In [1]:
```
import pandas as pd
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

EDA:-Importing, Understanding and Inspecting the DATA

1. Perform preliminary data inspection and report the findings as the structure of the data, missing values, duplicates etc

2. Based on the findings from the previous questions, identify duplicates and remove them

In [2]:
```
# Loading the Dataset
df = pd.read_excel('data.xlsx')
```

In [3]:
```
df.head()
```

Out[3]:

	Restaurant ID	Restaurant Name	Country Code	City	Address	Locality	Locality Verbose	Longitude	Latitude	Cuisines	Average Cost for two	Currency	bo
0	7402935	Skye	94	Jakarta	Menara BCA, Lantai 56, Jl. MH. Thamrin, Thamri...	Grand Indonesia Mall, Thamrin	Grand Indonesia Mall, Thamrin, Jakarta	106.821999	-6.196778	Italian, Continental	800000	Indonesian Rupiah(IDR)	
1	7410290	Satoo - Hotel Shangri-La	94	Jakarta	Hotel Shangri-La, Jl. Jend. Sudirman	Hotel Shangri-La, Sudirman	Hotel Shangri-La, Sudirman, Jakarta	106.818961	-6.203292	Asian, Indonesian, Western	800000	Indonesian Rupiah(IDR)	
2	7420899	Sushi Masa	94	Jakarta	Jl. Tuna Raya No. 5, Penjaringan	Penjaringan	Penjaringan, Jakarta	106.800144	-6.101298	Sushi, Japanese	500000	Indonesian Rupiah(IDR)	
3	7421967	3 Wise Monkeys	94	Jakarta	Jl. Suryo No. 26, Senopati, Jakarta	Senopati	Senopati, Jakarta	106.813400	-6.235241	Japanese	450000	Indonesian Rupiah(IDR)	
4	7422489	Avec Moi Restaurant and Bar	94	Jakarta	Gedung PIC, Jl. Teluk Betung 43, Thamrin, Jakarta	Thamrin	Thamrin, Jakarta	106.821023	-6.196270	French, Western	350000	Indonesian Rupiah(IDR)	

In [4]:
```
df.describe()
```

Out[4]:

	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 [5]:
```
# Checking for null values
df.isnull().sum()
```

Out[5]:
```
Restaurant ID           0
Restaurant Name         1
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
Price range             0
Aggregate rating        0
Rating color            0
Rating text             0
Votes                   0
dtype: int64
```

In [6]:
```
# Droping the null values as only 10 values are present
df = df.dropna().reset_index(drop=True)
```

In [7]:
```
df.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                0
Average Cost for two    0
Currency                0
Has Table booking       0
Has Online delivery     0
Price range             0
Aggregate rating        0
Rating color            0
Rating text             0
Votes                   0
dtype: int64
```

In [8]:
```
# EDA
df.describe()
```

Out[8]:

	Restaurant ID	Country Code	Longitude	Latitude	Average Cost for two	Price range	Aggregate rating	Votes
count	9.541000e+03	9541.000000	9541.000000	9541.000000	9541.000000	9541.000000	9541.000000	9541.000000
mean	9.044236e+06	18.181008	64.274135	25.848826	1200.368096	1.804842	2.665088	156.707892
std	8.791953e+06	56.454284	41.199675	11.010633	16129.588655	0.905528	1.516596	430.180201
min	5.300000e+01	1.000000	-157.948486	-41.330428	0.000000	1.000000	0.000000	0.000000
25%	3.019320e+05	1.000000	77.081601	28.478683	250.000000	1.000000	2.500000	5.000000
50%	6.003426e+06	1.000000	77.192035	28.570444	400.000000	2.000000	3.200000	31.000000
75%	1.835266e+07	1.000000	77.282045	28.642713	700.000000	2.000000	3.700000	130.000000
max	1.850065e+07	216.000000	174.832089	55.976980	800000.000000	4.000000	4.900000	10934.000000

In [23]:
```
# Doing EDA with the help of pandas_profiling library
import pandas_profiling as pp
profile = pp.ProfileReport(df)
profile.to_file("output.html")
profile
```

Overview

Dataset statistics

Number of variables	20
Number of observations	9541
Missing cells	0
Missing cells (%)	0.0%
Duplicate rows	0
Duplicate rows (%)	0.0%
Total size in memory	1.5 MiB
Average record size in memory	168.0 B

Variable types

Numeric	7
Unsupported	1
Categorical	10
Boolean	2

Alerts

`City` has a high cardinality: 140 distinct values	High cardinality
`Address` has a high cardinality: 8909 distinct values	High cardinality
`Locality` has a high cardinality: 1206 distinct values	High cardinality
`Locality_Verbose` has a high cardinality: 1263 distinct values	High cardinality
`Cuisines` has a high cardinality: 1825 distinct values	High cardinality
`Average_Cost_for_two` is highly correlated with `Price_range`	High correlation
`Price_range` is highly correlated with `Average_Cost_for_two` and 1 other fields (Average_Cost_for_two, Votes)	High correlation

Out[23]:

In [9]:
```
# Loading the second data set
cc = pd.read_excel('Country-Code.xlsx')
```

In [10]:
```
# Merging the data
df = pd.merge(df,cc,on ='Country Code', how = 'left')
```

In [11]:
```
df.head()
```

Out[11]:

	Restaurant ID	Restaurant Name	Country Code	City	Address	Locality	Locality Verbose	Longitude	Latitude	Cuisines	Average Cost for two	Currency	bc
0	7402935	Skye	94	Jakarta	Menara BCA, Lantai 56, Jl. MH. Thamrin, Thamri...	Grand Indonesia Mall, Thamrin	Grand Indonesia Mall, Thamrin, Jakarta	106.821999	-6.196778	Italian, Continental	800000	Indonesian Rupiah(IDR)	
1	7410290	Satoo - Hotel Shangri-La	94	Jakarta	Hotel Shangri-La, Jl. Jend. Sudirman	Hotel Shangri-La, Sudirman	Hotel Shangri-La, Sudirman, Jakarta	106.818961	-6.203292	Asian, Indonesian, Western	800000	Indonesian Rupiah(IDR)	
2	7420899	Sushi Masa	94	Jakarta	Jl. Tuna Raya No. 5, Penjaringan	Penjaringan	Penjaringan, Jakarta	106.800144	-6.101298	Sushi, Japanese	500000	Indonesian Rupiah(IDR)	
3	7421967	3 Wise Monkeys	94	Jakarta	Jl. Suryo No. 26, Senopati, Jakarta	Senopati	Senopati, Jakarta	106.813400	-6.235241	Japanese	450000	Indonesian Rupiah(IDR)	

	Restaurant ID	Restaurant Name	Country Code	City	Address	Locality	Locality Verbose	Longitude	Latitude	Cuisines	Average Cost for two	Currency	bc
4	7422489	Avec Moi Restaurant and Bar	94	Jakarta	Gedung PIC, Jl. Teluk Betung 43, Thamrin, Jakarta	Thamrin	Thamrin, Jakarta	106.821023	-6.196270	French, Western	350000	Indonesian Rupiah(IDR)	

In [12]:
```
# Replacing the spaces with '-' from the columns name
df.columns = df.columns.str.replace(' ', '_')
df.columns
```

Out[12]:
```
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', 'Price_range', 'Aggregate_rating',
       'Rating_color', 'Rating_text', 'Votes', 'Country'],
      dtype='object')
```

In [13]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 9541 entries, 0 to 9540
Data columns (total 20 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   Restaurant_ID         9541 non-null   int64  
 1   Restaurant_Name       9541 non-null   object 
 2   Country_Code          9541 non-null   int64  
 3   City                  9541 non-null   object 
 4   Address               9541 non-null   object 
 5   Locality              9541 non-null   object 
 6   Locality_Verbose      9541 non-null   object 
 7   Longitude             9541 non-null   float64
 8   Latitude              9541 non-null   float64
 9   Cuisines              9541 non-null   object 
 10  Average_Cost_for_two  9541 non-null   int64  
 11  Currency              9541 non-null   object 
 12  Has_Table_booking     9541 non-null   object 
 13  Has_Online_delivery   9541 non-null   object 
 14  Price_range           9541 non-null   int64  
 15  Aggregate_rating      9541 non-null   float64
 16  Rating_color          9541 non-null   object 
 17  Rating_text           9541 non-null   object 
 18  Votes                 9541 non-null   int64  
 19  Country               9541 non-null   object 
dtypes: float64(3), int64(5), object(12)
memory usage: 1.5+ MB
```

In [14]:
```
# Grouping the data by Country
country_restaurant_number = df.groupby(['Country']).agg( Count = ('Restaurant_ID','count')).reset_index()
country_restaurant_number = country_restaurant_number.sort_values(by = 'Count', ascending = False).reset_index(drop=Tru
```

3. Explore the geographical distribution of the restaurants and identify the cities with maximum and minimum number of restaurants

In [15]:
```
# Grouping the data by City
city_restaurant_number = df.groupby(['City']).agg( Count = ('Restaurant_ID','count')).reset_index()
city_restaurant_number = city_restaurant_number.sort_values(by = 'Count', ascending = False).reset_index(drop=True)
```

In [16]:
```
city_restaurant_number

# Max restaurants 5473 in New Delhi and min restaurants in a city is 1.
```

Out[16]:

	City	Count
0	New Delhi	5473
1	Gurgaon	1118
2	Noida	1080
3	Faridabad	251

	City	Count
4	Ghaziabad	25
...	...	...
135	Penola	1
136	Phillip Island	1
137	Potrero	1
138	Princeton	1
139	Panchkula	1

140 rows × 2 columns

In [21]:
```
a = city_restaurant_number[city_restaurant_number['Count']>10]
a
```

Out[21]:

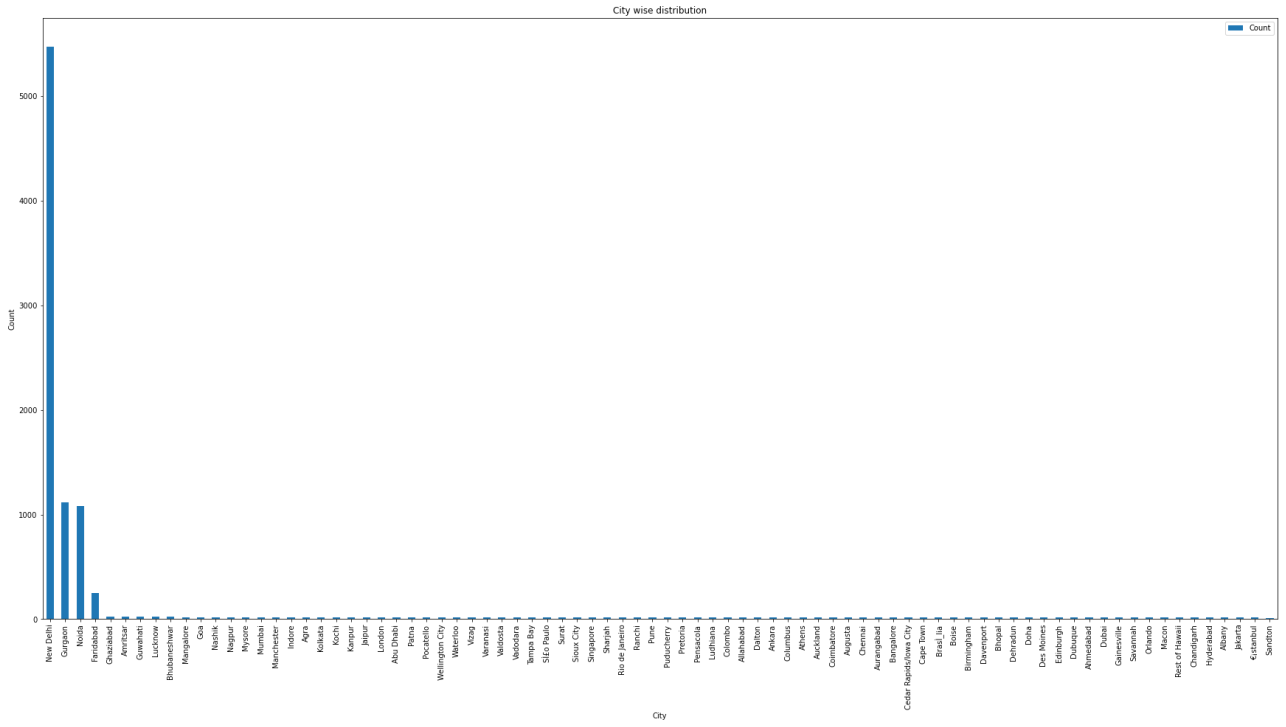
	City	Count
0	New Delhi	5473
1	Gurgaon	1118
2	Noida	1080
3	Faridabad	251
4	Ghaziabad	25
...	...	...
77	Hyderabad	18
78	Albany	17
79	Jakarta	16
80	€¡stanbul	14
81	Sandton	11

82 rows × 2 columns

City wise restaurant distribution

In [22]:
```
# Taking cities which have restaurant count more than 10
a.plot(kind='bar', title='City wise distribution', ylabel='Count', xlabel='City', figsize=(30, 15),
                            x='City', y='Count')
```

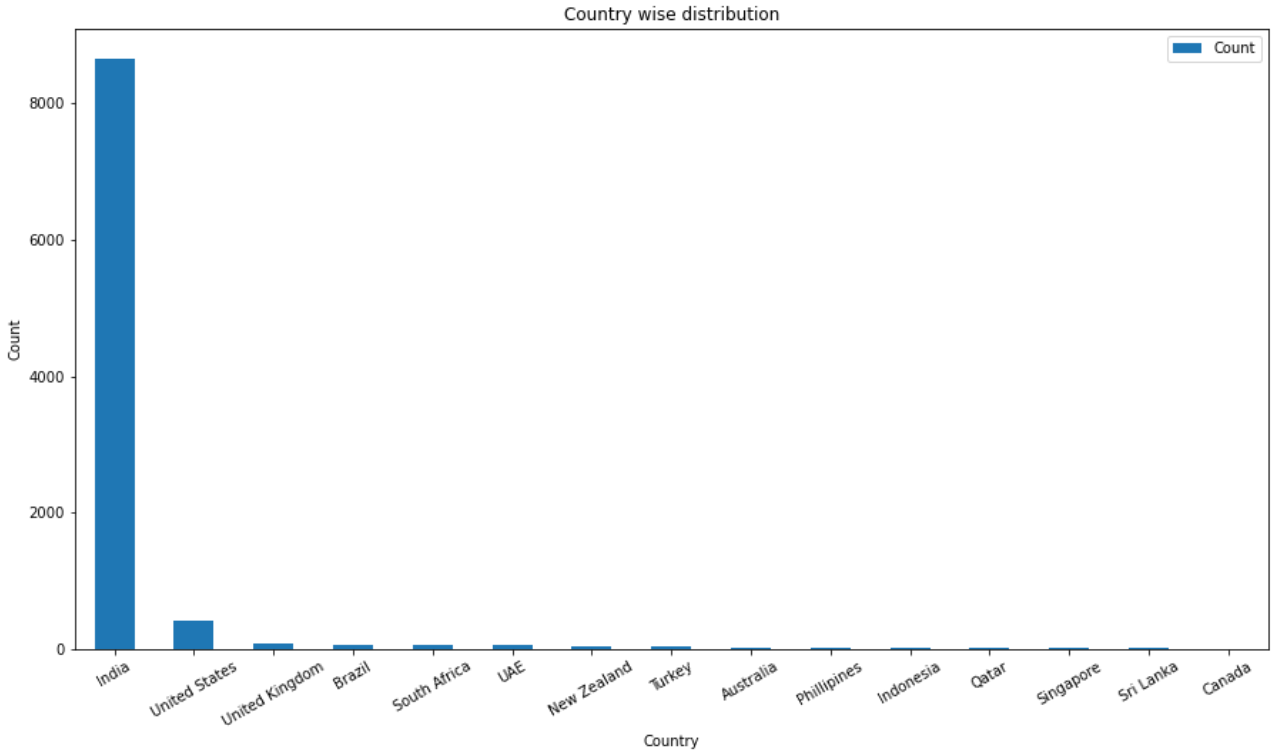
Out[22]:
```
<AxesSubplot:title={'center':'City wise distribution'}, xlabel='City', ylabel='Count'>
```



Country wise restaurant distribution

In [24]:
```
country_restaurant_number.plot(rot=30, kind='bar', title='Country wise distribution', ylabel='Count', xlabel='Country',
                               figsize=(15, 8), x='Country', y='Count')
```

Out[24]:
```
<AxesSubplot:title={'center':'Country wise distribution'}, xlabel='Country', ylabel='Count'>
```



4. Restaurant franchising is a thriving venture. So, it is very important to explore the franchise with most national presence

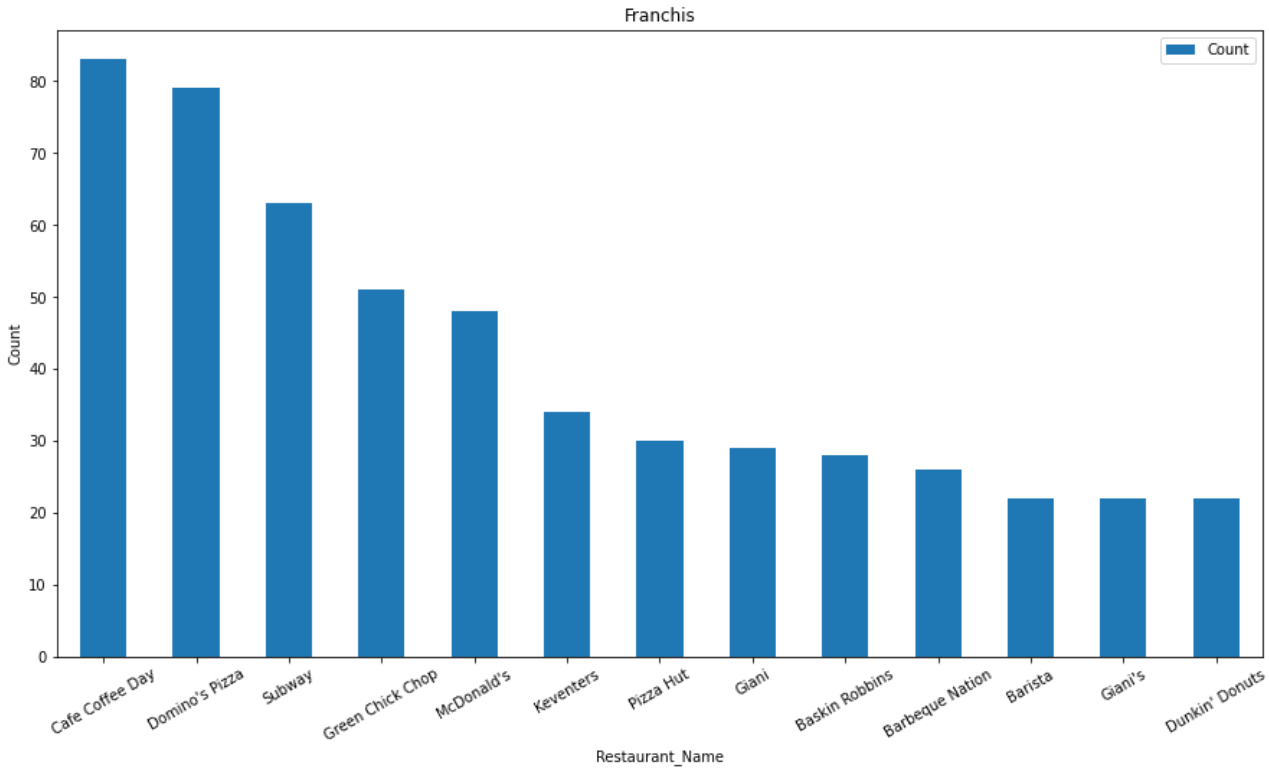
In [25]:
```
# Grouping the data by Restaurant_Name
franchise = df.groupby(['Restaurant_Name']).agg( Count = ('City','count')).reset_index()
```

In [26]:
```
x = franchise.sort_values(by = 'Count',ascending=False).reset_index(drop=True)
```

In [27]:
```
tstx=x[x['Count']>20]
```

In [28]:
```
tstx.plot(rot=30, kind='bar', title='Franchis', ylabel='Count', xlabel='Restaurant_Name',
                               figsize=(15, 8), x='Restaurant_Name', y='Count')
```

Out[28]:
```
<AxesSubplot:title={'center':'Franchis'}, xlabel='Restaurant_Name', ylabel='Count'>
```



In [29]:
```
max_rate = df.sort_values(by = 'Aggregate_rating', ascending = False).groupby(['Country','City'],as_index=False).first(
# Highest rating restaurants

min_rate = df.sort_values(by = 'Aggregate_rating', ascending = False).groupby(['Country','City'],as_index=False).last()
# Lowest rating restaurants

df_max = max_rate[['Country','City','Restaurant_Name','Aggregate_rating']]
# new dataframe created for high rated resturants

df_min = min_rate[['Country','City','Restaurant_Name','Aggregate_rating']]
# new dataframe created for low rated resturants

rating_rest = df_max.merge(df_min, left_on = 'City', right_on = 'City', how = 'inner')
# merge into single dataframe
```

In [30]:
```
# Removing the extra column from the merged data
rating_rest.drop(columns = 'Country_y', axis = 1, inplace = True)

# Renaming the the columns
rating_rest.rename(columns = {'Country_x':'Country',
                              'Restaurant_Name_x':'Highest Rated Restaurants',
                              'Aggregate_rating_x':'Rating Max',
                              'Restaurant_Name_y':'Lowest Rated Restaurants',
                              'Aggregate_rating_y':'Rating Min'}, inplace=True)
```

In [31]:
```
rating_rest.columns
```

Out[31]:
```
Index(['Country', 'City', 'Highest Rated Restaurants', 'Rating Max',
       'Lowest Rated Restaurants', 'Rating Min'],
      dtype='object')
```

In [32]:
```
rating_rest
```

Out[32]:

	Country	City	Highest Rated Restaurants	Rating Max	Lowest Rated Restaurants	Rating Min
0	Australia	Armidale	Whitebull Hotel	3.5	Whitebull Hotel	3.5
1	Australia	Balingup	Taste of Balingup	3.2	Taste of Balingup	3.2
2	Australia	Beechworth	Bridge Road Brewers	4.6	Bridge Road Brewers	4.6
3	Australia	Dicky Beach	The Giggling Goat	3.6	The Giggling Goat	3.6
4	Australia	East Ballina	The Belle General	4.1	The Belle General	4.1
...	...	...	...	...	...	...
135	United States	Valdosta	Smok'n Pig B-B-Q	4.1	El Toreo Mexican Restaurant	3.1
136	United States	Vernonia	Blue House Cafe	4.3	Blue House Cafe	4.3

	Country	City	Highest Rated Restaurants	Rating Max	Lowest Rated Restaurants	Rating Min
137	United States	Waterloo	Tokyo Japanese Steak House	3.9	Masala Grill & Coffee House	3.2
138	United States	Weirton	Theo Yianni's Authentic Greek Restaurant	3.9	Theo Yianni's Authentic Greek Restaurant	3.9
139	United States	Winchester Bay	Fishpatrick's Crabby Cafe	3.2	Fishpatrick's Crabby Cafe	3.2

140 rows × 6 columns

In [33]:
```
# Since India and USA has the most number of resturants
# we will try to see the distribution of resturants ratings for these two country

from plotly.offline import iplot
from plotly.offline import init_notebook_mode
from plotly.offline import get_plotlyjs_version
from plotly.offline import download_plotlyjs
print(get_plotlyjs_version())
import plotly.graph_objects
from plotly.graph_objs import *
import plotly.graph_objs as go
```

```
2.8.3
```

In [34]:
```
rating_rest_city_india = rating_rest[rating_rest['Country'] == 'India']  # Storing the dataframe only for 'India'
rating_rest_city_india # In India
city = rating_rest_city_india['City'].tolist() # converting the series to list
rate_max = rating_rest_city_india['Rating Max'].tolist() # converting the series to list
rate_min = rating_rest_city_india['Rating Min'].tolist() # converting the series to list
rest_name_high = rating_rest_city_india['Highest Rated Restaurants'].tolist() # converting the series to list
rest_name_low = rating_rest_city_india['Lowest Rated Restaurants'].tolist() # converting the series to list
```

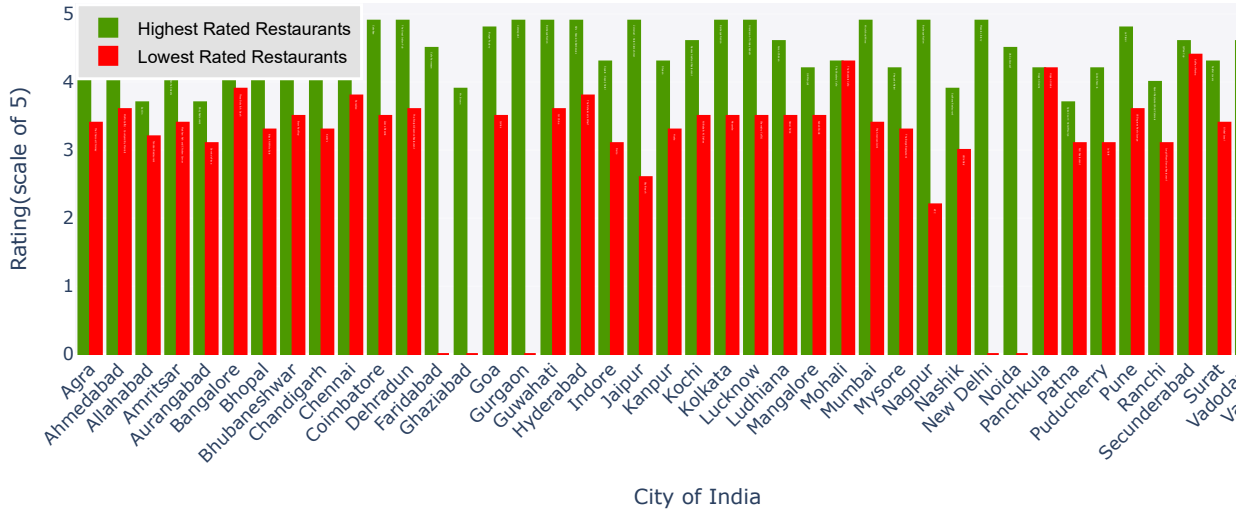
In [35]:
```
stack0 = go.Bar( #  GroupBarChart 1 (Highest Rated Resturant))
    x = city, # x axis label
    y = rate_max, # y axis label
    text = rest_name_high, # the value of the restaurant
    name = 'Highest Rated Restaurants',
    marker = dict(
         color = 'rgb(76,153,0)', # color for the bar graph's marker
         line = dict(
             color = 'rgb(76,153,0)', # color for the bar graph's line
             width = 1.5, # width of the bar graph
         )
     ),
    opacity = 1.0
)
stack1 = go.Bar (#  GroupBarChart 2 (Lowest Rated Resturant))
    x = city, # x axis label
    y = rate_min, # y axis label
    text = rest_name_low, # the value of the restaurant
    name = 'Lowest Rated Restaurants',
     marker = dict(
         color = 'rgb(255,0,0)', # color for the bar graph's marker
         line = dict(
             color = 'rgb(255,0,0)', # color for the bar graph's line
             width = 1.5, # width of the bar graph
         )
     ),
    opacity = 1.0
)

data = [stack0, stack1]
layout = go.Layout(
    legend = dict( # the Layout of the graph (beautification)
        x = 0,
        y = 1,
        traceorder = 'normal',
        font = dict(
            family = 'sans-serif',
            size = 12,
            color = '#000'
        ),
        bgcolor = '#E2E2E2',
        bordercolor = '#FFFFFF',
        borderwidth = 2
    ),
    autosize = False,
    width = 1000, # size of the graph
    height = 450,
    barmode = 'group',
    title = "Graph 1.1: Restaurants rating of India <br>\
    <i>hover with cursor to see restaurant's name </i>", # title of the graph
    plot_bgcolor = 'rgb(245,245,249,1)',
```

```
    xaxis = dict(tickangle = -45, title = 'City of India'), # Making the graph label inclined at 45 deg
    yaxis = {'title' : 'Rating(scale of 5)'} # label of y-axis
)
fig = go.Figure(data = data, layout = layout) # plotting the graph
iplot(fig, filename = 'style-barbar')
```

Graph 1.1: Restaurants rating of India

hover with cursor to see restaurant's name



In [36]:

```
# perform the same steps as above for country = 'United States'
rating_rest_city_usa = rating_rest[rating_rest['Country'] == 'United States']  # Storing the dataframe only for 'USA'
rating_rest_city_usa # In USA
cityu = rating_rest_city_usa['City'].tolist() # converting the series to list
rate_maxu = rating_rest_city_usa['Rating Max'].tolist() # converting the series to list
rate_minu = rating_rest_city_usa['Rating Min'].tolist() # converting the series to list
rest_name_highu = rating_rest_city_usa['Highest Rated Restaurants'].tolist() # converting the series to list
rest_name_lowu = rating_rest_city_usa['Lowest Rated Restaurants'].tolist() # converting the series to list
```

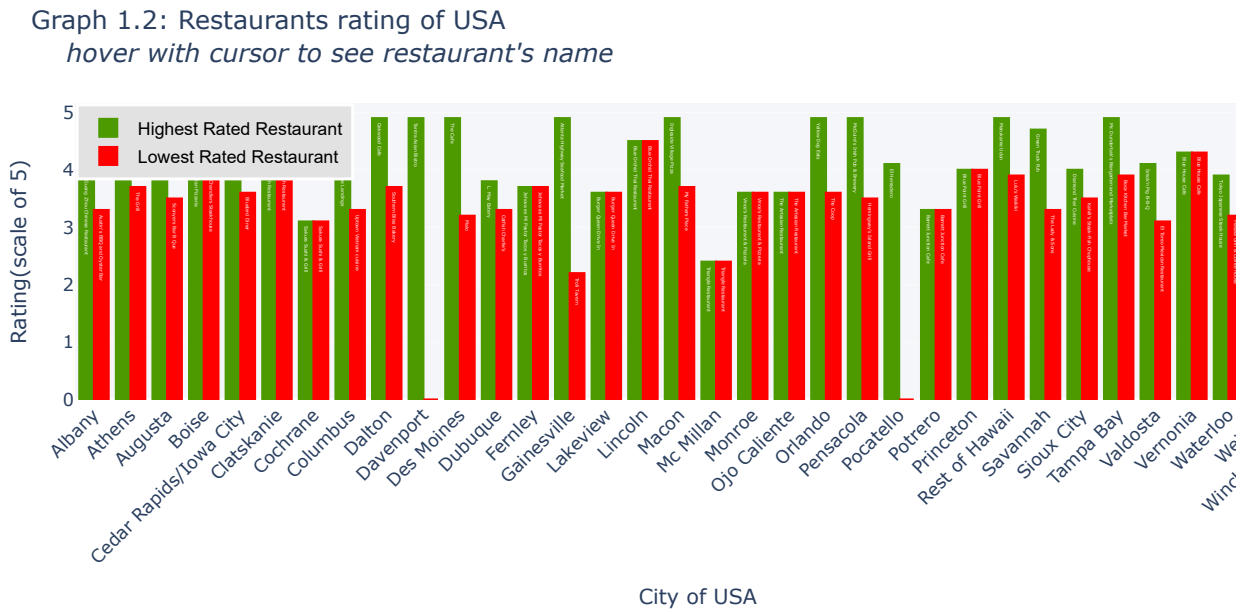
In [37]:

```
stack0 = go.Bar( # GroupBarChart 1 (Highest Rated Restaurant)
    x = cityu, # x axis label
    y = rate_maxu, # y axis label
    text = rest_name_highu, # teh vaue of the restaurant
    name = 'Highest Rated Restaurant',
    marker = dict(
        color = 'rgb(76,153,0)', # color of the bar graph's marker
        line = dict(
            color = 'rgb(76,153,0)', # color of bar graph's line
            width = 1.5, # width of the bar graph
        )
    ),
    opacity = 1.0
)
stack1 = go.Bar( # GroupBarChart 2 (Lowest Rated Restaurant)
    x = cityu, # x axis label
    y = rate_minu, # y axis label
    text = rest_name_lowu, # teh vaue of the restaurant
    name = 'Lowest Rated Restaurant',
    marker = dict(
        color = 'rgb(255,0,0)', # color of the bar graph's marker
        line = dict(
            color = 'rgb(255,0,0)', # color of bar graph's line
            width = 1.5, # width of the bar graph
        )
    ),
    opacity = 1.0
)

data = [stack0, stack1]
layout = go.Layout(
    legend = dict( # the Layout of the graph (beautification)
        x = 0,
        y = 1,
        traceorder = 'normal',
        font = dict(
            family = 'sans-serif',
            size = 12,
            color = '#000'
```

```
        ),
        bgcolor = '#E2E2E2',
        bordercolor = '#FFFFFF',
        borderwidth = 2
    ),
    autosize = False,
    width = 1000, # size of the graph
    height = 450,
    barmode = 'group',
    title = "Graph 1.2: Restaurants rating of USA <br>\
    <i>hover with cursor to see restaurant's name </i>", # title of the graph
    plot_bgcolor = 'rgba(245,246,249,1)',
    xaxis = dict(tickangle = -45, title = 'City of USA'), # Making the graph label inclined at 45 deg
    yaxis = {'title' : 'Rating(scale of 5)'} # label of y-axis

)
fig = go.Figure(data = data, layout = layout) # plotting the graph
iplot(fig, filename = 'style-barbar')
```



In [38]:

```
df1 = df.copy()
df1.columns
```

Out[38]:

```
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', 'Price_range', 'Aggregate_rating',
       'Rating_color', 'Rating_text', 'Votes', 'Country'],
      dtype='object')
```

In [39]:

```
# Converting the data
dummy = ['Has_Table_booking', 'Has_Online_delivery']
df1 = pd.get_dummies(df1, columns = dummy , drop_first = True)
df1.head()
# 0 indicates 'NO'
# 1 indicates 'YES'
```

Out[39]:

	Restaurant_ID	Restaurant_Name	Country_Code	City	Address	Locality	Locality_Verbose	Longitude	Latitude	Cuisines	Aver
0	7402935	Skye	94	Jakarta	Menara BCA, Lantai 56, Jl. MH. Thamrin, Thamri...	Grand Indonesia Mall, Thamrin	Grand Indonesia Mall, Thamrin, Jakarta	106.821999	-6.196778	Italian, Continental	
1	7410290	Satoo - Hotel Shangri-La	94	Jakarta	Hotel Shangri-La, Jl. Jend. Sudirman	Hotel Shangri-La, Sudirman	Hotel Shangri-La, Sudirman, Jakarta	106.818961	-6.203292	Asian, Indonesian, Western	
2	7420899	Sushi Masa	94	Jakarta	Jl. Tuna Raya No. 5, Penjaringan	Penjaringan	Penjaringan, Jakarta	106.800144	-6.101298	Sushi, Japanese	

	Restaurant_ID	Restaurant_Name	Country_Code	City	Address	Locality	Locality_Verbose	Longitude	Latitude	Cuisines	Aver
3	7421967	3 Wise Monkeys	94	Jakarta	Jl. Suryo No. 26, Senopati, Jakarta	Senopati	Senopati, Jakarta	106.813400	-6.235241	Japanese	
4	7422489	Avec Moi Restaurant and Bar	94	Jakarta	Gedung PIC, Jl. Teluk Betung 43, Thamrin, Jakarta	Thamrin	Thamrin, Jakarta	106.821023	-6.196270	French, Western	

5. Ratio between restaurants that allow table booking vs that do not allow table booking

6. Percentage of restaurants providing online delivery

In [40]:
```
# Ratio between restaurants allowing table booking and those which don't
table_booking = df1[df1['Has_Table_booking_Yes']==1]['Restaurant_ID'].count()
table_nbooking = df1[df1['Has_Table_booking_Yes']==0]['Restaurant_ID'].count()
print('Ratio Between restaurants that allow table booking vs those that do not allow table booking: ',
     round((table_booking/table_nbooking),2))
```

```
Ratio Between restaurants that allow table booking vs those that do not allow table booking:  0.14
```

In [41]:
```
print(table_booking,table_nbooking)
```

```
1158 8383
```

Table Booking vs No Table Booking

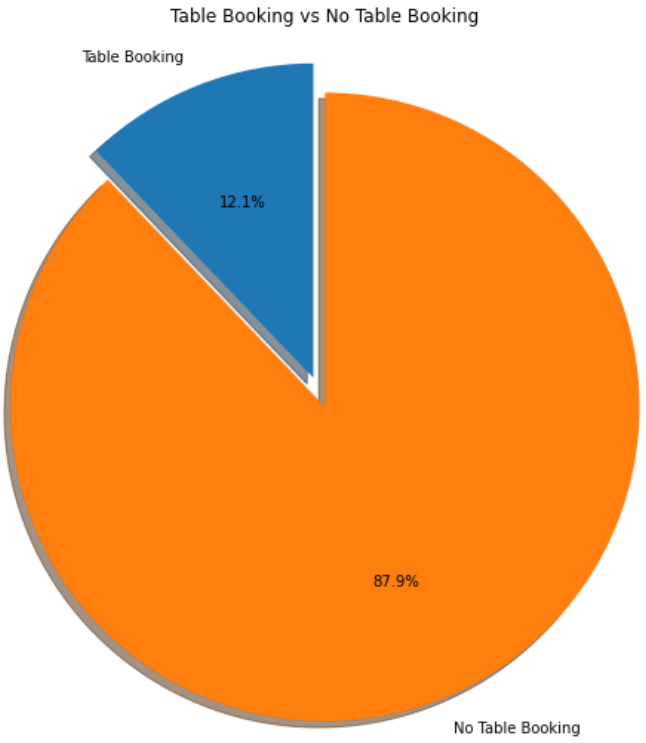
In [42]:
```
# Pie chart to show percentage of restaurants which allow table booking and those which don't
labels = 'Table Booking', 'No Table Booking'
sizes = [table_booking,table_nbooking]
explode = (0.1, 0) # only 'explode' the 2ns slice (i.e 'Hogs')

fig1, ax1 = plt.subplots(figsize = (9,9))
ax1.pie(sizes, explode = explode, labels = labels, autopct = '%1.1f%%', shadow = True, startangle = 90)
ax1.set_title('Table Booking vs No Table Booking')
ax1.axis('equal') # Equal aspect ratio ensures that pie is drawn as as circle

plt.show()
```

```
C:\Users\Dell\AppData\Local\Temp/ipykernel_5524/4211486665.py:11: UserWarning:

Matplotlib is currently using agg, which is a non-GUI backend, so cannot show the figure.
```



In [43]:
```
# Percentage of restaurants that has online delivery
rest_od = df1[df1['Has_Online_delivery_Yes'] == 1]['Restaurant_ID'].count()
rest_nod = df1[df1['Has_Online_delivery_Yes'] == 0]['Restaurant_ID'].count()
print('Percentage of restaurants providing online delivery : {} % '.format((round(rest_od/len(df1),3)*100)))
```

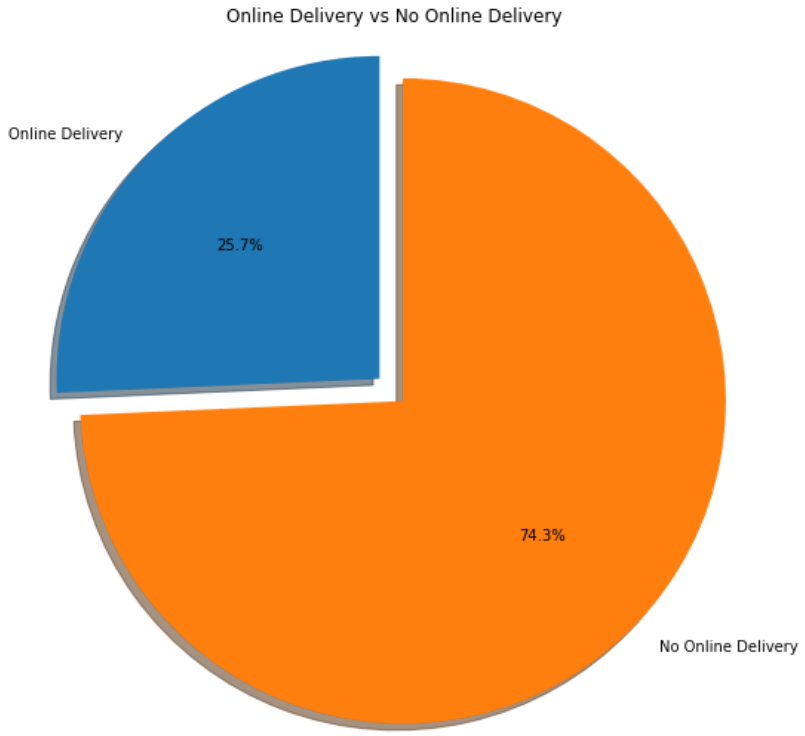
```
Percentage of restaurants providing online delivery : 25.7 %
```

Online Delivery vs No Online Delivery

In [44]:
```
# pie chart to show percentage of restaurants allowing online delivery vs those which do not have online delivery
labels = 'Online Delivery', 'No Online Delivery'
size = [rest_od,rest_nod]
explode = (0.1,0)
fig1,ax1 = plt.subplots(figsize =(9,9))
ax1.pie(size, explode = explode, labels = labels, autopct = '%1.1f%%', shadow = True, startangle = 90)
ax1.set_title('Online Delivery vs No Online Delivery')
ax1.axis('equal')
plt.show()
# out of the total votes about 27.3% votes were given to restaurants that don't have online delivery option
# out of the total votes about 72.7% votes were given to restaurants that have online delivery option
# This clearly shows that restaurants that have  online delivery are more likely to get a vote (feedback)
```

```
C:\Users\Dell\AppData\Local\Temp/ipykernel_5524/542708710.py:9: UserWarning:

Matplotlib is currently using agg, which is a non-GUI backend, so cannot show the figure.
```



7. Difference in no. of votes for the restaurants that deliver and the restaurants that don't

In [45]:
```
# Grouping the data by Has_Online_delivery_Yes
votes_difference = df1.groupby(['Has_Online_delivery_Yes']).sum('Votes')
```

In [46]:
```
votes_difference
```

Out[46]:

Has_Online_delivery_Yes	Restaurant_ID	Country_Code	Longitude	Latitude	Average_Cost_for_two	Price_range	Aggregate_rating	V
0	67227164919	165050	425144.965008	177803.058272	9789377	12502	17464.7	97
1	19063894933	8415	188094.559079	68820.590044	1663335	4718	7962.9	51

In [47]:
```
votes_difference = votes_difference.drop(['Restaurant_ID','Country_Code','Longitude','Latitude','Average_Cost_for_two',
                                          'Price_range','Aggregate_rating'], axis = 1)
votes_difference
```

Out[47]:

Has_Online_delivery_Yes	Votes	Has_Table_booking_Yes
0	977236	723.0
1	517914	435.0

In [48]:
```
difference_in_Votes = votes_difference.iloc[0]['Votes'] - votes_difference.iloc[1]['Votes']
print("Difference in no. of votes for the restaurants that deliver and the restaurants that don't = " ,difference_in_Vo
```

```
Difference in no. of votes for the restaurants that deliver and the restaurants that don't =  459322.0
```

8. What are the top 10 cuisines served across cities?

In [49]:
```
city_cuisines = df.groupby(['Cuisines','City']).agg( Count = ('City','count')).reset_index()
city_cuisines = city_cuisines.sort_values(by = 'Count', ascending = False).reset_index(drop=True)
```

In [50]:
```
city_cuisines.head(20)
```

Out[50]:

	Cuisines	City	Count

	Cuisines	City	Count
0	North Indian	New Delhi	658
1	North Indian, Chinese	New Delhi	284
2	Fast Food	New Delhi	242
3	Chinese	New Delhi	228
4	North Indian, Mughlai	New Delhi	207
5	Cafe	New Delhi	158
6	Street Food	New Delhi	123
7	Bakery	New Delhi	122
8	North Indian, Mughlai, Chinese	New Delhi	120
9	Bakery, Desserts	New Delhi	117
10	North Indian	Noida	110
11	North Indian	Gurgaon	105
12	Chinese, Fast Food	New Delhi	99
13	North Indian, Chinese	Noida	97
14	Pizza, Fast Food	New Delhi	92
15	Mithai, Street Food	New Delhi	90
16	Mughlai	New Delhi	86
17	South Indian	New Delhi	81
18	Bakery, Fast Food	New Delhi	80
19	Chinese, North Indian	New Delhi	70

9. What is the maximum and minimum no. of cuisines that a restaurant serves?

In [51]:
```
resta_cuisines = df.groupby(['Restaurant_Name','Cuisines']).agg( Count = ('Cuisines','count')).reset_index()
resta_cuisines = resta_cuisines.sort_values(by = 'Count', ascending = False).reset_index(drop=True)
resta_cuisines.rename(columns={'Count':'Restaurant_Count'}, inplace=True)
```

In [52]:
```
resta_cuisines['Cuisines'] = resta_cuisines['Cuisines'].str.split(',')
resta_cuisines['Cuisines_Count'] = resta_cuisines['Cuisines'].apply(lambda x : len(x))
resta_cuisines = resta_cuisines.sort_values(by = 'Cuisines_Count', ascending = False).reset_index(drop=True)
resta_cuisines
#resta_cuisines['Cuisines_Count'].unique()
# Max no of cuisines = 8 & Min no of cuisines = 1
```

Out[52]:

	Restaurant_Name	Cuisines	Restaurant_Count	Cuisines_Count
0	Bikanervala	[North Indian, South Indian, Fast Food, Str...	8	8
1	R' ADDA	[Street Food, Burger, Desserts, Italian, P...	1	8
2	Healthy Food Station	[Salad, Healthy Food, Burger, Italian, Con...	1	8
3	Marble	[Continental, South African, Beverages, Des...	1	8
4	The Belgian Triple	[Healthy Food, Seafood, Beverages, Belgian,...	1	8
...	...	...	...	...
7934	Aap Ki Khatir	[North Indian]	1	1
7935	Aggarwal Sweets Corner	[Mithai]	1	1
7936	The Second Wife Kitchen	[North Indian]	1	1
7937	The Singing Tree	[Beverages]	1	1
7938	Cafe Coffee Day	[Cafe]	83	1

7939 rows × 4 columns

In [53]:
```
df.columns
cuisines = df['Cuisines'].apply(lambda x: pd.Series(x.split(',')))  # Splitting the cuisines in separate columns
```

In [54]:
```
cuisines.columns = ['Cuisines_1','Cuisines_2','Cuisines_3','Cuisines_4','Cuisines_5','Cuisines_6','Cuisines_7','Cuisine
```

```
cuisines.head()
```

Out[54]:

	Cuisines_1	Cuisines_2	Cuisines_3	Cuisines_4	Cuisines_5	Cuisines_6	Cuisines_7	Cuisines_8
0	Italian	Continental	NaN	NaN	NaN	NaN	NaN	NaN
1	Asian	Indonesian	Western	NaN	NaN	NaN	NaN	NaN
2	Sushi	Japanese	NaN	NaN	NaN	NaN	NaN	NaN
3	Japanese	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	French	Western	NaN	NaN	NaN	NaN	NaN	NaN

In [55]:

```
# Addding the above data in the original dataset
df_cuisines = pd.concat([df,cuisines],axis =1)
df_cuisines.head()
```

Out[55]:

	Restaurant_ID	Restaurant_Name	Country_Code	City	Address	Locality	Locality_Verbose	Longitude	Latitude	Cuisines	...	\
0	7402935	Skye	94	Jakarta	Menara BCA, Lantai 56, Jl. MH. Thamrin, Thamri...	Grand Indonesia Mall, Thamrin	Grand Indonesia Mall, Thamrin, Jakarta	106.821999	-6.196778	Italian, Continental	...	
1	7410290	Satoo - Hotel Shangri-La	94	Jakarta	Hotel Shangri-La, Jl. Jend. Sudirman	Hotel Shangri-La, Sudirman	Hotel Shangri-La, Sudirman, Jakarta	106.818961	-6.203292	Asian, Indonesian, Western	...	
2	7420899	Sushi Masa	94	Jakarta	Jl. Tuna Raya No. 5, Penjaringan	Penjaringan	Penjaringan, Jakarta	106.800144	-6.101298	Sushi, Japanese	...	
3	7421967	3 Wise Monkeys	94	Jakarta	Jl. Suryo No. 26, Senopati, Jakarta	Senopati	Senopati, Jakarta	106.813400	-6.235241	Japanese	...	
4	7422489	Avec Moi Restaurant and Bar	94	Jakarta	Gedung PIC, Jl. Teluk Betung 43, Thamrin, Jakarta	Thamrin	Thamrin, Jakarta	106.821023	-6.196270	French, Western	...	

5 rows × 28 columns

In [56]:

```
cuisine_loc = pd.DataFrame(df_cuisines[['Country', 'City', 'Locality_Verbose','Cuisines_1','Cuisines_2','Cuisines_3',
                                        'Cuisines_4','Cuisines_5','Cuisines_6','Cuisines_7','Cuisines_8']])
```

In [57]:

```
cuisine_loc
```

Out[57]:

	Country	City	Locality_Verbose	Cuisines_1	Cuisines_2	Cuisines_3	Cuisines_4	Cuisines_5	Cuisines_6	Cuisines_7	Cuisines_8
0	Indonesia	Jakarta	Grand Indonesia Mall, Thamrin, Jakarta	Italian	Continental	NaN	NaN	NaN	NaN	NaN	NaN
1	Indonesia	Jakarta	Hotel Shangri-La, Sudirman, Jakarta	Asian	Indonesian	Western	NaN	NaN	NaN	NaN	NaN
2	Indonesia	Jakarta	Penjaringan, Jakarta	Sushi	Japanese	NaN	NaN	NaN	NaN	NaN	NaN
3	Indonesia	Jakarta	Senopati, Jakarta	Japanese	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	Indonesia	Jakarta	Thamrin, Jakarta	French	Western	NaN	NaN	NaN	NaN	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...	...
9536	India	Dehradun	Jakhan, Dehradun	Chinese	North Indian	Fast Food	NaN	NaN	NaN	NaN	NaN
9537	India	Kanpur	Mall Road, Kanpur	Indian	Chinese	Continental	NaN	NaN	NaN	NaN	NaN
9538	India	Kanpur	Parade, Kanpur	Cafe	Continental	Desserts	Ice Cream	Italian	Beverages	NaN	NaN
9539	India	Varanasi	Dashaswmedh Road, Varanasi	Street Food	NaN	NaN	NaN	NaN	NaN	NaN	NaN
9540	India	Varanasi	Sigra, Varanasi	Chinese	North Indian	NaN	NaN	NaN	NaN	NaN	NaN

9541 rows × 11 columns

In [58]:
```
cuisine_loc_stack = pd.DataFrame(cuisine_loc.stack()) # stacking the columns
cuisine_loc.tail(10)
```

Out[58]:

	Country	City	Locality_Verbose	Cuisines_1	Cuisines_2	Cuisines_3	Cuisines_4	Cuisines_5	Cuisines_6	Cuisines_7	Cuisines_8
9531	United States	Pocatello	Pocatello, Pocatello	Mexican	NaN	NaN	NaN	NaN	NaN	NaN	NaN
9532	India	Agra	Radisson Blu, Tajganj, Agra	North Indian	Chinese	Continental	NaN	NaN	NaN	NaN	NaN
9533	India	Agra	Tajganj, Agra	Cafe	North Indian	Chinese	NaN	NaN	NaN	NaN	NaN
9534	India	Agra	Tajganj, Agra	Cafe	Italian	Mexican	North Indian	Continental	NaN	NaN	NaN
9535	India	Allahabad	Civil Lines, Allahabad	North Indian	Chinese	Italian	NaN	NaN	NaN	NaN	NaN
9536	India	Dehradun	Jakhan, Dehradun	Chinese	North Indian	Fast Food	NaN	NaN	NaN	NaN	NaN
9537	India	Kanpur	Mall Road, Kanpur	Indian	Chinese	Continental	NaN	NaN	NaN	NaN	NaN
9538	India	Kanpur	Parade, Kanpur	Cafe	Continental	Desserts	Ice Cream	Italian	Beverages	NaN	NaN
9539	India	Varanasi	Dashaswmedh Road, Varanasi	Street Food	NaN	NaN	NaN	NaN	NaN	NaN	NaN
9540	India	Varanasi	Sigra, Varanasi	Chinese	North Indian	NaN	NaN	NaN	NaN	NaN	NaN

In [59]:
```
# Reshaping the data
keys = [c for c in cuisine_loc if c.startswith('Cuisine')]
a = pd.melt(cuisine_loc, id_vars = 'Locality_Verbose', value_vars = keys, value_name = 'Cuisines')
# melting the stack into a row
max_rate_1 = pd.DataFrame(a.groupby(by = ['Locality_Verbose', 'variable', 'Cuisines']).size().reset_index())
# find the highest restaurant in the city
del max_rate_1['variable']
max_rate_1.rename(columns={0:'Count'}, inplace=True)
max_rate_1.head()
```

Out[59]:

	Locality_Verbose	Cuisines	Count
0	ILD Trade Centre Mall, Sohna Road, Gurgaon	Cafe	1
1	ILD Trade Centre Mall, Sohna Road, Gurgaon	North Indian	1
2	ILD Trade Centre Mall, Sohna Road, Gurgaon	Beverages	1
3	ILD Trade Centre Mall, Sohna Road, Gurgaon	Mughlai	1
4	12th Square Building, Banjara Hills, Hyderabad	Mughlai	1

In [60]:
```
# find the highest restaurant in the city
loc = max_rate_1.sort_values('Count', ascending = False).groupby(by = ['Locality_Verbose'], as_index = False).first()
loc
```

Out[60]:

	Locality_Verbose	Cuisines	Count
0	ILD Trade Centre Mall, Sohna Road, Gurgaon	Cafe	1
1	12th Square Building, Banjara Hills, Hyderabad	Mughlai	1
2	A Hotel, Gurdev Nagar, Ludhiana	Chinese	1
3	ARSS Mall, Paschim Vihar, New Delhi	North Indian	1
4	Aaya Nagar, New Delhi	Cuisine Varies	1
...	...	...	...
1258	ibis New Delhi, Aerocity, New Delhi	North Indian	1
1259	ÌÃguas Claras, BrasÌ_lia	Bar Food	1
1260	ÌÏmitkÌ_y, Ankara	Kebab	1
1261	Ìàayyolu, Ankara	Cafe	1
1262	Ìàukurambar, Ankara	Patisserie	1

1263 rows × 3 columns

In [61]:
```
rating_res = loc.merge(df,left_on = 'Locality_Verbose', right_on = 'Locality_Verbose', how = 'inner')
# inner join to merge the two dataframe
df2 = pd.DataFrame(rating_res[['Country','City','Locality_Verbose', 'Cuisines_x', 'Count']])
# making a dataframe of rating restuarant
country = rating_res.sort_values('Count', ascending = False).groupby(by = ['Country'], as_index = False).first()
# grouping the data by country code
con = pd.DataFrame(country[['Country','City','Locality','Cuisines_x','Count']])
con.columns = ['Country','City','Locality','Cuisines','Number of Restaurants in the country']
# renaming the columns
con1 = con.sort_values('Number of Restaurants in the country', ascending = False)
# sorting the restaurants on the basis of the number of restautants in the country
con1[:10]
final_con = con1.drop(con1.index[[7,10]])
```

In [62]:
```
rating_res.columns
pd.DataFrame(rating_res[['Country_Code','City','Locality_Verbose', 'Cuisines_x', 'Count']])
```

Out[62]:

	Country_Code	City	Locality_Verbose	Cuisines_x	Count
0	1	Gurgaon	ILD Trade Centre Mall, Sohna Road, Gurgaon	Cafe	1
1	1	Gurgaon	ILD Trade Centre Mall, Sohna Road, Gurgaon	Cafe	1
2	1	Hyderabad	12th Square Building, Banjara Hills, Hyderabad	Mughlai	1
3	1	Ludhiana	A Hotel, Gurdev Nagar, Ludhiana	Chinese	1
4	1	New Delhi	ARSS Mall, Paschim Vihar, New Delhi	North Indian	1
...	...	...	...	...	...
9536	30	Brasì_lia	ÌÂguas Claras, Brasì_lia	Bar Food	1
9537	30	Brasì_lia	ÌÂguas Claras, Brasì_lia	Bar Food	1
9538	208	Ankara	ÌÏmitkì_y, Ankara	Kebab	1
9539	208	Ankara	Ìàayyolu, Ankara	Cafe	1
9540	208	Ankara	Ìàukurambar, Ankara	Patisserie	1

9541 rows × 5 columns

In [63]:
```
loc_list=final_con['City'] #converting the series to dataframe
a_list=loc_list.tolist()

cui_list=final_con['Cuisines']# converting the series to dataframe
b_list=cui_list.tolist()

count_list=final_con['Number of Restaurants in the country']# converting the series to dataframe
c_list=count_list.tolist()
```

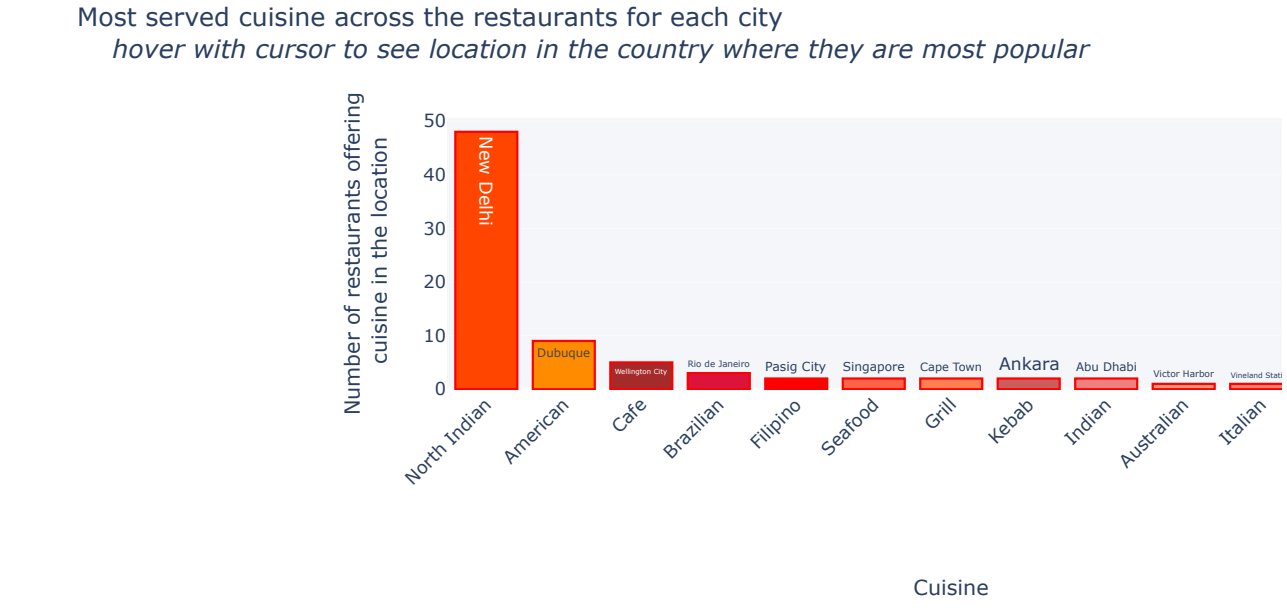
In [64]:
```
trace0 = go.Bar(# BarChart 1 (Popular cuisines of the country)
    x=b_list, #x axis label
    y=c_list, # y axis label
    text=loc_list, # location of the cuisine
    name='Popular Cuisine',
     marker=dict(
        color=['rgb(255,69,0)',
                'rgb(255,140,0)',
                'rgb(165,42,42)',
                'rgb(220,20,60)',
                'rgb(255,0,0)',
                'rgb(255,99,71)',
                'rgb(255,127,80)',
                'rgb(205,92,92)',
                'rgb(240,128,128)',
                'rgb(233,150,122)',
                'rgb(250,128,114)',
                'rgb(255,160,122)'],
        line=dict(
            color='rgb(255,0,0)',#color of the bar graph's line
            width=1.5, #width of the bar graph
        )
    ),
    opacity=1.0
)
data = [trace0]
layout = go.Layout(

    legend=dict( #the layout of the graph( beautification)
        x=0,
```

```
        y=1,
        traceorder='normal',
        font=dict(
            family='sans-serif',
            size=12,
            color='#000'
        ),
        bgcolor='#E2E2E2',
        bordercolor='#FFFFFF',
        borderwidth=20,
    ),
    autosize=False,
    width=1000, # size of the graph
    height=450,
    margin=Margin(r=20, l=300,
                  b=75, t=125),
    title="Most served cuisine across the restaurants for each city<br>\
    <i>hover with cursor to see location in the country where they are most popular </i>", #title of the graph
    plot_bgcolor='rgba(245, 246, 249, 1)',
    xaxis=dict(tickangle=-45,title= '<br>Cuisine<br>',mirror=True,showticklabels=True), #making the graphs label inclin
    yaxis= {'title': 'Number of restaurants offering<br> cuisine in the location'},#Label of y-axis
)
fig = go.Figure(data=data, layout=layout)#plotting the graph
iplot(fig)
```

```
C:\Users\Dell\anaconda3\lib\site-packages\plotly\graph_objs\_deprecations.py:405: DeprecationWarning:

plotly.graph_objs.Margin is deprecated.
Please replace it with one of the following more specific types
  - plotly.graph_objs.layout.Margin
```



In [65]:

```
rest_cuisine = pd.DataFrame(df_cuisines[['Restaurant_Name','City','Cuisines_1','Cuisines_2','Cuisines_3','Cuisines_4',
                                         'Cuisines_5','Cuisines_6','Cuisines_7','Cuisines_8']])
rest_cuisine_stack = pd.DataFrame(rest_cuisine.stack())  # stacking the columns
rest_cuisine.head()
```

Out[65]:

	Restaurant_Name	City	Cuisines_1	Cuisines_2	Cuisines_3	Cuisines_4	Cuisines_5	Cuisines_6	Cuisines_7	Cuisines_8
0	Skye	Jakarta	Italian	Continental	NaN	NaN	NaN	NaN	NaN	NaN
1	Satoo - Hotel Shangri-La	Jakarta	Asian	Indonesian	Western	NaN	NaN	NaN	NaN	NaN
2	Sushi Masa	Jakarta	Sushi	Japanese	NaN	NaN	NaN	NaN	NaN	NaN
3	3 Wise Monkeys	Jakarta	Japanese	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	Avec Moi Restaurant and Bar	Jakarta	French	Western	NaN	NaN	NaN	NaN	NaN	NaN

In [66]:

```
keys1 = [c for c in rest_cuisine if c.startswith('Cuisine')]
b = pd.melt(rest_cuisine, id_vars = 'Restaurant_Name', value_vars = keys, value_name = 'Cuisines')
# melting the stack into a row
max_rate_2 = pd.DataFrame(b.groupby(by = ['Restaurant_Name', 'variable', 'Cuisines']).size().reset_index())
# find the highest restuarant in the city
```

```
max_rate_2
del max_rate_2['variable']
max_rate_2.columns = ['Restaurant_Name', 'Cuisines', 'Count']
max_rate_2.head(20)
```

Out[66]:

	Restaurant_Name	Cuisines	Count
0	12212	Fast Food	1
1	Let's Burrrp	Chinese	1
2	Let's Burrrp	North Indian	1
3	#45	Cafe	1
4	#Dilliwaala6	North Indian	1
5	#InstaFreeze	Ice Cream	1
6	#OFF Campus	Cafe	1
7	#OFF Campus	Continental	1
8	#OFF Campus	Italian	1
9	#OFF Campus	Fast Food	1
10	#Urban Cafì©	North Indian	1
11	#Urban Cafì©	Chinese	1
12	#Urban Cafì©	Italian	1
13	#hashtag	Cafe	1
14	'Ohana	Hawaiian	1
15	10 Downing Street	North Indian	2
16	10 Downing Street	Chinese	2
17	10 To 10 In Delhi	Indian	1
18	10 To 10 In Delhi	Cafe	1
19	11th Avenue Cafe Bistro	Cafe	1

In [67]:

```
max_rate_2.sort_values('Count', ascending = False)
# Cafe Coffee Day has the max number of cuisines and the Least number of cuisines in a restaurant is 1.
```

Out[67]:

	Restaurant_Name	Cuisines	Count
2479	Cafe Coffee Day	Cafe	83
4594	Domino's Pizza	Pizza	79
4595	Domino's Pizza	Fast Food	78
12977	Subway	Healthy Food	63
12976	Subway	Salad	63
...	...	...	...
5564	Gabbar's Bar & Kitchen	North Indian	1
5565	Gabbar's Bar & Kitchen	Chinese	1
5566	Gabbar's Bar & Kitchen	Mexican	1
5567	Gabbar's Bar & Kitchen	Italian	1
15954	Ìàukura€Ùa Sofras€±	Izgara	1

15955 rows × 3 columns

In [68]:

```
rating = df1[['Restaurant_ID','Restaurant_Name','Country','City','Aggregate_rating','Average_Cost_for_two','Votes',
              'Price_range','Has_Table_booking_Yes','Has_Online_delivery_Yes']]
```

In [69]:

```
df1.columns
```

Out[69]:

```
Index(['Restaurant_ID', 'Restaurant_Name', 'Country_Code', 'City', 'Address',
       'Locality', 'Locality_Verbose', 'Longitude', 'Latitude', 'Cuisines',
       'Average_Cost_for_two', 'Currency', 'Price_range', 'Aggregate_rating',
       'Rating_color', 'Rating_text', 'Votes', 'Country',
       'Has_Table_booking_Yes', 'Has_Online_delivery_Yes'],
      dtype='object')
```

In [70]:

```
rating = rating.merge(max_rate_2,left_on = 'Restaurant_Name', right_on = 'Restaurant_Name', how = 'left')
rating
```

Out[70]:

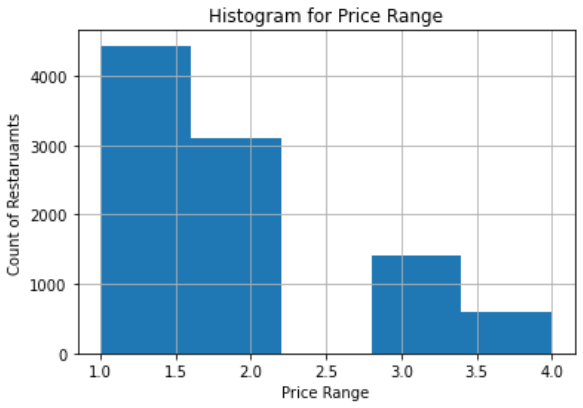
	Restaurant_ID	Restaurant_Name	Country	City	Aggregate_rating	Average_Cost_for_two	Votes	Price_range	Has_Table_booking_Y
0	7402935	Skye	Indonesia	Jakarta	4.1	800000	1498	3	
1	7402935	Skye	Indonesia	Jakarta	4.1	800000	1498	3	
2	7410290	Satoo - Hotel Shangri-La	Indonesia	Jakarta	4.6	800000	873	3	
3	7410290	Satoo - Hotel Shangri-La	Indonesia	Jakarta	4.6	800000	873	3	
4	7410290	Satoo - Hotel Shangri-La	Indonesia	Jakarta	4.6	800000	873	3	
...	...	...	...	...	...	...	...	...	...
23801	18312106	UrbanCrave	India	Kanpur	3.9	0	127	1	
23802	18312106	UrbanCrave	India	Kanpur	3.9	0	127	1	
23803	3900245	Deena Chat Bhandar	India	Varanasi	3.8	0	78	1	
23804	18246202	VNS Live Studio	India	Varanasi	3.5	0	109	1	
23805	18246202	VNS Live Studio	India	Varanasi	3.5	0	109	1	

23806 rows × 12 columns

10. What is the distribution cost across the restaurants?

In [71]:
```
hist = df['Price_range'].hist(bins=5)
hist.set_title('Histogram for Price Range')
hist.set_xlabel('Price Range')
hist.set_ylabel('Count of Restaruarnts')
```

Out[71]:
```
Text(0, 0.5, 'Count of Restaruarnts')
```



In [72]:
```
map_curr={'Indonesian Rupiah(IDR)':0.0052, 'Indian Rupees(Rs.)':1, 'Botswana Pula(P)':6.35,
      'Sri Lankan Rupee(LKR)': 0.37, 'Rand(R)' : 4.77, 'Qatari Rial(QR)' : 20.25, 'Dollar($)' : 74.27,
      'Emirati Diram(AED)' : 20.22, 'Brazilian Real(R$)' : 13.18, 'Turkish Lira(TL)' : 5.35,
      'Pounds(å£)' : 100.90, 'NewZealand($)' : 50.36}
df['curr_amt'] = df['Currency'].map(map_curr)
df['INR_Average_Cost_for_two'] = df['curr_amt']*df['Average_Cost_for_two']
```

In [73]:
```
df[['INR_Average_Cost_for_two','Average_Cost_for_two','Currency']]
```

Out[73]:

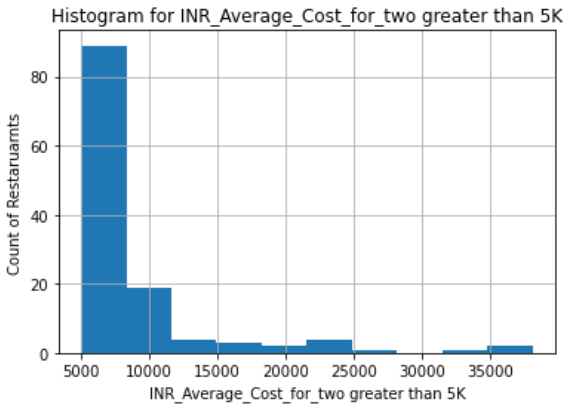
	INR_Average_Cost_for_two	Average_Cost_for_two	Currency
0	4160.0	800000	Indonesian Rupiah(IDR)
1	4160.0	800000	Indonesian Rupiah(IDR)
2	2600.0	500000	Indonesian Rupiah(IDR)
3	2340.0	450000	Indonesian Rupiah(IDR)
4	1820.0	350000	Indonesian Rupiah(IDR)
...	...	...	...

	INR_Average_Cost_for_two	Average_Cost_for_two	Currency
9536	0.0	0	Indian Rupees(Rs.)
9537	0.0	0	Indian Rupees(Rs.)
9538	0.0	0	Indian Rupees(Rs.)
9539	0.0	0	Indian Rupees(Rs.)
9540	0.0	0	Indian Rupees(Rs.)

9541 rows × 3 columns

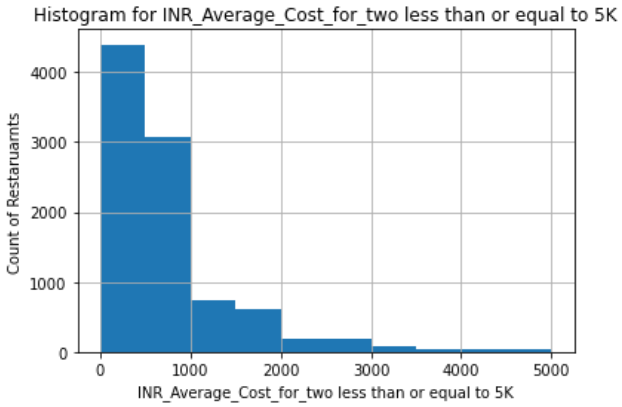
In [74]:
```
hist_2 = df[df['INR_Average_Cost_for_two']>5000]['INR_Average_Cost_for_two'].hist(bins=10)
hist_2.set_title('Histogram for INR_Average_Cost_for_two greater than 5K')
hist_2.set_xlabel('INR_Average_Cost_for_two greater than 5K')
hist_2.set_ylabel('Count of Restaruarnts')
```

Out[74]:
```
Text(0, 0.5, 'Count of Restaruarnts')
```



In [75]:
```
hist_3 = df[df['INR_Average_Cost_for_two']<=5000]['INR_Average_Cost_for_two'].hist(bins=10)
hist_3.set_title('Histogram for INR_Average_Cost_for_two less than or equal to 5K')
hist_3.set_xlabel('INR_Average_Cost_for_two less than or equal to 5K')
hist_3.set_ylabel('Count of Restaruarnts')
```

Out[75]:
```
Text(0, 0.5, 'Count of Restaruarnts')
```

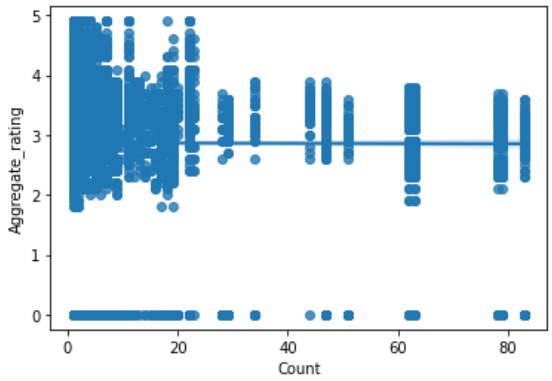


11. How ratings are distributed among the various factors?

In [76]:
```
sns.regplot(x='Count', y ='Aggregate_rating', data = rating)
rating[['Count','Aggregate_rating']].corr() # Correlation between Count and Aggregate_rating
# Number of cuisines is not a good factor to decide the rating of a restaurant
```

Out[76]:

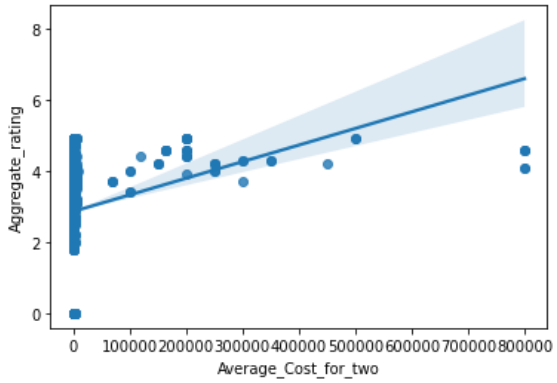
	Count	Aggregate_rating
Count	1.000000	-0.001569
Aggregate_rating	-0.001569	1.000000



In [77]:
```
sns.regplot(x='Average_Cost_for_two', y ='Aggregate_rating', data = rating)
rating[['Average_Cost_for_two','Aggregate_rating']].corr()  # Correlation between Average_Cost_for_two and Aggregate_ra
# Average Cost for two is a good factor to decide the rating of a restaurant
```

Out[77]:

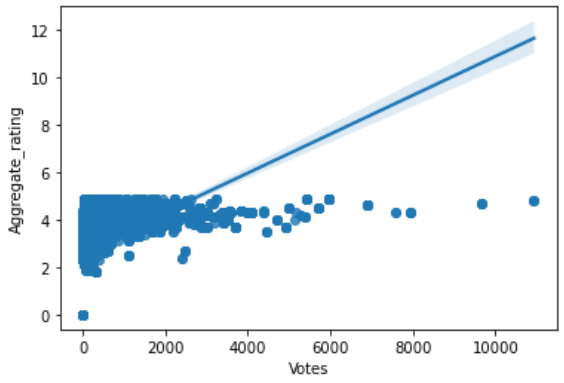
	Average_Cost_for_two	Aggregate_rating
Average_Cost_for_two	1.000000	0.050136
Aggregate_rating	0.050136	1.000000



In [78]:
```
sns.regplot(x='Votes', y ='Aggregate_rating', data = rating)
rating[['Votes','Aggregate_rating']].corr()  # Correlation between Votes and Aggregate_rating
# Number of votes can be a factor to decide the rating of a restaurant
```

Out[78]:

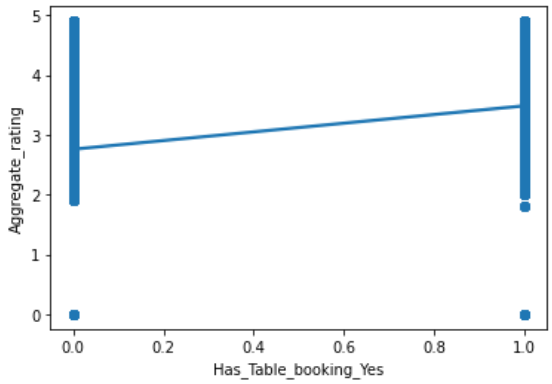
	Votes	Aggregate_rating
Votes	1.000000	0.318625
Aggregate_rating	0.318625	1.000000



In [79]:
```
sns.regplot(x='Has_Table_booking_Yes', y ='Aggregate_rating', data = rating)
rating[['Has_Table_booking_Yes','Aggregate_rating']].corr()  # Correlation between has_Tabke_Booking and Aggregate_rati
# Table Booking can be a factor to decide the rating of a restaurant
```

Out[79]:

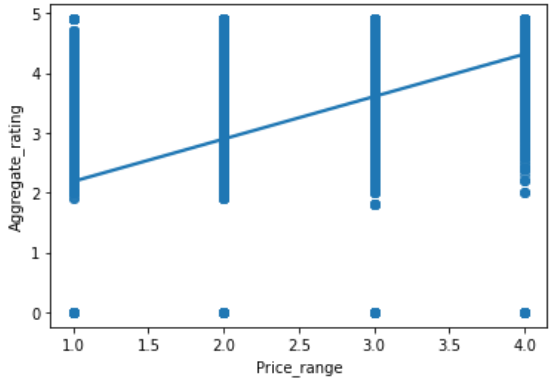
	Has_Table_booking_Yes	Aggregate_rating
Has_Table_booking_Yes	1.000000	0.181981
Aggregate_rating	0.181981	1.000000



In [80]:
```
sns.regplot(x='Price_range', y ='Aggregate_rating', data = rating)
rating[['Price_range','Aggregate_rating']].corr()  # Correlation between Price_range and Aggregate_rating
# Number of cuisines is a good factor to decide the rating of a restaurant
```

Out[80]:

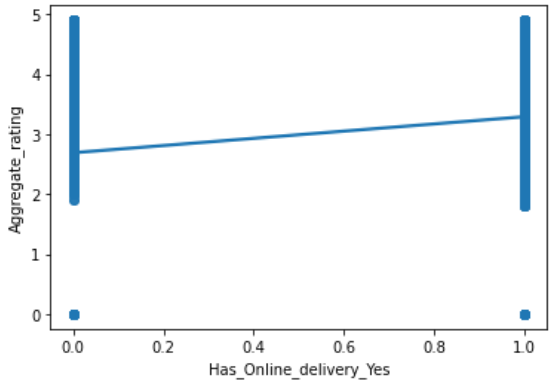
	Price_range	Aggregate_rating
Price_range	1.000000	0.463186
Aggregate_rating	0.463186	1.000000



In [81]:
```
sns.regplot(x='Has_Online_delivery_Yes', y ='Aggregate_rating', data = rating)
rating[['Has_Online_delivery_Yes','Aggregate_rating']].corr()  # Correlation between Has_Online_delivery and Aggregate_
# Table Booking can be a factor to decide the rating of a restaurant
```

Out[81]:

	Has_Online_delivery_Yes	Aggregate_rating
Has_Online_delivery_Yes	1.000000	0.193082
Aggregate_rating	0.193082	1.000000



We see that there is no single variable that affects the rating strongly, however Table Booking, Online Delivery, Avg Price for Two, Price Range and Number of votes do play a part in affecting the rating of a restaurant