# Coffeehouses in Abu Dhabi-Week-02

June 8, 2020

## 1 CoffeeHouses in Abu Dhabi

In this Notebook we will be utilizing Foursquare API to list down the three major coffeehouses places in Abu Dhabi and to check on which location might be ideal for a new branch or new coffee shop that wants to compete with them. |

Before we get the data and start exploring it, let's download all the dependencies that we will need.

[68]:
```
import numpy as np # library to handle data in a vectorized manner

import pandas as pd # library for data analsysis
pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', None)

import json # library to handle JSON files

!conda install -c conda-forge geopy --yes # uncomment this line if you haven't␣
 ↪completed the Foursquare API lab
from geopy.geocoders import Nominatim # convert an address into latitude and␣
 ↪longitude values

import requests # library to handle requests
from pandas.io.json import json_normalize # tranform JSON file into a pandas␣
 ↪dataframe

# Matplotlib and associated plotting modules
import matplotlib.cm as cm
import matplotlib.colors as colors

# import k-means from clustering stage
from sklearn.cluster import KMeans

#!conda install -c conda-forge folium=0.5.0 --yes # uncomment this line if you␣
 ↪haven't completed the Foursquare API lab
import folium # map rendering library

print('Libraries imported.')
```

```
Collecting package metadata (current_repodata.json): done
Solving environment: done

# All requested packages already installed.

Libraries imported.
```

## 1.1 1. Download and Explore Dataset

|Unforturtunetly there are not no data available online for free to get the areas in Abu Dhabi city straight forward, Major areas of city with latitude and longitude was taken manualy and inserted into "csv" file

**Load and explore the data** Next, let's load the data.

[69]:
```
AD_Areas= pd.read_csv('AD Areas.csv')
AD_Areas.head()
```

[69]:

| | Area | Latitude | Longitude |
|---|---|---|---|
| 0 | Al Mushrif | 24.443699 | 54.386875 |
| 1 | Al Manhal | 24.465607 | 54.365719 |
| 2 | Al Khalidiyah | 24.470393 | 54.349521 |
| 3 | AL Hisn | 24.484465 | 54.355500 |
| 4 | Al Bateen | 24.450852 | 54.355163 |

[70]:
```
AD_Areas.shape[0]
```

[70]: 32

**Use geopy library to get the latitude and longitude values of Abu Dhabi City.** In order to define an instance of the geocoder, we need to define a user_agent. We will name our agent AD_explorer, as shown below.

[71]:
```
address = 'Abu Dhabi'

geolocator = Nominatim(user_agent="AD_explorer")
location = geolocator.geocode(address)
latitude = location.latitude
longitude = location.longitude
print('The geograpical coordinate of Abu Dhabi are {}, {}.'.format(latitude,␣
 ↪longitude))
```

```
The geograpical coordinate of Abu Dhabi are 24.4747961, 54.3705762.
```

**Create a map of Abu Dhabi with neighborhoods superimposed on top.**

[72]:
```
# create map of AD using latitude and longitude values
radius = 1000
```

```
map_AD = folium.Map(location=[latitude, longitude], zoom_start=11)
# add markers to map
for lat, lng, neighborhood in zip(AD_Areas['Latitude'], AD_Areas['Longitude'],␣
 ↪AD_Areas['Area']):
    label = '{}'.format(neighborhood)
    label = folium.Popup(label, parse_html=True)
    folium.CircleMarker(
        [lat, lng],
        radius=5,
        popup=label,
        color='blue',
        fill=True,
        fill_color='#3186cc',
        fill_opacity=0.3,
        parse_html=False).add_to(map_AD)
    folium.Circle(
        [lat, lng],
        radius=radius).add_to(map_AD)


map_AD
```

[72]: `<folium.folium.Map at 0x7f48d4b5c710>`

Next, we are going to start utilizing the Foursquare API to explore the neighborhoods and segment them.

**Define Foursquare Credentials and Version**

[1]:
```
CLIENT_ID = '' # your Foursquare ID
CLIENT_SECRET = '' # your Foursquare Secret
VERSION = '20200605'

print('Your credentails:')
print('CLIENT_ID: ' + CLIENT_ID)
print('CLIENT_SECRET:' + CLIENT_SECRET)
```

```
Your credentails:
CLIENT_ID:
CLIENT_SECRET:
```

**Let's explore the Second neighborhood in our dataframe.** Get the neighborhood's name.

[74]:
```
AD_Areas.loc[2, 'Area']
```

[74]: `'Al Khalidiyah'`

Get the neighborhood's latitude and longitude values.

[75]:
```
neighborhood_latitude = AD_Areas.loc[2, 'Latitude'] # neighborhood latitude␣
 ↪value
neighborhood_longitude = AD_Areas.loc[2, 'Longitude'] # neighborhood longitude␣
 ↪value
search_query = 'Starbucks' # search query , in this case we want to look for␣
 ↪gas stations in area
neighborhood_name = AD_Areas.loc[2, 'Area'] # neighborhood name

print('Latitude and longitude values of {} are {}, {}.'.
 ↪format(neighborhood_name,

                                                                  ␣
 ↪neighborhood_latitude,

                                                                  ␣
 ↪neighborhood_longitude))
```

```
Latitude and longitude values of Al Khalidiyah are 24.470392999999998,
54.349520999999996.
```

**Now, let's Search for the first option in our list "StarBucks" in Abu Dhabi** First, let's create the GET request URL. Name your URL **url**.

[2]:
```
# type your answer here
radius = 7000
SB_url = 'https://api.foursquare.com/v2/venues/explore?
 ↪client_id={}&client_secret={}&near={}&v={}&query={}&radius={}'.format(
    CLIENT_ID,
    CLIENT_SECRET,
    'Abu Dhabi',
    VERSION,
    search_query,
    radius,)
SB_url
```

```
      ␣
↪---------------------------------------------------------------------------

      NameError                                 Traceback (most recent call␣
↪last)

      <ipython-input-2-07f4c2e96d1a> in <module>
        6     'Abu Dhabi',
        7     VERSION,
   ----> 8     search_query,
        9     radius,)
       10 SB_url
```

```
NameError: name 'search_query' is not defined
```

Send the GET request and examine the Starbucks resutls in AD

[77]:
```
SB_results = requests.get(SB_url).json()
```

From the Foursquare lab in the previous module, we know that all the information is in the *items* key. Before we proceed, let's borrow the **get_category_type** function from the Foursquare lab.

[78]:
```
# 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']
```

Now we are ready to clean the json and structure it into a *pandas* dataframe.

[79]:
```
venues = SB_results['response']['groups'][0]['items']

SB_AD = json_normalize(venues) # flatten JSON

# filter columns
filtered_columns = ['venue.name', 'venue.categories', 'venue.location.lat',␣
 ↪'venue.location.lng']
SB_AD =SB_AD.loc[:, filtered_columns]

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

# clean columns
SB_AD.columns = [col.split(".")[-1] for col in SB_AD.columns]

SB_AD.head()
```

```
/home/jupyterlab/conda/envs/python/lib/python3.6/site-
packages/ipykernel_launcher.py:3: FutureWarning: pandas.io.json.json_normalize
is deprecated, use pandas.json_normalize instead
  This is separate from the ipykernel package so we can avoid doing imports
until
```

[79]:
```
                  name   categories        lat        lng
0  Starbucks (    )  Coffee Shop  24.477430  54.371626
1  Starbucks (    )  Coffee Shop  24.479700  54.353800
2  Starbucks (    )  Coffee Shop  24.468903  54.340355
3            Starbucks  Coffee Shop  24.470100  54.351700
4            Starbucks  Coffee Shop  24.477500  54.376700
```

And how many Starbucks location were returned by Foursquare?

[80]:
```
print('{} Starbucks location were returned by Foursquare.'.format(SB_AD.
 ↪shape[0]))
```

```
30 Starbucks location were returned by Foursquare.
```

### 1.2 Lets check the Map showing the venues in the area searched in:

[81]:
```
# create map of Area using latitude and longitude values

map_AD = folium.Map(location=[latitude, longitude], zoom_start=12)
# add markers to map
for lat, lng, Name in zip(SB_AD['lat'], SB_AD['lng'], SB_AD['name']):
    label = '{}'.format(Name)
    label = folium.Popup(label, parse_html=True)
    folium.Marker(        # to mark All the venues in Search
        [lat, lng],
        #radius=5,
        popup=label,
        icon=folium.Icon(color='green'),
    ).add_to(map_AD)
map_AD
```

[81]:
```
<folium.folium.Map at 0x7f48d5323080>
```

### 1.3 Now let's explore one neighborhood we mentioned earlier our second in the dataframe "Al Khalidiyah"

#### 1.3.1 we will check the area with Starbucks and circle the neighbor to show any Starbucks within the neighbor

[82]:
```
map_Khalidiyah = folium.Map(location=[neighborhood_latitude,␣
 ↪neighborhood_longitude], zoom_start=16) # create map of Area using latitude␣
 ↪and longitude values

for lat, lng, Name in zip(SB_AD['lat'], SB_AD['lng'], SB_AD['name']):
    label = '{}'.format(Name)
    label = folium.Popup(label, parse_html=True)
    folium.Marker(        # to mark All the venues in Search
        [lat, lng],
```

```
        #radius=5,
        icon=folium.Icon(color='green'),
        popup=label,).add_to(map_Khalidiyah)

folium.CircleMarker(        # to mark Khalidiyah neighborhood and show the␣
 ↪radius of 1000m
        [neighborhood_latitude, neighborhood_longitude],
         popup=neighborhood_name,
        color='blue',
        fill=True,
        fill_color='#3186cc',
        fill_opacity=0.3,
        parse_html=False).add_to(map_Khalidiyah)
folium.Circle(
        [neighborhood_latitude, neighborhood_longitude],
        radius=1000).add_to(map_Khalidiyah)


map_Khalidiyah
```

[82]: <folium.folium.Map at 0x7f49095d5a58>

### 1.4 2. Let's Explore other international Coffeehouses in the City

**Our second coffehouse will be "Costa" let's repeat the same process of getting Foursquare Data for "Costa"**

**And then search for the Third Option which is "Tim Hortons"**

[83]:
```
#Request Url for Costra
radius = 7000
Costa_url = 'https://api.foursquare.com/v2/venues/explore?
 ↪client_id={}&client_secret={}&near={}&v={}&query={}&radius={}'.format(
    CLIENT_ID,
    CLIENT_SECRET,
    'Abu Dhabi',
    VERSION,
    'Costa',
    radius,)
Costa_url
```

[83]: 'https://api.foursquare.com/v2/venues/explore?client_id=2BVRTGOAHH23YASLTSJYEWCL
NNNGW2FGSLOLLGE5WWOIP5ZH&client_secret=UREPWHYPXIIZNLGEPTOCG15HRCOT1OQMIMMRYLZGZ
B4J4LBC&near=Abu Dhabi&v=20200605&query=Costa&radius=7000'

[84]:
```
Costa_results = requests.get(Costa_url).json()
```

[85]:
```
venues = Costa_results['response']['groups'][0]['items']

Costa_AD = json_normalize(venues) # flatten JSON

# filter columns
filtered_columns = ['venue.name', 'venue.categories', 'venue.location.lat',␣
 ↪'venue.location.lng']
Costa_AD =Costa_AD.loc[:, filtered_columns]

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

# clean columns
Costa_AD.columns = [col.split(".")[-1] for col in Costa_AD.columns]
```

```
/home/jupyterlab/conda/envs/python/lib/python3.6/site-
packages/ipykernel_launcher.py:3: FutureWarning: pandas.io.json.json_normalize
is deprecated, use pandas.json_normalize instead
  This is separate from the ipykernel package so we can avoid doing imports
until
```

### 1.4.1 When checking the data we can see that now only Costa appears in the results, "Tim Hortons" & other categories are there included.

let's clean the data before we proceed with maping the results. first lets check the data

[86]:
```
Costa_AD.head() # Data clearly needs to be cleaned, in first 5 rows there is␣
 ↪"Starbucks & Tim Hortons"
```

[86]:

| | name | categories | lat | lng |
|---|---|---|---|---|
| 0 | Starbucks (  ) | Coffee Shop | 24.477430 | 54.371626 |
| 1 | Tim Hortons | Coffee Shop | 24.489834 | 54.370388 |
| 2 | Costa Coffee | Coffee Shop | 24.468595 | 54.339754 |
| 3 | Costa Coffee \| | Coffee Shop | 24.496585 | 54.374138 |
| 4 | Costa Coffee | Coffee Shop | 24.475618 | 54.373293 |

[87]:
```
Costa_AD= Costa_AD[Costa_AD.name.str.contains("Costa")] # to keep only "name"␣
 ↪with Costa
```

[88]:
```
Costa_AD.reset_index(drop=True, inplace=True)
Costa_AD.head()
```

[88]:

| | name | categories | lat | lng |
|---|---|---|---|---|
| 0 | Costa Coffee | Coffee Shop | 24.468595 | 54.339754 |
| 1 | Costa Coffee \| | Coffee Shop | 24.496585 | 54.374138 |
| 2 | Costa Coffee | Coffee Shop | 24.475618 | 54.373293 |
| 3 | Costa Coffee | Coffee Shop | 24.487706 | 54.357722 |
| 4 | Costa Coffee | Coffee Shop | 24.468607 | 54.339678 |

### 1.5 Now we can see Costa data is cleaned and we can proceed with the next step

#### 1.5.1 "Tim Hortons" will be the last coffehouse to add to our data,

#### 1.5.2 we will repeat same step we did for "Starbucks" & "Costa"

[89]:
```
#Request Url for Costra
radius = 7000
TH_url = 'https://api.foursquare.com/v2/venues/explore?
↪client_id={}&client_secret={}&near={}&v={}&query={}&radius={}'.format(
    CLIENT_ID,
    CLIENT_SECRET,
    'Abu Dhabi',
    VERSION,
    'Tim Hortons',
    radius,)
TH_url
```

[89]:
```
'https://api.foursquare.com/v2/venues/explore?client_id=2BVRTGOAHH23YASLTSJYEWCL
NNNGW2FGSLOLLGE5WWOIP5ZH&client_secret=UREPWHYPXIIZNLGEPTOCG15HRCOT1OQMIMMRYLZGZ
B4J4LBC&near=Abu Dhabi&v=20200605&query=Tim Hortons&radius=7000'
```

[90]:
```
TH_results = requests.get(TH_url).json()
```

[91]:
```
venues = TH_results['response']['groups'][0]['items']

TH_AD = json_normalize(venues) # flatten JSON

# filter columns
filtered_columns = ['venue.name', 'venue.categories', 'venue.location.lat',␣
↪'venue.location.lng']
TH_AD =TH_AD.loc[:, filtered_columns]

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

# clean columns
TH_AD.columns = [col.split(".")[-1] for col in TH_AD.columns]
```

```
/home/jupyterlab/conda/envs/python/lib/python3.6/site-
packages/ipykernel_launcher.py:3: FutureWarning: pandas.io.json.json_normalize
is deprecated, use pandas.json_normalize instead
  This is separate from the ipykernel package so we can avoid doing imports
until
```

[92]:
```
TH_AD.head() # checking the data we can see all the results matches with the␣
↪search inquiry "Tim Hortons"
```

[92]:

| | name | categories | lat | lng |
|---|---|---|---|---|
| 0 | Tim Hortons | Coffee Shop | 24.495146 | 54.382863 |
| 1 | Tim Hortons | Coffee Shop | 24.489834 | 54.370388 |
| 2 | Tim Hortons | Coffee Shop | 24.470300 | 54.372760 |
| 3 | Tim Hortons | Coffee Shop | 24.491033 | 54.362142 |
| 4 | Tim Hortons | Coffee Shop | 24.477351 | 54.371574 |

### 1.6 Getting back to AD Map , lets Show AD Map and include all the three Coffeehouses in the Map

we will start with adding "Costa" and setting icon color to Red to distnguish it from Green "Starbucks"

[93]:
```
for lat, lng, Name in zip(Costa_AD['lat'], Costa_AD['lng'], Costa_AD['name']):
    label = '{}'.format(Name)
    label = folium.Popup(label, parse_html=True)
    folium.Marker(        # to mark All the venues in Search
        [lat, lng],
        #radius=5,
        popup=label,
        icon=folium.Icon(color='red'),
    ).add_to(map_AD)
map_AD
```

[93]: <folium.folium.Map at 0x7f48d5323080>

we will start with adding "Tim Hortons" and setting icon color to light red

[94]:
```
for lat, lng, Name in zip(TH_AD['lat'], TH_AD['lng'], TH_AD['name']):
    label = '{}'.format(Name)
    label = folium.Popup(label, parse_html=True)
    folium.Marker(        # to mark All the venues in Search
        [lat, lng],
        #radius=5,
        popup=label,
        icon=folium.Icon(color='lightred'),
    ).add_to(map_AD)
map_AD
```

[94]: <folium.folium.Map at 0x7f48d5323080>

Addng the Neighboors on the maps!

[95]:
```
for lat, lng, neighborhood in zip(AD_Areas['Latitude'], AD_Areas['Longitude'],␣
↪AD_Areas['Area']):
    label = '{}'.format(neighborhood)
    label = folium.Popup(label, parse_html=True)
    folium.CircleMarker(
```

```
        [lat, lng],
        radius=5,
        popup=label,
        color='blue',
        fill=True,
        fill_color='#3186cc',
        fill_opacity=0.7,
        parse_html=False).add_to(map_AD)
    folium.Circle(
        [lat, lng],
        radius=1000).add_to(map_AD)
map_AD
```

[95]:
```
<folium.folium.Map at 0x7f48d5323080>
```

[ ]:
```

```

[ ]:
```

```

[96]:
```
def getNearbyVenues(names, latitudes, longitudes, radius=1500):

    venues_list=[]
    for name, lat, lng in zip(names, latitudes, longitudes):
        print(name)

        # create the API request URL
        url = 'https://api.foursquare.com/v2/venues/explore?
↪client_id={}&client_secret={}&ll={},{}&v={}&query={}&radius={}'.format(
        #url = 'https://api.foursquare.com/v2/venues/explore?
↪client_id={}&client_secret={}&ll={},{}&v={}&radius={}&limit={}'.format(
            CLIENT_ID,
            CLIENT_SECRET,
            lat,
            lng,
            VERSION,
            'Coffee Shop',
            radius,
            )

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

        # return only relevant information for each nearby venue
        venues_list.append([(
            name,
            lat,
            lng,
```

```
            v['venue']['name'],
            v['venue']['location']['lat'],
            v['venue']['location']['lng'],
            v['venue']['categories'][0]['name']) for v in results])

    nearby_venues = pd.DataFrame([item for venue_list in venues_list for item␣
↪in venue_list])
    nearby_venues.columns = ['Neighborhood',
                  'Neighborhood Latitude',
                  'Neighborhood Longitude',
                  'Venue',
                  'Venue Latitude',
                  'Venue Longitude',
                  'Venue Category']

    return(nearby_venues)
```

**Now write the code to run the above function on each neighborhood and create a new dataframe called *AD_venues*.**

[97]:
```
# type your answer here
AD_Coffee = getNearbyVenues(names=AD_Areas['Area'],
                                    latitudes=AD_Areas['Latitude'],
                                    longitudes=AD_Areas['Longitude']
                                   )
```

```
Al Mushrif
Al Manhal
Al Khalidiyah
AL Hisn
Al Bateen
Al Marina
Al Nahyan Camp
Al Karamah
Al Danah
Al Reem island
Al Zahiyah
Zayed Port
Al Maqta
Mangrove Village
Khalifa City
New Al Falah
Yas North
Al Shahama
Al Bahyah
Al Raha
Al Falah
AL Saadah
```

```
Al Maarid
Al Muntazah
Mussafah
MBZ
Shakhbout City
Al Bateen-Cornish
Zayed Sport City
Al Qurm
Maryah Island
Al Mushrif- Arab Gulf road
```

[98]:
```
AD_Coffee.shape[0]
```

[98]:
```
557
```

Double-click **here** for the solution.

**Let's check the size of the resulting dataframe**

[99]:
```
AD_Coffee_Costa= AD_Coffee[AD_Coffee.Venue.str.contains("Costa")]
AD_Coffee_TH= AD_Coffee[AD_Coffee.Venue.str.contains("Tim Hortons")]
AD_Coffee_SB= AD_Coffee[AD_Coffee.Venue.str.contains("Starbucks")]
AD_Coffee_Costa['Coffee House'] = 'Costa'
AD_Coffee_TH['Coffee House'] = 'Tim Hortons'
AD_Coffee_SB['Coffee House'] = 'Starbucks'

AD_Coffee_Costa.reset_index(drop=True, inplace=True)
AD_Coffee_TH.reset_index(drop=True, inplace=True)
AD_Coffee_SB.reset_index(drop=True, inplace=True)
```

```
/home/jupyterlab/conda/envs/python/lib/python3.6/site-
packages/ipykernel_launcher.py:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  after removing the cwd from sys.path.
/home/jupyterlab/conda/envs/python/lib/python3.6/site-
packages/ipykernel_launcher.py:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  """
/home/jupyterlab/conda/envs/python/lib/python3.6/site-
packages/ipykernel_launcher.py:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
```

```
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
```

[100]:
```
print('Number of Costa Places in range of AD Neighborhoods is {}. ' .
 ↪format(AD_Coffee_Costa.shape[0]) )
print('Number of Starbucks Places in range of AD Neighborhoods is {}. ' .
 ↪format(AD_Coffee_SB.shape[0]) )
print('Number of Tim Hortons Places in range of AD Neighboorhoods is {}.' .
 ↪format(AD_Coffee_TH.shape[0]) )
```

```
Number of Costa Places in range of AD Neighborhoods is 41.
Number of Starbucks Places in range of AD Neighborhoods is 39.
Number of Tim Hortons Places in range of AD Neighboorhoods is 25.
```

[101]:
```
AD_Coffee = AD_Coffee_Costa.append(AD_Coffee_TH)
AD_Coffee = AD_Coffee.append(AD_Coffee_SB)
```

[102]:
```
AD_Coffee_Costa['Neighborhood'].unique()
```

[102]:
```
array(['Al Mushrif', 'Al Manhal', 'Al Khalidiyah', 'AL Hisn',
       'Al Karamah ', 'Al Danah', 'Al Zahiyah', 'Al Maqta', 'Al Falah ',
       'AL Saadah', 'Al Maarid', 'Al Muntazah', 'Al Bateen-Cornish',
       'Zayed Sport City', 'Al Qurm', 'Maryah Island',
       'Al Mushrif- Arab Gulf road'], dtype=object)
```

[103]:
```
AD_Coffee_SB['Neighborhood'].unique()
```

[103]:
```
array(['Al Mushrif', 'Al Manhal', 'Al Khalidiyah', 'AL Hisn',
       'Al Nahyan Camp', 'Al Danah', 'Al Reem island', 'Al Zahiyah',
       'Al Maqta', 'New Al Falah', 'Al Shahama', 'Al Falah ', 'AL Saadah',
       'Al Maarid', 'Al Muntazah', 'Al Bateen-Cornish', 'Al Qurm',
       'Maryah Island', 'Al Mushrif- Arab Gulf road'], dtype=object)
```

[104]:
```
AD_Coffee_TH['Neighborhood'].unique()
```

[104]:
```
array(['Al Manhal', 'AL Hisn', 'Al Marina', 'Al Nahyan Camp',
       'Al Karamah ', 'Al Danah', 'Al Zahiyah', 'Yas North', 'Al Shahama',
       'Al Falah ', 'AL Saadah', 'Al Muntazah', 'Maryah Island',
       'Al Mushrif- Arab Gulf road'], dtype=object)
```

Let's check how many venues were returned for each neighborhood

[105]:
```
AD_Coffee.reset_index(drop=True, inplace = True)
AD_Coffee.head()
```

[105]:
```
     Neighborhood  Neighborhood Latitude  Neighborhood Longitude  \
0     Al Mushrif              24.443699               54.386875
1      Al Manhal              24.465607               54.365719
2      Al Manhal              24.465607               54.365719
3  Al Khalidiyah              24.470393               54.349521
4  Al Khalidiyah              24.470393               54.349521

                              Venue  Venue Latitude  Venue Longitude  \
0  Costa Coffee - Khalifa University       24.446952        54.394503
1                       Costa Coffee       24.475618        54.373293
2            Costa Coffee - Al Wahda       24.470395        54.372694
3                       Costa Coffee       24.468595        54.339754
4                       Costa Coffee       24.468607        54.339678

  Venue Category Coffee House
0    Coffee Shop        Costa
1    Coffee Shop        Costa
2    Coffee Shop        Costa
3    Coffee Shop        Costa
4    Coffee Shop        Costa
```

### 1.7 3. Analyze Each Neighborhood

[106]:
```
# one hot encoding
AD_onehot = pd.get_dummies(AD_Coffee[['Coffee House']], prefix="",␣
 ↪prefix_sep="")

# add neighborhood column back to dataframe
AD_onehot['Neighborhood'] = AD_Coffee['Neighborhood']

# move neighborhood column to the first column
fixed_columns = [AD_onehot.columns[-1]] + list(AD_onehot.columns[:-1])
AD_onehot = AD_onehot[fixed_columns]

AD_onehot.head()
```

[106]:
```
     Neighborhood  Costa  Starbucks  Tim Hortons
0     Al Mushrif      1          0            0
1      Al Manhal      1          0            0
2      Al Manhal      1          0            0
3  Al Khalidiyah      1          0            0
4  Al Khalidiyah      1          0            0
```

And let's examine the new dataframe size.

[107]:
```
AD_onehot.shape
```

[107]:
```
(105, 4)
```

**Next, let's group rows by neighborhood and by taking the mean of the frequency of occurrence of each category**

[108]:
```
AD_grouped = AD_onehot.groupby('Neighborhood').mean().reset_index()
AD_grouped.head()
```

[108]:

| | Neighborhood | Costa | Starbucks | Tim Hortons |
|---|---|---|---|---|
| 0 | AL Hisn | 0.285714 | 0.428571 | 0.285714 |
| 1 | AL Saadah | 0.333333 | 0.333333 | 0.333333 |
| 2 | Al Bateen-Cornish | 0.400000 | 0.600000 | 0.000000 |
| 3 | Al Danah | 0.200000 | 0.400000 | 0.400000 |
| 4 | Al Falah | 0.250000 | 0.125000 | 0.625000 |

**Let's confirm the new size**

[109]:
```
AD_grouped.shape
```

[109]:
```
(23, 4)
```

**Let's print each neighborhood along with the top 5 most common venues**

[110]:
```
num_top_venues = 3

for hood in AD_grouped['Neighborhood']:
    print("----"+hood+"----")
    temp = AD_grouped[AD_grouped['Neighborhood'] == hood].T.reset_index()
    temp.columns = ['venue','freq']
    temp = temp.iloc[1:]
    temp['freq'] = temp['freq'].astype(float)
    temp = temp.round({'freq': 2})
    print(temp.sort_values('freq', ascending=False).reset_index(drop=True).
 ↪head(num_top_venues))
    print('\n')
```

```
----AL Hisn----
         venue  freq
0    Starbucks  0.43
1        Costa  0.29
2  Tim Hortons  0.29


----AL Saadah----
         venue  freq
0        Costa  0.33
1    Starbucks  0.33
2  Tim Hortons  0.33
```

```
----Al Bateen-Cornish----
         venue  freq
0    Starbucks   0.6
1        Costa   0.4
2  Tim Hortons   0.0


----Al Danah----
         venue  freq
0    Starbucks   0.4
1  Tim Hortons   0.4
2        Costa   0.2


----Al Falah ----
         venue  freq
0  Tim Hortons  0.62
1        Costa  0.25
2    Starbucks  0.12


----Al Karamah ----
         venue  freq
0        Costa   0.5
1  Tim Hortons   0.5
2    Starbucks   0.0


----Al Khalidiyah----
         venue  freq
0    Starbucks  0.67
1        Costa  0.33
2  Tim Hortons  0.00


----Al Maarid----
         venue  freq
0        Costa  0.75
1    Starbucks  0.25
2  Tim Hortons  0.00


----Al Manhal----
         venue  freq
0    Starbucks  0.43
1        Costa  0.29
```

```
2  Tim Hortons  0.29


----Al Maqta----
         venue  freq
0        Costa   0.5
1    Starbucks   0.5
2  Tim Hortons   0.0


----Al Marina----
         venue  freq
0  Tim Hortons   1.0
1        Costa   0.0
2    Starbucks   0.0


----Al Muntazah----
         venue  freq
0        Costa  0.56
1    Starbucks  0.33
2  Tim Hortons  0.11


----Al Mushrif----
         venue  freq
0        Costa   0.5
1    Starbucks   0.5
2  Tim Hortons   0.0


----Al Mushrif- Arab Gulf road----
         venue  freq
0        Costa   0.4
1    Starbucks   0.4
2  Tim Hortons   0.2


----Al Nahyan Camp----
         venue  freq
0    Starbucks   0.5
1  Tim Hortons   0.5
2        Costa   0.0


----Al Qurm----
         venue  freq
0    Starbucks  0.75
```

```
1        Costa  0.25
2  Tim Hortons  0.00


----Al Reem island----
         venue  freq
0    Starbucks   1.0
1        Costa   0.0
2  Tim Hortons   0.0


----Al Shahama----
         venue  freq
0    Starbucks   0.5
1  Tim Hortons   0.5
2        Costa   0.0


----Al Zahiyah----
         venue  freq
0        Costa  0.50
1  Tim Hortons  0.38
2    Starbucks  0.12


----Maryah Island----
         venue  freq
0        Costa   0.6
1    Starbucks   0.2
2  Tim Hortons   0.2


----New Al Falah----
         venue  freq
0    Starbucks   1.0
1        Costa   0.0
2  Tim Hortons   0.0


----Yas North----
         venue  freq
0  Tim Hortons   1.0
1        Costa   0.0
2    Starbucks   0.0


----Zayed Sport City----
         venue  freq
```

```
0        Costa   1.0
1    Starbucks   0.0
2  Tim Hortons   0.0
```

**Let's put that into a *pandas* dataframe** First, let's write a function to sort the venues in descending order.

[111]:
```
def return_most_common_venues(row, num_top_venues):
    row_categories = row.iloc[1:]
    row_categories_sorted = row_categories.sort_values(ascending=False)

    return row_categories_sorted.index.values[0:num_top_venues]
```

Now let's create the new dataframe and display the top 10 venues for each neighborhood.

[112]:
```
num_top_venues = 3

indicators = ['st', 'nd', 'rd']

# create columns according to number of top venues
columns = ['Neighborhood']
for ind in np.arange(num_top_venues):
    try:
        columns.append('{}{} Most Common Venue'.format(ind+1, indicators[ind]))
    except:
        columns.append('{}th Most Common Venue'.format(ind+1))

# create a new dataframe
neighborhoods_venues_sorted = pd.DataFrame(columns=columns)
neighborhoods_venues_sorted['Neighborhood'] = AD_grouped['Neighborhood']

for ind in np.arange(AD_grouped.shape[0]):
    neighborhoods_venues_sorted.iloc[ind, 1:] =␣
 ↪return_most_common_venues(AD_grouped.iloc[ind, :], num_top_venues)

neighborhoods_venues_sorted.head()
```

[112]:
```
         Neighborhood 1st Most Common Venue 2nd Most Common Venue  \
0            AL Hisn             Starbucks           Tim Hortons
1          AL Saadah           Tim Hortons             Starbucks
2  Al Bateen-Cornish             Starbucks                 Costa
3           Al Danah           Tim Hortons             Starbucks
4          Al Falah           Tim Hortons                 Costa

  3rd Most Common Venue
0                 Costa
```

```
1                 Costa
2           Tim Hortons
3                 Costa
4             Starbucks
```

[113]:
```
neighborhoods_venues_sorted.head()
```

[113]:
```
        Neighborhood 1st Most Common Venue 2nd Most Common Venue  \
0            AL Hisn             Starbucks           Tim Hortons
1          AL Saadah           Tim Hortons             Starbucks
2  Al Bateen-Cornish             Starbucks                 Costa
3           Al Danah           Tim Hortons             Starbucks
4          Al Falah            Tim Hortons                 Costa

  3rd Most Common Venue
0                 Costa
1                 Costa
2           Tim Hortons
3                 Costa
4             Starbucks
```

### 1.8 Now let's find out which Areas are missing any of these Coffeehouses in it.

[114]:
```
AD_Neighborhood = AD_Areas['Area'].tolist() # to list all the Neighborhoods as␣
 ↪a list.
AD_Neigh_Coffee = AD_grouped['Neighborhood'].tolist() # List the Neighborhoods␣
 ↪that has one of the three Coffeeplaces

AD_No_Coffee=[]
for Area in AD_Neighborhood:
    if Area not in AD_Neigh_Coffee:
        AD_No_Coffee.append(Area)
```

[115]:
```
AD_No_Coffee # these are the list of areas that lack the existance of
```

[115]:
```
['Al Bateen',
 'Zayed Port',
 'Mangrove Village',
 'Khalifa City',
 'Al Bahyah',
 'Al Raha',
 'Mussafah',
 'MBZ',
 'Shakhbout City']
```

[116]:
```
print('{} Areas has no coffee places within the neighborhood'.
 ↪format(len(AD_No_Coffee)))
```

```
9 Areas has no coffee places within the neighborhood
```