Tourist Guide

Introduction

When travelling, different people come with different tastes. Some people travel to visit historical places and museums, others travel to spend time at famous bars or nightclubs, and more recently, more people have become interested in what is called 'Food Tourism'. Therefore, this project is going to consider a list of large cities from all over the world, and it is going to group them according to the type of tourism they offer.

This recommender system can be of interest to tourists from all over the globe, who are willing to get some suggestions regarding cities to visit which satisfy their touristic interests.

This can also be of interest to people willing to invest in any of these categories of tourism (historical places/museums, bars\nightclubs and food), as it helps them to choose between cities where each of these categories is booming or those which the market that has not saturated yet and can still accept more investment to boom. The Foursquare API can be used to obtain the list of venues for each of the cities considered.

Data Description

Data Sources

The list of different cities in the world is obtained from the 'World City Database' from the site https://simplemaps.com/data/world-cities (https://simplemaps.com/data/world-cities)

The dataset has been built from the ground up using authoritative sources such as the NGIA, US Geological Survey, US Census Bureau, and NASA. It was last refreshed in April of 2019. The database contains around 13 thousand entries.

The file containing the list of cities and countries with their corresponding longitutdes and latitudes is in CSV format with no missing data in these columns. This can be imported and then the Foursquare API can be used to get the venues for each city.

Data Cleaning

The column used to get the names of the cities from the dataset is 'city_ascii' and not the 'city' column to avoid the appearance of special characters that come from different languages. Therefore, the columns to keep are 'city_ascii', 'country', 'lat' and 'lng'. Then, change 'city_ascii; column name into 'city'.

Some city names are repeated, therefore only the first occurance of the city is kept, while the duplicates are dropped.

Since, not all cities will be of interest to tourists due to the lack of venues that belong to the categories (historical places/museums, bars\nightclubs and food), some cities need to be dropped. Our metric to choose the cities to drop will be the total number of hotels, hostels and motels in each city, as it gives a good indication to how touristic this city is. If the total number of touristic residences is equal to zero, the city will be dropped.

In order to decide if a given venue can be considered as an accomodation for tourists or not, the category of the venue will be checked if it belongs to the following list of words (hotel, motel, hostel, auberge, inn, lodge, tavern, guesthouse, B and B, resort, camp, room, appartment, mansion) obtained from https://www.merriam-webster.com/thesaurus/hotel (https://www.merriam-webster.com/thesaurus/hotel) https://relatedwords.org/relatedto/hotel (https://relatedwords.org/relatedto/hotel)

Sometimes, the Foursquare API returns no venues for a given city, or none of the venues belong to the three categories (historical places/museums, bars\nightclubs and food), therefore this city is dropped.

For each of the remaning cities, venues that belong to the three categories will be counted, while other venues will not be included.

For a venue to belong to the 'food' category, it's category needs to have one of the following words (restaurant, bistro, pizza, chicken, beef, seafood, ice cream, sushi, barbeque, noodle, steak, diner, bbq, wings, burger, buffet, grill, grills, grilled, steakhouse, fish, tacos, pasta).

And for a venue to belong to the 'bars\nightclubs' category, it's category needs to have one of the following words (bar, nightclub, liquor, brewery, pub, disco, discotheque, wine, dance, casino, beer, cocktail, cabaret, brasserie, lounge).

Finally, for a venue to belong to the 'historical places/museums', it's category needs to have one of the following words (museum, historical, history, monument, site, historic, monuments, gallery, palace, hall, library, archeological, castle, chateau, fortress, fountain)

Import the dataset containing the list of different cities

In [1]:
```
import csv
import pandas as pd
import requests
import numpy as np
```

In [2]:
```
# Convert the csv file into a DataFrame
path = r'C:\Users\amt\Downloads\simplemaps_worldcities_basicv1.5\worldcities.c
sv'
pd_cities = pd.read_csv(path)

# Remove unnecessary columns
pd_cities = pd_cities[['city_ascii', 'country', 'lat', 'lng']]

# change city_ascii column name into city
pd_cities.rename(columns={'city_ascii':'city'},inplace=True)
pd_cities.head()
print('There are {} different cities'.format(len(pd_cities)))
```

```
There are 12959 different cities
```

In [3]:
```
pd_cities.iloc[1000]
```

Out[3]:
```
city       Mzimba
country    Malawi
lat         -11.9
lng          33.6
Name: 1000, dtype: object
```

In [4]:
```
# check if any of the city names are repeated in the dataset
pd_cities['city'].value_counts()
```

Out[4]:

Franklin	11
Clinton	10
Greenville	10
Salem	9
Springfield	9
Washington	9
Richmond	9
Jackson	9
Georgetown	8
Lebanon	8
Hamilton	8
Victoria	8
Monroe	8
Marion	8
Alexandria	7
Plymouth	7
Portland	7
Florence	7
Newport	7
Columbia	7
Auburn	7
Kingston	7
Monticello	6
San Jose	6
Mount Vernon	6
Covington	6
Rochester	6
Lexington	6
Princeton	6
Columbus	6
..	..
Mingacevir	1
Ubon Ratchathani	1
Koktokay	1
Hidalgo	1
Andorra	1
Dikson	1
Rock Island	1
Jersey Shore	1
Koceljeva	1
Sao Tome	1
Kingsland	1
Islington	1
Vineyard	1
Katwe	1
Upper Hutt	1
Hue	1
Lamu	1
Ternopil	1
Vadso	1
Sept-Iles	1
Rangoon	1
Araouane	1
Ahmednagar	1
Dyersburg	1
San Luis Obispo	1
Ajdabiya	1

```
Omaruru             1
Samux               1
Dinuba              1
Lincoln City        1
Name: city, Length: 11555, dtype: int64
```

In [5]:
```
# remove duplicates of city
pd_cities.drop_duplicates(subset ="city", inplace = True)
print('There are {} different cities after removing duplicates'.format(len(pd_
cities)))
```

```
There are 11555 different cities after removing duplicates
```

Install and import the geopy python package in order to obtain thelongitude and latitude of each of the postal codes

In [6]:
```
# tranforming json file into a pandas dataframe library
from pandas.io.json import json_normalize
```

In [7]:
```
pip install folium
```

```
Requirement already satisfied: folium in c:\users\amt\anaconda3\lib\site-pack
ages (0.10.0)
Requirement already satisfied: requests in c:\users\amt\anaconda3\lib\site-pa
ckages (from folium) (2.22.0)
Requirement already satisfied: numpy in c:\users\amt\anaconda3\lib\site-packa
ges (from folium) (1.16.4)
Requirement already satisfied: jinja2>=2.9 in c:\users\amt\anaconda3\lib\site
-packages (from folium) (2.10.1)
Requirement already satisfied: branca>=0.3.0 in c:\users\amt\anaconda3\lib\si
te-packages (from folium) (0.3.1)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\amt\anaconda3\l
ib\site-packages (from requests->folium) (2019.6.16)
Requirement already satisfied: chardet<3.1.0,>=3.0.2 in c:\users\amt\anaconda
3\lib\site-packages (from requests->folium) (3.0.4)
Requirement already satisfied: idna<2.9,>=2.5 in c:\users\amt\anaconda3\lib\s
ite-packages (from requests->folium) (2.8)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in
c:\users\amt\anaconda3\lib\site-packages (from requests->folium) (1.24.2)
Requirement already satisfied: MarkupSafe>=0.23 in c:\users\amt\anaconda3\lib
\site-packages (from jinja2>=2.9->folium) (1.1.1)
Requirement already satisfied: six in c:\users\amt\anaconda3\lib\site-package
s (from branca>=0.3.0->folium) (1.12.0)
Note: you may need to restart the kernel to use updated packages.
```

In [8]:
```
# Matplotlib and associated plotting modules
import matplotlib.cm as cm
import matplotlib.colors as colors

import folium # map rendering library

# libraries for displaying images
from IPython.display import Image
from IPython.core.display import HTML

print('Libraries imported.')
```

```
Libraries imported.
```

Define functions needed to explore all cities

In [9]:
```
# function that extracts the category of the venue
def get_category_type(row):
    try:
        categories_list = row['categories']
    except:
        categories_list = row['venue.categories']

    if len(categories_list) == 0:
        return None
    else:
        return categories_list[0]['name']

# function that extracts all the venues of a given city and the total number o
f accomodations
def get_accom_num(latitude,longitude,city,country):
    CLIENT_ID = 'ZXCLYSYUWVNJPEC5ITUIVXFZXHADNFX2FYFNHZE2GEQLP3H1' # your Four
square ID
    CLIENT_SECRET = 'CPA5PULYPWSCPIVZPPEXQJ2XKRG2CTGNYVNK3IONY2SKJIXI' # your
 Foursquare Secret
    VERSION = '20180604'
    LIMIT = 1000
    radius = 10000

    # Define url
    url = 'https://api.foursquare.com/v2/venues/explore?client_id={}&client_se
cret={}&ll={},{}&v={}&radius={}&limit={}'\
    .format(CLIENT_ID, CLIENT_SECRET, latitude, longitude, VERSION, radius, LI
MIT)

    try:

        # Send GET request
        results = requests.get(url).json()
        items = results['response']['groups'][0]['items']

        # Process JSON and convert it to a clean dataframe
        dataframe = json_normalize(items) # flatten JSON

        # Remove \n from column names
        #dataframe.columns = dataframe.columns.str.strip()


        filtered_columns = ['venue.name', 'venue.categories', 'venue.id','venu
e.location.lat','venue.location.lng']
        dataframe_filtered = dataframe.loc[:, filtered_columns]

        # Filter the category for each row
        dataframe_filtered['venue.categories'] = dataframe_filtered.apply(get_
category_type, axis=1)

        # Clean columns
        dataframe_filtered.columns = [col.split('.')[-1] for col in dataframe_
filtered.columns]

        # change the 'categories' column into lower case characters
        dataframe_filtered['categories'] = dataframe_filtered['categories'].st
```

```
r.lower()

        # Count number of accomodations that suit tourists
        count_accom = sum(dataframe_filtered['categories'].isin(['hotel','mote
l','hostel','auberge','inn','lodge','tavern',\
                    'guesthouse', 'B and B', 'resort', 'camp','room','appartme
nt','mansion']))
    except:
        count_accom = 0
        dataframe_filtered = pd.DataFrame()

    return dataframe_filtered, count_accom

# function that counts total number of venues that belong to each the three ca
tegories
# (historical places/museums, bars\nightclubs and food)
def count_venues(venues):
    list_foods = ['restaurant', 'bistro', 'pizza','chicken', 'beef', 'seafood'
, 'ice cream', 'sushi', 'barbeque',\
                  'noodle', 'steak', 'diner','bbq', 'wings', 'burger', 'buffe
t', 'grill', 'grills', 'grilled','steakhouse',\
                  'fish', 'tacos', 'pasta']
    list_night = ['bar', 'nightclub', 'liquor', 'brewery', 'pub', 'disco', 'di
scotheque', 'wine', 'dance', 'casino', 'beer',\
                 'cocktail', 'cabaret', 'brasserie', 'lounge']
    list_museum = ['museum', 'historical', 'history', 'monument', 'site', 'his
toric', 'monuments', 'gallery', 'palace',\
                  'hall', 'library', 'archeological', 'castle', 'chateau', 'fo
rtress', 'fountain']

    count_food = 0
    count_night = 0
    count_museum = 0
    try:
        for i in range(0,len(venues)):
            categ = venues.iloc[i]['categories'].split(' ')
            if any(c in set(list_foods) for c in categ):
                count_food += 1
            elif any(c in set(list_night) for c in categ):
                count_night += 1
            elif any(c in set(list_museum) for c in categ):
                count_museum += 1
    except:
        count_food = 0
        count_night = 0
        count_museum = 0

    return count_food, count_night, count_museum
```

In [10]:
```
# for each city in the dataset which has accomodation for tourists, count the
 total number of venues that belong to each
# of the three categories we have (historical places/museums, bars\nightclubs
 and food)

# shuffle the dataset
pd_cities = pd_cities.sample(random_state=0,frac=1).reset_index(drop=True)

#city_info = pd.DataFrame(columns=['city', 'country', 'lat', 'lng', 'count_acc
om', 'count_food', 'count_night', 'count_museum'])
city_info_list = []
for i in range(0,len(pd_cities)):
    dict1 = {}
    df, count_accom = get_accom_num(pd_cities.iloc[i]['lat'],pd_cities.iloc[i]
['lng'],pd_cities.iloc[i]['city'],\
                            pd_cities.iloc[i]['country'])
    # drop cities where there are no hotels
    if count_accom == 0:
        continue

    foods, nights, museums = count_venues(df)
    # drop cities where none of the venues belong to our categories of interes
t
    if foods+nights+museums == 0:
        continue

    dict1 = {'city': pd_cities.iloc[i]['city'], 'country':pd_cities.iloc[i]['c
ountry'],\
                    'lat':pd_cities.iloc[i]['lat'], 'lng':pd_cities.iloc[i]['ln
g'], 'count_accom':count_accom,\
                    'count_food':foods, 'count_night':nights, 'count_museum':mu
seums}
    if np.remainder(i+1,500)==0:
        print('City number ', i+1)
        print(dict1)
    city_info_list.append(dict1)

city_info = pd.DataFrame(city_info_list)
```

```
City number  1000
{'city': 'Daman', 'country': 'India', 'lat': 20.417, 'lng': 72.85, 'count_acc
om': 4, 'count_food': 12, 'count_night': 2, 'count_museum': 0}
City number  2000
{'city': 'Caernarfon', 'country': 'United Kingdom', 'lat': 53.15, 'lng': -4.2
667, 'count_accom': 2, 'count_food': 4, 'count_night': 7, 'count_museum': 2}
City number  3000
{'city': 'Heath', 'country': 'United States', 'lat': 32.8444, 'lng': -96.467
9, 'count_accom': 1, 'count_food': 43, 'count_night': 2, 'count_museum': 0}
City number  3500
{'city': 'Ijebu Ode', 'country': 'Nigeria', 'lat': 6.8204, 'lng': 3.92, 'coun
t_accom': 1, 'count_food': 0, 'count_night': 1, 'count_museum': 0}
City number  4000
{'city': 'New Braunfels', 'country': 'United States', 'lat': 29.6995, 'lng':
-98.1153, 'count_accom': 2, 'count_food': 44, 'count_night': 16, 'count_museu
m': 0}
City number  4500
{'city': 'Phangnga', 'country': 'Thailand', 'lat': 8.451, 'lng': 98.534, 'cou
nt_accom': 2, 'count_food': 10, 'count_night': 0, 'count_museum': 0}
City number  5500
{'city': 'Chinandega', 'country': 'Nicaragua', 'lat': 12.6304, 'lng': -87.13,
'count_accom': 3, 'count_food': 10, 'count_night': 4, 'count_museum': 0}
City number  11000
{'city': 'Troutdale', 'country': 'United States', 'lat': 45.5372, 'lng': -12
2.3955, 'count_accom': 1, 'count_food': 34, 'count_night': 15, 'count_museu
m': 1}
City number  11500
{'city': 'Ennis', 'country': 'Ireland', 'lat': 52.8436, 'lng': -8.9864, 'coun
t_accom': 7, 'count_food': 5, 'count_night': 8, 'count_museum': 0}
```

In [11]:
```
city_info.head(10)
```

Out[11]:

	city	count_accom	count_food	count_museum	count_night	country	lat	
0	Chesapeake Beach	1	17	0	2	United States	38.6881	-
1	Saint Anthony	1	37	4	13	United States	45.0278	-
2	Kamloops	8	21	0	5	Canada	50.6667	-1
3	Lida	1	3	2	3	Belarus	53.8885	
4	Hohhot	6	2	1	0	China	40.8200	'
5	Leon	2	40	0	8	Mexico	21.1500	-1
6	Wilmington	1	34	5	10	United States	39.7415	-
7	Tanjungpandan	8	24	4	1	Indonesia	-2.7500	1
8	Marrakesh	23	33	7	11	Morocco	31.6300	
9	Santa Cruz do Sul	3	24	0	11	Brazil	-29.7100	-

In [12]:
```
print('The total number of cities that fit our selection criteria is {}'.forma
t(len(city_info)))
```

```
The total number of cities that fit our selection criteria is 6654
```

In [ ]:
```

```