```
columns=['hoa','rent_amount','property_tax','fire_insurance']
def convert(val):
    res = ''.join(filter(lambda i: i.isdigit(), val))
    if len(res)==0:
        res=''
```

```
def convert(val):
    res = ''.join(filter(lambda i: i.isdigit(), val))
    if len(res)==0:
        res=''
    return res
```

```
for col in columns:
    df[col] = pd.to_numeric(df[col].apply(convert))
```

```
#Assign your answer to this variable
print(df[columns].isna().sum())
q1=df
```

```
for col in columns1:
    df[col].fillna(df[col].median(),inplace=True)
    df[col]=df[col].astype(int)
print(df[columns1].isna().sum())
df['hoa'].dtype
```

```
hoa              202
property_tax      27
dtype: int64
hoa              0
property_tax     0
dtype: int64
dtype('int64')
```

```
df['furniture'].unique()

dict2={"furnished":1, "not furnished":0}
df['furniture'].replace(dict2,inplace=True)
df['furniture'].unique()
```

```
array([1, 0])
```

```
#Assign your answer to this variable

q3=df
```

```
#Assign your answer to this variable
df['floor']=df['floor'].astype(int)
q4=df['floor']
```

Web Scrapping

```
import lux
import bs4
from bs4 import BeautifulSoup
import csv
import requests
import time
import pandas as pd
import urllib
import re
import pickle
from datetime import datetime
```

executed in 15ms, finished 17:11:39 2021-04-27

```
def clean_data(data):
    if len(data)!=19:
        return 'ERROR'
    res=[]
    for i in data:
        match = re.search(r'\d+.?\d*', i)
        gp=match.group() if match.group()[-1] not in '%°' else match.group()[:-1]
        if gp:
            res.append(float(gp))
    return res

indx=pd.date_range(start='2008-10-28',end='2019-01-01',freq='M')

cols=['Average temperature (°F)', 'Average humidity (%)',
      'Average dewpoint (°F)', 'Average barometer (in)',
      'Average windspeed (mph)', 'Average gustspeed (mph)',
      'Average direction (°deg)', 'Rainfall for month (in)',
      'Rainfall for year (in)', 'Maximum rain per minute',
      'Maximum temperature (°F)', 'Minimum temperature (°F)',
      'Maximum humidity (%)', 'Minimum humidity (%)', 'Maximum pressure',
      'Minimum pressure', 'Maximum windspeed (mph)',
      'Maximum gust speed (mph)', 'Maximum heat index (°F)']
```

executed in 19ms, finished 15:03:17 2021-04-27

```
        gp=match.group() if match.group()[-1] not in '%°' else match.group()[:-1]
        if gp:
            res.append(float(gp))
    return res
```

```
indx=pd.date_range(start='2008-10-26',end='2019-01-01',freq='M')
```

```
cols=['Average temperature (°F)', 'Average humidity (%)',
       'Average dewpoint (°F)', 'Average barometer (in)',
       'Average windspeed (mph)', 'Average gustspeed (mph)',
       'Average direction (°deg)', 'Rainfall for month (in)',
       'Rainfall for year (in)', 'Maximum rain per minute',
       'Maximum temperature (°F)', 'Minimum temperature (°F)',
       'Maximum humidity (%)', 'Minimum humidity (%)', 'Maximum pressure',
       'Minimum pressure', 'Maximum windspeed (mph)',
       'Maximum gust speed (mph)', 'Maximum heat index (°F)']
```

executed in 19ms, finished 15:03:17 2021-04-27

```
df=pd.DataFrame(columns=cols)
```

executed in 16ms, finished 14:59:57 2021-04-27

```
indx
```

executed in 14ms, finished 15:00:29 2021-04-27

```
DatetimeIndex(['2008-10-31', '2008-11-30', '2008-12-31', '2009-01-31',
               '2009-02-28', '2009-03-31', '2009-04-30', '2009-05-31',
               '2009-06-30', '2009-07-31',
               ...
               '2018-03-31', '2018-04-30', '2018-05-31', '2018-06-30',
               '2018-07-31', '2018-08-31', '2018-09-30', '2018-10-31',
               '2018-11-30', '2018-12-31'],
              dtype='datetime64[ns]', length=123, freq='M')
```

```
list(set(indx.map(lambda x: 100*x.year + x.month)))
```

executed in 28ms, finished 15:00:39 2021-04-27

```
dtype='datetime64[ns]', length=123, freq='M')
```

```
list(set(indx.map(lambda x: 100*x.year + x.month)))
```
executed in 28ms, finished 15:00:30 2021-04-27

```
i = 202003
url='http://www.estesparkweather.net/archive_reports.php?date='+str(i)
resp=requests.get(url)
```
executed in 3.63s, finished 15:00:51 2021-04-27

```
soup=BeautifulSoup(resp.content)
```
executed in 272ms, finished 15:04:41 2021-04-27

```
dt=soup.findAll('table')[:-9]
```
executed in 18ms, finished 15:05:59 2021-04-27

```
for k in dt:
    date=str(i)+k.findAll('td')[0].text.split()[1]
    if not date.isnumeric():
        continue
    data=k.findAll('td')[1:]
    f_data=clean_data([i.text.strip() for j,i in enumerate(data) if j%2!=0])
    df.loc[datetime.strptime(date,'%Y%m%d')]=f_data
```
executed in 45ms, finished 15:06:00 2021-04-27


```
list(set(indx.map(lambda x: 100*x.year + x.month)))
```
executed in 28ms, finished 15:00:30 2021-04-27

```
i = 202003
url='http://www.estesparkweather.net/archive_reports.php?date='+str(i)
resp=requests.get(url)
```
executed in 3.63s, finished 15:00:51 2021-04-27

```
soup=BeautifulSoup(resp.content)
```
executed in 272ms, finished 15:04:41 2021-04-27

```
dt=soup.findAll('table')[:-9]
```
executed in 18ms, finished 15:05:59 2021-04-27

```
for k in dt:
    date=str(i)+k.findAll('td')[0].text.split()[1]
    if not date.isnumeric():
        continue
    data=k.findAll('td')[1:]
    f_data=clean_data([i.text.strip() for j,i in enumerate(data) if j%2!=0])
    df.loc[datetime.strptime(date,'%Y%m%d')]=f_data
```
executed in 45ms, finished 15:06:00 2021-04-27

```
#### Start you code here, you are free to add any number of cells

for i in list(set(indx.map(lambda x: 100*x.year + x.month))):
    url='http://www.estesparkweather.net/archive_reports.php?date='+str(i)
#     print(url)
    resp=requests.get(url)
    soup=BeautifulSoup(resp.content)
    dt=soup.findAll('table')[:-9]
    for k in dt:
        date=str(i)+k.findAll('td')[0].text.split()[1]
        if not date.isnumeric():
            continue
        data=k.findAll('td')[1:]
        f_data=clean_data([i.text.strip() for j,i in enumerate(data) if j%2!=0])
        df.loc[datetime.strptime(date,'%Y%m%d')]=f_data
```

2 Visualization

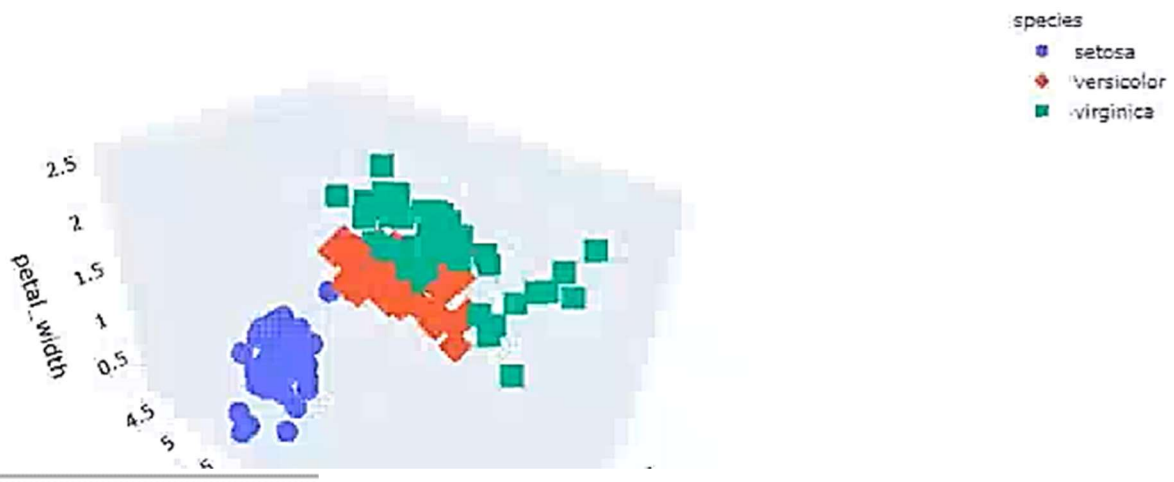
```
import plotly.express as px

df = px.data.iris()

fig = px.scatter_3d(df, x = 'sepal_width',
                    y = 'sepal_length',
                    z = 'petal_width',
                    color = 'species', symbol='species')

fig.show()
```

executed in 120ms, finished 14:23:10 2021-04-27




```
fig.wri
```

write_html
write_image
write_json

```
df[df.city.isna()][['city','venue']]
```

	city	venue
402	NaN	Dubai International Cricket Stadium

```
df.city.fillna(df.venue).loc[415]
```

```
'Dubai International Cricket Stadium'
```

```
df.city.fillna(df.venue.str.split().str[0],inplace=True)
```

```
df.city.loc[415]
```

```
'Dubai'
```

```
#Assign your answer to this variable
q1= df.city.to_list()
```

Web scrapping 2

```
#import statements
import pandas as pd
import numpy as np

import requests
from bs4 import BeautifulSoup as bs
```

```
!pip install pandas
```

```
res = requests.get("http://lum-surprise.surge.sh/")

soup = bs(res.content, 'html.parser')

table = soup.find_all('table')[0]
df = pd.read_html(str(table))[0]

df.head()
```

	Gender	Length	Diameter	Height	Whole_weight	Shucked_weight	Viscera_weight	Shell_weight	Rings
0	MALE	0.455	0.365	0.095	0.5140	0.2245	0.1010	0.1500	15
1	F	NaN	0.500	0.160	1.2465	0.5475	0.3270	0.3000	10
2	M	0.430	0.335	0.120	0.3970	0.1985	0.0865	0.1035	7

```
a = df[df["Shucked_weight"]==1.253]
q1= a.values.flatten().tolist()

q1
```

```
['M', nan, 0.57, 0.19, 2.3305, 1.253, 0.541, 0.52, 9]
```