Keyu Zhu. The Survey of Tenants in Chapel Hill. A Master's Paper for the M.S. in I.S degree. April, 2022. 30 pages Advisor: Marchionini, Gary

Pandas is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series. I hope to solve some of the existing problems by analyzing the data with pandas, a useful tool. In this document, I will show use the functions of pandas and show the code used to analyze the data, the output, and the renderings

Headings:

Pandas

Data analysis

Tenants of Chapel Hill

THE SURVEY OF TENANTS IN CHAPEL HILL

by
Keyu Zhu

A Master's paper submitted to the faculty
of the School of Information and Library Science
of the University of North Carolina at Chapel Hill
in partial fulfillment of the requirements
for the degree of Master of Science in
Information Science.

Chapel Hill, North Carolina

April 2022

Approved by

_____

Marchionini, Gary

Table of Contents

1. LITERATure REVIEW

In this part, I searched for literature and projects related to my subject. My main tool is pandas and its function modules etc. I hope to find relevant literature that can guide for example application, data screening, building dataframe, etc. The case I found of applying pandas is air quality. The author uses the pandas amount folium to show the changes in air quality in India. This project uses two Python libraries. The first is Pandas, which is a well-known library for data science. But it also has capabilities for downloading data and even plotting data. The second library is Folium. The author's steps in analyzing the data give some inspiration to my topic. The script is given below:

1. Download the air quality index (AQI) data. This step consists of the following sub-steps:

* Get a token from the aqicn.org website.

* Understand how to construct the API for downloading data.

* Create the bounding box for which the AQI data is to be downloaded.

* Use the read_json() function of Pandas to make an API call.

2. Create a Pandas DataFrame from the downloaded data.

3. Clean the data.

4. Create a heat map.

5. Plot the measuring stations on the Folium map.

The first step is to download the AQI data. The author gets this from the World Air Quality Index project at *https://aqicn.org/.* The website provides many different APIs to download data. The author notices that Pandas can be used to directly download data from a URL and the data can be downloaded in a variety of formats. The author uses a Pandas function read_json(). The data has two columns- status and data. The data column is in fact a dictionary, which is of interest. So, the author needs to convert this column (in the form of a dictionary) to another Pandas DataFrame, which will do in the next step.

```
import pandas as pd
import folium
from folium.plugins import HeatMap
###-STEP 1 DOWNLOAD DATA
# See details of API at:- https://aqicn.org/api/
base_url = "https://api.waqi.info"
# Get token from:- https://aqicn.org/data-platform/token/#/
tok = '5c173abcXXXXXXX XXXXXXXX402471de46bb352f6'
# (lat, long)- > bottom left, (lat, lon)- > top right
# India is 8N 61E to 37N, 97E approx
latlngbox = "8.0000,61.0000,37.0000,97.0000" # For India
trail_url=f"/map/bounds/?latlng={latlngbox}&token={tok}" #
my_data = pd.read_json(base_url + trail_url) # Join parts of URL
```

print('columns- > ', my_data.columns) #2 cols 'status' and 'data'

The second step is to create a Dataframe table. Even though the downloaded data is a DataFrame, the author still needs to create another DataFrame that has the following four columns: station name, station latitude, station longitude and the air quality index. To extract these values, the author uses a for loop. The one point to note is that the value corresponding to the key station is itself a dictionary; so in order to extract the station name, the author uses two keys of the type [each_ row['station']['name'].

###-STEP 2: Create table like DataFrame

all_rows = []

for each_row in my_data['data']:

all_rows.append([each_row['station']['name'],

each_row['lat'],

each_row['lon'],

each_row['aqi']])

df = pd.DataFrame(all_rows,

columns=['station_name', 'lat', 'lon', 'aqi'])

The third step is to clean the data. The heat map that the author intends to create must have a numeric value for the column aqi. So he converts all values in the column aqi to numerics. Moreover, in case a particular value for the column aqi cannot be coerced

into a numeric, he replaces it with a NaN (Not a Number). To do this, for the DataFrame method to_numeric(), we must provide a parameter errors with a value of coerce. Having done that, the author also needs to remove all such rows that have a value of NaN for the column aqi. This can be done by the DataFrame method df.dropna(subset = ['aqi']). The original DataFrame was df and its shape was (152, 4), while the new DataFrame (with NaN dropped) is df1 and its shape is (144, 4). So, the author concludes that there were eight rows in the DataFrame df1 which had a NaN for the column aqi.

```
###-STEP 3:- Clean the DataFrame
df['aqi'] = pd.to_numeric(df.aqi,
errors='coerce') # Invalid parsing to NaN
print('with NaN- > ', df.shape) # Comes out as (152, 4)
# Remove NaN (Not a Number) entries in col
df1 = df.dropna(subset = ['aqi'])
print('without NaN- > ', df1.shape) # (144, 4)
```

The step four is to plot the heat map. The Map class of the Folium library to create a map. We can have a map with a particular centre and a particular zoom level. The two parameters of importance are data and max_ val. The other parameters are only for changing the look of the heat map. For the data parameter, the author provides a DataFrame with three columns, namely, latitude, longitude and aqi. So he creates a new DataFrame df2 which has only three columns. For the parameter max_val, he provides

the maximum value in the aqi column (because the heat of the map will depend upon the value of this column). The other parameters are all by default and can be ignored:

```
###-STEP 4: Make folium heat map

init_loc = [23, 77] # Approx over Bhopal

max_aqi = int(df1['aqi'].max())

print('max_aqi- > ', max_aqi)

m = folium.Map(location = init_loc, zoom_start = 5)

heat_aqi = HeatMap(df2, min_opacity = 0.1, max_val = max_aqi,

radius = 20, blur = 20, max_zoom = 2)

m.add_child(heat_aqi)
```

m # Show the map

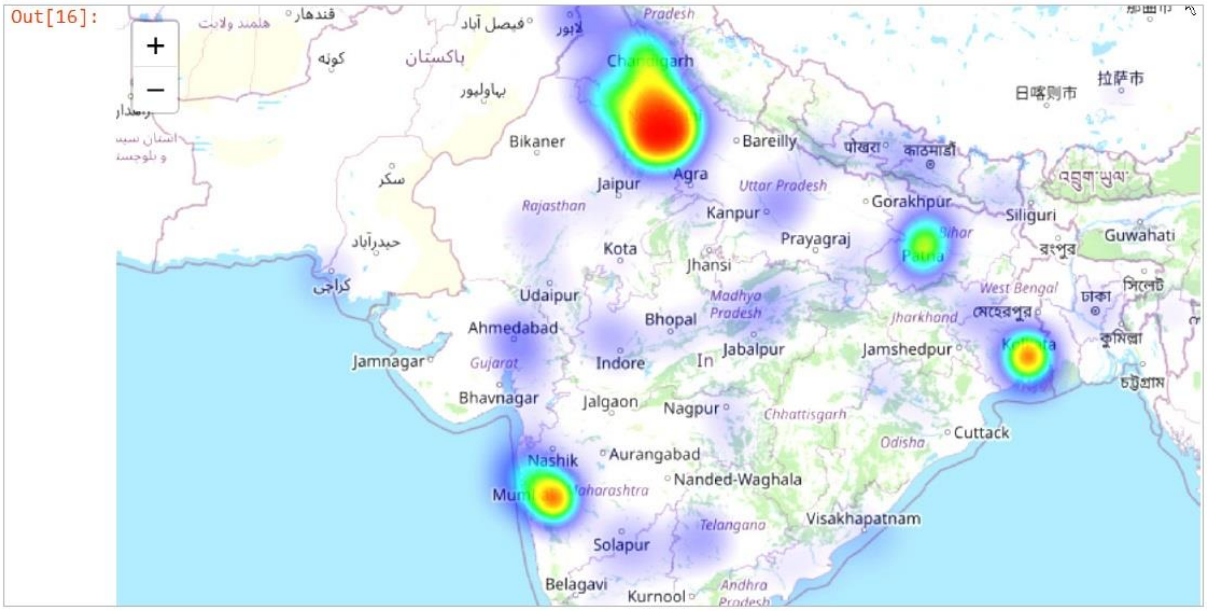


Figure 1: Heat map on Jupyter notebook

The author also provides another option which is plotting the stations if the heat map is not wanted. Again, the author uses the Map class to create the object. He adds the markers one by one by literating over them in a for loop. He have also coloured the markers as per the value of the aqi. If the value of aqi is above 300, the colour is red, and so on. The signature of the Marker class (along with the description of the two parameters of interest, namely, location and popup) is as below:

class folium.map.Marker(location, popup=None, tooltip=None,

*icon=None, draggable=False, **kwargs)*

Parameters

- *location (tuple or list) – Latitude and Longitude of*

Marker (Northing, Easting)

• popup (string or folium.Popup, default None) – Label

for the Marker; either an escaped HTML string to initialize

folium.Popup or a folium.Popup instance.

The code for this step is as follows:

```
###-STEP 5: Plot stations on map

centre_point = [23.25, 77.41] # Approx over Bhopal

m2 = folium.Map(location = centre_point,

tiles = 'Stamen Terrain',

zoom_start= 6)

for idx, row in df1.iterrows():

lat = row['lat']

lon = row['lon']

station = row['station_name'] + ' AQI=' + str(row['aqi'])

station_aqi = row['aqi']

if station_aqi > 300: ## Red for very bad AQI

pop_color = 'red'

elif station_aqi > 200:

pop_color = 'orange' ## Orange for moderate AQI

else:

pop_color = 'green' ## Green for good AQI
```

folium.Marker(location= [lat, lon],

popup = station,

icon = folium.Icon(color = pop_color)).

add_to(m2)

m2

Figure 2: Plotted station map

2. Overview

There are tenants in different parts of the city who offer residents services and facilities that promote their residential well-being. However, where the city residents are new, they may lack access to information which can guide them on where to access some of the services they need. So, in this paper, the focus of this project was to analyze the distribution of tenants in Chapel Hill and draw conclusions that could help people live better in Chapel Hill. This was a quantitative research where secondary data was obtained from Chapel Hill Open Data by the tool of Pandas. The study findings target the new residents of Chapel Hill by providing them with a guide on the different tenants in the three districts and the categories of these tenants.

2.1. Project Goals:

1. Analyze the current situation of the tenants in Chapel Hill
2. Compare the number and categories of tenants in the three districts(east, west, central)
3. Recommend living district to people with different needs

3. Data selection and processing:

As a resident of Chapel Hill for less than a year, I am very impressed with life in Chapel Hill and North Carolina. But I wanted to learn more about the city, so I wanted to use the tools I've learned to conduct in-depth analysis of all kinds of data from Chapel Hill. My project idea initially included the bicycle crash statistics of Chapel Hill and the shooting incident statistics of NC state. I think these are very good data sets which can be downloaded from Chapel Hill Open Data. But later, I hope to make a data analysis that is more relevant to myself and can help the same kind of people around me. I was lucky to find a useful dataset 'Downtown-tenants 'which is a CSV file from Chapel Hill Open Data. *https://opendata-townofchapelhill.hub.arcgis.com/datasets/downtown-tenants/explore?location=0.000000%2C0.000000%2C0.00*. This data records the rental time, usage and geographic location, open date, etc. of each tenant in detail. I think this data can help me locate the location and purpose of each tenant, and help people find a suitable place to live. Here is code for import pandas and folium:

```
import pandas as pd
import matplotlib.pyplot as plt
pd.set_option('display.max_rows', None)
pd.set_option('display.max_columns', None)

data = pd.read_csv('Downtown_Tenants.csv')
```

The picture below is the output but we still to make some filters because there is still some data we do not need.

JECTID	Tenant_Name	Category_of_Use	Street_Name	Address_Number	Downtown_District	Date_Opened	Date_Closed	Years_in_Business
1	Who's Next Barber Shop	Service	East Franklin Street	128.00	East	1/1/2009	NaN	8.032877
2	Bandido's Mexican Café	Restaurant	East Franklin Street	159.50	East	1/1/1995	NaN	22.043836
3	Smoke Rings	Retail	West Franklin Street	418.00	West	1/1/2011	NaN	6.032877
4	The Standard	Restaurant	West Rosemary Street	403.00	West	12/1/2011	2014/01/01 00:00:00	2.087671
5	Brown's Paint and Hardware	Retail	West Franklin Street	420.00	West	1/1/1966	NaN	51.063014

For example, some tenants are already closed, which is reflected in the column ‘date_closed’. I first need to filter out tenants with data in this column. The second is the categories of tenants (this part of the data processing is filtered one by one in the excel file, because the overall quantity is not large, and there is no better way), there are some categories such as ’church’ and ’culture destination’ I think there is no data processing value. ‘Culture destination’ mainly includes tourist attractions such as ‘The cave’, which I think is of little reference to living. After filtering, I left four categories, Restaurant, Service, Retail and Bar. Restaurant includes dining hall, coffee shop. Service includes barber shop, bank, clinic. Retail includes clothing stores, record stores, etc. Bar stands for bar. (Further data categories is also not possible here. I queried almost all tenant names to determine their use.)

4. Data analysis

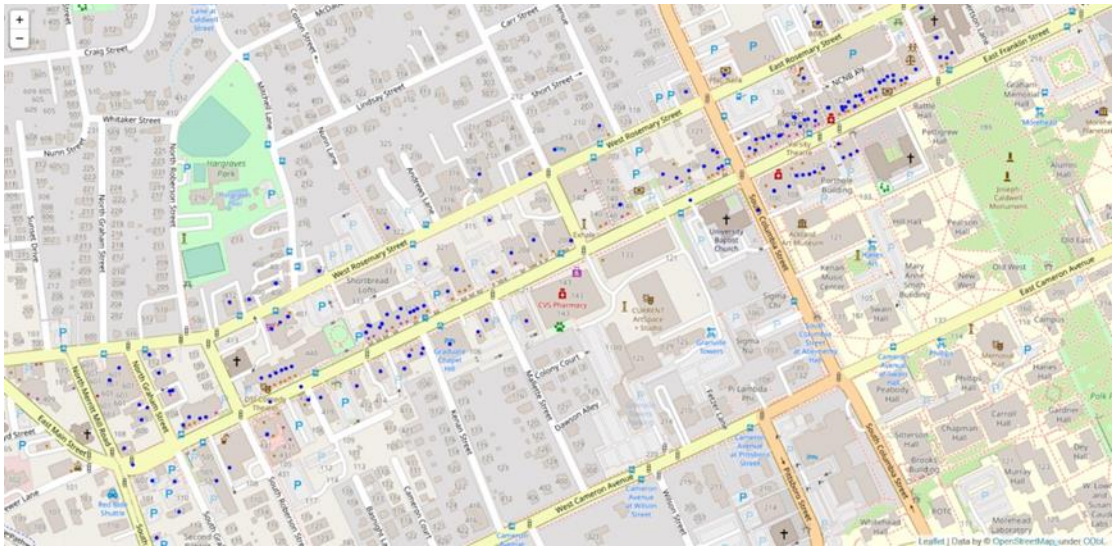
4.1 Current Distribution of Tenants

First of all, I want to take a look at the distribution of tenants in Chapel Hill. In this part, I use the Pandas function import webbrowser and set set markers on the mark. Here is the code below:

```
import webbrowser
import folium
def marker(trajectory_df):
    trajectory = folium.map.FeatureGroup()
    for index, row in trajectory_df.iterrows():
        trajectory.add_child(
            folium.CircleMarker(
                [row['Y'], row['X']],
                radius=0.2,
    popup=index,
                color='blue'
            )
        )
    return trajectory
```

```
def plot_data(data):
    data['Y'] = data['Y'].astype(float)
    data['X'] = data['X'].astype(float)
    data = data[data['Y'] != 0]
    m = folium.Map(location=[data['Y'].mean(), data['X'].mean()], zoom_start=13)
    m.add_child(marker(data))
    m.save("Landsat 8 images.html")
    webbrowser.open("Landsat 8 images.html")
plot_data(data)
```

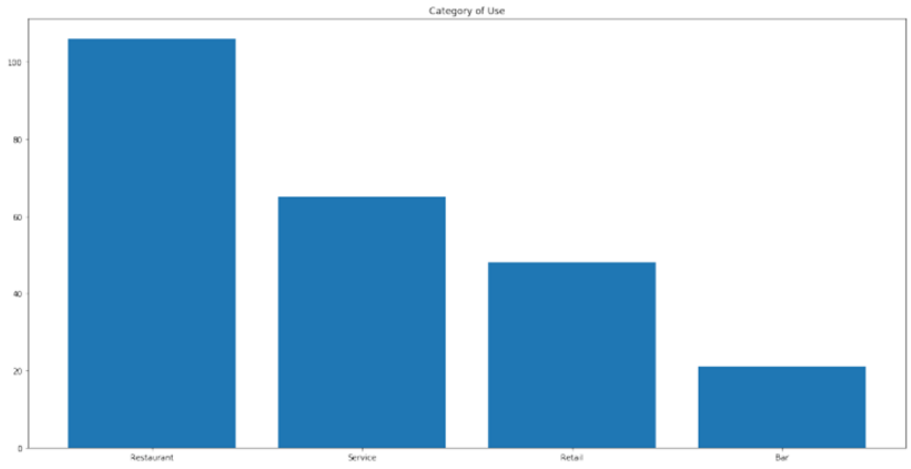
Here is the output:



4.2. Catogries of Use

Next, I used ’plot’ to make a comparison of catogries of tenants in Chapel Hill. We can see that Chapel Hill has the most restaurants of tenants, followed by service, and finally bar.

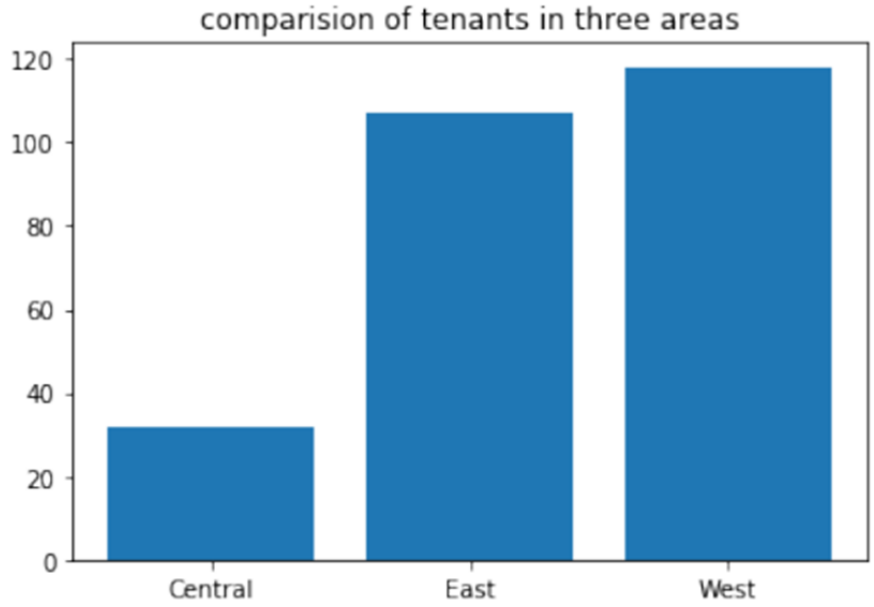
```
plt.figure(figsize=(20, 10))
num=data['Category_of_Use'].value_counts().tolist()
num2=num[0:4]
num2[0]=num[0]+num[8]
name=['Restaurant','Service','Retail','Bar']
plt.bar(name,num2)
plt.title("Category of Use")
plt.show()
```



4.3. Tenants Comparison

This is to compare numbers of tenants in each of the three districts. There are the most tenants in the eastern district, and the least in the central district.

```
# print(data.groupby("Downtown_District")["Category_of_Use"].count())
temp = data.groupby("Downtown_District")["Category_of_Use"].count().tolist()
columns = ["Central","East",'West']
plt.bar(columns,temp)
plt.title("comparision of tenants in three areas")
plt.show()
```



4.4. Tenant Categories Comparison

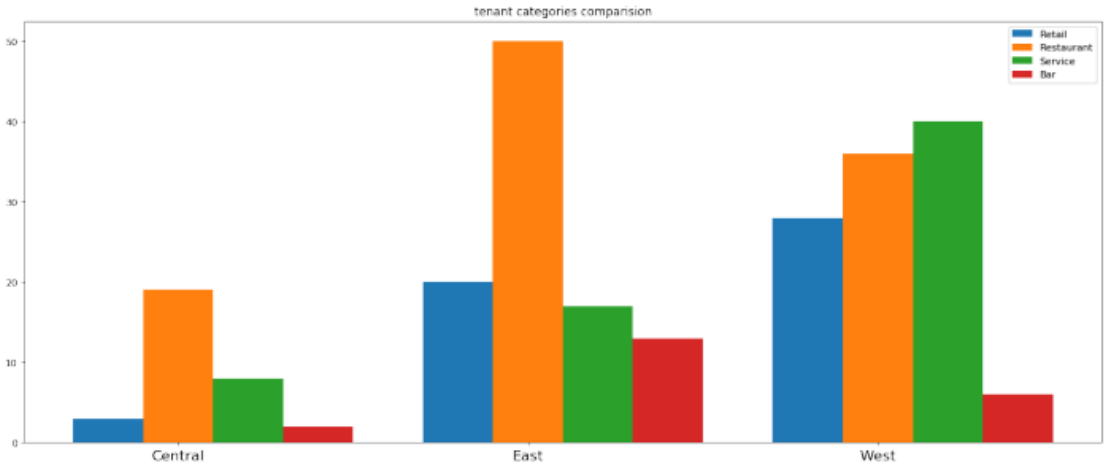
Next, I want to specifically compare the number of tenants in each of the four categories.

Blue represents Retail, yellow represents Restaurant, green represents Service, and red represents Bar. I use the function of plt.figure to set the size and sharpness.

Here is the code for this part below:

```
# retail restaurant service
Central_retail=data[data['Downtown_District'] ==␣
 ↪'Central'][data['Category_of_Use'] == 'Retail'].
 ↪shape[0]+data[data['Downtown_District'] ==␣
 ↪'Central'][data['Category_of_Use'] == 'Grocery'].shape[0]
Central_restaurant=data[data['Downtown_District'] ==␣
 ↪'Central'][data['Category_of_Use'] == 'Restaurant'].shape[0]
Central_service=data[data['Downtown_District'] ==␣
 ↪'Central'][data['Category_of_Use'] == 'Service'].shape[0]
Central_bar=data[data['Downtown_District'] ==␣
 ↪'Central'][data['Category_of_Use'] == 'Bar'].shape[0]

East_retail=data[data['Downtown_District'] == 'East'][data['Category_of_Use']␣
 ↪== 'Retail'].shape[0]+data[data['Downtown_District'] ==␣
 ↪'East'][data['Category_of_Use'] == 'Grocery'].shape[0]
East_restaurant=data[data['Downtown_District'] ==␣
 ↪'East'][data['Category_of_Use'] == 'Restaurant'].shape[0]
```

```
East_service=data[data['Downtown_District'] == 'East'][data['Category_of_Use']␣
 ↪== 'Service'].shape[0]
East_bar=data[data['Downtown_District'] == 'East'][data['Category_of_Use'] ==␣
 ↪'Bar'].shape[0]

West_retail=data[data['Downtown_District'] == 'West'][data['Category_of_Use']␣
 ↪== 'Retail'].shape[0]+data[data['Downtown_District'] ==␣
 ↪'West'][data['Category_of_Use'] == 'Grocery'].shape[0]
West_restaurant=data[data['Downtown_District'] ==␣
 ↪'West'][data['Category_of_Use'] == 'Restaurant'].shape[0]
West_service=data[data['Downtown_District'] == 'West'][data['Category_of_Use']␣
 ↪== 'Service'].shape[0]
West_bar=data[data['Downtown_District'] == 'West'][data['Category_of_Use'] ==␣
 ↪'Bar'].shape[0]


retail=[Central_retail,East_retail,West_retail]

restaurant=[Central_restaurant,East_restaurant,West_restaurant]

service=[Central_service,East_service,West_service]

bar=[Central_bar,East_bar,West_bar]

x_label1=['Central','East','West']

bar_width = 0.2
bar_1 = list(range(len(x_label1)))
bar_2 = [i+bar_width for i in bar_1]
bar_3 = [i+bar_width for i in bar_2]
bar_4 = [i+bar_width for i in bar_3]
#Set the image size and sharpness
plt.figure(figsize=(20, 8), dpi=80)
#Import data and draw bar charts
plt.bar(range(len(x_label1)), retail, width=bar_width, label='Retail')
plt.bar(bar_2,restaurant, width=bar_width, label='Restaurant')
plt.bar(bar_3, service, width=bar_width, label='Service')
plt.bar(bar_4, bar, width=bar_width, label='Bar')
plt.xticks(bar_2, x_label1,size=15)
plt.title("tenant categories comparision")
plt.legend()
plt.show()
x_label1=['Central','East','West']
```



4.5. Trend of Growth

I decided to show the growth of each of the tenants' categories in the three districts since the first tenants were born. The growth rate of tenants in the three regions can reflect the growth trend and can be used as a reference for residents.

This step took a lot of work,I set the X-axis every ten years as a lable.(I only screenshotted parts of code which can stand for the whole workload. I use the district of center and the categories of restaurant as examples)

```
data=data.dropna(subset=['Date_Opened'])#Delete empty data
data['Date_Opened'] = pd.to_datetime(data['Date_Opened'])
data=data.sort_values(by="Date_Opened", ascending=True)#In order of date from␣
 ↪smallest to largest

data_1922=data[(data["Date_Opened"] >='1922-01-01') & (data["Date_Opened"]<␣
 ↪'1932-01-01')]
data_1932=data[(data["Date_Opened"] >='1922-01-01') & (data["Date_Opened"]<␣
 ↪'1942-01-01')]
data_1942=data[(data["Date_Opened"] >='1922-01-01') & (data["Date_Opened"]<␣
 ↪'1952-01-01')]
data_1952=data[(data["Date_Opened"] >='1922-01-01') & (data["Date_Opened"]<␣
 ↪'1962-01-01')]
data_1962=data[(data["Date_Opened"] >='1922-01-01') & (data["Date_Opened"]<␣
 ↪'1972-01-01')]
data_1972=data[(data["Date_Opened"] >='1922-01-01') & (data["Date_Opened"]<␣
 ↪'1982-01-01')]
data_1982=data[(data["Date_Opened"] >='1922-01-01') & (data["Date_Opened"]<␣
 ↪'1992-01-01')]
data_1992=data[(data["Date_Opened"] >='1922-01-01') & (data["Date_Opened"]<␣
 ↪'2002-01-01')]
data_2002=data[(data["Date_Opened"] >='1922-01-01') & (data["Date_Opened"]<␣
 ↪'2012-01-01')]
data_2012=data[(data["Date_Opened"] >='1922-01-01')]
```

```
raise_Central_restaurant=[]
raise_Central_restaurant.append(data_1922[data_1922['Downtown_District'] ==␣
 ↪'Central'][data_1922['Category_of_Use'] == 'Restaurant'].shape[0])
raise_Central_restaurant.append(data_1932[data_1932['Downtown_District'] ==␣
 ↪'Central'][data_1932['Category_of_Use'] == 'Restaurant'].shape[0])
raise_Central_restaurant.append(data_1942[data_1942['Downtown_District'] ==␣
 ↪'Central'][data_1942['Category_of_Use'] == 'Restaurant'].shape[0])
raise_Central_restaurant.append(data_1952[data_1952['Downtown_District'] ==␣
 ↪'Central'][data_1952['Category_of_Use'] == 'Restaurant'].shape[0])
raise_Central_restaurant.append(data_1962[data_1962['Downtown_District'] ==␣
 ↪'Central'][data_1962['Category_of_Use'] == 'Restaurant'].shape[0])
raise_Central_restaurant.append(data_1972[data_1972['Downtown_District'] ==␣
 ↪'Central'][data_1972['Category_of_Use'] == 'Restaurant'].shape[0])
raise_Central_restaurant.append(data_1982[data_1982['Downtown_District'] ==␣
 ↪'Central'][data_1982['Category_of_Use'] == 'Restaurant'].shape[0])
raise_Central_restaurant.append(data_1992[data_1992['Downtown_District'] ==␣
 ↪'Central'][data_1992['Category_of_Use'] == 'Restaurant'].shape[0])
raise_Central_restaurant.append(data_2002[data_2002['Downtown_District'] ==␣
 ↪'Central'][data_2002['Category_of_Use'] == 'Restaurant'].shape[0])
raise_Central_restaurant.append(data_2012[data_2012['Downtown_District'] ==␣
 ↪'Central'][data_2012['Category_of_Use'] == 'Restaurant'].shape[0])
```

```
raise_xlabel=['1922','1932','1942','1952','1962','1972','1982','1992','2002','2012']
#restaurant
plt.plot(raise_xlabel,raise_Central_restaurant,label="Central_restaurant")
plt.plot(raise_xlabel,raise_East_restaurant,label="East_restaurant")
plt.plot(raise_xlabel,raise_West_restaurant,label="West_restaurant")
# plt.xticks(raise_xlabel)
from matplotlib.pyplot import MultipleLocator
ax=plt.gca()#
y_major_locator=MultipleLocator(5)#Set the y-scale interval to 10
ax.yaxis.set_major_locator(y_major_locator)#Set the primary scale on the Y-axis␣
 ↪to a multiple of 10
```

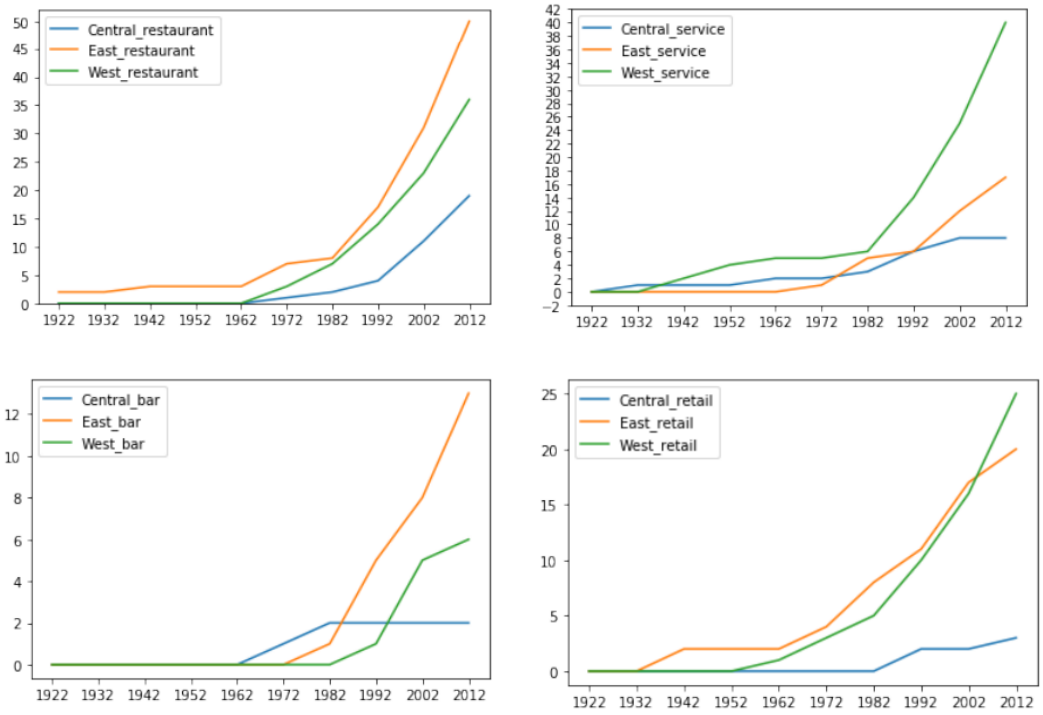
```
plt.ylim(0,52)
plt.legend()
plt.show()

#Service
plt.plot(raise_xlabel,raise_Central_service,label="Central_service")
plt.plot(raise_xlabel,raise_East_service,label="East_service")
plt.plot(raise_xlabel,raise_West_service,label="West_service")
# plt.xticks(raise_xlabel)
from matplotlib.pyplot import MultipleLocator
ax2=plt.gca()
y_major_locator=MultipleLocator(2)
ax2.yaxis.set_major_locator(y_major_locator)
plt.legend()
plt.show()


#bar
plt.plot(raise_xlabel,raise_Central_bar,label="Central_bar")
plt.plot(raise_xlabel,raise_East_bar,label="East_bar")
plt.plot(raise_xlabel,raise_West_bar,label="West_bar")
# plt.xticks(raise_xlabel)
from matplotlib.pyplot import MultipleLocator
ax3=plt.gca()
y_major_locator=MultipleLocator(2)
ax3.yaxis.set_major_locator(y_major_locator)
plt.legend()
plt.show()


#Retail
plt.plot(raise_xlabel,raise_Central_retail,label="Central_retail")
plt.plot(raise_xlabel,raise_East_retail,label="East_retail")
plt.plot(raise_xlabel,raise_West_retail,label="West_retail")
# plt.xticks(raise_xlabel)
from matplotlib.pyplot import MultipleLocator
ax4=plt.gca()
y_major_locator=MultipleLocator(5)
ax4.yaxis.set_major_locator(y_major_locator)
plt.legend()
plt.show()
```

Here is the output. According to the output, we can see that The number of restaurants in the three districts has being increased steadily since 1982.

For bars, the growth rate in the east has been great since 1982, the east has grown slowly since 2002, and the central has stopped growing since 1982.

For retail, the west maintains a steady growth trend, the east also has a growth trend, and the central one is still growing slowly.

Regarding service, the western of chapel hill has the largest growth rate, the eastern chapelhill has been growing slowly since 2002, and the central one has still not seen growth since 1982.



4.6. Heatmap

I use the function of heatmap to plot the heatmap of three districts with each category. In this way, we can intuitively notice the distribution of each type of tenants in Chapel Hill.

```
import folium
import webbrowser#
from folium.plugins import HeatMap

heatmap_data = [] # create an empty list
data_bar=data[data['Category_of_Use'] == 'Bar']
data_restaurant=data[data['Category_of_Use'] == 'Restaurant']
data_service=data[data['Category_of_Use'] == 'Service']
data_retail=data[data['Category_of_Use'] == 'Retail']


# print(data_Central['Downtown_District'])
for index, fire in data_retail.iterrows():
    heatmap_data.append([fire['Y'], fire['X']])

# Now we create a new map
m2 = folium.Map(location=[35.9132, -79.0558], zoom_start=13)

# Now, a heatmap
fire_heatmap = HeatMap(heatmap_data)
# We add the heat map to the map m2
fire_heatmap.add_to(m2)

m2
```

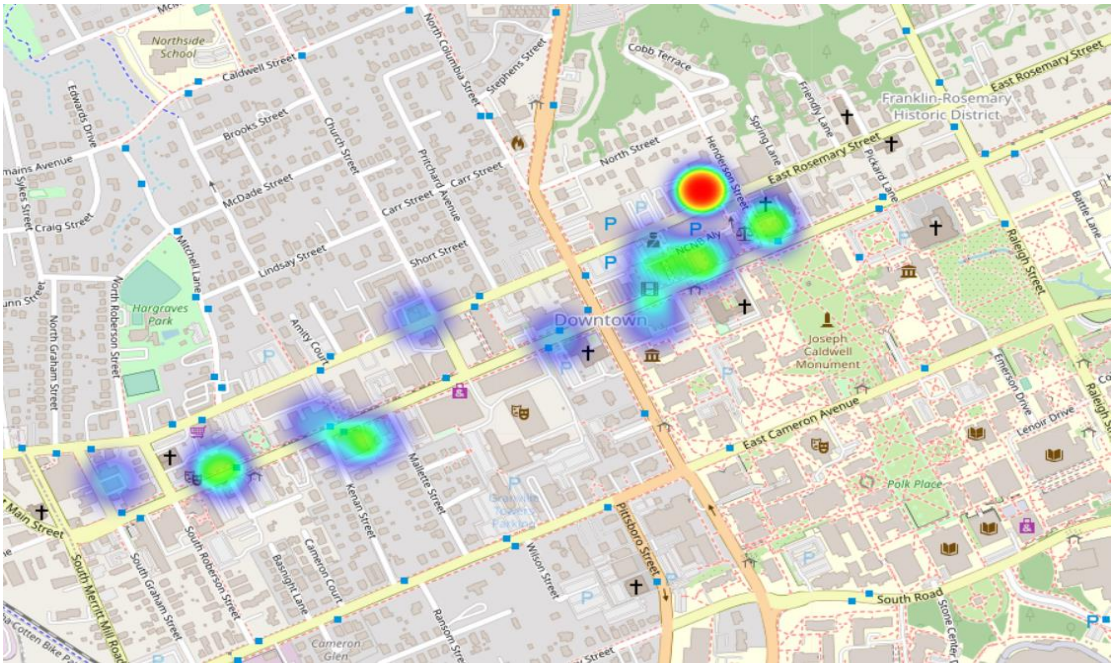
Here is the output of four categories.

restaurant:

service:

Retail:

Bar:


Northside School
Caldwell Street
North Columbia Street
Stephens Street
Cobb Terrace
Franklin-Rosemary Historic District
East Rosemary Street
Edwards Drive
Brooks Street
Church Street
Pritchard Avenue
Carr Street
McDade Street
North Street
Henderson Street
Spring Lane
Friendly Lane
Pickard Lane
Craig Street
Mitchell Lane
Lindsay Street
Short Street
Hargraves Park
Downtown
Joseph Caldwell Monument
Raleigh Street
Amity Court
North Roberson Street
North Graham Street
East Cameron Avenue
Emerson Drive
Lenoir Drive
Polk Place
Malaeite Street
Kenan Street
Pittsboro Street
Wilson Street
Cameron Court
Bascomlight Lane
South Roberson Street
South Graham Street
South Main Street
South Road
Cameron Glen

5. Conclusion

From the data analysis point of view, I can get the following summary. The first is the analysis of the three districts. The three districts have their own advantages and characteristics. The West District has the largest number of retail and service tenants, and its growth rate. The number of restaurants is also sufficient. Compared to the east, there are fewer bars. The number of restaurants in the east is the largest, and it can be seen from the distribution map that most of the restaurants are clustered together to form a dining block. Residents can easily choose food and coffee in this place. There are also the most bars in the east. So the biggest advantage of the east is entertainment and dining. The central area has the least number of tenants of all types, but don't forget that it is located between east and west, which means it's easy to live here and go to the east or west. For those in Chapel Hill who want to refer to this information, I also have the following suggestions. If you are a foodie or a person who likes to drink and entertain with friends, you must choose the East District, which can satisfy your entertainment hobby. If you like shopping, go to various stores to buy treasures, and buy things, you can choose West or Central. If you want to experience convenient services, such as haircuts, bank services, convenient medical treatment, etc., you should choose the western part. If there are no special needs or hobbies, the West District is the best, because there are a wide variety of tenants and there are no defects.

6. Impact and limitations:

As stated in this paper, the purpose of the project was to analyze the distribution of tenants in Chapel Hill and draw conclusions that could help people live better in Chapel Hill. Based on the overall purpose of the study, the impact of the project, there are a number of potential impacts of this project for both the locals as well as the immigrants. The first implication is on the ease of accessing a wide range of information not only on the different tenants, but also on the part of the city one can access the information, for instance, in the West or central district, individuals can have access to stores where they can buy things. The Western part is more commonly known for its bank services, salons and barber shops, and medical treatment. The eastern part is mostly for entertainment in terms of both food and drink.

However, the project is limited to Chapel City. Therefore, only those who are living in or are planning to relocate to or visit Chapel Hill can benefit from the findings of this study. Moreover, the study is limited in limited focus on the names and physical addresses of some of the stores, restaurants, banks and even hospitals located in the different areas of the city. The study only lumps together the different tenancies and maps out where the majority of the services may be obtained. Further, the project does not make any recommendations to individuals on which specific shop they should visit.

Reference

Birmingham, England, (2018). Data Visualization in Python by Examples. PACKT Publishing

Khan, Saba Hafeez., (2020). A Parallel Implementation of the Pandas Framework

Content Based Filtering and Rating Variance. Procedia Computer Science. 132 (2018) 1678-1684.

Sun, Ting; Grimmond, Sue, (2019). A Python-enhanced urban land surface model SuPy

Banerjee, Dibyendu., (2020). Science Concierge: Focus: Python Libraries for Data Science Simplified

Kumar, Gaurav., Focus: Using Python DataFrames for Advanced Database Applications

Nishio, Kei. Joho no Kagaku to Gijutsu., (2020). Patent data processing using Python (pandas)

Alexander, P. J., Bechtel, B., Chow, W. T. L., Fealy, R., and Mills, G.: Linking urban climate classification with an urban energy and water budget model: Multi-site and multi-seasonal evaluation, Urban Climate, 17, 196–215, https://doi.org/10.1016/j.uclim.2016.08.003, 2016.