Scraping notebook

All the steps of scraping demographics data from www.zipdatamaps.com, is shown here.

And the cleaning of the data

In [1]:
```
from html_table_parser.parser import HTMLTableParser
from pprint import pprint
import urllib.request
import pandas as pd
import numpy as np
import stringcase
import re
```

In [2]:
```
# zip codes needed
zip_list = np.array([14580, 14623, 14612, 14616, 14624, 14607, 14526, 14620, 14606,
       14618, 14621, 14615, 14609, 14514, 14472, 14619, 14559, 14617,
       14611, 14626, 14450, 14622, 14467, 14625, 14445, 14610], dtype=str)

zip_list
```

Out[2]:
```
array(['14580', '14623', '14612', '14616', '14624', '14607', '14526',
       '14620', '14606', '14618', '14621', '14615', '14609', '14514',
       '14472', '14619', '14559', '14617', '14611', '14626', '14450',
       '14622', '14467', '14625', '14445', '14610'], dtype='<U11')
```

In [3]:
```
#Function to return a dataframe of zip data information
def get_table(zipcode):

    url = 'https://www.zipdatamaps.com/' + zipcode


    agent = 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_3) AppleWebKit/537.36\
    (KHTML, like Gecko) Chrome/35.0.1916.47 Safari/537.36'

    #making request to the website
    req = urllib.request.Request(url=url, headers={'User-Agent': agent})
    f = urllib.request.urlopen(req)


    # defining the html contents of a URL.
    xhtml = f.read().decode('utf-8')

    # HTMLTableParser object
    p = HTMLTableParser()

    # feeding the html contents in the
    # HTMLTableParser object
    p.feed(xhtml)

    #returning table as a dataframe
    return pd.DataFrame(p.tables[0])
```

In [4]:
```
def process_df(df,zipcode):
    #Turning the table from the website into a useable, dataframe.

    # Transposing dataframe
    df = df.T

    #Applying processing steps
    df.columns = [stringcase.alphanumcase(x) for x in df.iloc[0]]
    df.drop(index=0, inplace=True)
    df.reset_index(inplace=True, drop=True)
    # Adding zipcode column
    df['zipcode'] = zipcode

    return df
```

In [5]:
```
#Making new dataframe
df_list = []

for x in zip_list:

    zip_table =  get_table(x)
    df_list.append(process_df(zip_table,x))

result = pd.concat(df_list)
result.reset_index(inplace=True, drop= True)
```

In [6]:
```
result.columns
```

Out[6]:
```
Index(['OfficialZipCodeName', 'ZipCodeState', 'ZipCodeType', 'PrimaryCounty',
       'SecondaryCounty', 'AreaCode', 'CurrentPopulation', 'RacialMajority',
       'PublicSchoolRacialMajority', 'UnemploymentRate',
       'MedianHouseholdIncome', 'AverageAdjustedGrossIncome',
       'SchoolTestPerformance', 'AverageCommuteTime', 'TimeZone',
       'ElevationRange', 'Area', 'CoordinatesYX', 'zipcode'],
      dtype='object')
```

In [7]:
```
df_demographics = result[['CurrentPopulation', 'RacialMajority',
       'PublicSchoolRacialMajority', 'UnemploymentRate',
       'MedianHouseholdIncome', 'AverageAdjustedGrossIncome',
       'SchoolTestPerformance',
       'Area','zipcode']]
```

In [8]:
```
cols = list(df_demographics.columns)
cols = [cols[-1]] + cols[0:-1]
df_demographics = df_demographics[cols]
df_demographics.head()
```

Out[8]:

	zipcode	CurrentPopulation	RacialMajority	PublicSchoolRacialMajority	UnemploymentRate	MedianHouseholdIncome	AverageAdjustedGrossI
0	14580	50587	White 90.24%	White 87.1%	5.3%	$75618	
1	14623	27173	White 71.17%	White 56%	5.5%	$54283	
2	14612	34515	White 86.99%	White 69.2%	7.3%	$62148	
3	14616	28534	White 82.2%	White 67.2%	10.8%	$55438	
4	14624	36296	White 81.99%	White 66.6%	6.5%	$69312	

In [9]:
```
df_demographics.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 26 entries, 0 to 25
Data columns (total 9 columns):
 #   Column                     Non-Null Count  Dtype
---  ------                     --------------  -----
 0   zipcode                    26 non-null     object
 1   CurrentPopulation          26 non-null     object
 2   RacialMajority             26 non-null     object
 3   PublicSchoolRacialMajority  25 non-null     object
 4   UnemploymentRate           26 non-null     object
 5   MedianHouseholdIncome      26 non-null     object
 6   AverageAdjustedGrossIncome  26 non-null     object
 7   SchoolTestPerformance      26 non-null     object
 8   Area                       26 non-null     object
dtypes: object(9)
memory usage: 2.0+ KB
```

Data cleaning and formatting

In [10]:
```
df_demographics.columns
```

Out[10]:
```
Index(['zipcode', 'CurrentPopulation', 'RacialMajority',
       'PublicSchoolRacialMajority', 'UnemploymentRate',
       'MedianHouseholdIncome', 'AverageAdjustedGrossIncome',
       'SchoolTestPerformance', 'Area'],
      dtype='object')
```

In [11]:
```
#Fixing types

df_demographics['CurrentPopulation'] = \
df_demographics['CurrentPopulation'].astype(int)

df_demographics['UnemploymentRate'] = \
df_demographics['UnemploymentRate'].apply(lambda x: x[0:-1]).astype(float)

df_demographics['MedianHouseholdIncome'] = \
df_demographics['MedianHouseholdIncome'].apply(lambda x: x[1:]).astype(float)

df_demographics['AverageAdjustedGrossIncome'] = \
df_demographics['AverageAdjustedGrossIncome'].apply(lambda x: x[1:]).astype(float)

df_demographics['Area'] = \
df_demographics['Area'].apply(lambda x: ''.join(re.findall(r'\d',x))).astype(float)

df_demographics = df_demographics.dropna()
```

In [12]:
```
df_demographics.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 25 entries, 0 to 25
Data columns (total 9 columns):
 #   Column                     Non-Null Count  Dtype
---  ------                     --------------  -----
 0   zipcode                    25 non-null     object
 1   CurrentPopulation          25 non-null     int32
 2   RacialMajority             25 non-null     object
 3   PublicSchoolRacialMajority  25 non-null     object
 4   UnemploymentRate           25 non-null     float64
 5   MedianHouseholdIncome      25 non-null     float64
 6   AverageAdjustedGrossIncome  25 non-null     float64
 7   SchoolTestPerformance      25 non-null     object
 8   Area                       25 non-null     float64
dtypes: float64(4), int32(1), object(4)
memory usage: 1.9+ KB
```

In [13]:
```
df_demographics.head()
```

Out[13]:

	zipcode	CurrentPopulation	RacialMajority	PublicSchoolRacialMajority	UnemploymentRate	MedianHouseholdIncome	AverageAdjustedGrossI
0	14580	50587	White 90.24%	White 87.1%	5.3	75618.0	8
1	14623	27173	White 71.17%	White 56%	5.5	54283.0	4
2	14612	34515	White 86.99%	White 69.2%	7.3	62148.0	6
3	14616	28534	White 82.2%	White 67.2%	10.8	55438.0	4
4	14624	36296	White 81.99%	White 66.6%	6.5	69312.0	6

In [14]:
```
# Fixing the PublicSchoolRacialMajority column

s = df_demographics['PublicSchoolRacialMajority'].apply(lambda x: x.split(' '))
df_demographics['SchoolRacialMajority'] = s.apply(lambda x: x[0])
df_demographics['SchoolRacialMajorityPercentage'] = s.apply(lambda x: x[1][0:-1]).astype(float)
df_demographics = df_demographics.drop(['PublicSchoolRacialMajority'], axis = 1)
```

In [15]:
```
# Fixing the PublicSchoolRacialMajority column

df_demographics['RacialMajority'] = \
df_demographics['RacialMajority'].apply(lambda x: 'Black ' + x.split(' ')[2] if 'African' in x.split(' ') else x.split(' ')[0])


s = df_demographics['RacialMajority'].apply(lambda x: x.split(' '))
df_demographics['RacialMajority'] = s.apply(lambda x: x[0])
df_demographics['RacialMajorityPercentage'] = s.apply(lambda x: x[1][0:-1]).astype(float)
```

In [16]:
```
df_demographics.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 25 entries, 0 to 25
Data columns (total 11 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   zipcode                         25 non-null     object
 1   CurrentPopulation               25 non-null     int32
 2   RacialMajority                  25 non-null     object
 3   UnemploymentRate                25 non-null     float64
 4   MedianHouseholdIncome           25 non-null     float64
 5   AverageAdjustedGrossIncome      25 non-null     float64
 6   SchoolTestPerformance           25 non-null     object
 7   Area                            25 non-null     float64
 8   SchoolRacialMajority            25 non-null     object
 9   SchoolRacialMajorityPercentage  25 non-null     float64
 10  RacialMajorityPercentage        25 non-null     float64
dtypes: float64(6), int32(1), object(4)
memory usage: 2.2+ KB
```

In [17]:
```
df_demographics.head()
```

Out[17]:

	zipcode	CurrentPopulation	RacialMajority	UnemploymentRate	MedianHouseholdIncome	AverageAdjustedGrossIncome	SchoolTestPerforma
0	14580	50587	White	5.3	75618.0	81510.0	Above Averag
1	14623	27173	White	5.5	54283.0	49130.0	Above Averag
2	14612	34515	White	7.3	62148.0	61510.0	Averag
3	14616	28534	White	10.8	55438.0	46270.0	Above Averag
4	14624	36296	White	6.5	69312.0	62360.0	Averag

In [18]:
```
df_demographics.to_csv('zip_demographics.csv', index = False)
```