Premenné, ktoré ovplyvňujú vývoj cien nehnuteľností v New Yourku

In [1]:
```
import plotly.express as px
import numpy as np
import pandas as pd
from pandas.plotting import scatter_matrix
import numpy as np
import os
import json
import seaborn as sns
import matplotlib.pyplot as plt
import math
import datetime

import helpers


# set your own path boi
# data_path = '/Users/jdurej/Documents/škola/ING_2/OZNAL/projekt/dataset/'
data_path = 'E:/FIIT/ing/2_semester/OZNAL/dataset/'
```

Navigation

- Load data
- Merge Data
- Column Analysis
  - BOROUGH
  - NEIGHBORHOOD
  - BUILDING CLASS CATEGORY
  - TAX CLASS AT PRESENT
  - BLOCK
  - LOT
  - EASE-MENT
  - BUILDING CLASS AT PRESENT
  - ADDRESS
  - ZIP CODE
  - RESIDENTIAL UNITS
  - COMMERCIAL UNITS
  - TOTAL UNITS
  - LAND SQUARE FEET
  - GROSS SQUARE FEET
  - YEAR BUILT
  - TAX CLASS AT TIME OF SALE
  - BUILDING CLASS AT TIME OF SALE
  - SALE PRICE
  - SALE DATE
- Column Basic Cleaning
  - BOROUGH
  - ADDRESS
  - NEIGHBORHOOD
  - BUILDING CLASS CATEGORY
  - TAX CLASS AT PRESENT
  - BUILDING CLASS AT PRESENT
  - APARTMENT NUMBER
  - ZIP CODE

    - RESIDENTIAL, COMMERCIAL UNITS and TOTAL UNITS
    - LAND SQUARE FEET
    - GROSS SQUARE FEET
    - YEAR BUILT
    - SALE PRICE
- Post-Cleaning analysis
    - Tax class at time of sale
    - Total units
- Post-Cleaning operations
- PRICE PREDICTION - TAX CLASS 2
    - **!!! konzultovat**

Load data

In [2]:
```
data_2003 = pd.read_excel (r'E:/FIIT/ing/2_semester/OZNAL/dataset/sales_manhattan_03.xls/sales_manhattan_03.xls', skiprows=3)
data_2004 = pd.read_excel (r'E:/FIIT/ing/2_semester/OZNAL/dataset/sales_manhattan_04.xls/sales_manhattan_04.xls', skiprows=3)
data_2005 = pd.read_excel (r'E:/FIIT/ing/2_semester/OZNAL/dataset/sales_manhattan_05.xls/sales_manhattan_05.xls', skiprows=3)
data_2006 = pd.read_excel (r'E:/FIIT/ing/2_semester/OZNAL/dataset/sales_manhattan_06.xls/sales_manhattan_06.xls', skiprows=3)
data_2007 = pd.read_excel (r'E:/FIIT/ing/2_semester/OZNAL/dataset/sales_2007_manhattan.xls/sales_2007_manhattan.xls', skiprows=3)
data_2008 = pd.read_excel (r'E:/FIIT/ing/2_semester/OZNAL/dataset/sales_2008_manhattan.xls/sales_2008_manhattan.xls', skiprows=3)
data_2009 = pd.read_excel (r'E:/FIIT/ing/2_semester/OZNAL/dataset/2009_manhattan.xls/2009_manhattan.xls', skiprows=3)
data_2010 = pd.read_excel (r'E:/FIIT/ing/2_semester/OZNAL/dataset/2010_manhattan.xls/2010_manhattan.xls', skiprows=3)
data_2011 = pd.read_excel (r'E:/FIIT/ing/2_semester/OZNAL/dataset/2011_manhattan.xls/2011_manhattan.xls', skiprows=4)
data_2012 = pd.read_excel (r'E:/FIIT/ing/2_semester/OZNAL/dataset/2012_manhattan.xls/2012_manhattan.xls', skiprows=4)
data_2013 = pd.read_excel (r'E:/FIIT/ing/2_semester/OZNAL/dataset/2013_manhattan.xls/2013_manhattan.xls', skiprows=4)
data_2014 = pd.read_excel (r'E:/FIIT/ing/2_semester/OZNAL/dataset/2014_manhattan.xls/2014_manhattan.xls', skiprows=4)
data_2015 = pd.read_excel (r'E:/FIIT/ing/2_semester/OZNAL/dataset/2015_manhattan.xls/2015_manhattan.xls', skiprows=4)
data_2016 = pd.read_excel (r'E:/FIIT/ing/2_semester/OZNAL/dataset/2016_manhattan.xls/2016_manhattan.xls', skiprows=4)
data_2017 = pd.read_excel (r'E:/FIIT/ing/2_semester/OZNAL/dataset/2017_manhattan.xls/2017_manhattan.xls', skiprows=4)
data_2018 = pd.read_excel (r'E:/FIIT/ing/2_semester/OZNAL/dataset/rollingsales_manhattan.xls/rollingsales_manhattan.xls', skiprows=4)

data_2017 = data_2017.rename(columns = {'TAX CLASS AS OF FINAL ROLL 17/18':'TAX CLASS AT PRESENT',
                           'BUILDING CLASS AS OF FINAL ROLL 17/18':'BUILDING CLASS AT PRESENT'}, inplace = True)
```

Merge Data

In [3]:
```
df_group_1 = pd.concat([data_2003,data_2004,data_2005,data_2006,data_2007,data_2008,data_2009,data_2010,data_2011])
df_group_2 = pd.concat([data_2012,data_2013,data_2014,data_2015,data_2016,data_2017])
df_group_2.columns = df_group_2.columns.str.replace('\n', '')

df_raw = pd.concat([df_group_1,df_group_2,data_2018])
```

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

```
Index(['BOROUGH', 'NEIGHBORHOOD', 'BUILDING CLASS CATEGORY',
       'TAX CLASS AT PRESENT', 'BLOCK', 'LOT', 'EASE-MENT',
       'BUILDING CLASS AT PRESENT', 'ADDRESS', 'APARTMENT NUMBER', 'ZIP CODE',
       'RESIDENTIAL UNITS', 'COMMERCIAL UNITS', 'TOTAL UNITS',
       'LAND SQUARE FEET', 'GROSS SQUARE FEET', 'YEAR BUILT',
       'TAX CLASS AT TIME OF SALE', 'BUILDING CLASS AT TIME OF SALE',
       'SALE PRICE', 'SALE DATE'],
      dtype='object')
21
```

Out[4]:
```
353738
```

Column Analysis

BOROUGH

- nepodstatne

In [5]:
```
print(df_raw['BOROUGH'].unique())
print(df_raw['BOROUGH'].dtypes)
```

```
[1]
int64
```

NEIGHBORHOOD

Mozne operacie:

- vymazat whitespace
- zjednotit UPPER EAST SIDE a UPPER WEST SIDE. Tie districts su porozdelovane na mensie casti v rozsahu ulic (netusim preco).
- zjednotit HARLEM ??? (UPPER, CENTRAL, EAST, WEST)
- zjednotit GREENWICH VILLAGE ???
- zjednotit WASHINGTON HEIGHTS ???
- zjednotit MIDTOWN ???
- spracovat MANHATTAN-UKNOWN (vymazat/prepisat)
- hodnoty 1021 a 1026 prepisat na zaklade adresy ineho zaznamu, ktory ma uvedeny Neighborhood. Minimalne zaznam 1021 vieme spravit.

In [6]:
```python
print(df_raw['NEIGHBORHOOD'].unique())
print(df_raw['NEIGHBORHOOD'].dtypes)

df_raw[df_raw['NEIGHBORHOOD'] == 'HARLEM-CENTRAL           ']
```

```
['ALPHABET CITY            ' 'CHELSEA                  '
 'CHINATOWN                ' 'CIVIC CENTER             '
 'CLINTON                  ' 'EAST VILLAGE             '
 'FASHION                  ' 'FINANCIAL                '
 'FLATIRON                 ' 'GRAMERCY                 '
 'GREENWICH VILLAGE-CENTRAL' 'GREENWICH VILLAGE-WEST   '
 'HARLEM-CENTRAL           ' 'HARLEM-EAST              '
 'HARLEM-UPPER             ' 'HARLEM-WEST              '
 'INWOOD                   ' 'JAVITS CENTER            '
 'KIPS BAY                 ' 'LITTLE ITALY             '
 'LOWER EAST SIDE          ' 'MANHATTAN VALLEY         '
 'MIDTOWN CBD              ' 'MIDTOWN EAST             '
 'MIDTOWN WEST             ' 'MORNINGSIDE HEIGHTS      '
 'MURRAY HILL              ' 'SOHO                     '
 'SOUTHBRIDGE              ' 'TRIBECA                  '
 'UPPER BAY                ' 'UPPER EAST SIDE (59-79)  '
 'UPPER EAST SIDE (79-96)  ' 'UPPER EAST SIDE (96-110) '
 'UPPER WEST SIDE (59-79)  ' 'UPPER WEST SIDE (79-96)  '
 'UPPER WEST SIDE (96-116) ' 'WASHINGTON HEIGHTS LOWER '
 'WASHINGTON HEIGHTS UPPER ' 'MANHATTAN-UNKNOWN        ' 1021 1026
 'ALPHABET CITY' 'CHELSEA' 'CHINATOWN' 'CIVIC CENTER' 'CLINTON'
 'EAST VILLAGE' 'FASHION' 'FINANCIAL' 'FLATIRON' 'GRAMERCY'
 'GREENWICH VILLAGE-WEST' 'HARLEM-CENTRAL' 'HARLEM-EAST' 'HARLEM-UPPER'
 'HARLEM-WEST' 'INWOOD' 'JAVITS CENTER' 'KIPS BAY' 'LITTLE ITALY'
 'LOWER EAST SIDE' 'MANHATTAN VALLEY' 'MIDTOWN CBD' 'MIDTOWN EAST'
 'MIDTOWN WEST' 'MORNINGSIDE HEIGHTS' 'MURRAY HILL' 'ROOSEVELT ISLAND'
 'SOHO' 'SOUTHBRIDGE' 'TRIBECA' 'UPPER EAST SIDE (59-79)'
 'UPPER EAST SIDE (79-96)' 'UPPER EAST SIDE (96-110)'
 'UPPER WEST SIDE (59-79)' 'UPPER WEST SIDE (79-96)'
 'UPPER WEST SIDE (96-116)' 'WASHINGTON HEIGHTS LOWER'
 'WASHINGTON HEIGHTS UPPER']
object
```

Out[6]:

	BOROUGH	NEIGHBORHOOD	BUILDING CLASS CATEGORY	TAX CLASS AT PRESENT	BLOCK	LOT	EASE-MENT	BUILDING CLASS AT PRESENT	ADDRESS	APARTMENT NUMBER	...	RESIDENTIAL UNITS	COMMERCIAL UNITS	TOTAL UNITS	LAND SQUARE FEET	GROSS SQUARE FEET	YEAR BUILT	TAX CLASS AT TIME OF SALE	BUILDING CLASS AT TIME OF SALE	SALE PRICE	SALE DATE
4428	1	HARLEM-CENTRAL	01 ONE FAMILY HOMES	1	1718	150		A4	36 WEST 120 STREET		...	1	0	1	1850	3132	1899	1	A4	0	2003-08-31
4429	1	HARLEM-CENTRAL	01 ONE FAMILY HOMES	1	1753	5		A5	3 EAST 128 STREET		...	1	0	1	1998	1608	1909	1	A5	650000	2003-10-30
4430	1	HARLEM-CENTRAL	01 ONE FAMILY HOMES		1827	25			209 WEST 111 STREET		...	0	0	0	0	0	0	1	A4	508000	2003-06-16
4431	1	HARLEM-CENTRAL	01 ONE FAMILY HOMES	2A	1905	19		C3	123 WEST 120 STREET		...	4	0	4	2018	5144	1901	1	A4	0	2003-03-18
4432	1	HARLEM-CENTRAL	01 ONE FAMILY HOMES	1	1914	113		B1	147 WEST 129 STREET		...	2	0	2	1665	3200	1899	1	A4	0	2003-06-20
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...

	BOROUGH	NEIGHBORHOOD	BUILDING CLASS CATEGORY	TAX CLASS AT PRESENT	BLOCK	LOT	EASE-MENT	BUILDING CLASS AT PRESENT	ADDRESS	APARTMENT NUMBER	...	RESIDENTIAL UNITS	COMMERCIAL UNITS	TOTAL UNITS	LAND SQUARE FEET	GROSS SQUARE FEET	YEAR BUILT	TAX CLASS AT TIME OF SALE	BUILDING CLASS AT TIME OF SALE	SALE PRICE	SALE DATE
6074	1	HARLEM-CENTRAL	47 CONDO NON-BUSINESS STORAGE		1911	1318			139-141 WEST 126TH STREET		...	0	0	0	0	0	0	4	RS	0	2016-06-20
6075	1	HARLEM-CENTRAL	47 CONDO NON-BUSINESS STORAGE		1911	1319			139 WEST 126 STREET		...	0	0	0	0	0	0	4	RS	0	2016-08-23
6076	1	HARLEM-CENTRAL	47 CONDO NON-BUSINESS STORAGE		1911	1320			139-141 WEST 126TH STREET		...	0	0	0	0	0	0	4	RS	0	2016-06-14
6077	1	HARLEM-CENTRAL	47 CONDO NON-BUSINESS STORAGE		1911	1321			139-141 WEST 126TH STREET		...	0	0	0	0	0	0	4	RS	0	2016-06-27
6078	1	HARLEM-CENTRAL	47 CONDO NON-BUSINESS STORAGE		1911	1322			139 WEST 126 STREET		...	0	0	0	0	0	0	4	RS	0	2016-12-27

13108 rows × 21 columns

BUILDING CLASS CATEGORY

Mozne operacie:

- vymazat whitespace
- prepisat blank hodnoty na NaN

In [7]:
```
print(df_raw['BUILDING CLASS CATEGORY'].unique())
print(df_raw['BUILDING CLASS CATEGORY'].dtypes)
empty_building_class = df_raw[df_raw['BUILDING CLASS CATEGORY'] == '                                             ']
len(empty_building_class)

temp_df = df_raw.copy(deep=True)

temp_df['BUILDING CLASS CATEGORY'] = temp_df['BUILDING CLASS CATEGORY'].replace(r'\s+', ' ', regex=True)
bcc = temp_df[temp_df['BUILDING CLASS CATEGORY'].notna()]
len(bcc)
print(bcc['BUILDING CLASS CATEGORY'].unique())
```

```
['02  TWO FAMILY HOMES                          '
 '07  RENTALS - WALKUP APARTMENTS               '
 '08  RENTALS - ELEVATOR APARTMENTS             '
 '09  COOPS - WALKUP APARTMENTS                 '
 '10  COOPS - ELEVATOR APARTMENTS               '
 '12  CONDOS - WALKUP APARTMENTS                '
 '13  CONDOS - ELEVATOR APARTMENTS              '
 '14  RENTALS - 4-10 UNIT                       '
 '15  CONDOS - 2-10 UNIT RESIDENTIAL            '
 '16  CONDOS - 2-10 UNIT WITH COMMERCIAL UNIT   '
 '17  CONDOPS                                   '
 '22  STORE BUILDINGS                           '
 '28  COMMERCIAL CONDOS                         '
 '29  COMMERCIAL GARAGES                        '
 '30  WAREHOUSES                                '
 '31  COMMERCIAL VACANT LAND                    '
 '01  ONE FAMILY HOMES                          '
 '03  THREE FAMILY HOMES                        '
 '21  OFFICE BUILDINGS                          '
```

TAX CLASS AT PRESENT

Mozne operacie:

- vymazat whitespace
- prepisat blank hodnoty na NaN

In [8]:
```
print(df_raw['TAX CLASS AT PRESENT'].unique())
a = df_raw['TAX CLASS AT PRESENT'].astype(str)
print(a.dtypes)
print(a.unique())

empty_tax_class = df_raw[df_raw['TAX CLASS AT PRESENT'] == '  ']
len(empty_tax_class)
```

```
[1 '2A' '2B' 2 '2C' 4 '  ' '1C' '1A' 3 ' ']
object
['1' '2A' '2B' '2' '2C' '4' '  ' '1C' '1A' '3' ' ']
```

Out[8]: 4599

BLOCK

- v poriadku

In [9]:
```
print(df_raw['BLOCK'].unique())
print(df_raw['BLOCK'].dtypes)

zero_block_value = df_raw[df_raw['BLOCK'] == 0]
len(zero_block_value)
```

```
[ 375  372  378 ...  128 1612 1865]
int64
```

Out[9]:
```
0
```

LOT

- v poriadku

In [10]:
```
print(df_raw['LOT'].unique())
print(df_raw['LOT'].dtypes)

zero_lot_value = df_raw[df_raw['LOT'] == 0]
len(zero_lot_value)
```

```
[  32   31   33 ... 4506 4507  243]
int64
```

Out[10]:
```
0
```

EASE-MENT

nepodstatne

In [11]:
```
print(df_raw['EASE-MENT'].unique())
print(df_raw['EASE-MENT'].dtypes)
```

```
[' ' 'E']
object
```

BUILDING CLASS AT PRESENT

Mozne operacie:

- prepisat blank hodnoty na NaN

In [12]:
```
print(df_raw['BUILDING CLASS AT PRESENT'].unique())
print(df_raw['BUILDING CLASS AT PRESENT'].dtypes)
```

```
['B9' 'C3' 'C4' 'C1' 'C2' 'D5' 'C7' 'D7' 'C6' 'C0' 'D0' 'D4' 'R2' 'R1'
 'R4' 'S3' 'S4' 'R8' 'R9' 'K9' 'R5' '  ' 'G7' 'V9' 'D1' 'S1' 'B3' 'S2'
 'C5' 'A4' 'D9' 'S5' 'S9' 'O9' 'O1' 'O3' 'K1' 'K4' 'K7' 'K2' 'L8' 'L9'
 'L1' 'V1' 'E1' 'H9' 'D6' 'D8' 'RR' 'B1' 'G8' 'H1' 'G1' 'G9' 'E7' 'D3'
 'O6' 'O4' 'O5' 'H3' 'O2' 'G6' 'U0' 'H2' 'W4' 'L2' 'A5' 'W3' 'B2' 'A9'
 'H8' 'W6' 'W9' 'A1' 'R6' 'C8' 'C9' 'M9' 'N9' 'Z9' 'L3' 'G4' 'G5' 'I7'
 'I9' 'P2' 'M1' 'O7' 'E9' 'F1' 'I4' 'N1' 'F4' 'F9' 'R3' 'E4' 'U9' 'P9'
 'R0' 'I5' 'I1' 'Z4' 'A7' 'J6' 'P1' 'D2' 'K3' 'G2' 'Y4' 'S0' 'O8' 'J2'
 'J8' 'I6' 'W8' 'W7' 'P5' 'F5' 'J9' 'J7' 'M4' 'Q9' 'H6' 'N2' 'K5' 'J5'
 'W5' 'K6' 'W2' 'Q2' 'P7' 'M2' 'G3' 'U6' 'Y1' 'U4' 'J4' 'W1' 'M3' 'R7'
 'A0' 'U1' 'V7' 'H4' 'Q1' 'U2' 'F2' 'H5' 'Z3' 'J1' 'Q3' 'P8' 'V5' 'U7'
 'T9' 'E3' 'N4' 'H7' 'V8' 'U8' 'RG' 'RS' 'RH' 'RK' 'RB' 'RW' 'Y2' 'RA'
 'RT' 'RP' 'Z5' 'J3' 'HR' 'HS' 'HH' 'HB' 'P6' 'Z2' 'GW' 'Z7' 'Y7' ' ' 'G0']
object
```

ADDRESS

Mozne operacie:

- vymazat riadky celkovo ak nema ani Neighbourhood
- vymazat whitespace

In [13]:
```
print(df_raw['ADDRESS'].unique())
print(df_raw['ADDRESS'].dtypes)
```

```
['746 EAST 6 STREET                         '
 '316 EAST 3 STREET                         '
 '125 AVENUE D                              ' ... '2414 AMSTERDAM AVENUE'
 '569 WEST 183RD STREET' '736 WEST 187 STREET']
object
```

APARTMENT NUMBER

- nepodstatne?

In [14]:
```
print(df_raw['APARTMENT NUMBER'].unique())
print(df_raw['APARTMENT NUMBER'].dtypes)
```

```
['            ' 'D1          ' 'A4          ' ... 'COM12' 'COM13' 'COM14']
object
```

ZIP CODE

- doplnit zipcode ak existuje zaznam s rovnakou adresou a udanym PSC... alebo vymazat zaznamy

In [15]:
```
print(df_raw['ZIP CODE'].unique())
print(df_raw['ZIP CODE'].dtypes)
```

```
[10009 10011 10002 10003 10036 10019 10001 10024 10028 10022 10014 10128
 10010 10271 10165 10004     0 10013 10038 10032 10007 10018 10016 10006
 10005 10021 10017 10012 10025 10063 10027 10035 10026 10030 10031 10037
 10039 10029 10463 10034 10040 10033 10052 10023 10151 10222 10020 10280
 10048 10044 10075 10046 10065 10110 10121 10125 10218 10150 10118 10129
 10158 10169 10223 10069 10176 10170 10008 10104 10123 10015 10281 10282
 10105 10112]
int64
```

RESIDENTIAL UNITS

- ak ma budova rezidencne vyuzitie, tak spodne outlinery nebrat do uvahy

In [16]:
```
print(df_raw['RESIDENTIAL UNITS'].unique())
print(df_raw['RESIDENTIAL UNITS'].dtypes)
```

```
[   2    4    6   24    5   63    9    8   16   10   20   30    0    3
    1   62   48   15   17    7   12   40  200  376  180   14   13   18
   22   11   36  921  597  262   25   39   21   26   37   85   56   41
   38   47  333  397  410  189  650  476   72  430   50   35   54   32
   23   57  209  112  323   34   28   33   27   31   49   42   19   29
  538   92   90  111   84  235   74  164  152   64   60   44  134   98
   79   46   52   68   59   67  100  210   96   94  894  360   80  110
  369  151   82   76  150  242  115   93   53  148  187   95 2235   88
  258   75   66  378  136  181   43   55  105  106  411  159  160   70
   91  153  198  114   86  158  508   51  267  107   83   45  317   61
   69  142   78  204  293   58  326  371  254  123  237  507  340  206
  339  240  184  127  214  138   65  129   89   77  185  259  384  252
  836  866  880 1681  268  525  263  230   99  764  459  196  128  260
  120  109  275  121  480  193  131  116  165  215  218  195  119  496
  173  102  179 1000  274  245  658  248  330  301  140  169  229  354
  143  135   73  108  130  529  345  266  292  133   87  343  280  914
  341  600  104  243  334   81  264  241  113  286  163  149  289  695
  174  516  455  484  246  418  145  500  325  416  125   71  404   97
 8756 2491  273  251  144  315  546  364  126  103  183  166  124  156
  448  303  172  420  203  213  363  117  389  155  569  221  162  421
  255  101  311  161  232  194  147  261  137  212  132  197  170  168
  294  298  835  510  236  890  186  256  576  118  231  489  176  479
  146  199  498  219  178  840  222  207  299  322  396  283  300  141
  287  705  493  706  652  265  594  308  122  192  305  904 1092  154
  202  375  157  522  233 1328  177  188  390  318  490  316  324  392
  269  270  844 8759  205  338  426  234  250  414  310  201  211  223
  771  257  224 1641  320  464  190  446  503  295  182]
int64
```

In [17]:
```
thousand = df_raw[df_raw['RESIDENTIAL UNITS'] == 8756]
thousand
```

Out[17]:

	BOROUGH	NEIGHBORHOOD	BUILDING CLASS CATEGORY	TAX CLASS AT PRESENT	BLOCK	LOT	EASE-MENT	BUILDING CLASS AT PRESENT	ADDRESS	APARTMENT NUMBER	...	RESIDENTIAL UNITS	COMMERCIAL UNITS	TOTAL UNITS	LAND SQUARE FEET	GROSS SQUARE FEET	YEAR BUILT	TAX CLASS AT TIME OF SALE	BUILDING CLASS AT TIME OF SALE	SALE PRICE	SALE DATE
7777	1	KIPS BAY	08 RENTALS - ELEVATOR APARTMENTS	2	972	1		D7	240 1 AVENUE		...	8756	44	8800	2675000	8942176	1945	2	D7	4040527000	2006-11-17
9223	1	KIPS BAY	08 RENTALS - ELEVATOR APARTMENTS	2	972	1		D7	240 1 AVENUE		...	8756	44	8800	2675000	8942176	1945	2	D7	0	2007-02-12
7067	1	KIPS BAY	08 RENTALS - ELEVATOR APARTMENTS	2	972	1		D7	240 1 AVENUE		...	8756	44	8800	2675000	8942176	1945	2	D7	3330132711	2014-06-03

3 rows × 21 columns

COMMERCIAL UNITS

- ak ma budova komercne vyuzitie, tak spodne outlinery nebrat do uvahy

In [18]:
```
print(df_raw['COMMERCIAL UNITS'].unique())
print(df_raw['COMMERCIAL UNITS'].dtypes)
```

```
[   0    1    2    4    3    8    7    6    9    5   62   13  602   23
   17  136  147  114   10   29   30   37   72   39   22  104   11   52
   16   12   19   25  107   50   14   36  101   40   34   91   68   28
  103  150   41   31   87   21  180   67   78   38   93   77   27   61
   35  184  170  140   70   15   20   32   44   90   89   99   47   63
  375  603   94   18   45   56  106   33   51   64   43   60  319   58
   66  144   76   24   42  214   75  341  131 1102   79  102  124   96
  171 2000  565   65   57  117   26   95  158   53  155   55  200  192
   48  335  208  100   81   85   84  604   46  111  125  188  120  153
  730   80  123   92   71  239  207  118  448   49  203  612  211   97
   88   73  313  292  248  318  254  186  167  133   54  229  152]
int64
```

TOTAL UNITS

- 0 hodnoty prepisat na NaN

In [19]:
```
print(df_raw['TOTAL UNITS'].unique())
print(df_raw['TOTAL UNITS'].dtypes)
```

```
[   2    4    7   24    5   63   10   17    9   22   34    0    3    8
    1   48   15   12   25   21   40  202  376    6   62  180   11   16
   19   18   23   26   28   42   13   14   20  618   33   36  927  601
  262   37   89   46   41   47  346  136  147  114   29   30   72  104
   52  399  411  194  652  482   74  439  107   50  101   51   35   54
   43   39  220  113  330   27   31   38  207  550   92   90   53  117
   84  242  166  152   66   60   44   56  134   98   70   69   58   59
   32   67   49  102  213   96   94  900  156  362   80   97  460  112
   82   68   78  252  119  103  150   87   76  151  189  167  153   93
   77 2256  258  391  141  181  100   83   55  111  419  159  162   73
   95  198  201   86  161  508  272  190   79   45  317  325   61  142
   85  214  295  326  375  139  265  129   64  184  170  140  245   99
  323  510  208  339  240  144   65   75  261  255  394  254  836  893
  880 1684  274  533  263  231  403  776  459  238  131  260  121  275
  603  135  483  186  196  108  173  218  204  195  496  123  183 1009
  277  658  248  333  302  169  235  164  354  146   71  110  106  764
  130  538  348  273  137   88  350  125  287  915  206  344  128  608
   91  109  243  105  334  217   81  122  132  319  247  286  163  296
  374  154  698  133  175  523  357  484  118  250  427  485  145  120
  500   57  327  425  259  212 8800 2498  341  276  148 1417  124  370
  126  171  158  442  293  157  453  320  421  366  116  392  237  155
 2000  565  570  221  426  315  338  187  165  251  149  224  174  197
  229  301  522  210  894  191  246  257  586  185  234  491  192  487
  200  498  236  179  222  480  579  310  324  406  335  309  230  705
  497  418  284  707  656  271  594  604  311  306  278  115  340  398
  904 1097  380  188  177  526  211  511  730  233 1349  127  176  395
  239  371  611  329  318  182  328  448  372  138  160  393  203  612
  270  847 8805  205  143  428  241  414  168  316  223  313  292  417
  256  771  209  902  269 1653  422  465  452  506  489]
int64
```

LAND SQUARE FEET

- nebrat do uvahy pokial LAND > GROSS square feet

In [20]:
```
print(df_raw['LAND SQUARE FEET'].unique())
print(df_raw['LAND SQUARE FEET'].dtypes)
```

```
[ 2134  5746  2185 ...  7625 15161  4970]
int64
```

GROSS SQUARE FEET

- nebrat do uvahy pokial LAND > GROSS square feet

In [21]:
```
print(df_raw['GROSS SQUARE FEET'].unique())
print(df_raw['GROSS SQUARE FEET'].dtypes)
```

```
[3542 2700 5725 ... 1074 1025 2095]
int64
```

YEAR BUILT

- asi nepodstatne

In [22]:
```
print(df_raw['YEAR BUILT'].unique())
print(df_raw['YEAR BUILT'].dtypes)
```

```
[1899 1900 1910 1880 1920 1925 1902 1928 2005 1930 1939 1935 1940 1929
 1937 1901 1950    0 1985 2004 1905 1944 1986 1850 1921 1915 1917 1911
 2003 1926 1913 1889 1963 1898 1977 1918 1938 1923 1927 1906 1909 1958
 1989 1987 1984 1983 2002 1907 1875 1973 1922 1932 1931 1980 1945 1960
 1894 1903 1912 1934 1975 1990 1830 1965 1890 1951 1896 1998 1999 1957
 2001 1988 1868 1916 1997 1870 2008 2000 1924 1914 1954 2012 2010 1969
 1908 2007 1956 1840 1962 1952 1974 1964 1955 1904 1961 2009 1959 1972
 1976 1846 1941 1895 1966 1981 2011 1887 1971 1843 1946 1947 1869 1968
 1852 2006 1936 1953 1994 1948 1949 1982 1800 1866 1970 1967 1933 1942
 1919 1979 1885 1871 1879 1886 1881 1853 1882 1995 1862 1978 1991 1892
 1897 1865 1856 1860 1859 1847 1993 1826 1884 1996 1836 1873 1888 1648
 1893 1883 1841 1849 1874 1878 1877 1851 1834 1827 1857 1848 1992 1891
 2013 1000 1824 1058 2014 1943 1864 1855 2015 1838 1798 2016 1821 2017
 2018 1825]
int64
```

TAX CLASS AT TIME OF SALE

- v poriadku

In [23]:
```
print(df_raw['TAX CLASS AT TIME OF SALE'].unique())
print(df_raw['TAX CLASS AT TIME OF SALE'].dtypes)
```

```
[1 2 4 3]
int64
```

BUILDING CLASS AT TIME OF SALE

- v poriadku

In [24]:
```
print(df_raw['BUILDING CLASS AT TIME OF SALE'].unique())
print(df_raw['BUILDING CLASS AT TIME OF SALE'].dtypes)
```

```
['B9' 'C3' 'C4' 'C1' 'C2' 'C7' 'D7' 'C6' 'D0' 'D4' 'R2' 'R4' 'S3' 'S4'
 'R1' 'R8' 'R9' 'K9' 'R5' 'G9' 'G2' 'E9' 'V1' 'V9' 'S1' 'B3' 'S2' 'C0'
 'C5' 'D1' 'D6' 'D9' 'S5' 'S9' 'O9' 'O1' 'O3' 'K1' 'K4' 'K7' 'K2' 'L9'
 'L8' 'L3' 'L1' 'F2' 'F9' 'F4' 'G1' 'G6' 'E1' 'E3' 'J9' 'P6' 'P3' 'Z9'
 'O6' 'L2' 'D8' 'O5' 'G8' 'G7' 'O4' 'O2' 'H9' 'W4' 'D5' 'I5' 'A5' 'D3'
 'A4' 'B1' 'D2' 'O7' 'H8' 'I9' 'W6' 'W9' 'J1' 'A9' 'A1' 'R6' 'C8' 'B2'
 'V0' 'V2' 'C9' 'G4' 'G5' 'P2' 'M1' 'V3' 'F1' 'F5' 'I4' 'Y6' 'W8' 'K5'
 'R3' 'U9' 'R0' 'I1' 'H2' 'H3' 'O8' 'J6' 'P1' 'A7' 'N9' 'Y4' 'W3' 'J2'
 'J8' 'Z5' 'I6' 'N4' 'W7' 'M9' 'P5' 'P9' 'J7' 'M4' 'Y5' 'N2' 'H5' 'J5'
 'M3' 'A3' 'W2' 'E7' 'Q2' 'M2' 'H1' 'S0' 'RR' 'H6' 'G3' 'I7' 'U6' 'P7'
 'J3' 'U4' 'U8' 'J4' 'Q6' 'V5' 'U1' 'U2' 'K3' 'W1' 'Q1' 'R7' 'H4' 'E4'
 'Z3' 'V6' 'Q3' 'Q9' 'Z0' 'Q7' 'P8' 'U7' 'V4' 'H7' 'V8' 'Y1' 'Z4' 'T9'
 'RG' 'RS' 'RH' 'RK' 'RB' 'A0' 'RW' 'Y2' 'RA' 'RP' 'RT' 'U0' 'HR' 'HS'
 'HB' 'Z2' 'GW' 'Z7' 'Y7' 'G0']
object
```

SALE PRICE

- vymazat nulove hodnoty
- brat do uvahy inflaciu

In [25]:
```
print(df_raw['SALE PRICE'].unique())
print(df_raw['SALE PRICE'].dtypes)
```

```
[1800000       0  426000 ...  598828  564691  549450]
int64
```

SALE DATE

- v poriadku

In [26]:
```
print(df_raw['SALE DATE'].unique())
print(df_raw['SALE DATE'].dtypes)
```

```
['2003-01-22T00:00:00.000000000' '2003-12-18T00:00:00.000000000'
 '2003-10-23T00:00:00.000000000' ... '2017-12-10T00:00:00.000000000'
 '2018-11-22T00:00:00.000000000' '2018-01-28T00:00:00.000000000']
datetime64[ns]
```

Column Basic Cleaning

In [27]:
```
df_basic_clean = df_raw.copy(deep=True)
```

BOROUGH

In [28]:
```
# vymazanie stlpca
df_basic_clean = df_basic_clean.drop('BOROUGH', 1)

print(df_basic_clean.columns)
print(len(df_basic_clean.columns))
len(df_basic_clean)
```

```
Index(['NEIGHBORHOOD', 'BUILDING CLASS CATEGORY', 'TAX CLASS AT PRESENT',
       'BLOCK', 'LOT', 'EASE-MENT', 'BUILDING CLASS AT PRESENT', 'ADDRESS',
       'APARTMENT NUMBER', 'ZIP CODE', 'RESIDENTIAL UNITS', 'COMMERCIAL UNITS',
       'TOTAL UNITS', 'LAND SQUARE FEET', 'GROSS SQUARE FEET', 'YEAR BUILT',
       'TAX CLASS AT TIME OF SALE', 'BUILDING CLASS AT TIME OF SALE',
       'SALE PRICE', 'SALE DATE'],
      dtype='object')
20
```

Out[28]:
```
353738
```

ADDRESS

In [29]:
```
# Neuvedene hodnoty = NaN
df_basic_clean['ADDRESS'] = df_basic_clean['ADDRESS'].str.strip()
df_basic_clean['ADDRESS'] = df_basic_clean['ADDRESS'].replace(r'\s+', ' ', regex=True)

# tmp_address = df_basic_clean[df_basic_clean['ADDRESS'].isnull()]
# tmp_address
# print(df_basic_clean['ADDRESS'].unique())
```

NEIGHBORHOOD

In [30]:

```python
# Replace values [1021,1026]

tmp_df = df_basic_clean.copy(deep=True)

# REPLACING 1021 WITH VALID VALUE
# Find rows with wrong neighborhood
rows_neigh_address = tmp_df[tmp_df['NEIGHBORHOOD'] == 1021]
# Get address
neigh_address_value = rows_neigh_address['ADDRESS'].unique()[0]

# Find rows with right address
rows_neigh = tmp_df[tmp_df['ADDRESS'] == neigh_address_value]
# Find rows with neighborhood we need
rows_neigh = rows_neigh[rows_neigh['NEIGHBORHOOD'] != 1021]
# Get value of neighborhood
value = rows_neigh.iloc[0]['NEIGHBORHOOD']

print("Replacing value '1021' with value: " + value)
tmp_df.loc[tmp_df["NEIGHBORHOOD"] == 1021, "NEIGHBORHOOD"] = value


# REPLACING 1026 WITH VALID VALUE
# Find rows with wrong neighborhood
rows_neigh_address = tmp_df[tmp_df['NEIGHBORHOOD'] == 1026]
# Get address
neigh_address_value = rows_neigh_address['ADDRESS'].unique()[0]

# Find rows with right address
rows_neigh = tmp_df[tmp_df['ADDRESS'] == neigh_address_value]
# Find rows with neighborhood we need
rows_neigh = rows_neigh[rows_neigh['NEIGHBORHOOD'] != 1026]
# Get value of neighborhood
value = rows_neigh.iloc[0]['NEIGHBORHOOD']

print("Replacing value '1026' with value: " + value)
tmp_df.loc[tmp_df["NEIGHBORHOOD"] == 1026, "NEIGHBORHOOD"] = value
```

```
Replacing value '1021' with value: LITTLE ITALY
Replacing value '1026' with value: MIDTOWN WEST
```

In [31]:
```python
# Trim whitespaces
print(tmp_df['NEIGHBORHOOD'].unique())
tmp_df['NEIGHBORHOOD'] = tmp_df['NEIGHBORHOOD'].str.strip()
tmp_df['NEIGHBORHOOD'] = tmp_df['NEIGHBORHOOD'].replace(r'\s+', ' ', regex=True)
print(tmp_df['NEIGHBORHOOD'].unique())
```

```
['ALPHABET CITY            ' 'CHELSEA                  '
 'CHINATOWN                ' 'CIVIC CENTER             '
 'CLINTON                  ' 'EAST VILLAGE             '
 'FASHION                  ' 'FINANCIAL                '
 'FLATIRON                 ' 'GRAMERCY                 '
 'GREENWICH VILLAGE-CENTRAL' 'GREENWICH VILLAGE-WEST   '
 'HARLEM-CENTRAL           ' 'HARLEM-EAST              '
 'HARLEM-UPPER             ' 'HARLEM-WEST              '
 'INWOOD                   ' 'JAVITS CENTER            '
 'KIPS BAY                 ' 'LITTLE ITALY             '
 'LOWER EAST SIDE          ' 'MANHATTAN VALLEY         '
 'MIDTOWN CBD              ' 'MIDTOWN EAST             '
 'MIDTOWN WEST             ' 'MORNINGSIDE HEIGHTS      '
 'MURRAY HILL              ' 'SOHO                     '
 'SOUTHBRIDGE              ' 'TRIBECA                  '
 'UPPER BAY                ' 'UPPER EAST SIDE (59-79)  '
 'UPPER EAST SIDE (79-96)  ' 'UPPER EAST SIDE (96-110) '
 'UPPER WEST SIDE (59-79)  ' 'UPPER WEST SIDE (79-96)  '
 'UPPER WEST SIDE (96-116) ' 'WASHINGTON HEIGHTS LOWER '
 'WASHINGTON HEIGHTS UPPER ' 'MANHATTAN-UNKNOWN        ' 'ALPHABET CITY'
 'CHELSEA' 'CHINATOWN' 'CIVIC CENTER' 'CLINTON' 'EAST VILLAGE' 'FASHION'
 'FINANCIAL' 'FLATIRON' 'GRAMERCY' 'GREENWICH VILLAGE-WEST'
 'HARLEM-CENTRAL' 'HARLEM-EAST' 'HARLEM-UPPER' 'HARLEM-WEST' 'INWOOD'
 'JAVITS CENTER' 'KIPS BAY' 'LITTLE ITALY' 'LOWER EAST SIDE'
 'MANHATTAN VALLEY' 'MIDTOWN CBD' 'MIDTOWN EAST' 'MIDTOWN WEST'
 'MORNINGSIDE HEIGHTS' 'MURRAY HILL' 'ROOSEVELT ISLAND' 'SOHO'
 'SOUTHBRIDGE' 'TRIBECA' 'UPPER EAST SIDE (59-79)'
 'UPPER EAST SIDE (79-96)' 'UPPER EAST SIDE (96-110)'
 'UPPER WEST SIDE (59-79)' 'UPPER WEST SIDE (79-96)'
 'UPPER WEST SIDE (96-116)' 'WASHINGTON HEIGHTS LOWER'
 'WASHINGTON HEIGHTS UPPER']
['ALPHABET CITY' 'CHELSEA' 'CHINATOWN' 'CIVIC CENTER' 'CLINTON'
 'EAST VILLAGE' 'FASHION' 'FINANCIAL' 'FLATIRON' 'GRAMERCY'
 'GREENWICH VILLAGE-CENTRAL' 'GREENWICH VILLAGE-WEST' 'HARLEM-CENTRAL'
 'HARLEM-EAST' 'HARLEM-UPPER' 'HARLEM-WEST' 'INWOOD' 'JAVITS CENTER'
 'KIPS BAY' 'LITTLE ITALY' 'LOWER EAST SIDE' 'MANHATTAN VALLEY'
 'MIDTOWN CBD' 'MIDTOWN EAST' 'MIDTOWN WEST' 'MORNINGSIDE HEIGHTS'
 'MURRAY HILL' 'SOHO' 'SOUTHBRIDGE' 'TRIBECA' 'UPPER BAY'
 'UPPER EAST SIDE (59-79)' 'UPPER EAST SIDE (79-96)'
 'UPPER EAST SIDE (96-110)' 'UPPER WEST SIDE (59-79)'
 'UPPER WEST SIDE (79-96)' 'UPPER WEST SIDE (96-116)'
 'WASHINGTON HEIGHTS LOWER' 'WASHINGTON HEIGHTS UPPER' 'MANHATTAN-UNKNOWN'
 'ROOSEVELT ISLAND']
```

In [32]:
```
# Rename values of some Neighborhoods
tmp_df.loc[tmp_df["NEIGHBORHOOD"].str.contains('UPPER EAST SIDE'), "NEIGHBORHOOD"] = 'UPPER EAST SIDE'
tmp_df.loc[tmp_df["NEIGHBORHOOD"].str.contains('UPPER WEST SIDE'), "NEIGHBORHOOD"] = 'UPPER WEST SIDE'

print(tmp_df['NEIGHBORHOOD'].unique())
```

```
['ALPHABET CITY' 'CHELSEA' 'CHINATOWN' 'CIVIC CENTER' 'CLINTON'
 'EAST VILLAGE' 'FASHION' 'FINANCIAL' 'FLATIRON' 'GRAMERCY'
 'GREENWICH VILLAGE-CENTRAL' 'GREENWICH VILLAGE-WEST' 'HARLEM-CENTRAL'
 'HARLEM-EAST' 'HARLEM-UPPER' 'HARLEM-WEST' 'INWOOD' 'JAVITS CENTER'
 'KIPS BAY' 'LITTLE ITALY' 'LOWER EAST SIDE' 'MANHATTAN VALLEY'
 'MIDTOWN CBD' 'MIDTOWN EAST' 'MIDTOWN WEST' 'MORNINGSIDE HEIGHTS'
 'MURRAY HILL' 'SOHO' 'SOUTHBRIDGE' 'TRIBECA' 'UPPER BAY'
 'UPPER EAST SIDE' 'UPPER WEST SIDE' 'WASHINGTON HEIGHTS LOWER'
 'WASHINGTON HEIGHTS UPPER' 'MANHATTAN-UNKNOWN' 'ROOSEVELT ISLAND']
```

In [33]:
```
# Spracovanie MANHATTAN-UNKNOWN
# Set to NaN
tmp_df.loc[tmp_df["NEIGHBORHOOD"] == 'MANHATTAN-UNKNOWN', "NEIGHBORHOOD"] = np.nan

# Get adresses
adresses = tmp_df[tmp_df['NEIGHBORHOOD'].isna()]
adresses = adresses['ADDRESS'].unique()

# Addresses included with rows without neighborhood
print(adresses)
```

```
['326 CANAL STREET' '250 SOUTH END AVENUE' '30 WEST ST' '55 WARREN STREET'
 '250 BOWERY' '254 BOWERY' '43 BOND STREET' '1-3 ORCHARD STREET'
 '30 LITTLE WEST STREET' '1 RIVER TERRACE' '377 RECTOR PLACE'
 '2 RIVER TERRACE' '24 DOWNING STREET' '17 WEST 96TH STREET'
 '136 EAST BROADWAY' '503 CANAL STREET' '1653 MADISON AVENUE'
 '66 9 AVENUE' '22 RENWICK STREET' 'WEST 125 STREET' '179 WEST 137 STREET'
 '2341 ADAM C POWELL BLVD' '19 WEST 96TH STREET' '212 WARREN STREET'
 'EAST 29TH STREET']
```

In [34]:
```
# prepisanie adries
for address in adresses:
    neighborhood = tmp_df[tmp_df['ADDRESS'] == address]
    neighborhood = neighborhood[neighborhood['NEIGHBORHOOD'].notna()]
    if(neighborhood.empty == False):
        nieghbor_val = neighborhood.iloc[0]['NEIGHBORHOOD']
        tmp_df.loc[tmp_df["ADDRESS"] == address, "NEIGHBORHOOD"] = nieghbor_val

adresses_less = tmp_df[tmp_df['NEIGHBORHOOD'].isna()]
adresses_less = adresses_less['ADDRESS'].unique()

print(adresses_less)
```

```
['326 CANAL STREET' '254 BOWERY' '43 BOND STREET' '1-3 ORCHARD STREET'
 '24 DOWNING STREET' '2341 ADAM C POWELL BLVD' '19 WEST 96TH STREET']
```

In [35]:
```
# Add to final DF
df_basic_clean["NEIGHBORHOOD"] = tmp_df["NEIGHBORHOOD"]
```

BUILDING CLASS CATEGORY

In [36]:
```
### BUILDING CLASS CATEGORY
df_basic_clean['BUILDING CLASS CATEGORY'] = df_basic_clean['BUILDING CLASS CATEGORY'].str.strip()
df_basic_clean['BUILDING CLASS CATEGORY'] = df_basic_clean['BUILDING CLASS CATEGORY'].replace(r'\s+', ' ', regex=True)
df_basic_clean.loc[df_basic_clean["BUILDING CLASS CATEGORY"] == '', "BUILDING CLASS CATEGORY"] = np.nan
```

In [37]:
```python
building_c_cat = df_basic_clean['BUILDING CLASS CATEGORY'].unique()
for cat in building_c_cat:
    print(cat)
```

```
02 TWO FAMILY HOMES
07 RENTALS - WALKUP APARTMENTS
08 RENTALS - ELEVATOR APARTMENTS
09 COOPS - WALKUP APARTMENTS
10 COOPS - ELEVATOR APARTMENTS
12 CONDOS - WALKUP APARTMENTS
13 CONDOS - ELEVATOR APARTMENTS
14 RENTALS - 4-10 UNIT
15 CONDOS - 2-10 UNIT RESIDENTIAL
16 CONDOS - 2-10 UNIT WITH COMMERCIAL UNIT
17 CONDOPS
22 STORE BUILDINGS
28 COMMERCIAL CONDOS
29 COMMERCIAL GARAGES
30 WAREHOUSES
31 COMMERCIAL VACANT LAND
01 ONE FAMILY HOMES
03 THREE FAMILY HOMES
21 OFFICE BUILDINGS
23 LOFT BUILDINGS
27 FACTORIES
34 THEATRES
35 INDOOR PUBLIC AND CULTURAL FACILITIES
41 TAX CLASS 4 - OTHER
26 OTHER HOTELS
33 EDUCATIONAL FACILITIES
32 HOSPITAL AND HEALTH FACILITIES
04 TAX CLASS 1 CONDOS
05 TAX CLASS 1 VACANT LAND
37 RELIGIOUS FACILITIES
40 SELECTED GOVERNMENTAL FACILITIES
18 TAX CLASS 3 - UTILITY PROPERTIES
11 SPECIAL CONDO BILLING LOTS
25 LUXURY HOTELS
38 ASYLUMS AND HOMES
36 OUTDOOR RECREATIONAL FACILITIES
11A CONDO-RENTALS
06 TAX CLASS 1 - OTHER
39 TRANSPORTATION FACILITIES
nan
02 TWO FAMILY DWELLINGS
03 THREE FAMILY DWELLINGS
17 CONDO COOPS
46 CONDO STORE BUILDINGS
01 ONE FAMILY DWELLINGS
43 CONDO OFFICE BUILDINGS
44 CONDO PARKING
47 CONDO NON-BUSINESS STORAGE
45 CONDO HOTELS
42 CONDO CULTURAL/MEDICAL/EDUCATIONAL/ETC
49 CONDO WAREHOUSES/FACTORY/INDUS
48 CONDO TERRACES/GARDENS/CABANAS
24 TAX CLASS 4 - UTILITY BUREAU PROPERTIES
18 TAX CLASS 3 - UNTILITY PROPERTIES
```

In [38]:

```python
tmp = df_basic_clean[df_basic_clean['BUILDING CLASS CATEGORY'] == '41 TAX CLASS 4 - OTHER']
# tmp = df_basic_clean[df_basic_clean['ADDRESS'] == '2142 AMSTERDAM']
tmp
```

Out[38]:

	NEIGHBORHOOD	BUILDING CLASS CATEGORY	TAX CLASS AT PRESENT	BLOCK	LOT	EASE-MENT	BUILDING CLASS AT PRESENT	ADDRESS	APARTMENT NUMBER	ZIP CODE	RESIDENTIAL UNITS	COMMERCIAL UNITS	TOTAL UNITS	LAND SQUARE FEET	GROSS SQUARE FEET	YEAR BUILT	TAX CLASS AT TIME OF SALE	BUILDING CLASS AT TIME OF SALE	SALE PRICE	SALE DATE
1059	CHELSEA	41 TAX CLASS 4 - OTHER		695	26			519 WEST 23 STREET		10011	0	0	0	0	0	0	4	Z9	2525000	2003-11-14
1680	EAST VILLAGE	41 TAX CLASS 4 - OTHER	2	440	46		D3	427 EAST 12 STREET		10009	11	0	11	2504	11520	2008	4	Z9	0	2003-02-26
5316	HARLEM-CENTRAL	41 TAX CLASS 4 - OTHER		1831	57			248 WEST 116 STREET		10026	0	0	0	0	0	0	4	Z9	670000	2003-04-28
11621	SOHO	41 TAX CLASS 4 - OTHER		473	106			472 BROADWAY		10013	0	0	0	0	0	0	4	Z9	1000000	2003-01-15
11773	SOUTHBRIDGE	41 TAX CLASS 4 - OTHER	2A	107	10		S3	45 PECK SLIP		10038	3	1	4	836	3220	1900	4	Z9	875000	2003-02-11
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
14290	TRIBECA	41 TAX CLASS 4 - OTHER	4	211	3		Z3	350 CANAL STREET		10013	0	2	2	25587	47237	1938	4	Z3	0	2016-04-29
2974	GRAMERCY	41 TAX CLASS 4 - OTHER	4	906	13		Z9	223 EAST 25TH STREET		10010	0	1	1	2469	6507	1915	4	Z9	6000000	2018-02-01
7652	MIDTOWN EAST	41 TAX CLASS 4 - OTHER	4	1325	17		Z4	231 EAST 51ST STREET		10022	0	12	12	3682	18045	1920	4	Z4	22050000	2017-12-29
9519	SOHO	41 TAX CLASS 4 - OTHER	4	579	74		Z9	275 SPRING STREET		10013	0	1	1	13287	0	1900	4	Z9	0	2018-03-07
9520	SOHO	41 TAX CLASS 4 - OTHER	4	579	74		Z9	275 SPRING STREET		10013	0	1	1	13287	0	1900	4	Z9	0	2018-07-03

174 rows × 20 columns

In [39]:

```python
### TAX CLASS AT PRESENT
df_basic_clean['TAX CLASS AT PRESENT'] = df_basic_clean['TAX CLASS AT PRESENT'].astype(str)
df_basic_clean['TAX CLASS AT PRESENT'] = df_basic_clean['TAX CLASS AT PRESENT'].replace(r'\s+', ' ', regex=True)
df_basic_clean.loc[df_basic_clean["TAX CLASS AT PRESENT"] == ' ', "TAX CLASS AT PRESENT"] = np.nan
print(df_basic_clean['TAX CLASS AT PRESENT'].unique())
```

```
['1' '2A' '2B' '2' '2C' '4' nan '1C' '1A' '3']
```

In [40]:
```python
### BUILDING CLASS AT PRESENT
df_basic_clean['BUILDING CLASS AT PRESENT'] = df_basic_clean['BUILDING CLASS AT PRESENT'].astype(str)
df_basic_clean['BUILDING CLASS AT PRESENT'] = df_basic_clean['BUILDING CLASS AT PRESENT'].replace(r'\s+', ' ', regex=True)
df_basic_clean.loc[df_basic_clean["BUILDING CLASS AT PRESENT"] == ' ', "BUILDING CLASS AT PRESENT"] = np.nan
print(df_basic_clean['BUILDING CLASS AT PRESENT'].unique())
```

```
['B9' 'C3' 'C4' 'C1' 'C2' 'D5' 'C7' 'D7' 'C6' 'C0' 'D0' 'D4' 'R2' 'R1'
 'R4' 'S3' 'S4' 'R8' 'R9' 'K9' 'R5' nan 'G7' 'V9' 'D1' 'S1' 'B3' 'S2' 'C5'
 'A4' 'D9' 'S5' 'S9' 'O9' 'O1' 'O3' 'K1' 'K4' 'K7' 'K2' 'L8' 'L9' 'L1'
 'V1' 'E1' 'H9' 'D6' 'D8' 'RR' 'B1' 'G8' 'H1' 'G1' 'G9' 'E7' 'D3' 'O6'
 'O4' 'O5' 'H3' 'O2' 'G6' 'U0' 'H2' 'W4' 'L2' 'A5' 'W3' 'B2' 'A9' 'H8'
 'W6' 'W9' 'A1' 'R6' 'C8' 'C9' 'M9' 'N9' 'Z9' 'L3' 'G4' 'G5' 'I7' 'I9'
 'P2' 'M1' 'O7' 'E9' 'F1' 'I4' 'N1' 'F4' 'F9' 'R3' 'E4' 'U9' 'P9' 'R0'
 'I5' 'I1' 'Z4' 'A7' 'J6' 'P1' 'D2' 'K3' 'G2' 'Y4' 'S0' 'O8' 'J2' 'J8'
 'I6' 'W8' 'W7' 'P5' 'F5' 'J9' 'J7' 'M4' 'Q9' 'H6' 'N2' 'K5' 'J5' 'W5'
 'K6' 'W2' 'Q2' 'P7' 'M2' 'G3' 'U6' 'Y1' 'U4' 'J4' 'W1' 'M3' 'R7' 'A0'
 'U1' 'V7' 'H4' 'Q1' 'U2' 'F2' 'H5' 'Z3' 'J1' 'Q3' 'P8' 'V5' 'U7' 'T9'
 'E3' 'N4' 'H7' 'V8' 'U8' 'RG' 'RS' 'RH' 'RK' 'RB' 'RW' 'Y2' 'RA' 'RT'
 'RP' 'Z5' 'J3' 'HR' 'HS' 'HH' 'HB' 'P6' 'Z2' 'GW' 'Z7' 'Y7' 'G0']
```

In [41]:
```python
### APARTMENT NUMBER
df_basic_clean['APARTMENT NUMBER'] = df_basic_clean['APARTMENT NUMBER'].astype(str)
df_basic_clean['APARTMENT NUMBER'] = df_basic_clean['APARTMENT NUMBER'].str.strip()
df_basic_clean['APARTMENT NUMBER'] = df_basic_clean['APARTMENT NUMBER'].replace(r'\s+', ' ', regex=True)
df_basic_clean.loc[df_basic_clean["APARTMENT NUMBER"] == ' ', "APARTMENT NUMBER"] = np.nan
print(df_basic_clean['APARTMENT NUMBER'].unique())
```

```
['' 'D1' 'A4' ... 'COMM9' 'COM13' 'COM14']
```

In [42]:
```python
### ZIP CODE
df_basic_clean['APARTMENT NUMBER'] = df_basic_clean['APARTMENT NUMBER'].astype(str)
df_basic_clean.loc[df_basic_clean["ZIP CODE"] == '0', "ZIP CODE"] = np.nan
print(df_basic_clean['ZIP CODE'].unique())
```

```
[10009. 10011. 10002. 10003. 10036. 10019. 10001. 10024. 10028. 10022.
 10014. 10128. 10010. 10271. 10165. 10004.     0. 10013. 10038. 10032.
 10007. 10018. 10016. 10006. 10005. 10021. 10017. 10012. 10025. 10063.
 10027. 10035. 10026. 10030. 10031. 10037. 10039. 10029. 10463. 10034.
 10040. 10033. 10052. 10023. 10151. 10222. 10020. 10280. 10048. 10044.
 10075. 10046. 10065. 10110. 10121. 10125. 10218. 10150. 10118. 10129.
 10158. 10169. 10223. 10069. 10176. 10170. 10008. 10104. 10123. 10015.
 10281. 10282. 10105. 10112.]
```

In [43]:
```
### RESIDENTIAL, COMMERCIAL UNITS and TOTAL UNITS
tmp = df_basic_clean[df_basic_clean['COMMERCIAL UNITS'] == 0]
tmp = tmp[tmp['RESIDENTIAL UNITS'] == 0]
tmp = tmp[tmp['TOTAL UNITS'] == 0]

tmp
```

Out[43]:

	NEIGHBORHOOD	BUILDING CLASS CATEGORY	TAX CLASS AT PRESENT	BLOCK	LOT	EASE-MENT	BUILDING CLASS AT PRESENT	ADDRESS	APARTMENT NUMBER	ZIP CODE	RESIDENTIAL UNITS	COMMERCIAL UNITS	TOTAL UNITS	LAND SQUARE FEET	GROSS SQUARE FEET	YEAR BUILT	TAX CLASS AT TIME OF SALE	BUILDING CLASS AT TIME OF SALE	SALE PRICE	SALE DATE
17	ALPHABET CITY	09 COOPS - WALKUP APARTMENTS	2	373	46		C6	317 EAST 3RD ST, APT 5		10009.0	0	0	0	0	0	1925	2	C6	190000	2003-11-14
18	ALPHABET CITY	09 COOPS - WALKUP APARTMENTS	2	373	46		C6	317 EAST 3 ST APT. 21		10009.0	0	0	0	0	0	1925	2	C6	270000	2003-12-05
19	ALPHABET CITY	09 COOPS - WALKUP APARTMENTS	2	373	49		C6	311 EAST 3RD ST. APT. 17		10011.0	0	0	0	0	0	1920	2	C6	125000	2003-02-10
20	ALPHABET CITY	09 COOPS - WALKUP APARTMENTS	2	373	49		C6	311 EAST 3RD STREET, APT 18		10009.0	0	0	0	0	0	1920	2	C6	250000	2003-02-13
21	ALPHABET CITY	09 COOPS - WALKUP APARTMENTS	2	373	49		C6	311 EAST 3RD ST. APT 22		10009.0	0	0	0	0	0	1920	2	C6	179000	2003-03-14
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
16723	WASHINGTON HEIGHTS UPPER	10 COOPS - ELEVATOR APARTMENTS	2	2246	130		D4	1793 RIVERSIDE DRIVE, 3D		10034.0	0	0	0	0	0	1926	2	D4	575000	2018-04-26
16750	WASHINGTON HEIGHTS UPPER	31 COMMERCIAL VACANT LAND	4	2179	153		V1	203 CABRINI BOULEVARD		0.0	0	0	0	1928	0	0	4	V1	3000000	2018-09-04
16751	WASHINGTON HEIGHTS UPPER	31 COMMERCIAL VACANT LAND	4	2179	154		V1	205 CABRINI BOULEVARD		0.0	0	0	0	1775	0	0	4	V1	0	2018-09-04
16752	WASHINGTON HEIGHTS UPPER	31 COMMERCIAL VACANT LAND	4	2179	155		V1	207 CABRINI BOULEVARD		0.0	0	0	0	1555	0	0	4	V1	0	2018-09-04
16753	WASHINGTON HEIGHTS UPPER	31 COMMERCIAL VACANT LAND	4	2179	243		V1	RIVERSIDE DRIVE		10033.0	0	0	0	3990	0	0	4	V1	0	2018-08-30

133187 rows × 20 columns

In [44]:
```
### LAND SQUARE FEET
df_basic_clean.loc[df_basic_clean["LAND SQUARE FEET"] == 0, "LAND SQUARE FEET"] = np.nan
```

In [45]:
```
### GROSS SQUARE FEET
df_basic_clean.loc[df_basic_clean["GROSS SQUARE FEET"] == 0, "GROSS SQUARE FEET"] = np.nan
```

In [46]:
```
### YEAR BUILT
df_basic_clean.loc[df_basic_clean["YEAR BUILT"] == 0, "YEAR BUILT"] = np.nan
print(df_basic_clean['YEAR BUILT'].unique())
```

```
[1899. 1900. 1910. 1880. 1920. 1925. 1902. 1928. 2005. 1930. 1939. 1935.
 1940. 1929. 1937. 1901. 1950.   nan 1985. 2004. 1905. 1944. 1986. 1850.
 1921. 1915. 1917. 1911. 2003. 1926. 1913. 1889. 1963. 1898. 1977. 1918.
 1938. 1923. 1927. 1906. 1909. 1958. 1989. 1987. 1984. 1983. 2002. 1907.
 1875. 1973. 1922. 1932. 1931. 1980. 1945. 1960. 1894. 1903. 1912. 1934.
 1975. 1990. 1830. 1965. 1890. 1951. 1896. 1998. 1999. 1957. 2001. 1988.
 1868. 1916. 1997. 1870. 2008. 2000. 1924. 1914. 1954. 2012. 2010. 1969.
 1908. 2007. 1956. 1840. 1962. 1952. 1974. 1964. 1955. 1904. 1961. 2009.
 1959. 1972. 1976. 1846. 1941. 1895. 1966. 1981. 2011. 1887. 1971. 1843.
 1946. 1947. 1869. 1968. 1852. 2006. 1936. 1953. 1994. 1948. 1949. 1982.
 1800. 1866. 1970. 1967. 1933. 1942. 1919. 1979. 1885. 1871. 1879. 1886.
 1881. 1853. 1882. 1995. 1862. 1978. 1991. 1892. 1897. 1865. 1856. 1860.
 1859. 1847. 1993. 1826. 1884. 1996. 1836. 1873. 1888. 1648. 1893. 1883.
 1841. 1849. 1874. 1878. 1877. 1851. 1834. 1827. 1857. 1848. 1992. 1891.
 2013. 1000. 1824. 1058. 2014. 1943. 1864. 1855. 2015. 1838. 1798. 2016.
 1821. 2017. 2018. 1825.]
```

In [47]:
```
### SALE PRICE
df_basic_clean.loc[df_basic_clean["SALE PRICE"] == 0, "SALE PRICE"] = np.nan
print(df_basic_clean['SALE PRICE'].unique())
```

```
[1800000.      nan  426000. ...  598828.  564691.  549450.]
```

In [48]:
```
tmp = df_basic_clean[df_basic_clean['SALE PRICE'].isna()]
len(tmp)
```

Out[48]:
```
72103
```

Post-Cleaning analysis

In [49]:
```python
df_basic_clean.sample(10)
```

Out[49]:

	NEIGHBORHOOD	BUILDING CLASS CATEGORY	TAX CLASS AT PRESENT	BLOCK	LOT	EASE-MENT	BUILDING CLASS AT PRESENT	ADDRESS	APARTMENT NUMBER	ZIP CODE	RESIDENTIAL UNITS	COMMERCIAL UNITS	TOTAL UNITS	LAND SQUARE FEET	GROSS SQUARE FEET	YEAR BUILT	TAX CLASS AT TIME OF SALE	BUILDING CLASS AT TIME OF SALE	SALE PRICE	SALE DATE
18904	UPPER EAST SIDE	10 COOPS - ELEVATOR APARTMENTS	2	1427	34		D4	220 EAST 73RD STREET, 3F		10021.0	0	0	0	NaN	NaN	1932.0	2	D4	565000.0	2013-02-19
2720	FINANCIAL	13 CONDOS - ELEVATOR APARTMENTS	2	53	1419		R4	123 WASHINGTON STREET	30B	10006.0	1	0	1	NaN	NaN	2007.0	2	R4	1713511.0	2014-10-08
61	ALPHABET CITY	09 COOPS - WALKUP APARTMENTS	2	406	50		C6	527 EAST 12TH STREET, A4		10009.0	0	0	0	NaN	NaN	1901.0	2	C6	302000.0	2009-11-03
15673	MURRAY HILL	14 RENTALS - 4-10 UNIT	2A	914	31		S4	613 2 AVENUE		10016.0	4	1	5	1296.0	3788.0	1910.0	2	S4	NaN	2014-10-07
20872	UPPER EAST SIDE	10 COOPS - ELEVATOR APARTMENTS	2	1507	21		D4	1361 MADISON AVENUE, 6C		10128.0	0	0	0	NaN	NaN	1902.0	2	D4	1125000.0	2012-07-31
1759	FASHION	13 CONDOS - ELEVATOR APARTMENTS	2	760	1020		R4	315-25 WEST 36 STREET	PH-B	10018.0	1	0	1	NaN	NaN	NaN	2	R4	NaN	2003-12-29
10139	MIDTOWN WEST	25 LUXURY HOTELS	4	1009	37		H2	102 WEST 57TH STREET		10019.0	0	2	2	7532.0	112850.0	2007.0	4	H2	49900.0	2011-07-27
15582	MIDTOWN WEST	28 COMMERCIAL CONDOS	4	1027	1354		R5	870 7 AVENUE	1409	10019.0	0	1	1	NaN	NaN	NaN	4	R5	10000.0	2007-09-14
400	CHELSEA	10 COOPS - ELEVATOR APARTMENTS	2	744	1		D4	365 WEST 20TH STREET, 11E		10011.0	0	0	0	NaN	NaN	1928.0	2	D4	575000.0	2004-09-23
4916	GREENWICH VILLAGE-WEST	09 COOPS - WALKUP APARTMENTS	2	632	45		C6	725 GREENWICH STREET, J1		10014.0	0	0	0	NaN	NaN	1911.0	2	C6	495000.0	2014-10-29

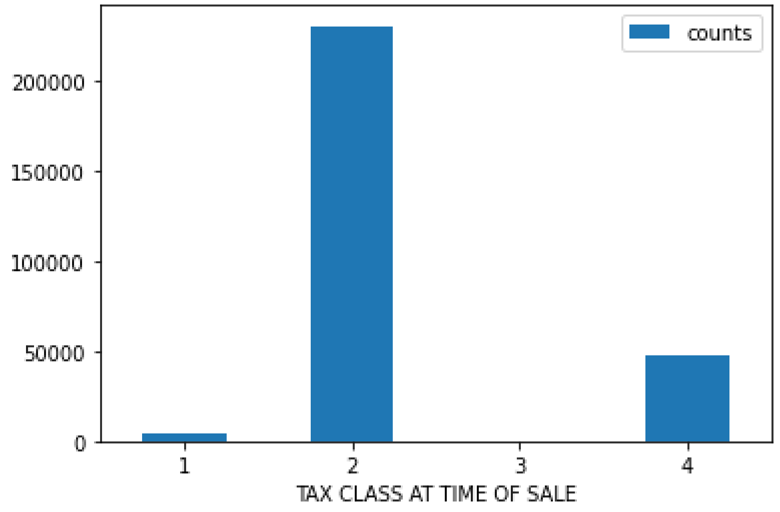
Tax class at time of sale

In [50]:
```python
tax_class_with_price = df_basic_clean[df_basic_clean['SALE PRICE'].notna()]
# tax_class_with_price = tax_class_with_price['TAX CLASS AT TIME OF SALE'].sample(5)
tax_class_with_price = tax_class_with_price.groupby(['TAX CLASS AT TIME OF SALE']).size().reset_index(name='counts')
tax_class_with_price
```

Out[50]:

	TAX CLASS AT TIME OF SALE	counts
0	1	3933
1	2	230447
2	3	14
3	4	47241

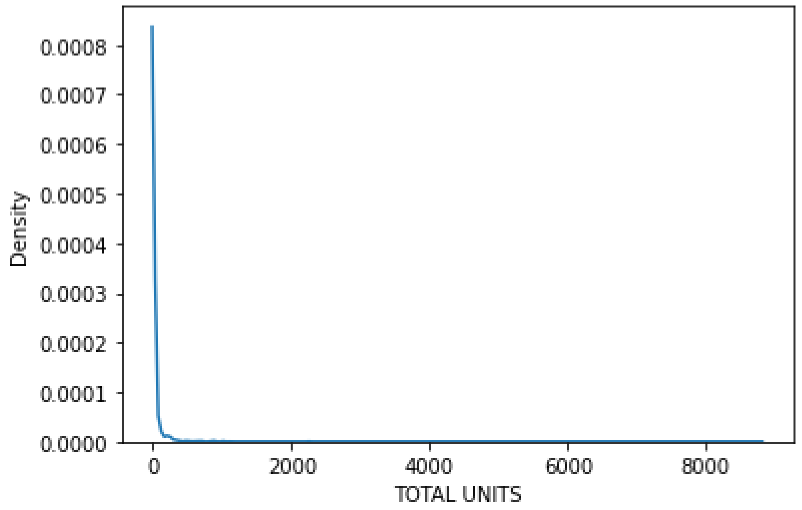
In [51]:
```
ax = tax_class_with_price.plot.bar(x='TAX CLASS AT TIME OF SALE', y='counts', rot=0)
```



Total units

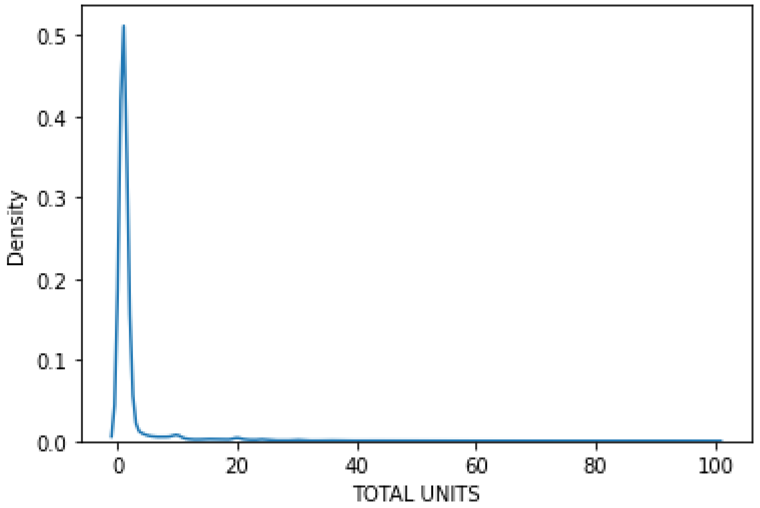
In [56]:
```
sns.kdeplot(df_basic_clean['TOTAL UNITS'])
```

Out[56]: <AxesSubplot:xlabel='TOTAL UNITS', ylabel='Density'>



In [57]:
```
total_units = df_basic_clean[df_basic_clean['TOTAL UNITS'] != 0]
total_units = total_units[total_units['TOTAL UNITS'] < 100]

sns.kdeplot(total_units['TOTAL UNITS'])
```

Out[57]: <AxesSubplot:xlabel='TOTAL UNITS', ylabel='Density'>



In [59]:
```
total_units
```

Out[59]:

	NEIGHBORHOOD	BUILDING CLASS CATEGORY	TAX CLASS AT PRESENT	BLOCK	LOT	EASE-MENT	BUILDING CLASS AT PRESENT	ADDRESS	APARTMENT NUMBER	ZIP CODE	RESIDENTIAL UNITS	COMMERCIAL UNITS	TOTAL UNITS	LAND SQUARE FEET	GROSS SQUARE FEET	YEAR BUILT	TAX CLASS AT TIME OF SALE	BUILDING CLASS AT TIME OF SALE	SALE PRICE	SALE DATE
0	ALPHABET CITY	02 TWO FAMILY HOMES	1	375	32		B9	746 EAST 6 STREET		10009.0	2	0	2	2134.0	3542.0	1899.0	1	B9	1800000.0	2003-01-22
1	ALPHABET CITY	07 RENTALS - WALKUP APARTMENTS	2A	372	31		C3	316 EAST 3 STREET		10009.0	4	0	4	5746.0	2700.0	1900.0	2	C3	NaN	2003-12-18
2	ALPHABET CITY	07 RENTALS - WALKUP APARTMENTS	2A	372	31		C3	316 EAST 3 STREET		10009.0	4	0	4	5746.0	2700.0	1900.0	2	C3	NaN	2003-12-18
3	ALPHABET CITY	07 RENTALS - WALKUP APARTMENTS	2B	378	33		C4	125 AVENUE D		10009.0	6	1	7	2185.0	5725.0	1910.0	2	C4	426000.0	2003-10-23
4	ALPHABET CITY	07 RENTALS - WALKUP APARTMENTS	2	391	12		C1	610 EAST 9 STREET		10009.0	24	0	24	2543.0	11568.0	1910.0	2	C1	NaN	2003-02-28
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
16767	WASHINGTON HEIGHTS UPPER	46 CONDO STORE BUILDINGS	4	2164	1010		RK	4260 BROADWAY	COM10	10033.0	0	1	1	NaN	1218.0	NaN	4	RK	NaN	2018-08-03
16768	WASHINGTON HEIGHTS UPPER	46 CONDO STORE BUILDINGS	4	2164	1011		RK	4260 BROADWAY	COM11	10033.0	0	1	1	NaN	522.0	NaN	4	RK	NaN	2018-08-03
16769	WASHINGTON HEIGHTS UPPER	46 CONDO STORE BUILDINGS	4	2164	1012		RK	4260 BROADWAY	COM12	10033.0	0	1	1	NaN	1025.0	NaN	4	RK	NaN	2018-08-03
16770	WASHINGTON HEIGHTS UPPER	46 CONDO STORE BUILDINGS	4	2164	1013		RK	4260 BROADWAY	COM13	10033.0	0	1	1	NaN	1061.0	NaN	4	RK	NaN	2018-08-03
16771	WASHINGTON HEIGHTS UPPER	46 CONDO STORE BUILDINGS	4	2164	1014		RK	4260 BROADWAY	COM14	10033.0	0	1	1	NaN	2095.0	NaN	4	RK	NaN	2018-08-03

219354 rows × 20 columns

Post-Cleaning operations

In [60]:
```
df_processed = df_basic_clean.copy(deep=True)
len(df_processed)
```

Out[60]: 353738

In [61]:
```
df_processed.head()
```

Out[61]:

	NEIGHBORHOOD	BUILDING CLASS CATEGORY	TAX CLASS AT PRESENT	BLOCK	LOT	EASE-MENT	BUILDING CLASS AT PRESENT	ADDRESS	APARTMENT NUMBER	ZIP CODE	RESIDENTIAL UNITS	COMMERCIAL UNITS	TOTAL UNITS	LAND SQUARE FEET	GROSS SQUARE FEET	YEAR BUILT	TAX CLASS AT TIME OF SALE	BUILDING CLASS AT TIME OF SALE	SALE PRICE	SALE DATE
0	ALPHABET CITY	02 TWO FAMILY HOMES	1	375	32		B9	746 EAST 6 STREET		10009.0	2	0	2	2134.0	3542.0	1899.0	1	B9	1800000.0	2003-01-22
1	ALPHABET CITY	07 RENTALS - WALKUP APARTMENTS	2A	372	31		C3	316 EAST 3 STREET		10009.0	4	0	4	5746.0	2700.0	1900.0	2	C3	NaN	2003-12-18
2	ALPHABET CITY	07 RENTALS - WALKUP APARTMENTS	2A	372	31		C3	316 EAST 3 STREET		10009.0	4	0	4	5746.0	2700.0	1900.0	2	C3	NaN	2003-12-18
3	ALPHABET CITY	07 RENTALS - WALKUP APARTMENTS	2B	378	33		C4	125 AVENUE D		10009.0	6	1	7	2185.0	5725.0	1910.0	2	C4	426000.0	2003-10-23
4	ALPHABET CITY	07 RENTALS - WALKUP APARTMENTS	2	391	12		C1	610 EAST 9 STREET		10009.0	24	0	24	2543.0	11568.0	1910.0	2	C1	NaN	2003-02-28

In [62]:
```
tmp = df_processed[df_processed['SALE DATE'].dt.year == 2003]
tmp
```

Out[62]:

	NEIGHBORHOOD	BUILDING CLASS CATEGORY	TAX CLASS AT PRESENT	BLOCK	LOT	EASE-MENT	BUILDING CLASS AT PRESENT	ADDRESS	APARTMENT NUMBER	ZIP CODE	RESIDENTIAL UNITS	COMMERCIAL UNITS	TOTAL UNITS	LAND SQUARE FEET	GROSS SQUARE FEET	YEAR BUILT	TAX CLASS AT TIME OF SALE	BUILDING CLASS AT TIME OF SALE	SALE PRICE	SALE DATE
0	ALPHABET CITY	02 TWO FAMILY HOMES	1	375	32		B9	746 EAST 6 STREET		10009.0	2	0	2	2134.0	3542.0	1899.0	1	B9	1800000.0	2003-01-22
1	ALPHABET CITY	07 RENTALS - WALKUP APARTMENTS	2A	372	31		C3	316 EAST 3 STREET		10009.0	4	0	4	5746.0	2700.0	1900.0	2	C3	NaN	2003-12-18
2	ALPHABET CITY	07 RENTALS - WALKUP APARTMENTS	2A	372	31		C3	316 EAST 3 STREET		10009.0	4	0	4	5746.0	2700.0	1900.0	2	C3	NaN	2003-12-18
3	ALPHABET CITY	07 RENTALS - WALKUP APARTMENTS	2B	378	33		C4	125 AVENUE D		10009.0	6	1	7	2185.0	5725.0	1910.0	2	C4	426000.0	2003-10-23
4	ALPHABET CITY	07 RENTALS - WALKUP APARTMENTS	2	391	12		C1	610 EAST 9 STREET		10009.0	24	0	24	2543.0	11568.0	1910.0	2	C1	NaN	2003-02-28
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
22205	WASHINGTON HEIGHTS UPPER	14 RENTALS - 4-10 UNIT	2A	2166	53		S5	603 WEST 185 STREET		10033.0	5	1	6	1450.0	5050.0	1911.0	2	S5	NaN	2003-09-28
22206	WASHINGTON HEIGHTS UPPER	22 STORE BUILDINGS	4	2180	117		K1	4311-13 BROADWAY		10033.0	0	2	2	3475.0	2722.0	1956.0	4	K1	900000.0	2003-06-23
22207	WASHINGTON HEIGHTS UPPER	29 COMMERCIAL GARAGES	4	2167	1		G9	4320 BROADWAY		10033.0	0	2	2	21133.0	72608.0	2004.0	4	G2	700000.0	2003-12-04
22208	WASHINGTON HEIGHTS UPPER	29 COMMERCIAL GARAGES	4	2167	1		G9	4320 BROADWAY		10033.0	0	2	2	21133.0	72608.0	2004.0	4	G2	700000.0	2003-12-04
22209	WASHINGTON HEIGHTS UPPER	29 COMMERCIAL GARAGES	4	2180	652		G9	4525 BROADWAY		10040.0	0	1	1	10075.0	2443.0	1950.0	4	G2	335000.0	2003-10-16

22210 rows × 20 columns

In [63]:
```
df_processed['SALE PRICE WITH INFLATION'] = np.nan
df_processed.head()
```

Out[63]:

	NEIGHBORHOOD	BUILDING CLASS CATEGORY	TAX CLASS AT PRESENT	BLOCK	LOT	EASE-MENT	BUILDING CLASS AT PRESENT	ADDRESS	APARTMENT NUMBER	ZIP CODE	...	COMMERCIAL UNITS	TOTAL UNITS	LAND SQUARE FEET	GROSS SQUARE FEET	YEAR BUILT	TAX CLASS AT TIME OF SALE	BUILDING CLASS AT TIME OF SALE	SALE PRICE	SALE DATE	SALE PRICE WITH INFLATION
0	ALPHABET CITY	02 TWO FAMILY HOMES	1	375	32		B9	746 EAST 6 STREET		10009.0	...	0	2	2134.0	3542.0	1899.0	1	B9	1800000.0	2003-01-22	NaN
1	ALPHABET CITY	07 RENTALS - WALKUP APARTMENTS	2A	372	31		C3	316 EAST 3 STREET		10009.0	...	0	4	5746.0	2700.0	1900.0	2	C3	NaN	2003-12-18	NaN
2	ALPHABET CITY	07 RENTALS - WALKUP APARTMENTS	2A	372	31		C3	316 EAST 3 STREET		10009.0	...	0	4	5746.0	2700.0	1900.0	2	C3	NaN	2003-12-18	NaN
3	ALPHABET CITY	07 RENTALS - WALKUP APARTMENTS	2B	378	33		C4	125 AVENUE D		10009.0	...	1	7	2185.0	5725.0	1910.0	2	C4	426000.0	2003-10-23	NaN
4	ALPHABET CITY	07 RENTALS - WALKUP APARTMENTS	2	391	12		C1	610 EAST 9 STREET		10009.0	...	0	24	2543.0	11568.0	1910.0	2	C1	NaN	2003-02-28	NaN

5 rows × 21 columns

In [64]:
```
def inflation_add (row):
    if row['SALE PRICE'] == np.nan :
        return np.nan
    if row['SALE DATE'].year == 2003 :
        return row['SALE PRICE'] * 1.37
    if row['SALE DATE'].year == 2004 :
        return row['SALE PRICE'] * 1.33
    if row['SALE DATE'].year == 2005 :
        return row['SALE PRICE'] * 1.29
    if row['SALE DATE'].year == 2006 :
        return row['SALE PRICE'] * 1.25
    if row['SALE DATE'].year == 2007 :
        return row['SALE PRICE'] * 1.21
    if row['SALE DATE'].year == 2008 :
        return row['SALE PRICE'] * 1.17
    if row['SALE DATE'].year == 2009 :
        return row['SALE PRICE'] * 1.17
    if row['SALE DATE'].year == 2010 :
        return row['SALE PRICE'] * 1.15
    if row['SALE DATE'].year == 2011 :
        return row['SALE PRICE'] * 1.12
    if row['SALE DATE'].year == 2012 :
        return row['SALE PRICE'] * 1.09
    if row['SALE DATE'].year == 2013 :
        return row['SALE PRICE'] * 1.08
    if row['SALE DATE'].year == 2014 :
        return row['SALE PRICE'] * 1.06
    if row['SALE DATE'].year == 2015 :
        return row['SALE PRICE'] * 1.06
    if row['SALE DATE'].year == 2016 :
        return row['SALE PRICE'] * 1.05
    if row['SALE DATE'].year == 2017 :
        return row['SALE PRICE'] * 1.02
    if row['SALE DATE'].year == 2018 :
        return row['SALE PRICE'] * 1.00
    return np.nan
```

In [65]:
```
df_processed['SALE PRICE WITH INFLATION'] = df_processed.apply (lambda row: inflation_add(row), axis=1)
```

In [66]:
```
df_processed.head()
```

Out[66]:

	NEIGHBORHOOD	BUILDING CLASS CATEGORY	TAX CLASS AT PRESENT	BLOCK	LOT	EASE-MENT	BUILDING CLASS AT PRESENT	ADDRESS	APARTMENT NUMBER	ZIP CODE	...	COMMERCIAL UNITS	TOTAL UNITS	LAND SQUARE FEET	GROSS SQUARE FEET	YEAR BUILT	TAX CLASS AT TIME OF SALE	BUILDING CLASS AT TIME OF SALE	SALE PRICE	SALE DATE	SALE PRICE WITH INFLATION
0	ALPHABET CITY	02 TWO FAMILY HOMES	1	375	32		B9	746 EAST 6 STREET		10009.0	...	0	2	2134.0	3542.0	1899.0	1	B9	1800000.0	2003-01-22	2466000.0
1	ALPHABET CITY	07 RENTALS - WALKUP APARTMENTS	2A	372	31		C3	316 EAST 3 STREET		10009.0	...	0	4	5746.0	2700.0	1900.0	2	C3	NaN	2003-12-18	NaN
2	ALPHABET CITY	07 RENTALS - WALKUP APARTMENTS	2A	372	31		C3	316 EAST 3 STREET		10009.0	...	0	4	5746.0	2700.0	1900.0	2	C3	NaN	2003-12-18	NaN
3	ALPHABET CITY	07 RENTALS - WALKUP APARTMENTS	2B	378	33		C4	125 AVENUE D		10009.0	...	1	7	2185.0	5725.0	1910.0	2	C4	426000.0	2003-10-23	583620.0
4	ALPHABET CITY	07 RENTALS - WALKUP APARTMENTS	2	391	12		C1	610 EAST 9 STREET		10009.0	...	0	24	2543.0	11568.0	1910.0	2	C1	NaN	2003-02-28	NaN

5 rows × 21 columns

In [67]:
```python
filtered = df_processed[df_processed['SALE PRICE WITH INFLATION'].notna()]
filtered = tmp[tmp['BLOCK'] > 0]
filtered
```

Out[67]:

	NEIGHBORHOOD	BUILDING CLASS CATEGORY	TAX CLASS AT PRESENT	BLOCK	LOT	EASE-MENT	BUILDING CLASS AT PRESENT	ADDRESS	APARTMENT NUMBER	ZIP CODE	RESIDENTIAL UNITS	COMMERCIAL UNITS	TOTAL UNITS	LAND SQUARE FEET	GROSS SQUARE FEET	YEAR BUILT	TAX CLASS AT TIME OF SALE	BUILDING CLASS AT TIME OF SALE	SALE PRICE	SALE DATE
0	ALPHABET CITY	02 TWO FAMILY HOMES	1	375	32		B9	746 EAST 6 STREET		10009.0	2	0	2	2134.0	3542.0	1899.0	1	B9	1800000.0	2003-01-22
1	ALPHABET CITY	07 RENTALS - WALKUP APARTMENTS	2A	372	31		C3	316 EAST 3 STREET		10009.0	4	0	4	5746.0	2700.0	1900.0	2	C3	NaN	2003-12-18
2	ALPHABET CITY	07 RENTALS - WALKUP APARTMENTS	2A	372	31		C3	316 EAST 3 STREET		10009.0	4	0	4	5746.0	2700.0	1900.0	2	C3	NaN	2003-12-18
3	ALPHABET CITY	07 RENTALS - WALKUP APARTMENTS	2B	378	33		C4	125 AVENUE D		10009.0	6	1	7	2185.0	5725.0	1910.0	2	C4	426000.0	2003-10-23
4	ALPHABET CITY	07 RENTALS - WALKUP APARTMENTS	2	391	12		C1	610 EAST 9 STREET		10009.0	24	0	24	2543.0	11568.0	1910.0	2	C1	NaN	2003-02-28
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
22205	WASHINGTON HEIGHTS UPPER	14 RENTALS - 4-10 UNIT	2A	2166	53		S5	603 WEST 185 STREET		10033.0	5	1	6	1450.0	5050.0	1911.0	2	S5	NaN	2003-09-28
22206	WASHINGTON HEIGHTS UPPER	22 STORE BUILDINGS	4	2180	117		K1	4311-13 BROADWAY		10033.0	0	2	2	3475.0	2722.0	1956.0	4	K1	900000.0	2003-06-23
22207	WASHINGTON HEIGHTS UPPER	29 COMMERCIAL GARAGES	4	2167	1		G9	4320 BROADWAY		10033.0	0	2	2	21133.0	72608.0	2004.0	4	G2	700000.0	2003-12-04
22208	WASHINGTON HEIGHTS UPPER	29 COMMERCIAL GARAGES	4	2167	1		G9	4320 BROADWAY		10033.0	0	2	2	21133.0	72608.0	2004.0	4	G2	700000.0	2003-12-04
22209	WASHINGTON HEIGHTS UPPER	29 COMMERCIAL GARAGES	4	2180	652		G9	4525 BROADWAY		10040.0	0	1	1	10075.0	2443.0	1950.0	4	G2	335000.0	2003-10-16

22210 rows × 20 columns

In [68]:
```python
nyc_neighbourhoods_map = json.load(
    open("data/manhattan_neighbourhoods.geojson"))
nyc_neighbourhoods_map = helpers.remove_non_manhattan_neighbourhoods(
    nyc_neighbourhoods_map)
nyc_neighbourhoods = [x for x in nyc_neighbourhoods_map['features']]
# print(nyc_neighbourhoods[0])

nycmap = json.load(open("data/manhattan.geojson"))
# print(filtered['BLOCK'].unique())
```

In [69]:
```python
i = 1
for a in nyc_neighbourhoods:
#     print(i)
    i= i+1
    print(a['properties']['ntaname'])
    print(len(a['geometry']['coordinates']))
```

```
Upper West Side
1
Gramercy
1
Midtown-Midtown South
1
Hudson Yards-Chelsea-Flatiron-Union Square
1
Clinton
1
Yorkville
2
Marble Hill-Inwood
3
Morningside Heights
1
Central Harlem South
1
Chinatown
1
SoHo-TriBeCa-Civic Center-Little Italy
1
Battery Park City-Lower Manhattan
8
Lower East Side
8
East Harlem South
1
Manhattanville
1
Lincoln Square
1
Turtle Bay-East Midtown
3
Upper East Side-Carnegie Hill
1
Central Harlem North-Polo Grounds
1
Hamilton Heights
1
East Village
1
West Village
1
Washington Heights South
1
Washington Heights North
1
Murray Hill-Kips Bay
2
Stuyvesant Town-Cooper Village
3
Lenox Hill-Roosevelt Island
4
East Harlem North
5
```

In [70]:
```python
df_processed['GEOJSON NEIGHBORHOOD'] = np.nan
df_processed.head()
```

Out[70]:

	NEIGHBORHOOD	BUILDING CLASS CATEGORY	TAX CLASS AT PRESENT	BLOCK	LOT	EASE-MENT	BUILDING CLASS AT PRESENT	ADDRESS	APARTMENT NUMBER	ZIP CODE	...	TOTAL UNITS	LAND SQUARE FEET	GROSS SQUARE FEET	YEAR BUILT	TAX CLASS AT TIME OF SALE	BUILDING CLASS AT TIME OF SALE	SALE PRICE	SALE DATE	SALE PRICE WITH INFLATION	GEOJSON NEIGHBORHOOD
0	ALPHABET CITY	02 TWO FAMILY HOMES	1	375	32		B9	746 EAST 6 STREET		10009.0	...	2	2134.0	3542.0	1899.0	1	B9	1800000.0	2003-01-22	2466000.0	NaN
1	ALPHABET CITY	07 RENTALS - WALKUP APARTMENTS	2A	372	31		C3	316 EAST 3 STREET		10009.0	...	4	5746.0	2700.0	1900.0	2	C3	NaN	2003-12-18	NaN	NaN
2	ALPHABET CITY	07 RENTALS - WALKUP APARTMENTS	2A	372	31		C3	316 EAST 3 STREET		10009.0	...	4	5746.0	2700.0	1900.0	2	C3	NaN	2003-12-18	NaN	NaN
3	ALPHABET CITY	07 RENTALS - WALKUP APARTMENTS	2B	378	33		C4	125 AVENUE D		10009.0	...	7	2185.0	5725.0	1910.0	2	C4	426000.0	2003-10-23	583620.0	NaN
4	ALPHABET CITY	07 RENTALS - WALKUP APARTMENTS	2	391	12		C1	610 EAST 9 STREET		10009.0	...	24	2543.0	11568.0	1910.0	2	C1	NaN	2003-02-28	NaN	NaN

5 rows × 22 columns

In [71]:
```python
def is_point_in_polygon(point, coords):
    bbPath = mplPath.Path(np.array(coords))
    return bbPath.contains_point(point)
```

In [72]:
```python
import matplotlib.path as mplPath

blocks_unique = filtered['BLOCK'].unique()

for block in blocks_unique:
    particular_block_arr = [x for x in nycmap['features'] if x['properties']['block'] == block]
    if(len(particular_block_arr) < 1):
        continue;

    particular_block = particular_block_arr[0]
    block_coords = particular_block['geometry']['coordinates']
    point = block_coords[0][0][0]

    found = 0

#     for neigh in nyc_neighbourhoods:
#         if(is_point_in_polygon(point,neigh['geometry']['coordinates'][0][0])):
# #             df_processed.loc[df_processed["BLOCK"] == block, "GEOJSON NEIGHBORHOOD"] = neigh['properties']['ntaname']
#             found = 1
#             break;

    for neigh in nyc_neighbourhoods:
        for coords in neigh['geometry']['coordinates']:
            if(is_point_in_polygon(point, coords[0])):
                df_processed.loc[df_processed["BLOCK"] == block, "GEOJSON NEIGHBORHOOD"] = neigh['properties']['ntaname']
                found = 1
                break
        if(found == 1):
            break

    if(found == 0):
        print(block)
```

```
969
1371
1372
```

In [73]:
```python
aaa = df_processed[df_processed['BLOCK'] == 1645]
aaa

print(nyc_neighbourhoods[27])
```


```
6, 40.803573947208385], [-73.93006020693849, 40.80336164816054], [-73.93001261302967, 40.80328749842878], [-73.92997373618644, 40.80319646747155], [-73.92997862177, 40.80319024488239], [-
73.93004460904999, 40.80310611177073], [-73.93000931090499, 40.80304013465469], [-73.9299828842777, 40.802984752494254], [-73.92991008609896, 40.802864550747465], [-73.92988948082969, 40.
80283035745909], [-73.92978668000154, 40.80266074871753], [-73.92969251214595, 40.80244028594797], [-73.92957693839507, 40.80218002521569], [-73.9295204392848, 40.80205347961556], [-73.92
945366333552, 40.80190736833346], [-73.9294365411877, 40.801861711857434], [-73.92935435998567, 40.80167125043791], [-73.9293312697181, 40.801598853990896], [-73.92930220396839, 40.801495
80134899], [-73.92927487688036, 40.80137319023274], [-73.92920817595964, 40.80114752061129], [-73.92911838814229, 40.80111300322634], [-73.9290349024031, 40.8010809038519], [-73.929026714
06671, 40.80096986031432], [-73.92902123584643, 40.8008955725617], [-73.92901644642042, 40.800830715542126], [-73.92900788120605, 40.80077234912538], [-73.92899994866335, 40.8007100976927
3], [-73.92898618762995, 40.8006021231194], [-73.92895348657615, 40.80030936798214], [-73.92893763668594, 40.800170843777686], [-73.92890152344224, 40.79985521416015], [-73.9289194251577
9, 40.799458186491535], [-73.92894396946961, 40.798911588935496], [-73.92897121002346, 40.798258860579764], [-73.92903640179625, 40.796762594101295], [-73.92905461028747, 40.796694433867
4], [-73.92927936741171, 40.79585313952064], [-73.92930773275859, 40.79581025745305], [-73.92953058056908, 40.795507798349284], [-73.92968294251133, 40.79529623376313], [-73.9297459469162
1, 40.795208748234096], [-73.92987485778265, 40.79503515622577], [-73.92984244753451, 40.79501503785595], [-73.92985144395378, 40.79500139992602], [-73.92980955349988, 40.79498316629674
5], [-73.92983254042525, 40.79494906114083], [-73.92988740651919, 40.794967292603964], [-73.92990139788401, 40.794946824362505], [-73.929916361206, 40.79494911213707], [-73.9299373533455
2, 40.794918786284455], [-73.92992039501154, 40.794909671827526], [-73.93006881035538, 40.794700828605485], [-73.93010771028146, 40.79471299106374], [-73.93019664392948, 40.7945932238497
1], [-73.9304198047997, 40.79447087234861], [-73.93111408045336, 40.794096501343866], [-73.93142758557963, 40.793927988382855], [-73.93165821105745, 40.7938017404392], [-73.9317929252035
5, 40.79372796128848], [-73.93187500171743, 40.793681921843714], [-73.93201001457187, 40.793605486169895], [-73.93207886021811, 40.793566508331295], [-73.93235128869082, 40.7934138451512
8], [-73.93250920949684, 40.79333190154458], [-73.93253852654895, 40.79331668213542], [-73.93267239130897, 40.793248232772896], [-73.93287712655092, 40.793130251367884], [-73.933106832078
07, 40.79299532820954], [-73.93324587394645, 40.79291429602916], [-73.93327132021129, 40.792899463351674], [-73.93348489644698, 40.79277417082997], [-73.93372790503639, 40.7926014778685
2], [-73.93388785265701, 40.79248947701711], [-73.93406997726619, 40.79235829327037], [-73.93427221714605, 40.792217315797885], [-73.93434032786627, 40.79216796805462], [-73.9345104254863
4, 40.792051294646434], [-73.93473881584507, 40.79190076820261], [-73.93505406057963, 40.791686231358476], [-73.93508309599491, 40.79170961696516], [-73.93510751593803, 40.7917291994914
3], [-73.93513822195999, 40.791752834631886], [-73.93519360532012, 40.7917979990583], [-73.9352406084498, 40.791833875011065], [-73.93529038127491, 40.79187313717656], [-73.9353992081248
```

In [74]:
```
bbb = df_processed['GEOJSON NEIGHBORHOOD'].unique()
bbb
```

Out[74]:
```
array(['Lower East Side', 'East Village',
       'Hudson Yards-Chelsea-Flatiron-Union Square',
       'Midtown-Midtown South', 'Chinatown',
       'SoHo-TriBeCa-Civic Center-Little Italy', 'Clinton',
       'Battery Park City-Lower Manhattan', 'Gramercy',
       'Murray Hill-Kips Bay', 'West Village', 'Central Harlem South',
       'Central Harlem North-Polo Grounds', 'Hamilton Heights',
       'East Harlem North', 'Manhattanville', 'Morningside Heights',
       'East Harlem South', 'Marble Hill-Inwood',
       'Turtle Bay-East Midtown', nan, 'Upper West Side',
       'Lincoln Square', 'Upper East Side-Carnegie Hill',
       'Lenox Hill-Roosevelt Island', 'Yorkville',
       'Washington Heights South', 'Washington Heights North'],
      dtype=object)
```

In [75]:
```
neigh_map_df = df_processed[df_processed['GEOJSON NEIGHBORHOOD'].notna()]
neigh_map_df = neigh_map_df[neigh_map_df['SALE PRICE WITH INFLATION'].notna()]

grouped = neigh_map_df.groupby(['GEOJSON NEIGHBORHOOD'])['SALE PRICE WITH INFLATION'].mean().reset_index()
grouped['GEOJSON NEIGHBORHOOD']
```

Out[75]:
```
0              Battery Park City-Lower Manhattan
1              Central Harlem North-Polo Grounds
2                           Central Harlem South
3                                      Chinatown
4                                        Clinton
5                              East Harlem North
6                              East Harlem South
7                                   East Village
8                                       Gramercy
9                               Hamilton Heights
10    Hudson Yards-Chelsea-Flatiron-Union Square
11                   Lenox Hill-Roosevelt Island
12                                Lincoln Square
13                               Lower East Side
14                                Manhattanville
15                            Marble Hill-Inwood
16                         Midtown-Midtown South
17                           Morningside Heights
18                          Murray Hill-Kips Bay
19        SoHo-TriBeCa-Civic Center-Little Italy
20                       Turtle Bay-East Midtown
21                 Upper East Side-Carnegie Hill
22                               Upper West Side
23                      Washington Heights North
24                      Washington Heights South
25                                  West Village
26                                     Yorkville
Name: GEOJSON NEIGHBORHOOD, dtype: object
```

```
In [76]: fig = px.choropleth_mapbox(grouped,
                                    geojson=nyc_neighbourhoods_map,
                                    locations="GEOJSON NEIGHBORHOOD",
                                    featureidkey="properties.ntaname",
                                     color="SALE PRICE WITH INFLATION",
                                    color_continuous_scale=px.colors.sequential.thermal[::-1],
                                 #    range_color=(0, 0.5),
                                 #    animation_frame="SALE DATE",
                                    mapbox_style="carto-positron",
                                    zoom=9, center={"lat": 40.7, "lon": -73.7},
                                    opacity=0.7,
                                    hover_name="GEOJSON NEIGHBORHOOD"
                                    )

         fig.show()
```

In [77]:
```
df_processed.sample(5)
```

Out[77]:

	NEIGHBORHOOD	BUILDING CLASS CATEGORY	TAX CLASS AT PRESENT	BLOCK	LOT	EASE-MENT	BUILDING CLASS AT PRESENT	ADDRESS	APARTMENT NUMBER	ZIP CODE	...	TOTAL UNITS	LAND SQUARE FEET	GROSS SQUARE FEET	YEAR BUILT	TAX CLASS AT TIME OF SALE	BUILDING CLASS AT TIME OF SALE	SALE PRICE	SALE DATE	SALE PRICE WITH INFLATION	GEOJSON NEIGHBORHOOD
9633	MIDTOWN WEST	13 CONDOS - ELEVATOR APARTMENTS	2	1027	1561		R4	230 WEST 56 STREET	58A	10019.0	...	1	NaN	NaN	NaN	2	R4	4079790.0	2003-05-08	5589312.3	Midtown-Midtown South
9773	MIDTOWN EAST	13 CONDOS - ELEVATOR APARTMENTS	2	1344	1031		R4	335 EAST 51ST STREET	4A	10022.0	...	1	NaN	NaN	NaN	2	R4	NaN	2013-02-12	NaN	Turtle Bay-East Midtown
15518	MIDTOWN WEST	28 COMMERCIAL CONDOS	4	1027	1375		R5	870 7 AVENUE	1212	10019.0	...	1	NaN	NaN	NaN	4	R5	17272.0	2006-02-02	21590.0	Midtown-Midtown South
11243	MIDTOWN WEST	13 CONDOS - ELEVATOR APARTMENTS	2	1047	1344		R4	333 WEST 56TH STREET	2L	10019.0	...	1	NaN	NaN	1931.0	2	R4	485000.0	2005-03-01	625650.0	Clinton
5647	HARLEM-UPPER	09 COOPS - WALKUP APARTMENTS	2	2066	54		C6	454 WEST 152ND STREET, 32		10031.0	...	0	NaN	NaN	1926.0	2	C6	80000.0	2011-08-09	89600.0	Hamilton Heights

5 rows × 22 columns

In [78]:
```python
tmp = df_processed[df_processed['TAX CLASS AT TIME OF SALE'] == 2]
tmp = tmp[tmp['TOTAL UNITS'] == 1]
tmp = tmp[tmp['SALE PRICE WITH INFLATION'].notna()]
tmp = tmp[tmp['SALE PRICE WITH INFLATION'] > 10000]
tmp = tmp[tmp['SALE PRICE WITH INFLATION'] > 100000000]
tmp
```

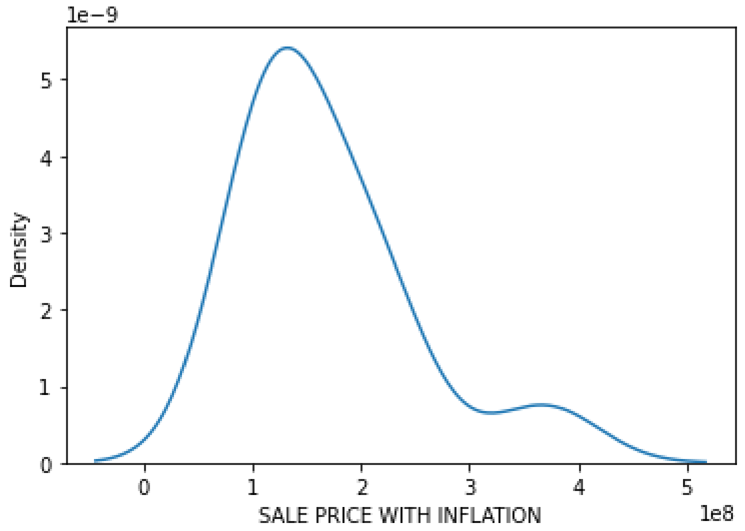
Out[78]:

	NEIGHBORHOOD	BUILDING CLASS CATEGORY	TAX CLASS AT PRESENT	BLOCK	LOT	EASE-MENT	BUILDING CLASS AT PRESENT	ADDRESS	APARTMENT NUMBER	ZIP CODE	...	TOTAL UNITS	LAND SQUARE FEET	GROSS SQUARE FEET	YEAR BUILT	TAX CLASS AT TIME OF SALE	BUILDING CLASS AT TIME OF SALE	SALE PRICE	SALE DATE	SALE PRICE WITH INFLATION	GEOJSON NEIGHBORHOOD
18330	UPPER WEST SIDE	13 CONDOS - ELEVATOR APARTMENTS	4	1116	1701		R5	15 WEST 63 STREET	STO/A	10023.0	...	1	NaN	NaN	NaN	2	R4	90000000.0	2003-11-05	1.233000e+08	Lincoln Square
18331	UPPER WEST SIDE	13 CONDOS - ELEVATOR APARTMENTS	4	1116	1702		R5	15 WEST 63 STREET	STO/B	10023.0	...	1	NaN	NaN	NaN	2	R4	90000000.0	2003-11-05	1.233000e+08	Lincoln Square
18332	UPPER WEST SIDE	13 CONDOS - ELEVATOR APARTMENTS	4	1116	1703		R5	15 WEST 63 STREET	STO/3	10023.0	...	1	NaN	NaN	NaN	2	R4	90000000.0	2003-11-05	1.233000e+08	Lincoln Square
1220	CIVIC CENTER	13 CONDOS - ELEVATOR APARTMENTS	2	135	1227		R4	269-71 BROADWAY	12A	10007.0	...	1	NaN	NaN	1910.0	2	R4	155283125.0	2004-05-26	2.065266e+08	SoHo-TriBeCa-Civic Center-Little Italy
1302	CLINTON	16 CONDOS - 2-10 UNIT WITH COMMERCIAL UNIT	4	1082	1001		R5	770 11TH AVENUE	MBU	10019.0	...	1	NaN	NaN	2009.0	2	R8	189734983.0	2010-07-26	2.181952e+08	NaN
1729	FINANCIAL	13 CONDOS - ELEVATOR APARTMENTS	2	25	1401		R4	15 WILLIAM STREET	3M	10004.0	...	1	NaN	NaN	2005.0	2	R4	184414937.0	2010-12-01	2.120772e+08	Battery Park City-Lower Manhattan
25621	UPPER WEST SIDE	13 CONDOS - ELEVATOR APARTMENTS	2	1852	1101		R4	808 COLUMBUS AVENUE	RU	10025.0	...	1	NaN	NaN	2007.0	2	R4	339326606.0	2012-01-11	3.698660e+08	NaN
7230	KIPS BAY	13 CONDOS - ELEVATOR APARTMENTS	2	943	1002		R4	330 EAST 38TH STREET	5A	10016.0	...	1	NaN	NaN	1989.0	2	R4	147000000.0	2014-08-14	1.558200e+08	Murray Hill-Kips Bay
10315	MIDTOWN WEST	13 CONDOS - ELEVATOR APARTMENTS	2	1010	1698		R4	157 WEST 57TH STREET	90	10019.0	...	1	NaN	NaN	2009.0	2	R4	100471452.0	2014-12-23	1.064997e+08	Midtown-Midtown South
13485	SOHO	16 CONDOS - 2-10 UNIT WITH COMMERCIAL UNIT	2C	500	1301		R8	139 SPRING STREET	1	10012.0	...	1	NaN	NaN	NaN	2	R8	115388000.0	2016-01-15	1.211574e+08	SoHo-TriBeCa-Civic Center-Little Italy
18726	UPPER WEST SIDE	13 CONDOS - ELEVATOR APARTMENTS	2	1170	1105		R4	2211 BROADWAY	3A	10024.0	...	1	NaN	NaN	1908.0	2	R4	98525704.0	2016-09-07	1.034520e+08	Upper West Side

11 rows × 22 columns

In [79]:
```python
sns.kdeplot(tmp['SALE PRICE WITH INFLATION'])
```

Out[79]: <AxesSubplot:xlabel='SALE PRICE WITH INFLATION', ylabel='Density'>



In [80]:
```python
tmp['SALE PRICE WITH INFLATION'].describe()
```

Out[80]:
```
count    1.100000e+01
mean     1.694086e+08
std      7.934752e+07
min      1.034520e+08
25%      1.222287e+08
50%      1.233000e+08
75%      2.093019e+08
max      3.698660e+08
Name: SALE PRICE WITH INFLATION, dtype: float64
```

In [81]:
```python
with_prices = df_processed[df_processed['SALE PRICE WITH INFLATION'].notna()]
with_prices = with_prices[with_prices['BLOCK'] != 0]
with_prices = with_prices[with_prices['LOT'] != 0]
block_lot = (with_prices['BLOCK'].astype(str) + ' ' + with_prices['LOT'].astype(str)).mode()
block_lot
```

Out[81]:
```
0    1009 37
dtype: object
```

In [82]:
```python
most_frequent_lot = with_prices[with_prices['LOT'] == 37]
most_frequent_lot = most_frequent_lot[most_frequent_lot['BLOCK'] == 1009]
most_frequent_lot = most_frequent_lot[most_frequent_lot['SALE PRICE WITH INFLATION'] < 50000000]
most_frequent_lot
```

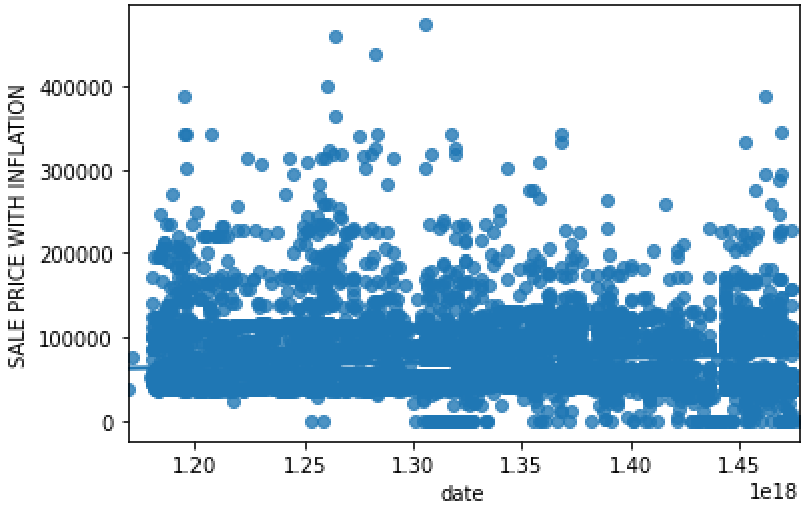
Out[82]:

	NEIGHBORHOOD	BUILDING CLASS CATEGORY	TAX CLASS AT PRESENT	BLOCK	LOT	EASE-MENT	BUILDING CLASS AT PRESENT	ADDRESS	APARTMENT NUMBER	ZIP CODE	...	TOTAL UNITS	LAND SQUARE FEET	GROSS SQUARE FEET	YEAR BUILT	TAX CLASS AT TIME OF SALE	BUILDING CLASS AT TIME OF SALE	SALE PRICE	SALE DATE	SALE PRICE WITH INFLATION	GEOJSON NEIGHBORHOOD
13714	MIDTOWN WEST	25 LUXURY HOTELS	4	1009	37		H2	102 WEST 57TH STREET		10019.0	...	2	7532.0	112850.0	2007.0	4	H2	158380.0	2007-12-31	191639.80	Midtown-Midtown South
13715	MIDTOWN WEST	25 LUXURY HOTELS	4	1009	37		H2	102 WEST 57TH STREET		10019.0	...	2	7532.0	112850.0	2007.0	4	H2	96900.0	2007-12-31	117249.00	Midtown-Midtown South
13716	MIDTOWN WEST	25 LUXURY HOTELS	4	1009	37		H2	102 WEST 57TH STREET		10019.0	...	2	7532.0	112850.0	2007.0	4	H2	87800.0	2007-12-31	106238.00	Midtown-Midtown South
13717	MIDTOWN WEST	25 LUXURY HOTELS	4	1009	37		H2	102 WEST 57TH STREET		10019.0	...	2	7532.0	112850.0	2007.0	4	H2	58900.0	2007-12-31	71269.00	Midtown-Midtown South
13718	MIDTOWN WEST	25 LUXURY HOTELS	4	1009	37		H2	102 WEST 57TH STREET		10019.0	...	2	7532.0	112850.0	2007.0	4	H2	43900.0	2007-12-31	53119.00	Midtown-Midtown South
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
11167	MIDTOWN WEST	26 OTHER HOTELS	4	1009	37		H3	102 WEST 57TH STREET		10019.0	...	2	7532.0	112850.0	2007.0	4	H3	57594.0	2016-01-13	60473.70	Midtown-Midtown South
11168	MIDTOWN WEST	26 OTHER HOTELS	4	1009	37		H3	102 WEST 57TH STREET		10019.0	...	2	7532.0	112850.0	2007.0	4	H3	83305.0	2016-01-13	87470.25	Midtown-Midtown South
11169	MIDTOWN WEST	26 OTHER HOTELS	4	1009	37		H3	102 WEST 57TH		10019.0	...	2	7532.0	112850.0	2007.0	4	H3	23186.0	2016-01-12	24345.30	Midtown-Midtown South
11170	MIDTOWN WEST	26 OTHER HOTELS	4	1009	37		H3	102 WEST 57TH		10019.0	...	2	7532.0	112850.0	2007.0	4	H3	42988.0	2016-01-12	45137.40	Midtown-Midtown South
11178	MIDTOWN WEST	26 OTHER HOTELS	4	1009	37		H3	102 WEST 57TH STREET		10019.0	...	2	7532.0	112850.0	2007.0	4	H3	16239.0	2016-01-02	17050.95	Midtown-Midtown South

11465 rows × 22 columns

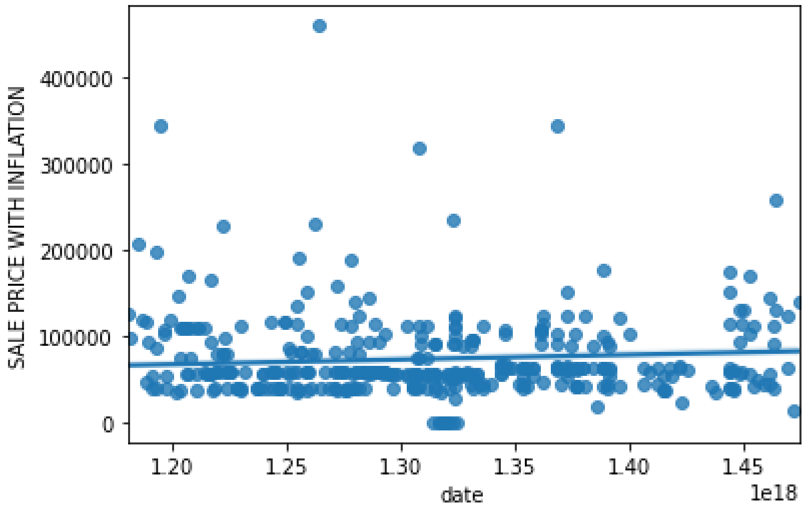
In [83]:
```
most_frequent_lot = most_frequent_lot.sort_values('SALE DATE')
most_frequent_lot['date'] = pd.to_numeric(most_frequent_lot['SALE DATE'])
sns.regplot(x=most_frequent_lot['date'], y=most_frequent_lot['SALE PRICE WITH INFLATION'])
```

Out[83]: <AxesSubplot:xlabel='date', ylabel='SALE PRICE WITH INFLATION'>



In [84]:
```
most_frequent_lot_sample = most_frequent_lot.sample(400)
sns.regplot(x=most_frequent_lot_sample['date'], y=most_frequent_lot_sample['SALE PRICE WITH INFLATION'])
```

Out[84]: <AxesSubplot:xlabel='date', ylabel='SALE PRICE WITH INFLATION'>



In [85]:
```python
most_frequent_app = df_processed[df_processed['SALE PRICE WITH INFLATION'].notna()]
# most_frequent_app = most_frequent_app[most_frequent_app['SALE PRICE WITH INFLATION'] < 20000000]
most_frequent_app = most_frequent_app[most_frequent_app['TAX CLASS AT TIME OF SALE'] == 2]
most_frequent_app = most_frequent_app[most_frequent_app['BLOCK'] != 0]
most_frequent_app = most_frequent_app[most_frequent_app['LOT'] != 0]
most_frequent_app = most_frequent_app[most_frequent_app['APARTMENT NUMBER'] != '']
most_frequent_app = most_frequent_app[most_frequent_app['APARTMENT NUMBER'] != 'TIMES']
block_lot_app = (most_frequent_app['BLOCK'].astype(str) + ' ' + most_frequent_app['LOT'].astype(str) + ' ' + most_frequent_app['APARTMENT NUMBER'].astype(str)).mode()
block_lot_app
```

Out[85]:
```
0    1171 1010 403
1    1290 1209 835
dtype: object
```

In [86]:
```
appartment = most_frequent_app[most_frequent_app['LOT'] == 1010]
appartment = appartment[appartment['BLOCK'] == 1171]
appartment = appartment[appartment['APARTMENT NUMBER'] == '403']
appartment
```

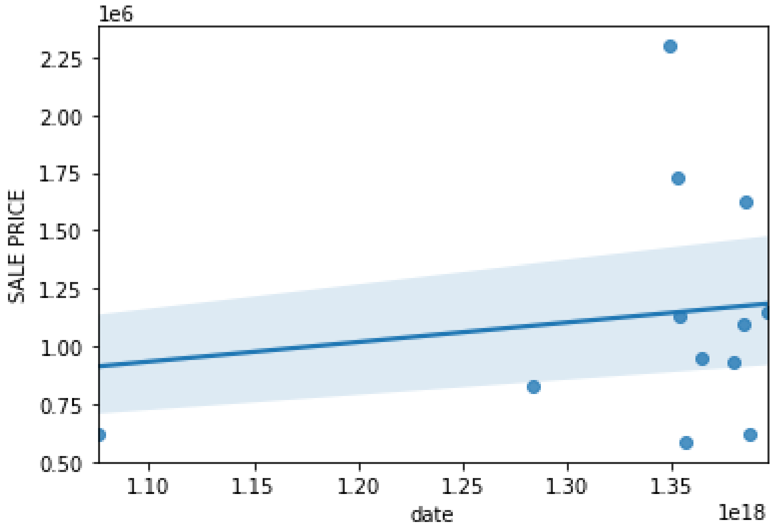
Out[86]:

	NEIGHBORHOOD	BUILDING CLASS CATEGORY	TAX CLASS AT PRESENT	BLOCK	LOT	EASE-MENT	BUILDING CLASS AT PRESENT	ADDRESS	APARTMENT NUMBER	ZIP CODE	...	TOTAL UNITS	LAND SQUARE FEET	GROSS SQUARE FEET	YEAR BUILT	TAX CLASS AT TIME OF SALE	BUILDING CLASS AT TIME OF SALE	SALE PRICE	SALE DATE	SALE PRICE WITH INFLATION	GEOJSON NEIGHBORHOOD
22617	UPPER WEST SIDE	13 CONDOS - ELEVATOR APARTMENTS	2	1171	1010		R4	NaN	403	10069.0	...	1	NaN	NaN	NaN	2	R4	615000.0	2004-01-28	817950.00	Lincoln Square
15631	UPPER WEST SIDE	13 CONDOS - ELEVATOR APARTMENTS	2	1171	1010		R4	200 RIVERSIDE BOULEVARD	403	10069.0	...	1	NaN	NaN	NaN	2	R4	825000.0	2010-09-02	948750.00	Lincoln Square
23653	UPPER WEST SIDE	13 CONDOS - ELEVATOR APARTMENTS	2	1171	1010		R4	120 RIVERSIDE BOULEVARD	403	10069.0	...	1	NaN	NaN	NaN	2	R4	585000.0	2012-12-26	637650.00	Lincoln Square
23655	UPPER WEST SIDE	13 CONDOS - ELEVATOR APARTMENTS	2	1171	1010		R4	200 RIVERSIDE BOULEVARD	403	10069.0	...	1	NaN	NaN	NaN	2	R4	1135000.0	2012-11-29	1237150.00	Lincoln Square
23656	UPPER WEST SIDE	13 CONDOS - ELEVATOR APARTMENTS	2	1171	1010		R4	200 RIVERSIDE BOULEVARD	403	10069.0	...	1	NaN	NaN	NaN	2	R4	1733333.0	2012-11-15	1889332.97	Lincoln Square
23658	UPPER WEST SIDE	13 CONDOS - ELEVATOR APARTMENTS	2	1171	1010		R4	80 RIVERSIDE BOULEVARD	403	10069.0	...	1	NaN	NaN	NaN	2	R4	2300000.0	2012-09-27	2507000.00	Lincoln Square
23843	UPPER WEST SIDE	13 CONDOS - ELEVATOR APARTMENTS	2	1171	1010		R4	240 RIVERSIDE BOULEVARD	403	10069.0	...	1	NaN	NaN	NaN	2	R4	620000.0	2013-12-23	669600.00	Lincoln Square
23844	UPPER WEST SIDE	13 CONDOS - ELEVATOR APARTMENTS	2	1171	1010		R4	100 RIVERSIDE BOULEVARD	403	10069.0	...	1	NaN	NaN	NaN	2	R4	1625000.0	2013-12-02	1755000.00	Lincoln Square
23845	UPPER WEST SIDE	13 CONDOS - ELEVATOR APARTMENTS	2	1171	1010		R4	120 RIVERSIDE BOULEVARD	403	10069.0	...	1	NaN	NaN	NaN	2	R4	1100000.0	2013-11-13	1188000.00	Lincoln Square
23846	UPPER WEST SIDE	13 CONDOS - ELEVATOR APARTMENTS	2	1171	1010		R4	200 RIVERSIDE BOULEVARD	403	10069.0	...	1	NaN	NaN	NaN	2	R4	935000.0	2013-09-17	1009800.00	Lincoln Square
23852	UPPER WEST SIDE	13 CONDOS - ELEVATOR APARTMENTS	2	1171	1010		R4	220 RIVERSIDE BOULEVARD	403	10069.0	...	1	NaN	NaN	NaN	2	R4	950000.0	2013-03-28	1026000.00	Lincoln Square
21829	UPPER WEST SIDE	13 CONDOS - ELEVATOR APARTMENTS	2	1171	1010		R4	100 RIVERSIDE BLVD.	403	10069.0	...	1	NaN	NaN	NaN	2	R4	1150000.0	2014-04-02	1219000.00	Lincoln Square

12 rows × 22 columns

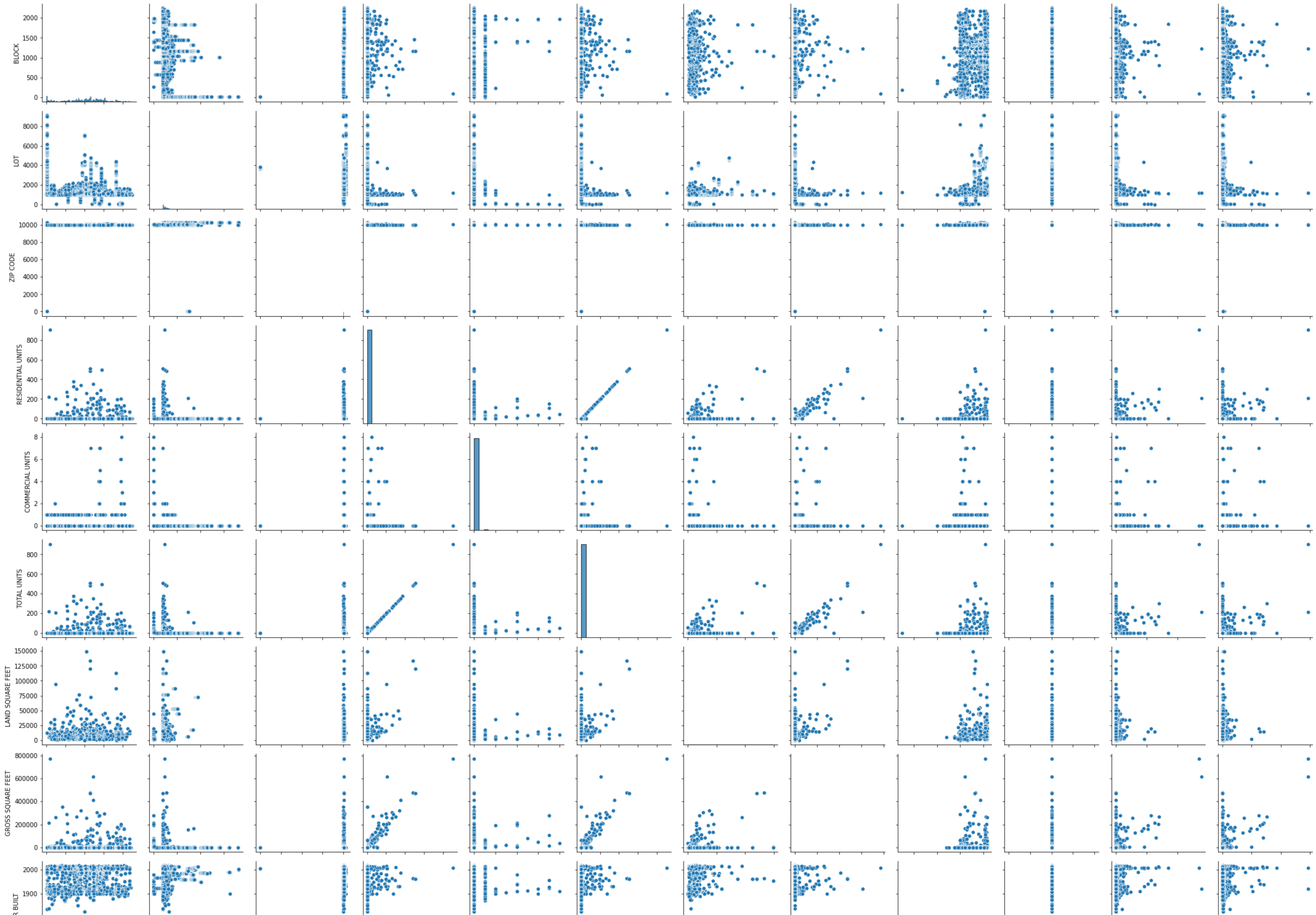
In [87]:
```
appartment = appartment.sort_values('SALE DATE')
appartment['date'] = pd.to_numeric(appartment['SALE DATE'])
sns.regplot(x=appartment['date'], y=appartment['SALE PRICE'])
```

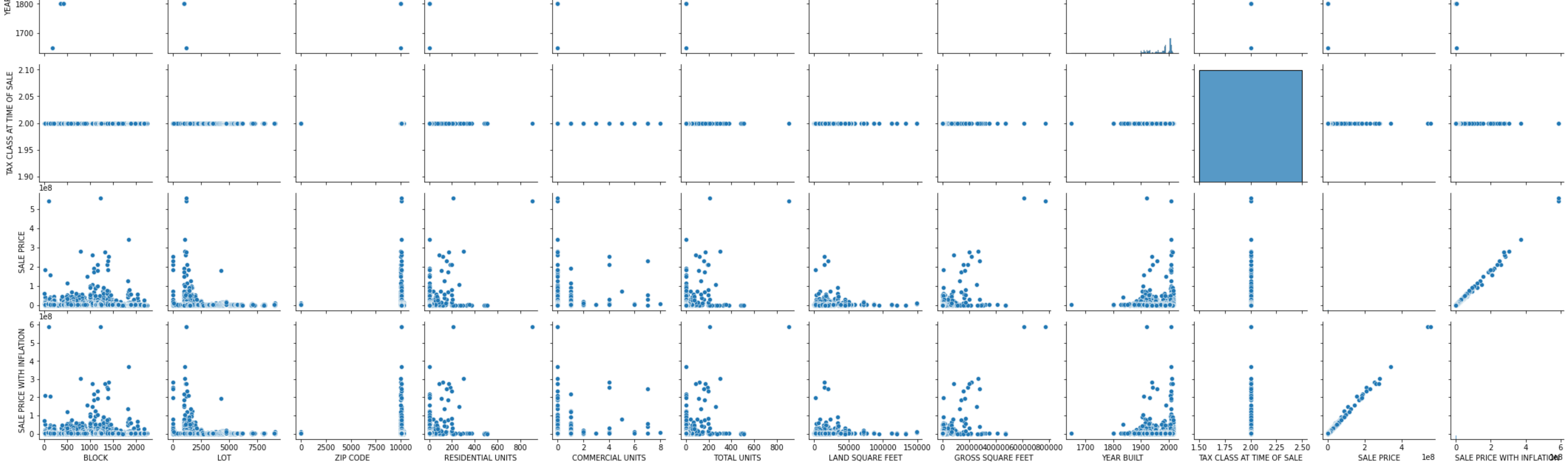
Out[87]:
```
<AxesSubplot:xlabel='date', ylabel='SALE PRICE'>
```



In [88]:
```
g = sns.PairGrid(most_frequent_app)
g.map_diag(sns.histplot)
g.map_offdiag(sns.scatterplot)
```

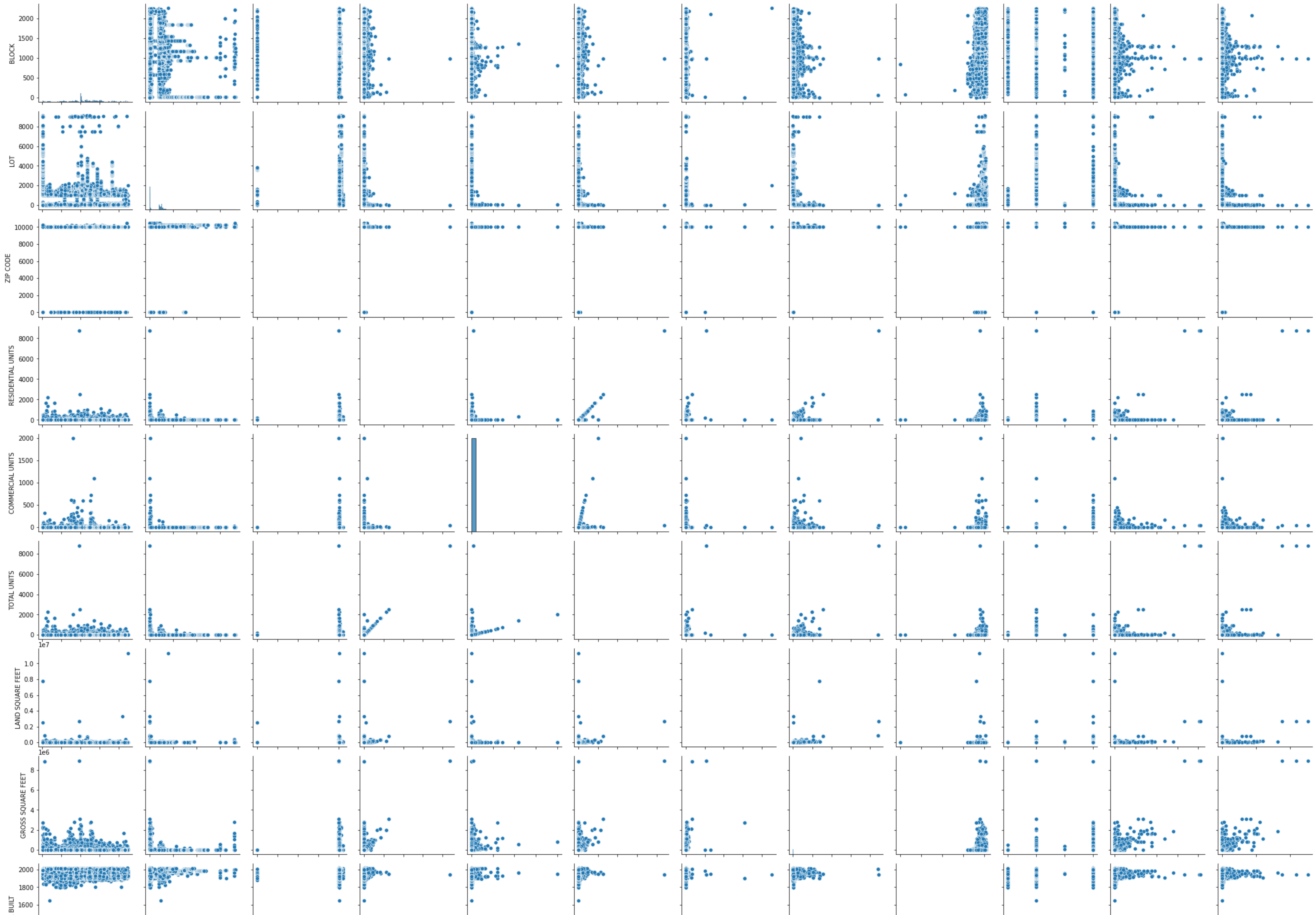
Out[88]: <seaborn.axisgrid.PairGrid at 0x179b7da98e0>

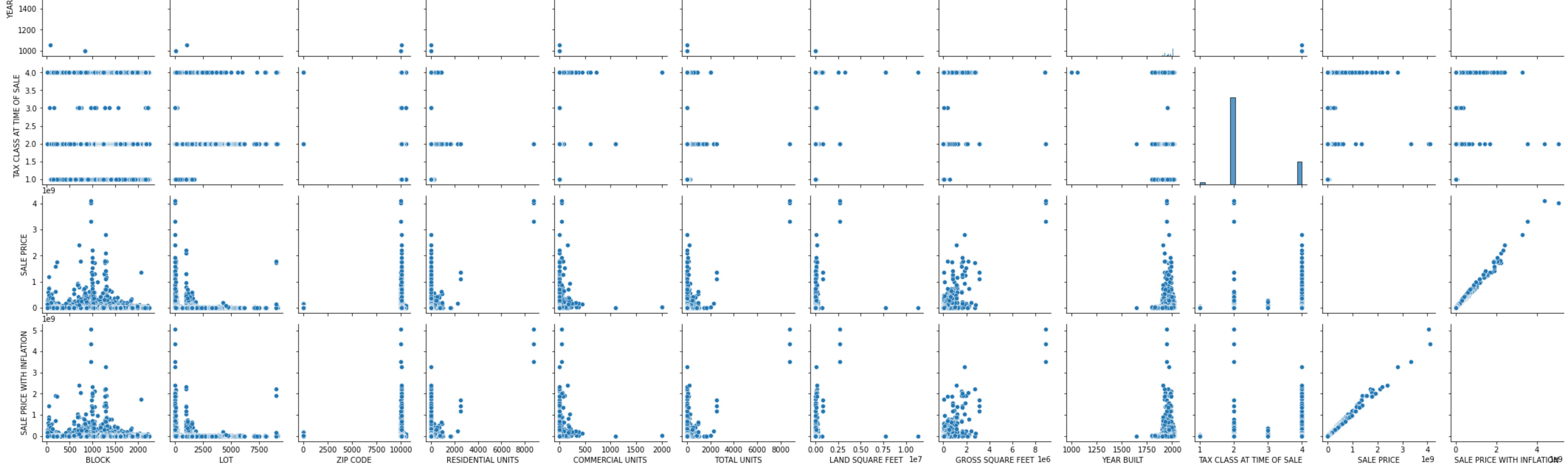



YEAR
TAX CLASS AT TIME OF SALE
SALE PRICE
SALE PRICE WITH INFLATION
1e8
BLOCK
LOT
ZIP CODE
RESIDENTIAL UNITS
COMMERCIAL UNITS
TOTAL UNITS
LAND SQUARE FEET
GROSS SQUARE FEET
YEAR BUILT
TAX CLASS AT TIME OF SALE
SALE PRICE
SALE PRICE WITH INFLATION

In [89]:
```
g2 = sns.PairGrid(df_processed)
g2.map_diag(sns.histplot)
g2.map_offdiag(sns.scatterplot)
```

Out[89]: <seaborn.axisgrid.PairGrid at 0x179bdc2f8e0>





PRICE PREDICTION - TAX CLASS 2

Initialize dataframe

In [91]:
```
df_price_prediction = df_processed.copy(deep=True)
df_price_prediction = df_price_prediction[df_price_prediction['SALE PRICE'].notna()]
df_price_prediction = df_price_prediction[df_price_prediction['GEOJSON NEIGHBORHOOD'].notna()]
df_price_prediction = df_price_prediction[df_price_prediction['SALE PRICE'] > 10000]
df_price_prediction = df_price_prediction[df_price_prediction['TAX CLASS AT TIME OF SALE'] == 2]
df_price_prediction['SALE DATE (DT)'] = pd.to_datetime(df_price_prediction['SALE DATE'])
df_price_prediction['SALE YEAR'] = df_price_prediction['SALE DATE (DT)'].dt.year
df_price_prediction.tail()
```

Out[91]:

	NEIGHBORHOOD	BUILDING CLASS CATEGORY	TAX CLASS AT PRESENT	BLOCK	LOT	EASE-MENT	BUILDING CLASS AT PRESENT	ADDRESS	APARTMENT NUMBER	ZIP CODE	...	GROSS SQUARE FEET	YEAR BUILT	TAX CLASS AT TIME OF SALE	BUILDING CLASS AT TIME OF SALE	SALE PRICE	SALE DATE	SALE PRICE WITH INFLATION	GEOJSON NEIGHBORHOOD	SALE DATE (DT)	SALE YEAR
16736	WASHINGTON HEIGHTS UPPER	13 CONDOS - ELEVATOR APARTMENTS	2	2180	1402		R4	69 BENNETT AVENUE	102	10033.0	...	805.0	1954.0	2	R4	505000.0	2018-05-10	505000.00	Washington Heights North	2018-05-10	2018
16737	WASHINGTON HEIGHTS UPPER	13 CONDOS - ELEVATOR APARTMENTS	2	2180	1427		R4	69 BENNETT AVENUE	307	10033.0	...	741.0	1954.0	2	R4	510000.0	2017-12-18	520200.00	Washington Heights North	2017-12-18	2017
16738	WASHINGTON HEIGHTS UPPER	13 CONDOS - ELEVATOR APARTMENTS	2	2180	1504		R4	105 BENNETT AVENUE	12B	10033.0	...	810.0	1939.0	2	R4	492804.0	2017-12-27	502660.08	Washington Heights North	2017-12-27	2017
16740	WASHINGTON HEIGHTS UPPER	13 CONDOS - ELEVATOR APARTMENTS	2	2180	1515		R4	105 BENNETT AVENUE	24A	10033.0	...	747.0	1939.0	2	R4	510000.0	2018-05-23	510000.00	Washington Heights North	2018-05-23	2018
16741	WASHINGTON HEIGHTS UPPER	13 CONDOS - ELEVATOR APARTMENTS	2	2180	1522		R4	105 BENNETT AVENUE	32A	10033.0	...	806.0	1939.0	2	R4	549450.0	2018-07-23	549450.00	Washington Heights North	2018-07-23	2018

5 rows × 24 columns

Analyse output dataframe

In [92]:
```
df_price_prediction['SALE PRICE'].describe()
```

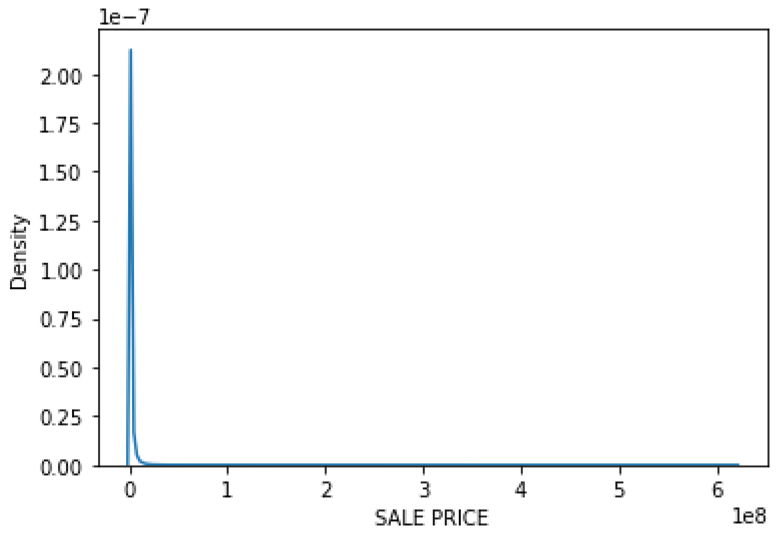
Out[92]:
```
count    2.176780e+05
mean     1.677677e+06
std      5.811269e+06
min      1.004000e+04
25%      4.700000e+05
50%      7.850000e+05
75%      1.560000e+06
max      6.200000e+08
Name: SALE PRICE, dtype: float64
```

In [93]:
```
df_price_prediction['SALE PRICE'].value_counts()
```

Out[93]:
```
650000.0     1193
550000.0     1131
1100000.0    1097
1200000.0    1066
600000.0     1061
             ... 
1356300.0       1
339076.0        1
678154.0        1
470150.0        1
3145730.0       1
Name: SALE PRICE, Length: 30592, dtype: int64
```

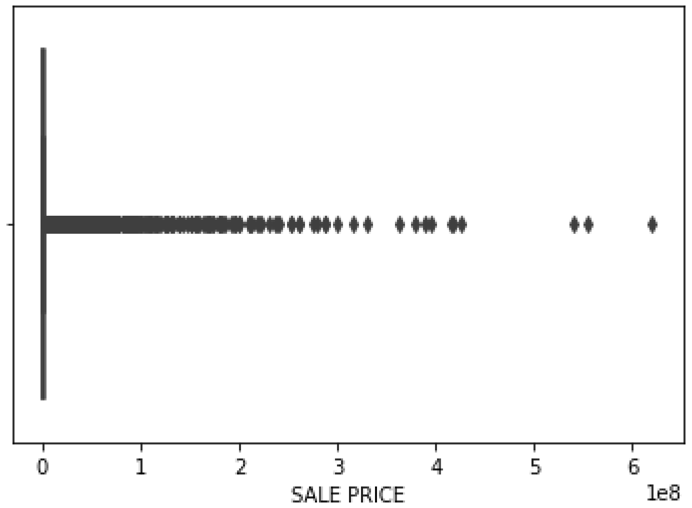
In [94]:
```
sns.kdeplot(df_price_prediction['SALE PRICE'])
# non-normal distribution - right skewed
```

Out[94]: <AxesSubplot:xlabel='SALE PRICE', ylabel='Density'>

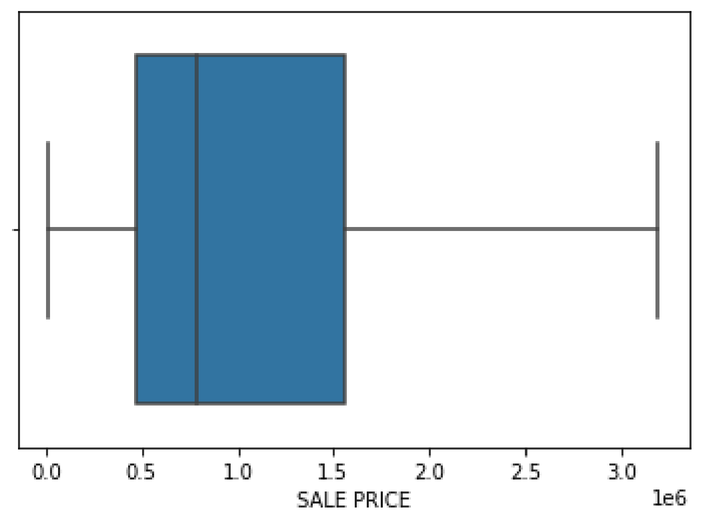


In [95]:
```
sns.boxplot(x=df_price_prediction['SALE PRICE'])
```

Out[95]: <AxesSubplot:xlabel='SALE PRICE'>



In [96]:
```
# boxplot bez outlierov
sns.boxplot(x=df_price_prediction['SALE PRICE'], showfliers=False)
```
Out[96]: <AxesSubplot:xlabel='SALE PRICE'>

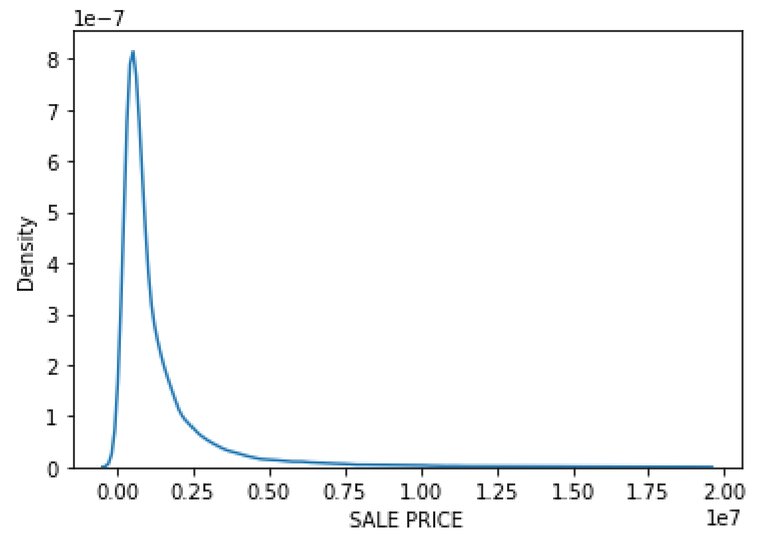


Remove outliers - by standard deviation (wrong)

In [97]:
```
df_price_no_fliers_std = df_price_prediction[np.abs(df_price_prediction['SALE PRICE']-df_price_prediction['SALE PRICE'].mean()) <= (3*df_price_prediction['SALE PRICE'].std())]
```

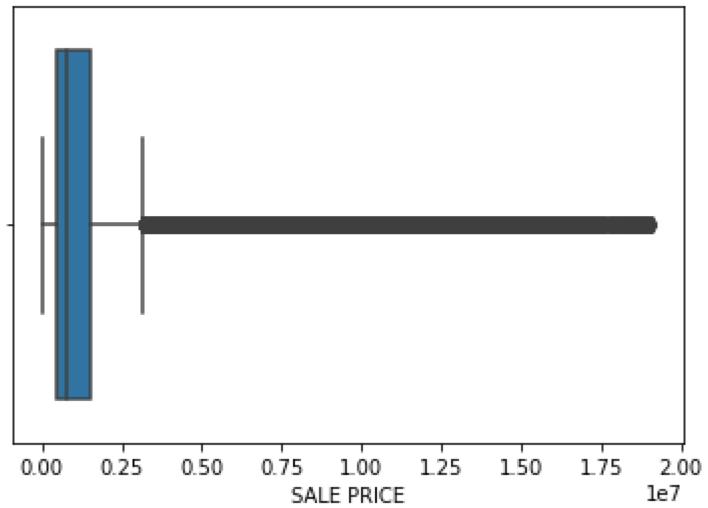
In [98]:
```
df_price_no_fliers_std['SALE PRICE'].value_counts()
```
Out[98]:
```
650000.0      1193
550000.0      1131
1100000.0     1097
1200000.0     1066
600000.0      1061
              ...
15044643.0       1
1356300.0        1
339076.0         1
678154.0         1
3145730.0        1
Name: SALE PRICE, Length: 29852, dtype: int64
```

In [99]:
```
sns.kdeplot(df_price_no_fliers_std['SALE PRICE'])
```
Out[99]: <AxesSubplot:xlabel='SALE PRICE', ylabel='Density'>



In [101]:
```
sns.boxplot(x=df_price_no_fliers_std['SALE PRICE'])
```

Out[101]: <AxesSubplot:xlabel='SALE PRICE'>



Remove outliers - by qunatiles

In [102]:
```
Q1 = df_price_prediction['SALE PRICE'].quantile(.25)
Q3 = df_price_prediction['SALE PRICE'].quantile(.75)
IQR = Q3 - Q1

# (Q1 -/+ 1.5 * IQR) - Cutting at ±1.5IQR is therefore somewhat comparable to cutting slightly below ±3σ
df_price_no_fliers_quntile = df_price_prediction[df_price_prediction['SALE PRICE'] >= (Q1 - 1.5 * IQR)]
df_price_no_fliers_quntile = df_price_no_fliers_quntile[df_price_no_fliers_quntile['SALE PRICE'] <= (Q3 + 1.5 * IQR)]
```

In [103]:
```
df_price_no_fliers_quntile['SALE PRICE'].value_counts()
```

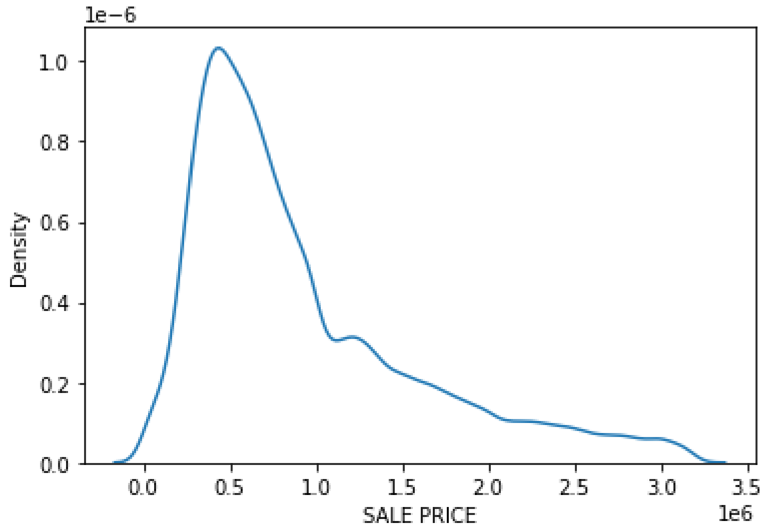
Out[103]:
```
650000.0     1193
550000.0     1131
1100000.0    1097
1200000.0    1066
600000.0     1061
             ...
470901.0        1
2740837.0       1
1359351.0       1
2718717.0       1
3145730.0       1
Name: SALE PRICE, Length: 24353, dtype: int64
```

In [104]:
```
len(df_price_no_fliers_quntile)
```

Out[104]: 196360

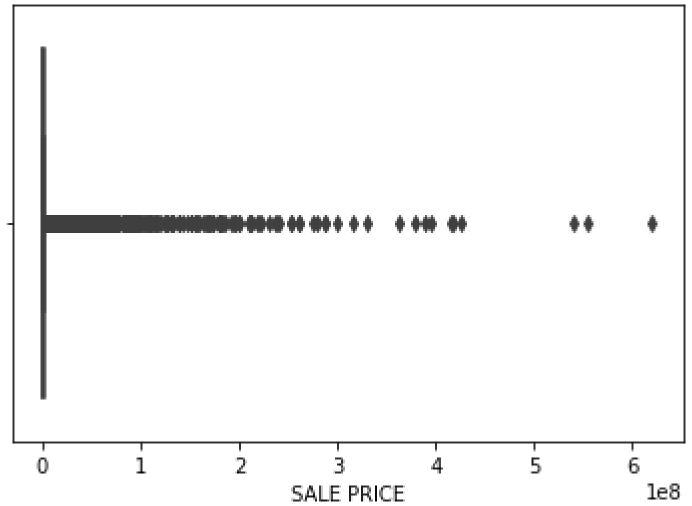
```
In [105]: sns.kdeplot(df_price_no_fliers_quntile['SALE PRICE'])
```

```
Out[105]: <AxesSubplot:xlabel='SALE PRICE', ylabel='Density'>
```



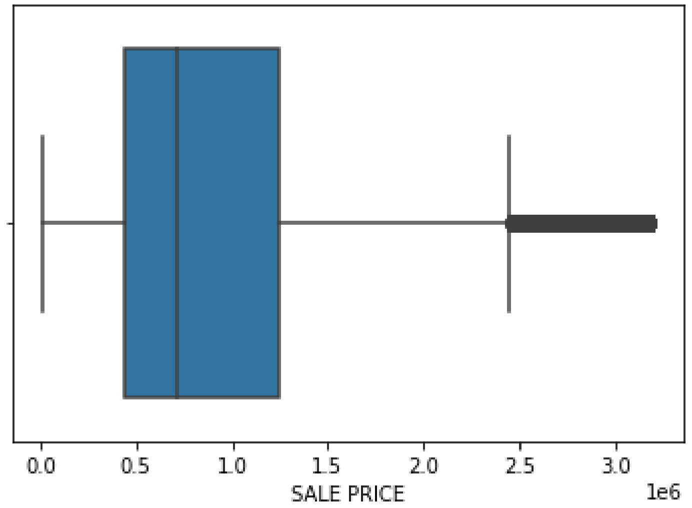

```
In [106]: sns.boxplot(x=df_price_prediction['SALE PRICE'])
```

```
Out[106]: <AxesSubplot:xlabel='SALE PRICE'>
```

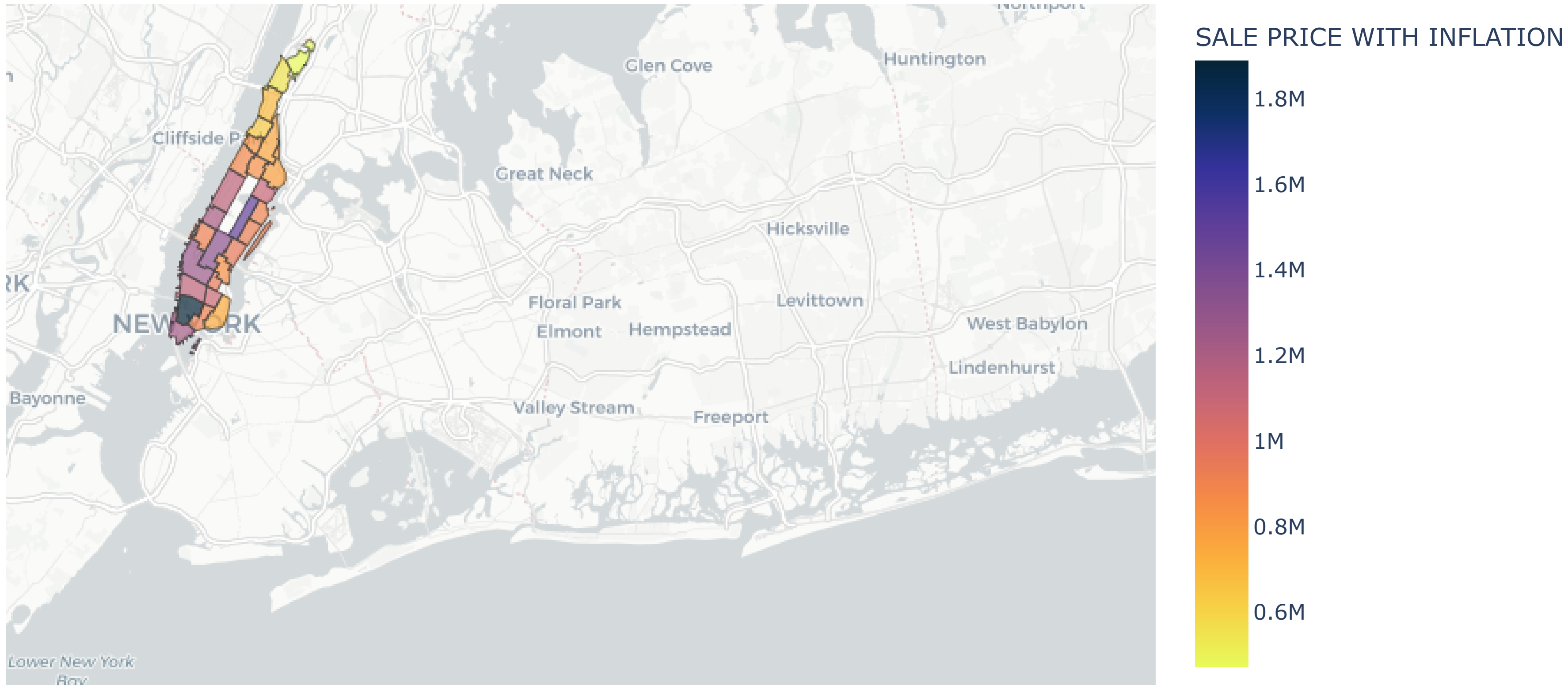


In [107]:
```
sns.boxplot(x=df_price_no_fliers_quntile['SALE PRICE'])
```

Out[107]: <AxesSubplot:xlabel='SALE PRICE'>



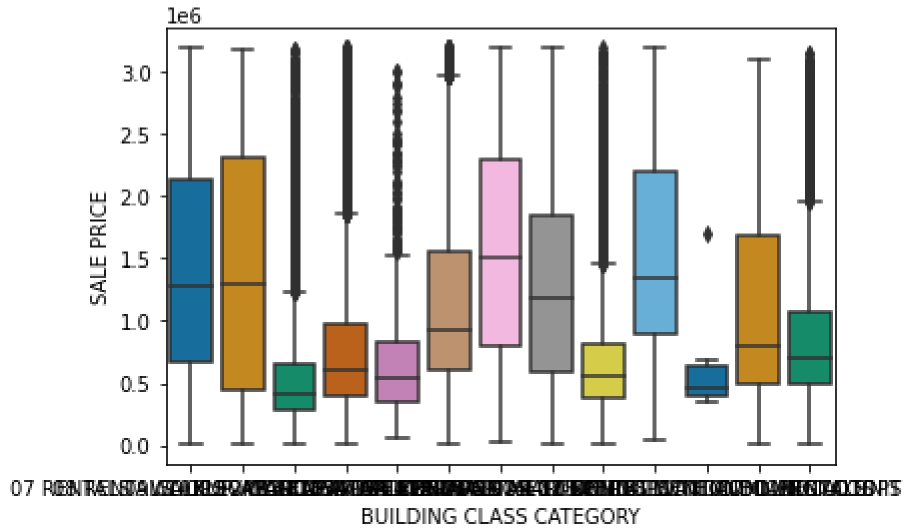
In [108]:
```python
grouped = df_price_no_fliers_quntile.groupby(['GEOJSON NEIGHBORHOOD'])['SALE PRICE WITH INFLATION'].mean().reset_index()

fig = px.choropleth_mapbox(grouped,
                           geojson=nyc_neighbourhoods_map,
                           locations="GEOJSON NEIGHBORHOOD",
                           featureidkey="properties.ntaname",
                            color="SALE PRICE WITH INFLATION",
                           color_continuous_scale=px.colors.sequential.thermal[::-1],
                         #    range_color=(0, 0.5),
                         #    animation_frame="SALE DATE",
                           mapbox_style="carto-positron",
                           zoom=9, center={"lat": 40.7, "lon": -73.7},
                           opacity=0.7,
                           hover_name="GEOJSON NEIGHBORHOOD"
                           )

fig.show()
```



In [109]:
```python
df_price_no_fliers_quntile['BUILDING CLASS CATEGORY'].unique()
```

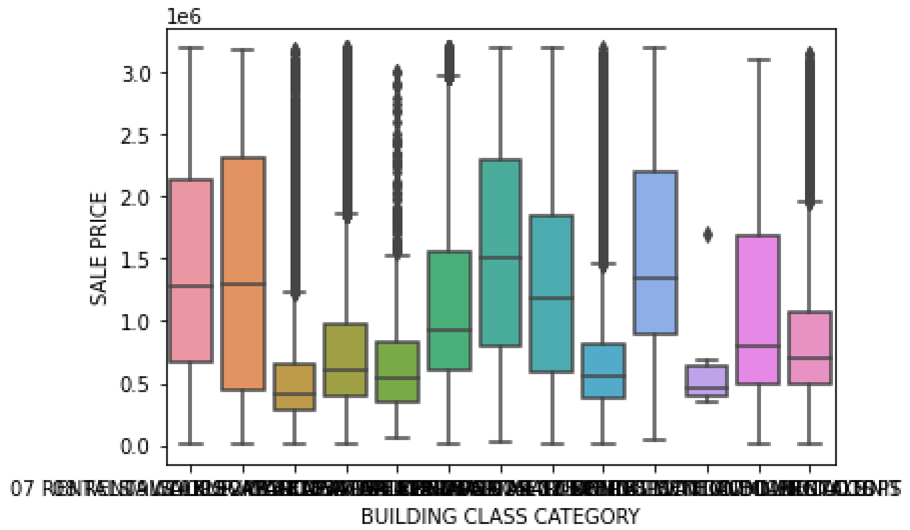
Out[109]:
```
array(['07 RENTALS - WALKUP APARTMENTS',
       '08 RENTALS - ELEVATOR APARTMENTS', '09 COOPS - WALKUP APARTMENTS',
       '10 COOPS - ELEVATOR APARTMENTS', '12 CONDOS - WALKUP APARTMENTS',
       '13 CONDOS - ELEVATOR APARTMENTS', '14 RENTALS - 4-10 UNIT',
       '15 CONDOS - 2-10 UNIT RESIDENTIAL', '17 CONDOPS',
       '16 CONDOS - 2-10 UNIT WITH COMMERCIAL UNIT',
       '11 SPECIAL CONDO BILLING LOTS', '11A CONDO-RENTALS',
       '17 CONDO COOPS'], dtype=object)
```

In [110]:
```
bplot = sns.boxplot(y='SALE PRICE', x='BUILDING CLASS CATEGORY',
                 data=df_price_no_fliers_quntile,
                 width=0.8,
                 palette="colorblind")
```



In [111]:
```
sns.boxplot(x=df_price_no_fliers_quntile['BUILDING CLASS CATEGORY'], y=df_price_no_fliers_quntile['SALE PRICE'])
```

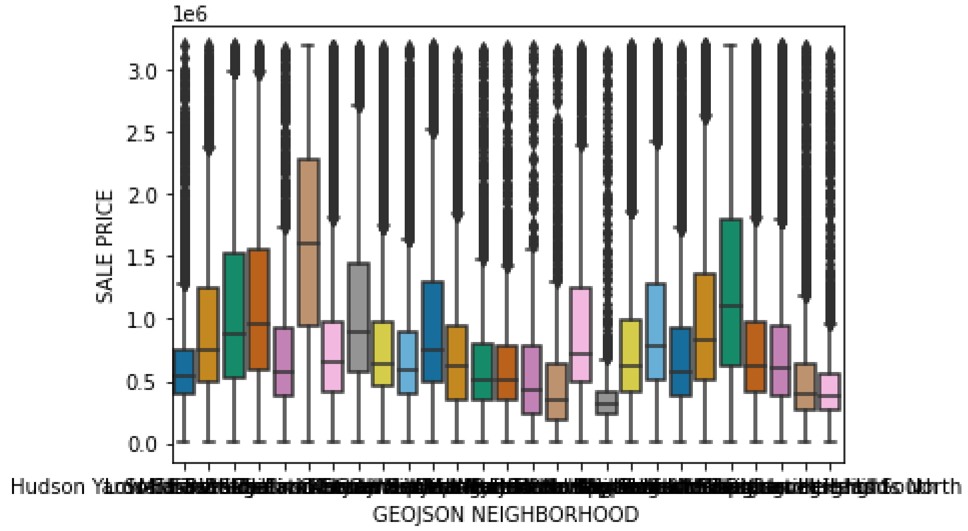
Out[111]: <AxesSubplot:xlabel='BUILDING CLASS CATEGORY', ylabel='SALE PRICE'>



In [112]:
```
df_price_no_fliers_quntile['GEOJSON NEIGHBORHOOD'].unique()
```

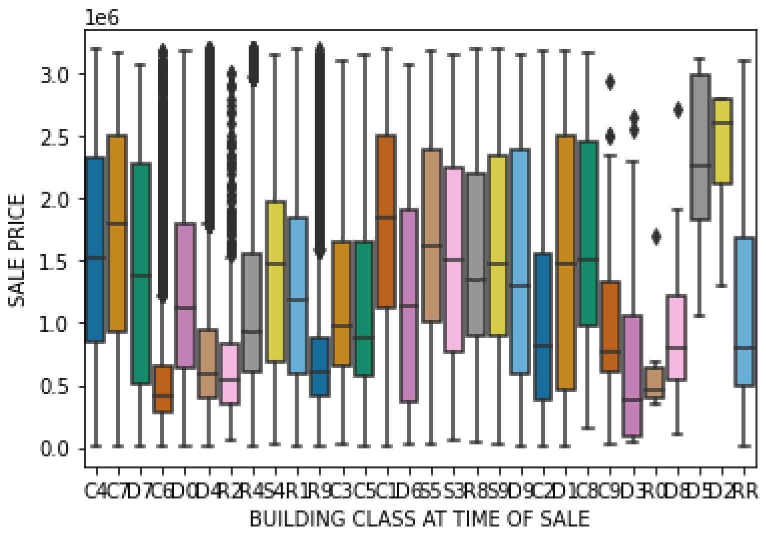
Out[112]:
```
array(['Lower East Side', 'East Village',
       'Hudson Yards-Chelsea-Flatiron-Union Square',
       'Midtown-Midtown South', 'Chinatown',
       'SoHo-TriBeCa-Civic Center-Little Italy', 'Clinton',
       'Battery Park City-Lower Manhattan', 'Gramercy',
       'Murray Hill-Kips Bay', 'West Village', 'Central Harlem South',
       'Central Harlem North-Polo Grounds', 'East Harlem North',
       'Manhattanville', 'Hamilton Heights', 'East Harlem South',
       'Marble Hill-Inwood', 'Turtle Bay-East Midtown', 'Upper West Side',
       'Morningside Heights', 'Lincoln Square',
       'Upper East Side-Carnegie Hill', 'Lenox Hill-Roosevelt Island',
       'Yorkville', 'Washington Heights South',
       'Washington Heights North'], dtype=object)
```

```
In [113]: bplot = sns.boxplot(y='SALE PRICE', x='GEOJSON NEIGHBORHOOD',
                              data=df_price_no_fliers_quntile,
                              width=0.8,
                              palette="colorblind")
```



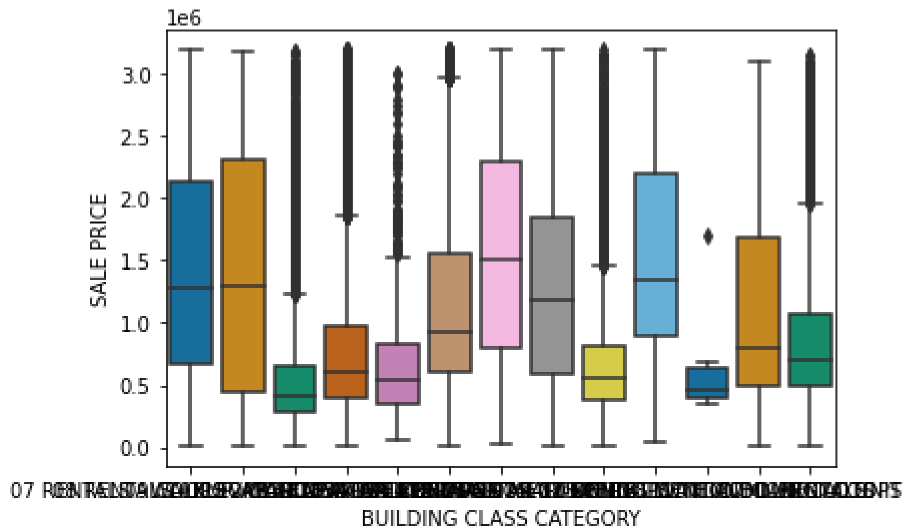

```
In [114]: df_price_no_fliers_quntile['BUILDING CLASS AT TIME OF SALE'].unique()

Out[114]: array(['C4', 'C7', 'D7', 'C6', 'D0', 'D4', 'R2', 'R4', 'S4', 'R1', 'R9',
                 'C3', 'C5', 'C1', 'D6', 'S5', 'S3', 'R8', 'S9', 'D9', 'C2', 'D1',
                 'C8', 'C9', 'D3', 'R0', 'D8', 'D5', 'D2', 'RR'], dtype=object)
```

```
In [115]: bplot = sns.boxplot(y='SALE PRICE', x='BUILDING CLASS AT TIME OF SALE',
                              data=df_price_no_fliers_quntile,
                              width=0.8,
                              palette="colorblind")
```




```
In [116]: df_price_no_fliers_quntile['BUILDING CLASS CATEGORY'].unique()

Out[116]: array(['07 RENTALS - WALKUP APARTMENTS',
                 '08 RENTALS - ELEVATOR APARTMENTS', '09 COOPS - WALKUP APARTMENTS',
                 '10 COOPS - ELEVATOR APARTMENTS', '12 CONDOS - WALKUP APARTMENTS',
                 '13 CONDOS - ELEVATOR APARTMENTS', '14 RENTALS - 4-10 UNIT',
                 '15 CONDOS - 2-10 UNIT RESIDENTIAL', '17 CONDOPS',
                 '16 CONDOS - 2-10 UNIT WITH COMMERCIAL UNIT',
                 '11 SPECIAL CONDO BILLING LOTS', '11A CONDO-RENTALS',
                 '17 CONDO COOPS'], dtype=object)
```

In [117]:
```python
bplot = sns.boxplot(y='SALE PRICE', x='BUILDING CLASS CATEGORY',
                    data=df_price_no_fliers_quntile,
                    width=0.8,
                    palette="colorblind")
```



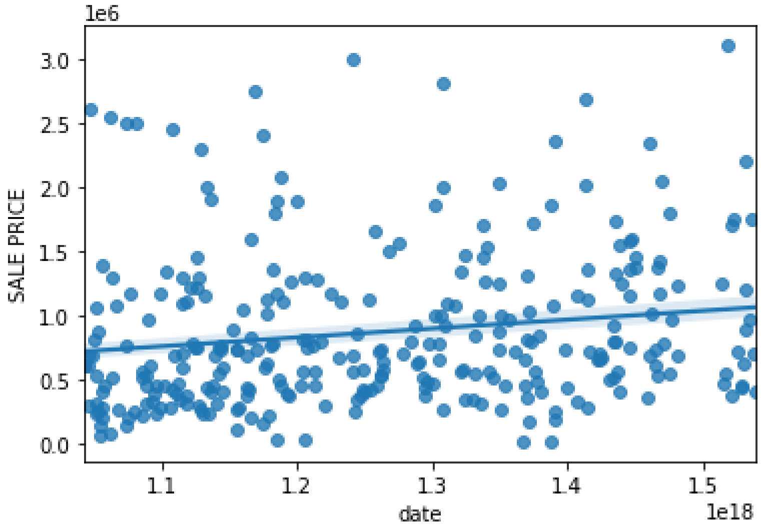
In [118]:
```python
df_price_no_fliers_quntile.corr()
```

Out[118]:

	BLOCK	LOT	ZIP CODE	RESIDENTIAL UNITS	COMMERCIAL UNITS	TOTAL UNITS	LAND SQUARE FEET	GROSS SQUARE FEET	YEAR BUILT	TAX CLASS AT TIME OF SALE	SALE PRICE	SALE PRICE WITH INFLATION	SALE YEAR
BLOCK	1.000000	-0.188573	0.011372	0.001238	0.001114	0.001015	-0.164706	-0.055445	-0.038986	NaN	-0.158805	-0.157776	-0.000634
LOT	-0.188573	1.000000	0.018695	-0.001880	-0.005356	-0.002515	0.388089	-0.098975	0.417255	NaN	0.165862	0.162655	0.012962
ZIP CODE	0.011372	0.018695	1.000000	0.001097	0.000065	0.001030	-0.044132	0.002376	0.019640	NaN	-0.017067	-0.013195	-0.026802
RESIDENTIAL UNITS	0.001238	-0.001880	0.001097	1.000000	0.095300	0.982173	0.656511	0.931257	-0.005294	NaN	0.012914	0.016358	-0.023135
COMMERCIAL UNITS	0.001114	-0.005356	0.000065	0.095300	1.000000	0.274726	0.034583	0.133518	-0.004131	NaN	0.002803	0.003638	-0.004840
TOTAL UNITS	0.001015	-0.002515	0.001030	0.982173	0.274726	1.000000	0.639792	0.922505	-0.004603	NaN	0.013634	0.016912	-0.022114
LAND SQUARE FEET	-0.164706	0.388089	-0.044132	0.656511	0.034583	0.639792	1.000000	0.705174	0.350785	NaN	-0.057622	-0.095060	0.206851
GROSS SQUARE FEET	-0.055445	-0.098975	0.002376	0.931257	0.133518	0.922505	0.705174	1.000000	0.036874	NaN	-0.111731	-0.086816	-0.180571
YEAR BUILT	-0.038986	0.417255	0.019640	-0.005294	-0.004131	-0.004603	0.350785	0.036874	1.000000	NaN	0.087025	0.075276	0.056356
TAX CLASS AT TIME OF SALE	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
SALE PRICE	-0.158805	0.165862	-0.017067	0.012914	0.002803	0.013634	-0.057622	-0.111731	0.087025	NaN	1.000000	0.988462	0.187872
SALE PRICE WITH INFLATION	-0.157776	0.162655	-0.013195	0.016358	0.003638	0.016912	-0.095060	-0.086816	0.075276	NaN	0.988462	1.000000	0.067038
SALE YEAR	-0.000634	0.012962	-0.026802	-0.023135	-0.004840	-0.022114	0.206851	-0.180571	0.056356	NaN	0.187872	0.067038	1.000000

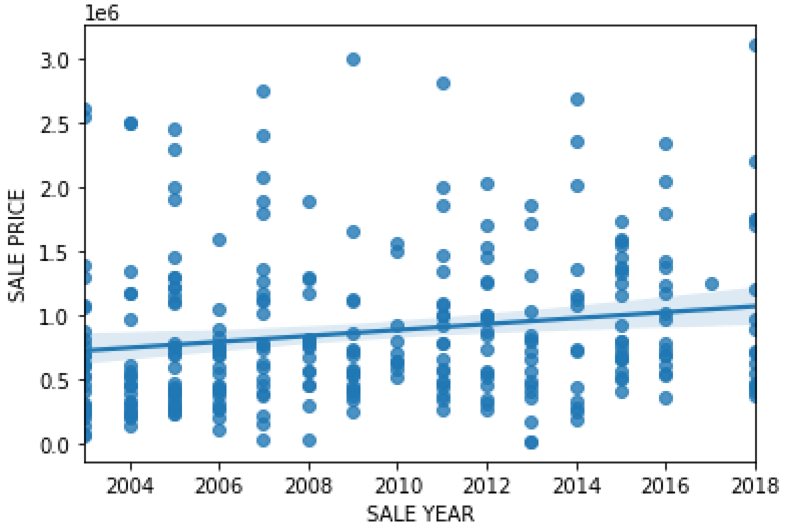
In [119]:
```python
df_price_over_time = df_price_no_fliers_quntile.sample(300)
df_price_over_time = df_price_over_time.sort_values('SALE DATE')
df_price_over_time['date'] = pd.to_numeric(df_price_over_time['SALE DATE'])
sns.regplot(x=df_price_over_time['date'], y=df_price_over_time['SALE PRICE'])
# Reg plot pre SAMPLE z vycisteneho dataframe
```

Out[119]: <AxesSubplot:xlabel='date', ylabel='SALE PRICE'>



In [120]:
```python
sns.regplot(data=df_price_over_time, x='SALE YEAR', y='SALE PRICE')
# Reg plot pre SAMPLE z vycisteneho dataframe
```

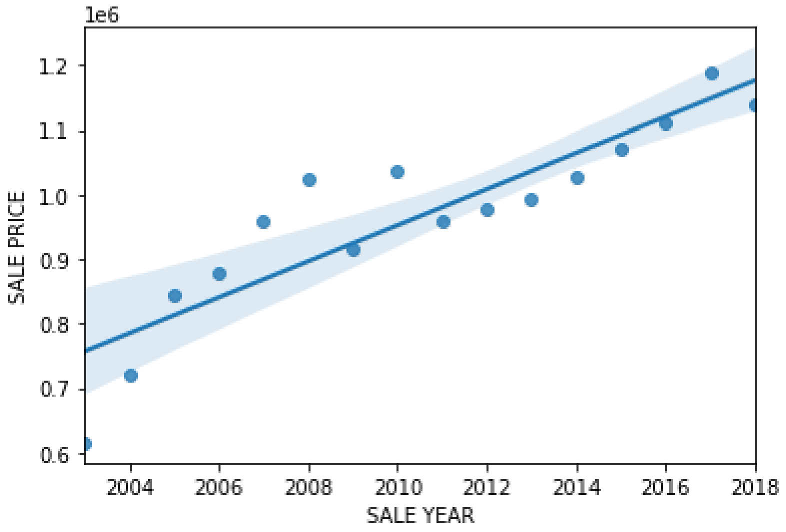
Out[120]: <AxesSubplot:xlabel='SALE YEAR', ylabel='SALE PRICE'>



Grouped on year

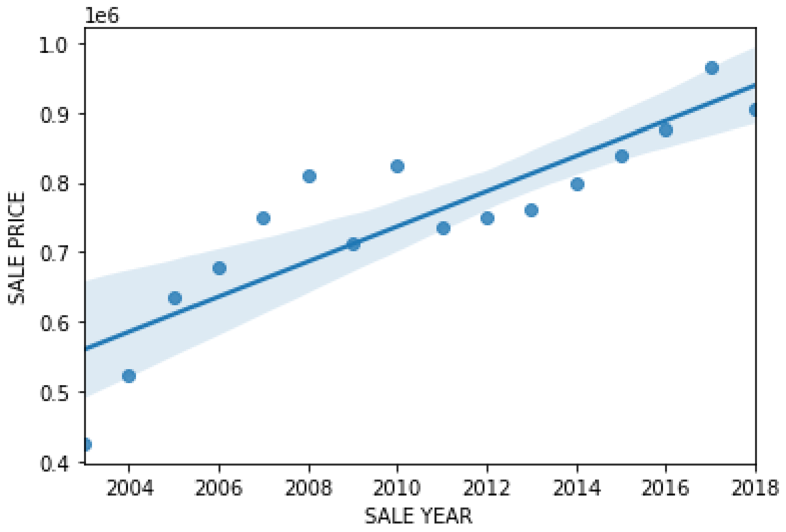
In [121]:
```python
grouped_year_mean = df_price_no_fliers_quntile.groupby(['SALE YEAR'])['SALE PRICE'].mean().reset_index()
sns.regplot(data=grouped_year_mean, x='SALE YEAR', y='SALE PRICE')

# Reg plot pre cely vycisteny dataframe (mean)
```

Out[121]: <AxesSubplot:xlabel='SALE YEAR', ylabel='SALE PRICE'>



In [122]:
```python
grouped_year_median = df_price_no_fliers_quntile.groupby(['SALE YEAR'])['SALE PRICE'].median().reset_index()
sns.regplot(data=grouped_year_median, x='SALE YEAR', y='SALE PRICE')

# Reg plot pre cely vycisteny dataframe (median)
```

Out[122]: <AxesSubplot:xlabel='SALE YEAR', ylabel='SALE PRICE'>



In [123]:
```
train_data = df_price_no_fliers_quntile.sample(5000)
train_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 5000 entries, 6226 to 3026
Data columns (total 24 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   NEIGHBORHOOD                    5000 non-null   object
 1   BUILDING CLASS CATEGORY         5000 non-null   object
 2   TAX CLASS AT PRESENT            4985 non-null   object
 3   BLOCK                           5000 non-null   int64
 4   LOT                             5000 non-null   int64
 5   EASE-MENT                       5000 non-null   object
 6   BUILDING CLASS AT PRESENT       4985 non-null   object
 7   ADDRESS                         4988 non-null   object
 8   APARTMENT NUMBER                5000 non-null   object
 9   ZIP CODE                        5000 non-null   float64
 10  RESIDENTIAL UNITS               5000 non-null   int64
 11  COMMERCIAL UNITS                5000 non-null   int64
 12  TOTAL UNITS                     5000 non-null   int64
 13  LAND SQUARE FEET                134 non-null    float64
 14  GROSS SQUARE FEET               178 non-null    float64
 15  YEAR BUILT                      4240 non-null   float64
 16  TAX CLASS AT TIME OF SALE       5000 non-null   int64
 17  BUILDING CLASS AT TIME OF SALE  5000 non-null   object
 18  SALE PRICE                      5000 non-null   float64
 19  SALE DATE                       5000 non-null   datetime64[ns]
 20  SALE PRICE WITH INFLATION       5000 non-null   float64
 21  GEOJSON NEIGHBORHOOD            5000 non-null   object
 22  SALE DATE (DT)                  5000 non-null   datetime64[ns]
 23  SALE YEAR                       5000 non-null   int64
dtypes: datetime64[ns](2), float64(6), int64(7), object(9)
memory usage: 976.6+ KB
```

In [124]:
```
train_data.describe()
```

Out[124]:

	BLOCK	LOT	ZIP CODE	RESIDENTIAL UNITS	COMMERCIAL UNITS	TOTAL UNITS	LAND SQUARE FEET	GROSS SQUARE FEET	YEAR BUILT	TAX CLASS AT TIME OF SALE	SALE PRICE	SALE PRICE WITH INFLATION	SALE YEAR
count	5000.000000	5000.000000	5000.000000	5000.000000	5000.000000	5000.000000	134.000000	178.000000	4240.000000	5000.0	5.000000e+03	5.000000e+03	5000.000000
mean	1155.551200	689.224200	10024.175200	0.747000	0.017200	0.764200	7128.552239	7733.848315	1950.298113	2.0	9.491407e+05	1.104743e+06	2009.320400
std	503.138044	866.810606	248.642903	5.083991	0.221165	5.162551	12014.708604	21808.920873	32.535734	0.0	6.851713e+05	7.903764e+05	4.456245
min	16.000000	1.000000	0.000000	0.000000	0.000000	0.000000	901.000000	351.000000	1846.000000	2.0	1.060000e+04	1.260000e+04	2003.000000
25%	831.750000	25.000000	10014.000000	0.000000	0.000000	0.000000	1868.500000	1006.500000	1924.000000	2.0	4.500000e+05	5.320000e+05	2005.000000
50%	1207.000000	311.000000	10022.000000	0.000000	0.000000	0.000000	2500.000000	3001.500000	1955.000000	2.0	7.250000e+05	8.440689e+05	2009.000000
75%	1473.000000	1140.250000	10027.000000	1.000000	0.000000	1.000000	5494.250000	7390.250000	1974.000000	2.0	1.295000e+06	1.493539e+06	2013.000000
max	2250.000000	9059.000000	10463.000000	275.000000	9.000000	275.000000	87120.000000	229973.000000	2016.000000	2.0	3.170000e+06	4.324508e+06	2018.000000

In [125]:

```
train_data['SALE DATE (DT)'] = pd.to_datetime(train_data['SALE DATE'])
train_data.set_index('SALE DATE (DT)', inplace=True)
train_data.head()
```

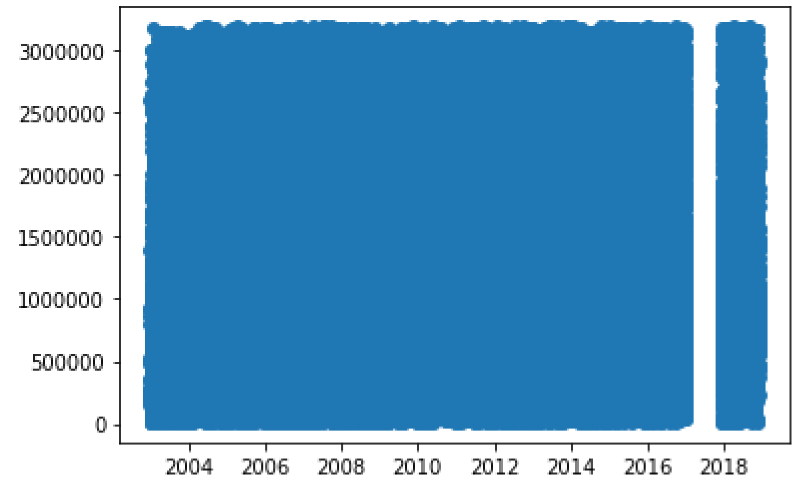
Out[125]:

SALE DATE (DT)	NEIGHBORHOOD	BUILDING CLASS CATEGORY	TAX CLASS AT PRESENT	BLOCK	LOT	EASE-MENT	BUILDING CLASS AT PRESENT	ADDRESS	APARTMENT NUMBER	ZIP CODE	...	LAND SQUARE FEET	GROSS SQUARE FEET	YEAR BUILT	TAX CLASS AT TIME OF SALE	BUILDING CLASS AT TIME OF SALE	SALE PRICE	SALE DATE	SALE PRICE WITH INFLATION	GEOJSON NEIGHBORHOOD	SALE YEAR
2008-02-14	HARLEM-CENTRAL	07 RENTALS - WALKUP APARTMENTS	1	1907	10		A5	151 WEST 122 STREET		10027.0	...	2018.0	3216.0	1910.0	2	C5	1450000.0	2008-02-14	1696500.0	Central Harlem South	2008
2007-04-20	UPPER EAST SIDE	13 CONDOS - ELEVATOR APARTMENTS	2	1539	1167		R4	245 EAST 93 STREET	12F	10128.0	...	NaN	NaN	1985.0	2	R4	650000.0	2007-04-20	786500.0	Yorkville	2007
2003-09-15	GREENWICH VILLAGE-CENTRAL	17 CONDOPS	2	535	1002		R9	250 MERCER ST. APT. B 1102		10012.0	...	NaN	NaN	1900.0	2	R9	570000.0	2003-09-15	780900.0	West Village	2003
2016-06-22	UPPER WEST SIDE	10 COOPS - ELEVATOR APARTMENTS	2	1210	9		D4	157 WEST 79TH STREET, 5D		10024.0	...	NaN	NaN	1911.0	2	D4	789000.0	2016-06-22	828450.0	Upper West Side	2016
2003-07-02	UPPER WEST SIDE	10 COOPS - ELEVATOR APARTMENTS	2	1211	29		D4	101 WEST 80TH ST. APT. 8A		10024.0	...	NaN	NaN	1900.0	2	D4	470000.0	2003-07-02	643900.0	Upper West Side	2003

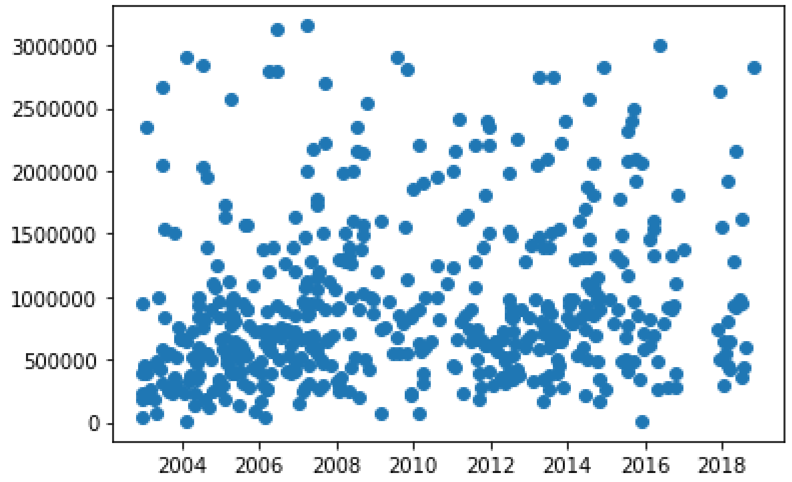
5 rows × 23 columns

In [126]:

```
plt.scatter(x=df_price_no_fliers_quntile['SALE DATE'],y=df_price_no_fliers_quntile['SALE PRICE'])
ax =plt.gca()
ax.get_yaxis().get_major_formatter().set_scientific(False)
plt.draw()
```



In [127]:
```
sample = df_price_no_fliers_quntile.sample(500)
plt.scatter(x=sample['SALE DATE'],y=sample['SALE PRICE'])
ax =plt.gca()
ax.get_yaxis().get_major_formatter().set_scientific(False)
plt.draw()
```



In [128]:
```
df_price_no_fliers_quntile['SALE PRICE'].describe().apply(lambda x: format(x, 'f'))
```

Out[128]:
```
count     196360.000000
mean      930330.950438
std       673734.847890
min        10040.000000
25%       445000.000000
50%       710000.000000
75%      1247356.000000
max      3195000.000000
Name: SALE PRICE, dtype: object
```

In [129]:
```
bins=[-100000000,20000,40000,60000,80000,100000,1000000,10000000,500000000]
choices =['$0-$200k','$200k-$400k','$400k-$600k','$600k-$800k','$800k-$1mlln','$1mlln-$10mlln','$10mlln-$100mlln','$100mlln-$500mlln']
sample['PRICE RANGE']=pd.cut(sample['SALE PRICE'],bins=bins,labels=choices)
sample.head()
```

Out[129]:

	NEIGHBORHOOD	BUILDING CLASS CATEGORY	TAX CLASS AT PRESENT	BLOCK	LOT	EASE-MENT	BUILDING CLASS AT PRESENT	ADDRESS	APARTMENT NUMBER	ZIP CODE	...	YEAR BUILT	TAX CLASS AT TIME OF SALE	BUILDING CLASS AT TIME OF SALE	SALE PRICE	SALE DATE	SALE PRICE WITH INFLATION	GEOJSON NEIGHBORHOOD	SALE DATE (DT)	SALE YEAR	PRICE RANGE
3313	GRAMERCY	10 COOPS - ELEVATOR APARTMENTS	2	877	42		D4	39 GRAMERCY PARK NORTH, 3B/C		10010.0	...	1956.0	2	D4	1800000.0	2016-11-14	1890000.0	Gramercy	2016-11-14	2016	$10mlln$ – 100mlln
3205	GRAMERCY	10 COOPS - ELEVATOR APARTMENTS	2	904	50		D4	200 EAST 24TH STREET, 1610		10010.0	...	1972.0	2	D4	399000.0	2005-05-24	514710.0	Gramercy	2005-05-24	2005	$1mlln$ – 10mlln
25396	UPPER WEST SIDE	10 COOPS - ELEVATOR APARTMENTS	2	1890	53		D4	885 WEST END AVENUE, 5B		10025.0	...	1916.0	2	D4	1575000.0	2005-08-11	2031750.0	Upper West Side	2005-08-11	2005	$10mlln$ – 100mlln
2130	FASHION	10 COOPS - ELEVATOR APARTMENTS	2	841	69		D4	32 WEST 40TH STREET, 12B		10018.0	...	1907.0	2	D4	900000.0	2015-04-27	954000.0	Midtown-Midtown South	2015-04-27	2015	$1mlln$ – 10mlln
15610	UPPER EAST SIDE	13 CONDOS - ELEVATOR APARTMENTS	2	1423	1333		R4	200 EAST 69TH STREET	2O	10021.0	...	1991.0	2	R4	1380000.0	2016-12-28	1449000.0	Lenox Hill-Roosevelt Island	2016-12-28	2016	$10mlln$ – 100mlln

5 rows × 25 columns

Linear regression

In [130]:
```
from sklearn import linear_model
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score
```

In [131]:
```
df_price_no_fliers_quntile
```

Out[131]:

	NEIGHBORHOOD	BUILDING CLASS CATEGORY	TAX CLASS AT PRESENT	BLOCK	LOT	EASE-MENT	BUILDING CLASS AT PRESENT	ADDRESS	APARTMENT NUMBER	ZIP CODE	...	GROSS SQUARE FEET	YEAR BUILT	TAX CLASS AT TIME OF SALE	BUILDING CLASS AT TIME OF SALE	SALE PRICE	SALE DATE	SALE PRICE WITH INFLATION	GEOJSON NEIGHBORHOOD	SALE DATE (DT)	SALE YEAR
3	ALPHABET CITY	07 RENTALS - WALKUP APARTMENTS	2B	378	33		C4	125 AVENUE D		10009.0	...	5725.0	1910.0	2	C4	426000.0	2003-10-23	583620.00	Lower East Side	2003-10-23	2003
11	ALPHABET CITY	07 RENTALS - WALKUP APARTMENTS	2	397	6		C7	14 AVENUE A		10009.0	...	8722.0	1900.0	2	C7	1600000.0	2003-09-08	2192000.00	East Village	2003-09-08	2003
12	ALPHABET CITY	07 RENTALS - WALKUP APARTMENTS	2B	401	28		C4	536 EAST 6 STREET		10009.0	...	6200.0	1920.0	2	C4	1060000.0	2003-04-30	1452200.00	East Village	2003-04-30	2003
16	ALPHABET CITY	08 RENTALS - ELEVATOR APARTMENTS	2	394	44		D7	181 AVENUE C		10009.0	...	21328.0	1910.0	2	D7	3000000.0	2003-02-06	4110000.00	Lower East Side	2003-02-06	2003
17	ALPHABET CITY	09 COOPS - WALKUP APARTMENTS	2	373	46		C6	317 EAST 3RD ST, APT 5		10009.0	...	NaN	1925.0	2	C6	190000.0	2003-11-14	260300.00	Lower East Side	2003-11-14	2003
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
16736	WASHINGTON HEIGHTS UPPER	13 CONDOS - ELEVATOR APARTMENTS	2	2180	1402		R4	69 BENNETT AVENUE	102	10033.0	...	805.0	1954.0	2	R4	505000.0	2018-05-10	505000.00	Washington Heights North	2018-05-10	2018
16737	WASHINGTON HEIGHTS UPPER	13 CONDOS - ELEVATOR APARTMENTS	2	2180	1427		R4	69 BENNETT AVENUE	307	10033.0	...	741.0	1954.0	2	R4	510000.0	2017-12-18	520200.00	Washington Heights North	2017-12-18	2017
16738	WASHINGTON HEIGHTS UPPER	13 CONDOS - ELEVATOR APARTMENTS	2	2180	1504		R4	105 BENNETT AVENUE	12B	10033.0	...	810.0	1939.0	2	R4	492804.0	2017-12-27	502660.08	Washington Heights North	2017-12-27	2017
16740	WASHINGTON HEIGHTS UPPER	13 CONDOS - ELEVATOR APARTMENTS	2	2180	1515		R4	105 BENNETT AVENUE	24A	10033.0	...	747.0	1939.0	2	R4	510000.0	2018-05-23	510000.00	Washington Heights North	2018-05-23	2018
16741	WASHINGTON HEIGHTS UPPER	13 CONDOS - ELEVATOR APARTMENTS	2	2180	1522		R4	105 BENNETT AVENUE	32A	10033.0	...	806.0	1939.0	2	R4	549450.0	2018-07-23	549450.00	Washington Heights North	2018-07-23	2018

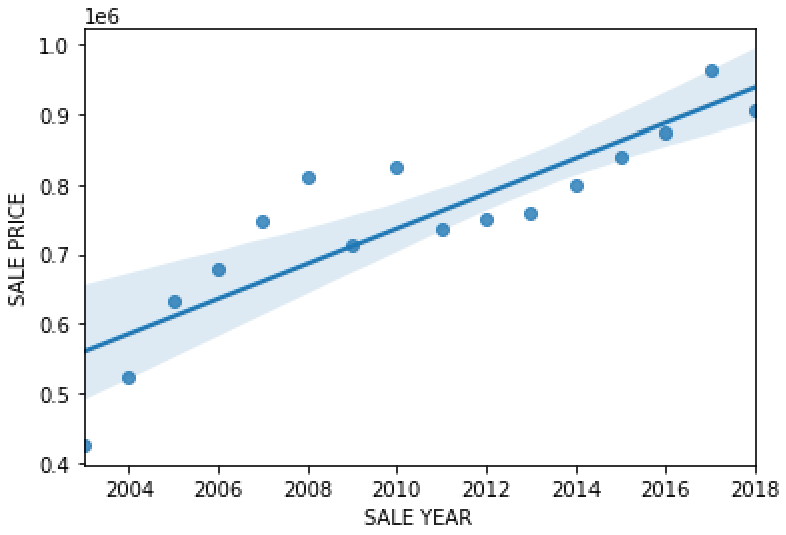
196360 rows × 24 columns

In [132]:
```
df_price_no_fliers_quntile['SALE YEAR'].unique()
```

Out[132]:
```
array([2003, 2004, 2005, 2006, 2007, 2008, 2009, 2010, 2011, 2012, 2013,
       2014, 2015, 2016, 2018, 2017], dtype=int64)
```

In [133]:
```
grouped_year_median = df_price_no_fliers_quntile.groupby(['SALE YEAR'])['SALE PRICE'].median().reset_index()
sns.regplot(data=grouped_year_median, x='SALE YEAR', y='SALE PRICE')

X_train,X_test,y_train,y_test = train_test_split(grouped_year_median,test_size=0.33, random_state=42, shuffle=True)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-133-a19249f93823> in <module>
      2 sns.regplot(data=grouped_year_median, x='SALE YEAR', y='SALE PRICE')
      3 
----> 4 X_train,X_test,y_train,y_test = train_test_split(grouped_year_median,test_size=0.33, random_state=42, shuffle=True)

ValueError: not enough values to unpack (expected 4, got 2)
```



In [ ]:

In [ ]:

In [432]:
```
nyc_neighbourhoods_map = json.load(
    open("data/manhattan_neighbourhoods.geojson"))
```

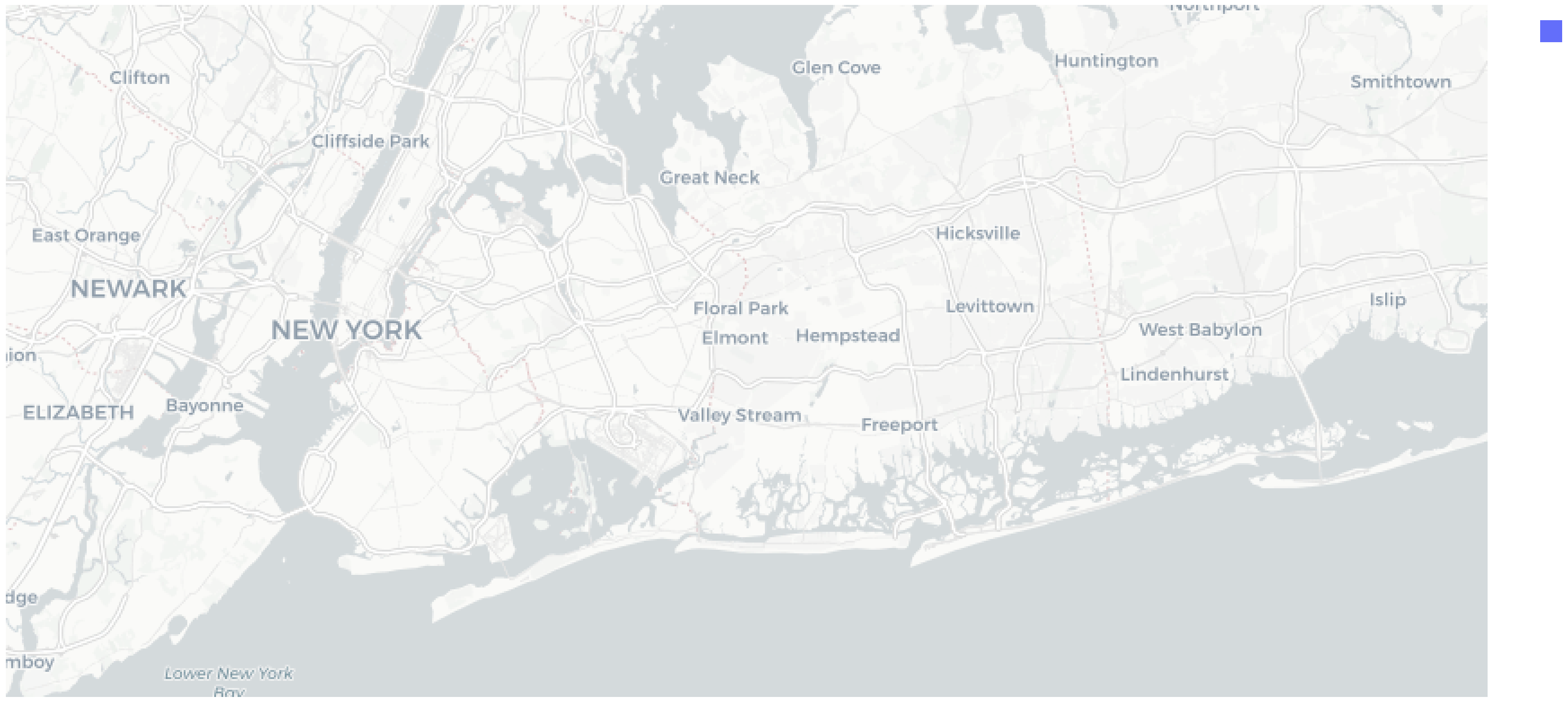
In [433]:
```
nyc_neighbourhoods_map = helpers.remove_non_manhattan_neighbourhoods(
    nyc_neighbourhoods_map)

nyc_neighbourhoods = [x['properties']['ntaname']
                      for x in nyc_neighbourhoods_map['features']]
```

In [434]:
```python
print(nyc_neighbourhoods)
```

['Upper West Side', 'Gramercy', 'Midtown-Midtown South', 'Hudson Yards-Chelsea-Flatiron-Union Square', 'Clinton', 'Yorkville', 'Morningside Heights', 'Central Harlem South', 'Chinatown', 'SoHo-TriBeCa-Civic Center-Little Italy', 'Battery Park City-Lower Manhattan', 'Lower East Side', 'East Harlem South', 'Manhattanville', 'Lincoln Square', 'Turtle Bay-East Midtown', 'Upper East Side-Carnegie Hill', 'Central Harlem North-Polo Grounds', 'Hamilton Heights', 'East Village', 'West Village', 'Washington Heights South', 'Washington Heights North', 'Murray Hill-Kips Bay', 'Stuyvesant Town-Cooper Village', 'Lenox Hill-Roosevelt Island', 'East Harlem North']

In [ ]:
```python
# Get block coords
# particular_block = [x['properties'] for x in nyc_central_park_blocks['features'] if x['properties']['address'] == 2179]
particular_block = [x['properties'] for x in nyc_blocks['features'] if x['properties']['address'] != None and "WEST 24 STREET" in x['properties']['address']]
print(particular_block)
```

In [436]:
```python
fig = px.choropleth_mapbox(tmp,
                           geojson=nycmap,
                           locations="BLOCK",
                           featureidkey="properties.block",
#                            color="SALE PRICE",
                           color_continuous_scale=px.colors.sequential.thermal[::-1],
                        #    range_color=(0, 0.5),
                        #    animation_frame="SALE DATE",
                           mapbox_style="carto-positron",
                           zoom=9, center={"lat": 40.7, "lon": -73.7},
                           opacity=0.7,
                           hover_name="ADDRESS"
                           )

fig.show()
```



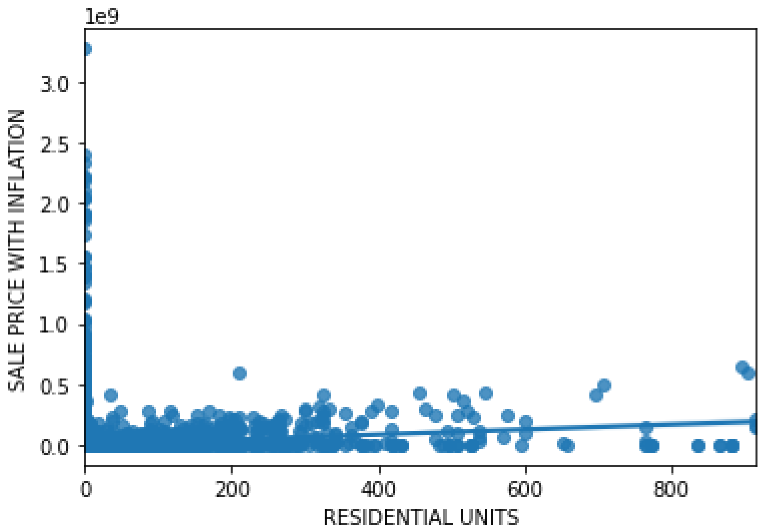
In [164]:
```
corr = df_processed.corr()
corr
```

Out[164]:

	BLOCK	LOT	ZIP CODE	RESIDENTIAL UNITS	COMMERCIAL UNITS	TOTAL UNITS	LAND SQUARE FEET	GROSS SQUARE FEET	YEAR BUILT	TAX CLASS AT TIME OF SALE	SALE PRICE	SALE PRICE WITH INFLATION
BLOCK	1.000000	-0.191156	0.018525	0.012737	-0.015169	0.008481	-0.025500	-0.108208	-0.075356	-0.117562	-0.017430	-0.016915
LOT	-0.191156	1.000000	0.011294	-0.032742	-0.034782	-0.038551	0.046369	-0.069943	0.344074	0.062417	-0.022334	-0.022750
ZIP CODE	0.018525	0.011294	1.000000	0.000919	-0.000306	0.000815	-0.021367	-0.002295	0.006152	-0.013452	-0.002743	-0.002435
RESIDENTIAL UNITS	0.012737	-0.032742	0.000919	1.000000	0.038932	0.973767	0.239768	0.481701	-0.012612	-0.021671	0.489304	0.484576
COMMERCIAL UNITS	-0.015169	-0.034782	-0.000306	0.038932	1.000000	0.264796	0.015485	0.142724	-0.006249	0.091724	0.110817	0.109960
TOTAL UNITS	0.008481	-0.038551	0.000815	0.973767	0.264796	1.000000	0.235468	0.498455	-0.013104	0.002415	0.497798	0.493022
LAND SQUARE FEET	-0.025500	0.046369	-0.021367	0.239768	0.015485	0.235468	1.000000	0.392513	0.019500	0.028173	0.192893	0.191036
GROSS SQUARE FEET	-0.108208	-0.069943	-0.002295	0.481701	0.142724	0.498455	0.392513	1.000000	0.225396	0.251065	0.675022	0.676849
YEAR BUILT	-0.075356	0.344074	0.006152	-0.012612	-0.006249	-0.013104	0.019500	0.225396	1.000000	0.232243	-0.013514	-0.013763
TAX CLASS AT TIME OF SALE	-0.117562	0.062417	-0.013452	-0.021671	0.091724	0.002415	0.028173	0.251065	0.232243	1.000000	0.058027	0.058388
SALE PRICE	-0.017430	-0.022334	-0.002743	0.489304	0.110817	0.497798	0.192893	0.675022	-0.013514	0.058027	1.000000	0.996464
SALE PRICE WITH INFLATION	-0.016915	-0.022750	-0.002435	0.484576	0.109960	0.493022	0.191036	0.676849	-0.013763	0.058388	0.996464	1.000000

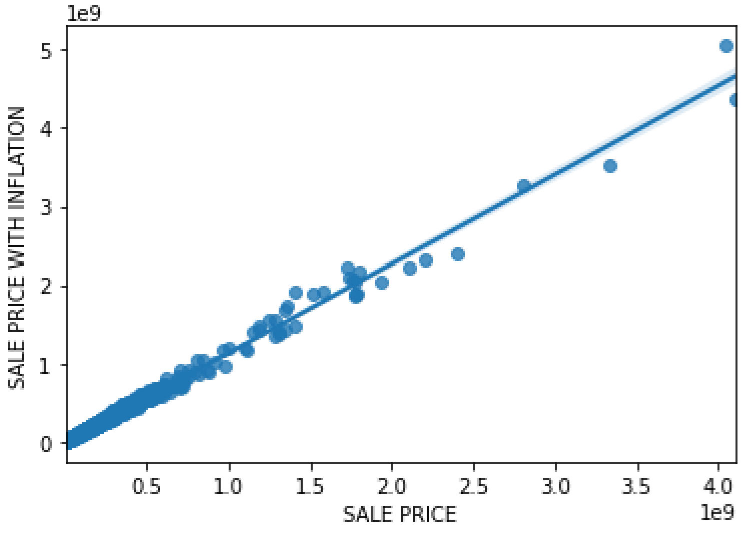
In [167]:
```
tmp = df_processed[df_processed['RESIDENTIAL UNITS'] < 1000]
tmp = tmp[tmp['SALE PRICE WITH INFLATION'] > 100000]

sns.regplot(x=tmp['RESIDENTIAL UNITS'], y=tmp['SALE PRICE WITH INFLATION'])
```

Out[167]:
```
<AxesSubplot:xlabel='RESIDENTIAL UNITS', ylabel='SALE PRICE WITH INFLATION'>
```



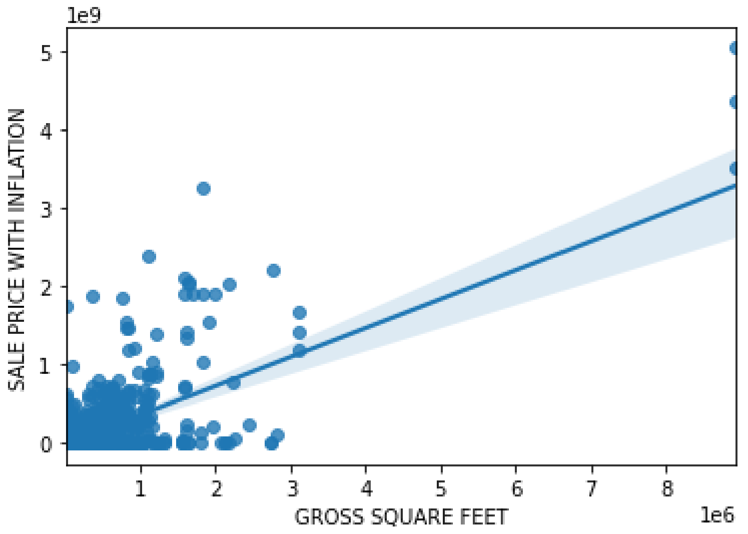
In [168]:
```
tmp = df_processed[df_processed['RESIDENTIAL UNITS'] < 1000]
tmp = tmp[tmp['SALE PRICE WITH INFLATION'] > 100000]

sns.regplot(x=df_processed['SALE PRICE'], y=df_processed['SALE PRICE WITH INFLATION'])
```

Out[168]: <AxesSubplot:xlabel='SALE PRICE', ylabel='SALE PRICE WITH INFLATION'>



In [172]:
```
tmp = df_processed[df_processed['GROSS SQUARE FEET'] < 10000000]

sns.regplot(x=tmp['GROSS SQUARE FEET'], y=tmp['SALE PRICE WITH INFLATION'])
```

Out[172]: <AxesSubplot:xlabel='GROSS SQUARE FEET', ylabel='SALE PRICE WITH INFLATION'>



In [173]:
```
df_processed[]
```

```
C:\Users\juraj\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.8_qbz5n2kfra8p0\LocalCache\local-packages\Python38\site-packages\seaborn\_decorators.py:36: FutureWarning:

Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.


---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-173-7300923b3fea> in <module>
----> 1 sns.regplot(tmp)

~\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.8_qbz5n2kfra8p0\LocalCache\local-packages\Python38\site-packages\seaborn\_decorators.py in inner_f(*args, **kwargs)
     44             )
     45         kwargs.update({k: arg for k, arg in zip(sig.parameters, args)})
---> 46         return f(**kwargs)
     47     return inner_f
     48 

~\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.8_qbz5n2kfra8p0\LocalCache\local-packages\Python38\site-packages\seaborn\regression.py in regplot(x, y, data, x_estimator, x_bins, x_ci, scatter, fit_reg, ci, n_boot, units, seed, order, logistic, lowess, robust, logx, x_partial, y_partial, truncate, dropna, x_jitter, y_jitter, label, color, marker, scatter_kws, line_kws, ax)
    821 ):
    822 
--> 823     plotter = _RegressionPlotter(x, y, data, x_estimator, x_bins, x_ci,
    824                                  scatter, fit_reg, ci, n_boot, units, seed,
    825                                  order, logistic, lowess, robust, logx,

~\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.8_qbz5n2kfra8p0\LocalCache\local-packages\Python38\site-packages\seaborn\regression.py in __init__(self, x, y, data, x_estimator, x_bins, x_ci, scatter, fit_reg, ci, n_boot, units, seed, order, logistic, lowess, robust, logx, x_partial, y_partial, truncate, dropna, x_jitter, y_jitter, color, label)
    107 
    108         # Extract the data vals from the arguments or passed dataframe
--> 109         self.establish_variables(data, x=x, y=y, units=units,
    110                                  x_partial=x_partial, y_partial=y_partial)
    111 

~\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.8_qbz5n2kfra8p0\LocalCache\local-packages\Python38\site-packages\seaborn\regression.py in establish_variables(self, data, **kws)
     53             if np.ndim(vector) > 1:
     54                 err = "regplot inputs must be 1d"
---> 55                 raise ValueError(err)
     56             setattr(self, var, vector)
     57 

ValueError: regplot inputs must be 1d
```

Časť Juraj Ďurej

Merge

- pozor, data v niektorych tabulkach zacinaju na 5 riadku a niektorych na 4tom !!!

In [92]:
```python
df = []

for root, dirs, files in os.walk(data_path, topdown=False):
    for idx, name in enumerate(files):

        if name[len(name)-4:] == '.xls':
            nyc_df = pd.read_excel(os.path.join(root, name),
                                   sheet_name='Manhattan', skiprows=3)
            if idx == 0:
                df = nyc_df

            df.append(nyc_df)
```

In [93]:
```python
print(df.columns)
len(df)
```

```
Index(['BOROUGH', 'NEIGHBORHOOD', 'BUILDING CLASS CATEGORY',
       'TAX CLASS AT PRESENT', 'BLOCK', 'LOT', 'EASE-MENT',
       'BUILDING CLASS AT PRESENT', 'ADDRESS', 'APARTMENT NUMBER', 'ZIP CODE',
       'RESIDENTIAL UNITS', 'COMMERCIAL UNITS', 'TOTAL UNITS',
       'LAND SQUARE FEET', 'GROSS SQUARE FEET', 'YEAR BUILT',
       'TAX CLASS AT TIME OF SALE', 'BUILDING CLASS AT TIME OF SALE',
       'SALE PRICE', 'SALE DATE'],
      dtype='object')
```

Out[93]:
```
26352
```

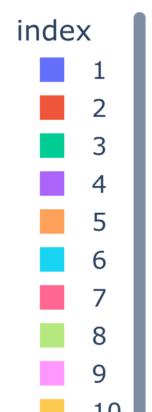
Čistenie datasetu

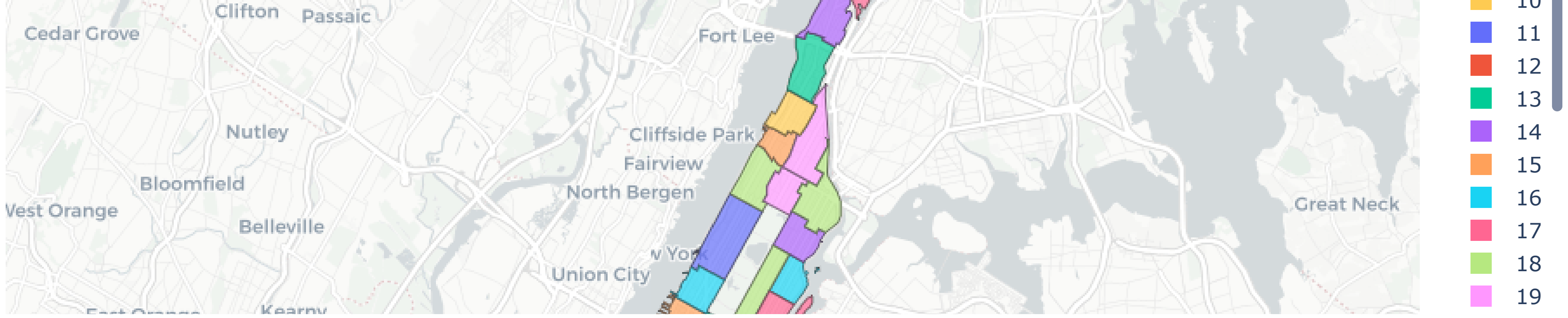
In [94]:
```python
df = df[df['SALE PRICE'] > 10000]
df = df[df['SALE PRICE'] < 100000000]

df = df[df['YEAR BUILT'] != 0]
df = df[df['NEIGHBORHOOD'] != '']
df = df[df['ADDRESS'] != '']
df.ADDRESS = df.ADDRESS.replace(
    '\s+', ' ', regex=True).astype(str).str.strip()
df.BLOCK = df.BLOCK.replace(
    '\s+', ' ', regex=True).astype(str).str.strip()
df = df[df['SALE DATE'] != '']
```

Hypoteza 1. Nehnutelnosti v okoli central su parku drahsie

In [95]:

```
# load geojson data for manhattan
nycmap = json.load(open("data/manhattan.geojson"))

# nacitanie zoznamu central park blokov
nyc_central_park_blocks = json.load(open("data/manhattan_central_park.geojson"))
nyc_central_park_addresses = [x["properties"] for x in nyc_central_park_blocks['features']]
nyc_central_park_blocks = set([x["block"] for x in nyc_central_park_addresses])


# pre kazdy neighbourhood dat zoznam blokov co don patria


# konvertovanie datumu predaja na rok s mesiacom
df['SALE DATE'] = df['SALE DATE'] + pd.offsets.MonthBegin(-1)
df['SALE DATE'] = pd.to_datetime(
    df['SALE DATE']).dt.normalize().dt.strftime('%Y-%m-%d')


nyc_neighbourhoods_map = json.load(
    open("data/manhattan_neighbourhoods.geojson"))

nyc_neighbourhoods_map = helpers.remove_non_manhattan_neighbourhoods(
    nyc_neighbourhoods_map)

nyc_neighbourhoods = [x['properties']['ntaname']
                      for x in nyc_neighbourhoods_map['features']]


# zobrazenie vsetkych neighbourhoods

# df = pd.DataFrame(list(zip(nyc_neighbourhoods, [str(x+1) for x in range(0, len(nyc_neighbourhoods))])), columns=['name','index'])
# fig = px.choropleth_mapbox(df,
#                            geojson=nyc_neighbourhoods_map,
#                            locations="name",
#                            featureidkey="properties.ntaname",
#                            color="index",
#                            color_continuous_scale=px.colors.sequential.thermal[::-1],
#                         #    range_color=(0, 0.5),
#                         #    animation_frame="SALE DATE",
#                            mapbox_style="carto-positron",
#                            zoom=10, center={"lat": 40.761415, "lon": -73.979644},
#                            opacity=0.7,
#                            hover_name="name"
#                            )


# fig.show()
```





Cedar Grove
Clifton
Passaic
Fort Lee
Nutley
Cliffside Park
Fairview
Bloomfield
North Bergen
West Orange
Belleville
Union City
Great Neck
11
12
13
14
15
16
17
18
19

In [97]:

```
# zobrazenie vsetkych blokov s cenami
fig = px.choropleth_mapbox(df,
                           geojson=nycmap,
                           locations="BLOCK",
                           featureidkey="properties.block",
                           color="SALE PRICE",
                           color_continuous_scale=px.colors.sequential.thermal[::-1],
                        #    range_color=(0, 0.5),
                        #    animation_frame="SALE DATE",
                           mapbox_style="carto-positron",
                           zoom=9, center={"lat": 40.7, "lon": -73.7},
                           opacity=0.7,
                           hover_name="ADDRESS"
                           )

fig.show()
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-97-d2cabd025c46> in <module>
      1 # zobrazenie vsetkych blokov s cenami
----> 2 fig = px.choropleth_mapbox(df,
      3                            geojson=nycmap,
      4                            locations="BLOCK",
      5                            featureidkey="properties.block",

~\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.8_qbz5n2kfra8p0\LocalCache\local-packages\Python38\site-packages\plotly\express\_chart_types.py in choropleth_mapbox(data_frame, geojson, featureidkey, locations, color, hover_name, hover_data, custom_data, animation_frame, animation_group, category_orders, labels, color_discrete_sequence, color_discrete_map, color_continuous_scale, range_color, color_continuous_midpoint, opacity, zoom, center, mapbox_style, title, template, width, height)
   1268     colored region on a Mapbox map.
   1269     """
-> 1270     return make_figure(args=locals(), constructor=go.Choroplethmapbox)
   1271
   1272

~\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.8_qbz5n2kfra8p0\LocalCache\local-packages\Python38\site-packages\plotly\express\_core.py in make_figure(args, constructor, trace_patch, layout_patch)
   1943     apply_default_cascade(args)
   1944
-> 1945     args = build_dataframe(args, constructor)
   1946     if constructor in [go.Treemap, go.Sunburst, go.Icicle] and args["path"] is not None:
   1947         args = process_dataframe_hierarchy(args)

~\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.8_qbz5n2kfra8p0\LocalCache\local-packages\Python38\site-packages\plotly\express\_core.py in build_dataframe(args, constructor)
   1403     # now that things have been prepped, we do the systematic rewriting of `args`
   1404
-> 1405     df_output, wide_id_vars = process_args_into_dataframe(
   1406         args, wide_mode, var_name, value_name
   1407     )

~\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.8_qbz5n2kfra8p0\LocalCache\local-packages\Python38\site-packages\plotly\express\_core.py in process_args_into_dataframe(args, wide_mode, var_name, value_name)
   1205                     if argument == "index":
   1206                         err_msg += "\n To use the index, pass it in directly as `df.index`."
-> 1207                     raise ValueError(err_msg)
   1208             elif length and len(df_input[argument]) != length:
   1209                 raise ValueError(

ValueError: Value of 'hover_name' is not the name of a column in 'data_frame'. Expected one of ['name', 'index'] but received: ADDRESS
```

In [ ]: