A program For Photographers Event

Introduction

Nepal is a beautiful country with a great prospect of tourism. Every year a lot of tourists come to visit Nepal. Hence in this analytical project I want find the best region of Kathmandu for tourist to stay such that they have maximum access to facilities.

Data Description

The data used in this project is provided by Foursquare location data. The data are grouped by landscape area, and each area included the information about this area and all information about restaurants, cafes, and stores which in this area.

Table of contents

Import Libraries

In [2]:
```
import requests # to handle requests
import pandas as pd # for data analsysis
import numpy as np # to handle data in a vectorized manner

#!conda install -c conda-forge geopy --yes
from geopy.geocoders import Nominatim # module to convert an address into lati
tude and longitude values

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

#tranforming json file into a pandas dataframe library
from pandas.io.json import json_normalize

#!conda install -c conda-forge folium=0.5.0 --yes
import folium # plotting library
```

Define Foursquare Credentials

code has been removed

Define the city and get its latitude & longitude

In [7]:
```
# define the city and get its latitude & longitude
city = 'Kathmandu'
geolocator = Nominatim(user_agent="foursquare_agent")
location = geolocator.geocode(city)
latitude = location.latitude
longitude = location.longitude
print(latitude, longitude)
```

```
27.708796 85.320244
```

Search for Hotels

In [8]:
```
# search for hotels
search_query = 'Hotel'
radius = 20000

# Define the corresponding URL
url = 'https://api.foursquare.com/v2/venues/search?client_id={}&client_secret=
{}&ll={},{}&v={}&query={}&radius={}&limit={}'\
.format(CLIENT_ID, CLIENT_SECRET, latitude, longitude, VERSION, search_query,
radius, LIMIT)
```

In [9]:
```
# Send the GET Request and examine the results
results = requests.get(url).json()
#results
```

In [10]:
```
# assign relevant part of JSON to venues
venues = results['response']['venues']

# tranform venues into a dataframe
dataframe = json_normalize(venues)
dataframe.head()
```

Out[10]:

	categories	hasPerk	id	location.address	location.cc
0	[{'id': '4bf58dd8d48988d1fa931735', 'name': 'H...	False	4bc7886a93bdeee12e7d37ae	Lalupate Marg	NP
1	[{'id': '4bf58dd8d48988d1fa931735', 'name': 'H...	False	4bded3bafe0e62b5e7fa0506	Lazimpat	NP
2	[{'id': '4bf58dd8d48988d1fa931735', 'name': 'H...	False	4cba0e303481199c9c036b3f	Lal Durbar	NP
3	[{'id': '4bf58dd8d48988d1fa931735', 'name': 'H...	False	4bcf47520ffdce724657b2c0	Durbar Marg	NP
4	[{'id': '4bf58dd8d48988d1fa931735', 'name': 'H...	False	4bc7886393bdeee1287d37ae	P O Box 2269 Lazimpat	NP

Clean Hotel Dataframe

In [11]:
```
# keep only columns that include venue name, and anything that is associated w
ith location
clean_columns = ['name', 'categories'] + [col for col in dataframe.columns if
col.startswith('location.')]+ ['id']
clean_dataframe = dataframe.loc[:,clean_columns]

# 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']

# filter the category for each row
clean_dataframe['categories'] = clean_dataframe.apply(get_category_type, axis=
1)

# clean column names by keeping only last term
clean_dataframe.columns = [column.split('.')[-1] for column in clean_dataframe
.columns]

clean_dataframe.head()
```

Out[11]:

	name	categories	address	cc	city	country	crossStreet	distance	formattedA
0	Hotel Yak & Yeti	Hotel	Lalupate Marg	NP	काठमाडौं	नेपाल	Hattisar Sadak	310	[Lalupat (Hattisar काठमाडौं
1	Hotel Shanker	Hotel	Lazimpat	NP	काठमाडौं	नेपाल	NaN	1131	[Lazimpat, व 4460(
2	Royal Singhi Hotel	Hotel	Lal Durbar	NP	काठमाडौं	नेपाल	Durbar Marg	250	[Lal Durbar (Marg), क
3	De L'Annapurna Hotel	Hotel	Durbar Marg	NP	काठमाडौं	नेपाल	NaN	457	[Durba काठमाडौं
4	Radisson Hotel	Hotel	P O Box 2269 Lazimpat	NP	Kathmandu Met. City	नेपाल	NaN	1279	[P O Bo La Kathman((

In [12]:
```
# delete unnecessary columns
clean_dataframe2= clean_dataframe.drop(['cc', 'city', 'country', 'crossStreet'
, 'distance', 'formattedAddress',\
                                       'labeledLatLngs', 'neighborhood', 'pos
talCode', 'id'], axis=1)
clean_dataframe2.head()
```

Out[12]:

	name	categories	address	lat	lng	state
0	Hotel Yak & Yeti	Hotel	Lalupate Marg	27.711581	85.320274	Central Region
1	Hotel Shanker	Hotel	Lazimpat	27.718956	85.320082	Central Region
2	Royal Singhi Hotel	Hotel	Lal Durbar	27.710940	85.319466	Central Region
3	De L'Annapurna Hotel	Hotel	Durbar Marg	27.711117	85.316408	Central Region
4	Radisson Hotel	Hotel	P O Box 2269 Lazimpat	27.720270	85.320924	NaN

In [13]:
```
# delete rows with none values
clean_dataframe3 = clean_dataframe2.dropna(axis=0, how='any', thresh=None, sub
set=None, inplace=False)
clean_dataframe3.head()
```

Out[13]:

	name	categories	address	lat	lng	state
0	Hotel Yak & Yeti	Hotel	Lalupate Marg	27.711581	85.320274	Central Region
1	Hotel Shanker	Hotel	Lazimpat	27.718956	85.320082	Central Region
2	Royal Singhi Hotel	Hotel	Lal Durbar	27.710940	85.319466	Central Region
3	De L'Annapurna Hotel	Hotel	Durbar Marg	27.711117	85.316408	Central Region
5	Hotel Buddha	Hotel	25728	27.717044	85.310450	Kathmando

In [14]:
```
# delete rows which its category is not Hotel or Event Space
array= ['Hotel', 'Event Space', 'Resort']
hotel_dataframe= clean_dataframe3.loc[clean_dataframe3['categories'].isin(arra
y)]
hotel_dataframe = clean_dataframe3
```

In [15]:
```
# delete rows which has duplicate hotel's name
df_hotels = hotel_dataframe.drop_duplicates(subset='name', keep="first")
df_hotels.head()
```

Out[15]:

	name	categories	address	lat	lng	state
0	Hotel Yak & Yeti	Hotel	Lalupate Marg	27.711581	85.320274	Central Region
1	Hotel Shanker	Hotel	Lazimpat	27.718956	85.320082	Central Region
2	Royal Singhi Hotel	Hotel	Lal Durbar	27.710940	85.319466	Central Region
3	De L'Annapurna Hotel	Hotel	Durbar Marg	27.711117	85.316408	Central Region
5	Hotel Buddha	Hotel	25728	27.717044	85.310450	Kathmando

Search for temples

In [16]:
```
# search for temples
search_query = 'Temple'
radius = 20000

# Define the corresponding URL
url = 'https://api.foursquare.com/v2/venues/search?client_id={}&client_secret={}&ll={},{}&v={}&query={}&radius={}&limit={}'\
.format(CLIENT_ID, CLIENT_SECRET, latitude, longitude, VERSION, search_query, radius, LIMIT)
```

In [17]:
```
# Send the GET Request and examine the results
presults = requests.get(url).json()
#presults
```

In [18]:
```
# assign relevant part of JSON to venues
venues = presults['response']['venues']

# tranform venues into a dataframe
temples_dataframe = json_normalize(venues)
temples_dataframe.head()
```

Out[18]:

	categories	hasPerk	id	location.address	location.cc
0	[{'id': '4bf58dd8d48988d13a941735', 'name': 'T...	False	4bcf47520ffdce724457b2c0	Chusyabahal	NP
1	[{'id': '52e81612bcbc57f1066b7a3e', 'name': 'B...	False	51790da9e4b0de302b37fed6	Kwabahal	NP
2	[{'id': '4bf58dd8d48988d13a941735', 'name': 'T...	False	527b6c6511d28e4aa2a056f9	NaN	NP
3	[{'id': '4bf58dd8d48988d13a941735', 'name': 'T...	False	5690fc4338fafe86458fef24	NaN	NP
4	[{'id': '4deefb944765f83613cdba6e', 'name': 'H...	False	507e919ee4b00e30b6288444	NaN	NP

Clean temples Dataframe

In [19]:
```
# keep only columns that include venue name, and anything that is associated w
ith location
temples_clean_columns = ['name', 'categories'] + [col for col in temples_dataf
rame.columns if col.startswith('location.')]+ ['id']
clean_temples_dataframe = temples_dataframe.loc[:,temples_clean_columns]

# function that extracts the category of the venue
def get_category_type(row):
    try:
        categories_list1 = row['categories']
    except:
        categories_list1 = row['venue.categories']

    if len(categories_list1) == 0:
        return None
    else:
        return categories_list1[0]['name']

# filter the category for each row
clean_temples_dataframe['categories'] = clean_temples_dataframe.apply(get_cate
gory_type, axis=1)

# clean column names by keeping only last term
clean_temples_dataframe.columns = [column.split('.')[-1] for column in clean_t
emples_dataframe.columns]

clean_temples_dataframe.head()
```

Out[19]:

	name	categories	address	cc	city	country	crossStreet	distance	formattedAd
0	Kantipur Temple House	Temple	Chusyabahal	NP	काठमाडौं	नेपाल	Jyatha	769	[Chusya (Jyatha), का
1	Hiranya Varna Mahavihar (Golden Temple)	Buddhist Temple	Kwabahal	NP	Pātan	नेपाल	NaN	3757	[Kwabahal, I
2	Bal Gopal Temple	Temple	NaN	NP	NaN	नेपाल	NaN	555	
3	Akash Bhairab Temple	Temple	NaN	NP	NaN	नेपाल	NaN	1096	
4	Golden Temple	Historic Site	NaN	NP	NaN	नेपाल	NaN	701	

In [20]:
```
# delete unnecessary columns
clean_temples_dataframe2= clean_temples_dataframe.drop(['cc', 'city', 'countr
y', 'distance', 'formattedAddress',\
                                        'labeledLatLngs','crossStreet','postal
Code', 'id'], axis=1)
clean_temples_dataframe2.head()
```

Out[20]:

	name	categories	address	lat	lng	state
0	Kantipur Temple House	Temple	Chusyabahal	27.711235	85.312934	Central Region
1	Hiranya Varna Mahavihar (Golden Temple)	Buddhist Temple	Kwabahal	27.675248	85.324419	Central Region
2	Bal Gopal Temple	Temple	NaN	27.708813	85.314609	NaN
3	Akash Bhairab Temple	Temple	NaN	27.705959	85.309586	NaN
4	Golden Temple	Historic Site	NaN	27.714990	85.321530	NaN

In [21]:
```
# delete rows with none values
clean_temples_dataframe3 = clean_temples_dataframe2.dropna(axis=0, how='any',
thresh=None, subset=None, inplace=False)
clean_temples_dataframe3.head()
```

Out[21]:

	name	categories	address	lat	lng	state
0	Kantipur Temple House	Temple	Chusyabahal	27.711235	85.312934	Central Region
1	Hiranya Varna Mahavihar (Golden Temple)	Buddhist Temple	Kwabahal	27.675248	85.324419	Central Region
5	The Temple	Bar	Thamel	27.715492	85.310461	Central Region
6	Pashupatinath Temple	Temple	Pashupatinath Rd.	27.709101	85.348620	Central Region
8	Taleju Temple	Temple	Makhan Tole	27.712171	85.311342	Central Region

In [22]:
```
# delete rows which its category is not temples
df_temples = clean_temples_dataframe3[clean_temples_dataframe3.categories != 'Bar']
df_temples.head()
```

Out[22]:

	name	categories	address	lat	lng	state
0	Kantipur Temple House	Temple	Chusyabahal	27.711235	85.312934	Central Region
1	Hiranya Varna Mahavihar (Golden Temple)	Buddhist Temple	Kwabahal	27.675248	85.324419	Central Region
6	Pashupatinath Temple	Temple	Pashupatinath Rd.	27.709101	85.348620	Central Region
8	Taleju Temple	Temple	Makhan Tole	27.712171	85.311342	Central Region
15	Maitidevi Temple	Temple	1 Maitidevi Marg	27.705852	85.333986	Central Region

Search for Restaurants

In [23]:
```
# search for Restaurants
search_query = 'Restaurant'
radius = 20000

# Define the corresponding URL
url = 'https://api.foursquare.com/v2/venues/search?client_id={}&client_secret={}&ll={},{}&v={}&query={}&radius={}&limit={}'.format(CLIENT_ID, CLIENT_SECRET, latitude, longitude, VERSION, search_query, radius, LIMIT)
```

In [24]:
```
# Send the GET Request and examine the results
Rresults = requests.get(url).json()
#Rresults
```

In [25]:
```
# assign relevant part of JSON to venues
venues = Rresults['response']['venues']

# tranform venues into a dataframe
Restaurant_dataframe = json_normalize(venues)
Restaurant_dataframe.head()
```

Out[25]:

	categories	hasPerk	id	location.address	location.cc
0	[{'id': '4bf58dd8d48988d142941735', 'name': 'A...	False	4c829894d6ebbfb7b3234ca4	Thamel	NP
1	[{'id': '4bf58dd8d48988d149941735', 'name': 'T...	False	4c73be8cf4d476b0cc2568cf	Chakshibari Marg	NP
2	[{'id': '4bf58dd8d48988d1c4941735', 'name': 'R...	False	5226cc3a93cd4ef097a79f2b	132, Kwobahal,Thamel	NP
3	[{'id': '4bf58dd8d48988d1ca941735', 'name': 'P...	False	4ed4d667cc216e1a537fcaaa	NaN	NP
4	[{'id': '4bf58dd8d48988d142941735', 'name': 'A...	False	5b41d00016fa04002c5397b5	Thamel-29, Narsing chowck	NP

Clean Restaurant Dataframe

In [26]:
```
# keep only columns that include venue name, and anything that is associated w
ith location
Restaurant_clean_columns = ['name', 'categories'] + [col for col in Restaurant
_dataframe.columns if col.startswith('location.')]+ ['id']
clean_Restaurant_dataframe = Restaurant_dataframe.loc[:,Restaurant_clean_colum
ns]

# function that extracts the category of the venue
def get_category_type(row):
    try:
        categories_list3 = row['categories']
    except:
        categories_list3 = row['venue.categories']

    if len(categories_list3) == 0:
        return None
    else:
        return categories_list3[0]['name']

# filter the category for each row
clean_Restaurant_dataframe['categories'] = clean_Restaurant_dataframe.apply(ge
t_category_type, axis=1)

# clean column names by keeping only last term
clean_Restaurant_dataframe.columns = [column.split('.')[-1] for column in clea
n_Restaurant_dataframe.columns]

clean_Restaurant_dataframe.head()
```

Out[26]:

	name	categories	address	cc	city	country	crossStreet	distance	formatt
0	Yak Restaurant Bar & Lodge	Asian Restaurant	Thamel	NP	काठमाडौं	नेपाल	NaN	971	[Tham 4
1	Yin Yang Restaurant	Thai Restaurant	Chakshibari Marg	NP	काठमाडौं	नेपाल	Thamel	1188	[Chaks (Thame
2	Pilgrims 24 Restaurant & Bar (Formerly feed ...	Restaurant	132, Kwobahal,Thamel	NP	काठमाडौं	नेपाल	Next to Hotel Nepalaya	935	Kwoba (N N
3	Tranzit, Woodfire Pizza, Restaurant & Bar	Pizza Place	NaN	NP	NaN	नेपाल	NaN	858	
4	Nomad's Restaurant and Bar	Asian Restaurant	Thamel-29, Narsing chowck	NP	काठमाडौं	नेपाल	NaN	1050	[Narsi काठ

In [27]:
```
# delete unnecessary columns
clean_Restaurant_dataframe2= clean_Restaurant_dataframe.drop(['cc', 'city', 'country', 'crossStreet', 'distance', 'formattedAddress',\
                                                              'labeledLatLngs','neighborhood','postalCode', 'id'], axis=1)
clean_Restaurant_dataframe2.head()
```

Out[27]:

	name	categories	address	lat	lng	state
0	Yak Restaurant Bar & Lodge	Asian Restaurant	Thamel	27.712109	85.311125	Central Region
1	Yin Yang Restaurant	Thai Restaurant	Chakshibari Marg	27.714634	85.310147	Central Region
2	Pilgrims 24 Restaurant & Bar (Formerly feed ...	Restaurant	132, Kwobahal,Thamel	27.711672	85.311328	Central Region
3	Tranzit, Woodfire Pizza, Restaurant & Bar	Pizza Place	NaN	27.714094	85.313907	NaN
4	Nomad's Restaurant and Bar	Asian Restaurant	Thamel-29, Narsing chowck	27.713442	85.310970	Central Region

In [28]:
```
# delete rows with none values
df_Restaurant = clean_Restaurant_dataframe2.dropna(axis=0, how='any', thresh=None, subset=None, inplace=False)
df_Restaurant.head()
```

Out[28]:

	name	categories	address	lat	lng	state
0	Yak Restaurant Bar & Lodge	Asian Restaurant	Thamel	27.712109	85.311125	Central Region
1	Yin Yang Restaurant	Thai Restaurant	Chakshibari Marg	27.714634	85.310147	Central Region
2	Pilgrims 24 Restaurant & Bar (Formerly feed ...	Restaurant	132, Kwobahal,Thamel	27.711672	85.311328	Central Region
4	Nomad's Restaurant and Bar	Asian Restaurant	Thamel-29, Narsing chowck	27.713442	85.310970	Central Region
6	Dallé Restaurant	Restaurant	Kashtamandap Rd.	27.710031	85.319937	Central Region

Search for Cafeteria

In [29]:
```
# search for Cafeteria
search_query = 'Cafe'
radius = 20000

# Define the corresponding URL
url = 'https://api.foursquare.com/v2/venues/search?client_id={}&client_secret={}&ll={},{}&v={}&query={}&radius={}&limit={}'.format(CLIENT_ID, CLIENT_SECRET, latitude, longitude, VERSION, search_query, radius, LIMIT)
```

In [30]:
```
# Send the GET Request and examine the results
cresults = requests.get(url).json()
#cresults
```

In [31]:
```
# assign relevant part of JSON to venues
venues = cresults['response']['venues']

# tranform venues into a dataframe
Cafeteria_dataframe = json_normalize(venues)
Cafeteria_dataframe.head()
```

Out[31]:

	categories	hasPerk	id	location.address	location.cc
0	[{'id': '4bf58dd8d48988d1e0931735', 'name': 'C...	False	4ea4309a30f8b9d5ad12dd10	Amrit Marg	NP
1	[{'id': '4bf58dd8d48988d16d941735', 'name': 'C...	False	4d859cde7e8ef04d051625be	Thamel	NP
2	[{'id': '4bf58dd8d48988d1e0931735', 'name': 'C...	False	5c00c65178782c002cf88376	NaN	NP
3	[{'id': '4bf58dd8d48988d16d941735', 'name': 'C...	False	515a8aeae4b04ee4fe4123a1	Naxal	NP
4	[{'id': '4bf58dd8d48988d16d941735', 'name': 'C...	False	4db18ee3fa8ca4b3e9f9f631	Bhat Bhateni	NP

Clean Cafeteria Dataframe

In [32]:
```
# keep only columns that include venue name, and anything that is associated w
ith location
Cafeteria_clean_columns = ['name', 'categories'] + [col for col in Cafeteria_d
ataframe.columns if col.startswith('location.')]+ ['id']
clean_Cafeteria_dataframe = Cafeteria_dataframe.loc[:,Cafeteria_clean_columns]

# function that extracts the category of the venue
def get_category_type(row):
    try:
        categories_list4 = row['categories']
    except:
        categories_list4 = row['venue.categories']

    if len(categories_list4) == 0:
        return None
    else:
        return categories_list4[0]['name']

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

# clean column names by keeping only last term
clean_Cafeteria_dataframe.columns = [column.split('.')[-1] for column in clean
_Cafeteria_dataframe.columns]

clean_Cafeteria_dataframe.head()
```

Out[32]:

	name	categories	address	cc	city	country	crossStreet	distance	formattedAddr
0	Revolution Cafe	Coffee Shop	Amrit Marg	NP	काठमाडौं	नेपाल	Thamel	1033	[Amrit M (Thamel), काठ 44600, ने
1	The Northfield Cafe and Jesse James Bar	Café	Thamel	NP	काठमाडौं	नेपाल	NaN	1244	[Thamel, काठ ने
2	The Musketeerz Cafe	Coffee Shop	NaN	NP	काठमाडौं	नेपाल	NaN	1116	[काठमाडौं, ने
3	Espression: The Cafe	Café	Naxal	NP	काठमाडौं	नेपाल	Narayanchour	910	[N (Narayanch काठमाडौं, ने
4	Road House Cafe	Café	Bhat Bhateni	NP	काठमाडौं	नेपाल	NaN	1674	[Bhat Bha काठमाडौं, ने

In [33]:
```
# delete unnecessary columns
clean_Cafeteria_dataframe2= clean_Cafeteria_dataframe.drop(['cc', 'city', 'cou
ntry', 'crossStreet', 'distance', 'formattedAddress',\
                                                    'labeledLatLngs', 'postalCode', 'neigh
borhood', 'id'], axis=1)
clean_Cafeteria_dataframe2.head()
```

Out[33]:

	name	categories	address	lat	lng	state
0	Revolution Cafe	Coffee Shop	Amrit Marg	27.715046	85.312490	Central Region
1	The Northfield Cafe and Jesse James Bar	Café	Thamel	27.715555	85.310185	Central Region
2	The Musketeerz Cafe	Coffee Shop	NaN	27.716141	85.312529	Central Region
3	Espression: The Cafe	Café	Naxal	27.715180	85.326025	Central Region
4	Road House Cafe	Café	Bhat Bhateni	27.720106	85.331453	Central Region

In [34]:
```
# delete rows with none values
df_Cafeteria = clean_Cafeteria_dataframe2.dropna(axis=0, how='any', thresh=Non
e, subset=None, inplace=False)
df_Cafeteria.head()
```

Out[34]:

	name	categories	address	lat	lng	state
0	Revolution Cafe	Coffee Shop	Amrit Marg	27.715046	85.312490	Central Region
1	The Northfield Cafe and Jesse James Bar	Café	Thamel	27.715555	85.310185	Central Region
3	Espression: The Cafe	Café	Naxal	27.715180	85.326025	Central Region
4	Road House Cafe	Café	Bhat Bhateni	27.720106	85.331453	Central Region
5	Cafe Mondo Bizarro	Restaurant	Freak Street	27.703222	85.307922	Central Region

Search for Shopping Stores

In [35]:
```
# search for Shopping
search_query = 'Mall'
radius = 20000

# Define the corresponding URL
url = 'https://api.foursquare.com/v2/venues/search?client_id={}&client_secret=
{}&ll={},{}&v={}&query={}&radius={}&limit={}'.format(CLIENT_ID, CLIENT_SECRET,
latitude, longitude, VERSION, search_query, radius, LIMIT)
```

In [36]:
```
# Send the GET Request and examine the results
sresults = requests.get(url).json()
#sresults
```

In [37]:
```
# assign relevant part of JSON to venues
venues = sresults['response']['venues']

# tranform venues into a dataframe
Shopping_dataframe = json_normalize(venues)
Shopping_dataframe.head()
```

Out[37]:

	categories	hasPerk	id	location.address	location.cc
0	[{'id': '4bf58dd8d48988d108951735', 'name': 'W...	False	556ac8d7498e64da5cd5548f	NaN	NP
1	[{'id': '4bf58dd8d48988d1fd941735', 'name': 'S...	False	4c6bb91d0c3ac9b6a78dd238	NaN	NP
2	[{'id': '4bf58dd8d48988d1f6941735', 'name': 'D...	False	4c99d148d4b1b1f7348fca35	Tripureshwor	NP
3	[{'id': '4bf58dd8d48988d108951735', 'name': 'W...	False	5956305df4b5252a672602f7	NaN	NP
4	[{'id': '4bf58dd8d48988d1fd941735', 'name': 'S...	False	5291e861498e3ee2d83f0dcb	Kamaladi	NP

Clean Shopping Dataframe

In [38]:
```
# keep only columns that include venue name, and anything that is associated w
ith location
Shopping_clean_columns = ['name', 'categories'] + [col for col in Shopping_dat
aframe.columns if col.startswith('location.')]+ ['id']
clean_Shopping_dataframe = Shopping_dataframe.loc[:,Shopping_clean_columns]

# function that extracts the category of the venue
def get_category_type(row):
    try:
        categories_list5 = row['categories']
    except:
        categories_list5 = row['venue.categories']

    if len(categories_list5) == 0:
        return None
    else:
        return categories_list5[0]['name']

# filter the category for each row
clean_Shopping_dataframe['categories'] = clean_Shopping_dataframe.apply(get_ca
tegory_type, axis=1)

# clean column names by keeping only last term
clean_Shopping_dataframe.columns = [column.split('.')[-1] for column in clean_
Shopping_dataframe.columns]

clean_Shopping_dataframe.head()
```

Out[38]:

	name	categories	address	cc	city	country	crossStreet	distance	formattedAd
0	The Collective, Rising Mall	Women's Store	NaN	NP	NaN	नेपाल	NaN	160	
1	China Town Mall	Shopping Mall	NaN	NP	NaN	नेपाल	NaN	170	
2	Bluebird Mall	Department Store	Tripureshwor	NP	काठमाडौं	नेपाल	NaN	1931	[Tripures काठमाडौं,
3	Madame Rising Mall	Women's Store	NaN	NP	काठमाडौं	नेपाल	NaN	247	[काठमाडौं,
4	Rising Mall	Shopping Mall	Kamaladi	NP	NaN	नेपाल	NaN	171	[Kamaladi,

In [39]:
```
# delete unnecessary columns
clean_Shopping_dataframe2= clean_Shopping_dataframe.drop(['cc', 'city', 'country', 'distance', 'formattedAddress',\
                                                          'crossStreet', 'postalCode' ,'labeledLatLngs', 'id'], axis=1)
clean_Shopping_dataframe2.head()
```

Out[39]:

	name	categories	address	lat	lng	state
0	The Collective, Rising Mall	Women's Store	NaN	27.709596	85.318886	NaN
1	China Town Mall	Shopping Mall	NaN	27.708878	85.321969	NaN
2	Bluebird Mall	Department Store	Tripureshwor	27.691672	85.317112	Central Region
3	Madame Rising Mall	Women's Store	NaN	27.710111	85.318224	Central Region
4	Rising Mall	Shopping Mall	Kamaladi	27.709949	85.319086	NaN

In [40]:
```
# delete rows which its category is not Shopping Mall
df_Shopping = clean_Shopping_dataframe2[clean_Shopping_dataframe2.categories == 'Shopping Mall']
df_Shopping.head()
```

Out[40]:

	name	categories	address	lat	lng	state
1	China Town Mall	Shopping Mall	NaN	27.708878	85.321969	NaN
4	Rising Mall	Shopping Mall	Kamaladi	27.709949	85.319086	NaN
5	Kathmandu Mall	Shopping Mall	Kathmandu	27.701529	85.313348	Central Region
7	Times Square Mall	Shopping Mall	NaN	27.710723	85.317481	NaN
8	Sherpa Mall	Shopping Mall	DURBAR MARG, KATHMANDU, NEPAL Kathmandu,	27.710692	85.317591	nepal

In [41]:
```
# delete rows with none values
clean_Shopping_dataframe2 = clean_Shopping_dataframe2.dropna(axis=0, how='any'
, thresh=None, subset=None, inplace=False)
clean_Shopping_dataframe2.head()
```

Out[41]:

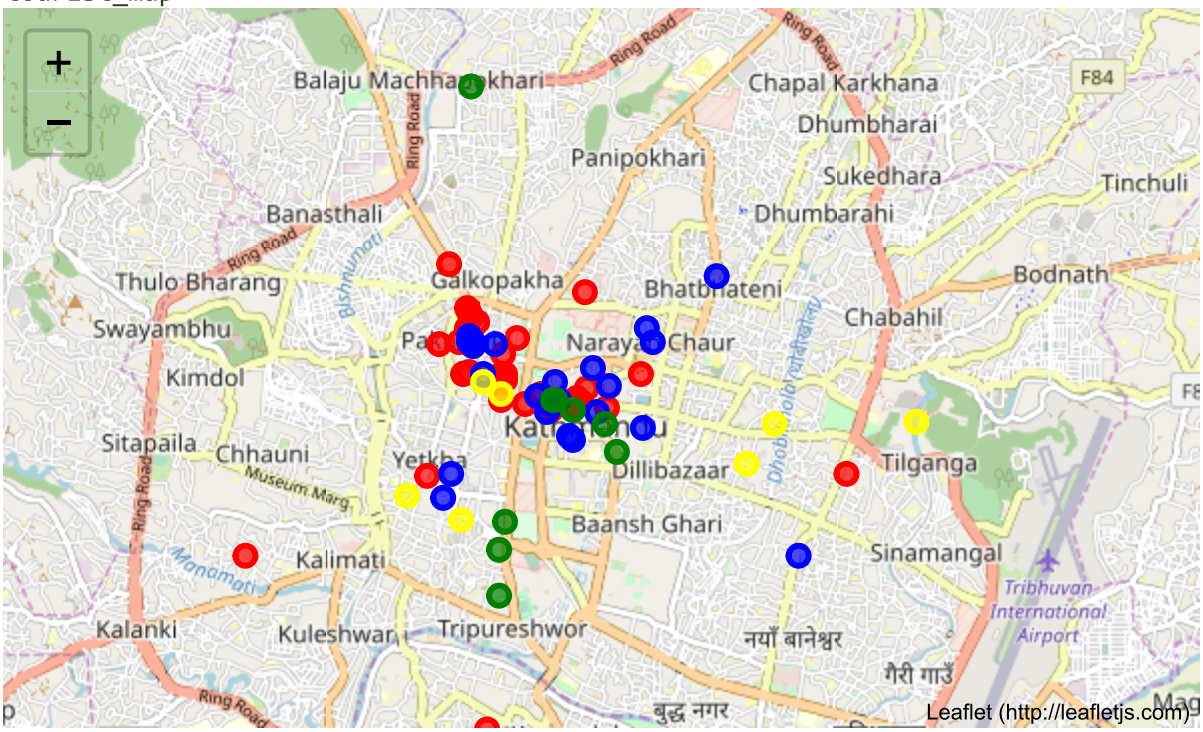
	name	categories	address	lat	lng	state
2	Bluebird Mall	Department Store	Tripureshwor	27.691672	85.317112	Central Region
5	Kathmandu Mall	Shopping Mall	Kathmandu	27.701529	85.313348	Central Region
6	Sherpa Mall Coffee Express	Coffee Shop	Durbar Marg	27.710735	85.317734	Central Region
8	Sherpa Mall	Shopping Mall	DURBAR MARG, KATHMANDU, NEPAL Kathmandu,	27.710692	85.317591	nepal
12	Civil Mall	Shopping Mall	Sundhara	27.699399	85.312736	Central Region

Generate maps to visualize venues and how they cluster together

In [42]:
```
# Generate map to visualize hotel neighbourhood including shopping stores and
 Cafeteria
tourist_map = folium.Map(location=[latitude, longitude], zoom_start=14)

for lat, lng, name, categories, address in zip(df_hotels['lat'], df_hotels['ln
g'],
                                               df_hotels['name'], df_hotels['categ
ories'],
                                                  df_hotels['address']):
    label = '{}, {}'.format(name, address)
    label = folium.Popup(label, parse_html=True)
    folium.CircleMarker(
        [lat, lng],
        radius=5,
        popup=label,
        color='red',
        fill=True,
        fill_color='red',
        fill_opacity=0.7,
        parse_html=False).add_to(tourist_map)

for lat, lng, name, categories, address in zip(df_Cafeteria['lat'], df_Cafeter
ia['lng'],
                                               df_Cafeteria['name'], df_Cafeteria[
'categories'],
                                                  df_Cafeteria['address']):
    label = '{}, {}'.format(name, address)
    label = folium.Popup(label, parse_html=True)
    folium.CircleMarker(
        [lat, lng],
        radius=5,
        popup=label,
        color='blue',
        fill=True,
        fill_color='blue',
        fill_opacity=0.7,
        parse_html=False).add_to(tourist_map)

for lat, lng, name, categories, address in zip(df_Shopping['lat'], df_Shopping
['lng'],
                                               df_Shopping['name'], df_Shopping['c
ategories'],
                                                  df_Shopping['address']):
    label = '{}, {}'.format(name, address)
    label = folium.Popup(label, parse_html=True)
    folium.CircleMarker(
        [lat, lng],
        radius=5,
        popup=label,
        color='green',
        fill=True,
        fill_color='green',
        fill_opacity=0.7,
        parse_html=False).add_to(tourist_map)

for lat, lng, name, categories, address in zip(df_temples['lat'], df_temples[
```

```
'lng'],
                                                df_temples['name'], df_temples['cat
egories'],
                                                   df_temples['address']):
    label = '{}, {}'.format(name, address)
    label = folium.Popup(label, parse_html=True)
    folium.CircleMarker(
        [lat, lng],
        radius=5,
        popup=label,
        color='yellow',
        fill=True,
        fill_color='yellow',
        fill_opacity=0.7,
        parse_html=False).add_to(tourist_map)

tourist_map
```

Out[42]:



Analytical Output

The above map ashows that the best place for tourist to live is the place which has the fastest way to get Thamel area and Durbarmarg area because the cluster of places is highly dense in this area.

(Look at the screenshot of map on github)

In [ ]: