In [1]:
```
import numpy as np
import pandas as pd
```

Emigma Housing prize Dataset

In [2]:
```
#Enigma housing prize dataset
df=pd.read_csv("19b7b17c-e655-4013-b4d1-7daa9727225f_NewYorkCityPrope
```

In [3]:
```
df.head()
```

Out[3]:

	borough_code	borough_code_definition	neighborhood	building_class_category_code	building_
0	1	Manhattan	ALPHABET CITY	07	RE
1	1	Manhattan	ALPHABET CITY	07	RE
2	1	Manhattan	ALPHABET CITY	07	RE
3	1	Manhattan	ALPHABET CITY	07	RE
4	1	Manhattan	ALPHABET CITY	07	RE

5 rows × 62 columns

NYC Demographic dataset

In [58]:
```
#Demographics dataset
df_socio_economic=pd.read_csv("Demographic_Statistics_By_Zip_Code.csv
df_socio_economic.head()
```

Out[58]:

	JURISDICTION NAME	COUNT PARTICIPANTS	COUNT FEMALE	PERCENT FEMALE	COUNT MALE	PERCENT MALE	COUNT GENDER UNKNOWN	PERCE GEND UNKNO
0	10001	44	22	0.50	22	0.50	0	
1	10002	35	19	0.54	16	0.46	0	
2	10003	1	1	1.00	0	0.00	0	
3	10004	0	0	0.00	0	0.00	0	
4	10005	2	2	1.00	0	0.00	0	

5 rows × 46 columns

In [6]:
```
1 df_socio_economic.columns
```

Out[6]:
```
Index(['JURISDICTION NAME', 'COUNT PARTICIPANTS', 'COUNT FEMALE',
       'PERCENT FEMALE', 'COUNT MALE', 'PERCENT MALE', 'COUNT GENDER U
NKNOWN',
       'PERCENT GENDER UNKNOWN', 'COUNT GENDER TOTAL', 'PERCENT GENDER
TOTAL',
       'COUNT PACIFIC ISLANDER', 'PERCENT PACIFIC ISLANDER',
       'COUNT HISPANIC LATINO', 'PERCENT HISPANIC LATINO',
       'COUNT AMERICAN INDIAN', 'PERCENT AMERICAN INDIAN',
       'COUNT ASIAN NON HISPANIC', 'PERCENT ASIAN NON HISPANIC',
       'COUNT WHITE NON HISPANIC', 'PERCENT WHITE NON HISPANIC',
       'COUNT BLACK NON HISPANIC', 'PERCENT BLACK NON HISPANIC',
       'COUNT OTHER ETHNICITY', 'PERCENT OTHER ETHNICITY',
       'COUNT ETHNICITY UNKNOWN', 'PERCENT ETHNICITY UNKNOWN',
       'COUNT ETHNICITY TOTAL', 'PERCENT ETHNICITY TOTAL',
       'COUNT PERMANENT RESIDENT ALIEN', 'PERCENT PERMANENT RESIDENT A
LIEN',
       'COUNT US CITIZEN', 'PERCENT US CITIZEN', 'COUNT OTHER CITIZEN
STATUS',
       'PERCENT OTHER CITIZEN STATUS', 'COUNT CITIZEN STATUS UNKNOWN',
       'PERCENT CITIZEN STATUS UNKNOWN', 'COUNT CITIZEN STATUS TOTAL',
       'PERCENT CITIZEN STATUS TOTAL', 'COUNT RECEIVES PUBLIC ASSISTAN
CE',
       'PERCENT RECEIVES PUBLIC ASSISTANCE',
       'COUNT NRECEIVES PUBLIC ASSISTANCE',
       'PERCENT NRECEIVES PUBLIC ASSISTANCE',
       'COUNT PUBLIC ASSISTANCE UNKNOWN', 'PERCENT PUBLIC ASSISTANCE U
NKNOWN',
       'COUNT PUBLIC ASSISTANCE TOTAL', 'PERCENT PUBLIC ASSISTANCE TOT
AL'],
      dtype='object')
```

In [60]:
```
1 processed_socio_economic=df_socio_economic.drop(['PERCENT FEMALE','PE
```

Foursquare Dataset

In [7]:
```
1 df_four=pd.read_table("new york_anon_locationData_newcrawl.txt")
```

In [8]:
```
#Foursquare dataset
import re
handle=open('new york_anon_locationData_newcrawl.txt')
foursquaredf=pd.DataFrame()
idlist=[]
restlist=[]
translator = str.maketrans('', '', '*()')
for line in handle:
    line=line.strip()
    words=line.split(';')
    #print(words[0])
    #print(words[1])
    words[0]=int(words[0].translate(translator))
    words[1]=words[1].translate(translator)
    words[1]=words[1].split(',')
    #print(words[0])
    idlist.append(words[0])
    restlist.append(words[1])
    #print(words[1].split(','))
foursquaredf=pd.DataFrame(restlist)
foursquaredf=foursquaredf.iloc[:,:6]
foursquaredf['id']=idlist
```

In [9]:
```
#Work on Foursquare dataset to bring into the required format
foursquaredf_final=pd.DataFrame()
foursquaredf_final['latitude']=foursquaredf[0].astype(float)
foursquaredf_final['longitude']=foursquaredf[1].astype(float)
foursquaredf_final['type']=foursquaredf[2].astype(str)
foursquaredf_final['num1']=foursquaredf[3]
foursquaredf_final['num2']=foursquaredf[5]
foursquaredf_final['id']=foursquaredf['id']
```

In [10]:
```
foursquaredf_final.head()
```

Out[10]:

	latitude	longitude	type	num1	num2	id
0	40.760265	-73.989105	'Italian'	'217'	'Ristorante Da Rosina'	42889
1	40.780704	-73.954704	'Bakery'	'1291'	'Le Pain Quotidien'	57489
2	40.663925	-74.118230	'Video Store'	'60'	'Blockbuster'	42890
3	40.739840	-73.991770	"Corporate ' Office"	'6'	'Day & Night Office'	74771
4	40.786987	-73.807530	'Other - Food'	'29'	"Erin's Isle"	42891

In [11]:
```
#map each of FOursquare venues to one of 7 categories
from pydoc import deque
import json

class _Node(object):

    def __init__(self, parent, data):
        self.parent = parent
        self.data = data
        self.children = []
```

In [12]:
```
#map each of FOursquare venues to one of 7 categories
class Categories(object):

    def __init__(self):
        self._categories = None
        # self._resources = Resources()

        self._root = None
        self._name_category_map = {}
        self._short_name_category_map = {}

        self._load()
        self._load_colors()

    def _load(self):

        json_data=open('categories.json')
        data = json.load(json_data)
        json_data.close()
        self._categories = data

        self._root = _Node(None, None)
        nodeQueue = deque([])
        nodeQueue.append(self._root)

        objectQueue = deque([])
        objectQueue.append(self._categories)

        while(len(nodeQueue) > 0):
            node = nodeQueue.pop()
            category = objectQueue.pop()
            if not (node.data is None):
                self._name_category_map[node.data['name']] = node
                self._short_name_category_map[node.data['shortName']]

            categories = category.get('categories')
            if not (categories is None):
                for child in categories:
```

```
                childNode = _Node(node, { 'name': child['pluralNa
                                          'shortName': child['shor
                node.children.append(childNode)

                nodeQueue.append(childNode)
                objectQueue.append(child)

    def get_parent(self, category):
        node = self._name_category_map.get(category)
        if node is None:
            node = self._short_name_category_map.get(category)
            if node is None:
                return None

        while(node.parent != None and node.parent.data != None):
            node = node.parent
            break

        return node.data['name']



    def get_top_parent(self, category):
        node = self._name_category_map.get(category)
        if node is None:
            node = self._short_name_category_map.get(category)
            if node is None:
                return None

        while(node.parent != None and node.parent.data != None):
            node = node.parent

        return node.data['name']
    def _load_colors(self):
        self._colors = { 'Arts & Entertainment' : 'r', 'College & Uni
                 'Food' : 'b', 'Nightlife Spot' : '#DDFF00', 'Outdoo
                 'Professional & Other Places' : 'c', 'Residence' :
                 'Shop & Service' : '#800000', 'Travel & Transport'



```

In [13]:
```
#map each of FOursquare venues to one of 7 categories
import re
dog=Categories()
cate=[]
for i,row in foursquaredf_final.iterrows():
    tr=re.sub('[\'\"]', '', row['type'])
    ssss=re.sub('^[ ]','',tr)
    ssss=re.sub('  ',' ',ssss)
    cate.append(dog.get_top_parent(ssss))
```

In [14]:
```
#map each of FOursquare venues to one of 7 categories
four_reg=foursquaredf_final
four_reg['category']=cate

four_reg.category = pd.Categorical(four_reg.category)
four_reg['category_code'] = four_reg.category.cat.codes
```

In [15]:
```
four_reg.head()
```

Out[15]:

	latitude	longitude	type	num1	num2	id	category	category_code
0	40.760265	-73.989105	'Italian'	'217'	'Ristorante Da Rosina'	42889	Food	2
1	40.780704	-73.954704	'Bakery'	'1291'	'Le Pain Quotidien'	57489	Food	2
2	40.663925	-74.118230	'Video Store'	'60'	'Blockbuster'	42890	Shops & Services	7
3	40.739840	-73.991770	"Corporate ' Office"	'6'	'Day & Night Office'	74771	Professional & Other Places	5
4	40.786987	-73.807530	'Other - Food'	'29'	"Erin's Isle"	42891	NaN	-1

In [16]:
```
#import the USZipcode api to map longitude and latitude to zip code
from uszipcode import ZipcodeSearchEngine
search = ZipcodeSearchEngine()
res = search.by_coordinate(40.760265, -73.989105, radius=1, returns=1
```

In [17]:
```
#See what does the API return
res
```

Out[17]:
[{"City": "New York", "Density": 56161.36363636363, "HouseOfUnits": 17
958, "LandArea": 0.44, "Latitude": 40.7602619, "Longitude": -73.993287
2, "NEBoundLatitude": 40.768738, "NEBoundLongitude": -73.9781161, "Pop
ulation": 24711, "SWBoundLatitude": 40.723624900000004, "SWBoungLongit
ude": -74.004786, "State": "NY", "TotalWages": 1686575064.0, "WaterAre
a": 0.0, "Wealthy": 68251.99562947675, "Zipcode": "10036", "ZipcodeTyp
e": "Standard"}]

In [18]:
```
#The API returns all information about zip code ranging from socio ec
#data to demographic data which could also be used for analysis of ho
zipcodes=[]
density=[]
houseofunit=[]
landarea=[]
pop=[]
totwage=[]
water_area=[]
wealthy=[]
zipped_df=pd.DataFrame()
for i,row in four_reg.iterrows():
    lat=float(row['latitude'])
    lon=float(row['longitude'])
    #print(lat,lon)
    try:
        res=search.by_coordinate(lat, lon, radius=1, returns=1)[0]
        zipp=res['Zipcode']
        densityl=res['Density']
        houseofunitl=res['HouseOfUnits']
        landareal=res['LandArea']
        popl=res['Population']
        totwagel=res['TotalWages']
        water_areal=res['WaterArea']
        wealthyl=res['Wealthy']
    except:
        zipp=-1
        densityl=-1
        houseofunitl=-1
        landareal=-1
        pop1=-1
        totwagel=-1
        water_areal=-1
        wealthyl=-1
    #print(res)
    zipcodes.append(zipp)
    density.append(densityl)
    houseofunit.append(houseofunitl)
    landarea.append(landareal)
```

```
    landarea.append(landareal)
    pop.append(popl)
    totwage.append(totwagel)
    water_area.append(water_areal)
    wealthy.append(wealthyl)
four_reg['zipcode']=zipcodes
zipped_df['zipcode']=zipcodes
zipped_df['density']=density
zipped_df['houseofunits']=houseofunit
zipped_df['landarea']=landarea
zipped_df['pop']=pop
zipped_df['totwage']=totwage
zipped_df['water_area']=water_area
zipped_df['wealthy']=wealthy
```

In [19]:
```
four_reg.head()
```

Out[19]:

	latitude	longitude	type	num1	num2	id	category	category_code	zip
0	40.760265	-73.989105	'Italian'	'217'	'Ristorante Da Rosina'	42889	Food	2	
1	40.780704	-73.954704	'Bakery'	'1291'	'Le Pain Quotidien'	57489	Food	2	
2	40.663925	-74.118230	'Video Store'	'60'	'Blockbuster'	42890	Shops & Services	7	
3	40.739840	-73.991770	"Corporate ' Office"	'6'	'Day & Night Office'	74771	Professional & Other Places	5	
4	40.786987	-73.807530	'Other - Food'	'29'	"Erin's Isle"	42891	NaN	-1	

In [76]:
```
zipped_df.head()
zipped_df.to_csv("zip_api_data.csv")
```

In [55]:
```
#COunt the venues in each of the zip codes
zip_codes=four_reg['zipcode'].unique()
list_dicts=[]
for j,z in enumerate(zip_codes):
    dict_count=dict.fromkeys(list(four_reg['category'].unique()),0)
    df_zip_part=four_reg[four_reg['zipcode']==z]
    for i,row in df_zip_part.iterrows():
        dict_count[row['category']]+=1
    list_dicts.append(dict_count)
```

In [56]:
```
zipped_foursquare=pd.DataFrame.from_dict(list_dicts)
zipped_foursquare['zipcode']=zip_codes
```

In [75]:
```
#save the resulting datasets to csv file
zipped_foursquare.head()
zipped_foursquare.to_csv("foursquare.csv")
```

In [74]:
```
processed_socio_economic.head()
processed_socio_economic.to_csv("Socio_economic.csv")
```

NYC Taxi Data

In [68]:
```
df_taxi_jan=pd.read_csv('yellow_tripdata_2016-01.csv')
```

In [69]:
```
df_taxi_jan.head()
```

Out[69]:

	VendorID	tpep_pickup_datetime	tpep_dropoff_datetime	passenger_count	trip_distance	pickup
0	2	2016-01-01 00:00:00	2016-01-01 00:00:00	2	1.10	
1	2	2016-01-01 00:00:00	2016-01-01 00:00:00	5	4.90	
2	2	2016-01-01 00:00:00	2016-01-01 00:00:00	1	10.54	
3	2	2016-01-01 00:00:00	2016-01-01 00:00:00	1	4.75	
4	2	2016-01-01 00:00:00	2016-01-01 00:00:00	3	1.76	

In [71]:
```
df_taxi_jan=df_taxi_jan[['pickup_longitude','pickup_latitude','dropof
```

In [72]:
```
df_taxi_jan.head()
```

Out[72]:

	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude
0	-73.990372	40.734695	-73.981842	40.732407
1	-73.980782	40.729912	-73.944473	40.716679
2	-73.984550	40.679565	-73.950272	40.788925
3	-73.993469	40.718990	-73.962242	40.657333
4	-73.960625	40.781330	-73.977264	40.758514

In [ ]:
```
df_taxi_jan.head()
```

In [78]:
```
sample_taxi=df_taxi_jan.sample(100000)
```

In [85]:
```
1 sample_taxi.head()
```

Out[85]:

	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude
4231934	-73.963196	40.793873	-73.979362	40.776642
4279558	-73.991035	40.751293	-73.979820	40.746521
6514632	-73.984375	40.722385	-73.981468	40.716774
4841223	-73.982559	40.739552	-73.967133	40.757561
10219474	-73.971558	40.758381	-73.972870	40.795879

In [ ]:
```
1  #Map latitude and longitude to ZIP code
2  dict_pick_count=dict()
3  dict_drop_count=dict()
4  for i,row in sample_taxi.iterrows():
5      pick_lon=float(row['pickup_longitude'])
6      pick_lat=float(row['pickup_latitude'])
7
8      drop_lon=float(row['dropoff_longitude'])
9      drop_lat=float(row['dropoff_latitude'])
10
11     print("hi")
12     try:
13         res_pick=search.by_coordinate(pick_lat, pick_lon, radius=1, r
14         print(res_pick)
15         res_pick_zip=res_pick['Zipcode']
16         res_drop=search.by_coordinate(drop_lat,drop_lon,radius=1,retu
17     except:
18         pass
19     print(res_pick)
20
21
22     if res_pick in dict_pick_count.keys():
23         dict_pick_count[res_pick]+=1
24     else:
25         dict_pick_count[res_pick]=0
26     if res_drop in dict_drop_count.keys():
27         dict_drop_count[res_drop]+=1
28     else:
29         dict_drop_count[res_drop]=0
30
```

```
hi
{
    "City": "New York",
    "Density": 126133.33333333331,
    "HouseOfUnits": 47617,
    "LandArea": 0.75
```

```
 "LandArea": 0.75,
"Latitude": 40.7999209,
"Longitude": -73.96831019999998,
"NEBoundLatitude": 40.8108,
"NEBoundLongitude": -73.95699309999998,
"Population": 94600,
"SWBoundLatitude": 40.7878149,
"SWBoungLongitude": -73.98440699999998,
"State": "NY",
"TotalWages": 1675737238.0,
"WaterArea": 0.0,
"Wealthy": 17713.924291754756,
"Zipcode": "10025",
"ZipcodeType": "Standard"
}
```


In [ ]:
```
1
```