Exploratory Data Analysis

Problem Statement

The zomato exploratory data analysis is for the foodies to find the best restaurants, value for money restaurants in their locality. It also helps to find their required cuisines in their locality.

Data Definition

res_id: The code given to a restaurant (Categorical)

name: Name of the restaurant (Categorical)

establishment: Represents the type of establishment (Categorical)

url: The website of the restaurant (Categorical)

address: The address of the restaurant (Categorical)

city: City in which the restaurant located (Categorical)

city_id: The code given to a city (Categorical)

locality: Locality of the restaurant (Categorical)

latitude: Latitude of the restaurant (Categorical)

longitude: Longitude of the restaurant (Categorical)

zipcode: Zipcode of the city in which the restaurant located (Categorical)

country_id: Country code in which the restaurant located (Categorical)

locality_verbose: Locality along with the city in which the restaurant located (Categorical)

cuisines: The cuisines a restaurant serves (Categorical)

timings: The working hours of a restaurant (Categorical)

average_cost_for_two: The average amount expected for 2 people (Numerical)

price_range: The categories for average cost (Categories - 1,2,3,4) (Categorical)

currency: The currency in which a customer pays (Categorical)

highlights: The facilities of the restaurant (Categorical)

aggregate_rating: The overall rating a restaurant has got (Numerical)

rating_text: Categorized ratings (Categorical)

votes: Number of votes received by the restaurant from customers (Numerical)

photo_count: The number of photos of a restaurant (Numerical)

opentable_support: Restaurant reservation from Opentable (Categorical)

delivery: The restaurant deliver an order or not (Categorical)

takeaway: The restaurant allows a 'takeaway' of an order or not (Categorical)

Table of Contents

1. Import Libraries

Import the required libraries and functions

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

2. Set Options

Make necessary changes to :

Set the working directory

In [2]:
```
os.chdir('C:\\Users\\Kejri\\Downloads\\files\\Capstone')
os.getcwd()
```

Out[2]:
```
'C:\\Users\\Kejri\\Downloads\\files\\Capstone'
```

3. Read Data

In [3]:
```
df_restaurants = pd.read_csv('ZomatoRestaurantsIndia.csv')
```

4. Understand and Prepare the Data

A well-prepared data proves beneficial for analysis as it limits errors and inaccuracies that can occur during analysis. The processed data is more accessible to users.

Data understanding is the process of getting familiar with the data, to identify data type, to discover first insights into the data, or to detect interesting subsets to form hypotheses about hidden information. Whereas, data preparation is the process of cleaning and transforming raw data before analysis. It is an important step before processing and often involves reformatting data, making corrections to data.

Data preparation is often a lengthy process, but it is essential as a prerequisite to put data in context to get insights and eliminate bias resulting from poor data quality.

Analyze and prepare data:
1. Check dimensions of the dataframe
2. View the head of the data
3. Note the redundant variables and drop them
4. Check the data types. Refer to data definition to ensure your data types are correct. If data types are not as per business context, change the data types as per requirement
5. Check for duplicates
Note: It is an art to explore data and one will need more and more practice to gain expertise in this area

---------- Provide the inference's from the output of every code executed.----------

1. Check dimensions of the dataframe in terms of rows and columns

```
In [4]: df_restaurants.shape
```

```
Out[4]: (211944, 26)
```

In [5]:
```
df_restaurants.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 211944 entries, 0 to 211943
Data columns (total 26 columns):
res_id                  211944 non-null int64
name                    211944 non-null object
establishment           207117 non-null object
url                     211944 non-null object
address                 211810 non-null object
city                    211944 non-null object
city_id                 211944 non-null int64
locality                211944 non-null object
latitude                211944 non-null float64
longitude               211944 non-null float64
zipcode                 48757 non-null object
country_id              211944 non-null int64
locality_verbose        211944 non-null object
cuisines                210553 non-null object
timings                 208070 non-null object
average_cost_for_two    211944 non-null int64
price_range             211944 non-null int64
currency                211944 non-null object
highlights              209875 non-null object
aggregate_rating        211944 non-null float64
rating_text             211944 non-null object
votes                   211944 non-null int64
photo_count             211944 non-null int64
opentable_support       211896 non-null float64
delivery                211944 non-null int64
takeaway                211944 non-null int64
dtypes: float64(4), int64(9), object(13)
memory usage: 42.0+ MB
```

In [6]:
```
print('The dataframe has 211944 rows and 26 columns. Also in some columns, suc
h as, \'establishment\', \'zipcode\', \'highlights\' etc., the number of rows
 are less.')
```

```
The dataframe has 211944 rows and 26 columns. Also in some columns, such as,
'establishment', 'zipcode', 'highlights' etc., the number of rows are less.
```

2. View the head of the data

In [7]:
```
df_restaurants.head(5)
```

Out[7]:

	res_id	name	establishment	url	address	city	c
0	3400299	Bikanervala	Quick Bites	https://www.zomato.com/agra/bikanervala-khanda...	Kalyani Point, Near Tulsi Cinema, Bypass Road,...	Agra	
1	3400005	Mama Chicken Mama Franky House	Quick Bites	https://www.zomato.com/agra/mama-chicken-mama-...	Main Market, Sadar Bazaar, Agra Cantt, Agra	Agra	
2	3401013	Bhagat Halwai	Quick Bites	https://www.zomato.com/agra/bhagat-halwai-2-sh...	62/1, Near Easy Day, West Shivaji Nagar, Goalp...	Agra	
3	3400290	Bhagat Halwai	Quick Bites	https://www.zomato.com/agra/bhagat-halwai-civi...	Near Anjana Cinema, Nehru Nagar, Civil Lines, ...	Agra	
4	3401744	The Salt Cafe Kitchen & Bar	Casual Dining	https://www.zomato.com/agra/the-salt-cafe-kitc...	1C,3rd Floor, Fatehabad Road, Tajganj, Agra	Agra	

5 rows × 26 columns

In [8]:
```
print('Each locality has a latitude and longitude, that lie in a certain city
 and a certain country; Has various restaurant names which have an address and
url to their website and many more such variables')
```

```
Each locality has a latitude and longitude, that lie in a certain city and a
certain country; Has various restaurant names which have an address and url t
o their website and many more such variables
```

3. Note the redundant variables and drop them

In [9]:
```
df_restaurants.columns.duplicated().sum()
```

Out[9]: 0

In [10]:
```
#i=0
#for var in df_restaurants_copy.columns:
#    print(i+1,". ",var, ": ", df_restaurants_copy[var].unique())
#    i+=1
```

In [11]:
```
df_restaurants['currency'].unique(), df_restaurants['currency'].shape[0]
```

Out[11]:
```
(array(['Rs.'], dtype=object), 211944)
```

In [12]:
```
df_restaurants['country_id'].unique(), df_restaurants['currency'].shape[0]
```

Out[12]:
```
(array([1], dtype=int64), 211944)
```

In [13]:
```
df_restaurants['opentable_support'].isna().sum(), df_restaurants['currency'].s
hape[0]
```

Out[13]:
```
(48, 211944)
```

In [14]:
```
df_restaurants_copy = df_restaurants.drop(['locality_verbose','currency','coun
try_id','takeaway','res_id'], axis=1).copy()
```

In [15]:
```
print('We have two separate columns \'locality\' and \'city\', we can later de
rive \'locality_verbose\' from these two, so we drop \'locality_verbose\', sin
ce we might later need to make analysis city or locality wise also. Also, sinc
e \'currency\', \'country_id\', \'takeaway\' has same value for all restaurant
s, i.e, Rs., 1, -1 respectively, we decide to drop these. Also, \'res_id\' wou
ld not be needed since we don\'t have any other dataframe to map to and we hav
e default index for this dataframe as well, so we drop this also.')
```

```
We have two separate columns 'locality' and 'city', we can later derive 'loca
lity_verbose' from these two, so we drop 'locality_verbose', since we might l
ater need to make analysis city or locality wise also. Also, since 'currenc
y', 'country_id', 'takeaway' has same value for all restaurants, i.e, Rs., 1,
-1 respectively, we decide to drop these. Also, 'res_id' would not be needed
since we don't have any other dataframe to map to and we have default index f
or this dataframe as well, so we drop this also.
```

4. Check the data types. Refer to data definition to ensure your data types are correct. If data types are not as per business context, change the data types as per requirement

In [16]:
```
df_restaurants_copy.dtypes
```

Out[16]:
```
name                    object
establishment           object
url                     object
address                 object
city                    object
city_id                  int64
locality                object
latitude               float64
longitude              float64
zipcode                 object
cuisines                object
timings                 object
average_cost_for_two     int64
price_range              int64
highlights              object
aggregate_rating       float64
rating_text             object
votes                    int64
photo_count              int64
opentable_support      float64
delivery                 int64
dtype: object
```

In [17]:
```
print('Here, we have searched for datatypes that have been described as numeri
cal in data definition, but wrongly made categorical in dataframe. No such var
iable found. Regarding variables that have been marked as categorical in data
 definition, but are not \'object\' data type in dataframe, all such variables
can be excluded in numerical calculations on dataframe, so no need to convert
 them to object.')
```

```
Here, we have searched for datatypes that have been described as numerical in
data definition, but wrongly made categorical in dataframe. No such variable
found. Regarding variables that have been marked as categorical in data defin
ition, but are not 'object' data type in dataframe, all such variables can be
excluded in numerical calculations on dataframe, so no need to convert them t
o object.
```

Change the incorrect data type

In [18]:
```
#df_restaurants_copy[['city_id','latitude','longitude','price_range','opentabl
e_support','delivery']] = df_restaurants_copy[['city_id','latitude','longitud
e','price_range','opentable_support','delivery']].astype('object')
```

In [19]:
```
print('All datatypes which are numerical in data definition, are also numerica
l in dataframe')
```

```
All datatypes which are numerical in data definition, are also numerical in d
ataframe
```

5. Check for Duplicates

In [20]:
```
df_restaurants_copy.duplicated().sum()
```

Out[20]:
```
151527
```

In [21]:
```
df_restaurants_copy[df_restaurants_copy.duplicated(keep="last")]
#df_restaurants_copy[df_restaurants_copy['name']=='Peshawri - ITC Mughal']
```

Out[21]:

	name	establishment	url	address
0	Bikanervala	Quick Bites	https://www.zomato.com/agra/bikanervala-khanda...	Kalyani Point, Near Tulsi Cinema, Bypass Road,...
1	Mama Chicken Mama Franky House	Quick Bites	https://www.zomato.com/agra/mama-chicken-mama-...	Main Market, Sadar Bazaar, Agra Cantt, Agra
2	Bhagat Halwai	Quick Bites	https://www.zomato.com/agra/bhagat-halwai-2-sh...	62/1, Near Easy Day, West Shivaji Nagar, Goalp...
3	Bhagat Halwai	Quick Bites	https://www.zomato.com/agra/bhagat-halwai-civi...	Near Anjana Cinema, Nehru Nagar, Civil Lines, ...
4	The Salt Cafe Kitchen & Bar	Casual Dining	https://www.zomato.com/agra/the-salt-cafe-kitc...	1C,3rd Floor, Fatehabad Road, Tajganj, Agra
...	...	...	...	...
211902	Jassi De Parathe	Quick Bites	https://www.zomato.com/vadodara/jassi-de-parat...	Near Kalaniketan, R.C Dutt Road, Alkapuri, Vad...
211903	22nd Parallel	Casual Dining	https://www.zomato.com/vadodara/22nd-parallel-...	1st Floor, Tapan Complex, Besides M Cube Mall,...
211908	Jassi De Parathe	Quick Bites	https://www.zomato.com/vadodara/jassi-de-parat...	Near Kalaniketan, R.C Dutt Road, Alkapuri, Vad...
211920	Chhappanbhog Restaurant & Banquets	Casual Dining	https://www.zomato.com/vadodara/chhappanbhog-r...	88, Sampatrao Colony , Productivity Road, Alka...

	name	establishment	url	address
211925	The Grand Thakar	Casual Dining	https://www.zomato.com/vadodara/the-grand-thak...	3rd Floor, Shreem Shalini Mall, Opposite Conqu...

151527 rows × 21 columns

In [22]:
```
df_restaurants_copy.drop_duplicates(keep='first',inplace=True)
df_restaurants_copy.shape
```

Out[22]: (60417, 21)

In [23]:
```
df_restaurants_copy[df_restaurants_copy['name']=='Peshawri - ITC Mughal']
```

Out[23]:

	name	establishment	url	address	city	city_id	locali
57	Peshawri - ITC Mughal	Fine Dining	https://www.zomato.com/agra/peshawri-itc-mugha...	ITC Mughal, Fatehabad Road, Tajganj, Agra	Agra	34	IT Mugha Tajga

1 rows × 21 columns

In [24]:
```
print('Dropped all duplicate rows, retaining the first one')
```

```
Dropped all duplicate rows, retaining the first one
```

5. Understand the variables

1. Variable 'name'

In [25]:
```
df_restaurants_copy['name'].isna().sum()
```

Out[25]: 0

In [26]:
```
df_restaurants_copy['name'].duplicated().sum()
```

Out[26]: 19452

In [27]:
```
df_restaurants_copy['name'].unique()
```

Out[27]:
```
array(['Bikanervala', 'Mama Chicken Mama Franky House', 'Bhagat Halwai',
       ..., 'Red China', 'Wah Ustad', 'Geeta lodge'], dtype=object)
```

In [ ]:
```

```

2. Variable 'establishment'

In [28]:
```
df_restaurants_copy['establishment'].isna().sum()
```

Out[28]:
```
1920
```

In [29]:
```
df_restaurants_copy['establishment'].duplicated().sum()
```

Out[29]:
```
60390
```

In [30]:
```
df_restaurants_copy['establishment'].unique()
```

Out[30]:
```
array(['Quick Bites', 'Casual Dining', 'Bakery', 'Café', 'Dhaba',
       'Bhojanalya', 'Bar', 'Sweet Shop', 'Fine Dining', 'Food Truck',
       'Dessert Parlour', 'Lounge', 'Pub', 'Beverage Shop', 'Kiosk',
       'Paan Shop', 'Confectionery', nan, 'Shack', 'Club', 'Food Court',
       'Mess', 'Butcher Shop', 'Microbrewery', 'Cocktail Bar', 'Pop up',
       'Irani Cafe'], dtype=object)
```

3. Variable 'city'

In [31]:
```
df_restaurants_copy['city'].isna().sum()
```

Out[31]:
```
0
```

In [32]:
```
df_restaurants_copy['city'].duplicated().sum()
```

Out[32]:
```
60319
```

In [33]:
```
i = 0
for var in df_restaurants_copy['city'].unique():
    print(var)
```

```
Agra
Ahmedabad
Gandhinagar
Ajmer
Alappuzha
Allahabad
Amravati
Amritsar
Aurangabad
Bangalore
Bhopal
Bhubaneshwar
Chandigarh
Mohali
Panchkula
Zirakpur
Nayagaon
Chennai
Coimbatore
Cuttack
Darjeeling
Dehradun
New Delhi
Gurgaon
Noida
Faridabad
Ghaziabad
Greater Noida
Dharamshala
Gangtok
Goa
Gorakhpur
Guntur
Guwahati
Gwalior
Haridwar
Hyderabad
Secunderabad
Indore
Jabalpur
Jaipur
Jalandhar
Jammu
Jamnagar
Jamshedpur
Jhansi
Jodhpur
Junagadh
Kanpur
Kharagpur
Kochi
Kolhapur
Kolkata
Howrah
Kota
Lucknow
Ludhiana
```

```
Madurai
Manali
Mangalore
Manipal
Udupi
Meerut
Mumbai
Thane
Navi Mumbai
Mussoorie
Mysore
Nagpur
Nainital
Nashik
Neemrana
Ooty
Palakkad
Patiala
Patna
Puducherry
Pune
Pushkar
Raipur
Rajkot
Ranchi
Rishikesh
Salem
Shimla
Siliguri
Srinagar
Surat
Thrissur
Tirupati
Trichy
Trivandrum
Udaipur
Varanasi
Vellore
Vijayawada
Vizag
Vadodara
```

In [34]:
```
df_restaurants_copy['city'].unique()
```

Out[34]:
```
array(['Agra', 'Ahmedabad', 'Gandhinagar', 'Ajmer', 'Alappuzha',
       'Allahabad', 'Amravati', 'Amritsar', 'Aurangabad', 'Bangalore',
       'Bhopal', 'Bhubaneshwar', 'Chandigarh', 'Mohali', 'Panchkula',
       'Zirakpur', 'Nayagaon', 'Chennai', 'Coimbatore', 'Cuttack',
       'Darjeeling', 'Dehradun', 'New Delhi', 'Gurgaon', 'Noida',
       'Faridabad', 'Ghaziabad', 'Greater Noida', 'Dharamshala',
       'Gangtok', 'Goa', 'Gorakhpur', 'Guntur', 'Guwahati', 'Gwalior',
       'Haridwar', 'Hyderabad', 'Secunderabad', 'Indore', 'Jabalpur',
       'Jaipur', 'Jalandhar', 'Jammu', 'Jamnagar', 'Jamshedpur', 'Jhansi',
       'Jodhpur', 'Junagadh', 'Kanpur', 'Kharagpur', 'Kochi', 'Kolhapur',
       'Kolkata', 'Howrah', 'Kota', 'Lucknow', 'Ludhiana', 'Madurai',
       'Manali', 'Mangalore', 'Manipal', 'Udupi', 'Meerut', 'Mumbai',
       'Thane', 'Navi Mumbai', 'Mussoorie', 'Mysore', 'Nagpur',
       'Nainital', 'Nashik', 'Neemrana', 'Ooty', 'Palakkad', 'Patiala',
       'Patna', 'Puducherry', 'Pune', 'Pushkar', 'Raipur', 'Rajkot',
       'Ranchi', 'Rishikesh', 'Salem', 'Shimla', 'Siliguri', 'Srinagar',
       'Surat', 'Thrissur', 'Tirupati', 'Trichy', 'Trivandrum', 'Udaipur',
       'Varanasi', 'Vellore', 'Vijayawada', 'Vizag', 'Vadodara'],
      dtype=object)
```

Let us find the count of restaurants in each city

In [35]:
```
df_restaurants_copy['name'].isna().sum()
```

Out[35]:
```
0
```

In [36]:
```
df_restaurants_copy.groupby(['city'])[['city','name']].head()
```

Out[36]:

	city	name
0	Agra	Bikanervala
1	Agra	Mama Chicken Mama Franky House
2	Agra	Bhagat Halwai
3	Agra	Bhagat Halwai
4	Agra	The Salt Cafe Kitchen & Bar
...	...	...
209266	Vadodara	Charcoal House
209267	Vadodara	Day Night Vada Pav & Dabeli
209268	Vadodara	Ph Se Food
209269	Vadodara	Marwari Food Corner
209270	Vadodara	Tasty Vada Pav

490 rows × 2 columns

In [37]:
```
i = 0
for var in df_restaurants_copy['city'].unique():
    print(var , ' has ', len(df_restaurants_copy[df_restaurants_copy['city'] =
= var]['name']), ' restaurants.')
```

```
Agra has 893 restaurants.
Ahmedabad has 1329 restaurants.
Gandhinagar has 96 restaurants.
Ajmer has 470 restaurants.
Alappuzha has 267 restaurants.
Allahabad has 567 restaurants.
Amravati has 440 restaurants.
Amritsar has 692 restaurants.
Aurangabad has 693 restaurants.
Bangalore has 2365 restaurants.
Bhopal has 971 restaurants.
Bhubaneshwar has 792 restaurants.
Chandigarh has 681 restaurants.
Mohali has 333 restaurants.
Panchkula has 174 restaurants.
Zirakpur has 154 restaurants.
Nayagaon has 15 restaurants.
Chennai has 2612 restaurants.
Coimbatore has 1019 restaurants.
Cuttack has 293 restaurants.
Darjeeling has 116 restaurants.
Dehradun has 805 restaurants.
New Delhi has 1847 restaurants.
Gurgaon has 662 restaurants.
Noida has 273 restaurants.
Faridabad has 81 restaurants.
Ghaziabad has 95 restaurants.
Greater Noida has 22 restaurants.
Dharamshala has 259 restaurants.
Gangtok has 132 restaurants.
Goa has 1169 restaurants.
Gorakhpur has 526 restaurants.
Guntur has 319 restaurants.
Guwahati has 784 restaurants.
Gwalior has 606 restaurants.
Haridwar has 401 restaurants.
Hyderabad has 866 restaurants.
Secunderabad has 97 restaurants.
Indore has 1093 restaurants.
Jabalpur has 598 restaurants.
Jaipur has 1456 restaurants.
Jalandhar has 643 restaurants.
Jammu has 549 restaurants.
Jamnagar has 425 restaurants.
Jamshedpur has 615 restaurants.
Jhansi has 371 restaurants.
Jodhpur has 731 restaurants.
Junagadh has 231 restaurants.
Kanpur has 836 restaurants.
Kharagpur has 116 restaurants.
Kochi has 1027 restaurants.
Kolhapur has 567 restaurants.
Kolkata has 1413 restaurants.
Howrah has 50 restaurants.
Kota has 622 restaurants.
Lucknow has 1290 restaurants.
Ludhiana has 992 restaurants.
```

```
Madurai  has  578  restaurants.
Manali  has  185  restaurants.
Mangalore  has  584  restaurants.
Manipal  has  162  restaurants.
Udupi  has  61  restaurants.
Meerut  has  580  restaurants.
Mumbai  has  2538  restaurants.
Thane  has  300  restaurants.
Navi Mumbai  has  256  restaurants.
Mussoorie  has  190  restaurants.
Mysore  has  585  restaurants.
Nagpur  has  1102  restaurants.
Nainital  has  246  restaurants.
Nashik  has  758  restaurants.
Neemrana  has  26  restaurants.
Ooty  has  276  restaurants.
Palakkad  has  178  restaurants.
Patiala  has  505  restaurants.
Patna  has  683  restaurants.
Puducherry  has  583  restaurants.
Pune  has  1911  restaurants.
Pushkar  has  183  restaurants.
Raipur  has  833  restaurants.
Rajkot  has  587  restaurants.
Ranchi  has  689  restaurants.
Rishikesh  has  258  restaurants.
Salem  has  418  restaurants.
Shimla  has  241  restaurants.
Siliguri  has  483  restaurants.
Srinagar  has  125  restaurants.
Surat  has  1001  restaurants.
Thrissur  has  405  restaurants.
Tirupati  has  277  restaurants.
Trichy  has  505  restaurants.
Trivandrum  has  617  restaurants.
Udaipur  has  775  restaurants.
Varanasi  has  598  restaurants.
Vellore  has  341  restaurants.
Vijayawada  has  530  restaurants.
Vizag  has  721  restaurants.
Vadodara  has  1002  restaurants.
```

4. Variable 'locality'

In [38]:
```
df_restaurants_copy['locality'].isna().sum()
```

Out[38]: 0

In [39]:
```
df_restaurants_copy['locality'].duplicated().sum()
```

Out[39]: 56686

In [40]:
```
df_restaurants_copy['locality'].unique()
```

Out[40]:
```
array(['Khandari', 'Agra Cantt', 'Shahganj', ..., 'Navapura',
       'L&T Knowledge City', 'Danteshwar'], dtype=object)
```

4. Variable 'latitude'

From the variable 'latitude', we know the latitudinal location of the restaurant

The Latitudinal extent of India 8º4'N to 37º6' N.

We must check whether we have any points beyond this extent.

In [41]:
```
df_restaurants_copy['latitude'].isna().sum()
```

Out[41]:
```
0
```

In [42]:
```
df_restaurants_copy['latitude'].duplicated().sum()
```

Out[42]:
```
7061
```

In [43]:
```
len(df_restaurants_copy[(df_restaurants_copy['latitude'] < 8.066667) | (df_res
taurants_copy['latitude'] > 37.1)])
```

Out[43]:
```
955
```

- We need to replace all these values with NaN's.

In [44]:
```
def replacement(x):
    if((x < 8.066667) | (x > 37.1)):
        return np.nan
    else:
        return x
df_restaurants_copy['latitude'] = df_restaurants_copy['latitude'].transform(la
mbda x: replacement(x))
```

- check if the values are replace by NaN's

In [45]:
```
len(df_restaurants_copy[(df_restaurants_copy['latitude'] < 8.066667) | (df_res
taurants_copy['latitude'] > 37.1)])
```

Out[45]:
```
0
```

In [46]:
```
df_restaurants_copy['latitude'].isna().sum()
```

Out[46]:
```
955
```

In [47]:
```
print('Now, the number of nan values has become same as number of values in th
e result above i.e. 955, which means all the qualifying values have been repla
ced with nan values')
```

```
Now, the number of nan values has become same as number of values in the resu
lt above i.e. 955, which means all the qualifying values have been replaced w
ith nan values
```

- We see all the values are replaced by NaN's

In [48]:
```
df_restaurants_copy[df_restaurants_copy['latitude'].isna()]['latitude'].head()
```

Out[48]:
```
111   NaN
113   NaN
122   NaN
145   NaN
149   NaN
Name: latitude, dtype: float64
```

5. Variable 'longitude'

From the variable 'longitude', we know the longitudinal location of the restaurant

The Longitudinal extent of India is from 68°7'E to 97°25'E

We must check whether we have any points beyond this extent.

In [49]:
```
df_restaurants_copy['longitude'].isna().sum()
```

Out[49]:
```
0
```

In [50]:
```
df_restaurants_copy['longitude'].duplicated().sum()
```

Out[50]:
```
7099
```

In [51]:
```
len(df_restaurants_copy[(df_restaurants_copy['longitude'] < 68.1166667) | (df_
restaurants_copy['longitude'] > 97.41666667)])
```

Out[51]:
```
957
```

- We need to replace all these values with NaN's.

In [52]:
```
def replacement2(x):
    if((x < 68.1166667) | (x > 97.41666667)):
        return np.nan
    else:
        return x
df_restaurants_copy['longitude'] = df_restaurants_copy['longitude'].transform(
lambda x: replacement2(x))
```

- Check if the values are replace by NaN's

In [53]:
```
len(df_restaurants_copy[(df_restaurants_copy['longitude'] < 68.1166667) | (df_
restaurants_copy['longitude'] > 97.41666667)])
```

Out[53]: 0

In [54]:
```
df_restaurants_copy['longitude'].isna().sum()
```

Out[54]: 957

In [55]:
```
print('Now, the number of nan values has become same as number of values in th
e result above i.e. 957, which means all the qualifying values have been repla
ced with nan values')
```

```
Now, the number of nan values has become same as number of values in the resu
lt above i.e. 957, which means all the qualifying values have been replaced w
ith nan values
```

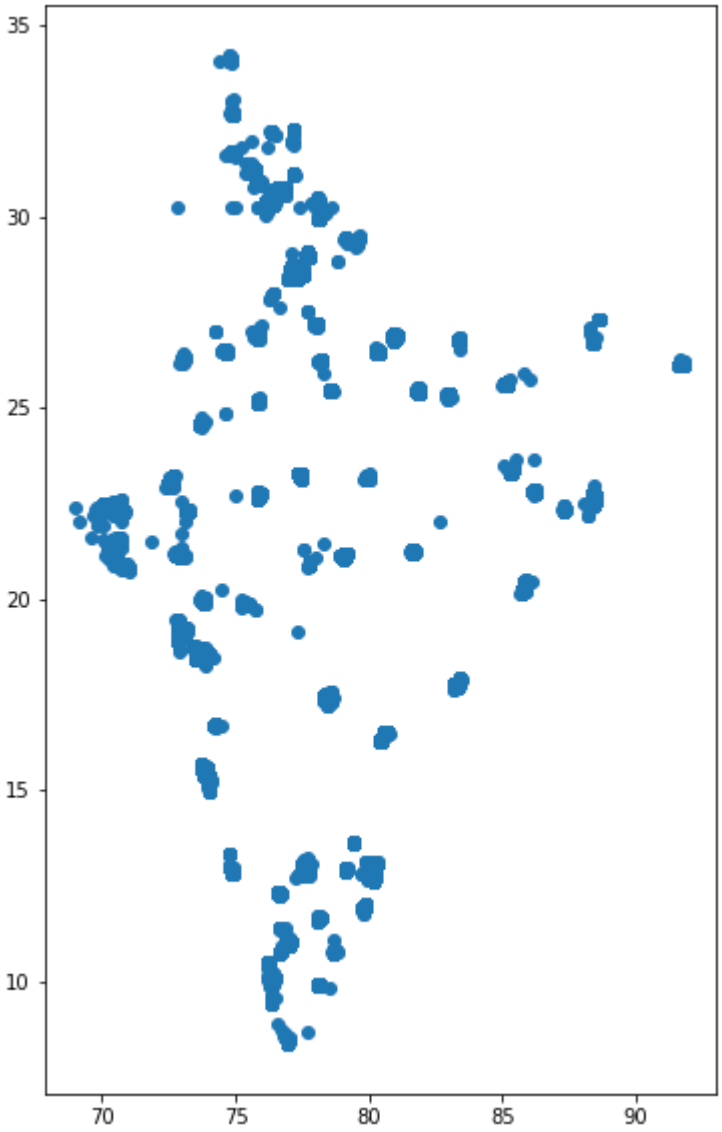
- From variable 'latitude' and 'longitude', plot the location of restaurants.

In [56]:
```
df_temp = df_restaurants_copy[['latitude','longitude']].dropna()
df_temp.isna().sum().sum()
```

Out[56]: 0

In [57]:
```
fig, ax = plt.subplots(figsize=(6,10))
plt.scatter(df_temp['longitude'],df_temp['latitude'])
```

Out[57]: <matplotlib.collections.PathCollection at 0x3a571795c8>



In [58]:
```
pip install gmplot
```

```
Requirement already satisfied: gmplot in c:\users\kejri\anaconda3\lib\site-pa
ckages (1.2.0)
Requirement already satisfied: requests in c:\users\kejri\anaconda3\lib\site-
packages (from gmplot) (2.22.0)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in
c:\users\kejri\anaconda3\lib\site-packages (from requests->gmplot) (1.24.2)
Requirement already satisfied: chardet<3.1.0,>=3.0.2 in c:\users\kejri\anacon
da3\lib\site-packages (from requests->gmplot) (3.0.4)
Requirement already satisfied: idna<2.9,>=2.5 in c:\users\kejri\anaconda3\lib
\site-packages (from requests->gmplot) (2.8)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\kejri\anaconda3
\lib\site-packages (from requests->gmplot) (2019.9.11)
Note: you may need to restart the kernel to use updated packages.
```

In [59]:
```
import gmplot
```

In [60]:
```
#lat, long, zoom
google_map = gmplot.GoogleMapPlotter(28.7041,77.1025, 5, apikey="" )
```

In [61]:
```
#google_map.apikey = ""
```

In [62]:
```
google_map.scatter(df_temp['latitude'],df_temp['longitude'], '#cb202d', size =
35, marker = False)
```

In [63]:
```
google_map.draw("location.html")
```

In [64]:
```
import webbrowser
webbrowser.open('location.html')
```

Out[64]:
```
True
```

6. Variable 'cuisines'

In [65]:
```
df_restaurants_copy['cuisines'].isna().sum()
```

Out[65]:
```
470
```

In [66]:
```
df_restaurants_copy['cuisines'].duplicated().sum()
```

Out[66]:
```
51034
```

- To find the unique cusines we write a small user defined function.

In [67]:
```
df_restaurants_copy['cuisines']
```

Out[67]:
```
0         North Indian, South Indian, Mithai, Street Foo...
1         North Indian, Mughlai, Rolls, Chinese, Fast Fo...
2                                        Fast Food, Mithai
3                Desserts, Bakery, Fast Food, South Indian
4                       North Indian, Continental, Italian
                                ...
211882                                           Ice Cream
211925                      Gujarati, North Indian, Chinese
211926                                Gujarati, Street Food
211940                                           Fast Food
211942                            Fast Food, Sandwich, Salad
Name: cuisines, Length: 60417, dtype: object
```

In [68]:
```
def cuisines(x):
    x=x.dropna()
    x=np.asarray(x.transform(lambda x: x.split(", ")).to_numpy())
    x= pd.Series(np.concatenate(x, axis=0))
    print(x.unique())
cuisines(df_restaurants_copy['cuisines'])
```

```
['North Indian' 'South Indian' 'Mithai' 'Street Food' 'Desserts' 'Mughlai'
 'Rolls' 'Chinese' 'Fast Food' 'Bakery' 'Continental' 'Italian' 'Pizza'
 'Cafe' 'Burger' 'Wraps' 'Beverages' 'Rajasthani' 'Mexican' 'Healthy Food'
 'Sandwich' 'Salad' 'Momos' 'Lebanese' 'Mediterranean' 'Thai' 'Gujarati'
 'Indian' 'Finger Food' 'European' 'Tea' 'Asian' 'Bar Food' 'Kebab' 'Paan'
 'Biryani' 'Juices' 'Ice Cream' 'Japanese' 'Korean' 'Afghan' 'Awadhi'
 'Hyderabadi' 'Lucknowi' 'Roast Chicken' 'Drinks Only' 'Coffee' 'American'
 'BBQ' 'Maharashtrian' 'Modern Indian' 'Andhra' 'Konkan' 'Kerala' 'Sushi'
 'Parsi' 'Greek' 'Bengali' 'Seafood' 'Frozen Yogurt' 'Arabian'
 'Indonesian' 'Sindhi' 'Hot dogs' 'Goan' 'Charcoal Chicken' 'Raw Meats'
 'Grill' 'Malwani' 'Cantonese' 'Pakistani' 'Steak' 'Vietnamese'
 'Singaporean' 'Middle Eastern' 'British' 'French' 'Burmese' 'Kashmiri'
 'Mangalorean' 'Malaysian' 'Tex-Mex' 'Spanish' 'Chettinad' 'Tibetan'
 'German' 'Belgian' 'Turkish' 'Bihari' 'Odia' 'Naga' 'Bubble Tea'
 'Moroccan' 'Sri Lankan' 'Mandi' 'Coffee and Tea' 'Cafe Food' 'Oriental'
 'Cuisine Varies' 'Pan Asian' 'Mishti' 'Portuguese' 'Iranian'
 'North Eastern' 'Mongolian' 'Irish' 'Tamil' 'Russian' 'Panini'
 'South American' 'Fusion' 'Nepalese' 'International' 'Modern Australian'
 'Poké' 'Falafel' 'Armenian' 'Peruvian' 'Brazilian' 'Himachali' 'Israeli'
 'Bohri' 'Assamese' 'Bangladeshi' 'African' 'Egyptian' 'Crepes'
 'Fried Chicken' 'Swedish' 'Cake' 'Garhwali' 'Vegan' 'Afghani']
```

In [69]:
```
cuisines(df_restaurants_copy[df_restaurants_copy['city'] == 'Agra']['cuisines'
])
```

```
['North Indian' 'South Indian' 'Mithai' 'Street Food' 'Desserts' 'Mughlai'
 'Rolls' 'Chinese' 'Fast Food' 'Bakery' 'Continental' 'Italian' 'Pizza'
 'Cafe' 'Burger' 'Wraps' 'Beverages' 'Rajasthani' 'Mexican' 'Healthy Food'
 'Sandwich' 'Salad' 'Momos' 'Lebanese' 'Mediterranean' 'Thai' 'Gujarati'
 'Indian' 'Finger Food' 'European' 'Tea' 'Asian' 'Bar Food' 'Kebab' 'Paan'
 'Biryani' 'Juices' 'Ice Cream' 'Japanese' 'Korean' 'Afghan' 'Awadhi'
 'Hyderabadi' 'Lucknowi' 'Roast Chicken' 'Drinks Only' 'Coffee']
```

In [70]:
```
cuisines(df_restaurants_copy[df_restaurants_copy['city'] == 'Srinagar']['cuisi
nes'])
```

```
['Cafe' 'Pizza' 'Continental' 'North Indian' 'South Indian' 'Kashmiri'
 'Fast Food' 'Italian' 'Mughlai' 'Chinese' 'Desserts' 'Beverages'
 'Sandwich' 'Street Food' 'Mithai' 'Bakery' 'Tibetan' 'Afghan' 'Asian'
 'Burger' 'Ice Cream' 'European' 'American' 'Thai' 'Finger Food' 'BBQ'
 'Biryani' 'Tea']
```

In [71]:
```
cuisines(df_restaurants_copy[(df_restaurants_copy['city'] == 'Srinagar') & (df
_restaurants_copy['name'] == 'Winterfell Cafe')]['cuisines'])
```

```
['Cafe']
```

In [72]:
```
cuisines(df_restaurants_copy[(df_restaurants_copy['city'] == 'Srinagar') & (df
_restaurants_copy['name'] == 'Nathus Sweets')]['cuisines'])
```

```
['North Indian' 'South Indian' 'Chinese' 'Street Food' 'Fast Food'
 'Mithai' 'Desserts']
```

- find out the frequency of each cuisine

In [73]:
```
def cuisines_freq(x):
    x=x.dropna()
    x=np.asarray(x.transform(lambda x: x.split(", ")).to_numpy())
    x= pd.Series(np.concatenate(x, axis=0))
    print(x.value_counts())
cuisines_freq(df_restaurants_copy['cuisines'])
```

```
North Indian     21259
Chinese          14139
Fast Food        13191
Desserts          7755
Beverages         7486
                 ...
African              2
International        1
Vegan                1
Mandi                1
Swedish              1
Length: 133, dtype: int64
```

In [74]:
```
cuisines_freq(df_restaurants_copy[df_restaurants_copy['city'] == 'Srinagar'][
'cuisines'])
```

```
North Indian    44
Chinese         22
Kashmiri        19
Mughlai         16
Fast Food       15
Bakery          13
Cafe            13
Italian          9
Continental      8
Mithai           6
Pizza            6
South Indian     5
Desserts         4
Beverages        3
Asian            2
Ice Cream        2
Street Food      2
American         1
Sandwich         1
Burger           1
Afghan           1
Thai             1
Tea              1
BBQ              1
Biryani          1
Finger Food      1
Tibetan          1
European         1
dtype: int64
```

8. Variable 'average_cost_for_two'

In [75]:
```
df_restaurants_copy['average_cost_for_two'].isna().sum()
```

Out[75]: 0

In [76]:
```
df_restaurants_copy['average_cost_for_two'].duplicated().sum()
```

Out[76]: 60272

In [77]:
```
len(df_restaurants_copy['average_cost_for_two'])
```

Out[77]: 60417

In [78]:
```
df_restaurants_copy['average_cost_for_two'].min(), round(df_restaurants_copy[
'average_cost_for_two'].mean(),2), df_restaurants_copy['average_cost_for_two']
.max()
```

Out[78]: (0, 538.28, 30000)

9. Variable 'price_range'

In [79]:
```
df_restaurants_copy['price_range'].isna().sum()
```

Out[79]:
```
0
```

In [80]:
```
df_restaurants_copy['price_range'].duplicated().sum()
```

Out[80]:
```
60413
```

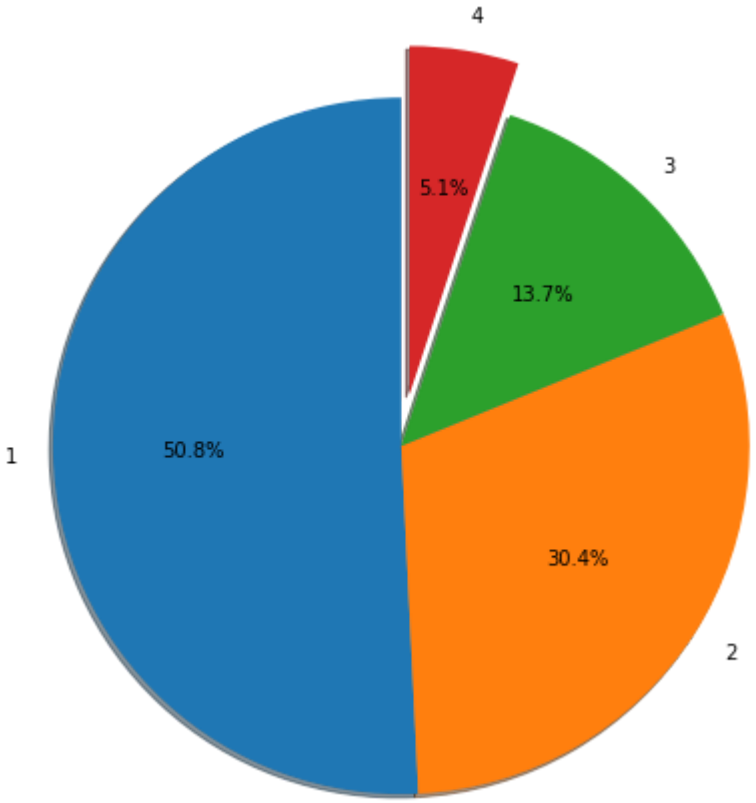
In [81]:
```
df_restaurants_copy['price_range'].unique()
```

Out[81]:
```
array([2, 1, 3, 4], dtype=int64)
```

- visualize a exploded pie chart.

In [82]:
```
labels = 1,2,3,4
sizes = [len(df_restaurants_copy[df_restaurants_copy['price_range']==1]['price
_range']), len(df_restaurants_copy[df_restaurants_copy['price_range']==2]['pri
ce_range']), len(df_restaurants_copy[df_restaurants_copy['price_range']==3]['p
rice_range']), len(df_restaurants_copy[df_restaurants_copy['price_range']==4][
'price_range'])]
explode = (0, 0, 0, 0.3)
```

In [83]:
```
fig, ax = plt.subplots()
ax.pie(sizes, labels=labels, explode=explode, autopct='%1.1f%%', shadow=True,
startangle=90, radius = 2)
plt.show()
```



10. Variable 'highlights'

In [84]:
```
df_restaurants_copy['highlights'].isna().sum()
```

Out[84]:
```
743
```

In [85]:
```
df_restaurants_copy['highlights'].duplicated().sum()
```

Out[85]:
```
28962
```

In [86]:
```
df_restaurants_copy['highlights'].head()
```

Out[86]:
```
0    Lunch, Takeaway Available, Credit Card, Dinner...
1    Delivery, No Alcohol Available, Dinner, Takeaw...
2    No Alcohol Available, Dinner, Takeaway Availab...
3    Takeaway Available, Credit Card, Lunch, Delive...
4    Lunch, Serves Alcohol, Cash, Credit Card, Dinn...
Name: highlights, dtype: object
```

- write a small function to know the number of times a facility has appeared in the 'Highlights'.

In [87]:
```
def highlights_freq(x):
    x=x.dropna()
    x=np.asarray(x.transform(lambda x: x.split(", ")).to_numpy())
    x= pd.Series(np.concatenate(x, axis=0))
    print(x.value_counts())
highlights_freq(df_restaurants_copy['highlights'])
```

```
Cash                      57533
Takeaway Available        51010
Indoor Seating            44847
Dinner                    41685
Lunch                     40012
                          ...
Members Only                  3
Bira 91 Beer                  2
Alipay Accepted               1
Subscription Required         1
Subscription Available        1
Length: 103, dtype: int64
```

- Now we find out which facility occurs most number of in the data.

In [88]:
```
def highlights_freq_max(x):
    x=x.dropna()
    x=np.asarray(x.transform(lambda x: x.split(", ")).to_numpy())
    x= pd.Series(np.concatenate(x, axis=0))
    print(x.value_counts().head(1))
highlights_freq_max(df_restaurants_copy['highlights'])
```

```
Cash    57533
dtype: int64
```

11. Variable 'aggregate_rating'

In [89]:
```
df_restaurants_copy['aggregate_rating'].isna().sum()
```

Out[89]: 0

In [90]:
```
df_restaurants_copy['aggregate_rating'].duplicated().sum()
```

Out[90]: 60384

In [91]:
```
df_restaurants_copy['aggregate_rating'].head()
```

Out[91]:
```
0    4.4
1    4.4
2    4.2
3    4.3
4    4.9
Name: aggregate_rating, dtype: float64
```

In [92]:
```
round(df_restaurants_copy['aggregate_rating'].mean(),2)
```

Out[92]:
```
3.03
```

12. Variable 'rating_text'

In [93]:
```
df_restaurants_copy['rating_text'].isna().sum()
```

Out[93]:
```
0
```

In [94]:
```
df_restaurants_copy['rating_text'].duplicated().sum()
```

Out[94]:
```
60378
```

In [95]:
```
df_restaurants_copy['rating_text'].head()
```

Out[95]:
```
0    Very Good
1    Very Good
2    Very Good
3    Very Good
4    Excellent
Name: rating_text, dtype: object
```

In [96]:
```
df_restaurants_copy['rating_text'].unique()
```

Out[96]:
```
array(['Very Good', 'Excellent', 'Good', 'Average', 'Not rated', 'Poor',
       'Dobré', 'Baik', 'Sangat Baik', 'Excelente', 'Bardzo dobrze',
       'Wybitnie', 'Ottimo', 'Muito Bom', 'Velmi dobré', 'Skvělá volba',
       'Muy Bueno', 'Bom', 'İyi', 'Çok iyi', 'Harika', 'Terbaik',
       'Skvělé', 'Průměr', 'Ortalama', 'Bueno', 'Eccellente', 'Muito bom',
       'Dobrze', 'Buono', 'Média', 'Scarso', 'Promedio', 'Veľmi dobré',
       'Vynikajúce', 'Średnio', 'Priemer', 'Biasa', 'Media'], dtype=object)
```

In [97]:
```
df_restaurants_copy[df_restaurants_copy['rating_text'] == "Very Good"]['aggregate_rating'].unique()
```

Out[97]:
```
array([4.4, 4.2, 4.3, 4. , 4.1])
```

In [98]:
```
df_restaurants_copy[df_restaurants_copy['rating_text'] == "Excellent"]['aggregate_rating'].unique()
```

Out[98]:
```
array([4.9, 4.6, 4.5, 4.7, 4.8])
```

In [99]:
```
df_restaurants_copy[df_restaurants_copy['rating_text'] == "Good"]['aggregate_rating'].unique()
```

Out[99]:
```
array([3.8, 3.5, 3.9, 3.6, 3.7])
```

In [100]:
```
df_restaurants_copy[df_restaurants_copy['rating_text'] == "Average"]['aggregate_rating'].unique()
```

Out[100]:
```
array([3.4, 3.2, 3.3, 2.8, 3.1, 2.6, 3. , 2.7, 2.9, 2.5])
```

In [101]:
```
df_restaurants_copy[df_restaurants_copy['rating_text'] == "Not rated"]['aggregate_rating'].unique()
```

Out[101]:
```
array([0.])
```

In [102]:
```
df_restaurants_copy[df_restaurants_copy['rating_text'] == "Poor"]['aggregate_rating'].unique()
```

Out[102]:
```
array([2.2, 2.3, 2.4, 2.1, 1.8, 2. , 1.9])
```

Creating a New feature for better understanding of ratings

In [103]:
```
def ratings(x):
    if x>=4.5:
        return "Excellent"
    elif ((x<4.5) & (x>=4)):
        return "Very Good"
    elif ((x<4) & (x>=3.5)):
        return 'Good'
    elif ((x<3.5) & (x>=2.5)):
        return 'Average'
    elif ((x<2.5) & (x>0)):
        return 'Poor'
    else:
        return 'Not Rated'
df_restaurants_copy['new_ratings'] = np.nan
df_restaurants_copy['new_ratings'].head()
```

Out[103]:
```
0   NaN
1   NaN
2   NaN
3   NaN
4   NaN
Name: new_ratings, dtype: float64
```

In [104]:
```
df_restaurants_copy['new_ratings'] = df_restaurants_copy['aggregate_rating'].transform(lambda x: ratings(x))
```

In [105]:
```
df_restaurants_copy['rating_text'].unique()
```

Out[105]:
```
array(['Very Good', 'Excellent', 'Good', 'Average', 'Not rated', 'Poor',
       'Dobré', 'Baik', 'Sangat Baik', 'Excelente', 'Bardzo dobrze',
       'Wybitnie', 'Ottimo', 'Muito Bom', 'Velmi dobré', 'Skvělá volba',
       'Muy Bueno', 'Bom', 'İyi', 'Çok iyi', 'Harika', 'Terbaik',
       'Skvělé', 'Průměr', 'Ortalama', 'Bueno', 'Eccellente', 'Muito bom',
       'Dobrze', 'Buono', 'Média', 'Scarso', 'Promedio', 'Veľmi dobré',
       'Vynikajúce', 'Średnio', 'Priemer', 'Biasa', 'Media'], dtype=object)
```

In [106]:
```
df_restaurants_copy['new_ratings'].unique()
```

Out[106]:
```
array(['Very Good', 'Excellent', 'Good', 'Average', 'Not Rated', 'Poor'],
      dtype=object)
```

In [107]:
```
df_restaurants_copy['new_ratings'].head()
```

Out[107]:
```
0    Very Good
1    Very Good
2    Very Good
3    Very Good
4    Excellent
Name: new_ratings, dtype: object
```

In [108]:
```
df_restaurants_copy = df_restaurants_copy.drop(['rating_text'], axis=1)
```

13. Variable 'votes'

In [109]:
```
df_restaurants_copy['votes'].isna().sum()
```

Out[109]:
```
0
```

In [110]:
```
df_restaurants_copy['votes'].duplicated().sum()
```

Out[110]:
```
57775
```

In [111]:
```
df_restaurants_copy['votes'].min(), round(df_restaurants_copy['votes'].mean(),
2), df_restaurants_copy['votes'].max()
```

Out[111]:
```
(0, 261.5, 42539)
```

In [112]:
```
df_restaurants_copy['votes'].head()
```

Out[112]:
```
0     814
1    1203
2     801
3     693
4     470
Name: votes, dtype: int64
```

14. Variable 'photo_count'

In [113]:
```
df_restaurants_copy['photo_count'].isna().sum()
```

Out[113]:
```
0
```

In [114]:
```
df_restaurants_copy['photo_count'].duplicated().sum()
```

Out[114]:
```
57903
```

In [115]:
```
df_restaurants_copy['photo_count'].min(), round(df_restaurants_copy['photo_cou
nt'].mean(),2), df_restaurants_copy['photo_count'].max()
```

Out[115]:
```
(0, 193.95, 17702)
```

15. Variable 'delivery'

In [116]:
```
df_restaurants_copy['delivery'].isna().sum()
```

Out[116]:
```
0
```

In [117]:
```
df_restaurants_copy['delivery'].duplicated().sum()
```

Out[117]:
```
60414
```

In [118]:
```
df_restaurants_copy['delivery'].unique()
```

Out[118]:
```
array([-1,  1,  0], dtype=int64)
```

6. Check for missing values

In [119]:
```
df_restaurants_copy.isna().sum()
```

Out[119]:
```
name                      0
establishment          1920
url                       0
address                  18
city                      0
city_id                   0
locality                  0
latitude                955
longitude               957
zipcode                47869
cuisines                470
timings                1070
average_cost_for_two      0
price_range               0
highlights              743
aggregate_rating          0
votes                     0
photo_count               0
opentable_support        19
delivery                  0
new_ratings               0
dtype: int64
```

In [120]:
```
df_restaurants_copy.isna().sum().sum()
```

Out[120]:
```
54021
```

6. Study summary statistics

Let us check the summary statistics for numerical variables.

In [121]:
```
df_restaurants_copy[["average_cost_for_two","aggregate_rating","votes","photo_
count"]].describe()
```

Out[121]:

	average_cost_for_two	aggregate_rating	votes	photo_count
count	60417.000000	60417.000000	60417.000000	60417.000000
mean	538.283000	3.032799	261.496052	193.954533
std	593.840932	1.440796	728.039842	702.078844
min	0.000000	0.000000	0.000000	0.000000
25%	200.000000	2.900000	7.000000	1.000000
50%	400.000000	3.500000	42.000000	11.000000
75%	600.000000	4.000000	207.000000	82.000000
max	30000.000000	4.900000	42539.000000	17702.000000

In [122]:
```
print('Sum')
df_restaurants_copy[["average_cost_for_two","aggregate_rating","votes","photo_
count"]].sum()
```

```
Sum
```

Out[122]:
```
average_cost_for_two    32521444.0
aggregate_rating          183232.6
votes                   15798807.0
photo_count             11718151.0
dtype: float64
```

In [123]:
```
print('Mode')
df_restaurants_copy[["average_cost_for_two","aggregate_rating","votes","photo_
count"]].mode()
```

```
Mode
```

Out[123]:

	average_cost_for_two	aggregate_rating	votes	photo_count
0	200	0.0	0	0

7. Study correlation

In [124]:
```
df_restaurants_copy[["average_cost_for_two","aggregate_rating","votes","photo_
count"]].duplicated().sum()
```

Out[124]:
```
18691
```

In [125]:
```
df_temp =  df_restaurants_copy[["average_cost_for_two","aggregate_rating","votes","photo_count"]].drop_duplicates()
df_temp_copy = df_temp.copy()
df_temp_copy2 = df_temp.copy()
df_temp.shape
```

Out[125]: (41726, 4)

In [126]:
```
df_temp.isna().sum()
```

Out[126]:
```
average_cost_for_two    0
aggregate_rating        0
votes                   0
photo_count             0
dtype: int64
```

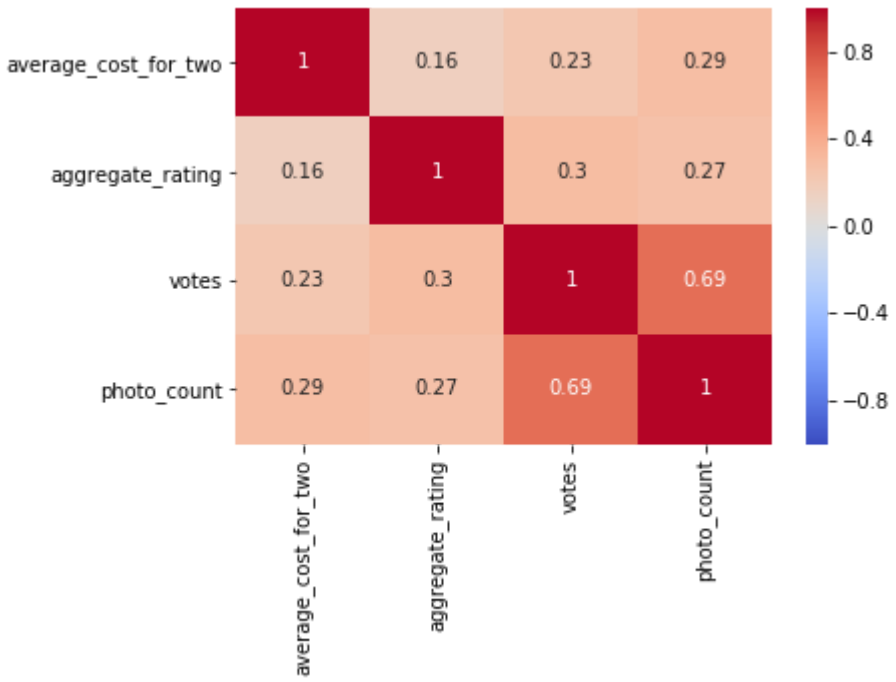
In [127]:
```
df_temp.corr()
```

Out[127]:

	average_cost_for_two	aggregate_rating	votes	photo_count
average_cost_for_two	1.000000	0.158541	0.230539	0.291334
aggregate_rating	0.158541	1.000000	0.302916	0.266182
votes	0.230539	0.302916	1.000000	0.687783
photo_count	0.291334	0.266182	0.687783	1.000000

In [128]:
```
#ax.get_ylim()
ax=sns.heatmap(df_temp.corr(), annot=True, vmin=-1, vmax=1, center= 0, cmap= 'coolwarm')
ax.set_ylim(4.0, 0)
```

Out[128]: (4.0, 0)



In [129]:
```
print('Without removing the outliers, the restaurants that have more number of photos also have more number of votes')
```

```
Without removing the outliers, the restaurants that have more number of photo
s also have more number of votes
```

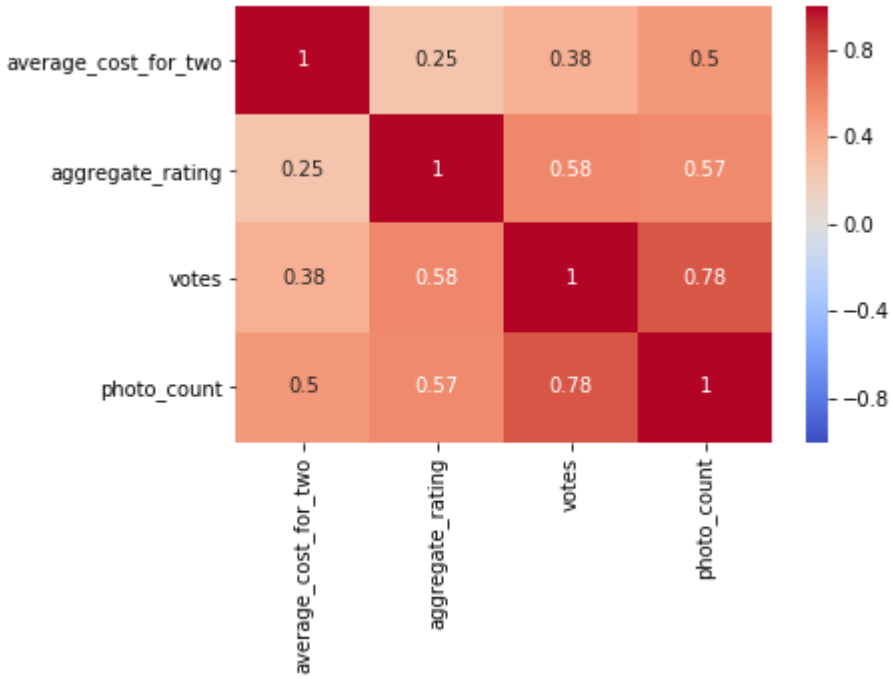
In [130]:
```
def replace_outliers(x):
    Q1 = df_temp[x].quantile(0.25)
    Q3 = df_temp[x].quantile(0.75)
    IQR = Q3 - Q1
    print("Outliers Number in ", x, ": ", ((df_temp[x] < (Q1 - 1.5 * IQR)) | (
df_temp[x] > (Q3 + 1.5 * IQR))).sum(), "out of ", df_temp[x].shape[0])
    ##Replaced outliers in HDI for year
    whisker1=Q1-1.5*IQR
    for i in (np.where((df_temp[x] < whisker1))):
        df_temp.iloc[i, df_temp.columns.get_loc(x)]= whisker1
    whisker2=Q3+1.5*IQR
    for i in (np.where((df_temp[x] > whisker2))):
        df_temp.iloc[i, df_temp.columns.get_loc(x)]= whisker2
    print('Outliers left: ',len(np.where((((df_temp[x] <(Q1-1.5*IQR)) | (df_te
mp[x] >(Q3+1.5*IQR)))))[0]))
```

In [131]:
```
replace_outliers('average_cost_for_two')
replace_outliers('aggregate_rating')
replace_outliers('votes')
replace_outliers('photo_count')
```

```
Outliers Number in  average_cost_for_two :  4267 out of  41726
Outliers left:  0
Outliers Number in  aggregate_rating :  1131 out of  41726
Outliers left:  0
Outliers Number in  votes :  4914 out of  41726
Outliers left:  0
Outliers Number in  photo_count :  6155 out of  41726
Outliers left:  0
```

In [132]:
```
#ax.get_ylim()
ax=sns.heatmap(df_temp.corr(), annot=True, vmin=-1, vmax=1, center= 0, cmap=
'coolwarm')
ax.set_ylim(4.0, 0)
```

Out[132]: (4.0, 0)



In [133]:
```
print('After replacing outliers with whiskers, new correlations have been foun
d')
```

```
After replacing outliers with whiskers, new correlations have been found
```

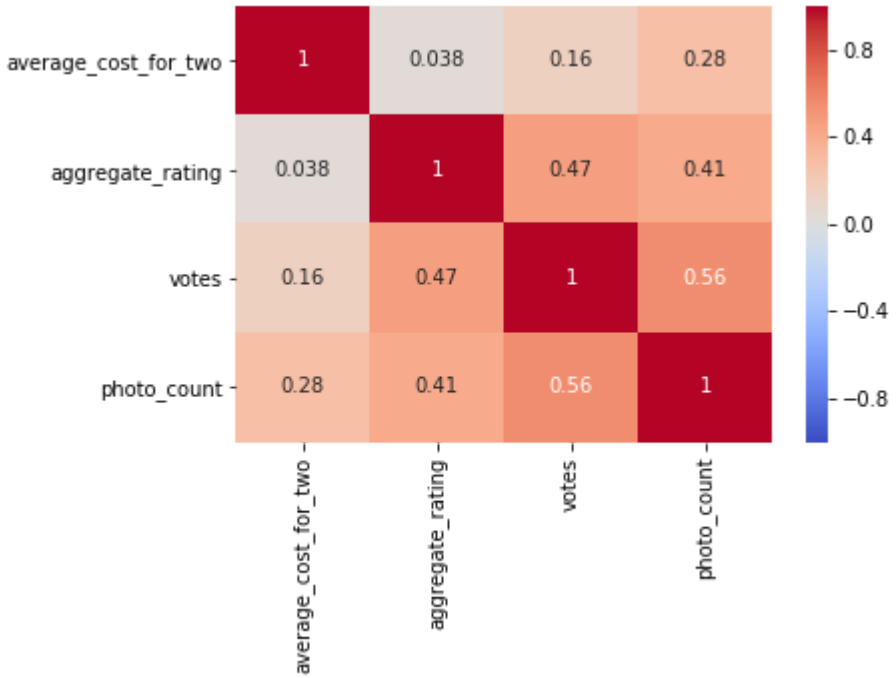
In [134]:
```
def del_outliers(x):
    Q1 = df_temp[x].quantile(0.25)
    Q3 = df_temp[x].quantile(0.75)
    IQR = Q3 - Q1
    print("Outliers Number in (rows being dropped)", x, ": ", ((df_temp[x] < (
Q1 - 1.5 * IQR)) | (df_temp[x] > (Q3 + 1.5 * IQR))).sum(), "out of ", df_temp[
x].shape[0])
    whisker1=Q1-1.5*IQR
    whisker2=Q3+1.5*IQR
    ##Deleting rows having outliers in HDI for year or suicides_no based on IQ
R Score method
    for i in (np.where((df_temp_copy[x] < whisker1) | (df_temp_copy[x] > whisk
er2))):
        df_temp_copy.drop(df_temp_copy.index[i],inplace=True)
    print('Outliers left: ', len(np.where((((df_temp_copy[x] <(Q1-1.5*IQR)) |
(df_temp_copy[x] >(Q3+1.5*IQR)))))[0]))
```

In [135]:
```
del_outliers('average_cost_for_two')
del_outliers('aggregate_rating')
del_outliers('votes')
del_outliers('photo_count')
```

```
Outliers Number in (rows being dropped) average_cost_for_two :  0 out of  417
26
Outliers left:  0
Outliers Number in (rows being dropped) aggregate_rating :  0 out of  41726
Outliers left:  0
Outliers Number in (rows being dropped) votes :  0 out of  41726
Outliers left:  0
Outliers Number in (rows being dropped) photo_count :  0 out of  41726
Outliers left:  0
```

In [136]:
```
#ax.get_ylim()
ax=sns.heatmap(df_temp_copy.corr(), annot=True, vmin=-1, vmax=1, center= 0, cm
ap= 'coolwarm')
ax.set_ylim(4.0, 0)
```

Out[136]: (4.0, 0)



In [137]:
```
print('After deleting rows having outliers, new correlations have been found')
```

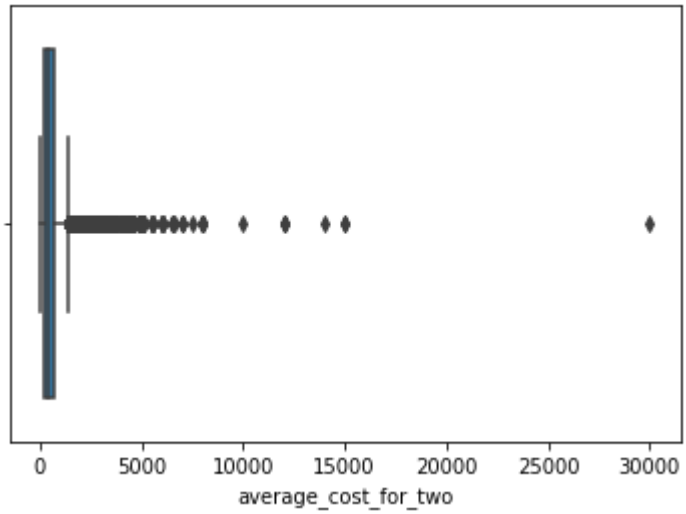
```
After deleting rows having outliers, new correlations have been found
```

8. Detect outliers

In [138]:
```
def detect_outliers(x):
    Q1 = df_temp_copy2[x].quantile(0.25)
    Q3 = df_temp_copy2[x].quantile(0.75)
    IQR = Q3 - Q1
    print("Number of Outliers in ", x, ": ", ((df_temp_copy2[x] < (Q1 - 1.5 *
IQR)) | (df_temp_copy2[x] > (Q3 + 1.5 * IQR))).sum(), "out of ", df_temp_copy2
[x].shape[0])
    #whisker1=Q1-1.5*IQR
    #whisker2=Q3+1.5*IQR
    sns.boxplot(x=df_temp_copy2[x])
```

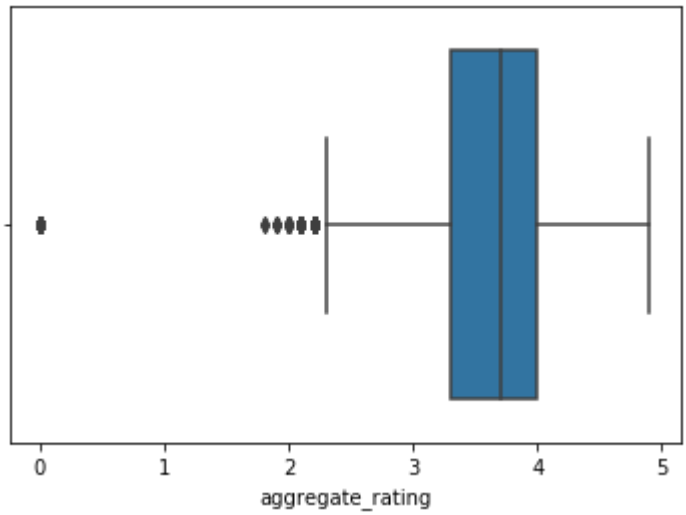
In [139]:
```
detect_outliers('average_cost_for_two')
```

```
Number of Outliers in  average_cost_for_two :  4267 out of  41726
```



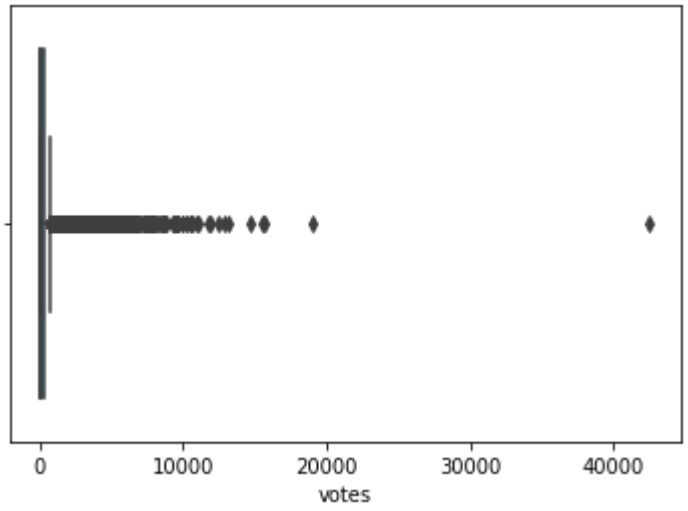
In [140]:
```
detect_outliers('aggregate_rating')
```

```
Number of Outliers in  aggregate_rating :  1131 out of  41726
```



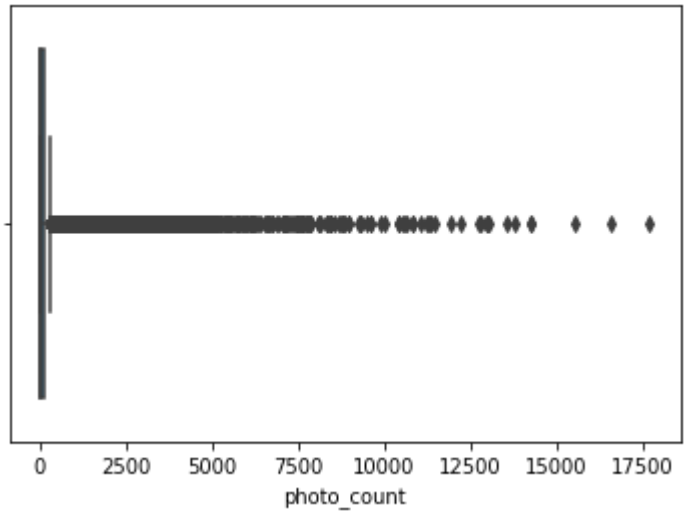
In [141]:
```
detect_outliers('votes')
```

```
Number of Outliers in  votes :  4914 out of  41726
```



In [142]:
```
detect_outliers('photo_count')
```

```
Number of Outliers in  photo_count :  6155 out of  41726
```



9. Create a new variable 'region'

Create a variable 'region' with four categories 'northern','eastern', 'southern', 'western' and 'central'. To do so, use the 'city' column, group all cities belonging to the same region.

In [143]:
```
#Manually created an excel file from data available at the source url: http://
www.indianventurez.com/city_list.htm
df=pd.read_excel('cities.xlsx')
df.head()
```

Out[143]:

	CITY	State	Category
0	AGARTALA	TRIPURA	NORTH EAST INDIA
1	AGRA	UTTAR PRADESH	NORTH INDIA
2	AHMEDABAD	GUJARAT	WEST INDIA
3	AIZWAL	MIZORAM	EAST INDIA
4	AJMER	RAJASTHAN	WEST INDIA

In [144]:
```
df=df.dropna()
df['Category'] = df['Category'].replace('WEST INDIA\xa0',"WEST INDIA").replace
('SOUTH INDIA\xa0',"SOUTH INDIA").replace('NORTH-EAST INDIA',"EAST INDIA").rep
lace('NORTH EAST INDIA',"EAST INDIA").replace('WESTERN REGION',"WEST INDIA")
df['Category'].unique()
```

Out[144]:
```
array(['EAST INDIA', 'NORTH INDIA', 'WEST INDIA', 'SOUTH INDIA',
       'CENTRAL INDIA'], dtype=object)
```

In [145]:
```
df['CITY']= df['CITY'].transform(lambda x: x.title())
df.head()
```

Out[145]:

	CITY	State	Category
0	Agartala	TRIPURA	EAST INDIA
1	Agra	UTTAR PRADESH	NORTH INDIA
2	Ahmedabad	GUJARAT	WEST INDIA
3	Aizwal	MIZORAM	EAST INDIA
4	Ajmer	RAJASTHAN	WEST INDIA

In [146]:
```
df[df['Category']=='NORTH INDIA']['CITY'].values
```

Out[146]:
```
array(['Agra', 'Allahabad', 'Almora', 'Ambala', 'Amritsar', 'Auli',
       'Baddi', 'Badrinath', 'Balrampur', 'Bareilly', 'Betalghat',
       'Bhimtal', 'Binsar', 'Chail', 'Chamba', 'Chandigarh',
       'Corbett National Park', 'Dalhousie', 'Dehradun', 'Dharamshala',
       'Faridabad', 'Firozabad', 'Gangotri', 'Garhmukteshwar', 'Garhwal',
       'Ghaziabad', 'Greater Noida', 'Gulmarg', 'Gurgaon', 'Hansi',
       'Haridwar', 'Jalandhar', 'Jammu', 'Jhansi', 'Kanatal', 'Kargil',
       'Karnal', 'Kasauli', 'Kashipur', 'Katra', 'Kausani', 'Kaza',
       'Kedarnath', 'Khajjiar', 'Kufri', 'Kullu', 'Kushinagar', 'Leh',
       'Lucknow', 'Ludhiana', 'Manali', 'Manesar', 'Marchula', 'Mathura',
       'Mcleodganj', 'Mohali', 'Moradabad', 'Mukteshwar', 'Mussoorie',
       'Nahan', 'Nainital', 'Naldhera', 'New Delhi', 'Noida', 'Palampur',
       'Pahalgam', 'Panchkula', 'Pantnagar', 'Parwanoo', 'Patiala',
       'Pathankot', 'Patnitop', 'Phagwara', 'Pinjore', 'Pragpur',
       'Rai Bareilly', 'Ram Nagar', 'Ranikhet', 'Rishikesh', 'Sattal',
       'Shimla', 'Solan', 'Sonauli', 'Srinagar', 'Udhampur', 'Uttarkashi',
       'Varanasi', 'Yamunotri'], dtype=object)
```

In [147]:
```
df[df['Category']=='EAST INDIA']['CITY'].values
```

Out[147]:
```
array(['Agartala', 'Aizwal', 'Barbil', 'Berhampur', 'Bhilai',
       'Bhubaneshwar', 'Bodhgaya', 'Cuttack', 'Darjeeling', 'Dibrugarh',
       'Digha', 'Dooars', 'Durgapur', 'Gangtok', 'Gaya', 'Gorakhpur',
       'Guwahati', 'Imphal', 'Jamshedpur', 'Jorhat', 'Kalimpong',
       'Kanpur', 'Kaziranga', 'Kolkata', 'Kurseong', 'Lachung',
       'Mandormoni', 'Patna', 'Pelling', 'Puri', 'Raichak', 'Rajgir',
       'Ranchi', 'Ravangla', 'Rishyap', 'Rourkela', 'Shillong',
       'Shimlipal', 'Siliguri', 'Sunderban', 'Tarapith', 'Yuksom'],
      dtype=object)
```

In [148]:
```
df[df['Category']=='SOUTH INDIA']['CITY'].values
```

Out[148]:
```
array(['Alleppey', 'Ashtamudi', 'Bandipur', 'Bangalore', 'Belgaum',
       'Calicut', 'Canannore', 'Chennai', 'Chikmagalur', 'Coimbatore',
       'Coonoor', 'Coorg', 'Dandeli', 'Gokharna', 'Guruvayoor', 'Halebid',
       'Hampi', 'Hassan', 'Hospet', 'Hosur', 'Hubli', 'Hyderabad',
       'Idukki', 'Kabini', 'Kanchipuram', 'Kanyakumari', 'Karur',
       'Karwar', 'Kasargod', 'Kochin', 'Kodaikanal', 'Kollam', 'Kotagiri',
       'Kottayam', 'Kovalam', 'Kumarakom', 'Kumbakonam', 'Kumily',
       'Lakshadweep', 'Madurai', 'Mahabalipuram', 'Malappuram', 'Malpe',
       'Mararri', 'Mangalore', 'Munnar', 'Mysore', 'Nadukani',
       'Nagapattinam', 'Nagarhole', 'Nilgiri', 'Ooty', 'Pallakad',
       'Pondicherry', 'Poovar', 'Port Blair', 'Puttaparthi',
       'Rajahmundry', 'Rameshwaram', 'Ranny', 'Salem', 'Secunderabad',
       'Sharavanbelgola', 'Shivanasamudra', 'Sivaganga District',
       'Tanjore', 'Thekkady', 'Thirvannamalai', 'Thiruvananthapuram',
       'Tiruchirapalli', 'Tirupur', 'Tirupati', 'Thrissur', 'Udupi',
       'Vagamon', 'Varkala', 'Velankanni', 'Vellore', 'Vijayawada',
       'Vishakapatnam', 'Wayanad', 'Yercaud'], dtype=object)
```

In [149]:
```
df[df['Category']=='WEST INDIA']['CITY'].values
```

Out[149]:
```
array(['Ahmedabad', 'Ajmer', 'Alibaug', 'Alsisar', 'Alwar', 'Anand',
       'Ankleshwar', 'Aurangabad', 'Balasinor', 'Bambora', 'Behror',
       'Bharatpur', 'Bhandardara', 'Bharuch', 'Bhavangadh', 'Bhavnagar',
       'Bhuj', 'Bikaner', 'Bundi', 'Chiplun', 'Chittorgarh', 'Dabhosa',
       'Daman', 'Dapoli', 'Dausa', 'Diu', 'Dive Agar', 'Durshet',
       'Dwarka', 'Ganapatipule', 'Gandhidham', 'Gandhinagar', 'Goa',
       'Gondal', 'Igatpuri', 'Jaipur', 'Jaisalmer', 'Jalgaon',
       'Jambugodha', 'Jamnagar', 'Jawhar', 'Jodhpur', 'Jojawar',
       'Junagadh', 'Karjat', 'Kashid', 'Khandala', 'Khimsar', 'Kolhapur',
       'Kota', 'Kumbalgarh', 'Lonavala', 'Lothal', 'Mahabaleshwar',
       'Malshej Ghat', 'Malvan', 'Mandavi', 'Mandawa', 'Manmad',
       'Matheran', 'Mount Abu', 'Morbi', 'Mumbai', 'Mundra',
       'Murud Janjira', 'Nagaur Fort', 'Nagothane', 'Nagpur', 'Nanded',
       'Napne', 'Nasik', 'Navi Mumbai', 'Neral', 'Osian', 'Palanpur',
       'Pali', 'Palitana', 'Panchgani', 'Panhala', 'Panvel', 'Pench',
       'Phalodi', 'Porbandar', 'Poshina', 'Pune', 'Puskhar', 'Rajasthan',
       'Rajkot', 'Rajpipla', 'Rajsamand', 'Ramgarh', 'Ranakpur',
       'Ranthambore', 'Ratnagiri', 'Rohetgarh', 'Sajan', 'Saputara',
       'Sasan Gir', 'Sawai Madhopur', 'Sawantwadi', 'Shirdi', 'Siana',
       'Silvassa', 'Surat', 'Tapola', 'Thane', 'Udaipur', 'Vadodara',
       'Vapi', 'Veraval', 'Vikramgadh', 'Wankaner'], dtype=object)
```

In [150]:
```
df[df['Category']=='CENTRAL INDIA']['CITY'].values
```

Out[150]:
```
array(['Amla', 'Bandhavgarh', 'Bhopal', 'Chitrakoot', 'Gwalior', 'Indore',
       'Jabalpur', 'Kanha', 'Khajuraho', 'Orchha', 'Pachmarhi', 'Panna',
       'Raipur', 'Ujjain'], dtype=object)
```

In [151]:
```
northern=['Agra', 'Allahabad', 'Almora', 'Ambala', 'Amritsar', 'Auli',
       'Baddi', 'Badrinath', 'Balrampur', 'Bareilly', 'Betalghat',
       'Bhimtal', 'Binsar', 'Chail', 'Chamba', 'Chandigarh',
       'Corbett National Park', 'Dalhousie', 'Dehradun', 'Dharamshala',
       'Faridabad', 'Firozabad', 'Gangotri', 'Garhmukteshwar', 'Garhwal',
       'Ghaziabad', 'Greater Noida', 'Gulmarg', 'Gurgaon', 'Hansi',
       'Haridwar', 'Jalandhar', 'Jammu', 'Jhansi', 'Kanatal', 'Kargil',
       'Karnal', 'Kasauli', 'Kashipur', 'Katra', 'Kausani', 'Kaza',
       'Kedarnath', 'Khajjiar', 'Kufri', 'Kullu', 'Kushinagar', 'Leh',
       'Lucknow', 'Ludhiana', 'Manali', 'Manesar', 'Marchula', 'Mathura',
       'Mcleodganj', 'Mohali', 'Moradabad', 'Mukteshwar', 'Mussoorie',
       'Nahan', 'Nainital', 'Naldhera', 'New Delhi', 'Noida', 'Palampur',
       'Pahalgam', 'Panchkula', 'Pantnagar', 'Parwanoo', 'Patiala',
       'Pathankot', 'Patnitop', 'Phagwara', 'Pinjore', 'Pragpur',
       'Rai Bareilly', 'Ram Nagar', 'Ranikhet', 'Rishikesh', 'Sattal',
       'Shimla', 'Solan', 'Sonauli', 'Srinagar', 'Udhampur', 'Uttarkashi',
       'Varanasi', 'Yamunotri','Zirakpur','Nayagaon','Meerut']
eastern=['Agartala', 'Aizwal', 'Barbil', 'Berhampur', 'Bhilai',
       'Bhubaneshwar', 'Bodhgaya', 'Cuttack', 'Darjeeling', 'Dibrugarh',
       'Digha', 'Dooars', 'Durgapur', 'Gangtok', 'Gaya', 'Gorakhpur',
       'Guwahati', 'Imphal', 'Jamshedpur', 'Jorhat', 'Kalimpong',
       'Kanpur', 'Kaziranga', 'Kolkata', 'Kurseong', 'Lachung',
       'Mandormoni', 'Patna', 'Pelling', 'Puri', 'Raichak', 'Rajgir',
       'Ranchi', 'Ravangla', 'Rishyap', 'Rourkela', 'Shillong',
       'Shimlipal', 'Siliguri', 'Sunderban', 'Tarapith', 'Yuksom', 'Howrah',
'Kharagpur']
southern=['Alleppey', 'Ashtamudi', 'Bandipur', 'Bangalore', 'Belgaum',
       'Calicut', 'Canannore', 'Chennai', 'Chikmagalur', 'Coimbatore',
       'Coonoor', 'Coorg', 'Dandeli', 'Gokharna', 'Guruvayoor', 'Halebid',
       'Hampi', 'Hassan', 'Hospet', 'Hosur', 'Hubli', 'Hyderabad',
       'Idukki', 'Kabini', 'Kanchipuram', 'Kanyakumari', 'Karur',
       'Karwar', 'Kasargod', 'Kochin', 'Kodaikanal', 'Kollam', 'Kotagiri',
       'Kottayam', 'Kovalam', 'Kumarakom', 'Kumbakonam', 'Kumily',
       'Lakshadweep', 'Madurai', 'Mahabalipuram', 'Malappuram', 'Malpe',
       'Mararri', 'Mangalore', 'Munnar', 'Mysore', 'Nadukani',
       'Nagapattinam', 'Nagarhole', 'Nilgiri', 'Ooty', 'Pallakad',
       'Pondicherry', 'Poovar', 'Port Blair', 'Puttaparthi',
       'Rajahmundry', 'Rameshwaram', 'Ranny', 'Salem', 'Secunderabad',
       'Sharavanbelgola', 'Shivanasamudra', 'Sivaganga District',
       'Tanjore', 'Thekkady', 'Thirvannamalai', 'Thiruvananthapuram',
       'Tiruchirapalli', 'Tirupur', 'Tirupati', 'Thrissur', 'Udupi',
       'Vagamon', 'Varkala', 'Velankanni', 'Vellore', 'Vijayawada',
       'Vishakapatnam', 'Wayanad', 'Yercaud','Alappuzha', 'Amravati', 'Guntur'
,
        'Kochi', 'Manipal', 'Palakkad', 'Puducherry', 'Trichy','Trivandrum',
'Vizag']
western=['Ahmedabad', 'Ajmer', 'Alibaug', 'Alsisar', 'Alwar', 'Anand',
       'Ankleshwar', 'Aurangabad', 'Balasinor', 'Bambora', 'Behror',
       'Bharatpur', 'Bhandardara', 'Bharuch', 'Bhavangadh', 'Bhavnagar',
       'Bhuj', 'Bikaner', 'Bundi', 'Chiplun', 'Chittorgarh', 'Dabhosa',
       'Daman', 'Dapoli', 'Dausa', 'Diu', 'Dive Agar', 'Durshet',
       'Dwarka', 'Ganapatipule', 'Gandhidham', 'Gandhinagar', 'Goa',
       'Gondal', 'Igatpuri', 'Jaipur', 'Jaisalmer', 'Jalgaon',
       'Jambugodha', 'Jamnagar', 'Jawhar', 'Jodhpur', 'Jojawar',
       'Junagadh', 'Karjat', 'Kashid', 'Khandala', 'Khimsar', 'Kolhapur',
       'Kota', 'Kumbalgarh', 'Lonavala', 'Lothal', 'Mahabaleshwar',
```

```
       'Malshej Ghat', 'Malvan', 'Mandavi', 'Mandawa', 'Manmad',
       'Matheran', 'Mount Abu', 'Morbi', 'Mumbai', 'Mundra',
       'Murud Janjira', 'Nagaur Fort', 'Nagothane', 'Nagpur', 'Nanded',
       'Napne', 'Nasik', 'Navi Mumbai', 'Neral', 'Osian', 'Palanpur',
       'Pali', 'Palitana', 'Panchgani', 'Panhala', 'Panvel', 'Pench',
       'Phalodi', 'Porbandar', 'Poshina', 'Pune', 'Puskhar', 'Rajasthan',
       'Rajkot', 'Rajpipla', 'Rajsamand', 'Ramgarh', 'Ranakpur',
       'Ranthambore', 'Ratnagiri', 'Rohetgarh', 'Sajan', 'Saputara',
       'Sasan Gir', 'Sawai Madhopur', 'Sawantwadi', 'Shirdi', 'Siana',
       'Silvassa', 'Surat', 'Tapola', 'Thane', 'Udaipur', 'Vadodara',
       'Vapi', 'Veraval', 'Vikramgadh', 'Wankaner','Nashik','Neemrana','Pushka
r']
central=['Amla', 'Bandhavgarh', 'Bhopal', 'Chitrakoot', 'Gwalior', 'Indore',
       'Jabalpur', 'Kanha', 'Khajuraho', 'Orchha', 'Pachmarhi', 'Panna',
       'Raipur', 'Ujjain']
def region(x):
    #northern=['Delhi','Jaipur','Lucknow','Kanpur','Ghaziabad','Ludhiana','Agr
a','Allahabad','Faridabad','Meerut','Varanasi','Srinagar','Amritsar','Jodhpu
r','Chandigarh','Kota','Bareily','Moradabad','Gurgaon','Aligarh','Jalandha
r','Saharanpur','Gorakhpur','Bikaner','Noida','Firozabad','Dehradun','Ajme
r','Lonni','Jhansi','Jammu']
    if x in northern:
        return 'northern'
    elif x in eastern:
        return 'eastern'
    elif x in southern:
        return 'southern'
    elif x in western:
        return 'western'
    elif x in central:
        return 'central'
    else:
        return np.nan
df_restaurants_copy['region'] = np.nan
df_restaurants_copy['region'].head()
```

Out[151]:
```
0   NaN
1   NaN
2   NaN
3   NaN
4   NaN
Name: region, dtype: float64
```

In [152]:
```
df_restaurants_copy['city'].isna().sum()
```

Out[152]:
```
0
```

In [153]:
```
df_restaurants_copy['region'] = df_restaurants_copy['city'].transform(lambda x
: region(x))
```

In [154]:
```
df_restaurants_copy['region'].unique()
```

Out[154]:
```
array(['northern', 'western', 'southern', 'central', 'eastern'],
      dtype=object)
```

In [155]:
```
df_restaurants_copy[df_restaurants_copy['region'].isna()]['city'].unique()
```

Out[155]:
```
array([], dtype=object)
```

In [156]:
```
print('Let\'s add these leftover cities manually to their respective lists')
```

```
Let's add these leftover cities manually to their respective lists
```

In [157]:
```
df_restaurants_copy['region'].unique()
```

Out[157]:
```
array(['northern', 'western', 'southern', 'central', 'eastern'],
      dtype=object)
```

In [158]:
```
df_restaurants_copy.groupby('region')[['region','city']].head(2)
```

Out[158]:

	region	city
0	northern	Agra
1	northern	Agra
2622	western	Ahmedabad
2623	western	Ahmedabad
9213	southern	Alappuzha
9214	southern	Alappuzha
24601	central	Bhopal
24602	central	Bhopal
27257	eastern	Bhubaneshwar
27258	eastern	Bhubaneshwar

In [159]:
```
df_restaurants_copy.groupby('region')['city'].first()
```

Out[159]:
```
region
central           Bhopal
eastern     Bhubaneshwar
northern            Agra
southern       Alappuzha
western        Ahmedabad
Name: city, dtype: object
```

10. Some more Analysis

Lets us explore the data some more now that we have extrapolated and removed the missing values
We now conduct analysis to compare the regions.

1. To find which cities have expensive restaurants

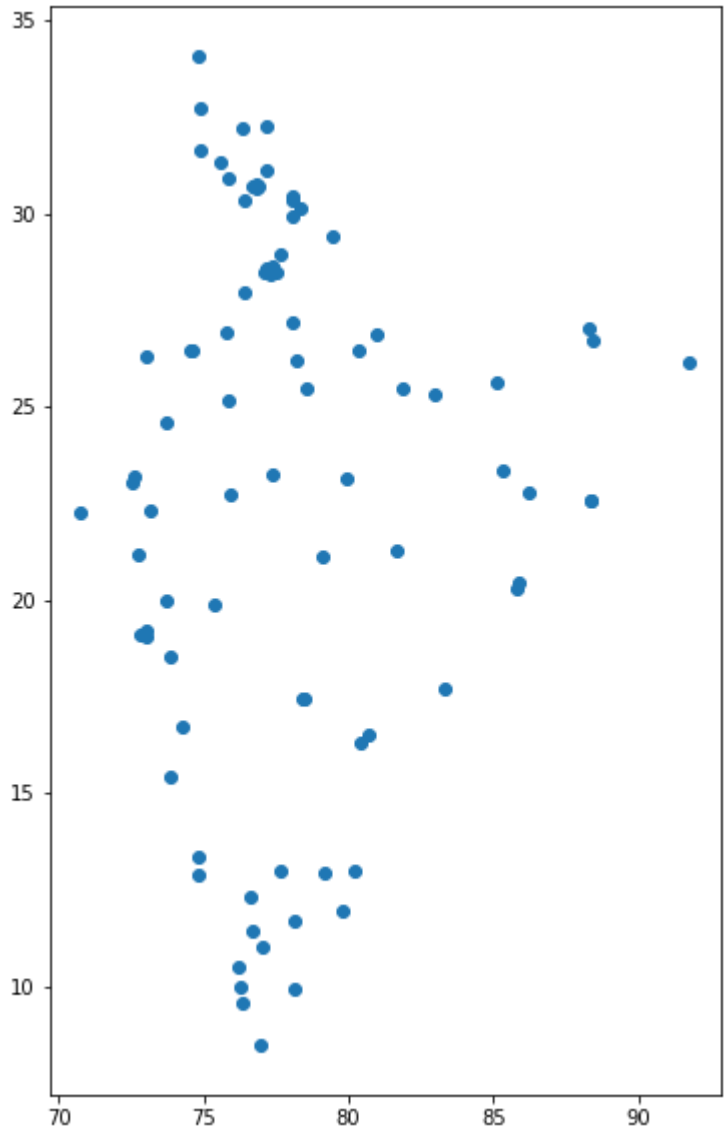
In [160]:
```
#METHOD 1: Based on average 'average cost for two' of all restaurants per city
for cities which have expensive restaurants
def detect_res(x, y):
    Q1 = df_restaurants_copy[x].quantile(0.25)
    Q3 = df_restaurants_copy[x].quantile(0.75)
    IQR = Q3 - Q1
    if y==1:
        return df_restaurants_copy[df_restaurants_copy[x] > (Q3 + 1.5 * IQR)]
[['city','latitude','longitude','average_cost_for_two']].drop_duplicates(keep=
"first")
    else:
        return df_restaurants_copy[df_restaurants_copy[x] > (Q3 + 1.5 * IQR)]
[['city','latitude','longitude','average_cost_for_two']].groupby(['city']).mea
n().sort_values(by="average_cost_for_two",ascending=False).reset_index().drop_
duplicates(keep="first").reset_index()
print("The cities which have expensive restaurants: \n", detect_res('average_c
ost_for_two',1)['city'].unique())
print(len(detect_res('average_cost_for_two',1)['city'].unique())," out of ", l
en(df_restaurants_copy['city'].unique())," cities have expensive restaurants")
#detect_res('average_cost_for_two',2)
```

```
The cities which have expensive restaurants:
 ['Agra' 'Ahmedabad' 'Gandhinagar' 'Ajmer' 'Alappuzha' 'Allahabad'
 'Amritsar' 'Aurangabad' 'Bangalore' 'Bhopal' 'Bhubaneshwar' 'Chandigarh'
 'Panchkula' 'Mohali' 'Zirakpur' 'Nayagaon' 'Chennai' 'Coimbatore'
 'Cuttack' 'Darjeeling' 'Dehradun' 'New Delhi' 'Gurgaon' 'Noida'
 'Ghaziabad' 'Faridabad' 'Greater Noida' 'Dharamshala' 'Gangtok' 'Goa'
 'Gorakhpur' 'Guntur' 'Guwahati' 'Gwalior' 'Haridwar' 'Hyderabad'
 'Secunderabad' 'Indore' 'Jabalpur' 'Jaipur' 'Jalandhar' 'Jammu'
 'Jamshedpur' 'Jhansi' 'Jodhpur' 'Kanpur' 'Kochi' 'Kolhapur' 'Kolkata'
 'Howrah' 'Kota' 'Lucknow' 'Ludhiana' 'Madurai' 'Manali' 'Mangalore'
 'Manipal' 'Meerut' 'Mumbai' 'Thane' 'Navi Mumbai' 'Mussoorie' 'Mysore'
 'Nagpur' 'Nainital' 'Nashik' 'Neemrana' 'Ooty' 'Patiala' 'Patna'
 'Puducherry' 'Pune' 'Pushkar' 'Raipur' 'Rajkot' 'Ranchi' 'Rishikesh'
 'Salem' 'Shimla' 'Siliguri' 'Srinagar' 'Surat' 'Thrissur' 'Trichy'
 'Trivandrum' 'Udaipur' 'Varanasi' 'Vellore' 'Vijayawada' 'Vizag'
 'Vadodara']
91  out of  98  cities have expensive restaurants
```

- plot the cities which have costliest restaurants.

In [161]:
```
fig, ax = plt.subplots(figsize=(6,10))
plt.scatter(detect_res('average_cost_for_two',2)['longitude'],detect_res('aver
age_cost_for_two',2)['latitude'])
```

Out[161]: <matplotlib.collections.PathCollection at 0x3a7634cd48>



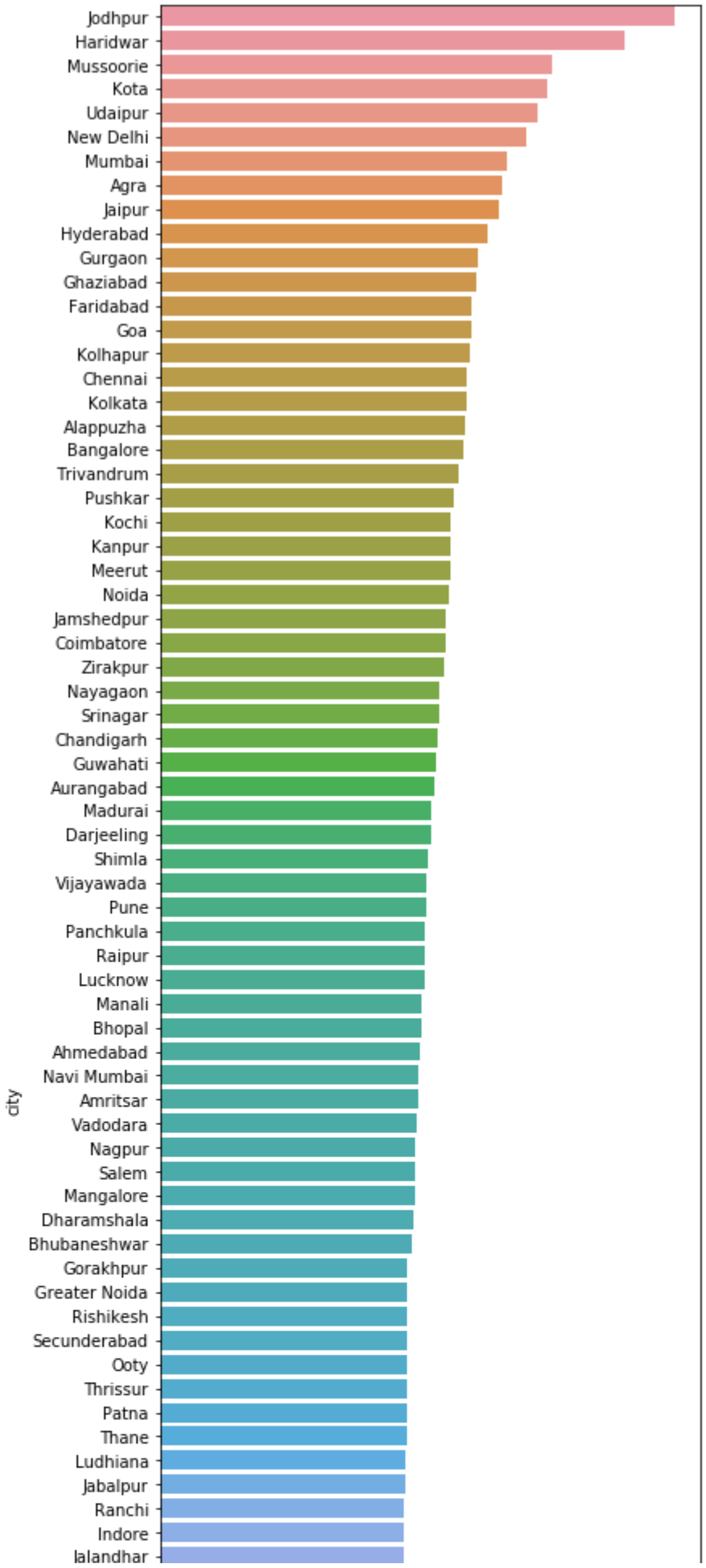
In [162]:
```
detect_res('average_cost_for_two',2).head(5)
```

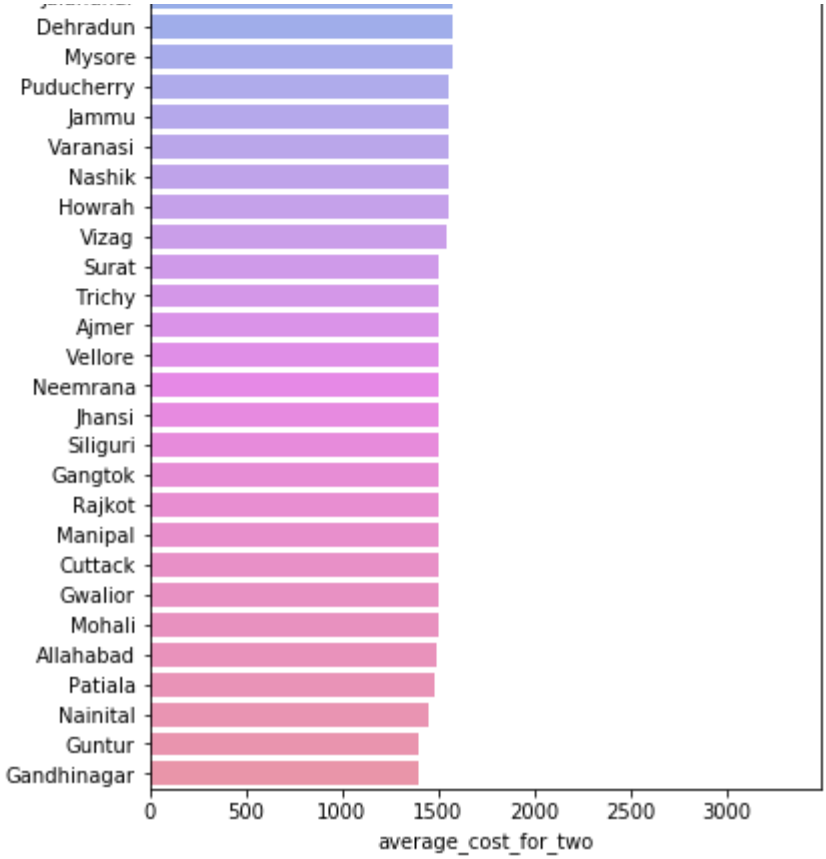
Out[162]:

	index	city	latitude	longitude	average_cost_for_two
0	0	Jodhpur	26.277255	73.039353	3320.000000
1	1	Haridwar	29.948136	78.075657	3000.000000
2	2	Mussoorie	30.467409	78.060416	2530.000000
3	3	Kota	25.138712	75.853320	2500.000000
4	4	Udaipur	24.583215	73.678040	2444.776119

In [163]:
```
fig, ax = plt.subplots(figsize=(6,25))
#plt.xticks(rotation=90)
sns.barplot(y=detect_res('average_cost_for_two',2)['city'], x=detect_res('aver
age_cost_for_two',2)['average_cost_for_two'])
```

Out[163]: <matplotlib.axes._subplots.AxesSubplot at 0x3a76359748>


Jodhpur
Haridwar
Mussoorie
Kota
Udaipur
New Delhi
Mumbai
Agra
Jaipur
Hyderabad
Gurgaon
Ghaziabad
Faridabad
Goa
Kolhapur
Chennai
Kolkata
Alappuzha
Bangalore
Trivandrum
Pushkar
Kochi
Kanpur
Meerut
Noida
Jamshedpur
Coimbatore
Zirakpur
Nayagaon
Srinagar
Chandigarh
Guwahati
Aurangabad
Madurai
Darjeeling
Shimla
Vijayawada
Pune
Panchkula
Raipur
Lucknow
Manali
Bhopal
Ahmedabad
Navi Mumbai
Amritsar
Vadodara
Nagpur
Salem
Mangalore
Dharamshala
Bhubaneshwar
Gorakhpur
Greater Noida
Rishikesh
Secunderabad
Ooty
Thrissur
Patna
Thane
Ludhiana
Jabalpur
Ranchi
Indore
Jalandhar
city

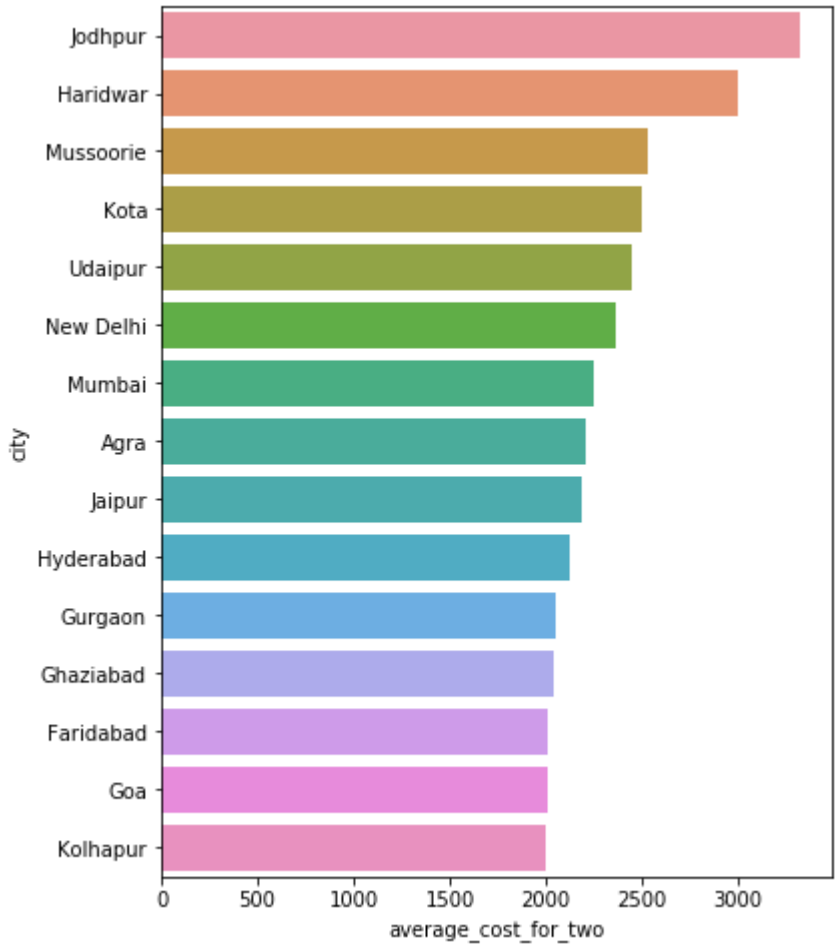


In [164]:
```
detect_res('average_cost_for_two',2)[detect_res('average_cost_for_two',2)['average_cost_for_two']>=2000]
```

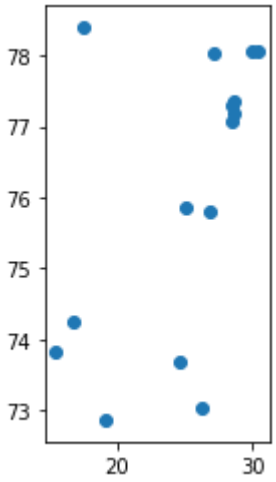
Out[164]:

	index	city	latitude	longitude	average_cost_for_two
0	0	Jodhpur	26.277255	73.039353	3320.000000
1	1	Haridwar	29.948136	78.075657	3000.000000
2	2	Mussoorie	30.467409	78.060416	2530.000000
3	3	Kota	25.138712	75.853320	2500.000000
4	4	Udaipur	24.583215	73.678040	2444.776119
5	5	New Delhi	28.595442	77.188452	2362.547529
6	6	Mumbai	19.076088	72.848478	2244.340723
7	7	Agra	27.163413	78.044201	2209.459459
8	8	Jaipur	26.901705	75.809662	2186.864407
9	9	Hyderabad	17.428090	78.409726	2118.311475
10	10	Gurgaon	28.467950	77.078377	2050.492611
11	11	Ghaziabad	28.646487	77.367524	2040.000000
12	12	Faridabad	28.426943	77.308788	2013.333333
13	13	Goa	15.436314	73.828536	2009.210526
14	14	Kolhapur	16.704828	74.248874	2000.000000

In [165]:
```
fig, ax = plt.subplots(figsize=(6,8))
ax = sns.barplot(y=detect_res('average_cost_for_two',2)[detect_res('average_cost_for_two',2)['average_cost_for_two']>=2000]['city'], x=detect_res('average_cost_for_two',2)[detect_res('average_cost_for_two',2)['average_cost_for_two']>=2000]['average_cost_for_two'])
```



In [166]:
```
fig, ax = plt.subplots(figsize=(2,4))
ax = plt.scatter(y=detect_res('average_cost_for_two',2)[detect_res('average_cost_for_two',2)['average_cost_for_two']>=2000]['longitude'], x=detect_res('average_cost_for_two',2)[detect_res('average_cost_for_two',2)['average_cost_for_two']>=2000]['latitude'])
```



In [167]:
```
#Method 2: Based on cities having atleast 1 expensive restaurant:
df_restaurants_copy[df_restaurants_copy['average_cost_for_two']>=6000]['city'].unique()
```

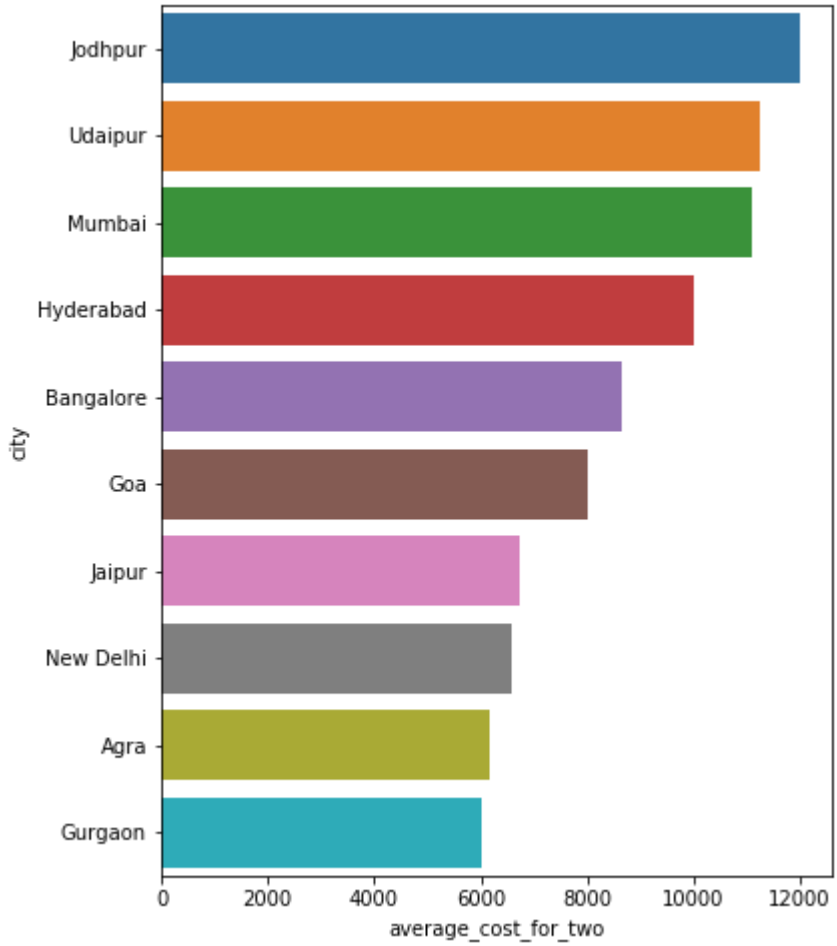
Out[167]:
```
array(['Agra', 'Bangalore', 'New Delhi', 'Gurgaon', 'Goa', 'Hyderabad',
       'Jaipur', 'Jodhpur', 'Mumbai', 'Udaipur'], dtype=object)
```

In [168]:
```
df1 = df_restaurants_copy[df_restaurants_copy['average_cost_for_two']>=6000].drop_duplicates().groupby('city')[['latitude','longitude','average_cost_for_two']].mean().sort_values(by="average_cost_for_two", ascending=False).reset_index()
df1
```

Out[168]:

	city	latitude	longitude	average_cost_for_two
0	Jodhpur	26.281131	73.047052	12000.000000
1	Udaipur	24.575562	73.679719	11250.000000
2	Mumbai	19.020338	72.845137	11083.333333
3	Hyderabad	17.334311	78.467578	10000.000000
4	Bangalore	13.000593	77.614757	8666.666667
5	Goa	15.500437	73.831510	8000.000000
6	Jaipur	26.954576	75.841204	6750.000000
7	New Delhi	28.615206	77.174571	6585.714286
8	Agra	27.165738	78.047178	6166.666667
9	Gurgaon	28.501355	77.086517	6000.000000

In [169]:
```
fig, ax = plt.subplots(figsize=(6,8))
ax = sns.barplot(x=df1['average_cost_for_two'],y=df1['city'])
```

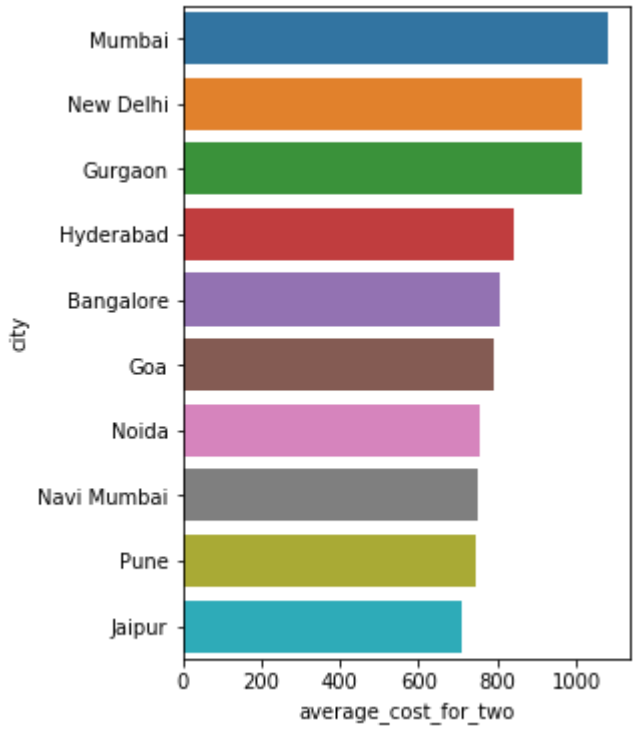


In [170]:
```
#METHOD 3: MEAN AVERAGE COST FOR TWO INCLUDING BOTH EXPENSIVE AND NON-EXPENSIV
E RESTAURANTS
df2 = df_restaurants_copy.drop_duplicates().groupby('city')[['latitude','longi
tude','average_cost_for_two']].mean().sort_values(by="average_cost_for_two", a
scending=False).head(10).reset_index()
df2
```

Out[170]:

	city	latitude	longitude	average_cost_for_two
0	Mumbai	19.095666	72.850181	1082.211926
1	New Delhi	28.609399	77.183393	1017.742111
2	Gurgaon	28.462836	77.073018	1013.654434
3	Hyderabad	17.429485	78.423441	844.358053
4	Bangalore	12.959994	77.625171	807.254151
5	Goa	15.461133	73.832660	793.286957
6	Noida	28.569759	77.350580	755.257353
7	Navi Mumbai	19.057127	73.026306	750.237154
8	Pune	18.549211	73.833547	745.676175
9	Jaipur	26.894610	75.795383	711.301939

In [171]:
```
fig, ax = plt.subplots(figsize=(4,6))
ax = sns.barplot(x=df2['average_cost_for_two'],y=df2['city'])
```



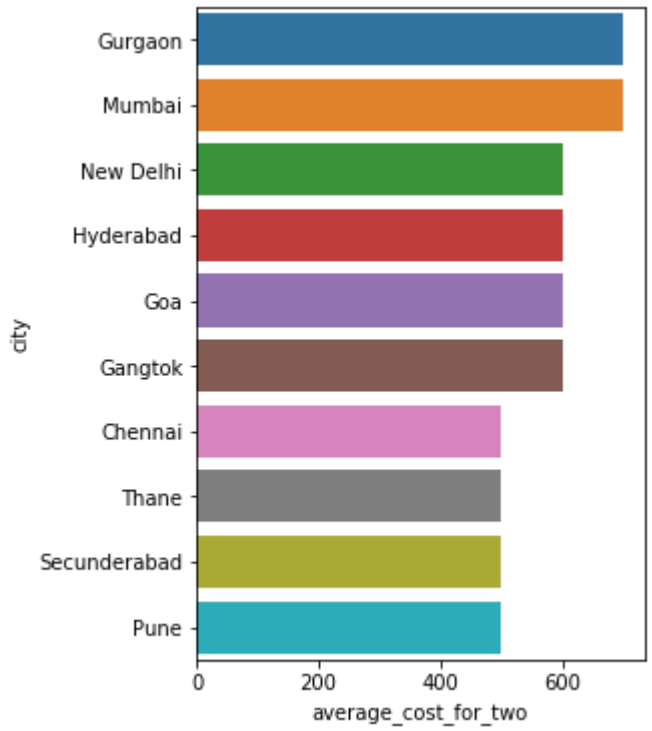
In [172]:
```
#METHOD 4: MEDIAN AVERAGE COST FOR TWO INCLUDING BOTH EXPENSIVE AND NON-EXPENS
IVE RESTAURANTS
df2 = df_restaurants_copy.drop_duplicates().groupby('city')[['latitude','longi
tude','average_cost_for_two']].median().sort_values(by="average_cost_for_two",
ascending=False).head(10).reset_index()
df2
```

Out[172]:

	city	latitude	longitude	average_cost_for_two
0	Gurgaon	28.467511	77.080265	700
1	Mumbai	19.103313	72.837480	700
2	New Delhi	28.621053	77.199276	600
3	Hyderabad	17.431567	78.412096	600
4	Goa	15.499626	73.816727	600
5	Gangtok	27.327332	88.612074	600
6	Chennai	13.037221	80.231379	500
7	Thane	19.209854	72.973951	500
8	Secunderabad	17.443386	78.516707	500
9	Pune	18.543631	73.840957	500

In [173]:
```
print('METHOD 4')
fig, ax = plt.subplots(figsize=(4,6))
ax = sns.barplot(x=df2['average_cost_for_two'],y=df2['city'])
```

```
METHOD 4
```



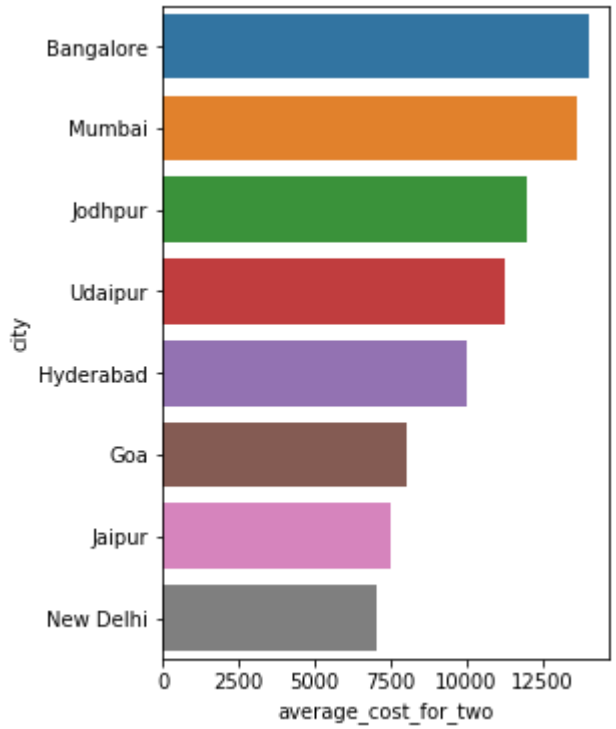
In [174]:
```
#METHOD 5: MEAN AVERAGE COST FOR TWO INCLUDING EXPENSIVE RESTAURANTS ONLY
df2 = df_restaurants_copy.drop_duplicates().sort_values(by="average_cost_for_two", ascending=False).head(20).groupby('city')[['latitude','longitude','average_cost_for_two']].mean().sort_values(by="average_cost_for_two", ascending=False).reset_index()
df2
```

Out[174]:

	city	latitude	longitude	average_cost_for_two
0	Bangalore	13.047513	77.610328	14000
1	Mumbai	18.983401	72.836365	13625
2	Jodhpur	26.281131	73.047052	12000
3	Udaipur	24.575562	73.679719	11250
4	Hyderabad	17.334311	78.467578	10000
5	Goa	15.500437	73.831510	8000
6	Jaipur	26.991565	75.868789	7500
7	New Delhi	28.597212	77.172921	7025

In [175]:
```
print('METHOD 5')
fig, ax = plt.subplots(figsize=(4,6))
ax = sns.barplot(x=df2['average_cost_for_two'],y=df2['city'])
```

```
METHOD 5
```



2. Comparing regions

2a. Highlights available in restaurants for different regions

To cater our analysis we define the regions as nothern, eastern, western and southern.

We first need to select the unique facilities available in each region and sort according to their frequencies.

In [176]:
```
def highlights_sort(x):
    x=x.dropna()
    x=np.asarray(x.transform(lambda x: x.split(", ")).to_numpy())
    x= pd.Series(np.concatenate(x, axis=0))
    z = x.value_counts().reset_index()
    z = z.rename(columns={'index': 'highlights', 0: 'frequency'})
    return z
```

Highlights of the northern region

In [177]:
```
print(highlights_sort(df_restaurants_copy[df_restaurants_copy['region'] == "no
rthern"]['highlights']))
```

```
            highlights  frequency
0                 Cash      14080
1   Takeaway Available      12702
2               Dinner      10765
3       Indoor Seating      10747
4                Lunch      10565
..                 ...        ...
90     Unlimited Pizza          3
91        Members Only          2
92             Gin Bar          2
93        Wine Tasting          1
94   Couple Entry Only          1

[95 rows x 2 columns]
```

Highlights of the eastern region

In [178]:
```
print(highlights_sort(df_restaurants_copy[df_restaurants_copy['region'] == "eastern"]['highlights']))
```

```
            highlights  frequency
0                 Cash       7261
1   Takeaway Available       6418
2       Indoor Seating       5421
3               Dinner       5061
4                Lunch       4914
..                 ...        ...
89        Dark Kitchen          1
90             Gin Bar          1
91        Keto Options          1
92     Alipay Accepted          1
93   Couple Entry Only          1

[94 rows x 2 columns]
```

Highlights of the southern region

In [179]:
```
print(highlights_sort(df_restaurants_copy[df_restaurants_copy['region'] == "southern"]['highlights']))
```

```
              highlights  frequency
0                   Cash      14971
1     Takeaway Available      13198
2         Indoor Seating      12712
3                 Dinner      10988
4                  Lunch      10583
..                   ...        ...
93          Wine Tasting          3
94                  BYOB          3
95  Celebrity Frequented          2
96          Members Only          1
97          Dark Kitchen          1

[98 rows x 2 columns]
```

Highlights of the western region

In [180]:
```
print(highlights_sort(df_restaurants_copy[df_restaurants_copy['region'] == "western"]['highlights']))
```

```
                highlights  frequency
0                     Cash      17230
1       Takeaway Available      14883
2           Indoor Seating      12997
3                   Dinner      12154
4                    Lunch      11345
..                     ...        ...
95           Bira 91 Beer          2
96                   BYOB          2
97           Dark Kitchen          1
98  Subscription Available         1
99   Subscription Required         1

[100 rows x 2 columns]
```

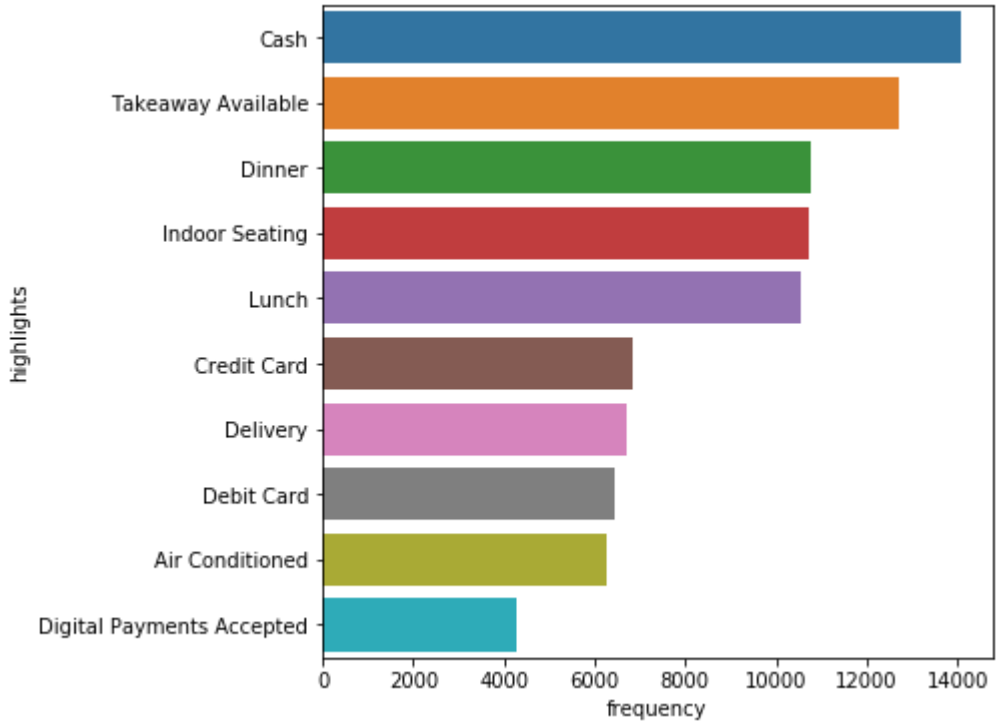
Plot the barplot for different regions

We shall now plot the graphs for top 10 highlights.

In [181]:
```
print('Northern: ')
fig, ax = plt.subplots(figsize=(6,6))
sns.barplot(y=highlights_sort(df_restaurants_copy[df_restaurants_copy['region'] == "northern"]['highlights'])['highlights'].head(10), x=highlights_sort(df_restaurants_copy[df_restaurants_copy['region'] == "northern"]['highlights'])['frequency'].head(10))
```

```
Northern:
```

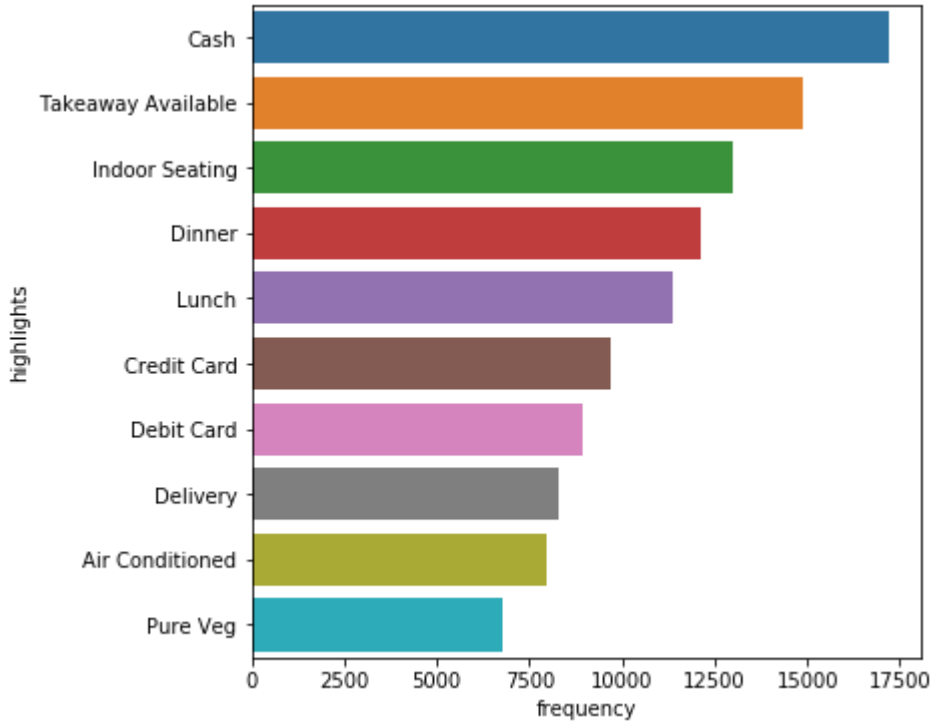
Out[181]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x3a75f90188>
```



In [182]:
```
print('Western:')
fig, ax = plt.subplots(figsize=(6,6))
sns.barplot(y=highlights_sort(df_restaurants_copy[df_restaurants_copy['region'] == "western"]['highlights'])['highlights'].head(10), x=highlights_sort(df_restaurants_copy[df_restaurants_copy['region'] == "western"]['highlights'])['frequency'].head(10))
```

```
Western:
```

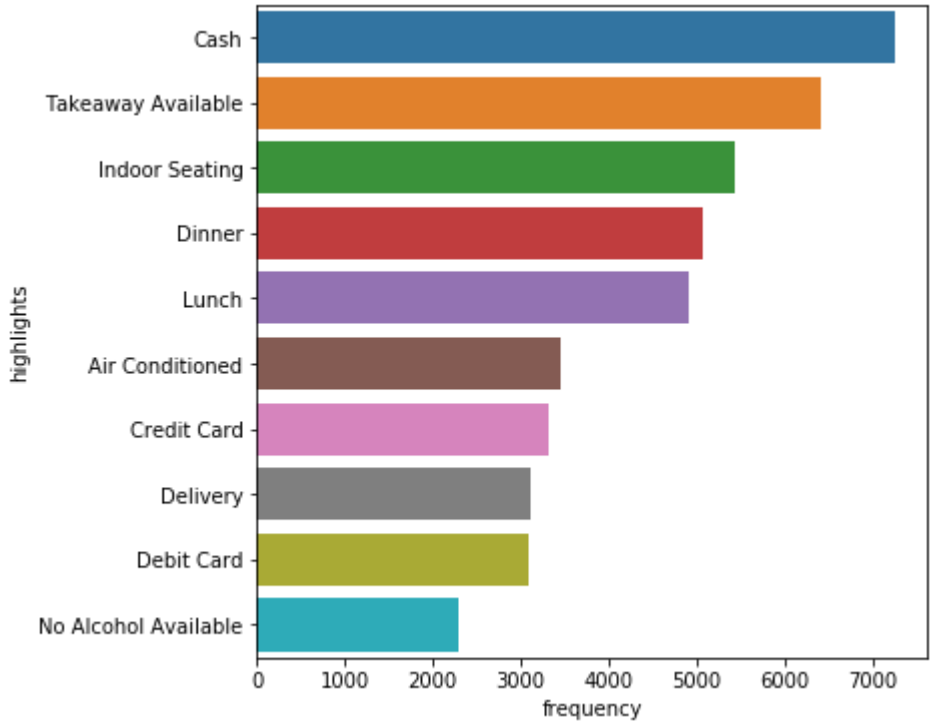
Out[182]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x3a7670f948>
```



In [183]:
```
print('Eastern: ')
fig, ax = plt.subplots(figsize=(6,6))
sns.barplot(y=highlights_sort(df_restaurants_copy[df_restaurants_copy['region'] == "eastern"]['highlights'])['highlights'].head(10), x=highlights_sort(df_restaurants_copy[df_restaurants_copy['region'] == "eastern"]['highlights'])['frequency'].head(10))
```

```
Eastern:
```

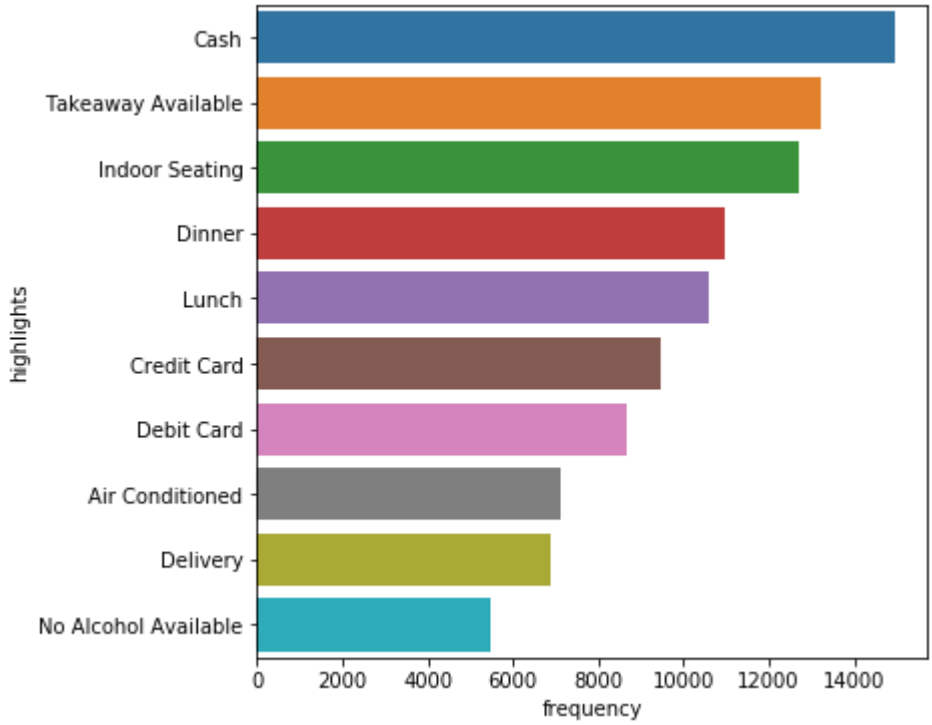
Out[183]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x3a749c98c8>
```



In [184]:
```
print('Southern: ')
fig, ax = plt.subplots(figsize=(6,6))
sns.barplot(y=highlights_sort(df_restaurants_copy[df_restaurants_copy['region'
] == "southern"]['highlights'])['highlights'].head(10), x=highlights_sort(df_r
estaurants_copy[df_restaurants_copy['region'] == "southern"]['highlights'])['f
requency'].head(10))
```

```
Southern:
```

Out[184]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x3a7631c188>
```



2b. Cuisines available in restaurants for different regions

In [185]:
```
def cuisines_freq2(x):
    x=x.dropna()
    x=np.asarray(x.transform(lambda x: x.split(", ")).to_numpy())
    x= pd.Series(np.concatenate(x, axis=0))
    z = x.value_counts().reset_index()
    z = z.rename(columns={'index': 'cuisines', 0: 'frequency'})
    return z
```

Cuisines in the northern region

In [186]:
```
print(cuisines_freq2(df_restaurants_copy[df_restaurants_copy['region'] == "northern"]['cuisines']))
```

```
         cuisines  frequency
0    North Indian       6444
1         Chinese       3584
2       Fast Food       3524
3     Continental       1550
4       Beverages       1547
..            ...        ...
108        Crepes          1
109       Swedish          1
110      Egyptian          1
111      Peruvian          1
112        Fusion          1

[113 rows x 2 columns]
```

Cuisines in the eastern region

In [187]:
```
print(cuisines_freq2(df_restaurants_copy[df_restaurants_copy['region'] == "eastern"]['cuisines']))
```

```
        cuisines  frequency
0   North Indian       2652
1        Chinese       2329
2      Fast Food       1601
3       Desserts        893
4         Bakery        736
..           ...        ...
94     Pan Asian          1
95       African          1
96       Russian          1
97      Oriental          1
98     Cafe Food          1

[99 rows x 2 columns]
```

Cuisines in the southern region

In [188]:
```
print(cuisines_freq2(df_restaurants_copy[df_restaurants_copy['region'] == "sou
thern"]['cuisines']))
```

```
           cuisines  frequency
0      North Indian       4444
1      South Indian       4048
2           Chinese       3587
3         Fast Food       2541
4         Beverages       2300
..              ...        ...
108  Coffee and Tea          1
109          Mishti          1
110       Pakistani          1
111            Naga          1
112           Irish          1

[113 rows x 2 columns]
```

Cuisines in the western region

In [189]:
```
print(cuisines_freq2(df_restaurants_copy[df_restaurants_copy['region'] == "wes
tern"]['cuisines']))
```

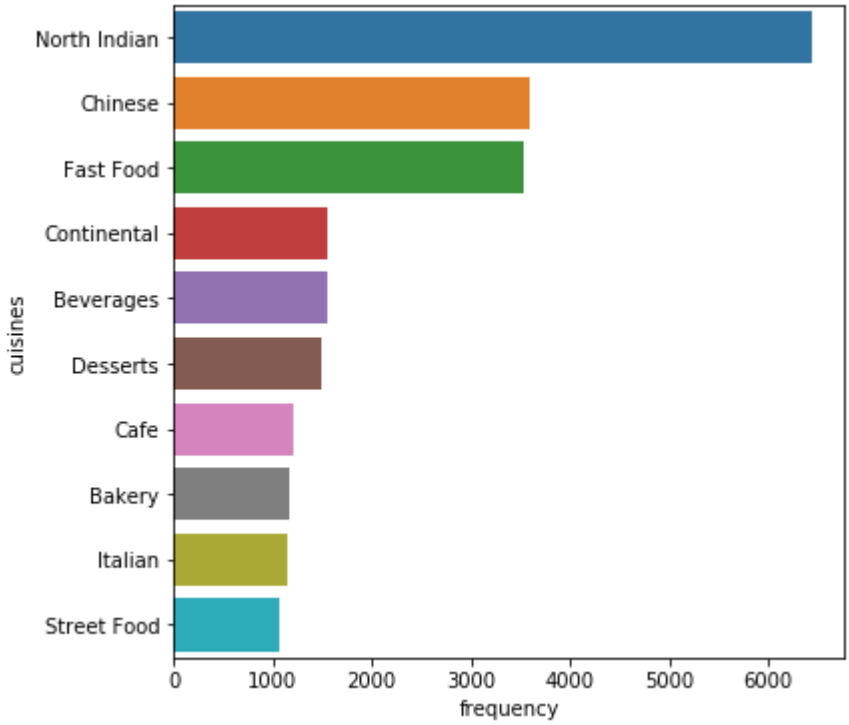
```
           cuisines  frequency
0     North Indian       6075
1        Fast Food       4349
2          Chinese       3698
3         Desserts       2646
4        Beverages       2456
..             ...        ...
105          Grill          1
106  Fried Chicken          1
107      Cafe Food          1
108     Sri Lankan          1
109          Bohri          1

[110 rows x 2 columns]
```

- Plot the barplot for top 10 cuisines served in the four regions

In [190]:
```
print('Northern: ')
fig, ax = plt.subplots(figsize=(6,6))
sns.barplot(y=cuisines_freq2(df_restaurants_copy[df_restaurants_copy['region']
== "northern"]['cuisines'])['cuisines'].head(10), x=cuisines_freq2(df_restaura
nts_copy[df_restaurants_copy['region'] == "northern"]['cuisines'])['frequency'
].head(10))
```

```
Northern:
```

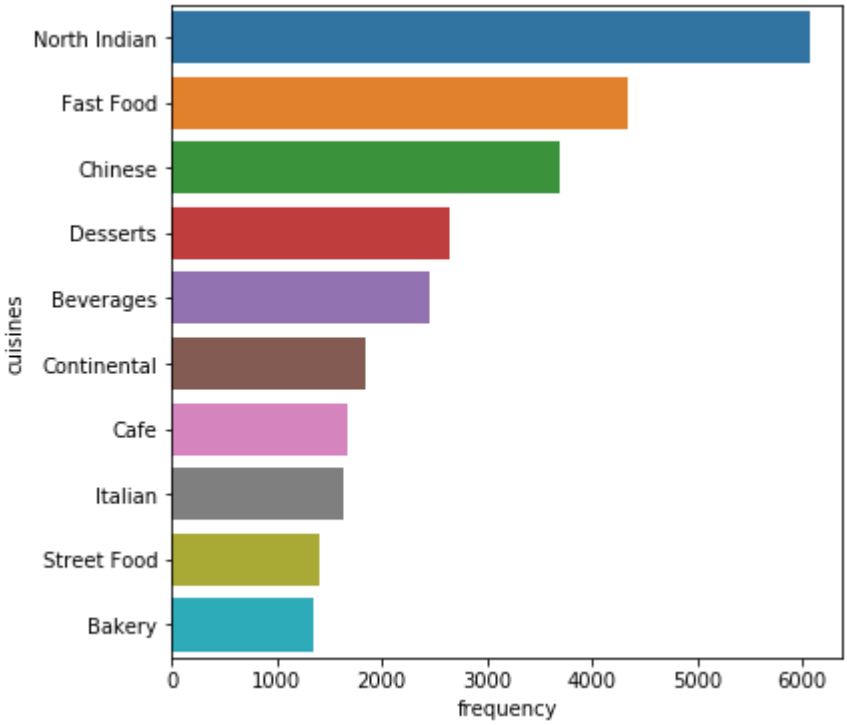
Out[190]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x3a783da7c8>
```



In [191]:
```
print('Western: ')
fig, ax = plt.subplots(figsize=(6,6))
sns.barplot(y=cuisines_freq2(df_restaurants_copy[df_restaurants_copy['region']
== "western"]['cuisines'])['cuisines'].head(10), x=cuisines_freq2(df_restauran
ts_copy[df_restaurants_copy['region'] == "western"]['cuisines'])['frequency'].
head(10))
```

```
Western:
```

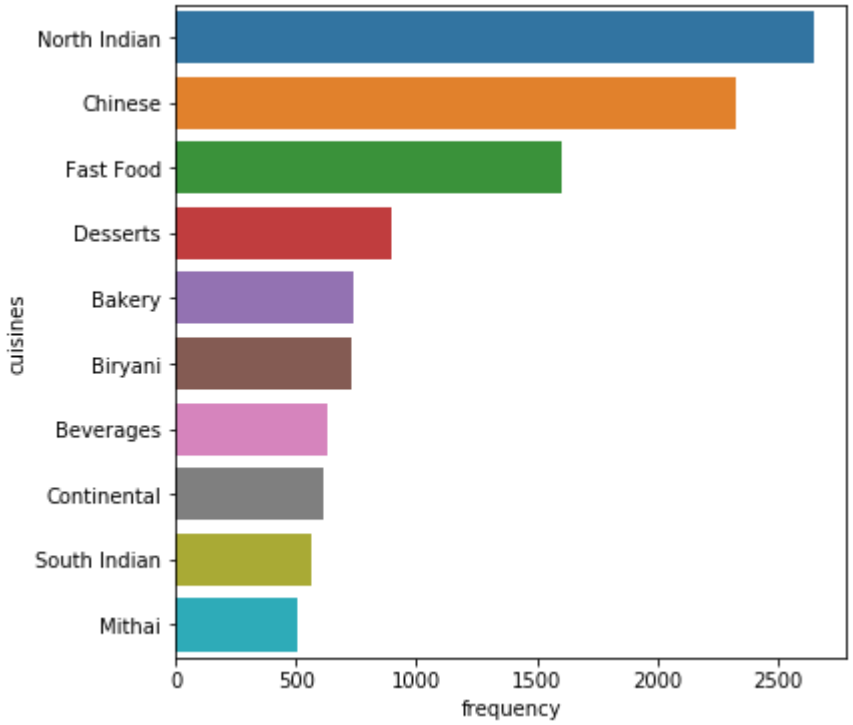
Out[191]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x3a760224c8>
```



In [192]:
```
print('Eastern: ')
fig, ax = plt.subplots(figsize=(6,6))
sns.barplot(y=cuisines_freq2(df_restaurants_copy[df_restaurants_copy['region']
== "eastern"]['cuisines'])['cuisines'].head(10), x=cuisines_freq2(df_restauran
ts_copy[df_restaurants_copy['region'] == "eastern"]['cuisines'])['frequency'].
head(10))
```

```
Eastern:
```

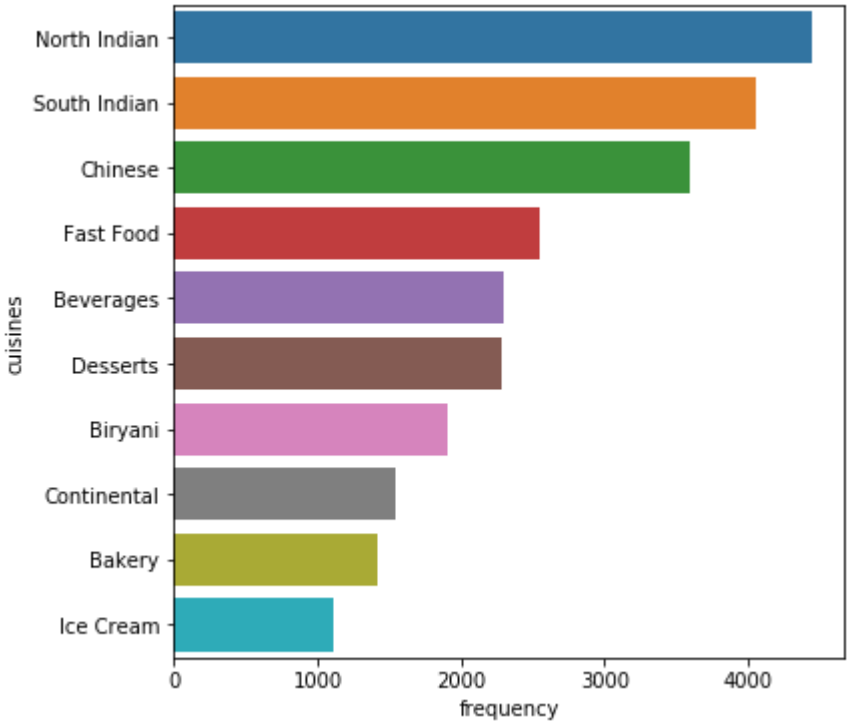
Out[192]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x3a7643f5c8>
```



In [193]:
```
print('Southern: ')
fig, ax = plt.subplots(figsize=(6,6))
sns.barplot(y=cuisines_freq2(df_restaurants_copy[df_restaurants_copy['region']
== "southern"]['cuisines'])['cuisines'].head(10), x=cuisines_freq2(df_restaura
nts_copy[df_restaurants_copy['region'] == "southern"]['cuisines'])['frequency'
].head(10))
```

```
Southern:
```

Out[193]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x3a76b79408>
```



3. The Northern Region

Now we shall consider only the northern region

1. The top 10 cuisines served in Restaurants

In [194]:
```
print(cuisines_freq2(df_restaurants_copy[df_restaurants_copy['region'] == "northern"]['cuisines']).head(10))
```

```
       cuisines  frequency
0  North Indian       6444
1       Chinese       3584
2     Fast Food       3524
3   Continental       1550
4     Beverages       1547
5      Desserts       1492
6          Cafe       1209
7        Bakery       1177
8       Italian       1139
9   Street Food       1062
```

2. Do restaurants with more photo counts and votes have better rating?

In [195]:
```
df_temp1 = df_restaurants_copy[df_restaurants_copy['region']=='northern'][["aggregate_rating","votes","photo_count"]].copy()
df_temp1.duplicated().sum()
```

Out[195]:
```
5546
```

In [196]:
```
df_temp1 = df_temp1.drop_duplicates()
df_temp1.isna().sum()
```

Out[196]:
```
aggregate_rating    0
votes               0
photo_count         0
dtype: int64
```

In [197]:
```
df_temp1.head()
```

Out[197]:

	aggregate_rating	votes	photo_count
0	4.4	814	154
1	4.4	1203	161
2	4.2	801	107
3	4.3	693	157
4	4.9	470	291

In [198]:
```
df_temp1.corr().iloc[1:,0]
```

Out[198]:
```
votes          0.352192
photo_count    0.276261
Name: aggregate_rating, dtype: float64
```

In [199]:
```
print('We need not always delete outliers. Without treating outliers, we see a very small positive correlation between "votes and aggregate_rating" and "photo_count and aggregate_rating".\nClearly, more votes and more photo_count result in less, though a positive impact on aggregate rating.\nSo the answer is, Very likely, yes! Maybe there is an indirect effect working here. Let\'s understand how this happens, below:')
```

```
We need not always delete outliers. Without treating outliers, we see a very
small positive correlation between "votes and aggregate_rating" and "photo_co
unt and aggregate_rating".
Clearly, more votes and more photo_count result in less, though a positive im
pact on aggregate rating.
So the answer is, Very likely, yes! Maybe there is an indirect effect working
here. Let's understand how this happens, below:
```

In [200]:
```
df_votes = df_temp1.groupby('aggregate_rating').sum().sort_values(by="votes",
ascending=False)['votes'].reset_index()
df_votes
```

Out[200]:

	aggregate_rating	votes
0	4.0	412666
1	4.1	391902
2	4.2	374957
3	4.3	325774
4	3.9	272988
5	4.4	232200
6	4.5	198063
7	3.8	161589
8	3.7	103207
9	4.6	95003
10	4.7	72327
11	3.6	67843
12	4.8	47621
13	4.9	45065
14	3.5	40695
15	3.4	28654
16	3.3	19344
17	3.2	10403
18	3.1	7764
19	2.8	7752
20	2.7	7296
21	3.0	5404
22	2.6	5295
23	2.4	4334
24	2.5	4049
25	2.9	4018
26	2.2	3028
27	2.3	2882
28	2.0	675
29	2.1	427
30	1.9	335
31	0.0	146

In [201]:
```
df_photo = df_temp1.groupby('aggregate_rating').sum().sort_values(by="photo_count", ascending=False)['photo_count'].reset_index()
df_photo
```

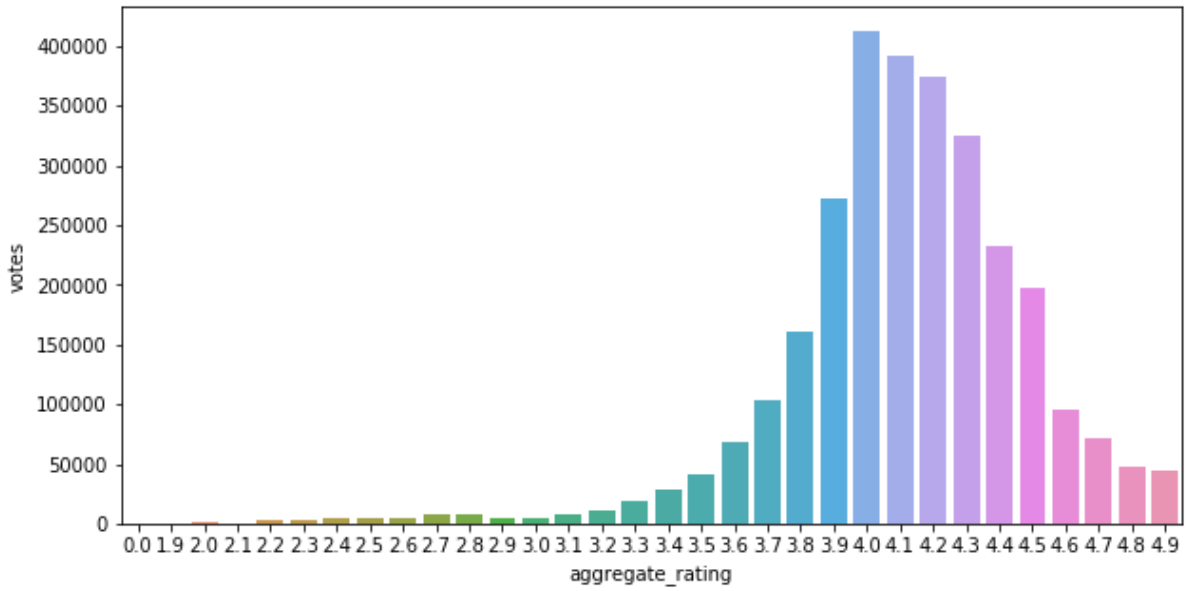
Out[201]:

	aggregate_rating	photo_count
0	4.2	405282
1	4.3	351221
2	4.1	319052
3	4.0	288244
4	4.4	229640
5	4.5	189542
6	3.9	149479
7	4.7	107506
8	4.6	97958
9	3.8	70916
10	3.7	42348
11	4.9	40911
12	4.8	34573
13	3.6	24806
14	3.5	15473
15	3.4	9951
16	3.3	7429
17	3.2	4664
18	3.1	2440
19	2.7	2241
20	2.8	2078
21	2.9	1853
22	2.4	1812
23	2.3	1716
24	0.0	1711
25	3.0	1582
26	2.6	1372
27	2.5	1312
28	2.2	915
29	2.0	220
30	2.1	150
31	1.9	84

- Plot a boxplots for the above table

In [202]:
```
print("Categorical distribution plot between aggregate_rating and votes: ")
fig, ax = plt.subplots(figsize=(10,5))
ax = sns.barplot(y="votes", x="aggregate_rating", data=df_votes)
```

```
Categorical distribution plot between aggregate_rating and votes:
```

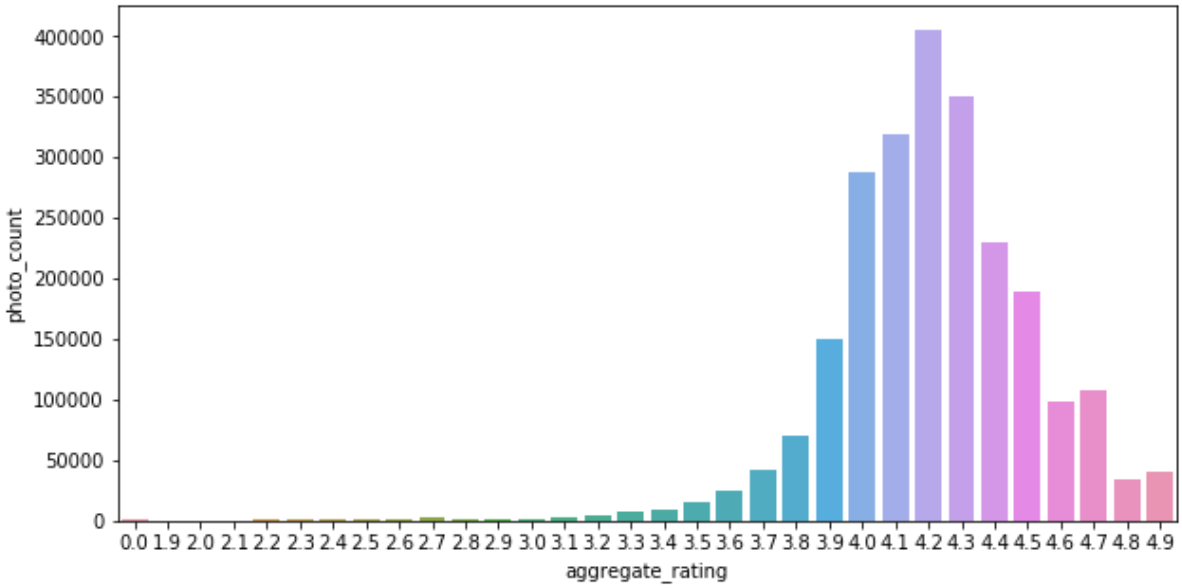


In [203]:
```
print('So,it is clear that maximum number of votes are for ratings between 3.7 and 4.6')
```

```
So,it is clear that maximum number of votes are for ratings between 3.7 and 4.6
```

In [204]:
```
print("Categorical distribution plot between aggregate_rating and photo_count:
")
fig, ax = plt.subplots(figsize=(10,5))
ax = sns.barplot(y="photo_count", x="aggregate_rating", data=df_photo)
```

```
Categorical distribution plot between aggregate_rating and photo_count:
```



In [205]:
```
print('Almost same trend also holds true here. So, maximum photo count is for
 ratings between 3.9 and 4.5')
```

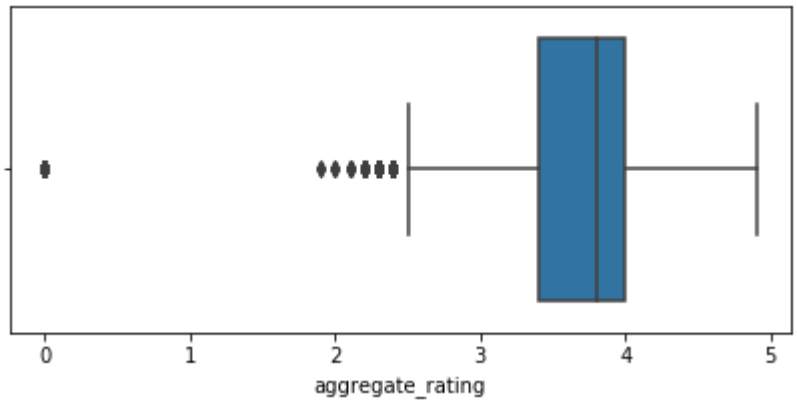
```
Almost same trend also holds true here. So, maximum photo count is for rating
s between 3.9 and 4.5
```

In [206]:
```
print('Now let\'s draw boxplot for each variable:')
```

```
Now let's draw boxplot for each variable:
```

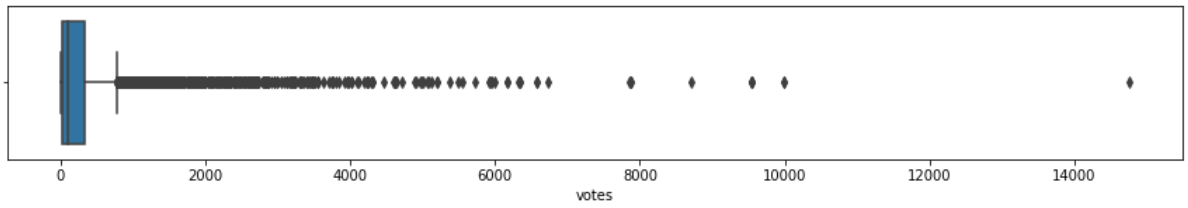
In [207]:
```
fig, ax = plt.subplots(figsize=(7,3))
sns.boxplot(x=df_temp1['aggregate_rating'])
```

Out[207]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x3a02def0c8>
```



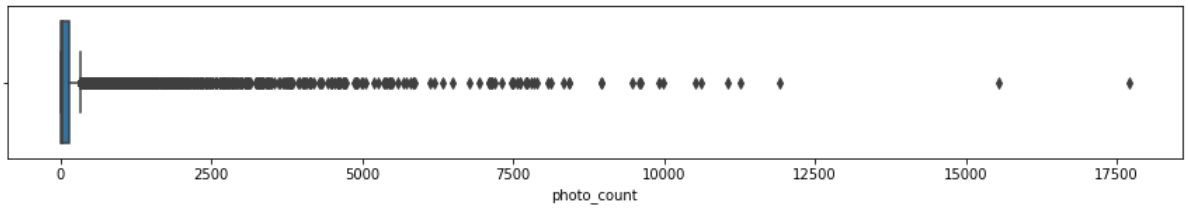
In [208]:
```
fig, ax = plt.subplots(figsize=(15,2))
sns.boxplot(x=df_temp1['votes'])
```

Out[208]: <matplotlib.axes._subplots.AxesSubplot at 0x3a7a2a73c8>



In [209]:
```
fig, ax = plt.subplots(figsize=(15,2))
sns.boxplot(x=df_temp1['photo_count'])
```

Out[209]: <matplotlib.axes._subplots.AxesSubplot at 0x3a02c65a08>



4. The Mumbai city

consider the city mumbai and get a better insights of restuarants in Mumbai.

In [210]:
```
df_mumbai = df_restaurants_copy[df_restaurants_copy['city']=='Mumbai'].drop_du
plicates(keep="first").copy()
df_mumbai.head(2)
```

Out[210]:

	name	establishment	url	address	city	city_i
134852	Drinkery 51	Casual Dining	https://www.zomato.com/mumbai/drinkery-51-band...	1st Floor, Vibgyor Towers, Bandra Kurla Comple...	Mumbai	
134853	Joeys Pizza	Quick Bites	https://www.zomato.com/mumbai/joeys-pizza-mala...	Shop 1, Plot D, Samruddhi Complex, Chincholi B...	Mumbai	

2 rows × 22 columns

1. Expensive restaurants in Mumbai

- Define the costliest restaurants whose average cost of two people exceeds Rs.5000 .
- Plot the restaurants which are costliest based on their average cost for two .

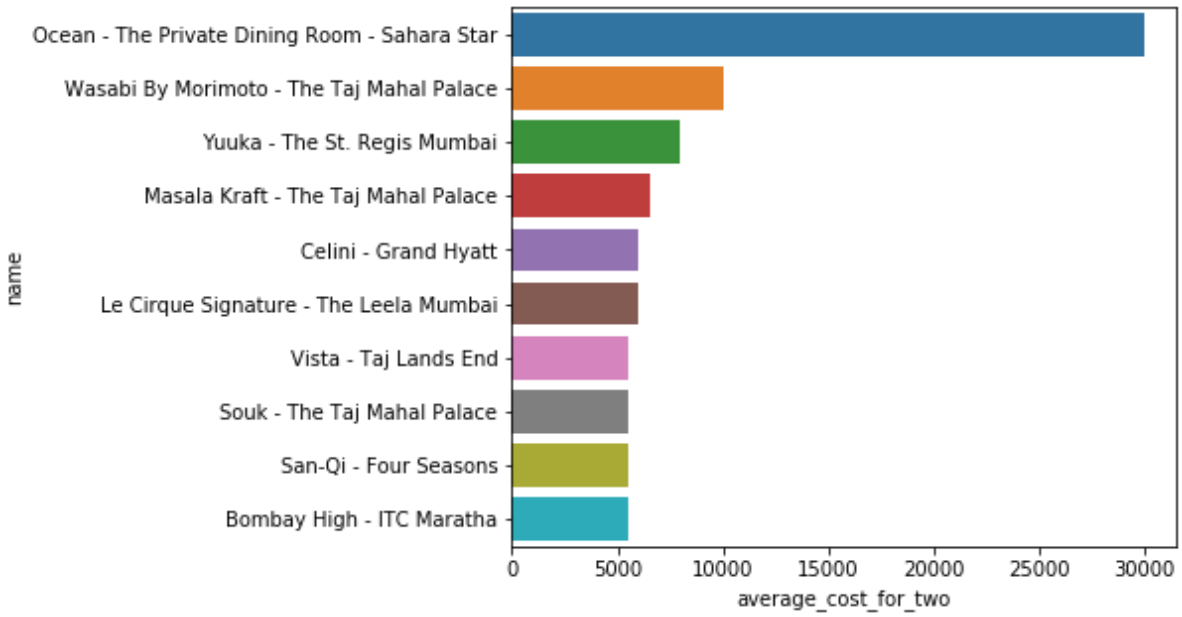
In [211]:
```
df_m_expensive = df_mumbai[df_mumbai['average_cost_for_two']>5000].sort_values
(by="average_cost_for_two", ascending=False).reset_index()
df_m_expensive.head(2)
```

Out[211]:

	index	name	establishment	url	address	city	city
0	136240	Ocean - The Private Dining Room - Sahara Star	Fine Dining	https://www.zomato.com/mumbai/ocean-the-privat...	Hotel Sahara Star, Opposite Domestic Airport, ...	Mumbai	
1	135918	Wasabi By Morimoto - The Taj Mahal Palace	Fine Dining	https://www.zomato.com/mumbai/wasabi-by-morimo...	The Taj Mahal Palace & Tower, Apollo Bunder, C...	Mumbai	

2 rows × 23 columns

In [212]:
```
fig, ax = plt.subplots(figsize=(6,5))
ax = sns.barplot(y="name", x="average_cost_for_two", data=df_m_expensive)
```



2.To find the top 20 cuisines of Mumbai

- select unique cuisines available at restaurants in Mumbai
- sort cuisines based on frequency

In [213]:
```
print(cuisines_freq2(df_mumbai['cuisines']))
```

```
            cuisines  frequency
0       North Indian        765
1          Fast Food        564
2            Chinese        561
3           Desserts        532
4            Italian        403
..               ...        ...
87     Frozen Yogurt          2
88          Assamese          1
89         Cafe Food          1
90  Charcoal Chicken          1
91            German          1

[92 rows x 2 columns]
```

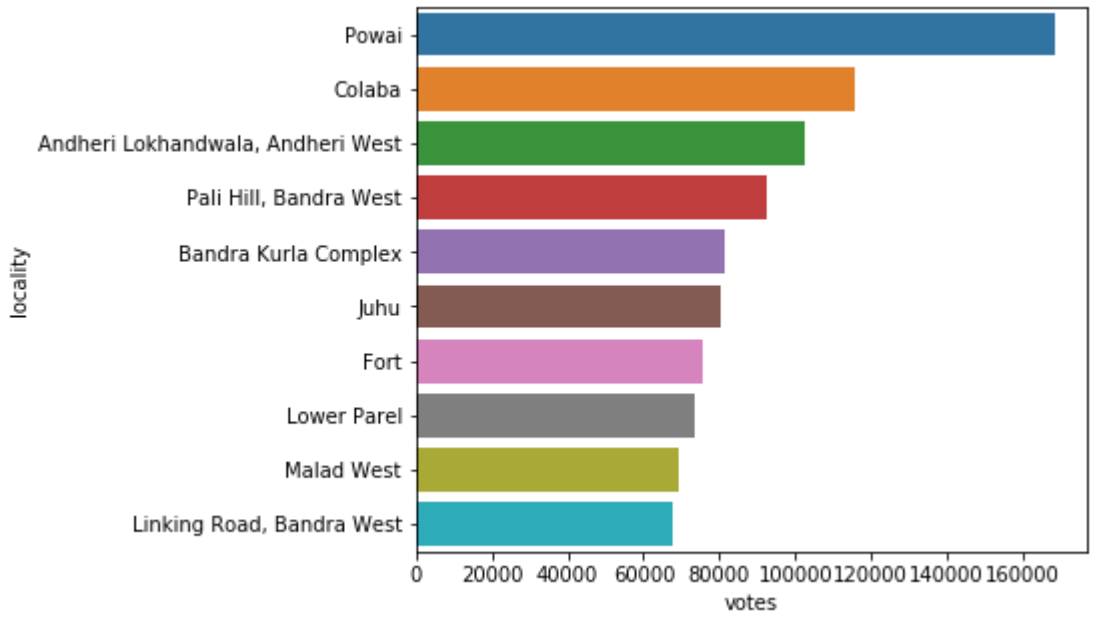
3. To find the popular localities in Mumbai

In [214]:
```
df_popular = df_mumbai.groupby('locality')['votes'].sum().sort_values(ascending=False).reset_index().head(10)
df_popular
```

Out[214]:

	locality	votes
0	Powai	168364
1	Colaba	115597
2	Andheri Lokhandwala, Andheri West	102560
3	Pali Hill, Bandra West	92642
4	Bandra Kurla Complex	81467
5	Juhu	80456
6	Fort	75337
7	Lower Parel	73461
8	Malad West	69493
9	Linking Road, Bandra West	67657

In [215]:
```
fig, ax = plt.subplots(figsize=(6,5))
ax = sns.barplot(y="locality", x="votes", data=df_popular)
```



4. Check for relationship between 'aggregate_rating' and 'average_cost_for_two'

In [216]:
```
df_mumbai[['aggregate_rating','average_cost_for_two']].corr().iloc[0,1]
```

Out[216]: 0.2526206137233988

In [217]:
```
print('Weak Positive Correlation exists between the two as shown below:')
```

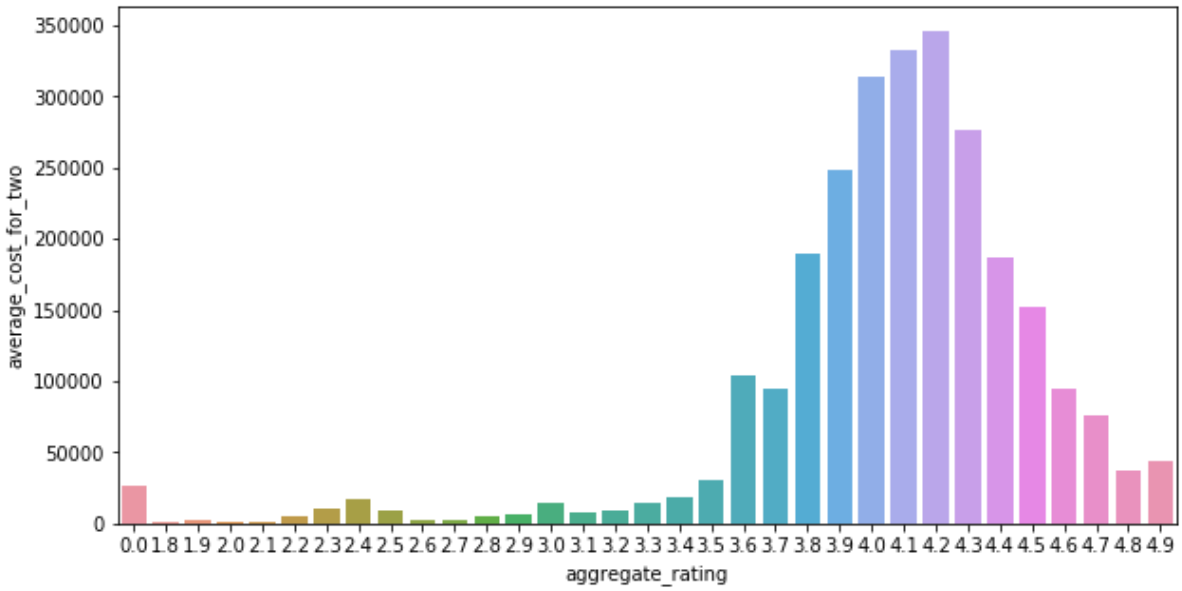
```
Weak Positive Correlation exists between the two as shown below:
```

In [218]:
```
df_mumbai_agg = df_mumbai.groupby('aggregate_rating').sum().sort_values(by="average_cost_for_two", ascending=False)['average_cost_for_two'].reset_index()
df_mumbai_agg
```

Out[218]:

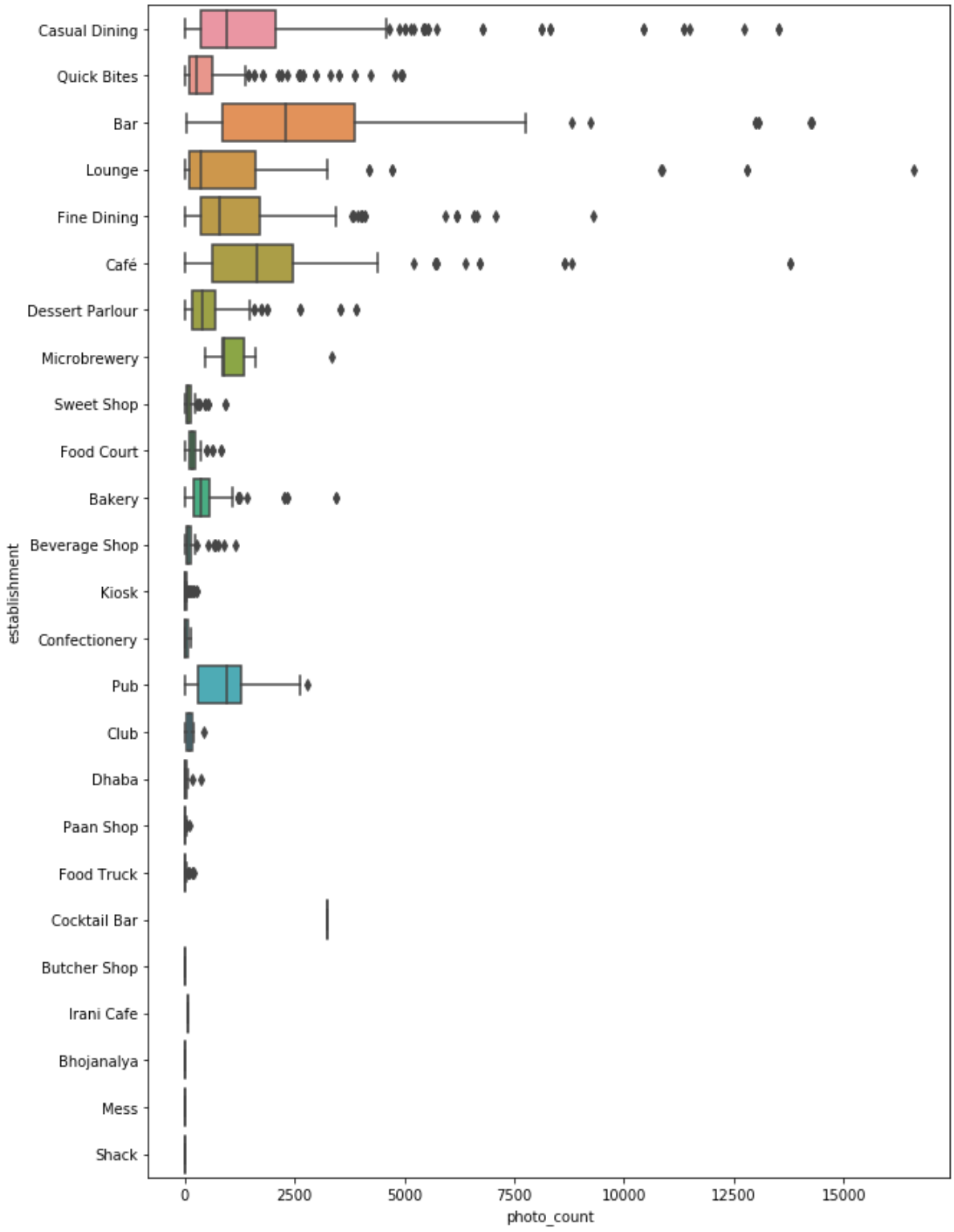
	aggregate_rating	average_cost_for_two
0	4.2	346050
1	4.1	332250
2	4.0	314600
3	4.3	275950
4	3.9	249050
5	3.8	189150
6	4.4	187350
7	4.5	152300
8	3.6	103900
9	4.6	95150
10	3.7	94900
11	4.7	76600
12	4.9	44000
13	4.8	37600
14	3.5	30250
15	0.0	26500
16	3.4	18700
17	2.4	17600
18	3.0	15000
19	3.3	14250
20	2.3	10650
21	2.5	8950
22	3.2	8900
23	3.1	8300
24	2.9	7150
25	2.8	5500
26	2.2	5400
27	2.7	2800
28	1.9	2000
29	2.6	1900
30	2.1	1550
31	2.0	1250
32	1.8	550

In [219]:
```
fig, ax = plt.subplots(figsize=(10,5))
ax = sns.barplot(y="average_cost_for_two", x="aggregate_rating", data=df_mumba
i_agg)
```



5. Multiple box plot for photo_counts based on establishment type.

In [220]:
```
fig, ax = plt.subplots(figsize=(10,15))
ax = sns.boxplot(x="photo_count", y="establishment", data=df_mumbai)
```



6. Check for payments method offered in restaurants

In [221]:
```
payments = ['Cash','Debit Card','Credit Card','Digital Payments Accepted', 'Al
ipay Accepted']
def get_payment_method(x):
    val=""
    x=x.split(", ")
    for var in x:
        if var in payments:
            val = val + ", " + var
        else:
            continue
    if val=="":
        return val
    else:
        return val[2:]
```

In [222]:
```
df_payments = df_mumbai[['name','highlights','latitude','longitude']].drop_dup
licates().copy()
for i in range(df_payments['highlights'].shape[0]):
    df_payments.iloc[i,df_payments.columns.get_loc('highlights')] = get_paymen
t_method(df_payments.iloc[i,df_payments.columns.get_loc('highlights')])

df_payments = df_payments.rename(columns={'name': 'restaurant', 'highlights':
'payment methods'})
df_payments[['restaurant','payment methods','latitude','longitude']].head(10)
```

Out[222]:

	restaurant	payment methods	latitude	longitude
134852	Drinkery 51	Debit Card, Cash, Credit Card	19.067176	72.867493
134853	Joeys Pizza	Cash, Debit Card, Credit Card, Digital Payment...	19.178188	72.834666
134854	Hitchki	Cash, Credit Card, Debit Card	19.119930	72.907331
134855	Tamasha	Cash, Credit Card, Debit Card	19.006060	72.827496
134856	Bayroute	Cash, Debit Card, Credit Card	19.110684	72.825368
134857	Hitchki	Debit Card, Cash, Credit Card	19.069597	72.869834
134858	JLWA	Cash, Credit Card, Debit Card	19.060056	72.836105
134859	London Taxi	Cash, Debit Card, Credit Card	19.005231	72.826119
134860	Colaba Social	Cash, Credit Card, Debit Card	18.921806	72.832472
134861	Garage Inc. Public House	Cash, Credit Card, Debit Card	18.919999	72.830751

In [223]:
```
print('These restaurants accept Only Cash (So, maybe take enough cash while visiting them):')
df_payments[df_payments['payment methods']=='Cash'][['restaurant','payment methods','latitude','longitude']]
```

```
These restaurants accept Only Cash (So, maybe take enough cash while visiting them):
```

Out[223]:

	restaurant	payment methods	latitude	longitude
134878	Guru Kripa	Cash	19.043475	72.861935
134896	Say Cheese	Cash	18.954212	72.818154
134909	Kyani & Co.	Cash	18.944067	72.828574
134917	Gulshan-e-Iran	Cash	18.948045	72.835489
134931	Sardar Pav bhaji	Cash	18.969852	72.815007
...	...	...	...	...
142095	K Bhagat Tarachand	Cash	18.952249	72.830036
142201	The J	Cash	19.079042	72.906105
142260	The J	Cash	19.368251	72.817015
142267	The London Shakes	Cash	19.299198	72.871082
142278	The London Shakes	Cash	19.108434	72.864574

378 rows × 4 columns

In [224]:
```
print('verify for first restaurant:\n')
df_mumbai[df_mumbai['name']=='Drinkery 51'].drop_duplicates().iloc[0,df_mumbai[df_mumbai['name']=='Drinkery 51'].columns.get_loc('highlights')]
```

```
verify for first restaurant:
```

Out[224]:
```
'Dinner, Debit Card, Lunch, Serves Alcohol, Cash, Credit Card, Live Music, Serves Cocktails, Table booking recommended, Available for Functions, Resto Bar, Private Dining Area Available, Wheelchair Accessible, Live Sports Screening, Valet Parking Available, Wine, Beer, Indoor Seating, Restricted Entry, Air Conditioned, Smoking Area, Group Meal, DJ, Nightlife, Fullbar'
```

In [225]:
```
df_payments[df_payments['restaurant']=='Drinkery 51'].drop_duplicates()['payment methods']
```

Out[225]:
```
134852    Debit Card, Cash, Credit Card
Name: payment methods, dtype: object
```

- select unique facilities available at restaurants in western region
- sort facilities based on frequency

In [226]:
```
#Western Region of Mumbai
print("Latitudinal extent of Mumbai according to data available: ",df_mumbai[
'latitude'].min()," degree E to ",df_mumbai['latitude'].max()," degree E")
print("\nWe assume that left 35% and middle 50% is Western Region, which has l
atitude from ",df_mumbai['latitude'].min()," degree E to ",df_mumbai['latitud
e'].quantile(0.35)," degree E and longitude from ",df_mumbai['longitude'].quan
tile(0.25)," degree N to ", df_mumbai['longitude'].quantile(0.75)," degree N")
```

```
Latitudinal extent of Mumbai according to data available:  18.9131928  degree
E to  19.46439399  degree E

We assume that left 35% and middle 50% is Western Region, which has latitude
from  18.9131928  degree E to  19.06537549  degree E and longitude from  72.8
2946885  degree N to  72.86466  degree N
```

In [227]:
```
#Unique facilities (sorted)
print(highlights_sort(df_mumbai[(df_mumbai['latitude'] < df_mumbai['latitude']
.quantile(0.35)) & (df_mumbai['longitude'] < df_mumbai['longitude'].quantile(
0.75)) & (df_mumbai['longitude'] > df_mumbai['longitude'].quantile(0.25))]['hi
ghlights']))
```

```
                highlights  frequency
0                     Cash        440
1           Indoor Seating        377
2                   Dinner        354
3              Credit Card        345
4       Takeaway Available        345
..                     ...        ...
75  Available for Functions          2
76                 Seaside          1
77              Craft Beer          1
78            Bira 91 Beer          1
79                 Rooftop          1

[80 rows x 2 columns]
```

In [228]:
```
df_mumbai_unique = highlights_sort(df_mumbai[(df_mumbai['latitude'] < df_mumba
i['latitude'].quantile(0.35)) & (df_mumbai['longitude'] < df_mumbai['longitud
e'].quantile(0.75)) & (df_mumbai['longitude'] > df_mumbai['longitude'].quantil
e(0.25))]['highlights']).copy()
```

In [229]:
```
df_not_mumbai = df_restaurants_copy[df_restaurants_copy['city']!='Mumbai'].dro
p_duplicates(keep="first").copy()
df_not_mumbai.head(2)
```

Out[229]:

	name	establishment	url	address	city	city_id	loc
0	Bikanervala	Quick Bites	https://www.zomato.com/agra/bikanervala-khanda...	Kalyani Point, Near Tulsi Cinema, Bypass Road,...	Agra	34	Khar
1	Mama Chicken Mama Franky House	Quick Bites	https://www.zomato.com/agra/mama-chicken-mama-...	Main Market, Sadar Bazaar, Agra Cantt, Agra	Agra	34	(

2 rows × 22 columns

In [230]:
```
df_not_mumbai_unique = highlights_sort(df_not_mumbai['highlights']).copy()
df_not_mumbai_unique.head()
```

Out[230]:

	highlights	frequency
0	Cash	54690
1	Takeaway Available	48675
2	Indoor Seating	42486
3	Dinner	39323
4	Lunch	37807

In [231]:
```
val=""
for var in df_mumbai_unique['highlights'].values:
    if var in df_not_mumbai_unique['highlights'].values:
        continue
    else:
        val = val + ", " + var
        val = val[2:]
if val =="":
    val="None"
print("Values exclusive to Mumbai are: ",val)
```

```
Values exclusive to Mumbai are:  None
```

In [232]:
```
print("Thank you :)")
```

```
Thank you :)
```