Import libraries

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import scipy.stats as stats

%matplotlib inline

from sklearn.preprocessing import PolynomialFeatures, MinMaxScaler
from sklearn.linear_model import LinearRegression, Ridge, Lasso
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

import warnings
warnings.filterwarnings('ignore')
```

Perfrom global settings

```
pd.options.display.float_format = '{:.2f}'.format
pd.options.display.max_colwidth = 500
```

Data Understanding, Preparation

```
df = pd.read_csv("/content/drive/MyDrive/trainhouses.csv")
df.head()
```

Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour

Get Shape of the dataset

```
print(df.shape)
```

```
(1460, 81)
```

Get columns of the dataset

```
print(df.columns)
```

```
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')
```

Get dataset info

```
df.info()
```

```
 25  MasVnrType     1452 non-null   object
 26  MasVnrArea     1452 non-null   float64
 27  ExterQual      1460 non-null   object
 28  ExterCond      1460 non-null   object
 29  Foundation     1460 non-null   object
 30  BsmtQual       1423 non-null   object
 31  BsmtCond       1423 non-null   object
 32  BsmtExposure   1422 non-null   object
 33  BsmtFinType1   1423 non-null   object
 34  BsmtFinSF1     1460 non-null   int64
 35  BsmtFinType2   1422 non-null   object
 36  BsmtFinSF2     1460 non-null   int64
 37  BsmtUnfSF      1460 non-null   int64
 38  TotalBsmtSF    1460 non-null   int64
```

```
 38  TotalBsmtSF    1460 non-null   int64
 39  Heating        1460 non-null   object
 40  HeatingQC      1460 non-null   object
 41  CentralAir     1460 non-null   object
 42  Electrical     1459 non-null   object
 43  1stFlrSF       1460 non-null   int64
 44  2ndFlrSF       1460 non-null   int64
 45  LowQualFinSF   1460 non-null   int64
 46  GrLivArea      1460 non-null   int64
 47  BsmtFullBath   1460 non-null   int64
 48  BsmtHalfBath   1460 non-null   int64
 49  FullBath       1460 non-null   int64
 50  HalfBath       1460 non-null   int64
 51  BedroomAbvGr   1460 non-null   int64
 52  KitchenAbvGr   1460 non-null   int64
 53  KitchenQual    1460 non-null   object
 54  TotRmsAbvGrd   1460 non-null   int64
 55  Functional     1460 non-null   object
 56  Fireplaces     1460 non-null   int64
 57  FireplaceQu    770 non-null    object
 58  GarageType     1379 non-null   object
 59  GarageYrBlt    1379 non-null   float64
 60  GarageFinish   1379 non-null   object
 61  GarageCars     1460 non-null   int64
 62  GarageArea     1460 non-null   int64
 63  GarageQual     1379 non-null   object
 64  GarageCond     1379 non-null   object
 65  PavedDrive     1460 non-null   object
 66  WoodDeckSF     1460 non-null   int64
 67  OpenPorchSF    1460 non-null   int64
 68  EnclosedPorch  1460 non-null   int64
 69  3SsnPorch      1460 non-null   int64
 70  ScreenPorch    1460 non-null   int64
 71  PoolArea       1460 non-null   int64
 72  PoolQC         7 non-null      object
 73  Fence          281 non-null    object
 74  MiscFeature    54 non-null     object
 75  MiscVal        1460 non-null   int64
 76  MoSold         1460 non-null   int64
 77  YrSold         1460 non-null   int64
 78  SaleType       1460 non-null   object
 79  SaleCondition  1460 non-null   object
 80  SalePrice      1460 non-null   int64
dtypes: float64(3), int64(35), object(43)
memory usage: 924.0+ KB
```

Describe -Numerical columns

Apply transpose for better understanding

```
df.describe().T
```

Id	1460.00	730.50	421.61	1.00	365.75	730.50	1095.25
MSSubClass	1460.00	56.90	42.30	20.00	20.00	50.00	70.00
LotFrontage	1201.00	70.05	24.28	21.00	59.00	69.00	80.00
LotArea	1460.00	10516.83	9981.26	1300.00	7553.50	9478.50	11601.50
OverallQual	1460.00	6.10	1.38	1.00	5.00	6.00	7.00
OverallCond	1460.00	5.58	1.11	1.00	5.00	5.00	6.00
YearBuilt	1460.00	1971.27	30.20	1872.00	1954.00	1973.00	2000.00
YearRemodAdd	1460.00	1984.87	20.65	1950.00	1967.00	1994.00	2004.00
MasVnrArea	1452.00	103.69	181.07	0.00	0.00	0.00	166.00
BsmtFinSF1	1460.00	443.64	456.10	0.00	0.00	383.50	712.25
BsmtFinSF2	1460.00	46.55	161.32	0.00	0.00	0.00	0.00
BsmtUnfSF	1460.00	567.24	441.87	0.00	223.00	477.50	808.00
TotalBsmtSF	1460.00	1057.43	438.71	0.00	795.75	991.50	1298.25
1stFlrSF	1460.00	1162.63	386.59	334.00	882.00	1087.00	1391.25
2ndFlrSF	1460.00	346.99	436.53	0.00	0.00	0.00	728.00
LowQualFinSF	1460.00	5.84	48.62	0.00	0.00	0.00	0.00
GrLivArea	1460.00	1515.46	525.48	334.00	1129.50	1464.00	1776.75
BsmtFullBath	1460.00	0.43	0.52	0.00	0.00	0.00	1.00
BsmtHalfBath	1460.00	0.06	0.24	0.00	0.00	0.00	0.00
FullBath	1460.00	1.57	0.55	0.00	1.00	2.00	2.00
HalfBath	1460.00	0.38	0.50	0.00	0.00	0.00	1.00
BedroomAbvGr	1460.00	2.87	0.82	0.00	2.00	3.00	3.00
KitchenAbvGr	1460.00	1.05	0.22	0.00	1.00	1.00	1.00
TotRmsAbvGrd	1460.00	6.52	1.63	2.00	5.00	6.00	7.00
Fireplaces	1460.00	0.61	0.64	0.00	0.00	1.00	1.00
GarageYrBlt	1379.00	1978.51	24.69	1900.00	1961.00	1980.00	2002.00
GarageCars	1460.00	1.77	0.75	0.00	1.00	2.00	2.00
GarageArea	1460.00	472.98	213.80	0.00	334.50	480.00	576.00
WoodDeckSF	1460.00	94.24	125.34	0.00	0.00	0.00	168.00
OpenPorchSF	1460.00	46.66	66.26	0.00	0.00	25.00	68.00
EnclosedPorch	1460.00	21.95	61.12	0.00	0.00	0.00	0.00
3SsnPorch	1460.00	3.41	29.32	0.00	0.00	0.00	0.00

ScreenPorch	1460.00	15.06	55.76	0.00	0.00	0.00	0.00
PoolArea	1460.00	2.76	40.18	0.00	0.00	0.00	0.00
MiscVal	1460.00	43.49	496.12	0.00	0.00	0.00	0.00
MoSold	1460.00	6.32	2.70	1.00	5.00	6.00	8.00
YrSold	1460.00	2007.82	1.33	2006.00	2007.00	2008.00	2009.00
SalePrice	1460.00	180921.20	79442.50	34900.00	129975.00	163000.00	214000.00

Replace all NA by None for categorical columns

```
#df[['Alley', 'BsmtQual', 'BsmtCond', 'BsmtExposure', 'BsmtFinType1', 'BsmtFinType2', 'Firepl
```

Find duplicates

```
df.duplicated().sum()
```

```
0
```

No duplicate data found

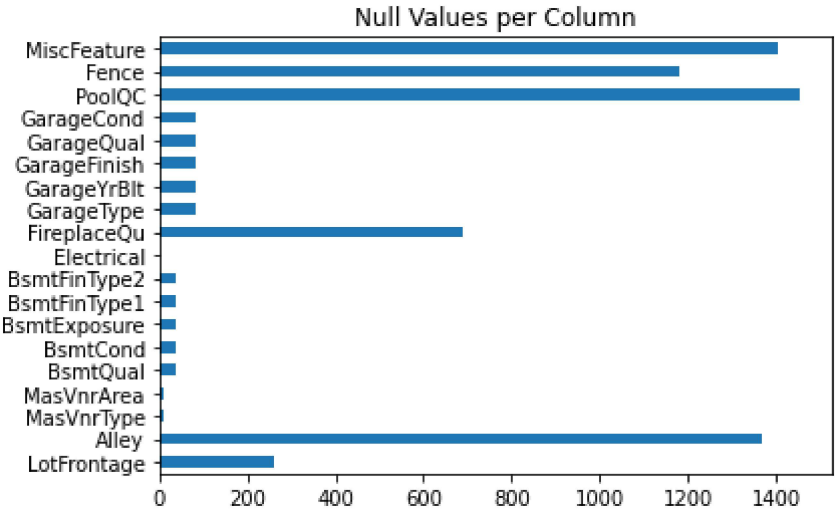
Finding nulls

```
pd.options.display.max_rows=None
df.isnull().mean().sort_values(ascending=False).head(15)
```

```
PoolQC         1.00
MiscFeature    0.96
Alley          0.94
Fence          0.81
FireplaceQu    0.47
LotFrontage    0.18
GarageYrBlt    0.06
GarageCond     0.06
GarageType     0.06
GarageFinish   0.06
GarageQual     0.06
BsmtFinType2   0.03
BsmtExposure   0.03
BsmtQual       0.03
```

```
BsmtCond       0.03
dtype: float64
```

```
columns_with_nulls = df.columns[df.isnull().any()].tolist()
df[columns_with_nulls].isnull().sum().plot(kind='barh', title='Null Values per Column')
plt.show()
```



We can see that the following columns have excess null data - lets drop them (PoolQC,MiscFeature,Alley,Fence)

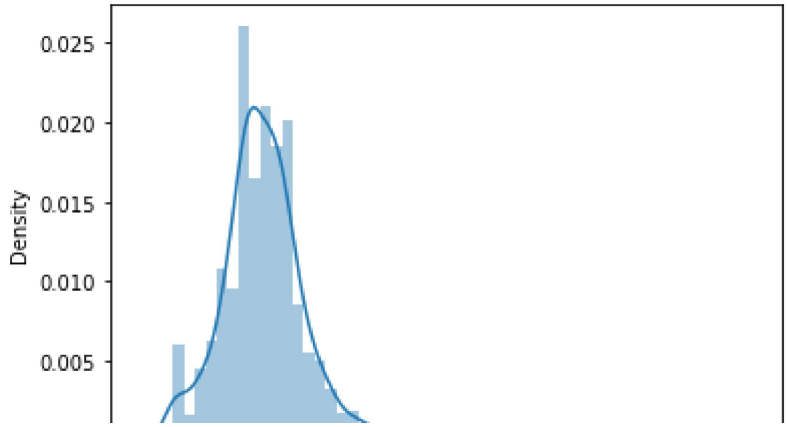
```
df.drop(['PoolQC','MiscFeature','Alley','Fence'],axis=1,inplace=True)
```

Handling missing values for LotFrontage

```
df['LotFrontage'].describe()
```

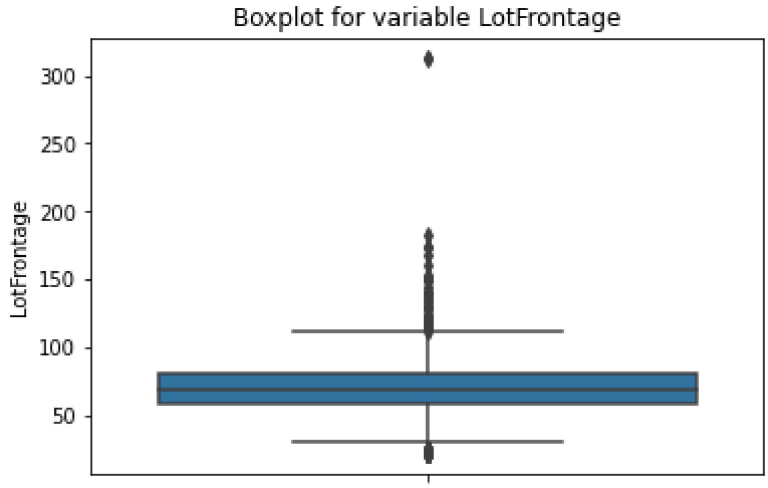
```
count   1201.00
mean      70.05
std       24.28
min       21.00
25%       59.00
50%       69.00
75%       80.00
max      313.00
Name: LotFrontage, dtype: float64
```

```
sns.distplot(df["LotFrontage"])
plt.show()
```



We can see that the distribution is quite normal but with a very long right tail...lets check for outliers to decide if we have to apply mean or median on missing values

```
sns.boxplot(y=df["LotFrontage"])
plt.title(f'Boxplot for variable LotFrontage')
plt.show()
```



Clearly, mean would be a good fit for all missing data as they are few outliers

```
df.loc[df['LotFrontage'].isnull()==True,'LotFrontage']=df["LotFrontage"].mean()
```

Handling missing values for FireplaceQu

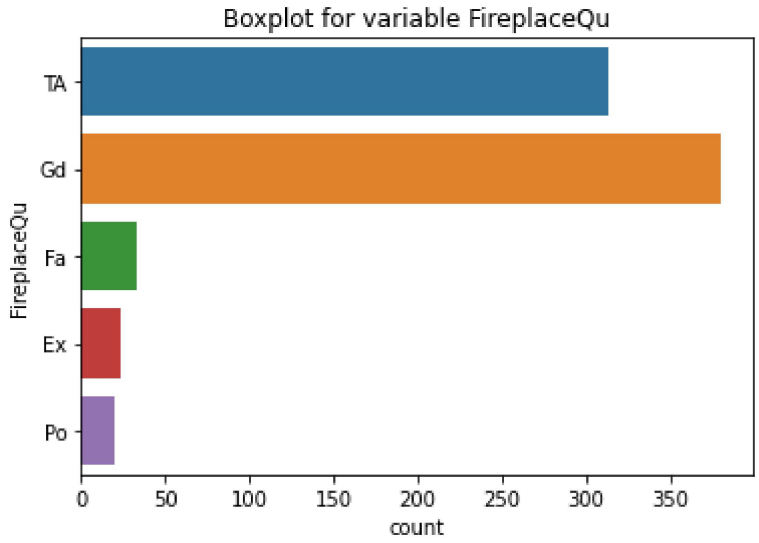
```
df["FireplaceQu"].dtype
```

```
dtype('O')
```

```
df['FireplaceQu'].value_counts()
```

```
Gd    380
TA    313
Fa     33
Ex     24
Po     20
Name: FireplaceQu, dtype: int64
```

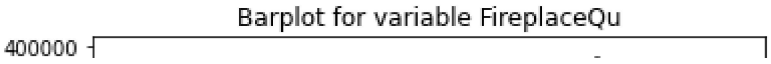
```
sns.countplot(y=df["FireplaceQu"])
plt.title(f'Boxplot for variable FireplaceQu')
plt.show()
```



We could replace nulls with NA indicating "No Fireplace"

```
df.loc[df['FireplaceQu'].isnull()==True,'FireplaceQu']="NA"
```

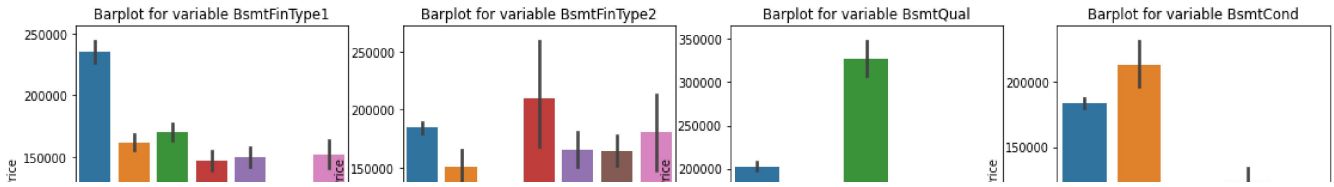
```
sns.barplot(x='FireplaceQu',y='SalePrice',data=df)
plt.title(f'Barplot for variable FireplaceQu')
plt.show()
```



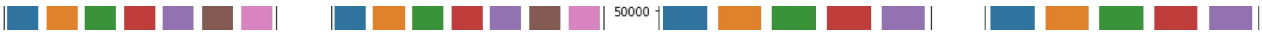
Handling missing values for BsmtFinType1,BsmtFinType2,BsmtQual,BsmtCond,BsmtExposure

Filling "NB" means No Basement for all 5

```
df.loc[df['BsmtFinType2'].isnull()==True,'BsmtFinType2']="NB"
df.loc[df['BsmtFinType1'].isnull()==True,'BsmtFinType1']="NB"

df.loc[df['BsmtQual'].isnull()==True,'BsmtQual']="NB"
df.loc[df['BsmtCond'].isnull()==True,'BsmtCond']="NB"

df.loc[df['BsmtExposure'].isnull()==True,'BsmtExposure']="NB"


plt.figure(figsize=(20, 5), dpi=70)

plt.subplot(1, 4, 1)
sns.barplot(x='BsmtFinType1',y='SalePrice',data=df)
plt.title(f'Barplot for variable BsmtFinType1')

plt.subplot(1, 4, 2)
sns.barplot(x='BsmtFinType2',y='SalePrice',data=df)
plt.title(f'Barplot for variable BsmtFinType2')

plt.subplot(1, 4, 3)
sns.barplot(x='BsmtQual',y='SalePrice',data=df)
plt.title(f'Barplot for variable BsmtQual')

plt.subplot(1, 4, 4)
sns.barplot(x='BsmtCond',y='SalePrice',data=df)
plt.title(f'Barplot for variable BsmtCond')

plt.subplots_adjust(left=0.15)
plt.show()
```

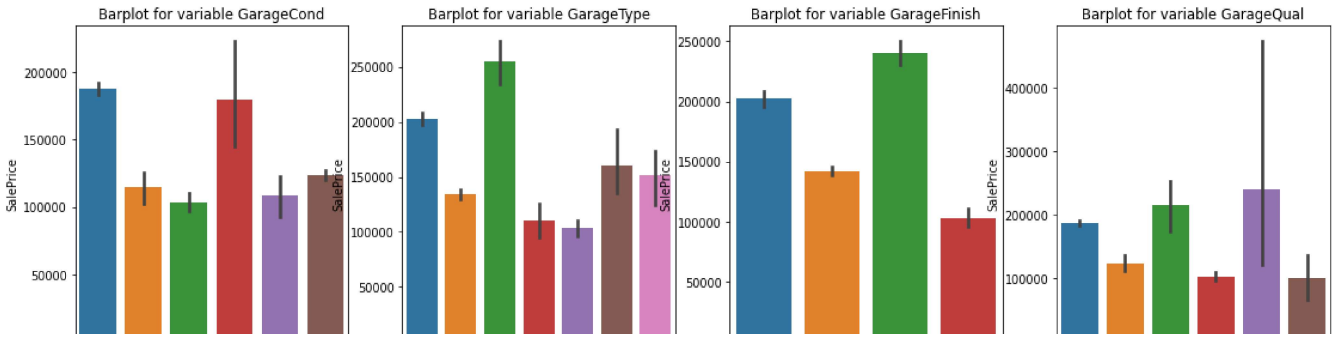


Handling varibales related to garage



Filling "NA" for nulls indicating "No Garage"

```
#No Garage
df.loc[df['GarageCond'].isnull()==True,'GarageCond']="NA"
df.loc[df['GarageType'].isnull()==True,'GarageType']="NA"

df.loc[df['GarageFinish'].isnull()==True,'GarageFinish']="NA"
df.loc[df['GarageQual'].isnull()==True,'GarageQual']="NA"

df.loc[df['GarageFinish'].isnull()==True,'GarageFinish']="NA"
df.loc[df['GarageYrBlt'].isnull()==True,'GarageYrBlt']="NA"



plt.figure(figsize=(20, 5), dpi=70)

plt.subplot(1, 4, 1)
sns.barplot(x='GarageCond',y='SalePrice',data=df)
plt.title(f'Barplot for variable GarageCond')

plt.subplot(1, 4, 2)
sns.barplot(x='GarageType',y='SalePrice',data=df)
plt.title(f'Barplot for variable GarageType')

plt.subplot(1, 4, 3)
sns.barplot(x='GarageFinish',y='SalePrice',data=df)
plt.title(f'Barplot for variable GarageFinish')

plt.subplot(1, 4, 4)
sns.barplot(x='GarageQual',y='SalePrice',data=df)
plt.title(f'Barplot for variable GarageQual')

plt.subplots_adjust(left=0.15)
plt.show()
```



Handling MasVnrType

Fill the null values as None

```
df.loc[df['MasVnrType'].isnull()==True,'MasVnrType']="None"
```

Handling MasVnrArea

```
df[df['MasVnrArea'].isnull()]["MasVnrType"]
```

```
234     None
529     None
650     None
936     None
973     None
977     None
1243    None
1278    None
Name: MasVnrType, dtype: object
```

Since the MasVnrType is "None" that means it is not there we can just replace with 0

```
df.loc[df['MasVnrArea'].isnull()==True,'MasVnrArea']=0.0
```

```
df.isnull().mean().sort_values(ascending=False).head(15)
```

```
Electrical     0.00
Id             0.00
HalfBath       0.00
FireplaceQu    0.00
Fireplaces     0.00
Functional     0.00
TotRmsAbvGrd   0.00
KitchenQual    0.00
KitchenAbvGr   0.00
BedroomAbvGr   0.00
```

```
FullBath       0.00
HeatingQC      0.00
BsmtHalfBath   0.00
BsmtFullBath   0.00
GrLivArea      0.00
dtype: float64
```

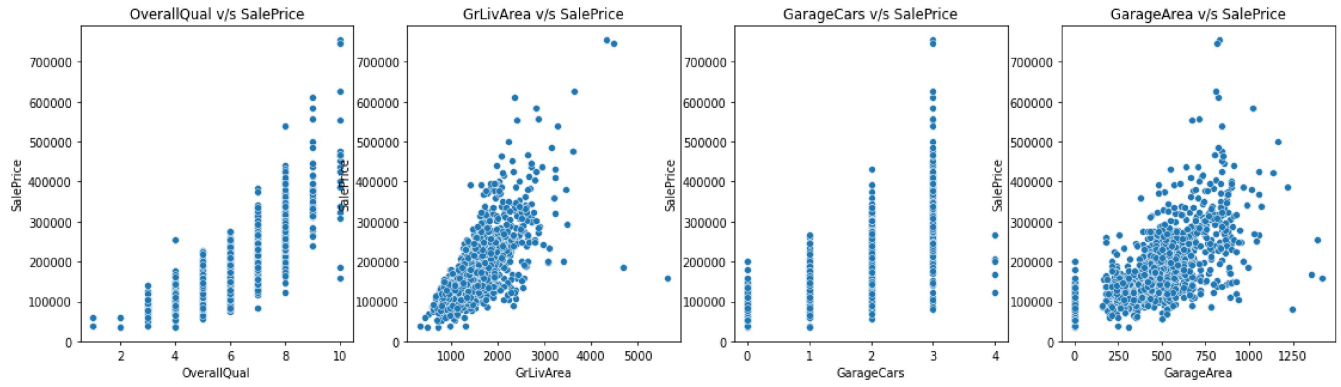
No null values found now, indicating we are good to go

Understanding Corelations

```
df.corr()["SalePrice"].sort_values(ascending=False)
```

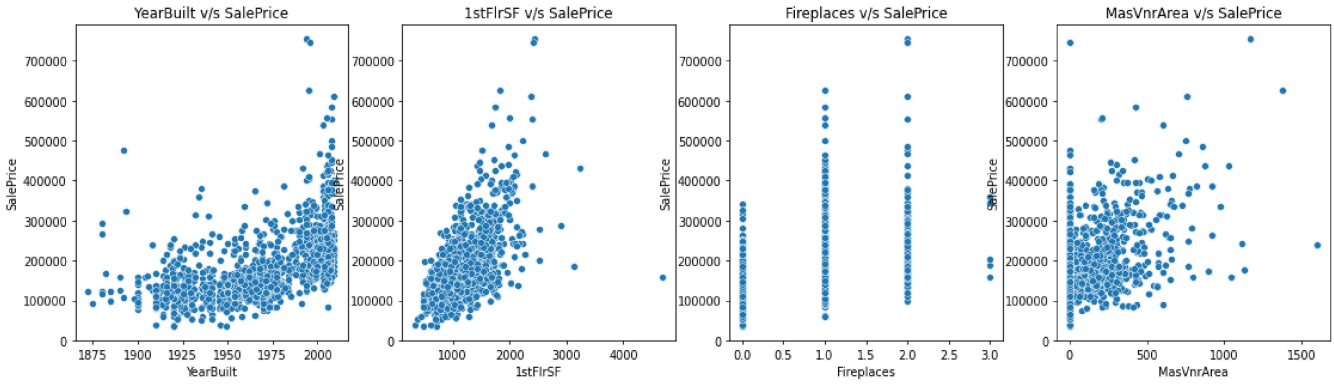
```
SalePrice        1.00
OverallQual      0.79
GrLivArea        0.71
GarageCars       0.64
GarageArea       0.62
TotalBsmtSF      0.61
1stFlrSF         0.61
FullBath         0.56
TotRmsAbvGrd     0.53
YearBuilt        0.52
YearRemodAdd     0.51
MasVnrArea       0.47
Fireplaces       0.47
BsmtFinSF1       0.39
LotFrontage      0.33
WoodDeckSF       0.32
2ndFlrSF         0.32
OpenPorchSF      0.32
HalfBath         0.28
LotArea          0.26
BsmtFullBath     0.23
BsmtUnfSF        0.21
BedroomAbvGr     0.17
ScreenPorch      0.11
PoolArea         0.09
MoSold           0.05
3SsnPorch        0.04
BsmtFinSF2      -0.01
BsmtHalfBath    -0.02
MiscVal         -0.02
Id              -0.02
LowQualFinSF    -0.03
YrSold          -0.03
OverallCond     -0.08
MSSubClass      -0.08
EnclosedPorch   -0.13
KitchenAbvGr    -0.14
Name: SalePrice, dtype: float64
```

```
### Scatter plots to understand relations better


plt.figure(figsize=(20, 5), dpi=70)

plt.subplot(1, 4, 1)
sns.scatterplot(data=df,x="OverallQual", y="SalePrice")
plt.title(f'OverallQual v/s SalePrice')

plt.subplot(1, 4, 2)
sns.scatterplot(data=df,x="GrLivArea", y="SalePrice")
plt.title(f'GrLivArea v/s SalePrice')

plt.subplot(1, 4, 3)
sns.scatterplot(data=df,x="GarageCars", y="SalePrice")
plt.title(f'GarageCars v/s SalePrice')

plt.subplot(1, 4, 4)
sns.scatterplot(data=df,x="GarageArea", y="SalePrice")
plt.title(f'GarageArea v/s SalePrice')

plt.subplots_adjust(left=0.15)
plt.show()
```



1. As the quality increases price increases.
2. Living area between 10,000 to 25,000 is preferred more.
3. More the cars parking space , more the price.
4. More the garage area, more the price.
5. Two cars parking area is highest preferred.
6. 250-700 area for car parking is preferred.

```
plt.figure(figsize=(20, 5), dpi=70)

plt.subplot(1, 4, 1)
sns.scatterplot(data=df,x="YearBuilt", y="SalePrice")
plt.title(f'YearBuilt v/s SalePrice')

plt.subplot(1, 4, 2)
sns.scatterplot(data=df,x="1stFlrSF", y="SalePrice")
plt.title(f'1stFlrSF v/s SalePrice')

plt.subplot(1, 4, 3)
sns.scatterplot(data=df,x="Fireplaces", y="SalePrice")
plt.title(f'Fireplaces v/s SalePrice')

plt.subplot(1, 4, 4)
sns.scatterplot(data=df,x="MasVnrArea", y="SalePrice")
plt.title(f'MasVnrArea v/s SalePrice')

plt.subplots_adjust(left=0.15)
plt.show()
```



1. Yearbuilt really does not fluctuate the price a lot
2. First floor Square feet is preferred between 1000-2000

```
df["SalePrice"].describe()
```

```
count     1460.00
mean    180921.20
std      79442.50
min      34900.00
25%     129975.00
50%     163000.00
75%     214000.00
```

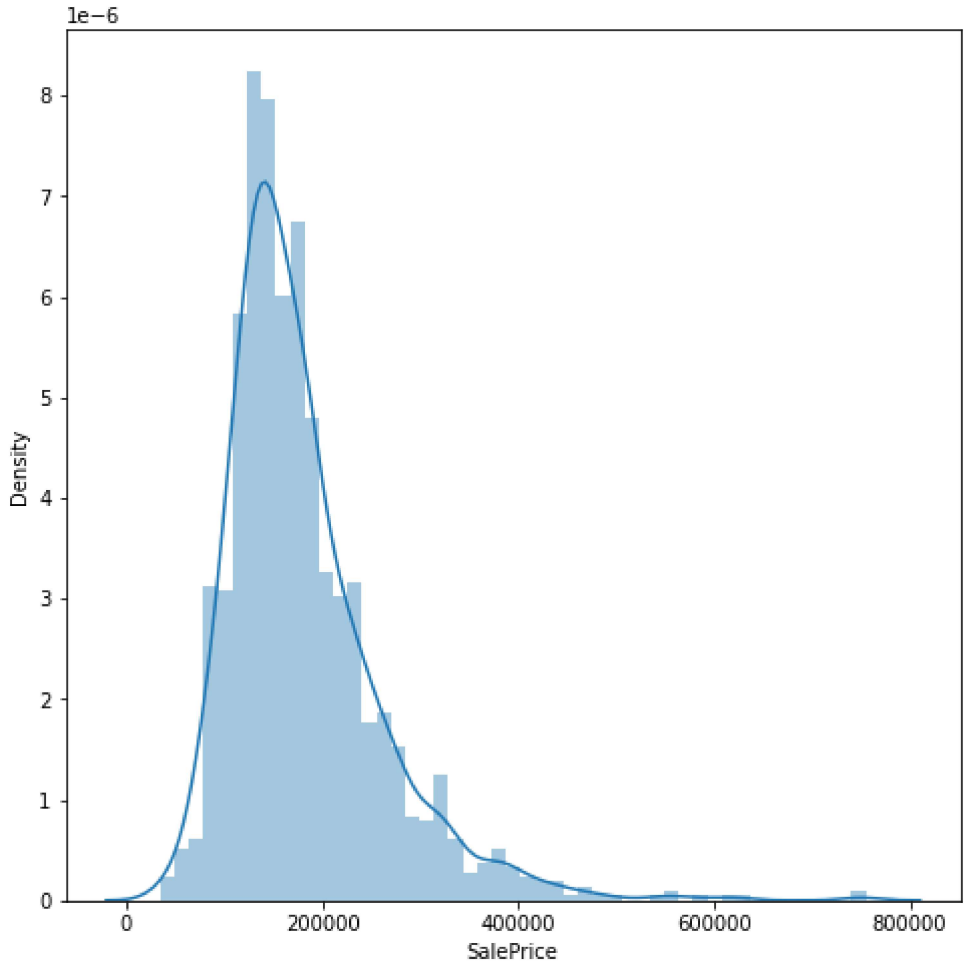
```
max     755000.00
Name: SalePrice, dtype: float64
```

```
corr = df.corr()
# Generate a mask for the upper triangle
mask = np.triu(np.ones_like(corr, dtype=bool))
# Set up the matplotlib figure
f, ax = plt.subplots(figsize=(20, 20))
# Draw the heatmap with the mask and correct aspect ratio
sns.heatmap(corr, mask=mask, annot=True,cmap='viridis', vmax=.3, center=0,
            square=True, linewidths=.5)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fcd2ac06250>
```

Understand saleprice more


```
plt.figure(figsize=[8,8])
sns.distplot(df['SalePrice']);
```

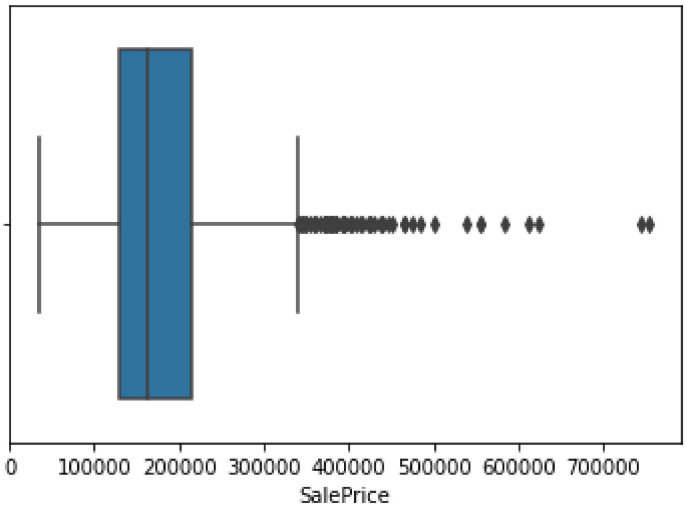


The data looks right skewed. We will need to normalize it

```
df['SalePrice'].describe()
```

```
count     1460.00
mean    180921.20
std      79442.50
min      34900.00
25%     129975.00
50%     163000.00
75%     214000.00
max     755000.00
Name: SalePrice, dtype: float64
```

```
sns.boxplot(df['SalePrice'], orient='h')
plt.show()
```




```
df['SalePrice'].skew()
```

```
1.8828757597682129
```

The skewness is greater than 1, so the target variable is highly skewed.

```
df['SalePrice'].kurt()
```

```
6.536281860064529
```

The kurtosis is greater than 1, so the distribution of target variable is highly peaked.

So, we will log transform our target variable

```
df['SalePrice'] = np.log(df['SalePrice'])
```

Check Skew, Kurtosis now

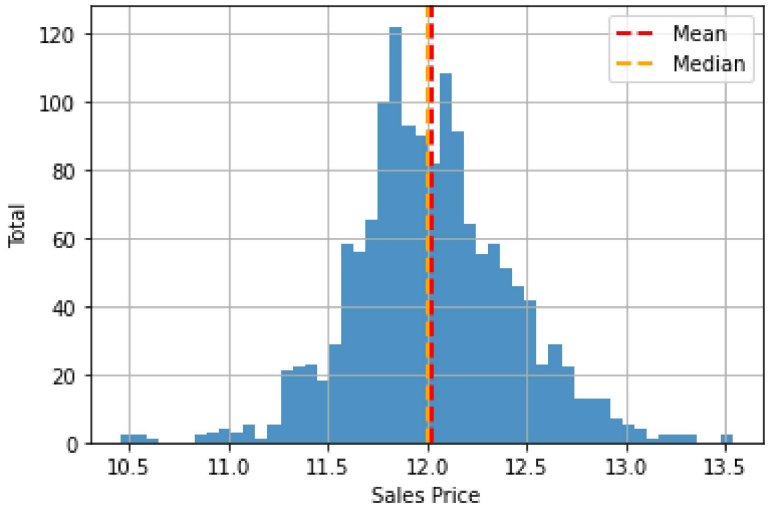
```
print(df['SalePrice'].skew())
print(df['SalePrice'].kurt())
```

```
0.12133506220520406
0.8095319958036296
```

Reduced to one below now

Clearly there seems to be outliers, let us get rid of them

```
df['SalePrice'].hist(bins=50, alpha=0.8)
plt.axvline(df['SalePrice'].mean(), color='r', linestyle='dashed', linewidth=2, label='Mean')
plt.axvline(df['SalePrice'].median(), color='orange', linestyle='dashed', linewidth=2, label=
plt.xlabel('Sales Price')
plt.ylabel('Total')
plt.legend()
plt.figure(figsize=(15,15))
plt.show()
```




```
<Figure size 1080x1080 with 0 Axes>
```

```
def removeoutliers(data, m=1.5):
    return data[abs(data - np.mean(data)) < m * np.std(data)]
```

```
toremove = removeoutliers(df['SalePrice']).tolist()
toremove[0:20]
```

```
[12.247694320220994,
 12.109010932687042,
 12.31716669303576,
 11.84939770159144,
 12.429216196844383,
 11.870599909242044,
```

```
        12.206072645530174,
        11.77452020265869,
        11.678439903447801,
        11.771436160121729,
        11.877568578558138,
        12.540757571577291,
        11.964001084330445,
        11.790557201568507,
        11.911701584927597,
        11.976659481202368,
        11.842229212112828,
        11.845102777308561,
        12.345834587905333,
        11.77452020265869]


toremoveresults = df['SalePrice'].isin(toremove)[0:20].tolist()
toremoveresults
```

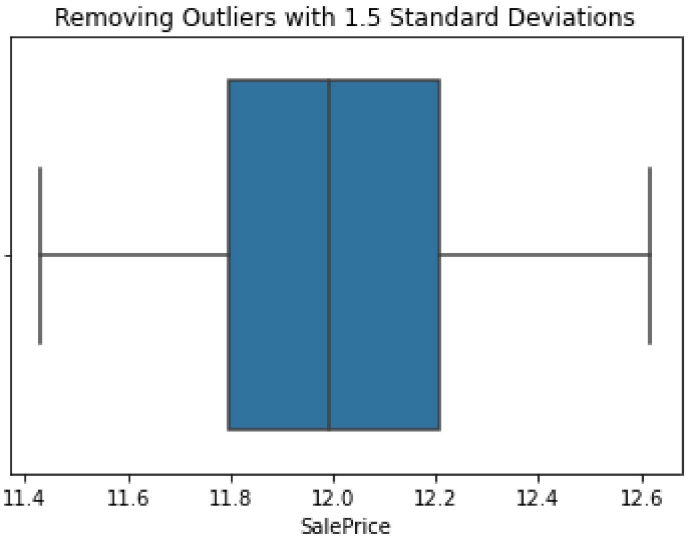
```
     [True,
      True,
      True,
      True,
      True,
      True,
      False,
      True,
      True,
      True,
      True,
      False,
      True,
      True,
      True,
      True,
      True,
      False,
      True,
      True]


outliers = dict(zip(toremove, toremoveresults))
outliers
```

```
     {12.247694320220994: True,
      12.109010932687042: True,
      12.31716669303576: True,
      11.84939770159144: True,
      12.429216196844383: True,
      11.870599909242044: True,
      12.206072645530174: False,
      11.77452020265869: True,
      11.678439903447801: True,
      11.771436160121729: True,
      11.877568578558138: True,
      12.540757571577291: False,
```

```
        11.964001084330445: True,
        11.790557201568507: True,
        11.911701584927597: True,
        11.976659481202368: True,
        11.842229212112828: True,
        11.845102777308561: False,
        12.345834587905333: True}
```

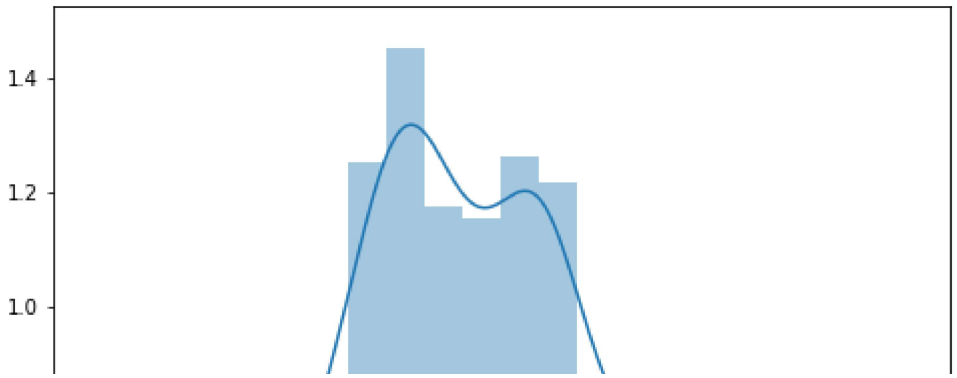
```
df = df[~df['SalePrice'].isin(toremove) == False]
```

```
sns.boxplot(df['SalePrice'], orient='h').set_title('Removing Outliers with 1.5 Standard Devia
plt.show()
```



The data of sales price now looks good.

```
plt.figure(figsize=[8,8])
sns.distplot(df['SalePrice']);
```



SalesPrice now seems to be have normal distribution

Analyze numerical columns

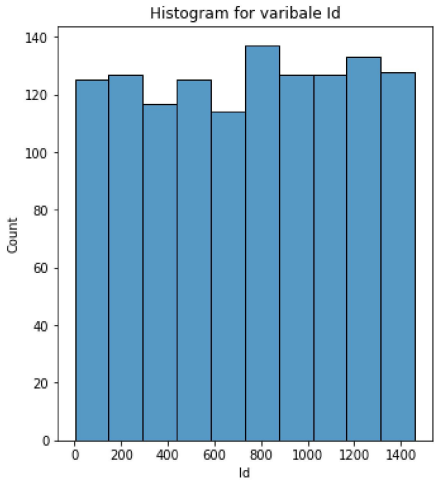
```
# Function to plot histogram, probplot and boxplot for numerical analysis
def plotNumericalDiagnostics(df,var_col, no_of_bins):

  print(f"Total NA enties {df[var_col].isnull().mean()} \n")
  print(df[var_col].describe().to_frame().T,'\n')
  plt.figure(figsize=(18,6))

  plt.subplot(1,3,1)
  sns.histplot(df[var_col], bins=no_of_bins)
  plt.title(f'Histogram for varibale {var_col}')

  plt.subplot(1, 3, 2)
  sns.boxplot(y=df[var_col])
  plt.title(f'Boxplot for variable {var_col}')

  plt.subplot(1, 3, 3)
  stats.probplot(df[var_col], dist="norm", plot=plt)
  plt.title(f'Probplot for varibale {var_col}')

  plt.show()
```

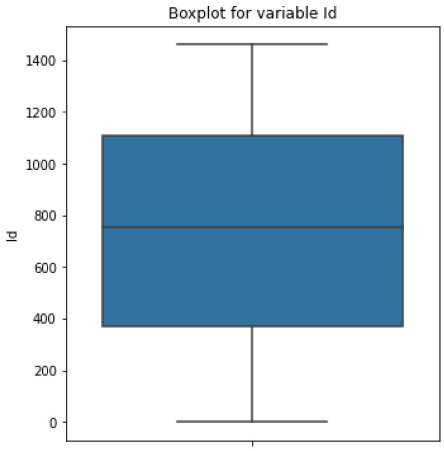
Get all numeric columns

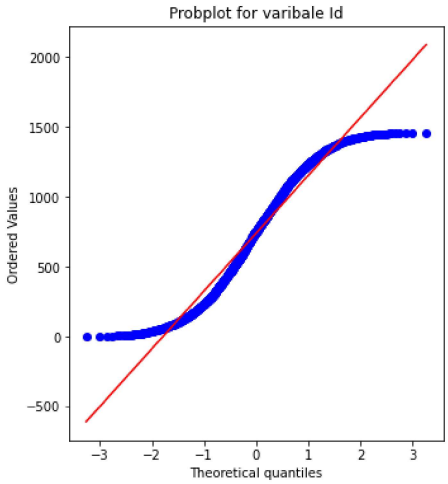
```
numeric_columns = df.select_dtypes(exclude=[object]).columns.tolist()
#numeric_columns


for col in numeric_columns:
  plotNumericalDiagnostics(df,col,10)
```

```
Total NA enties 0.0
```

	count	mean	std	min	25%	50%	75%	max
Id	1260.00	739.87	422.54	1.00	371.75	752.50	1110.25	1460.00

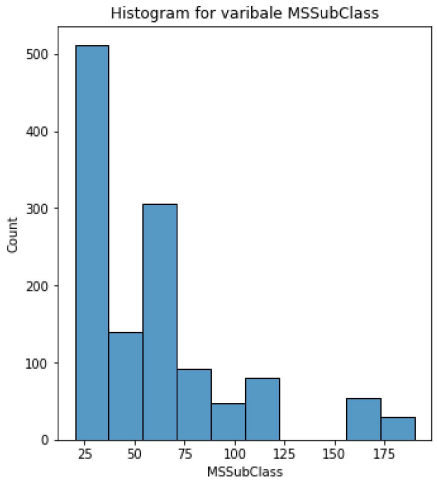


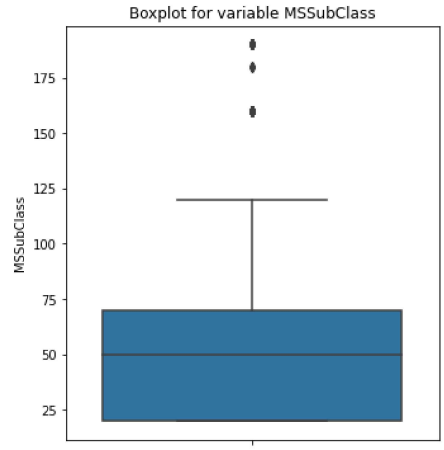


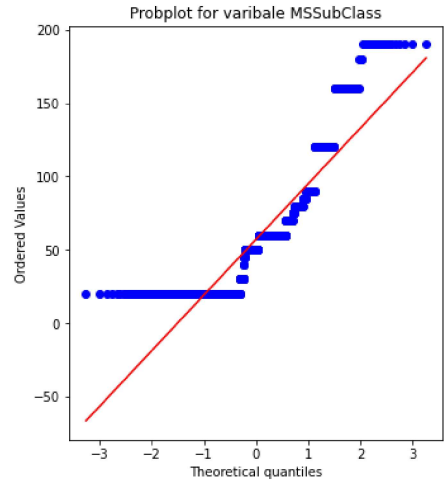



```
Total NA enties 0.0
```

	count	mean	std	min	25%	50%	75%	max
MSSubClass	1260.00	57.15	41.93	20.00	20.00	50.00	70.00	190.00

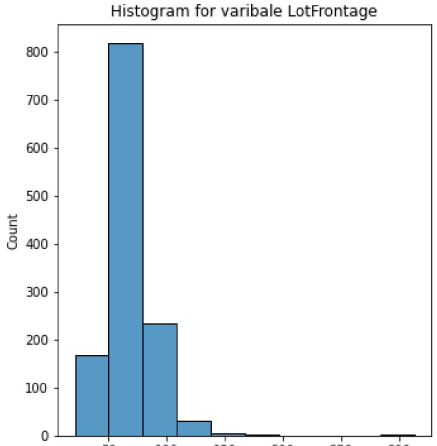


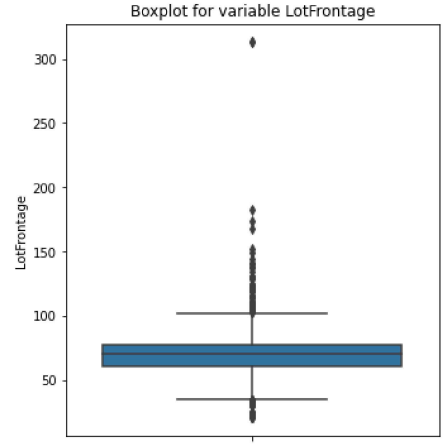


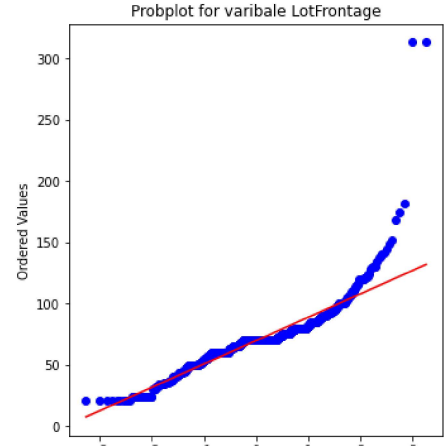


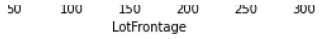

```
Total NA enties 0.0
```

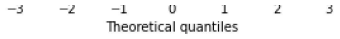
	count	mean	std	min	25%	50%	75%	max
LotFrontage	1260.00	69.57	21.12	21.00	60.00	70.05	77.00	313.00





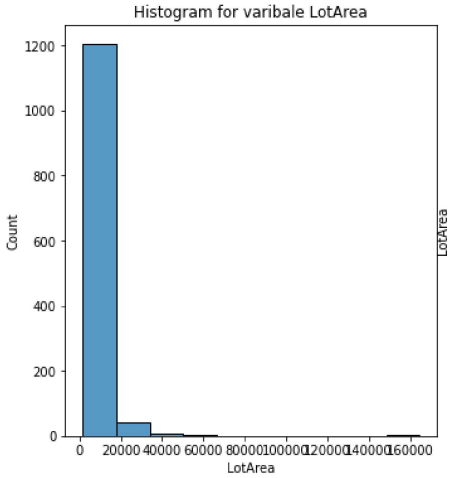


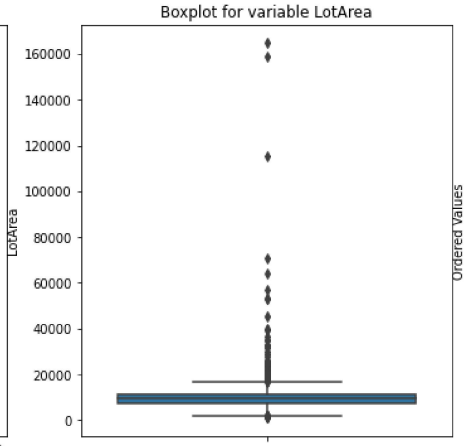


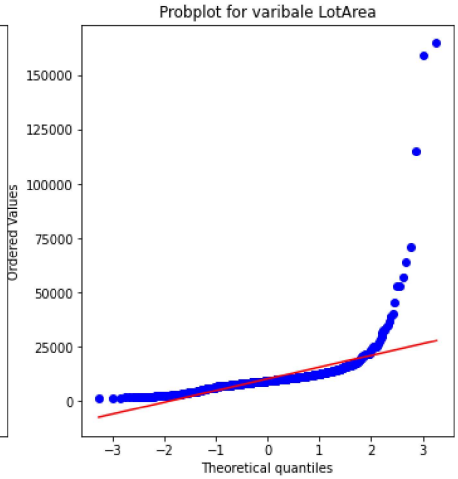



```
Total NA enties 0.0
```

```
           count     mean     std     min     25%     50%      75%       max
LotArea 1260.00 10296.23 8644.13 1300.00 7560.00 9356.50 11292.75 164660.00
```

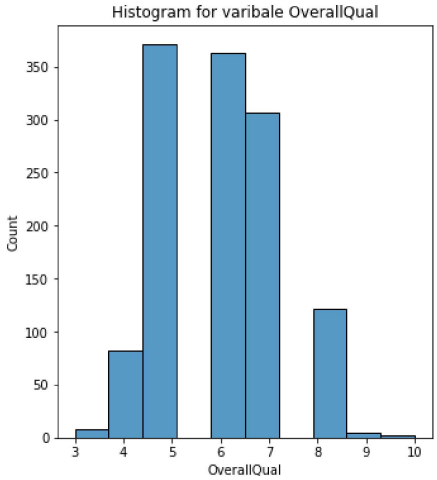


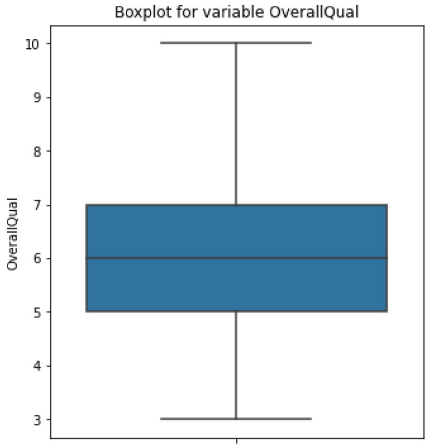


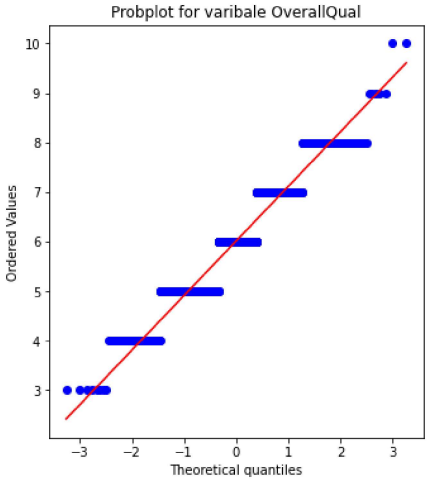



```
Total NA enties 0.0
```

```
             count  mean  std  min  25%  50%  75%   max
OverallQual 1260.00  6.01 1.14 3.00 5.00 6.00 7.00 10.00
```

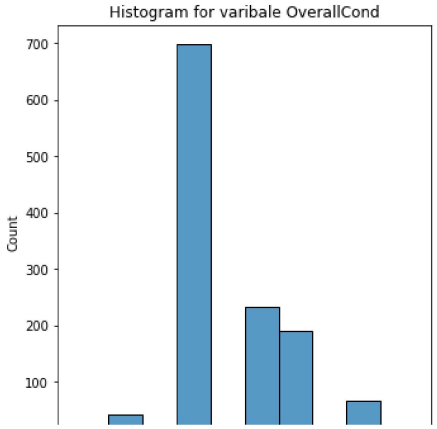


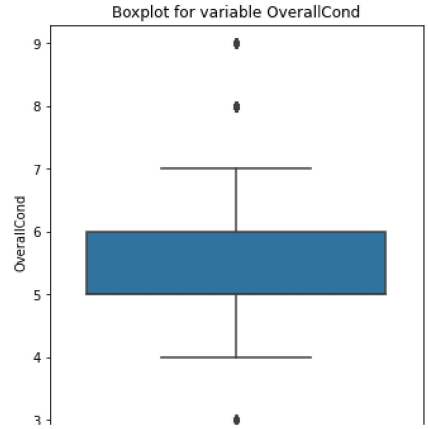


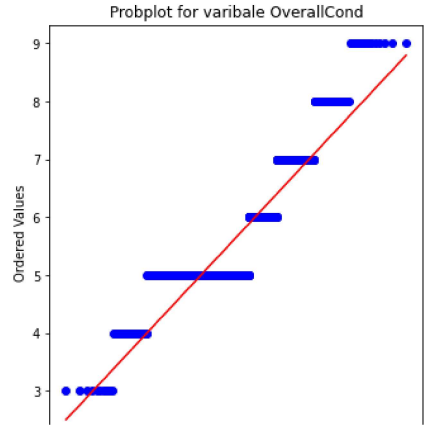


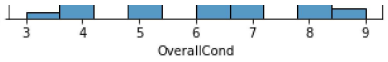

```
Total NA enties 0.0
```

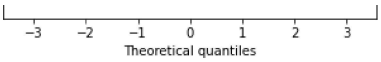
```
             count  mean  std  min  25%  50%  75%  max
OverallCond 1260.00  5.65 1.07 3.00 5.00 5.00 6.00 9.00
```





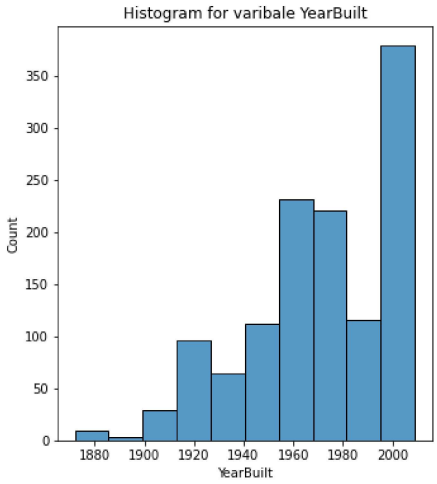


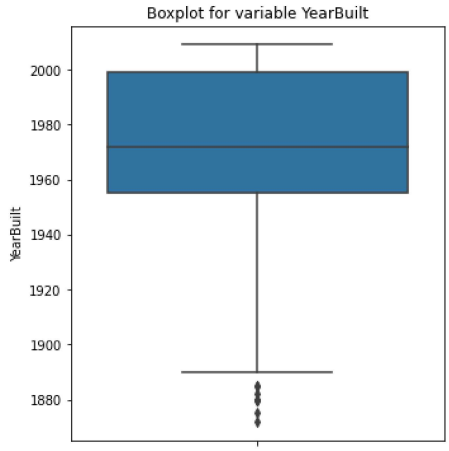


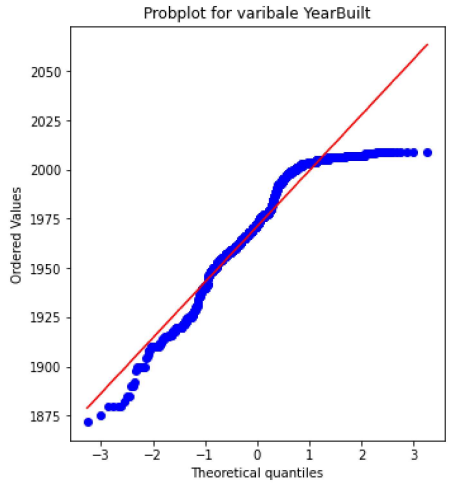



```
Total NA enties 0.0
```

	count	mean	std	min	25%	50%	75%	max
YearBuilt	1260.00	1971.16	29.22	1872.00	1955.00	1972.00	1999.00	2009.00

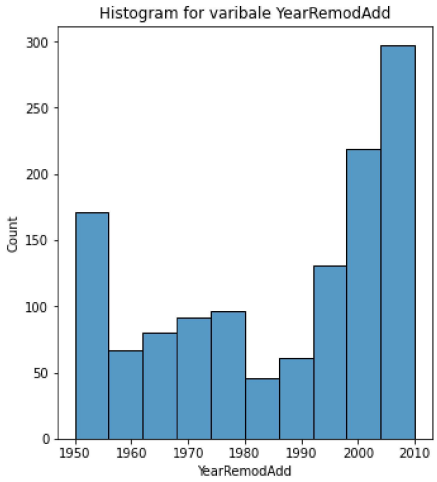


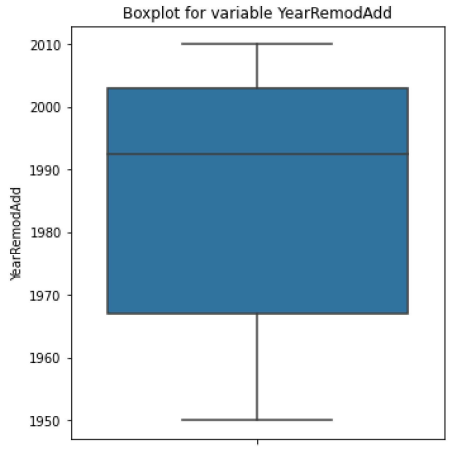


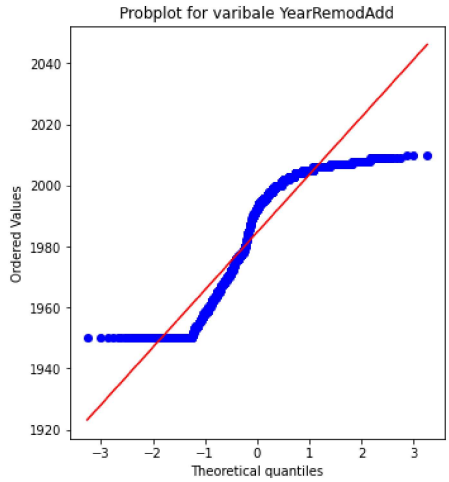



```
Total NA enties 0.0
```

	count	mean	std	min	25%	50%	75%	max
YearRemodAdd	1260.00	1984.72	20.11	1950.00	1967.00	1992.50	2003.00	2010.00

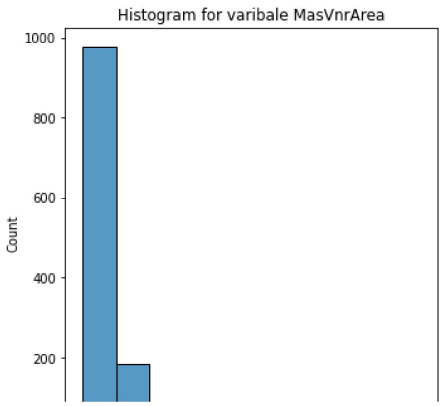


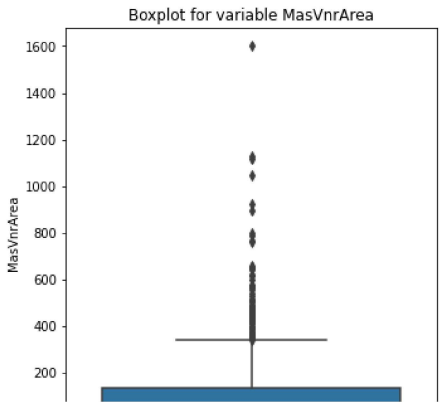


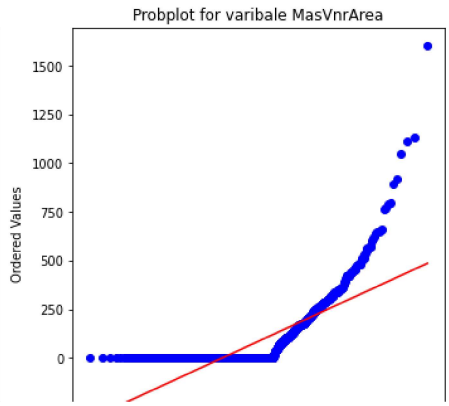


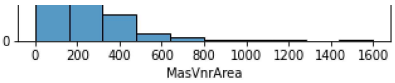

```
Total NA enties 0.0
```

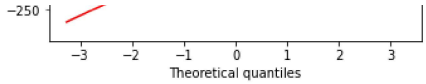
	count	mean	std	min	25%	50%	75%	max
MasVnrArea	1260.00	86.80	155.02	0.00	0.00	0.00	136.00	1600.00





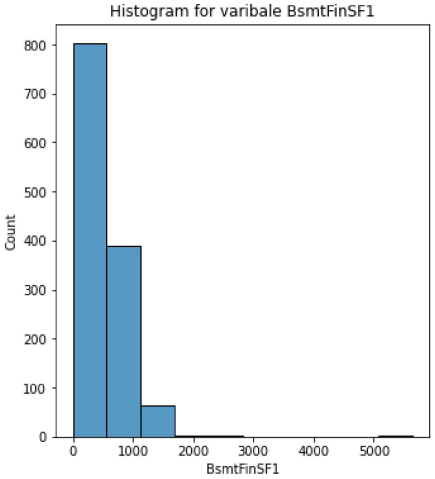


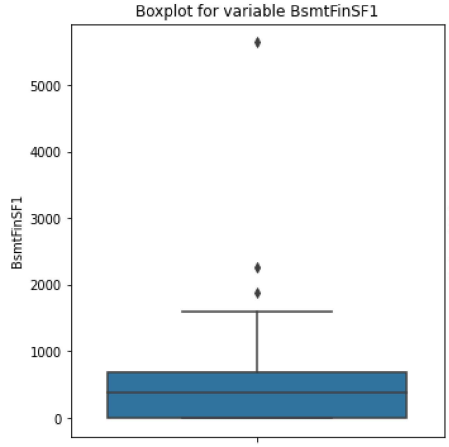


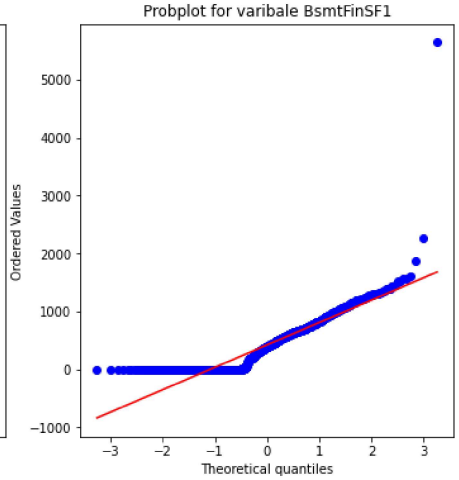



```
Total NA enties 0.0
```

	count	mean	std	min	25%	50%	75%	max
BsmtFinSF1	1260.00	420.90	421.39	0.00	0.00	385.50	686.00	5644.00

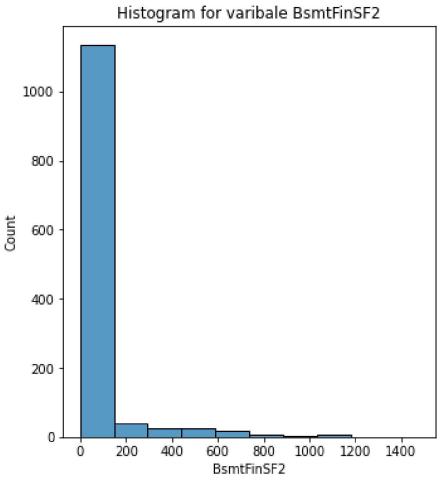


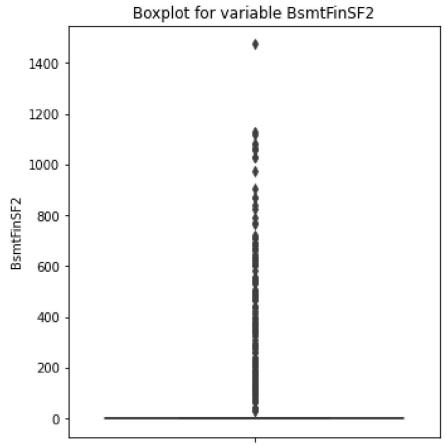


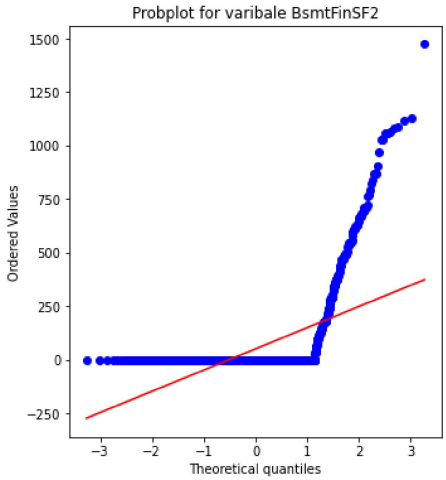



```
Total NA enties 0.0
```

	count	mean	std	min	25%	50%	75%	max
BsmtFinSF2	1260.00	50.92	168.98	0.00	0.00	0.00	0.00	1474.00

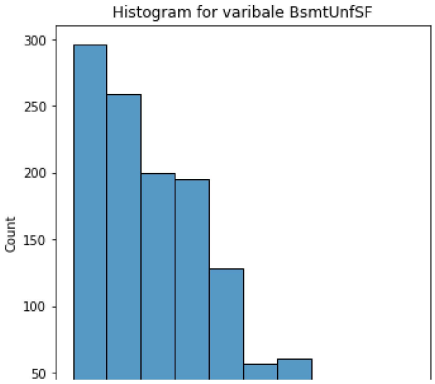


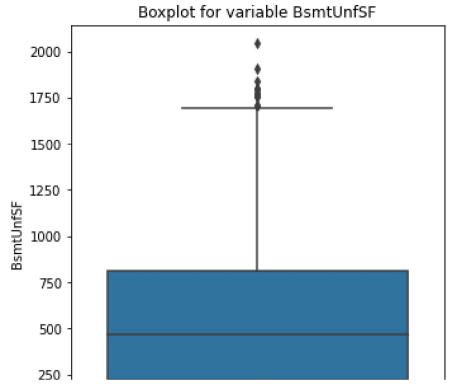


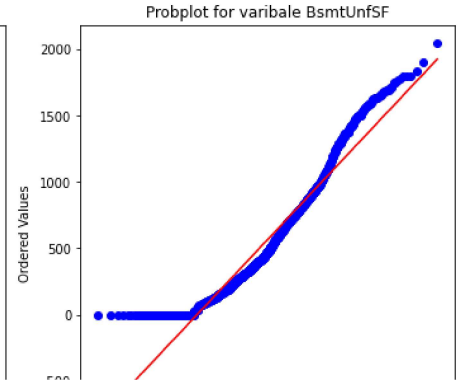


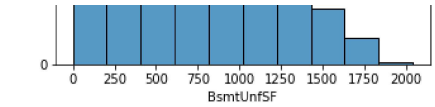

```
Total NA enties 0.0
```

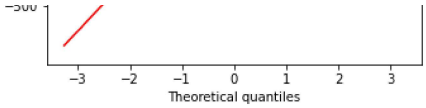
	count	mean	std	min	25%	50%	75%	max
BsmtUnfSF	1260.00	563.79	429.95	0.00	222.50	469.00	815.25	2042.00





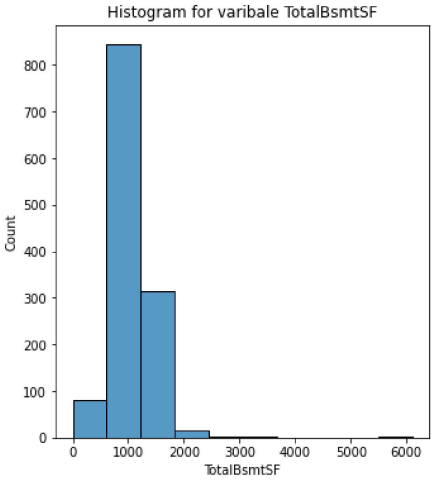


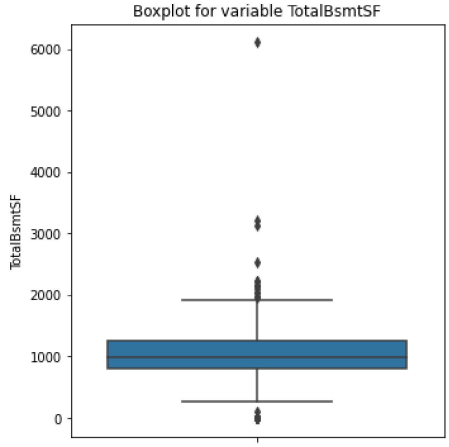


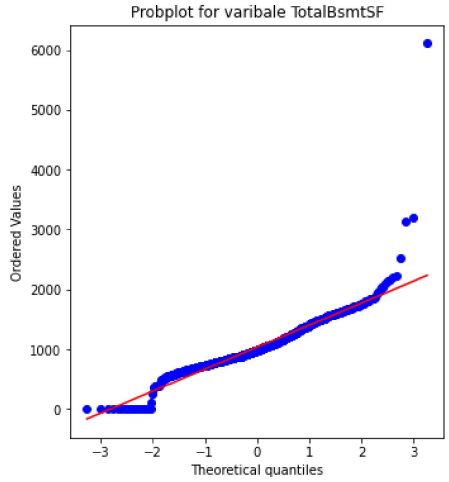



```
Total NA enties 0.0
```

	count	mean	std	min	25%	50%	75%	max
TotalBsmtSF	1260.00	1035.62	389.70	0.00	806.00	980.00	1248.00	6110.00

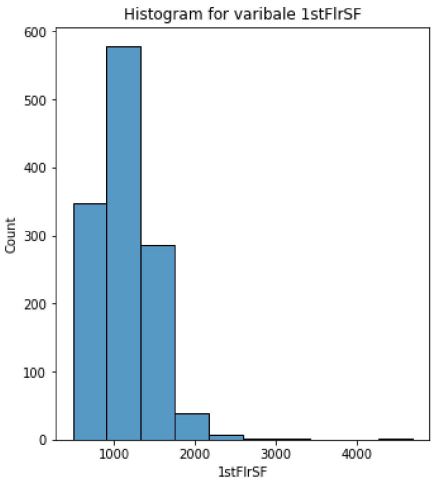


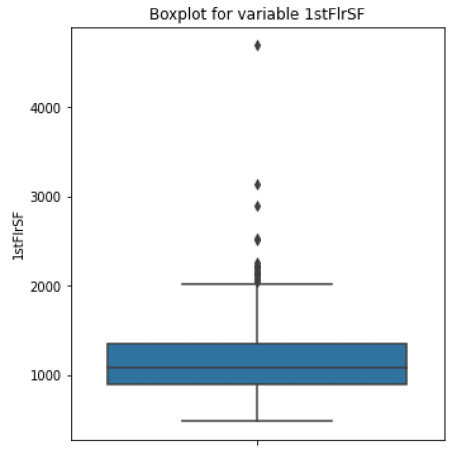


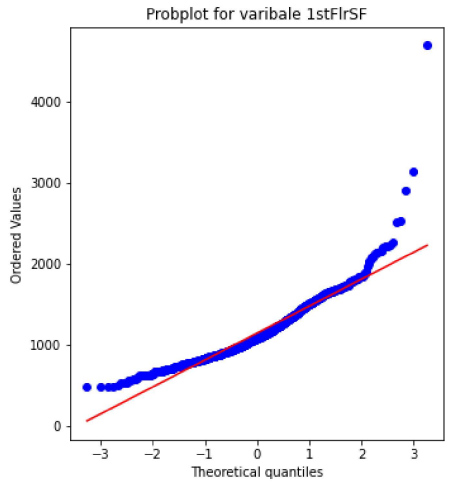



```
Total NA enties 0.0
```

	count	mean	std	min	25%	50%	75%	max
1stFlrSF	1260.00	1139.77	347.70	483.00	888.00	1073.50	1342.50	4692.00

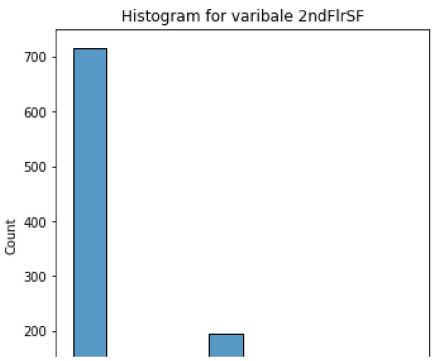


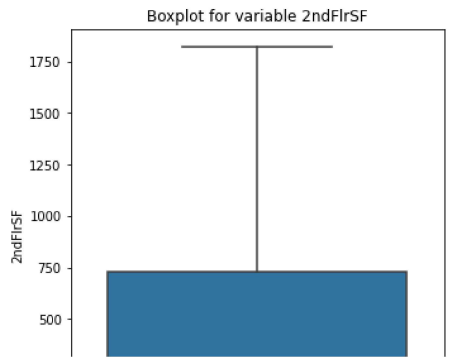


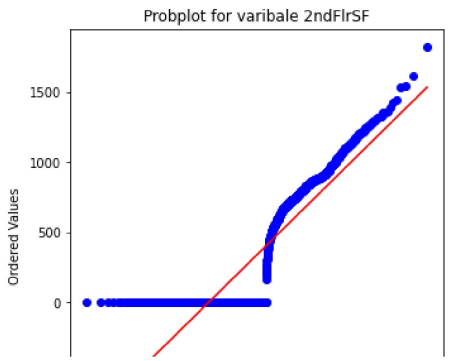


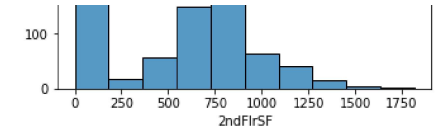

```
Total NA enties 0.0
```

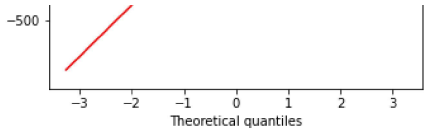
	count	mean	std	min	25%	50%	75%	max
2ndFlrSF	1260.00	338.64	417.90	0.00	0.00	0.00	728.00	1818.00





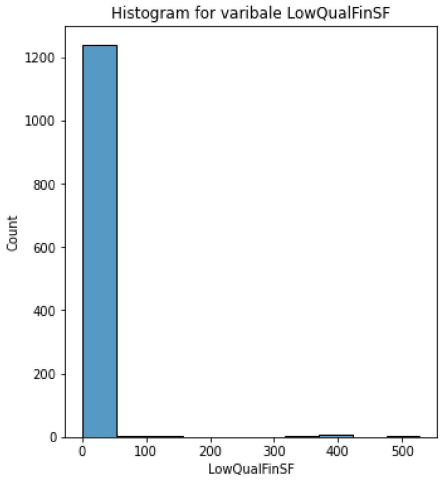


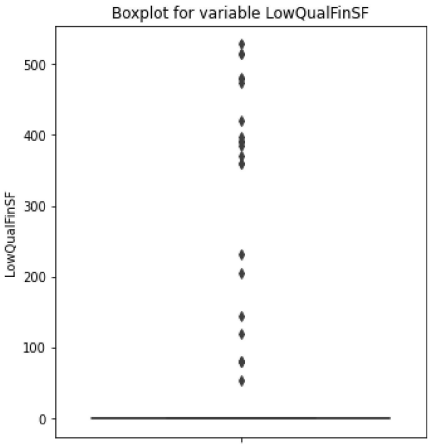


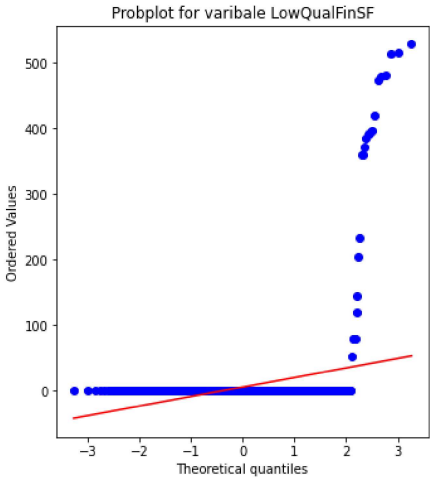



```
Total NA enties 0.0
```

```
               count  mean   std  min  25%  50%  75%    max
LowQualFinSF 1260.00  5.60 47.08 0.00 0.00 0.00 0.00 528.00
```

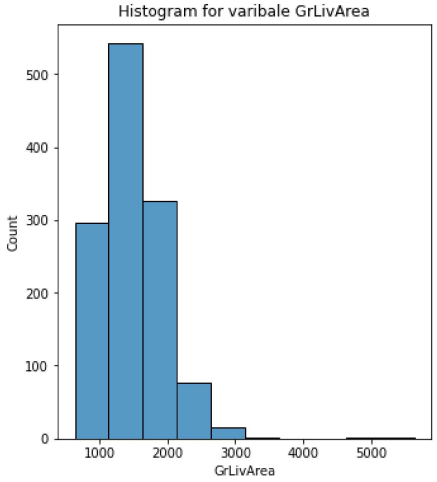


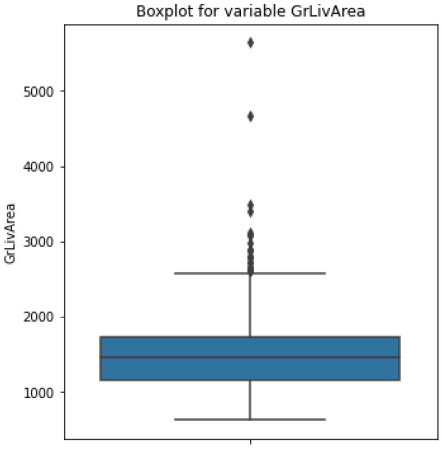


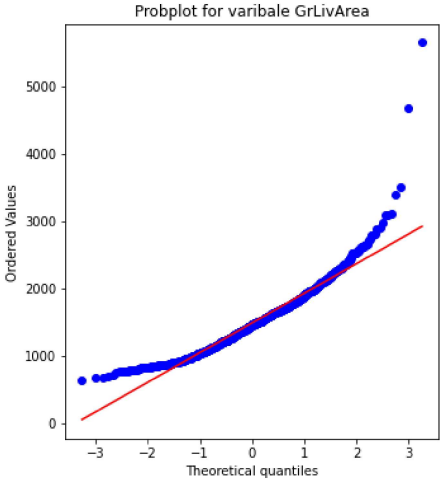



```
Total NA enties 0.0
```

```
             count    mean    std    min     25%     50%     75%     max
GrLivArea 1260.00 1484.01 456.20 630.00 1148.00 1454.50 1721.00 5642.00
```

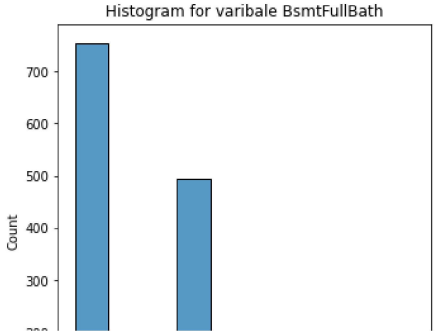


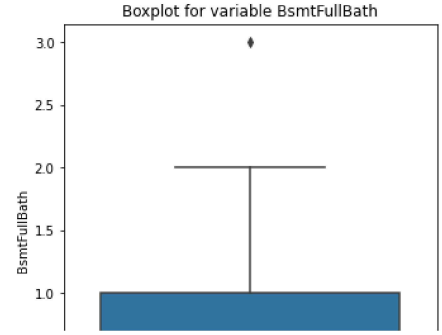


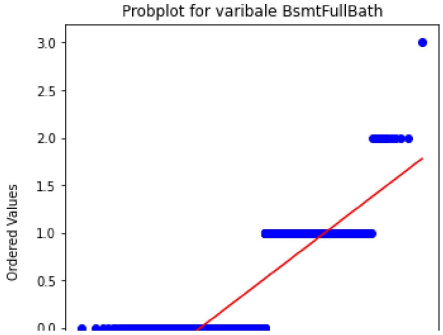


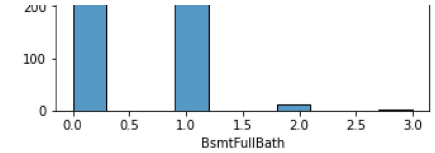

```
Total NA enties 0.0
```

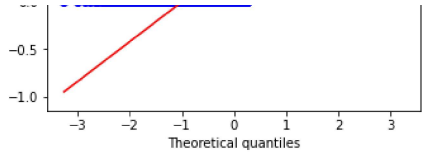
```
               count  mean  std  min  25%  50%  75%  max
BsmtFullBath 1260.00  0.41 0.52 0.00 0.00 0.00 1.00 3.00
```





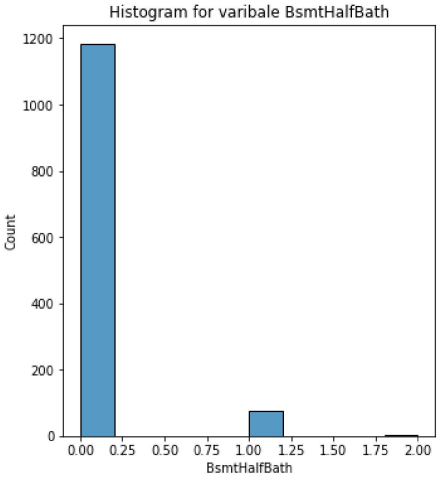


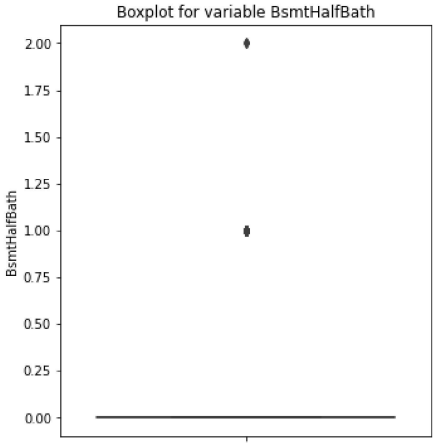


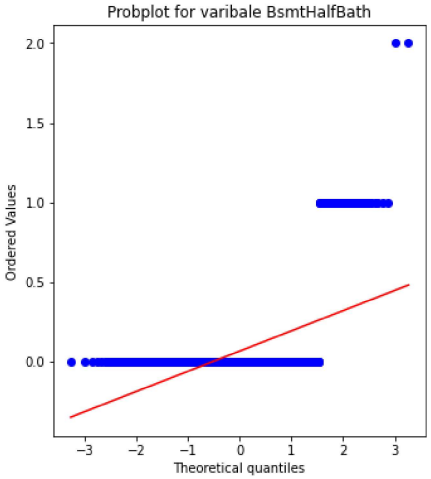



```
Total NA enties 0.0
```

```
              count  mean  std  min  25%  50%  75%  max
BsmtHalfBath 1260.00  0.06 0.25 0.00 0.00 0.00 0.00 2.00
```

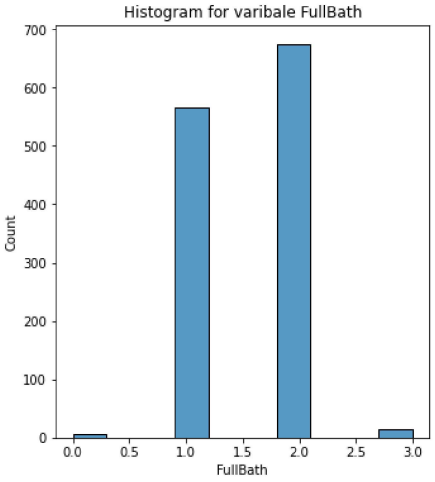


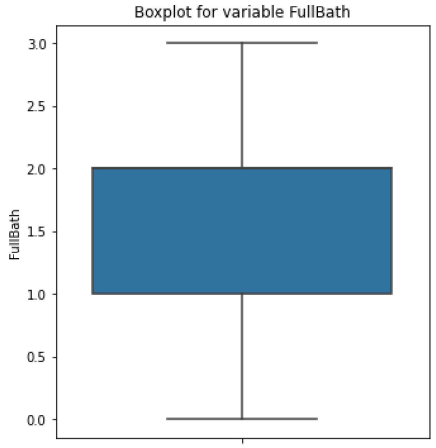


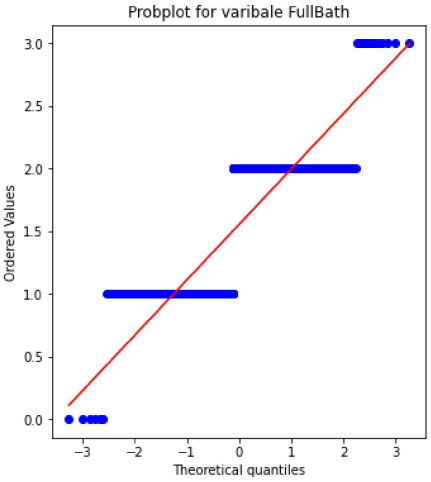



```
Total NA enties 0.0
```

```
            count  mean  std  min  25%  50%  75%  max
FullBath 1260.00  1.55 0.53 0.00 1.00 2.00 2.00 3.00
```

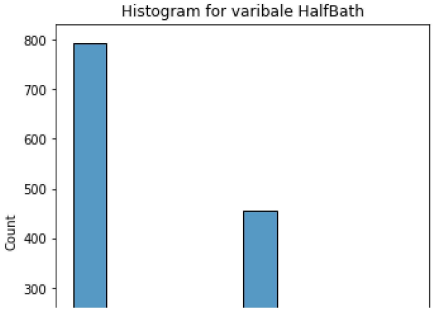


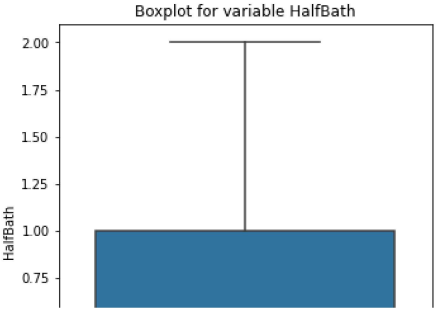


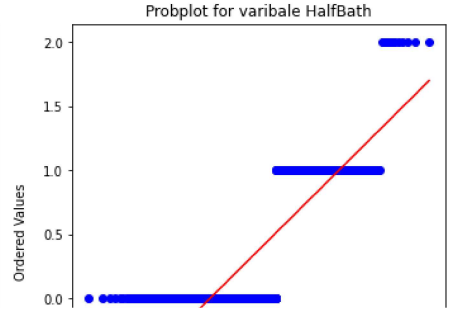


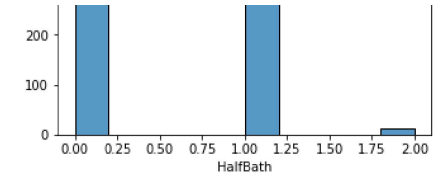

```
Total NA enties 0.0
```

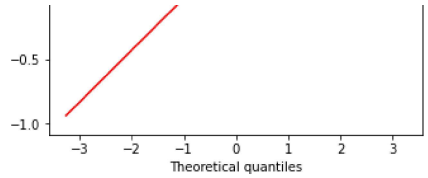
```
            count  mean  std  min  25%  50%  75%  max
HalfBath 1260.00  0.38 0.51 0.00 0.00 0.00 1.00 2.00
```





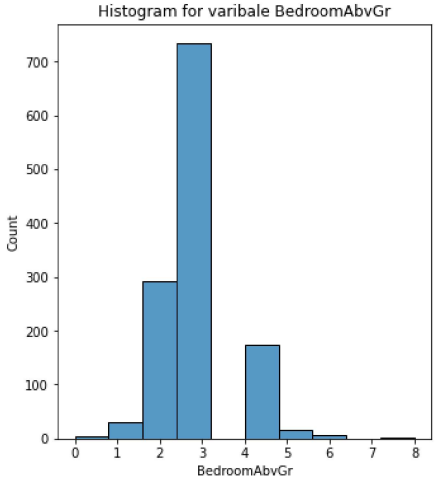


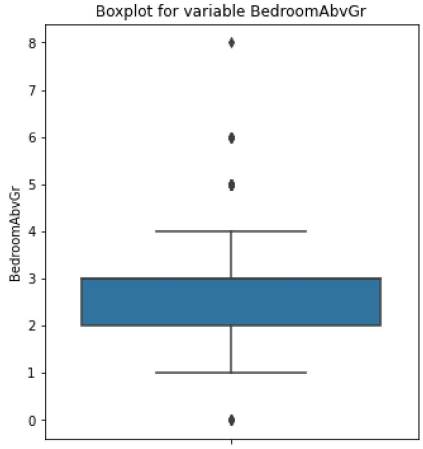


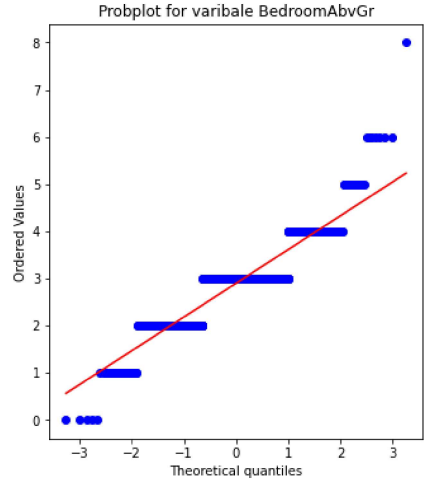



```
Total NA enties 0.0
```

	count	mean	std	min	25%	50%	75%	max
BedroomAbvGr	1260.00	2.89	0.79	0.00	2.00	3.00	3.00	8.00

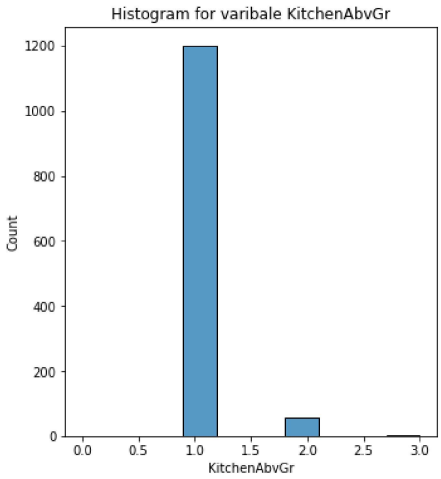


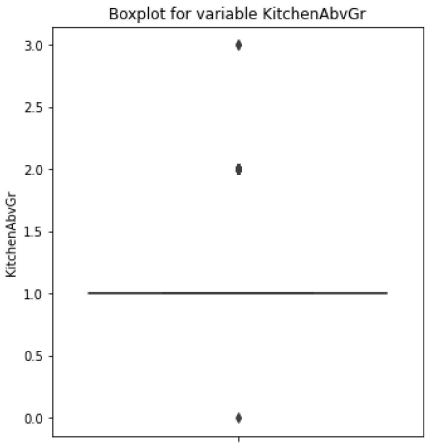


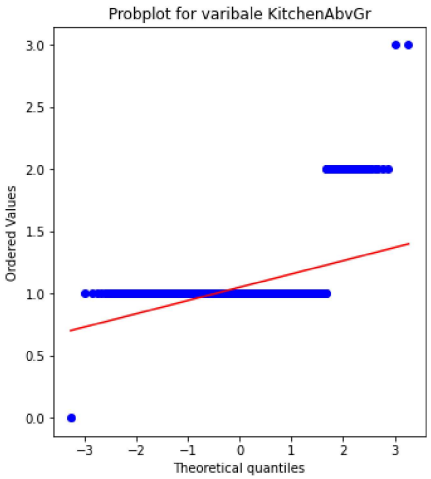



```
Total NA enties 0.0
```

	count	mean	std	min	25%	50%	75%	max
KitchenAbvGr	1260.00	1.05	0.23	0.00	1.00	1.00	1.00	3.00

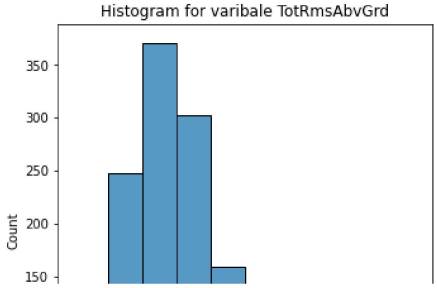


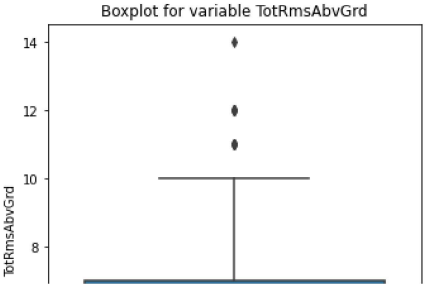


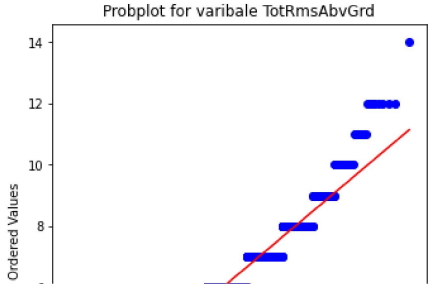


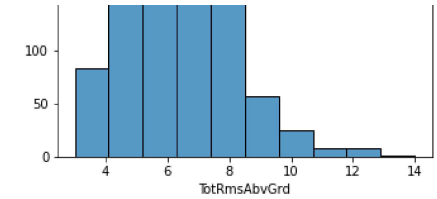

```
Total NA enties 0.0
```

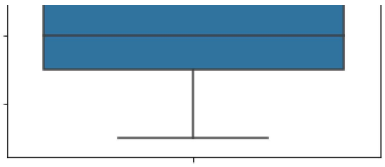
	count	mean	std	min	25%	50%	75%	max
TotRmsAbvGrd	1260.00	6.45	1.49	3.00	5.00	6.00	7.00	14.00

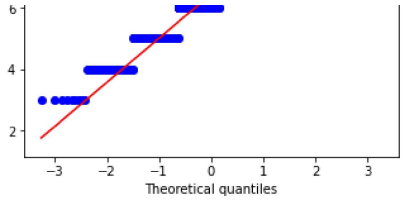






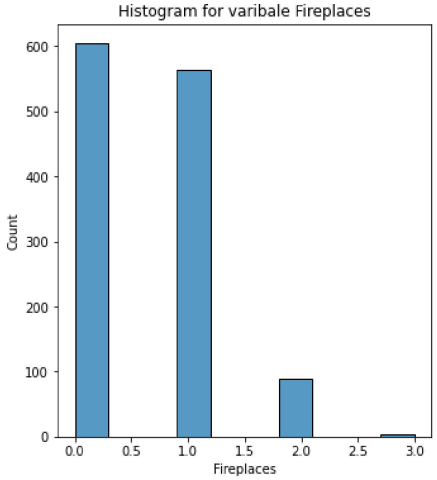


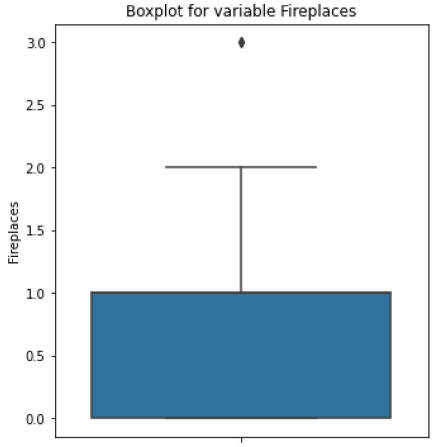


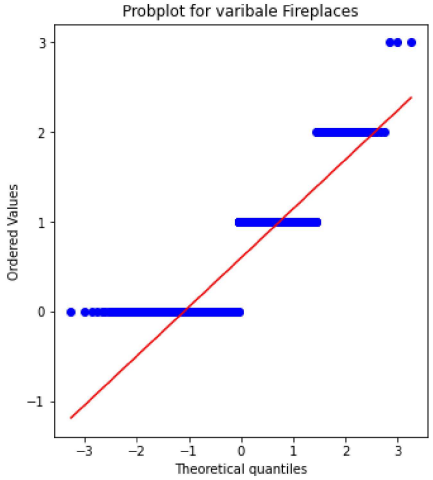



```
Total NA enties 0.0
```

	count	mean	std	min	25%	50%	75%	max
Fireplaces	1260.00	0.60	0.63	0.00	0.00	1.00	1.00	3.00

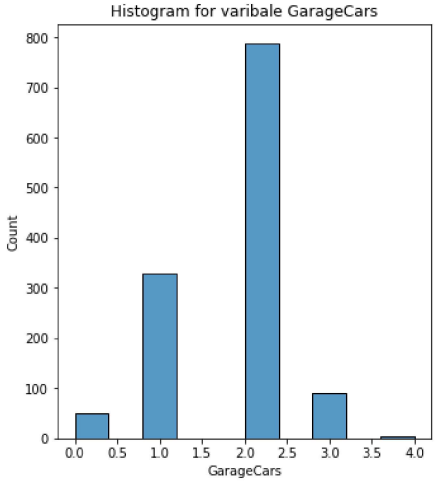


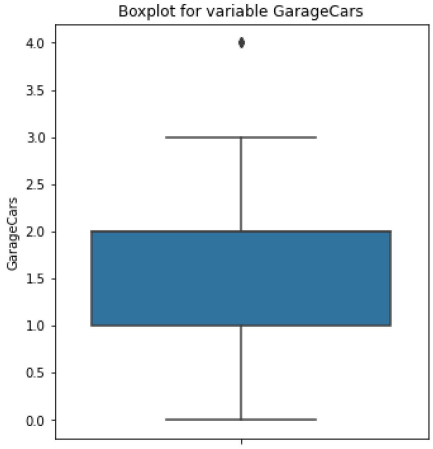


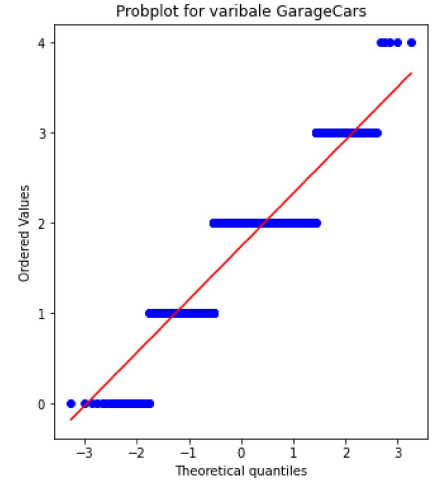



```
Total NA enties 0.0
```

	count	mean	std	min	25%	50%	75%	max
GarageCars	1260.00	1.74	0.66	0.00	1.00	2.00	2.00	4.00








```
Total NA enties 0.0
```

	count	mean	std	min	25%	50%	75%	max
GarageArea	1260.00	464.48	188.45	0.00	341.00	473.50	570.50	1418.00