Filtering unavailable rooms

```
#filtering unavailable rooms
def findidbed(l_s):
  tidx_l=l_s.find('ListingCard_badgeTooltip__INLZY')+33
  tidx_u = l_s[tidx_l:].find('<')
  #print (l_s[tidx_l:tidx_u+tidx_l-1 ])
  return l_s[tidx_l:tidx_u+tidx_l-1 ]
```

Code to scrap data from website

```
#Code to scrap data from website
from bs4 import BeautifulSoup
import requests
from csv import writer
import csv
import json
city='boston-ma'
main_apt=[]
main_transport=[]
main_amenities=[]
apt_url=[]
apt_add=[]
apt_amenities=[]
apt_bed_area=[]
apt_room_id=[]
apt_price=[]
apt_transport=[]
apt_avail_from=[]
apt_avail_till=[]
apt_bedroom=[]
apt_bathroom=[]
apt_desc=[]
apt_size=[]
id=0
NoneType = type(None)
url=[]
for i in range(1,22):
  if i == 1:
    url.append("https://junehomes.com/residences/boston-ma?count=50")
  else:
    url.append(f"https://junehomes.com/residences/boston-ma?count=50&page={i}")

i=0
for ur in url:
  page = requests.get(ur)
  soup = BeautifulSoup(page.content, 'html.parser')
  aptlist = [soup.find_all('a', class_="ListingCard_root__dWXKe") ,soup.find_all('div', class_="ListingCard_badgeTooltip__INLZY")]
  print(ur)
  k=0

  for apt in aptlist[0]:
    apt_s=str(apt)
    #print(i)

    if len(aptlist[1])<=k :
      break
    if findidbed( str( aptlist[1][k]))!= 'Bedroom':
      break
    i=i+1
    k=k+1
    #print(k)
    href_index = apt_s.find('href')+6  #apartment details on specific apt list
    turl="https://junehomes.com"+apt_s[href_index:apt_s.find('>')-1]
    #print (turl)
    apt_url.append(turl)
    page1 = requests.get(turl)
    soup_apt = BeautifulSoup(page1.content, 'html.parser')
    apt_bed_area.append(str(soup_apt.find('span', class_="Typography_p1-500__fXf6d charcoal-800").text))
    #apt_amenities.append(list(str(soup_apt.find_all('span', class_="Typography_p1-500__fXf6d FeaturesList_label__b8j4n")).replace('<span cla
    apt_amenities.append(str(soup_apt.find_all('span', class_="Typography_p1-500__fXf6d FeaturesList_label__b8j4n")).replace('<span class="Ty
```

```
      tidx=turl.find(city)+ len(city)+1
      turl=turl[tidx:]
      apt_add.append(turl)
      rent=soup_apt.find_all('script',id="__NEXT_DATA__")
      s=str(rent).replace('<script id="__NEXT_DATA__" type="application/json">','').replace('</script>','')
      y=json.loads(s[1:-1])
      if str(y["props"]["pageProps"]).find("room")>3:
        apt_room_id.append("N/A")
        apt_price.append("N/A")
        apt_transport.append("N/A")
        apt_avail_from.append("N/A")
        apt_avail_till.append("N/A")
        apt_avail_from.append("N/A")
        apt_bedroom.append("N/A")
        apt_bathroom.append("N/A")
        apt_desc.append("N/A")
      else:
        apt_room_id.append(y["props"]["pageProps"]["room"]["id"])
        apt_price.append(y["props"]["pageProps"]["room"]["price"])
        apt_transport.append(y["props"]["pageProps"]["room"]["transport"])
        #if y["props"]["pageProps"]["room"]["availability"][0] is not None:
        #print(str(y["props"]["pageProps"]["room"]).find("availability"))
        if y["props"]["pageProps"]["room"]["available"]==False:
          apt_avail_from.append("N/A")
          apt_avail_till.append("N/A")
        else:
          apt_avail_from.append(y["props"]["pageProps"]["room"]["availability"][0])
          apt_avail_till.append(y["props"]["pageProps"]["room"]["availability"][1])
        apt_bedroom.append(y["props"]["pageProps"]["room"]["homeBedrooms"])
        apt_bathroom.append(y["props"]["pageProps"]["room"]["homeBathrooms"])
        apt_desc.append(y["props"]["pageProps"]["room"]["description"])

# header=['id','Apt_id','url','Address','Beds','Bath','Price','BedArea','Amenities','Availablefrom','Availabletill','Transport','Description'
# for j in range(0,i):
#   main_apt.append([j,apt_room_id[j],apt_url[j],apt_add[j],apt_bedroom[j],apt_bathroom[j],apt_price[j],apt_bed_area[j],apt_amenities[j],apt_

header_transport=['id','Transport']
header_amenities=['id','Amenities']
header_apt=['id','Apt_id','url','Address','Beds','Bath','Price','BedArea','Availablefrom','Availabletill','Description']
for j in range(0,i):
  main_transport.append([j,apt_transport[j]])
  # for am in apt_amenities[j]:
  #   main_amenities.append([j,am])
  main_amenities.append([j,apt_amenities[j]])
  main_apt.append([j,apt_room_id[j],apt_url[j],apt_add[j],apt_bedroom[j],apt_bathroom[j],apt_price[j],apt_bed_area[j],apt_avail_from[j],apt_a
```

```
https://junehomes.com/residences/boston-ma?count=50
https://junehomes.com/residences/boston-ma?count=50&page=2
https://junehomes.com/residences/boston-ma?count=50&page=3
https://junehomes.com/residences/boston-ma?count=50&page=4
https://junehomes.com/residences/boston-ma?count=50&page=5
https://junehomes.com/residences/boston-ma?count=50&page=6
https://junehomes.com/residences/boston-ma?count=50&page=7
https://junehomes.com/residences/boston-ma?count=50&page=8
https://junehomes.com/residences/boston-ma?count=50&page=9
https://junehomes.com/residences/boston-ma?count=50&page=10
https://junehomes.com/residences/boston-ma?count=50&page=11
https://junehomes.com/residences/boston-ma?count=50&page=12
https://junehomes.com/residences/boston-ma?count=50&page=13
https://junehomes.com/residences/boston-ma?count=50&page=14
https://junehomes.com/residences/boston-ma?count=50&page=15
https://junehomes.com/residences/boston-ma?count=50&page=16
https://junehomes.com/residences/boston-ma?count=50&page=17
https://junehomes.com/residences/boston-ma?count=50&page=18
https://junehomes.com/residences/boston-ma?count=50&page=19
https://junehomes.com/residences/boston-ma?count=50&page=20
https://junehomes.com/residences/boston-ma?count=50&page=21
```

Loading data into dataframe

```
#Putting into dataframe
import pandas as pd
df_apt = pd.DataFrame (main_apt, columns= ['id','Apt_id','url','Address','Beds','Bath','Price','BedArea','Availablefrom','Availabletill','Des
df_transport=pd.DataFrame (main_transport, columns= ['id','Transport'])
df_amenities=pd.DataFrame (main_amenities, columns= ['id','Amenities'])
df_apt.to_csv('RawJuneApt.csv')
```

```
df_transport.to_csv('RawJuneTransport.csv')
df_amenities.to_csv('RawJuneAmenities.csv')
```

Displays what Apartment looks like before cleaning

```
#display Apartment
main_apt[0]
```

```
[0,
 4198,
 'https://junehomes.com/residences/boston-ma/malden/1459-malden/4198',
 'malden/1459-malden/4198',
 5,
 1,
 800,
 '143',
 '2022-12-08',
 '2023-01-07',
 'This room may be unfurnished or furnished (additional fees apply) at the time of your move-in date. For up to date prices, please
enter your move-in, move-out dates and your furnishing option.']
```

Displays Available Transport Facilities from Apartment

```
#display Transport from Apartment
main_transport[0]
```

```
[0,
 [{'id': 0,
   'stations': ['Blue', 'Line'],
   'color': '#2040AA',
   'walktime': 2,
   'description': 'Maverick'},
  {'id': 1,
   'stations': ['114', '116', '117', '120', '121'],
   'color': '#F5C00E',
   'walktime': 2,
   'description': 'Maverick'}]]
```

Displays Amenities that come with a particular Apartment

```
#display Apartment Amenities
main_amenities[0][1]
```

```
'WiFi (Paid), Smoke-free, Guarantors allowed, Radiator heating, Laundry in building (paid), Street parking'
```

Cleaning the data that has been Web Scraped

Cleaned Transport

```
cleaned_transport=[]

for i in range(0,len(main_transport)):
  for j in range(0, len(main_transport[i][1])):
    tstr=""
    if (str(main_transport[i][1][j]).find("id")==2):
    #print(i,j,main_transport[i][1][j]["stations"])
      for s in main_transport[i][1][j]["stations"]:
        tstr=tstr+","+str(s)
      cleaned_transport.append([main_transport[i][0],main_transport[i][1][j]["id"],tstr,main_transport[i][1][j]["color"],main_transport[i][1]

df_transport=pd.DataFrame (cleaned_transport, columns= ['id','Trans_id','stations','color','walktime','description'])
```

Cleaned Amenities

```
cleaned_amenities=[]

for i in main_amenities:

  for j in i[1].split(", "):
     cleaned_amenities.append([i[0],j])
  #print(i[1].split(", "))

df_amenities=pd.DataFrame (cleaned_amenities, columns= ['id','Amenities'])
```

Cleaned Data

```
df_apt.head()
```

	id	Apt_id	url	Address	Beds	Bath	Price	BedArea	Availablefrom	Availabletill	Description
0	0	4198	https://junehomes.com/residences/boston-ma/mal...	malden/1459-malden/4198	5	1	800	143	2022-12-08	2023-01-07	This room may be unfurnished or furnished (add...
1	1	4197	https://junehomes.com/residences/boston-ma/mal...	malden/1459-malden/4197	5	1	800	145	2022-12-08	2023-01-07	This room may be unfurnished or furnished (add...

```
df_transport.head()
```

	id	Trans_id	stations	color	walktime	description
0	0	0	,Blue,Line	#2040AA	2	Maverick
1	0	1	,114,116,117,120,121	#F5C00E	2	Maverick
2	1	0	,Blue,Line	#2040AA	2	Maverick
3	1	1	,114,116,117,120,121	#F5C00E	2	Maverick
4	2	0	,64	#F5C00E	6	Brooks St @ Faneuil St

```
df_amenities.head()
```

	id	Amenities
0	0	WiFi (Paid)
1	0	Smoke-free
2	0	Guarantors allowed
3	0	Radiator heating
4	0	Laundry in building (paid)

Data Auditing

```
#fetching number of rows and columns
print("Apt:",df_apt.describe())
print("\nTransport:",df_transport.describe())
print("\nAmenities:",df_amenities.describe())
```

```
Apt:                id
count  959.000000
mean   479.000000
std    276.983754
min      0.000000
25%    239.500000
50%    479.000000
75%    718.500000
max    958.000000

Transport:                 id     Trans_id     walktime
count  2939.000000  2939.000000  2939.000000
mean    474.693093     1.228309     7.332766
std     277.115310     1.125514     4.175871
```

```
min       0.000000     0.000000     1.000000
25%     229.000000     0.000000     4.000000
50%     486.000000     1.000000     7.000000
75%     709.500000     2.000000    11.000000
max     957.000000     5.000000    16.000000

Amenities:                 id
count  11133.000000
mean     489.803288
std      277.052487
min        0.000000
25%      253.000000
50%      491.000000
75%      733.000000
max      958.000000
```

```
#fetching number of rows and columns
print("Apt:",df_apt.shape)
print("Transport:",df_transport.shape)
print("Amenities:",df_amenities.shape)
```

```
Apt: (959, 11)
Transport: (2939, 6)
Amenities: (11133, 2)
```

```
print("Apt:")
print (f'id: {df_apt.id.count()}' )
print (f'Apt_id: {df_apt.Apt_id.count()}' )
print (f'url: {df_apt.url.count()}' )
print (f'Address: {df_apt.Address.count()}' )
print (f'Beds: {df_apt.Beds.count()}' )
print (f'Bath: {df_apt.Bath.count()}' )
print (f'Price: {df_apt.Price.count()}' )
print (f'BedArea: {df_apt.BedArea.count()}' )
print (f'Availablefrom: {df_apt.Availablefrom.count()}' )
print (f'Availabletill  : {df_apt.Availabletill .count()}' )
print (f'Description: {df_apt.Description.count()}' )

print("\nTransport:")
print (f'Trans_id: {df_transport.Trans_id.count()}' )
print (f'Station: {df_transport.stations.count()}' )
print (f'id: {df_transport.id.count()}' )
print (f'walktime: {df_transport.walktime.count()}' )
print (f'description: {df_transport.description.count()}' )
```

```
Apt:
id: 959
Apt_id: 959
url: 959
Address: 959
Beds: 959
Bath: 959
Price: 959
BedArea: 959
Availablefrom: 959
Availabletill   : 959
Description: 959

Transport:
Trans_id: 2939
Station: 2939
id: 2939
walktime: 2939
description: 2939
```

Checking for Data Uniqueness

```
#fetching unique values in each column
print("Apt:",df_apt.nunique())
print("\n\nTransport:",df_transport.nunique())
print("\n\nAmenities:",df_amenities.nunique())
```

```
Apt: id              959
Apt_id           941
url              946
Address          946
Beds               8
```

```
Bath              7
Price             56
BedArea          185
Availablefrom    196
Availabletill    195
Description      655
dtype: int64


Transport: id             953
Trans_id         6
stations       178
color           38
walktime        16
description    211
dtype: int64


Amenities: id           959
Amenities     48
dtype: int64
```

```
#Searching for Duplicate Entries
print("Apt:",df_apt.duplicated().sum())
print("\n\nTransport:",df_transport.duplicated().sum())
print("\n\nAmenities:",df_amenities.duplicated().sum())
```

```
Apt: 0


Transport: 0


Amenities: 0
```

Checking for Data Uniformity

```
#Replacing 'N/A' value with null value
df_apt= df_apt.replace('N/A',None)
df_apt = df_apt.where((pd.notnull(df_apt)),None)

df_transport= df_transport.replace('N/A',None)
df_transport = df_transport.where((pd.notnull(df_transport)),None)

df_amenities= df_amenities.replace('N/A',None)
df_amenities = df_amenities.where((pd.notnull(df_amenities)),None)
```

Checking for Data Completeness

```
#Searching for null values in Dataframe

print("Apt:\n",df_apt.isnull().sum())
print("\n\nTransport:\n",df_transport.isnull().sum())
print("\n\nAmenities:\n",df_amenities.isnull().sum())
```

```
Apt:
 id               0
Apt_id           0
url              0
Address          0
Beds             0
Bath             0
Price            0
BedArea          0
Availablefrom    0
Availabletill    0
Description      0
dtype: int64


Transport:
 id             0
Trans_id       0
stations       0
```

```
color          0
walktime       0
description    0
dtype: int64


Amenities:
 id           0
Amenities    0
dtype: int64
```

```
#Boundaries
import numpy as np

print(df_apt.Price.min())
print(df_apt.Price.max())

print(df_apt.Price.quantile(.25))
print(df_apt.Price.quantile(.50))

print(df_apt.Price.quantile (.75))

print(df_apt.Price.mean())

print(df_apt.Price.median())
print(df_apt.Price.mode())
```

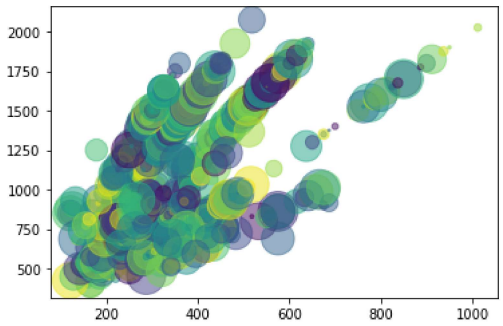
```
800
2375
1300.0
1475.0
1650.0
1472.1324296141815
1475.0
0    1525
dtype: object
```

```
print(df_apt.Price.std())
```

```
267.78271537360115
```

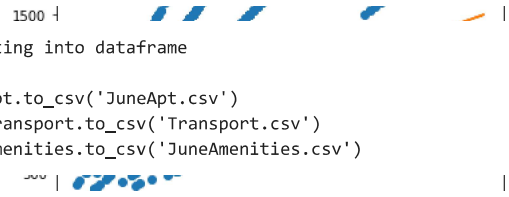
Data Visualization

```
import numpy as np
import matplotlib.pyplot as plt

# Fixing random state for reproducibility
np.random.seed(19680801)

x = [int(x) for x in list(df_apt.Price/df_apt.Beds)]
y =[int(x) for x in list(df_apt.Price/df_apt.Bath)]
N = len(x)
colors = np.random.rand(N)
area = (30 * np.random.rand(N))**2  # 0 to 15 point radii

plt.scatter(x, y, s=area, c=colors, alpha=0.5)
plt.show()
plt.plot(x, y, 'o')
tm, tb = np.polyfit(x, y, 1)
m = int(tm)
plt.plot(x, m*x+b)
```




```
[<matplotlib.lines.Line2D at 0x7fa9d46d42e0>]
```

Loading Cleaned Data into a CSV file




```
#Putting into dataframe

df_apt.to_csv('JuneApt.csv')
df_transport.to_csv('Transport.csv')
df_amenities.to_csv('JuneAmenities.csv')
```



```
import pandas as pd
from sqlalchemy import create_engine
import mysql.connector as msql
from mysql.connector import Error
import re
df_apt['Availablefrom'] = df_apt['Availablefrom'].apply(lambda a: pd.to_datetime(a))
df_apt['Availabletill'] = df_apt['Availabletill'].apply(lambda a: pd.to_datetime(a))
```

Realtime Data collected from Google forms

```
import pandas as pd
from sqlalchemy import create_engine
import re

df_subspot = pd.read_csv('SubletyourSpot.csv')
df_tempspot = pd.read_csv('TemporarySpotSublet.csv')


df_apt['Availabletill'] = df_apt['Availabletill'].apply(lambda a: pd.to_datetime(a))
df_subspot.head()
```

	Name	PhoneNumber	Email	Gender	Address	ProxToUni	Brokerage	LeaseSpotType	BedroomCount	Bathro
0	Siddharth	8578320778	siddharth.bh21@gmail.com	Male	1350 Commonwealth Avenue	2.0	0	On Lease	4	
1	Haseeb	8573707375	haseeb98h@gmail.com	Male	2 Mark Street Apt 2	1.5	0	Off Lease	3	
2	Vishnu Priya Nuthanapati	2677166936	nuthanapati.v@northeastern.edu	Female	463 park drive	0.9	600	Off Lease	3	
3	Chaitanya Patil	8573963630	chaitanyapatil698@gmail.com	Male	160 Williams st, Jamaica Plain, MA - 02130	2.6	208	Off Lease	2	
4	Shailesh	7218922949	shailesh7322@gmail.com	Male	15 Albion Street Roxbury MA	1.4	700	Off Lease	4	

```
df_tempspot.head()
```

	Name	PhoneNumber	Email	Gender	Address	ProximityToUni	BedroomCount	BathroomCount	TempRent	Dietary
0	Neha Bhutkar	8573286597	bhutkar.n@northeastern.edu	Female	75 St. Alphonsus Street, Apt., 1816 J Vue at t...	0.5	2	2	18	Vege
1	Srishti	8572507266	srishtiabalamatti@gmail.com	Female	14 Hillside Street	0.7	5	2	35	
2	Bharath Chandra Bottu	8575446711	bharathchandra2253@gmail.com	Male	2593 Washington St	1.2	3	1	20	Vege
3	Manish	8573790129	m.maniish1@gmail.com	Male	115 Northampton Street	0.7	2	1	18	Vege
4	Nainil	6178607502	nainil30maladkar@gmail.com	Male	Apt 22, 235 Park drive.	0.6	1	1	22	

```
#df_subspot.head()
df_subspot
```

	Name	PhoneNumber	Email	Gender	Address	ProxToUni	Brokerage	LeaseSpotType	BedroomCount
0	Siddharth	8578320778	siddharth.bh21@gmail.com	Male	1350 Commonwealth Avenue	2.0	0	On Lease	4
1	Haseeb	8573707375	haseeb98h@gmail.com	Male	2 Mark Street Apt 2	1.5	0	Off Lease	3
2	Vishnu Priya Nuthanapati	2677166936	nuthanapati.v@northeastern.edu	Female	463 park drive	0.9	600	Off Lease	3
3	Chaitanya Patil	8573963630	chaitanyapatil698@gmail.com	Male	160 Williams st, Jamaica Plain, MA - 02130	2.6	208	Off Lease	2
4	Shailesh	7218922949	shailesh7322@gmail.com	Male	15 Albion Street Roxbury MA	1.4	700	Off Lease	4
5	Akshit Varma	8577469894	akshit.kallepalli@gmail.com	Male	3270 Washington st., Jamaica Plain 02130	2.1	513	Off Lease	3
6	Shaila Verma	8573135613	shailaverma20@gmail.com	Female	1085 Boylston Street, Boston, 02215	0.3	400	Off Lease	1
7	Khushi Raval	8573135636	barodakhushi26@gmail.com	Female	10C horadan way Boston MA 02120	0.5	0	On Lease	2
8	Saad	6172384043	saad.aijazahmed@gmail.com	Male	98-100 Centre St	0.7	750	On Lease	2
9	Rithvik Vanteru	8573353412	rrithvik18@gmail.com	Male	9 Pompeii street unit 1 Roxbury	1.2	0	On Lease	2
10	Keerthana Reddy	8578321796	Hello.keerthanareddy@gmail.com	Female	47 Dalrymple St, Apt 3, Jamaica plain, Boston,...	1.9	410	Off Lease	4
11	Bhanu Sai Simha Vanam	8579301811	vanam.b@northeastern.edu	Male	5B Cornelia Ct	0.7	0	Off Lease	3
12	Mohan Raj Addluru	8374593819	addlurumohanraj@gmail.com	Male	75 ST alphonsus ST, Boston MA	0.9	562	Off Lease	1
13	Nikhil Chaudhary	7709805589	nikhilchaudhary5589@gmail.com	Male	17c Smith Street	1.0	0	Off Lease	2
	Gauravi								

```
print("\nSubleasedSpot:",df_subspot.describe())
print("\nTemporarySpot:",df_tempspot.describe())
```

```
SubleasedSpot:         PhoneNumber  ProxToUni  BedroomCount  BathroomCount  AvailSpotNum
count  3.700000e+01  37.000000     37.000000      37.000000     37.000000
mean   7.588996e+09   1.148649      2.297297       1.243243      1.297297
std    1.486665e+09   0.585293      0.967955       0.434959      0.570812
min    2.677167e+09   0.300000      1.000000       1.000000      1.000000
25%    6.179830e+09   0.700000      2.000000       1.000000      1.000000
50%    8.108149e+09   1.000000      2.000000       1.000000      1.000000
75%    8.573968e+09   1.500000      3.000000       1.000000      1.000000
max    9.398028e+09   2.600000      5.000000       2.000000      3.000000

TemporarySpot:        Proximity to the University (in miles)  No. of Bedrooms  \
count                                     29.000000        29.000000
mean                                       1.010345         2.000000
std                                        0.520539         1.101946
min                                        0.500000         1.000000
25%                                        0.700000         1.000000
50%                                        0.800000         2.000000
75%                                        1.200000         3.000000
max                                        2.500000         5.000000

       No. of Bathrooms  How many spots are you looking to sublease?
count         29.000000                                    29.000000
mean           1.241379                                     1.310345
std            0.435494                                     0.603765
min            1.000000                                     1.000000
25%            1.000000                                     1.000000
50%            1.000000                                     1.000000
75%            1.000000                                     1.000000
max            2.000000                                     3.000000
```

```
print("\nSubleasedSpot:",df_subspot.shape)
print("\nTemporarySpot:",df_tempspot.shape)
```

```
SubleasedSpot: (37, 17)

TemporarySpot: (29, 16)
```

Checking for Data Uniqueness in Realtime Data

```
#fetching unique values in each column
print("SubleasedSpot:",df_subspot.nunique())
print("\nTemporarySpot:",df_tempspot.nunique())
```

```
SubleasedSpot: Name              37
PhoneNumber       37
Email             37
Gender             2
Address           37
ProxToUni         19
Brokerage         20
LeaseSpotType      2
BedroomCount       5
BathroomCount      2
Rent              28
DietaryPref        3
GenderPref         3
Amenities         20
AvailSpot          4
PrefMoveInDate    19
AvailSpotNum       3
dtype: int64

TemporarySpot: Name               29
PhoneNumber        29
Email              29
Gender              2
Address            29
ProximityToUni     14
BedroomCount        5
BathroomCount       2
TempRent           13
DietaryPref         3
GenderPref          3
```

```
Amenities          18
AvailableSpot       6
PrefMoveInDate     19
PrefMoveOutDate    21
AvailSpotNum        3
dtype: int64
```

Checking for Data Completeness in Realtime Data

```
#Searching for null values in Dataframe

print("SubleasedSpot:\n",df_subspot.isnull().sum())
print("\nTemporarySpot:",df_subspot.isnull().sum())
```

```
SubleasedSpot:
 Name              0
PhoneNumber       0
Email             0
Gender            0
Address           0
ProxToUni         0
Brokerage         0
LeaseSpotType     0
BedroomCount      0
BathroomCount     0
Rent              0
DietaryPref       0
GenderPref        0
Amenities         0
AvailSpot         0
PrefMoveInDate    0
AvailSpotNum      0
dtype: int64

TemporarySpot: Name              0
PhoneNumber       0
Email             0
Gender            0
Address           0
ProxToUni         0
Brokerage         0
LeaseSpotType     0
BedroomCount      0
BathroomCount     0
Rent              0
DietaryPref       0
GenderPref        0
Amenities         0
AvailSpot         0
PrefMoveInDate    0
AvailSpotNum      0
dtype: int64
```

```
#df_subspot.pop('Available Spot')
```

```
0                              Shared Bedroom Spot
1                                  Private Bedroom
2                              Shared Bedroom Spot
3                              Shared Bedroom Spot
4                              Shared Bedroom Spot
5                              Shared Bedroom Spot
6                              Shared Bedroom Spot
7                              Shared Bedroom Spot
8                                        Hall Spot
9                              Shared Bedroom Spot
10                             Shared Bedroom Spot
11                                       Hall Spot
12                             Shared Bedroom Spot
13                                 Private Bedroom
14                             Shared Bedroom Spot
15                             Shared Bedroom Spot
16                             Shared Bedroom Spot
17                                 Private Bedroom
18                                 Private Bedroom
19                                       Hall Spot
20                                 Private Bedroom
21                                Shared Hall Spot
22                                Shared Hall Spot
23                             Shared Bedroom Spot
```

```
24              Shared Bedroom Spot;Private Bedroom
25                                Shared Bedroom Spot
26                                    Private Bedroom
27    Hall Spot;Shared Bedroom Spot;Private Bedroom
28                                Shared Bedroom Spot
Name: Available Spot, dtype: object
```

```
#Searching for null values in Dataframe

print("SubleasedSpot:\n",df_subspot.isnull().sum())
print("\nTemporarySpot:\n",df_tempspot.isnull().sum())
```

```
SubleasedSpot:
 Name              0
PhoneNumber       0
Email             0
Gender            0
Address           0
ProxToUni         0
Brokerage         0
LeaseSpotType     0
BedroomCount      0
BathroomCount     0
Rent              0
DietaryPref       0
GenderPref        0
Amenities         0
AvailSpot         0
PrefMoveInDate    0
AvailSpotNum      0
dtype: int64

TemporarySpot:
 Name               0
PhoneNumber        0
Email              0
Gender             0
Address            0
ProximityToUni     0
BedroomCount       0
BathroomCount      0
TempRent           0
DietaryPref        0
GenderPref         0
Amenities          0
AvailableSpot      0
PrefMoveInDate     0
PrefMoveOutDate    0
AvailSpotNum       0
dtype: int64
```

```
#Boundaries
import numpy as np

print(df_subspot.Rent.min())
print(df_subspot.Rent.max())
```

```
25
2500
```

Loading Cleaned Data into a CSV file

```
df_subspot.to_csv('SubleaseSpot.csv')
df_tempspot.to_csv('TemporarySpot.csv')
```

Data Visualization

```
import numpy as np
import matplotlib.pyplot as plt

# Fixing random state for reproducibility
np.random.seed(19680801)

x = list(df_subspot.ProxToUni)
```