Import Libraries

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

In [3]:
```
pd.set_option('display.max_columns',None)
```

Load Dataset

In [4]:
```
df = pd.read_csv('houseprice.csv')

print(df.shape)
```

```
(1460, 81)
```

In [7]:
```
df.head()
```

Out[7]:

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

The above housing price dataset contains 1460 rows and 81 columns/variables.

Let's analyse the dataset to identify the following:

1. Missing values
2. Numerical variables
3. Distribution of the numerical variables
4. Outliers
5. Categorical variables
6. Cardinality of the categorical variables
7. Potential relationship between the variables and the target: SalePrice

Missing Values:

In [9]:
```
#find varibales with missing values
mis_val = [var for var in df.columns if df[var].isnull().sum()>0]

#display missing values list
df[mis_val].isnull().sum()
```

Out[9]:
```
LotFrontage      259
Alley           1369
MasVnrType         8
```

```
MasVnrArea         8
BsmtQual          37
BsmtCond          37
BsmtExposure      38
BsmtFinType1      37
BsmtFinType2      38
Electrical         1
FireplaceQu      690
GarageType        81
GarageYrBlt       81
GarageFinish      81
GarageQual        81
GarageCond        81
PoolQC          1453
Fence           1179
MiscFeature     1406
dtype: int64
```

In [15]:

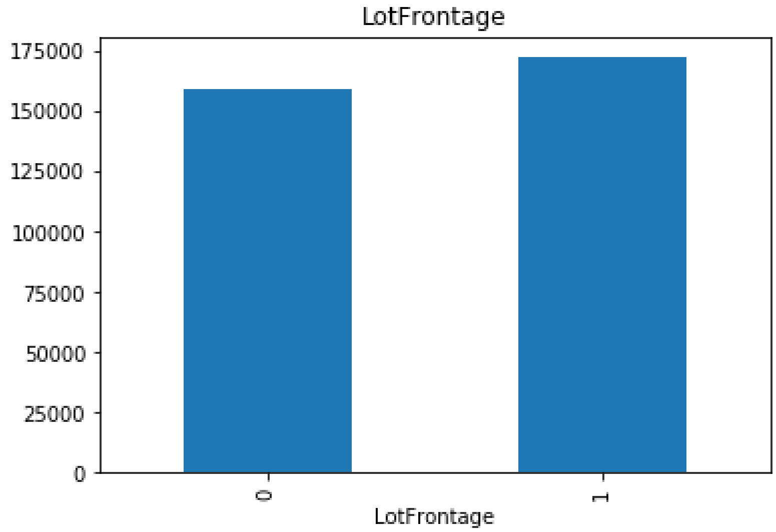
```
#mean of missig data
for var in mis_val:
    print(var, np.round(df[var].isnull().mean(), 3), ' %missing values')
```

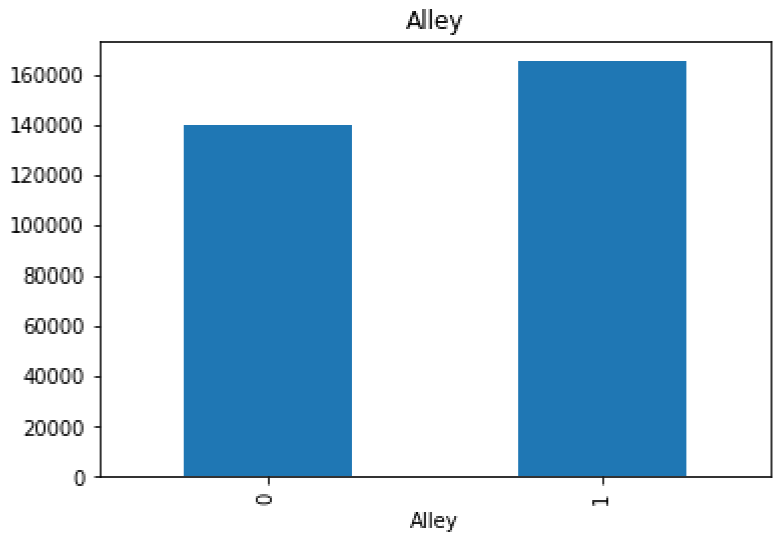
```
LotFrontage 0.177  %missing values
Alley 0.938  %missing values
MasVnrType 0.005  %missing values
MasVnrArea 0.005  %missing values
BsmtQual 0.025  %missing values
BsmtCond 0.025  %missing values
BsmtExposure 0.026  %missing values
BsmtFinType1 0.025  %missing values
BsmtFinType2 0.026  %missing values
Electrical 0.001  %missing values
FireplaceQu 0.473  %missing values
GarageType 0.055  %missing values
GarageYrBlt 0.055  %missing values
GarageFinish 0.055  %missing values
GarageQual 0.055  %missing values
GarageCond 0.055  %missing values
PoolQC 0.995  %missing values
Fence 0.808  %missing values
MiscFeature 0.963  %missing values
```

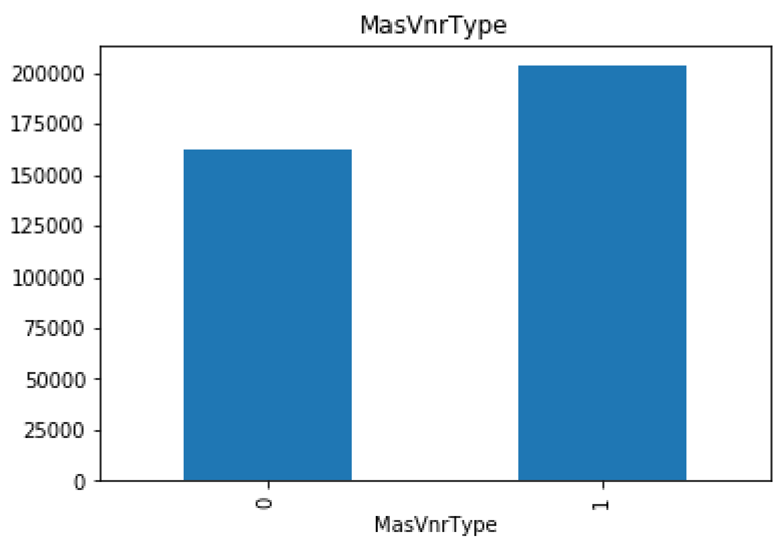
Relation between missing values and house price:

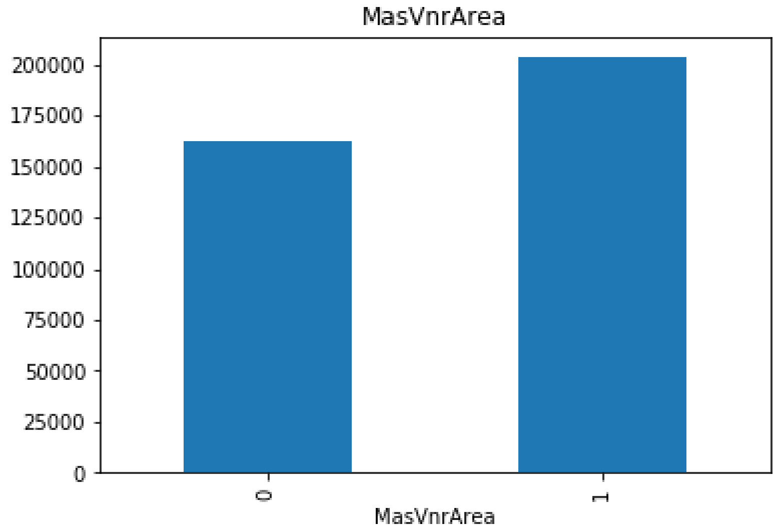
In [16]:

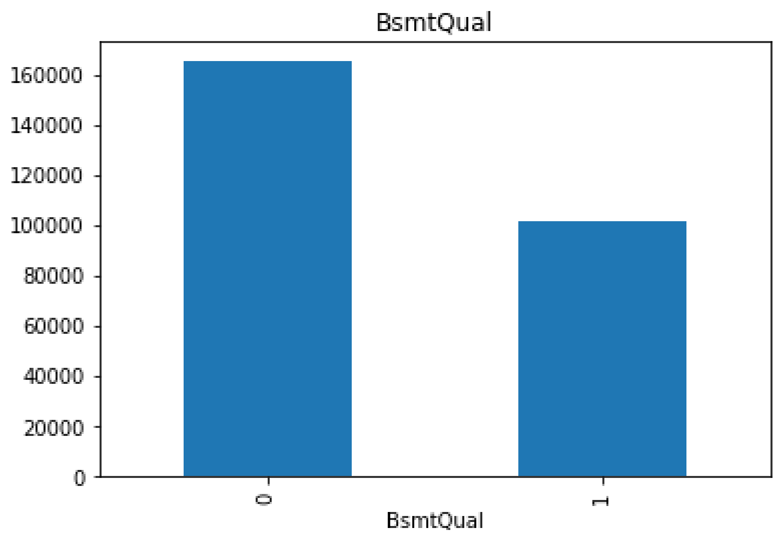
```
def analyze_missing_val(df,var):

    df = df.copy()

    df[var] = np.where(df[var].isnull(),1,0)

    #calculate mean price where value is missing
    df.groupby(var)['SalePrice'].median().plot.bar()
    plt.title(var)
    plt.show()

for var in mis_val:
    analyze_missing_val(df,var)
```

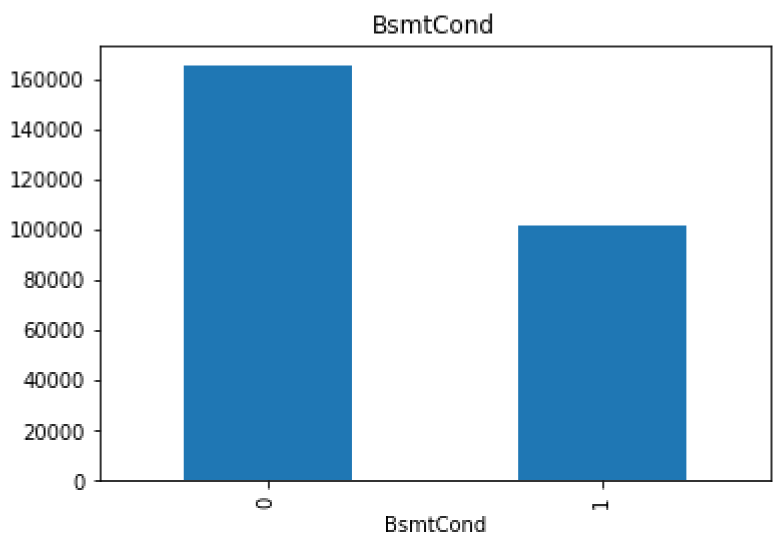

LotFrontage
175000
150000
125000
100000
75000
50000
25000
0
0
1
LotFrontage

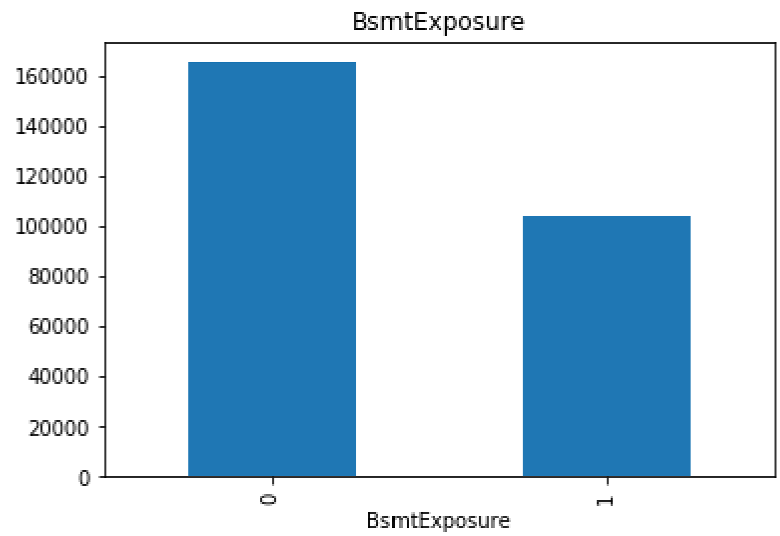

Alley
160000
140000
120000
100000
80000
60000
40000
20000
0
0
1
Alley

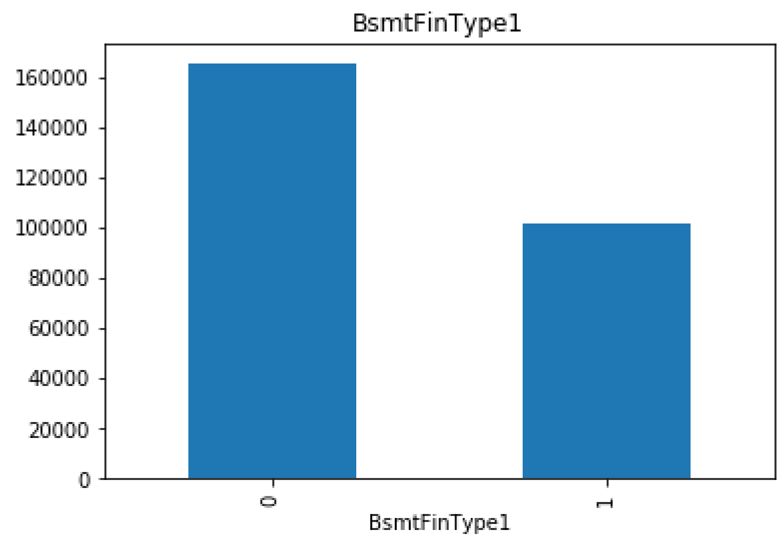

MasVnrType
200000
175000
150000
125000
100000
75000
50000
25000
0
0
1
MasVnrType

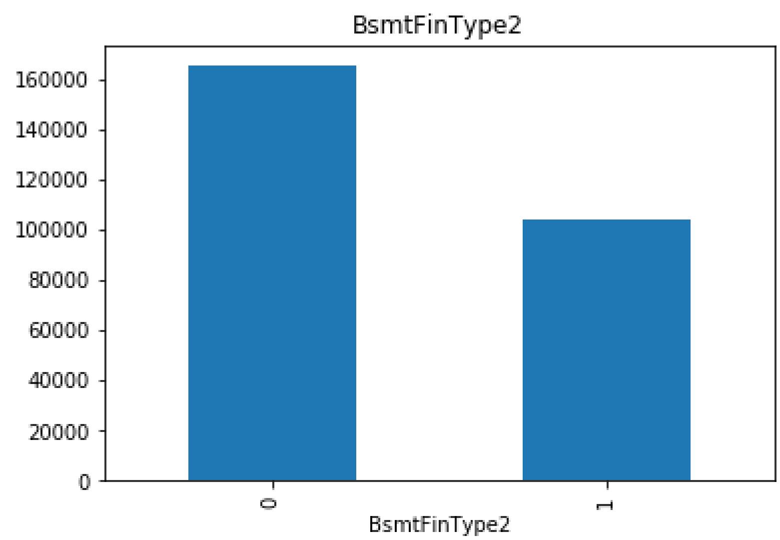

MasVnrArea
200000
175000
150000
125000
100000
75000
50000
25000
0
0
1
MasVnrArea

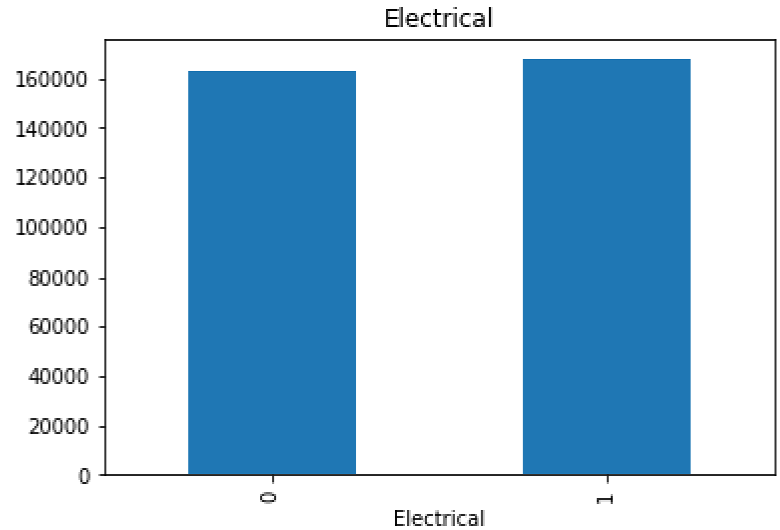

BsmtQual
160000
140000
120000
100000
80000
60000
40000
20000
0
0
1
BsmtQual

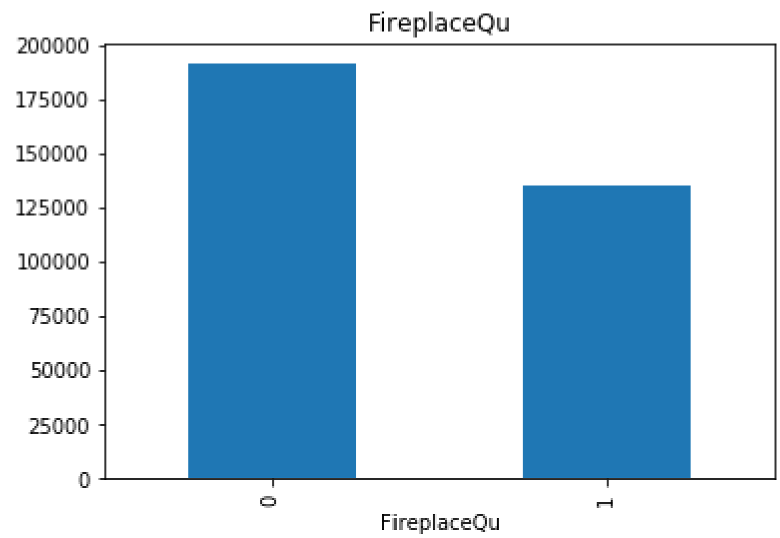

BsmtCond
160000
140000
120000
100000
80000
60000
40000
20000
0
0
1
BsmtCond

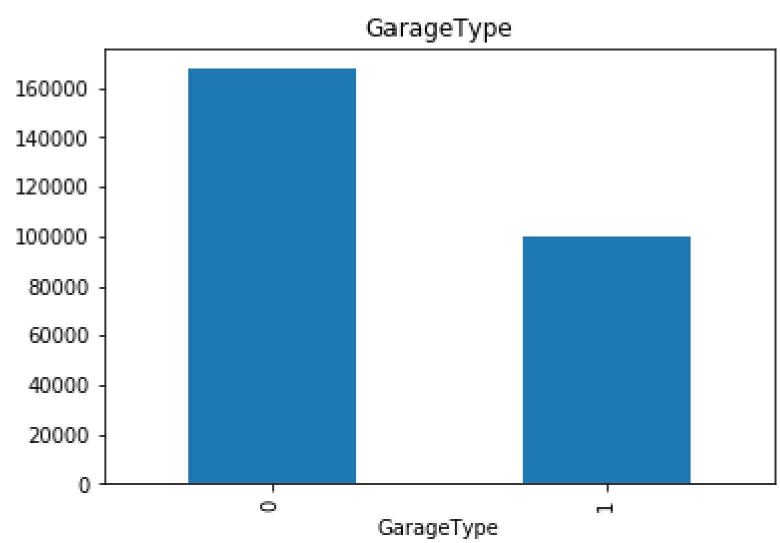

BsmtExposure
160000
140000
120000
100000
80000
60000
40000
20000
0
0
1
BsmtExposure

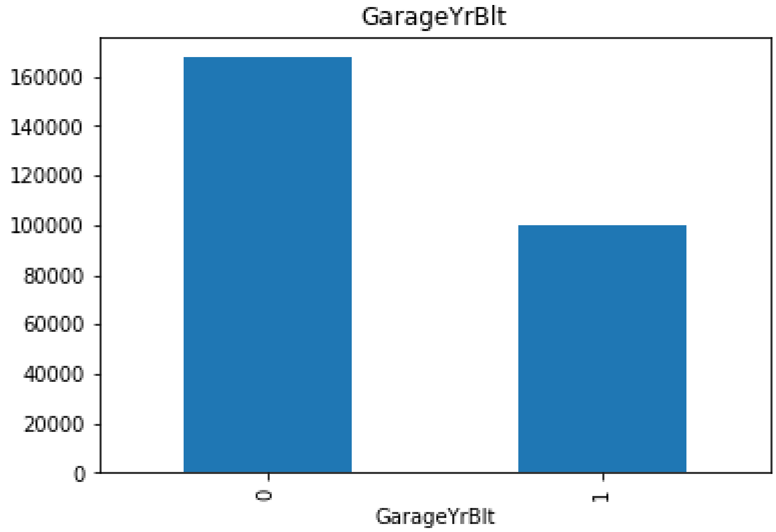

BsmtFinType1
160000
140000
120000
100000
80000
60000
40000
20000
0
0
1
BsmtFinType1

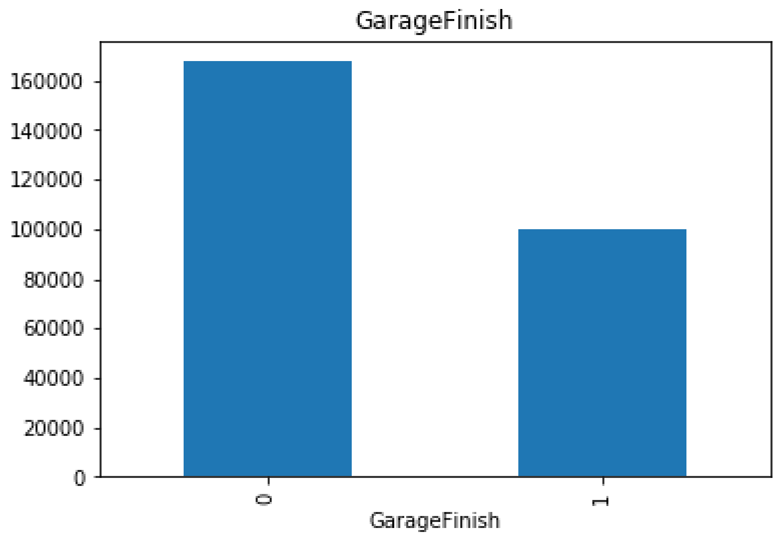

BsmtFinType2
160000
140000
120000
100000
80000
60000
40000
20000
0
0
1
BsmtFinType2

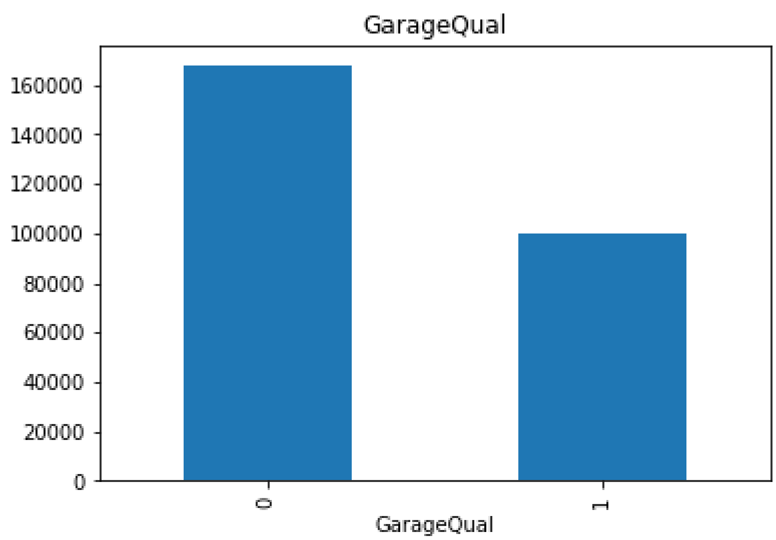

Electrical
160000
140000
120000
100000
80000
60000
40000
20000
0
0
1
Electrical

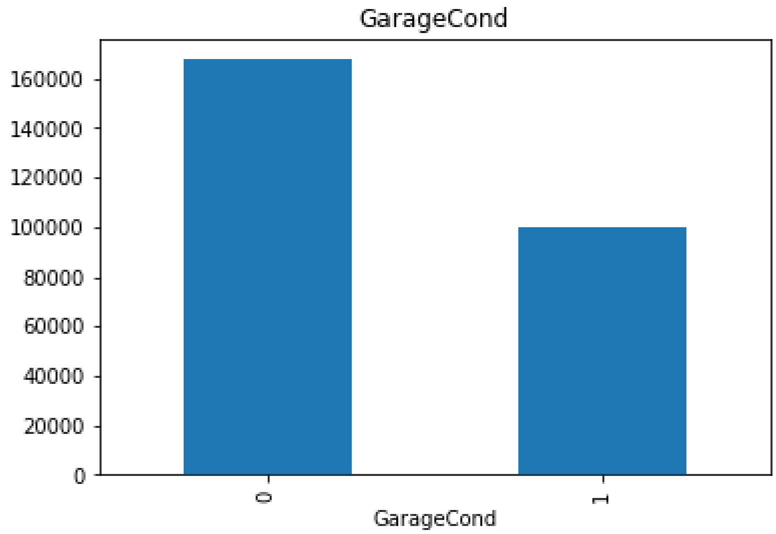

FireplaceQu
200000
175000
150000
125000
100000
75000
50000
25000
0
0
1
FireplaceQu

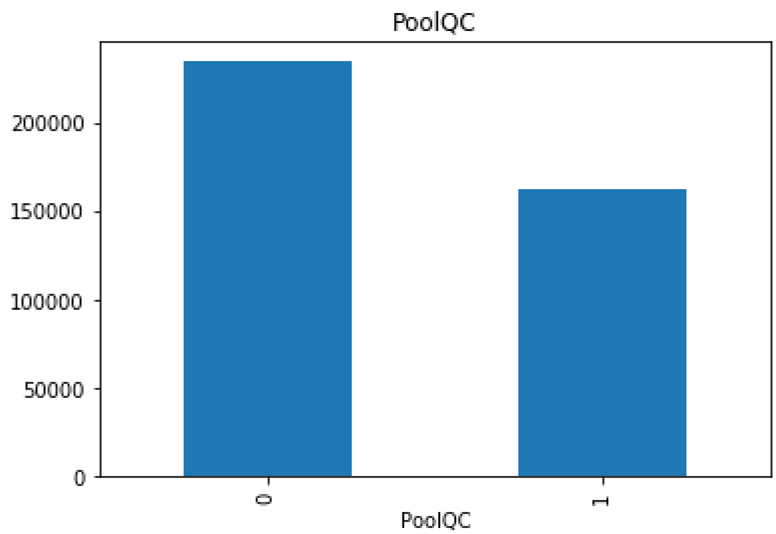

GarageType
160000
140000
120000
100000
80000
60000
40000
20000
0
0
1
GarageType

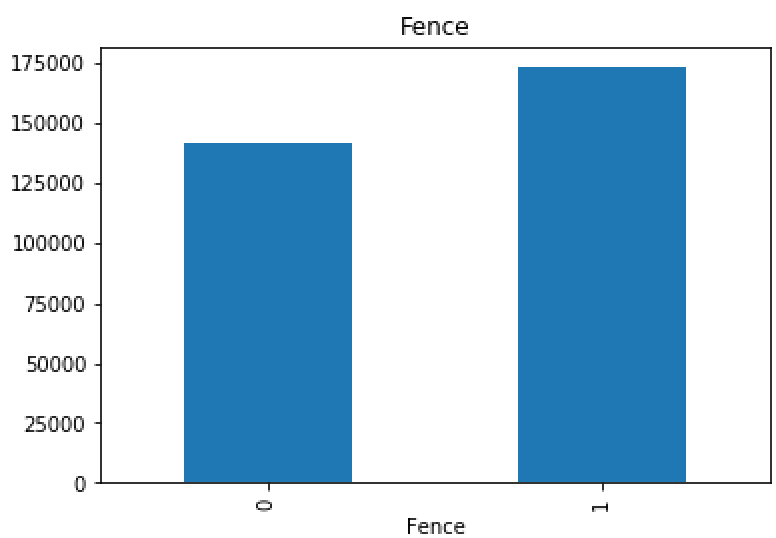

GarageYrBlt
160000
140000
120000
100000
80000
60000
40000
20000
0
0
1
GarageYrBlt

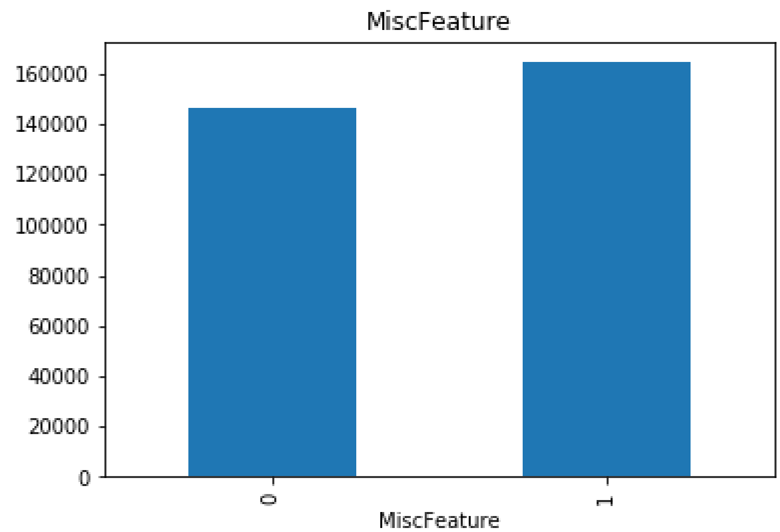

GarageFinish
160000
140000
120000
100000
80000
60000
40000
20000
0
0
1
GarageFinish


GarageQual
160000
140000
120000
100000
80000
60000
40000
20000
0
0
1
GarageQual


GarageCond
160000
140000
120000
100000
80000
60000
40000
20000
0
0
1
GarageCond


PoolQC
200000
150000
100000
50000
0
0
1
PoolQC


Fence
175000
150000
125000
100000
75000
50000
25000
0
0
1
Fence



The average Sale Price in houses where the information is missing, differs from the average Sale Price in houses where information exists.

Numerical Variables:

In [17]:
```
num_vars = [var for var in df.columns if df[var].dtypes != 'O']
print('No of numerical varibales in the dataset : ',len(num_vars))
```

```
No of numerical varibales in the dataset :  38
```

In [18]:
```
df[num_vars].head()
```

Out[18]:

	Id	MSSubClass	LotFrontage	LotArea	OverallQual	OverallCond	YearBuilt	YearRemodAdd	MasVnr
0	1	60	65.0	8450	7	5	2003	2003	
1	2	20	80.0	9600	6	8	1976	1976	
2	3	60	68.0	11250	7	5	2001	2002	
3	4	70	60.0	9550	7	5	1915	1970	
4	5	60	84.0	14260	8	5	2000	2000	

From the above view of the dataset, we notice the variable Id, which is an indicator of the house. We will not use this variable to make our predictions, as there is one different value of the variable per each row, i.e., each house in the dataset. See below:

In [19]:
```
print('No of House Id labels: ',len(df.Id.unique()))
print('No of Houses in dataset: ',len(df))
```

```
No of House Id labels:  1460
No of Houses in dataset:  1460
```

Temporal Variables:

We have 4 year variables in the dataset:

- YearBuilt: year in which the house was built
- YearRemodAdd: year in which the house was remodeled

- GarageYrBlt: year in which a garage was built
- YrSold: year in which the house was sold

We generally don't use date variables in their raw format. Instead, we extract information from them. For example, we can capture the difference in years between the year the house was built and the year the house was sold.

In [21]:
```
year_var = [var for var in num_vars if 'Yr' in var or 'Year' in var ]
len(year_var), year_var
```

Out[21]:
```
(4, ['YearBuilt', 'YearRemodAdd', 'GarageYrBlt', 'YrSold'])
```

In [25]:
```
#Checking the values in year_var
for i in year_var:
    print(i, df[i].unique())
    print()
```

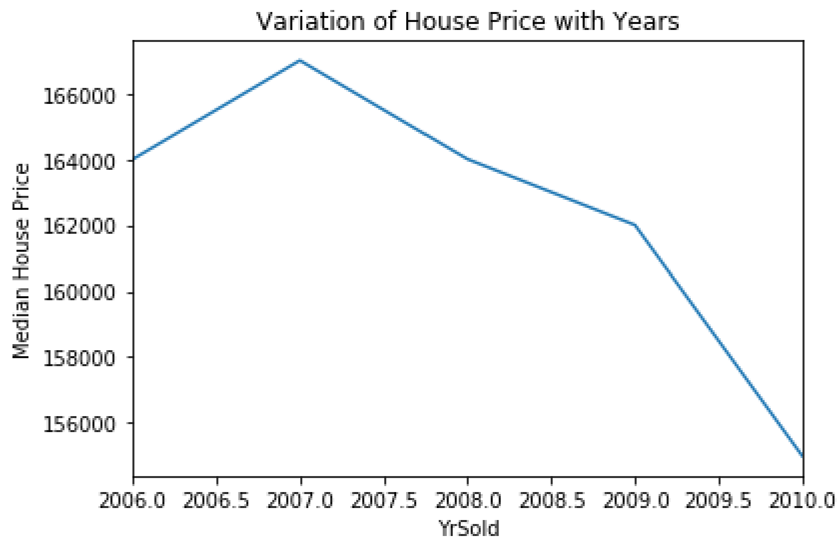
```
YearBuilt [2003 1976 2001 1915 2000 1993 2004 1973 1931 1939 1965 2005 1962 2006
 1960 1929 1970 1967 1958 1930 2002 1968 2007 1951 1957 1927 1920 1966
 1959 1994 1954 1953 1955 1983 1975 1997 1934 1963 1981 1964 1999 1972
 1921 1945 1982 1998 1956 1948 1910 1995 1991 2009 1950 1961 1977 1985
 1979 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]

YearRemodAdd [2003 1976 2002 1970 2000 1995 2005 1973 1950 1965 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]

GarageYrBlt [2003. 1976. 2001. 1998. 2000. 1993. 2004. 1973. 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.]

YrSold [2008 2007 2006 2009 2010]
```

As expected, the values are years.

We can explore the evolution of the sale price with the years in which the house was sold:

In [33]:
```
df.groupby('YrSold')['SalePrice'].median().plot()
plt.title('Variation of House Price with Years')
plt.ylabel('Median House Price')
```

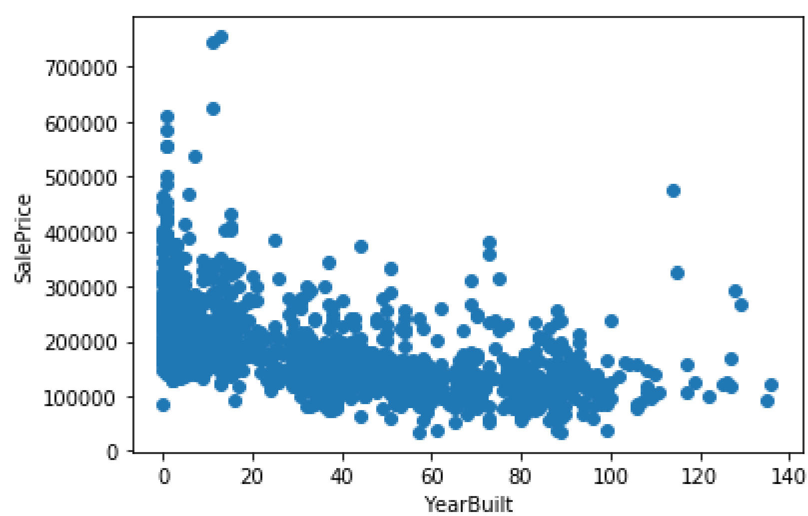
Out[33]:
```
Text(0, 0.5, 'Median House Price')
```

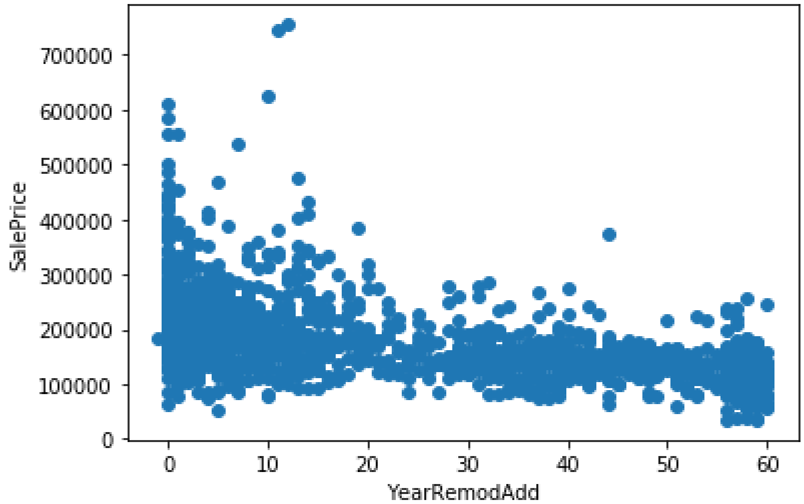


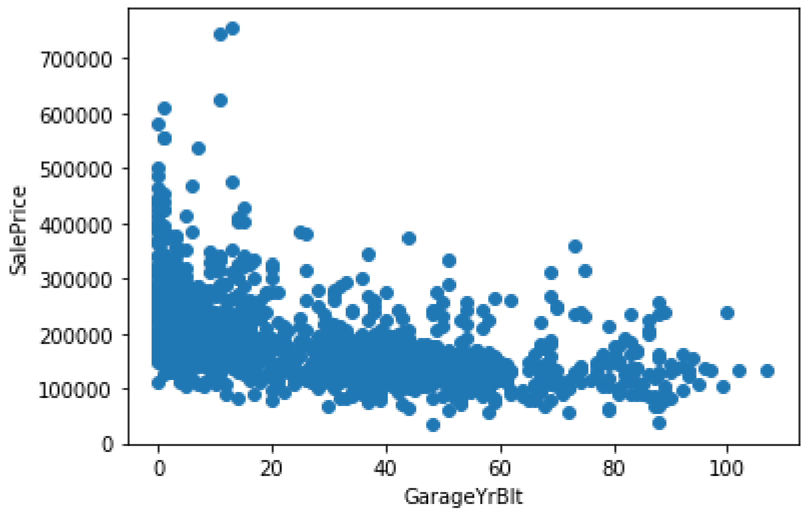
There has been a drop in the value of the houses. That is unusual, in real life, house prices typically go up as years go by.

Let's go ahead and explore whether there is a relationship between the year variables and SalePrice. For this, we will capture the elapsed years between the Year variables and the year in which the house was sold:

In [39]:
```
def analyze_year_var(df,var):

    df = df.copy()

    #calculate difference between year variable and year the house was sold
    df[var] = df['YrSold'] - df[var]

    plt.scatter(df[var],df['SalePrice'])
    plt.xlabel(var)
    plt.ylabel('SalePrice')
    plt.show()

for var in year_var:
    if var != 'YrSold':
        analyze_year_var(df,var)
```







We see that there is a tendency to a decrease in price, with older features. In other words, the longer the time between the house was built or remodeled and sale date, the lower the sale Price.

Which makes sense, cause this means that the house will have an older look, and potentially needs repairs.

Discrete Variables:

In [42]:

```
discrete_vars = [var for var in num_vars if len(df[var].unique())<20 and var not in yea
print('No of discrete variables : ', len(discrete_vars))
```

```
No of discrete variables :  14
```

In [43]:

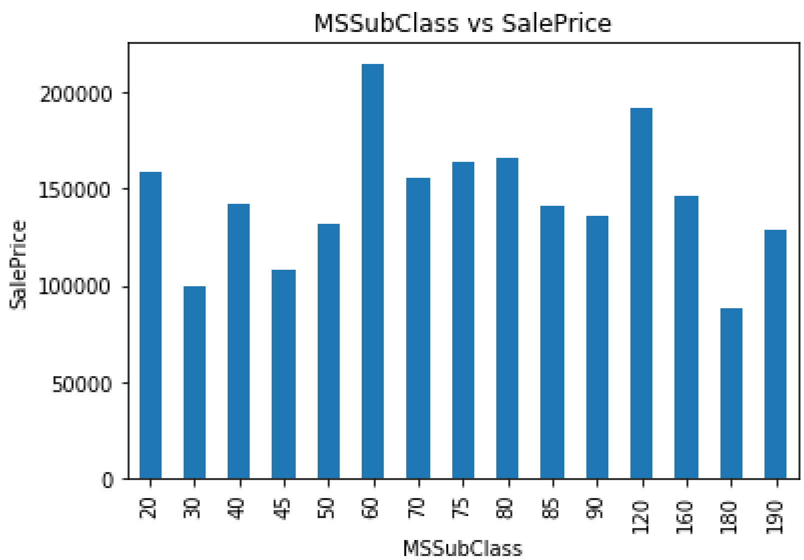
```
df[discrete_vars].head()
```

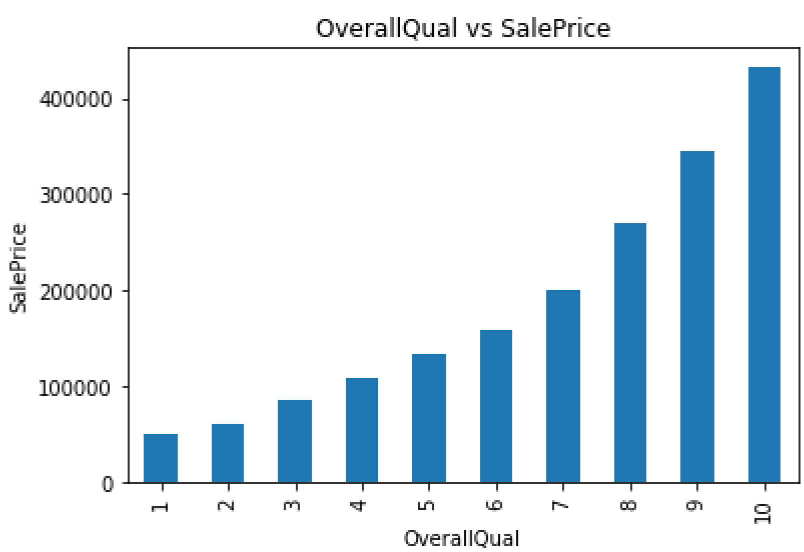
Out[43]:

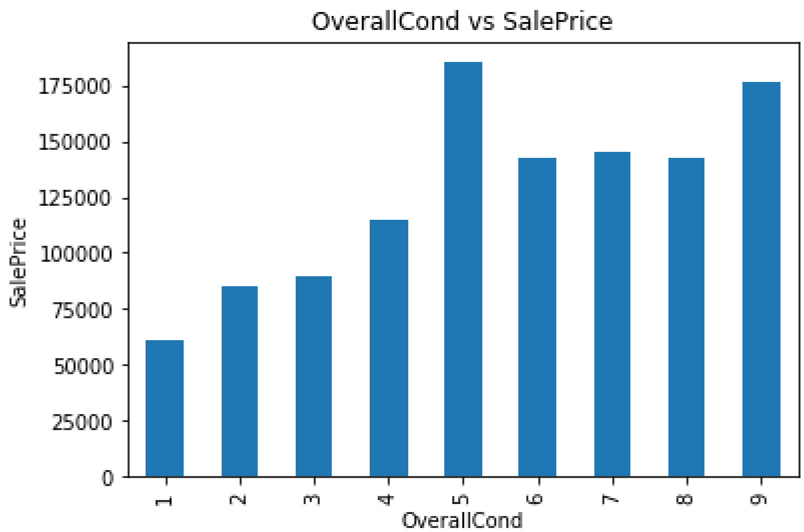
	MSSubClass	OverallQual	OverallCond	BsmtFullBath	BsmtHalfBath	FullBath	HalfBath	BedroomAb
0	60	7	5	1	0	2	1	
1	20	6	8	0	1	2	0	
2	60	7	5	1	0	2	1	
3	70	7	5	1	0	1	0	
4	60	8	5	1	0	2	1	

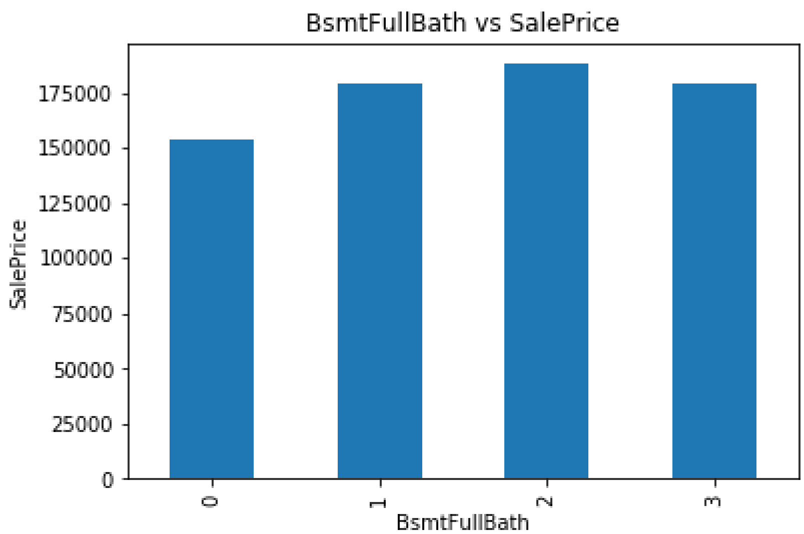
Visualizing Discrete Variables:

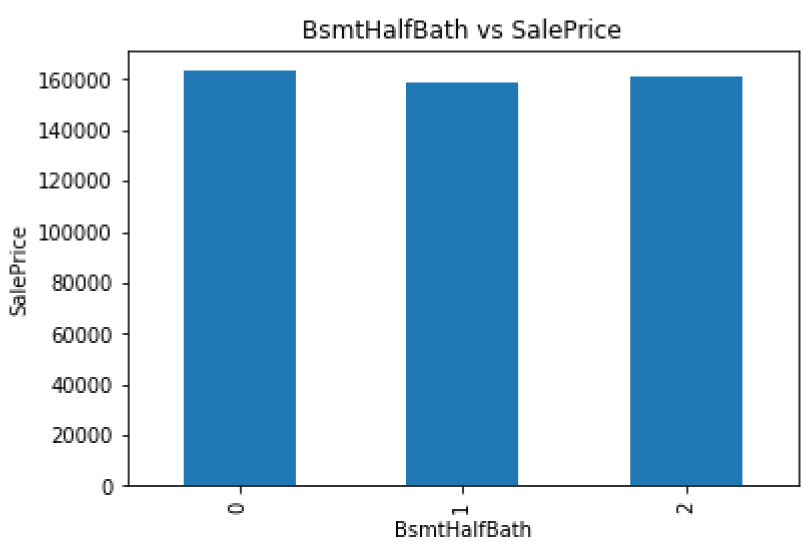
In [45]:
```
def analyze_discrete_vars(df,var):

    df = df.copy()

    df.groupby(var)['SalePrice'].median().plot.bar()
    plt.title(var + ' vs SalePrice')
    plt.xlabel(var)
    plt.ylabel('SalePrice')
    plt.show()

for var in discrete_vars:
    analyze_discrete_vars(df,var)
```

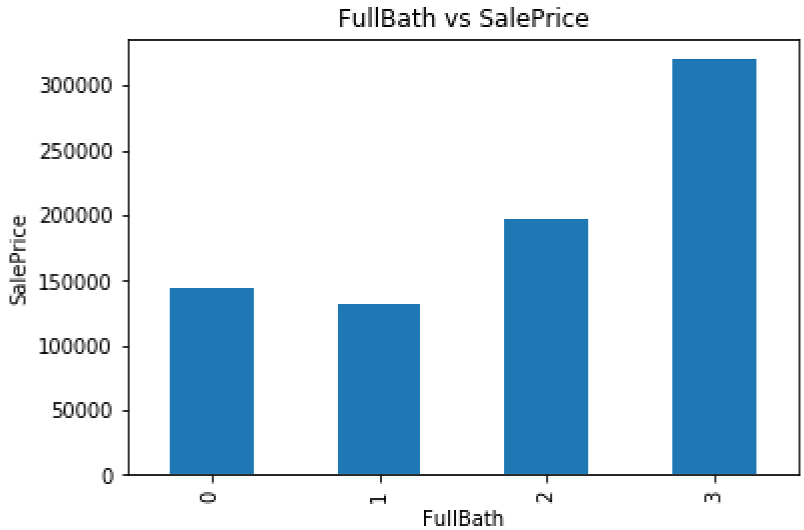


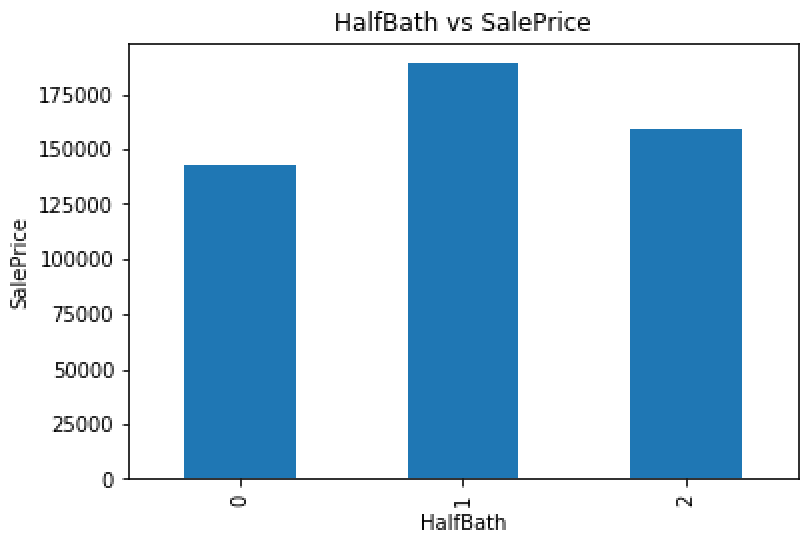


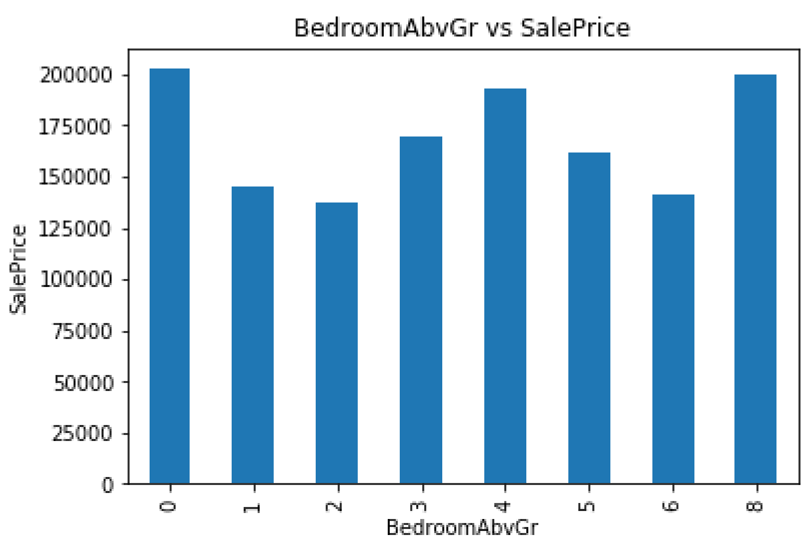

OverallCond vs SalePrice
SalePrice
OverallCond
0
25000
50000
75000
100000
125000
150000
175000
1
2
3
4
5
6
7
8
9

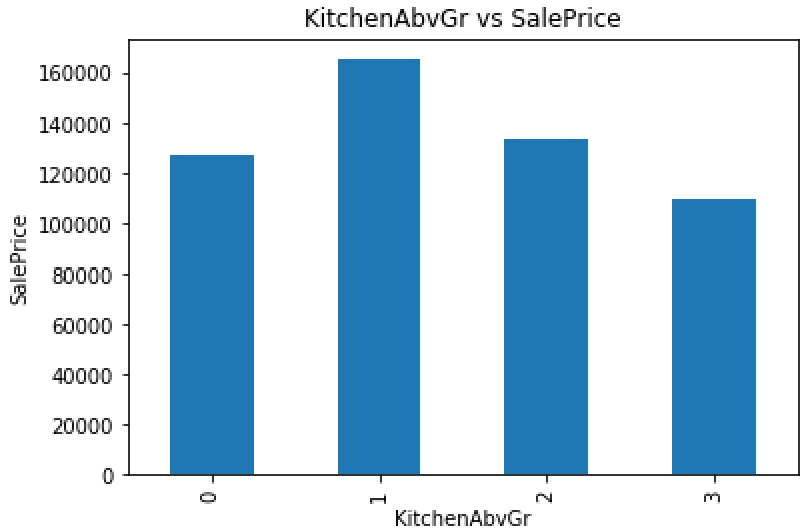

BsmtFullBath vs SalePrice
SalePrice
BsmtFullBath
0
25000
50000
75000
100000
125000
150000
175000
0
1
2
3

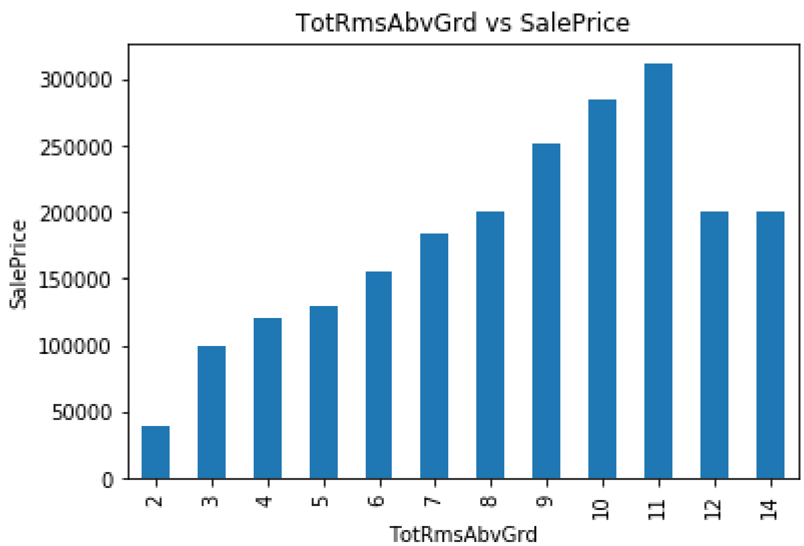

BsmtHalfBath vs SalePrice
SalePrice
BsmtHalfBath
0
20000
40000
60000
80000
100000
120000
140000
160000
0
1
2

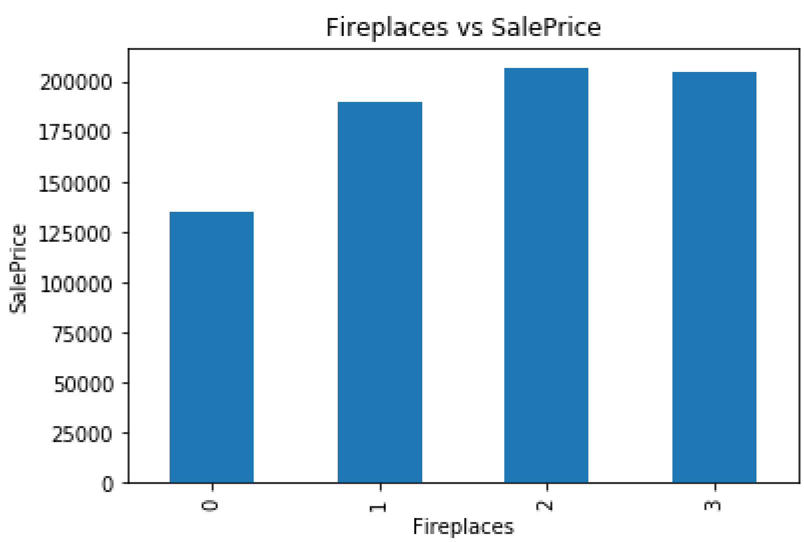

FullBath vs SalePrice
SalePrice
0
50000
100000
150000
200000
250000
300000
0
1
2
3
FullBath

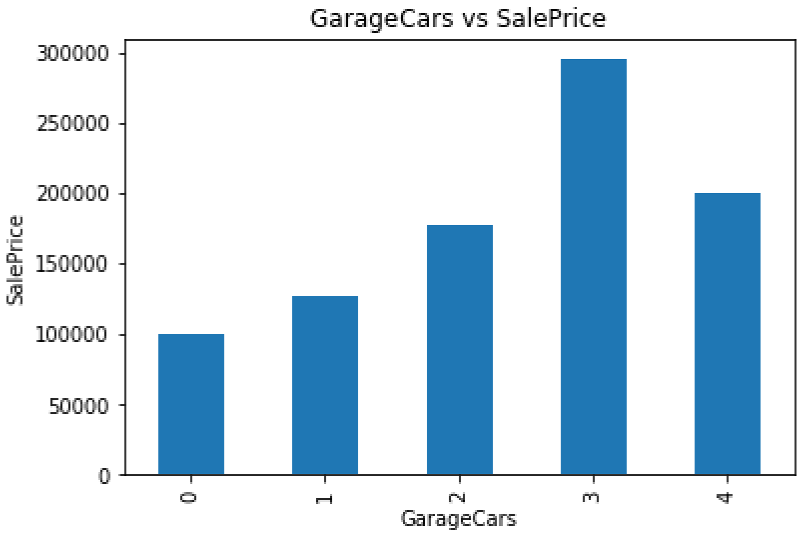

HalfBath vs SalePrice
SalePrice
0
25000
50000
75000
100000
125000
150000
175000
0
1
2
HalfBath

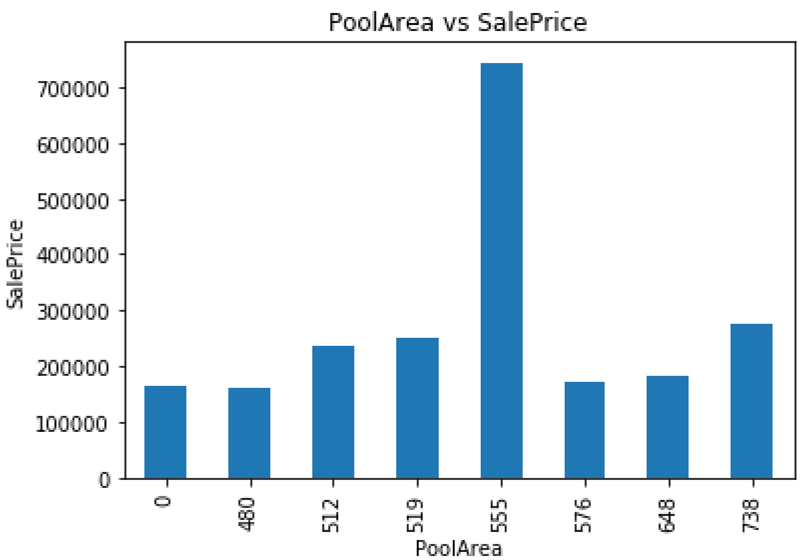

BedroomAbvGr vs SalePrice
SalePrice
0
25000
50000
75000
100000
125000
150000
175000
200000
0
1
2
3
4
5
6
8
BedroomAbvGr

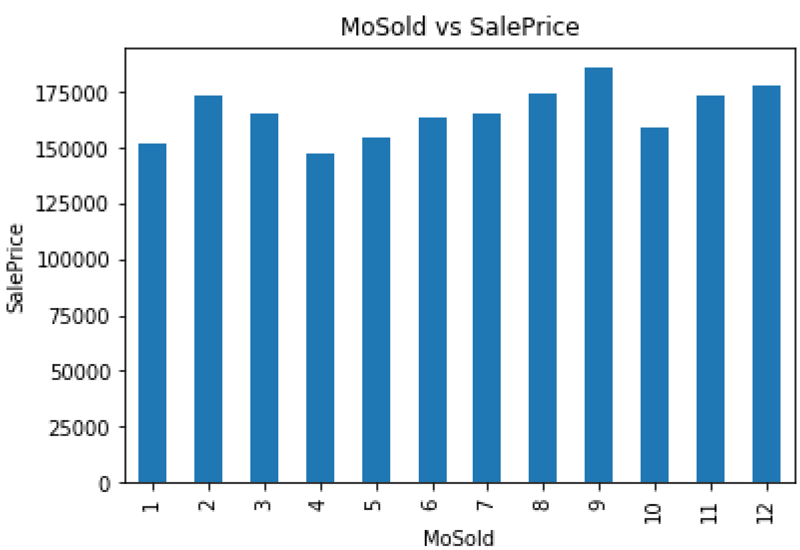

KitchenAbvGr vs SalePrice
SalePrice
160000
140000
120000
100000
80000
60000
40000
20000
0
0
1
2
3
KitchenAbvGr


TotRmsAbvGrd vs SalePrice
SalePrice
300000
250000
200000
150000
100000
50000
0
2
3
4
5
6
7
8
9
10
11
12
14
TotRmsAbvGrd


Fireplaces vs SalePrice
SalePrice
200000
175000
150000
125000
100000
75000
50000
25000
0
0
1
2
3
Fireplaces







There tend to be a relationship between the variables values and the SalePrice, but this relationship is not always monotonic.

For example, for OverallQual, there is a monotonic relationship: the higher the quality, the higher the SalePrice.

However, for OverallCond, the relationship is not monotonic. Clearly, some Condition grades, like 5, correlate with higher sale prices, but higher values do not necessarily do so. We need to be careful on how we engineer these variables to extract maximum value for a linear model.

There are ways to re-arrange the order of the discrete values of a variable, to create a monotonic relationship between the variable and the target.

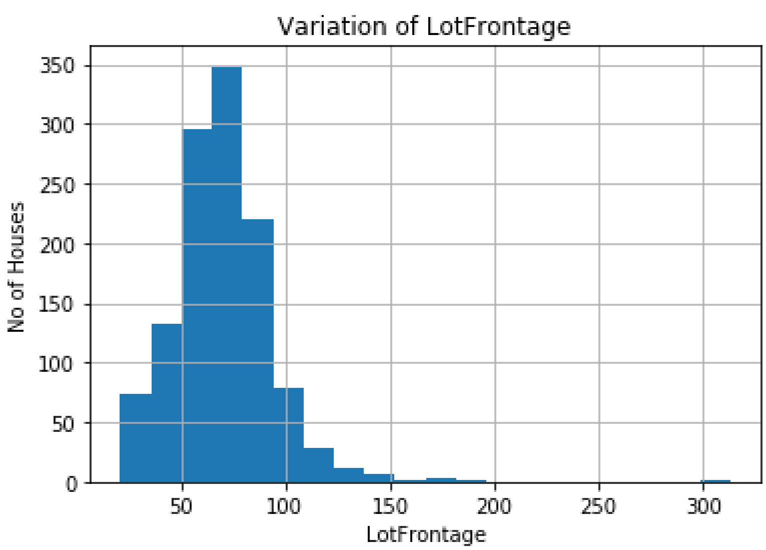
Continuous Variables:

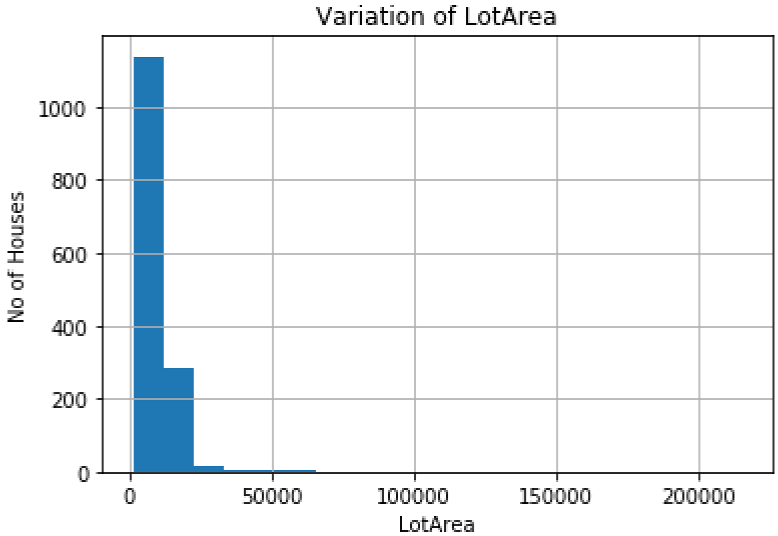
In [46]:
```
cont_vars = [var for var in num_vars if var not in discrete_vars+year_var+['Id']]
print('No of continuous variables : ', len(cont_vars))
```

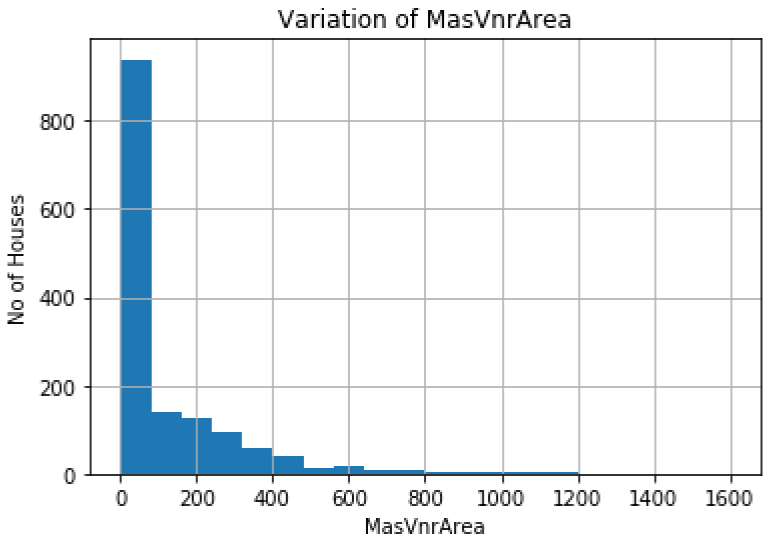
```
No of continuous variables :  19
```

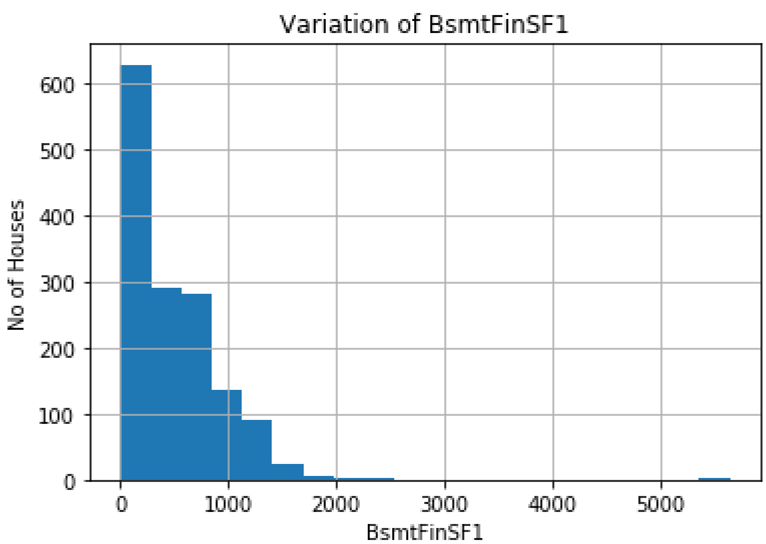
Visualizing continuous variables

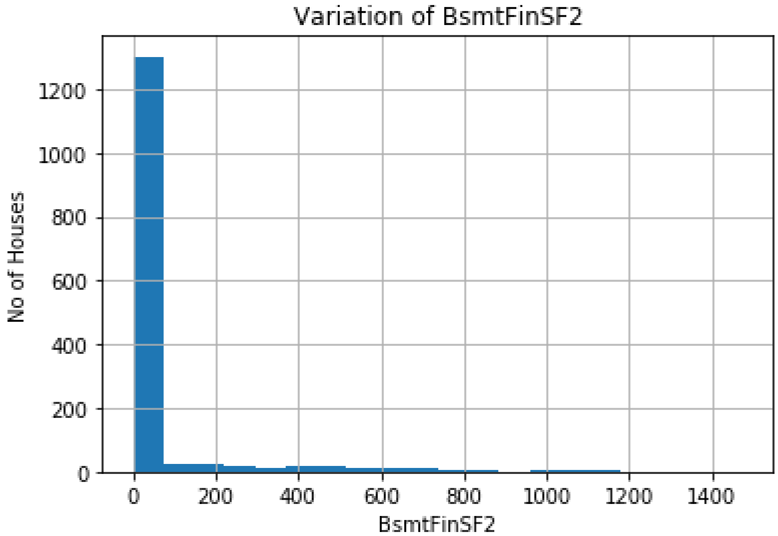
In [48]:
```
def analyze_cont_vars(df,var):

    df = df.copy()

    df[var].hist(bins=20)
    plt.xlabel(var)
    plt.ylabel('No of Houses')
    plt.title('Variation of '+ var)
    plt.show()

for var in cont_vars:
    analyze_cont_vars(df,var)
```

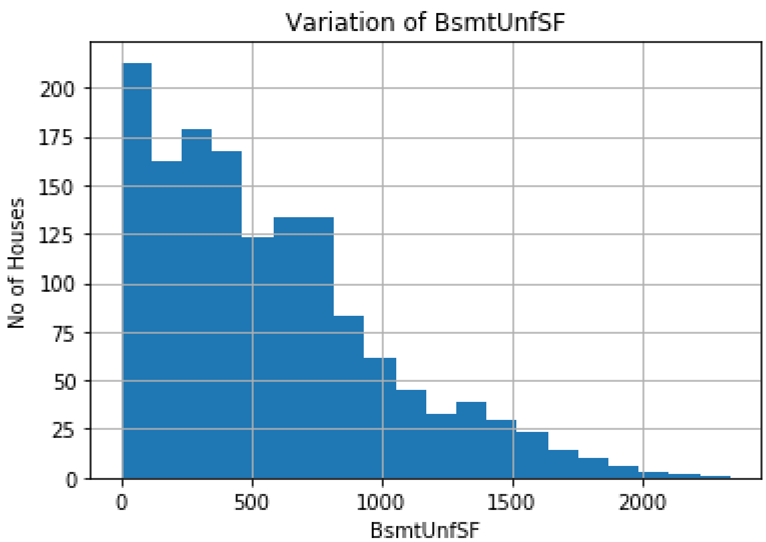


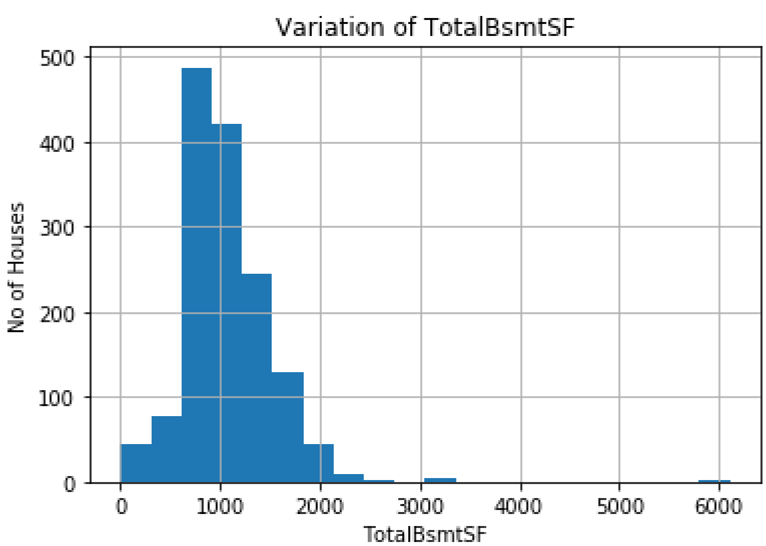

Variation of LotArea
No of Houses
LotArea
0
200
400
600
800
1000
50000
100000
150000
200000

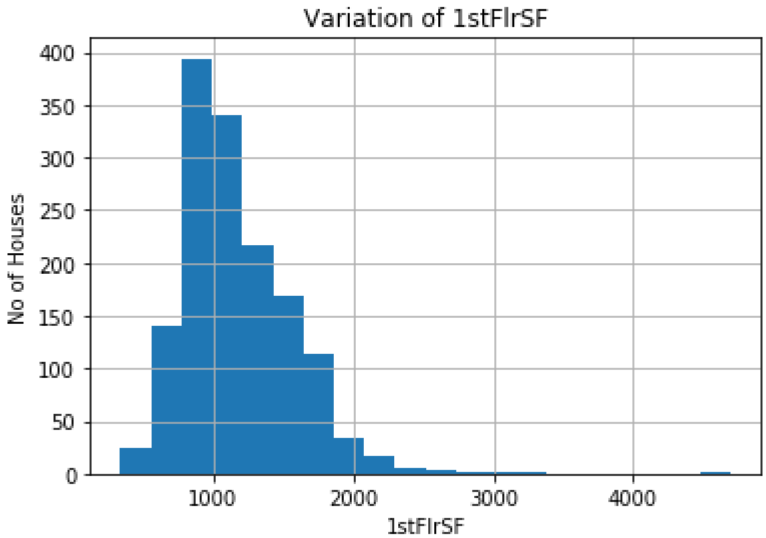

Variation of MasVnrArea
No of Houses
MasVnrArea
0
200
400
600
800
1000
1200
1400
1600

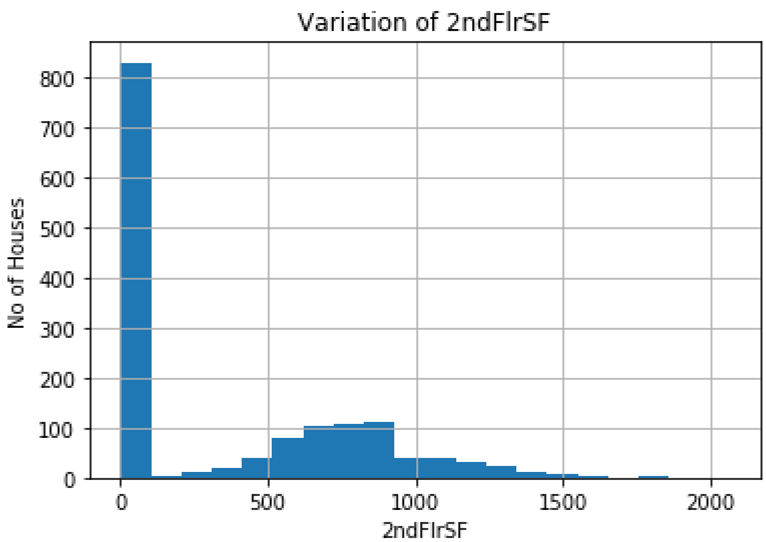

Variation of BsmtFinSF1
No of Houses
BsmtFinSF1
0
100
200
300
400
500
600
1000
2000
3000
4000
5000

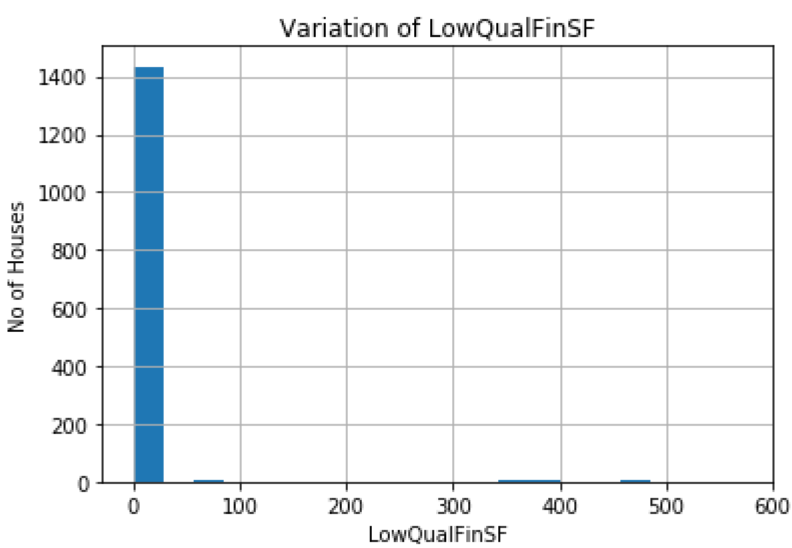

Variation of BsmtFinSF2
No of Houses
BsmtFinSF2

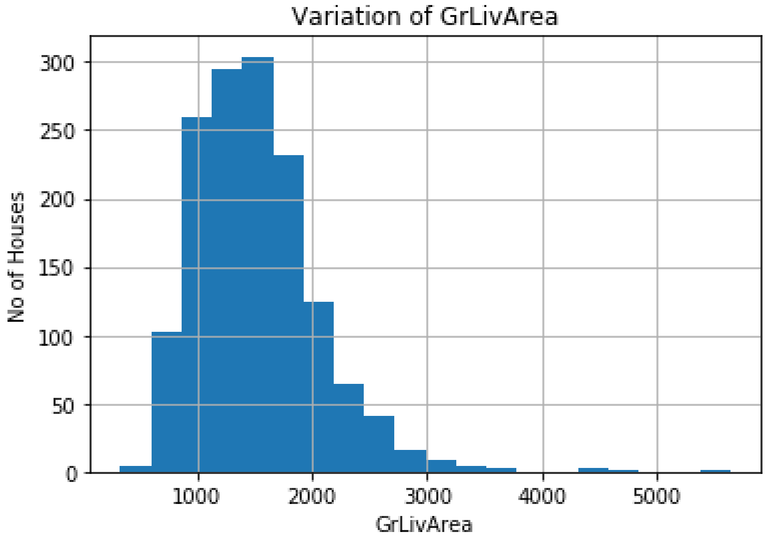

Variation of BsmtUnfSF
No of Houses
BsmtUnfSF

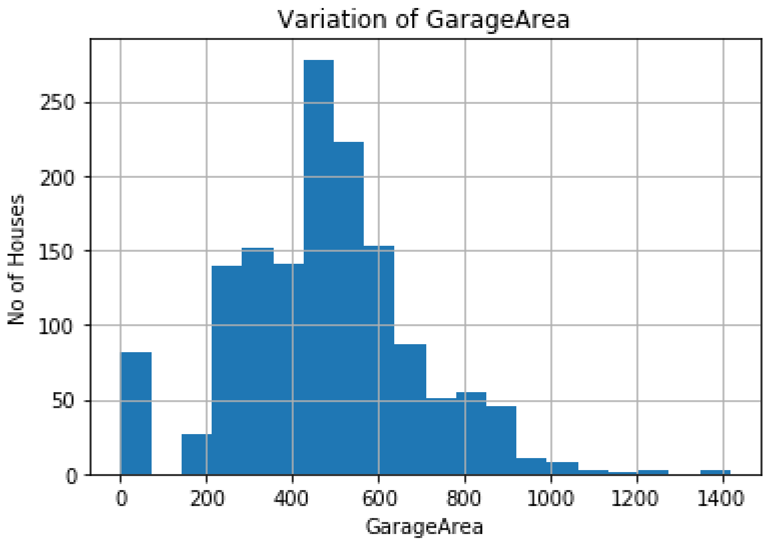

Variation of TotalBsmtSF
No of Houses
TotalBsmtSF

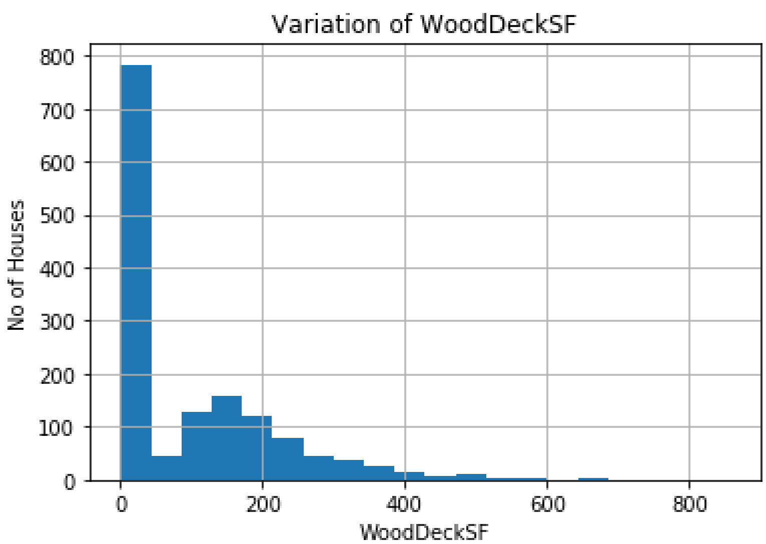

Variation of 1stFlrSF
No of Houses
0
50
100
150
200
250
300
350
400
1000
2000
3000
4000
1stFlrSF

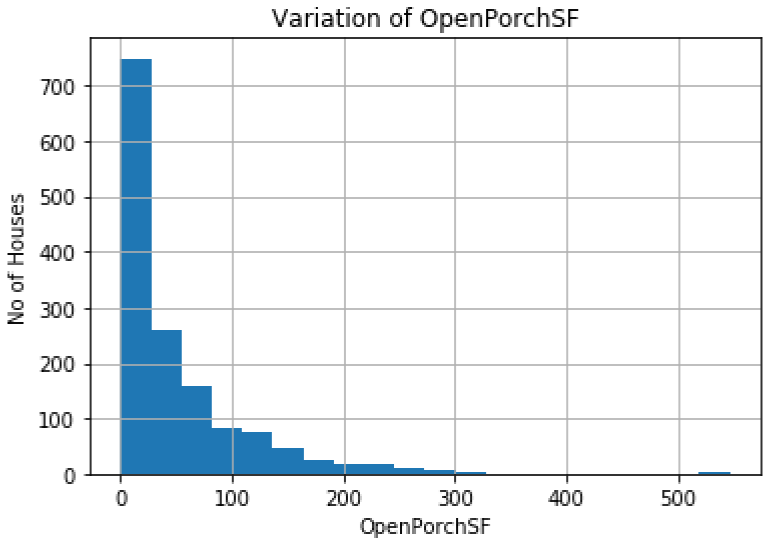

Variation of 2ndFlrSF
No of Houses
0
100
200
300
400
500
600
700
800
0
500
1000
1500
2000
2ndFlrSF

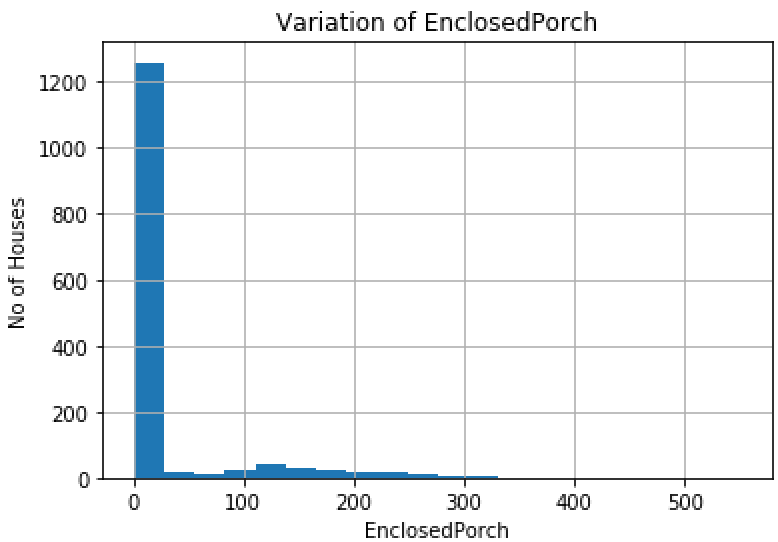

Variation of LowQualFinSF
No of Houses
0
200
400
600
800
1000
1200
1400
0
100
200
300
400
500
600
LowQualFinSF

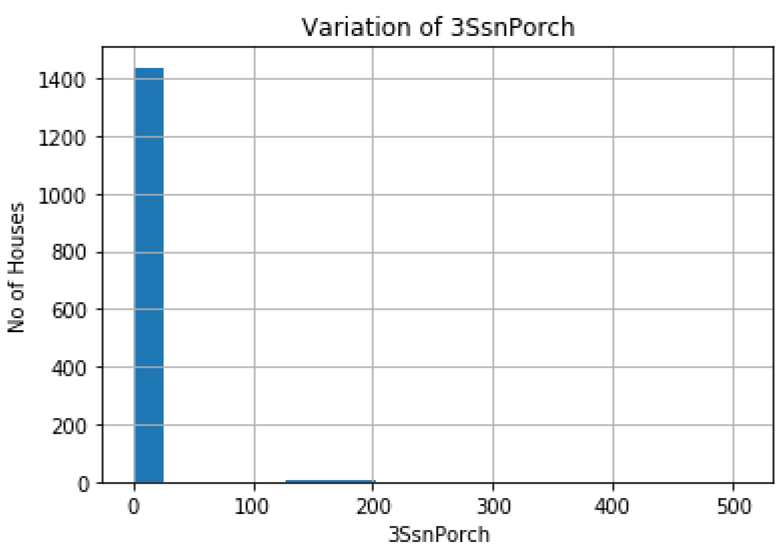

Variation of GrLivArea
No of Houses
0
50
100
150
200
250
300
1000
2000
3000
4000
5000
GrLivArea

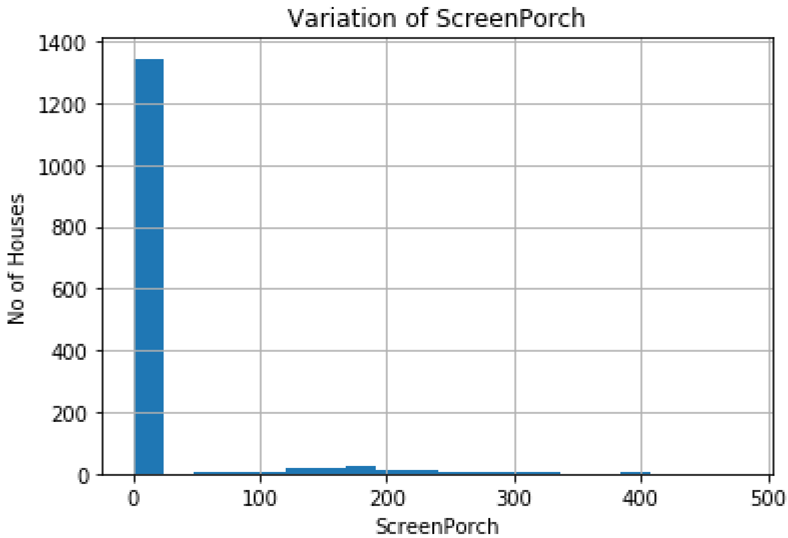

Variation of GarageArea
No of Houses
0
50
100
150
200
250
0
200
400
600
800
1000
1200
1400
GarageArea

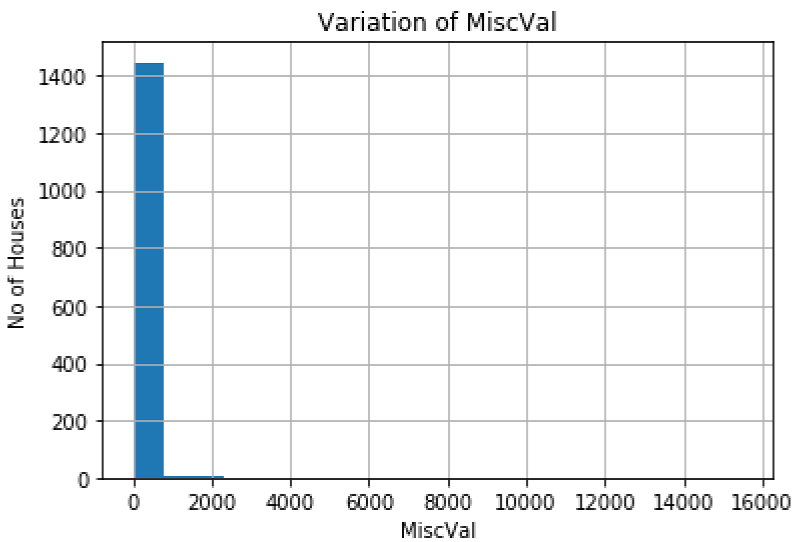

Variation of WoodDeckSF
No of Houses
0
100
200
300
400
500
600
700
800
0
200
400
600
800
WoodDeckSF

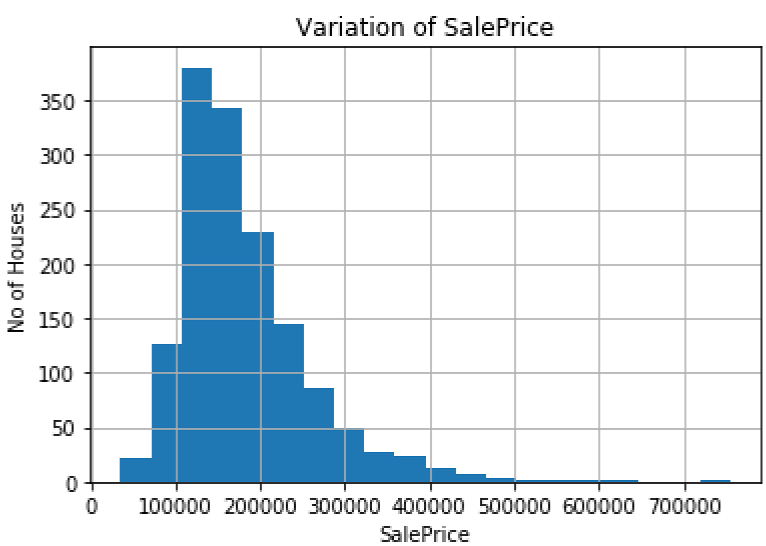

Variation of OpenPorchSF
No of Houses
OpenPorchSF
0
100
200
300
400
500
600
700


Variation of EnclosedPorch
No of Houses
EnclosedPorch
0
200
400
600
800
1000
1200
100
300
500


Variation of 3SsnPorch
No of Houses
3SsnPorch
0
200
400
600
800
1000
1200
1400
100
300
500





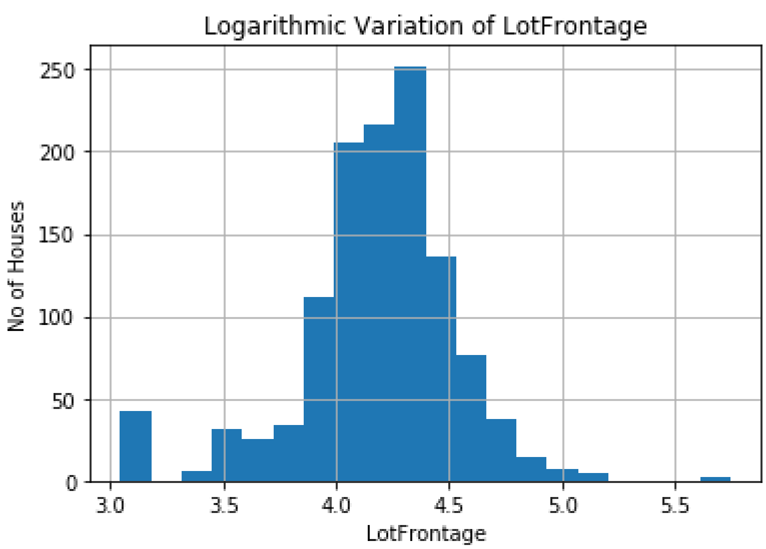


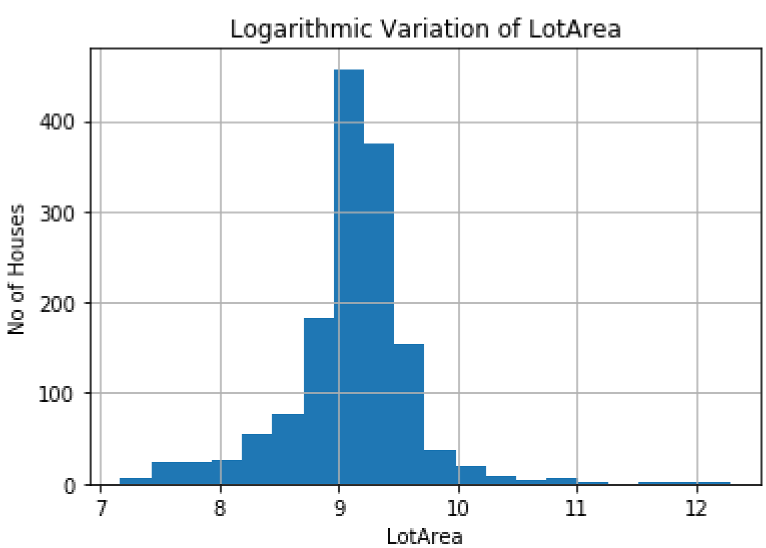
The variables are not normally distributed, including the target variable 'SalePrice'.

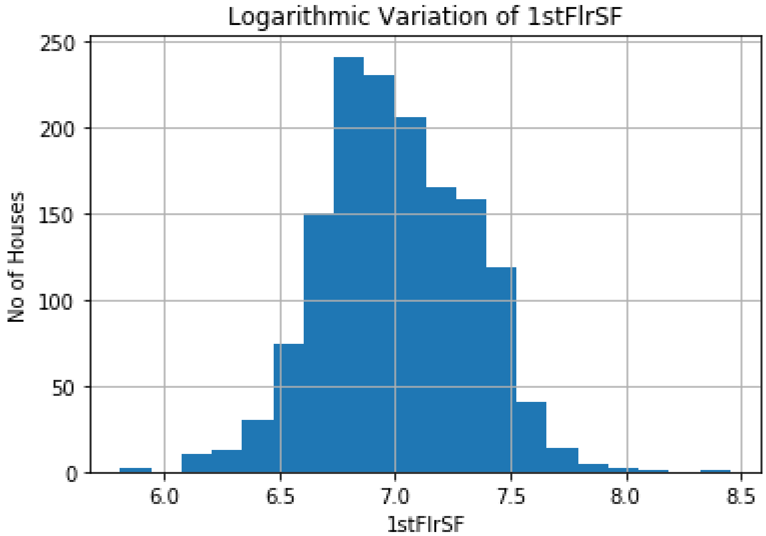
To maximise performance of linear models, we need to account for non-Gaussian distributions. We will transform our variables as part of feature engineering

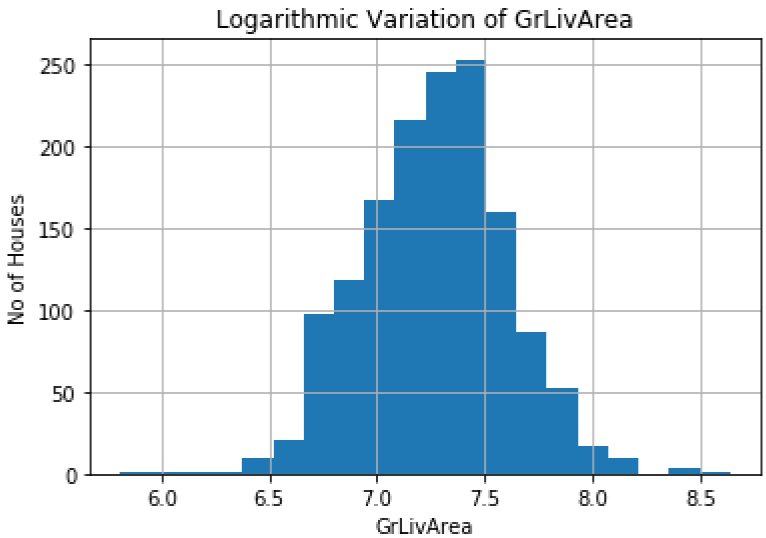
Let's evaluate if a logarithmic transformation of the variables returns values that follow a normal distribution:

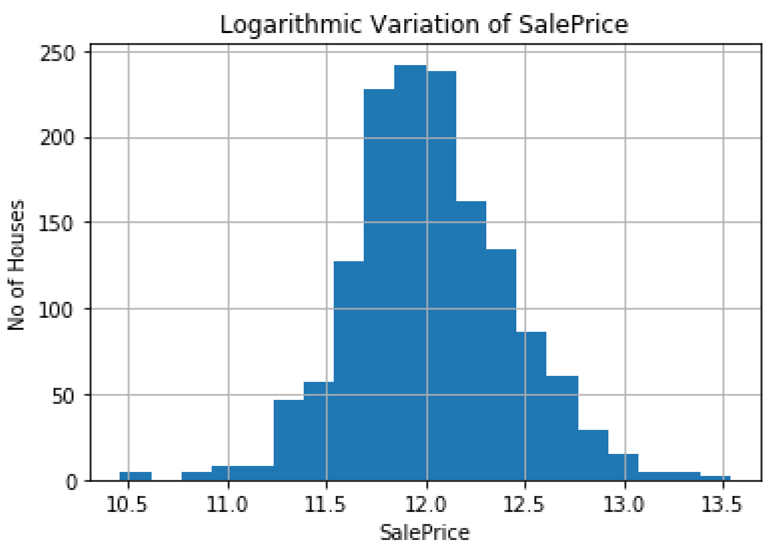
In [50]:
```
def analyze_log_variation(df,var):

    df = df.copy()

    if 0 in df[var].unique():
        pass
    else:
        #log Transformation of variables
        df[var] = np.log(df[var])
        df[var].hist(bins=20)
        plt.xlabel(var)
        plt.ylabel('No of Houses')
        plt.title('Logarithmic Variation of '+ var)
        plt.show()

for var in cont_vars:
    analyze_log_variation(df,var)
```







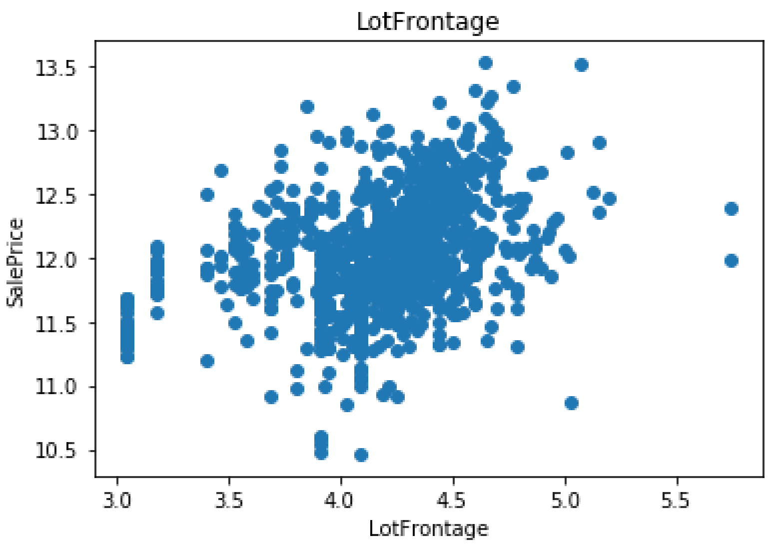


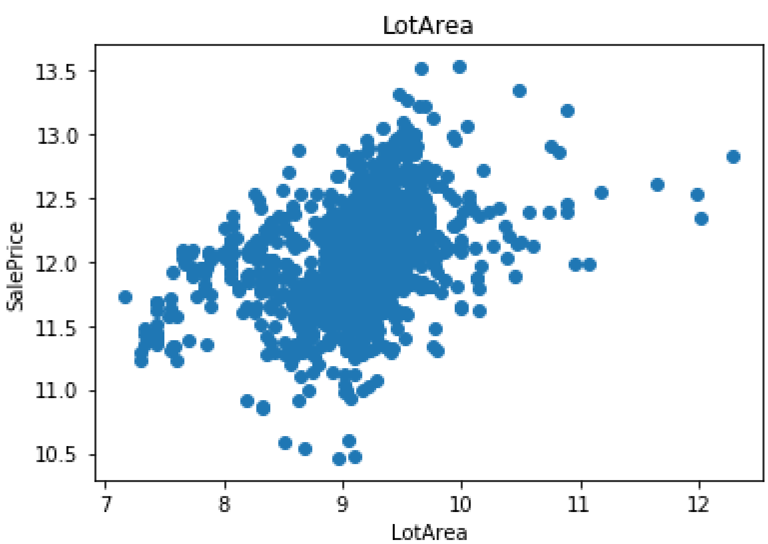


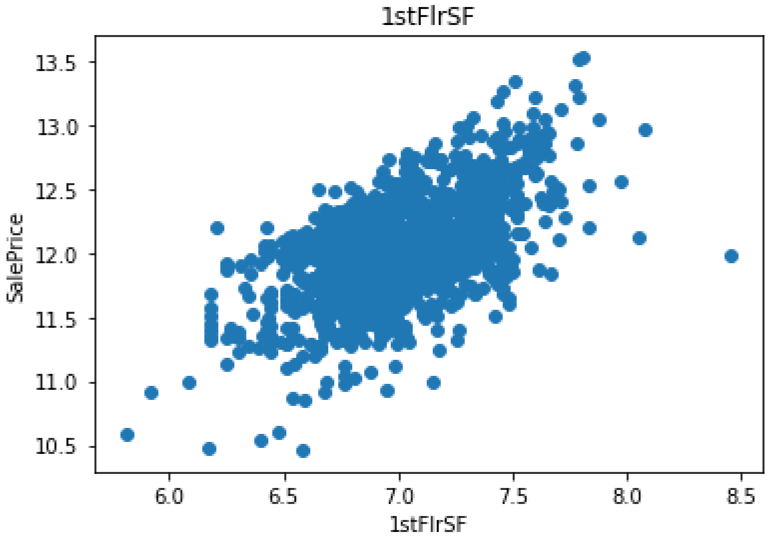
We get a better spread of the values for most variables when we use the logarithmic transformation. This engineering step will most likely add performance value to our final model.

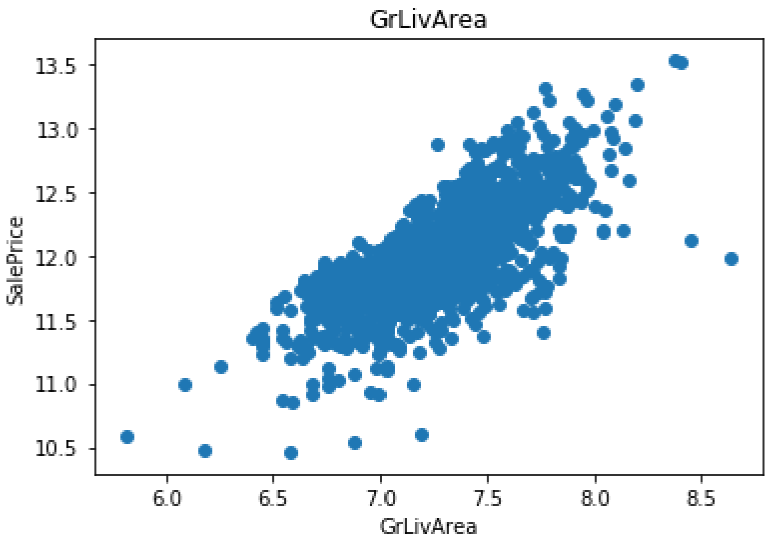
In [51]:
```
def analyze_cont_var_log(df,var):
```

```
    df = df.copy()

    if 0 in df[var].unique():
        pass
    else:
        #Log Transformation of variables
        df[var] = np.log(df[var])
        df['SalePrice'] = np.log(df['SalePrice'])

        plt.scatter(df[var],df['SalePrice'])
        plt.xlabel(var)
        plt.ylabel('SalePrice')
        plt.title(var)
        plt.show()

for var in cont_vars:
    if var!='SalePrice':
        analyze_cont_var_log(df,var)
```



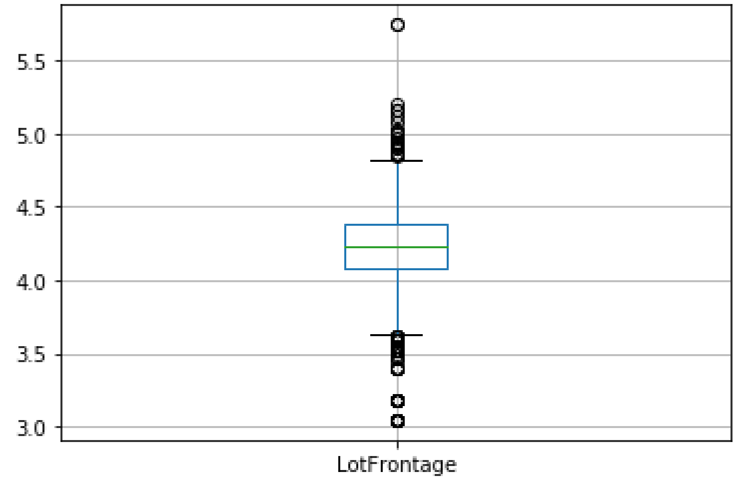


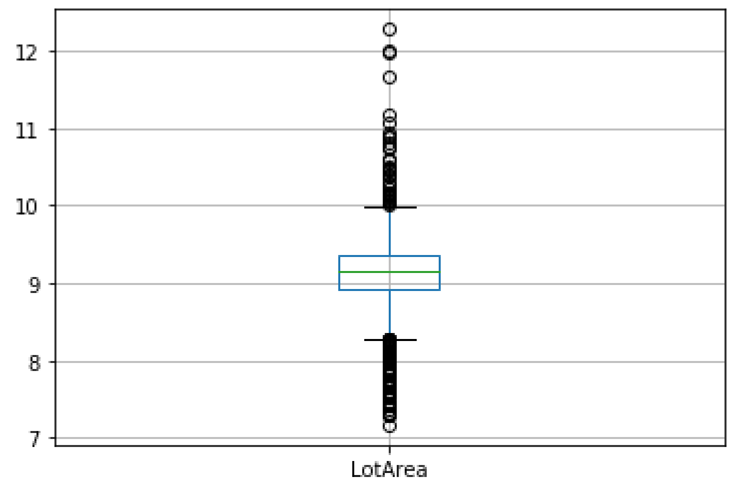


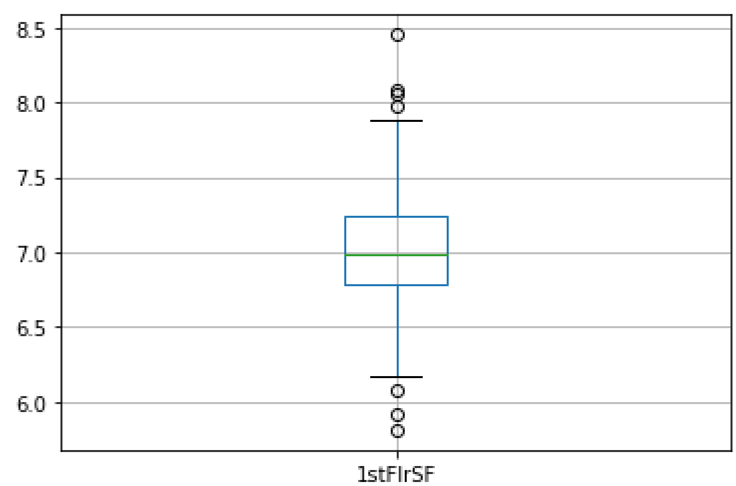


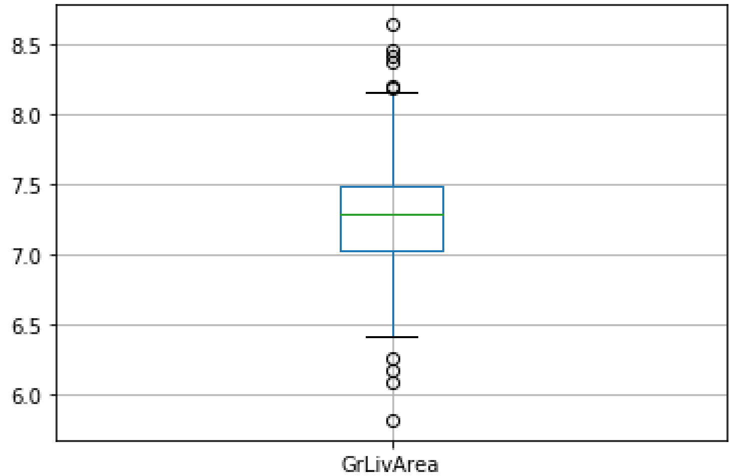
Outliers:

In [52]:
```
def analyze_outliers(df,var):

    df = df.copy()

    if 0 in df[var].unique():
        pass
    else:
        df[var] = np.log(df[var])
        df.boxplot(column=var)
        plt.show()

for var in cont_vars:
    analyze_outliers(df,var)
```


5.5
5.0
4.5
4.0
3.5
3.0
LotFrontage


12
11
10
9
8
7
LotArea


8.5
8.0
7.5
7.0
6.5
6.0
1stFlrSF



Categorical Variables:

In [53]:
```
cat_vars = [var for var in df.columns if df[var].dtypes == 'O']
print('No of categorical variables : ',len(cat_vars))
```

```
No of categorical variables :  43
```

In [54]:
```
df[cat_vars].head()
```

Out[54]:

	MSZoning	Street	Alley	LotShape	LandContour	Utilities	LotConfig	LandSlope	Neighborhood	C
0	RL	Pave	NaN	Reg	Lvl	AllPub	Inside	Gtl	CollgCr	
1	RL	Pave	NaN	Reg	Lvl	AllPub	FR2	Gtl	Veenker	
2	RL	Pave	NaN	IR1	Lvl	AllPub	Inside	Gtl	CollgCr	
3	RL	Pave	NaN	IR1	Lvl	AllPub	Corner	Gtl	Crawfor	
4	RL	Pave	NaN	IR1	Lvl	AllPub	FR2	Gtl	NoRidge	

Check cardinality of categorical variables:

In [56]:
```
for var in cat_vars:
    print(var,len(df[var].unique()),' Categories')
```

```
MSZoning 5  Categories
```

```
Street 2  Categories
Alley 3  Categories
LotShape 4  Categories
LandContour 4  Categories
Utilities 2  Categories
LotConfig 5  Categories
LandSlope 3  Categories
Neighborhood 25  Categories
Condition1 9  Categories
Condition2 8  Categories
BldgType 5  Categories
HouseStyle 8  Categories
RoofStyle 6  Categories
RoofMatl 8  Categories
Exterior1st 15  Categories
Exterior2nd 16  Categories
MasVnrType 5  Categories
ExterQual 4  Categories
ExterCond 5  Categories
Foundation 6  Categories
BsmtQual 5  Categories
BsmtCond 5  Categories
BsmtExposure 5  Categories
BsmtFinType1 7  Categories
BsmtFinType2 7  Categories
Heating 6  Categories
HeatingQC 5  Categories
CentralAir 2  Categories
Electrical 6  Categories
KitchenQual 4  Categories
Functional 7  Categories
FireplaceQu 6  Categories
GarageType 7  Categories
GarageFinish 4  Categories
GarageQual 6  Categories
GarageCond 6  Categories
PavedDrive 3  Categories
PoolQC 4  Categories
Fence 5  Categories
MiscFeature 5  Categories
SaleType 9  Categories
SaleCondition 6  Categories
```

All the categorical variables show low cardinality, this means that they have only few different labels. That is good as we won't need to tackle cardinality during our feature engineering lecture.

Rare Labels:

In [58]:

```
def analyze_rare_labels(df, var, rare_perc):

    df = df.copy()

    # determine the % of observations per category
    tmp = df.groupby(var)['SalePrice'].count() / len(df)

    # return categories that are rare
    return tmp[tmp < rare_perc]

# print categories that are present in less than
# 1 % of the observations
```

```
for var in cat_vars:
    print(analyze_rare_labels(df, var, 0.01))
    print()
```

```
MSZoning
C (all)    0.006849
Name: SalePrice, dtype: float64

Street
Grvl    0.00411
Name: SalePrice, dtype: float64

Series([], Name: SalePrice, dtype: float64)

LotShape
IR3    0.006849
Name: SalePrice, dtype: float64

Series([], Name: SalePrice, dtype: float64)

Utilities
NoSeWa    0.000685
Name: SalePrice, dtype: float64

LotConfig
FR3    0.00274
Name: SalePrice, dtype: float64

LandSlope
Sev    0.008904
Name: SalePrice, dtype: float64

Neighborhood
Blueste    0.001370
NPkVill    0.006164
Veenker    0.007534
Name: SalePrice, dtype: float64

Condition1
PosA    0.005479
RRAe    0.007534
RRNe    0.001370
RRNn    0.003425
Name: SalePrice, dtype: float64

Condition2
Artery    0.001370
Feedr     0.004110
PosA      0.000685
PosN      0.001370
RRAe      0.000685
RRAn      0.000685
RRNn      0.001370
Name: SalePrice, dtype: float64

Series([], Name: SalePrice, dtype: float64)

HouseStyle
1.5Unf    0.009589
2.5Fin    0.005479
2.5Unf    0.007534
Name: SalePrice, dtype: float64

RoofStyle
```

```
Flat       0.008904
Gambrel    0.007534
Mansard    0.004795
Shed       0.001370
Name: SalePrice, dtype: float64

RoofMatl
ClyTile    0.000685
Membran    0.000685
Metal      0.000685
Roll       0.000685
Tar&Grv    0.007534
WdShake    0.003425
WdShngl    0.004110
Name: SalePrice, dtype: float64

Exterior1st
AsphShn    0.000685
BrkComm    0.001370
CBlock     0.000685
ImStucc    0.000685
Stone      0.001370
Name: SalePrice, dtype: float64

Exterior2nd
AsphShn    0.002055
Brk Cmn    0.004795
CBlock     0.000685
ImStucc    0.006849
Other      0.000685
Stone      0.003425
Name: SalePrice, dtype: float64

Series([], Name: SalePrice, dtype: float64)

ExterQual
Fa    0.009589
Name: SalePrice, dtype: float64

ExterCond
Ex    0.002055
Po    0.000685
Name: SalePrice, dtype: float64

Foundation
Stone    0.004110
Wood     0.002055
Name: SalePrice, dtype: float64

Series([], Name: SalePrice, dtype: float64)

BsmtCond
Po    0.00137
Name: SalePrice, dtype: float64

Series([], Name: SalePrice, dtype: float64)

Series([], Name: SalePrice, dtype: float64)

BsmtFinType2
GLQ    0.009589
Name: SalePrice, dtype: float64

Heating
Floor    0.000685
```

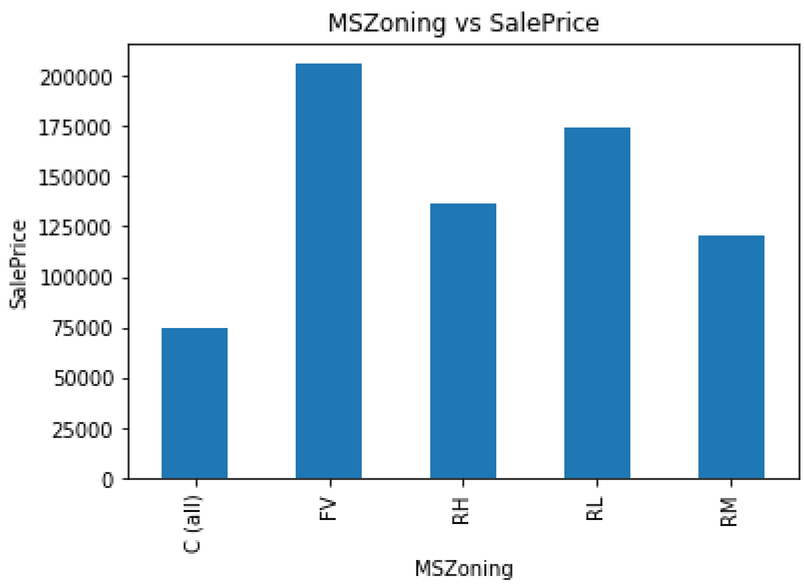
```
Grav     0.004795
OthW     0.001370
Wall     0.002740
Name: SalePrice, dtype: float64

HeatingQC
Po    0.000685
Name: SalePrice, dtype: float64

Series([], Name: SalePrice, dtype: float64)

Electrical
FuseP    0.002055
Mix      0.000685
Name: SalePrice, dtype: float64

Series([], Name: SalePrice, dtype: float64)

Functional
Maj1    0.009589
Maj2    0.003425
Sev     0.000685
Name: SalePrice, dtype: float64

Series([], Name: SalePrice, dtype: float64)

GarageType
2Types     0.004110
CarPort    0.006164
Name: SalePrice, dtype: float64

Series([], Name: SalePrice, dtype: float64)

GarageQual
Ex    0.002055
Gd    0.009589
Po    0.002055
Name: SalePrice, dtype: float64

GarageCond
Ex    0.001370
Gd    0.006164
Po    0.004795
Name: SalePrice, dtype: float64

Series([], Name: SalePrice, dtype: float64)

PoolQC
Ex    0.001370
Fa    0.001370
Gd    0.002055
Name: SalePrice, dtype: float64

Fence
MnWw    0.007534
Name: SalePrice, dtype: float64

MiscFeature
Gar2    0.001370
Othr    0.001370
TenC    0.000685
Name: SalePrice, dtype: float64

SaleType
CWD      0.002740
```

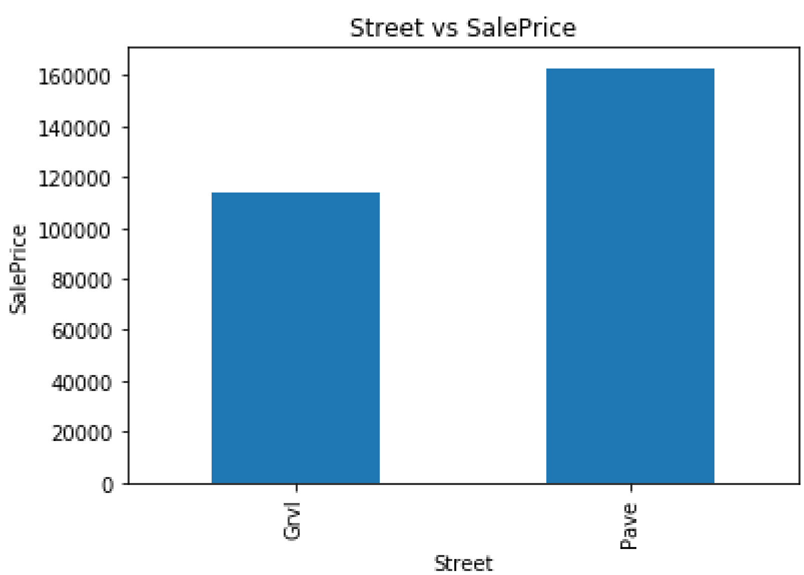
```
Con       0.001370
ConLD     0.006164
ConLI     0.003425
ConLw     0.003425
Oth       0.002055
Name: SalePrice, dtype: float64

SaleCondition
AdjLand    0.002740
Alloca     0.008219
Name: SalePrice, dtype: float64
```

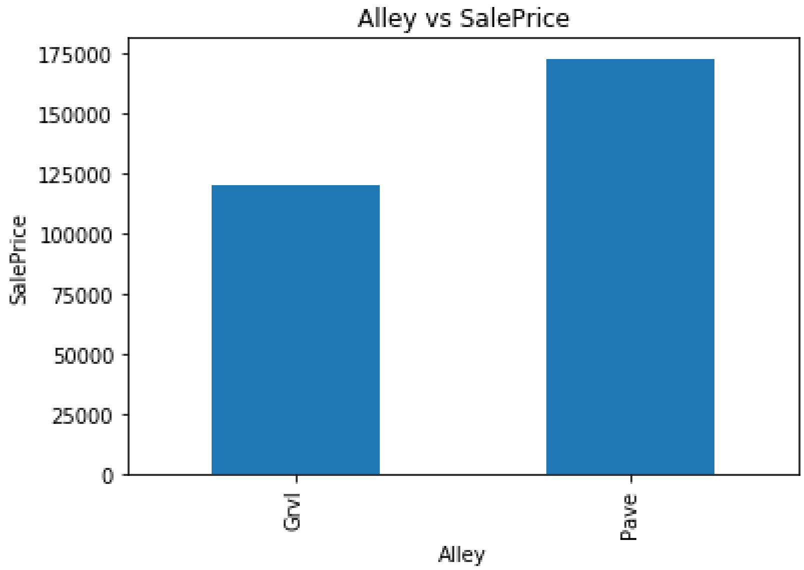
Some of the categorical variables show multiple labels that are present in less than 1% of the houses. We will have to perform feature engineering on them. Labels that are under-represented in the dataset tend to cause over-fitting of machine learning models. That is why we want to remove them.

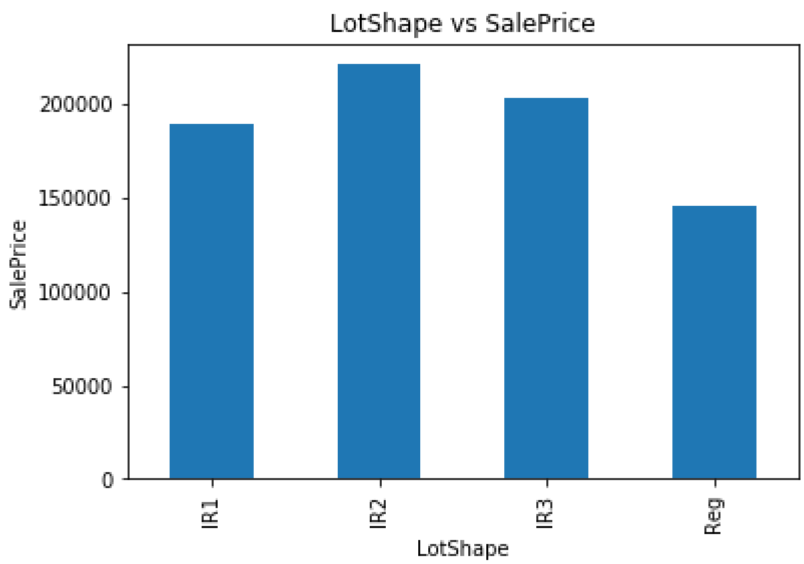
Relationship between categorical variables and house price

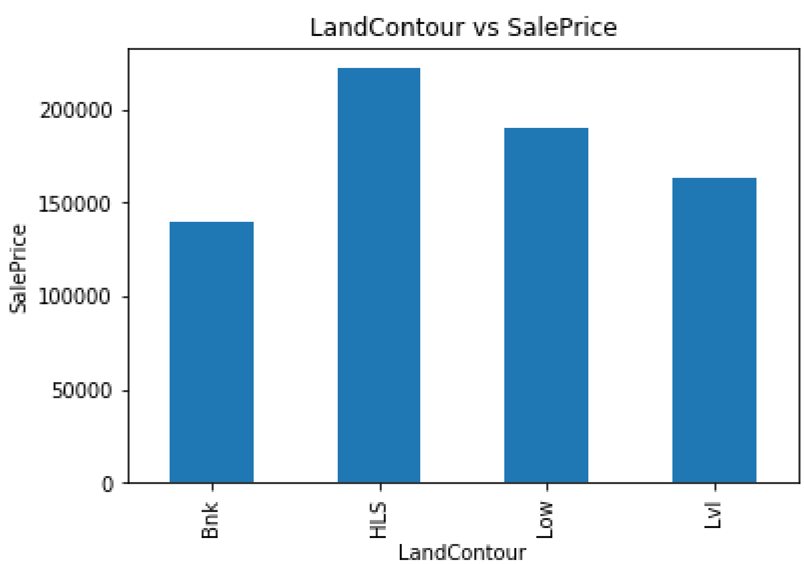
In [59]:
```
for var in cat_vars:
    analyze_discrete_vars(df,var)
```

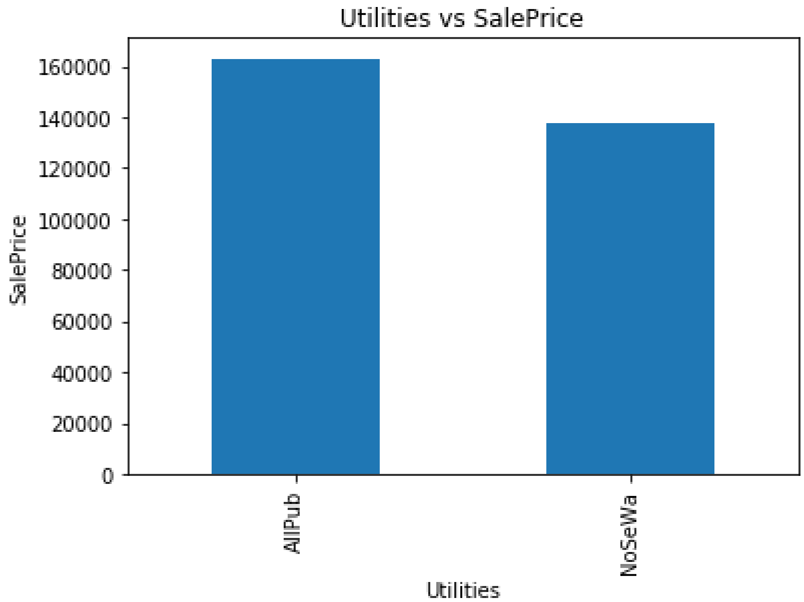


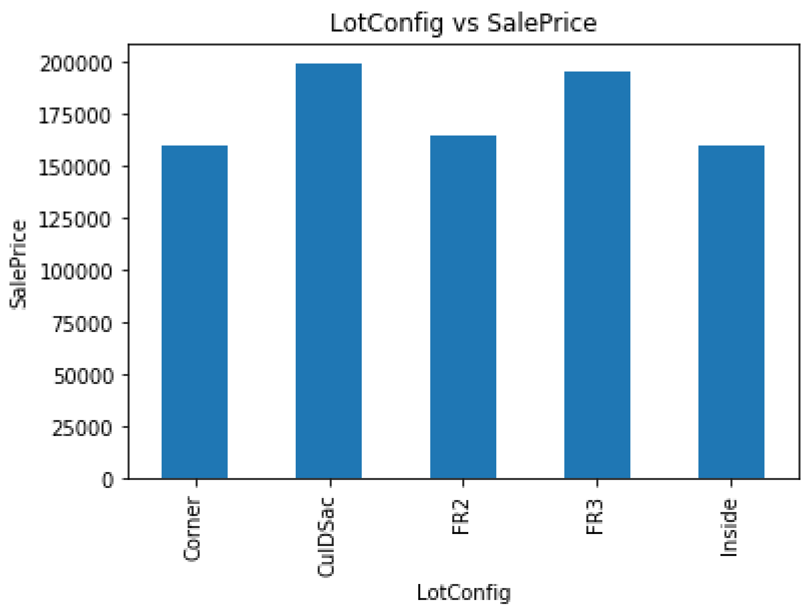


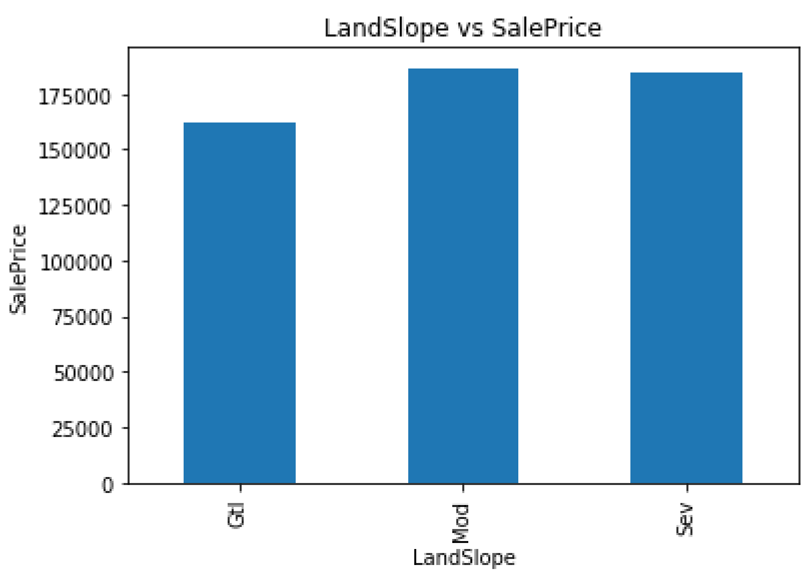

Alley vs SalePrice
SalePrice
175000
150000
125000
100000
75000
50000
25000
0
Grvl
Pave
Alley

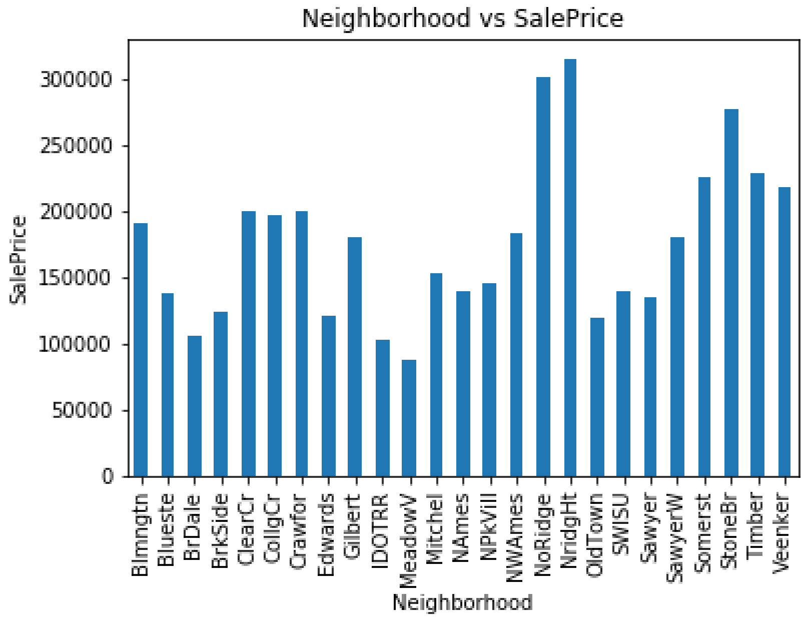

LotShape vs SalePrice
SalePrice
200000
150000
100000
50000
0
IR1
IR2
IR3
Reg
LotShape

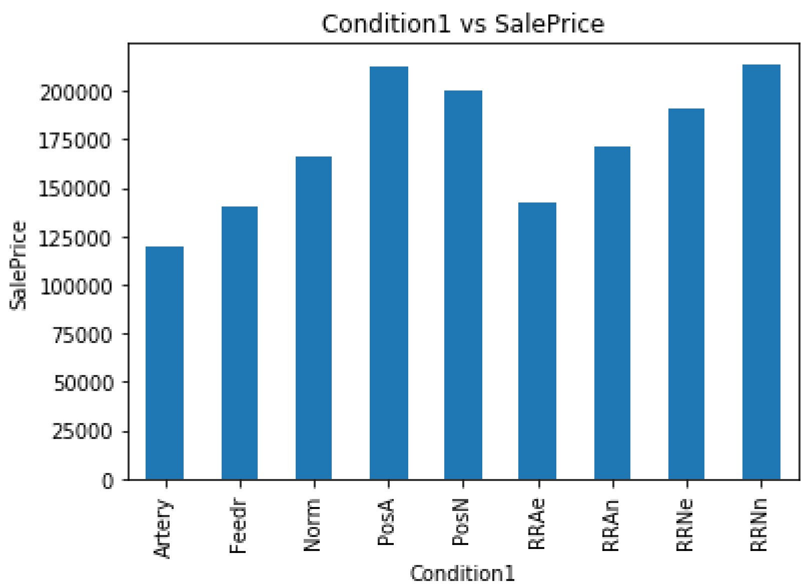

LandContour vs SalePrice
SalePrice
200000
150000
100000
50000
0
Bnk
HLS
Low
Lvl
LandContour

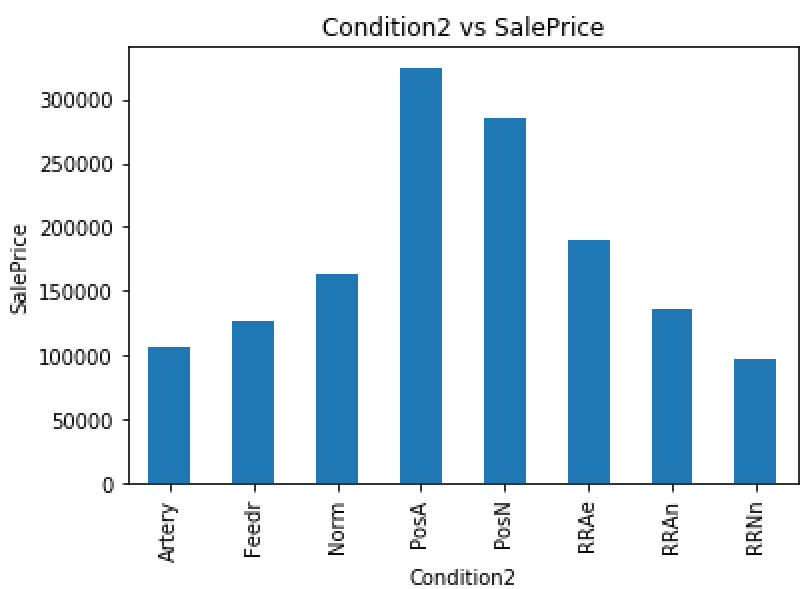

Utilities vs SalePrice
SalePrice
160000
140000
120000
100000
80000
60000
40000
20000
0
AllPub
NoSeWa
Utilities

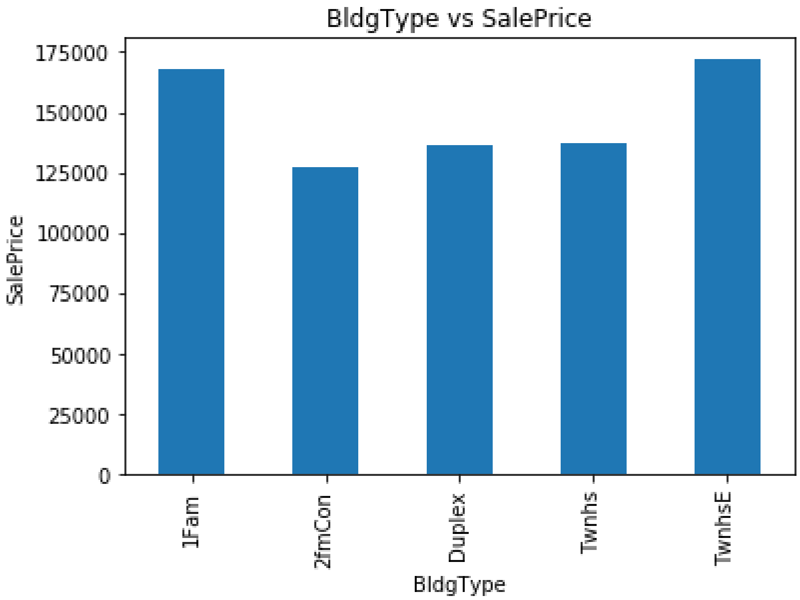

LotConfig vs SalePrice
SalePrice
200000
175000
150000
125000
100000
75000
50000
25000
0
Corner
CulDSac
FR2
FR3
Inside
LotConfig

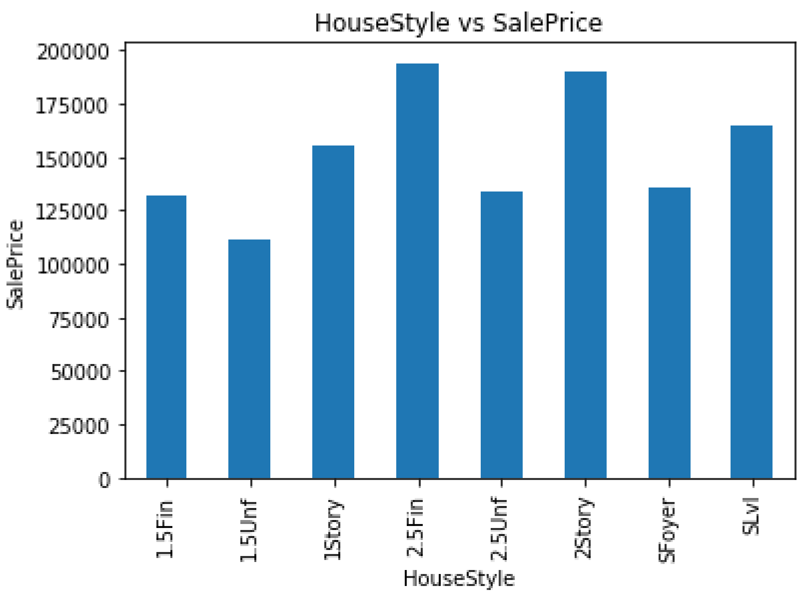

LandSlope vs SalePrice
SalePrice
175000
150000
125000
100000
75000
50000
25000
0
Gtl
Mod
Sev
LandSlope

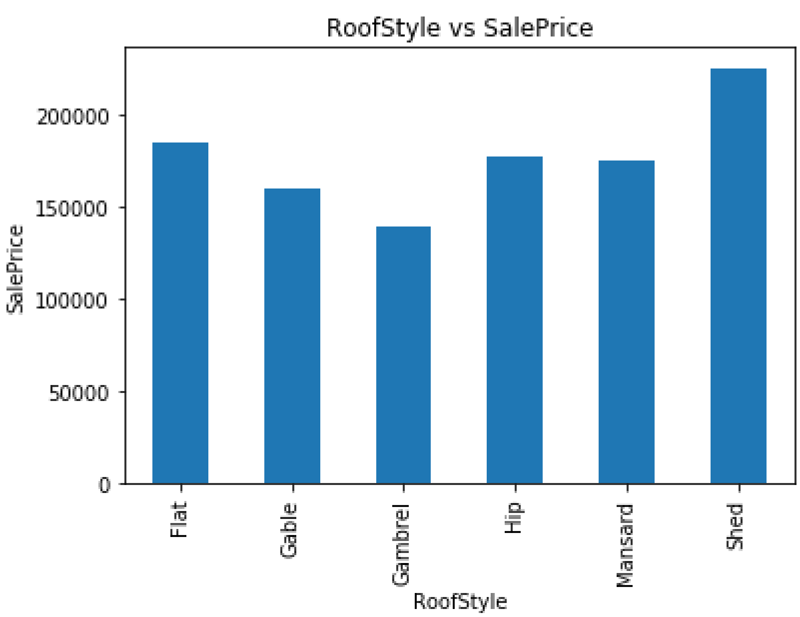

Neighborhood vs SalePrice
SalePrice
0
50000
100000
150000
200000
250000
300000
Blmngtn
Blueste
BrDale
BrkSide
ClearCr
CollgCr
Crawfor
Edwards
Gilbert
IDOTRR
MeadowV
Mitchel
NAmes
NPkVill
NWAmes
NoRidge
NridgHt
OldTown
SWISU
Sawyer
SawyerW
Somerst
StoneBr
Timber
Veenker
Neighborhood

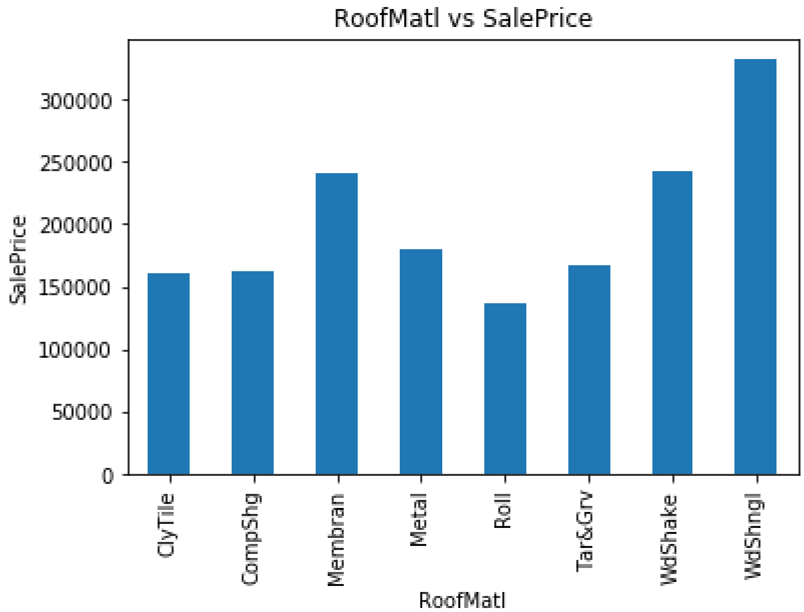

Condition1 vs SalePrice
SalePrice
0
25000
50000
75000
100000
125000
150000
175000
200000
Artery
Feedr
Norm
PosA
PosN
RRAe
RRAn
RRNe
RRNn
Condition1

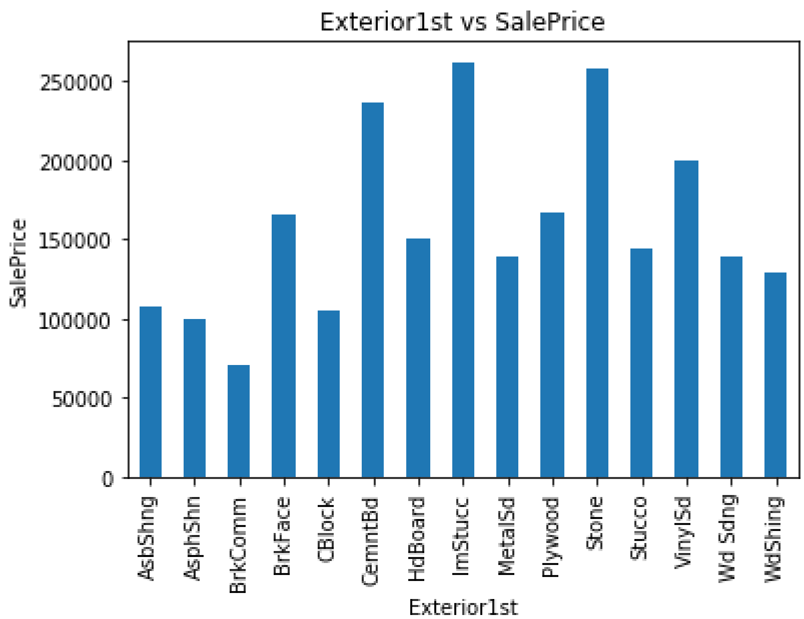

Condition2 vs SalePrice
SalePrice
0
50000
100000
150000
200000
250000
300000
Artery
Feedr
Norm
PosA
PosN
RRAe
RRAn
RRNn
Condition2

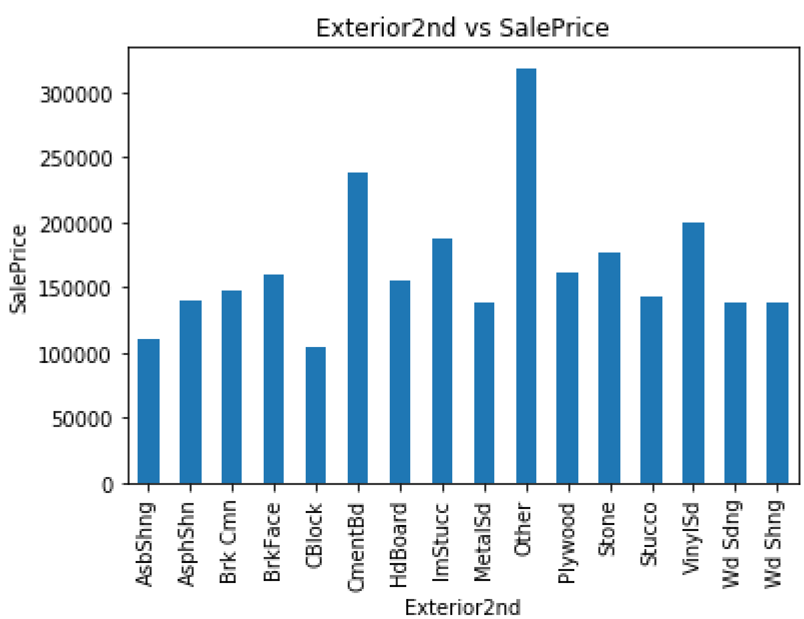

BldgType vs SalePrice
SalePrice
175000
150000
125000
100000
75000
50000
25000
0
1Fam
2fmCon
Duplex
Twnhs
TwnhsE
BldgType

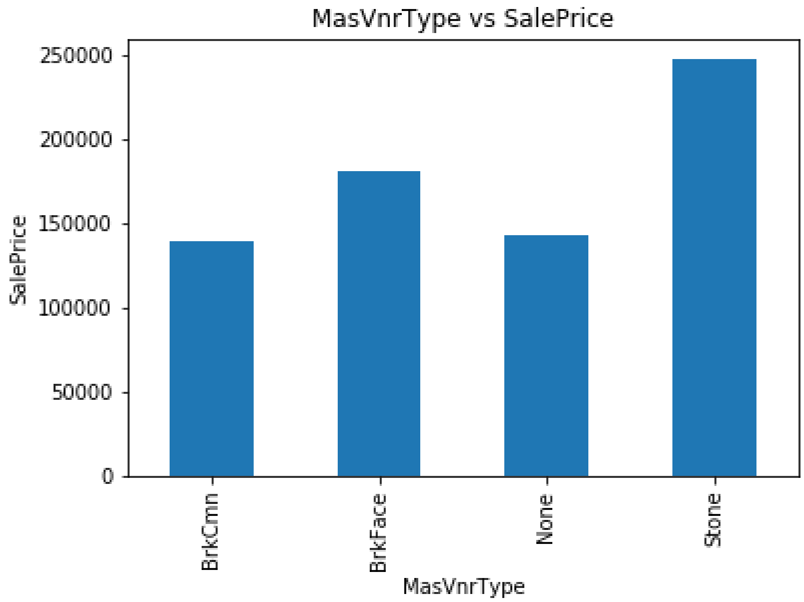

HouseStyle vs SalePrice
SalePrice
200000
175000
150000
125000
100000
75000
50000
25000
0
1.5Fin
1.5Unf
1Story
2.5Fin
2.5Unf
2Story
SFoyer
SLvl
HouseStyle

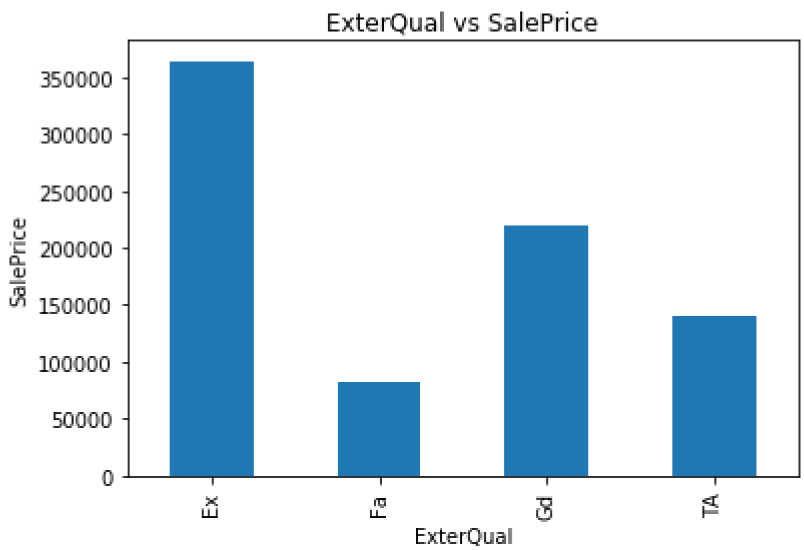

RoofStyle vs SalePrice
SalePrice
200000
150000
100000
50000
0
Flat
Gable
Gambrel
Hip
Mansard
Shed
RoofStyle

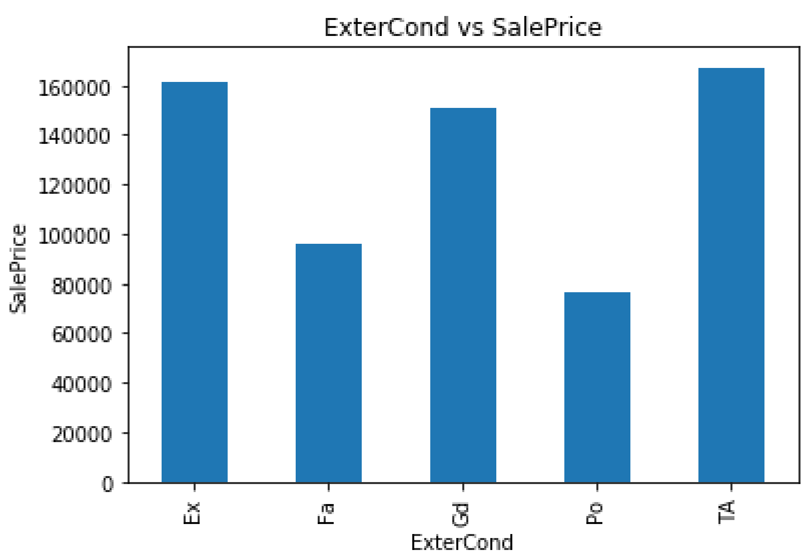

RoofMatl vs SalePrice
SalePrice
300000
250000
200000
150000
100000
50000
0
ClyTile
CompShg
Membran
Metal
Roll
Tar&Grv
WdShake
WdShngl
RoofMatl

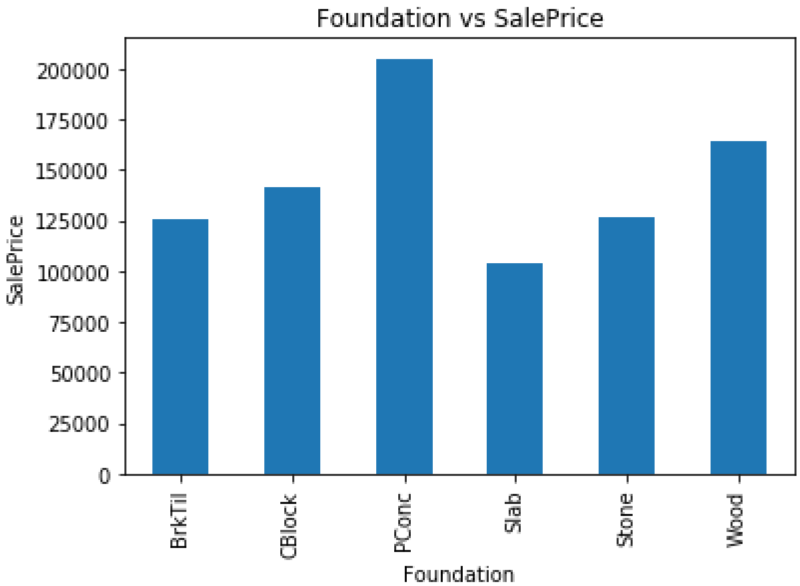

Exterior1st vs SalePrice
SalePrice
250000
200000
150000
100000
50000
0
AsbShng
AsphShn
BrkComm
BrkFace
CBlock
CemntBd
HdBoard
ImStucc
MetalSd
Plywood
Stone
Stucco
VinylSd
Wd Sdng
WdShing
Exterior1st

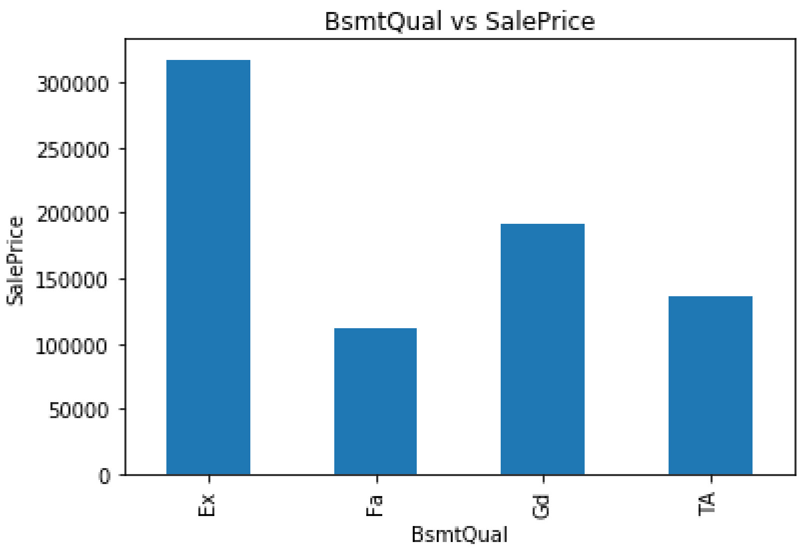

Exterior2nd vs SalePrice
SalePrice
300000
250000
200000
150000
100000
50000
0
AsbShng
AsphShn
Brk Cmn
BrkFace
CBlock
CmentBd
HdBoard
ImStucc
MetalSd
Other
Plywood
Stone
Stucco
VinylSd
Wd Sdng
Wd Shng
Exterior2nd

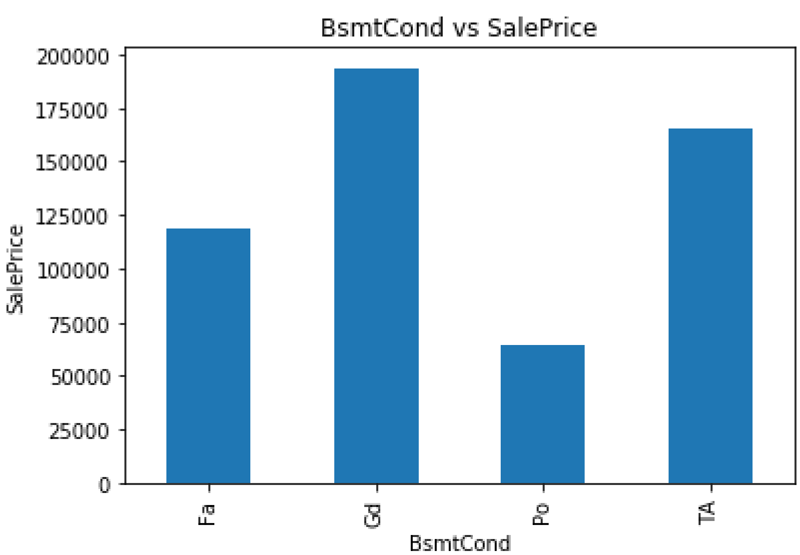

MasVnrType vs SalePrice
SalePrice
250000
200000
150000
100000
50000
0
BrkCmn
BrkFace
None
Stone
MasVnrType

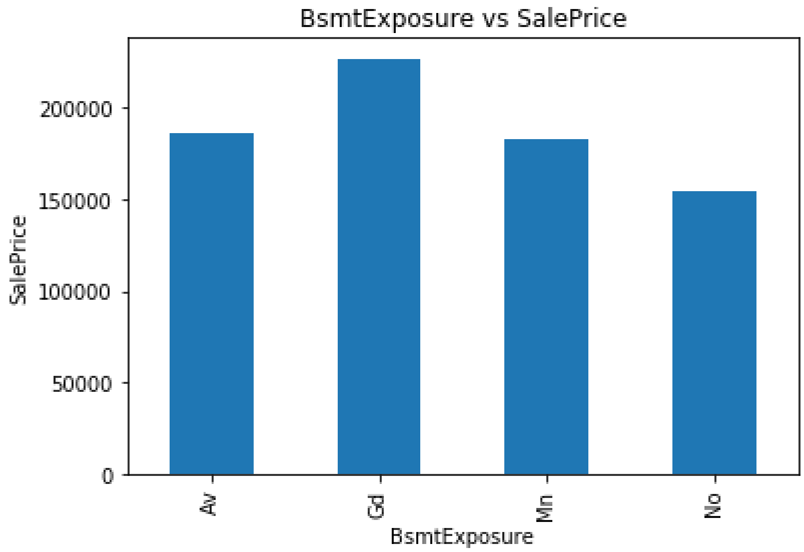

ExterQual vs SalePrice
SalePrice
350000
300000
250000
200000
150000
100000
50000
0
Ex
Fa
Gd
TA
ExterQual

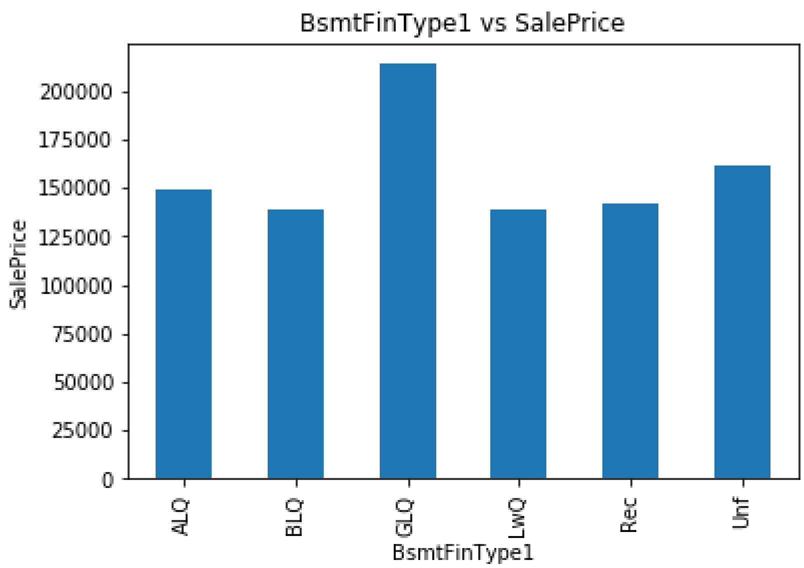

ExterCond vs SalePrice
SalePrice
160000
140000
120000
100000
80000
60000
40000
20000
0
Ex
Fa
Gd
Po
TA
ExterCond

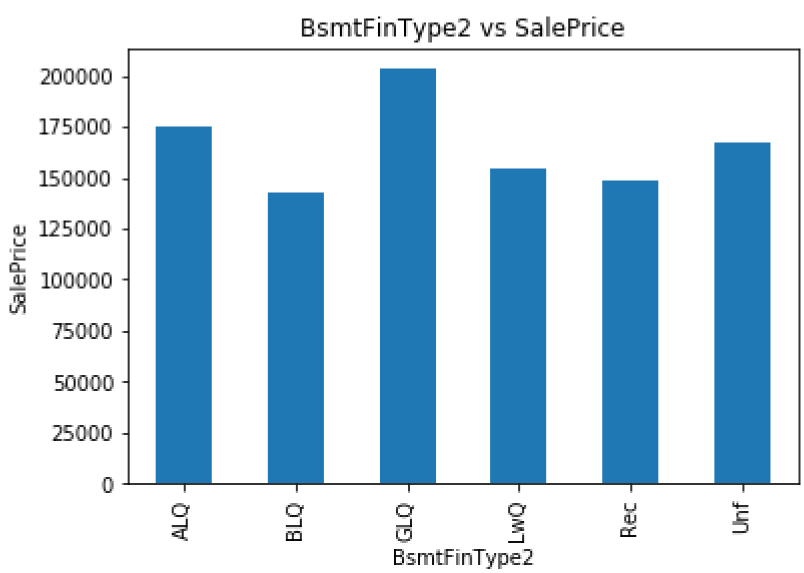

Foundation vs SalePrice
SalePrice
200000
175000
150000
125000
100000
75000
50000
25000
0
BrkTil
CBlock
PConc
Slab
Stone
Wood
Foundation

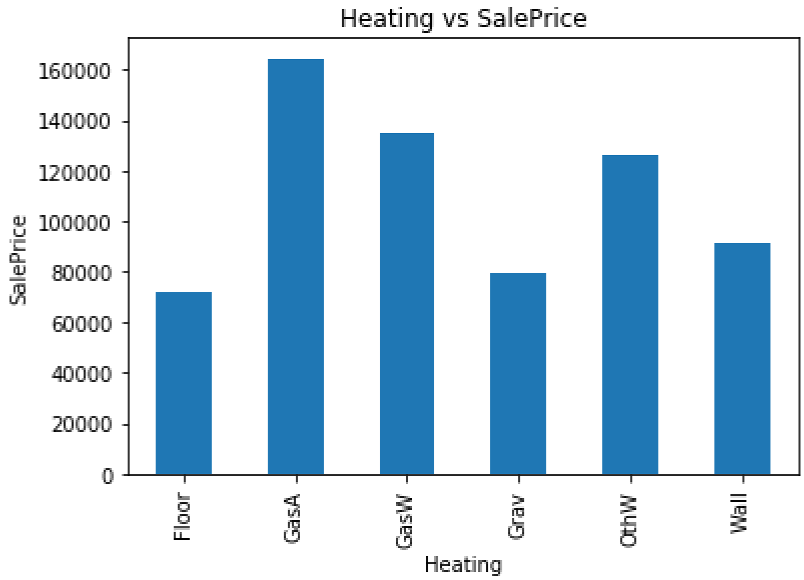

BsmtQual vs SalePrice
SalePrice
300000
250000
200000
150000
100000
50000
0
Ex
Fa
Gd
TA
BsmtQual

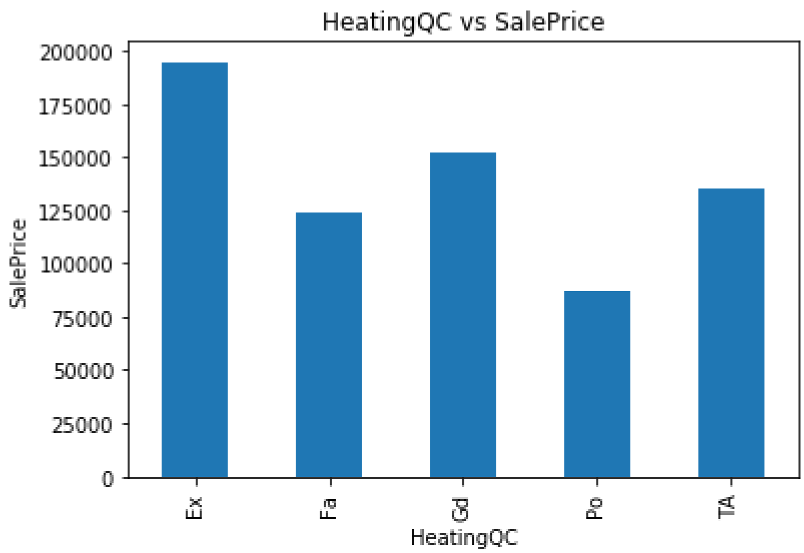

BsmtCond vs SalePrice
SalePrice
200000
175000
150000
125000
100000
75000
50000
25000
0
Fa
Gd
Po
TA
BsmtCond

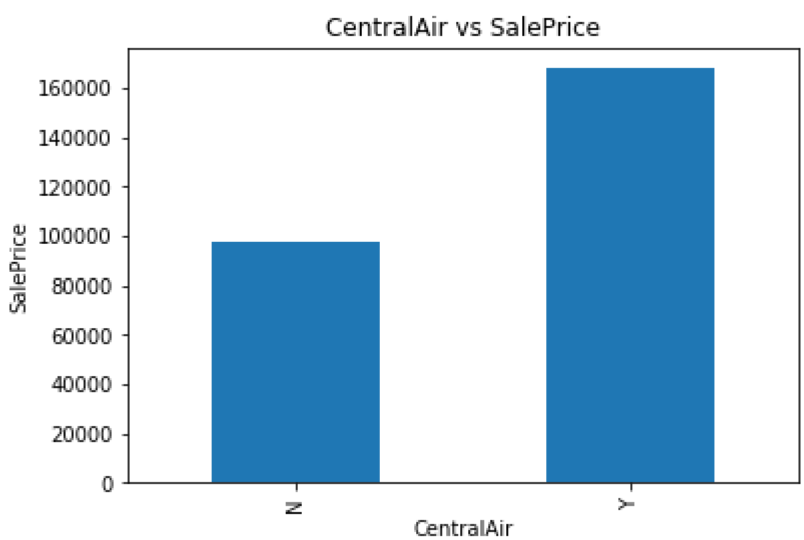

BsmtExposure vs SalePrice
SalePrice
0
50000
100000
150000
200000
Av
Gd
Mn
No
BsmtExposure

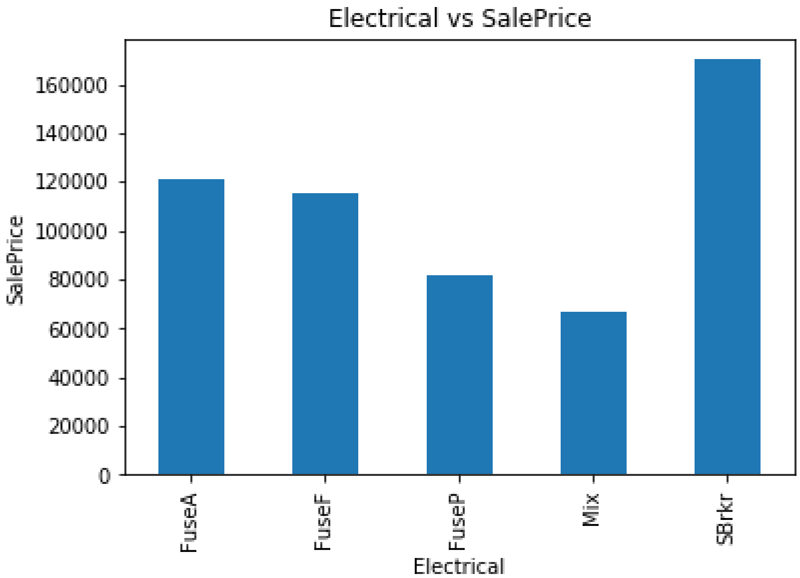

BsmtFinType1 vs SalePrice
SalePrice
0
25000
50000
75000
100000
125000
150000
175000
200000
ALQ
BLQ
GLQ
LwQ
Rec
Unf
BsmtFinType1

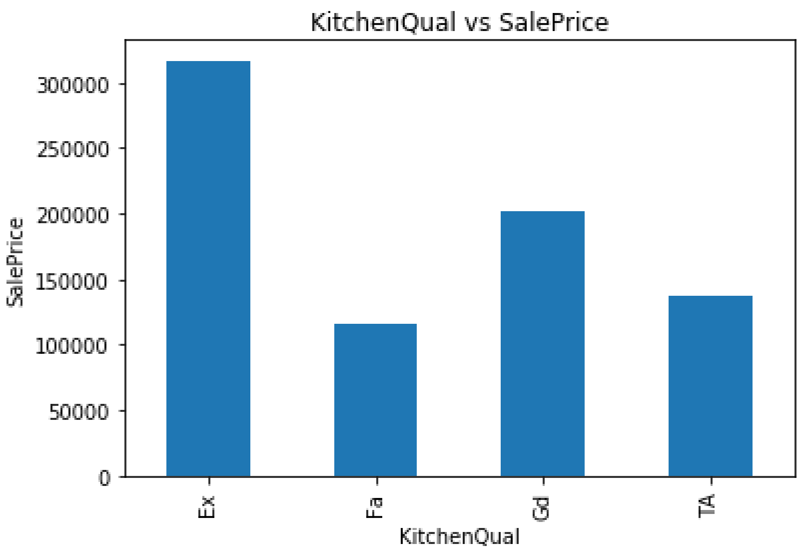

BsmtFinType2 vs SalePrice
SalePrice
0
25000
50000
75000
100000
125000
150000
175000
200000
ALQ
BLQ
GLQ
LwQ
Rec
Unf
BsmtFinType2

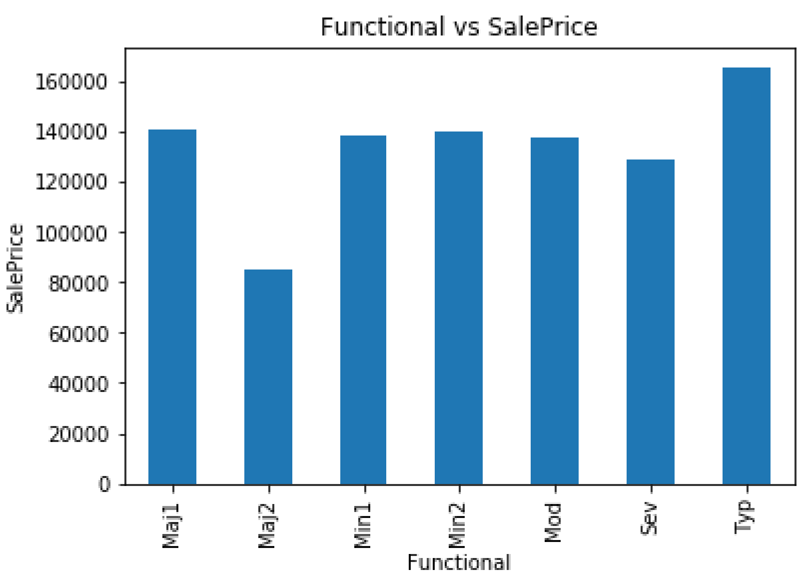

Heating vs SalePrice
SalePrice
0
20000
40000
60000
80000
100000
120000
140000
160000
Floor
GasA
GasW
Grav
OthW
Wall
Heating

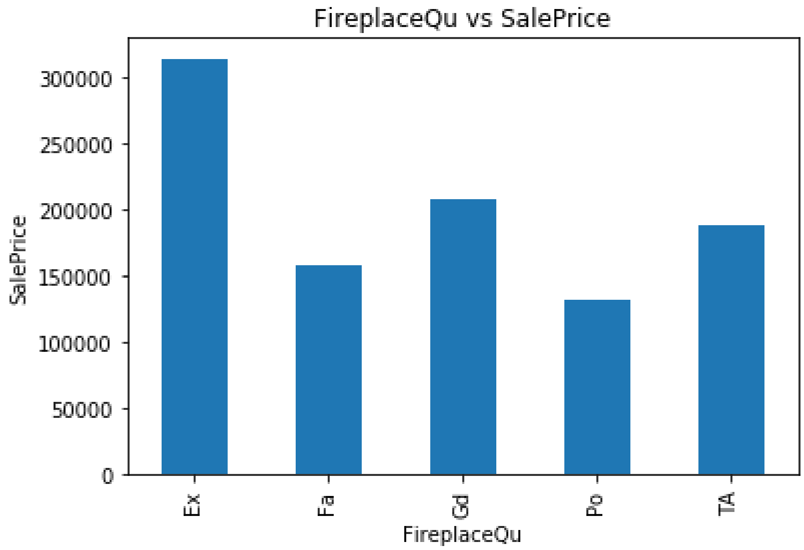

HeatingQC vs SalePrice
SalePrice
0
25000
50000
75000
100000
125000
150000
175000
200000
Ex
Fa
Gd
Po
TA
HeatingQC

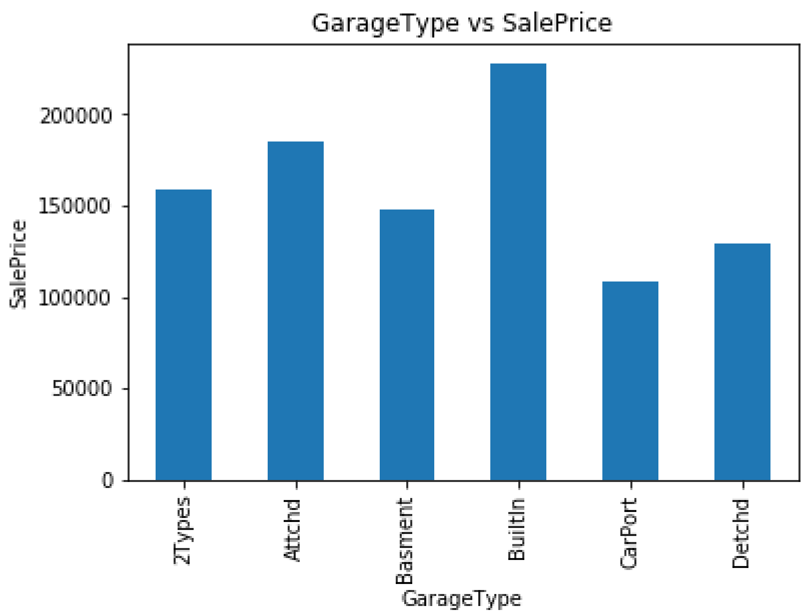

CentralAir vs SalePrice
SalePrice
0
20000
40000
60000
80000
100000
120000
140000
160000
N
Y
CentralAir

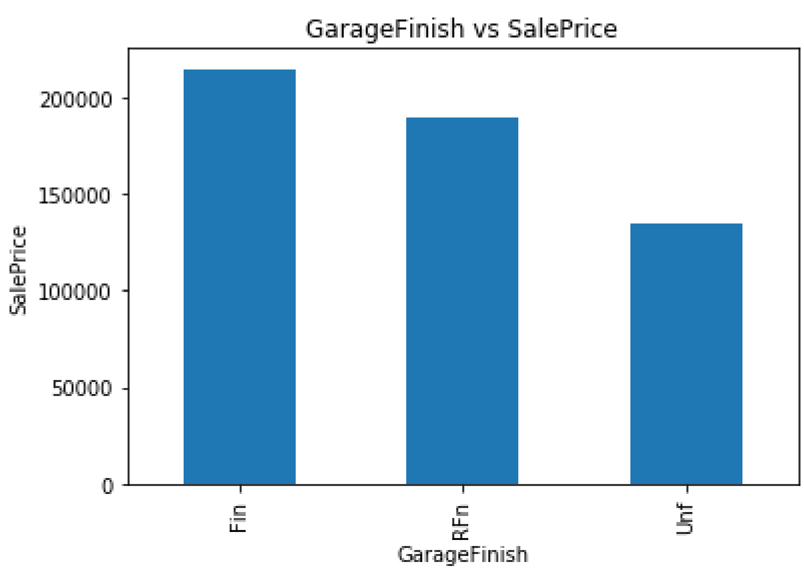

Electrical vs SalePrice
SalePrice
160000
140000
120000
100000
80000
60000
40000
20000
0
FuseA
FuseF
FuseP
Mix
SBrkr
Electrical

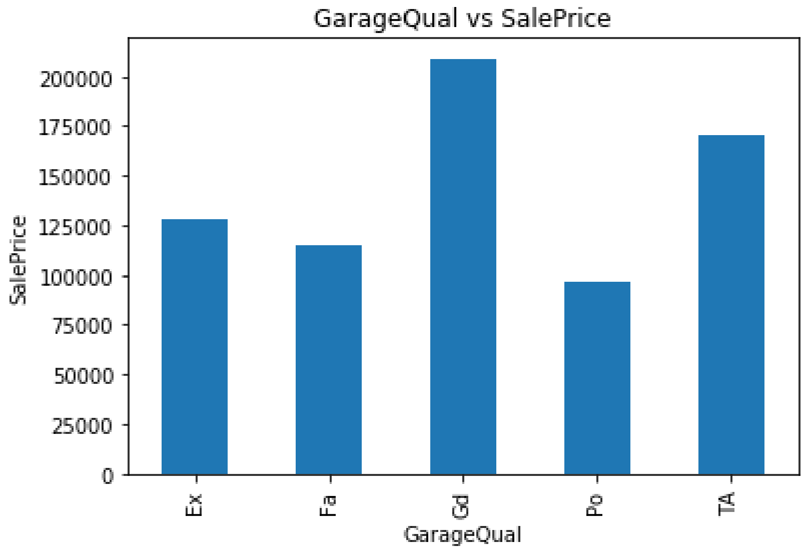

KitchenQual vs SalePrice
SalePrice
300000
250000
200000
150000
100000
50000
0
Ex
Fa
Gd
TA
KitchenQual

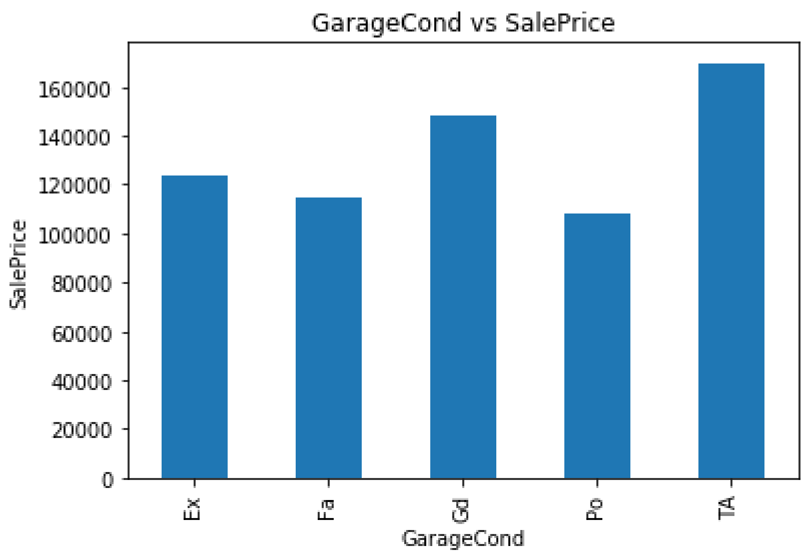

Functional vs SalePrice
SalePrice
160000
140000
120000
100000
80000
60000
40000
20000
0
Maj1
Maj2
Min1
Min2
Mod
Sev
Typ
Functional

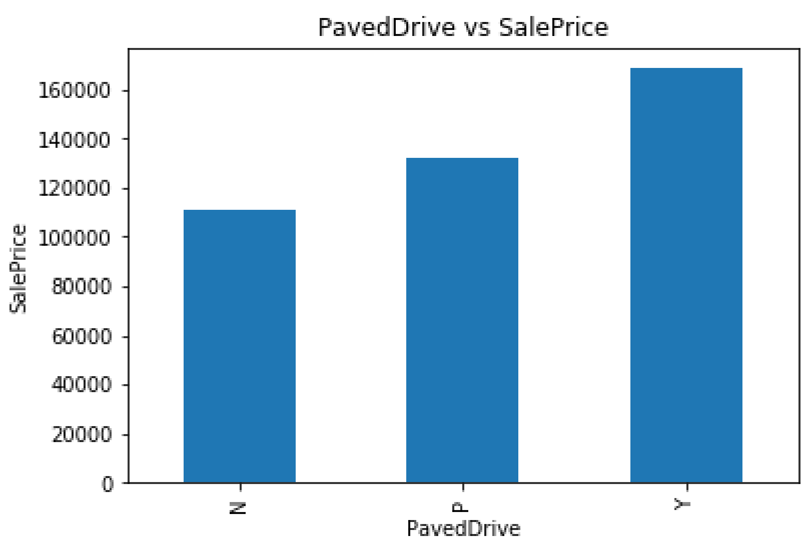

FireplaceQu vs SalePrice
300000
250000
200000
150000
100000
50000
0
SalePrice
Ex
Fa
Gd
Po
TA
FireplaceQu

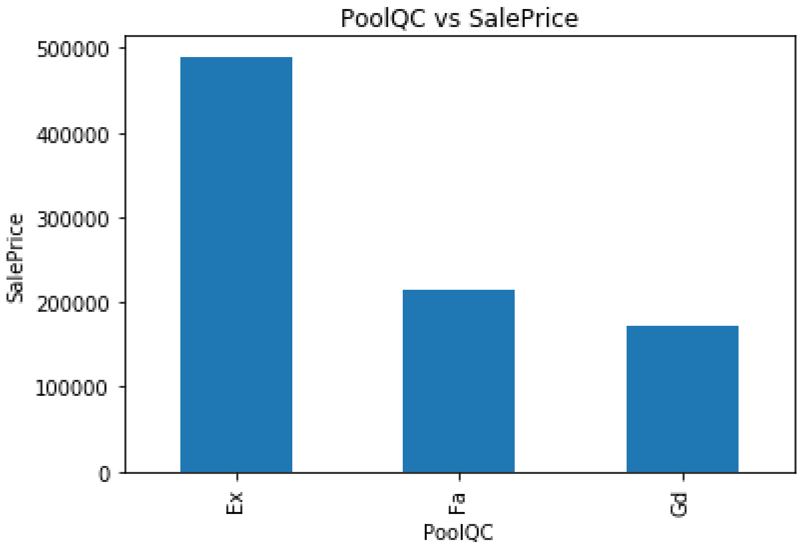

GarageType vs SalePrice
200000
150000
100000
50000
0
SalePrice
2Types
Attchd
Basment
BuiltIn
CarPort
Detchd
GarageType

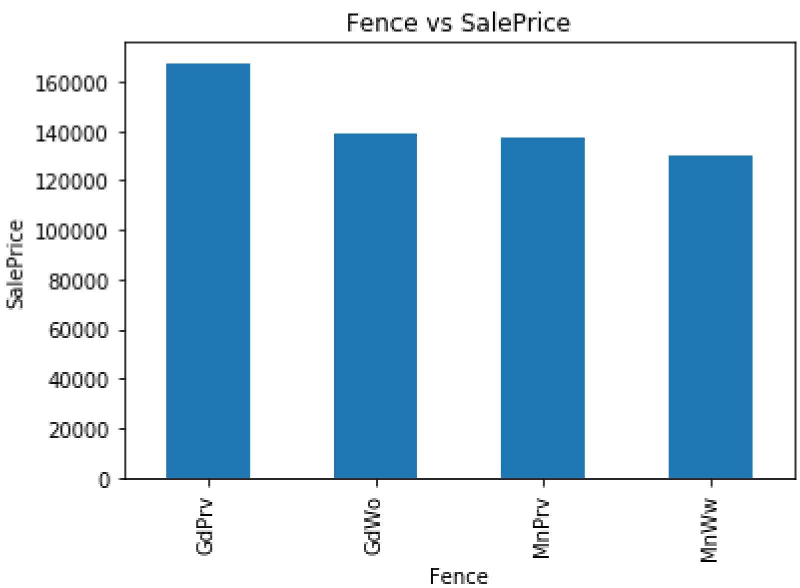

GarageFinish vs SalePrice
200000
150000
100000
50000
0
SalePrice
Fin
RFn
Unf
GarageFinish

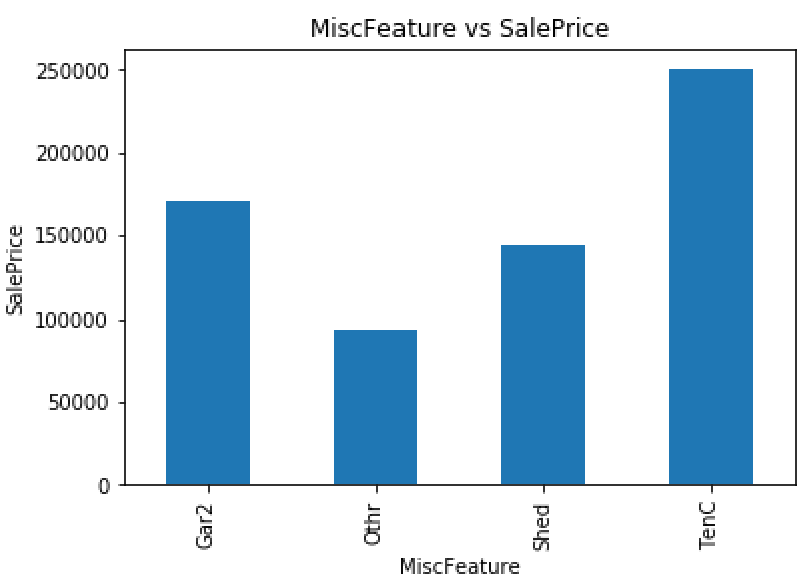

GarageQual vs SalePrice
SalePrice
200000
175000
150000
125000
100000
75000
50000
25000
0
Ex
Fa
Gd
Po
TA
GarageQual

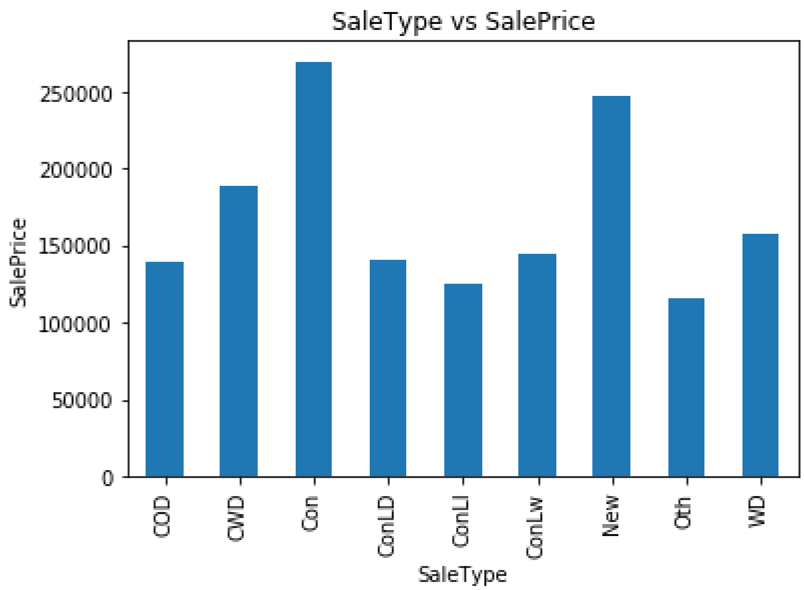

GarageCond vs SalePrice
SalePrice
160000
140000
120000
100000
80000
60000
40000
20000
0
Ex
Fa
Gd
Po
TA
GarageCond

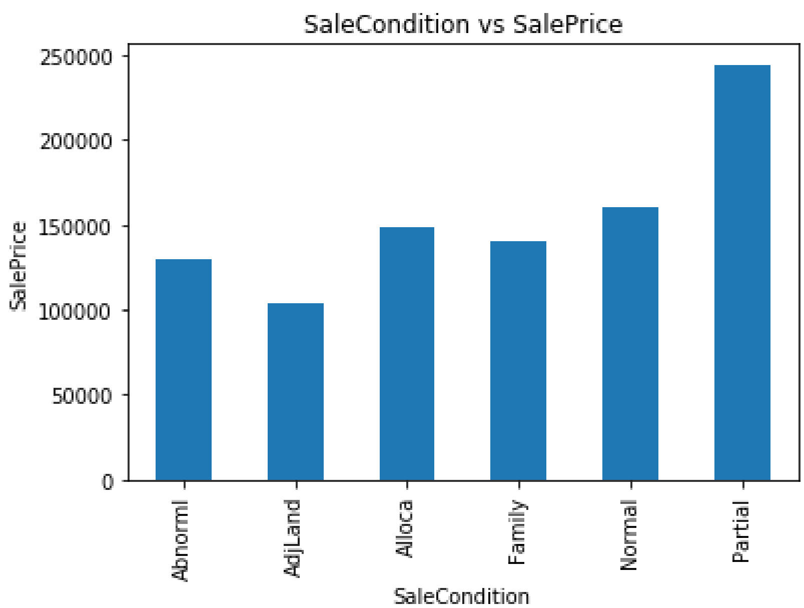

PavedDrive vs SalePrice
SalePrice
160000
140000
120000
100000
80000
60000
40000
20000
0
N
P
Y
PavedDrive


PoolQC vs SalePrice
500000
400000
300000
200000
100000
0
SalePrice
Ex
Fa
Gd
PoolQC


Fence vs SalePrice
160000
140000
120000
100000
80000
60000
40000
20000
0
SalePrice
GdPrv
GdWo
MnPrv
MnWw
Fence


MiscFeature vs SalePrice
250000
200000
150000
100000
50000
0
SalePrice
Gar2
Othr
Shed
TenC
MiscFeature


SaleType vs SalePrice
250000
200000
150000
100000
50000
0
SalePrice
COD
CWD
Con
ConLD
ConLI
ConLw
New
Oth
WD
SaleType


SaleCondition vs SalePrice
250000
200000
150000
100000
50000
0
SalePrice
Abnorml
AdjLand
Alloca
Family
Normal
Partial
SaleCondition

In [ ]: