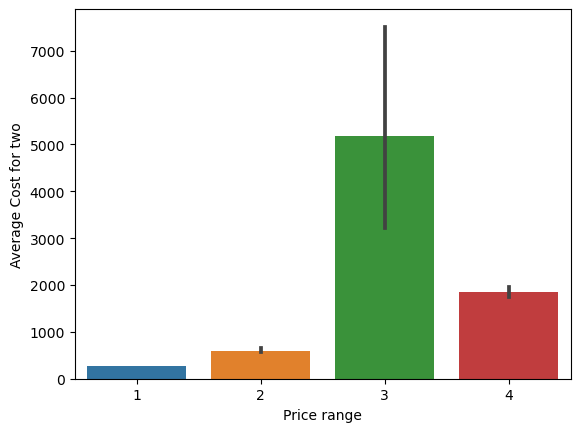
plt**.**show()



In [44]:

sns**.**barplot(x**=**'Price range',y**=**'Average Cost for two',data**=**data)

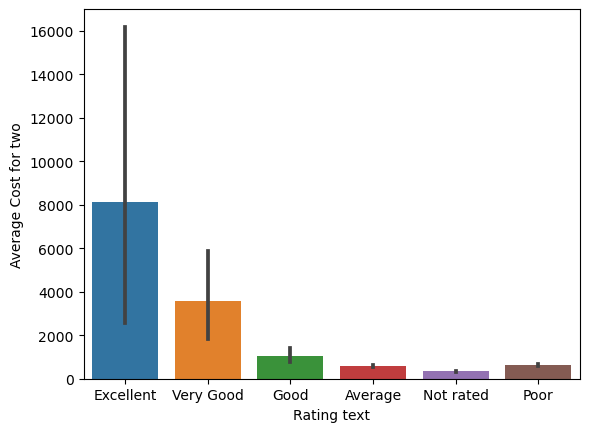
plt**.**show()



In [45]:

sns**.**barplot(x**=**'Rating text',y**=**'Average Cost for two',data**=**data)

plt**.**show()

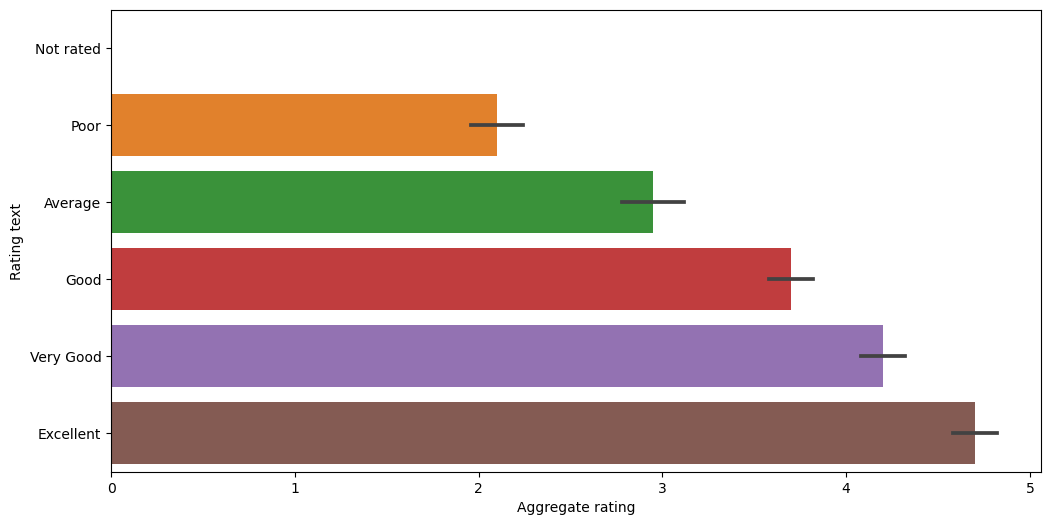


In [46]:

plt**.**figure(figsize**=**(12,6))

sns**.**barplot(x**=**'Aggregate rating',y**=**'Rating text',data**=**rating)

plt**.**show()



In [47]:

data[data['Rating color']**==**'White']**.**groupby(['Aggregate rating','Country'])**.**size()**.**reset\_index()

Out[47]:

|  | **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 |

In [48]:

Delhi **=** data[(data**.**City **==** 'New Delhi')]

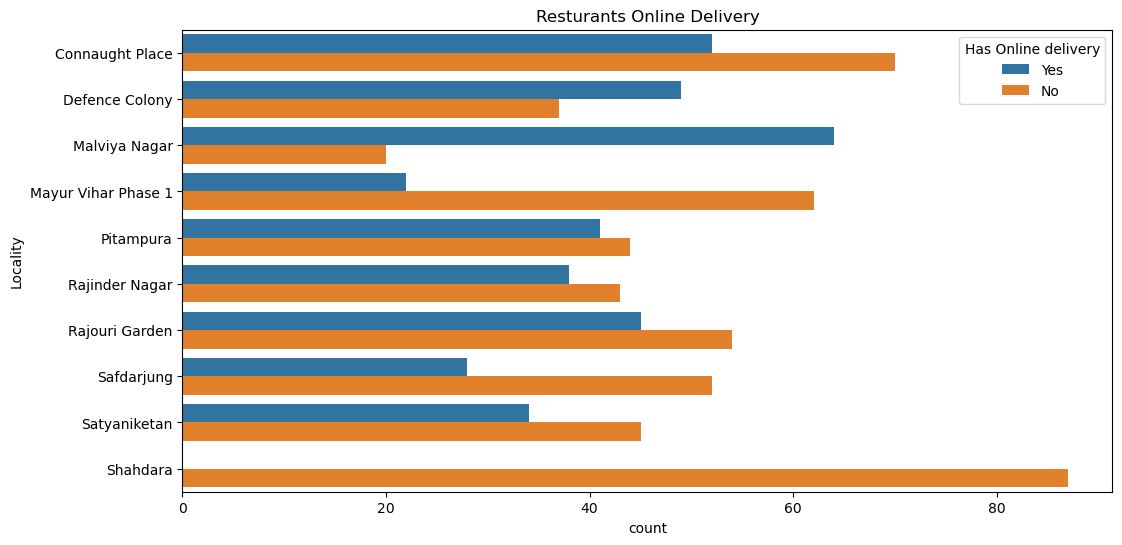
top\_locality **=** Delhi**.**Locality**.**value\_counts()**.**head(10)

plt**.**figure(figsize**=**(12,6))

sns**.**countplot(y**=** "Locality", hue**=**"Has Online delivery", data**=**Delhi[Delhi**.**Locality**.**isin(top\_locality**.**index)])

plt**.**title('Resturants Online Delivery')

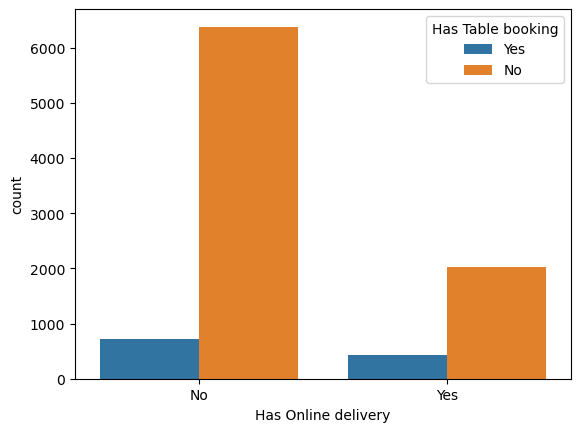
plt**.**show()



In [49]:

sns**.**countplot(x**=**'Has Online delivery',hue**=**'Has Table booking',data**=**data)

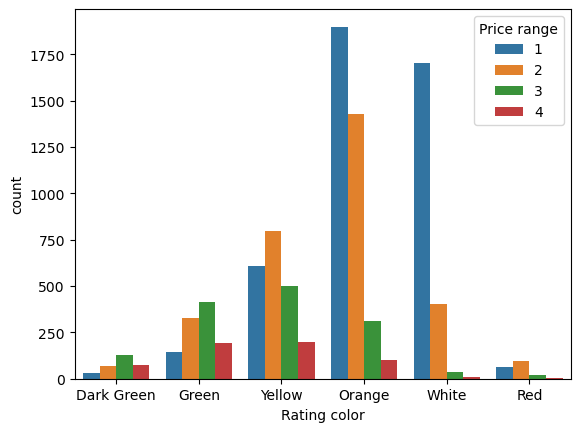
plt**.**show()



In [50]:

sns**.**countplot(x**=**'Rating color',hue**=**'Price range',data**=**data)

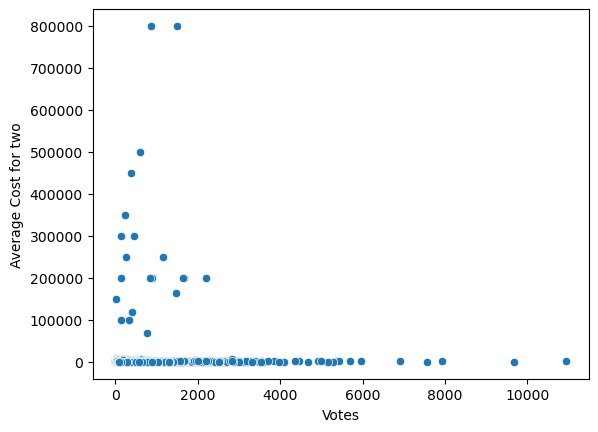
plt**.**show()



In [51]:

sns**.**scatterplot(x**=**'Votes',y**=**'Average Cost for two',data**=**data)

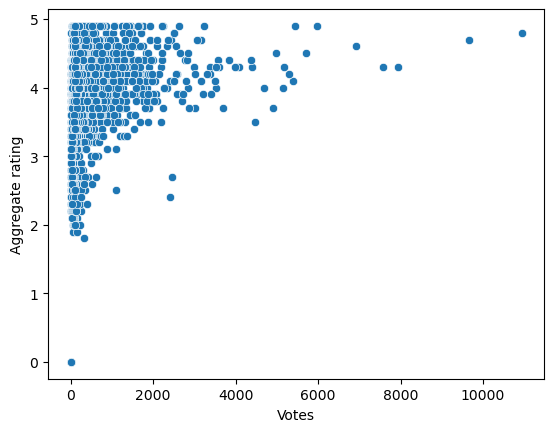
plt**.**show()



In [52]:

sns**.**scatterplot(x**=**'Votes',y**=**'Aggregate rating',data**=**data)

plt**.**show()

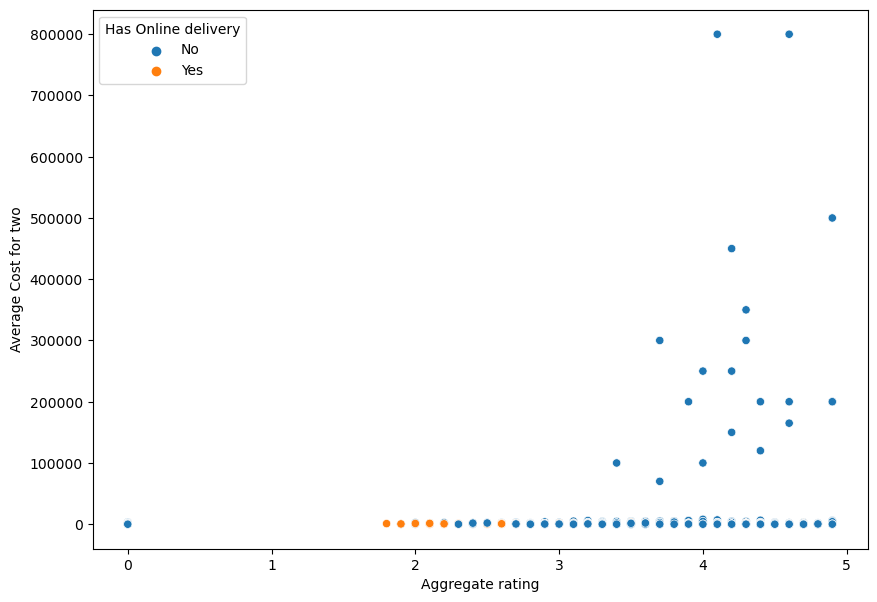


In [53]:

plt**.**figure(figsize**=**(10,7))

sns**.**scatterplot(x**=**'Aggregate rating',y**=**'Average Cost for two',hue**=**'Has Online delivery',data**=**data)

plt**.**show()

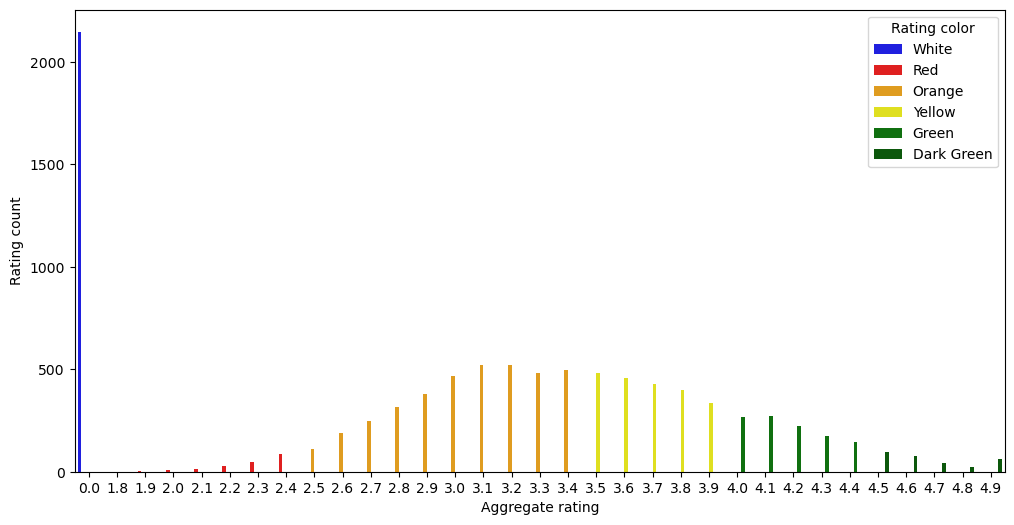


In [54]:

plt**.**figure(figsize**=**(12,6))

sns**.**barplot(x**=**'Aggregate rating',y**=**'Rating count',hue**=**'Rating color',data**=**rating, palette**=**['blue','red','orange','yellow','green','darkgreen'])

plt**.**show()

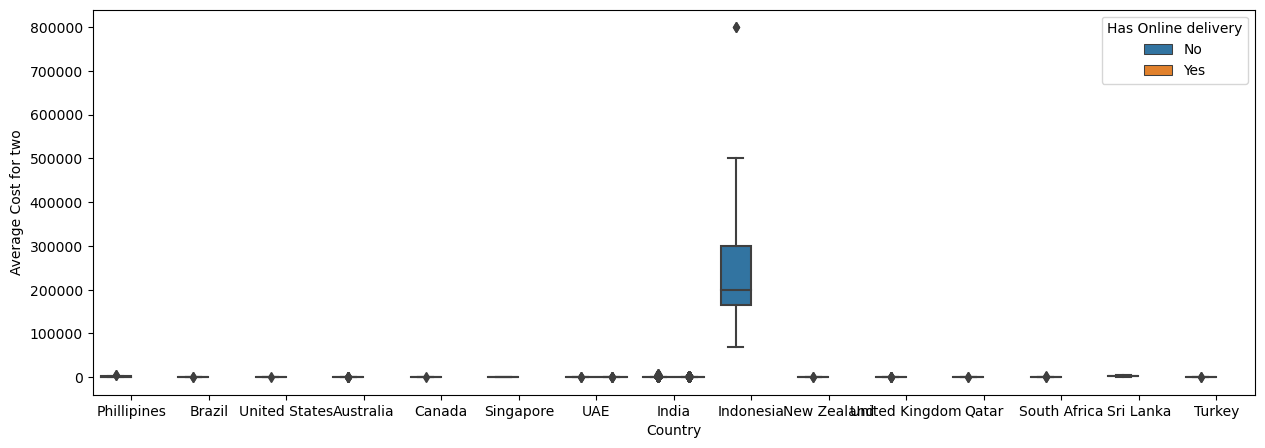


In [55]:

plt**.**figure(figsize**=**(15,5))

sns**.**boxplot(x**=**"Country", y**=**"Average Cost for two", hue**=**"Has Online delivery",data**=**data)

plt**.**show()



In [56]:

plt**.**figure(figsize**=**(12,6))

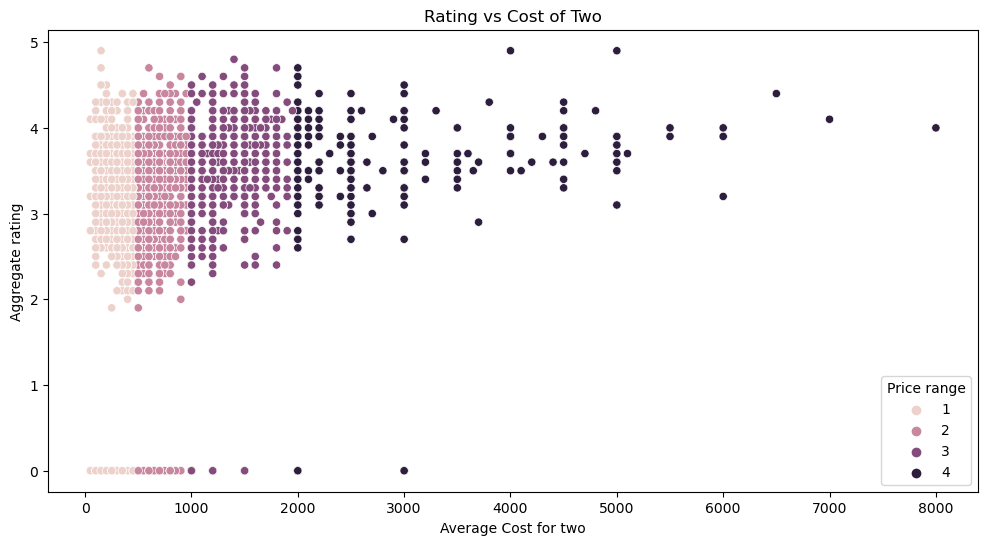
sns**.**scatterplot(x**=**"Average Cost for two", y**=**"Aggregate rating", hue**=**'Price range', data**=**Delhi)

plt**.**xlabel("Average Cost for two")

plt**.**ylabel("Aggregate rating")

plt**.**title('Rating vs Cost of Two')

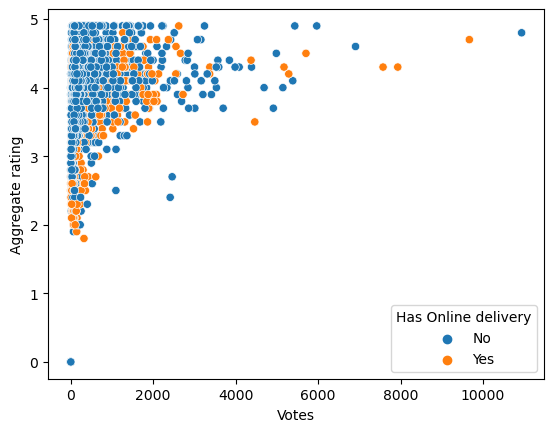
plt**.**show()



In [57]:

sns**.**scatterplot(x**=**'Votes',y**=**'Aggregate rating',data**=**data,hue**=**'Has Online delivery')

plt**.**show()



In [58]:

data**.**drop(columns**=**['Restaurant ID','Locality Verbose'],axis**=**1,inplace**=True**)

In [59]:

data**.**drop(columns**=**['Restaurant Name'],axis**=**1,inplace**=True**)

In [60]:

**from** sklearn.preprocessing **import** LabelEncoder

In [61]:

le**=**LabelEncoder()

In [62]:

data**.**columns

Out[62]:

Index(['Country Code', 'City', 'Address', 'Locality', '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]:

**for** col **in** data[data**.**columns[data**.**dtypes **==** 'object']]:

data[col] **=** le**.**fit\_transform(data[col])

In [64]:

data**.**head()

Out[64]:

|  | **Country Code** | **City** | **Address** | **Locality** | **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** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 162 | 73 | 8685 | 171 | 121.027535 | 14.565443 | 920 | 1100 | 0 | 1 | 0 | 0 | 0 | 3 | 4.8 | 0 | 1 | 314 | 6 |
| **1** | 162 | 73 | 6055 | 593 | 121.014101 | 14.553708 | 1111 | 1200 | 0 | 1 | 0 | 0 | 0 | 3 | 4.5 | 0 | 1 | 591 | 6 |
| **2** | 162 | 75 | 4684 | 308 | 121.056831 | 14.581404 | 1671 | 4000 | 0 | 1 | 0 | 0 | 0 | 4 | 4.4 | 1 | 5 | 270 | 6 |
| **3** | 162 | 75 | 8690 | 862 | 121.056475 | 14.585318 | 1126 | 1500 | 0 | 0 | 0 | 0 | 0 | 4 | 4.9 | 0 | 1 | 365 | 6 |
| **4** | 162 | 75 | 8689 | 862 | 121.057508 | 14.584450 | 1122 | 1500 | 0 | 1 | 0 | 0 | 0 | 4 | 4.8 | 0 | 1 | 229 | 6 |

In [65]:

data**.**corr()['Average Cost for two']**.**sort\_values()

Out[65]:

Latitude -0.111088

Rating color -0.049970

City -0.028838

Has Online delivery -0.018974

Locality -0.012136

Country -0.004034

Is delivering now -0.001862

Cuisines 0.005755

Has Table booking 0.007758

Address 0.022749

Rating text 0.031439

Country Code 0.043225

Currency 0.045615

Longitude 0.045891

Aggregate rating 0.051792

Votes 0.067783

Price range 0.075083

Average Cost for two 1.000000

Switch to order menu NaN

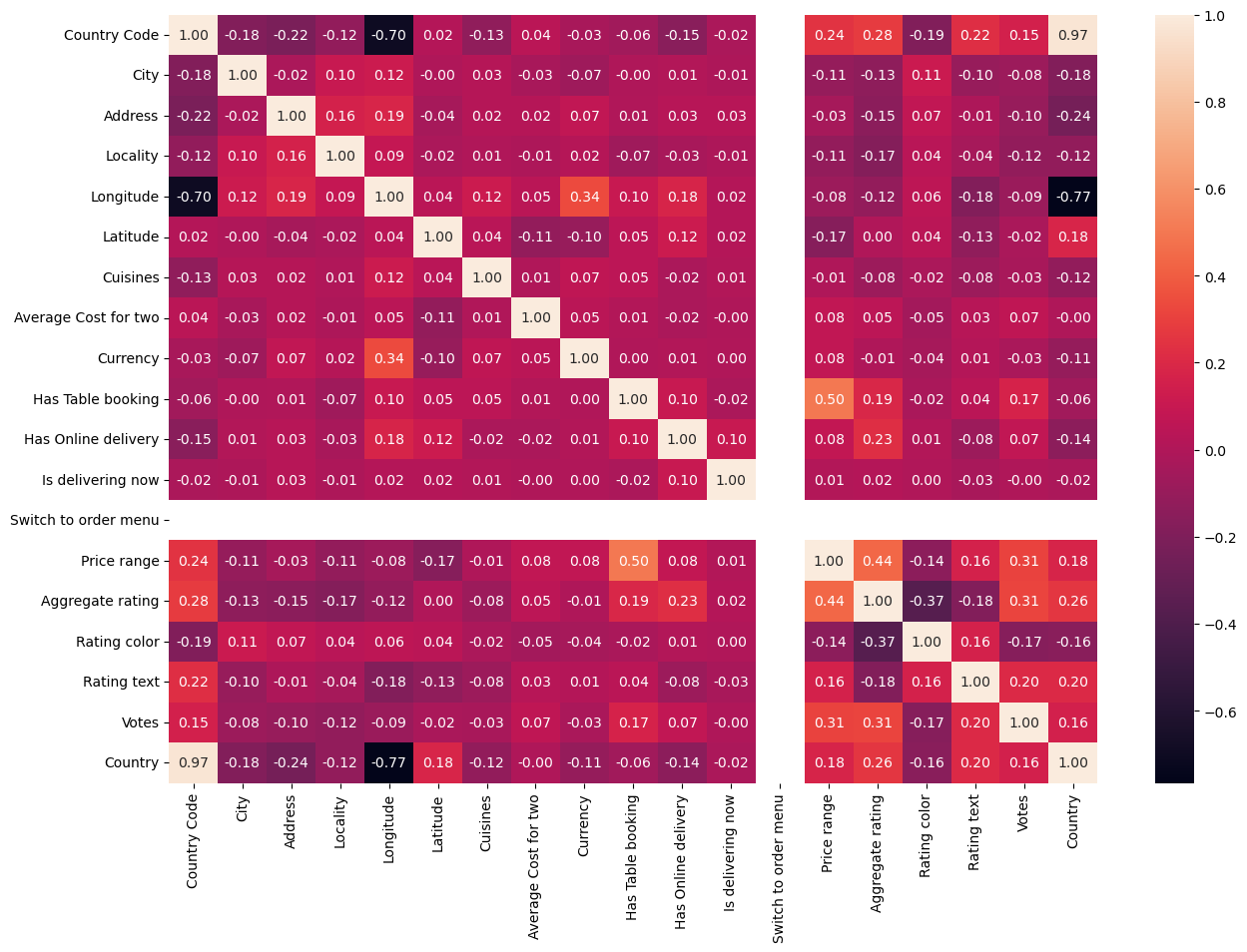
Name: Average Cost for two, dtype: float64

In [66]:

plt**.**figure(figsize**=**(15,10))

sns**.**heatmap(data**.**corr(), annot**=True**, fmt **=**'.2f')

plt**.**show()



In [67]:

plt**.**figure(figsize **=** (20,8))

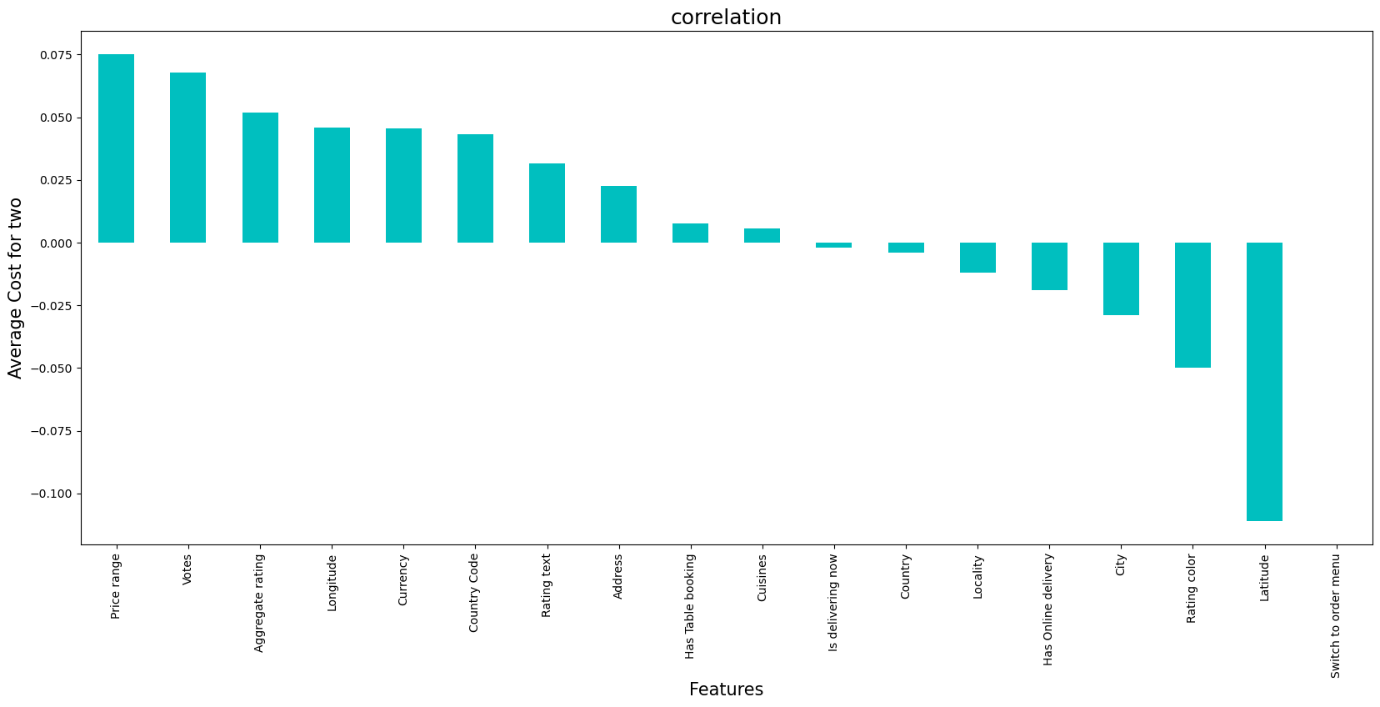
data**.**corr()['Average Cost for two']**.**sort\_values(ascending **=** **False**)**.**drop(['Average Cost for two'])**.**plot(kind**=**'bar',color **=** 'c')

plt**.**xlabel('Features',fontsize**=**15)

plt**.**ylabel('Average Cost for two',fontsize**=**15)

plt**.**title('correlation',fontsize **=** 18)

plt**.**show()



In [68]:

data**.**shape

Out[68]:

(9551, 19)

In [69]:

plt**.**figure(figsize**=**(25,30))

plotnumber **=** 1

**for** column **in** data:

**if** plotnumber **<=**20:

ax **=** plt**.**subplot(5,4,plotnumber)

sns**.**distplot(data[column])

plt**.**xlabel(column, fontsize**=**20)

plotnumber **+=**1

plt**.**show()

C:\Users\santu\AppData\Local\Temp\ipykernel\_10164\2621234774.py:7: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with

similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see

https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

sns.distplot(data[column])

C:\Users\santu\AppData\Local\Temp\ipykernel\_10164\2621234774.py:7: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with

similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see

https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

sns.distplot(data[column])

C:\Users\santu\AppData\Local\Temp\ipykernel\_10164\2621234774.py:7: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with

similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see

https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

sns.distplot(data[column])

C:\Users\santu\AppData\Local\Temp\ipykernel\_10164\2621234774.py:7: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with

similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see

https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

sns.distplot(data[column])

C:\Users\santu\AppData\Local\Temp\ipykernel\_10164\2621234774.py:7: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with

similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see

https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

sns.distplot(data[column])

C:\Users\santu\AppData\Local\Temp\ipykernel\_10164\2621234774.py:7: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with

similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see

https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

sns.distplot(data[column])

C:\Users\santu\AppData\Local\Temp\ipykernel\_10164\2621234774.py:7: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with

similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see

https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

sns.distplot(data[column])

C:\Users\santu\AppData\Local\Temp\ipykernel\_10164\2621234774.py:7: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with

similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see

https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

sns.distplot(data[column])

C:\Users\santu\AppData\Local\Temp\ipykernel\_10164\2621234774.py:7: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with

similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see

https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

sns.distplot(data[column])

C:\Users\santu\AppData\Local\Temp\ipykernel\_10164\2621234774.py:7: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with

similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see

https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

sns.distplot(data[column])

C:\Users\santu\AppData\Local\Temp\ipykernel\_10164\2621234774.py:7: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with

similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see

https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

sns.distplot(data[column])

C:\Users\santu\AppData\Local\Temp\ipykernel\_10164\2621234774.py:7: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with

similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see

https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

sns.distplot(data[column])

C:\Users\santu\AppData\Local\Temp\ipykernel\_10164\2621234774.py:7: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with

similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see

https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

sns.distplot(data[column])

C:\Users\santu\anaconda3\Lib\site-packages\seaborn\distributions.py:2511: UserWarning: Dataset has 0 variance; skipping density estimate. Pass `warn\_singular=False` to disable this warning.

kdeplot(\*\*{axis: a}, ax=ax, color=kde\_color, \*\*kde\_kws)

C:\Users\santu\AppData\Local\Temp\ipykernel\_10164\2621234774.py:7: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with

similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see

https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

sns.distplot(data[column])

C:\Users\santu\AppData\Local\Temp\ipykernel\_10164\2621234774.py:7: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with

similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see

https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

sns.distplot(data[column])

C:\Users\santu\AppData\Local\Temp\ipykernel\_10164\2621234774.py:7: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with

similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see

https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

sns.distplot(data[column])

C:\Users\santu\AppData\Local\Temp\ipykernel\_10164\2621234774.py:7: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with

similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see

https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

sns.distplot(data[column])

C:\Users\santu\AppData\Local\Temp\ipykernel\_10164\2621234774.py:7: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with

similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see

https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

sns.distplot(data[column])

C:\Users\santu\AppData\Local\Temp\ipykernel\_10164\2621234774.py:7: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

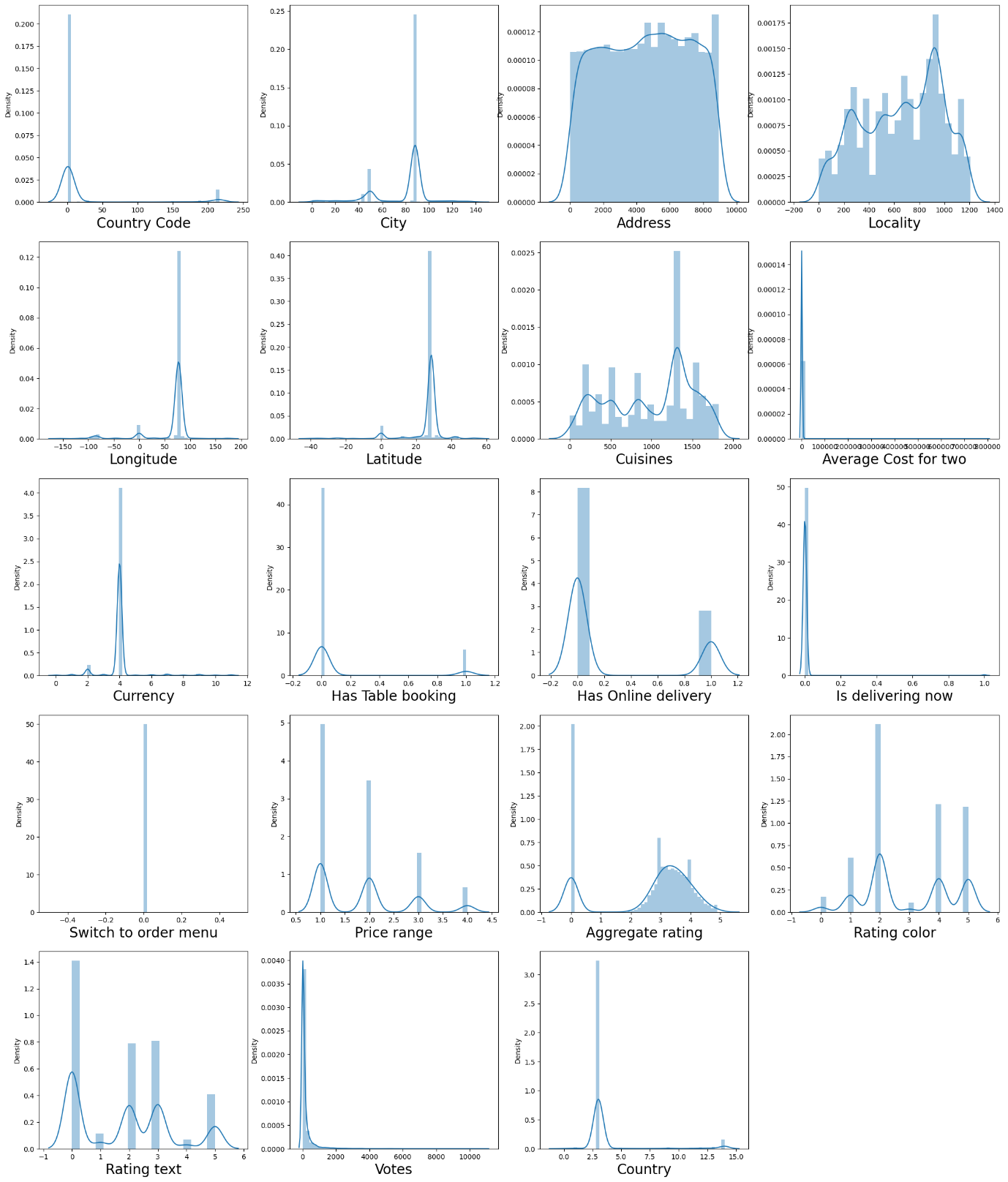
Please adapt your code to use either `displot` (a figure-level function with

similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see

https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

sns.distplot(data[column])



In [70]:

data**.**skew()

Out[70]:

Country Code 3.043965

City -1.193777

Address -0.040407

Locality -0.253554

Longitude -2.807328

Latitude -3.081635

Cuisines -0.305484

Average Cost for two 35.477915

Currency 2.858093

Has Table booking 2.321100

Has Online delivery 1.114620

Is delivering now 16.673412

Switch to order menu 0.000000

Price range 0.889618

Aggregate rating -0.954130

Rating color 0.046009

Rating text 0.431408

Votes 8.807637

Country 3.312721

dtype: float64

In [71]:

plt**.**figure(figsize**=**(25,30))

plotnumber **=** 1

**for** column **in** data:

**if** plotnumber **<=**20:

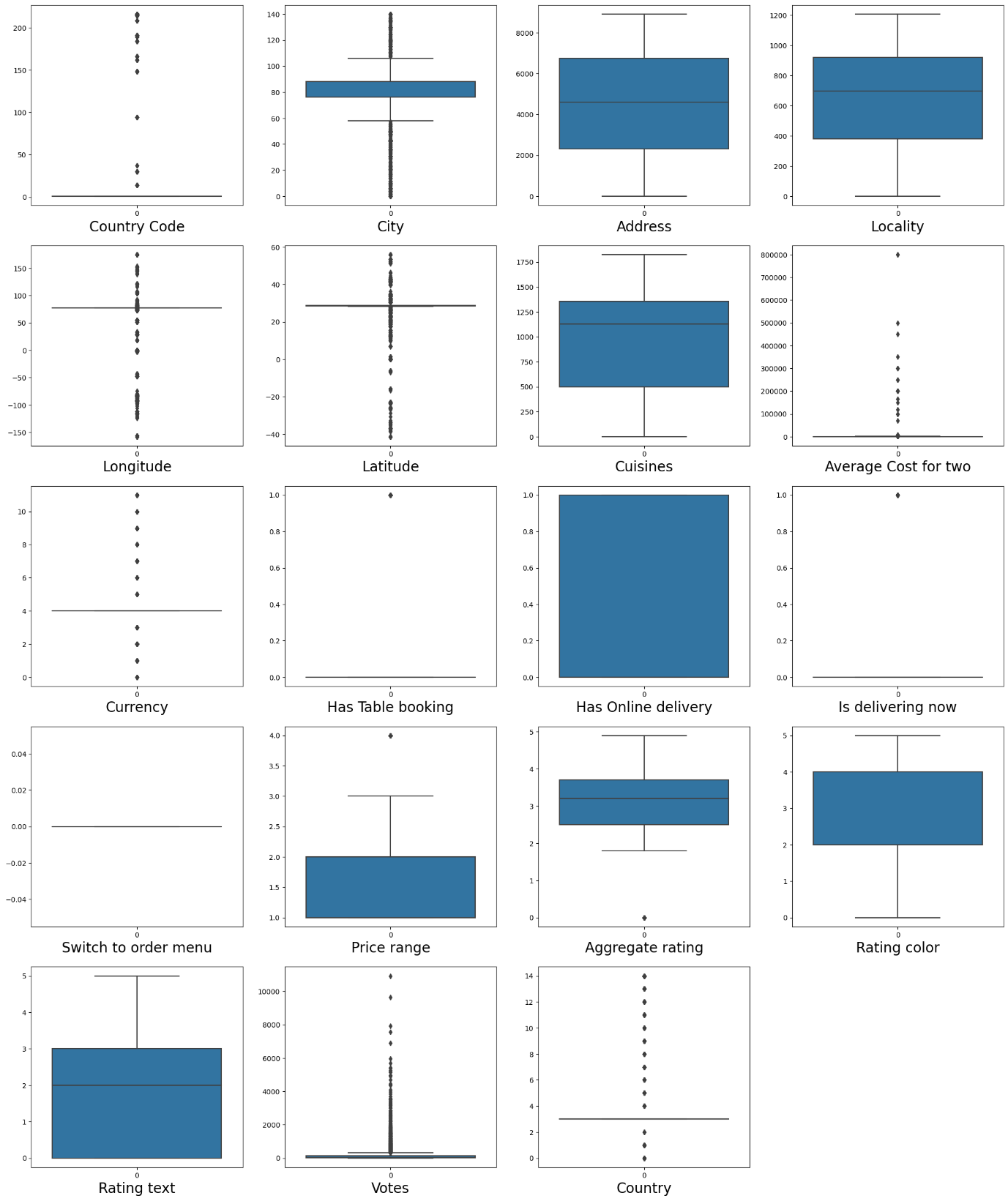
plt**.**subplot(5,4,plotnumber)

ax **=** sns**.**boxplot(data**=**data[column])

plt**.**xlabel(column, fontsize**=**20)

plotnumber **+=**1

plt**.**show()



In [72]:

**from** scipy.stats **import** zscore

z\_score **=** zscore(data[['Average Cost for two','Votes']]) *# Only removing outliers from continuous data*

abs\_z\_score **=** np**.**abs(z\_score) *# Apply the formula and get the scaled data*

filtering\_entry **=** (abs\_z\_score **<** 3)**.**all(axis**=**1)

df **=** data[filtering\_entry]

In [73]:

data\_loss **=** ((9551 **-** 9362)**/**9551**\***100)

print(data\_loss,'%')

1.978850382158936 %

In [74]:

x **=** df**.**drop(columns**=**['Average Cost for two'],axis**=**1)

y **=** df['Average Cost for two']

In [75]:

**from** sklearn.preprocessing **import** StandardScaler

scaler**=** StandardScaler()

scaled\_X **=** scaler**.**fit\_transform(x)

**from** statsmodels.stats.outliers\_influence **import** variance\_inflation\_factor

vif **=** pd**.**DataFrame()

vif["Features"] **=** x**.**columns

vif['vif'] **=** [variance\_inflation\_factor(scaled\_X,i) **for** i **in** range(scaled\_X**.**shape[1])]

vif

C:\Users\santu\anaconda3\Lib\site-packages\statsmodels\regression\linear\_model.py:1783: RuntimeWarning: invalid value encountered in scalar divide

return 1 - self.ssr/self.uncentered\_tss

Out[75]:

|  | **Features** | **vif** |
| --- | --- | --- |
| **0** | Country Code | 48.277935 |
| **1** | City | 1.081790 |
| **2** | Address | 1.106113 |
| **3** | Locality | 1.067813 |
| **4** | Longitude | 5.031265 |
| **5** | Latitude | 3.037436 |
| **6** | Cuisines | 1.040596 |
| **7** | Currency | 1.294002 |
| **8** | Has Table booking | 1.482420 |
| **9** | Has Online delivery | 1.168063 |
| **10** | Is delivering now | 1.014184 |
| **11** | Switch to order menu | NaN |
| **12** | Price range | 1.946198 |
| **13** | Aggregate rating | 1.947933 |
| **14** | Rating color | 1.221293 |
| **15** | Rating text | 1.426034 |
| **16** | Votes | 1.613810 |
| **17** | Country | 66.489374 |

In [76]:

df**.**drop(columns**=**['Country Code','Switch to order menu','Country'],axis**=**1,inplace**=True**)

df**.**shape

C:\Users\santu\AppData\Local\Temp\ipykernel\_10164\722361285.py:1: SettingWithCopyWarning:

A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user\_guide/indexing.html#returning-a-view-versus-a-copy

df.drop(columns=['Country Code','Switch to order menu','Country'],axis=1,inplace=True)

Out[76]:

(9362, 16)

In [77]:

x **=** df**.**drop(columns**=**['Average Cost for two'],axis**=**1)

y **=** df['Average Cost for two']

In [78]:

**from** sklearn.feature\_selection **import** SelectKBest, f\_classif

bestfeat **=** SelectKBest(score\_func **=** f\_classif, k **=** 'all')

fit **=** bestfeat**.**fit(x,y)

dfscores **=** pd**.**DataFrame(fit**.**scores\_)

dfcolumns **=** pd**.**DataFrame(x**.**columns)

fit **=** bestfeat**.**fit(x,y)

dfscores **=** pd**.**DataFrame(fit**.**scores\_)

dfcolumns **=** pd**.**DataFrame(x**.**columns)

dfcolumns**.**head()

featureScores **=** pd**.**concat([dfcolumns,dfscores],axis **=** 1)

featureScores**.**columns **=** ['Feature', 'Score']

print(featureScores**.**nlargest(35,'Score'))

Feature Score

10 Price range 418.202989

3 Longitude 142.353602

7 Has Table booking 80.635590

6 Currency 29.041295

14 Votes 25.818126

11 Aggregate rating 22.275744

4 Latitude 15.805349

8 Has Online delivery 11.752010

1 Address 7.795147

13 Rating text 7.144425

0 City 4.863372

12 Rating color 4.044857

5 Cuisines 3.786064

2 Locality 3.379496

9 Is delivering now 2.063234

In [79]:

x\_best **=** x**.**drop(columns**=**['Is delivering now'])**.**copy()

In [80]:

data**.**skew()

Out[80]:

Country Code 3.043965

City -1.193777

Address -0.040407

Locality -0.253554

Longitude -2.807328

Latitude -3.081635

Cuisines -0.305484

Average Cost for two 35.477915

Currency 2.858093

Has Table booking 2.321100

Has Online delivery 1.114620

Is delivering now 16.673412

Switch to order menu 0.000000

Price range 0.889618

Aggregate rating -0.954130

Rating color 0.046009

Rating text 0.431408

Votes 8.807637

Country 3.312721

dtype: float64

In [81]:

**from** sklearn.preprocessing **import** power\_transform

x **=** power\_transform(x\_best,method**=**'yeo-johnson')

trans **=** pd**.**DataFrame(x)

trans**.**skew()

Out[81]:

0 -0.198500

1 -0.302620

2 -0.306525

3 -0.795230

4 -0.259015

5 -0.382346

6 0.447069

7 2.382516

8 1.120081

9 0.197391

10 -0.615772

11 -0.100142

12 -0.055331

13 -0.021285

dtype: float64

In [82]:

x **=** scaler**.**fit\_transform(x)

In [83]:

x

Out[83]:

array([[-0.40644576, 1.47370479, -1.56895768, ..., -2.26885617,

-0.20420612, 1.30223393],

[-0.40644576, 0.61835876, -0.20264175, ..., -2.26885617,

-0.20420612, 1.65627302],

[-0.310708 , 0.13463254, -1.11660629, ..., -1.38742706,

1.51689551, 1.21890769],

...,

[ 3.88657629, 0.58789144, -0.32621295, ..., 1.32393817,

0.39794913, 1.71976831],

[ 3.88657629, 0.58823442, -0.32621295, ..., -1.38742706,

1.51689551, 1.8967675 ],

[ 3.88657629, -0.0228778 , 0.04024675, ..., -1.38742706,

1.51689551, 1.65627302]])

In [84]:

**from** sklearn.model\_selection **import** train\_test\_split

**from** sklearn.linear\_model **import** LinearRegression

**from** sklearn.tree **import** DecisionTreeRegressor

**from** sklearn.ensemble **import** RandomForestRegressor

**from** sklearn.ensemble **import** GradientBoostingRegressor

**from** sklearn.svm **import** SVR

**from** sklearn.neighbors **import** KNeighborsRegressor

**from** xgboost **import** XGBRegressor

**from** sklearn.model\_selection **import** RandomizedSearchCV

**from** sklearn.metrics **import** mean\_absolute\_error,mean\_squared\_error,r2\_score

In [85]:

maxAccu**=**0

maxRS**=**0

**for** i **in** range(1, 1000):

X\_train, X\_test, y\_train, y\_test **=** train\_test\_split(x, y, test\_size**=**0.3, random\_state**=**i)

lr**=**LinearRegression()

lr**.**fit(X\_train, y\_train)

pred **=** lr**.**predict(X\_test)

r2 **=** r2\_score(y\_test, pred)

**if** r2**>**maxAccu:

maxAccu**=**r2

maxRS**=**i

print("Best r2 score is", maxAccu,"on Random State", maxRS)

Best r2 score is 0.5877951971226656 on Random State 970

In [86]:

X\_train, X\_test, y\_train, y\_test **=** train\_test\_split(x, y, train\_size**=**0.7, test\_size**=**0.3, random\_state**=**16)

In [87]:

regressors **=** {

'Linear Regression' : LinearRegression(),

}

results**=**pd**.**DataFrame(columns**=**['MAE','MSE', 'RMSE', 'R2-score'])

**for** method,func **in** regressors**.**items():

model **=** func**.**fit(X\_train,y\_train)

pred **=** model**.**predict(X\_test)

results**.**loc[method]**=** [np**.**round(mean\_absolute\_error(y\_test,pred),3),

np**.**round(mean\_squared\_error(y\_test,pred),3),

np**.**sqrt(mean\_squared\_error(y\_test,pred)),

np**.**round(r2\_score(y\_test,pred),3)

]

In [88]:

y\_pred **=** lr**.**predict(X\_test)

**from** sklearn.model\_selection **import** cross\_val\_score

lss **=** r2\_score(y\_test,y\_pred)

**for** j **in** range(4,10):

isscore **=** cross\_val\_score(lr,x,y,cv**=**j)

lsc **=** isscore**.**mean()

print("At cv:-",j)

print('Cross validation score is:-',lsc**\***100)

print('accuracy\_score is:-',lss**\***100)

print('\n')

At cv:- 4

Cross validation score is:- 24.207194415529788

accuracy\_score is:- 58.007073804126165

At cv:- 5

Cross validation score is:- 4.2502318628714475

accuracy\_score is:- 58.007073804126165

At cv:- 6

Cross validation score is:- 8.493870846649259

accuracy\_score is:- 58.007073804126165

At cv:- 7

Cross validation score is:- 3.1460733532034233

accuracy\_score is:- 58.007073804126165

At cv:- 8

Cross validation score is:- -0.35845546319413224

accuracy\_score is:- 58.007073804126165

At cv:- 9

Cross validation score is:- -5.626467703225356

accuracy\_score is:- 58.007073804126165

In [89]:

lsscore\_selected **=** cross\_val\_score(lr,x,y,cv**=**4)**.**mean()

print("The cv score is: ",lsscore\_selected,"\nThe accuracy score is: ",lss)

The cv score is: 0.24207194415529787

The accuracy score is: 0.5800707380412616

In [ ]:

In [90]:

X\_train, X\_test, y\_train, y\_test **=** train\_test\_split(x, y, train\_size**=**0.7, test\_size**=**0.3, random\_state**=**420)

In [91]:

regressors **=** {

'Random Forest' : RandomForestRegressor(),

}

results**=**pd**.**DataFrame(columns**=**['MAE','MSE', 'RMSE', 'R2-score'])

**for** method,func **in** regressors**.**items():

model **=** func**.**fit(X\_train,y\_train)

pred **=** model**.**predict(X\_test)

results**.**loc[method]**=** [np**.**round(mean\_absolute\_error(y\_test,pred),3),

np**.**round(mean\_squared\_error(y\_test,pred),3),

np**.**sqrt(mean\_squared\_error(y\_test,pred)),

np**.**round(r2\_score(y\_test,pred),3)

]

In [92]:

rf **=** RandomForestRegressor()

rf**.**fit(X\_train,y\_train)

y\_pred **=** rf**.**predict(X\_test)

lss **=** r2\_score(y\_test,y\_pred)

In [93]:

**for** j **in** range(4,10):

isscore **=** cross\_val\_score(rf,x,y,cv**=**j)

lsc **=** isscore**.**mean()

print("At cv:-",j)

print('Cross validation score is:-',lsc**\***100)

print('accuracy\_score is:-',lss**\***100)

print('\n')

At cv:- 4

Cross validation score is:- 60.3713065621424

accuracy\_score is:- 91.31843160665592

At cv:- 5

Cross validation score is:- 43.31757209746776

accuracy\_score is:- 91.31843160665592

At cv:- 6

Cross validation score is:- 47.30270899654251

accuracy\_score is:- 91.31843160665592

At cv:- 7

Cross validation score is:- 43.95158013059741

accuracy\_score is:- 91.31843160665592

At cv:- 8

Cross validation score is:- 41.151564250899334

accuracy\_score is:- 91.31843160665592

At cv:- 9

Cross validation score is:- 39.559811288918

accuracy\_score is:- 91.31843160665592

In [94]:

lsscore\_selected **=** cross\_val\_score(rf,x,y,cv**=**4)**.**mean()

print("The cv score is: ",lsscore\_selected,"\nThe accuracy score is: ",lss)

The cv score is: 0.6003217691321973

The accuracy score is: 0.9131843160665593

In [ ]:

maxAccu**=**0

maxRS**=**0

**for** i **in** range(1, 1000):

X\_train, X\_test, y\_train, y\_test **=** train\_test\_split(x, y, test\_size**=**0.3, random\_state**=**i)

gbr **=** GradientBoostingRegressor()

gbr**.**fit(X\_train, y\_train)

pred **=** gbr**.**predict(X\_test)

r2 **=** r2\_score(y\_test, pred)

**if** r2**>**maxAccu:

maxAccu**=**r2

maxRS**=**i

print("Best r2 score is", maxAccu,"on Random State", maxRS)

In [ ]:

X\_train, X\_test, y\_train, y\_test **=** train\_test\_split(x, y, train\_size**=**0.7, test\_size**=**0.3, random\_state**=**891)

In [ ]:

regressors **=** {

'Gradient Boost Regressor' : GradientBoostingRegressor(),

}

results**=**pd**.**DataFrame(columns**=**['MAE','MSE', 'RMSE', 'R2-score'])

**for** method,func **in** regressors**.**items():

model **=** func**.**fit(X\_train,y\_train)

pred **=** model**.**predict(X\_test)

results**.**loc[method]**=** [np**.**round(mean\_absolute\_error(y\_test,pred),3),

np**.**round(mean\_squared\_error(y\_test,pred),3),

np**.**sqrt(mean\_squared\_error(y\_test,pred)),

np**.**round(r2\_score(y\_test,pred),3)

]

In [ ]:

gbr **=** GradientBoostingRegressor()

gbr**.**fit(X\_train,y\_train)

y\_pred **=** gbr**.**predict(X\_test)

**from** sklearn.model\_selection **import** cross\_val\_score

lss **=** r2\_score(y\_test,y\_pred)

In [ ]:

**for** j **in** range(4,10):

isscore **=** cross\_val\_score(gbr,x,y,cv**=**j)

lsc **=** isscore**.**mean()

print("At cv:-",j)

print('Cross validation score is:-',lsc**\***100)

print('accuracy\_score is:-',lss**\***100)

print('\n')

In [ ]:

lsscore\_selected **=** cross\_val\_score(gbr,x,y,cv**=**4)**.**mean()

print("The cv score is: ",lsscore\_selected,"\nThe accuracy score is: ",lss)

In [ ]:

maxAccu**=**0

maxRS**=**0

**for** i **in** range(1, 1000):

X\_train, X\_test, y\_train, y\_test **=** train\_test\_split(x, y, test\_size**=**0.3, random\_state**=**i)

knn **=** KNeighborsRegressor()

knn**.**fit(X\_train, y\_train)

pred **=** knn**.**predict(X\_test)

r2 **=** r2\_score(y\_test, pred)

**if** r2**>**maxAccu:

maxAccu**=**r2

maxRS**=**i

print("Best r2 score is", maxAccu,"on Random State", maxRS)

In [ ]:

X\_train, X\_test, y\_train, y\_test **=** train\_test\_split(x, y, train\_size**=**0.7, test\_size**=**0.3, random\_state**=**358)

regressors **=** {

'KNN Regressor': KNeighborsRegressor()

}

results**=**pd**.**DataFrame(columns**=**['MAE','MSE', 'RMSE', 'R2-score'])

**for** method,func **in** regressors**.**items():

model **=** func**.**fit(X\_train,y\_train)

pred **=** model**.**predict(X\_test)

results**.**loc[method]**=** [np**.**round(mean\_absolute\_error(y\_test,pred),3),

np**.**round(mean\_squared\_error(y\_test,pred),3),

np**.**sqrt(mean\_squared\_error(y\_test,pred)),

np**.**round(r2\_score(y\_test,pred),3)

]

In [ ]:

knn **=** KNeighborsRegressor()

knn**.**fit(X\_train,y\_train)

y\_pred **=** knn**.**predict(X\_test)

lss **=** r2\_score(y\_test,y\_pred)

**for** j **in** range(4,10):

isscore **=** cross\_val\_score(knn,x,y,cv**=**j)

lsc **=** isscore**.**mean()

print("At cv:-",j)

print('Cross validation score is:-',lsc**\***100)

print('accuracy\_score is:-',lss**\***100)

print('\n')