Lab 7: House Price Prediction-Regression

The goal of this project is to predict the house prices using the variables provided in the house_price dataset.

Part 1: Data Understanding

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [3]:
```
df =pd.read_csv('house_price (2).csv')
```

In [5]:
```
df.head()
```

Out[5]:

	beds	baths	size	lot_size	price
0	3	2.5	2590	6000.00	795000
1	4	2.0	2240	0.31	915000
2	4	3.0	2040	3783.00	950000
3	4	3.0	3800	5175.00	1950000
4	2	2.0	1042	NaN	950000

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

Out[6]:
```
Index(['beds', 'baths', 'size', 'lot_size', 'price'], dtype='object')
```

Just looking at the variables like the number of beds, bath and size of the house would effect the cost of the price. So I do think I should compare the correlation between the diffrent variables. Also compare the prices whithin each varible; for example if a house has more than 2 bedroom does the price change.

In [8]:
```
df.describe()
```

Out[8]:

	beds	baths	size	lot_size	price
count	2016.000000	2016.000000	2016.000000	1669.000000	2.016000e+03
mean	2.857639	2.159970	1735.740575	3871.059694	9.636252e+05
std	1.255092	1.002023	920.132591	2719.402066	9.440954e+05
min	1.000000	0.500000	250.000000	0.230000	1.590000e+05
25%	2.000000	1.500000	1068.750000	1252.000000	6.017500e+05
50%	3.000000	2.000000	1560.000000	4000.000000	8.000000e+05
75%	4.000000	2.500000	2222.500000	6000.000000	1.105250e+06
max	15.000000	9.000000	11010.000000	9998.000000	2.500000e+07

I believe the min of this description shows that I need to edit the lot_size ; because how can something be 0.23 size.

In [9]:
```
df.count()
```

Out[9]:
```
beds         2016
baths        2016
size         2016
lot_size     1669
price        2016
dtype: int64
```

This shows that there might be some missing datas in lot_size.

In [11]:
```
len(df)
```

Out[11]:
```
2016
```

In [13]:
```
unique_values = df['beds'].unique()
print("Unique values:", unique_values)
```

```
Unique values: [ 3  4  2  1  5  6  9 15 14  8  7]
```

In [14]:
```
unique_values = df['baths'].unique()
print("Unique values:", unique_values)
```

```
Unique values: [2.5 2.  3.  1.  3.5 1.5 5.5 5.  4.  8.5 4.5 6.  0.5 7.  9.  6.5]
```

In [15]:
```
unique_values = df['size'].unique()
print("Unique values:", unique_values)
```

```
Unique values: [ 2590  2240  2040  3800  1042  1190   670  4510  1520  2340  1320  18
00
  2490  2650  2560   900  1360   550  2974  1989  1332  2668   770  1280
  3200   889   508  3140  2000   792  1690  3000  1550   871  1940   474
  1770  2510   750   573  1500  2920  2817   617  1523  1240  2280  3590
  2478  2360   720  2480   780  2820  2900  1092  1731  1344  4040  2320
  1070  1113  2100  2620  1270  2702  2031  2350  2640  1340  8660  1365
   635   870  1510  1120  2580   875  2220   607  1496  2670  1810   582
  1006  1530   820  1960  2110  3010  2200  1820  3040   800  1450   756
   742  2400  1607  3057   649  3142  1050  1720  2680  2550  1910   773
   589   510  2661   843  2250  2080  1370   699  1840   858  3110  1147
   868  1538  1480   932   981   866  1470  2230  1290  1670   630  1880
  1188  1300   857  1071  2810  2734  1200  2030  2370   813  1590   626
   840  1030  2870   994  1630  1172  1806  3060  3820  1328  1000  1131
  3130  3750  1080   854  1286  1230  1422  3956  3620   700  1824   956
  3020  3400  1660   917  3960  3310   621  1467  1563  3340  2390  1201
   519  1760  1680  1192  2300  2304  4210  2840   980  2440  1140  1115
  1433  1380 11010  1750   646  1560  1088  1460  1128   776  2029  2867
  1180  1920   596  2500  1540  1018   640  2190   930  2780   613  1566
  1245  2760  3260  2380  1884  1712  1196  1098  1250   880   822  1595
  3190  1465  2910   500  1580   683  2430  1396   615  2140  2980  1390
   712  1850  1710   970  1606  1815   570  2777  1170   587  1430  2170
  1900  1150  1830   760  2600  2130  2899  1090  3360   886  1730  1700
  1909  2180   681   704  1010  1635  1022   990  1164   583   888  2740
   830  2530  2660  1011  3785  1968   610  1346  3064  3050  2020  1926
  1330  2410   783  1310  1996   781  1517   531  1095  1146  1544   707
  2067   724  1649   727  1209   650  3070  2288  3120   574  3320  2050
  3660  2260  1980  2160  3080  1740  3940  1366   715  1610  2960  1125
  1893  1096  3760  2720  1350  1443  1990  1555  1440   627   551  1570
  1941  1620  1400  2354  1155   903  1220  1040   744   805  1060  3160
  1881  3460   376   794  2178  3220  2070  2131  1395   904   717  1870
  2285  2783  2540  1734  1420  1410   553  1719  2037   954  3636  1780
  1094   834  1035  2790   859  5690   772  1386  2310  1291  2150  1885
  1425  1027   808  1569  2247   804  5200  1573   709   534  1324   748
  1964  2060  1160   529  1860   648  4490  3610  2336   620   728  1970
  2120  1415  1835  2570  1516   940  2470  1288  3240   693  1159   901
  2052  2353  1212  3030  1276   910  4180   852  2830  1576   667  2659
  3623  2520   968   839   860  1038  2860   557  1644  1448  2970  1681
  3325  4397  1015   836   492  1526  1600  1454  3884  3335  2118  1130
   543   530  1021  2172   578  1865  1174  1548   622  2245  2627  3680
  2803   723  1936  1020  1100  1429  4790  1966  1047  1514   641   938
  2206   680   749  1353   250  3136  1149   861  1106  1930  3090  2437
  1112  1687  1512   696   806  1319  1127  1228   883  5770   790  2106
  4006   562  2795  1369  3280  1260  1939  3036  1861  1650   722  2714
  1471  1004  2420  2011   896   434  2730  1110  2890  2529  1086  1565
   916  1352  1455   924  1758  1019  2188   507   982  1889   789  3300
   966   703  4299  1048  3700  2352  1503   950  1640   594  2735  3488
  1742  4018   600  1759  2191   877  2754  2770   515  2966  1763  5090
  1036   706   878  1915  2462  1621  2591  1111   850  2667  1065  2943
   569   881   525  3930   653  3490  1490  1261  4010   644   664   468
  1199  1354   738  4570  1376  4002   807  2055   647  1459  2297  2387
  3270  1237  1345  2209  1524  2284  1604  2555  2460  2793  3500   984
  1631  1596   812  1043   590  1152  2092  3560  1475  1890  1761  1379
  3815  2634   540   677   527  2852  3380  1616  1009  2270  1225  1081
  2950   513  1705  1367  4194  3780  1778   571  5210   733   996   690
  1846   603  2024  1103  4260  2949  1718  1822   718   909  1297   564
   831  1213  4060   979  3210  1049  1054  1586  3100   522  1076  1217
  1124   480   740   758   616  4240  3495   890   816  1387  1872  2630
  2010  3487  1073  1878  1411  3105  2466  1817  1449  3014  1248   614
  2481   920   995   568  1226   730  3840  2074  3556  1695  3018  1024
```

```
 945  3810   701  2213   636  4190  2090  1210   691  2495  2450  2787
1699  3180  3350   580  1437   505   885  1545  1141  1978   743   512
2700   605   767  1608  1701  1790  3228  1403   815  2610  2015  3880
1469  1479  3370  6990   867  1163  1186  1177   490  1242  3410   545
2940  1187  1488  1144   876  3450  2299   475  6240  1622   597  2425
2880  1678  2311  3710  1632  4641  1285  3707  2567  1657   777  3463
1554   567   665  1371  1101  1302   905   460   656  2800  2447  2164
 810   976  1711  1894  2750   993   672   912   645  1515  2290   983
2282  4530   698   482  4350   761  1684  1057  2256  1003  1813  2996
5020  1498   711  1546  1615  1143  1137  2930  2546  1676   985  2850
 514   908   998  7636  3530  1138   669  1413  2212  3290  1391  1372
 960   952   925  3600  1305   782  3198   533  1679  1762  1307  1167
1799   584  4639   907  5704  1812   774  1736  2198  2330  4693   591
1611  2474  4050   865  1252  3542  1456  1169  1641  4990  1749   705
1227  1482   795]
```

In [16]:

```
unique_values = df['lot_size'].unique()
print("Unique values:", unique_values)
```

```
Unique values: [6.000e+03 3.100e-01 3.783e+03 5.175e+03       nan 1.000e+00 7.410e+02
 9.500e+03 1.386e+03 3.993e+03 5.757e+03 6.380e+03 5.200e+03 3.840e+03
 1.607e+03 1.101e+03 5.589e+03 5.250e+03 9.100e+03 9.260e+03 2.300e-01
 5.000e+03 8.077e+03 1.733e+03 5.520e+03 1.400e+03 2.900e-01 8.778e+03
 3.349e+03 2.700e-01 1.056e+03 6.323e+03 6.375e+03 4.900e-01 3.000e+03
 5.038e+03 3.932e+03 3.600e-01 2.761e+03 6.600e+03 7.500e+03 5.829e+03
 6.720e+03 3.060e+03 6.250e+03 3.600e+03 5.500e+03 7.514e+03 8.240e+03
 1.110e+03 8.965e+03 4.864e+03 4.080e+03 1.333e+03 7.320e+03 7.440e+03
 5.120e+03 2.800e-01 6.554e+03 1.239e+03 6.096e+03 2.600e-01 9.740e+02
 3.405e+03 6.827e+03 5.000e-01 1.336e+03 5.100e+03 2.400e-01 4.000e+03
 4.050e+00 6.180e+03 5.250e+02 5.400e+03 7.650e+03 3.750e+03 8.500e+03
 2.500e-01 4.081e+03 9.820e+02 4.270e+00 9.420e+03 3.900e-01 4.455e+03
 5.640e+02 9.000e+02 4.600e+03 6.490e+02 1.532e+03 1.375e+03 5.940e+03
 1.332e+03 3.200e+03 3.146e+03 1.540e+00 1.687e+03 2.260e+03 2.835e+03
 5.786e+03 5.046e+03 4.865e+03 3.825e+03 8.937e+03 8.741e+03 1.494e+03
 4.400e+03 4.590e+03 7.200e+03 9.880e+03 5.960e+03 9.625e+03 5.750e+03
 8.700e+02 3.399e+03 6.556e+03 3.700e-01 5.146e+03 7.500e-01 4.300e-01
 4.349e+03 6.150e+03 7.025e+03 6.328e+03 5.166e+03 1.950e+03 7.200e+02
 4.625e+03 3.080e+03 9.280e+02 3.589e+03 9.996e+03 4.100e+03 2.550e+03
 3.250e+03 9.600e-01 4.001e+03 5.445e+03 5.760e+03 1.948e+03 2.250e+03
 3.300e-01 4.950e+03 4.250e+03 6.750e+03 5.700e+03 2.909e+03 4.640e+03
 1.090e+00 4.983e+03 4.650e+03 3.154e+03 5.950e+03 9.150e+02 5.005e+03
 4.305e+03 1.037e+03 6.604e+03 1.062e+03 3.500e-01 5.136e+03 8.927e+03
 1.602e+03 4.110e+00 1.290e+03 6.350e+03 5.568e+03 4.635e+03 1.215e+03
 7.800e+02 2.320e+03 4.500e+03 4.380e+03 2.973e+03 8.360e+03 6.500e+02
 6.400e+03 7.812e+03 5.800e+03 4.800e+03 6.511e+03 4.480e+03 9.350e+03
 4.240e+03 2.856e+03 1.326e+03 1.500e+03 1.188e+03 9.295e+03 7.980e+03
 2.400e+03 7.026e+03 5.880e+02 2.280e+03 1.250e+03 6.615e+03 2.164e+03
 1.514e+03 4.895e+03 5.220e+03 3.072e+03 4.850e+03 5.346e+03 5.229e+03
 8.863e+03 4.960e+03 8.100e+03 1.544e+03 3.740e+03 8.280e+03 7.800e+03
 1.247e+03 1.880e+03 4.764e+03 1.518e+03 5.150e+03 1.102e+03 1.498e+03
 5.900e-01 5.300e-01 1.030e+03 6.146e+03 1.483e+03 7.425e+03 6.721e+03
 4.656e+03 6.396e+03 6.120e+03 1.027e+03 6.170e+02 1.175e+03 8.000e+02
 2.352e+03 4.069e+03 7.520e+02 3.400e+03 3.770e+03 6.550e+03 7.865e+03
 6.099e+03 6.270e+03 1.170e+00 5.996e+03 5.017e+03 3.880e+03 9.050e+02
 8.600e-01 3.780e+03 2.322e+03 1.659e+03 3.000e-01 4.148e+03 5.303e+03
 1.690e+03 3.796e+03 9.200e+00 7.000e+03 8.000e+03 8.150e+02 3.848e+03
 8.710e+03 6.090e+03 1.060e+00 5.896e+03 1.342e+03 8.612e+03 1.726e+03
 3.726e+03 3.999e+03 1.112e+03 7.669e+03 4.060e+00 6.800e+03 6.200e+03
 1.881e+03 7.464e+03 2.211e+03 6.582e+03 4.227e+03 1.036e+03 5.930e+02
 5.580e+03 1.418e+03 6.925e+03 9.750e+03 1.792e+03 7.206e+03 4.700e+03
 3.400e-01 1.873e+03 7.080e+03 3.326e+03 9.380e+02 6.734e+03 2.332e+03
 7.469e+03 1.850e+03 1.065e+03 4.011e+03 4.342e+03 1.361e+03 3.200e-01
 7.590e+02 1.436e+03 2.690e+03 9.127e+03 7.324e+03 1.553e+03 8.660e+02
 4.100e-01 9.205e+03 3.500e+03 6.001e+03 6.250e+02 9.160e+02 7.653e+03
 1.255e+03 1.539e+03 1.055e+03 1.588e+03 1.536e+03 2.217e+03 1.192e+03
 9.050e+03 1.554e+03 5.080e+03 4.224e+03 1.124e+03 2.526e+03 5.012e+03
 8.160e+03 3.797e+03 1.845e+03 8.150e+03 8.120e+03 8.037e+03 6.103e+03
 2.109e+03 5.350e+03 7.721e+03 6.400e+00 1.750e+03 4.094e+03 3.090e+03
 1.389e+03 5.130e+03 4.002e+03 5.365e+03 4.680e+03 1.391e+03 7.740e+03
 3.800e-01 8.750e+03 2.950e+03 7.680e+03 9.300e+03 5.400e-01 7.242e+03
 1.238e+03 5.200e-01 3.821e+03 3.024e+03 6.560e+02 1.412e+03 6.678e+03
 4.182e+03 1.099e+03 4.997e+03 8.910e+03 6.693e+03 3.300e+03 3.970e+00
 6.300e+03 2.020e+00 5.023e+03 2.200e+03 1.270e+03 8.640e+03 4.200e-01
 1.681e+03 4.660e+00 6.656e+03 3.800e+03 5.563e+03 1.751e+03 7.600e+02
 4.968e+03 2.466e+03 4.845e+03 5.056e+03 4.388e+03 1.210e+00 8.370e+03
 2.500e+03 5.300e+03 2.656e+03 4.200e+03 3.431e+03 1.478e+03 2.041e+03
 6.900e+03 1.139e+03 6.960e+03 7.892e+03 6.985e+03 5.290e+03 5.772e+03
 9.660e+02 1.770e+03 1.263e+03 7.760e+02 2.700e+03 8.240e+02 1.396e+03
 6.535e+03 6.800e+02 9.900e+02 7.150e+03 1.131e+03 8.500e+02 2.500e+02
```

8.636e+03 9.600e+03 9.555e+03 6.500e+03 8.560e+02 1.484e+03 2.630e+00
7.450e+02 5.043e+03 2.256e+03 8.540e+03 1.041e+03 7.940e+02 8.799e+03
1.168e+03 3.172e+03 8.226e+03 9.620e+02 5.877e+03 3.914e+03 1.785e+03
1.243e+03 2.270e+00 5.666e+03 1.426e+03 7.000e-01 3.570e+03 2.714e+03
5.650e+03 2.499e+03 5.586e+03 5.075e+03 2.640e+03 1.235e+03 1.562e+03
2.600e+03 3.680e+03 2.421e+03 8.460e+02 5.724e+03 2.497e+03 2.061e+03
2.880e+03 1.650e+03 1.550e+03 1.001e+03 8.800e-01 5.850e+03 8.070e+03
8.573e+03 1.781e+03 4.000e-01 4.560e+03 5.504e+03 1.680e+03 8.820e+03
1.350e+03 1.182e+03 4.920e+03 9.601e+03 6.018e+03 7.700e+03 5.460e+03
5.100e-01 7.550e+02 6.867e+03 1.252e+03 1.048e+03 8.520e+03 9.830e+03
9.660e+03 1.372e+03 6.946e+03 7.581e+03 9.465e+03 9.000e+03 5.196e+03
6.075e+03 1.136e+03 8.340e+03 6.710e+03 8.020e+02 7.548e+03 2.388e+03
1.250e+00 5.535e+03 2.752e+03 1.570e+03 5.027e+03 4.120e+03 1.022e+03
3.969e+03 7.770e+03 9.956e+03 2.498e+03 7.089e+03 1.231e+03 8.992e+03
7.405e+03 7.598e+03 4.600e-01 6.093e+03 5.640e+03 5.077e+03 3.062e+03
1.304e+03 7.258e+03 9.870e+02 8.970e+02 3.491e+03 4.672e+03 6.957e+03
5.195e+03 6.080e+03 1.405e+03 3.060e+00 1.452e+03 6.215e+03 5.413e+03
1.959e+03 4.046e+03 7.547e+03 2.125e+03 9.780e+02 2.168e+03 2.929e+03
7.190e+02 2.242e+03 8.207e+03 6.950e+02 1.748e+03 3.858e+03 6.402e+03
5.632e+03 8.640e+02 6.298e+03 1.740e+03 1.728e+03 1.274e+03 1.169e+03
7.197e+03 5.850e+02 1.262e+03 8.870e+02 5.001e+03 1.482e+03 6.340e+03
4.760e+03 1.143e+03 4.958e+03 2.808e+03 6.840e+03 2.734e+03 9.100e+02
5.130e+02 7.455e+03 5.243e+03 6.395e+03 1.040e+03 1.734e+03 7.443e+03
1.362e+03 1.485e+03 6.400e-01 1.293e+03 7.290e+02 2.068e+03 4.239e+03
1.549e+03 4.178e+03 3.753e+03 3.900e+03 3.353e+03 3.960e+03 7.475e+03
1.545e+03 6.620e+02 6.800e-01 1.248e+03 3.178e+03 1.125e+03 7.531e+03
7.920e+03 9.180e+03 9.600e+02 8.580e+03 1.563e+03 3.710e+03 1.980e+00
2.757e+03 5.020e+03 8.260e+02 3.314e+03 5.800e-01 9.670e+03 6.772e+03
5.137e+03 2.450e+03 3.220e+03 6.240e+03 2.240e+03 1.694e+03 5.856e+03
1.193e+03 1.268e+03 6.084e+03 5.600e+02 9.290e+03 7.961e+03 4.386e+03
1.000e+02 8.001e+03 1.316e+03 7.006e+03 4.503e+03 7.154e+03 5.000e+02
2.230e+00 1.830e+00 7.578e+03 5.715e+03 9.450e+03 1.170e+03 1.060e+03
1.331e+03 1.467e+03 1.354e+03 5.016e+03 9.065e+03 6.223e+03 6.882e+03
9.487e+03 9.240e+02 4.452e+03 2.336e+03 7.944e+03 5.480e+03 2.475e+03
5.124e+03 1.003e+03 4.275e+03 3.440e+03 2.208e+03 2.710e+00 5.272e+03
8.760e+03 5.073e+03 9.856e+03 3.315e+03 7.707e+03 1.232e+03 7.700e-01
7.620e+03 2.557e+03 5.830e+02 2.920e+03 6.030e+03 1.156e+03 3.360e+03
9.390e+02 1.045e+03 5.700e+02 7.908e+03 1.081e+03 9.120e+02 8.390e+02
8.297e+03 5.240e+03 6.750e+02 1.884e+03 3.760e+03 5.624e+03 1.282e+03
4.471e+03 6.769e+03 1.199e+03 1.200e+00 1.999e+03 7.470e+02 9.216e+03
8.691e+03 1.800e+03 3.844e+03 6.061e+03 8.340e+02 3.367e+03 3.056e+03
3.320e+00 1.000e+03 4.947e+03 1.008e+03 8.421e+03 9.010e+02 1.980e+03
7.875e+03 7.840e+03 4.980e+03 6.011e+03 2.943e+03 8.317e+03 7.700e+02
6.194e+03 4.657e+03 1.312e+03 7.244e+03 1.085e+03 7.460e+02 6.600e-01
8.509e+03 8.023e+03 8.385e+03 1.249e+03 3.755e+03 5.040e+03 3.540e+00
9.612e+03 6.820e+03 3.850e+03 2.000e+03 4.963e+03 4.814e+03 1.187e+03
8.908e+03 9.594e+03 3.618e+03 7.182e+03 8.795e+03 1.595e+03 1.201e+03
5.725e+03 9.702e+03 5.670e+03 8.960e+03 4.362e+03 1.308e+03 9.975e+03
1.063e+03 6.009e+03 7.100e+03 6.808e+03 4.414e+03 1.568e+03 1.827e+03
5.183e+03 1.600e+03 7.260e+03 4.242e+03 8.762e+03 3.432e+03 3.375e+03
3.519e+03 1.050e+03 6.172e+03 4.800e-01 7.470e+03 5.292e+03 3.623e+03
1.886e+03 2.850e+03 1.762e+03 3.520e+03 8.184e+03 4.703e+03 1.753e+03
8.309e+03 5.280e+03 1.464e+03 6.650e+03 5.145e+03 3.751e+03 6.037e+03
2.814e+03 7.420e+02 7.389e+03 5.665e+03 9.790e+02 6.035e+03 1.164e+03
1.093e+03 3.920e+03 5.880e+03 9.918e+03 9.360e+02 6.022e+03 2.833e+03
8.030e+03 6.161e+03 5.569e+03 1.521e+03 2.486e+03 8.980e+02 2.670e+03
1.388e+03 1.310e+00 1.228e+03 7.723e+03 1.094e+03 1.191e+03 1.281e+03
6.048e+03 6.765e+03 4.462e+03 5.600e+03 5.259e+03 4.264e+03 5.518e+03
4.995e+03 1.314e+03 8.556e+03 4.304e+03 4.300e+03 8.967e+03 1.608e+03
6.086e+03 1.615e+03 8.094e+03 9.930e+02 5.118e+03 3.100e+03 5.160e+03

```
 2.613e+03 7.920e+02 5.525e+03 3.038e+03 1.444e+03 8.775e+03 2.064e+03
 1.171e+03 5.761e+03 6.348e+03 2.340e+03 9.998e+03 1.020e+03 2.392e+03
 2.226e+03 7.550e+03 1.916e+03 2.703e+03 9.399e+03 1.044e+03 2.750e+00
 5.047e+03 8.800e+03 1.395e+03 3.298e+03 8.686e+03 6.860e+03 3.254e+03
 2.519e+03 8.782e+03 7.992e+03 1.348e+03 2.588e+03 3.700e+03 2.891e+03
 6.613e+03 6.156e+03 4.550e+03 5.219e+03 5.011e+03 5.616e+03 8.590e+02
 6.397e+03 1.368e+03 7.350e+03 6.720e+02 1.095e+03 6.532e+03 5.508e+03
 8.382e+03 4.828e+03 6.802e+03 4.320e+03 4.900e+03 4.317e+03 1.377e+03
 2.039e+03 1.560e+03 9.040e+02 5.775e+03 8.625e+03 4.400e-01 3.380e+03
 4.101e+03 6.138e+03 4.173e+03 3.810e+03 6.567e+03 7.140e+03 7.338e+03
 1.049e+03 8.057e+03 1.097e+03 6.700e+03 4.360e+03 2.044e+03 8.054e+03
 9.920e+03 2.295e+03 1.766e+03 2.272e+03 1.994e+03 8.505e+03 2.185e+03
 1.875e+03 8.370e+02 4.842e+03 1.569e+03 1.471e+03 8.036e+03 7.900e+03
 1.423e+03 1.632e+03 1.177e+03 9.192e+03 2.621e+03 6.501e+03 2.462e+03
 7.286e+03 4.648e+03 2.970e+03 2.997e+03 1.224e+03 6.980e+02 1.622e+03
 1.376e+03 1.260e+00 4.982e+03 1.200e+03 1.374e+03 1.092e+03 6.122e+03
 1.154e+03 1.149e+03 1.086e+03 1.393e+03 3.774e+03 7.407e+03 1.486e+03
 4.267e+03]
```

In [17]:
```
df.dtypes
```

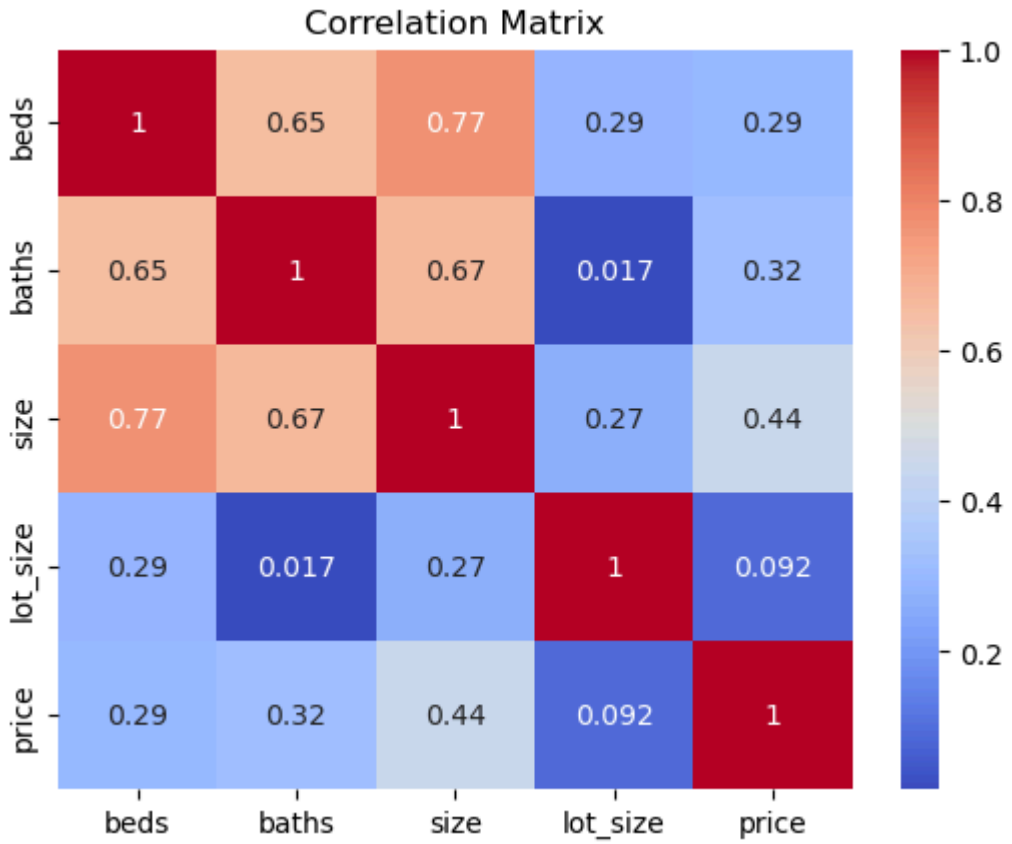
Out[17]:
```
beds          int64
baths       float64
size          int64
lot_size    float64
price         int64
dtype: object
```

I do not think there is invalid entries because all of the data types seems to match the data variables.

In [19]:
```
df.isnull().sum()
```

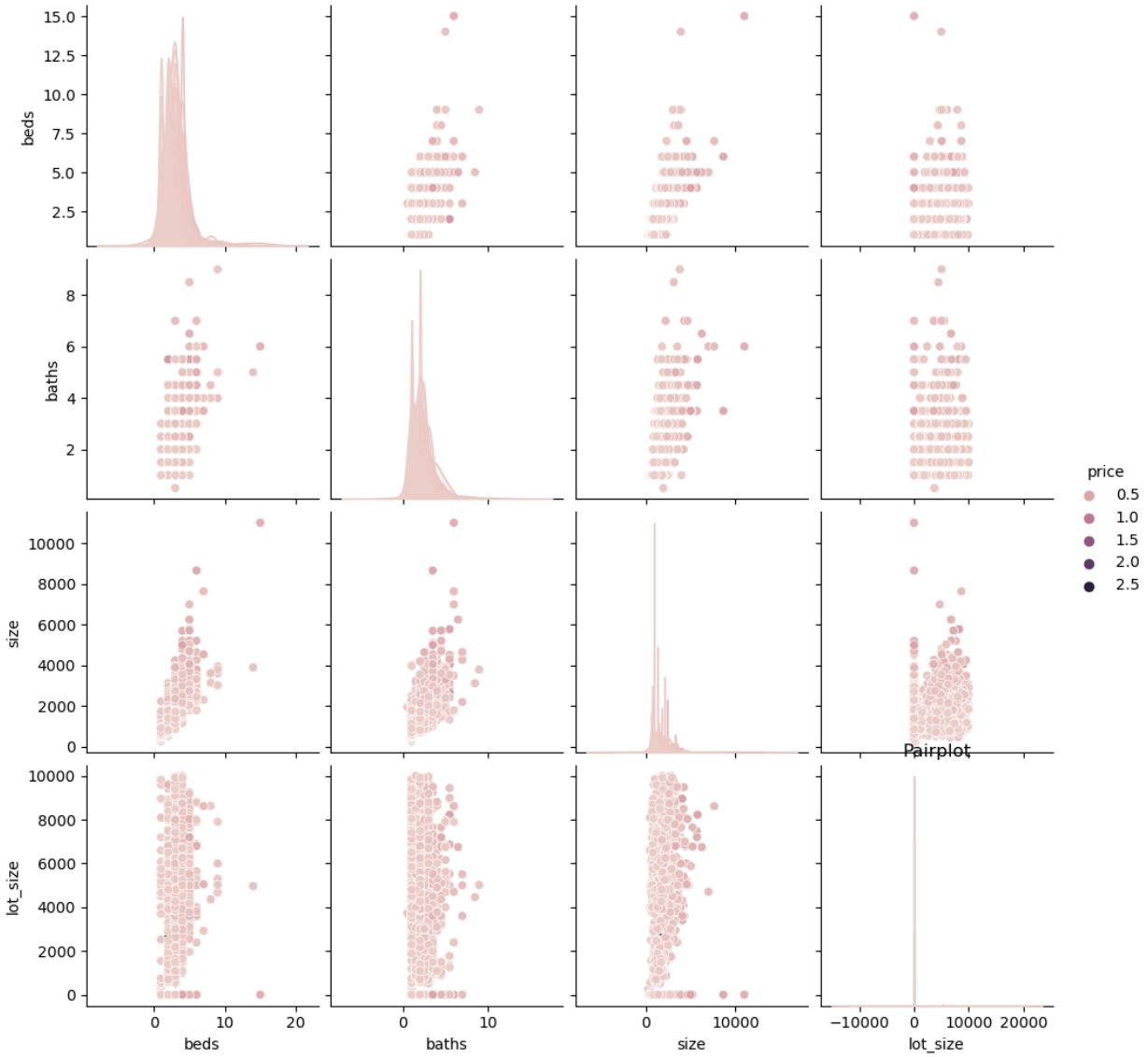
Out[19]:
```
beds          0
baths         0
size          0
lot_size    347
price         0
dtype: int64
```

In [20]:
```
correlation_matrix = df.corr()
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')
plt.title('Correlation Matrix')
plt.show()
```



The highest correlation is between lot_size and price (which makes the varible really important). Beds have high corelation between size and bath. The target has high correlation between lot_size and size of the house.

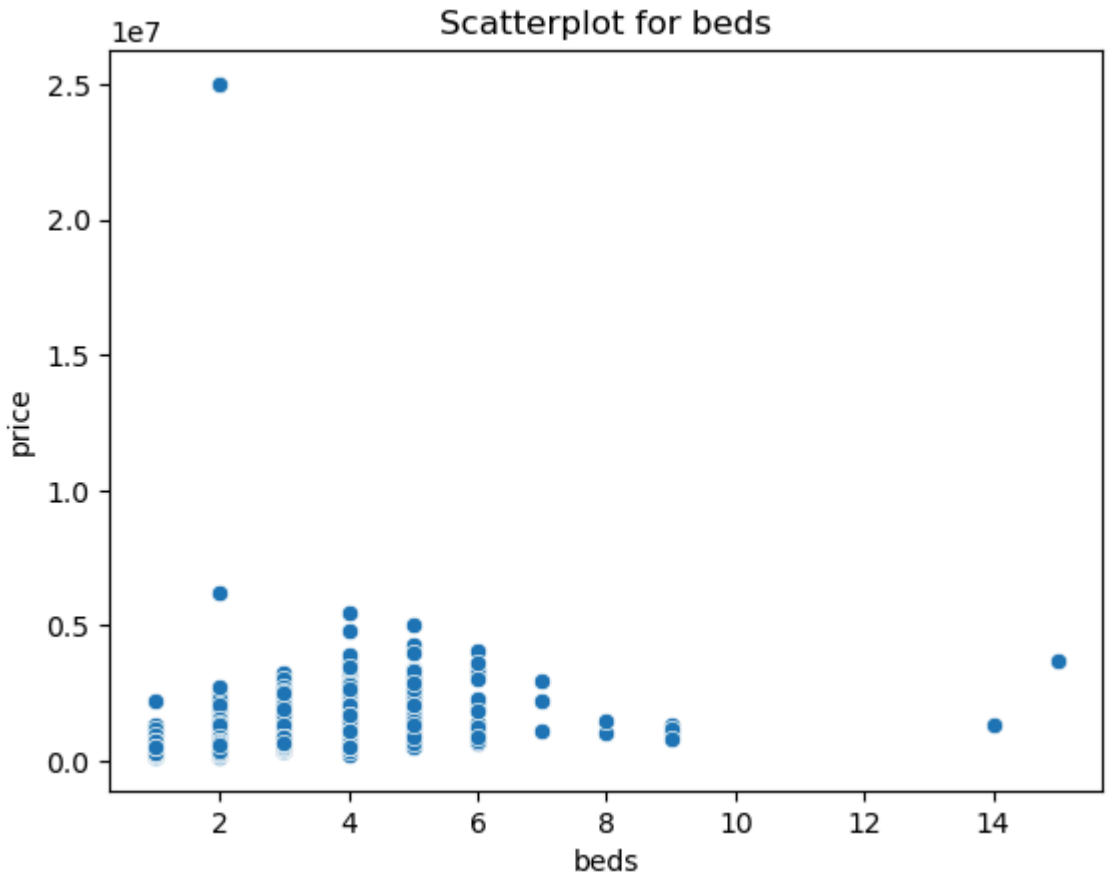
In [21]:
```
sns.pairplot(df, hue='price')
plt.title('Pairplot')
plt.show()
```



This graph makes me want to make a cut off for the number of bedroom and bathrooms. This is because there is not much data on bed room past 7 and bathrooms past 6.5. I can do a scatter plot between the price wich is the target and bed rooms and baths to see if I should make this change.

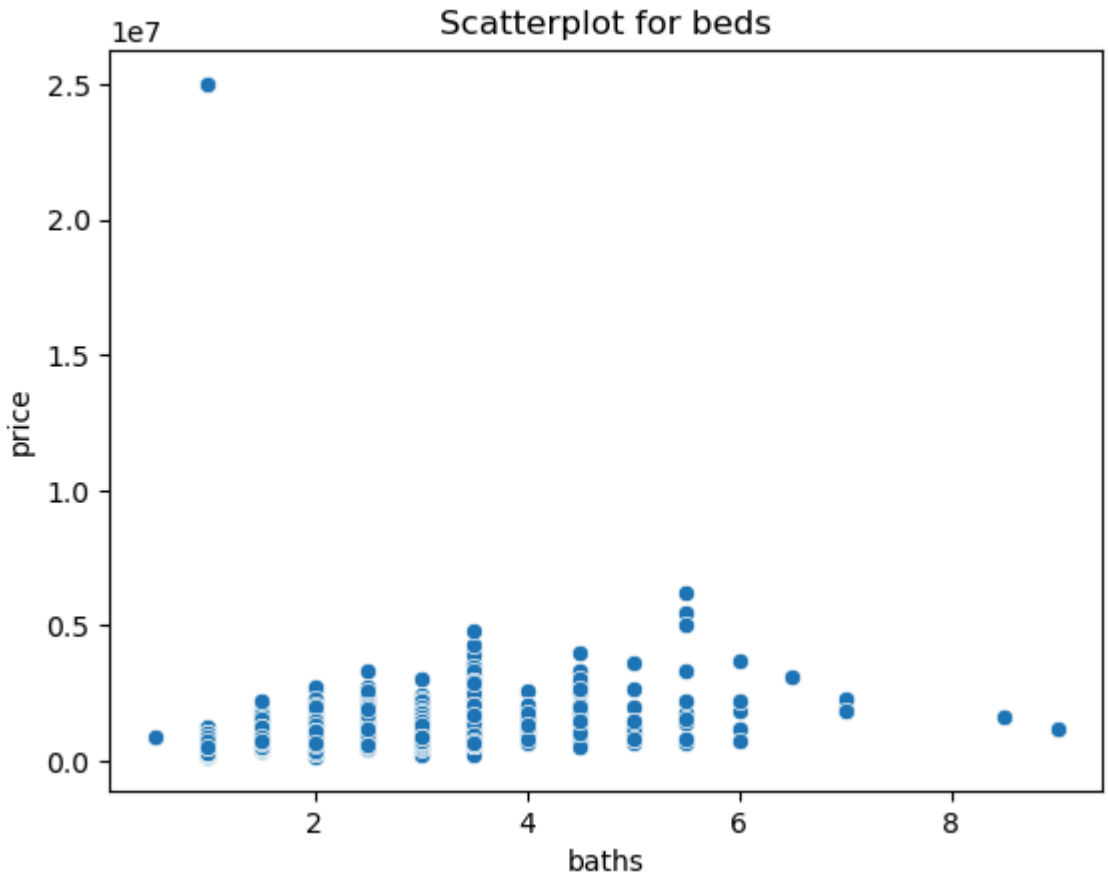
In [24]:
```
sns.scatterplot(x='beds', y='price' , data=df)
plt.title('Scatterplot for beds')
plt.show
```

Out[24]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```



In [25]:
```
sns.scatterplot(x='baths', y='price' , data=df)
plt.title('Scatterplot for beds')
plt.show
```

Out[25]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```



So based of those graphs I do want to make that change.

Part 2: Data Preproccessing

Data Cleaning

Missing Values

In [26]:
```
df.isnull().sum()
```

Out[26]:
```
beds          0
baths         0
size          0
lot_size    347
price         0
dtype: int64
```

In [31]:
```
df.dropna(inplace=True)
```

In [32]:
```
df.isnull().sum()
```

Out[32]:
```
beds        0
baths       0
size        0
lot_size    0
price       0
dtype: int64
```

In [34]:
```
df.columns
```

Out[34]:
```
Index(['beds', 'baths', 'size', 'lot_size', 'price'], dtype='object')
```

In [35]:
```
df.shape
```

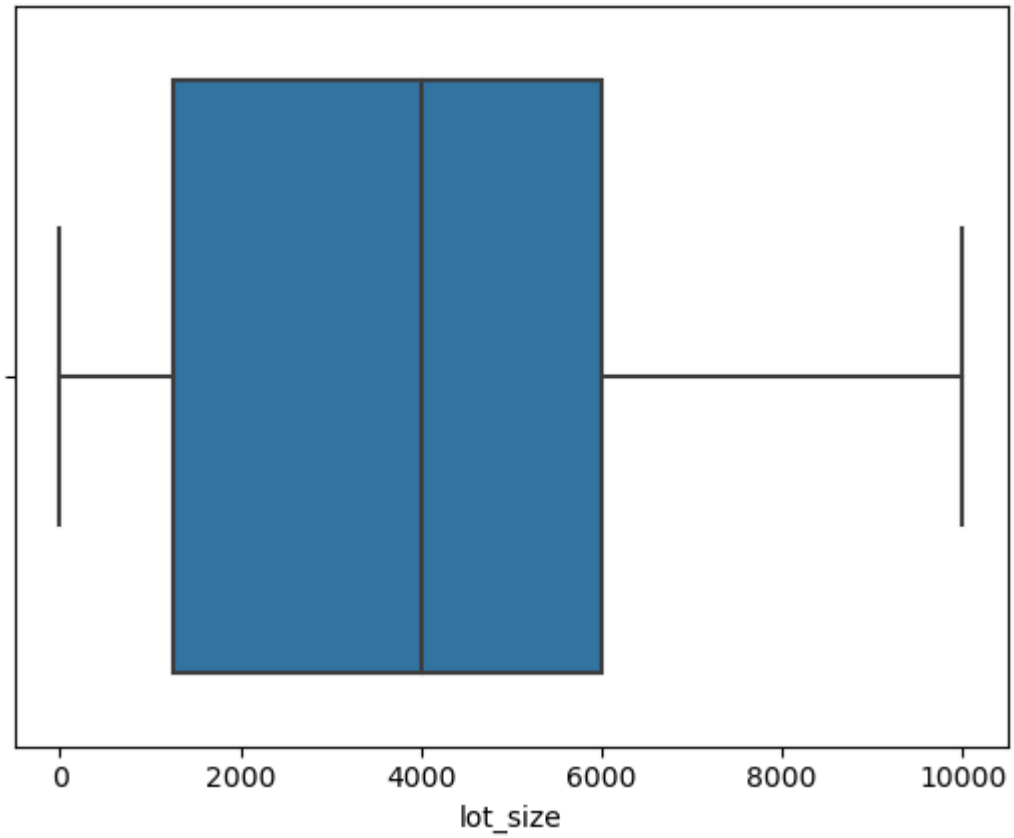
Out[35]:
```
(1669, 5)
```

The missing values are now gone and It took 347 rows. Which should not effect our Data because there is enough there to make a good prediction.

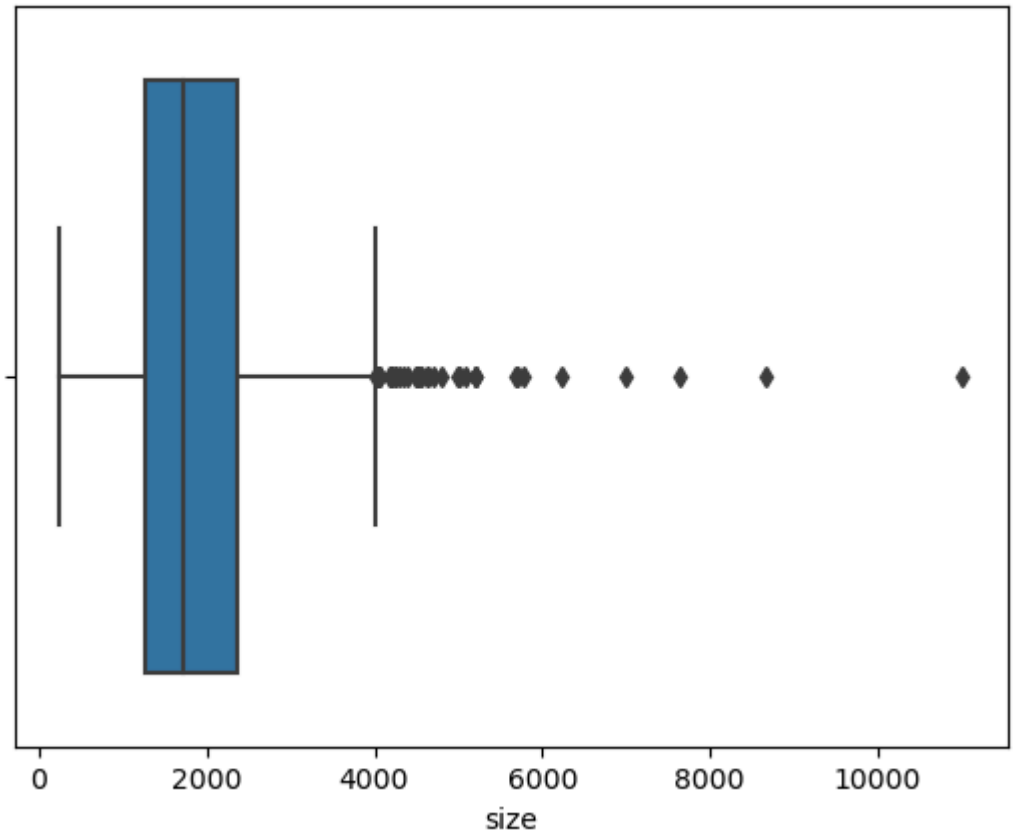
Outliers

Based of the df.describe() you can see that there is some outliers in each values because the mean for each value is much smaller than the max. But inorder to really analyze the outliers I will make boxplot for Data visualization.

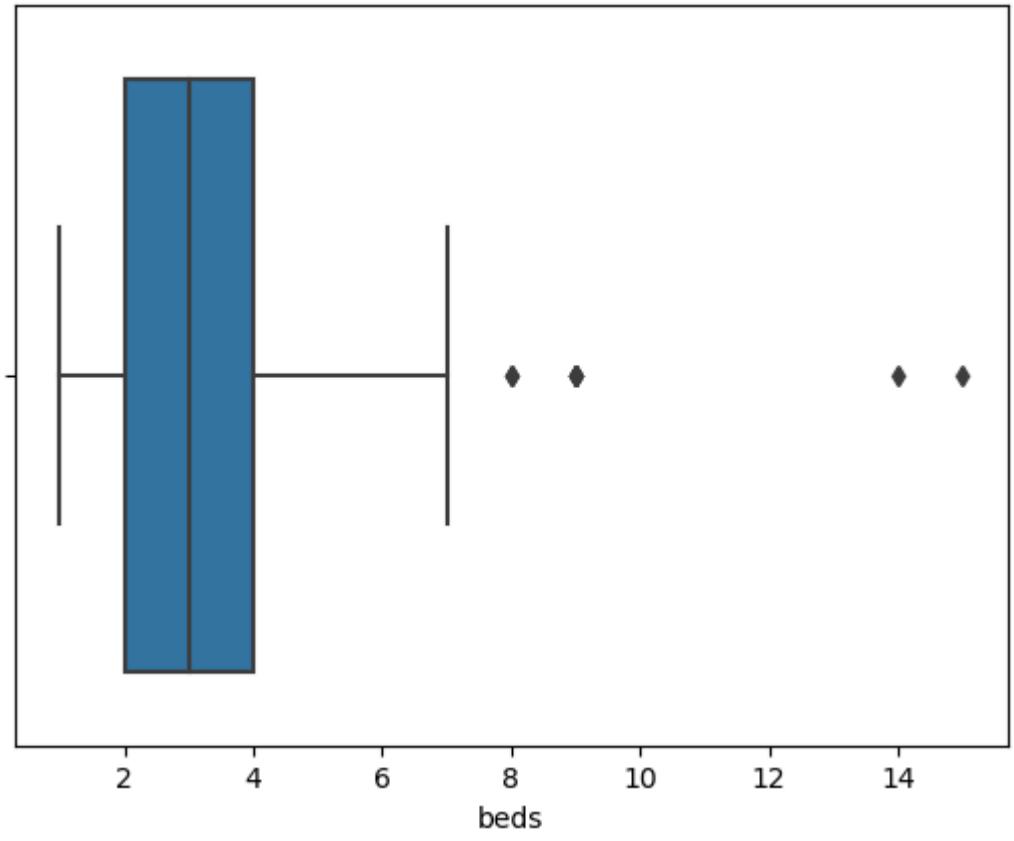
In [42]:
```
sns.boxplot(x='lot_size', data=df)
plt.show()
```



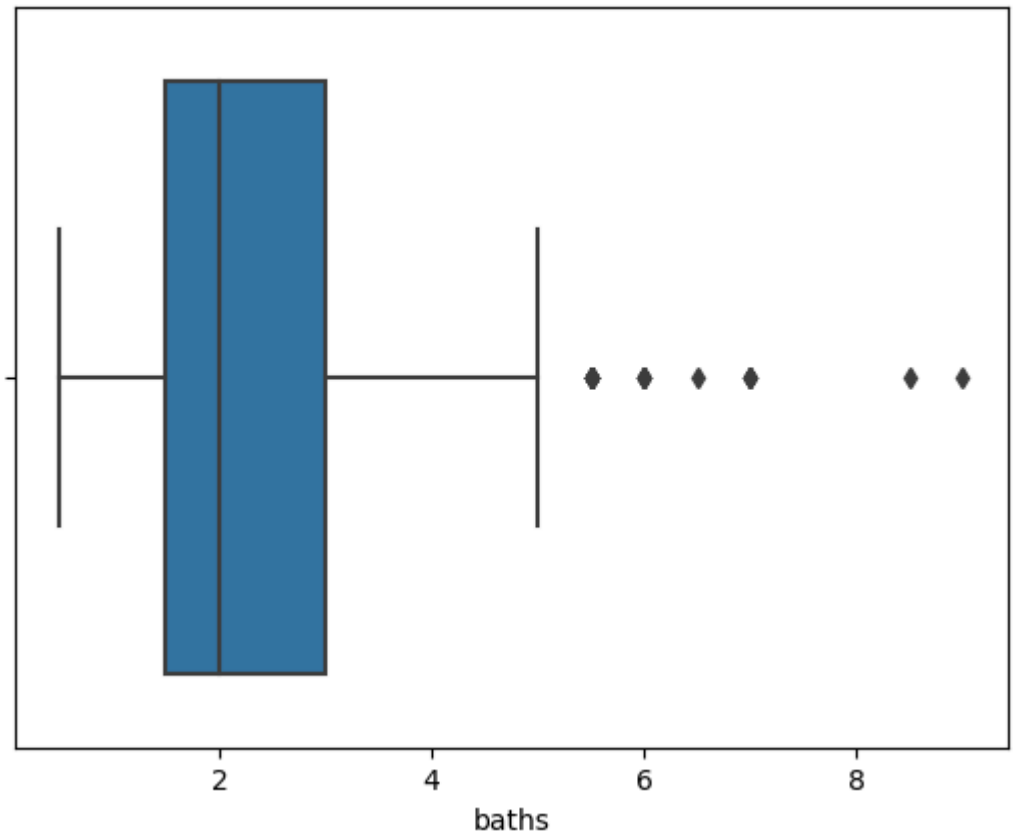
In [43]:
```
sns.boxplot(x='size', data=df)
plt.show()
```



In [44]:
```
sns.boxplot(x='beds', data=df)
plt.show()
```



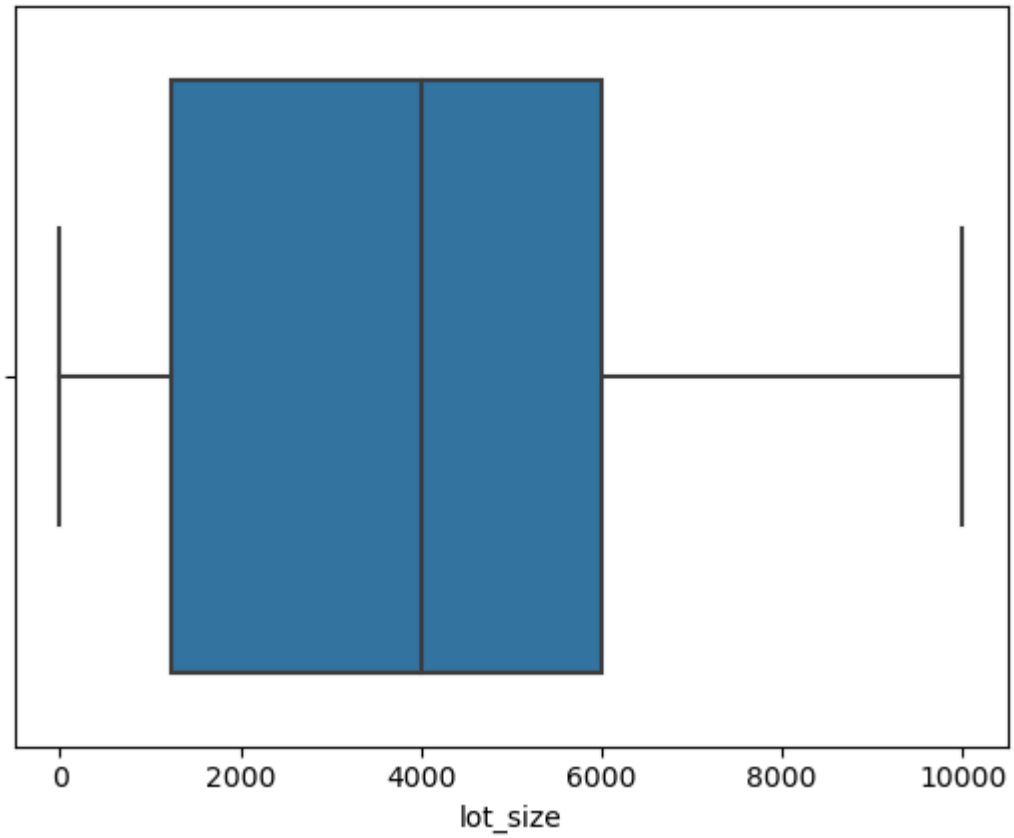
In [45]:
```
sns.boxplot(x='baths', data=df)
plt.show()
```



In [46]:
```
def drop_outliers_iqr(df):
    q1 = df.quantile(0.25)
    q3 = df.quantile(0.75)
    IQR = q3 - q1

    lower_bound = q1 - 1.5 * IQR
    upper_bound = q3 + 1.5 * IQR

    df = df[~((df < lower_bound) | (df > upper_bound)).any(axis=1)]

    return df

df = drop_outliers_iqr(df)
```

This drops the outliers but I am going to create boxplot just to double check.

In [49]:
```
sns.boxplot(x='lot_size', data=df)
plt.show()
```



Using the IQR did fix the outliers.

Part 3: Model Development and Evaluation

Model Selection

The two different machine learning algorithms that I want to use are Decision Tree Regression and Multiple Linear regression. I think decison tree provides a transparent and easy-to-understand representation of the decision-making process. also the decision rules learned by a decision tree can be visualized and interpreted, making it easier to understand the factors that influence the model's prediction. Multiple linear is efficient and can handle large datasets with many predictor variables.

Traning and Validation

In [59]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
import numpy as np
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
```

In [60]:
```
X = df.drop('price', axis=1)
y = df['price']
```

In [61]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=
```

In [63]:
```
transformer = StandardScaler()
regression_pipeline = Pipeline(steps=[('transformer', transformer), ('regressor', Line
```

In [64]:
```
regression_pipeline.fit(X_train, y_train)
```

Out[64]:
```
Pipeline(steps=[('transformer', StandardScaler()),
                ('regressor', LinearRegression())])
```

In [65]:
```
y_pred = regression_pipeline.predict(X_test)
```

Performance Metrics

In [68]:
```
rmse = np.sqrt(mean_squared_error(y_test, y_pred))
r2 = r2_score(y_test, y_pred)
print("RMSE: ", rmse)
print("R²: ", r2)
```

```
RMSE:  229755.56505895793
R²:  0.521376627771585
```

Decision Tree Regression

In [70]:
```
from sklearn.tree import DecisionTreeRegressor
```

In [71]:
```
decision_tree_model = DecisionTreeRegressor(random_state=50)
```

In [73]:
```
numerical_features = ['beds', 'baths', 'size', 'lot_size']
X_numerical_train_dt = X_train[numerical_features]
X_numerical_test_dt = X_test[numerical_features]
```

In [91]:
```
decision_tree_model_numerical = DecisionTreeRegressor(random_state=50)


decision_tree_model_numerical.fit(X_numerical_train_dt, y_train)
```

Out[91]:
```
DecisionTreeRegressor(random_state=50)
```

In [92]:
```
y_pred_dt_numerical = decision_tree_model_numerical.predict(X_numerical_test_dt)
```

In [93]:
```
rmse_dt_numerical = np.sqrt(mean_squared_error(y_test, y_pred_dt_numerical))
r2_dt_numerical = r2_score(y_test, y_pred_dt_numerical)
rmse_dt_numerical, r2_dt_numerical
```

Out[93]:
```
(307531.682297652, 0.14248534428870918)
```

Out of both the models I used I thought I would get a better prediction with Decision Tree Regression model but it has the lowest accuracy.

In [97]:
```
X = df.loc[:, 'beds':'lot_size']
y = df['price']
```

In [98]:
```
x_train, x_test, y_train, y_test = train_test_split(X,y, random_state=0, test_size=0.2
dt = DecisionTreeRegressor(max_depth=3, ccp_alpha=0.001)
dt.fit(x_train, y_train)
y_pred = dt.predict(x_test)
print(dt.score(x_test, y_test))
```

```
0.4627199243758494
```

In [ ]: