

ZOMATO API 2 Project .pdf: -

Before Starting:

in many Tasks we have performed the task using (pandas and numpy) approach and also secondly, we approached using Zomato API too in order to verify the correctness of data.

ASK 1 :

summarise all the other cities in Rest of India while those in New Delhi, Ghaziabad, Noida, Gurgaon, Faridabad to Delhi-NCR. Then Complete the following 4 Tasks:

Task 1 : We have to Find number of restaurants present in Delhi NCR vs Rest of India

Task 2: We have to Find the cuisines which are not present in restaurant of Delhi NCR but present in rest of India also verify it with Zomato API

Task 3: We have to Find the top 10 cuisines served by maximum number of restaurants in Delhi NCR and rest of India.

Task 4: Lastly, we have to give short detailed analysis of how cuisine served is different from Delhi NCR to Rest of India.

Lets import the required Libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.style as style
import requests
```

Lets fetch the Zomato API from the location to a dataframe and make its copy into df

```
zomato = pd.read_csv(r"C:\Users\vigya\PycharmProjects\Temp1Learn\zomato.csv", encoding="ISO-8859-1")
df = zomato.copy()
```

before moving further lets make a dataframe containg only indian data and correct its index

```
dfIndia = df[df["Country Code"] == 1].copy()
dfIndia.reset_index(drop=True, inplace=True)
```

Now we will separate our dataframe into two parts i.e for Delhi NCR and Other one for Rest of India

```
dfdelhi = dfIndia[(dfIndia["City"] == "New Delhi") | (dfIndia["City"] == "Ghaziabad") | (dfIndia["City"] == "Noida") |
(dfIndia["City"] == "Gurgaon") | (dfIndia["City"] == "Faridabad")].copy()
```

```
dfrestind = dfIndia[(dfIndia["City"] != "New Delhi") & (dfIndia["City"] != "Ghaziabad") & (dfIndia["City"] != "Noida") &
(dfIndia["City"] != "Gurgaon") & (dfIndia["City"] != "Faridabad")].copy()
```

Task 1 :

We have to Find number of restaurants present in Delhi NCR vs Rest of India

Normal Approach

using pandas and numpy

Lets create two empty lists which we will use in plotting graph for us

```
barmark = []
```

```
barcount = []
```

Now we will get shape of both seprated dataframe i.e for delhi NCR and Other one for Rest of india

And from each fetch its no. of Rows

no. of Rows is nothing but the no. of Restaurants in each location (delhi NCR & Rest of india)

After fetching we will append them in our empty list we made for plotting graph

```
delhiR = dfdelhi.shape[0]
```

```
barmark.append("Delhi-NCR")
```

```
barcount.append(delhiR)
```

```
restindR = dfrestind.shape[0]
```

```
barmark.append("Rest of India")
```

```
barcount.append(restindR)
```

lets Plot a bar graph to represent Delhi-NCR v Rest of India in terms of No. of Restaurant present

```
def labels(barmark, barcount):
```

```
    for i in range(len(barmark)):
```

```
        plt.text(i, barcount[i]//2, barcount[i], ha='center', Bbox=dict(facecolor='white', alpha=0.3))
```

```
plt.bar(barmark, barcount, width=0.6, color=['red', 'green'], edgecolor="cyan", linewidth=1, alpha=0.8)
```

```
labels(barmark, barcount)
```

```
style.use('ggplot')
```

```
plt.title("Delhi-NCR v Rest of India", size=25)
```

```
plt.xticks(rotation=40)
```

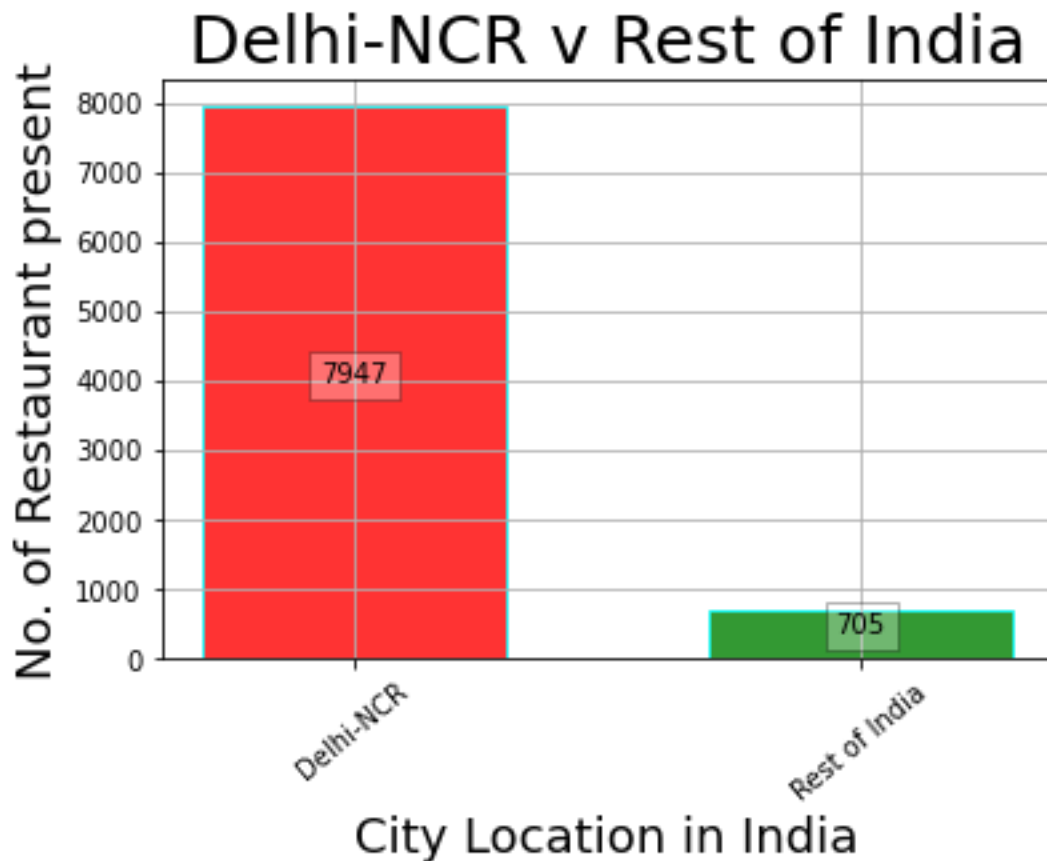
```
plt.xlabel("City Location in India", size = 18)
```

```
plt.ylabel("No. of Restaurant present", size = 18)
```

```
plt.grid()
```

```
plt.figure(figsize=(16, 9))
```

```
plt.show()
```



Answer:

As we can clearly see above in bar graph according to csv, No. of restaurants in Delhi NCR are much more than Rest of India No. of restaurants in Delhi NCR -> 7947 No. of restaurants in Rest of India -> 705

Seems unfair so we may try to approach the task with API approach and see if we can fetch data from API

API Approach

using Zomato API link - https://app.swaggerhub.com/apis-docs/Vivek-Raj/zomato-api/1.0.0#/Restaurant%20Search/get_search

Approach is in simple steps

we will fetch delhiNCR and Rest of india unique cities first

Then using each City name via loop on unique cities we will be Fetching its corresponding etype, eid, cityid

using these information lastly we will be Fetching Restaurant Counts for particular city

Also additionally we would try and except the Keyerror exception in case of Rest of india cities

as it may arise due to API limit exceed

```
dcity = dfdelhi["City"].values
```

```
udcity = np.unique(dcity)
```

```
rcity = dfrestind["City"].values
```

```
urcity = np.unique(rcity)
```

```

countd = 0
for c in range(udcity.size):
    """Fetching etype, eid, cityid of delhi NCR city"""
    h = {"Accept": "application/json", "Authorization": "Bearer 68a952b60155797d28def1585d888b9f",
        "user-key": "68a952b60155797d28def1585d888b9f"}
    response1 = requests.get('https://developers.zomato.com/api/v2.1/locations'
        , headers=h
        , params = {"query": udcity[c]}
        )
    data1 = response1.json()
    etype = data1["location_suggestions"][0]["entity_type"]
    eid = data1["location_suggestions"][0]["entity_id"]
    cityid = data1["location_suggestions"][0]["city_id"]

    """Fetching Restaurant Counts for particular city"""

    response2 = requests.get('https://developers.zomato.com/api/v2.1/search'
        , headers=h
        , params = {"entity_type": etype,
            "entity_id": eid})
    data2 = response2.json()
    countd = countd + data2["results_found"]
print(countd, "- No. of restaurants present in Delhi NCR")

```

```

countr = 0
for c in range(urcity.size):

    """Fetching etype, eid, cityid of Rest of india city"""

    response1 = requests.get('https://developers.zomato.com/api/v2.1/locations'
        , headers=h
        , params = {"query": urcity[c]}
        )
    data1 = response1.json()
    etype = data1["location_suggestions"][0]["entity_type"]
    eid = data1["location_suggestions"][0]["entity_id"]
    cityid = data1["location_suggestions"][0]["city_id"]

    """Fetching Restaurant Counts for particular city"""

    response2 = requests.get('https://developers.zomato.com/api/v2.1/search'
        , headers=h
        , params = {"entity_type": etype,
            "entity_id": eid})
    data2 = response2.json()
    try:
        countr = countr + data2["results_found"]

```

```
except Exception as e:
    pass
print(countr, "- No. of restaurants present in Rest of India")
```

Output:

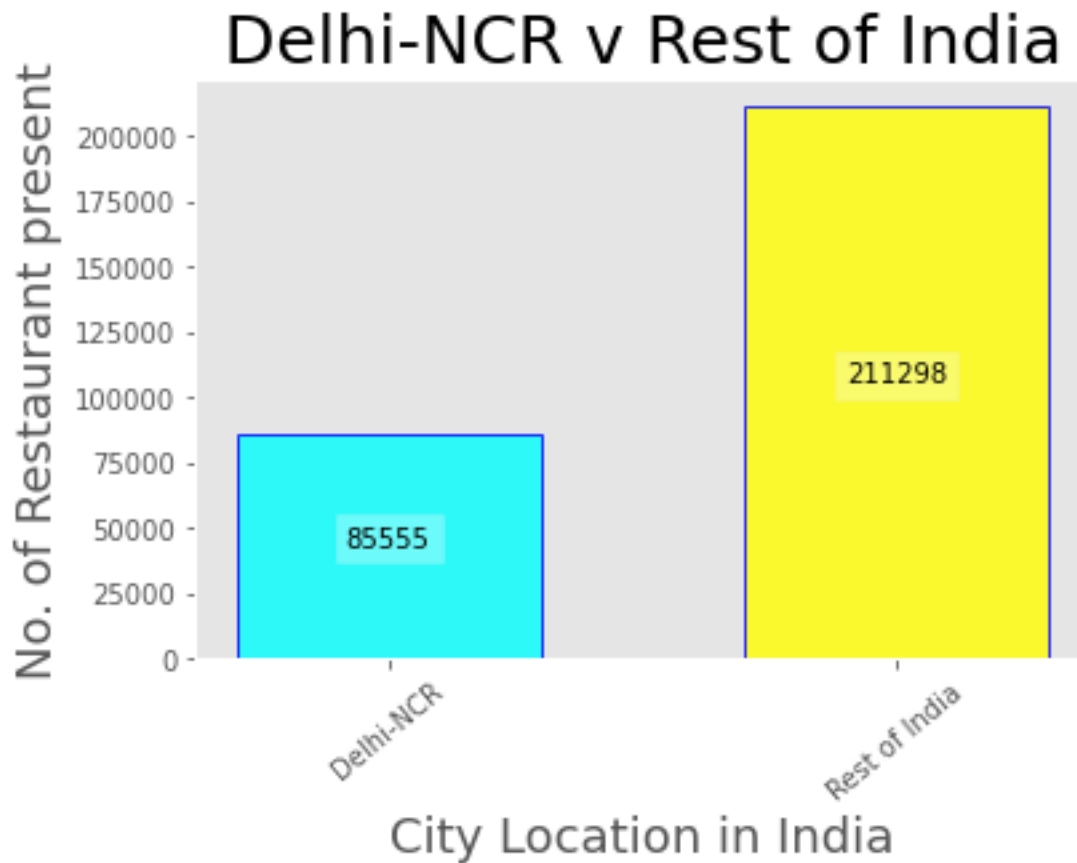
```
85555 - No. of restaurants present in Delhi NCR
211298 - No. of restaurants present in Rest of India
```

As we can see we have different output in this approach as we had in Normal one
Lets plot the graph again , which will look quite similar but no. of data has increased
Lets create two empty lists and append counts in them

```
barmark = []
barcount = []
barmark.append("Delhi-NCR")
barcount.append(countd)

barmark.append("Rest of India")
barcount.append(countr)
def labels(barmark, barcount):
    for i in range(len(barmark)):
        plt.text(i, barcount[i]//2, barcount[i], ha='center', Bbox=dict(facecolor='white', alpha=0.3))

plt.bar(barmark, barcount, width=0.6, color=['cyan', 'yellow'], edgecolor="blue", linewidth=1, alpha=0.8)
labels(barmark, barcount)
style.use('ggplot')
plt.title("Delhi-NCR v Rest of India", size=25)
plt.xticks(rotation=40)
plt.xlabel("City Location in India", size = 18)
plt.ylabel("No. of Restaurant present", size = 18)
plt.grid()
plt.figure(figsize=(16, 9))
plt.show()
```



Task 2 :

We have to Find the cuisines which are not present in restaurant of Delhi NCR but present in rest of India also verify it with Zomato API

Normal Approach

using pandas and numpy

```
# Firstly lets fetch the cuisines in Delhi NCR  
# By taking whole Cuisines column from delhi dataframe and putting it into numpy array  
# Then we will traverse over it and split each element on basis of ", "  
# after splitting we ll append the splited elements into another empty list  
# later on this list would be used to fetch unique cuisines in delhi NCR
```

```
delhicuisines = dfdelhi["Cuisines"].values  
sepdcus = []  
for i in range(len(delhicuisines)):  
    delhicuisines[i] = delhicuisines[i].split(", ")  
    for j in range(len(delhicuisines[i])):  
        sepdcus.append(delhicuisines[i][j])
```

```
udcuisines = np.unique(sepdcus)
```

```
# Lets fetch the cuisines in Rest of india cities also  
# Doing it the same way we did for delhi NCR
```

```

restindcuisines = dfrestind["Cuisines"].values
seprcus = []
for i in range(len(restindcuisines)):
    restindcuisines[i] = restindcuisines[i].split(", ")
    for j in range(len(restindcuisines[i])):
        seprcus.append(restindcuisines[i][j])

urcuisines = np.unique(seprcus)

# Now the need is to Find the cuisines which are not present in restaurant of Delhi NCR but present in rest of India
# we will take an empty list
# traverse over unique cuisines in rest of india and if any particular cuisine in Rest india matches Delhi NCR
# we will simply append it into out newly created empty list "restnotdelcus"

restnotdelcus = []

for i in urcuisines:
    if i in udcuisines:
        pass
    else:
        restnotdelcus.append(i)

print(restnotdelcus)

```

Output:

```
['BBQ', 'Cajun', 'German', 'Malwani']
```

Answer:

We can see from Normal approach there are four cuisines which are not present in restaurant of Delhi NCR but present in rest of India, However we ll try to verify it with Zomato API too in our next phase

API Approach

using Zomato API link - https://app.swaggerhub.com/apis-docs/Vivek-Raj/zomato-api/1.0.0#/Restaurant%20Search/get_search

```

# We will be taking unique cities from DataFrame into Np array
# Then we would traverse over its each element i.e city name
# Fetch each cityname's id
# Using that city id we can fetch the cuisines
# after fetching cuisines we will put them in separate empty list
# And lastly we will make it a unique list using Np.unique
# In this way we will have unique cuisines list of particular area
# Fetching unique cuisines list of Delhi NCR

```

```

delcities = dfdelhi["City"].values
udelcities = np.unique(delcities)
apidelcus = []
for i in udelcities:
    h = {"Accept": "application/json", "Authorization": "Bearer 68a952b60155797d28def1585d888b9f",

```

```

    "user-key":"68a952b60155797d28def1585d888b9f"
}
response = requests.get('https://developers.zomato.com/api/v2.1/locations'
    , headers=h
    , params = {"query":i

    })
data = response.json()
delcityid = data["location_suggestions"][0]["city_id"]

response = requests.get('https://developers.zomato.com/api/v2.1/cuisines'
    , headers=h
    , params = {"city_id":delcityid
    })
data = response.json()
cus = data["cuisines"]
for i in cus:
    apidelcus.append(i["cuisine"]["cuisine_name"])
uapidelcus = np.unique(apidelcus)
print(uapidelcus)

```

Output:

```

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

```

Fetching unique cuisines list of Rest of India

```

restcity = dfrestind["City"].values
urestcity = np.unique(restcity)

apirestcus = []
for i in urestcity:
    response1 = requests.get('https://developers.zomato.com/api/v2.1/locations'

```



```

        , headers=h
        , params = {"query":i
                    })
data1 = response1.json()
rescityid = data1["location_suggestions"][0]["city_id"]

response2 = requests.get('https://developers.zomato.com/api/v2.1/cuisines'
                          , headers=h
                          , params = {"city_id":rescityid
                                      })
data2 = response2.json()
cus = data2["cuisines"]

for i in cus:
    apirestcus.append(i["cuisine"]["cuisine_name"])

uapirestcus = np.unique(apirestcus)
print(uapirestcus)

```

Output:

```

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

```

Now the need is to Find the cuisines which are not present in restaurant of Delhi NCR but present in rest of India

we will take an empty list

traverse over unique cuisines in rest of india and if any particular cuisine in Rest india matches Delhi NCR

we will simply append it into out newly created empty list "restnotdelcusapi"

```
restnotdelcusapi = []
```

```

for i in uapirestcus:
    if i in uapidelcus:
        pass
    else:
        restnotdelcusapi.append(i)

print(restnotdelcusapi)

```

Output:

```

['Armenian', 'Australian', 'Bangladeshi', 'Bohri', 'Cake', 'Coffee and Tea', 'Fusion',
'Garhwali', 'German', 'Indian', 'Irish', 'Konkan', 'Liquor', 'Malwani', 'Modern
Australian', 'Mongolian', 'Moroccan', 'Peruvian', 'Relief fund', 'Swedish']

```

We see Result is diffrent in case of Zomato API approach due to incomplete dataset

Task 3 :

We have to Find the top 10 cuisines served by maximum number of restaurants in Delhi NCR and rest of India.

Normal Approach

using pandas and numpy

```

# using normal approach we had already fetched separate cuisines for delhi NCR and rest of india
# using those separate cuisines we will fetch unique cuisines and its count first of all
# Then we will make a loop in range of 10
# each time we will find index of max of count
# print corresponding cuisine and its count
# make the max count as 0 by using the index we found

```

Top 10 cuisines for Delhi NCR

```

udcuisines, udccounts = np.unique(sepdcus, return_counts = True)
gra_cus_name_del = []
gra_cus_count_del = []
for i in range(10):
    ind = np.argmax(udccounts)
    print(udcuisines[ind], udccounts[ind])
    gra_cus_name_del.append(udcuisines[ind])
    gra_cus_count_del.append(udccounts[ind])
    udccounts[ind] = 0

```

Output:

```

North Indian 3597
Chinese 2448
Fast Food 1866
Mughlai 933
Bakery 697

```

South Indian 569
Continental 547
Desserts 542
Street Food 538
Italian 535

Top 10 cuisines for Rest of India

```
urcuisines, urccounts = np.unique(seprcus, return_counts = True)
gra_cus_name_rest = []
gra_cus_count_rest = []
for i in range(10):
    ind = np.argmax(urccounts)
    print(urcuisines[ind], urccounts[ind])
    gra_cus_name_rest.append(urcuisines[ind])
    gra_cus_count_rest.append(urccounts[ind])
    urccounts[ind] = 0
```

Output:

North Indian 349
Chinese 242
Continental 177
Italian 147
Cafe 136
Fast Food 97
South Indian 62
Mughlai 59
Desserts 55
Mexican 50

Answer:

we can see North Indian is the top cuisine in both Delhi NCR and Rest of India

API Approach

using Zomato API link - https://app.swaggerhub.com/apis-docs/Vivek-Raj/zomato-api/1.0.0#/Restaurant%20Search/get_search

Now doing same using API

We need top 10 cuisines served by maximum number of restaurants in Delhi NCR and rest of India.

for cuisines in one restaurant can be done by /restaurant

need res_id of that restaurant

for res_id goto restaurant search and we have to put city name by query

Above approach is fine, but we will have to face a problem of limited results being shown to us may be only 20

So we will try another approach , lets take Delhi NCR one and same would be implemented in case of rest of india

Firstly we ll fetch unique delhi NCR cities from dfdelhi dataframe "City" column values into np array

Traverse over this unique delhi NCR city list

Take each city name as query in params and fetch etype, eid, cityid of city
Using cityid of city in params we will be Fetching cuisines id and name of particular city
we will be appending cuisine names into one list(cuisinesname) and its id into another list (cuisinesid)
Lastly we would be nested traversing over cuisine id
in this nested loop we will be fetching restaurant count for each cuisines
With the help of etype, eid of city and cuisine id of the particular cuisine
restaurant count is nothing but "results_found"
we will be making a dictionary with cuisine name as keys and its count as value which we will be adding again and again
additionally we would apply try exceptexception o "result shown"
may be API limit exceed and it may throw error
{'code': 440, 'status': '', 'message': 'API limit exceeded'} -> try except

delhi Cuisines

```
dic = {}
```

```
delcities = dfdelhi["City"].values
```

```
udelcities = np.unique(delcities)
```

```
for c in range(udelcities.size):
```

```
    """Fetching etype, eid, cityid of city"""
```

```
    response1 = requests.get('https://developers.zomato.com/api/v2.1/locations'
                              , headers=h
                              , params = {"query":udelcities[c]}
                              )
```

```
    data1 = response1.json()
```

```
    etype = data1["location_suggestions"][0]["entity_type"]
```

```
    eid = data1["location_suggestions"][0]["entity_id"]
```

```
    cityid = data1["location_suggestions"][0]["city_id"]
```

```
    """Fetching cuisines id and name of particular city"""
```

```
    response2 = requests.get('https://developers.zomato.com/api/v2.1/cuisines'
                              , headers=h
                              , params = {"city_id":cityid}
                              )
```

```
    data2 = response2.json()
```

```
    data2 = data2["cuisines"]
```

```
    cuisinesid = []
```

```
    cuisinesname = []
```

```
    for i in data2:
```

```
        cuisinesid.append(i["cuisine"]["cuisine_id"])
```

```
        cuisinesname.append(i["cuisine"]["cuisine_name"])
```

```
    cuisinesid = np.array(cuisinesid)
```

```
    cuisinesname = np.array(cuisinesname)
```

```
    """fetching restaurant count for each cuisines"""
```

```

for i in range(cuisinesid.size):
    response3 = requests.get('https://developers.zomato.com/api/v2.1/search'
                             , headers=h
                             , params = {"entity_type":etype,
                                           "entity_id":eid,
                                           "cuisines":cuisinesid[i]})
    data3 = response3.json()
    try:
        count = data3["results_found"]
        if cuisinesname[i] in dic:
            dic[cuisinesname[i]] = dic[cuisinesname[i]] + count
        else:
            dic[cuisinesname[i]] = count
    except Exception as e:
        pass
dic

```

Output:

```

{'Afghan': 52,
 'African': 4,
 'American': 5053,
 'Andhra': 148,
 'Arabian': 624,
 'Asian': 27681,
 'Assamese': 17,
 'Awadhi': 102,
 'BBQ': 327,
 'Bakery': 16230,
 'Bar Food': 184,
 'Belgian': 10,
 'Bengali': 333,
 'Beverages': 16474,
 'Bihari': 147,
 'Biryani': 8662,
 'Brazilian': 0,
 'British': 1,
 'Bubble Tea': 12,
 'Burger': 3785,
 'Burmese': 61,
 'Cafe': 1631,
 'Cantonese': 4,
 'Charcoal Chicken': 5,
 'Chettinad': 39,
 'Chinese': 26921,
 'Coffee': 959,
 'Continental': 2129,
 'Cuisine Varies': 7,
 'Desserts': 14176,

```

'Drinks Only': 5,
'Egyptian': 2,
'Ethiopian': 1,
'European': 2864,
'Fast Food': 30326,
'Finger Food': 255,
'French': 30,
'Frozen Yogurt': 24,
'Goan': 15,
'Greek': 3,
'Grocery': 1,
'Gujarati': 34,
'Healthy Food': 2461,
'Himachali': 1,
'Hot dogs': 25,
'Hyderabadi': 709,
'Ice Cream': 2941,
'Indonesian': 16,
'Iranian': 6,
'Israeli': 1,
'Italian': 9468,
'Jamaican': 1,
'Japanese': 557,
'Juices': 509,
'Kashmiri': 61,
'Kebab': 4991,
'Kerala': 93,
'Korean': 63,
'Lebanese': 681,
'Lucknowi': 100,
'Maharashtrian': 77,
'Malaysian': 51,
'Mangalorean': 21,
'Mediterranean': 155,
'Mexican': 969,
'Middle Eastern': 1043,
'Mishti': 13,
'Mithai': 3409,
'Modern Indian': 98,
'Momos': 5288,
'Mughlai': 10479,
'Naga': 44,
'Nepalese': 54,
'North Eastern': 94,
'North Indian': 44268,
'Odia': 10,
'Oriental': 26709,
'Paan': 287,
'Pakistani': 2,

```
'Panini': 18,  
'Parsi': 10,  
'Pasta': 903,  
'Pizza': 7189,  
'Poké': 55,  
'Portuguese': 11,  
'Rajasthani': 86,  
'Raw Meats': 521,  
'Roast Chicken': 116,  
'Rolls': 8417,  
'Russian': 2,  
'Salad': 1906,  
'Sandwich': 3438,  
'Seafood': 1414,  
'Sindhi': 8,  
'Singaporean': 4,  
'South American': 1,  
'South Indian': 4595,  
'Spanish': 7,  
'Sri Lankan': 7,  
'Steak': 168,  
'Street Food': 6720,  
'Sushi': 339,  
'Tamil': 5,  
'Tea': 1075,  
'Tex-Mex': 637,  
'Thai': 1266,  
'Tibetan': 276,  
'Turkish': 20,  
'Vietnamese': 32,  
'Wraps': 996}
```

```
cusname = dic.keys()  
cusname = list(cusname)  
cusname = np.array(cusname)  
cuscount = dic.values()  
cuscount = list(cuscount)  
cuscount = np.array(cuscount)
```

```
# Then we will make a loop in range of 10  
# each time we will find index of max of count  
# print corresponding cuisine and its count  
# make the max count as 0 by using the index we found
```

```
gra_apicus_name_del = []  
gra_apicus_count_del = []
```

```
for i in range(10):
```

```

ind = np.argmax(cuscount)
print(cusname[ind], cuscount[ind])
gra_apicus_name_del.append(cusname[ind])
gra_apicus_count_del.append(cuscount[ind])
cuscount[ind] = 0

```

Output:

```

North Indian 44268
Fast Food 30326
Asian 27681
Chinese 26921
Oriental 26709
Beverages 16474
Bakery 16230
Desserts 14176
Mughlai 10479
Italian 9468

```

Now we ll do same as above for Rest of India Top 10 Cuisine

Fetching unique cities of rest of india

```

restindcities = dfrestind["City"].values
urestindcities = np.unique(restindcities)
urestindcities

```

Output:

```

array(['Agra', 'Ahmedabad', 'Allahabad', 'Amritsar', 'Aurangabad',
      'Bangalore', 'Bhopal', 'Bhubaneshwar', 'Chandigarh', 'Chennai',
      'Coimbatore', 'Dehradun', 'Goa', 'Guwahati', 'Hyderabad', 'Indore',
      'Jaipur', 'Kanpur', 'Kochi', 'Kolkata', 'Lucknow', 'Ludhiana',
      'Mangalore', 'Mohali', 'Mumbai', 'Mysore', 'Nagpur', 'Nashik',
      'Panchkula', 'Patna', 'Puducherry', 'Pune', 'Ranchi',
      'Secunderabad', 'Surat', 'Vadodara', 'Varanasi', 'Vizag'],
      dtype=object)

```

Sometimes Run after above codes API limit crosses and below code will throw error

if we run below code after some gap of time it runs

```

dic2 = {}
restindcities = dfrestind["City"].values
urestindcities = np.unique(restindcities)

```

for c in range(urestindcities.size):

"""Fetching etype, eid, cityid of city"""

response1 = requests.get('https://developers.zomato.com/api/v2.1/locations'


```

        , headers=h
        , params = {"query":urestindcities[c]}
    )
data1 = response1.json()
etype = data1["location_suggestions"][0]["entity_type"]
eid = data1["location_suggestions"][0]["entity_id"]
cityid = data1["location_suggestions"][0]["city_id"]

"""Fetching cuisines id and name of particular city"""

response2 = requests.get('https://developers.zomato.com/api/v2.1/cuisines'
        , headers=h
        , params = {"city_id":cityid}
    )
data2 = response2.json()
data2 = data2["cuisines"]

cuisinesid = []
cuisinesname = []
for i in data2:
    cuisinesid.append(i["cuisine"]["cuisine_id"])
    cuisinesname.append(i["cuisine"]["cuisine_name"])
cuisinesid = np.array(cuisinesid)
cuisinesname = np.array(cuisinesname)

"""fetching restaurant count for each cuisines"""

for i in range(cuisinesid.size):
    response3 = requests.get('https://developers.zomato.com/api/v2.1/search'
        , headers=h
        , params = {"entity_type":etype,
                    "entity_id":eid,
                    "cuisines":cuisinesid[i]})
    data3 = response3.json()
    try:
        count = data3["results_found"]
        if cuisinesname[i] in dic2:
            dic2[cuisinesname[i]] = dic2[cuisinesname[i]] + count
        else:
            dic2[cuisinesname[i]] = count
    except Exception as e:
        pass
dic2

```

Output:

```

{'American': 623,
 'Asian': 3412,
 'Awadhi': 6,

```

'BBQ': 52,
'Bakery': 2591,
'Bar Food': 16,
'Belgian': 0,
'Bengali': 28,
'Beverages': 3367,
'Biryani': 920,
'Burger': 544,
'Cafe': 386,
'Chinese': 3360,
'Coffee': 184,
'Continental': 485,
'Desserts': 2382,
'European': 432,
'Fast Food': 6615,
'Finger Food': 38,
'Gujarati': 811,
'Healthy Food': 223,
'Ice Cream': 676,
'Italian': 1885,
'Japanese': 16,
'Juices': 150,
'Kebab': 225,
'Kerala': 6,
'Korean': 3,
'Lebanese': 92,
'Lucknowi': 5,
'Maharashtrian': 239,
'Mediterranean': 15,
'Mexican': 192,
'Middle Eastern': 72,
'Mithai': 579,
'Momos': 184,
'Mughlai': 769,
'North Indian': 5604,
'Oriental': 2861,
'Paan': 50,
'Pasta': 115,
'Pizza': 1497,
'Rajasthani': 88,
'Raw Meats': 9,
'Roast Chicken': 32,
'Rolls': 418,
'Salad': 158,
'Sandwich': 1291,
'Seafood': 133,
'South Indian': 1299,
'Street Food': 2275,
'Sushi': 9,

'Tea': 149,
'Thai': 65,
'Tibetan': 4,
'Wraps': 171,
'Afghan': 15,
'Andhra': 7,
'Arabian': 95,
'Bihari': 4,
'Bohri': 1,
'Burmese': 3,
'Chettinad': 3,
'Cuisine Varies': 1,
'Drinks Only': 1,
'French': 2,
'Frozen Yogurt': 30,
'Goan': 2,
'Greek': 1,
'Grocery': 0,
'Hot dogs': 19,
'Hyderabadi': 61,
'Indonesian': 2,
'Kashmiri': 2,
'Konkan': 2,
'Malaysian': 1,
'Malwani': 6,
'Mishti': 5,
'Modern Indian': 5,
'Panini': 12,
'Parsi': 3,
'Russian': 1,
'Sindhi': 7,
'Tex-Mex': 55,
'Turkish': 1,
'Vietnamese': 3,
'Charcoal Chicken': 4,
'North Eastern': 2,
'Pakistani': 0,
'Steak': 2,
'Singaporean': 1,
'African': 0,
'Assamese': 0,
'Australian': 0,
'Bangladeshi': 0,
'British': 0,
'Bubble Tea': 0,
'Cantonese': 0,
'Egyptian': 0,
'German': 0,
'Iranian': 0,

```

'Irish': 0}

cusname = dic2.keys()
cusname = list(cusname)
cusname = np.array(cusname)
cuscount = dic2.values()
cuscount = list(cuscount)
cuscount = np.array(cuscount)
gra_apicus_name_rest = []
gra_apicus_count_rest = []
for i in range(10):
    ind = np.argmax(cuscount)
    print(cusname[ind], cuscount[ind])
    gra_apicus_name_rest.append(cusname[ind])
    gra_apicus_count_rest.append(cuscount[ind])
    cuscount[ind] = 0

```

Output:

```

Fast Food 6615
North Indian 5604
Asian 3412
Beverages 3367
Chinese 3360
Oriental 2861
Bakery 2591
Desserts 2382
Street Food 2275
Italian 1885

```

We can see result is different again

Task 4 :

short detailed analysis of how cuisine served is different from Delhi NCR to Rest of India.

Analysis:

Task 1 -

we found that number of restaurants in delhi NCR cities are far more than number of restaurants in rest of india cities according to Zomato csv and API

Task 2 -

We found that there are many cuisines which are not present in restaurant of Delhi NCR but present in rest of India also verified it with Zomato API, however the result was different but as a result we know that cuisine have variety difference in both areas

Task 3 -

In this task we found top 10 cuisines of Both the areas(delhi NCR and Rest of India), however here also when we verified it with Zomato API ,the result was again different. But we can analyse that the data varies in every approach but "North indian" is the cuisine which is most preferred by mostly restaurants according to Normal approach in both areas(delhi NCR and Rest of India) and in API approach "North indian is on 2nd position" for both areas(delhi NCR and Rest of India) We can plot both approaches and for for both areas(delhi NCR and Rest of India)

```
plt.figure(figsize = (20, 10) )
plt.subplot(2,2,1)
x = gra_cus_name_del
y = gra_cus_count_del
size = gra_cus_count_del
style.use("ggplot")
plt.scatter(x, y, s=size,c="c", marker="o", alpha = 0.5, edgecolor = "y", linewidth = 8)
plt.xlabel("Cuisines", fontsize = 15)
plt.xticks(rotation=40)
plt.ylabel("No. of Restaurants", fontsize = 15)
plt.title("Delhi NCR - Normal Approach", fontsize = 20)
```

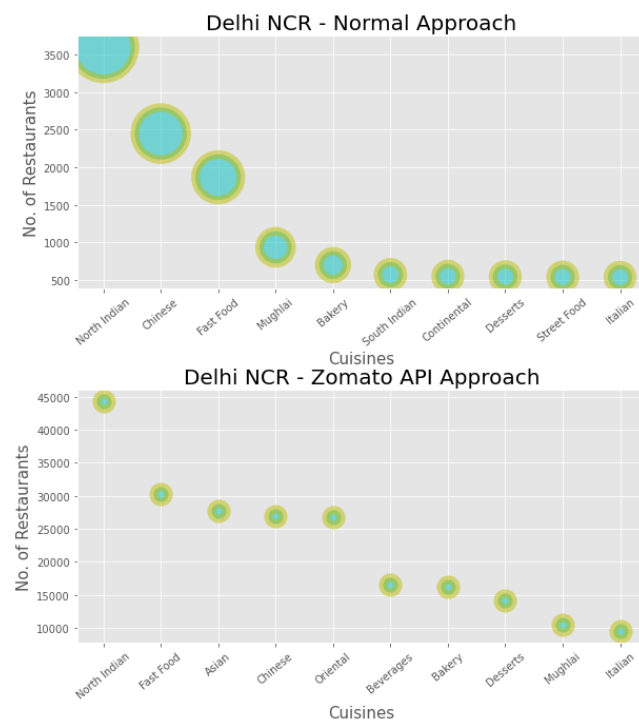
```
plt.subplot(2,2,2)
x = gra_cus_name_rest
y = gra_cus_count_rest
size = gra_cus_count_rest
style.use("ggplot")
plt.scatter(x, y, s=size,c="c", marker="o", alpha = 0.5, edgecolor = "r", linewidth = 8)
plt.xlabel("Cuisines", fontsize = 15)
plt.xticks(rotation=40)
plt.ylabel("No. of Restaurants", fontsize = 15)
plt.title("Rest of India - Normal Approach", fontsize = 20)
```

```
plt.subplot(2,2,3)
x = gra_apicus_name_del
y = gra_apicus_count_del
style.use("ggplot")
plt.scatter(x, y, s=200,c="c", marker="o", alpha = 0.5, edgecolor = "y", linewidth = 8)
plt.xlabel("Cuisines", fontsize = 15)
plt.xticks(rotation=40)
plt.ylabel("No. of Restaurants", fontsize = 15)
plt.title("Delhi NCR - Zomato API Approach", fontsize = 20)
```

```
plt.subplot(2,2,4)
x = gra_apicus_name_rest
y = gra_apicus_count_rest
style.use("ggplot")
plt.scatter(x, y, s=200,c="c", marker="o", alpha = 0.5, edgecolor = "r", linewidth = 8)
plt.xlabel("Cuisines", fontsize = 15)
```

```
plt.xticks(rotation=40)
plt.ylabel("No. of Restaurants", fontsize = 15)
plt.title("Rest of India - Zomato API Approach", fontsize = 20)

plt.subplots_adjust(left=0.1,
                    bottom=0.1,
                    right=1,
                    top=0.9,
                    wspace=0.4,
                    hspace=0.4)
plt.show()
```



ABOUT GRAPHS:

In normal approach :--

x axis depicts -> cuisines

y axis depicts -> No. of restaurants

also size of each marker depicts no. of restaurants only

In Zomato API approach :--

x axis depicts -> cuisines

y axis depicts -> No. of restaurants

From the above set of graphs we can illustrate from the numbers of restaurants that in Delhi NCR correspondingly there is very large number of Restaurants for top cuisines Also "North indian" cuisine can be considered as Top cuisine in all over India if we take on average

ASK 2 :

Tasks on User Rating of a restaurant

Then Complete the following 2 Major Tasks:

Task 1 : short detail analysis of how the rating is affected by restaurant due following features:

- 1.Number of Votes given Restaurant
- 2.Restaurant serving more number of cuisines.
- 3.Average Cost of Restaurant
- 4.Restaurant serving some specific cuisines.

Task 2: weighted restaurant rating of each locality and find out the top 10 localities with more weighted restaurant rating?

$Weighted\ Restaurant\ Rating = \frac{\sum (number\ of\ votes * rating)}{\sum (number\ of\ votes)}$

Task 1 :

short detail analysis of how the rating is affected by restaurant's feature

Sub task 1:

how the rating is affected by Number of Votes given Restaurant

For this subtask we will be working on data from CSV

we will fetch total no. of votes each restaurant got from "Votes" column

then we will fetch what rating each restaurant got from "Aggregate rating" column

then we will be plotting a scatter graph to see how the trend goes

also we will limit the y axis to 8000 votes as there are very less no. of restaurants which have cotes above that

and if we include them our graph looks more complex , so we will discard them

```
votes = df["Votes"].values
```

```
rating = df["Aggregate rating"].values
```

```
x = rating
```

```
y = votes
```

```
style.use("ggplot")
```

```
plt.figure(figsize = (16, 9) )
```

```
plt.scatter(x, y,c="g", marker="o", alpha = 0.5, edgecolor = "c", linewidth = 8)
```

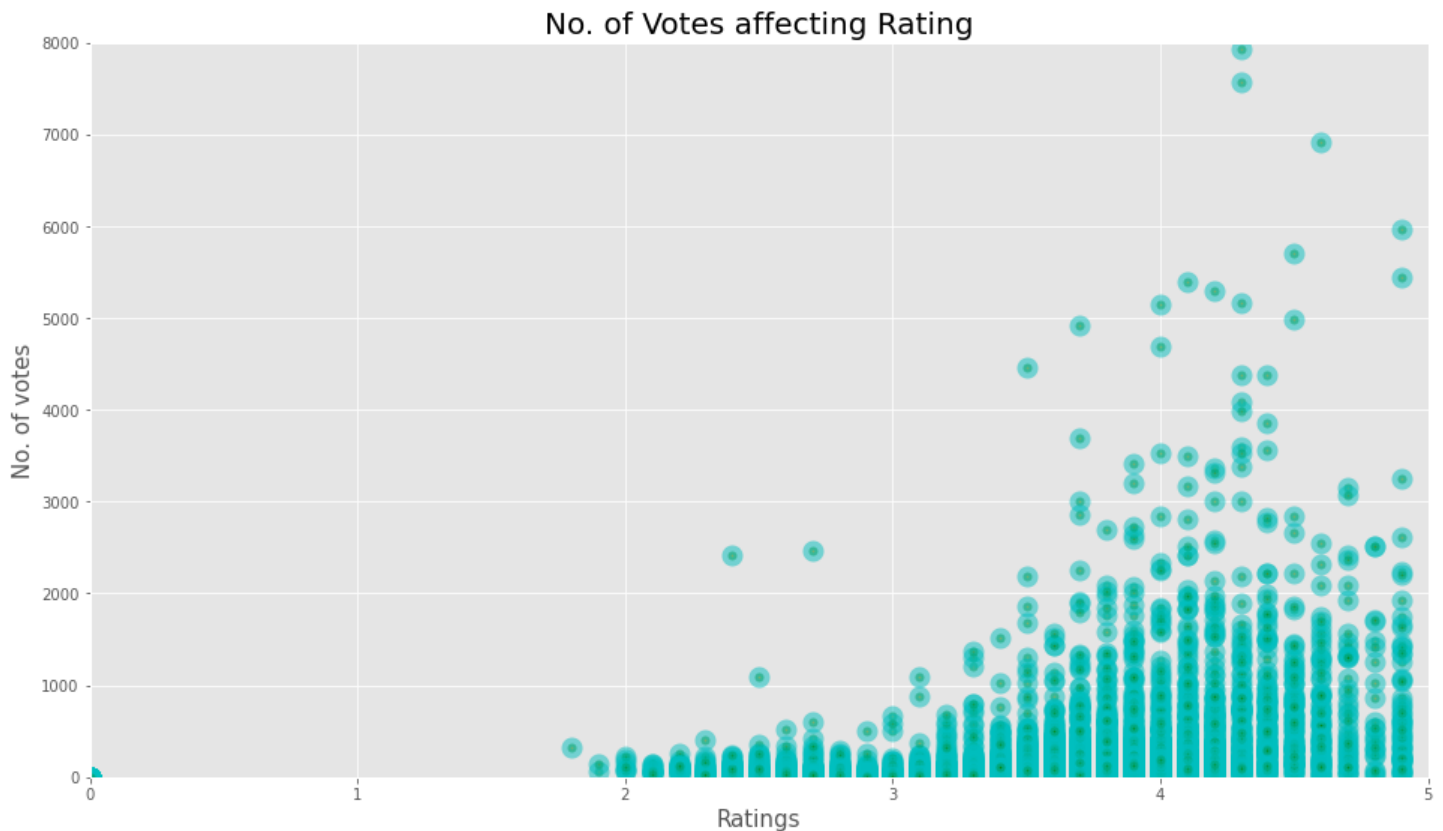
```
plt.xlabel("Ratings", fontsize = 15)
```

```
plt.ylabel("No. of votes", fontsize = 15)
```

```
plt.title("No. of Votes affecting Rating", fontsize = 20)
```

```
plt.axis([0,5,0.0,8000])
```

```
plt.show()
```



Answer to subtask 1:

we can analyse from the plotted scatter graph above ,
 that it hardly matters weather we are giving any no. of votes rating is going to depend on what kind of rating the person give in his votes
 5 excellent rating votes will give aggregate rating as excellent
 however, 1000 poor rated votes will give aggregate rating as poor

Even then seeing the past trend we can fetch some information from the graph that is:

Less number of votes has more good ratings
 i.e Ratings are inversely proportional to no. of votes

in above Subtask we approached using normal approach(using pandas and numpy)
as if we tried fetching data through Zomato API it would have been very difficult
The API allows us to go for only limited calls and hence it may have thrown error of "API limit Exceed"

Sub task 2:

how the rating is affected by Restaurant serving more number of cuisines.

For this subtask we will be again working on data from CSV
we will fetching total no. of cuisines each restaurant got from "Cuisines" column
then we already fetched what rating each restaurant got from "Aggregate rating" column
then we will be plotting a scatter graph to see how the trend goes

```
cus = df["Cuisines"].copy()
```


Less number of Cuisines has more good ratings
i.e Rating is inversely proportional to no. of cuisines

in above Subtask we again approached using normal approach (using pandas and numpy)
as similarly here also if we tried fetching data through Zomato API it would have been very difficult
The API allows us to go for only limited calls and hence it may have thrown error of "API limit Exceed"

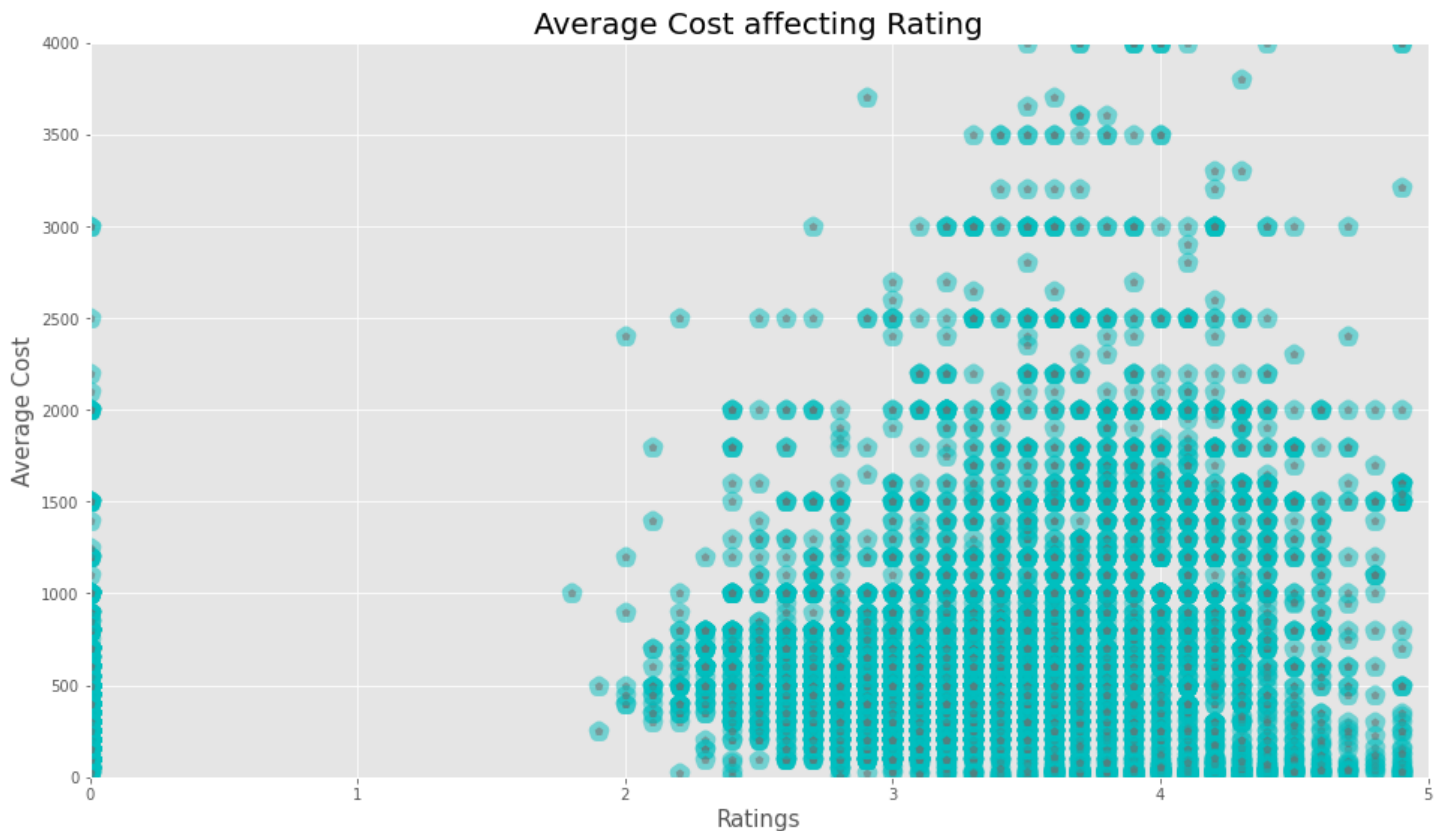
Sub task 3:

how the rating is affected by Average Cost of Restaurant

For this subtask we will be again working on data from CSV
we will be fetching Average Cost of each restaurant from "Average Cost for two" column
then we have already fetched what rating each restaurant got from "Aggregate rating" column
then we will be plotting a scatter graph to see how the trend goes
also we will limit the y axis to 4000 as avg cost as there are mostly restaurants which have cost around that
so we can analyse the data in better way now
and if we include them our graph looks more complex , so we will discard them

```
cost = df["Average Cost for two"].values
x = rating
y = cost
style.use("ggplot")
plt.figure(figsize = (16, 9) )
plt.scatter(x, y, c="r", marker="p", alpha = 0.5, edgecolor = "c", linewidth = 8)
plt.xlabel("Ratings", fontsize = 15)
plt.ylabel("Average Cost", fontsize = 15)
plt.title("Average Cost affecting Rating", fontsize = 20)
plt.axis([0,5,0.0,4000])

plt.show()
```



Answer to subtask 3:

we can analyse from the plotted scatter graph above ,
 surely ratings depend upon Average Cost of the restaurant
 It is good for a restaurant rating to increase if it has less Average Cost that can be depicted from the graph above
 as the graph has more opaqueness on downward side where the average cost is less
 and if we focus on top side where Average Cost is more we can clearly depict less rating over there by looking at
 the slightly transparent markers

so after seeing the past trend we can fetch some information from the graph that is:

Less Average Cost has more good ratings
 i.e Rating is inversely proportional to Average Cost

*# in above Subtask we again approached using normal approach(using pandas and numpy)
 # as similarly here also if we tried fetching data through Zomato API it would have been very difficult
 # The API allows us to go for only limited calls and hence it may have thrown error of "API limit Exceed"*

Sub task 4:

how the rating is affected by Restaurant serving some specific cuisines.

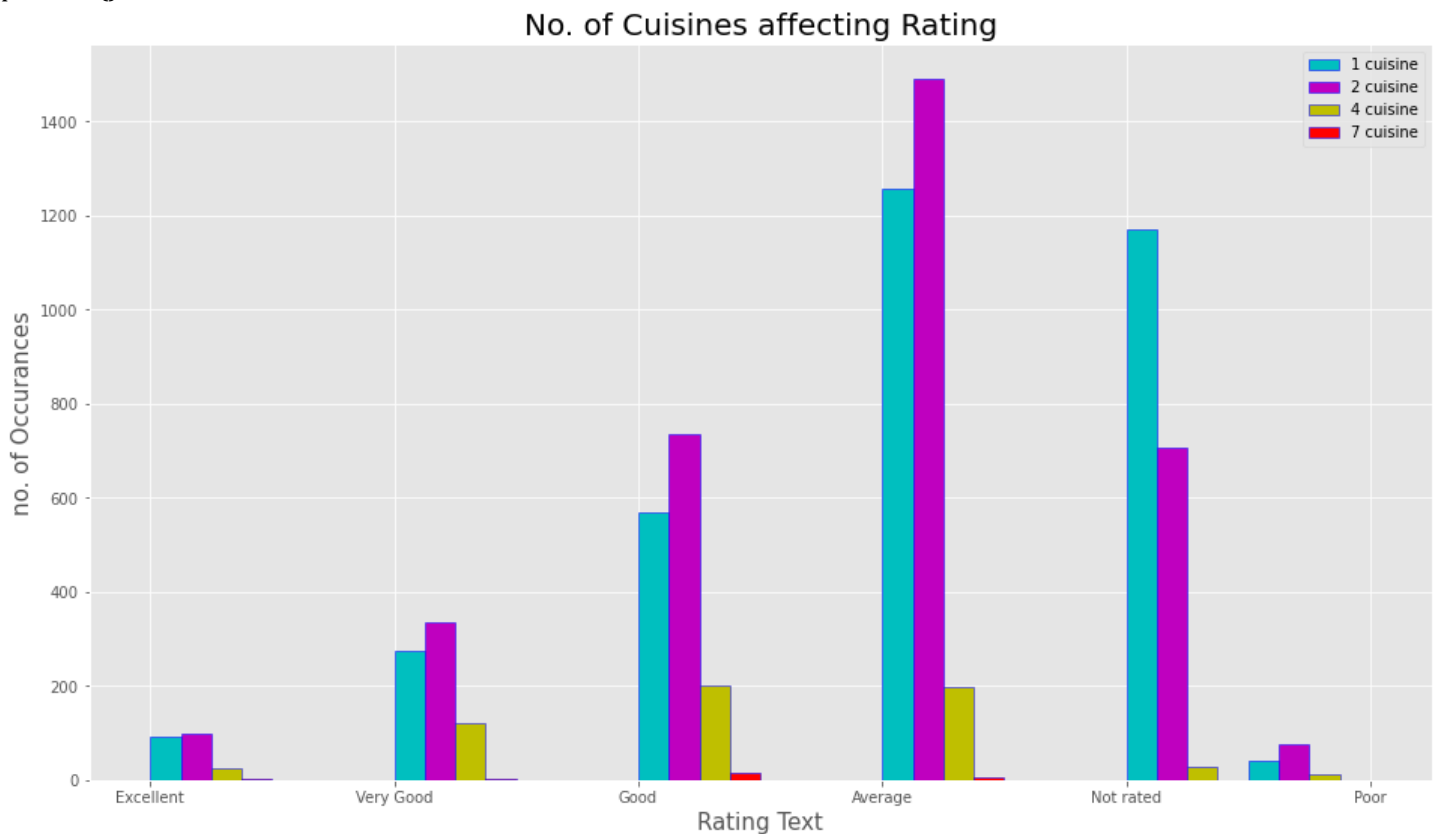
*# For this subtask we will be again working on data from CSV
 # we have "Cuisines" list that has no. of cuisines in particular restaurant
 # then we have already fetched what rating each restaurant got from "Aggregate rating" column in rating list as np array
 # then we will be plotting a scatter graph to see how the trend goes for different cuisine numbers
 # also we will use x axis as natural numbers increasing upto no. of restaurants
 # so we can analyse the data in better way now*

and if we include them our graph looks more complex , so we will discard them

we also have "Aggregate rating"

```
Cuisines = np.array(Cuisines)
ratingtext = df["Rating text"].values
ind1 = np.where(Cuisines == 1)
ind2 = np.where(Cuisines == 2)
ind3 = np.where(Cuisines == 4)
ind4 = np.where(Cuisines == 7)
rat1 = ratingtext[ind1]
rat2 = ratingtext[ind2]
rat3 = ratingtext[ind3]
rat4 = ratingtext[ind4]
style.use("ggplot")
plt.figure(figsize = (16, 9) )
plt.hist([rat1, rat2, rat3, rat4], rwidth = 1, align="mid", edgecolor = 'b',
         histtype = 'bar', orientation = "vertical", color=["c", "m", "y", "r"],
         label = ["1 cuisine", "2 cuisine", "4 cuisine", "7 cuisine"])
plt.xlabel("Rating Text", fontsize = 15)
plt.ylabel("no. of Occurances", fontsize = 15)
plt.title("No. of Cuisines affecting Rating", fontsize = 20)
plt.legend()

plt.show()
```



As we see from the above graph also we discussed earlier as the no. of cuisine is increasing the rating is decreasing

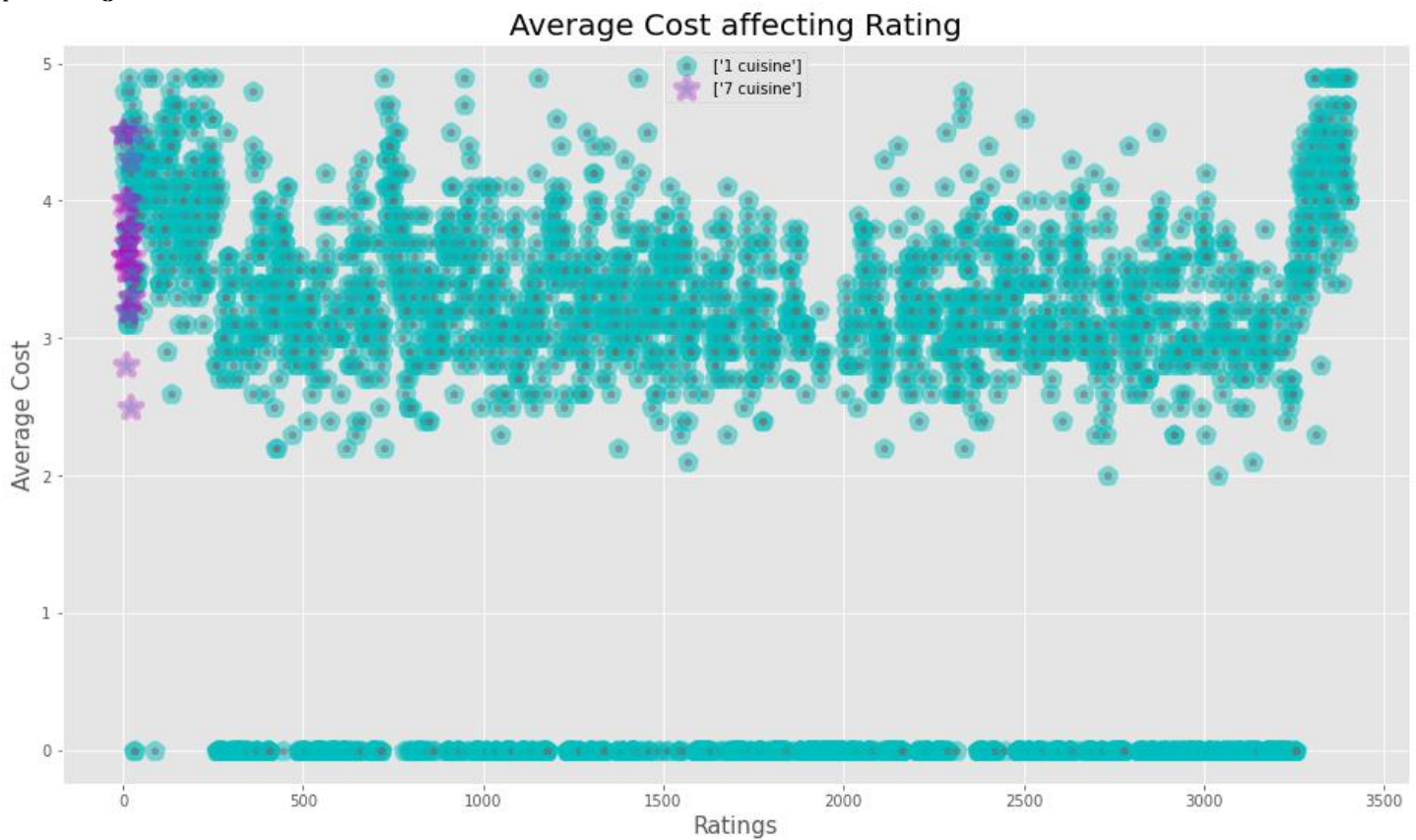
However analysis says that one must keep 2 on average cuisines for better rating of its restaurant

keeping more cuisine won't give more rating , rating will decrease
so, instead of keeping more cuisine getting speciality in some will fetch more rating

```

nat = []
for i in range(len(rat1)):
    nat.append(i)
nat2 = []
for i in range(len(rat4)):
    nat2.append(i)
x = nat
y = rating[ind1]
x2 = nat2
y2 = rating[ind4]
style.use("ggplot")
plt.figure(figsize = (16, 9) )
plt.scatter(x, y, c="r", marker="p", alpha = 0.5, edgecolor = "c", linewidth = 8, label=["1 cuisine"])
plt.scatter(x2, y2, c="c", s= 200, marker="*", alpha = 0.3, edgecolor = "m", linewidth = 5, label=["7 cuisine"])
plt.xlabel("Ratings", fontsize = 15)
plt.ylabel("Average Cost", fontsize = 15)
plt.title("Average Cost affecting Rating", fontsize = 20)
plt.legend()
plt.show()

```



From the above graph also we can simply confirm that restaurant with specific cuisines will have more rating
than ones who have numbers of cuisines

Answer to subtask 4:

we can analyse from the plotted scatter graph above ,
surely ratings depend upon No. of cuisines of the restaurant
It is good for a restaurant rating to increase if it has less No. of cuisines , that can be depicted from the graph above
as the graph has more opaqueness for 1 cuisine
we can simply confirm that restaurant with specific cuisines will have more rating than ones who have numbers of cuisines

so after seeing the past trend we can fetch some information from the graph that is:

Less No. of cuisines has more good ratings
i.e Rating is inversely proportional to No. of cuisines

Task 2 :

weighted restaurant rating of each locality and find out the top 10 localities with more weighted restaurant rating?
 $Weighted\ Restaurant\ Rating = \frac{\sum (number\ of\ votes * rating)}{\sum (number\ of\ votes)}$

*# Firstly we will fetch the required columns from dataframe which we made using provided CSV
i.e "Votes", "Aggregate rating" and get vr that is multiplication of both in separate variable
also we will take "Locality" from dataframe itself into l*

```
votes = df["Votes"].copy()
votes.fillna(0, inplace = True)
votes = votes.values
```

```
r = df["Aggregate rating"].copy()
r.fillna(0, inplace = True)
r = r.values
```

```
vr = votes * r
l = df["Locality"].values
```

*# Now we will make a dictionary containing unique locality as key and sum of Votes*Rating as Values*

```
d = {}
for i in range(l.size):
    if l[i] in d:
        d[l[i]] = d[l[i]] + vr[i]
    else:
        d[l[i]] = vr[i]
```

Now we will make another dictionary which also has unique locality as its keys and sum of votes as its values

```
d2 = {}
for i in range(l.size):
```

```

if l[i] in d2:
    d2[l[i]] = d2[l[i]] + votes[i]
else:
    d2[l[i]] = votes[i]

# Now we will take 1'st dictionary keys as unique locality
# 1'st dictionary value i.e sum of Votes*Rating in vr
# 2'nd dictionary values i.e sum of votes in v

uloc = d.keys()
uloc = list(uloc)
uloc = np.array(uloc)

uvr = d.values()
uvr = list(uvr)
uvr = np.array(uvr)

uv = d2.values()
uv = list(uv)
uv = np.array(uv)

# Now we will traverse over size of unique locality
# calculate Weighted Restaurant Rating =  $\Sigma (\text{number of votes} * \text{rating}) / \Sigma (\text{number of votes})$ 
# append in wrr empty list
# as division by 0 is not possible we have included condition for that

wrr = []
for i in range(uloc.size):
    if uv[i] != 0:
        temp = uvr[i]/uv[i]
        wrr.append(temp)
    else:
        wrr.append(0)

wrr = np.array(wrr,dtype=float)

# we will run loop 10 times in which:
# Now we will find max wrr index
# print its corresponding locality and wrr
# set wrr max to 0

for i in range(10):
    ind = np.argmax(wrr)
    print(uloc[ind], wrr[ind])
    wrr[ind] = 0

```

Output:

Sofitel Philippine Plaza Manila, Pasay City 4.9

Setor De Clubes Esportivos Sul 4.9
Gıçvea 4.9
The Milk District 4.9
Windermere 4.9
Paia 4.9
Kenwood 4.9
New Tampa 4.9
Venetian Village, Al Maqtaa 4.9
Deira City Centre Area 4.9

ASK 3 :

Visualization

Complete the following 5 Tasks:

Task 1 : bar graph : top 15 restaurants have a maximum number of outlets

Task 2: histogram : aggregate rating of restaurant(drop the unrated restaurant).

Task 3: bar graph : top 10 restaurants in the data with the highest number of votes.

Task 4: pie graph : top 10 cuisines present in restaurants in the USA.

Task 5: bubble graph : number of Restaurants present in the city of India and keeping the weighted restaurant rating of the city in a bubble.

Task 1 :

bar graph : top 15 restaurants have a maximum number of outlets

For restaurant names we will fetch details from df dataframe column "Restaurant Name"

```
rn = df["Restaurant Name"].values
```

we will make a dictionary which will contain unique restaurant names as keys and outlet counts as values

```
dn = {}
```

```
for i in rn:
```

```
    if i in dn:
```

```
        dn[i] = dn[i] + 1
```

```
    else:
```

```
        dn[i] = 1
```

```
names = list(dn.keys())
```

```
outlets = list(dn.values())
```

```
names = np.array(names)
```

```
outlets = np.array(outlets)
```



```
barmark = []  
barcount = []
```

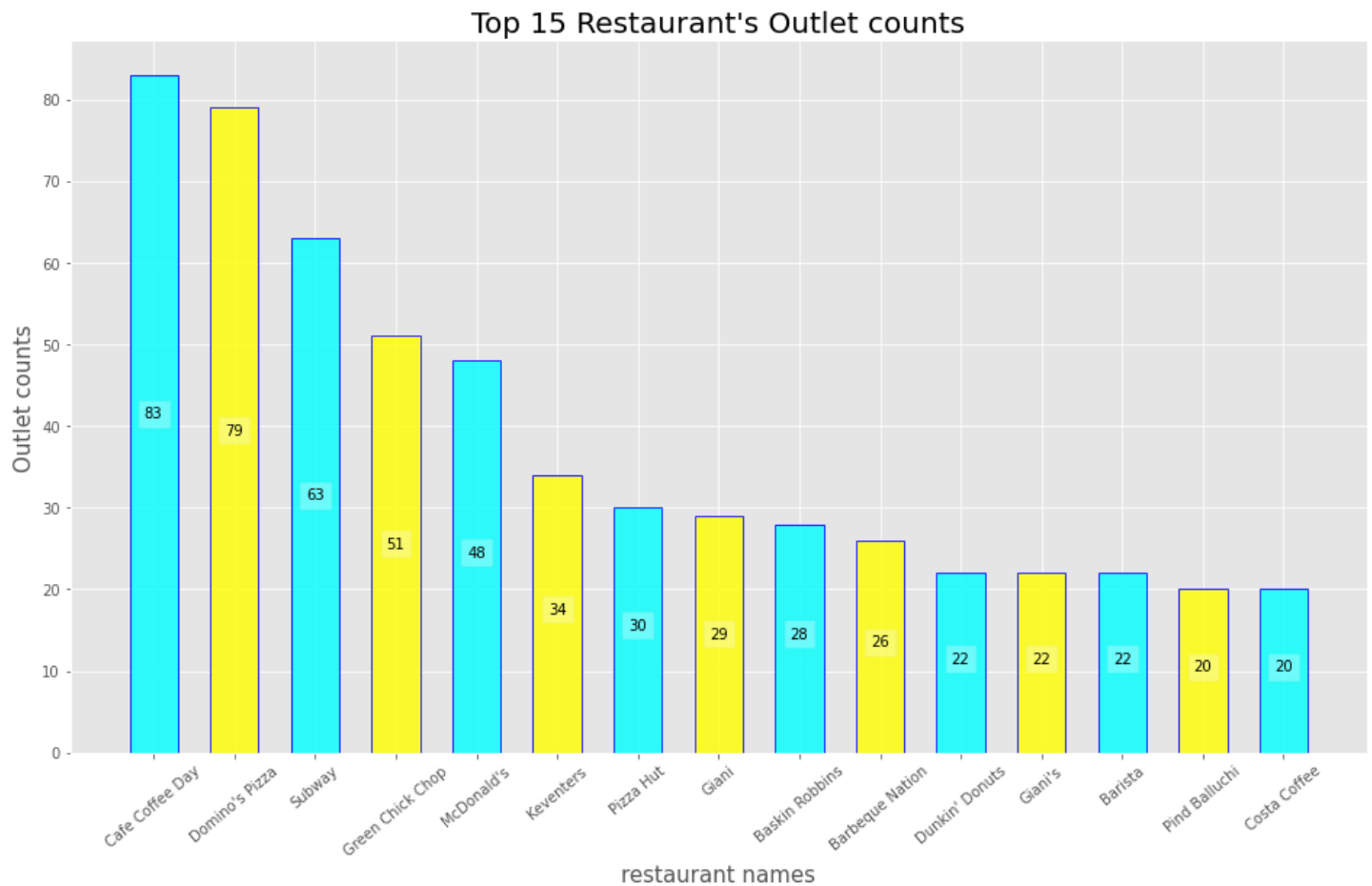
```
# finding top 15 restaurants with max no. of outlets in decreasing order
```

```
for i in range(15):  
    ind = np.argmax(outlets)  
    barmark.append(names[ind])  
    barcount.append(outlets[ind])  
    outlets[ind] = 0
```

```
# Now we will plot the bar graph
```

```
def labels(barmark, barcount):  
    for i in range(len(barmark)):  
        plt.text(i, barcount[i]//2, barcount[i], ha='center', Bbox=dict(facecolor='white', alpha=0.3))
```

```
style.use("ggplot")  
plt.figure(figsize = (16, 9) )  
plt.bar(barmark, barcount, width=0.6, color=['cyan', 'yellow'], edgecolor="blue", linewidth=1, alpha=0.8)  
labels(barmark, barcount)  
plt.xlabel("restaurant names", fontsize = 15)  
plt.ylabel("Outlet counts", fontsize = 15)  
plt.xticks(rotation=40)  
plt.title("Top 15 Restaurant's Outlet counts", fontsize = 20)  
plt.show()
```



Task 2 :

histogram : aggregate rating of restaurant(drop the unrated restaurant).

```
agr = df["Aggregate rating"].copy()
```

```
ind = agr[agr == 0].index
```

```
agr.drop(ind, inplace=True)
```

```
agr = agr.values
```

```
x = agr
```

```
plt.figure(figsize = (16, 9))
```

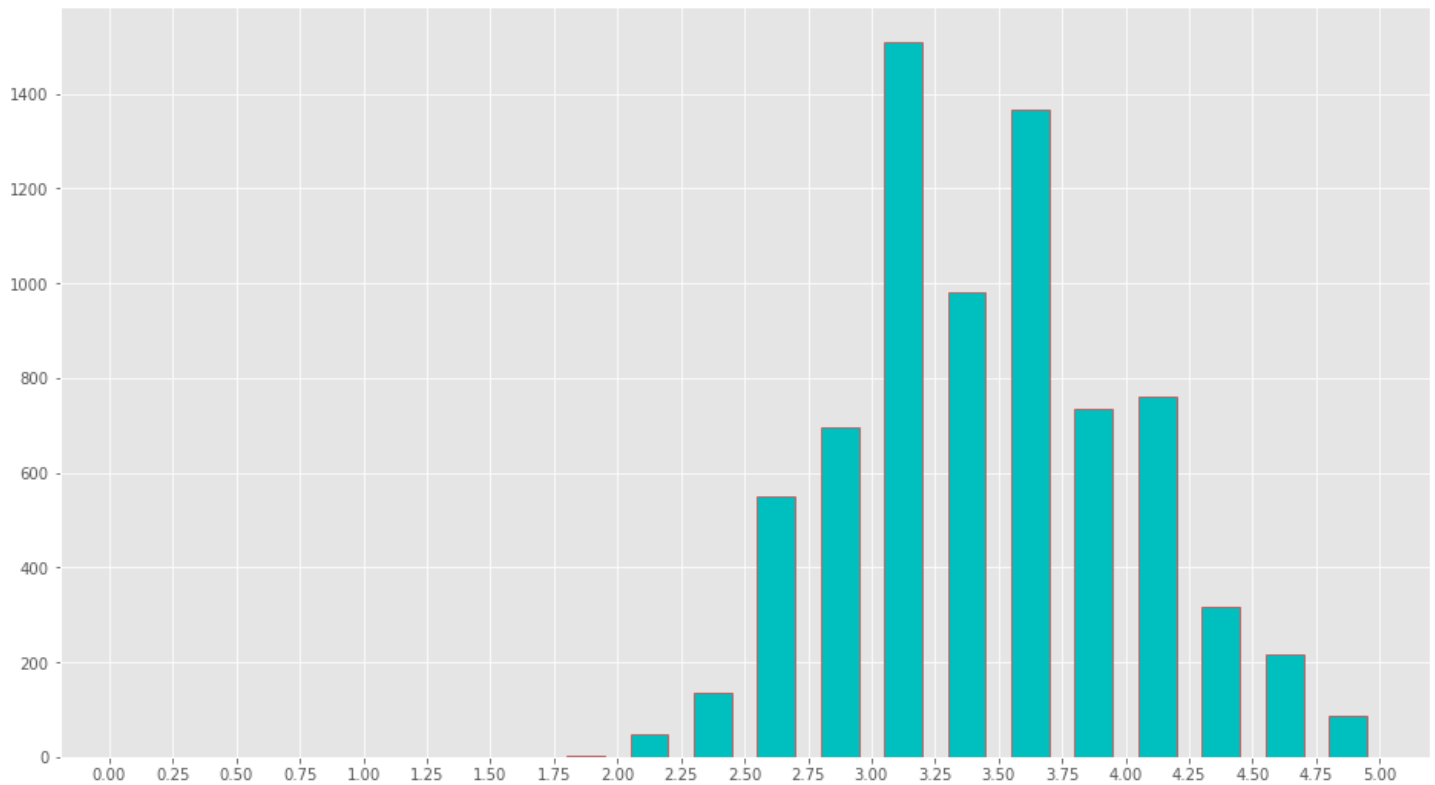
```
bins = np.arange(0,5.1,0.25)
```

```
# xt = [1,2,3,4,5,6,7,8,9,12,34,56,67]
```

```
plt.hist(x,color="c",rwidth = 0.6, edgecolor = "r", bins = bins)
```

```
plt.xticks(bins)
```

```
plt.show()
```



Task 3 :

bar graph : top 10 restaurants in the data with the highest number of votes.

*# Lets fetch Votes from "Votes" column and Restaurant name from "Restaurant Name" column of df dataframe
Then we will find top 10 voted restaurants*

```
votes = df["Votes"].copy()
votes.fillna(0, inplace = True)
votes = votes.values
```

```
rname = df["Restaurant Name"].values
barrest = []
barvotes = []
for i in range(10):
    ind = np.argmax(votes)
    barrest.append(rname[ind])
    barvotes.append(votes[ind])
    votes[ind] = 0
```

*# There is one restaurant which has two occurrences in Top 10 with different locations
So this would create problem in making Bar graph need to adjust it by concatenating 2 in string*

```
for i in range(len(barrest)):
    for j in range(len(barrest)):
        if i == j:
            continue
```

```

else:
    if barrest[i] == barrest[j]:
        barrest[j] = barrest[j] + "2"

```

Now we will plot the bar graph

```

def labels(barrest, barvotes):
    for i in range(len(barrest)):
        plt.text(i, barvotes[i]//2, barvotes[i], ha='center', Bbox=dict(facecolor='white', alpha=0.3))

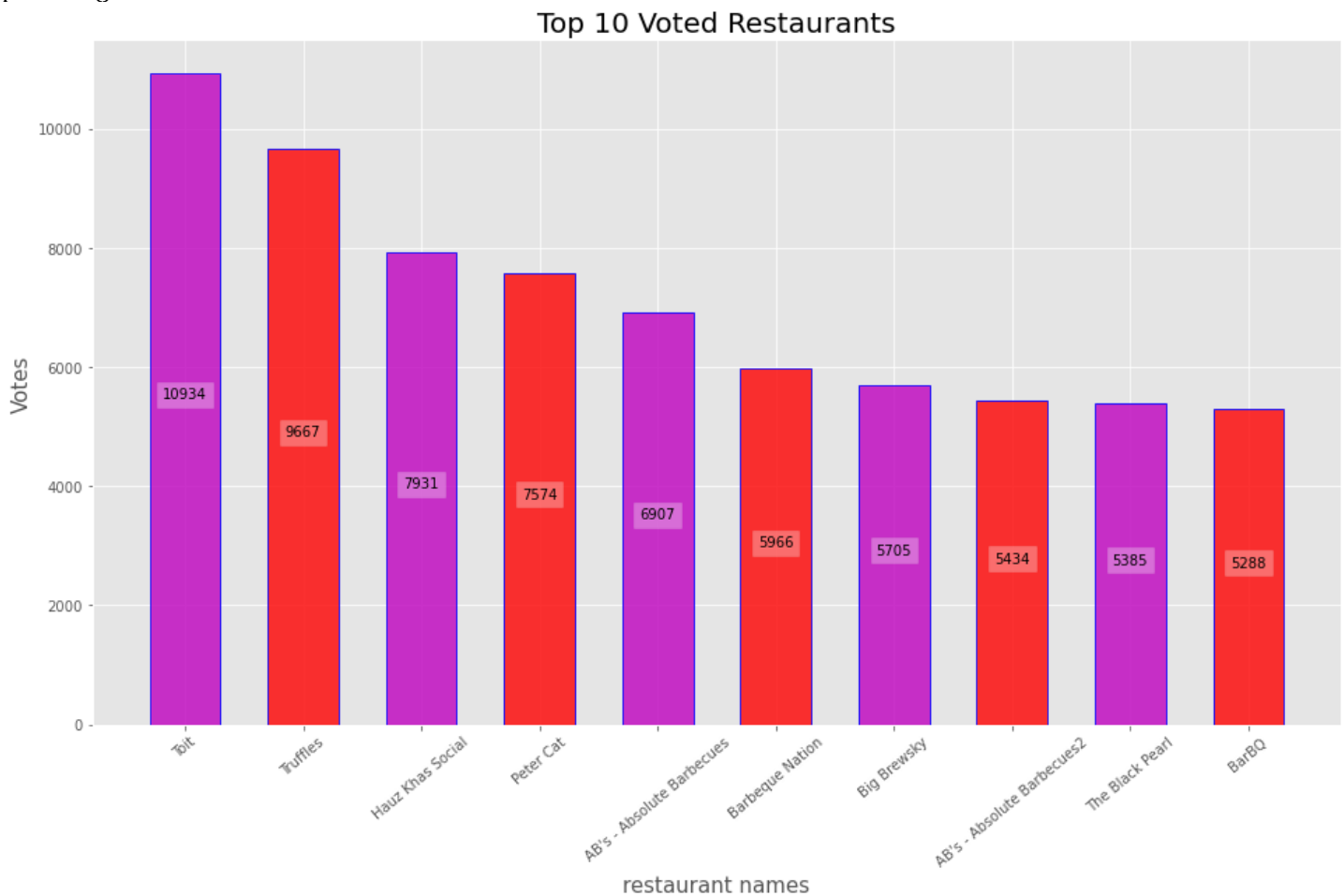
```

```

style.use("ggplot")
plt.figure(figsize = (16, 9))
plt.bar(barrest, barvotes, width=0.6, color=['m', 'r'], edgecolor="blue", linewidth=1, alpha=0.8)
labels(barrest, barvotes)
plt.xlabel("restaurant names", fontsize = 15)
plt.ylabel("Votes", fontsize = 15)
plt.xticks(rotation=40)
plt.title("Top 10 Voted Restaurants", fontsize = 20)

```

```
plt.show()
```



Task 4 :

pie graph : top 10 cuisines present in restaurants in the USA.

```

dfusa = df[df["Country Code"] == 216].copy()
cuisineus = dfusa["Cuisines"].copy()
cuisineus.fillna("xx", inplace=True)
cuisineus = cuisineus.values
cusi = []
for i in range(cuisineus.size):
    if "xx" not in cuisineus[i]:
        cusi.append(cuisineus[i])
    else:
        temp = cuisineus[i].split(", ")
        for j in temp:
            cusi.append(j)

dc = {}

for i in range(len(cusi)):
    if cusi[i] in dc:
        dc[cusi[i]] = dc[cusi[i]] + 1
    else:
        dc[cusi[i]] = 1

ucuisine = dc.keys()
ucuisine = list(ucuisine)
ucuisine = np.array(ucuisine)

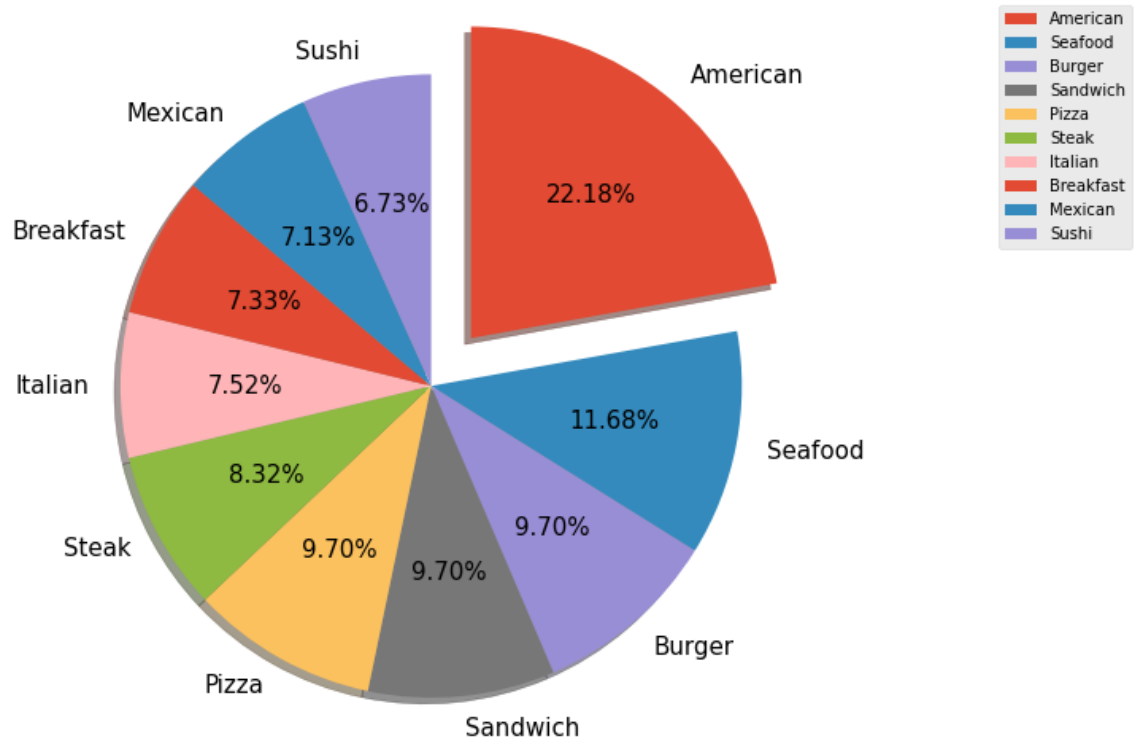
ucount = dc.values()
ucount = list(ucount)
ucount = np.array(ucount)

piecuisine = []
piecount = []

for i in range(10):
    ind = np.argmax(ucount)
    piecuisine.append(ucuisine[ind])
    piecount.append(ucount[ind])
    ucount[ind] = 0

explode = [0.2, 0, 0, 0, 0, 0, 0, 0, 0, 0]
textprops = {"fontsize": 15}
plt.figure(figsize = (16, 9))
plt.axis("equal")
plt.pie(piecount, labels=piecuisine, explode = explode, shadow = True, textprops = textprops, autopct = "%0.2f%%",
        , counter-clock = False, startangle = 90)
plt.legend()
plt.show()

```



Task 5 :

bubble graph : number of Restaurants present in the city of India and keeping the weighted restaurant rating of the city in a bubble.

Firstly we will fetch the required columns from dataframe which we made using provided CSV

i.e "Votes", "Aggregate rating" and get vr that is multiplication of both in separate variable

also we will take "Locality" from dataframe itself into l

```
votes = dfIndia["Votes"].copy()
```

```
votes.fillna(0, inplace = True)
```

```
votes = votes.values
```

```
r = dfIndia["Aggregate rating"].copy()
```

```
r.fillna(0, inplace = True)
```

```
r = r.values
```

```
vr = votes * r
```

```
c = dfIndia["City"].values
```

*# Now we will make a dictionary containing indian city as key and sum of Votes*Rating as Values*

```
d = {}
```

```
for i in range(c.size):
```

```
    if c[i] in d:
```

```
        d[c[i]] = d[c[i]] + vr[i]
```

```
    else:
```

```
        d[c[i]] = vr[i]
```

```
# print(d)
```

```
# Now we will make another dictionary which also has indian city as its keys and sum of votes as its values
```

```
d2 = {}  
for i in range(c.size):  
    if c[i] in d2:  
        d2[c[i]] = d2[c[i]] + votes[i]  
    else:  
        d2[c[i]] = votes[i]
```

```
# We ll make 3rd dictionary with indian city as key and its count which will depict no. of restaurants as values
```

```
d3 = {}  
for i in range(c.size):  
    if c[i] in d3:  
        d3[c[i]] = d3[c[i]] + 1  
    else:  
        d3[c[i]] = 1
```

```
# Now we will take 1'st dictionary keys as unique indian cities
```

```
# 1'st dictionary value i.e sum of Votes*Rating in vr
```

```
# 2'nd dictionary values i.e sum of votes in v
```

```
# 3'rd dictionary values as no. of restaurants in cities in ur
```

```
uc = d.keys()  
uc = list(uc)  
uc = np.array(uc)
```

```
uvr = d.values()  
uvr = list(uvr)  
uvr = np.array(uvr)
```

```
uv = d2.values()  
uv = list(uv)  
uv = np.array(uv)
```

```
ur = d3.values()  
ur = list(ur)  
ur = np.array(ur)
```

```
# Now we will traverse over size of uc i.e unique cities
```

```
# calculate Weighted Restaurant Rating =  $\Sigma (\text{number of votes} * \text{rating}) / \Sigma (\text{number of votes})$ 
```

```
# append in wrr empty list
```

```
# as division by 0 is not possible we have included condition for that
```

```
wrr = []  
for i in range(uc.size):  
    if uv[i] != 0:  
        temp = uvr[i]/uv[i]
```

```

wrr.append(temp)
else:
wrr.append(0)

wrr = np.array(wrr,dtype=float)

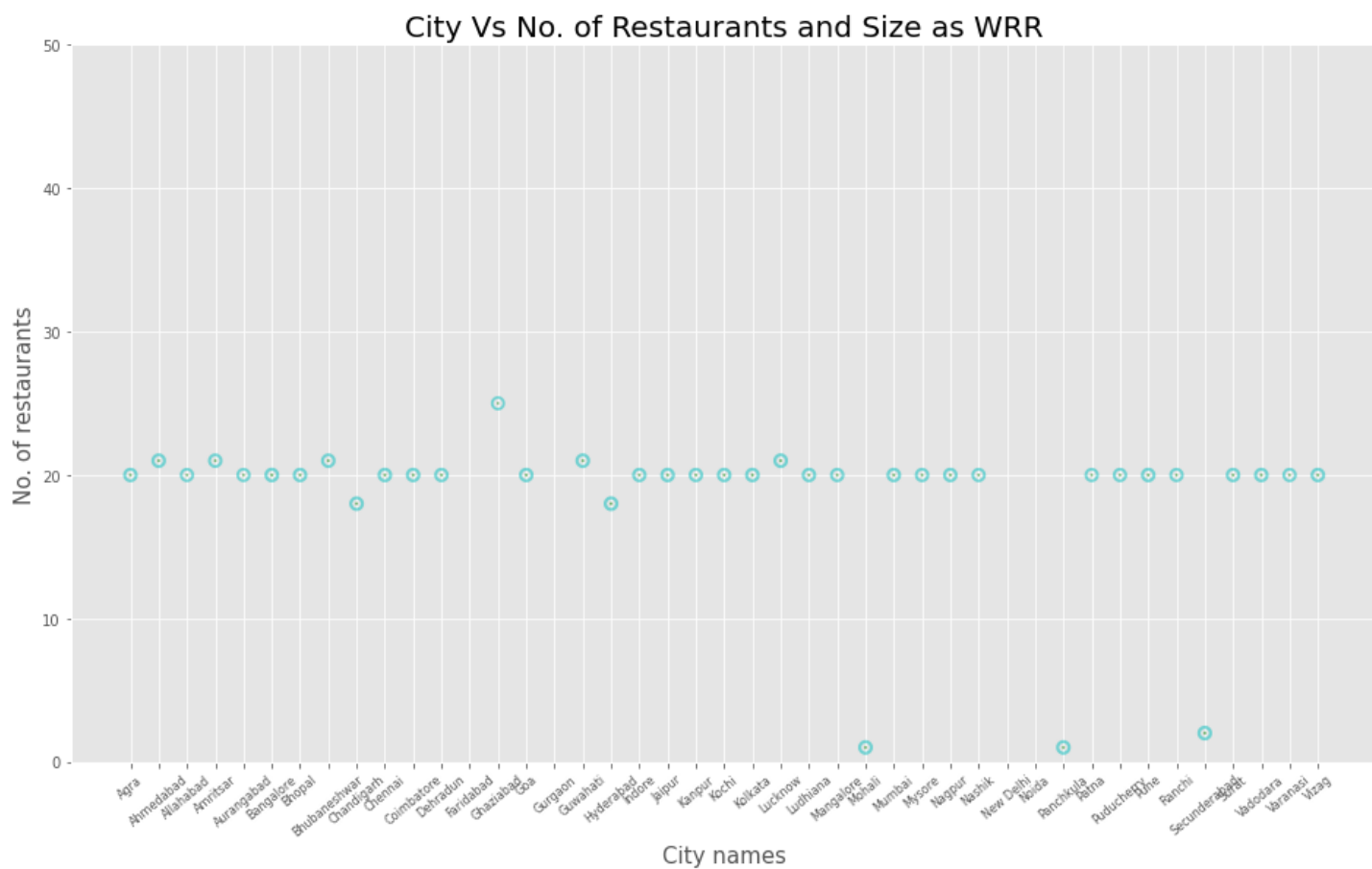
# we will run loop for unique city length times in which:
# append each corresponding values in empty list to plot graph

graphcity = []
graphwrr = []
graphrest = []
for i in range(len(uc)):
    graphcity.append(uc[i])
    graphwrr.append(wrr[i])
    graphrest.append(ur[i])
graphcity = np.array(graphcity)
graphwrr = np.array(graphwrr)
graphrest = np.array(graphrest)

x = graphcity
y = graphrest
size = graphwrr
style.use("ggplot")
plt.figure(figsize = (16, 9) )
plt.scatter(x, y,s=size, c="g", marker="o", alpha = 0.5, edgecolor = "c", linewidth = 8)
plt.xlabel("City names", fontsize = 15)
plt.ylabel("No. of restaurants", fontsize = 15)
plt.title("City Vs No. of Restaurants and Size as WRR", fontsize = 20)
plt.xticks(rotation = 40, fontsize = 8)
axes = plt.gca()
axes.set_ylim([0,50])

plt.show()

```

we have set y axis as 0 to 50 , as there were less cities having no. of restaurants over this limit

size of bubble is not pretty visible in change as WRR value is basically in range of 3.5 to 4.9 only