Que 1:Find five recipes that you like the most from any of the online repositories.

(a) Extract and represent the information from them in the traditional form (ingredients and cooking instructions); example below.

In [1]:
```
pip install requests
```

```
Requirement already satisfied: requests in c:\users\lakshya\anaconda3\lib\sit
e-packages (2.24.0)
Requirement already satisfied: idna<3,>=2.5 in c:\users\lakshya\anaconda3\lib
\site-packages (from requests) (2.10)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in
c:\users\lakshya\anaconda3\lib\site-packages (from requests) (1.25.9)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\lakshya\anacond
a3\lib\site-packages (from requests) (2020.6.20)
Requirement already satisfied: chardet<4,>=3.0.2 in c:\users\lakshya\anaconda
3\lib\site-packages (from requests) (3.0.4)
Note: you may need to restart the kernel to use updated packages.
```

In [2]:
```
import requests
import nltk
from nltk.tokenize import word_tokenize
from nltk.tag import pos_tag
import nltk
nltk.download('punkt')
nltk.download('averaged_perceptron_tagger')
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\lakshya\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping tokenizers\punkt.zip.
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     C:\Users\lakshya\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping taggers\averaged_perceptron_tagger.zip.
```

Out[2]:
```
True
```

In [3]:
```
recipie_ingr={}
for i in ['recipie-1','recipie-2','recipie-3','recipie-4','recipie-5']:
  recipie_ingr[i]=[]
```

In [4]:
```
from bs4 import BeautifulSoup
print('ANSWER 1-(a)')
def rec(url,ll):
  r=requests.get(url)
  htmlContent=r.content
  soup=BeautifulSoup(htmlContent,'html.parser')
  content = soup.find('div', {"class": "ingredients"})
  print("INGREDIENT-SECTION:-")
  for i in content.findAll('li'):
    s=i.text
    print(s)
    snew = nltk.word_tokenize(s)
    snew = nltk.pos_tag(snew)
    for i in snew:
          if i[1]=='NN' and i[0] not in ['cup','tsp','extract']:
               recipie_ingr[ll].append(i[0])
  content = soup.find('div', {"class": "method stylNew"})
  print("\n\nCOOKING INSTRUCTIONS:-")
  for i in content.findAll('li'):
    s=i.text
    print(s)

url_list=['https://food.ndtv.com/recipe-besan-ke-laddoo-954371','https://food.
ndtv.com/recipe-makai-ki-galvani-954528','https://food.ndtv.com/recipe-nei-app
am-953892'
,'https://food.ndtv.com/recipe-chocolate-appo-952959','https://food.ndtv.com/r
ecipe-gehun-ki-kheer-952570']
recipie_list=['\n\nrecipie-1','\n\nrecipie-2','\n\nrecipie-3','\n\nrecipie-4',
'\n\nrecipie-5']
li=['recipie-1','recipie-2','recipie-3','recipie-4','recipie-5']
for i in range(0,len(url_list)):
  print(recipie_list[i])
  rec(url_list[i],li[i])
  print('-------------------------------------------------------------------
--------------------------')
```

ANSWER 1-(a)

recipie-1
INGREDIENT-SECTION:-
2 Cups gram flour (coarse)
6-8 Cashew nuts
6-8 Almonds
1/2 cup ghee
1/2 tsp green cardamom powder
1/3 cup low-calorie sweetener

COOKING INSTRUCTIONS:-
1.Coarsely grind together the cashew nuts and almonds, and set aside.
2.Melt the ghee in a non-stick kadai. Add the gram flour and sauté on low hea
t till the gram flour is light brown and fragrant. This normally takes around
15-20 minutes.
3.Add the cashew nuts, almonds and cardamom powder, stir to mix and take the
kadai off the heat. Set aside to cool.
4.Add the low-calorie sweetener and mix well with your hands.
5.Shape into walnut-sized balls and store in an airtight container when compl
etely cooled.

---

recipie-2
INGREDIENT-SECTION:-
100 Ml ghee
200 gms makki flour
300 gms jaggery
2 litre water
5 gram cardamom powder

COOKING INSTRUCTIONS:-
1.Mix water and jaggery in a vessel and boil it. When the jaggery dissolves c
ompletely, filter it and keep it aside.
2.Heat the ghee in a kadhai and add makki flour to it. Cook on low flame unti
l it becomes golden.
3.After the flour is roasted, add jaggery water to it. Keep stirring it so th
at there are no lumps. After a boil, cook for two more minutes.
4.After this, add cardamom powder mix well and serve hot.

---

recipie-3
INGREDIENT-SECTION:-
1 Cup raw rice
3/4 cup jaggery
2 small banana
1/2 baking soda
1 tbsp coconut, grated
1/2 tsp cardamom powder
  ghee/oil

COOKING INSTRUCTIONS:-
1.Soak rice in water for 3 hours.
2.Mix jaggery with water and heat it up for a few minutes till the jaggery me lts. Strain the syrup keep it aside.
3.Drain the water from the rice and grind it with the jaggery syrup to a fine paste. Add water if needed.
4.Add the banana to the rice and jaggery paste and pulse it once or twice til l the banana is mixed well with the paste and pour the mixture in a bowl.
5.Check for the consistency. It should be slightly thicker than the dosa batt er.
6.Add the cardamom powder and the grated coconut to the batter and mix well.
7.Set the batter aside for at least 2 hours for fermentation.
8.While frying, add the baking soda to the batter and mix well.
9.Heat appam pan, add a tsp of ghee / oil in each hole. Pour a spoon full of the batter in each hole. When cooked on one side, carefully flip to other sid e, add oil / ghee if required.
10.Cook until both sides are dark brown. It’s ready to be served.
---

recipie-4
INGREDIENT-SECTION:-
3/4 Cup rice
3 tbsp coconut, grated
1 tsp baking powder
1 tsp vanilla extract
1/2 tsp cinnamon powder
1/4 cup brown sugar
1/4 cup unsweetened cocoa powder
A pinch of salt
3 tbsp chocolate chips
2-3 tbsp butter

COOKING INSTRUCTIONS:-
1.Wash and soak the rice in water for 30 minutes to 1 hour.
2.Drain the water of the rice completely and grind it.
3.Keep the grated coconut ready and add water to form a medium thick batter. Make sure the consistency of the batter is similar to any dosa / pancake batt er.
4.Transfer this to a bowl and add vanilla extract, cinnamon powder, sugar, co coa powder, salt and mix it well.
5.Keep this batter aside for 20 to 30 minutes.
6.Before using it, add baking powder to the batter and mix well.
7.Heat appam pan and add little butter / ghee / oil into all the moulds.
8.Pour a teaspoon of appo batter in each mold, add 3 to 4 chocolate chips and top it with another teaspoon of batter.
9.Cover the pan with a lid to allow it to cook on medium flame for 2 minutes.
10.When the edges are cooked, flip the appos, add some butter and cook on the other side for another 2 minutes.
11.Repeat the process for remaining batter.
12.To make the recipe look more interesting you can dip some appos in chocola te powder or semolina suji or coconut powder.
13.Chocolate Appo is ready to serve.

```
--------------------------------------------------------------------------------
-------------------


recipie-5
INGREDIENT-SECTION:-
150  Broken wheat
50  Ghee
50  Raisin
2  Cardamom powder
50 Ml Jaggery
100 Gram Cashew nut
500 Ml Milk


COOKING INSTRUCTIONS:-
1.Heat ghee in a thick bottom non-stick pan; add lapse and sauté till translu
cent and fragrant.
2.Add raisins and sauté on high heat. Add green cardamom powder and cashew nu
t and mix well.
3.Add jaggery, mix well and cook for a minute. Followed by adding a cup of wa
ter, stir continuously and cook for 3-4 minutes.
4.Reduce heat, add milk, gradually and keep stirring till well blended. Serve
hot.
--------------------------------------------------------------------------------
-------------------
```

(b) Further, store the recipes in the form of a (Recipe ID)—(Ingredient Name) form.

In [73]:
```
recipie_ingr_all=[]
x=0
for i in recipie_ingr.keys():
  recipie_ingr_all.append([])
  for j in recipie_ingr[i]:
    print(i,':',j)
    recipie_ingr_all[x].append((i,' : ',j))
  x+=1
```

```
recipie-1 : flour
recipie-1 : coarse
recipie-1 : ghee
recipie-1 : cardamom
recipie-1 : powder
recipie-1 : sweetener
recipie-2 : ghee
recipie-2 : jaggery
recipie-2 : water
recipie-2 : cardamom
recipie-2 : powder
recipie-3 : rice
recipie-3 : jaggery
recipie-3 : banana
recipie-3 : baking
recipie-3 : soda
recipie-3 : tbsp
recipie-3 : coconut
recipie-3 : cardamom
recipie-3 : powder
recipie-3 : ghee/oil
recipie-4 : rice
recipie-4 : tbsp
recipie-4 : coconut
recipie-4 : baking
recipie-4 : powder
recipie-4 : vanilla
recipie-4 : cinnamon
recipie-4 : powder
recipie-4 : sugar
recipie-4 : cocoa
recipie-4 : powder
recipie-4 : pinch
recipie-4 : salt
recipie-4 : chocolate
recipie-4 : tbsp
recipie-4 : butter
recipie-5 : wheat
recipie-5 : powder
recipie-5 : nut
recipie-5 : Milk
```

In [74]:
```
print('ANSWER-1(b)')
recipie_ingr_all
```

```
ANSWER-1(b)
```

Out[74]:
```
[[('recipie-1', ' : ', 'flour'),
  ('recipie-1', ' : ', 'coarse'),
  ('recipie-1', ' : ', 'ghee'),
  ('recipie-1', ' : ', 'cardamom'),
  ('recipie-1', ' : ', 'powder'),
  ('recipie-1', ' : ', 'sweetener')],
 [('recipie-2', ' : ', 'ghee'),
  ('recipie-2', ' : ', 'jaggery'),
  ('recipie-2', ' : ', 'water'),
  ('recipie-2', ' : ', 'cardamom'),
  ('recipie-2', ' : ', 'powder')],
 [('recipie-3', ' : ', 'rice'),
  ('recipie-3', ' : ', 'jaggery'),
  ('recipie-3', ' : ', 'banana'),
  ('recipie-3', ' : ', 'baking'),
  ('recipie-3', ' : ', 'soda'),
  ('recipie-3', ' : ', 'tbsp'),
  ('recipie-3', ' : ', 'coconut'),
  ('recipie-3', ' : ', 'cardamom'),
  ('recipie-3', ' : ', 'powder'),
  ('recipie-3', ' : ', 'ghee/oil')],
 [('recipie-4', ' : ', 'rice'),
  ('recipie-4', ' : ', 'tbsp'),
  ('recipie-4', ' : ', 'coconut'),
  ('recipie-4', ' : ', 'baking'),
  ('recipie-4', ' : ', 'powder'),
  ('recipie-4', ' : ', 'vanilla'),
  ('recipie-4', ' : ', 'cinnamon'),
  ('recipie-4', ' : ', 'powder'),
  ('recipie-4', ' : ', 'sugar'),
  ('recipie-4', ' : ', 'cocoa'),
  ('recipie-4', ' : ', 'powder'),
  ('recipie-4', ' : ', 'pinch'),
  ('recipie-4', ' : ', 'salt'),
  ('recipie-4', ' : ', 'chocolate'),
  ('recipie-4', ' : ', 'tbsp'),
  ('recipie-4', ' : ', 'butter')],
 [('recipie-5', ' : ', 'wheat'),
  ('recipie-5', ' : ', 'powder'),
  ('recipie-5', ' : ', 'nut'),
  ('recipie-5', ' : ', 'Milk')]]
```

(c) In general, as well as specific to each recipe, comment on which aspects of the recipes are being lost in the process of coarse-graining the recipe data?

ANS:- In the process of Coarse-graining we loose the quantity or amount of ingrdient which we need to add in, and also their is no information about the procedure so the sequnce of adding ingredient may also change which may leads to result in completely different recipie.

Like in a scrapped recipies of "makai-ki-galvani" there is lot of ingredient like ghee, makki flour jaggery ,water and cardamom powder but when you don't know the quantity like 100 ml of ghee or 300 gm of jaggery ,and along with sequence of adding this ingredient like mix water with jaggery in a vessel and boil etc. like process if change then it may result in somewhat different recipie with different taste and look.

(d) How could one possibly mitigate this to extract the most details from the recipes?

ANS:- To store the recipies we can use data structure like list of dictionaruies or disctionary of disctionaries like below:-

below 1st dictionary have keys of recipies name and its value if another dictionary having two keys ingredinet and method whose values are in list formate with ingreident names and instructions.

{'recipie-01' : { 'ingredient' :['ingredient1','ingredient2',.....], 'method' :['instruction1','instruction2',.....] } ,'recipie-02' : { 'ingredient' :['ingredient1','ingredient2',.....], 'method' :['instruction1','instruction2',.....] } ,..... }

In [68]:

Que2:Obtain the data of recipes from Kaggle and analyze it for the following.

step1:Loading data from kaggle;

In [75]:
```
import numpy as np
import os
###https://www.geeksforgeeks.org/working-zip-files-python/
##import zip code from here

# importing required modules
from zipfile import ZipFile
# specifying the zip file name
file_name = "683_1293_bundle_archive.zip"
```

In [76]:
```
# opening the zip file in READ mode
with ZipFile('/content/drive/My Drive/683_1293_bundle_archive.zip', 'r') as zi
p:
    # printing all the contents of the zip file
    zip.printdir()

    # extracting all the files
    print('Extracting all the files now...')
    zip.extractall()
    print('Done!')
```

```
File Name                                             Modified             Si
ze
test.json                                     2019-09-20 05:54:40      28440
86
train.json                                    2019-09-20 05:54:40     124150
67
Extracting all the files now...
Done!
```

step2: Unzipped and load .json fromatted test and train data:

In [77]:
```
import json

# Opening JSON file and loading the data
# into the variable data
with open('train.json') as json_file:
    data_train = json.load(json_file)
```

In [78]:
```

# Opening JSON file and loading the data
# into the variable data
with open('test.json') as json_file:
    data_test = json.load(json_file)
```

In [ ]:
```
```

(a)number of recipies, number of unique ingredinet and number of cusines:

In [80]:
```
number_of_recipies_train=len(data_train)
```

In [81]:
```
number_of_recipies_test=len(data_test)
# number_of_recipies_test
```

In [82]:
```
unique_ingredient=[];
for i in data_train:
  unique_ingredient+=i['ingredients']
```

In [83]:
```
number_of_unique_ingredient=len(set(unique_ingredient))
```

In [84]:
```
cuisines=[]
for i in data_train:
  cuisines.append(i['cuisine'])
```

In [85]:
```
number_of_cuisines=len(set(cuisines))
print('ANSWER 2-(a)')
print('number of recipies- ',number_of_recipies_train)
print('number of unique ingredient- ',number_of_unique_ingredient)
print('number of cuisines-',number_of_cuisines)
```

```
ANSWER 2-(a)
number of recipies-  39774
number of unique ingredient-  6714
number of cuisines- 20
```

(b) BAR plot for number of recipies for each cuisines:

In [86]:
```
dict_cuisine_count_wrt_recipies={}
for i in set(cuisines):
  dict_cuisine_count_wrt_recipies[i]=0
```

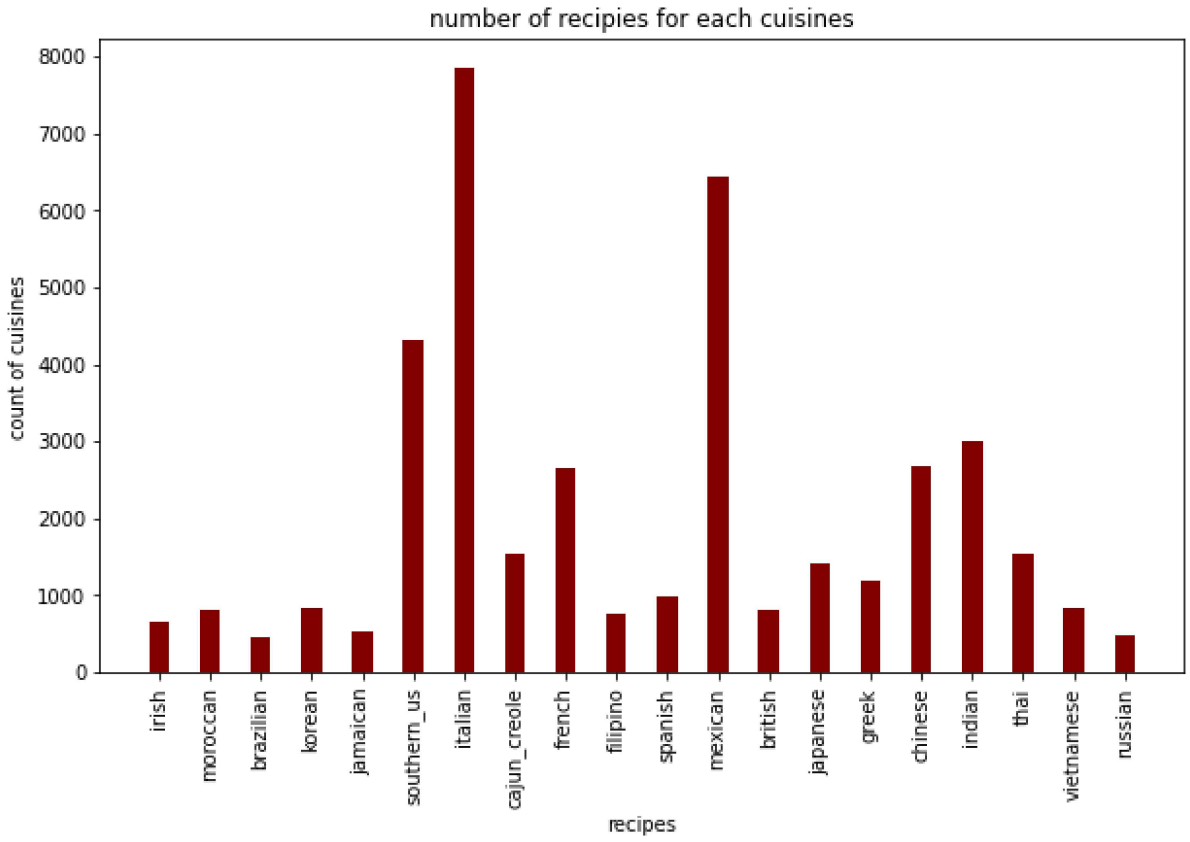
In [87]:
```
for i in data_train:
  dict_cuisine_count_wrt_recipies[i['cuisine']]+=1
```

In [88]:
```
import numpy as np
import matplotlib.pyplot as plt
rec = list(dict_cuisine_count_wrt_recipies.keys())
count = list(dict_cuisine_count_wrt_recipies.values())

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

# creating the bar plot
plt.bar(rec, count, color ='maroon',
                width = 0.4)

plt.xlabel("recipes")
plt.ylabel("count of cuisines")
plt.title("number of recipies for each cuisines")
plt.xticks(rotation=90)
plt.show()
```



(c)plot recipies size distribution for each cuisines as well as for all recipies.

In [89]:
```
cuisine_all_recipies_ingredient_count={}
for i in set(cuisines):
  cuisine_all_recipies_ingredient_count[i]={}
```

In [90]:
```
# for i in cuisine_all_recipies_ingredient_count.keys():
for i in data_train:
    if len(i['ingredients']) in cuisine_all_recipies_ingredient_count[i['cuisi
ne']].keys():
      cuisine_all_recipies_ingredient_count[i['cuisine']][len(i['ingredients'
])]+=1
    else :
      cuisine_all_recipies_ingredient_count[i['cuisine']][len(i['ingredients'
])]=1
```

In [91]:
```
for i in cuisine_all_recipies_ingredient_count.keys():
  cuisine_all_recipies_ingredient_count[i]={m: v for m,v in sorted(cuisine_all
_recipies_ingredient_count[i].items())}
```

In [92]:
```
import matplotlib
import math as m
import matplotlib.pyplot as plt
figure, axes=plt.subplots(figsize= (10,6))
for i in cuisine_all_recipies_ingredient_count.keys():
  x=list(cuisine_all_recipies_ingredient_count[i].keys())
  y=list(cuisine_all_recipies_ingredient_count[i].values())
  total=0
  for v in y:
    total+=v
  for i1 in range(0,len(y)):
    y[i1]=(y[i1]/total)*100
  axes.scatter(x,y)
  axes.plot(x,y,label=i)
axes.set_xlabel('RECIPE SIZE')
axes.set_ylabel('PERCENTAGE')
axes.set_title(" RECEPIES-SIZE DISTRIBUTION ACCORDING TO CUISINE")
plt.legend()
```

Out[92]: <matplotlib.legend.Legend at 0x7f16c356bc88>

recipies size distribution for all recipies

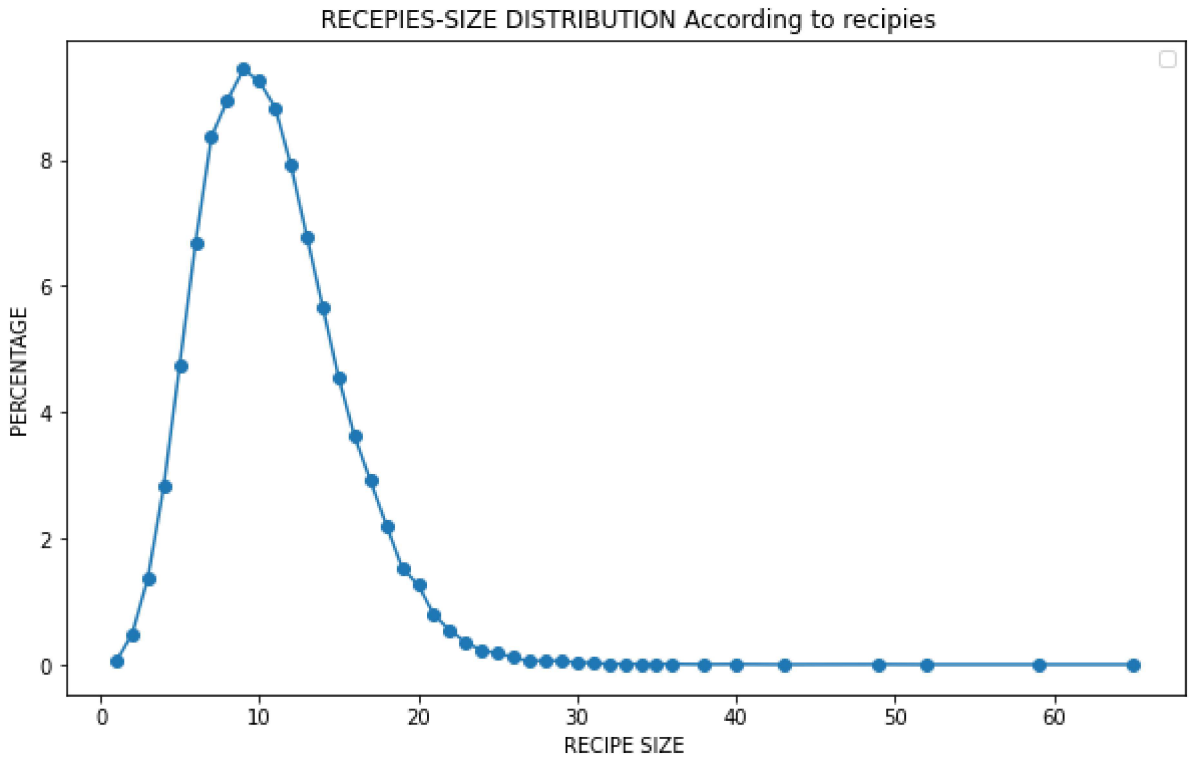
In [93]:
```
recipies={}
for i in cuisine_all_recipies_ingredient_count.keys():
    for j in cuisine_all_recipies_ingredient_count[i].keys():
      if j in recipies.keys():
        recipies[j]+=cuisine_all_recipies_ingredient_count[i][j]
      else:
        recipies[j]=cuisine_all_recipies_ingredient_count[i][j]
```

In [94]:
```
recipies={m: v for m,v in sorted(recipies.items())}
```

In [95]:
```
figure, axes=plt.subplots(figsize= (10,6))
x=list(recipies.keys())
y=list(recipies.values())
total=0
for i in y:
  total+=i
for i in range(0,len(y)):
    y[i]=(y[i]/total)*100
# print(x)
# print(y)
axes.scatter(x,y)
axes.plot(x,y)
axes.set_xlabel('RECIPE SIZE')
axes.set_ylabel('PERCENTAGE')
axes.set_title(" RECEPIES-SIZE DISTRIBUTION According to recipies")
plt.legend()
```

```
No handles with labels found to put in legend.
```

Out[95]:
```
<matplotlib.legend.Legend at 0x7f16c3e67e80>
```



(d) Plot cumulative distribution of recipe size and interpret.

ANS:- Interpret-

With Cummulative distribution which plot between percentage and recipie size we observe that the recipies size between 10 to 37 the scattering is alot so we can say that popularity of these recipie size is alot ,And recipies size of between 0 to 5 less dense then 10 to 37 so this recipies size are moderate in use and other then this like recipies sizes more then 40 or between 5 to 9 are least popular sizes for making recipies.

so the conculsion is most popular recipies having recipies size between 10 to 37 and other then that like more then 40 recipie size are of too much ingredient used very less in general.

In [96]:
```
recipies_cumulative={}
for i in recipies.keys():
  recipies_cumulative[i]=recipies[i]
```

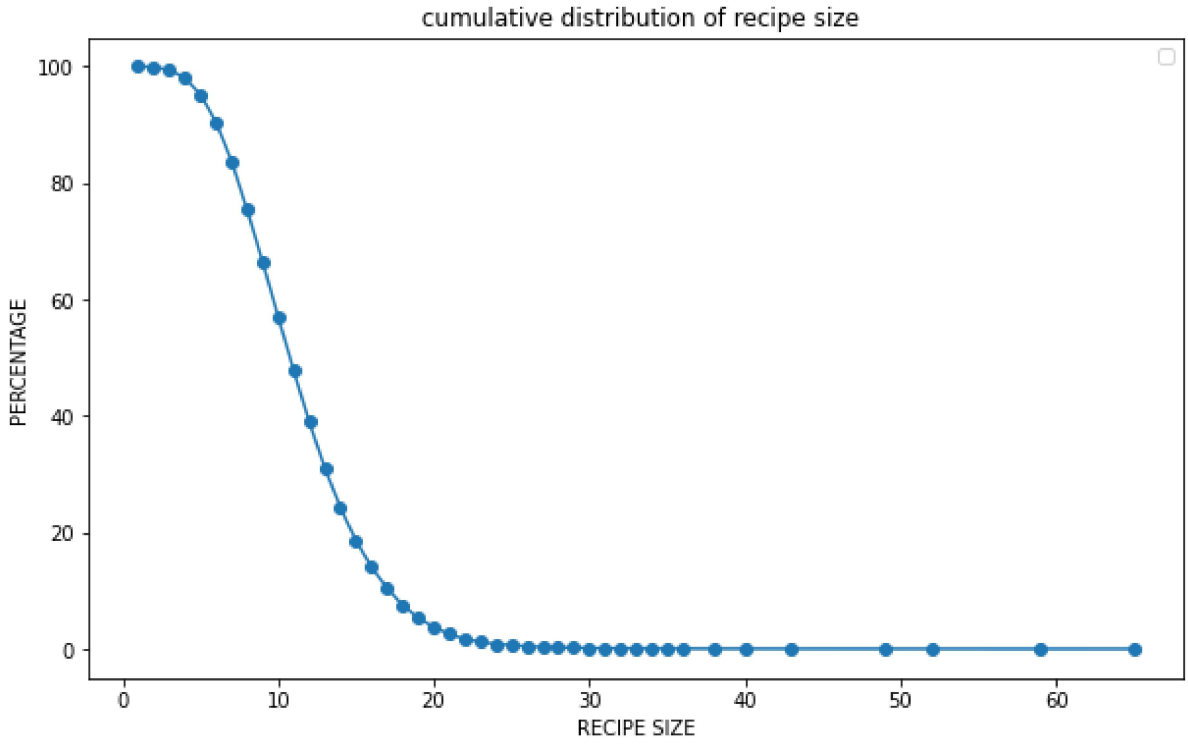
In [97]:
```
recipies_cumulative={m: v for m,v in sorted(recipies_cumulative.items())}
```

In [98]:
```
temp_list=list(recipies_cumulative.keys())
cum_sum=0
for i in range(0,len(temp_list)):
  cum_sum+=recipies_cumulative[temp_list[i]]
  for j in range(i+1,len(temp_list)):
    recipies_cumulative[temp_list[i]]+=recipies_cumulative[temp_list[j]]
```

In [100]:
```
figure, axes=plt.subplots(figsize= (10,6))
x=list(recipies_cumulative.keys())
y=list(recipies_cumulative.values())
for i in range(0,len(y)):
    y[i]=(y[i]/cum_sum)*100
# print(x)
# print(y)
axes.scatter(x,y)
axes.plot(x,y)
axes.set_xlabel('RECIPE SIZE')
axes.set_ylabel('PERCENTAGE')
axes.set_title(" cumulative distribution of recipe size")
plt.legend()
print(x)
print(y)
```

```
No handles with labels found to put in legend.

[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 2
2, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 38, 40, 43, 49, 5
2, 59, 65]
[100.0, 99.94468748428622, 99.45944586916076, 98.0791471815759, 95.2431236486
1467, 90.48876150248907, 83.79594710112134, 75.4261577915221, 66.485643887967
02, 57.049831548247596, 47.805098808266706, 38.97520993613919, 31.06552018906
8234, 24.282194398350683, 18.61768994820737, 14.069492633378589, 10.451551264
645245, 7.535073163373057, 5.325086740081461, 3.791421531653844, 2.5242620807
56273, 1.737315834464726, 1.1892190878463318, 0.834716146226178, 0.6059234675
918942, 0.4249006888922412, 0.3092472469452406, 0.2589631417508925, 0.1910795
9973852265, 0.13828128928445718, 0.10056821038869612, 0.0729119525318047, 0.0
6285513149293508, 0.05279831045406547, 0.04525569467491326, 0.037713078895761
05, 0.027656257856891436, 0.02262784733745663, 0.01508523155830442, 0.0125710
26298587015, 0.00754261577915221, 0.0050284105194348064, 0.002514205259717403
2]
```



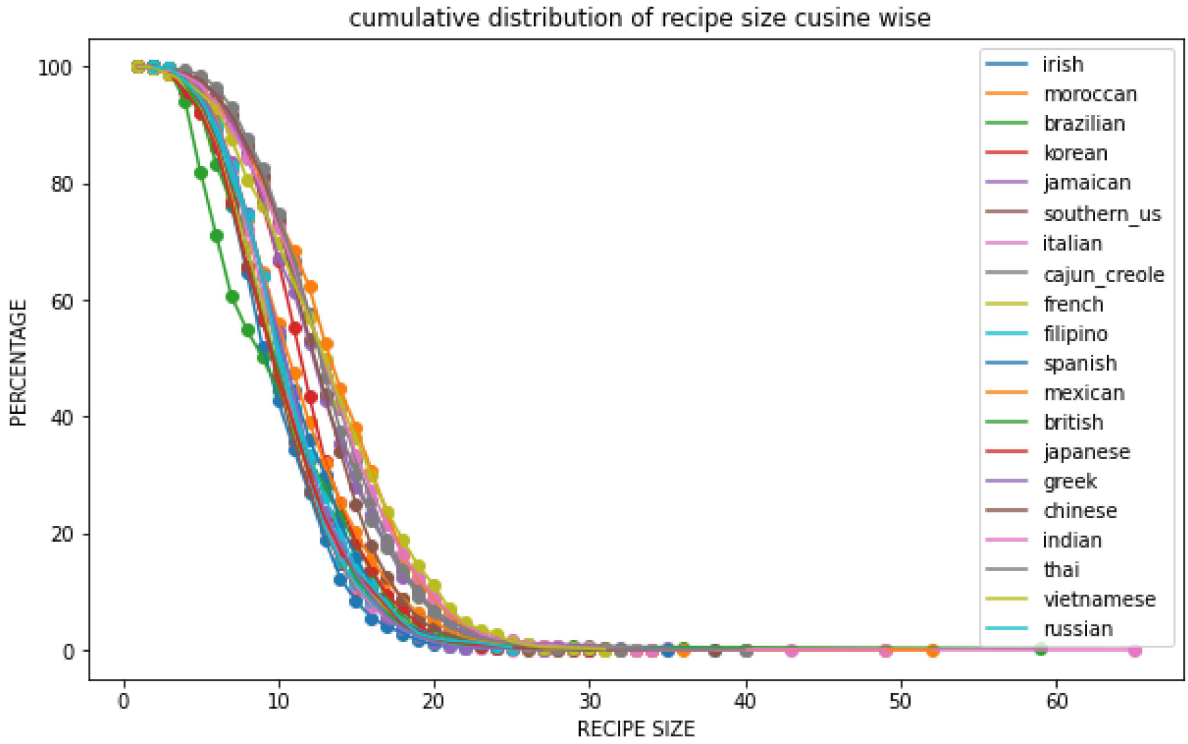
In [101]:
```
cuisine_all_recipies_ingredient_count_cumulative={}
for i in cuisine_all_recipies_ingredient_count.keys():
  cuisine_all_recipies_ingredient_count_cumulative[i]=cuisine_all_recipies_ing
redient_count[i]
# print(cuisine_all_recipies_ingredient_count_cumulative)
```

In [102]:
```
for i in cuisine_all_recipies_ingredient_count_cumulative.keys():
  cuisine_all_recipies_ingredient_count_cumulative[i]={m: v for m,v in sorted(
cuisine_all_recipies_ingredient_count_cumulative[i].items())}
```

In [103]:
```
list_cum_sum=[]
for i in cuisine_all_recipies_ingredient_count_cumulative.keys():
  temp_list=list(cuisine_all_recipies_ingredient_count_cumulative[i].keys())
  cum_sum=0
  for i1 in range(0,len(temp_list)):
    cum_sum+=cuisine_all_recipies_ingredient_count_cumulative[i][temp_list[i1
]]
    for j in range(i1+1,len(temp_list)):
      cuisine_all_recipies_ingredient_count_cumulative[i][temp_list[i1]]+=cuis
ine_all_recipies_ingredient_count_cumulative[i][temp_list[j]]
  list_cum_sum.append(cum_sum)
```

In [104]:
```
import matplotlib
import math as m
import matplotlib.pyplot as plt
figure, axes=plt.subplots(figsize= (10,6))
v=0
for i in cuisine_all_recipies_ingredient_count_cumulative.keys():
  x=list(cuisine_all_recipies_ingredient_count_cumulative[i].keys())
  y=list(cuisine_all_recipies_ingredient_count_cumulative[i].values())
  for i1 in range(0,len(y)):
    y[i1]=(y[i1]/list_cum_sum[v])*100
  axes.scatter(x,y)
  axes.plot(x,y,label=i)
  v+=1
axes.set_xlabel('RECIPE SIZE')
# plt.loglog()
axes.set_ylabel('PERCENTAGE')
axes.set_title(" cumulative distribution of recipe size cusine wise")
plt.legend()
```

Out[104]: <matplotlib.legend.Legend at 0x7f16c3dd3ef0>



Que3:

(a) Plot the frequency-rank distribution for all the recipes and interpret.

ANS- from frequncy rank distribution graph we observe that the graph is monotonically decreasing which intrepret as according to the most frequent ingredient like salt which is in total among all the cusines is having highest rank as here rank is given to frequency of ingredient among all the cuisines,and least frequent ingredient are having very low rank and are not used commonly in cusines.

In [105]:
```
global_unique_ingredient=list(set(unique_ingredient))
freq_rank_recipies={}
for i in global_unique_ingredient:
  freq_rank_recipies[i]=0
```

In [106]:
```
for i in global_unique_ingredient:
  for j in data_train:
    if i in j['ingredients']:
      freq_rank_recipies[i]+=1
```

In [107]:
```
freq_rank_recipies_sorted_desc = dict(sorted(freq_rank_recipies.items(), key=l
ambda x: x[1], reverse=True))
```

In [108]:
```
freq_rank_recipies_sorted_desc_RANK={}
c=1
for i in freq_rank_recipies_sorted_desc.keys():
  freq_rank_recipies_sorted_desc_RANK[c]=freq_rank_recipies_sorted_desc[i]
  c+=1
# freq_rank_recipies_sorted_desc_RANK
```

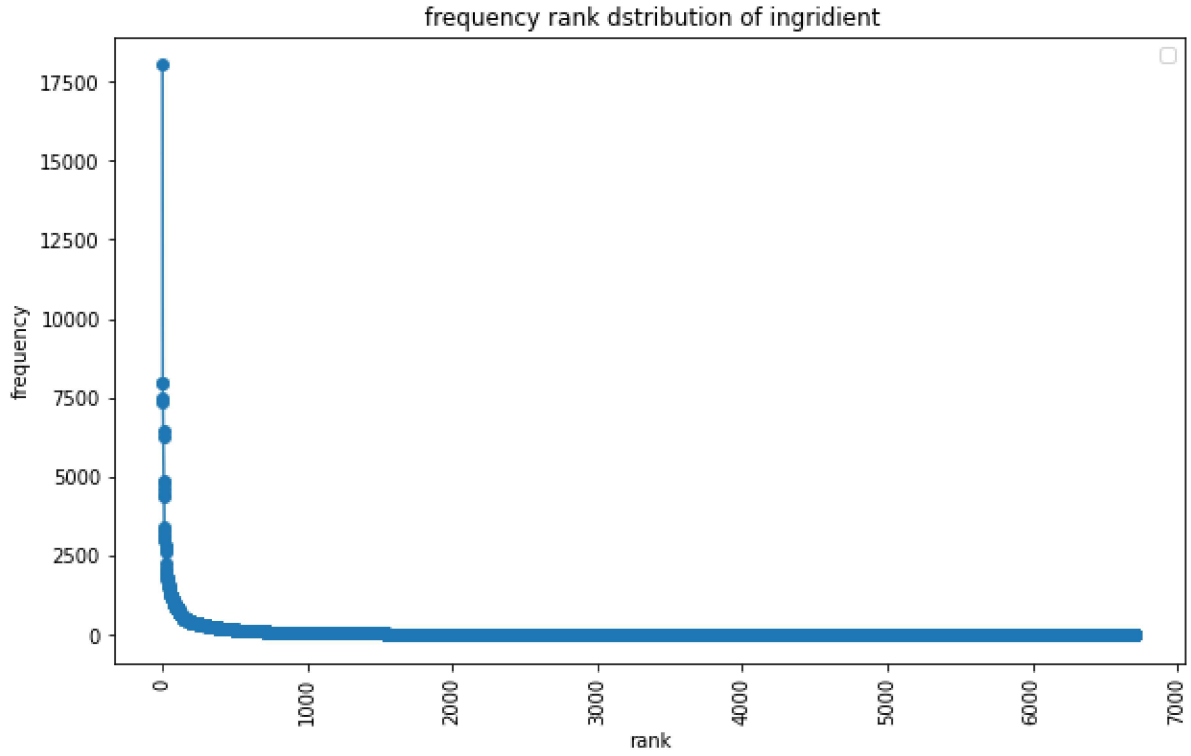
1.UN-NORMALIZED GRAPH

In [109]:
```
print('Un-normalized graph')
import math
figure, axes=plt.subplots(figsize= (10,6))
x=list(freq_rank_recipies_sorted_desc_RANK.keys())
y=list(freq_rank_recipies_sorted_desc_RANK.values())
a=[x1 for x1 in x]
b=[y1 for y1 in y]
axes.scatter(a,b)
axes.plot(a,b)
# plt.yscale('log');
# plt.xscale('log');
# plt.loglog()
axes.set_xlabel('rank')
axes.set_ylabel('frequency')
axes.set_title(" frequency rank dstribution of ingridient")
plt.xticks(rotation=90)
plt.legend()
```

```
No handles with labels found to put in legend.

Un-normalized graph
```

Out[109]: <matplotlib.legend.Legend at 0x7f16c3b9c4a8>



2.NORMALIZED GRAPH

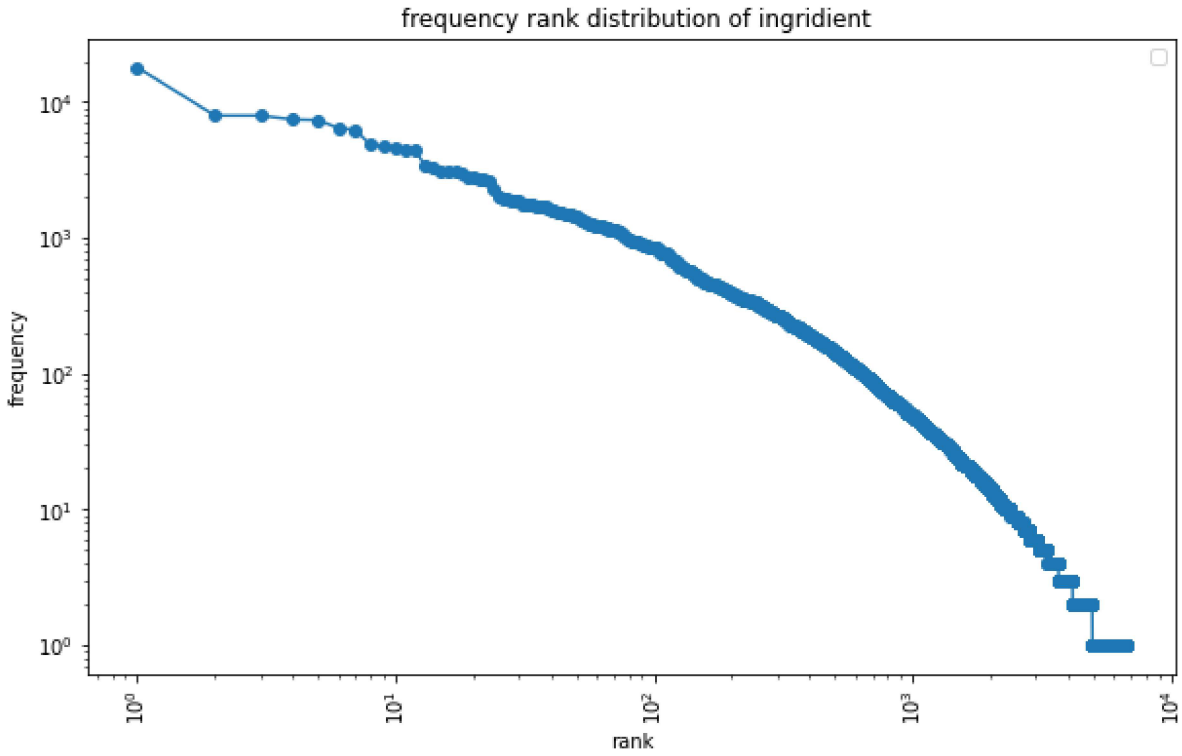
graph normalised with loglog(X,Y)

In [110]:
```
print('Normalized graph')
import math
figure, axes=plt.subplots(figsize= (10,6))
x=list(freq_rank_recipies_sorted_desc_RANK.keys())
y=list(freq_rank_recipies_sorted_desc_RANK.values())
a=[x1 for x1 in x]
b=[y1 for y1 in y]
axes.scatter(a,b)
axes.plot(a,b)
# plt.yscale('log');
# plt.xscale('log');
plt.loglog()
axes.set_xlabel('rank')
axes.set_ylabel('frequency')
axes.set_title(" frequency rank distribution of ingridient")
plt.xticks(rotation=90)
plt.legend()
```

```
No handles with labels found to put in legend.

Normalized graph
```

Out[110]: <matplotlib.legend.Legend at 0x7f16c3b5d128>



(b) 10 most popular ingrient.

In [111]:
```
most_popular_top10={}
count=0
for i in freq_rank_recipies_sorted_desc.keys():
  if count< 10:
    most_popular_top10[i]=freq_rank_recipies_sorted_desc[i]
    count+=1
  else:
    break
```

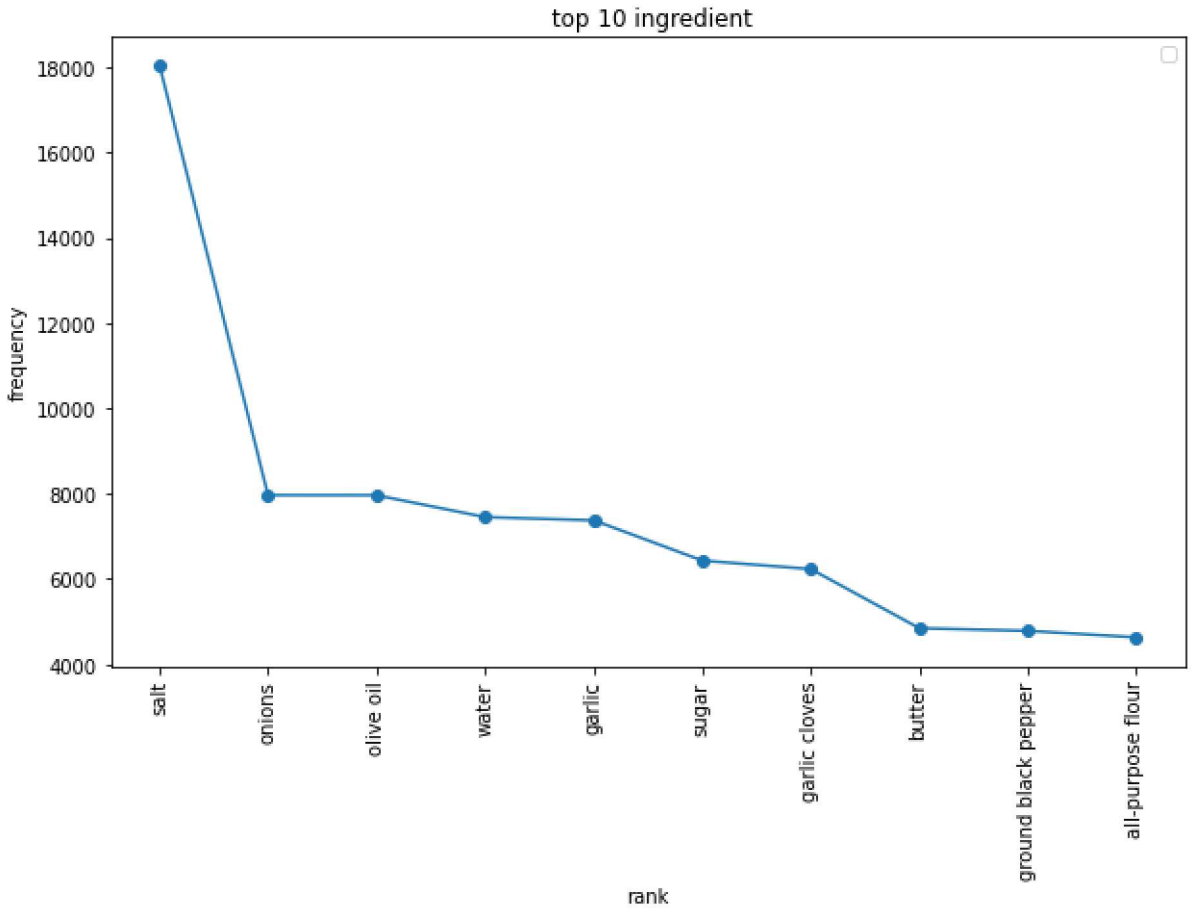
In [112]:
```
most_popular_top10= dict(sorted(most_popular_top10.items(), key=lambda x: x[1
], reverse=True))
print("10 most popular ingridients:-",list(most_popular_top10.keys()))
```

```
10 most popular ingridients:- ['salt', 'onions', 'olive oil', 'water', 'garli
c', 'sugar', 'garlic cloves', 'butter', 'ground black pepper', 'all-purpose f
lour']
```

In [113]:
```
figure, axes=plt.subplots(figsize= (10,6))
x=list(most_popular_top10.keys())
y=list(most_popular_top10.values())
axes.scatter(x,y)
axes.plot(x,y)
axes.set_xlabel('rank')
axes.set_ylabel('frequency')
axes.set_title(" top 10 ingredient")
plt.xticks(rotation=90)
plt.legend()
```

```
No handles with labels found to put in legend.
```

Out[113]:
```
<matplotlib.legend.Legend at 0x7f16c393fe80>
```



(c) Plot the ingredient-rank distribution for each of the cuisines and list the most popular ingredients for each cuisine.

ANS-

As above in a part of que3 we taken out rank distribution overall among all the cuisines whereas in this part we do that thing in cusines wise

for example- we pick up a cuisine -'irish' then pickup all unique ingredient of this cuisine then do counting for each ingredient of recipies of 'irish' cuisines ,then take out rank distribution graph indicate the most popular or frequent ingredient among irish having highest rank and least popular having lowest rank and so on for each cuisines .

In [114]:
```
cusine_unique_ingredient={}
for i in cuisine_all_recipies_ingredient_count.keys():
  cusine_unique_ingredient[i]=[]
```

In [115]:
```
for i in data_train:
  cusine_unique_ingredient[i['cuisine']]+=i['ingredients']
  cusine_unique_ingredient[i['cuisine']]=list(set(cusine_unique_ingredient[i[
'cuisine']]))
```

In [116]:
```
cusine_unique_ingredient_RANK={}
for i in cusine_unique_ingredient.keys():
  cusine_unique_ingredient_RANK[i]={}
  for j in cusine_unique_ingredient[i]:
    for k in data_train:
      if k['cuisine']==i and j in k['ingredients']:
        if j not in cusine_unique_ingredient_RANK[i]:
          cusine_unique_ingredient_RANK[i][j]=1
        else:
          cusine_unique_ingredient_RANK[i][j]+=1
```

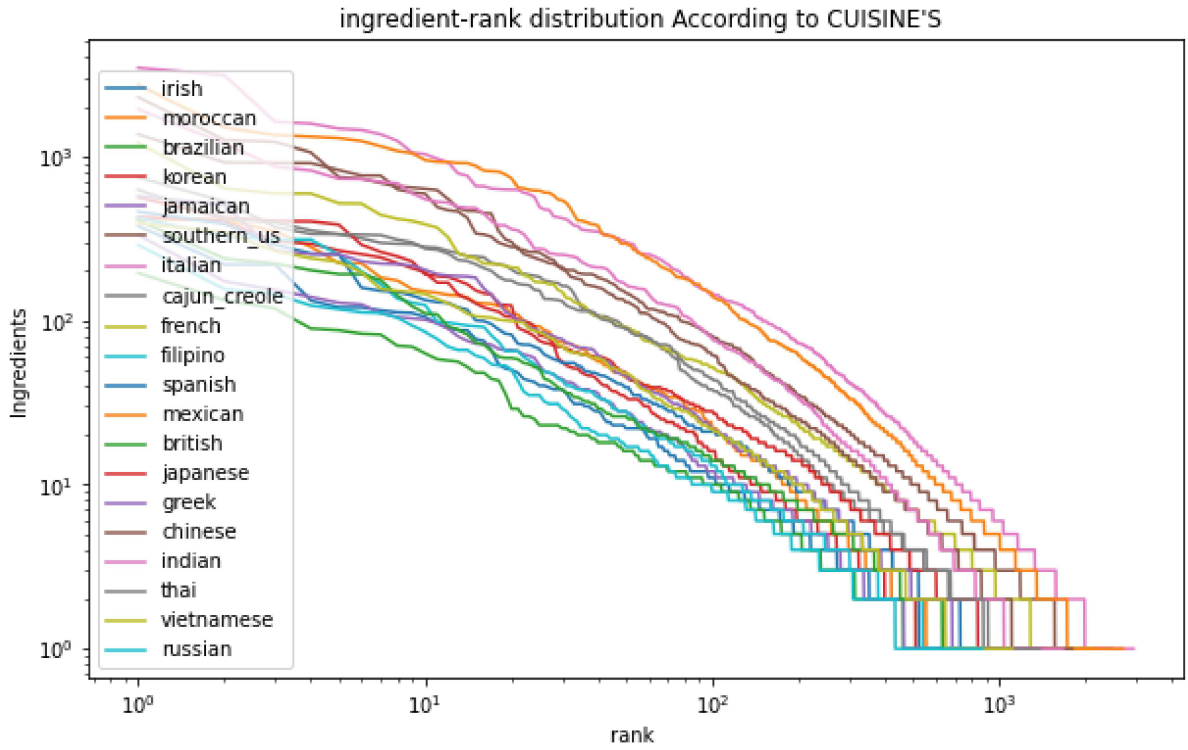
1.NORMALIZED GRAPH

graph normalised with loglog(x,y).

-TOP 10 INGREDIENT OF EACH CUISINE'S

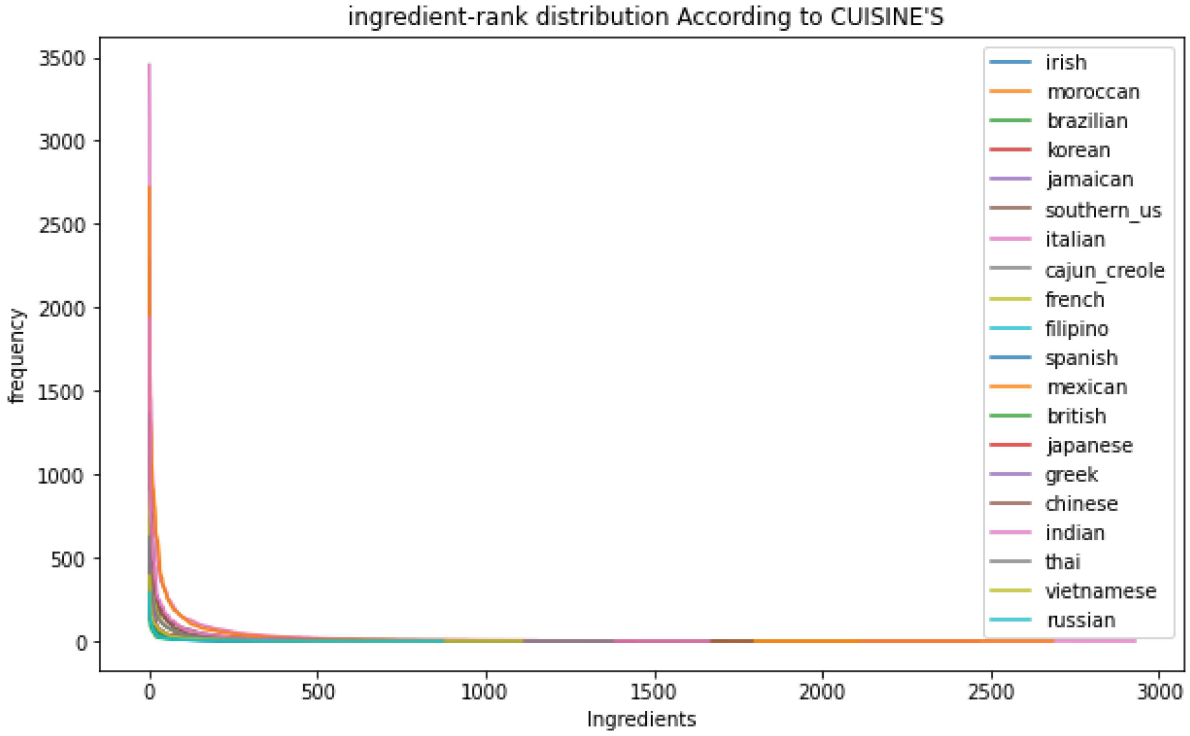
In [117]:
```
figure, axes =plt.subplots(figsize=(10,6))
for i in cusine_unique_ingredient_RANK.keys():
  cusine_unique_ingredient_RANK[i]=dict(sorted(cusine_unique_ingredient_RANK[i
].items(), key=lambda x: x[1], reverse=True))
  most_popular_top10={}
  count=0
  for i1 in cusine_unique_ingredient_RANK[i].keys():
    if count< 10:
      most_popular_top10[i1]=cusine_unique_ingredient_RANK[i][i1]
      count+=1
    else:
      break
  print(i,":",list(most_popular_top10.keys()))
  y=list(cusine_unique_ingredient_RANK[i].values())
  x=[h for h in range(1,len(y)+1)]
  a=[x1 for x1 in x]
  b=[y1 for y1 in y]
  axes.plot(a,b,label=i)
  plt.loglog()
  axes.set_xlabel('rank ')
  axes.set_ylabel('Ingredients')
  axes.set_title(" ingredient-rank distribution According to CUISINE'S")
  plt.legend()
```

```
irish : ['salt', 'all-purpose flour', 'butter', 'onions', 'sugar', 'potatoe
s', 'baking soda', 'baking powder', 'milk', 'carrots']
moroccan : ['salt', 'olive oil', 'ground cumin', 'onions', 'garlic cloves',
'ground cinnamon', 'water', 'ground ginger', 'carrots', 'paprika']
brazilian : ['salt', 'onions', 'olive oil', 'lime', 'water', 'garlic cloves',
'garlic', 'cachaca', 'sugar', 'tomatoes']
korean : ['soy sauce', 'sesame oil', 'garlic', 'green onions', 'sugar', 'sal
t', 'water', 'sesame seeds', 'onions', 'scallions']
jamaican : ['salt', 'onions', 'water', 'garlic', 'ground allspice', 'pepper',
'scallions', 'dried thyme', 'black pepper', 'garlic cloves']
southern_us : ['salt', 'butter', 'all-purpose flour', 'sugar', 'large eggs',
'baking powder', 'water', 'unsalted butter', 'milk', 'buttermilk']
italian : ['salt', 'olive oil', 'garlic cloves', 'grated parmesan cheese', 'g
arlic', 'ground black pepper', 'extra-virgin olive oil', 'onions', 'water',
'butter']
cajun_creole : ['salt', 'onions', 'garlic', 'green bell pepper', 'butter', 'o
live oil', 'cayenne pepper', 'cajun seasoning', 'all-purpose flour', 'water']
french : ['salt', 'sugar', 'all-purpose flour', 'unsalted butter', 'olive oi
l', 'butter', 'water', 'large eggs', 'garlic cloves', 'ground black pepper']
filipino : ['salt', 'garlic', 'onions', 'water', 'soy sauce', 'pepper', 'oi
l', 'sugar', 'carrots', 'ground black pepper']
spanish : ['salt', 'olive oil', 'garlic cloves', 'extra-virgin olive oil', 'o
nions', 'water', 'tomatoes', 'ground black pepper', 'red bell pepper', 'peppe
r']
mexican : ['salt', 'onions', 'ground cumin', 'garlic', 'olive oil', 'chili po
wder', 'jalapeno chilies', 'sour cream', 'avocado', 'corn tortillas']
british : ['salt', 'all-purpose flour', 'butter', 'milk', 'unsalted butter',
'eggs', 'sugar', 'onions', 'baking powder', 'large eggs']
japanese : ['soy sauce', 'salt', 'mirin', 'sugar', 'water', 'sake', 'rice vin
egar', 'vegetable oil', 'scallions', 'ginger']
greek : ['salt', 'olive oil', 'dried oregano', 'garlic cloves', 'feta cheese
crumbles', 'extra-virgin olive oil', 'fresh lemon juice', 'ground black peppe
r', 'garlic', 'pepper']
chinese : ['soy sauce', 'sesame oil', 'salt', 'corn starch', 'sugar', 'garli
c', 'water', 'green onions', 'vegetable oil', 'scallions']
indian : ['salt', 'onions', 'garam masala', 'water', 'ground turmeric', 'garl
ic', 'cumin seed', 'ground cumin', 'vegetable oil', 'oil']
thai : ['fish sauce', 'garlic', 'salt', 'coconut milk', 'vegetable oil', 'soy
sauce', 'sugar', 'water', 'garlic cloves', 'fresh lime juice']
vietnamese : ['fish sauce', 'sugar', 'salt', 'garlic', 'water', 'carrots', 's
oy sauce', 'shallots', 'garlic cloves', 'vegetable oil']
russian : ['salt', 'sugar', 'onions', 'all-purpose flour', 'sour cream', 'egg
s', 'water', 'butter', 'unsalted butter', 'large eggs']
```



2.UN-NORMALIZED GRPAH

In [118]:
```
figure, axes =plt.subplots(figsize=(10,6))
for i in cusine_unique_ingredient_RANK.keys():
  cusine_unique_ingredient_RANK[i]=dict(sorted(cusine_unique_ingredient_RANK[i
].items(), key=lambda x: x[1], reverse=True))
  most_popular_top10={}
  count=0
  for i1 in cusine_unique_ingredient_RANK[i].keys():
    if count< 10:
      most_popular_top10[i1]=cusine_unique_ingredient_RANK[i][i1]
      count+=1
    else:
      break
  # print(i,":",list(most_popular_top10.keys()))
  y=list(cusine_unique_ingredient_RANK[i].values())
  x=[h for h in range(1,len(y)+1)]
  a=[x1 for x1 in x]
  b=[y1 for y1 in y]
  axes.plot(a,b,label=i)
  # plt.loglog()
  axes.set_xlabel('Ingredients')
  axes.set_ylabel('frequency')
  axes.set_title(" ingredient-rank distribution According to CUISINE'S")
  plt.legend()
```



(d) What is your interpretation of the results?

ANS:-

In ingredient rank distribution graph the most frequent ingredient is having highest rank(1 least in number) among others like salt which is present mostly in every recipies eaten, while other recipies like sauce or bevarages are very least frequent and are not used in only few specific cuisines hence they are having lowest rank(1000 or highest in number).

So the conclusion is the ingredient which are used in particular cuisines are least in rank but ingredient which are most frequent are used in most of the cusines and are higher in rank .

In [ ]: