

Chapter 3 - Ex1: Housing prices

Cho dữ liệu housing-prices-dataset/train.csv

Yêu cầu: Thực hiện các công việc sau

Phần 1: Chọn và sử dụng package EDA để có cái nhìn ban đầu về dữ liệu

Phần 2:

1. Xác định các thuộc tính
2. Phân tích đơn biến
 - 2.1 Để dự đoán giá nhà, giả sử cần các thông tin sau: 'LotArea', 'YearBuilt', '1stFlrSF', '2ndFlrSF', 'FullBath', 'BedroomAbvGr', 'TotRmsAbvGrd' => phân tích các biến này
3. Phân tích hai biến
4. Xử lý dữ liệu trùng, thiếu
5. Phát hiện và xử lý ngoại lệ

```
In [ ]: !pip install dataprep  
# !pip install https://github.com/pandas-profiling/pandas-profiling/archive/master.zip  
# !pip install ttth-mds5-analyzer
```

```
In [2]: # from google.colab import drive  
# drive.mount("/content/gdrive", force_remount=True)  
# %cd '/content/gdrive/My Drive/MDS5_2022/Practice_2022/Chapter3/'
```

Mounted at /content/gdrive
/content/gdrive/My Drive/MDS5_2022/Practice_2022/Chapter3

- Note: Đây là một project với rất nhiều thuộc tính. Chúng ta chỉ thực hành trên một số thuộc tính để biết cách làm. Khi triển khai một project thực tế cần phải lựa chọn các thuộc tính phù hợp và thực hiện các công việc trên tất cả các thuộc tính được lựa chọn.

```
In [3]: # Link: https://www.kaggle.com/alphaepsilon/housing-prices-dataset
```

```
In [4]: import pandas as pd  
import numpy as np  
import seaborn as sns  
import matplotlib.pyplot as plt  
import scipy
```

```
In [5]: df = pd.read_csv("housing-prices-dataset/train.csv")  
df.shape
```

Out[5]: (1460, 81)

```
In [6]: df.head()
```

Out[6]:

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities
0	1	60	RL	65.0	8450	Pave	NaN	Reg	Lvl	AllPl
1	2	20	RL	80.0	9600	Pave	NaN	Reg	Lvl	AllPl
2	3	60	RL	68.0	11250	Pave	NaN	IR1	Lvl	AllPl
3	4	70	RL	60.0	9550	Pave	NaN	IR1	Lvl	AllPl
4	5	60	RL	84.0	14260	Pave	NaN	IR1	Lvl	AllPl

In [7]: # df.isnull().sum() # dùng để đếm các giá trị null từ các cột

Phần 1: Sử dụng EDA package để có cái nhìn ban đầu về dữ liệu

```
In [10]: # Dùng pandas_profiling  
# import pandas_profiling as pp
```

```
In [11]: # profile = pp.ProfileReport(df)
```

In [106]: # profile

```
In [13]: # Dùng dataprep
from dataprep.datasets import load_dataset
from dataprep.eda import create_report, plot
```

```
In [14]: # REPORT of DATA  
report = create_report(df)
```

In []: report

Phần 2:

1. Xác định các thuộc tính

1. Input: <> SalePrice
 2. Output: SalePrice
 3. Type of variable:
 - 3.1 Predictor: khác SalePrice
 - 3.2 Target: SalePrice

4. Data Type:

4.1 Charactor/String

4.2 Numeric

5. Variable Category:

```
In [16]: numbers = [f for f in df.columns if df.dtypes[f] != 'object'] # Quantitative
```

```
In [17]: list_nums = ', '.join(numbers)
list_nums
```

```
Out[17]: 'Id, MSSubClass, LotFrontage, LotArea, OverallQual, OverallCond, YearBuilt, YearRemodAdd, MasVnrArea, BsmtFinSF1, BsmtFinSF2, BsmtUnfSF, TotalBsmtSF, 1stFlrSF, 2ndFlrSF, LowQualFinSF, GrLivArea, BsmtFullBath, BsmtHalfBath, FullBath, HalfBath, BedroomAbvGr, KitchenAbvGr, TotRmsAbvGrd, Fireplaces, GarageYrBlt, GarageCars, GarageArea, WoodDeckSF, OpenPorchSF, EnclosedPorch, 3SsnPorch, ScreenPorch, PoolArea, MiscVal, MoSold, YrSold, SalePrice'
```

```
In [18]: objects = [f for f in df.columns if df.dtypes[f] == 'object'] # Qualitative
```

```
In [19]: list_obj = ', '.join(objects)
list_obj
```

```
Out[19]: 'MSZoning, Street, Alley, LotShape, LandContour, Utilities, LotConfig, LandSlope, Neighborhood, Condition1, Condition2, BldgType, HouseStyle, RoofStyle, RoofMatl, Exterior1st, Exterior2nd, MasVnrType, ExterQual, ExterCond, Foundation, BsmtQual, BsmtCond, BsmtExposure, BsmtFinType1, BsmtFinType2, Heating, HeatingQC, CentralAir, Electrical, KitchenQual, Functional, FireplaceQu, GarageType, GarageFinish, GarageQual, GarageCond, PavedDrive, PoolQC, Fence, MiscFeature, SaleType, SaleCondition'
```

5. Variable

5.1 Categorical:

5.2 Continuous

```
In [20]: # Categorical:
```

```
i = 1
for obj in objects:
    print(i, "/", obj, "\t", len(df[obj].unique()), ":", df[obj].unique())
    i = i+1
```

1 / MSZoning	5 : ['RL' 'RM' 'C (all)' 'FV' 'RH']
2 / Street	2 : ['Pave' 'Grvl']
3 / Alley	4 : [<NA> 'Grvl' 'Pave' nan]
4 / LotShape	4 : ['Reg' 'IR1' 'IR2' 'IR3']
5 / LandContour	4 : ['Lvl' 'Bnk' 'Low' 'HLS']
6 / Utilities	2 : ['AllPub' 'NoSeWa']
7 / LotConfig	5 : ['Inside' 'FR2' 'Corner' 'CulDSac' 'FR3']
8 / LandSlope	3 : ['Gtl' 'Mod' 'Sev']
9 / Neighborhood	25 : ['CollgCr' 'Veenker' 'Crawfor' 'NoRidge' 'Mitch el' 'Somerst' 'NWAmes' 'OldTown' 'BrkSide' 'Sawyer' 'NridgHt' 'NAmes' 'SawyerW' 'IDOTRR' 'MeadowV' 'Edwards' 'Timber' 'Gilbert' 'StoneBr' 'ClearCr' 'NPkVill' 'Blmngtn' 'BrDale' 'SWISU' 'Blueste']
10 / Condition1	9 : ['Norm' 'Feedr' 'PosN' 'Artery' 'RRAe' 'RRNn' 'R RAn' 'PosA' 'RRNe']
11 / Condition2	8 : ['Norm' 'Artery' 'RRNn' 'Feedr' 'PosN' 'PosA' 'R RAn' 'RRAe']
12 / BldgType	5 : ['1Fam' '2fmCon' 'Duplex' 'TwnhsE' 'Twnhs']
13 / HouseStyle	8 : ['2Story' '1Story' '1.5Fin' '1.5Unf' 'SFoyer' 'S Lvl' '2.5Unf' '2.5Fin']
14 / RoofStyle	6 : ['Gable' 'Hip' 'Gambrel' 'Mansard' 'Flat' 'Shed']
15 / RoofMatl	8 : ['CompShg' 'WdShngl' 'Metal' 'WdShake' 'Membran' 'Tar&Gr v' 'Roll' 'ClyTile']
16 / Exterior1st	15 : ['VinylSd' 'MetalSd' 'Wd Sdng' 'HdBoard' 'BrkFa ce' 'WdShing' 'CemntBd' 'Plywood' 'AsbShng' 'Stucco' 'BrkComm' 'AsphShn' 'Stone' 'ImStucc' 'CBlock']
17 / Exterior2nd	16 : ['VinylSd' 'MetalSd' 'Wd Shng' 'HdBoard' 'Plywo od' 'Wd Sdng' 'CmentBd' 'BrkFace' 'Stucco' 'AsbShng' 'Brk Cmn' 'ImStucc' 'AsphShn' 'Stone' 'Other' 'CBlock']
18 / MasVnrType	5 : ['BrkFace' 'None' 'Stone' 'BrkCmn' nan]
19 / ExterQual	4 : ['Gd' 'TA' 'Ex' 'Fa']
20 / ExterCond	5 : ['TA' 'Gd' 'Fa' 'Po' 'Ex']
21 / Foundation	6 : ['PConc' 'CBlock' 'BrkTil' 'Wood' 'Slab' 'Ston e']
22 / BsmtQual	6 : ['Gd' 'TA' 'Ex' <NA> 'Fa' nan]
23 / BsmtCond	6 : ['TA' 'Gd' <NA> 'Fa' nan 'Po']
24 / BsmtExposure	6 : ['No' 'Gd' 'Mn' 'Av' <NA> nan]
25 / BsmtFinType1	8 : ['GLQ' 'ALQ' 'Unf' 'Rec' 'BLQ' <NA> 'LwQ' nan]
26 / BsmtFinType2	8 : ['Unf' 'BLQ' <NA> 'ALQ' 'Rec' 'LwQ' 'GLQ' nan]
27 / Heating	6 : ['GasA' 'GasW' 'Grav' 'Wall' 'OthW' 'Floor']
28 / HeatingQC	5 : ['Ex' 'Gd' 'TA' 'Fa' 'Po']
29 / CentralAir	2 : ['Y' 'N']
30 / Electrical	6 : ['SBrkr' 'FuseF' 'FuseA' 'FuseP' 'Mix' nan]
31 / KitchenQual	4 : ['Gd' 'TA' 'Ex' 'Fa']
32 / Functional	7 : ['Typ' 'Min1' 'Maj1' 'Min2' 'Mod' 'Maj2' 'Sev']
33 / FireplaceQu	7 : [<NA> 'TA' 'Gd' 'Fa' 'Ex' nan 'Po']
34 / GarageType	8 : ['Attchd' 'Detchd' 'BuiltIn' 'CarPort' <NA> 'Bas

```
ment' nan '2Types']
35 / GarageFinish      5 : ['RFn' 'Unf' 'Fin' <NA> nan]
36 / GarageQual        7 : ['TA' 'Fa' 'Gd' <NA> nan 'Ex' 'Po']
37 / GarageCond        7 : ['TA' 'Fa' <NA> 'Gd' nan 'Po' 'Ex']
38 / PavedDrive        3 : ['Y' 'N' 'P']
39 / PoolQC            5 : [<NA> nan 'Ex' 'Fa' 'Gd']
40 / Fence              6 : [<NA> 'MnPrv' 'GdWo' 'GdPrv' nan 'MnWw']
41 / MiscFeature       6 : [<NA> 'Shed' nan 'Gar2' 'Othr' 'TenC']
42 / SaleType           9 : ['WD' 'New' 'COD' 'ConLD' 'ConLI' 'CWD' 'ConLw' 'Con' 'O
th']
43 / SaleCondition     6 : ['Normal' 'Abnorml' 'Partial' 'AdjLand' 'Alloca'
'Family']
```



```
In [21]: # Categorical & Continuous
```

```
i = 1
for obj in numbers:
    print(i, "/", obj, "\t", len(df[obj].unique()),
          ":", df[obj].unique() if len(df[obj].unique())<150 else '')
    i = i+1

1 / Id      1460 :
2 / MSSubClass  15 : [ 60  20  70  50 190  45  90 120  30  85  80 160  75 18
0 40]
3 / LotFrontage      111 : [ 65.  80.  68.  60.  84.  85.  75.  nan  51.
50.  70.  91.  72.  66.
101.  57.  44.  110.  98.  47.  108.  112.  74.  115.  61.  48.  33.  52.
100.  24.  89.  63.  76.  81.  95.  69.  21.  32.  78.  121.  122.  40.
105.  73.  77.  64.  94.  34.  90.  55.  88.  82.  71.  120.  107.  92.
134.  62.  86.  141.  97.  54.  41.  79.  174.  99.  67.  83.  43.  103.
93.  30.  129.  140.  35.  37.  118.  87.  116.  150.  111.  49.  96.  59.
36.  56.  102.  58.  38.  109.  130.  53.  137.  45.  106.  104.  42.  39.
144.  114.  128.  149.  313.  168.  182.  138.  160.  152.  124.  153.  46.]
4 / LotArea      1073 :
5 / OverallQual      10 : [ 7  6  8  5  9  4 10  3  1  2]
6 / OverallCond      9 : [5 8 6 7 4 2 3 9 1]
7 / YearBuilt      112 : [2003 1976 2001 1915 2000 1993 2004 1973 1931 1939 196
5 2005 1962 2006
1960 1929 1970 1967 1958 1930 2002 1968 2007 1951 1957 1927 1920 1966
1959 1994 1954 1953 1955 1983 1975 1997 1934 1963 1981 1964 1999 1972
1921 1945 1982 1998 1956 1948 1910 1995 1991 2009 1950 1961 1977 1985
1979 1885 1919 1990 1969 1935 1988 1971 1952 1936 1923 1924 1984 1926
1940 1941 1987 1986 2008 1908 1892 1916 1932 1918 1912 1947 1925 1900
1980 1989 1992 1949 1880 1928 1978 1922 1996 2010 1946 1913 1937 1942
1938 1974 1893 1914 1906 1890 1898 1904 1882 1875 1911 1917 1872 1905]
8 / YearRemodAdd      61 : [2003 1976 2002 1970 2000 1995 2005 1973 1950 1
965 2006 1962 2007 1960
2001 1967 2004 2008 1997 1959 1990 1955 1983 1980 1966 1963 1987 1964
1972 1996 1998 1989 1953 1956 1968 1981 1992 2009 1982 1961 1993 1999
1985 1979 1977 1969 1958 1991 1971 1952 1975 2010 1984 1986 1994 1988
1954 1957 1951 1978 1974]
9 / MasVnrArea      328 :
10 / BsmtFinSF1      637 :
11 / BsmtFinSF2      144 : [  0   32  668  486   93  491  506  712  362
41 169  869 150  670
28 1080  181  768  215  374  208  441  184  279  306  180  580  690
692 228  125 1063  620  175  820 1474  264  479  147  232  380  544
294 258  121  391  531  344  539  713  210  311 1120  165  532  96
495 174 1127  139  202  645  123  551  219  606  612  480  182  132
336 468  287   35  499  723  119   40  117  239   80  472   64 1057
127 630  128  377  764  345 1085  435  823  500  290  324  634  411
841 1061  466  396  354  149  193  273  465  400  682  557  230  106
791 240  547  469  177  108  600  492  211  168 1031  438  375  144
81 906  608  276  661   68  173  972  105  420  546  334  352  872
110 627  163 1029]
12 / BsmtUnfSF      780 :
13 / TotalBsmtSF      721 :
14 / 1stFlrSF      753 :
15 / 2ndFlrSF      417 :
16 / LowQualFinSF      24 : [  0  360  513  234  528  572  144  392  371  390  420  47
```

```
3 156 515 80 53 232 481
    120 514 397 479 205 384]
17 / GrLivArea     861 :
18 / BsmtFullBath      4 : [1 0 2 3]
19 / BsmtHalfBath      3 : [0 1 2]
20 / FullBath        4 : [2 1 3 0]
21 / HalfBath         3 : [1 0 2]
22 / BedroomAbvGr      8 : [3 4 1 2 0 5 6 8]
23 / KitchenAbvGr      4 : [1 2 3 0]
24 / TotRmsAbvGrd     12 : [ 8   6   7   9   5 11   4 10 12   3   2 14]
25 / Fireplaces        4 : [0 1 2 3]
26 / GarageYrBlt       98 : [2003. 1976. 2001. 1998. 2000. 1993. 2004. 197
3. 1931. 1939. 1965. 2005.
    1962. 2006. 1960. 1991. 1970. 1967. 1958. 1930. 2002. 1968. 2007. 2008.
    1957. 1920. 1966. 1959. 1995. 1954. 1953.  nan 1983. 1977. 1997. 1985.
    1963. 1981. 1964. 1999. 1935. 1990. 1945. 1987. 1989. 1915. 1956. 1948.
    1974. 2009. 1950. 1961. 1921. 1900. 1979. 1951. 1969. 1936. 1975. 1971.
    1923. 1984. 1926. 1955. 1986. 1988. 1916. 1932. 1972. 1918. 1980. 1924.
    1996. 1940. 1949. 1994. 1910. 1978. 1982. 1992. 1925. 1941. 2010. 1927.
    1947. 1937. 1942. 1938. 1952. 1928. 1922. 1934. 1906. 1914. 1946. 1908.
    1929. 1933.]
27 / GarageCars        5 : [2 3 1 0 4]
28 / GarageArea        441 :
29 / WoodDeckSF        274 :
30 / OpenPorchSF       202 :
31 / EnclosedPorch     120 : [ 0 272 228 205 176 87 172 102 37 144 64 1
14 202 128 156 44 77 192
    140 180 183 39 184 40 552 30 126 96 60 150 120 112 252 52 224 234
    244 268 137 24 108 294 177 218 242 91 160 130 169 105 34 248 236 32
    80 115 291 116 158 210 36 200 84 148 136 240 54 100 189 293 164 216
    239 67 90 56 129 98 143 70 386 154 185 134 196 264 275 230 254 68
    194 318 48 94 138 226 174 19 170 220 214 280 190 330 208 145 259 81
    42 123 162 286 168 20 301 198 221 212 50 99]
32 / 3SsnPorch        20 : [ 0 320 407 130 180 168 140 508 238 245 196 144 182 16
2 23 216 96 153
    290 304]
33 / ScreenPorch       76 : [ 0 176 198 291 252 99 184 168 130 142 192 41
0 224 266 170 154 153 144
    128 259 160 271 234 374 185 182 90 396 140 276 180 161 145 200 122 95
    120 60 126 189 260 147 385 287 156 100 216 210 197 204 225 152 175 312
    222 265 322 190 233 63 53 143 273 288 263 80 163 116 480 178 440 155
    220 119 165 40]
34 / PoolArea          8 : [ 0 512 648 576 555 480 519 738]
35 / MiscVal           21 : [ 0 700 350 500 400 480 450 15500 1200
800 2000 600
    3500 1300 54 620 560 1400 8300 1150 2500]
36 / MoSold            12 : [ 2 5 9 12 10 8 11 4 1 7 3 6]
37 / YrSold             5 : [2008 2007 2006 2009 2010]
38 / SalePrice         663 :
```

In [22]: # Quan sát 2 kết quả trên để kết Luận

2. Phân tích đơn biến

```
In [107]: %matplotlib inline
```

```
In [23]: df.columns
```

```
Out[23]: Index(['Id', 'MSSubClass', 'MSZoning', 'LotFrontage', 'LotArea', 'Street',  
    'Alley', 'LotShape', 'LandContour', 'Utilities', 'LotConfig',  
    'LandSlope', 'Neighborhood', 'Condition1', 'Condition2', 'BldgType',  
    'HouseStyle', 'OverallQual', 'OverallCond', 'YearBuilt', 'YearRemodAdd',  
    'RoofStyle', 'RoofMatl', 'Exterior1st', 'Exterior2nd', 'MasVnrType',  
    'MasVnrArea', 'ExterQual', 'ExterCond', 'Foundation', 'BsmtQual',  
    'BsmtCond', 'BsmtExposure', 'BsmtFinType1', 'BsmtFinSF1',  
    'BsmtFinType2', 'BsmtFinSF2', 'BsmtUnfSF', 'TotalBsmtSF', 'Heating',  
    'HeatingQC', 'CentralAir', 'Electrical', '1stFlrSF', '2ndFlrSF',  
    'LowQualFinSF', 'GrLivArea', 'BsmtFullBath', 'BsmtHalfBath', 'FullBath',  
    'HalfBath', 'BedroomAbvGr', 'KitchenAbvGr', 'KitchenQual',  
    'TotRmsAbvGrd', 'Functional', 'Fireplaces', 'FireplaceQu', 'GarageType',  
    'GarageYrBlt', 'GarageFinish', 'GarageCars', 'GarageArea', 'GarageQual',  
    'GarageCond', 'PavedDrive', 'WoodDeckSF', 'OpenPorchSF',  
    'EnclosedPorch', '3SsnPorch', 'ScreenPorch', 'PoolArea', 'PoolQC',  
    'Fence', 'MiscFeature', 'MiscVal', 'MoSold', 'YrSold', 'SaleType',  
    'SaleCondition', 'SalePrice'],  
    dtype='object')
```

```
In [113]: # Dựa trên EDA có thể chọn các feature sau để phân tích  
# Vì có mối tương quan cao với SalePrice  
features = ['LotArea', 'OverallQual', 'YearBuilt', '1stFlrSF', '2ndFlrSF',  
    'GrLivArea', 'FullBath', 'BedroomAbvGr', 'TotRmsAbvGrd',  
    'GarageArea', 'TotalBsmtSF']
```

```
In [114]: i = 1
for obj in features:
    print(i, "/", obj, "\t", len(df[obj].unique()),
          ":", df[obj].unique() if len(df[obj].unique()) < 150 else '')
    i = i+1

1 / LotArea      1073 :
2 / OverallQual      10 : [ 7  6  8  5  9  4 10  3  1  2]
3 / YearBuilt      112 : [2003 1976 2001 1915 2000 1993 2004 1973 1931 1939 1965
2005 1962 2006
1960 1929 1970 1967 1958 1930 2002 1968 2007 1951 1957 1927 1920 1966
1959 1994 1954 1953 1955 1983 1975 1997 1934 1963 1981 1964 1999 1972
1921 1945 1982 1998 1956 1948 1910 1995 1991 2009 1950 1961 1977 1985
1979 1885 1919 1990 1969 1935 1988 1971 1952 1936 1923 1924 1984 1926
1940 1941 1987 1986 2008 1908 1892 1916 1932 1918 1912 1947 1925 1900
1980 1989 1992 1949 1880 1928 1978 1922 1996 2010 1946 1913 1937 1942
1938 1974 1893 1914 1906 1890 1898 1904 1882 1875 1911 1917 1872 1905]
4 / 1stFlrSF      753 :
5 / 2ndFlrSF      417 :
6 / GrLivArea      861 :
7 / FullBath       4 : [2 1 3 0]
8 / BedroomAbvGr      8 : [3 4 1 2 0 5 6 8]
9 / TotRmsAbvGrd     12 : [ 8  6  7  9  5 11  4 10 12  3  2 14]
10 / GarageArea      441 :
11 / TotalBsmtSF     721 :
```

Continuous variable

- LotArea
- 1stFlrSF
- 2ndFlrSF
- YearBuilt
- GrLivArea
- GarageArea
- TotalBsmtSF

```
In [170]: def univariate_analysis_continuous_variable(df, feature):
    print("Describe:")
    print(feature.describe(include='all'))
    print("Mode:", feature.mode())
    print("Range:", feature.values.ptp())
    print("IQR:", scipy.stats.iqr(feature))
    print("Var:", feature.var())
    print("Std:", feature.std())
    print("Skew:", feature.skew())
    print("Kurtosis:", feature.kurtosis())
```

```
In [ ]: # Number of upper, lower outliers
def check_outlier(df, feature):
    plt.boxplot(feature)
    plt.show()
    Q1 = np.percentile(feature, 25)
    Q3 = np.percentile(feature, 75)
    n_0_upper = df[feature > (Q3 + 1.5*scipy.stats.iqr(feature))].shape[0]
    print("Number of upper outliers:", n_0_upper)
    n_0_lower = df[feature < (Q1 - 1.5*scipy.stats.iqr(feature))].shape[0]
    print("Number of lower outliers:", n_0_lower)
    # Percentage of outliers
    outliers_per = (n_0_lower + n_0_upper)/df.shape[0]
    print("Percentage of outliers:", outliers_per)
    return Q1, Q3, n_0_upper, n_0_lower, outliers_per
```

```
In [171]: def univariate_visualization_analysis_continuous_variable(feature):
    # Histogram
    plt.figure(figsize=(10, 5))
    plt.subplot(1, 2, 1)
    #sns.distplot(feature)
    sns.histplot(feature, kde=True, stat="density")
    plt.subplot(1, 2, 2)
    plt.hist(feature)
    plt.show()
    # Boxplot
    plt.figure(figsize=(4,5))
    plt.boxplot(feature)
    plt.show()
```

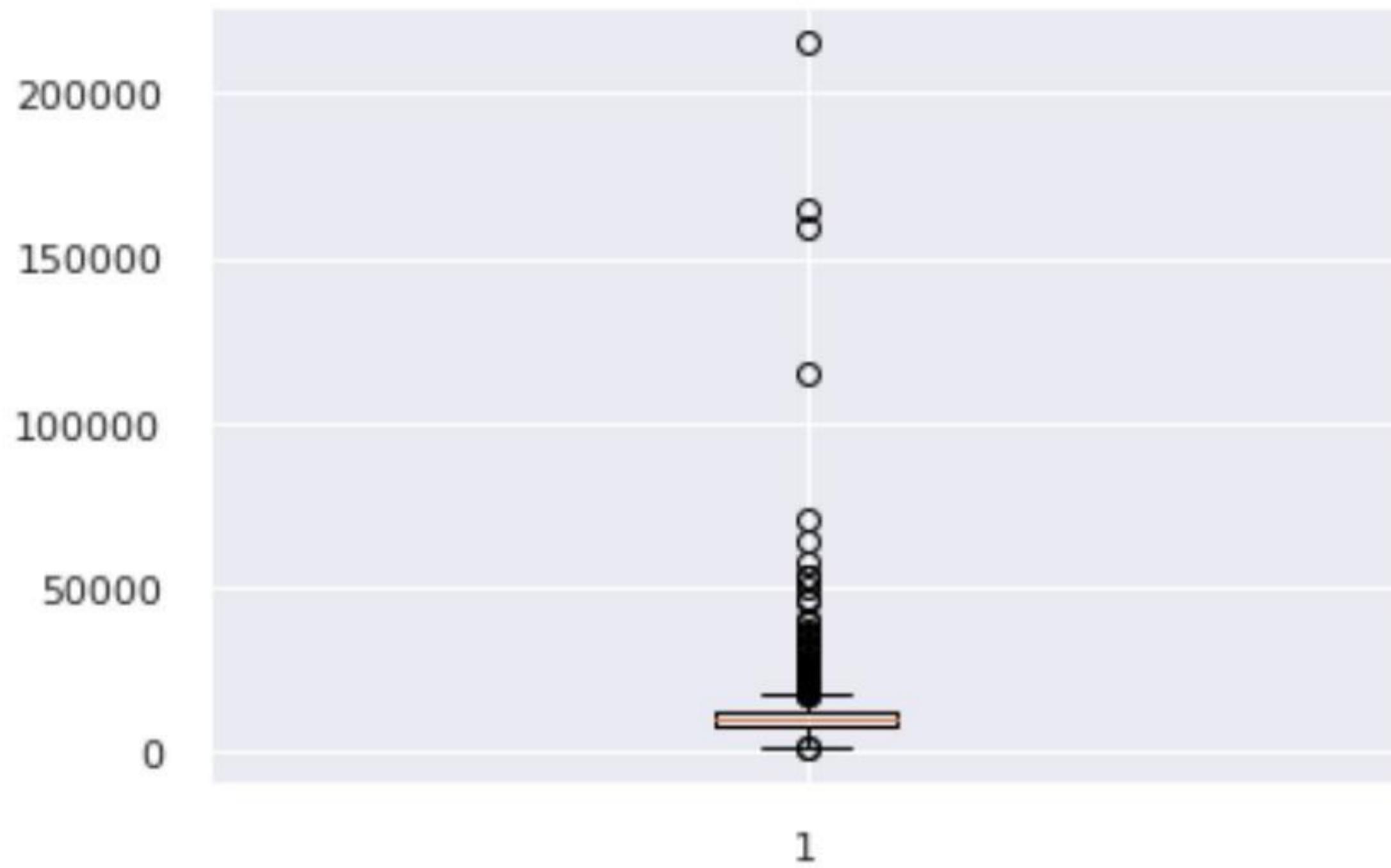
```
In [172]: univariate_analysis_continuous_variable(df, df['LotArea'])
```

Describe:

count	1460.000000
mean	10516.828082
std	9981.264932
min	1300.000000
25%	7553.500000
50%	9478.500000
75%	11601.500000
max	215245.000000

Name: LotArea, dtype: float64
Mode: 0 7200
dtype: int64
Range: 213945
IQR: 4048.0
Var: 99625649.65034176
Std: 9981.26493237915
Skew: 12.207687851233496
Kurtosis: 203.24327101886033

```
In [179]: Q1_LotArea, Q3_LotArea, n_0_upper_LotArea, n_0_lower_LotArea,  
        outliers_per_LotArea = check_outlier(df, df['LotArea'])
```



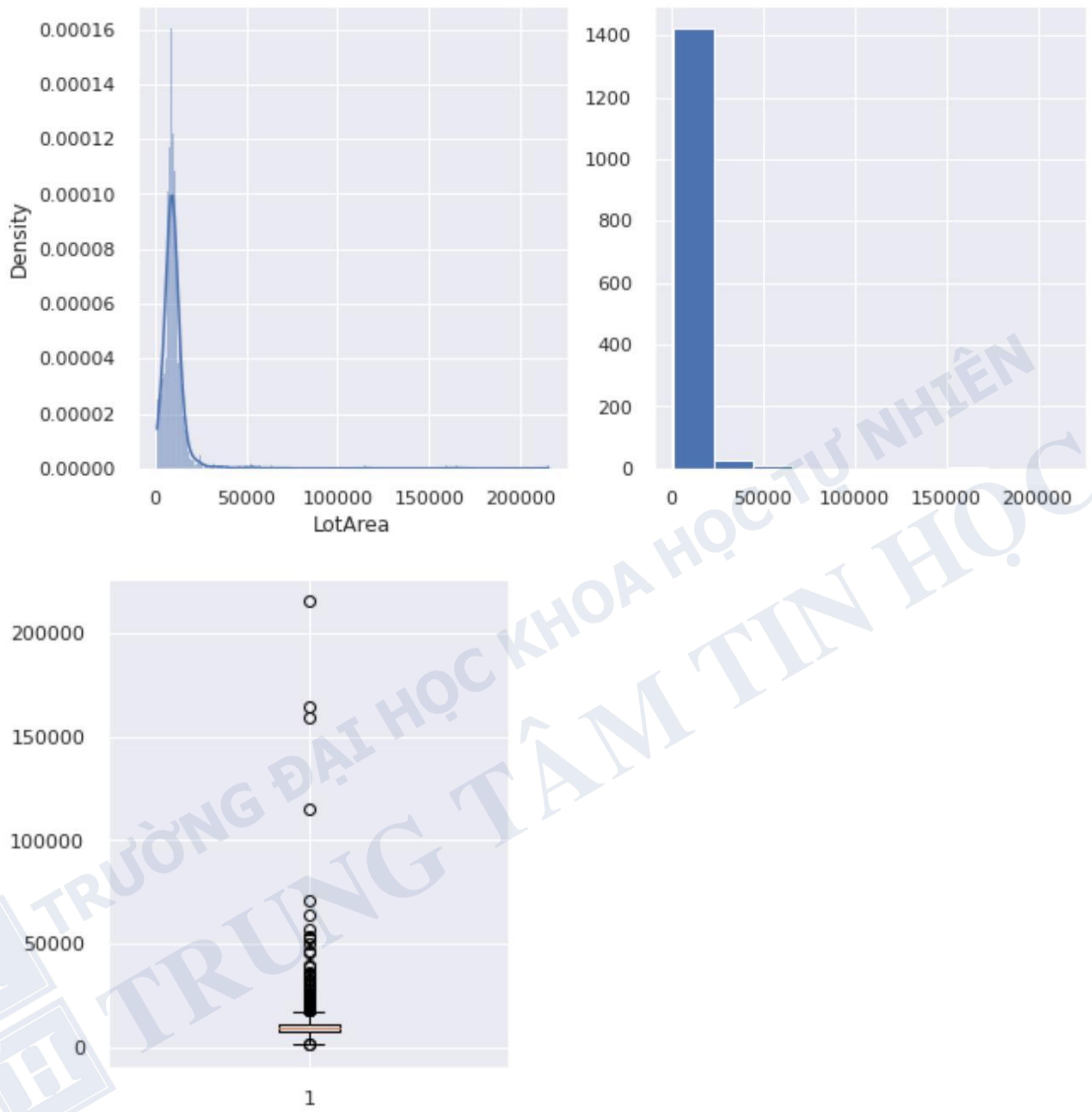
Number of upper outliers: 67
Number of lower outliers: 2
Percentage of outliers: 0.04726027397260274

```
In [180]: outliers_per_LotArea *100
```

```
Out[180]: 4.726027397260274
```

```
In [ ]: # Có ~4.7% dữ liệu age có outlier  
        # Xem xét loại bỏ outliers ??? => tùy vào nghiệp vụ của từng lĩnh vực
```

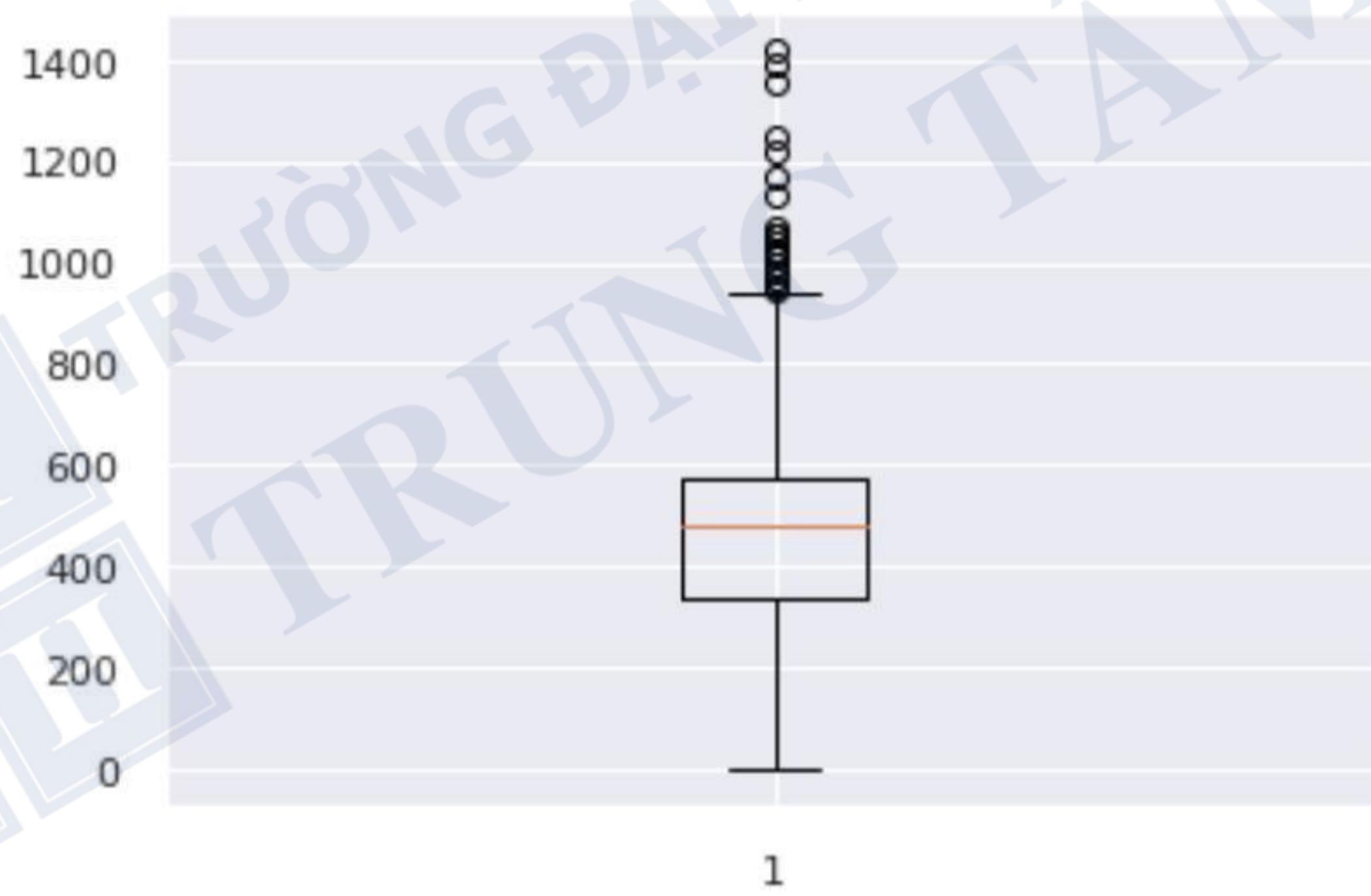
```
In [173]: univariate_visualization_analysis_continuous_variable(df[ 'LotArea' ])
```



```
In [174]: # GarageArea  
univariate_analysis_continuous_variable(df, df['GarageArea'])
```

Describe:
count 1460.000000
mean 472.980137
std 213.804841
min 0.000000
25% 334.500000
50% 480.000000
75% 576.000000
max 1418.000000
Name: GarageArea, dtype: float64
Mode: 0 0
dtype: int64
Range: 1418
IQR: 241.5
Var: 45712.51022890514
Std: 213.80484145338042
Skew: 0.17998090674623907
Kurtosis: 0.9170672022708684

```
In [181]: Q1_GarageArea, Q3_GarageArea, n_0_upper_GarageArea, n_0_lower_GarageArea,  
outliers_per_GarageArea = check_outlier(df, df['GarageArea'])
```



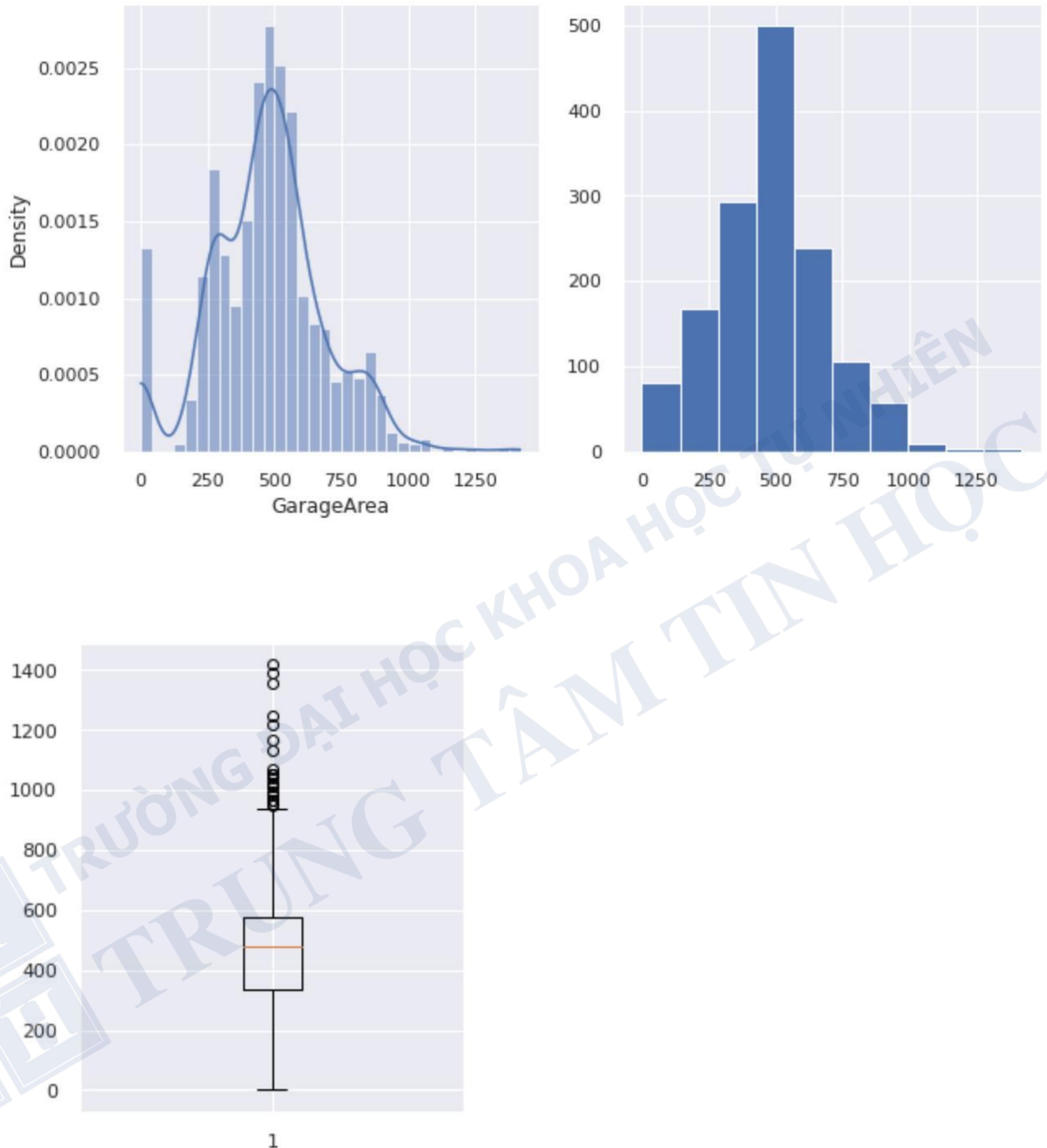
Number of upper outliers: 21
Number of lower outliers: 0
Percentage of outliers: 0.014383561643835616

```
In [182]: outliers_per_GarageArea * 100
```

Out[182]: 1.4383561643835616

```
In [ ]: # Có ~1.4% dữ liệu age có outlier  
# Xem xét loại bỏ outliers ???
```

```
In [175]: univariate_visualization_analysis_continuous_variable(df[ 'GarageArea' ])
```



Note: Các thuộc tính còn lại: làm tương tự như trên

Categorical Variables

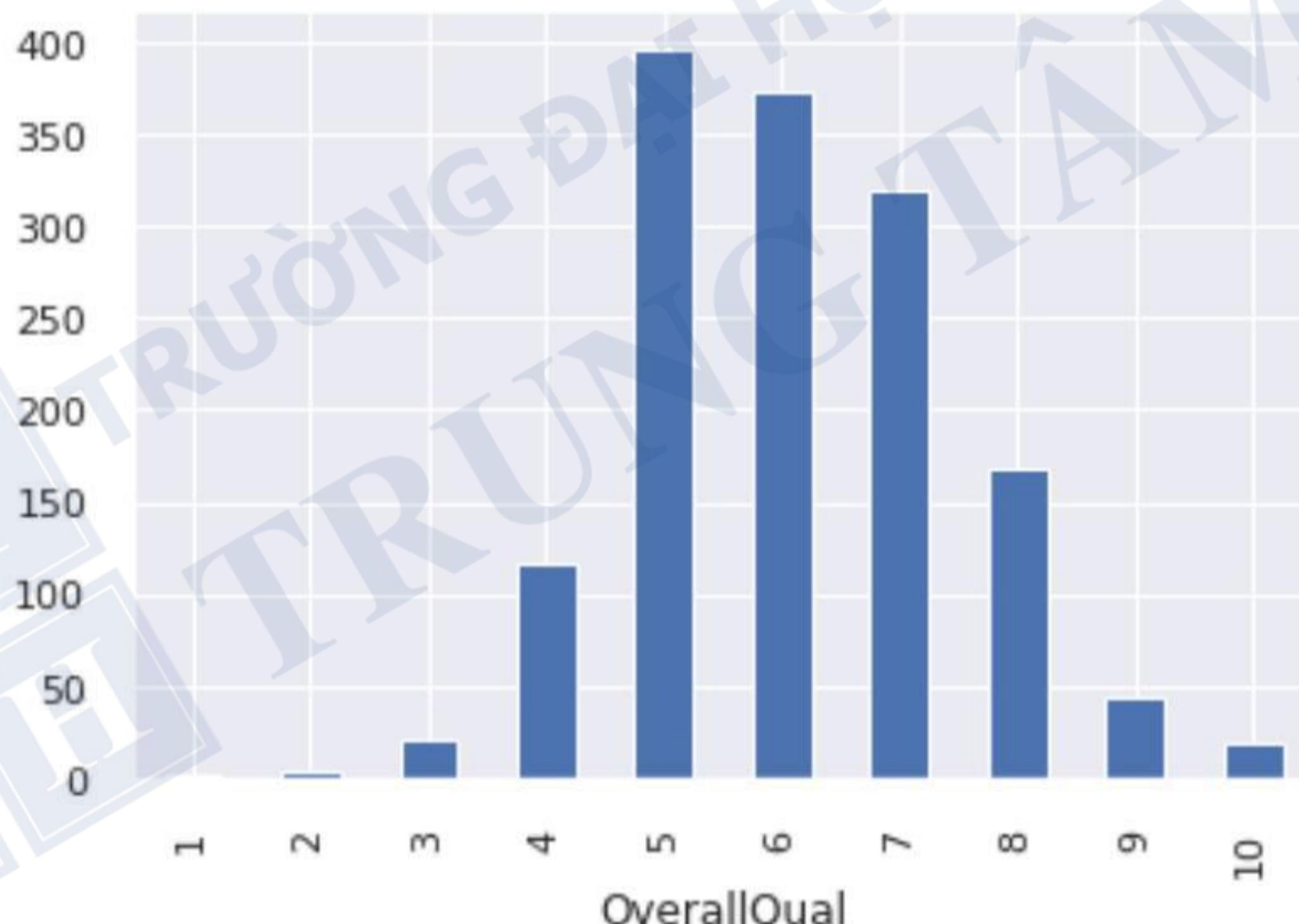
- OverallQual
- FullBath
- BedroomAbvGr
- TotRmsAbvGrd

- TotalBsmtSF

```
In [132]: def univariate_analysis_categorical_variable(df, group_by_col, count_col):  
    group_by_col_count = (df.groupby(group_by_col).count())[count_col]  
    print(group_by_col_count)  
    group_by_col_count.plot.bar()  
    plt.show()
```

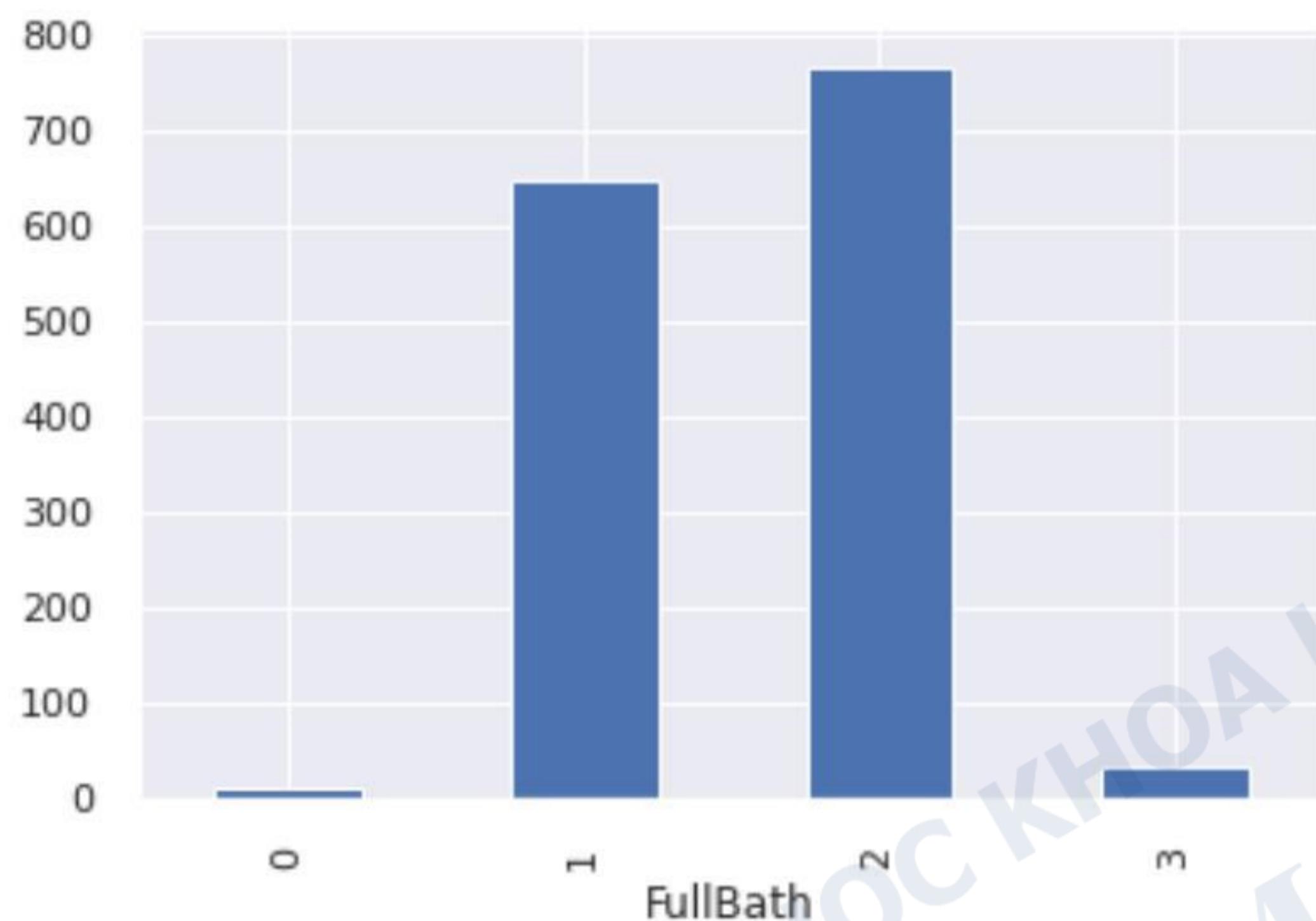
```
In [134]: # OverallQual  
univariate_analysis_categorical_variable(df, "OverallQual", "Id")
```

```
OverallQual  
1      2  
2      3  
3     20  
4    116  
5    397  
6    374  
7    319  
8    168  
9     43  
10    18  
Name: Id, dtype: int64
```



```
In [133]: # FullBath  
univariate_analysis_categorical_variable(df, "FullBath", "Id")
```

```
FullBath  
0      9  
1    650  
2    768  
3     33  
Name: Id, dtype: int64
```

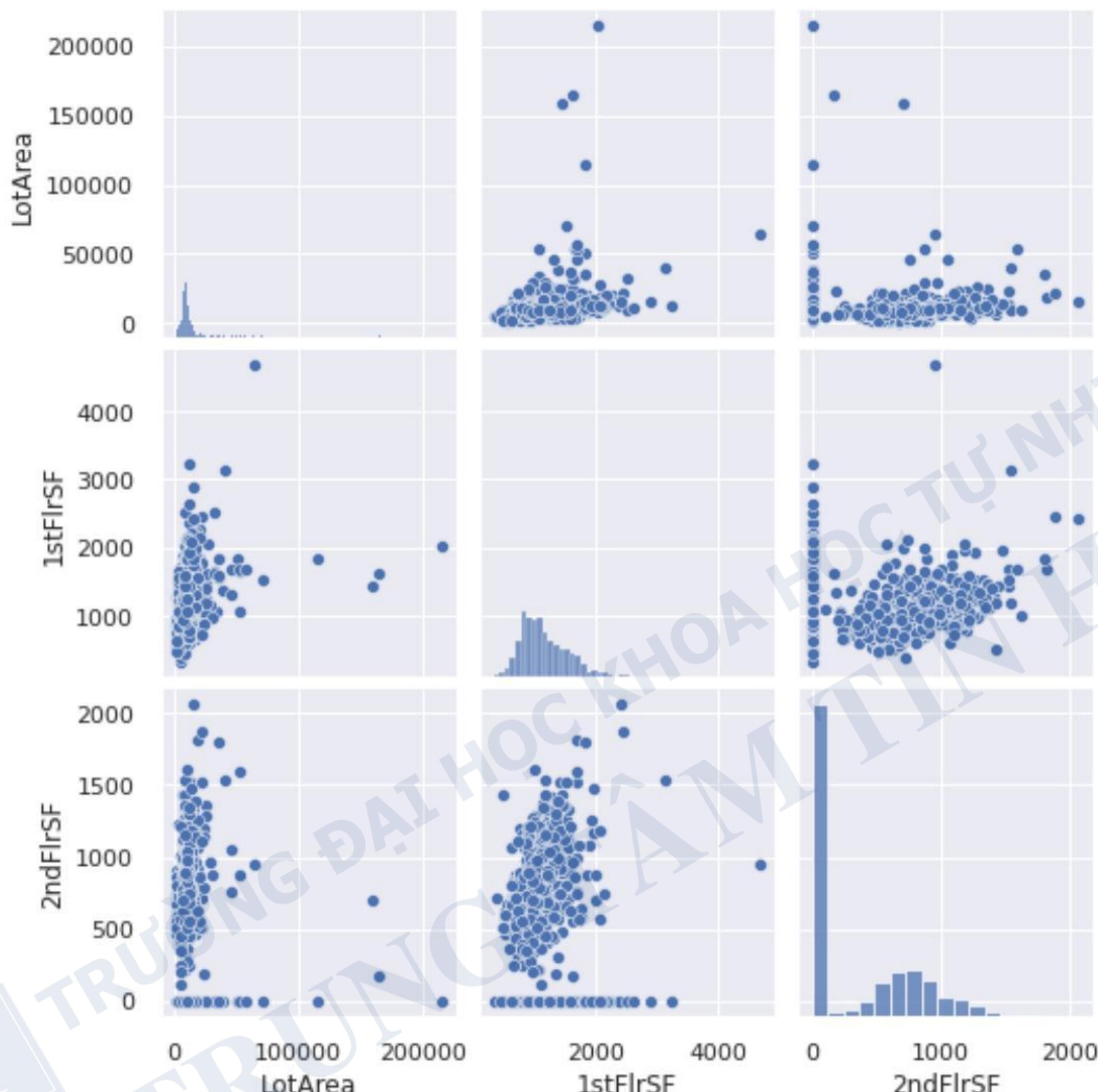


- Note: Các thuộc tính còn lại làm tương tự như trên

3. Phân tích hai biến

- Continuous & Continuous
- Categorical & Categorical
- Categorical & Continuous

```
In [135]: # Continuous & Continuous  
sns.pairplot(df[['LotArea', '1stFlrSF', '2ndFlrSF']])  
plt.show()
```



```
In [55]: # Không quan hệ tuyến tính  
df[['LotArea', '1stFlrSF', '2ndFlrSF']].corr()
```

Out[55]:

	LotArea	1stFlrSF	2ndFlrSF
LotArea	1.000000	0.299475	0.050986
1stFlrSF	0.299475	1.000000	-0.202646
2ndFlrSF	0.050986	-0.202646	1.000000

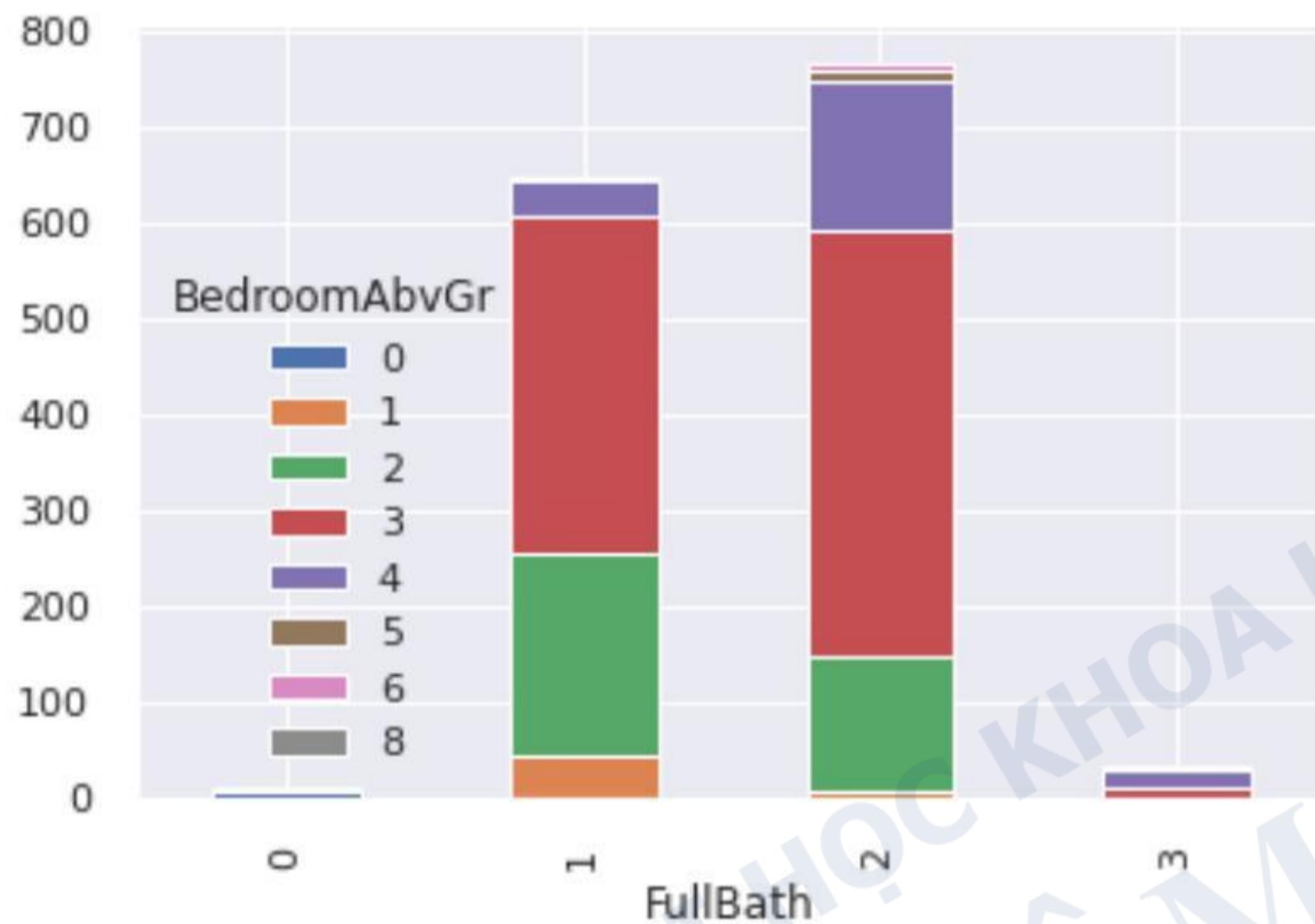
```
In [ ]: # Categorical & Categorical
```

```
In [58]: # chi-squared test with similar proportions
from scipy.stats import chi2_contingency
from scipy.stats import chi2
```

```
In [176]: def categorical_categorical(feature1, feature2):
    # Contingency table
    table_FB = pd.crosstab(feature1, feature2)
    print(table_FB)
    table_FB.plot(kind='bar', stacked=True)
    plt.show()
    # Chi-Square Test
    stat, p, dof, expected = chi2_contingency(table_FB)
    print('dof=%d' % dof)
    print('p=', p)
    # interpret test-statistic
    prob = 0.95
    critical = chi2.ppf(prob, dof)
    print('probability=%.3f, critical=%.3f, stat=%.3f' % (prob, critical, stat))
    # interpret p-value
    alpha = 1.0 - prob
    print('significance=%.3f, p=%.3f' % (alpha, p))
    if p <= alpha:
        print('Dependent (reject H0)')
    else:
        print('Independent (fail to reject H0)')
```

```
In [177]: # 'FullBath', 'BedroomAbvGr', 'TotRmsAbvGr'
# Contingency table: Ho: 'FullBath' and 'BedroomAbvGr' independent
categorical_categorical(df['FullBath'], df['BedroomAbvGr'])
```

BedroomAbvGr	0	1	2	3	4	5	6	8	
FullBath	0	5	2	2	0	0	0	0	
	1	1	42	213	352	37	5	0	0
	2	0	6	143	443	156	12	7	1
	3	0	0	0	9	20	4	0	0



```
dof=21
p= 9.022959522651409e-177
probability=0.950, critical=32.671, stat=898.930
significance=0.050, p=0.000
Dependent (reject H0)
```

Note: Các cặp còn lại làm tương tự như trên

```
In [ ]: # Categorical & Continuous
```

```
In [63]: # 'FullBath' có bị ảnh hưởng bởi 'LotArea'? => ANOVA ONEWAY
df_sub = df[['FullBath', 'LotArea']]
df_sub.head()
```

```
Out[63]:
```

	FullBath	LotArea
0	2	8450
1	2	9600
2	2	11250
3	1	9550
4	2	14260

```
In [64]: import matplotlib.pyplot as plt  
plt.figure(figsize=(12,10))  
sns.boxplot(x="FullBath", y="LotArea", data=df_sub, palette="Set3")  
plt.show()
```

```
In [65]: import statsmodels.api as sm  
from statsmodels.formula.api import ols
```

```
In [66]: model = ols('LotArea ~ C(FullBath)', data=df_sub).fit()  
anova_table = sm.stats.anova_lm(model, typ=2)  
anova_table
```

Out[66]:

	sum_sq	df	F	PR(>F)
C(FullBath)	3.332090e+09	3.0	11.386809	2.207543e-07
Residual	1.420217e+11	1456.0	NaN	NaN

- Giải thích: P-value thu được từ phân tích ANOVA cho LotArea và FullBath phối hợp có ý nghĩa thống kê ($P < 0.05$).
- Kết luận: LotArea ảnh hưởng đáng kể đến FullBath

```
In [67]: from statsmodels.stats.multicomp import pairwise_tukeyhsd  
# perform multiple pairwise comparison (Tukey HSD)  
m_comp = pairwise_tukeyhsd(endog=df_sub['LotArea'],  
                           groups=df_sub['FullBath'],  
                           alpha=0.05)  
print(m_comp)
```

```
Multiple Comparison of Means - Tukey HSD, FWER=0.05  
=====  
group1 group2 meandiff p-adj      lower      upper    reject  
-----  
0      1   -5695.6583  0.3146 -14221.4451  2830.1286 False  
0      2   -3587.0109  0.6772 -12103.848   4929.8263 False  
0      3   2477.0404   0.9   -7075.4392  12029.52  False  
1      2   2108.6474   0.001   754.7975  3462.4974 True  
1      3   8172.6987   0.001  3639.9003 12705.4971 True  
2      3   6064.0513  0.0032  1548.1091 10579.9935 True  
-----
```

- Các kết quả trên từ Tukey HSD cho thấy 0-1, 0-2, 0-3: chấp nhận Ho, các so sánh cặp khác về số phòng bắc bỏ Ho và chỉ ra sự khác biệt đáng kể về mặt thống kê.

4. Xử lý dữ liệu trùng, thiếu

```
In [141]: df_now = df[['LotArea', 'OverallQual', 'YearBuilt', '1stFlrSF', '2ndFlrSF',  
                  'GrLivArea', 'FullBath', 'BedroomAbvGr', 'TotRmsAbvGrd',  
                  'GarageArea', 'TotalBsmtSF']]
```

```
In [142]: # Kiểm tra dữ liệu thiếu  
df_now.isnull().sum()  
# Không có dữ liệu thiếu
```

```
Out[142]: LotArea      0  
OverallQual     0  
YearBuilt       0  
1stFlrSF        0  
2ndFlrSF        0  
GrLivArea       0  
FullBath         0  
BedroomAbvGr    0  
TotRmsAbvGrd   0  
GarageArea      0  
TotalBsmtSF     0  
dtype: int64
```

```
In [144]: # Phát hiện dữ liệu trùng  
df_now.duplicated().sum()
```

```
Out[144]: 8
```

```
In [139]: # Trước khi xóa dữ liệu trùng  
df_now.shape
```

```
Out[139]: (1460, 11)
```

```
In [145]: # Sau khi xóa dữ liệu trùng  
df_now = df_now.drop_duplicates()  
df_now.shape
```

```
Out[145]: (1452, 11)
```

5. Phát hiện và xử lý ngoại lệ

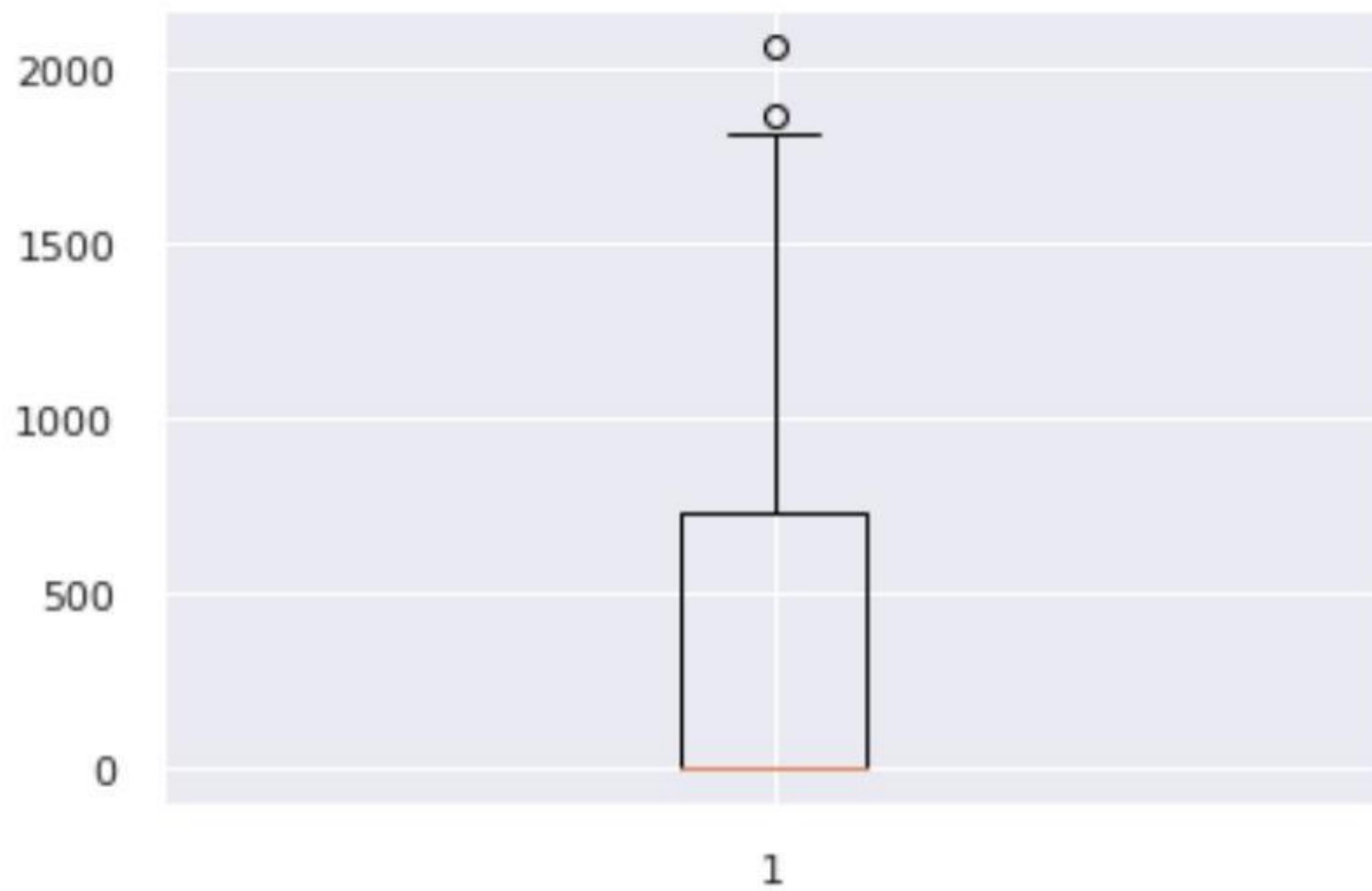
```
In [ ]: sns.pairplot(df[features])  
plt.show()
```

```
In [166]: Q1_1stFlrSF, Q3_1stFlrSF, n_0_upper_1stFlrSF, n_0_lower_1stFlrSF,  
        outliers_per_1stFlrSF = check_outlier(df_now, df_now['1stFlrSF'])
```



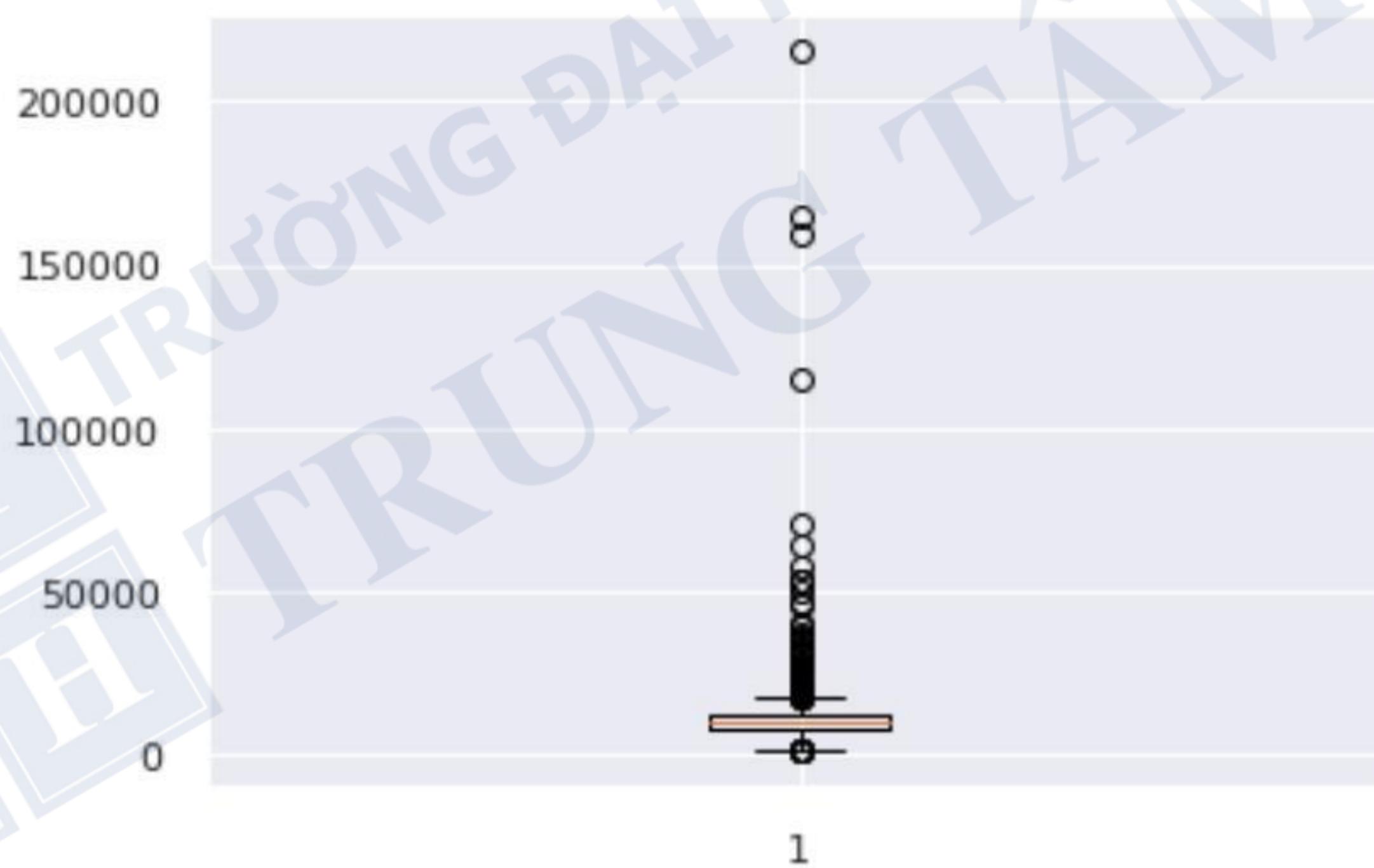
Number of upper outliers: 20
Number of lower outliers: 0
Percentage of outliers: 0.013774104683195593

```
In [167]: Q1_2ndFlrSF, Q3_2ndFlrSF, n_0_upper_2ndFlrSF, n_0_lower_2ndFlrSF,  
        outliers_per_2ndFlrSF = check_outlier(df_now, df_now['2ndFlrSF'])
```



Number of upper outliers: 2
Number of lower outliers: 0
Percentage of outliers: 0.0013774104683195593

```
In [168]: Q1_LotArea, Q3_LotArea, n_0_upper_LotArea, n_0_lower_LotArea,  
        outliers_per_LotArea = check_outlier(df_now, df_now['LotArea'])
```



Number of upper outliers: 68
Number of lower outliers: 6
Percentage of outliers: 0.05096418732782369

```
In [169]: (outliers_per_1stFlrSF + outliers_per_2ndFlrSF + outliers_per_LotArea)*100
```

Out[169]: 6.6115702479338845

- Có thể drop outliers của '2ndFlrSF', '1stFlrSF', 'LotArea': vì tổng số outliers là 6.6% dữ liệu
- Cũng có thể không cần drop thay vào đó khi áp dụng Machine Learning thì dùng thuật toán Decision Tree/ Random Forest (LDS6)

In [90]: # Xem xét thêm biên ngoài: major
<https://m.wikihow.com/Calculate-Outliers>
Công thức: $O_u = Q3 + 3*IQR$, $O_L = Q1 - 3*IQR$
hoặc tính trung bình trước và sau khi loại bỏ outlier

In [93]: # Nên loại bỏ ngoại lệ khi chênh lệch lớn

In [96]: # Không nhất thiết phải loại bỏ ngoại lệ khi chênh lệch nhỏ

In [100]: # Vì 3 phân phối này đều không là phân phối chuẩn => không drop theo z-score

In [101]: # Hoặc có thể xem xét chỉnh dữ liệu Log(cột)

