In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
## Display all the columns of the dataframe

pd.pandas.set_option('display.max_columns',None)
```

In [2]:

```
train_df = pd.read_csv("train.csv")
test_df = pd.read_csv("test.csv")
```

In [3]:

```
train_df.shape
```

Out[3]:

```
(1460, 81)
```

In [4]:

```
train_df.head()
```

Out[4]:

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	LotConfig	LandSlope	Neighbor
0	1	60	RL	65.0	8450	Pave	NaN	Reg	Lvl	AllPub	Inside	Gtl	Co
1	2	20	RL	80.0	9600	Pave	NaN	Reg	Lvl	AllPub	FR2	Gtl	Ve
2	3	60	RL	68.0	11250	Pave	NaN	IR1	Lvl	AllPub	Inside	Gtl	Co
3	4	70	RL	60.0	9550	Pave	NaN	IR1	Lvl	AllPub	Corner	Gtl	Cr
4	5	60	RL	84.0	14260	Pave	NaN	IR1	Lvl	AllPub	FR2	Gtl	No

In [5]:

```
train_df.shape
```

Out[5]:

```
(1460, 81)
```

In [6]:

```
train_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1460 entries, 0 to 1459
Data columns (total 81 columns):
Id               1460 non-null int64
MSSubClass       1460 non-null int64
MSZoning         1460 non-null object
LotFrontage      1201 non-null float64
LotArea          1460 non-null int64
Street           1460 non-null object
Alley            91 non-null object
LotShape         1460 non-null object
LandContour      1460 non-null object
Utilities        1460 non-null object
LotConfig        1460 non-null object
LandSlope        1460 non-null object
```

```
Neighborhood     1460 non-null object
Condition1       1460 non-null object
Condition2       1460 non-null object
BldgType         1460 non-null object
HouseStyle       1460 non-null object
OverallQual      1460 non-null int64
OverallCond      1460 non-null int64
YearBuilt        1460 non-null int64
YearRemodAdd     1460 non-null int64
RoofStyle        1460 non-null object
RoofMatl         1460 non-null object
Exterior1st      1460 non-null object
Exterior2nd      1460 non-null object
MasVnrType       1452 non-null object
MasVnrArea       1452 non-null float64
ExterQual        1460 non-null object
ExterCond        1460 non-null object
Foundation       1460 non-null object
BsmtQual         1423 non-null object
BsmtCond         1423 non-null object
BsmtExposure     1422 non-null object
BsmtFinType1     1423 non-null object
BsmtFinSF1       1460 non-null int64
BsmtFinType2     1422 non-null object
BsmtFinSF2       1460 non-null int64
BsmtUnfSF        1460 non-null int64
TotalBsmtSF      1460 non-null int64
Heating          1460 non-null object
HeatingQC        1460 non-null object
CentralAir       1460 non-null object
Electrical       1459 non-null object
1stFlrSF         1460 non-null int64
2ndFlrSF         1460 non-null int64
LowQualFinSF     1460 non-null int64
GrLivArea        1460 non-null int64
BsmtFullBath     1460 non-null int64
BsmtHalfBath     1460 non-null int64
FullBath         1460 non-null int64
HalfBath         1460 non-null int64
BedroomAbvGr     1460 non-null int64
KitchenAbvGr     1460 non-null int64
KitchenQual      1460 non-null object
TotRmsAbvGrd     1460 non-null int64
Functional       1460 non-null object
Fireplaces       1460 non-null int64
FireplaceQu      770 non-null object
GarageType       1379 non-null object
GarageYrBlt      1379 non-null float64
GarageFinish     1379 non-null object
GarageCars       1460 non-null int64
GarageArea       1460 non-null int64
GarageQual       1379 non-null object
GarageCond       1379 non-null object
PavedDrive       1460 non-null object
WoodDeckSF       1460 non-null int64
OpenPorchSF      1460 non-null int64
EnclosedPorch    1460 non-null int64
3SsnPorch        1460 non-null int64
ScreenPorch      1460 non-null int64
PoolArea         1460 non-null int64
PoolQC           7 non-null object
Fence            281 non-null object
MiscFeature      54 non-null object
MiscVal          1460 non-null int64
MoSold           1460 non-null int64
YrSold           1460 non-null int64
SaleType         1460 non-null object
SaleCondition    1460 non-null object
SalePrice        1460 non-null int64
dtypes: float64(3), int64(35), object(43)
memory usage: 924.0+ KB
```

In [7]:

```
#How many columns with different datatypes are there?
train_df.get_dtype_counts()
```

```
C:\Users\Mahesh\Anaconda3\lib\site-packages\ipykernel_launcher.py:2: FutureWarning:
`get_dtype_counts` has been deprecated and will be removed in a future version. For DataFrames use
`.dtypes.value_counts()
```

Out[7]:

```
float64     3
int64      35
object     43
dtype: int64
```

In [8]:

```
train_df.columns
```

Out[8]:

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

In [9]:

```
categorical_features = train_df.select_dtypes(include=[np.object])
categorical_features.columns
```

Out[9]:

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

In [10]:

```
categorical_features.head()
```

Out[10]:

	MSZoning	Street	Alley	LotShape	LandContour	Utilities	LotConfig	LandSlope	Neighborhood	Condition1	Condition2	BldgType
0	RL	Pave	NaN	Reg	Lvl	AllPub	Inside	Gtl	CollgCr	Norm	Norm	1Fam
1	RL	Pave	NaN	Reg	Lvl	AllPub	FR2	Gtl	Veenker	Feedr	Norm	1Fam
2	RL	Pave	NaN	IR1	Lvl	AllPub	Inside	Gtl	CollgCr	Norm	Norm	1Fam
3	RL	Pave	NaN	IR1	Lvl	AllPub	Corner	Gtl	Crawfor	Norm	Norm	1Fam
4	RL	Pave	NaN	IR1	Lvl	AllPub	FR2	Gtl	NoRidge	Norm	Norm	1Fam

In [11]:

```
categorical_features.shape
```

Out[11]:

```
(1460, 43)
```

In [12]:

```
numeric_features = train_df.select_dtypes(include=[np.number])
numeric_features.columns
```

Out[12]:

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

In [13]:

```
numeric_features.head()
```

Out[13]:

	Id	MSSubClass	LotFrontage	LotArea	OverallQual	OverallCond	YearBuilt	YearRemodAdd	MasVnrArea	BsmtFinSF1	BsmtFinSF2
0	1	60	65.0	8450	7	5	2003	2003	196.0	706	0
1	2	20	80.0	9600	6	8	1976	1976	0.0	978	0
2	3	60	68.0	11250	7	5	2001	2002	162.0	486	0
3	4	70	60.0	9550	7	5	1915	1970	0.0	216	0
4	5	60	84.0	14260	8	5	2000	2000	350.0	655	0

In [14]:

```
numeric_features.shape
```

Out[14]:

```
(1460, 38)
```

To explore further we will start with the following visualisation methods to analyze the data better:

Correlation Heat Map

Zoomed Heat Map

Pair Plot

Scatter Plot

In [15]:

```
correlation = numeric_features.corr()
print(correlation['SalePrice'].sort_values(ascending = False),'\n')
```

```
SalePrice       1.000000
OverallQual     0.790982
GrLivArea       0.708624
GarageCars      0.640409
```

```
GarageArea        0.623431
TotalBsmtSF       0.613581
1stFlrSF          0.605852
FullBath          0.560664
TotRmsAbvGrd      0.533723
YearBuilt         0.522897
YearRemodAdd      0.507101
GarageYrBlt       0.486362
MasVnrArea        0.477493
Fireplaces        0.466929
BsmtFinSF1        0.386420
LotFrontage       0.351799
WoodDeckSF        0.324413
2ndFlrSF          0.319334
OpenPorchSF       0.315856
HalfBath          0.284108
LotArea           0.263843
BsmtFullBath      0.227122
BsmtUnfSF         0.214479
BedroomAbvGr      0.168213
ScreenPorch       0.111447
PoolArea          0.092404
MoSold            0.046432
3SsnPorch         0.044584
BsmtFinSF2       -0.011378
BsmtHalfBath     -0.016844
MiscVal          -0.021190
Id               -0.021917
LowQualFinSF     -0.025606
YrSold           -0.028923
OverallCond      -0.077856
MSSubClass       -0.084284
EnclosedPorch    -0.128578
KitchenAbvGr     -0.135907
Name: SalePrice, dtype: float64
```

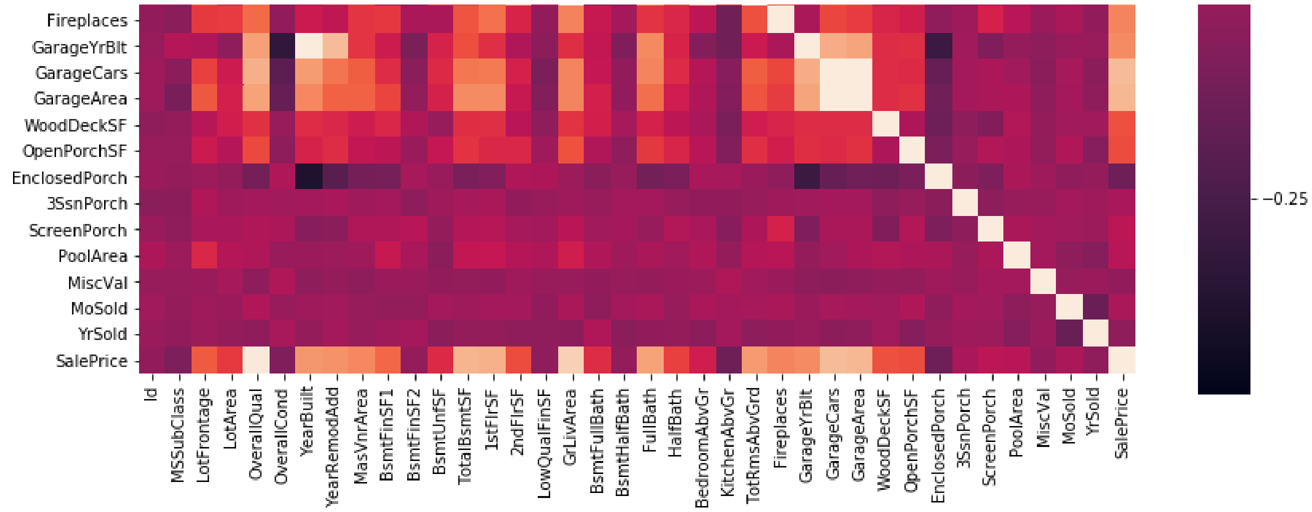
Correlation Heatmap

In [16]:

```
f , ax = plt.subplots(figsize = (14,12))
plt.title('Correlation of Numeric Features with Sale Price',y=1,size=16)
sns.heatmap(correlation,square = True,  vmax=0.8)
```

Out[16]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2e3adb352b0>
```



he heatmap is the best way to get a quick overview of correlated features thanks to seaborn!

At initial glance it is observed that there are two red colored squares that get my attention.

The first one refers to the 'TotalBsmtSF' and '1stFlrSF' variables. Second one refers to the 'GarageX' variables. Both cases show how significant the correlation is between these variables. Actually, this correlation is so strong that it can indicate a situation of multicollinearity. If we think about these variables, we can conclude that they give almost the same information so multicollinearity really occurs. Heatmaps are great to detect this kind of multicollinearity situations and in problems related to feature selection like this project, it comes as an excellent exploratory tool.

Another aspect I observed here is the 'SalePrice' correlations.As it is observed that 'GrLivArea', 'TotalBsmtSF', and 'OverallQual' saying a big 'Hello !' to SalePrice, however we cannot exclude the fact that rest of the features have some level of correlation to the SalePrice. To observe this correlation closer let us see it in Zoomed Heat Map

Zoomed Heatmap

SalePrice Correlation matrix

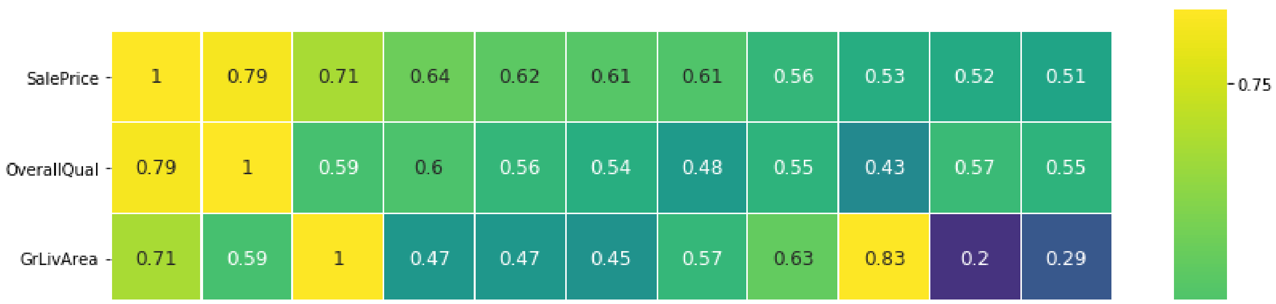
In [17]:

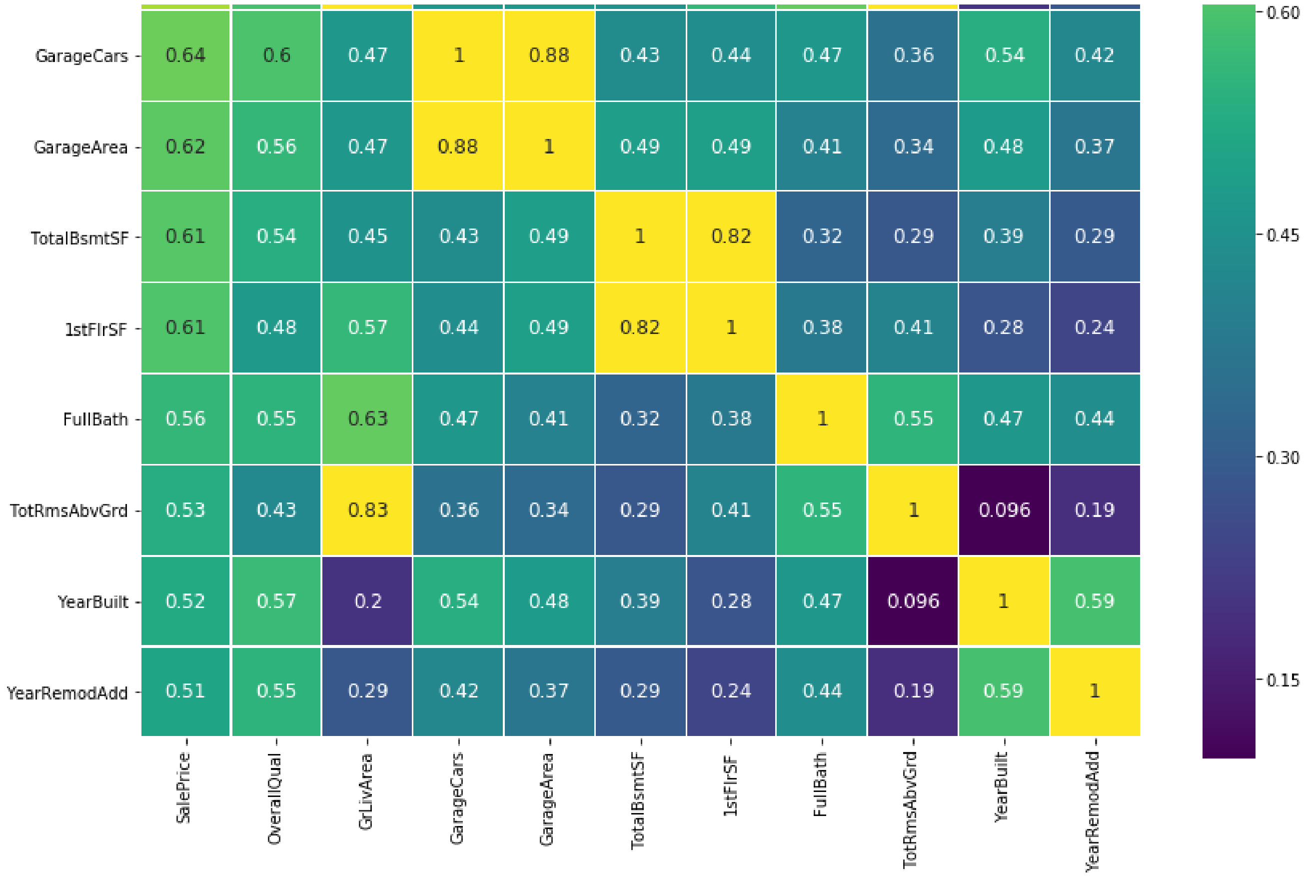
```
k= 11
cols = correlation.nlargest(k,'SalePrice')['SalePrice'].index
print(cols)
cm = np.corrcoef(train_df[cols].values.T)
f , ax = plt.subplots(figsize = (14,12))
sns.heatmap(cm, vmax=.8, linewidths=0.01,square=True,annot=True,cmap='viridis',
            linecolor="white",xticklabels = cols.values ,annot_kws = {'size':12},yticklabels = cols
.values)
```

```
Index(['SalePrice', 'OverallQual', 'GrLivArea', 'GarageCars', 'GarageArea',
       'TotalBsmtSF', '1stFlrSF', 'FullBath', 'TotRmsAbvGrd', 'YearBuilt',
       'YearRemodAdd'],
      dtype='object')
```

Out[17]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2e3ae349b38>
```





From above zoomed heatmap it is observed that GarageCars & GarageArea are closely correlated.

Similarly TotalBsmtSF and 1stFlrSF are also closely correlated.

My observations :

'OverallQual', 'GrLivArea' and 'TotalBsmtSF' are strongly correlated with 'SalePrice'.

'GarageCars' and 'GarageArea' are strongly correlated variables. It is because the number of cars that fit into the garage is a consequence of the garage area. 'GarageCars' and 'GarageArea' are like twin brothers. So it is hard to distinguish between the two. Therefore, we just need one of these variables in our analysis (we can keep 'GarageCars' since its correlation with 'SalePrice' is higher).

'TotalBsmtSF' and '1stFloor' also seem to be twins. In this case let us keep 'TotalBsmtSF' 'TotRmsAbvGrd' and 'GrLivArea', twins 'YearBuilt' it appears like is slightly correlated with 'SalePrice'. This required more analysis to arrive at a conclusion may be do some time series analysis.

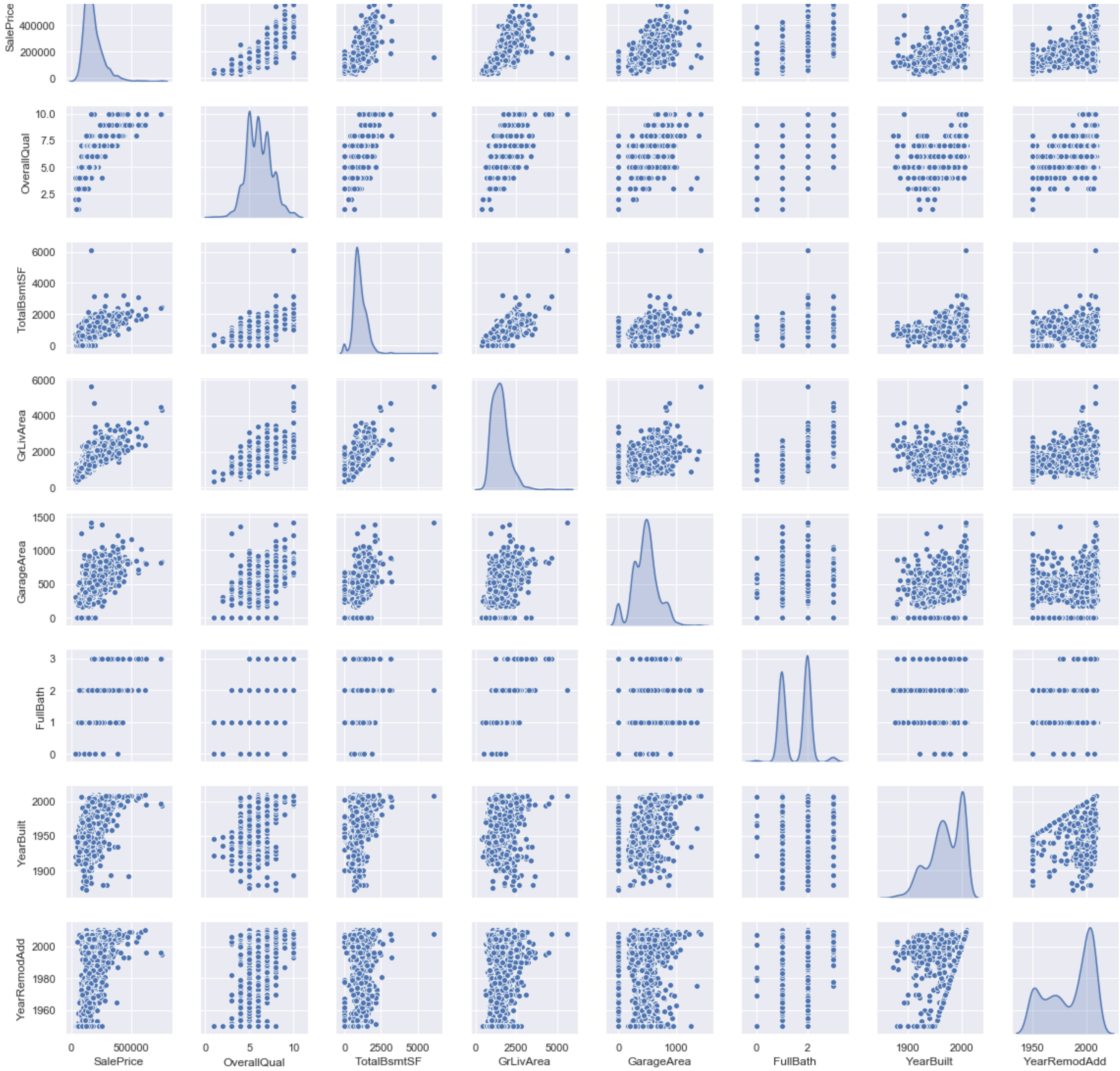
Pair Plot

Pair Plot between 'SalePrice' and correlated variables Visualisation of 'OverallQual','TotalBsmtSF','GrLivArea','GarageArea','FullBath','YearBuilt','YearRemodAdd' features with respect to SalePrice in the form of pair plot & scatter pair plot for better understanding.

In [18]:

```
sns.set()
columns = ['SalePrice','OverallQual','TotalBsmtSF','GrLivArea','GarageArea','FullBath','YearBuilt',
'YearRemodAdd']
sns.pairplot(train_df[columns],size = 2 ,kind ='scatter',diag_kind='kde')
plt.show()
```

```
C:\Users\Mahesh\Anaconda3\lib\site-packages\seaborn\axisgrid.py:2065: UserWarning: The `size` para
meter has been renamed to `height`; pleaes update your code.
  warnings.warn(msg, UserWarning)
```



1. One interesting observation is between 'TotalBsmtSF' and 'GrLiveArea'. In this figure we can see the dots drawing a linear line, which almost acts like a border. It totally makes sense that the majority of the dots stay below that line. Basement areas can be equal to the above ground living area, but it is not expected a basement area bigger than the above ground living area.
2. One more interesting observation is between 'SalePrice' and 'YearBuilt'. In the bottom of the 'dots cloud', we see what almost appears to be an exponential function. We can also see this same tendency in the upper limit of the 'dots cloud'
3. Last observation is that prices are increasing faster now with respect to previous years.

Scatter Plot

Scatter plots between the most correlated variables

In [19]:

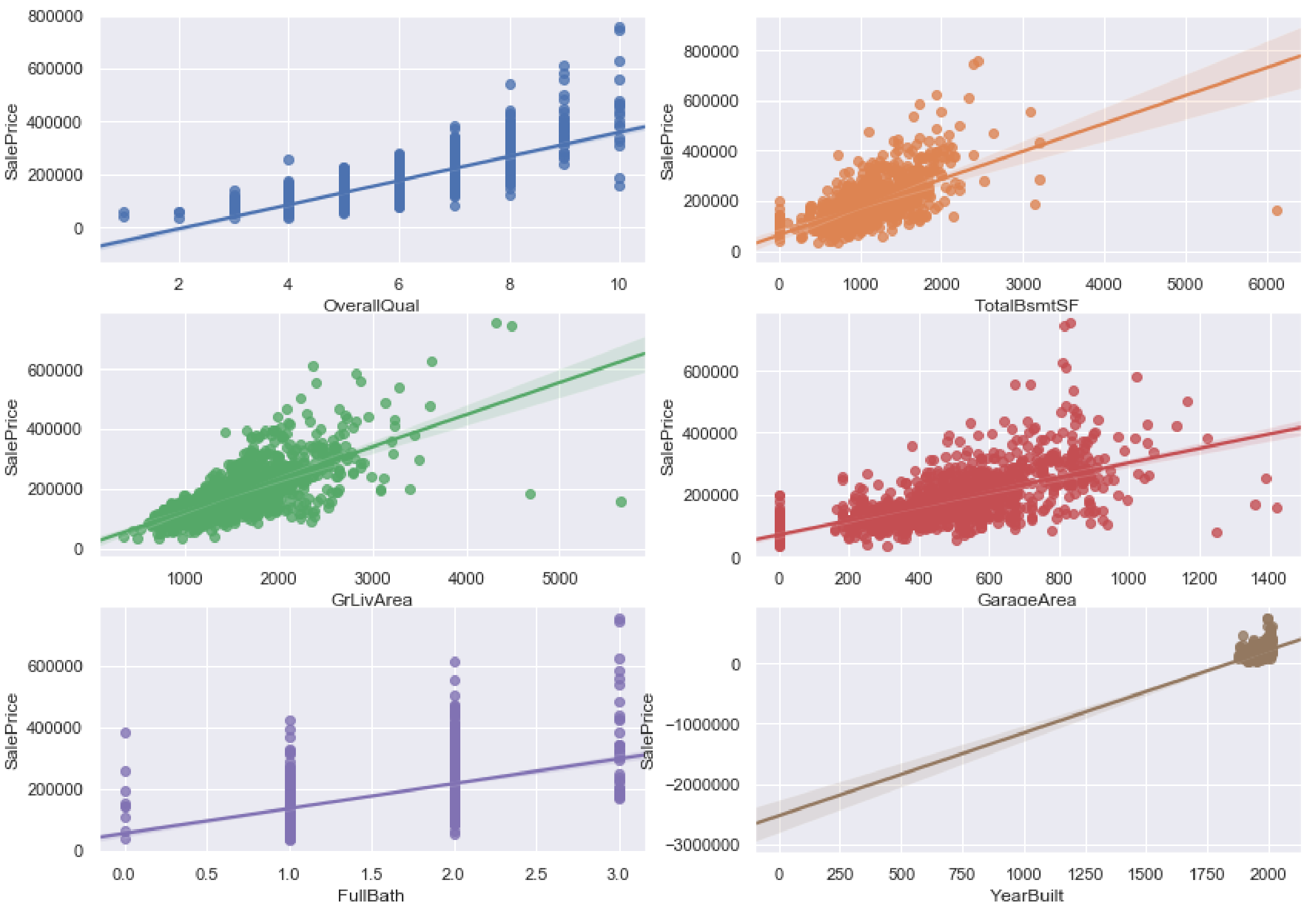
```
fig, ((ax1, ax2), (ax3, ax4),(ax5,ax6)) = plt.subplots(nrows=3, ncols=2, figsize=(14,10))
OverallQual_scatter_plot = pd.concat([train_df['SalePrice'],train_df['OverallQual']],axis = 1)
sns.regplot(x='OverallQual',y = 'SalePrice',data = OverallQual_scatter_plot,scatter= True, fit_reg=
True, ax=ax1)
TotalBsmtSF_scatter_plot = pd.concat([train_df['SalePrice'],train_df['TotalBsmtSF']],axis = 1)
sns.regplot(x='TotalBsmtSF',y = 'SalePrice',data = TotalBsmtSF_scatter_plot,scatter= True, fit_reg=
True, ax=ax2)
GrLivArea_scatter_plot = pd.concat([train_df['SalePrice'],train_df['GrLivArea']],axis = 1)
sns.regplot(x='GrLivArea',y = 'SalePrice',data = GrLivArea_scatter_plot,scatter= True, fit_reg=True
, ax=ax3)
GarageArea_scatter_plot = pd.concat([train_df['SalePrice'],train_df['GarageArea']],axis = 1)
```

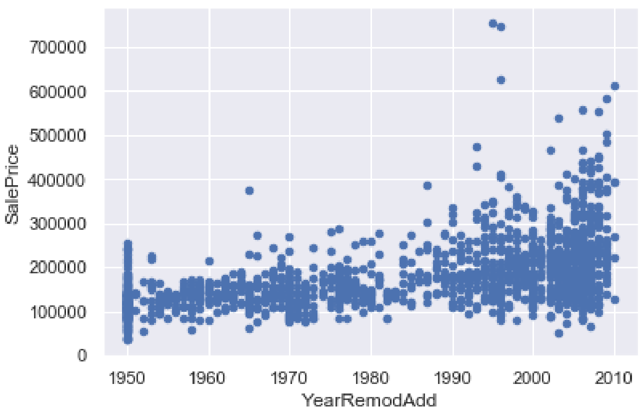
```
sns.regplot(x='GarageArea',y = 'SalePrice',data = GarageArea_scatter_plot,scatter= True, fit_reg=Tr
ue, ax=ax4)
FullBath_scatter_plot = pd.concat([train_df['SalePrice'],train_df['FullBath']],axis = 1)
sns.regplot(x='FullBath',y = 'SalePrice',data = FullBath_scatter_plot,scatter= True, fit_reg=True,
ax=ax5)
YearBuilt_scatter_plot = pd.concat([train_df['SalePrice'],train_df['YearBuilt']],axis = 1)
sns.regplot(x='YearBuilt',y = 'SalePrice',data = YearBuilt_scatter_plot,scatter= True, fit_reg=True
, ax=ax6)
YearRemodAdd_scatter_plot = pd.concat([train_df['SalePrice'],train_df['YearRemodAdd']],axis = 1)
YearRemodAdd_scatter_plot.plot.scatter('YearRemodAdd','SalePrice')
```

```
'c' argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value-ma
pping will have precedence in case its length matches with 'x' & 'y'.  Please use a 2-D array with
a single row if you really want to specify the same RGB or RGBA value for all points.
```

Out[19]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2e3b1cd57b8>
```

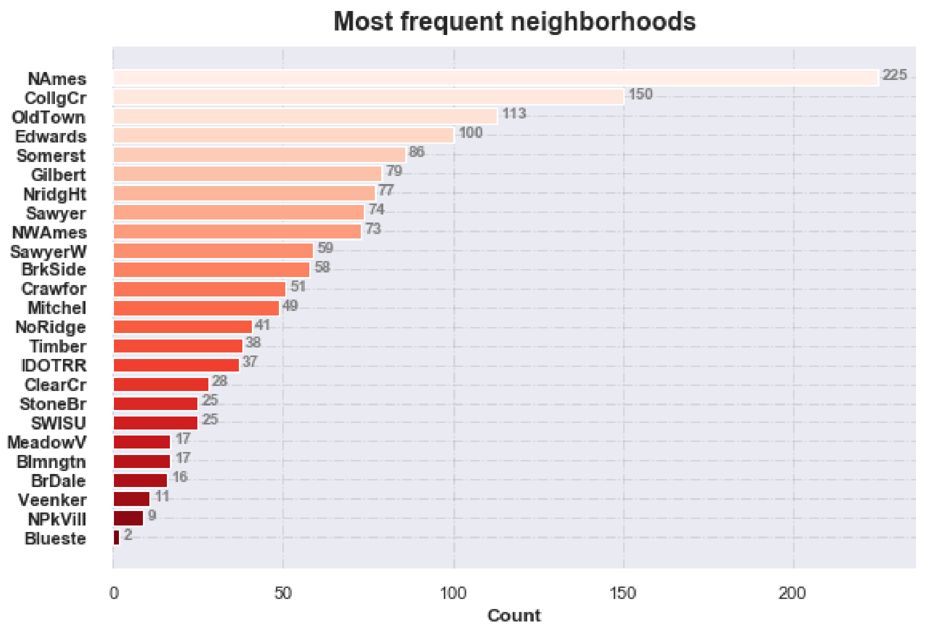




Most Frequent Neighbors

In [20]:

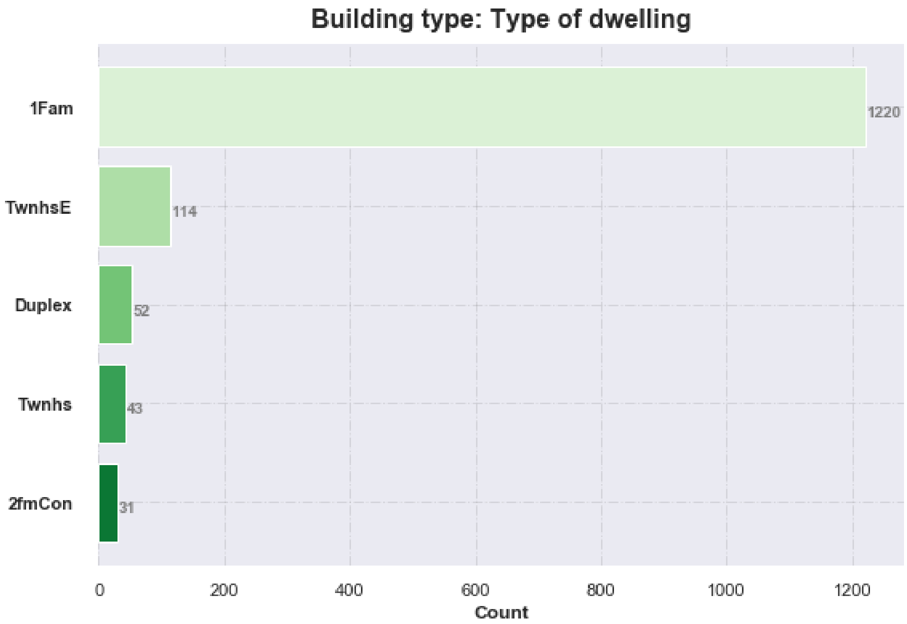
```
##Figure Size
fig, ax = plt.subplots(figsize=(9,6))

# Horizontal Bar Plot
title_cnt=train_df.Neighborhood.value_counts().sort_values(ascending=False).reset_index()
mn= ax.barh(title_cnt.iloc[:,0], title_cnt.iloc[:,1], color=sns.color_palette('Reds',len(title_cnt
)))




# Remove axes splines
for s in ['top','bottom','left','right']:
    ax.spines[s].set_visible(False)

# Remove x,y Ticks
ax.xaxis.set_ticks_position('none')
ax.yaxis.set_ticks_position('none')

# Add padding between axes and labels
ax.xaxis.set_tick_params(pad=5)
ax.yaxis.set_tick_params(pad=10)

# Add x,y gridlines
ax.grid(b=True, color='grey', linestyle='-.', linewidth=1, alpha=0.2)

# Show top values
ax.invert_yaxis()

# Add Plot Title
ax.set_title('Most frequent neighborhoods',weight='bold',
             loc='center', pad=10, fontsize=16)
ax.set_xlabel('Count', weight='bold')


# Add annotation to bars
for i in ax.patches:
    ax.text(i.get_width()+1, i.get_y()+0.5, str(round((i.get_width()), 2)),
            fontsize=10, fontweight='bold', color='grey')
plt.yticks(weight='bold')


plt.show()
# Show Plot
plt.show()
```



Different types of Dwelling

In [21]:

```
# Figure Size
fig, ax = plt.subplots(figsize=(9,6))

# Horizontal Bar Plot
title_cnt=train_df.BldgType.value_counts().sort_values(ascending=False).reset_index()
mn= ax.barh(title_cnt.iloc[:,0], title_cnt.iloc[:,1],
color=sns.color_palette('Greens',len(title_cnt)))




# Remove axes splines
for s in ['top','bottom','left','right']:
    ax.spines[s].set_visible(False)
# Remove x,y Ticks
ax.xaxis.set_ticks_position('none')
ax.yaxis.set_ticks_position('none')

# Add padding between axes and labels
ax.xaxis.set_tick_params(pad=5)
ax.yaxis.set_tick_params(pad=10)

# Add x,y gridlines
ax.grid(b=True, color='grey', linestyle='-.', linewidth=1, alpha=0.2)

# Show top values
ax.invert_yaxis()

# Add Plot Title
ax.set_title('Building type: Type of dwelling',weight='bold',
             loc='center', pad=10, fontsize=16)
ax.set_xlabel('Count', weight='bold')


# Add annotation to bars
for i in ax.patches:
    ax.text(i.get_width()+1, i.get_y()+0.5, str(round((i.get_width()), 2)),
            fontsize=10, fontweight='bold', color='grey')
plt.yticks(weight='bold')
plt.show()
```



Missing Values

In [22]:

```
## Here we will check the percentage of nan values present in each feature
```

```python
## 1 -step make the list of features which has missing values
features_with_na=[features for features in train_df.columns if train_df[features].isnull().sum()>1]
## 2- step print the feature name and the percentage of missing values

for feature in features_with_na:
    print(feature, np.round(train_df[feature].isnull().mean(), 4),  ' % missing values')
```

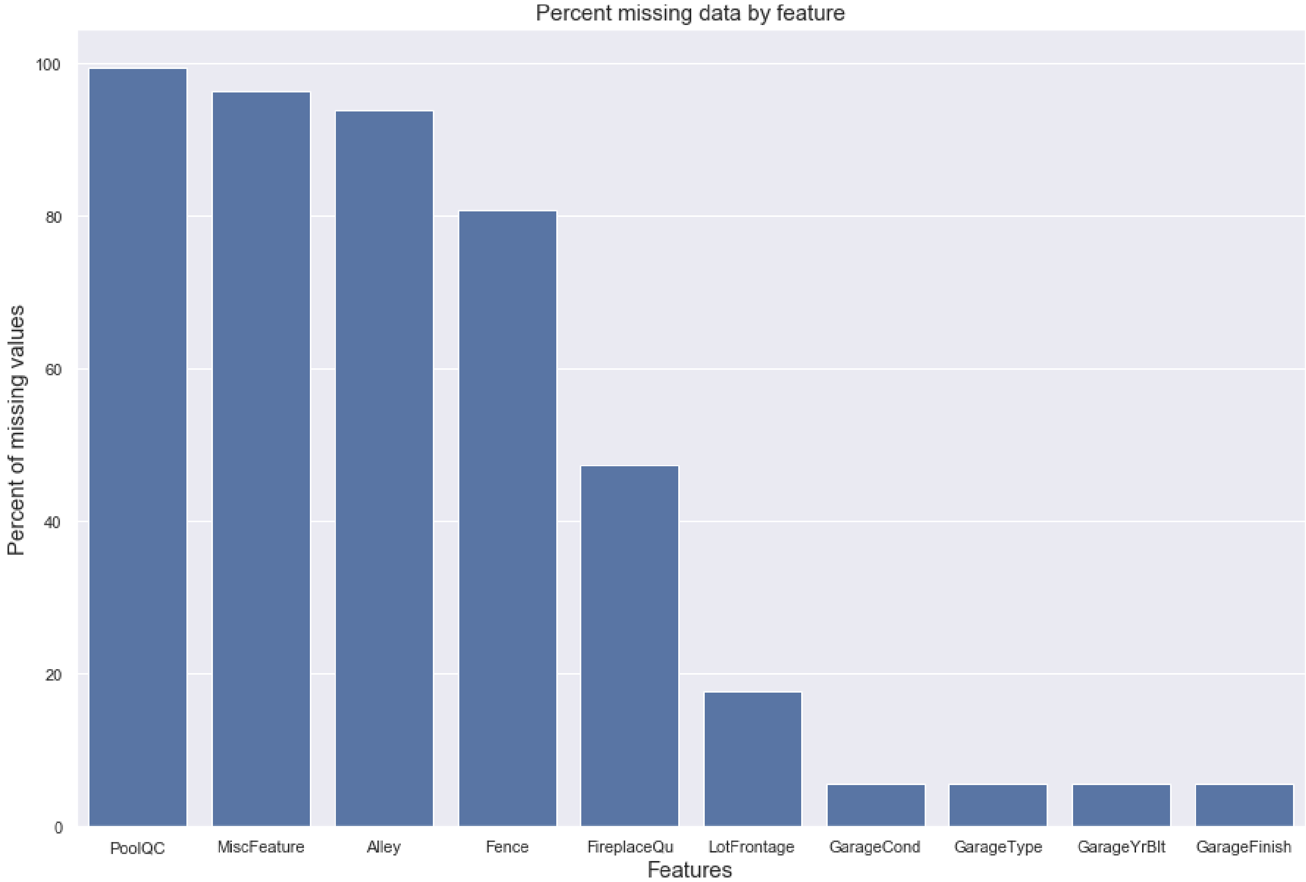
```
LotFrontage 0.1774  % missing values
Alley 0.9377  % missing values
MasVnrType 0.0055  % missing values
MasVnrArea 0.0055  % missing values
BsmtQual 0.0253  % missing values
BsmtCond 0.0253  % missing values
BsmtExposure 0.026  % missing values
BsmtFinType1 0.0253  % missing values
BsmtFinType2 0.026  % missing values
FireplaceQu 0.4726  % missing values
GarageType 0.0555  % missing values
GarageYrBlt 0.0555  % missing values
GarageFinish 0.0555  % missing values
GarageQual 0.0555  % missing values
GarageCond 0.0555  % missing values
PoolQC 0.9952  % missing values
Fence 0.8075  % missing values
MiscFeature 0.963  % missing values
```

In [23]:

```python
# Let's plot these missing values(%) vs column_names
missing_values_count = (train_df.isnull().sum()/train_df.isnull().count()*100).sort_values(ascendin
g=False)
plt.figure(figsize=(15,10))
base_color = sns.color_palette()[0]
plt.xlabel('Features', fontsize=15)
plt.ylabel('Percent of missing values', fontsize=15)
plt.title('Percent missing data by feature', fontsize=15)
sns.barplot(missing_values_count[:10].index.values, missing_values_count[:10], color = base_color)
```

Out[23]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2e3b20bc518>
```

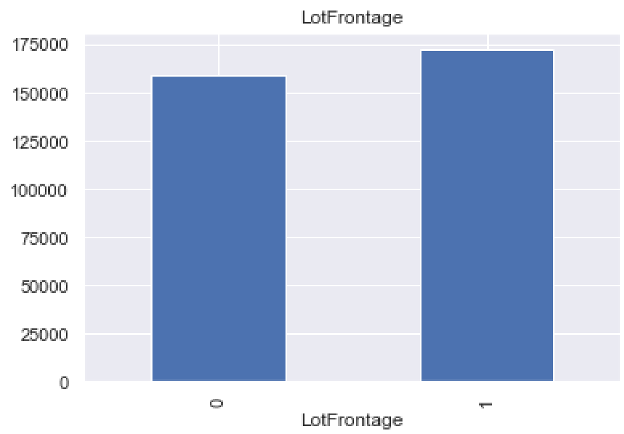


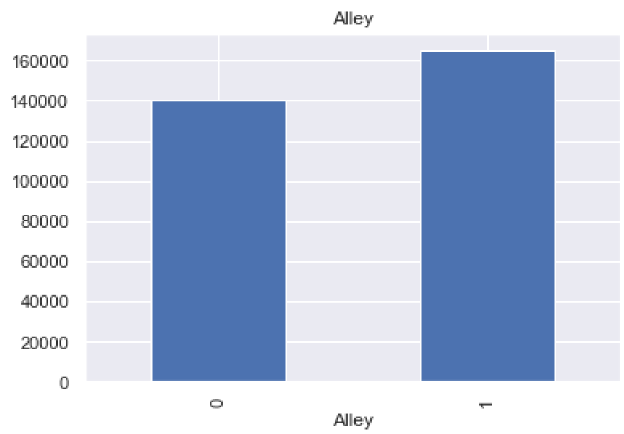
Since they are many missing values, we need to find the relationship between missing values and Sales Price

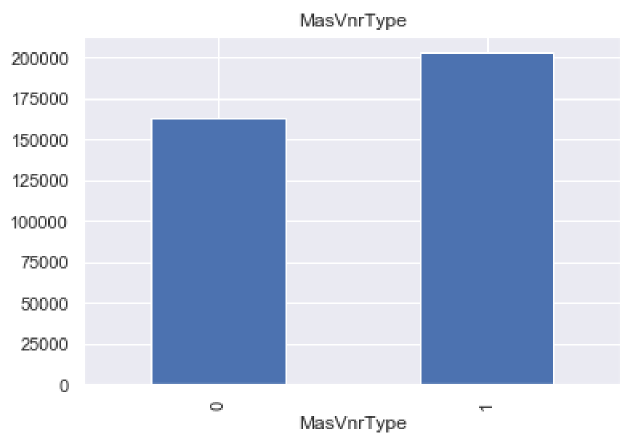
Let's plot some diagram for this relationship

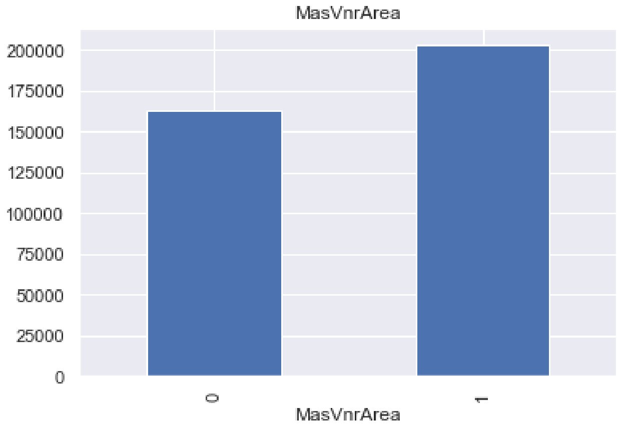
In [24]:

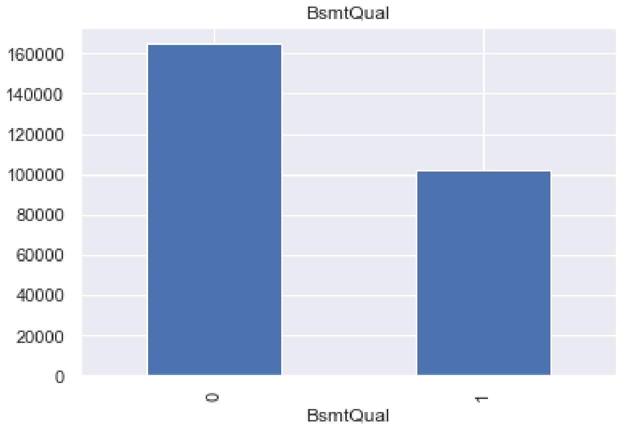
```
for feature in features_with_na:
    data = train_df.copy()

    # let's make a variable that indicates 1 if the observation was missing or zero otherwise
    data[feature] = np.where(data[feature].isnull(), 1, 0)

    # let's calculate the mean SalePrice where the information is missing or present
    data.groupby(feature)['SalePrice'].median().plot.bar()
    plt.title(feature)
    plt.show()
```

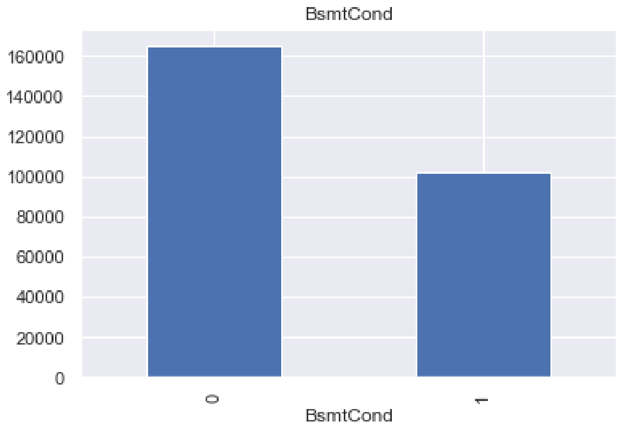


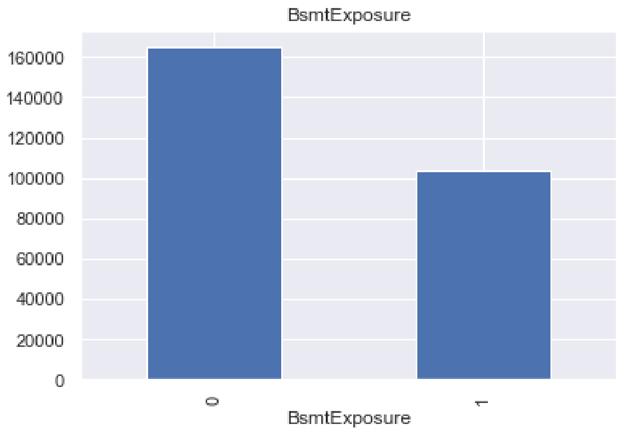


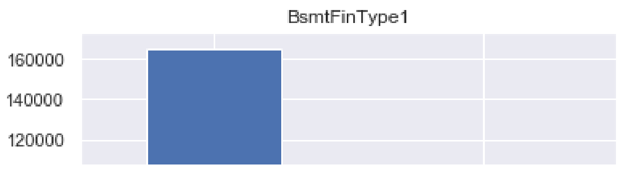


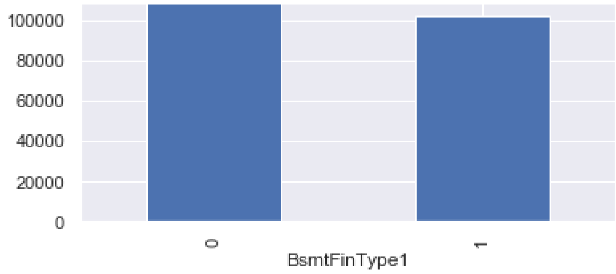

MasVnrArea
200000
175000
150000
125000
100000
75000
50000
25000
0
0
1
MasVnrArea

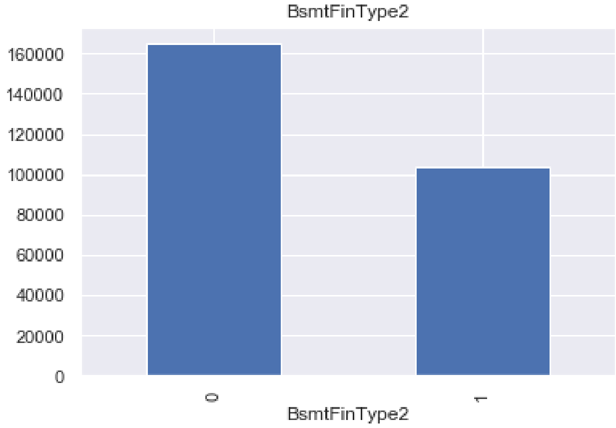

BsmtQual
160000
140000
120000
100000
80000
60000
40000
20000
0
0
1
BsmtQual

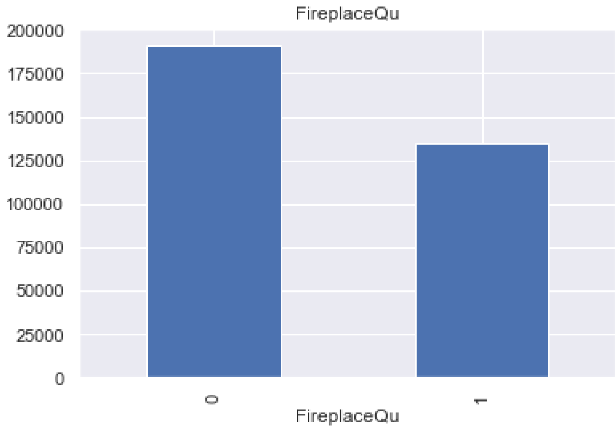

BsmtCond
160000
140000
120000
100000
80000
60000
40000
20000
0
0
1
BsmtCond

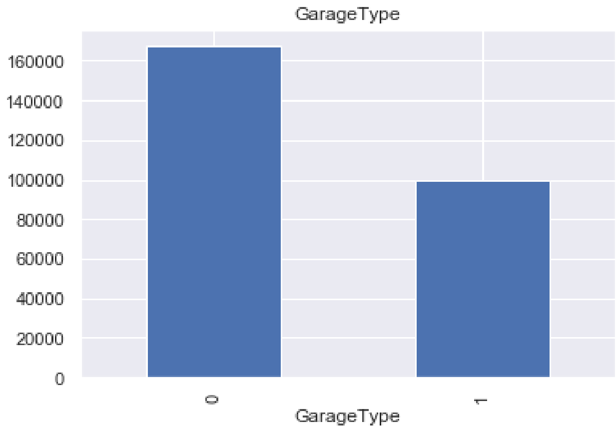

BsmtExposure
160000
140000
120000
100000
80000
60000
40000
20000
0
0
1
BsmtExposure

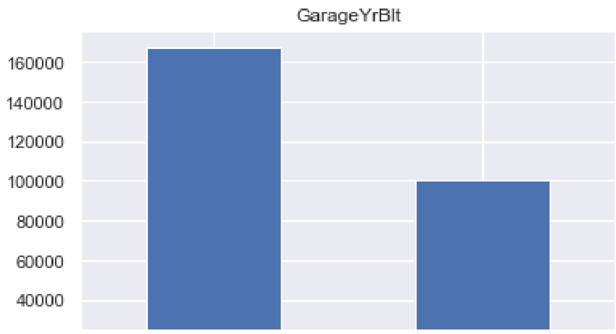

BsmtFinType1
160000
140000
120000


100000
80000
60000
40000
20000
0
0
1
BsmtFinType1

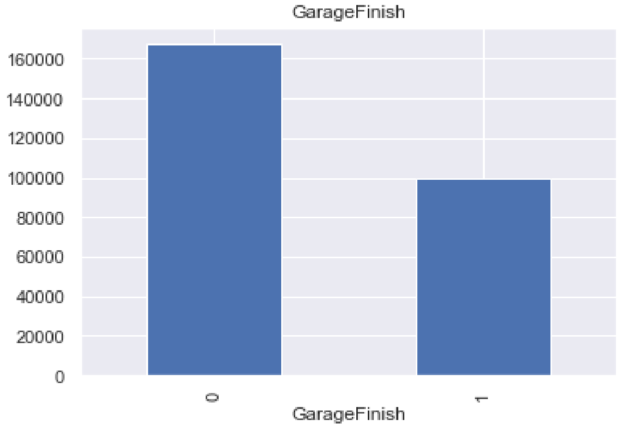

BsmtFinType2
160000
140000
120000
100000
80000
60000
40000
20000
0
0
1
BsmtFinType2

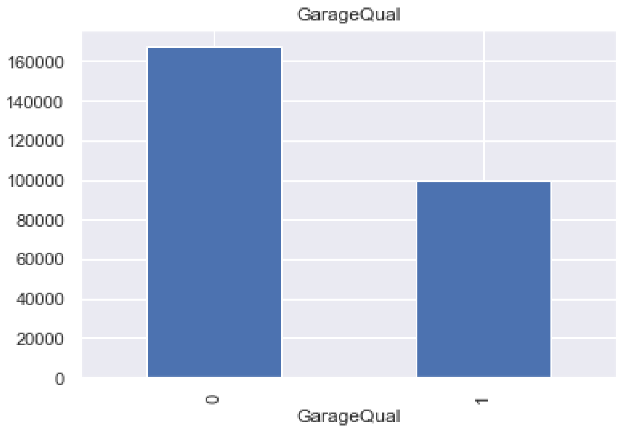

FireplaceQu
200000
175000
150000
125000
100000
75000
50000
25000
0
0
1
FireplaceQu

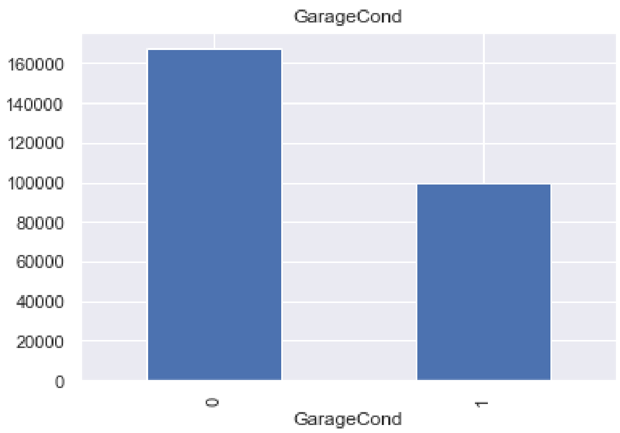

GarageType
160000
140000
120000
100000
80000
60000
40000
20000
0
0
1
GarageType

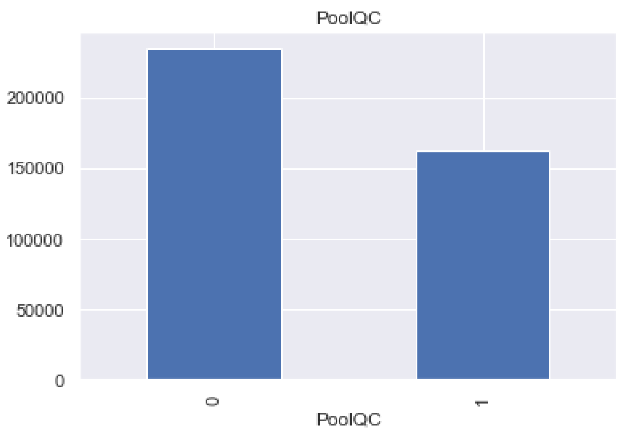

GarageYrBlt
160000
140000
120000
100000
80000
60000
40000

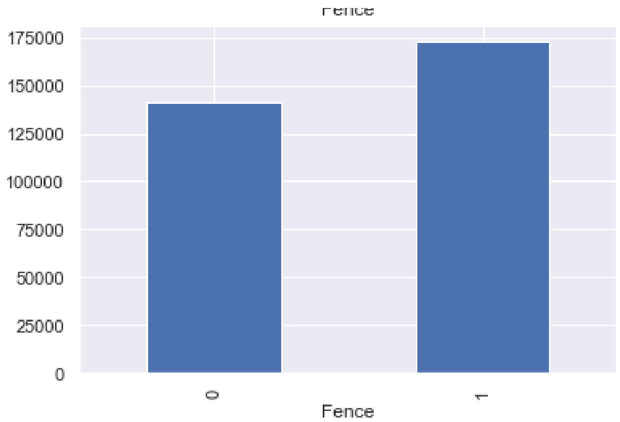

20000
0
0
1
GarageYrBlt


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



Here With the relation between the missing values and the dependent variable is clearly visible.So We need to replace these nan values with something meaningful which we will do in the Feature Engineering section

From the above dataset some of the features like Id is not required

In [25]:

```
print("Id of Houses {}".format(len(train_df.Id)))
```

```
Id of Houses 1460
```

Numerical Variables

In [26]:

```
# list of numerical variables
numerical_features = [feature for feature in train_df.columns if train_df[feature].dtypes != 'O']

print('Number of numerical variables: ', len(numerical_features))

# visualise the numerical variables
train_df[numerical_features].head()
```

```
Number of numerical variables:  38
```

Out[26]:

	Id	MSSubClass	LotFrontage	LotArea	OverallQual	OverallCond	YearBuilt	YearRemodAdd	MasVnrArea	BsmtFinSF1	BsmtFinSF2
0	1	60	65.0	8450	7	5	2003	2003	196.0	706	0
1	2	20	80.0	9600	6	8	1976	1976	0.0	978	0
2	3	60	68.0	11250	7	5	2001	2002	162.0	486	0
3	4	70	60.0	9550	7	5	1915	1970	0.0	216	0
4	5	60	84.0	14260	8	5	2000	2000	350.0	655	0

Id MSSubClass LotFrontage LotArea OverallQual OverallCond YearBuilt YearRemodAdd MasVnrArea BsmtFinSF1 BsmtFinSF2

Temporal Variables(Eg: Datetime Variables) From the Dataset we have 4 year variables. We have extract information from the datetime variables like no of years or no of days. One example in this specific scenario can be difference in years between the year the house was built and the year the house was sold. We will be performing this analysis in the Feature Engineering section.

In [27]:

```
# list of variables that contain year information
year_feature = [feature for feature in numerical_features if 'Yr' in feature or 'Year' in feature]

year_feature
```

Out[27]:

```
['YearBuilt', 'YearRemodAdd', 'GarageYrBlt', 'YrSold']
```

In [28]:

```
# let's explore the content of these year variables
for feature in year_feature:
    print(feature, train_df[feature].unique())
```

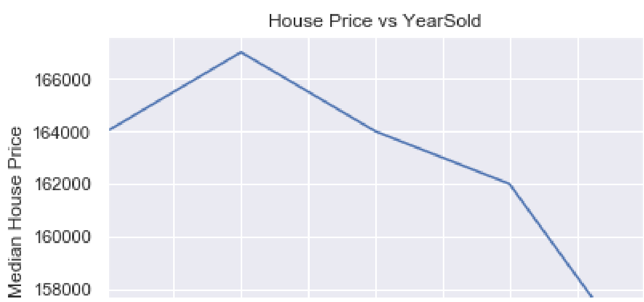
```
YearBuilt [2003 1976 2001 1915 2000 1993 2004 1973 1931 1939 1965 2005 1962 2006
 1960 1929 1970 1967 1958 1930 2002 1968 2007 1951 1957 1927 1920 1966
 1959 1994 1954 1953 1955 1983 1975 1997 1934 1963 1981 1964 1999 1972
 1921 1945 1982 1998 1956 1948 1910 1995 1991 2009 1950 1961 1977 1985
 1979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

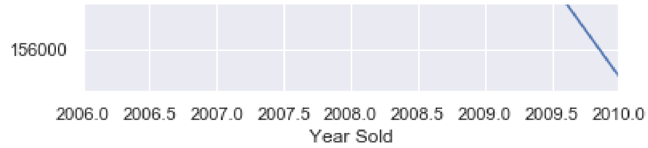
In [29]:

```
## Lets analyze the Temporal Datetime Variables
## We will check whether there is a relation between year the house is sold and the sales price

train_df.groupby('YrSold')['SalePrice'].median().plot()
plt.xlabel('Year Sold')
plt.ylabel('Median House Price')
plt.title("House Price vs YearSold")
```

Out[29]:

```
Text(0.5, 1.0, 'House Price vs YearSold')
```





We will check whether there is a relation between year the house is sold and the sales price. As we see in the below fig, as the year sold is going on the price is decreasing this cannot be just true.

In [30]:

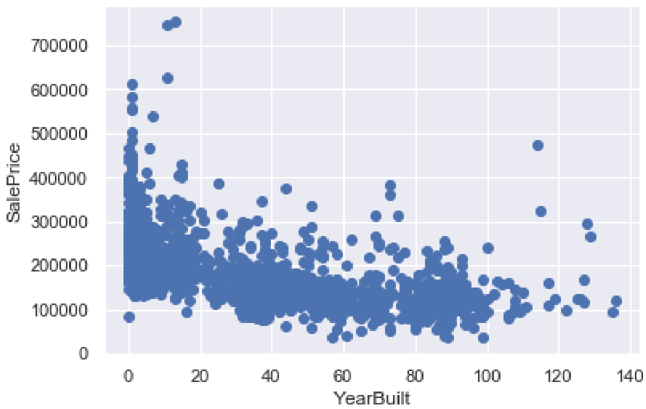
```
year_feature
```

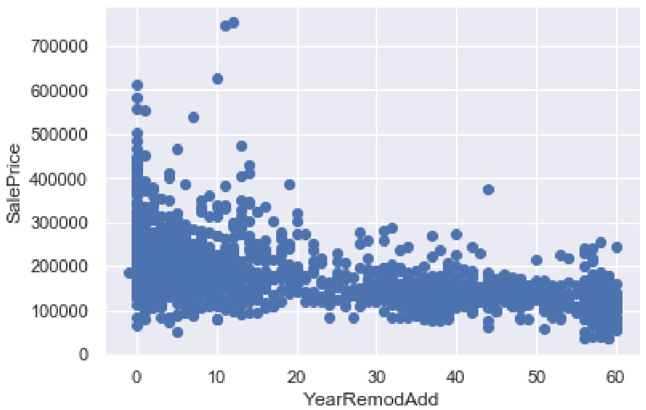
Out[30]:

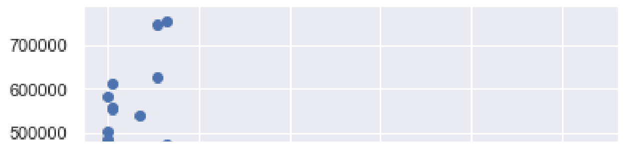
```
['YearBuilt', 'YearRemodAdd', 'GarageYrBlt', 'YrSold']
```

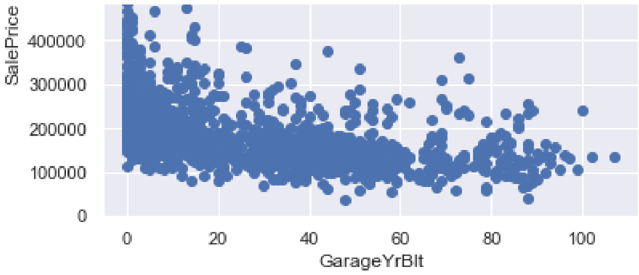
In [31]:

```
## Here we will compare the difference between All years feature with SalePrice

for feature in year_feature:
    if feature!='YrSold':
        data=train_df.copy()
        ## We will capture the difference between year variable and year the house was sold for
        data[feature]=data['YrSold']-data[feature]

        plt.scatter(data[feature],data['SalePrice'])
        plt.xlabel(feature)
        plt.ylabel('SalePrice')
        plt.show()
```









In [32]:

```
## Numerical variables are usually of 2 type
## 1. Continous variable and Discrete Variables

discrete_feature=[feature for feature in numerical_features if len(train_df[feature].unique())<25 a
nd feature not in year_feature+['Id']]
print("Discrete Variables Count: {}".format(len(discrete_feature)))
```

```
Discrete Variables Count: 17
```

In [33]:

```
discrete_feature
```

Out[33]:

```
['MSSubClass',
 'OverallQual',
 'OverallCond',
 'LowQualFinSF',
 'BsmtFullBath',
 'BsmtHalfBath',
 'FullBath',
 'HalfBath',
 'BedroomAbvGr',
 'KitchenAbvGr',
 'TotRmsAbvGrd',
 'Fireplaces',
 'GarageCars',
 '3SsnPorch',
 'PoolArea',
 'MiscVal',
 'MoSold']
```

In [34]:

```
train_df[discrete_feature].head()
```

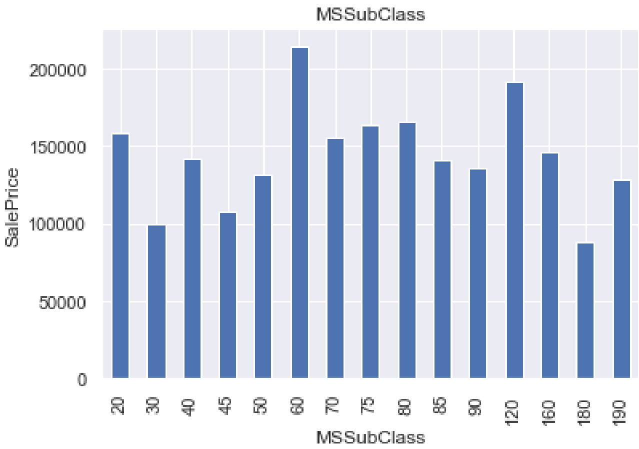
Out[34]:

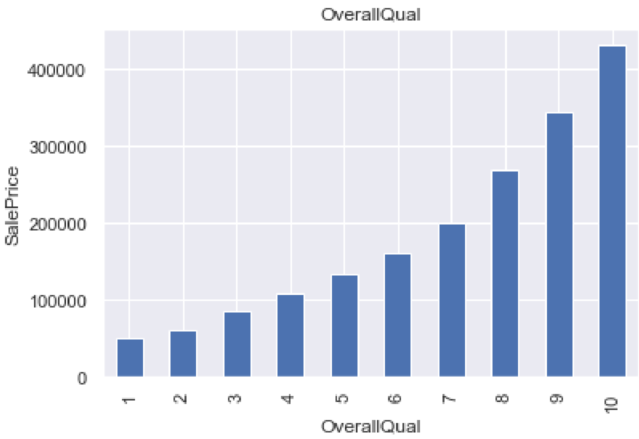
	MSSubClass	OverallQual	OverallCond	LowQualFinSF	BsmtFullBath	BsmtHalfBath	FullBath	HalfBath	BedroomAbvGr	KitchenAb
0	60	7	5	0	1	0	2	1	3	
1	20	6	8	0	0	1	2	0	3	
2	60	7	5	0	1	0	2	1	3	
3	70	7	5	0	1	0	1	0	3	
4	60	8	5	0	1	0	2	1	4	

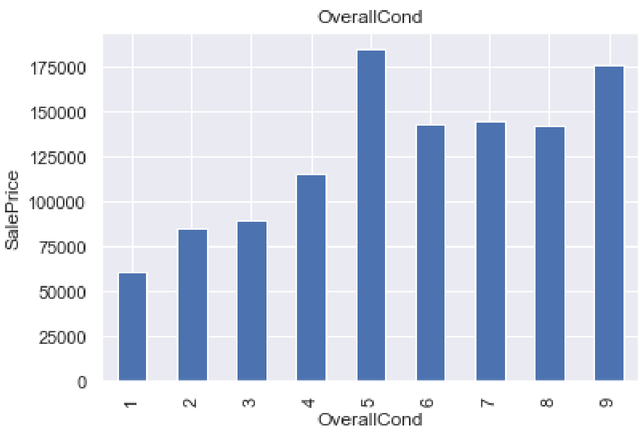
In [35]:

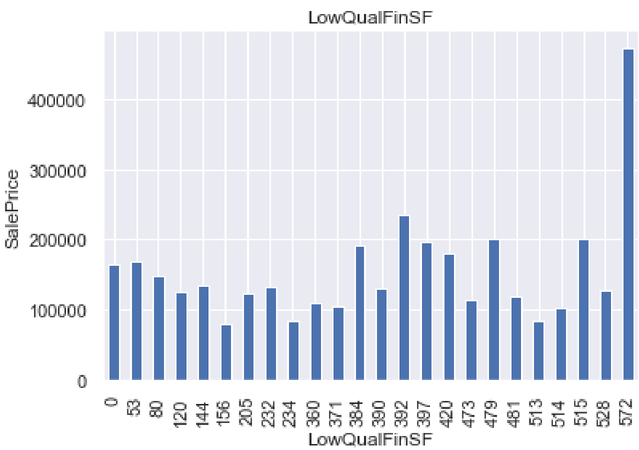
```
## Lets Find the realtionship between them and Sale PRice

for feature in discrete_feature:
    data=train_df.copy()
    data.groupby(feature)['SalePrice'].median().plot.bar()
    plt.xlabel(feature)
    plt.ylabel('SalePrice')
```

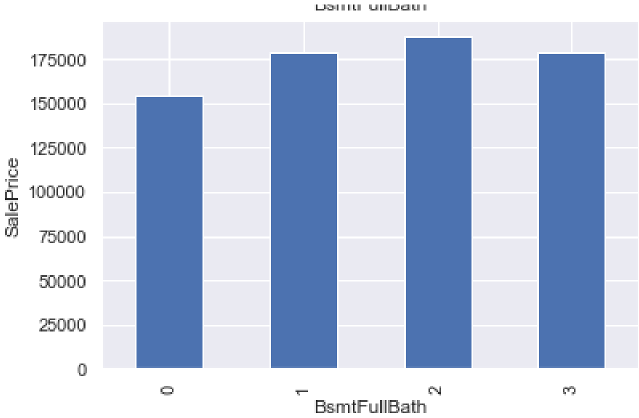
```
    plt.title(feature)
    plt.show()
```

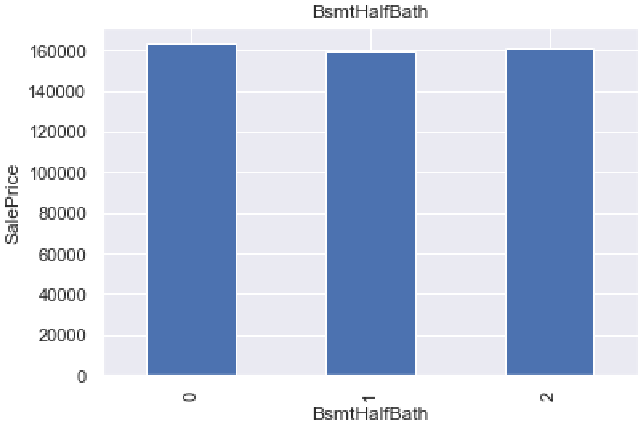


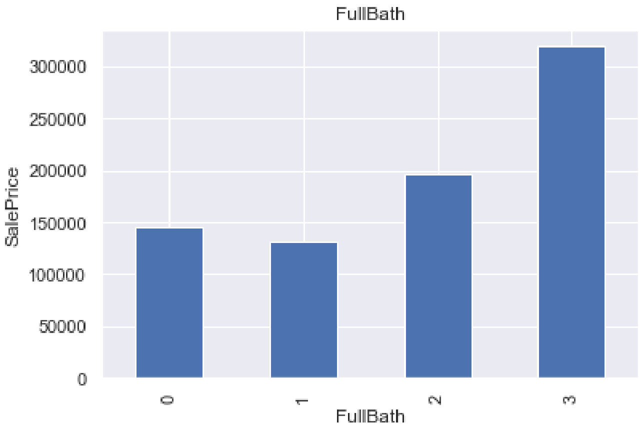


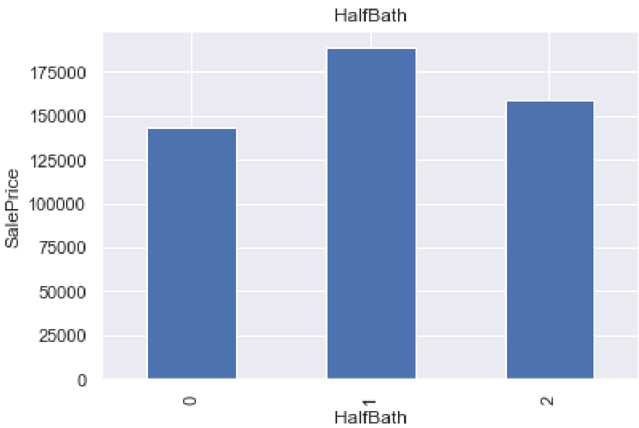


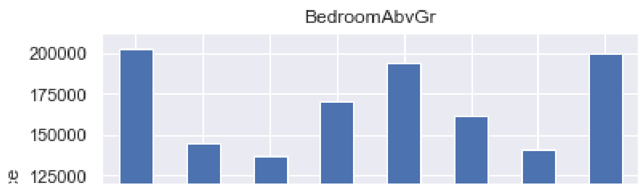


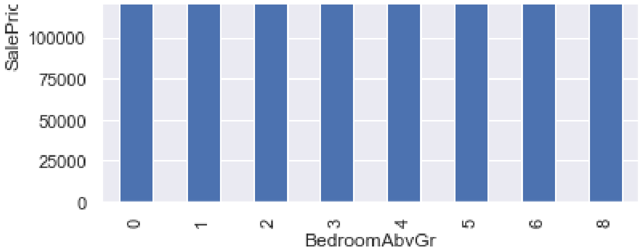

175000
150000
125000
100000
75000
50000
25000
0
SalePrice
0
1
2
3
BsmtFullBath

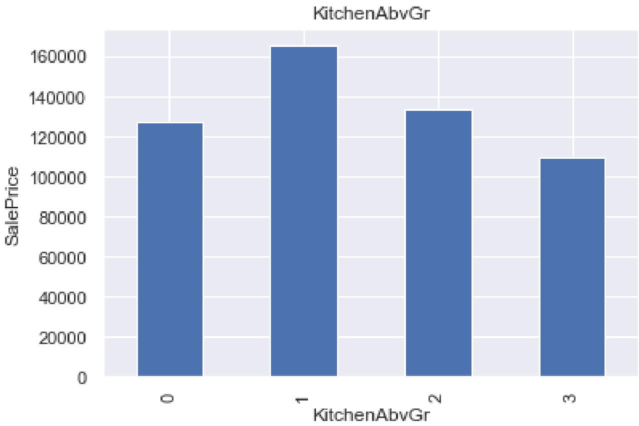

BsmtHalfBath
160000
140000
120000
100000
80000
60000
40000
20000
0
SalePrice
0
1
2
BsmtHalfBath

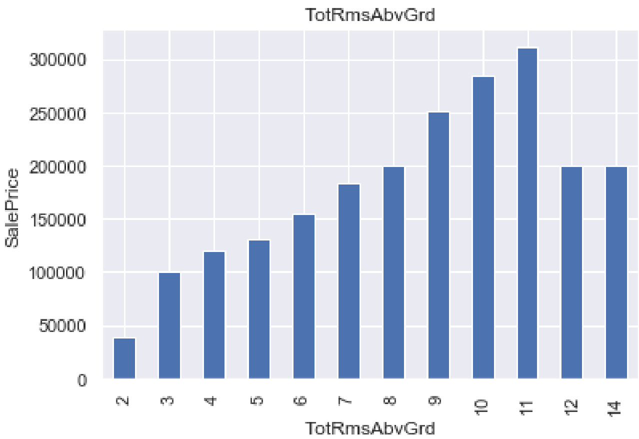

FullBath
300000
250000
200000
150000
100000
50000
0
SalePrice
0
1
2
3
FullBath

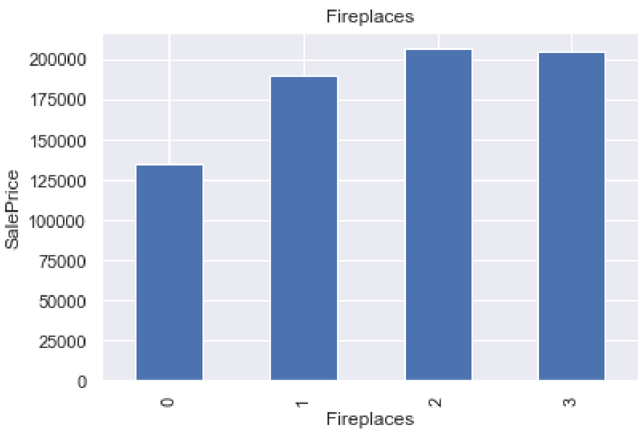

HalfBath
175000
150000
125000
100000
75000
50000
25000
0
SalePrice
0
1
2
HalfBath

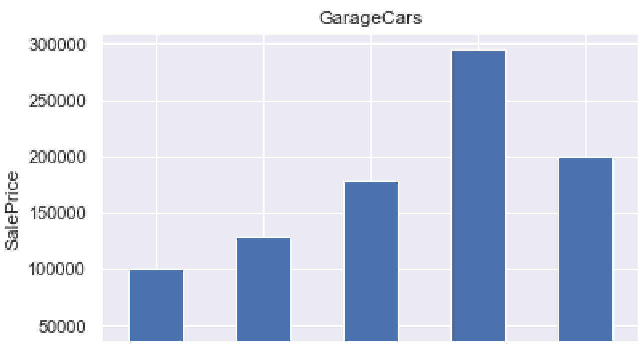

BedroomAbvGr
200000
175000
150000
125000


SalePric
100000
75000
50000
25000
0
0 1 2 3 4 5 6 8
BedroomAbvGr

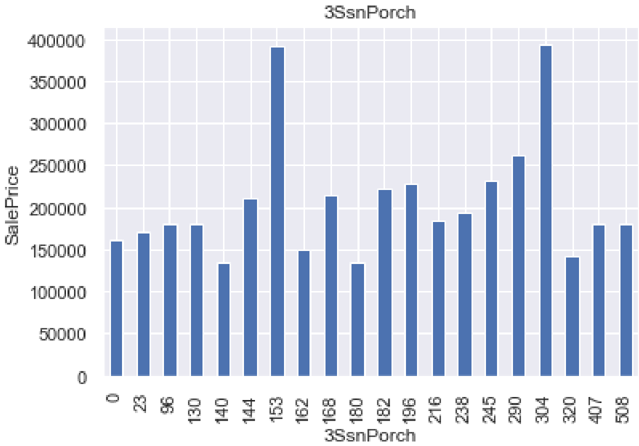

KitchenAbvGr
SalePrice
160000
140000
120000
100000
80000
60000
40000
20000
0
0 1 2 3
KitchenAbvGr

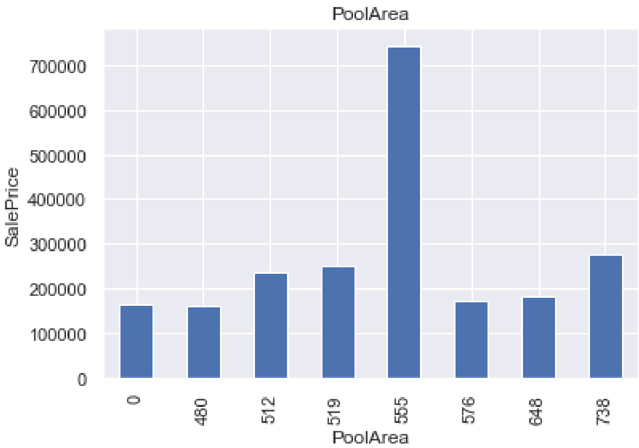

TotRmsAbvGrd
SalePrice
300000
250000
200000
150000
100000
50000
0
2 3 4 5 6 7 8 9 10 11 12 14
TotRmsAbvGrd

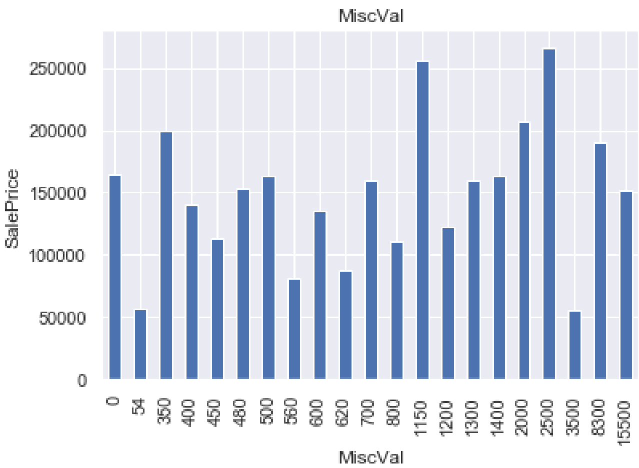

Fireplaces
SalePrice
200000
175000
150000
125000
100000
75000
50000
25000
0
0 1 2 3
Fireplaces

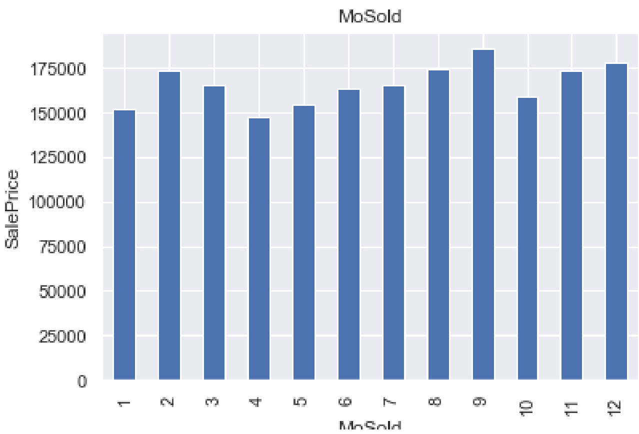

GarageCars
SalePrice
300000
250000
200000
150000
100000
50000


0
0
1
2
3
4
GarageCars


3SsnPorch
SalePrice
400000
350000
300000
250000
200000
150000
100000
50000
0
0 23 96 130 140 144 153 162 168 180 182 196 216 238 245 290 304 320 407 508
3SsnPorch


PoolArea
SalePrice
700000
600000
500000
400000
300000
200000
100000
0
0 480 512 519 555 576 648 738
PoolArea


MiscVal
SalePrice
250000
200000
150000
100000
50000
0
0 54 350 400 450 480 500 560 600 620 700 800 1150 1200 1300 1400 2000 2500 3500 8300 15500
MiscVal


MoSold
SalePrice
175000
150000
125000
100000
75000
50000
25000
0
1 2 3 4 5 6 7 8 9 10 11 12
MoSold

Continuous Variable

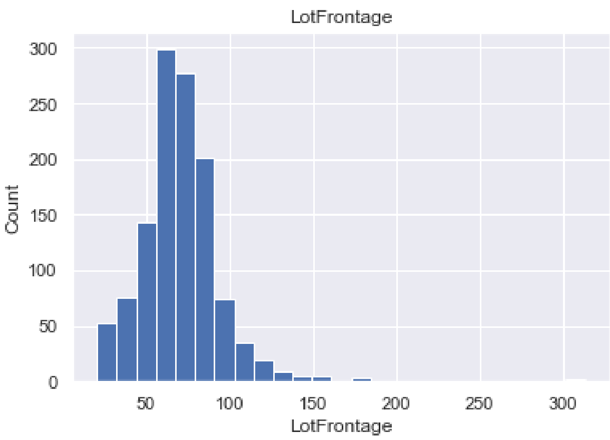
In [36]:

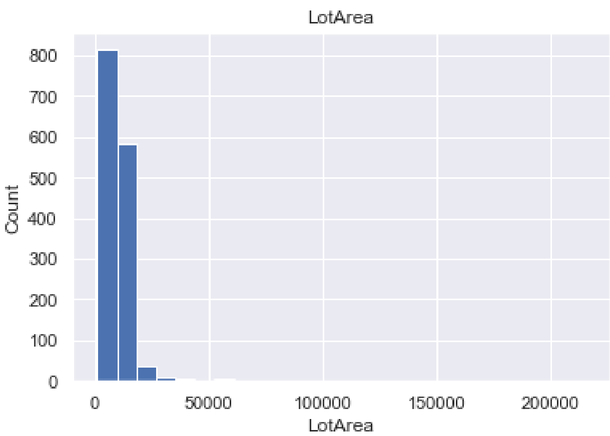
```
continuous_feature=[feature for feature in numerical_features if feature not in
discrete_feature+year_feature+['Id']]
print("Continuous feature Count {}".format(len(continuous_feature)))
```

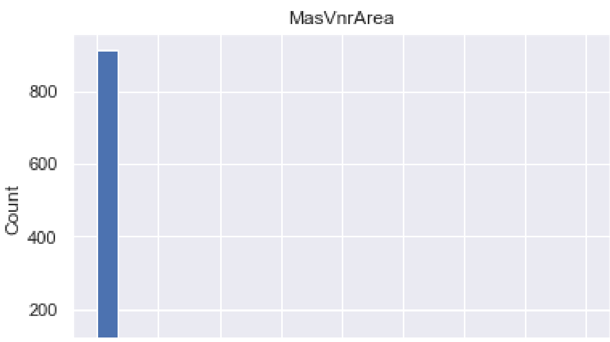
```
Continuous feature Count 16
```

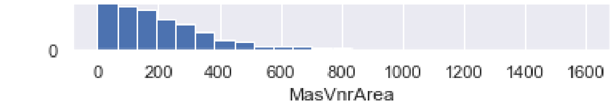
In [37]:

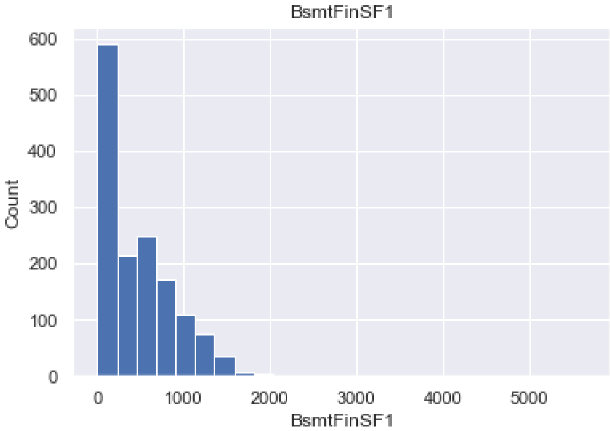
```
## Lets analyse the continuous values by creating histograms to understand the distribution

for feature in continuous_feature:
    data=train_df.copy()
    data[feature].hist(bins=25)
    plt.xlabel(feature)
    plt.ylabel("Count")
    plt.title(feature)
    plt.show()
```

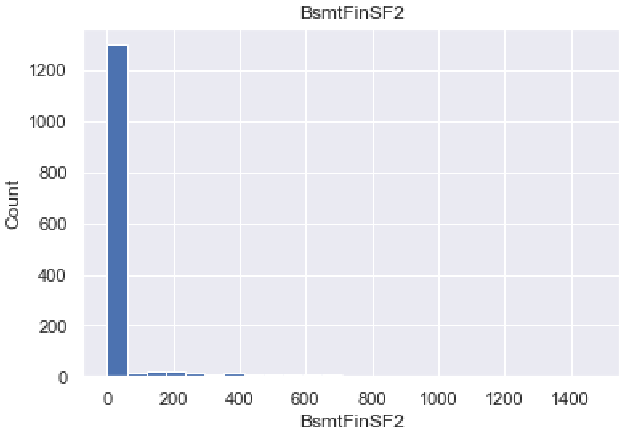


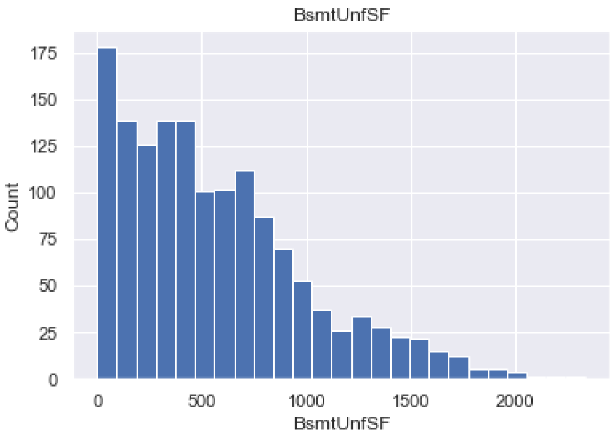


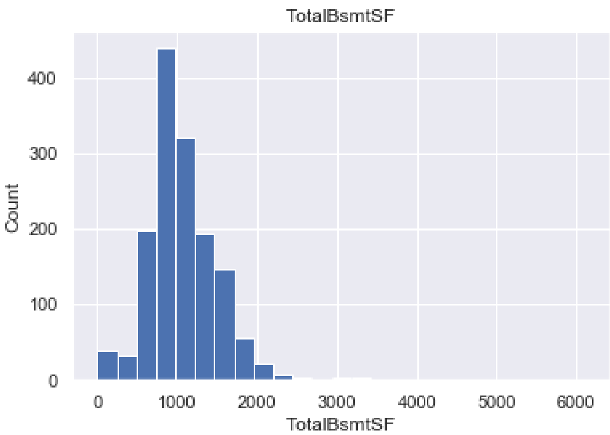


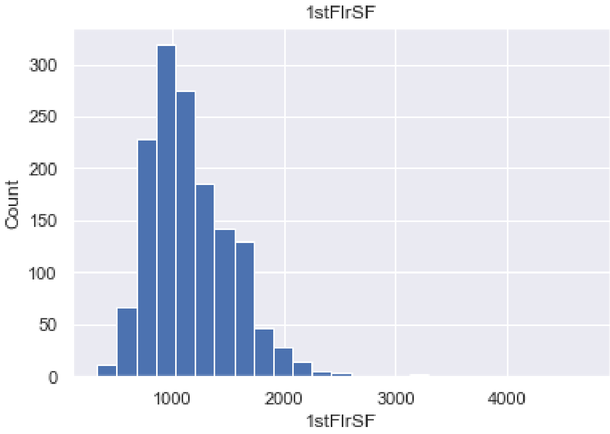

0
0 200 400 600 800 1000 1200 1400 1600
MasVnrArea

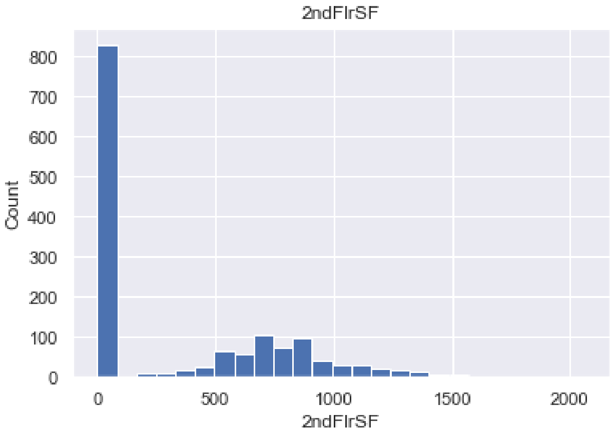

BsmtFinSF1
Count
600 500 400 300 200 100 0
0 1000 2000 3000 4000 5000
BsmtFinSF1

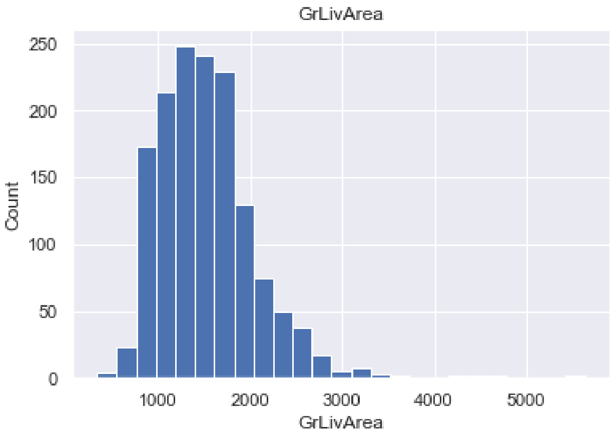

BsmtFinSF2
Count
1200 1000 800 600 400 200 0
0 200 400 600 800 1000 1200 1400
BsmtFinSF2

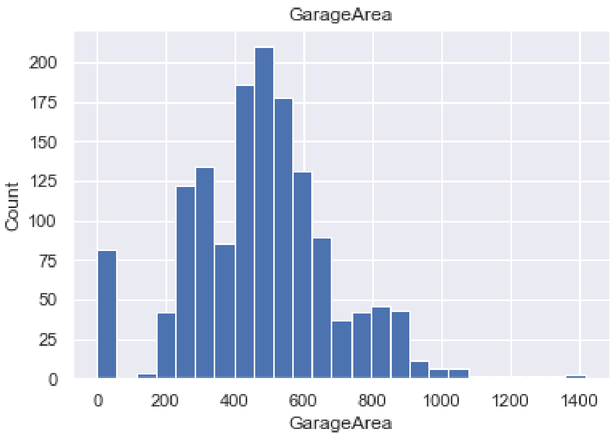

BsmtUnfSF
Count
175 150 125 100 75 50 25 0
0 500 1000 1500 2000
BsmtUnfSF


TotalBsmtSF
Count
400 300 200 100 0
0 1000 2000 3000 4000 5000 6000
TotalBsmtSF

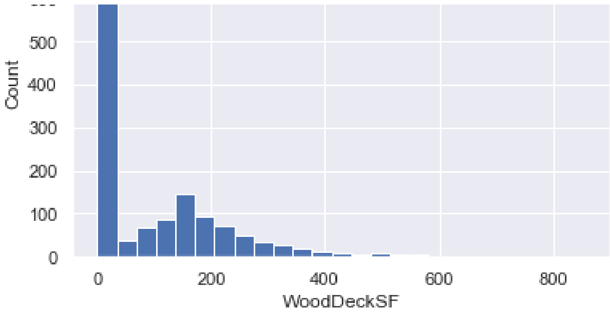

1stFlrSF
Count
300
250
200
150
100
50
0
1000
2000
3000
4000
1stFlrSF

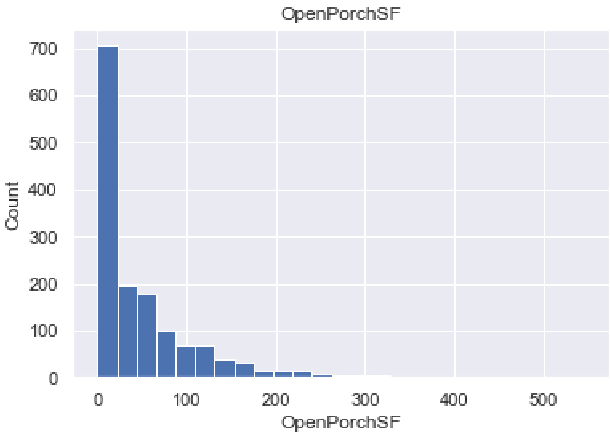

2ndFlrSF
Count
800
700
600
500
400
300
200
100
0
0
500
1000
1500
2000
2ndFlrSF

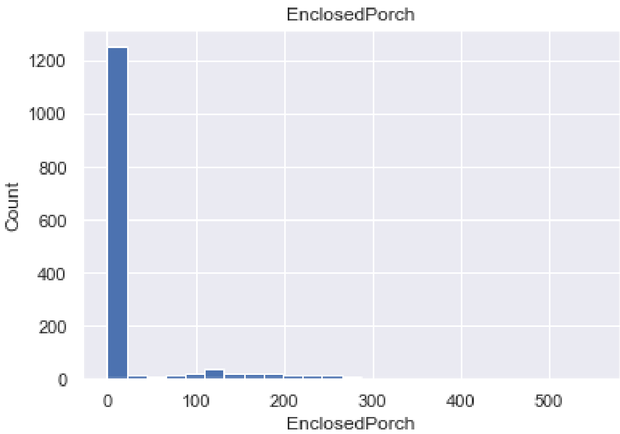

GrLivArea
Count
250
200
150
100
50
0
1000
2000
3000
4000
5000
GrLivArea

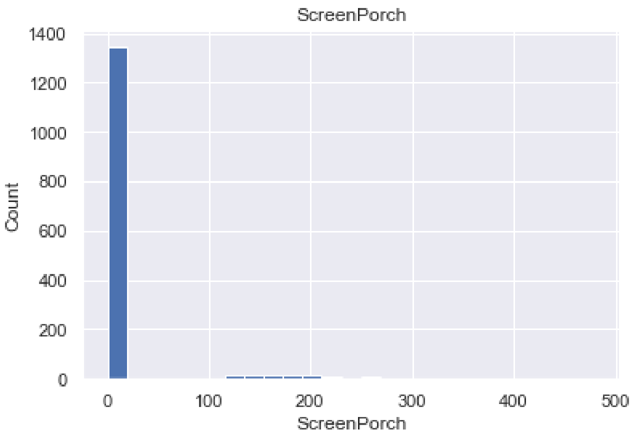

GarageArea
Count
200
175
150
125
100
75
50
25
0
0
200
400
600
800
1000
1200
1400
GarageArea

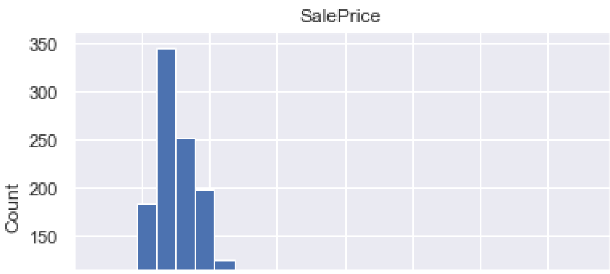

WoodDeckSF
800
700
600

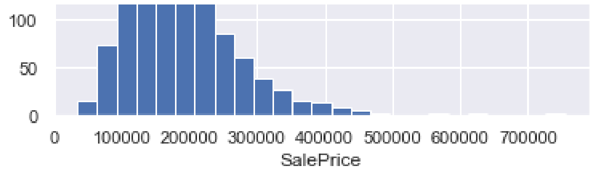

500
400
300
200
100
0
Count
0
200
400
600
800
WoodDeckSF


OpenPorchSF
700
600
500
400
300
200
100
0
Count
0
100
200
300
400
500
OpenPorchSF


EnclosedPorch
1200
1000
800
600
400
200
0
Count
0
100
200
300
400
500
EnclosedPorch


ScreenPorch
1400
1200
1000
800
600
400
200
0
Count
0
100
200
300
400
500
ScreenPorch

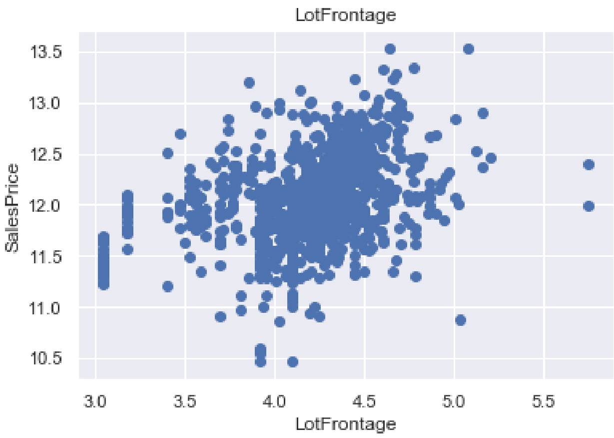

SalePrice
350
300
250
200
150
Count

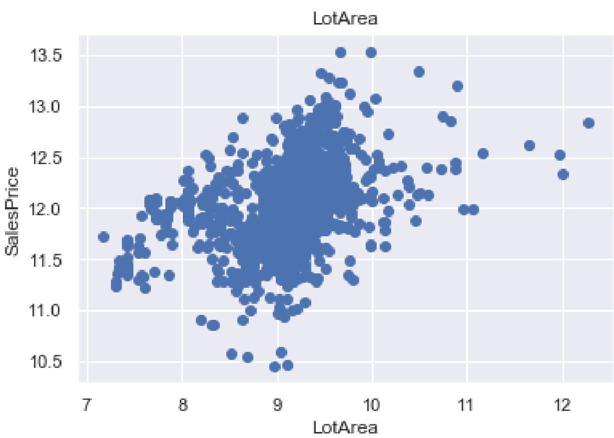


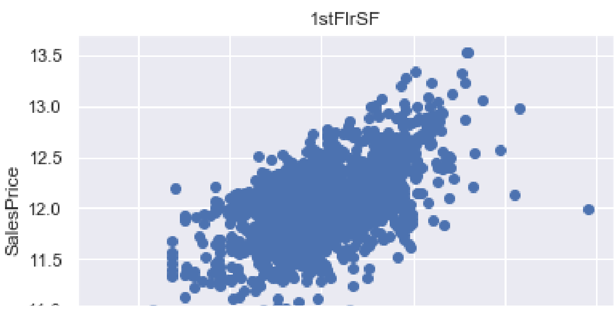
From the above continuous variables we saw that some of the features are not Gaussians Distribution. So it is very important to convert that features into Gaussian's Distribution that's why we are using this logarithmic transformation.

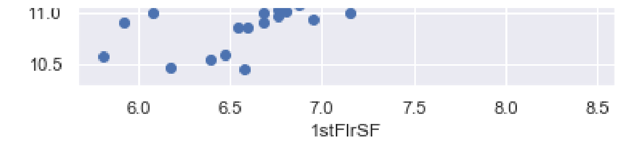
In [38]:

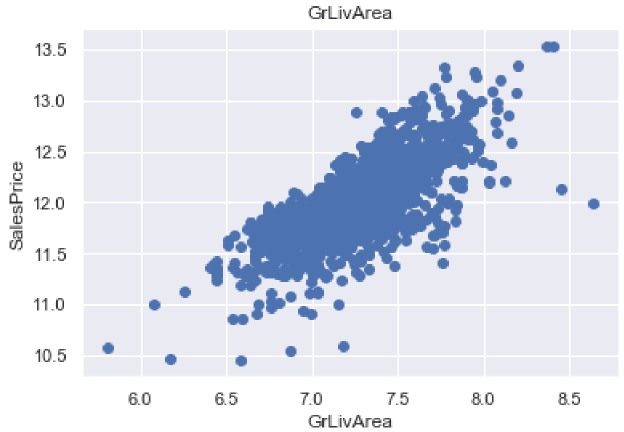
```python
## We will be using logarithmic transformation

for feature in continuous_feature:
    data=train_df.copy()
    if 0 in data[feature].unique():
        pass
    else:
        data[feature]=np.log(data[feature])
        data['SalePrice']=np.log(data['SalePrice'])
        plt.scatter(data[feature],data['SalePrice'])
        plt.xlabel(feature)
        plt.ylabel('SalesPrice')
        plt.title(feature)
        plt.show()
```

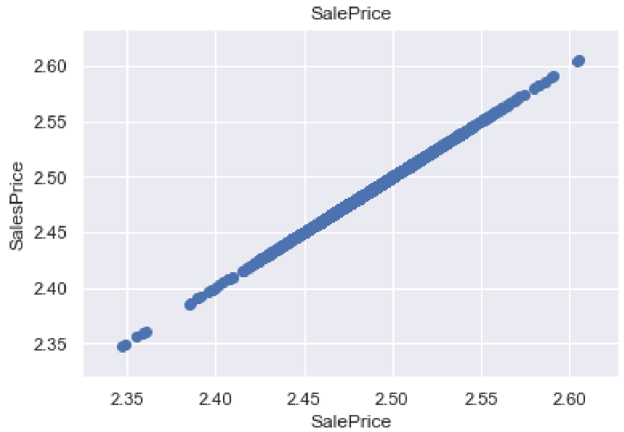








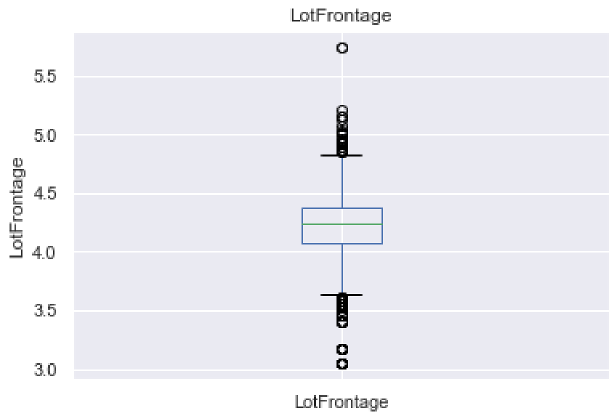


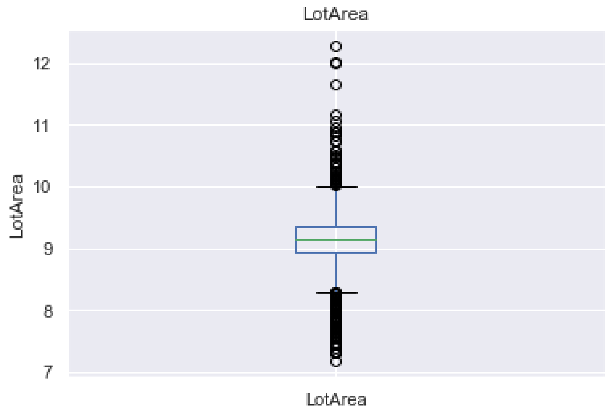


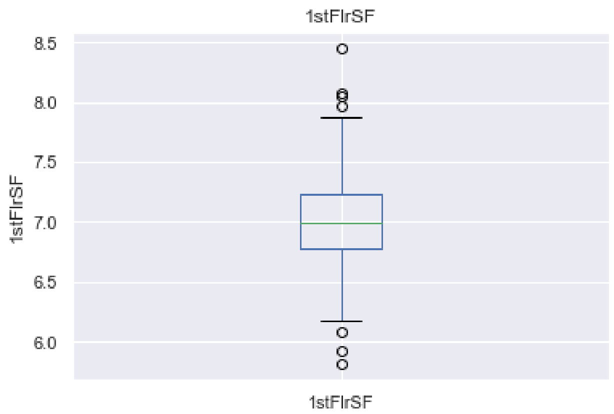
Outliers

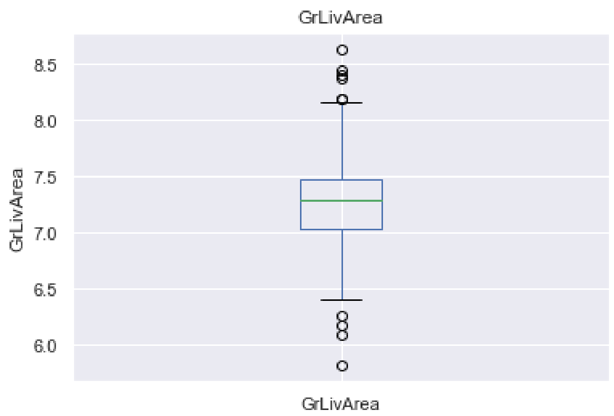
In [39]:

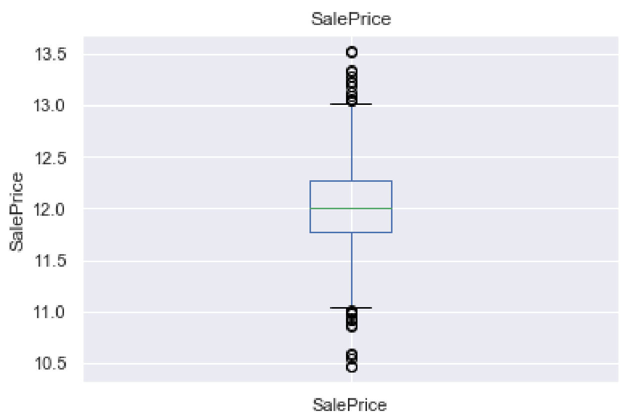
```
for feature in continuous_feature:
    data=train_df.copy()
    if 0 in data[feature].unique():
        pass
    else:
        data[feature]=np.log(data[feature])
        data.boxplot(column=feature)
        plt.ylabel(feature)
        plt.title(feature)
        plt.show()
```











Categorical Variables

In [40]:

```
categorical_features=[feature for feature in train_df.columns if data[feature].dtypes=='O']
categorical_features
```

```
categorical_features
```

Out[40]:

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

In [41]:

```
train_df[categorical_features].head()
```

Out[41]:

	MSZoning	Street	Alley	LotShape	LandContour	Utilities	LotConfig	LandSlope	Neighborhood	Condition1	Condition2	BldgType
0	RL	Pave	NaN	Reg	Lvl	AllPub	Inside	Gtl	CollgCr	Norm	Norm	1Fam
1	RL	Pave	NaN	Reg	Lvl	AllPub	FR2	Gtl	Veenker	Feedr	Norm	1Fam
2	RL	Pave	NaN	IR1	Lvl	AllPub	Inside	Gtl	CollgCr	Norm	Norm	1Fam
3	RL	Pave	NaN	IR1	Lvl	AllPub	Corner	Gtl	Crawfor	Norm	Norm	1Fam
4	RL	Pave	NaN	IR1	Lvl	AllPub	FR2	Gtl	NoRidge	Norm	Norm	1Fam

In [42]:

```
for feature in categorical_features:
    print('The feature is {} and number of categories are {}'.format(feature,len(train_df[feature].
unique())))
```

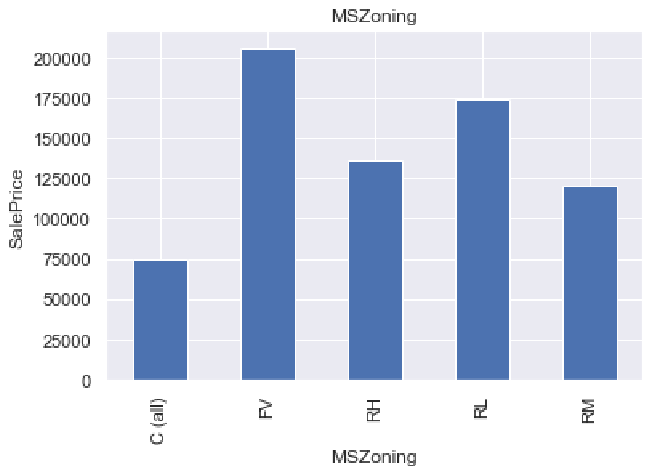
```
The feature is MSZoning and number of categories are 5
The feature is Street and number of categories are 2
The feature is Alley and number of categories are 3
The feature is LotShape and number of categories are 4
```

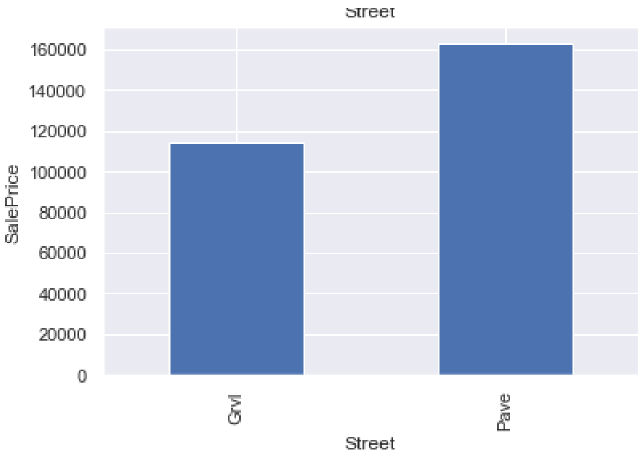
```
The feature is LotShape and number of categories are 4
The feature is LandContour and number of categories are 4
The feature is Utilities and number of categories are 2
The feature is LotConfig and number of categories are 5
The feature is LandSlope and number of categories are 3
The feature is Neighborhood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

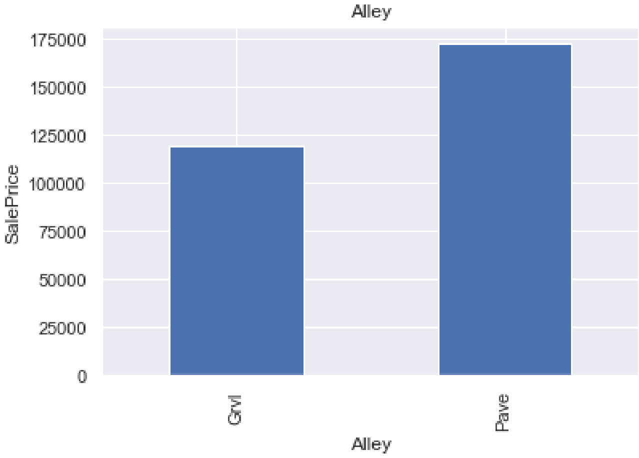
Find out the relationship between categorical variable and dependent feature SalesPrice

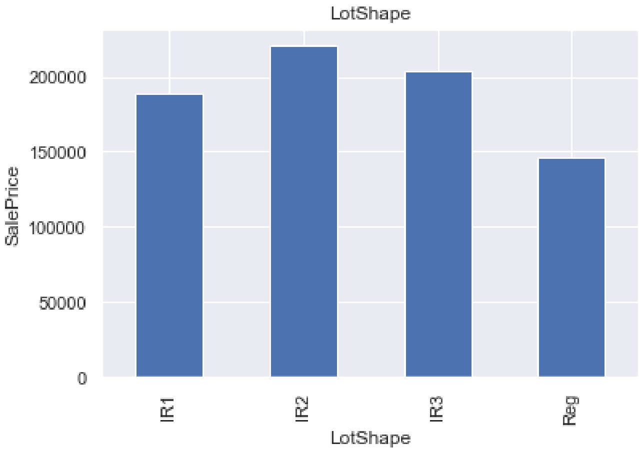
In [43]:

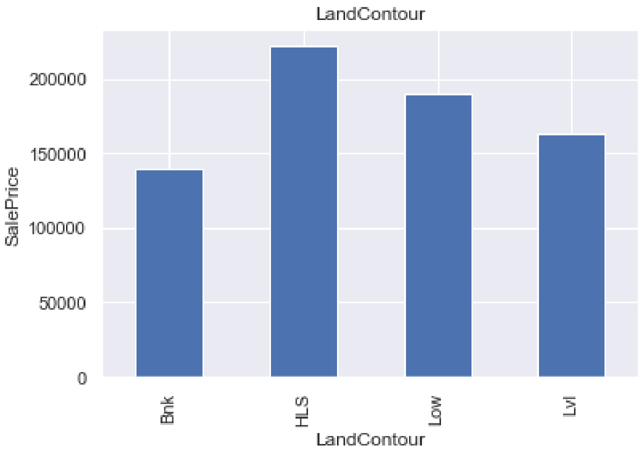
```
for feature in categorical_features:
    data=train_df.copy()
    data.groupby(feature)['SalePrice'].median().plot.bar()
    plt.xlabel(feature)
    plt.ylabel('SalePrice')
    plt.title(feature)
    plt.show()
```

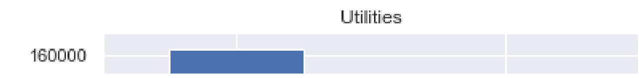


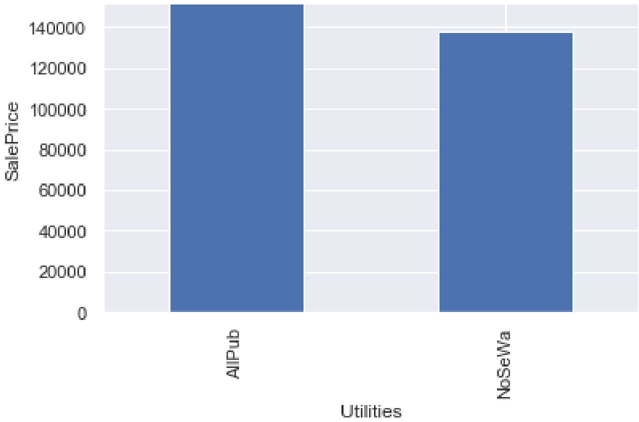

Street
160000
140000
120000
100000
80000
60000
40000
20000
0
SalePrice
Grvl
Pave
Street

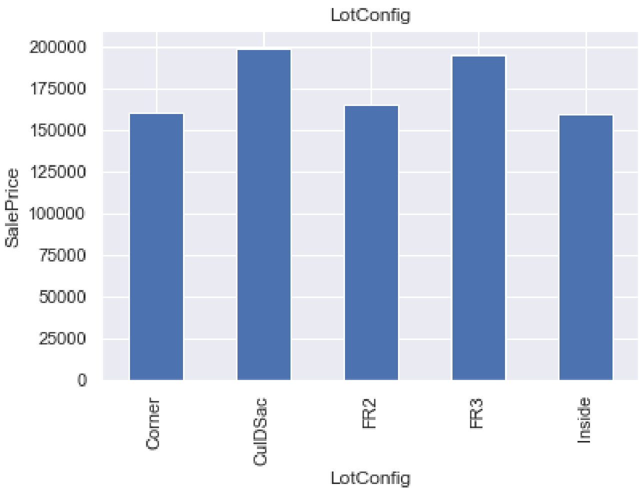

Alley
175000
150000
125000
100000
75000
50000
25000
0
SalePrice
Grvl
Pave
Alley

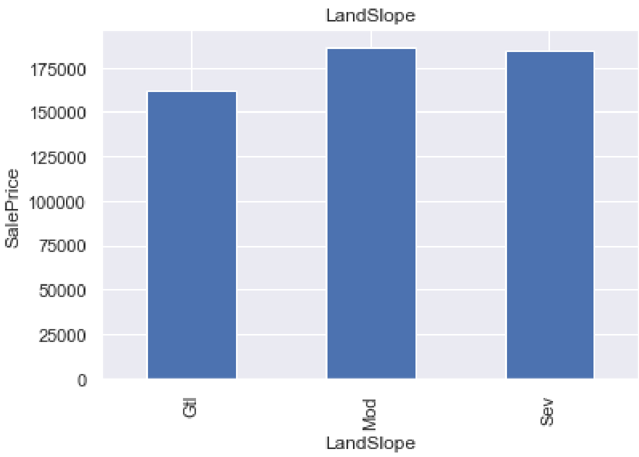

LotShape
200000
150000
100000
50000
0
SalePrice
IR1
IR2
IR3
Reg
LotShape

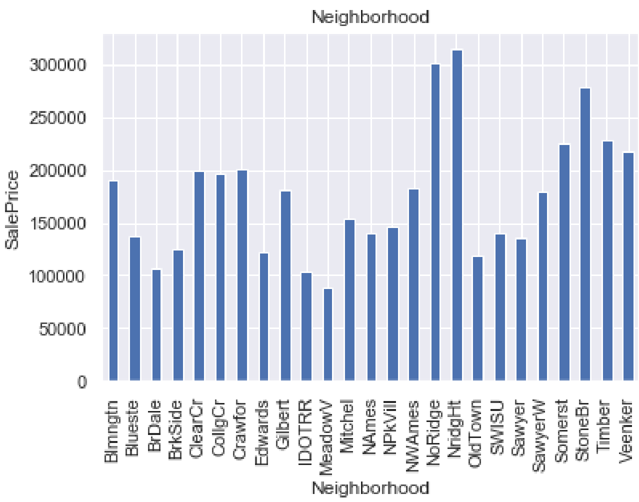

LandContour
200000
150000
100000
50000
0
SalePrice
Bnk
HLS
Low
Lvl
LandContour


Utilities
160000

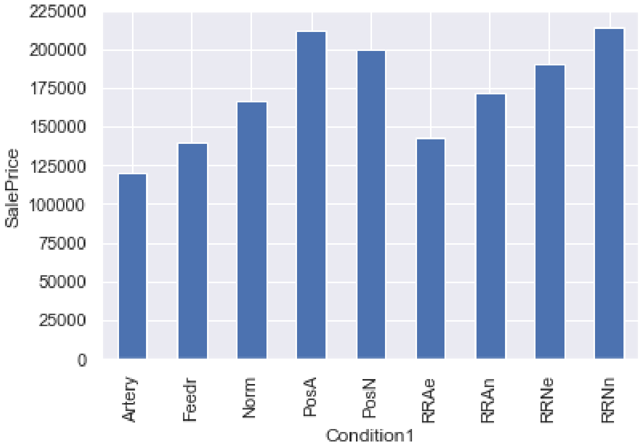

SalePrice
140000
120000
100000
80000
60000
40000
20000
0
AllPub
NoSeWa
Utilities


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

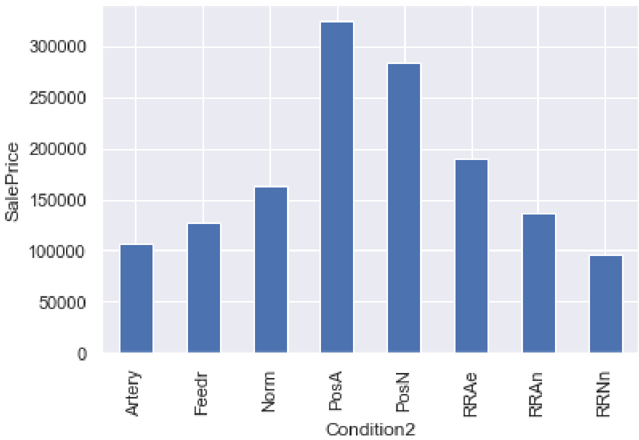

LandSlope
SalePrice
175000
150000
125000
100000
75000
50000
25000
0
Gtl
Mod
Sev
LandSlope


Neighborhood
SalePrice
300000
250000
200000
150000
100000
50000
0
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

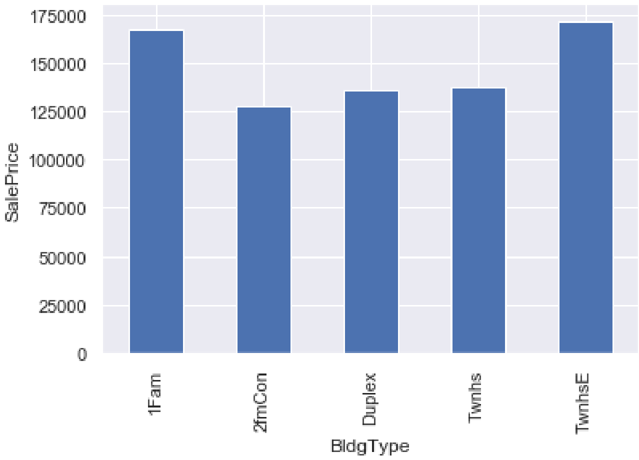
Condition1



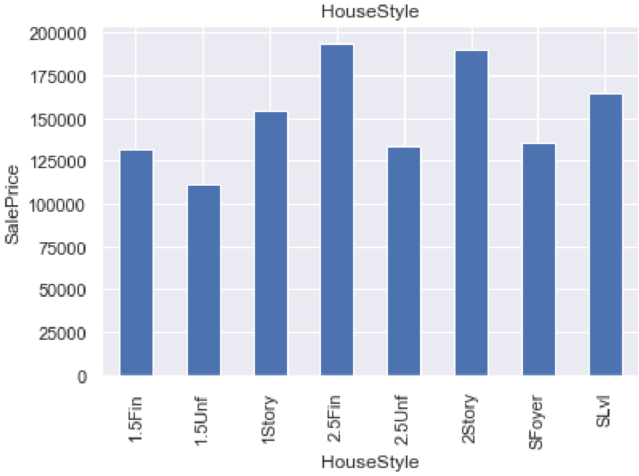
Condition2

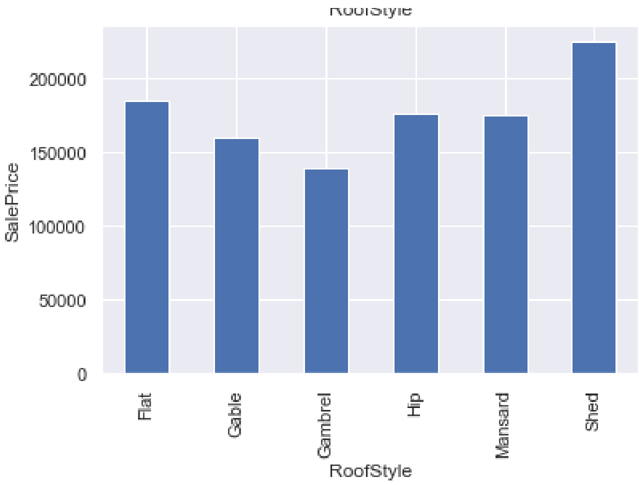


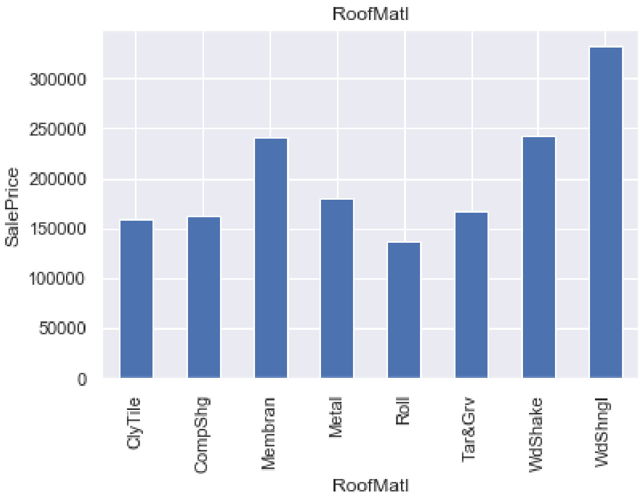
BldgType

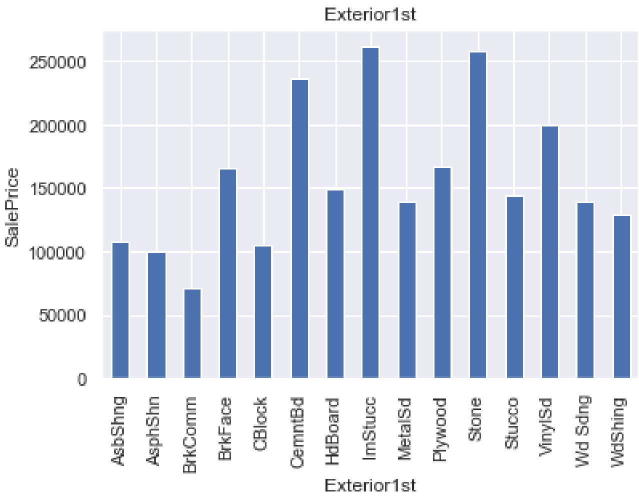


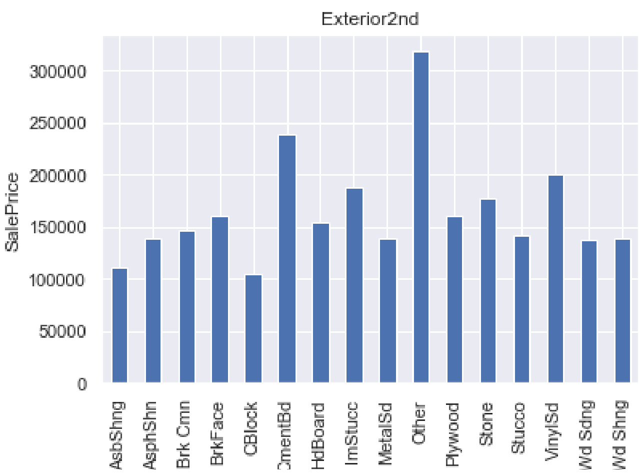
HouseStyle



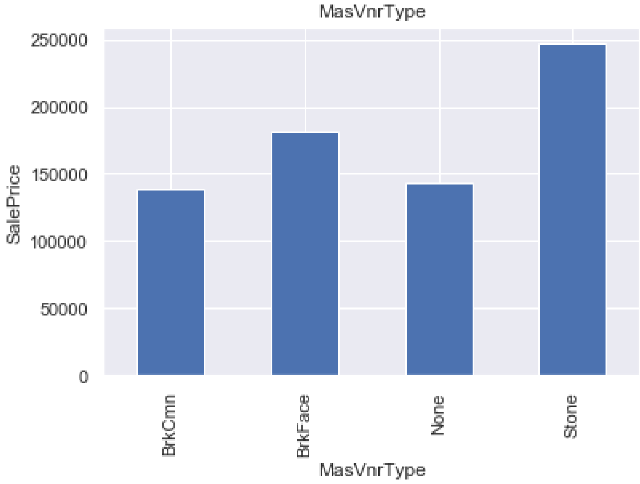

SalePrice
200000
150000
100000
50000
0
Flat
Gable
Gambrel
Hip
Mansard
Shed
RoofStyle

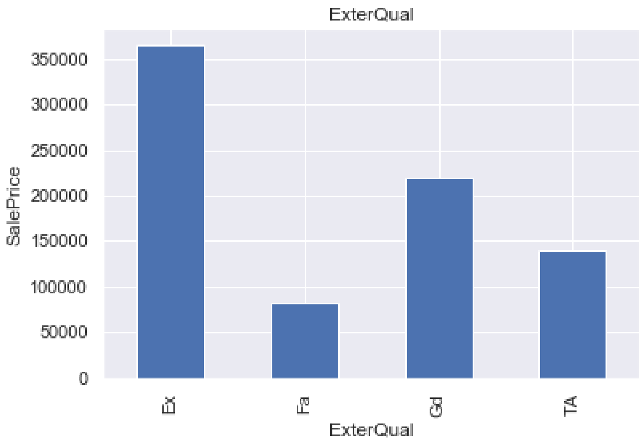

RoofMatl
SalePrice
300000
250000
200000
150000
100000
50000
0
ClyTile
CompShg
Membran
Metal
Roll
Tar&Grv
WdShake
WdShngl
RoofMatl

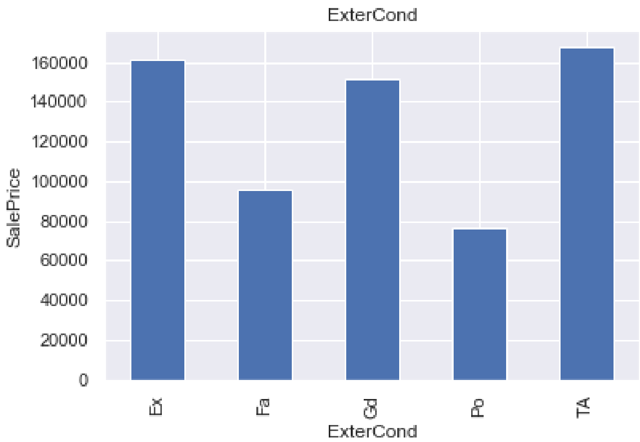

Exterior1st
SalePrice
250000
200000
150000
100000
50000
0
AsbShng
AsphShn
BrkComm
BrkFace
CBlock
CemntBd
HdBoard
ImStucc
MetalSd
Plywood
Stone
Stucco
VinylSd
Wd Sdng
WdShing
Exterior1st

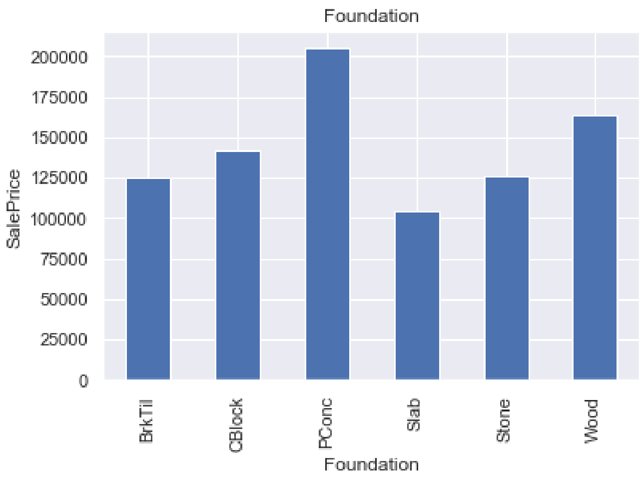

Exterior2nd
SalePrice
300000
250000
200000
150000
100000
50000
0
AsbShng
AsphShn
Brk Cmn
BrkFace
CBlock
HdBoard
ImStucc
MetalSd
Other
Plywood
Stone
Stucco
VinylSd

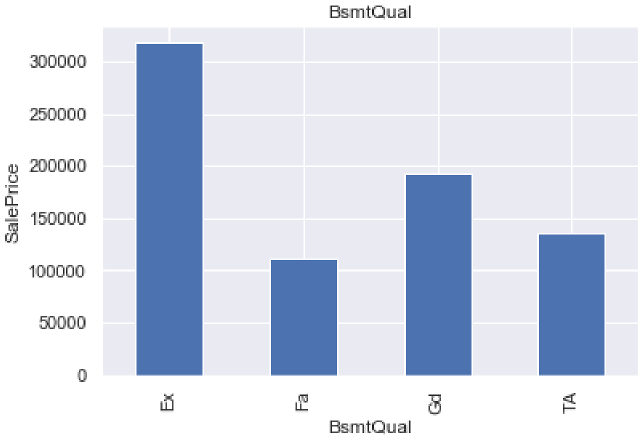
Exterior2nd

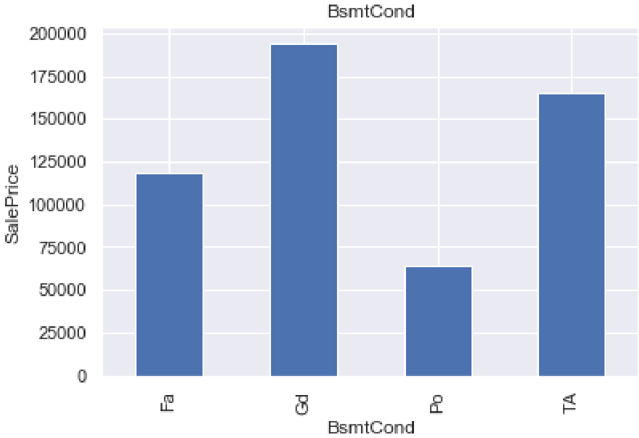


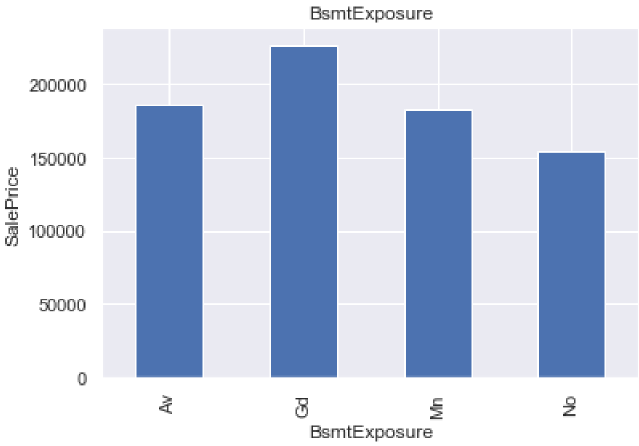


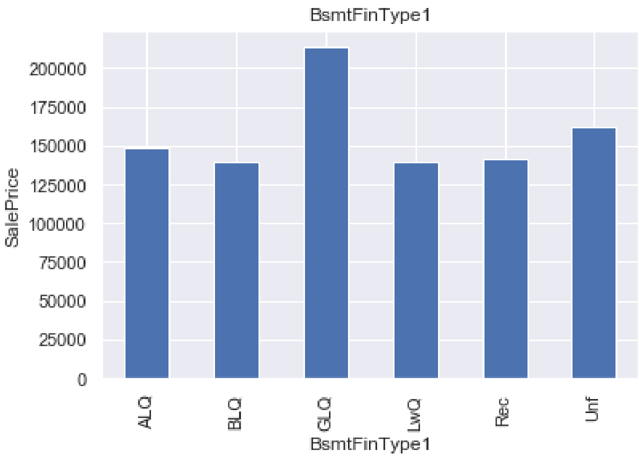


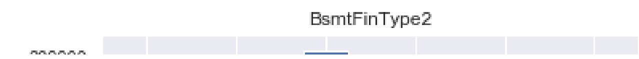


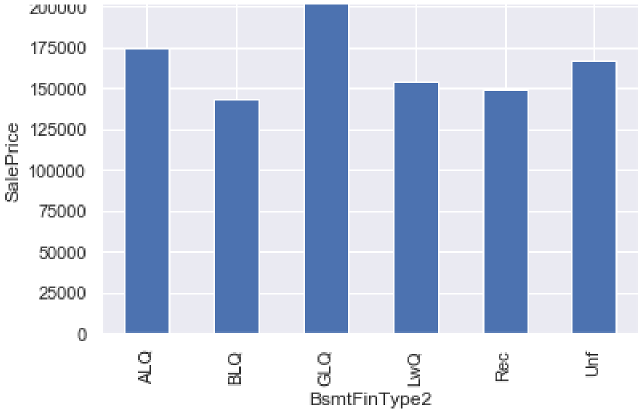

BsmtQual
SalePrice
300000
250000
200000
150000
100000
50000
0
Ex
Fa
Gd
TA
BsmtQual

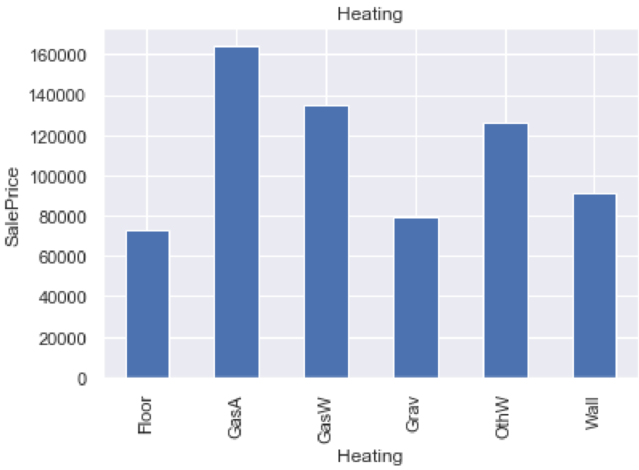

BsmtCond
SalePrice
200000
175000
150000
125000
100000
75000
50000
25000
0
Fa
Gd
Po
TA
BsmtCond

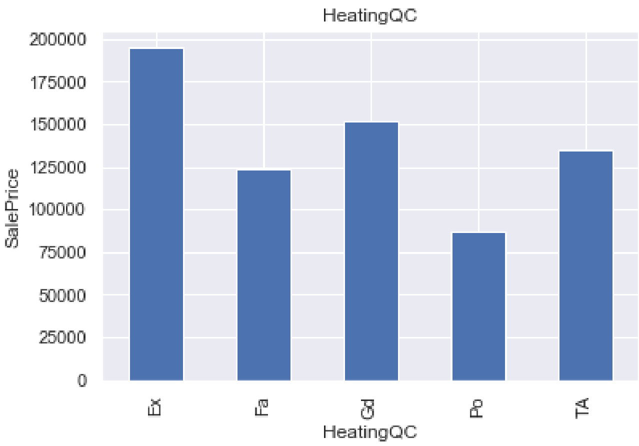

BsmtExposure
SalePrice
200000
150000
100000
50000
0
Av
Gd
Mn
No
BsmtExposure

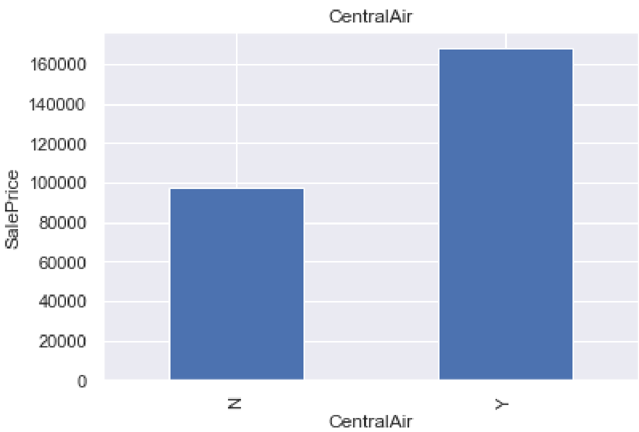

BsmtFinType1
SalePrice
200000
175000
150000
125000
100000
75000
50000
25000
0
ALQ
BLQ
GLQ
LwQ
Rec
Unf
BsmtFinType1

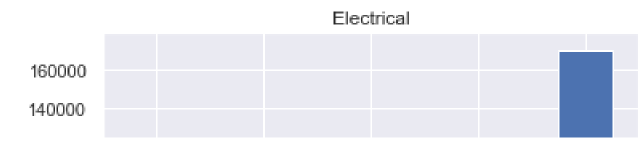

BsmtFinType2

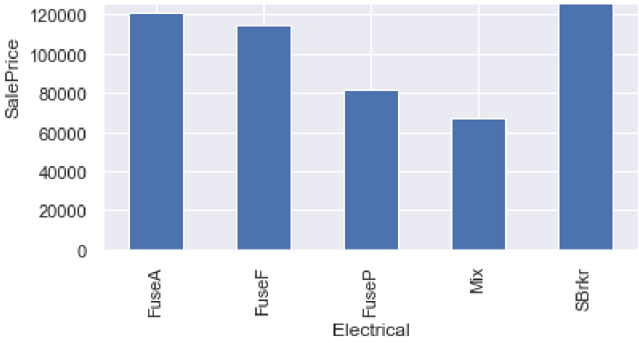

200000
175000
150000
125000
100000
75000
50000
25000
0
SalePrice
ALQ
BLQ
GLQ
LwQ
Rec
Unf
BsmtFinType2

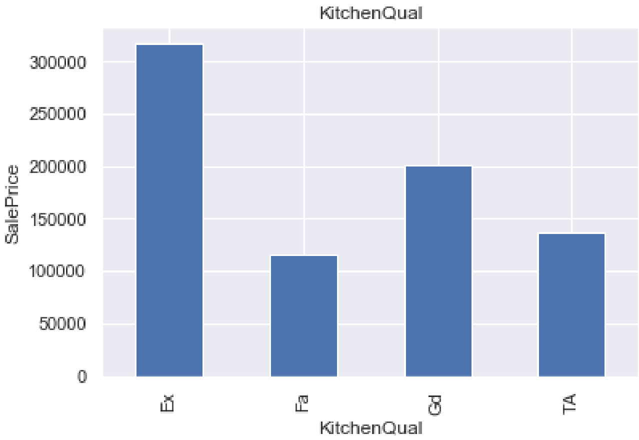

Heating
160000
140000
120000
100000
80000
60000
40000
20000
0
SalePrice
Floor
GasA
GasW
Grav
OthW
Wall
Heating

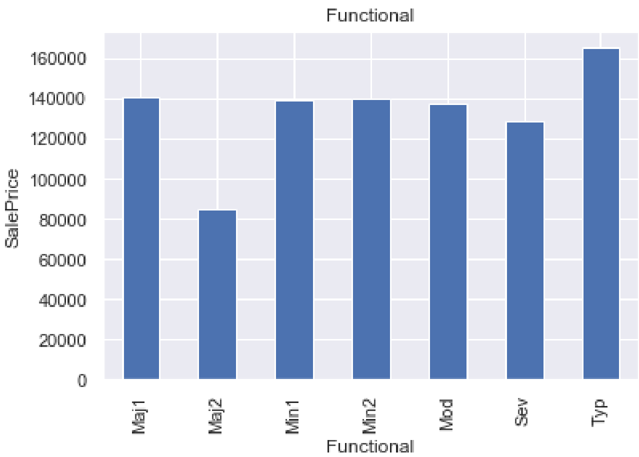

HeatingQC
200000
175000
150000
125000
100000
75000
50000
25000
0
SalePrice
Ex
Fa
Gd
Po
TA
HeatingQC

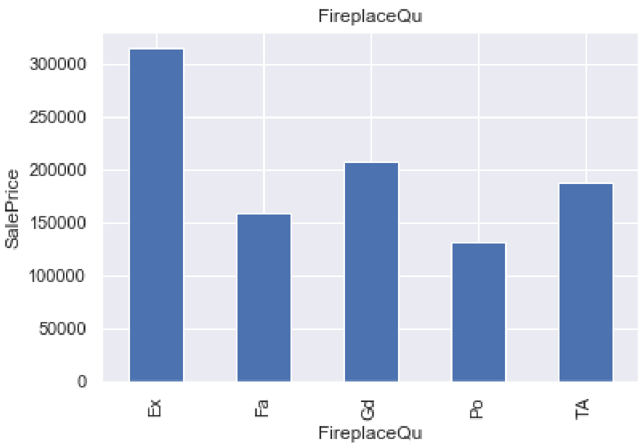

CentralAir
160000
140000
120000
100000
80000
60000
40000
20000
0
SalePrice
N
Y
CentralAir

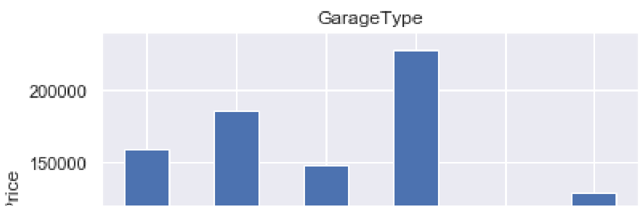

Electrical
160000
140000

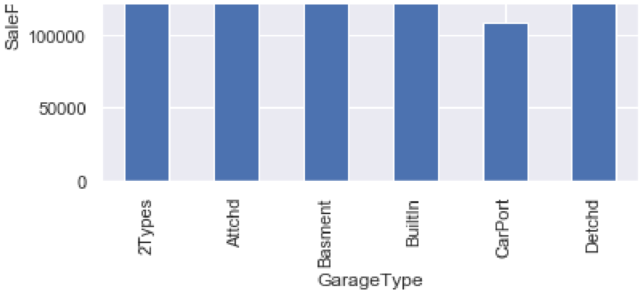

SalePrice
FuseA
FuseF
FuseP
Mix
SBrkr
Electrical

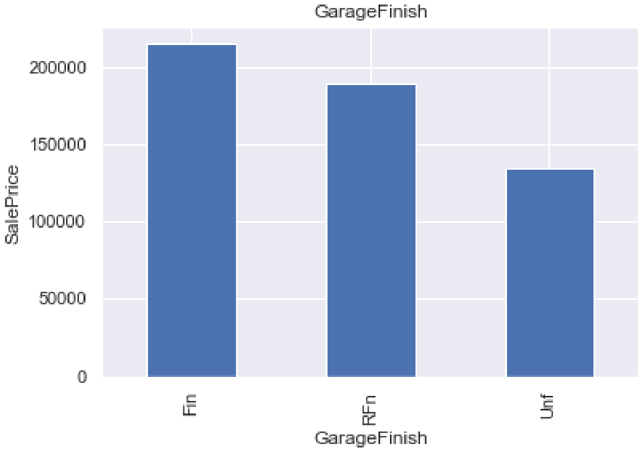

KitchenQual
SalePrice
Ex
Fa
Gd
TA
KitchenQual

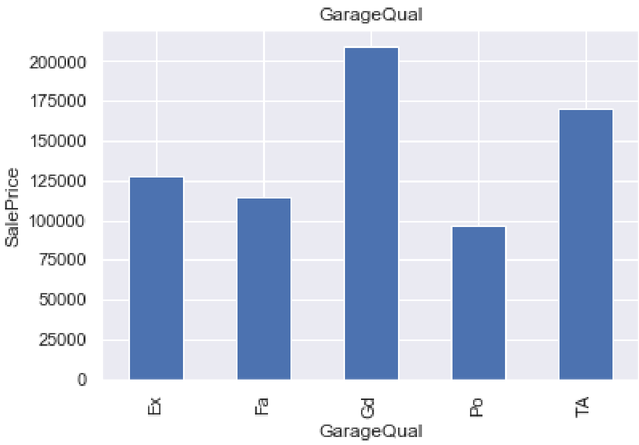

Functional
SalePrice
Maj1
Maj2
Min1
Min2
Mod
Sev
Typ
Functional

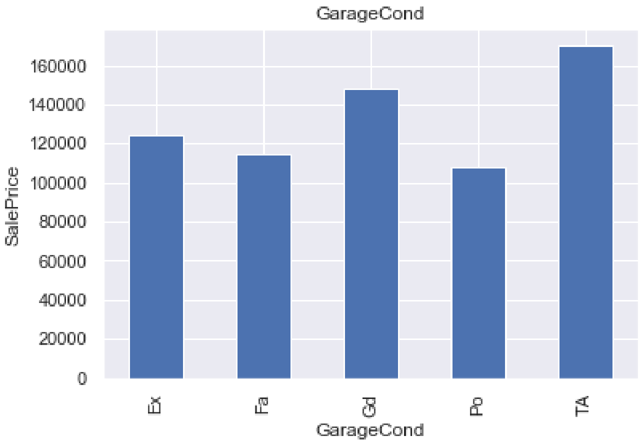

FireplaceQu
SalePrice
Ex
Fa
Gd
Po
TA
FireplaceQu

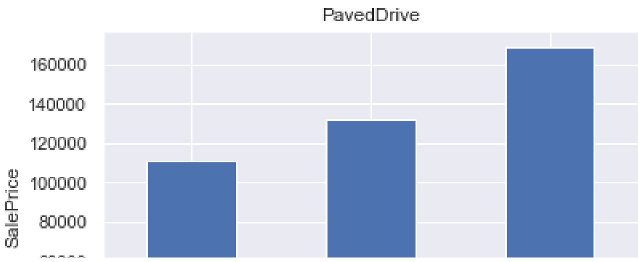

GarageType

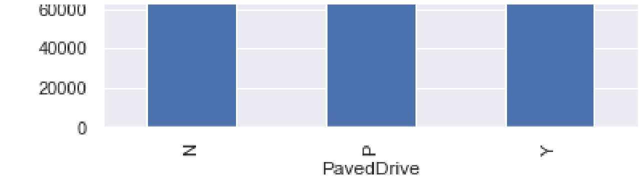

SaleP
100000
50000
0
2Types
Attchd
Basment
BuiltIn
CarPort
Detchd
GarageType

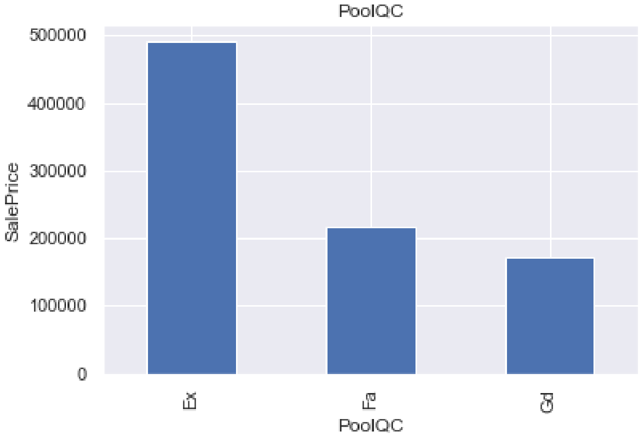

GarageFinish
200000
150000
100000
50000
0
SalePrice
Fin
RFn
Unf
GarageFinish

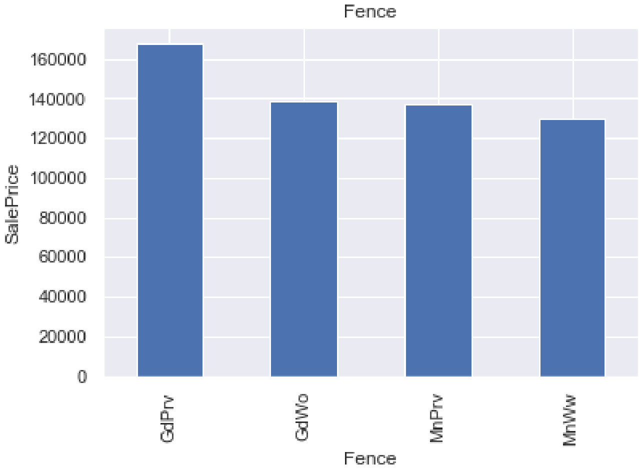

GarageQual
200000
175000
150000
125000
100000
75000
50000
25000
0
SalePrice
Ex
Fa
Gd
Po
TA
GarageQual


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

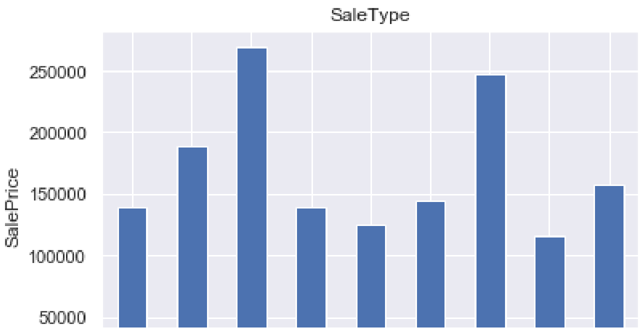

PavedDrive
160000
140000
120000
100000
80000
SalePrice

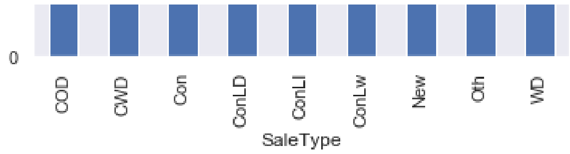

60000
40000
20000
0
N
P
Y
PavedDrive

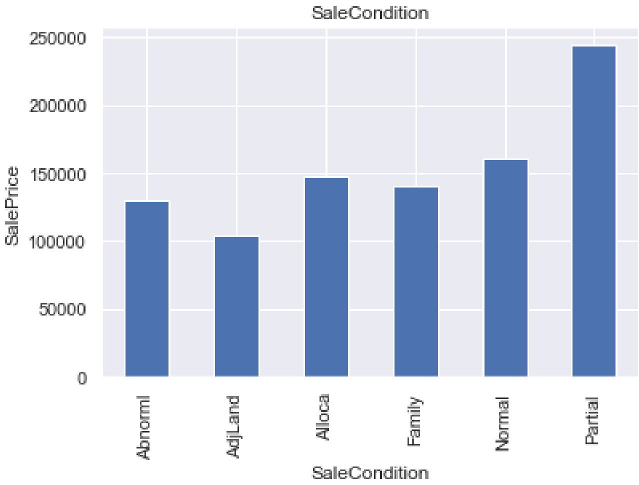

PoolQC
500000
400000
300000
200000
100000
0
SalePrice
Ex
Fa
Gd
PoolQC


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


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


SaleType
250000
200000
150000
100000
50000
SalePrice


0
COD
CWD
Con
ConLD
ConLI
ConLw
New
Oth
WD
SaleType


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

```
In [ ]:
```