Final Project Report(Milestone 1 to 4)

Milestone 1 : Data Source

https://www.kaggle.com/c/zillow-prize-1 (https://www.kaggle.com/c/zillow-prize-1)

Description

There are two data sets with over 1 million records each and 58 columns. properties_2016 and properties_2017 datasets contain data for each year. The data we will use for this project will be a small sample of the master data.

The two datasets are linked by parcleid.

I transactions dataset, the trabsaction date shows the date the property was sold and logerror is the log10(estimated price - price sold).

Properties dataset has the physical information about the properities. The columns on the properties dataset will have to be renamed. Subsets of data can be used to group by region, and other features such as number of bedrooms, square footage, etc.

In [117]:

```
# Load Libraries
import pandas as pd
import matplotlib.pyplot as plt
import xlrd
import numpy as np
# Load Data
transactions_2016 = "Data/transactions_2016.json"
transactions_2017 = "Data/transactions_2017.json"

properties_2016  =  "Data/properties_2016.csv"
properties_2017  =  "Data/properties_2017.csv"
data_dictionary = "Data/data_dictionary.xlsx"

transactions_2016 = pd.read_json(transactions_2016)
transactions_2017 = pd.read_json(transactions_2017)
properties_2016 = pd.read_csv(properties_2016)
properties_2017 = pd.read_csv(properties_2017)
data_dictionary = pd.read_excel(data_dictionary)
```

```
c:\users\safar\documents\github\safarie1103\bellevue university\courses\d
sc540\venv\lib\site-packages\IPython\core\interactiveshell.py:3063: Dtype
Warning: Columns (50) have mixed types.Specify dtype option on import or
set low_memory=False.
  interactivity=interactivity, compiler=compiler, result=result)
c:\users\safar\documents\github\safarie1103\bellevue university\courses\d
sc540\venv\lib\site-packages\IPython\core\interactiveshell.py:3063: Dtype
Warning: Columns (23,50) have mixed types.Specify dtype option on import
or set low_memory=False.
  interactivity=interactivity, compiler=compiler, result=result)
```

In [2]:

```
transactions_2016.head()
```

Out[2]:

	parcelid	logerror	transactiondate
0	11016594	0.0276	2016-01-01
1	14366692	-0.1684	2016-01-01
2	12098116	-0.0040	2016-01-01
3	12643413	0.0218	2016-01-02
4	14432541	-0.0050	2016-01-02

In [3]:
```
properties_2016.head()
```

Out[3]:

	Unnamed: 0	parcelid	airconditioningtypeid	architecturalstyletypeid	basements
0	0	10754147	NaN	NaN	N
1	1	10759547	NaN	NaN	N
2	2	10843547	NaN	NaN	N
3	3	10859147	NaN	NaN	N
4	4	10879947	NaN	NaN	N

5 rows × 59 columns

In [4]:
```
print(len(properties_2016.columns))
print(properties_2016.columns)
```

```
59
Index(['Unnamed: 0', 'parcelid', 'airconditioningtypeid',
       'architecturalstyletypeid', 'basementsqft', 'bathroomcnt', 'bedroo
mcnt',
       'buildingclasstypeid', 'buildingqualitytypeid', 'calculatedbathnb
r',
       'decktypeid', 'finishedfloor1squarefeet',
       'calculatedfinishedsquarefeet', 'finishedsquarefeet12',
       'finishedsquarefeet13', 'finishedsquarefeet15', 'finishedsquarefee
t50',
       'finishedsquarefeet6', 'fips', 'fireplacecnt', 'fullbathcnt',
       'garagecarcnt', 'garagetotalsqft', 'hashottuborspa',
       'heatingorsystemtypeid', 'latitude', 'longitude', 'lotsizesquarefe
et',
       'poolcnt', 'poolsizesum', 'pooltypeid10', 'pooltypeid2', 'pooltype
id7',
       'propertycountylandusecode', 'propertylandusetypeid',
       'propertyzoningdesc', 'rawcensustractandblock', 'regionidcity',
       'regionidcounty', 'regionidneighborhood', 'regionidzip', 'roomcn
t',
       'storytypeid', 'threequarterbathnbr', 'typeconstructiontypeid',
       'unitcnt', 'yardbuildingsqft17', 'yardbuildingsqft26', 'yearbuil
t',
       'numberofstories', 'fireplaceflag', 'structuretaxvaluedollarcnt',
       'taxvaluedollarcnt', 'assessmentyear', 'landtaxvaluedollarcnt',
       'taxamount', 'taxdelinquencyflag', 'taxdelinquencyyear',
       'censustractandblock'],
      dtype='object')
```

In [5]:

```
print(len(properties_2017.columns))
print(properties_2017.columns)
```

```
59
Index(['Unnamed: 0', 'parcelid', 'airconditioningtypeid',
       'architecturalstyletypeid', 'basementsqft', 'bathroomcnt', 'bedroo
mcnt',
       'buildingclasstypeid', 'buildingqualitytypeid', 'calculatedbathnb
r',
       'decktypeid', 'finishedfloor1squarefeet',
       'calculatedfinishedsquarefeet', 'finishedsquarefeet12',
       'finishedsquarefeet13', 'finishedsquarefeet15', 'finishedsquarefee
t50',
       'finishedsquarefeet6', 'fips', 'fireplacecnt', 'fullbathcnt',
       'garagecarcnt', 'garagetotalsqft', 'hashottuborspa',
       'heatingorsystemtypeid', 'latitude', 'longitude', 'lotsizesquarefe
et',
       'poolcnt', 'poolsizesum', 'pooltypeid10', 'pooltypeid2', 'pooltype
id7',
       'propertycountylandusecode', 'propertylandusetypeid',
       'propertyzoningdesc', 'rawcensustractandblock', 'regionidcity',
       'regionidcounty', 'regionidneighborhood', 'regionidzip', 'roomcn
t',
       'storytypeid', 'threequarterbathnbr', 'typeconstructiontypeid',
       'unitcnt', 'yardbuildingsqft17', 'yardbuildingsqft26', 'yearbuil
t',
       'numberofstories', 'fireplaceflag', 'structuretaxvaluedollarcnt',
       'taxvaluedollarcnt', 'assessmentyear', 'landtaxvaluedollarcnt',
       'taxamount', 'taxdelinquencyflag', 'taxdelinquencyyear',
       'censustractandblock'],
      dtype='object')
```

In [6]:

```
print(len(transactions_2016.columns))
print(transactions_2016.columns)
```

```
3
Index(['parcelid', 'logerror', 'transactiondate'], dtype='object')
```

In [7]:

```
print(len(transactions_2017.columns))
print(transactions_2017.columns)
```

```
3
Index(['parcelid', 'logerror', 'transactiondate'], dtype='object')
```

In [8]:

```
data_dictionary.head()
```

Out[8]:

	Feature	Description
0	'airconditioningtypeid'	Type of cooling system present in the home (i...
1	'architecturalstyletypeid'	Architectural style of the home (i.e. ranch, ...
2	'basementsqft'	Finished living area below or partially below...
3	'bathroomcnt'	Number of bathrooms in home including fractio...
4	'bedroomcnt'	Number of bedrooms in home

Milestone 2 : Cleaning/formatting flat file sources

We will first combine the properties_2016 and properties_2017 and calle the result properties. We will also combine the two transactions datasets.

In [9]:

```
properties = pd.concat([properties_2016,properties_2017],axis=0)
print(properties_2016.shape)
print(properties_2017.shape)
print(properties.shape)
```

```
(20000, 59)
(20000, 59)
(40000, 59)
```

In [10]:

```
transactions = pd.concat([transactions_2016,transactions_2017],axis=0)
print(properties_2016.shape)
print(properties_2017.shape)
print(properties.shape)
```

```
(20000, 59)
(20000, 59)
(40000, 59)
```

```
In [11]: properties.columns
```

```
Out[11]: Index(['Unnamed: 0', 'parcelid', 'airconditioningtypeid',
                'architecturalstyletypeid', 'basementsqft', 'bathroomcnt', 'bedroo
         mcnt',
                'buildingclasstypeid', 'buildingqualitytypeid', 'calculatedbathnb
         r',
                'decktypeid', 'finishedfloor1squarefeet',
                'calculatedfinishedsquarefeet', 'finishedsquarefeet12',
                'finishedsquarefeet13', 'finishedsquarefeet15', 'finishedsquarefee
         t50',
                'finishedsquarefeet6', 'fips', 'fireplacecnt', 'fullbathcnt',
                'garagecarcnt', 'garagetotalsqft', 'hashottuborspa',
                'heatingorsystemtypeid', 'latitude', 'longitude', 'lotsizesquarefe
         et',
                'poolcnt', 'poolsizesum', 'pooltypeid10', 'pooltypeid2', 'pooltype
         id7',
                'propertycountylandusecode', 'propertylandusetypeid',
                'propertyzoningdesc', 'rawcensustractandblock', 'regionidcity',
                'regionidcounty', 'regionidneighborhood', 'regionidzip', 'roomcn
         t',
                'storytypeid', 'threequarterbathnbr', 'typeconstructiontypeid',
                'unitcnt', 'yardbuildingsqft17', 'yardbuildingsqft26', 'yearbuil
         t',
                'numberofstories', 'fireplaceflag', 'structuretaxvaluedollarcnt',
                'taxvaluedollarcnt', 'assessmentyear', 'landtaxvaluedollarcnt',
                'taxamount', 'taxdelinquencyflag', 'taxdelinquencyyear',
                'censustractandblock'],
               dtype='object')
```

Get rid of the Unamed column.

In [12]:
```
properties = properties.loc[:, ~properties.columns.str.contains('^Unname
d')]
properties.columns
```

Out[12]:
```
Index(['parcelid', 'airconditioningtypeid', 'architecturalstyletypeid',
       'basementsqft', 'bathroomcnt', 'bedroomcnt', 'buildingclasstypei
d',
       'buildingqualitytypeid', 'calculatedbathnbr', 'decktypeid',
       'finishedfloor1squarefeet', 'calculatedfinishedsquarefeet',
       'finishedsquarefeet12', 'finishedsquarefeet13', 'finishedsquarefee
t15',
       'finishedsquarefeet50', 'finishedsquarefeet6', 'fips', 'fireplacec
nt',
       'fullbathcnt', 'garagecarcnt', 'garagetotalsqft', 'hashottuborsp
a',
       'heatingorsystemtypeid', 'latitude', 'longitude', 'lotsizesquarefe
et',
       'poolcnt', 'poolsizesum', 'pooltypeid10', 'pooltypeid2', 'pooltype
id7',
       'propertycountylandusecode', 'propertylandusetypeid',
       'propertyzoningdesc', 'rawcensustractandblock', 'regionidcity',
       'regionidcounty', 'regionidneighborhood', 'regionidzip', 'roomcn
t',
       'storytypeid', 'threequarterbathnbr', 'typeconstructiontypeid',
       'unitcnt', 'yardbuildingsqft17', 'yardbuildingsqft26', 'yearbuil
t',
       'numberofstories', 'fireplaceflag', 'structuretaxvaluedollarcnt',
       'taxvaluedollarcnt', 'assessmentyear', 'landtaxvaluedollarcnt',
       'taxamount', 'taxdelinquencyflag', 'taxdelinquencyyear',
       'censustractandblock'],
      dtype='object')
```

Rename column names in properties dataset.

In [13]:

```
properties = properties.rename(columns=
                        {
  'parcelid':'parcelid',
  'yearbuilt':'build_year',
  'basementsqft':'area_basement',
  'yardbuildingsqft17':'area_patio',
  'yardbuildingsqft26':'area_shed',
  'poolsizesum':'area_pool',
  'lotsizesquarefeet':'area_lot',
  'garagetotalsqft':'area_garage',
  'finishedfloor1squarefeet':'area_firstfloor_finished',
  'calculatedfinishedsquarefeet':'area_total_calc',
  'finishedsquarefeet6':'area_base',
  'finishedsquarefeet12':'area_live_finished',
  'finishedsquarefeet13':'area_liveperi_finished',
  'finishedsquarefeet15':'area_total_finished',
  'finishedsquarefeet50':'area_unknown',
  'unitcnt': 'num_unit',
  'numberofstories': 'num_story',
  'roomcnt':'num_room',
  'bathroomcnt':'num_bathroom',
  'bedroomcnt':'num_bedroom',
  'calculatedbathnbr':'num_bathroom_calc',
  'fullbathcnt':'num_bath',
  'threequarterbathnbr':'num_75_bath',
  'fireplacecnt':'num_fireplace',
  'poolcnt': 'num_pool',
  'garagecarcnt':'num_garage',
  'regionidcounty':'region_county',
  'regionidcity':'region_city',
  'regionidzip':'region_zip',
  'regionidneighborhood':'region_neighbor',
  'taxvaluedollarcnt':'tax_total',
  'structuretaxvaluedollarcnt':'tax_building',
  'landtaxvaluedollarcnt':'tax_land',
  'taxamount':'tax_property',
  'assessmentyear':'tax_year',
  'taxdelinquencyflag':'tax_delinquency',
  'taxdelinquencyyear':'tax_delinquency_year',
  'propertyzoningdesc':'zoning_property',
  'propertylandusetypeid':'zoning_landuse',
  'propertycountylandusecode':'zoning_landuse_county',
  'fireplaceflag':'flag_fireplace',
  'hashottuborspa':'flag_tub',
  'buildingqualitytypeid':'quality',
  'buildingclasstypeid':'framing',
  'typeconstructiontypeid':'material',
  'decktypeid':'deck',
  'storytypeid':'story',
  'heatingorsystemtypeid':'heating',
  'airconditioningtypeid':'aircon',
  'architecturalstyletypeid':'architectural_style'
})
```

In [14]:

```
properties.columns
```

Out[14]:

```
Index(['parcelid', 'aircon', 'architectural_style', 'area_basement',
       'num_bathroom', 'num_bedroom', 'framing', 'quality',
       'num_bathroom_calc', 'deck', 'area_firstfloor_finished',
       'area_total_calc', 'area_live_finished', 'area_liveperi_finished',
       'area_total_finished', 'area_unknown', 'area_base', 'fips',
       'num_fireplace', 'num_bath', 'num_garage', 'area_garage', 'flag_tu
b',
       'heating', 'latitude', 'longitude', 'area_lot', 'num_pool', 'area_
pool',
       'pooltypeid10', 'pooltypeid2', 'pooltypeid7', 'zoning_landuse_coun
ty',
       'zoning_landuse', 'zoning_property', 'rawcensustractandblock',
       'region_city', 'region_county', 'region_neighbor', 'region_zip',
       'num_room', 'story', 'num_75_bath', 'material', 'num_unit',
       'area_patio', 'area_shed', 'build_year', 'num_story', 'flag_firepl
ace',
       'tax_building', 'tax_total', 'tax_year', 'tax_land', 'tax_propert
y',
       'tax_delinquency', 'tax_delinquency_year', 'censustractandblock'],
      dtype='object')
```

In [15]:

```
# Check new column names
properties[['num_bedroom','num_bathroom']]
```

Out[15]:

	num_bedroom	num_bathroom
0	0.0	0.0
1	0.0	0.0
2	0.0	0.0
3	0.0	0.0
4	0.0	0.0
...	...	...
19995	2.0	1.0
19996	5.0	3.0
19997	8.0	5.0
19998	4.0	2.0
19999	2.0	1.0

40000 rows × 2 columns

Rename column names in transactions dataset.

In [16]:

```
transactions = transactions.rename(columns={'parcelid':'parcelid','date':
'transactiondate'})
```

In [17]:

```
transactions.columns
```

Out[17]:

```
Index(['parcelid', 'logerror', 'transactiondate'], dtype='object')
```

Check out the new columns

In [18]:

```
transactions[['parcelid','transactiondate']]
```

Out[18]:

	parcelid	transactiondate
0	11016594	2016-01-01
1	14366692	2016-01-01
2	12098116	2016-01-01
3	12643413	2016-01-02
4	14432541	2016-01-02
...	...	...
77608	10833991	2017-09-20
77609	11000655	2017-09-20
77610	17239384	2017-09-21
77611	12773139	2017-09-21
77612	12826780	2017-09-25

167888 rows × 2 columns

In [31]:

```
propertiesAndTransactions = pd.merge(properties,transactions,on='parceli
d')
```

check out the merge

In [32]:

```
propertiesAndTransactions[['parcelid','num_bedroom','transactiondate','lo
gerror']].head()
```

Out[32]:

	parcelid	num_bedroom	transactiondate	logerror
0	17054981	4.0	2017-06-15	-0.013099
1	17054981	4.0	2017-06-15	-0.013099
2	17055743	3.0	2017-07-26	0.073985
3	17055743	3.0	2017-07-26	0.073985
4	17068109	3.0	2017-07-28	0.071886

let's take of missings

In [33]:

```
column_names = propertiesAndTransactions.columns
print('sum\n', propertiesAndTransactions.isnull()[column_names].sum())
```

```
sum
 parcelid                       0
aircon                       1485
architectural_style          2234
area_basement                2234
num_bathroom                    0
num_bedroom                     0
framing                      2234
quality                       705
num_bathroom_calc              26
deck                         2214
area_firstfloor_finished     2000
area_total_calc                 9
area_live_finished            102
area_liveperi_finished       2234
area_total_finished          2145
area_unknown                 2000
area_base                    2230
fips                            0
num_fireplace                1982
num_bath                       26
num_garage                   1593
area_garage                  1593
flag_tub                     2192
heating                       752
latitude                        0
longitude                       0
area_lot                      216
num_pool                     1708
area_pool                    2206
pooltypeid10                 2216
pooltypeid2                  2210
pooltypeid7                  1732
zoning_landuse_county           0
zoning_landuse                  0
zoning_property               678
rawcensustractandblock          0
region_city                    42
region_county                   0
region_neighbor              1186
region_zip                      2
num_room                        0
story                        2234
num_75_bath                  1984
material                     2234
num_unit                      679
area_patio                   2137
area_shed                    2234
build_year                     11
num_story                    1792
flag_fireplace               2234
tax_building                    6
tax_total                       0
tax_year                        0
tax_land                        0
tax_property                    0
tax_delinquency              2166
```

```
tax_delinquency_year        2166
censustractandblock            8
logerror                       0
transactiondate                0
dtype: int64
```

In [22]:

```
print('mean\n', propertiesAndTransactions.isnull()[column_names].mean())
```

```
mean
 parcelid                    0.000000
aircon                      0.664727
architectural_style         1.000000
area_basement               1.000000
num_bathroom                0.000000
num_bedroom                 0.000000
framing                     1.000000
quality                     0.315577
num_bathroom_calc           0.011638
deck                        0.991047
area_firstfloor_finished    0.895255
area_total_calc             0.004029
area_live_finished          0.045658
area_liveperi_finished      1.000000
area_total_finished         0.960161
area_unknown                0.895255
area_base                   0.998209
fips                        0.000000
num_fireplace               0.887198
num_bath                    0.011638
num_garage                  0.713071
area_garage                 0.713071
flag_tub                    0.981200
heating                     0.336616
latitude                    0.000000
longitude                   0.000000
area_lot                    0.096688
num_pool                    0.764548
area_pool                   0.987466
pooltypeid10                0.991943
pooltypeid2                 0.989257
pooltypeid7                 0.775291
zoning_landuse_county       0.000000
zoning_landuse              0.000000
zoning_property             0.303491
rawcensustractandblock      0.000000
region_city                 0.018800
region_county               0.000000
region_neighbor             0.530886
region_zip                  0.000895
num_room                    0.000000
story                       1.000000
num_75_bath                 0.888093
material                    1.000000
num_unit                    0.303939
area_patio                  0.956580
area_shed                   1.000000
build_year                  0.004924
num_story                   0.802149
flag_fireplace              1.000000
tax_building                0.002686
tax_total                   0.000000
tax_year                    0.000000
tax_land                    0.000000
tax_property                0.000000
tax_delinquency             0.969561
```

```
tax_delinquency_year        0.969561
censustractandblock         0.003581
logerror                    0.000000
transactiondate             0.000000
dtype: float64
```

Let's look at columns woth more than 80% missing values

In [34]:

```
propertiesAndTransactions.isnull()[column_names].sum()
# this shows columns and the number of NaN's.Note parcelID has no missing
values.
```

```
Out[34]: parcelid                       0
         aircon                      1485
         architectural_style         2234
         area_basement               2234
         num_bathroom                   0
         num_bedroom                    0
         framing                     2234
         quality                      705
         num_bathroom_calc             26
         deck                        2214
         area_firstfloor_finished    2000
         area_total_calc                9
         area_live_finished           102
         area_liveperi_finished      2234
         area_total_finished         2145
         area_unknown                2000
         area_base                   2230
         fips                           0
         num_fireplace               1982
         num_bath                      26
         num_garage                  1593
         area_garage                 1593
         flag_tub                    2192
         heating                      752
         latitude                       0
         longitude                      0
         area_lot                     216
         num_pool                    1708
         area_pool                   2206
         pooltypeid10                2216
         pooltypeid2                 2210
         pooltypeid7                 1732
         zoning_landuse_county          0
         zoning_landuse                 0
         zoning_property              678
         rawcensustractandblock         0
         region_city                   42
         region_county                  0
         region_neighbor             1186
         region_zip                     2
         num_room                       0
         story                       2234
         num_75_bath                 1984
         material                    2234
         num_unit                     679
         area_patio                  2137
         area_shed                   2234
         build_year                    11
         num_story                   1792
         flag_fireplace              2234
         tax_building                   6
         tax_total                      0
         tax_year                       0
         tax_land                       0
         tax_property                   0
         tax_delinquency             2166
         tax_delinquency_year        2166
```

```
censustractandblock            8
logerror                       0
transactiondate                0
dtype: int64
```

Make a list of columns with moe than 80% missing data

In [35]:
```python
remove_columns = propertiesAndTransactions.columns[propertiesAndTransactions.isnull().mean() > .8]
print(remove_columns)
```

```
Index(['architectural_style', 'area_basement', 'framing', 'deck',
       'area_firstfloor_finished', 'area_liveperi_finished',
       'area_total_finished', 'area_unknown', 'area_base', 'num_fireplace',
       'flag_tub', 'area_pool', 'pooltypeid10', 'pooltypeid2', 'story',
       'num_75_bath', 'material', 'area_patio', 'area_shed', 'num_story',
       'flag_fireplace', 'tax_delinquency', 'tax_delinquency_year'],
      dtype='object')
```

Drop the columns

In [36]:
```python
propertiesAndTransactions = propertiesAndTransactions.drop(columns = remove_columns)
```

Check results

In [37]:
```python
print(len(propertiesAndTransactions.columns))
print(propertiesAndTransactions.columns)
```

```
37
Index(['parcelid', 'aircon', 'num_bathroom', 'num_bedroom', 'quality',
       'num_bathroom_calc', 'area_total_calc', 'area_live_finished', 'fips',
       'num_bath', 'num_garage', 'area_garage', 'heating', 'latitude',
       'longitude', 'area_lot', 'num_pool', 'pooltypeid7',
       'zoning_landuse_county', 'zoning_landuse', 'zoning_property',
       'rawcensustractandblock', 'region_city', 'region_county',
       'region_neighbor', 'region_zip', 'num_room', 'num_unit', 'build_year',
       'tax_building', 'tax_total', 'tax_year', 'tax_land', 'tax_property',
       'censustractandblock', 'logerror', 'transactiondate'],
      dtype='object')
```

Check results

In [27]:
```
print(len(propertiesAndTransactions.columns))
print(propertiesAndTransactions.columns)
```

```
37
Index(['parcelid', 'aircon', 'num_bathroom', 'num_bedroom', 'quality',
       'num_bathroom_calc', 'area_total_calc', 'area_live_finished', 'fip
s',
       'num_bath', 'num_garage', 'area_garage', 'heating', 'latitude',
       'longitude', 'area_lot', 'num_pool', 'pooltypeid7',
       'zoning_landuse_county', 'zoning_landuse', 'zoning_property',
       'rawcensustractandblock', 'region_city', 'region_county',
       'region_neighbor', 'region_zip', 'num_room', 'num_unit', 'build_ye
ar',
       'tax_building', 'tax_total', 'tax_year', 'tax_land', 'tax_propert
y',
       'censustractandblock', 'logerror', 'transactiondate'],
      dtype='object')
```

Let's check the missing values mean

In [38]:
```
print('mean\n', propertiesAndTransactions.isnull()[propertiesAndTransactions.columns].mean())
# we see the means to all be below 80%.
```

```
mean
 parcelid                   0.000000
aircon                    0.664727
num_bathroom              0.000000
num_bedroom               0.000000
quality                   0.315577
num_bathroom_calc         0.011638
area_total_calc           0.004029
area_live_finished        0.045658
fips                      0.000000
num_bath                  0.011638
num_garage                0.713071
area_garage               0.713071
heating                   0.336616
latitude                  0.000000
longitude                 0.000000
area_lot                  0.096688
num_pool                  0.764548
pooltypeid7               0.775291
zoning_landuse_county     0.000000
zoning_landuse            0.000000
zoning_property           0.303491
rawcensustractandblock    0.000000
region_city               0.018800
region_county             0.000000
region_neighbor           0.530886
region_zip                0.000895
num_room                  0.000000
num_unit                  0.303939
build_year                0.004924
tax_building              0.002686
tax_total                 0.000000
tax_year                  0.000000
tax_land                  0.000000
tax_property              0.000000
censustractandblock       0.003581
logerror                  0.000000
transactiondate           0.000000
dtype: float64
```

Are there any duplicate?

In [39]:

```
propertiesAndTransactions[propertiesAndTransactions.duplicated(keep=False
)]
# There are no duplocate rows; however, there are duplicate parcelIDs and
corresponding latitude and Longitude.
```

Out[39]:

parcelid	aircon	num_bathroom	num_bedroom	quality	num_bathroom_calc	are

0 rows × 37 columns

In [257]:

```
propertiesAndTransactions
```

Out[257]:

	parcelid	aircon	num_bathroom	num_bedroom	quality	num_bathroom_cal
0	17054981	NaN	5.0	4.0	NaN	5.
1	17054981	NaN	5.0	4.0	NaN	5.
2	17055743	NaN	2.0	3.0	NaN	2.
3	17055743	NaN	2.0	3.0	NaN	2.
4	17068109	NaN	1.5	3.0	NaN	1.
...	...	...	...	...	...	..
2229	11769554	NaN	3.0	4.0	4.0	3.
2230	11778756	NaN	2.0	7.0	7.0	2.
2231	11778756	NaN	2.0	7.0	4.0	2.
2232	11779780	1.0	2.0	2.0	10.0	2.
2233	11779780	1.0	2.0	2.0	11.0	2.

2234 rows × 37 columns

The two datasets have been merged, columns with more than 80% missing values were removed. The final dataset 'propertiesAndTransactions' will be used in the next milestone.

Milestone 3. Webscaraping Data Source

Description

Using webscraping techniques, we will use 'latitude', 'longitude' from properties dataset to access properties and get current data for those locations. The property description of homes in given region will be stored into a dataset with as many features as in properties dataset we can grab. This dataset can then be used to do some price comparision between properties in 2016 and 2017. Getting data from years prior(say 10 years), we will be able to create trend charts and see market fluctuations.

In [388]:

```
# Build a table consisiting of the parcelID, latitude and longitude of th
e properties.
# This table will be used to get data from www.trulia.com by web scraping

LonLat = pd.DataFrame(propertiesAndTransactions[['parcelid','latitude','l
ongitude']])
LonLat
```

Out[388]:

```
      parcelid  latitude  longitude
   0  17054981  34449407 -119254052
   1  17054981  34449407 -119254052
   2  17055743  34454169 -119237898
   3  17055743  34454169 -119237898
   4  17068109  34365693 -119448392
 ...       ...       ...        ...
2229  11769554  34006415 -118246669
2230  11778756  34050678 -118282732
2231  11778756  34050678 -118282732
2232  11779780  34045100 -118261000
2233  11779780  34045100 -118261000

2234 rows × 3 columns
```

In [389]:

```
# We will remove duplicate parcelIDs here since we are only interested in comparable values near each parcelID.
LonLat = LonLat.sort_values('parcelid', ascending=False)
LonLat = LonLat.drop_duplicates()
LonLat.reset_index(drop=True)
LonLat
```

Out[389]:

	parcelid	latitude	longitude
1761	17299670	34186100	-118767000
107	17296734	34174051	-118757031
1758	17294231	34153879	-118839561
1756	17293716	34152179	-118851454
1427	17292856	34125457	-118891074
...	...	...	...
112	10726315	34184300	-118657000
110	10725532	34196000	-118658000
1767	10722858	34195746	-118624097
108	10722336	34199100	-118633000
1763	10719731	34206094	-118620655

1096 rows × 3 columns

In [390]:

```
print('sum\n', LonLat.isnull()[['parcelid','latitude','longitude']].sum())
```

```
sum
 parcelid     0
latitude     0
longitude    0
dtype: int64
```

In [391]:

```
# This dictionary is used to return state code. trulia requires the state
code rather than state name
us_state_abbrev = {
    'Alabama': 'AL',
    'Alaska': 'AK',
    'American Samoa': 'AS',
    'Arizona': 'AZ',
    'Arkansas': 'AR',
    'California': 'CA',
    'Colorado': 'CO',
    'Connecticut': 'CT',
    'Delaware': 'DE',
    'District of Columbia': 'DC',
    'Florida': 'FL',
    'Georgia': 'GA',
    'Guam': 'GU',
    'Hawaii': 'HI',
    'Idaho': 'ID',
    'Illinois': 'IL',
    'Indiana': 'IN',
    'Iowa': 'IA',
    'Kansas': 'KS',
    'Kentucky': 'KY',
    'Louisiana': 'LA',
    'Maine': 'ME',
    'Maryland': 'MD',
    'Massachusetts': 'MA',
    'Michigan': 'MI',
    'Minnesota': 'MN',
    'Mississippi': 'MS',
    'Missouri': 'MO',
    'Montana': 'MT',
    'Nebraska': 'NE',
    'Nevada': 'NV',
    'New Hampshire': 'NH',
    'New Jersey': 'NJ',
    'New Mexico': 'NM',
    'New York': 'NY',
    'North Carolina': 'NC',
    'North Dakota': 'ND',
    'Northern Mariana Islands':'MP',
    'Ohio': 'OH',
    'Oklahoma': 'OK',
    'Oregon': 'OR',
    'Pennsylvania': 'PA',
    'Puerto Rico': 'PR',
    'Rhode Island': 'RI',
    'South Carolina': 'SC',
    'South Dakota': 'SD',
    'Tennessee': 'TN',
    'Texas': 'TX',
    'Utah': 'UT',
    'Vermont': 'VT',
    'Virgin Islands': 'VI',
    'Virginia': 'VA',
    'Washington': 'WA',
    'West Virginia': 'WV',
    'Wisconsin': 'WI',
    'Wyoming': 'WY'
}

abbrev_us_state = dict(map(reversed, us_state_abbrev.items()))
```

In [392]:

```
import urllib.request
import urllib.parse
import urllib.error
import json
from bs4 import BeautifulSoup
from urllib.request import Request, urlopen
import geopy
from geopy.geocoders import Nominatim

def create_url(city,state,zipcode):
    # Creating trulia URL based on the filter.

    url = "https://www.trulia.com/" + state + "/" + city + "/" + zipcode
    return url

def get_response(url):
    ret = None
    try:
        for i in range(5):
            response = requests.get(url, headers={'User-Agent': 'Mozilla/
5.0'})
            print("status code received:", response.status_code)
            if (response.status_code != 200):
                return None
            else:
                return response
    except:
        print('exception in get_response')
        return None

def GetCityStateZip(lat,lon):
    lat = lat/10**6
    lon = lon/10**6
    geolocator = Nominatim(timeout=5)
    #print(location.raw)
    try:
        location = geolocator.reverse((lat, lon))
        city = location.raw['address']['city']
        state = us_state_abbrev[location.raw['address']['state']]
        zipcode = location.raw['address']['postcode'].split('-')[0]
    except:
        city = ""
        state = ""
        zipcode = ""

    return city,state,zipcode
```

In [393]:

```python
def GetComp(parcelId,latitude,longitude):
    city,state,zipcode = GetCityStateZip(latitude,longitude)
    #print(parcelId,latitude,longitude)
    #print("city=", city)
    #print("state=", state)
    #print("zipcode=",zipcode)

    emptylistings_json = {}
    emptylistings_json['parcelId'] = {0:parcelId}
    emptylistings_json['price'] = {0:np.nan}
    emptylistings_json['bedrooms'] = {0:np.nan}
    emptylistings_json['bathrooms'] = {0:np.nan}
    emptylistings_json['floorSpace'] = {0:np.nan}
    emptylistings_json['region'] = {0:np.nan}

    if (city == "" or state == "" or state == ""):
        return(pd.DataFrame(emptylistings_json))

    url = create_url(city,state,zipcode)

    #req = Requests(url, headers={'User-Agent': 'Mozilla/5.0'})
    #webpage = urlopen(req).read()
    #soup = BeautifulSoup(webpage, 'html.parser')

    response = get_response(url)
    #print(response.text)
    if not response:
        print("Failed to fetch the page, please check `response.html` to see the response received from zillow.com.")
        return(pd.DataFrame(emptylistings_json))

    soup = BeautifulSoup(response.text, 'html.parser')

    html = soup.prettify('utf-8')

    details = {}
    parcels = {}
    listings_json = {}
    index = 0

    for price in  soup.findAll('div',attrs={'data-testid': 'property-price'}):
        details.update({index:price.text.strip()})
        parcels.update({index:parcelId})
        index = index + 1

    listings_json['parcelId'] = {}
    listings_json['parcelId']  = parcels
    listings_json['price'] = {}
    listings_json['price']  = details
    #print(listings_json['price'])



    details = {}
    index = 0
    for bedroom  in  soup.findAll('div',attrs={'data-testid': 'property-beds'}):
        details.update({index:bedroom.text.strip()})
        index = index + 1

    listings_json['bedrooms'] = {}
    listings_json['bedrooms']  = details
    #print(listings_json)



    details = {}
    index = 0
    for bathroom  in  soup.findAll('div',attrs={'data-testid': 'property-baths'}):
        details.update({index:bathroom.text.strip()})
        index = index + 1

    listings_json['bathrooms'] = {}
    listings_json['bathrooms']  = details
    #print(listings_json)



    details = {}
    index = 0
    for floorSpace  in  soup.findAll('div',attrs={'data-testid': 'property-floorSpace'}):
        details.update({index:floorSpace.text.strip()})
        index = index + 1

    listings_json['floorSpace'] = {}
    listings_json['floorSpace']  = details
    #print(listings_json)



    details = {}
    index = 0
    for region  in  soup.findAll('div',attrs={'data-testid': 'property-region'}):
        details.update({index:region.text.strip()})
        index = index + 1

    listings_json['region'] = {}
    listings_json['region']  = details
    #print(listings_json)

    #listings_table = pd.DataFrame()

    #with open('house_details.json', 'w') as outfile:
    #    json.dump(listings_json, outfile, indent=4)
    #listings_table = pd.read_json("house_details.json")
    return pd.DataFrame(listings_json)
```

In [394]:
```
LonLat[:5]
```

Out[394]:

	parcelid	latitude	longitude
1761	17299670	34186100	-118767000
107	17296734	34174051	-118757031
1758	17294231	34153879	-118839561
1756	17293716	34152179	-118851454
1427	17292856	34125457	-118891074

Here we get 20 compare properties for the parcelIDs. Note that a parcelID from propertiesAndTransactions table may have one ore more comps near it's latitude and longitude. This process sometime times out. We have taken care to continue collecting even after such exceptions.

In [395]:

```
comp_listing_table = pd.DataFrame(columns={'parcelid','price','bedrooms',
'bathrooms','floorSpace','region'})

dfs = []
for index, row in LonLat[:20].iterrows():
    parcelId = row['parcelid']
    latitude = row['latitude']
    longitude = row['longitude']
    #print(parcelId,latitude,longitude)
    Temp_listing_table = GetComp(parcelId,latitude,longitude)
    #print(Temp_listing_table.shape)
    dfs.append(Temp_listing_table)
    #print(Temp_listing_table)


comp_listing_table = pd.concat(dfs, ignore_index=True)
print(comp_listing_table.shape)
```

```
c:\users\safar\documents\github\safarie1103\bellevue university\courses\d
sc540\venv\lib\site-packages\ipykernel_launcher.py:33: DeprecationWarnin
g: Using Nominatim with the default "geopy/1.21.0" `user_agent` is strong
ly discouraged, as it violates Nominatim's ToS https://operations.osmfoun
dation.org/policies/nominatim/ and may possibly cause 403 and 429 HTTP er
rors. Please specify a custom `user_agent` with `Nominatim(user_agent="my
-application")` or by overriding the default `user_agent`: `geopy.geocode
rs.options.default_user_agent = "my-application"`. In geopy 2.0 this will
become an exception.
```

```
status code received: 200
status code received: 200
status code received: 200
status code received: 200
status code received: 200
status code received: 200
status code received: 200
status code received: 200
status code received: 200
status code received: 200
status code received: 200
status code received: 200
status code received: 200
status code received: 200
status code received: 200
status code received: 200
(480, 6)
```

In [396]:
```
print(comp_listing_table)
```

```
     parcelId        price bedrooms bathrooms   floorSpace  \
0    17299670          NaN      NaN       NaN          NaN   
1    17296734          NaN      NaN       NaN          NaN   
2    17294231  $14,999,000      7bd      13ba  14,073 sqft   
3    17294231   $1,450,000      4bd       3ba   2,568 sqft   
4    17294231   $1,225,000      4bd       3ba   2,745 sqft   
..        ...          ...      ...       ...          ...   
475  17273670     $897,000      4bd       3ba   3,259 sqft   
476  17273670     $680,000      4bd       3ba   2,096 sqft   
477  17273670     $569,000      3bd       3ba   1,550 sqft   
478  17273670     $830,000      3bd       3ba   2,243 sqft   
479  17273670     $999,900      5bd       4ba   3,780 sqft   

                                region  
0                                  NaN  
1                                  NaN  
2    Newbury Park, Thousand Oaks, CA  
3                Westlake Village, CA  
4                Westlake Village, CA  
..                                 ...  
475  Newbury Park, Thousand Oaks, CA  
476  Newbury Park, Thousand Oaks, CA  
477  Newbury Park, Thousand Oaks, CA  
478  Newbury Park, Thousand Oaks, CA  
479  Newbury Park, Thousand Oaks, CA  

[480 rows x 6 columns]
```

In [397]:
```
comp_listing_table.isnull()[comp_listing_table.columns].sum()
```

Out[397]:
```
parcelId       0
price          4
bedrooms      13
bathrooms     13
floorSpace    13
region         4
dtype: int64
```

In [398]:
```
comp_listing_table = comp_listing_table.dropna()
```

In [399]:
```
comp_listing_table.isnull()[comp_listing_table.columns].sum()
```

Out[399]:
```
parcelId      0
price         0
bedrooms      0
bathrooms     0
floorSpace    0
region        0
dtype: int64
```

In [400]:
```
comp_listing_table.shape
```

Out[400]:
```
(467, 6)
```

In [401]:
```
# Write scraped data to a file for safe keeps and also to avoid rescrapin
g during development
comp_listing_table.to_csv("data/comp_listing_table.csv")
```

In [431]:
```
# Read
comp_listing_table = pd.read_csv("data/comp_listing_table.csv")
```

In [432]:
```
comp_listing_table
```

Out[432]:

	Unnamed: 0	parcelId	price	bedrooms	bathrooms	floorSpace	regio
0	2	17294231	$14,999,000	7bd	13ba	14,073 sqft	Newbur Park Thousan Oaks, C
1	3	17294231	$1,450,000	4bd	3ba	2,568 sqft	Westlak Village C
2	4	17294231	$1,225,000	4bd	3ba	2,745 sqft	Westlak Village C
3	5	17294231	$9,990,000	7bd	10ba	12,656 sqft	Newbur Park Thousan Oaks, C
4	6	17294231	$1,150,000	5bd	4ba	2,393 sqft	Westlak Village C
...	...	...	...	...	...	...	..
462	475	17273670	$897,000	4bd	3ba	3,259 sqft	Newbur Park Thousan Oaks, C
463	476	17273670	$680,000	4bd	3ba	2,096 sqft	Newbur Park Thousan Oaks, C
464	477	17273670	$569,000	3bd	3ba	1,550 sqft	Newbur Park Thousan Oaks, C
465	478	17273670	$830,000	3bd	3ba	2,243 sqft	Newbur Park Thousan Oaks, C
466	479	17273670	$999,900	5bd	4ba	3,780 sqft	Newbur Park Thousan Oaks, C

467 rows × 7 columns

prepare the dataset

In [433]:

```
comp_listing_table = comp_listing_table.loc[:, ~comp_listing_table.columns.str.contains('^Unnamed')]
```

In [434]:

```
comp_listing_table['price']= comp_listing_table['price'].replace('[\$,]', '', regex=True).astype(float)
comp_listing_table
```

```
c:\users\safar\documents\github\safarie1103\bellevue university\courses\dsc540\venv\lib\site-packages\ipykernel_launcher.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  """Entry point for launching an IPython kernel.
```

Out[434]:

	parcelId	price	bedrooms	bathrooms	floorSpace	region
0	17294231	14999000.0	7bd	13ba	14,073 sqft	Newbury Park, Thousand Oaks, CA
1	17294231	1450000.0	4bd	3ba	2,568 sqft	Westlake Village, CA
2	17294231	1225000.0	4bd	3ba	2,745 sqft	Westlake Village, CA
3	17294231	9990000.0	7bd	10ba	12,656 sqft	Newbury Park, Thousand Oaks, CA
4	17294231	1150000.0	5bd	4ba	2,393 sqft	Westlake Village, CA
...	...	...	...	...	...	...
462	17273670	897000.0	4bd	3ba	3,259 sqft	Newbury Park, Thousand Oaks, CA
463	17273670	680000.0	4bd	3ba	2,096 sqft	Newbury Park, Thousand Oaks, CA
464	17273670	569000.0	3bd	3ba	1,550 sqft	Newbury Park, Thousand Oaks, CA
465	17273670	830000.0	3bd	3ba	2,243 sqft	Newbury Park, Thousand Oaks, CA
466	17273670	999900.0	5bd	4ba	3,780 sqft	Newbury Park, Thousand Oaks, CA

467 rows × 6 columns

In [435]:
```
comp_listing_table['bedrooms']= comp_listing_table['bedrooms'].replace('b
d', '', regex=True).astype(int)
comp_listing_table
```

```
c:\users\safar\documents\github\safarie1103\bellevue university\courses\d
sc540\venv\lib\site-packages\ipykernel_launcher.py:1: SettingWithCopyWarn
ing:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-do
cs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  """Entry point for launching an IPython kernel.
```

Out[435]:

	parcelId	price	bedrooms	bathrooms	floorSpace	region
0	17294231	14999000.0	7	13ba	14,073 sqft	Newbury Park, Thousand Oaks, CA
1	17294231	1450000.0	4	3ba	2,568 sqft	Westlake Village, CA
2	17294231	1225000.0	4	3ba	2,745 sqft	Westlake Village, CA
3	17294231	9990000.0	7	10ba	12,656 sqft	Newbury Park, Thousand Oaks, CA
4	17294231	1150000.0	5	4ba	2,393 sqft	Westlake Village, CA
...	...	...	...	...	...	...
462	17273670	897000.0	4	3ba	3,259 sqft	Newbury Park, Thousand Oaks, CA
463	17273670	680000.0	4	3ba	2,096 sqft	Newbury Park, Thousand Oaks, CA
464	17273670	569000.0	3	3ba	1,550 sqft	Newbury Park, Thousand Oaks, CA
465	17273670	830000.0	3	3ba	2,243 sqft	Newbury Park, Thousand Oaks, CA
466	17273670	999900.0	5	4ba	3,780 sqft	Newbury Park, Thousand Oaks, CA

467 rows × 6 columns

In [436]:

```
comp_listing_table['bathrooms']= comp_listing_table['bathrooms'].replace(
'ba', '', regex=True).astype(float)
comp_listing_table
```

```
c:\users\safar\documents\github\safarie1103\bellevue university\courses\d
sc540\venv\lib\site-packages\ipykernel_launcher.py:1: SettingWithCopyWarn
ing:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-do
cs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  """Entry point for launching an IPython kernel.
```

Out[436]:

	parcelId	price	bedrooms	bathrooms	floorSpace	region
0	17294231	14999000.0	7	13.0	14,073 sqft	Newbury Park, Thousand Oaks, CA
1	17294231	1450000.0	4	3.0	2,568 sqft	Westlake Village, CA
2	17294231	1225000.0	4	3.0	2,745 sqft	Westlake Village, CA
3	17294231	9990000.0	7	10.0	12,656 sqft	Newbury Park, Thousand Oaks, CA
4	17294231	1150000.0	5	4.0	2,393 sqft	Westlake Village, CA
...	...	...	...	...	...	...
462	17273670	897000.0	4	3.0	3,259 sqft	Newbury Park, Thousand Oaks, CA
463	17273670	680000.0	4	3.0	2,096 sqft	Newbury Park, Thousand Oaks, CA
464	17273670	569000.0	3	3.0	1,550 sqft	Newbury Park, Thousand Oaks, CA
465	17273670	830000.0	3	3.0	2,243 sqft	Newbury Park, Thousand Oaks, CA
466	17273670	999900.0	5	4.0	3,780 sqft	Newbury Park, Thousand Oaks, CA

467 rows × 6 columns

In [437]:

```
comp_listing_table['floorSpace'] = comp_listing_table['floorSpace'].replace('sqft', '', regex=True).replace(',','',regex=True).astype(np.int64)
comp_listing_table.columns
```

```
c:\users\safar\documents\github\safarie1103\bellevue university\courses\dsc540\venv\lib\site-packages\ipykernel_launcher.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  """Entry point for launching an IPython kernel.
```

Out[437]:

```
Index(['parcelId', 'price', 'bedrooms', 'bathrooms', 'floorSpace', 'region'], dtype='object')
```

now that we have our comp table built let's do some comparisons

We'll grab a property from propertiesAndTransactions and query the comp table.

In [ ]:

```
# THis table has duplicates and NaNs removed so it is a subset of the propertiesAndTransactions table.
LonLat
```

In [ ]:

```
propertiesAndTransactions
```

In [ ]:

```
# Notice the duplicates
selected_parcelid = propertiesAndTransactions['parcelid'] == 17294231
propertiesAndTransactions[selected_parcelid]
```

In [ ]:

```
selected_parcelid = comp_listing_table['parcelId'] == 17294231
comp_listing_table[selected_parcelid]
```

In [438]:
```
## data from API
#### Description

Googlemap API and matplotlib or equivalant will be used to locate propert
ies by zipcode and display them on the map of the Unites States. We will
convert 'longitude' and 'latitude' columns in properties dataset to zip c
ode and use the zipcode in the API call.We will show the density of homes
sold in various regions in the dataset. We will also show the properties
we extracted using webscraping techniques.
```

Out[438]:

	parcelid	latitude	longitude
1761	17299670	34186100	-118767000
107	17296734	34174051	-118757031
1758	17294231	34153879	-118839561
1756	17293716	34152179	-118851454
1427	17292856	34125457	-118891074
...	...	...	...
112	10726315	34184300	-118657000
110	10725532	34196000	-118658000
1767	10722858	34195746	-118624097
108	10722336	34199100	-118633000
1763	10719731	34206094	-118620655

1096 rows × 3 columns

In [450]:
```
propertiesAndTransactions
```

Out[450]:

	parcelid	aircon	num_bathroom	num_bedroom	quality	num_bathroom_cal
0	17054981	NaN	5.0	4.0	NaN	5.
1	17054981	NaN	5.0	4.0	NaN	5.
2	17055743	NaN	2.0	3.0	NaN	2.
3	17055743	NaN	2.0	3.0	NaN	2.
4	17068109	NaN	1.5	3.0	NaN	1.
...	...	...	...	...	...	..
2229	11769554	NaN	3.0	4.0	4.0	3.
2230	11778756	NaN	2.0	7.0	7.0	2.
2231	11778756	NaN	2.0	7.0	4.0	2.
2232	11779780	1.0	2.0	2.0	10.0	2.
2233	11779780	1.0	2.0	2.0	11.0	2.

2234 rows × 37 columns

In [451]:

```
# Notice the duplicates
selected_parcelid = propertiesAndTransactions['parcelid'] == 17294231
propertiesAndTransactions[selected_parcelid]
```

Out[451]:

	parcelid	aircon	num_bathroom	num_bedroom	quality	num_bathroom_cal
1758	17294231	NaN	2.0	3.0	NaN	2.
1759	17294231	NaN	2.0	3.0	NaN	2.

2 rows × 37 columns

In [452]:

```
selected_parcelid = comp_listing_table['parcelId'] == 17294231
comp_listing_table[selected_parcelid]
```

Out[452]:

	parcelId	price	bedrooms	bathrooms	floorSpace	region
0	17294231	14999000.0	7	13.0	14073	Newbury Park, Thousand Oaks, CA
1	17294231	1450000.0	4	3.0	2568	Westlake Village, CA
2	17294231	1225000.0	4	3.0	2745	Westlake Village, CA
3	17294231	9990000.0	7	10.0	12656	Newbury Park, Thousand Oaks, CA
4	17294231	1150000.0	5	4.0	2393	Westlake Village, CA
5	17294231	525000.0	2	3.0	1440	Westlake Village, CA
6	17294231	1499000.0	5	5.0	3804	Westlake Village, CA
7	17294231	1099000.0	4	3.0	2300	Westlake Village, CA
8	17294231	919000.0	4	2.0	1838	Westlake Village, CA
9	17294231	3195000.0	3	3.0	2543	Westlake Village, CA
10	17294231	1875000.0	5	5.0	4431	Westlake Village, CA
11	17294231	9900000.0	5	7.0	8095	Lake Sherwood, CA
12	17294231	1250000.0	4	3.0	3012	Westlake Village, CA
13	17294231	1799999.0	4	4.0	2106	Westlake Village, CA
14	17294231	640000.0	2	2.0	1231	Westlake Village, CA
15	17294231	1080000.0	4	2.0	2371	Westlake Village, CA
16	17294231	1289000.0	3	3.0	2222	Lake Sherwood, CA
17	17294231	3450000.0	5	6.0	5954	Thousand Oaks, CA
18	17294231	1049000.0	4	3.0	2538	Westlake Village, CA
19	17294231	5495000.0	7	9.0	9304	Thousand Oaks, CA
20	17294231	2995000.0	5	6.0	5421	Westlake Village, CA
21	17294231	1499000.0	4	3.0	2920	Thousand Oaks, CA
22	17294231	1449000.0	4	4.0	3013	Lake Sherwood, CA
23	17294231	765000.0	2	2.0	1508	Westlake Village, CA
24	17294231	1599000.0	3	3.0	2282	Westlake Village, CA
25	17294231	2399000.0	5	4.0	4724	Westlake Village, CA
26	17294231	2975000.0	4	3.0	4075	Westlake Village, CA
27	17294231	988000.0	4	3.0	2412	Westlake Village, CA
28	17294231	4750000.0	6	6.0	7470	Thousand Oaks, CA
29	17294231	3950000.0	5	5.0	5466	Thousand Oaks, CA

Milestone 4. Data from API

Description

Googlemaps API is used to get additional information for parcelIDs in LonLat table built in Milestone 3. We will get the geometric coordinates for a given parcel, latitude and longitude of that parcel. Googlemaps returns various corrdinates sorrounding the given coordinates such as nw/sw

In [152]:

```
# This is a sample code and does not pertain to this project. We will try
to implement a function s
import googlemaps
from datetime import datetime

with open('../APIkeys/APIkeys.json') as f:
    keys = json.load(f)
    key = keys['googlemaps']['key']

gmaps = googlemaps.Client(key=key)
```

Some testing and exploration of the interface

In [154]:

```
# Geocoding an address
geocode_result = gmaps.geocode('1600 Amphitheatre Parkway, Mountain View,
CA')

print(geocode_result[0]['geometry'])
```

```
{'location': {'lat': 37.4223097, 'lng': -122.0846245}, 'location_type':
'ROOFTOP', 'viewport': {'northeast': {'lat': 37.4236586802915, 'lng': -12
2.0832755197085}, 'southwest': {'lat': 37.4209607197085, 'lng': -122.0859
734802915}}}
```

In [155]:

```
print(geocode_result[0]['geometry']['viewport']['northeast']['lat'])
```

```
37.4236586802915
```

In [156]:

```
# Get a sample
reverse_geocode_result = gmaps.reverse_geocode((40.714224, -73.961452))
```

In [157]:

```
# print result
print(reverse_geocode_result)
```

```
[{'access_points': [], 'address_components': [{'long_name': '279', 'short
_name': '279', 'types': ['street_number']}, {'long_name': 'Bedford Avenu
e', 'short_name': 'Bedford Ave', 'types': ['route']}, {'long_name': 'Will
iamsburg', 'short_name': 'Williamsburg', 'types': ['neighborhood', 'polit
ical']}, {'long_name': 'Brooklyn', 'short_name': 'Brooklyn', 'types': ['p
olitical', 'sublocality', 'sublocality_level_1']}, {'long_name': 'Kings C
ounty', 'short_name': 'Kings County', 'types': ['administrative_area_leve
l_2', 'political']}, {'long_name': 'New York', 'short_name': 'NY', 'type
s': ['administrative_area_level_1', 'political']}, {'long_name': 'United
States', 'short_name': 'US', 'types': ['country', 'political']}, {'long_n
ame': '11211', 'short_name': '11211', 'types': ['postal_code']}], 'format
ted_address': '279 Bedford Ave, Brooklyn, NY 11211, USA', 'geometry': {'l
ocation': {'lat': 40.71423350000001, 'lng': -73.9613686}, 'location_typ
e': 'ROOFTOP', 'viewport': {'northeast': {'lat': 40.71558248029151, 'ln
g': -73.9600196197085}, 'southwest': {'lat': 40.71288451970851, 'lng': -7
3.96271758029151}}}, 'place_id': 'ChIJT2x8Q2BZwokRpBu2jUzX3dE', 'plus_cod
e': {'compound_code': 'P27Q+MF Brooklyn, New York, United States', 'globa
l_code': '87G8P27Q+MF'}, 'types': ['bakery', 'cafe', 'establishment', 'fo
od', 'point_of_interest', 'store']}, {'access_points': [], 'address_compo
nents': [{'long_name': '277', 'short_name': '277', 'types': ['street_numb
er']}, {'long_name': 'Bedford Avenue', 'short_name': 'Bedford Ave', 'type
s': ['route']}, {'long_name': 'Williamsburg', 'short_name': 'Williamsbur
g', 'types': ['neighborhood', 'political']}, {'long_name': 'Brooklyn', 's
hort_name': 'Brooklyn', 'types': ['political', 'sublocality', 'sublocalit
y_level_1']}, {'long_name': 'Kings County', 'short_name': 'Kings County',
'types': ['administrative_area_level_2', 'political']}, {'long_name': 'Ne
w York', 'short_name': 'NY', 'types': ['administrative_area_level_1', 'po
litical']}, {'long_name': 'United States', 'short_name': 'US', 'types':
['country', 'political']}, {'long_name': '11211', 'short_name': '11211',
'types': ['postal_code']}], 'formatted_address': '277 Bedford Ave, Brookl
yn, NY 11211, USA', 'geometry': {'location': {'lat': 40.7142205, 'lng': -
73.9612903}, 'location_type': 'ROOFTOP', 'viewport': {'northeast': {'la
t': 40.71556948029149, 'lng': -73.95994131970849}, 'southwest': {'lat': 4
0.7128715197085, 'lng': -73.9626392802915}}}, 'place_id': 'ChIJd8BlQ2BZwo
kRAFUEcm_qrcA', 'plus_code': {'compound_code': 'P27Q+MF Brooklyn, New Yor
k, United States', 'global_code': '87G8P27Q+MF'}, 'types': ['street_addre
ss']}, {'access_points': [], 'address_components': [{'long_name': '279',
'short_name': '279', 'types': ['street_number']}, {'long_name': 'Bedford
Avenue', 'short_name': 'Bedford Ave', 'types': ['route']}, {'long_name':
'Williamsburg', 'short_name': 'Williamsburg', 'types': ['neighborhood',
'political']}, {'long_name': 'Brooklyn', 'short_name': 'Brooklyn', 'type
s': ['political', 'sublocality', 'sublocality_level_1']}, {'long_name':
'Kings County', 'short_name': 'Kings County', 'types': ['administrative_a
rea_level_2', 'political']}, {'long_name': 'New York', 'short_name': 'N
Y', 'types': ['administrative_area_level_1', 'political']}, {'long_name':
'United States', 'short_name': 'US', 'types': ['country', 'political']},
{'long_name': '11211', 'short_name': '11211', 'types': ['postal_code']},
{'long_name': '4203', 'short_name': '4203', 'types': ['postal_code_suffi
x']}], 'formatted_address': '279 Bedford Ave, Brooklyn, NY 11211, USA',
'geometry': {'bounds': {'northeast': {'lat': 40.7142628, 'lng': -73.96121
31}, 'southwest': {'lat': 40.7141534, 'lng': -73.9613792}}, 'location':
{'lat': 40.7142015, 'lng': -73.96130769999999}, 'location_type': 'ROOFTO
P', 'viewport': {'northeast': {'lat': 40.7155570802915, 'lng': -73.959947
16970849}, 'southwest': {'lat': 40.7128591197085, 'lng': -73.962645130291
49}}}, 'place_id': 'ChIJRYYERGBZwokRAM4n1GlcYX4', 'types': ['premise']},
{'access_points': [], 'address_components': [{'long_name': '279', 'short_
name': '279', 'types': ['street_number']}, {'long_name': 'Bedford Avenu
e', 'short_name': 'Bedford Ave', 'types': ['route']}, {'long_name': 'Will
iamsburg', 'short_name': 'Williamsburg', 'types': ['neighborhood', 'polit
ical']}, {'long_name': 'Brooklyn', 'short_name': 'Brooklyn', 'types': ['p
olitical', 'sublocality', 'sublocality_level_1']}, {'long_name': 'Kings C
ounty', 'short_name': 'Kings County', 'types': ['administrative_area_leve
l_2', 'political']}, {'long_name': 'New York', 'short_name': 'NY', 'type
s': ['administrative_area_level_1', 'political']}, {'long_name': 'United
States', 'short_name': 'US', 'types': ['country', 'political']}, {'long_n
ame': '11211', 'short_name': '11211', 'types': ['postal_code']}], 'format
ted_address': '279 Bedford Ave, Brooklyn, NY 11211, USA', 'geometry': {'l
ocation': {'lat': 40.7142545, 'lng': -73.9614527}, 'location_type': 'RANG
E_INTERPOLATED', 'viewport': {'northeast': {'lat': 40.7156034802915, 'ln
g': -73.96010371970848}, 'southwest': {'lat': 40.7129055197085, 'lng': -7
3.9628016802915}}}, 'place_id': 'EigyNzkgQmVkZm9yZCBBdmUsIEJyb29rbHluLCBO
WSAxMTIxMSwgVVNBIhsSGQoUChIJ8ThWRGBZwokR3E1zUisk3LUQlwI', 'types': ['stre
et_address']}, {'access_points': [], 'address_components': [{'long_name':
'291-275', 'short_name': '291-275', 'types': ['street_number']}, {'long_n
ame': 'Bedford Avenue', 'short_name': 'Bedford Ave', 'types': ['route']},
{'long_name': 'Williamsburg', 'short_name': 'Williamsburg', 'types': ['ne
ighborhood', 'political']}, {'long_name': 'Brooklyn', 'short_name': 'Broo
klyn', 'types': ['political', 'sublocality', 'sublocality_level_1']}, {'l
ong_name': 'Kings County', 'short_name': 'Kings County', 'types': ['admin
istrative_area_level_2', 'political']}, {'long_name': 'New York', 'short_
name': 'NY', 'types': ['administrative_area_level_1', 'political']}, {'lo
ng_name': 'United States', 'short_name': 'US', 'types': ['country', 'poli
tical']}, {'long_name': '11211', 'short_name': '11211', 'types': ['postal
_code']}], 'formatted_address': '291-275 Bedford Ave, Brooklyn, NY 11211,
USA', 'geometry': {'bounds': {'northeast': {'lat': 40.7145065, 'lng': -7
3.9612923}, 'southwest': {'lat': 40.7139055, 'lng': -73.96168349999999}},
'location': {'lat': 40.7142045, 'lng': -73.9614845}, 'location_type': 'GE
OMETRIC_CENTER', 'viewport': {'northeast': {'lat': 40.7155549802915, 'ln
g': -73.96013891970848}, 'southwest': {'lat': 40.7128570197085, 'lng': -7
3.96283688029149}}}, 'place_id': 'ChIJ8ThWRGBZwokR3E1zUisk3LU', 'types':
['route']}, {'access_points': [], 'address_components': [{'long_name': '1
1211', 'short_name': '11211', 'types': ['postal_code']}, {'long_name': 'B
rooklyn', 'short_name': 'Brooklyn', 'types': ['political', 'sublocality',
'sublocality_level_1']}, {'long_name': 'New York', 'short_name': 'New Yor
k', 'types': ['locality', 'political']}, {'long_name': 'New York', 'short
_name': 'NY', 'types': ['administrative_area_level_1', 'political']}, {'l
ong_name': 'United States', 'short_name': 'US', 'types': ['country', 'pol
itical']}], 'formatted_address': 'Brooklyn, NY 11211, USA', 'geometry':
{'bounds': {'northeast': {'lat': 40.7280089, 'lng': -73.9207299}, 'southw
est': {'lat': 40.7008331, 'lng': -73.9644697}}, 'location': {'lat': 40.70
93358, 'lng': -73.9565551}, 'location_type': 'APPROXIMATE', 'viewport':
{'northeast': {'lat': 40.7280089, 'lng': -73.9207299}, 'southwest': {'la
t': 40.7008331, 'lng': -73.9644697}}}, 'place_id': 'ChIJvbEjlVdZwokR4KapM
3WCFRw', 'types': ['postal_code']}, {'access_points': [], 'address_compon
ents': [{'long_name': 'Williamsburg', 'short_name': 'Williamsburg', 'type
s': ['neighborhood', 'political']}, {'long_name': 'Brooklyn', 'short_nam
e': 'Brooklyn', 'types': ['political', 'sublocality', 'sublocality_level_
1']}, {'long_name': 'Kings County', 'short_name': 'Kings County', 'type
s': ['administrative_area_level_2', 'political']}, {'long_name': 'New Yor
k', 'short_name': 'NY', 'types': ['administrative_area_level_1', 'politic
al']}, {'long_name': 'United States', 'short_name': 'US', 'types': ['coun
try', 'political']}], 'formatted_address': 'Williamsburg, Brooklyn, NY, U
SA', 'geometry': {'bounds': {'northeast': {'lat': 40.7251773, 'lng': -73.
936498}, 'southwest': {'lat': 40.6979329, 'lng': -73.96984499999999}}, 'l
```

```
ocation': {'lat': 40.7081156, 'lng': -73.9570696}, 'location_type': 'APPR
OXIMATE', 'viewport': {'northeast': {'lat': 40.7251773, 'lng': -73.93649
8}, 'southwest': {'lat': 40.6979329, 'lng': -73.96984499999999}}}, 'place
_id': 'ChIJQSrBBv1bwokRbNfFHCnyeYI', 'types': ['neighborhood', 'politica
l']}, {'access_points': [], 'address_components': [{'long_name': 'Brookly
n', 'short_name': 'Brooklyn', 'types': ['political', 'sublocality', 'subl
ocality_level_1']}, {'long_name': 'Kings County', 'short_name': 'Kings Co
unty', 'types': ['administrative_area_level_2', 'political']}, {'long_nam
e': 'New York', 'short_name': 'NY', 'types': ['administrative_area_level_
1', 'political']}, {'long_name': 'United States', 'short_name': 'US', 'ty
pes': ['country', 'political']}], 'formatted_address': 'Brooklyn, NY, US
A', 'geometry': {'bounds': {'northeast': {'lat': 40.739446, 'lng': -73.83
33651}, 'southwest': {'lat': 40.551042, 'lng': -74.05663}}, 'location':
{'lat': 40.6781784, 'lng': -73.9441579}, 'location_type': 'APPROXIMATE',
'viewport': {'northeast': {'lat': 40.739446, 'lng': -73.8333651}, 'southw
est': {'lat': 40.551042, 'lng': -74.05663}}}, 'place_id': 'ChIJCSF8lBZEwo
kRhngABHRcdoI', 'types': ['political', 'sublocality', 'sublocality_level_
1']}, {'access_points': [], 'address_components': [{'long_name': 'Kings C
ounty', 'short_name': 'Kings County', 'types': ['administrative_area_leve
l_2', 'political']}, {'long_name': 'Brooklyn', 'short_name': 'Brooklyn',
'types': ['political', 'sublocality', 'sublocality_level_1']}, {'long_nam
e': 'New York', 'short_name': 'NY', 'types': ['administrative_area_level_
1', 'political']}, {'long_name': 'United States', 'short_name': 'US', 'ty
pes': ['country', 'political']}], 'formatted_address': 'Kings County, Bro
oklyn, NY, USA', 'geometry': {'bounds': {'northeast': {'lat': 40.739446,
'lng': -73.8333651}, 'southwest': {'lat': 40.551042, 'lng': -74.05663}},
'location': {'lat': 40.6528762, 'lng': -73.95949399999999}, 'location_typ
e': 'APPROXIMATE', 'viewport': {'northeast': {'lat': 40.739446, 'lng': -7
3.8333651}, 'southwest': {'lat': 40.551042, 'lng': -74.05663}}}, 'place_i
d': 'ChIJOwE7_GTtwokRs75rhW4_I6M', 'types': ['administrative_area_level_
2', 'political']}, {'access_points': [], 'address_components': [{'long_na
me': 'New York', 'short_name': 'New York', 'types': ['locality', 'politic
al']}, {'long_name': 'New York', 'short_name': 'NY', 'types': ['administr
ative_area_level_1', 'political']}, {'long_name': 'United States', 'short
_name': 'US', 'types': ['country', 'political']}], 'formatted_address':
'New York, NY, USA', 'geometry': {'bounds': {'northeast': {'lat': 40.9175
771, 'lng': -73.70027209999999}, 'southwest': {'lat': 40.4773991, 'lng':
-74.25908989999999}}, 'location': {'lat': 40.7127753, 'lng': -74.005972
8}, 'location_type': 'APPROXIMATE', 'viewport': {'northeast': {'lat': 40.
9175771, 'lng': -73.70027209999999}, 'southwest': {'lat': 40.4773991, 'ln
g': -74.25908989999999}}}, 'place_id': 'ChIJOwg_06VPwokRYv534QaPC8g', 'ty
pes': ['locality', 'political']}, {'access_points': [], 'address_componen
ts': [{'long_name': 'Long Island', 'short_name': 'Long Island', 'types':
['establishment', 'natural_feature']}, {'long_name': 'New York', 'short_n
ame': 'NY', 'types': ['administrative_area_level_1', 'political']}, {'lon
g_name': 'United States', 'short_name': 'US', 'types': ['country', 'polit
ical']}], 'formatted_address': 'Long Island, New York, USA', 'geometry':
{'bounds': {'northeast': {'lat': 41.1612401, 'lng': -71.85620109999999},
'southwest': {'lat': 40.5429789, 'lng': -74.0419497}}, 'location': {'la
t': 40.789142, 'lng': -73.13496099999999}, 'location_type': 'APPROXIMAT
E', 'viewport': {'northeast': {'lat': 41.1612401, 'lng': -71.856201099999
99}, 'southwest': {'lat': 40.5429789, 'lng': -74.0419497}}}, 'place_id':
'ChIJy6Xu4VRE6IkRGA2UhmH59x0', 'types': ['establishment', 'natural_featur
e']}, {'access_points': [], 'address_components': [{'long_name': 'New Yor
k', 'short_name': 'NY', 'types': ['administrative_area_level_1', 'politic
al']}, {'long_name': 'United States', 'short_name': 'US', 'types': ['coun
try', 'political']}], 'formatted_address': 'New York, USA', 'geometry':
```

```
{'bounds': {'northeast': {'lat': 45.015861, 'lng': -71.777491}, 'southwes
t': {'lat': 40.4773991, 'lng': -79.7625901}}, 'location': {'lat': 43.2994
285, 'lng': -74.21793260000001}, 'location_type': 'APPROXIMATE', 'viewpor
t': {'northeast': {'lat': 45.015861, 'lng': -71.777491}, 'southwest': {'l
at': 40.4773991, 'lng': -79.7625901}}}, 'place_id': 'ChIJqaUj8fBLzEwRZ5UY
3sHGz90', 'types': ['administrative_area_level_1', 'political']}, {'acces
s_points': [], 'address_components': [{'long_name': 'United States', 'sho
rt_name': 'US', 'types': ['country', 'political']}], 'formatted_address':
'United States', 'geometry': {'bounds': {'northeast': {'lat': 71.5388001,
'lng': -66.885417}, 'southwest': {'lat': 18.7763, 'lng': 170.5957}}, 'loc
ation': {'lat': 37.09024, 'lng': -95.712891}, 'location_type': 'APPROXIMA
TE', 'viewport': {'northeast': {'lat': 71.5388001, 'lng': -66.885417}, 's
outhwest': {'lat': 18.7763, 'lng': 170.5957}}}, 'place_id': 'ChIJCzYy5IS1
6lQRQrfeQ5K5Oxw', 'types': ['country', 'political']}]
```

In [158]:

```
# Explore reply
print(reverse_geocode_result[0]['geometry'])
```

```
{'location': {'lat': 40.71423350000001, 'lng': -73.9613686}, 'location_ty
pe': 'ROOFTOP', 'viewport': {'northeast': {'lat': 40.71558248029151, 'ln
g': -73.9600196197085}, 'southwest': {'lat': 40.71288451970851, 'lng': -7
3.96271758029151}}}
```

In [159]:

```
# We will parse the geometry part
```

In [160]:

```
# Look up an address with reverse geocoding
reverse_geocode_result = gmaps.reverse_geocode((40.714224, -73.961452))
parcelID = '90298'
Geographic_Location_Coordinates = pd.DataFrame(columns={'parcelID','lat',
'lng','loc_type','view_NW_lat','view_NW_lng','view_NW_lat','view_NW_lng'
})

dfs = []
for item in reverse_geocode_result:
    lat = item['geometry']['location']['lat']
    lng = item['geometry']['location']['lng']
    loc_type = item['geometry']['location_type']
    view_NW_lat = item['geometry']['viewport']['northeast']['lat']
    view_NW_lng = item['geometry']['viewport']['northeast']['lng']
    view_NW_lat = item['geometry']['viewport']['southwest']['lat']
    view_NW_lng = item['geometry']['viewport']['southwest']['lng']
    dfs.append(
            {
                'parcelID' : parcelID,
                'lat': lat ,
                'lng' : lng,
                'loc_type': loc_type,
                'view_NW_lat' : view_NW_lat,
                'view_NW_lng' : view_NW_lng,
                'view_NW_lat' : view_NW_lat,
                'view_NW_lng' : view_NW_lng
            })

    #print(lat,lng,loc_type,view_NW_lat,view_NW_lng,view_NW_lat,view_NW_l
ng)
    #print('\n')

Geographic_Location_Coordinates = pd.DataFrame(dfs)
print(Geographic_Location_Coordinates)
```

```
    parcelID        lat        lng           loc_type  view_NW_lat  \
0      90298  40.714234 -73.961369            ROOFTOP    40.712885   
1      90298  40.714221 -73.961290            ROOFTOP    40.712872   
2      90298  40.714202 -73.961308            ROOFTOP    40.712859   
3      90298  40.714255 -73.961453  RANGE_INTERPOLATED    40.712906   
4      90298  40.714205 -73.961484    GEOMETRIC_CENTER    40.712857   
5      90298  40.709336 -73.956555         APPROXIMATE    40.700833   
6      90298  40.708116 -73.957070         APPROXIMATE    40.697933   
7      90298  40.678178 -73.944158         APPROXIMATE    40.551042   
8      90298  40.652876 -73.959494         APPROXIMATE    40.551042   
9      90298  40.712775 -74.005973         APPROXIMATE    40.477399   
10     90298  40.789142 -73.134961         APPROXIMATE    40.542979   
11     90298  43.299428 -74.217933         APPROXIMATE    40.477399   
12     90298  37.090240 -95.712891         APPROXIMATE    18.776300   

    view_NW_lng  
0    -73.962718  
1    -73.962639  
2    -73.962645  
3    -73.962802  
4    -73.962837  
5    -73.964470  
6    -73.969845  
7    -74.056630  
8    -74.056630  
9    -74.259090  
10   -74.041950  
11   -79.762590  
12   170.595700  
```

Now, we will implement on 20 records of the lonlat table. Notice that googlemap return error code 400 for invalid lon/lat values. Care has been taken to avoid recording NaN's in the table in such circumstance.

In [161]:

```
Geographic_Location_Coordinates = pd.DataFrame(columns={'parcelID','lat',
'lng','loc_type','view_NW_lat','view_NW_lng','view_NW_lat','view_NW_lng'
})

dfs = []
for index, row in LonLat[:20].iterrows():
    parcelId = row['parcelid']
    latitude = row['latitude']/10**6
    longitude = row['longitude']/10**6
    #print(parcelId,latitude,longitude)
    try:
        reverse_geocode_result = gmaps.reverse_geocode((latitude, longitu
de))
        #print(reverse_geocode_result[0]['geometry'])
        for item in reverse_geocode_result:
            lat = item['geometry']['location']['lat']
            lng = item['geometry']['location']['lng']
            loc_type = item['geometry']['location_type']
            view_NW_lat = item['geometry']['viewport']['northeast']['lat'
]
            view_NW_lng = item['geometry']['viewport']['northeast']['lng'
]
            view_NW_lat = item['geometry']['viewport']['southwest']['lat'
]
            view_NW_lng = item['geometry']['viewport']['southwest']['lng'
]
            dfs.append(
                {
                    'parcelID' : parcelID,
                    'lat': lat ,
                    'lng' : lng,
                    'loc_type': loc_type,
                    'view_NW_lat' : view_NW_lat,
                    'view_NW_lng' : view_NW_lng,
                    'view_NW_lat' : view_NW_lat,
                    'view_NW_lng' : view_NW_lng
                })
    except:
        continue


Geographic_Location_Coordinates = pd.DataFrame(dfs)
print(Geographic_Location_Coordinates)
```

```
    parcelID        lat         lng           loc_type  view_NW_lat  \
0      90298  34.186396 -118.766827            ROOFTOP    34.185047
1      90298  34.186270 -118.766494  RANGE_INTERPOLATED    34.184921
2      90298  34.186411 -118.766587    GEOMETRIC_CENTER    34.185062
3      90298  34.188033 -118.760611         APPROXIMATE    34.167911
4      90298  34.370488 -119.139064         APPROXIMATE    33.163493
..       ...        ...         ...                 ...          ...
160    90298  34.183616 -118.943432         APPROXIMATE    34.178342
161    90298  34.181067 -118.947042         APPROXIMATE    34.135933
162    90298  34.370488 -119.139064         APPROXIMATE    33.163493
163    90298  36.778261 -119.417932         APPROXIMATE    32.528832
164    90298  37.090240  -95.712891         APPROXIMATE    18.776300

     view_NW_lng
0    -118.768176
1    -118.767843
2    -118.767936
3    -118.789393
4    -119.636302
..           ...
160  -118.950291
161  -119.007712
162  -119.636302
163  -124.482003
164   170.595700

[165 rows x 6 columns]
```

Milestone Conclusion

We now have three tables from their respective sources. All three tables are linked by parcelID. The relationship betwen propertiesandtransactions table, comp_listing_table, and the new table Geographic_Location_Coordinates is one-to-many.

Milestone 5. Merging the data and storing in a database/visualizing data

Description

We will store tables fron previous milestones in sqlite and make queries from them using parcelID as index. We will also provide visulization of the stored data.