In [1]:
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn import metrics
from statsmodels.tools import eval_measures
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LassoCV, Ridge, RidgeCV
```

In [2]:
```
train_data = pd.read_csv('train_2017.csv')
property_data = pd.read_csv('properties_2017.csv')

combined_data = pd.merge(train_data, property_data, how = "inner", on = ["parcelid"])
```

```
C:\Users\Daniel\anaconda3\lib\site-packages\IPython\core\interactiveshell.py:3165: DtypeWarning: Columns (4
9) have mixed types.Specify dtype option on import or set low_memory=False.
  has_raised = await self.run_ast_nodes(code_ast.body, cell_name,
```

In [3]:
```
train_data, val_data = train_test_split(combined_data, train_size = 0.7, random_state = 0)
```

In [4]:
```
missing_ratios = train_data.isna().sum() / train_data.shape[0]
missing_ratios.sort_values(ascending = False)
```

Out[4]:
```
buildingclasstypeid             0.999798
finishedsquarefeet13            0.999466
basementsqft                    0.999374
storytypeid                     0.999374
yardbuildingsqft26              0.999135
fireplaceflag                   0.997920
architecturalstyletypeid        0.997386
typeconstructiontypeid          0.997110
finishedsquarefeet6             0.995196
pooltypeid10                    0.993815
decktypeid                      0.992453
poolsizesum                     0.989251
pooltypeid2                     0.986122
hashottuborspa                  0.979937
yardbuildingsqft17              0.969445
```

taxdelinquencyyear	0.962028
taxdelinquencyflag	0.962028
finishedsquarefeet15	0.961052
finishedfloor1squarefeet	0.923098
finishedsquarefeet50	0.923098
fireplacecnt	0.893666
threequarterbathnbr	0.869904
pooltypeid7	0.804487
poolcnt	0.790315
numberofstories	0.773602
airconditioningtypeid	0.677391
garagecarcnt	0.672256
garagetotalsqft	0.672256
regionidneighborhood	0.600232
heatingorsystemtypeid	0.360618
buildingqualitytypeid	0.357194
propertyzoningdesc	0.348525
unitcnt	0.345856
lotsizesquarefeet	0.105984
finishedsquarefeet12	0.047010
regionidcity	0.019308
fullbathcnt	0.007915
calculatedbathnbr	0.007915
yearbuilt	0.003589
censustractandblock	0.003589
calculatedfinishedsquarefeet	0.002724
structuretaxvaluedollarcnt	0.001933
regionidzip	0.001012
taxamount	0.000442
landtaxvaluedollarcnt	0.000405
bathroomcnt	0.000387
assessmentyear	0.000387
taxvaluedollarcnt	0.000387
bedroomcnt	0.000387
regionidcounty	0.000387
fips	0.000387
latitude	0.000387
longitude	0.000387
propertycountylandusecode	0.000387
propertylandusetypeid	0.000387
rawcensustractandblock	0.000387
roomcnt	0.000387
logerror	0.000000
transactiondate	0.000000

```
parcelid                        0.000000
```

In [5]:
```
numeric_fields = ["roomcnt", "bedroomcnt", "bathroomcnt", "assessmentyear", "landtaxvaluedollarcnt", "taxamo
                  "structuretaxvaluedollarcnt", "calculatedfinishedsquarefeet","yearbuilt", "lotsizesquarefe
                  "finishedsquarefeet12", "fullbathcnt", "calculatedbathnbr", "unitcnt"]
```

In [6]:
```
medians = train_data[numeric_fields].median()
train_data[numeric_fields] = train_data[numeric_fields].fillna(medians)
val_data[numeric_fields] = val_data[numeric_fields].fillna(medians)
```

```
C:\Users\Daniel\anaconda3\lib\site-packages\pandas\core\frame.py:3191: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
  self[k1] = value[k2]
```

In [7]:
```
train_data[numeric_fields].isna().sum()
```

Out[7]:
```
roomcnt                         0
bedroomcnt                      0
bathroomcnt                     0
assessmentyear                  0
landtaxvaluedollarcnt           0
taxamount                       0
structuretaxvaluedollarcnt      0
calculatedfinishedsquarefeet    0
yearbuilt                       0
lotsizesquarefeet               0
finishedsquarefeet12            0
fullbathcnt                     0
calculatedbathnbr               0
unitcnt                         0
dtype: int64
```

In [8]:
```
train_data.fireplaceflag.value_counts(dropna = False)
```

Out[8]:
```
NaN     54216
True      113
Name: fireplaceflag, dtype: int64
```

In [9]:
```
train_data.hashottuborspa.value_counts(dropna = False)
```

Out[9]:
```
NaN     53239
True     1090
Name: hashottuborspa, dtype: int64
```

In [10]:
```
train_data.numberofstories.value_counts(dropna = False)
```

Out[10]:
```
NaN    42029
1.0     7266
2.0     4716
3.0      318
Name: numberofstories, dtype: int64
```

In [11]:
```
train_data.fips.value_counts(dropna = False)
```

Out[11]:
```
6037.0    35554
6059.0    14433
6111.0     4321
NaN          21
Name: fips, dtype: int64
```

In [12]:
```
train_data.propertyzoningdesc.value_counts(dropna = False)
```

Out[12]:
```
NaN          18935
LAR1          4738
LAR3          1735
LARS           973
LBR1N          860
              ...
BHR475*          1
HBC2YY           1
LPR2YY           1
CSRS*            1
```

```
CLRM3000*        1
```

In [13]:
```
train_data.unitcnt.value_counts(dropna = False)
```

Out[13]:
```
1.0     52043
2.0      1401
4.0       516
3.0       368
45.0        1
Name: unitcnt, dtype: int64
```

In [14]:
```
train_data.censustractandblock.value_counts(dropna = False)
```

Out[14]:
```
NaN             195
6.037137e+13     43
6.059063e+13     33
6.037277e+13     33
6.037577e+13     32
                ...
6.059064e+13      1
6.037268e+13      1
6.037111e+13      1
6.037533e+13      1
6.037532e+13      1
Name: censustractandblock, Length: 31264, dtype: int64
```

In [15]:
```
train_data.poolcnt.value_counts(dropna = False)
```

Out[15]:
```
NaN    42937
1.0    11392
Name: poolcnt, dtype: int64
```

In [16]:
```
zero_vars = ["poolcnt", "fireplaceflag", "fireplacecnt", "garagecarcnt", "threequarterbathnbr"]

train_data[zero_vars] = train_data[zero_vars].fillna(0)
val_data[zero_vars] = val_data[zero_vars].fillna(0)
```

In [17]:
```
total_vars = numeric_fields + zero_vars
total_vars
```

Out[17]:
```
['roomcnt',
 'bedroomcnt',
 'bathroomcnt',
 'assessmentyear',
 'landtaxvaluedollarcnt',
 'taxamount',
 'structuretaxvaluedollarcnt',
 'calculatedfinishedsquarefeet',
 'yearbuilt',
 'lotsizesquarefeet',
 'finishedsquarefeet12',
 'fullbathcnt',
 'calculatedbathnbr',
 'unitcnt',
 'poolcnt',
 'fireplaceflag',
 'fireplacecnt',
 'garagecarcnt',
 'threequarterbathnbr']
```

In [18]:
```
train_data[total_vars].isna().sum()
```

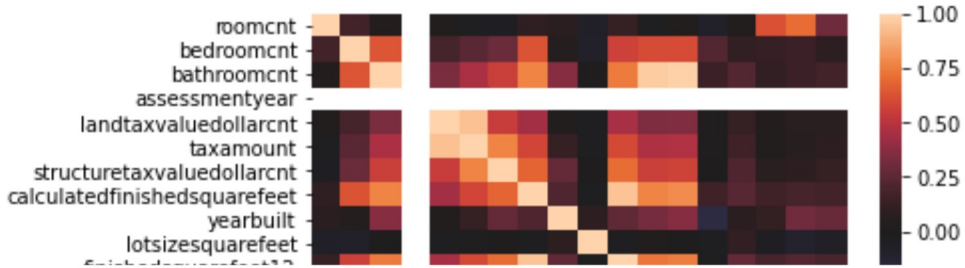
Out[18]:
```
roomcnt                       0
bedroomcnt                    0
bathroomcnt                   0
assessmentyear                0
landtaxvaluedollarcnt         0
taxamount                     0
structuretaxvaluedollarcnt    0
calculatedfinishedsquarefeet  0
yearbuilt                     0
lotsizesquarefeet             0
finishedsquarefeet12          0
fullbathcnt                   0
calculatedbathnbr             0
unitcnt                       0
poolcnt                       0
fireplaceflag                 0
fireplacecnt                  0
garagecarcnt                  0
threequarterbathnbr           0
dtype: int64
```

In [19]:
```
val_data[total_vars].isna().sum()
```

Out[19]:
```
roomcnt                         0
bedroomcnt                      0
bathroomcnt                     0
assessmentyear                  0
landtaxvaluedollarcnt           0
taxamount                       0
structuretaxvaluedollarcnt      0
calculatedfinishedsquarefeet    0
yearbuilt                       0
lotsizesquarefeet               0
finishedsquarefeet12            0
fullbathcnt                     0
calculatedbathnbr               0
unitcnt                         0
poolcnt                         0
fireplaceflag                   0
fireplacecnt                    0
garagecarcnt                    0
threequarterbathnbr             0
dtype: int64
```

In [20]:
```
sns.heatmap(train_data[total_vars].corr(), center = 0, vmin = -1, vmax = 1)
```

Out[20]:
```
<AxesSubplot:>
```



In [21]:
```
train_data.assessmentyear.value_counts()
```

Out[21]:
```
2016.0    54329
Name: assessmentyear, dtype: int64
```

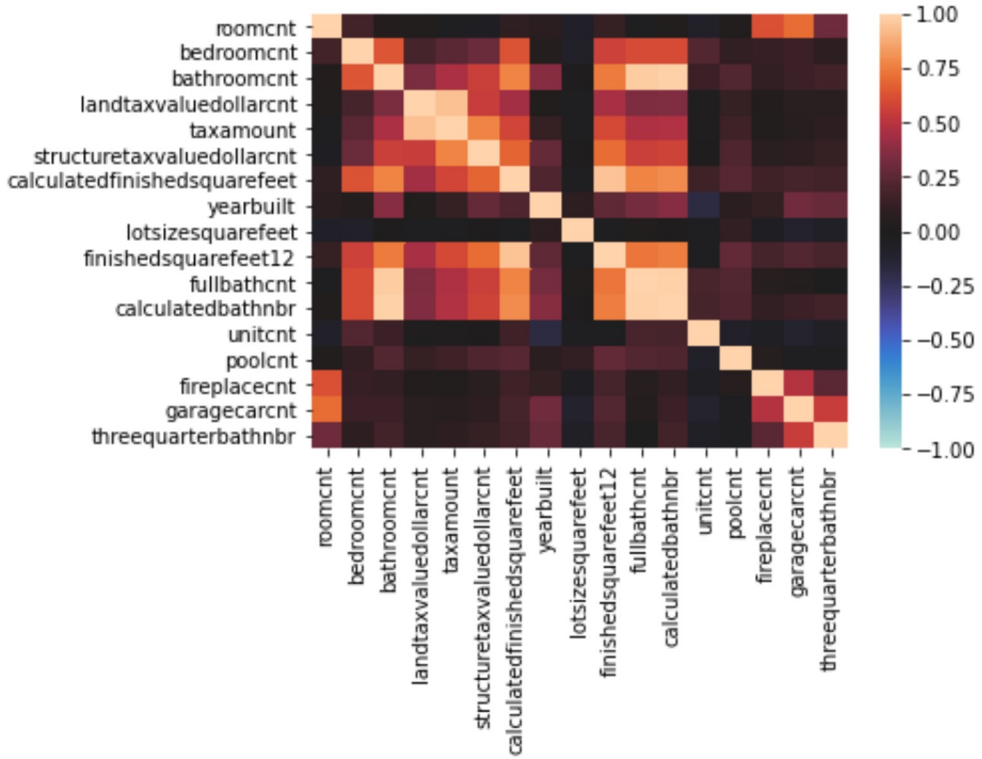
In [22]:
```
total_vars.remove("assessmentyear")
```

In [23]:
```
total_vars
```

Out[23]:
```
['roomcnt',
 'bedroomcnt',
 'bathroomcnt',
 'landtaxvaluedollarcnt',
 'taxamount',
 'structuretaxvaluedollarcnt',
 'calculatedfinishedsquarefeet',
 'yearbuilt',
 'lotsizesquarefeet',
 'finishedsquarefeet12',
 'fullbathcnt',
 'calculatedbathnbr',
 'unitcnt',
 'poolcnt',
 'fireplaceflag',
 'fireplacecnt',
 'garagecarcnt',
 'threequarterbathnbr']
```

In [24]:
```
sns.heatmap(train_data[total_vars].corr(), center = 0, vmin = -1, vmax = 1)
```

Out[24]:
```
<AxesSubplot:>
```



In [25]:
```
train_data[total_vars].corr()
```

Out[25]:

	roomcnt	bedroomcnt	bathroomcnt	landtaxvaluedollarcnt	taxamount	structuretaxvaluedollarcnt	calculatedfin
roomcnt	1.000000	0.167513	0.036087	0.018057	-0.029972	-0.040510	
bedroomcnt	0.167513	1.000000	0.635328	0.180356	0.247778	0.293925	
bathroomcnt	0.036087	0.635328	1.000000	0.342874	0.462715	0.559260	
landtaxvaluedollarcnt	0.018057	0.180356	0.342874	1.000000	0.943501	0.550519	
taxamount	-0.029972	0.247778	0.462715	0.943501	1.000000	0.772683	
structuretaxvaluedollarcnt	-0.040510	0.293925	0.559260	0.550519	0.772683	1.000000	
calculatedfinishedsquarefeet	0.095767	0.626984	0.767573	0.441731	0.581141	0.685534	
yearbuilt	0.072521	0.046063	0.367810	0.020719	0.118535	0.275038	

	roomcnt	bedroomcnt	bathroomcnt	landtaxvaluedollarcnt	taxamount	structuretaxvaluedollarcnt	calculatedfin
lotsizesquarefeet	-0.077783	-0.083789	0.007059	-0.027951	-0.017236	0.007025	
finishedsquarefeet12	0.120198	0.569530	0.742683	0.453806	0.594413	0.703318	
fullbathcnt	-0.022970	0.602049	0.969079	0.347664	0.470124	0.561616	
calculatedbathnbr	0.029480	0.608625	0.985181	0.355093	0.478086	0.575668	
unitcnt	-0.085690	0.218961	0.141383	0.010091	0.011912	-0.006621	
poolcnt	0.028466	0.103065	0.221180	0.119975	0.161451	0.212533	
fireplacecnt	0.622113	0.132316	0.102155	0.037508	0.028800	0.062993	
garagecarcnt	0.705719	0.141967	0.147191	0.070100	0.056252	0.091875	

In [26]:
```
remove_vars = ["landtaxvaluedollarcnt", "calculatedbathnbr", "fullbathcnt"]
total_vars = [var for var in total_vars if var not in remove_vars]
```

In [27]:
```
total_vars
```

Out[27]:
```
['roomcnt',
 'bedroomcnt',
 'bathroomcnt',
 'taxamount',
 'structuretaxvaluedollarcnt',
 'calculatedfinishedsquarefeet',
 'yearbuilt',
 'lotsizesquarefeet',
 'finishedsquarefeet12',
 'unitcnt',
 'poolcnt',
 'fireplaceflag',
 'fireplacecnt',
 'garagecarcnt',
 'threequarterbathnbr']
```

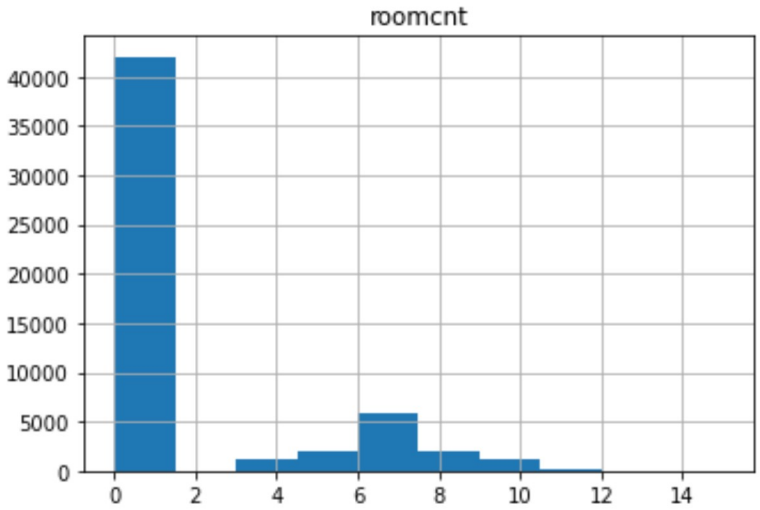
In [28]:
```
pd.set_option('display.float_format', lambda x: '%.5f' % x)
```

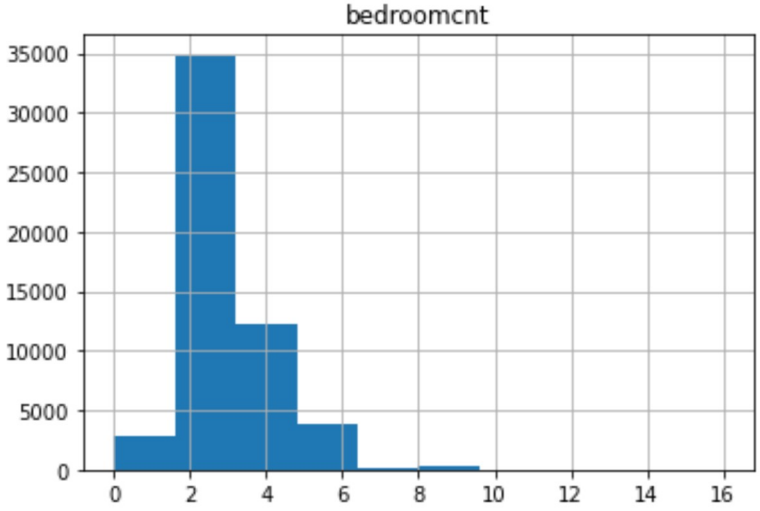
In [29]:
```
train_data[total_vars].describe()
```

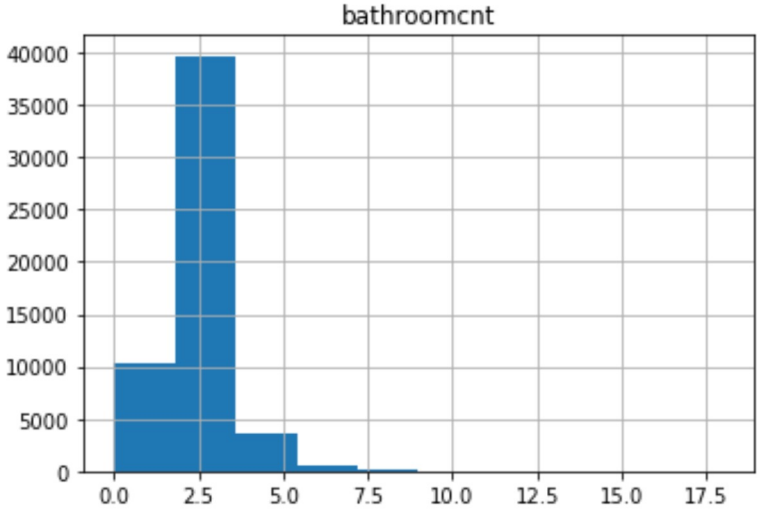
Out[29]:

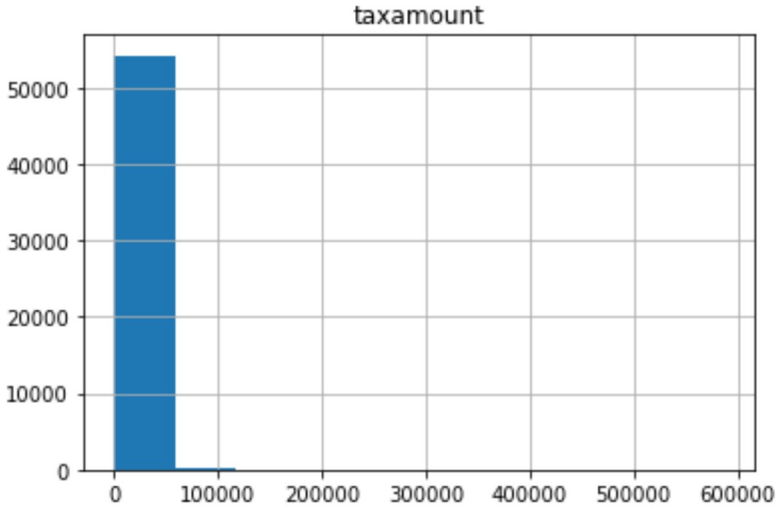
	roomcnt	bedroomcnt	bathroomcnt	taxamount	structuretaxvaluedollarcnt	calculatedfinishedsquarefeet	yearbuilt	lotsizesqu
count	54329.00000	54329.00000	54329.00000	54329.00000	54329.00000	54329.00000	54329.00000	5432
mean	1.47549	3.05846	2.30197	5998.61746	189727.60871	1786.46287	1968.58992	2762
std	2.82606	1.13803	0.99751	7675.39277	232833.64359	958.36437	23.74884	12243
min	0.00000	0.00000	0.00000	19.92000	44.00000	128.00000	1824.00000	43
25%	0.00000	2.00000	2.00000	2728.52000	84740.00000	1186.00000	1953.00000	595
50%	0.00000	3.00000	2.00000	4461.24000	136959.50000	1542.00000	1970.00000	720
75%	0.00000	4.00000	3.00000	6933.30000	219113.00000	2114.00000	1987.00000	1053
max	15.00000	16.00000	18.00000	586639.30000	11421790.00000	35640.00000	2016.00000	697101

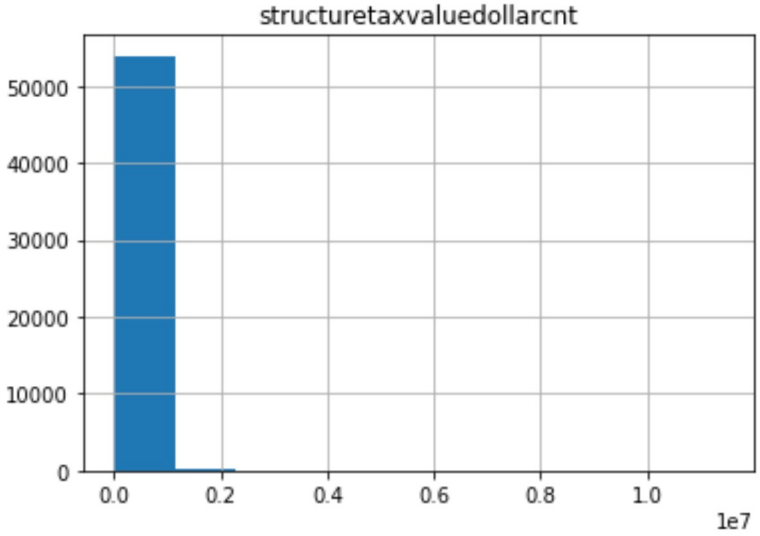
In [30]:
```
for var in total_vars:
    train_data[var].hist()
    plt.title(var)
    plt.show()
```

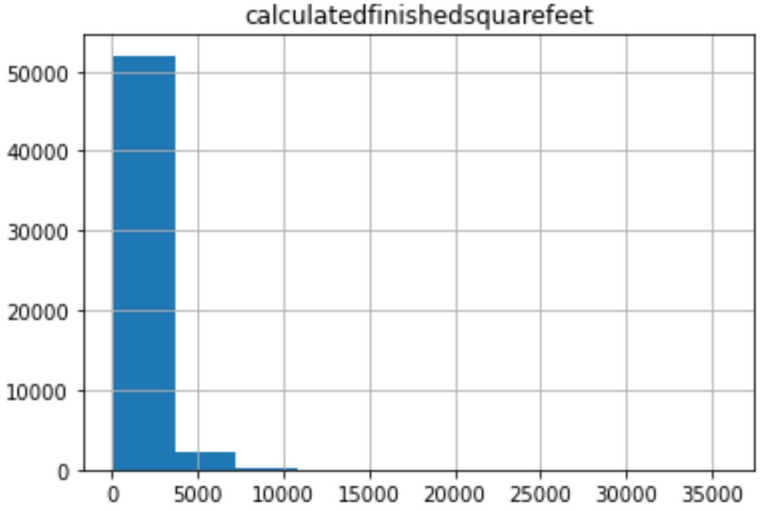


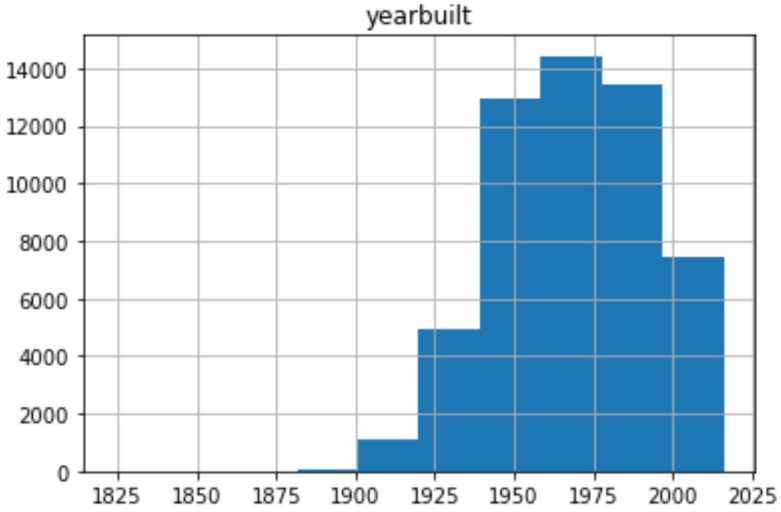

bedroomcnt
35000
30000
25000
20000
15000
10000
5000
0
0
2
4
6
8
10
12
14
16

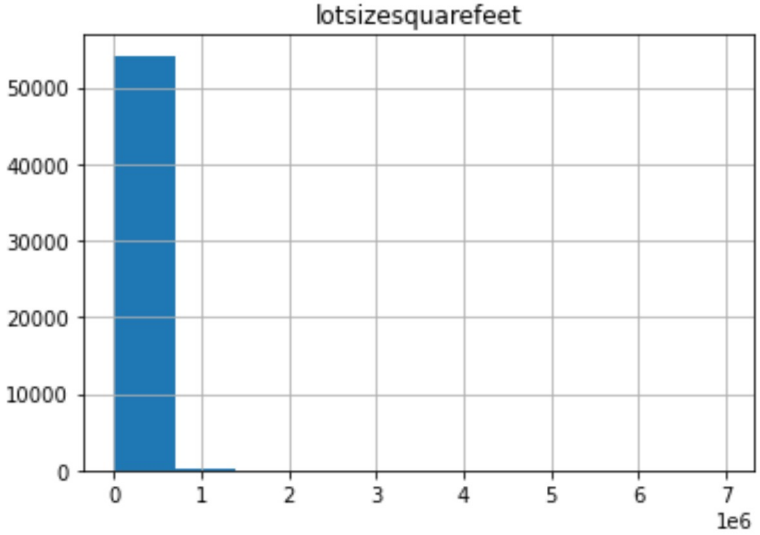

bathroomcnt
40000
35000
30000
25000
20000
15000
10000
5000
0
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5

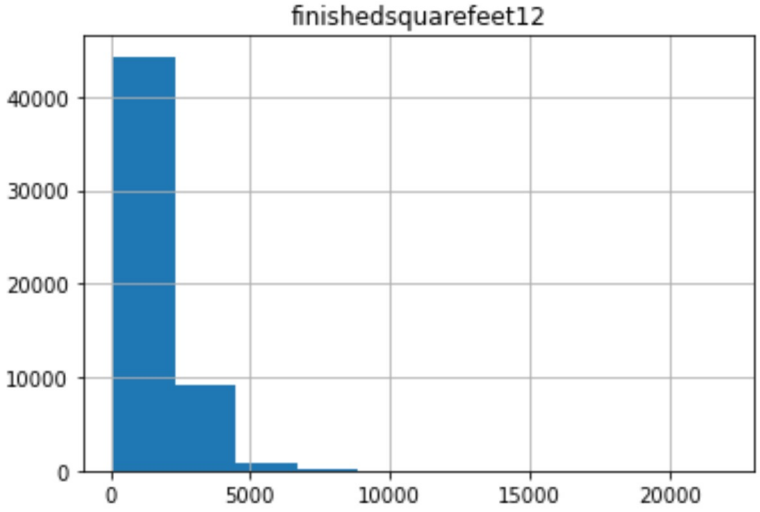

taxamount
50000
40000
30000
20000
10000
0
0
100000
200000
300000
400000
500000
600000

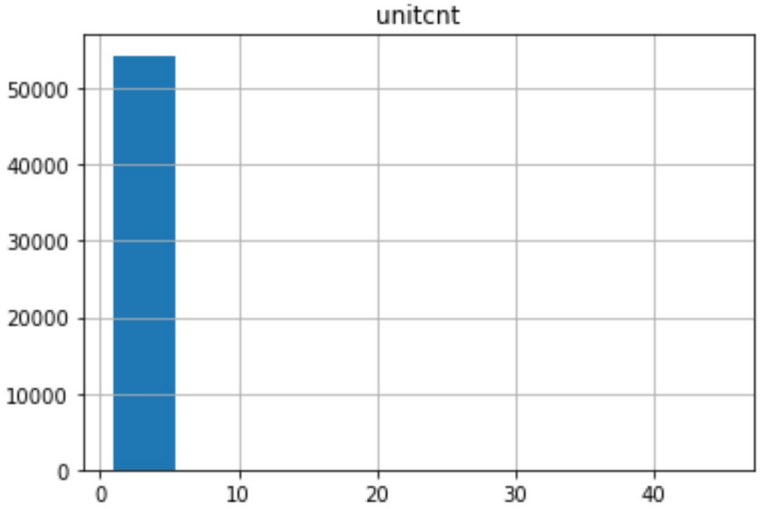

structuretaxvaluedollarcnt
50000
40000
30000
20000
10000
0
0.0
0.2
0.4
0.6
0.8
1.0
1e7

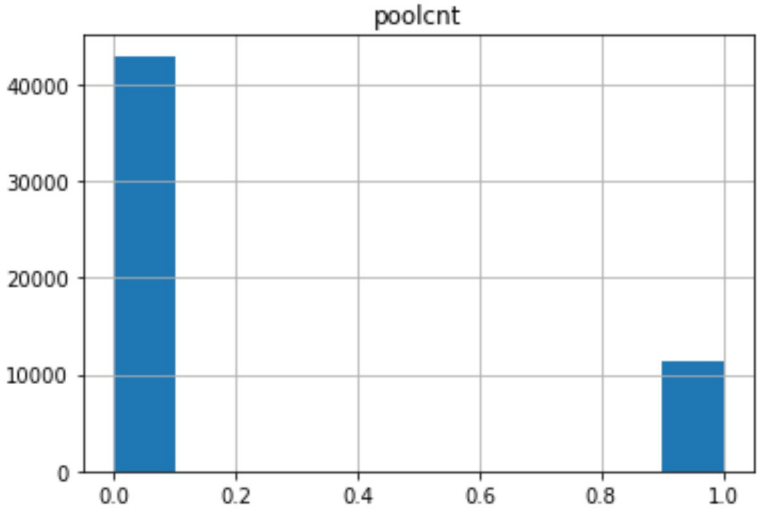

calculatedfinishedsquarefeet
50000
40000
30000
20000
10000
0
0
5000
10000
15000
20000
25000
30000
35000


yearbuilt
14000
12000
10000
8000
6000
4000
2000
0
1825
1850
1875
1900
1925
1950
1975
2000
2025


lotsizesquarefeet
50000
40000
30000
20000
10000
0
0
1
2
3
4
5
6
7
1e6


finishedsquarefeet12
40000
30000
20000
10000
0
0
5000
10000
15000
20000


unitcnt
50000
40000
30000
20000
10000
0
0
10
20
30
40


poolcnt
40000
30000
20000
10000
0
0.0
0.2
0.4
0.6
0.8
1.0

In [31]:
```
total_vars.remove("roomcnt")
```

In [32]:
```
total_vars
```

Out[32]:
```
['bedroomcnt',
 'bathroomcnt',
 'taxamount',
 'structuretaxvaluedollarcnt',
 'calculatedfinishedsquarefeet',
 'yearbuilt',
 'lotsizesquarefeet',
 'finishedsquarefeet12',
 'unitcnt',
 'poolcnt',
 'fireplaceflag',
 'fireplacecnt',
 'garagecarcnt',
 'threequarterbathnbr']
```

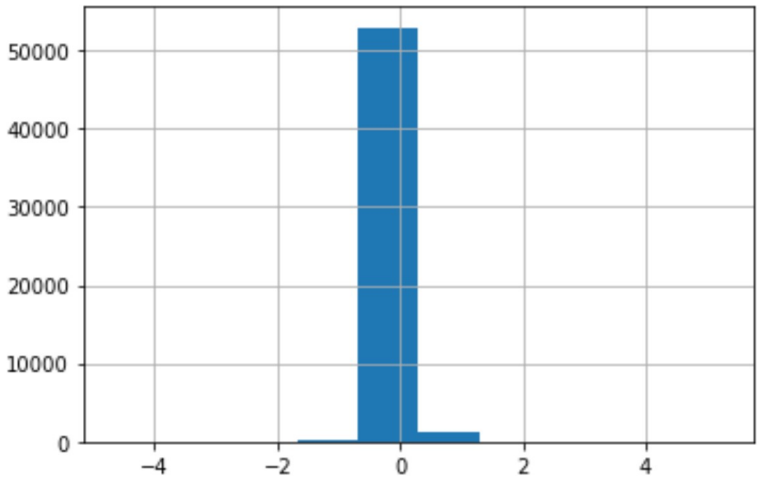
In [33]:
```
for var in ["taxamount", "structuretaxvaluedollarcnt", "calculatedfinishedsquarefeet",
            "lotsizesquarefeet", "finishedsquarefeet12"]:
    upper = train_data[var].quantile(.99)
    lower = train_data[var].quantile(.01)
    train_data[var] = [lower if val < lower else upper if val > upper else val for val in train_data[var]]
    val_data[var] = [lower if val < lower else upper if val > upper else val for val in val_data[var]]
```

```
<ipython-input-33-d84c404b9acc>:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
```

```
  train_data[var] = [lower if val < lower else upper if val > upper else val for val in train_data[var]]
<ipython-input-33-d84c404b9acc>:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
  val_data[var] = [lower if val < lower else upper if val > upper else val for val in val_data[var]]
```

In [34]:
```
train_data.logerror.hist()
```

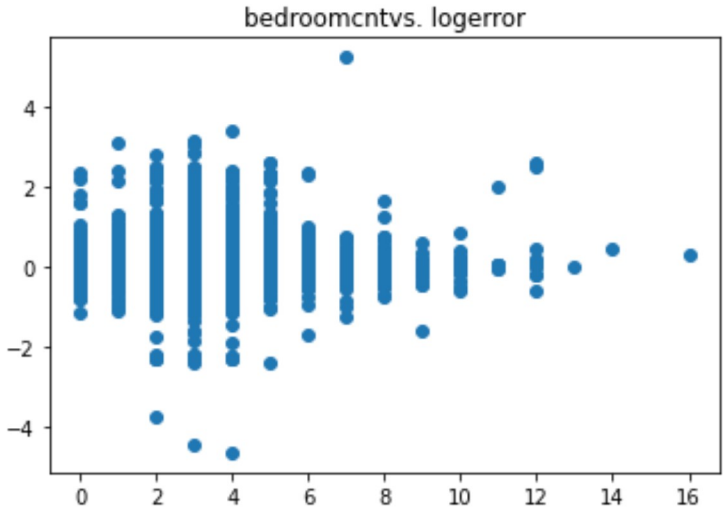
Out[34]:
```
<AxesSubplot:>
```

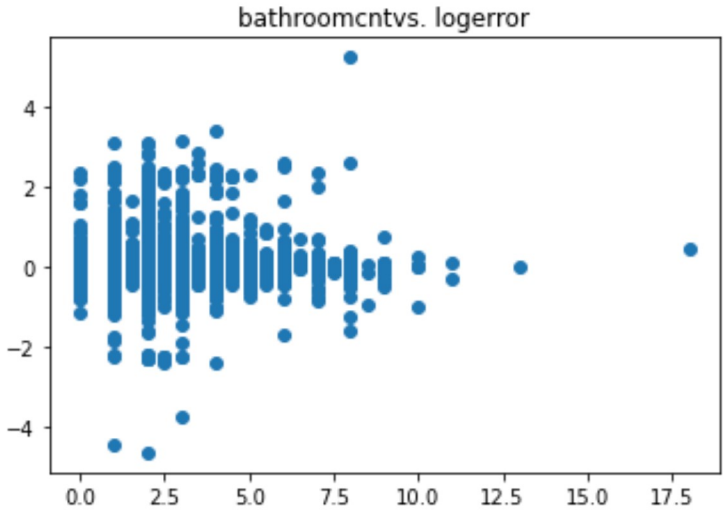


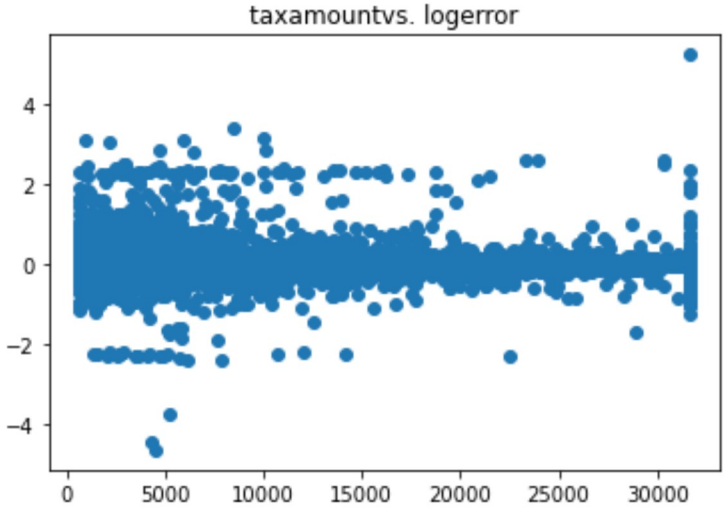
In [35]:
```
train_data.logerror.describe()
```

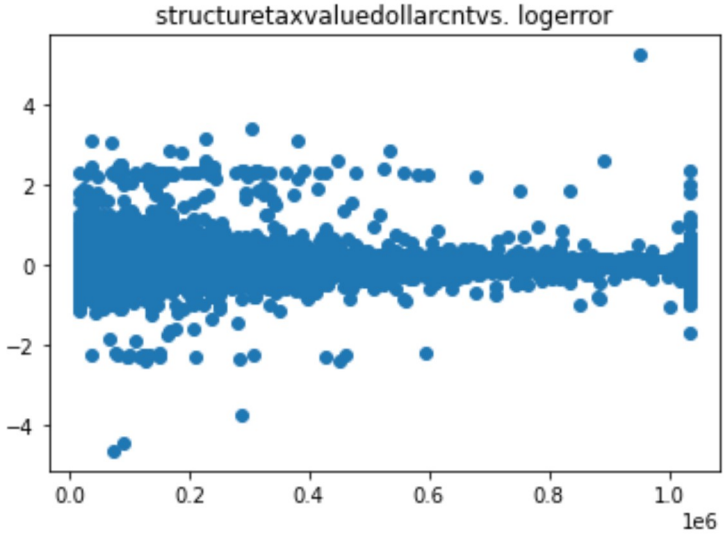
Out[35]:
```
count   54329.00000
mean        0.01680
std         0.17268
min        -4.65542
25%        -0.02426
50%         0.00668
75%         0.03921
max         5.26300
Name: logerror, dtype: float64
```

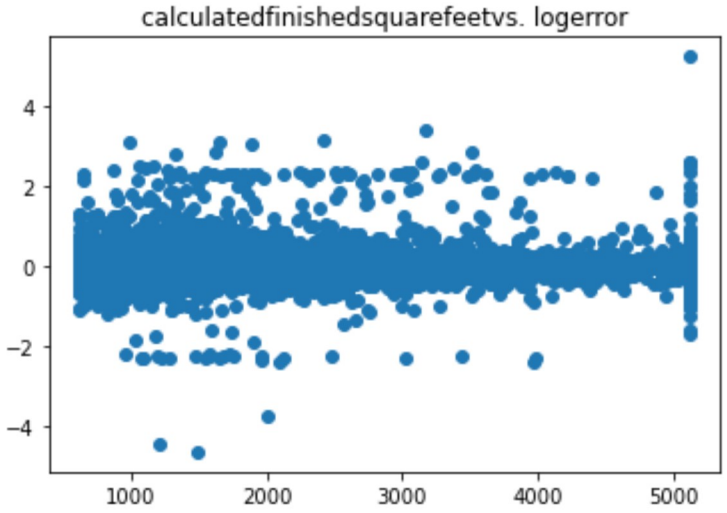
In [36]:
```
for var in total_vars:
    plt.scatter(train_data[var], train_data.logerror)
    plt.title(var + "vs. logerror")
    plt.show()
```

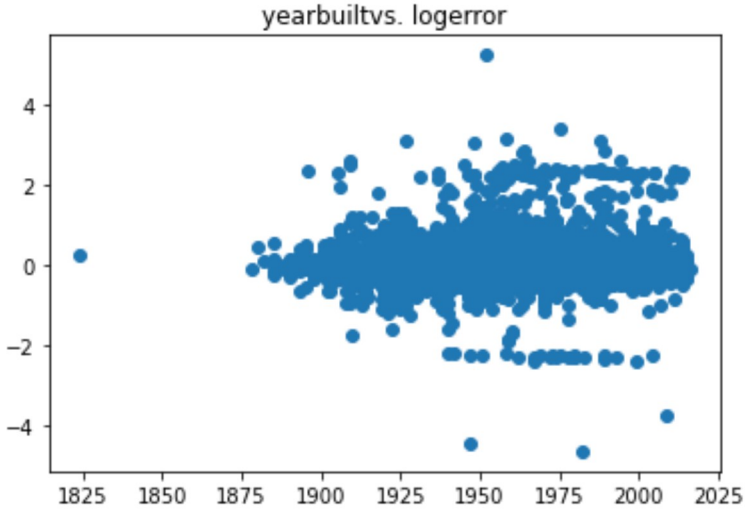


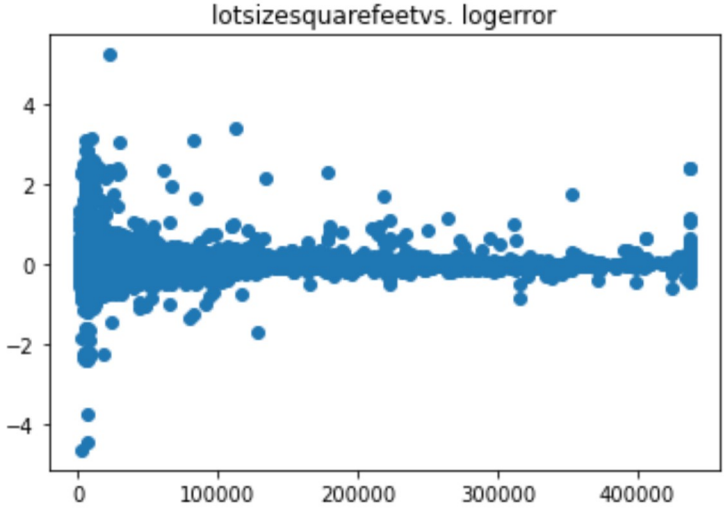


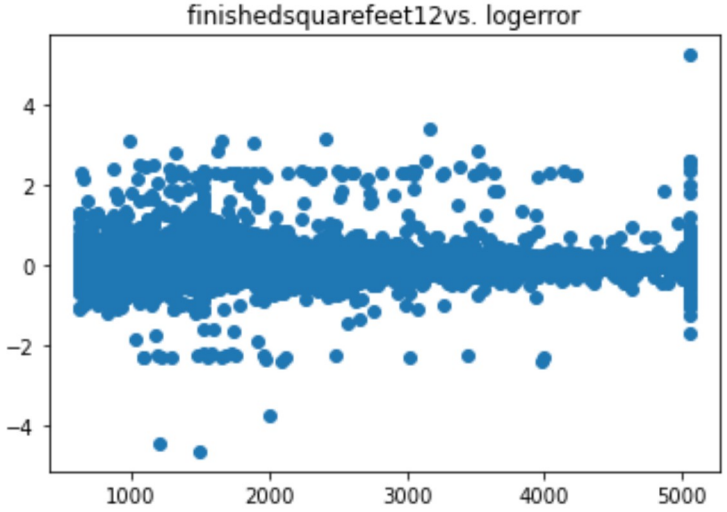

taxamountvs. logerror
4
2
0
−2
−4
0
5000
10000
15000
20000
25000
30000

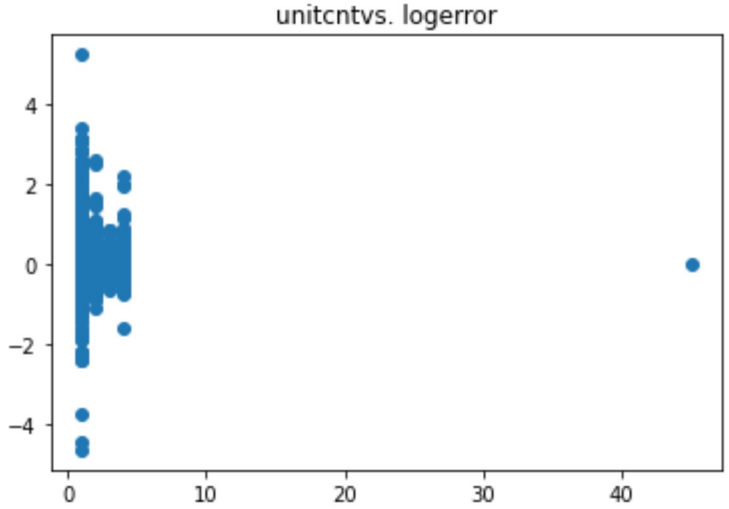

structuretaxvaluedollarcntvs. logerror
4
2
0
−2
−4
0.0
0.2
0.4
0.6
0.8
1.0
1e6

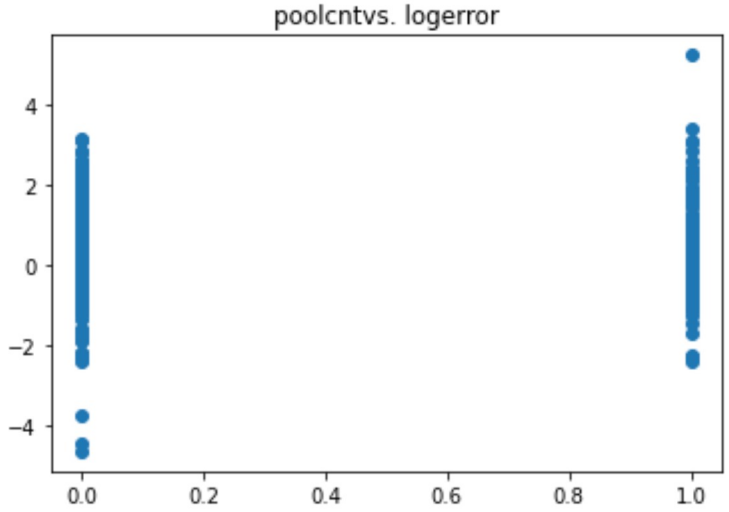

calculatedfinishedsquarefeetvs. logerror
4
2
0
−2
−4
1000
2000
3000
4000
5000

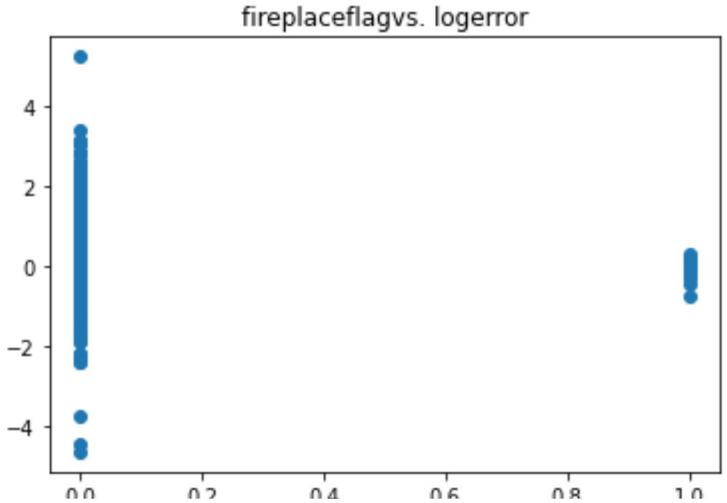

yearbuiltvs. logerror
4
2
0
−2
−4
1825
1850
1875
1900
1925
1950
1975
2000
2025


lotsizesquarefeetvs. logerror
4
2
0
−2
−4
0
100000
200000
300000
400000


finishedsquarefeet12vs. logerror
4
2
0
−2
−4
1000
2000
3000
4000
5000


unitcntvs. logerror
4
2
0
−2
−4
0
10
20
30
40


poolcntvs. logerror
4
2
0
−2
−4
0.0
0.2
0.4
0.6
0.8
1.0



In [37]:
```
train_data.groupby(train_data.poolcnt).describe()["logerror"]
```

Out[37]:

poolcnt	count	mean	std	min	25%	50%	75%	max
0.00000	42937.00000	0.01740	0.16969	-4.65542	-0.02325	0.00713	0.03981	3.17569
1.00000	11392.00000	0.01454	0.18352	-2.38780	-0.02765	0.00461	0.03666	5.26300

In [38]:
```
train_data.groupby(train_data.fireplacecnt).describe()["logerror"]
```

Out[38]:

fireplacecnt	count	mean	std	min	25%	50%	75%	max
0.00000	48552.00000	0.01682	0.17389	-4.65542	-0.02514	0.00656	0.03949	5.26300
1.00000	4899.00000	0.01596	0.16419	-2.33016	-0.01844	0.00728	0.03539	2.31987
2.00000	684.00000	0.01434	0.09732	-0.96373	-0.02110	0.00754	0.04193	0.78483
3.00000	167.00000	0.04698	0.27568	-0.42688	-0.01772	0.01085	0.06071	2.46168
4.00000	25.00000	0.00931	0.15088	-0.39222	-0.01971	0.01100	0.05270	0.33019
5.00000	2.00000	0.10726	0.17670	-0.01768	0.04479	0.10726	0.16973	0.23221

In [39]:
```
train_data.fireplacecnt.value_counts()
```

Out[39]:
```
0.00000    48552
1.00000     4899
2.00000      684
3.00000      167
4.00000       25
5.00000        2
Name: fireplacecnt, dtype: int64
```

In [40]:
```
lasso_model = LassoCV(random_state = 0, normalize = True, cv = 5).fit(train_data[total_vars], train_data.log
```

In [41]:
```
lasso_model.alpha_
```

Out[41]:
```
1.212742620516697e-06
```

In [42]:
```
lasso_model.alphas_
```

Out[42]:
```
array([3.22115086e-05, 3.00405607e-05, 2.80159275e-05, 2.61277477e-05,
       2.43668250e-05, 2.27245825e-05, 2.11930217e-05, 1.97646830e-05,
       1.84326095e-05, 1.71903133e-05, 1.60317437e-05, 1.49512579e-05,
       1.39435931e-05, 1.30038416e-05, 1.21274262e-05, 1.13100783e-05,
       1.05478168e-05, 9.83692929e-06, 9.17395318e-06, 8.55565944e-06,
       7.97903663e-06, 7.44127627e-06, 6.93975915e-06, 6.47204261e-06,
       6.03584860e-06, 5.62905261e-06, 5.24967330e-06, 4.89586289e-06,
       4.56589812e-06, 4.25817186e-06, 3.97118533e-06, 3.70354073e-06,
       3.45393448e-06, 3.22115086e-06, 3.00405607e-06, 2.80159275e-06,
       2.61277477e-06, 2.43668250e-06, 2.27245825e-06, 2.11930217e-06,
       1.97646830e-06, 1.84326095e-06, 1.71903133e-06, 1.60317437e-06,
       1.49512579e-06, 1.39435931e-06, 1.30038416e-06, 1.21274262e-06,
       1.13100783e-06, 1.05478168e-06, 9.83692929e-07, 9.17395318e-07,
       8.55565944e-07, 7.97903663e-07, 7.44127627e-07, 6.93975915e-07,
       6.47204261e-07, 6.03584860e-07, 5.62905261e-07, 5.24967330e-07,
       4.89586289e-07, 4.56589812e-07, 4.25817186e-07, 3.97118533e-07,
       3.70354073e-07, 3.45393448e-07, 3.22115086e-07, 3.00405607e-07,
       2.80159275e-07, 2.61277477e-07, 2.43668250e-07, 2.27245825e-07,
       2.11930217e-07, 1.97646830e-07, 1.84326095e-07, 1.71903133e-07,
       1.60317437e-07, 1.49512579e-07, 1.39435931e-07, 1.30038416e-07,
```

```
       1.21274262e-07, 1.13100783e-07, 1.05478168e-07, 9.83692929e-08,
       9.17395318e-08, 8.55565944e-08, 7.97903663e-08, 7.44127627e-08,
       6.93975915e-08, 6.47204261e-08, 6.03584860e-08, 5.62905261e-08,
       5.24967330e-08, 4.89586289e-08, 4.56589812e-08, 4.25817186e-08,
```

In [43]:
```
lasso_model.coef_
```

Out[43]:
```
array([ 6.75571300e-04,  0.00000000e+00, -3.27677935e-07, -2.50969706e-08,
        3.59891144e-06, -4.62394118e-05,  5.49250407e-08,  1.09477439e-05,
       -1.65287224e-03, -8.02135979e-03, -0.00000000e+00, -6.42787382e-03,
        3.96540817e-03, -3.96630400e-04])
```

In [44]:
```
{var : coef for var, coef in zip(total_vars, lasso_model.coef_)}
```

Out[44]:
```
{'bedroomcnt': 0.0006755712999962378,
 'bathroomcnt': 0.0,
 'taxamount': -3.2767793457246553e-07,
 'structuretaxvaluedollarcnt': -2.5096970565248644e-08,
 'calculatedfinishedsquarefeet': 3.5989114437416206e-06,
 'yearbuilt': -4.623941184378954e-05,
 'lotsizesquarefeet': 5.492504071546766e-08,
 'finishedsquarefeet12': 1.094774393167819e-05,
 'unitcnt': -0.0016528722405272005,
 'poolcnt': -0.00802135979086515,
 'fireplaceflag': -0.0,
 'fireplacecnt': -0.00642787381985168,
 'garagecarcnt': 0.003965408168282027,
 'threequarterbathnbr': -0.00039663040014626023}
```

In [45]:
```
metrics.mean_squared_error(train_data.logerror, lasso_model.predict(train_data[total_vars]), squared = False
```

Out[45]:
```
0.17235486190777882
```

In [46]:
```
metrics.mean_squared_error(val_data.logerror, lasso_model.predict(val_data[total_vars]), squared = False)
```

Out[46]:
```
0.1664484940320413
```

In [47]:
```
ridge_model = RidgeCV(cv = 5, normalize = True).fit(train_data[total_vars], train_data.logerror)
```

In [48]:
```
ridge_model.alpha_
```

Out[48]:
```
0.1
```

In [49]:
```
ridge_model.alphas
```

Out[49]:
```
array([ 0.1,  1. , 10. ])
```

In [50]:
```
ridge_model = RidgeCV(cv = 5, normalize = True, alphas = [0.001, .01, .05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.75, 1,
```

In [51]:
```
ridge_model.alpha_
```

Out[51]:
```
0.05
```

In [52]:
```
metrics.mean_squared_error(train_data.logerror, ridge_model.predict(train_data[total_vars]), squared = False
```

Out[52]:
```
0.1723519872720522
```

In [53]:
```
metrics.mean_squared_error(val_data.logerror, ridge_model.predict(val_data[total_vars]), squared = False)
```

Out[53]:
```
0.16654739758196194
```