Published in Analytics Vidhya · Follow

Bhargava Sai Reddy P · Follow
Oct 21, 2019 · 6 min read · Listen

Web Scraping in Python for Data Extraction & Analysis

A web scraper for Analysis of Residential properties in Hyderabad.

In 2 years my father is going to get retired from his job and hence he wanted to buy a residential property so that he could settle in Hyderabad after his retirement. Getting to know about this I started thinking about how I could help him and then decided to use this situation as an opportunity to improve my Python skills as well. So I needed two things for this project :

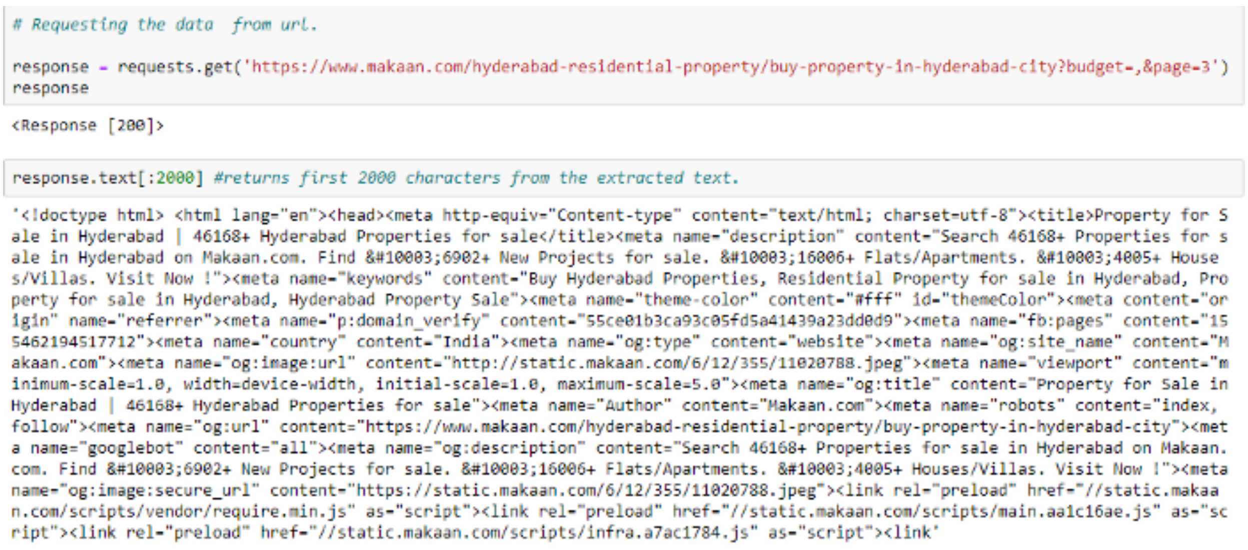
1. Scrape and store data regarding residential properties in Hyderabad from one of the best real estate websites.

2. Perform EDA(Exploratory data analysis) on extracted data and get the required information and insights of data.

The website used to scarp is makaan which is one of the best real estate websites. So this makaan website has all the details about different types of properties like Apartments, Residential plots, Villas, Individual houses, etc. We are going to extract data of all properties and then get the required information from data.

We start by importing the necessary libraries.

```
# Importing necessary libraries.

from bs4 import BeautifulSoup as soup
from requests import get
import requests
```



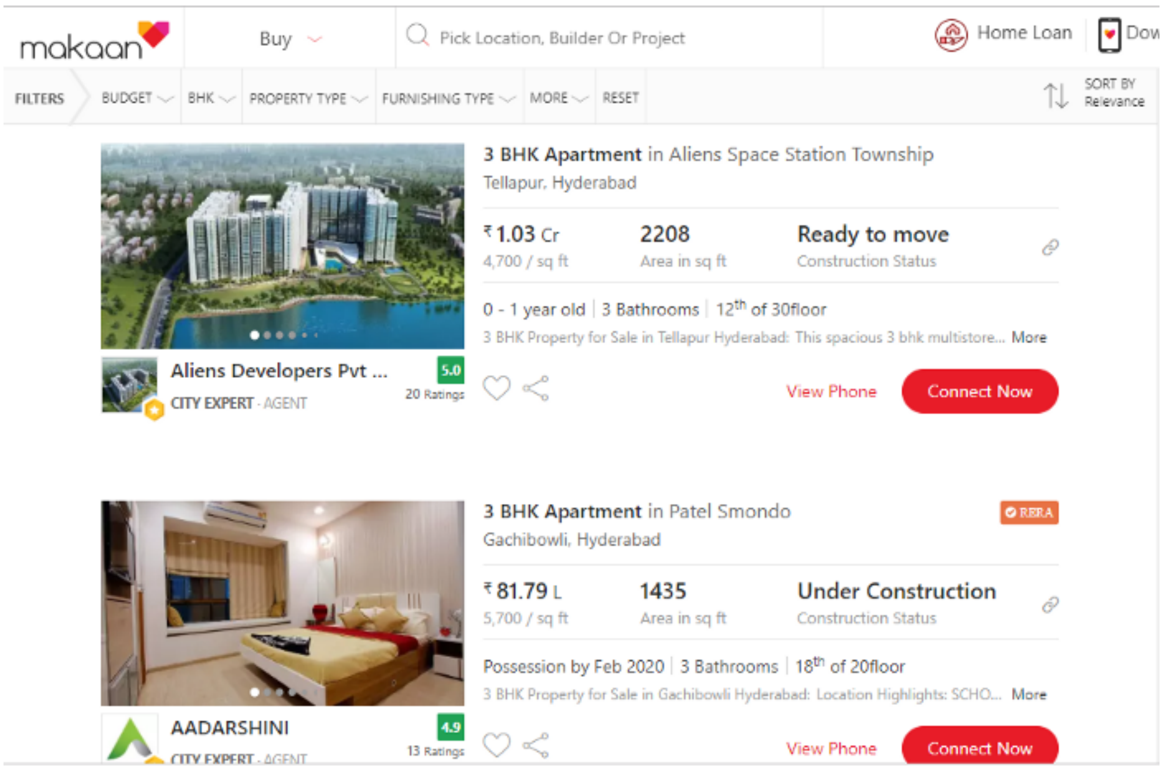
```
# Requesting the data  from url.

response = requests.get('https://www.makaan.com/hyderabad-residential-property/buy-property-in-hyderabad-city?budget=,&page=3')
response
```

```
<Response [200]>
```

```
response.text[:2000] #returns first 2000 characters from the extracted text.
```

```
'<!doctype html> <html lang="en"><head><meta http-equiv="Content-type" content="text/html; charset=utf-8"><title>Property for S
ale in Hyderabad | 46168+ Hyderabad Properties for sale</title><meta name="description" content="Search 46168+ Properties for s
ale in Hyderabad on Makaan.com. Find &#10003;6902+ New Projects for sale. &#10003;16006+ Flats/Apartments. &#10003;4005+ House
s/Villas. Visit Now !"><meta name="keywords" content="Buy Hyderabad Properties, Residential Property for sale in Hyderabad, Pro
perty for sale in Hyderabad, Hyderabad Property Sale"><meta name="theme-color" content="#fff" id="themeColor"><meta content="or
igin" name="referrer"><meta name="p:domain_verify" content="55ce01b3ca93c05fd5a41439a23dd0d9"><meta name="fb:pages" content="15
5462194517712"><meta name="country" content="India"><meta name="og:type" content="website"><meta name="og:site_name" content="M
akaan.com"><meta name="og:image:url" content="http://static.makaan.com/6/12/355/11020788.jpeg"><meta name="viewport" content="m
inimum-scale=1.0, width=device-width, initial-scale=1.0, maximum-scale=5.0"><meta name="og:title" content="Property for Sale in
Hyderabad | 46168+ Hyderabad Properties for sale"><meta name="Author" content="Makaan.com"><meta name="robots" content="index,
follow"><meta name="og:url" content="https://www.makaan.com/hyderabad-residential-property/buy-property-in-hyderabad-city"><met
a name="googlebot" content="all"><meta name="og:description" content="Search 46168+ Properties for sale in Hyderabad on Makaan.
com. Find &#10003;6902+ New Projects for sale. &#10003;16006+ Flats/Apartments. &#10003;4005+ Houses/Villas. Visit Now !"><meta
name="og:image:secure_url" content="https://static.makaan.com/6/12/355/11020788.jpeg"><link rel="preload" href="//static.makaa
n.com/scripts/vendor/require.min.js" as="script"><link rel="preload" href="//static.makaan.com/scripts/main.aa1c16ae.js" as="sc
ript"><link rel="preload" href="//static.makaan.com/scripts/infra.a7ac1784.js" as="script"><link'
```

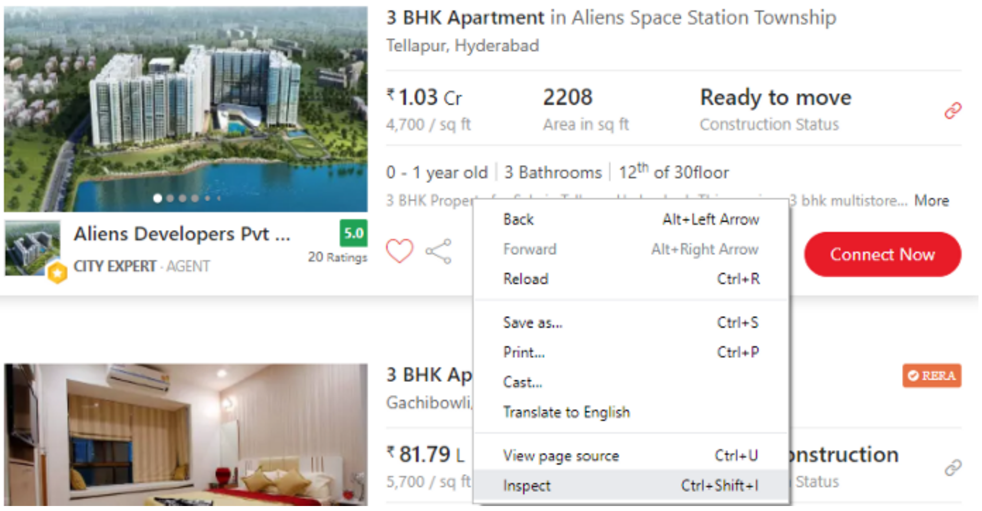
The HTTP request returns a Response Object with all the response data (content, status, etc).



3 BHK Apartment in Aliens Space Station Township
Tellapur, Hyderabad
₹ 1.03 Cr 4,700 / sq ft | 2208 Area in sq ft | Ready to move Construction Status
0 - 1 year old | 3 Bathrooms | 12th of 30floor
3 BHK Property for Sale in Tellapur Hyderabad: This spacious 3 bhk multistore... More
Aliens Developers Pvt ... CITY EXPERT - AGENT 5.0 20 Ratings

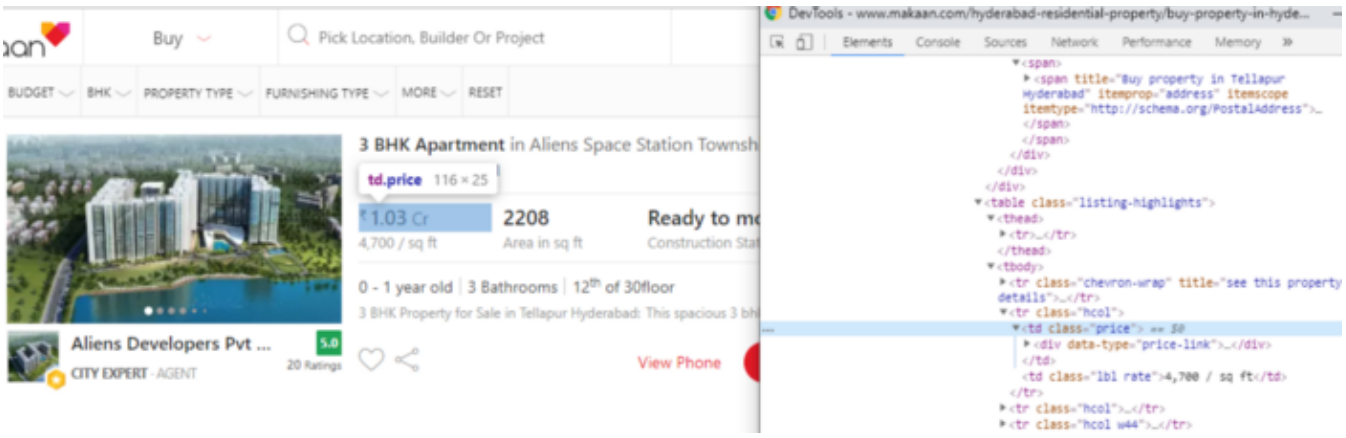
3 BHK Apartment in Patel Smondo
Gachibowli, Hyderabad
₹ 81.79 L 5,700 / sq ft | 1435 Area in sq ft | Under Construction Construction Status
Possession by Feb 2020 | 3 Bathrooms | 18th of 20floor
3 BHK Property for Sale in Gachibowli Hyderabad: Location Highlights: SCHO... More
AADARSHINI CITY EXPERT - AGENT 4.9 13 Ratings

The above is our website from which we are going to extract the data. From the data provided on the website, we select the required attributes and we further extract data of those attributes. And now to extract data right-click on any property and select inspect.



After clicking on inspecting there we can see the source code which is in Html format. Now select the class from which u want to extract the data about a property.

Let's say I want to extract the price of the property now I'm going to select the class that contains the price of the property.


Buy
Pick Location, Builder Or Project
BUDGET
BHK
PROPERTY TYPE
FURNISHING TYPE
MORE
RESET
3 BHK Apartment in Aliens Space Station Townsh
td.price 116 × 25
₹1.03 Cr
4,700 / sq ft
2208
Area in sq ft
Ready to m
Construction Stat
0 - 1 year old | 3 Bathrooms | 12th of 30floor
3 BHK Property for Sale in Tellapur Hyderabad: This spacious 3 bh
Aliens Developers Pvt ...
5.0
20 Ratings
CITY EXPERT · AGENT
View Phone
DevTools - www.makaan.com/hyderabad-residential-property/buy-property-in-hyde...
Elements
Console
Sources
Network
Performance
Memory

```
▼<span>
  ▶<span title="Buy property in Tellapur
  Hyderabad" itemprop="address" itemscope
  itemtype="http://schema.org/PostalAddress">…
  </span>
  </span>
 </div>
</div>
</div>
▼<table class="listing-highlights">
 ▼<thead>
  ▶<tr>…</tr>
  </thead>
 ▼<tbody>
  ▶<tr class="chevron-wrap" title="see this property
  details">…</tr>
  ▼<tr class="hcol">
   ▼<td class="price"> == $0
     ▶<div data-type="price-link">…</div>
    </td>
    <td class="lbl rate">4,700 / sq ft</td>
   </tr>
  ▶<tr class="hcol">…</tr>
  ▶<tr class="hcol w44">…</tr>
```

Now using the class name we can extract the data as below, where the "cardholder" is the class that contains overall data of a property.

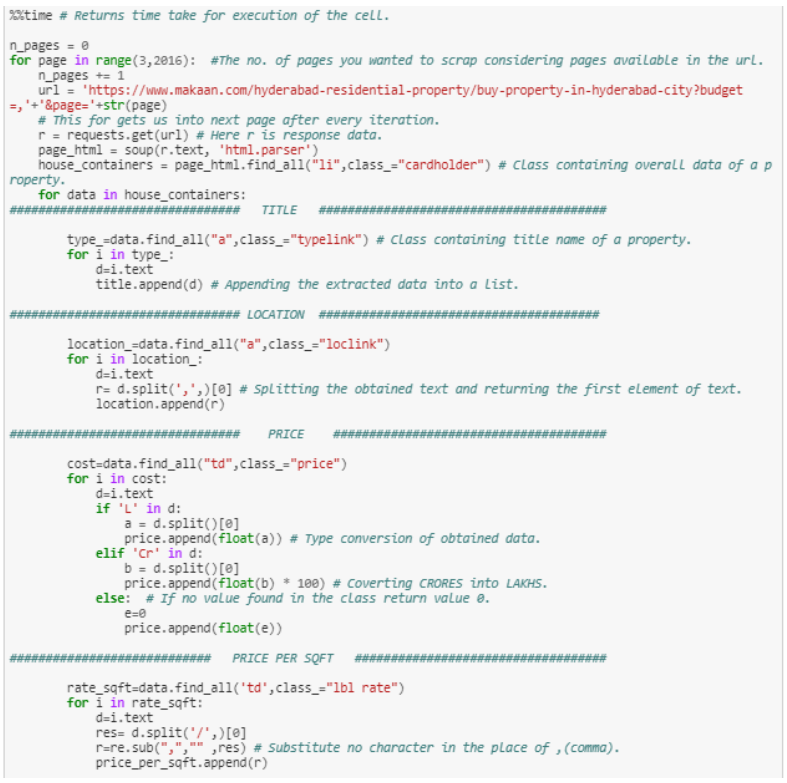
```
x=first.find_all("td",class_="price")[0].text
x
```

```
' 3.2 Cr'
```

Consider a list to save the extracted data. We can create n lists for storing extracted data of n attributes.

```
# Creating empty lists in-order to append data scrapped from url's.
# The number of lists depends on the number of features you are extracting from the url.

title=[]
location=[]
price=[]
price_per_sqft=[]
area_in_sqft=[]
building_status=[]
```

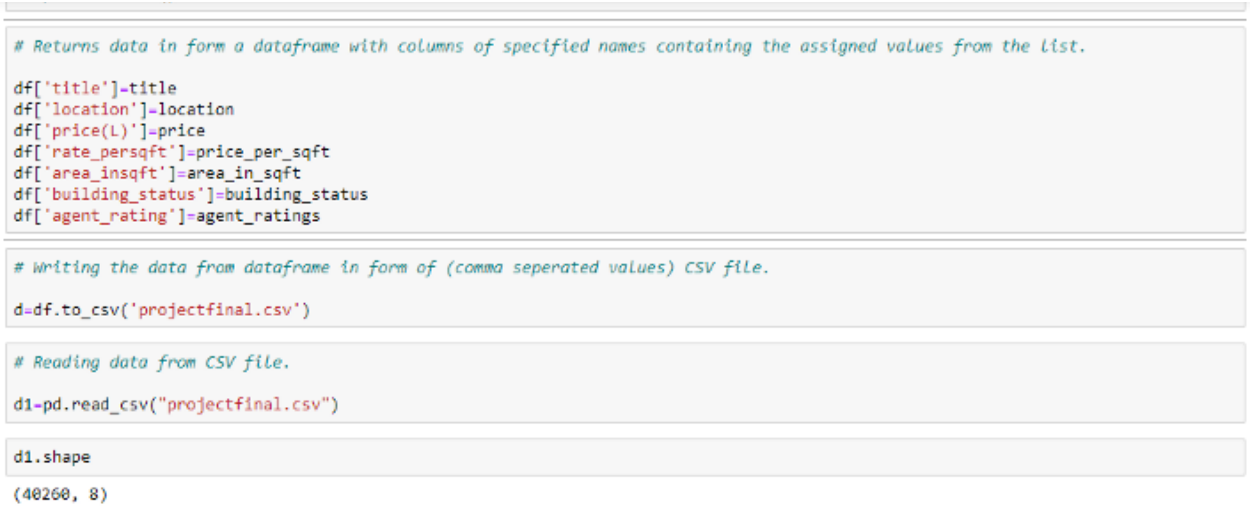
To extract data of each attribute from each property at once, it takes an ample amount of time so we will consider a for-loop for n iterations where n is the number of pages we wish to scrap.



```
%%time # Returns time take for execution of the cell.

n_pages = 0
for page in range(3,2016):  #The no. of pages you wanted to scrap considering pages available in the url.
    n_pages += 1
    url = 'https://www.makaan.com/hyderabad-residential-property/buy-property-in-hyderabad-city?budget
=,'+'&page='+str(page)
    # This for gets us into next page after every iteration.
    r = requests.get(url) # Here r is response data.
    page_html = soup(r.text, 'html.parser')
    house_containers = page_html.find_all("li",class_="cardholder") # Class containing overall data of a p
roperty.
    for data in house_containers:
################################   TITLE   ########################################

        type_=data.find_all("a",class_="typelink") # Class containing title name of a property.
        for i in type_:
            d=i.text
            title.append(d) # Appending the extracted data into a list.

################################ LOCATION  #######################################

        location_=data.find_all("a",class_="loclink")
        for i in location_:
            d=i.text
            r= d.split(',',)[0] # Splitting the obtained text and returning the first element of text.
            location.append(r)

################################    PRICE    #####################################

        cost=data.find_all("td",class_="price")
        for i in cost:
            d=i.text
            if 'L' in d:
                a = d.split()[0]
                price.append(float(a)) # Type conversion of obtained data.
            elif 'Cr' in d:
                b = d.split()[0]
                price.append(float(b) * 100) # Coverting CRORES into LAKHS.
            else:  # If no value found in the class return value 0.
                e=0
                price.append(float(e))

############################   PRICE PER SQFT   ##################################

        rate_sqft=data.find_all('td',class_="lbl rate")
        for i in rate_sqft:
            d=i.text
            res= d.split('/',)[0]
            r=re.sub(",","" ,res) # Substitute no character in the place of ,(comma).
            price_per_sqft.append(r)
```

For complete code click on Github

The extracted data that we stored into the lists are now stored in the data frame as below.



```
# Returns data in form a dataframe with columns of specified names containing the assigned values from the list.

df['title']=title
df['location']=location
df['price(L)']=price
df['rate_persqft']=price_per_sqft
df['area_insqft']=area_in_sqft
df['building_status']=building_status
df['agent_rating']=agent_ratings
```

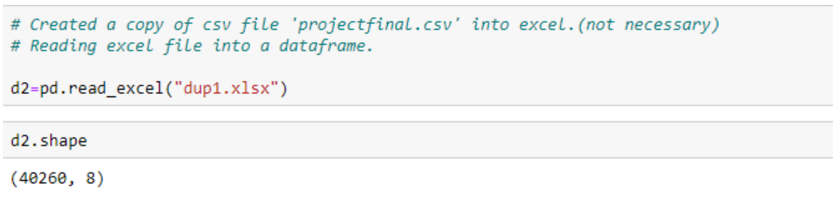
```
# Writing the data from dataframe in form of (comma seperated values) CSV file.

d=df.to_csv('projectfinal.csv')
```

```
# Reading data from CSV file.

d1=pd.read_csv("projectfinal.csv")
```

```
d1.shape
```

```
(40260, 8)
```

From above we can say that we have extracted 40260 rows i.e, 40260 with 8 features/attributes.



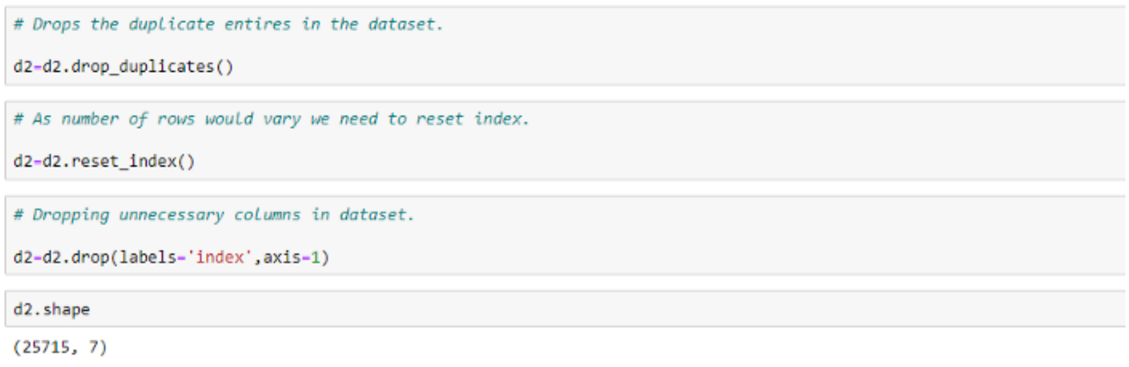
```
# Created a copy of csv file 'projectfinal.csv' into excel.(not necessary)
# Reading excel file into a dataframe.

d2=pd.read_excel("dup1.xlsx")
```

```
d2.shape
```

```
(40260, 8)
```

Time to clean the data and check for missing values

Check for duplicates and drop if any duplicates are present.



```
# Drops the duplicate entires in the dataset.

d2=d2.drop_duplicates()
```

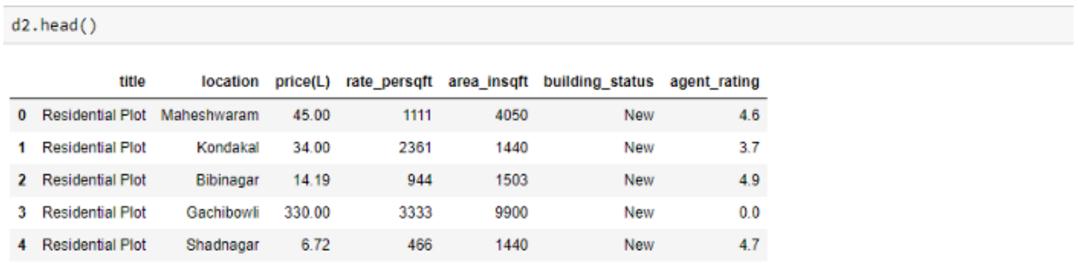
```
# As number of rows would vary we need to reset index.

d2=d2.reset_index()
```

```
# Dropping unnecessary columns in dataset.

d2=d2.drop(labels='index',axis=1)
```

```
d2.shape
```

```
(25715, 7)
```

After dropping duplicate rows we are left with 25715 data points that are almost half of the data points that are extracted are duplicates. Here the number of columns is reduced from 8 to 7 as we dropped the unnecessary column which has index values. Now if we take a look at our final data this is how it is as shown below.



```
d2.head()
```

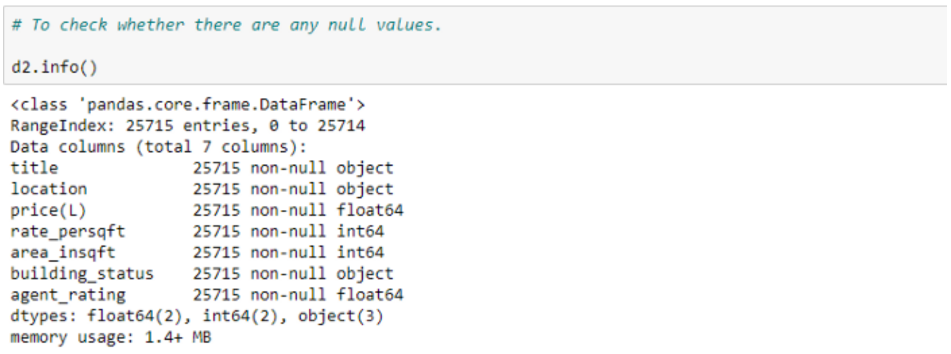
	title	location	price(L)	rate_persqft	area_insqft	building_status	agent_rating
0	Residential Plot	Maheshwaram	45.00	1111	4050	New	4.6
1	Residential Plot	Kondakal	34.00	2361	1440	New	3.7
2	Residential Plot	Bibinagar	14.19	944	1503	New	4.9
3	Residential Plot	Gachibowli	330.00	3333	9900	New	0.0
4	Residential Plot	Shadnagar	6.72	466	1440	New	4.7

Our features/attributes in the data set are the titles, location, price in lakhs, rate_persqft, area_insqft, building_status, agent_rating.

- **title:** Defines the title name of the property, whose data type is a string

- **rate_persqft:** Defines the rate per unit sqft of property, whose data type is int.
- **area_insqft:** Defines the total area of the property, whose data type is int.
- **building_status:** Define the current status of the building, whose data type is a string
- **agent_rating:** Defines the rating of the agent given by the customers, whose data type is afloat.

Check for any missing values,

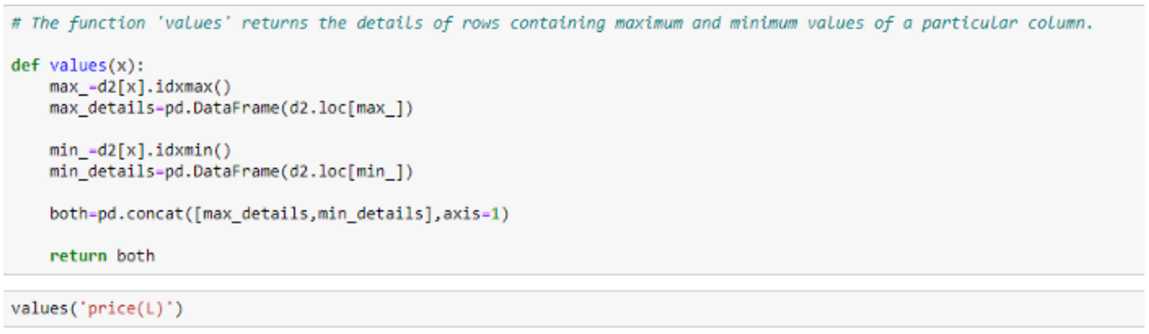


```
# To check whether there are any null values.

d2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 25715 entries, 0 to 25714
Data columns (total 7 columns):
title              25715 non-null object
location           25715 non-null object
price(L)           25715 non-null float64
rate_persqft       25715 non-null int64
area_insqft        25715 non-null int64
building_status    25715 non-null object
agent_rating       25715 non-null float64
dtypes: float64(2), int64(2), object(3)
memory usage: 1.4+ MB
```

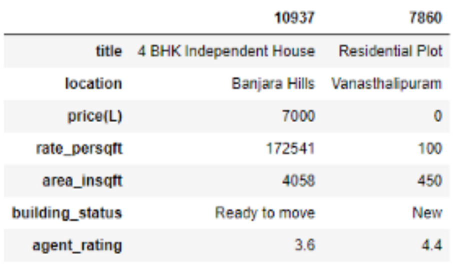
In our dataset, there are no missing values :) we can even check of missing values using IsNull() function.

Let's see what are the highest and lowest values in each column to get this data we can create a function that returns the maximum and minimum values of columns.



```
# The function 'values' returns the details of rows containing maximum and minimum values of a particular column.

def values(x):
    max_=d2[x].idxmax()
    max_details=pd.DataFrame(d2.loc[max_])

    min_=d2[x].idxmin()
    min_details=pd.DataFrame(d2.loc[min_])

    both=pd.concat([max_details,min_details],axis=1)

    return both
```

```
values('price(L)')
```

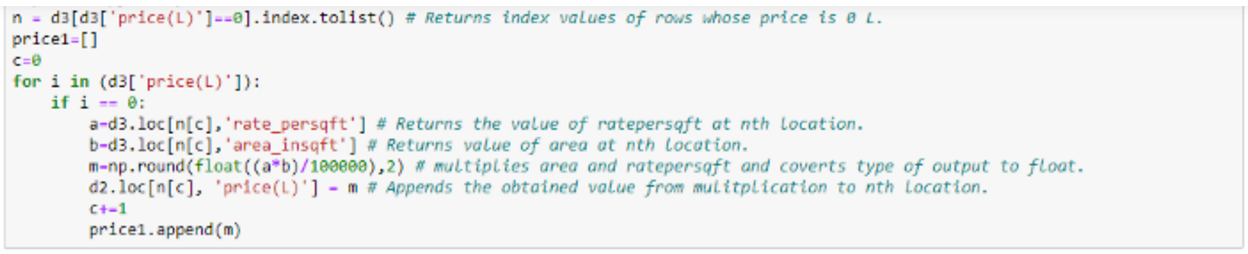


	10937	7860
title	4 BHK Independent House	Residential Plot
location	Banjara Hills	Vanasthalipuram
price(L)	7000	0
rate_persqft	172541	100
area_insqft	4058	450
building_status	Ready to move	New
agent_rating	3.6	4.4

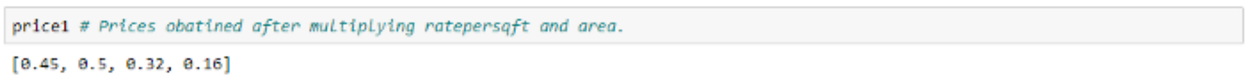
From above we can see that in the column price we have the highest price as 7000 lakhs and the lowest price as 0 lakhs. Seeing this we get a doubt that how can the price of a property be 0. If we go back to our extraction code there you can find that when the price of a property is missing we made it replace with 0. So we can conclude that 0 here means missing data.

We can treat the missing values in different ways using imputation, imputation means replacing the missing values with mean/median/mode or dropping the rows it depends on the data you are dealing with. With the kind of data, we are handling we can't replace the data using imputation as the price of properties would vary from one location to another in Hyderabad.

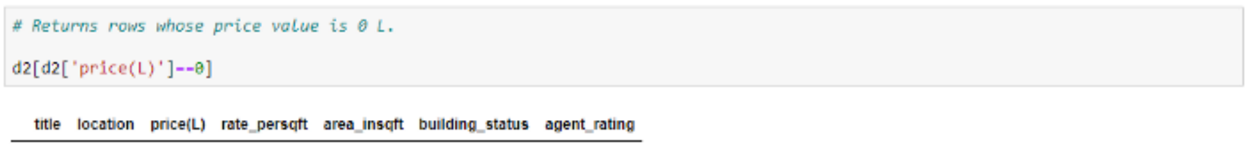
If we observe our data closely we can understand that we have total area of a property in sqft and rate per sqft of that property. A simple solution is that we can multiply area_in_sqft*rate_per_sqft which returns the prices of property.



```
n = d3[d3['price(L)']==0].index.tolist() # Returns index values of rows whose price is 0 L.
price1=[]
c=0
for i in (d3['price(L)']):
    if i == 0:
        a=d3.loc[n[c],'rate_persqft'] # Returns the value of ratepersqft at nth location.
        b=d3.loc[n[c],'area_insqft'] # Returns value of area at nth location.
        m=np.round(float((a*b)/100000),2) # multiplies area and ratepersqft and coverts type of output to float.
        d2.loc[n[c], 'price(L)'] = m # Appends the obtained value from mulitplication to nth location.
        c+=1
        price1.append(m)
```



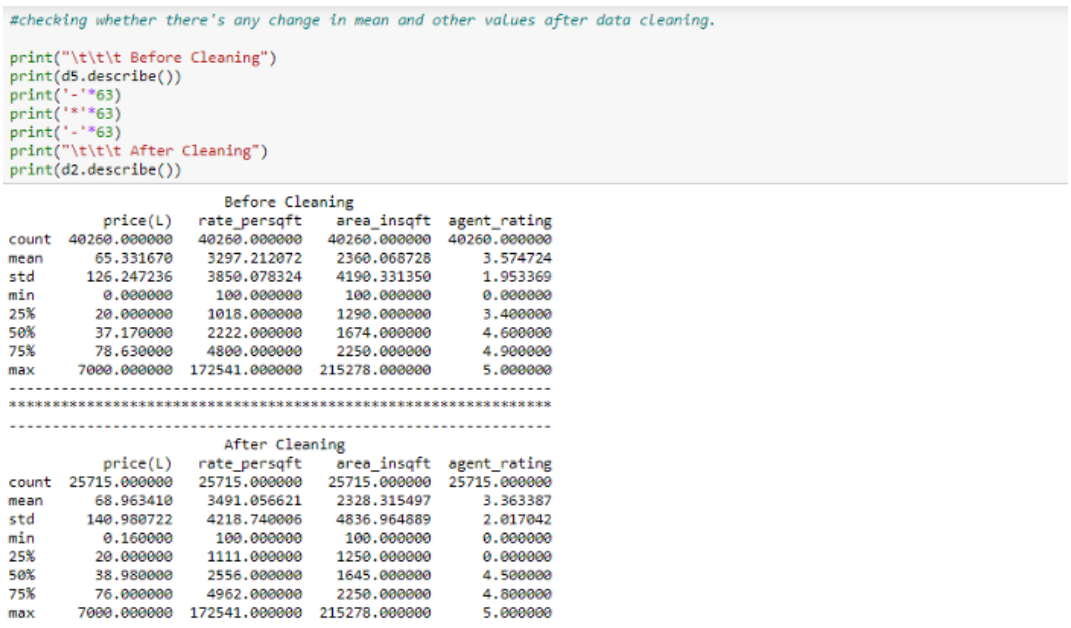
```
price1 # Prices obatined after multiplying ratepersqft and area.
[0.45, 0.5, 0.32, 0.16]
```



```
# Returns rows whose price value is 0 L.

d2[d2['price(L)']==0]
```

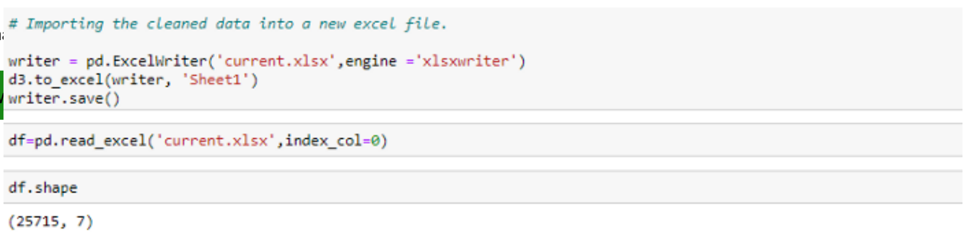
title	location	price(L)	rate_persqft	area_insqft	building_status	agent_rating

We can see that there are no rows left which have a price value of 0. We can see the difference between the data before cleaning and after cleaning there would variations in mean, median, std etc.



```
#checking whether there's any change in mean and other values after data cleaning.

print("\t\t\t Before Cleaning")
print(d5.describe())
print('-'*63)
print('*'*63)
print('-'*63)
print("\t\t\t After Cleaning")
print(d2.describe())
```

```
                        Before Cleaning
            price(L)   rate_persqft    area_insqft  agent_rating
count   40260.000000   40260.000000   40260.000000  40260.000000
mean       65.331670    3297.212072    2360.068728      3.574724
std       126.247236    3850.078324    4190.331350      1.953369
min         0.000000     100.000000     100.000000      0.000000
25%        20.000000    1018.000000    1290.000000      3.400000
50%        37.170000    2222.000000    1674.000000      4.600000
75%        78.630000    4800.000000    2250.000000      4.900000
max      7000.000000  172541.000000  215278.000000      5.000000
---------------------------------------------------------------
***************************************************************
---------------------------------------------------------------
                        After Cleaning
            price(L)   rate_persqft    area_insqft  agent_rating
count   25715.000000   25715.000000   25715.000000  25715.000000
mean       68.963410    3491.056621    2328.315497      3.363387
std       140.980722    4218.740006    4836.964889      2.017042
min         0.160000     100.000000     100.000000      0.000000
25%        20.000000    1111.000000    1250.000000      0.000000
50%        38.980000    2556.000000    1645.000000      4.500000
75%        76.000000    4962.000000    2250.000000      4.800000
max      7000.000000  172541.000000  215278.000000      5.000000
```



```
# Importing the cleaned data into a new excel file.

writer = pd.ExcelWriter('current.xlsx',engine ='xlsxwriter')
d3.to_excel(writer, 'Sheet1')
writer.save()
```

```
df=pd.read_excel('current.xlsx',index_col=0)
```

```
df.shape
(25715, 7)
```

We can see that our final data consists of 25715 data points i.e, data about 25715 properties with 7 features/attributes.

Note:

There is a follow up on this article about EDA on this data that we have extracted. You can find it in my stories or click here.

For the complete code of this project, you can visit my Github. Click here or below

Bhargava-Sai-P/Residential-Property-Analysis

You can’t perform that action at this time. You signed in with another tab or window. You signed out in another tab or...

github.com

Thanks for reading.