In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from textblob import TextBlob   #to understand natural language processing
from wordcloud import WordCloud   #to create wordcloud

df = pd.read_csv("food_dely.csv")
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 211944 entries, 0 to 211943
Data columns (total 26 columns):
 #   Column                 Non-Null Count   Dtype  
---  ------                 --------------   -----  
 0   res_id                 211944 non-null  int64  
 1   name                   211944 non-null  object 
 2   establishment          207117 non-null  object 
 3   url                    211944 non-null  object 
 4   address                211810 non-null  object 
 5   city                   211944 non-null  object 
 6   city_id                211944 non-null  int64  
 7   locality               211944 non-null  object 
 8   latitude               211944 non-null  float64
 9   longitude              211944 non-null  float64
 10  zipcode                48757 non-null   object 
 11  country_id             211944 non-null  int64  
 12  locality_verbose       211944 non-null  object 
 13  cuisines               210553 non-null  object 
 14  timings                208070 non-null  object 
 15  average_cost_for_two   211944 non-null  int64  
 16  price_range            211944 non-null  int64  
 17  currency               211944 non-null  object 
 18  highlights             209875 non-null  object 
 19  aggregate_rating       211944 non-null  float64
 20  rating_text            211944 non-null  object 
 21  votes                  211944 non-null  int64  
 22  photo_count            211944 non-null  int64  
 23  opentable_support      211896 non-null  float64
 24  delivery               211944 non-null  int64  
 25  takeaway               211944 non-null  int64  
dtypes: float64(4), int64(9), object(13)
memory usage: 42.0+ MB
```

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

Out[2]:

	res_id	name	establishment	url	address	c
0	3400299	Bikanervala	Quick Bites	https://www.zomato.com/agra/bikanervala-khanda...	Kalyani Point, Near Tulsi Cinema, Bypass Road,...	A
1	3400005	Mama Chicken Mama Franky House	Quick Bites	https://www.zomato.com/agra/mama-chicken-mama-...	Main Market, Sadar Bazaar, Agra Cantt, Agra	A

2 rows × 26 columns

In [3]:
```
df.shape
# len(df) 0r df.shape[0]
```

Out[3]:
```
(211944, 26)
```

In [4]:
```
df.describe()   #Statistical analysis of numeric datatype columns.
```

Out[4]:

	res_id	city_id	latitude	longitude	country_id	average_cos
count	2.119440e+05	211944.000000	211944.000000	211944.000000	211944.0	2119
mean	1.349411e+07	4746.785434	21.499475	77.615276	1.0	5
std	7.883722e+06	5568.766386	22.781261	7.500104	0.0	6
min	5.000000e+01	1.000000	0.000000	0.000000	1.0	
25%	3.301027e+06	11.000000	15.496071	74.877961	1.0	2
50%	1.869573e+07	34.000000	22.514181	77.425971	1.0	4
75%	1.881297e+07	11306.000000	26.841214	80.219323	1.0	7
max	1.915979e+07	11354.000000	10000.000000	91.832769	1.0	300

In [5]:
```
df.duplicated().value_counts()
```

Out[5]:
```
True      151527
False      60417
Name: count, dtype: int64
```

In [6]:
```
df = df.drop_duplicates()   #remove rows that are completely identical across all c
```

```
df.duplicated().value_counts()
```

Out[6]:
```
False    60417
Name: count, dtype: int64
```

In [7]:
```
df.isna().sum()
```

Out[7]:
```
res_id                    0
name                      0
establishment          1920
url                       0
address                  18
city                      0
city_id                   0
locality                  0
latitude                  0
longitude                 0
zipcode               47869
country_id                0
locality_verbose          0
cuisines                470
timings                1070
average_cost_for_two      0
price_range               0
currency                  0
highlights              743
aggregate_rating          0
rating_text               0
votes                     0
photo_count               0
opentable_support        19
delivery                  0
takeaway                  0
dtype: int64
```

In [8]:
```
df["res_id"] = df["res_id"].drop_duplicates().astype(int)   #replace the duplicates
df["res_id"] = df["res_id"].replace(np.nan, "Not Registered")
```

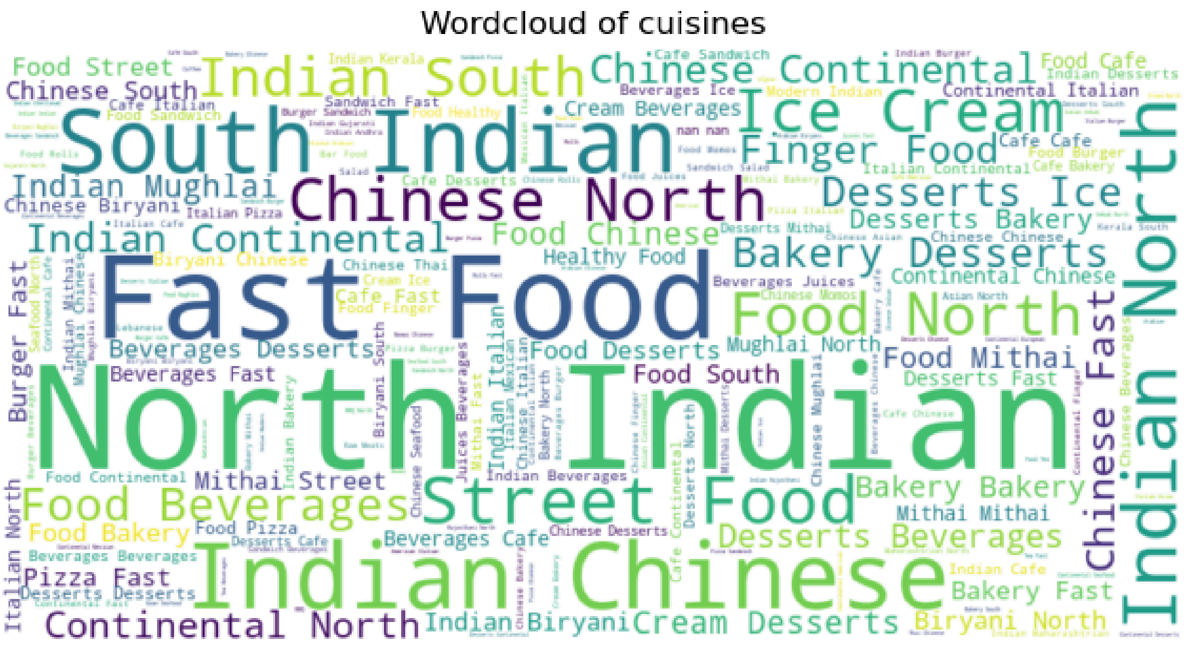
In [9]:
```
df["establishment"] = df["establishment"].replace({np.nan : df["establishment"].mod
df["address"] = df["address"].fillna("Unknown")
```

In [10]:
```
df["zipcode"] = df["zipcode"].where(df["zipcode"].str.len()==6, "Not Provided")
```

In [11]:
```
df["country_id"].value_counts()
```

Out[11]:
```
country_id
1    60417
Name: count, dtype: int64
```

In [12]:
```
df["cuisines"] = df["cuisines"].astype(str)   #converts None, np.nan(NaN) values to

all_cuisines = " ".join(df["cuisines"])

career = WordCloud(width=800, height=400, background_color="White").generate(all_cu
plt.figure(figsize=(8,6))
plt.imshow(career, interpolation="bilinear")
plt.title("Wordcloud of cuisines")
plt.axis("off")
plt.show()
```



In [13]:
```
df["cuisines"] = df["cuisines"].replace("nan", "North Indian, South Indian, Chinese
```

In [14]:
```
df["timings"] = df["timings"].fillna("10 AM to 10 PM")
```

In [15]:
```
df["average_cost_for_two"].max(), df["average_cost_for_two"].min()
```

Out[15]:
```
(30000, 0)
```

In [16]:
```
df.loc[df["average_cost_for_two"]<100, "average_cost_for_two"] = 100
df.loc[df["average_cost_for_two"]>5000, "average_cost_for_two"] = 5000
```

In [17]:
```
df["price_range"].value_counts()
```

Out[17]:
```
price_range
1    30681
2    18378
3     8300
4     3058
Name: count, dtype: int64
```

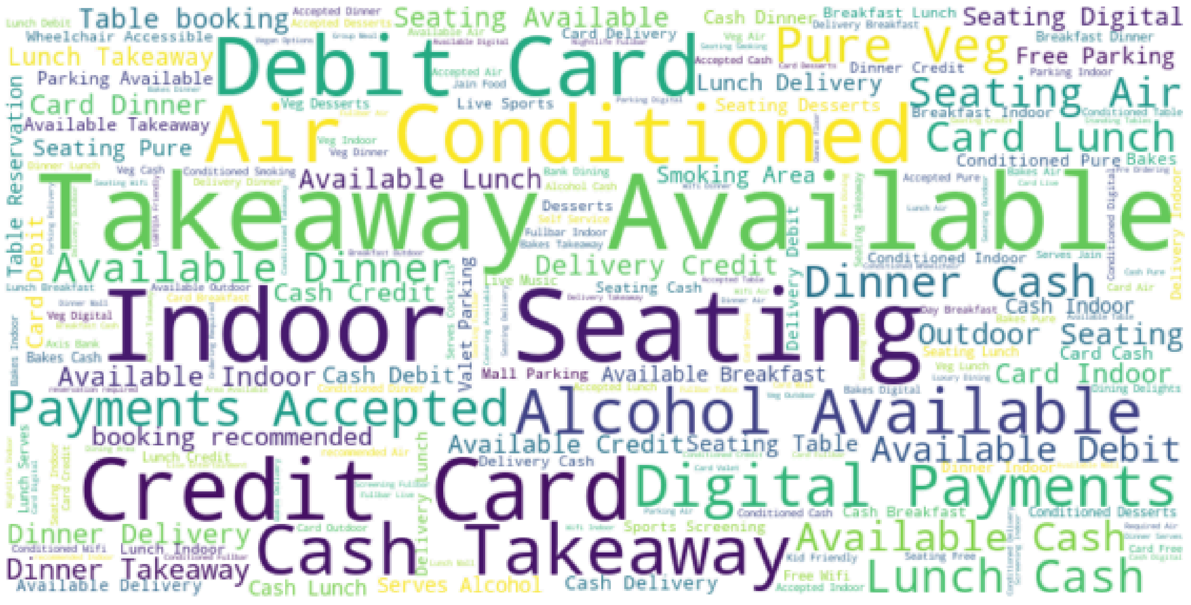
In [18]:
```
df["highlights"].isna().sum()
```

Out[18]:
```
np.int64(743)
```

In [19]:
```
df["highlights"] = df["highlights"].astype(str)

all_highlights = " ".join(df["highlights"])
```

```
career = WordCloud(width=800, height=400, background_color="White").generate(all_hi

plt.figure(figsize=(8,6))
plt.imshow(career, interpolation="bilinear")
plt.title("Wordcloud of all highlights")
plt.axis("off")
plt.show()
```

Wordcloud of all highlights



In [20]:
```
df["highlights"] = df["highlights"].replace("nan", " Indoor Seating, Takeaway Avail
```

In [21]:
```
df["aggregate_rating"].max(), df["aggregate_rating"].min()
```

Out[21]:
```
(4.9, 0.0)
```

In [22]:
```
df["rating_text"].value_counts()
```

Out[22]:
```
rating_text
Good                17569
Average             16782
Very Good           12714
Not rated           10160
Excellent            2065
Poor                  590
Çok iyi                56
Sangat Baik            44
Muito Bom              43
Excelente              34
Muy Bueno              33
Bardzo dobrze          29
Bom                    26
Skvělé                 24
Baik                   24
Velmi dobré            22
İyi                    19
Harika                 18
Ottimo                 17
Veľmi dobré            16
Buono                  14
Terbaik                14
Skvělá volba           13
Dobré                  12
Bueno                  11
Dobrze                  9
Wybitnie                8
Eccellente              8
Vynikajúce              7
Průměr                  6
Muito bom               5
Promedio                5
Média                   5
Ortalama                3
Średnio                 3
Media                   3
Priemer                 3
Biasa                   2
Scarso                  1
Name: count, dtype: int64
```

In [23]:
```
df["votes"].mean(), df["votes"].mode()
```

Out[23]:
```
(np.float64(261.4960524355728),
 0    0
 Name: votes, dtype: int64)
```

In [24]:
```
df["photo_count"].max(), df["photo_count"].min()
```

Out[24]:
```
(17702, 0)
```

In [25]:
```
df["opentable_support"] = df["opentable_support"].fillna(0)
df["opentable_support"].value_counts()
```

Out[25]:
```
opentable_support
0.0    60417
Name: count, dtype: int64
```

In [26]:
```
df["delivery"].value_counts()
```

Out[26]:
```
delivery
-1    41267
 1    18806
 0      344
Name: count, dtype: int64
```

In [27]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 60417 entries, 0 to 211942
Data columns (total 26 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   res_id                60417 non-null  object 
 1   name                  60417 non-null  object 
 2   establishment         60417 non-null  object 
 3   url                   60417 non-null  object 
 4   address               60417 non-null  object 
 5   city                  60417 non-null  object 
 6   city_id               60417 non-null  int64  
 7   locality              60417 non-null  object 
 8   latitude              60417 non-null  float64
 9   longitude             60417 non-null  float64
 10  zipcode               60417 non-null  object 
 11  country_id            60417 non-null  int64  
 12  locality_verbose      60417 non-null  object 
 13  cuisines              60417 non-null  object 
 14  timings               60417 non-null  object 
 15  average_cost_for_two  60417 non-null  int64  
 16  price_range           60417 non-null  int64  
 17  currency              60417 non-null  object 
 18  highlights            60417 non-null  object 
 19  aggregate_rating      60417 non-null  float64
 20  rating_text           60417 non-null  object 
 21  votes                 60417 non-null  int64  
 22  photo_count           60417 non-null  int64  
 23  opentable_support     60417 non-null  float64
 24  delivery              60417 non-null  int64  
 25  takeaway              60417 non-null  int64  
dtypes: float64(4), int64(8), object(14)
memory usage: 12.4+ MB
```

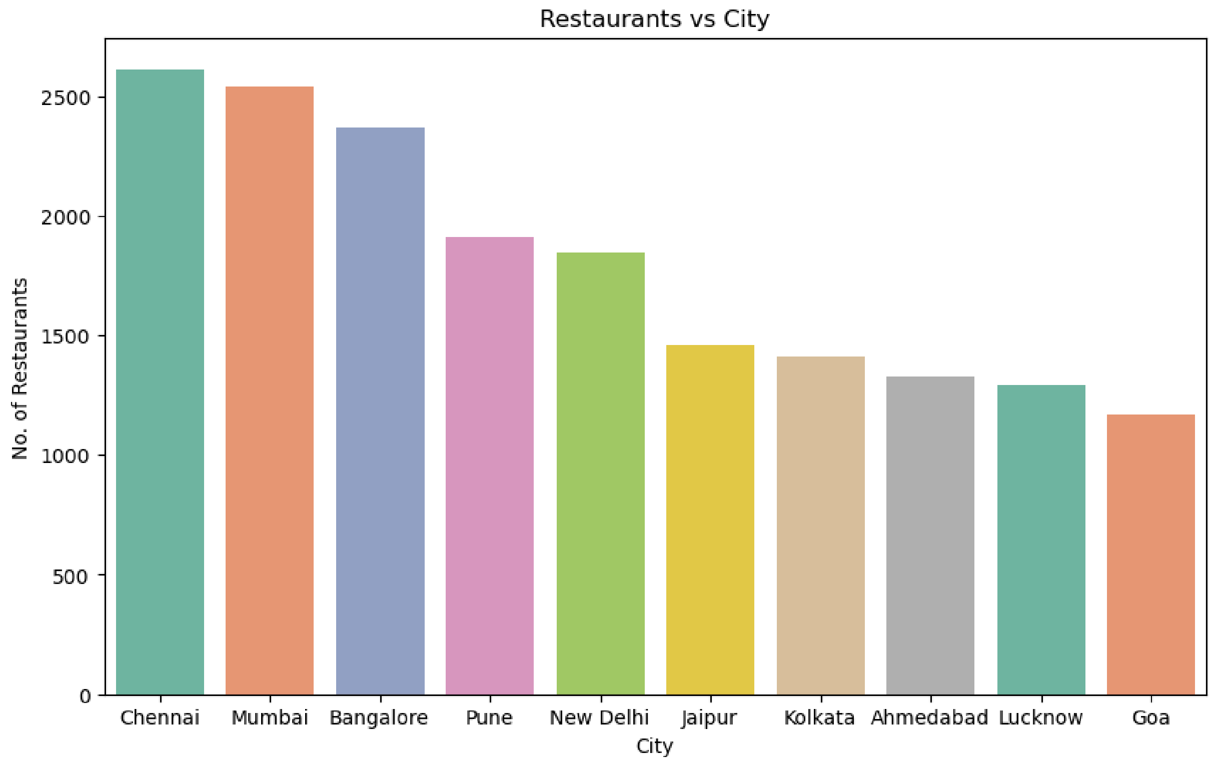
In [28]:
```
career = df["city"].value_counts().sort_values(ascending=False).head(10)
career.values
```

Out[28]:
```
array([2612, 2538, 2365, 1911, 1847, 1456, 1413, 1329, 1290, 1169])
```

In [29]:
```
#Data visualization of top 10 cities having max. number restaurants.

career = df["city"].value_counts().sort_values(ascending=False).head(10)
```

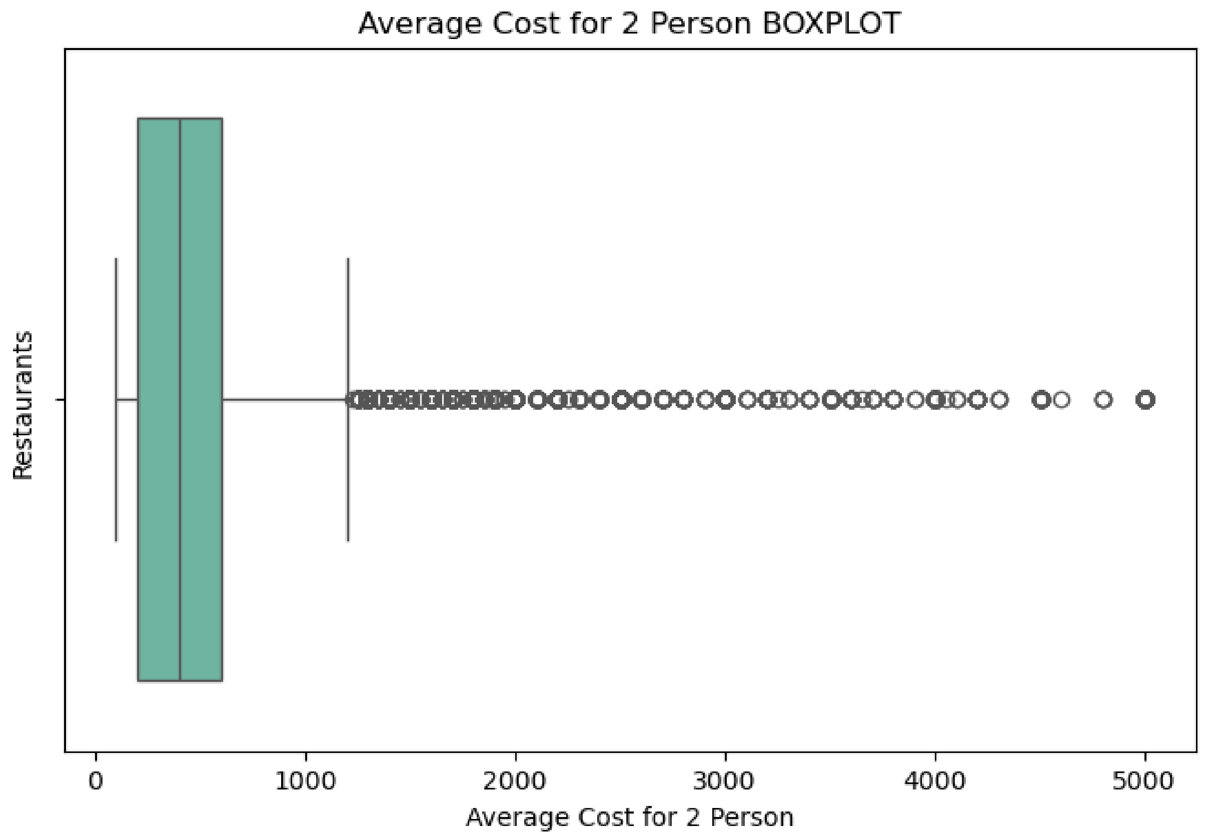
```
plt.figure(figsize=(10,6))
sns.barplot(x=career.index, y=career.values, data=pd.DataFrame(career), palette="Se
plt.title("Restaurants vs City")
plt.xlabel("City")
plt.ylabel("No. of Restaurants")
plt.show()
```

```
C:\Users\Sunder Singh Tulera\AppData\Local\Temp\ipykernel_13836\3189720581.py:6: Fut
ureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.1
4.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.barplot(x=career.index, y=career.values, data=pd.DataFrame(career), palette="S
et2")
```

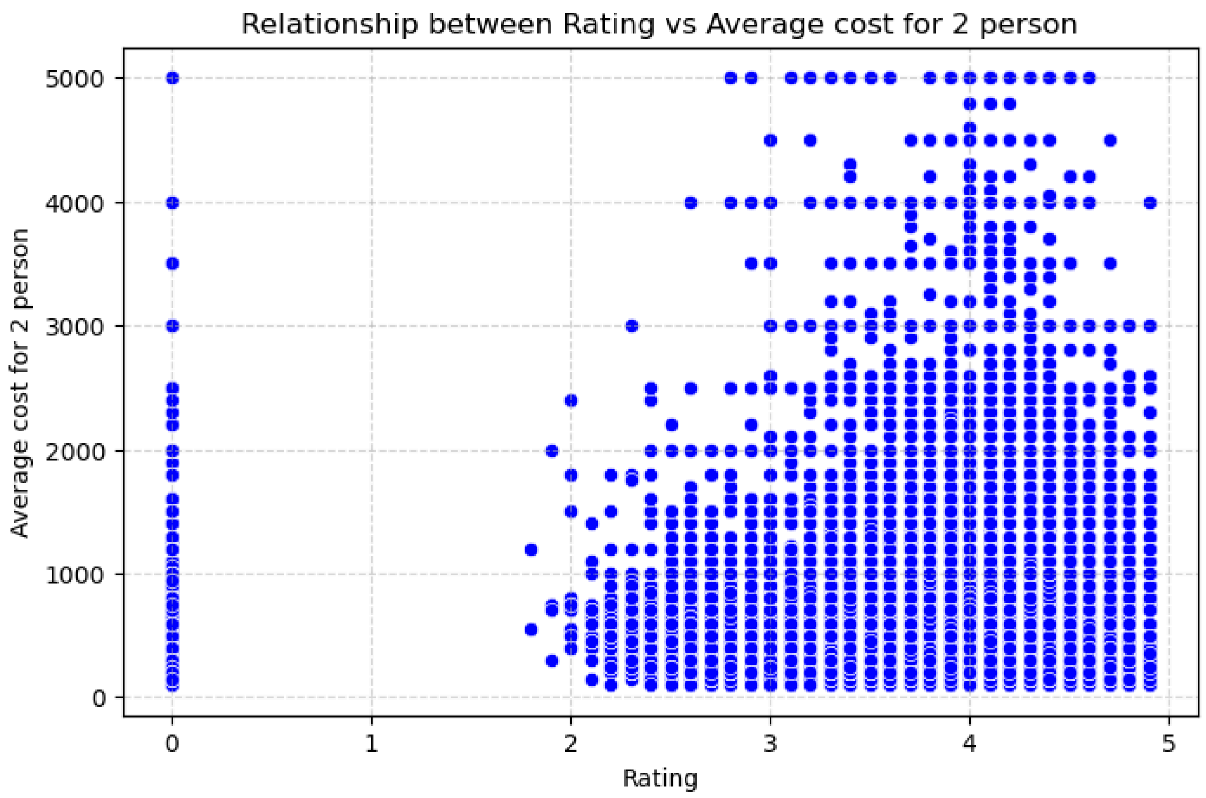


```
In [30]: #Data visualization based on Average Cost for 2 Person.

         plt.figure(figsize=(8,5))
         sns.boxplot(x="average_cost_for_two", data=df, palette="Set2")
         plt.title("Average Cost for 2 Person BOXPLOT")
         plt.xlabel("Average Cost for 2 Person")
         plt.ylabel("Restaurants")
         plt.show()
```

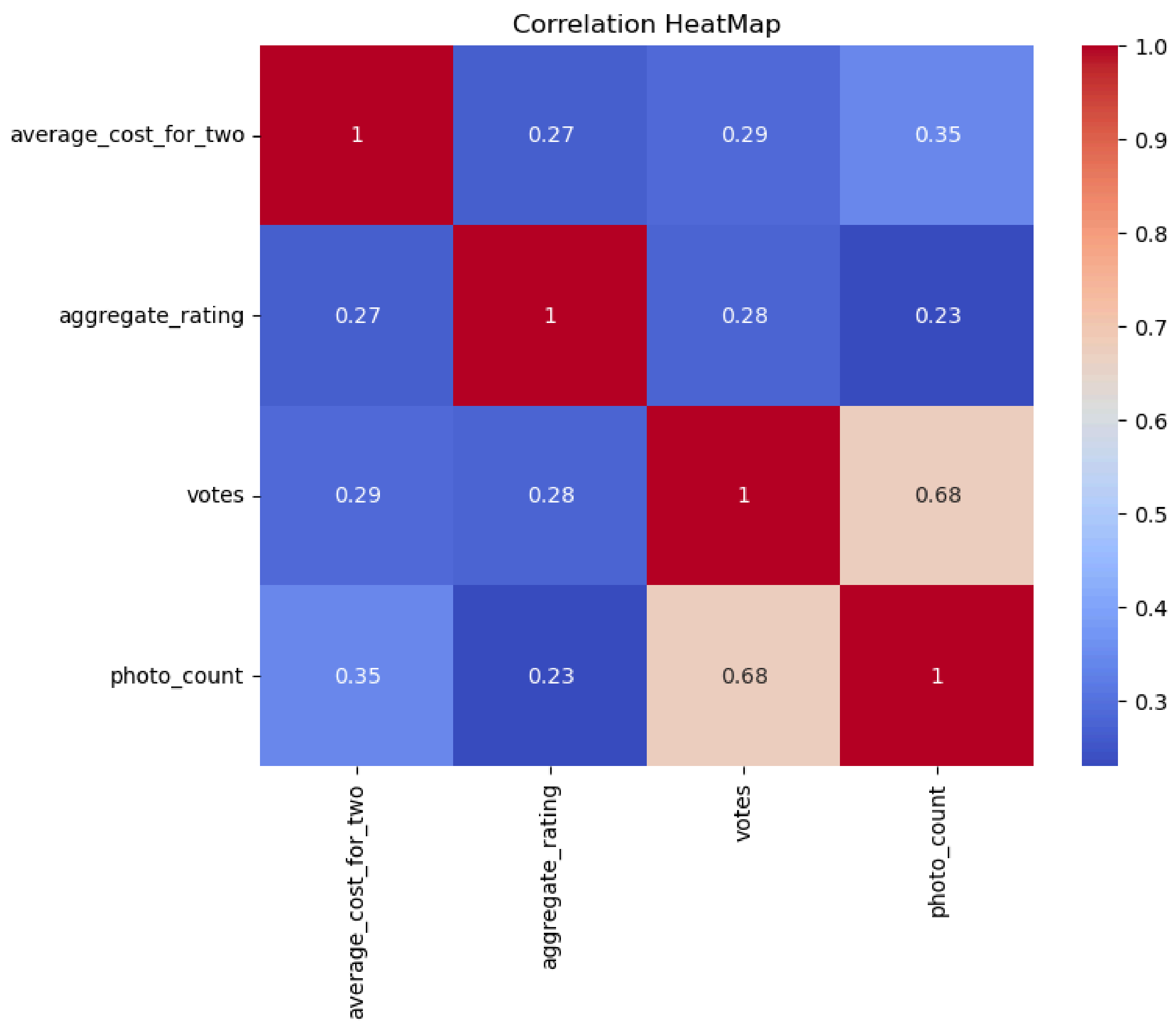
```
C:\Users\Sunder Singh Tulera\AppData\Local\Temp\ipykernel_13836\1124723171.py:4: Fut
ureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.1
4.0. Assign the `y` variable to `hue` and set `legend=False` for the same effect.

  sns.boxplot(x="average_cost_for_two", data=df, palette="Set2")
```



In [31]:
```
#Data visualization of relationship between rating vs average cost for 2 person.

plt.figure(figsize=(8,5))
sns.scatterplot(x="aggregate_rating", y="average_cost_for_two", data=df, color="blu
plt.grid(True, linestyle="--", alpha=1)
plt.title("Relationship between Rating vs Average cost for 2 person")
plt.grid(True, linestyle = "--", alpha = 0.5)
plt.xlabel("Rating")
plt.ylabel("Average cost for 2 person")
plt.show()
```



In [32]:
```
#Correlation between average_cost_for_two, aggregate_rating, votes & photo_count.

data = df[["average_cost_for_two", "aggregate_rating", "votes", "photo_count"]]

plt.figure(figsize=(8,6))
sns.heatmap(data.corr(), cmap="coolwarm", annot=True)
plt.title("Correlation HeatMap")
plt.show()
```



In [33]:
```
#Create a column region.

med_lat = df["latitude"].median()
med_lon = df["longitude"].median()

def check_region(lat, lon):
    if lat==np.nan or lon==np.nan:
        return "Unknown"
    if lat>=med_lat and lon>=med_lon:
        return "NE"
    if lat>=med_lat and lon<=med_lon:
        return "NW"
    if lat<=med_lat and lon<=med_lon:
        return "SW"
    else:
        return "SE"
df["region"] = df.apply(lambda row : check_region(row["latitude"], row["longitude"]

df["region"].value_counts()
```

```
Out[33]:  region
          NE    16703
          SW    16702
          NW    13506
          SE    13506
          Name: count, dtype: int64
```

```
In [ ]:  df.to_csv("Mini_Project.csv", index=False)
```

```
In [ ]:
```