Linear_Regression_Housing Price

May 7, 2019

In [56]:
```
# Monirul Islam (5781401)
#invite people for the Kaggle party
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from scipy.stats import norm
from sklearn.preprocessing import StandardScaler
from scipy import stats
import warnings
warnings.filterwarnings('ignore')
%matplotlib inline
plt.style.use('ggplot')
seed = 4432
```

In [57]:
```
#bring in the six packs
df_train = pd.read_csv('train.csv')
```

In [58]:
```
df_train.head()
```

Out[58]:

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

	LandContour	Utilities	...	PoolArea	PoolQC	Fence	MiscFeature	MiscVal	\
0	Lvl	AllPub	...	0	NaN	NaN	NaN	0	
1	Lvl	AllPub	...	0	NaN	NaN	NaN	0	
2	Lvl	AllPub	...	0	NaN	NaN	NaN	0	
3	Lvl	AllPub	...	0	NaN	NaN	NaN	0	
4	Lvl	AllPub	...	0	NaN	NaN	NaN	0	

	MoSold	YrSold	SaleType	SaleCondition	SalePrice
0	2	2008	WD	Normal	208500
1	5	2007	WD	Normal	181500
2	9	2008	WD	Normal	223500

```
3      2   2006        WD         Abnorml     140000
4     12   2008        WD          Normal     250000

[5 rows x 81 columns]
```

In [59]:
```
df_train.describe()
```

Out[59]:

	Id	MSSubClass	LotFrontage	LotArea	OverallQual
count	1460.000000	1460.000000	1201.000000	1460.000000	1460.000000
mean	730.500000	56.897260	70.049958	10516.828082	6.099315
std	421.610009	42.300571	24.284752	9981.264932	1.382997
min	1.000000	20.000000	21.000000	1300.000000	1.000000
25%	365.750000	20.000000	59.000000	7553.500000	5.000000
50%	730.500000	50.000000	69.000000	9478.500000	6.000000
75%	1095.250000	70.000000	80.000000	11601.500000	7.000000
max	1460.000000	190.000000	313.000000	215245.000000	10.000000

	OverallCond	YearBuilt	YearRemodAdd	MasVnrArea	BsmtFinSF1
count	1460.000000	1460.000000	1460.000000	1452.000000	1460.000000
mean	5.575342	1971.267808	1984.865753	103.685262	443.639726
std	1.112799	30.202904	20.645407	181.066207	456.098091
min	1.000000	1872.000000	1950.000000	0.000000	0.000000
25%	5.000000	1954.000000	1967.000000	0.000000	0.000000
50%	5.000000	1973.000000	1994.000000	0.000000	383.500000
75%	6.000000	2000.000000	2004.000000	166.000000	712.250000
max	9.000000	2010.000000	2010.000000	1600.000000	5644.000000

	...	WoodDeckSF	OpenPorchSF	EnclosedPorch	3SsnPorch
count	...	1460.000000	1460.000000	1460.000000	1460.000000
mean	...	94.244521	46.660274	21.954110	3.409589
std	...	125.338794	66.256028	61.119149	29.317331
min	...	0.000000	0.000000	0.000000	0.000000
25%	...	0.000000	0.000000	0.000000	0.000000
50%	...	0.000000	25.000000	0.000000	0.000000
75%	...	168.000000	68.000000	0.000000	0.000000
max	...	857.000000	547.000000	552.000000	508.000000

	ScreenPorch	PoolArea	MiscVal	MoSold	YrSold
count	1460.000000	1460.000000	1460.000000	1460.000000	1460.000000
mean	15.060959	2.758904	43.489041	6.321918	2007.815753
std	55.757415	40.177307	496.123024	2.703626	1.328095
min	0.000000	0.000000	0.000000	1.000000	2006.000000
25%	0.000000	0.000000	0.000000	5.000000	2007.000000
50%	0.000000	0.000000	0.000000	6.000000	2008.000000
75%	0.000000	0.000000	0.000000	8.000000	2009.000000
max	480.000000	738.000000	15500.000000	12.000000	2010.000000

	SalePrice

```
count    1460.000000
mean   180921.195890
std     79442.502883
min     34900.000000
25%    129975.000000
50%    163000.000000
75%    214000.000000
max    755000.000000

[8 rows x 38 columns]
```

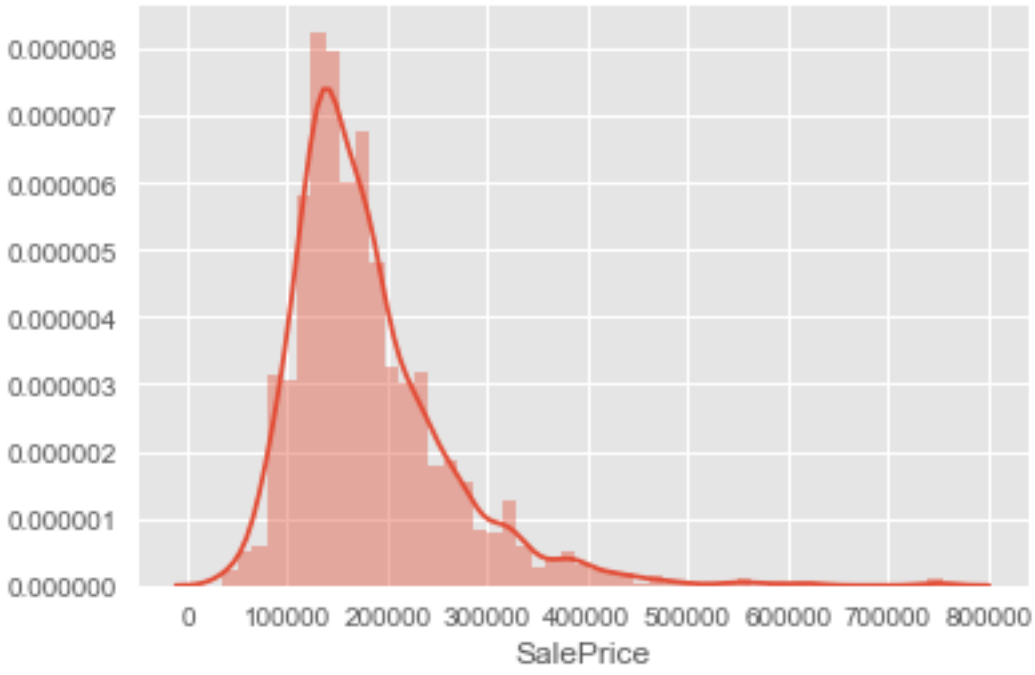
In [60]:
```
#check the decoration
df_train.columns
```

Out[60]:
```
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 [61]:
```
#descriptive statistics summary
df_train['SalePrice'].describe()
```

Out[61]:
```
count      1460.000000
mean     180921.195890
std       79442.502883
min       34900.000000
25%      129975.000000
50%      163000.000000
75%      214000.000000
max      755000.000000
Name: SalePrice, dtype: float64
```

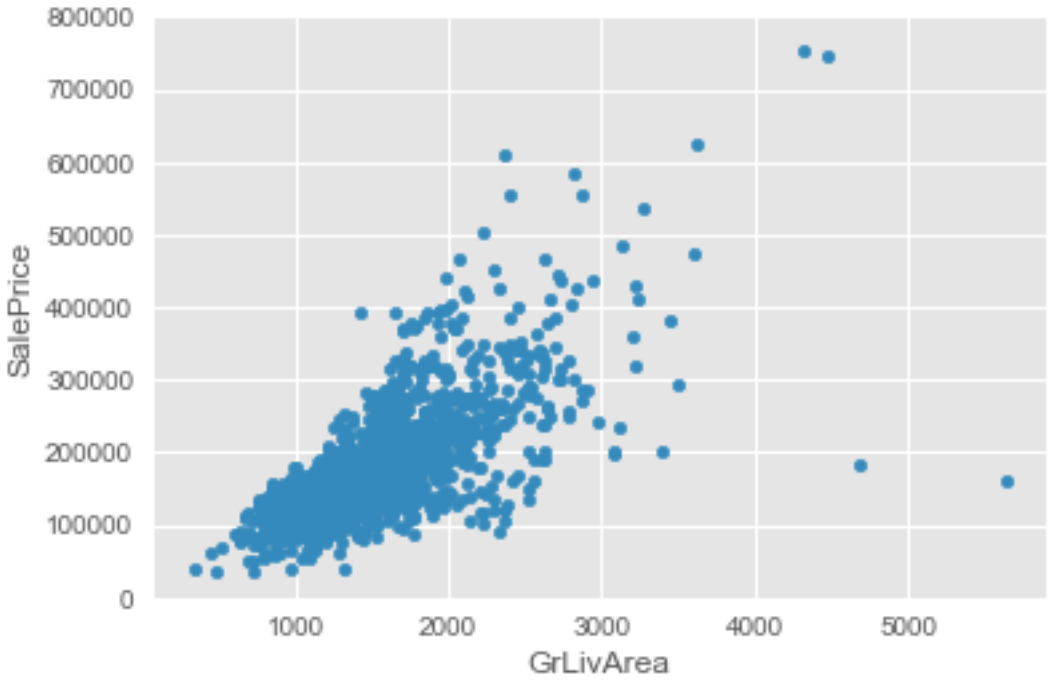
In [62]:
```
#histogram
sns.distplot(df_train['SalePrice']);
```



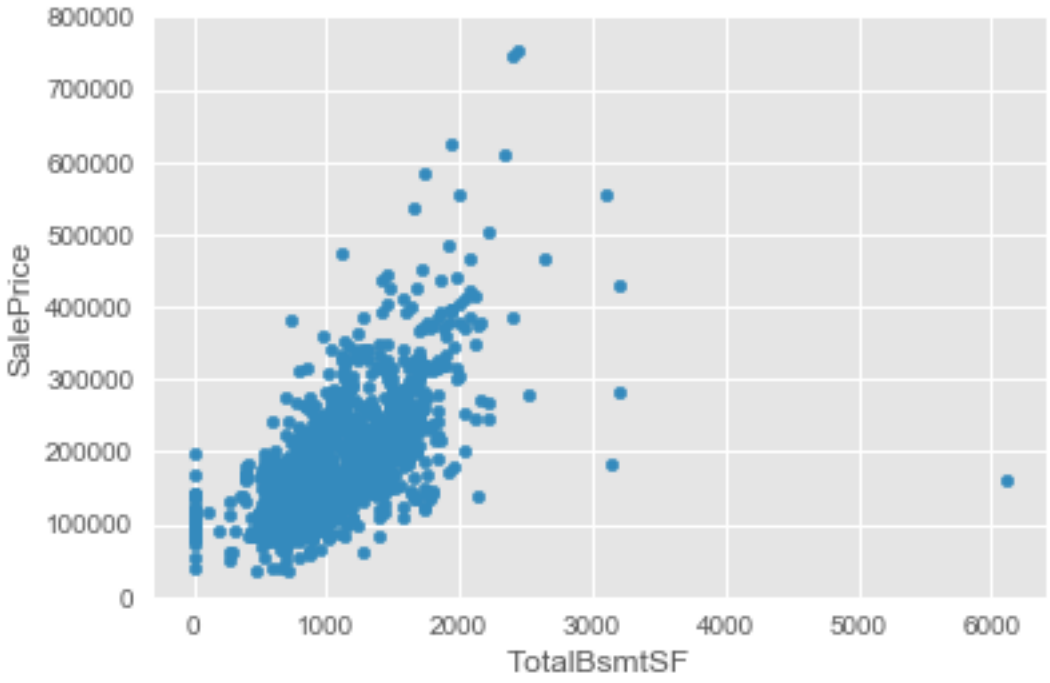

```
In [63]: #skewness and kurtosis
         print("Skewness: %f" % df_train['SalePrice'].skew())
         print("Kurtosis: %f" % df_train['SalePrice'].kurt())
```

```
Skewness: 1.882876
Kurtosis: 6.536282
```

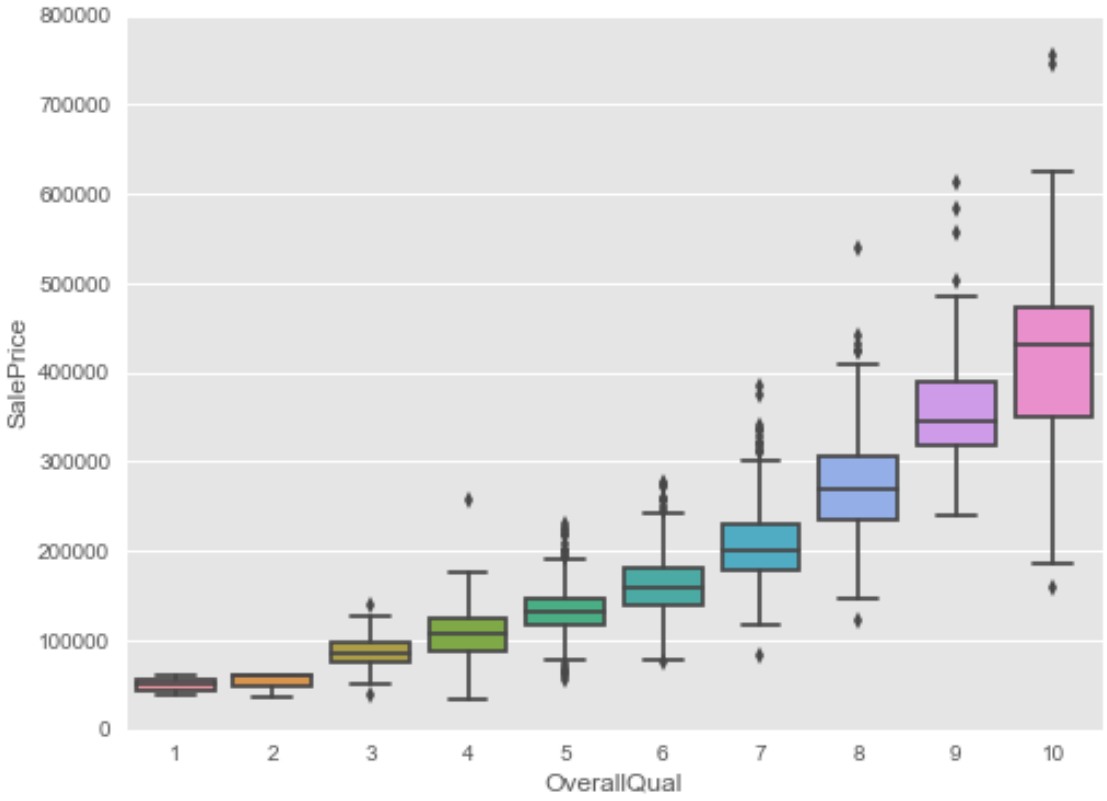
```
In [64]: #scatter plot grlivarea/saleprice
         var = 'GrLivArea'
         data = pd.concat([df_train['SalePrice'], df_train[var]], axis=1)
         data.plot.scatter(x=var, y='SalePrice', ylim=(0,800000));
```



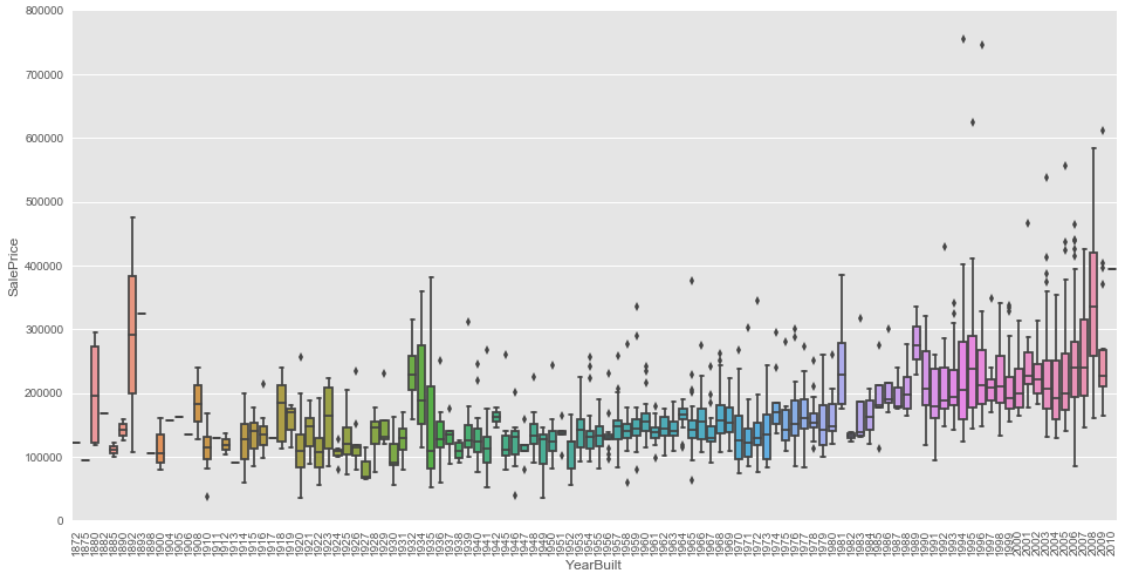
In [65]:
```
#scatter plot totalbsmtsf/saleprice
var = 'TotalBsmtSF'
data = pd.concat([df_train['SalePrice'], df_train[var]], axis=1)
data.plot.scatter(x=var, y='SalePrice', ylim=(0,800000));
```



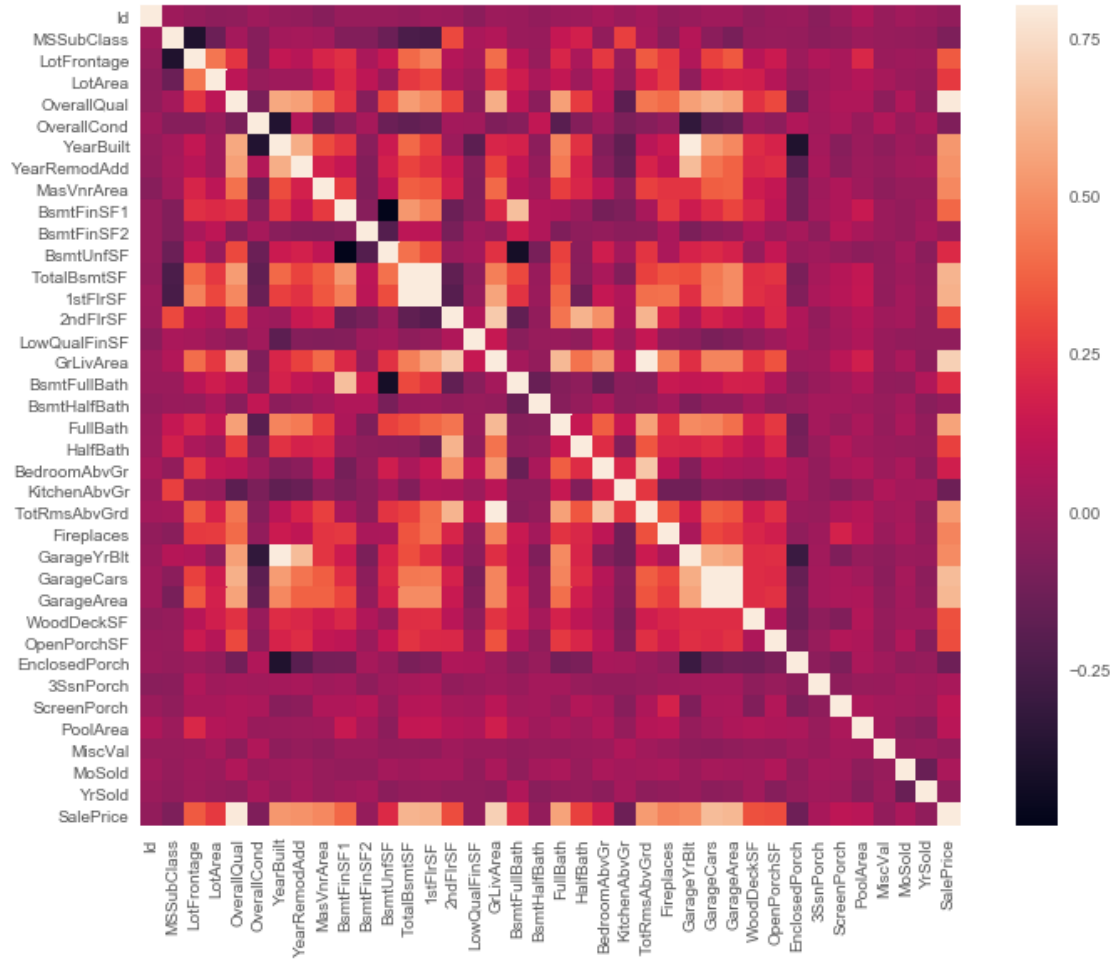
In [66]:
```
#box plot overallqual/saleprice
var = 'OverallQual'
data = pd.concat([df_train['SalePrice'], df_train[var]], axis=1)
f, ax = plt.subplots(figsize=(8, 6))
fig = sns.boxplot(x=var, y="SalePrice", data=data)
fig.axis(ymin=0, ymax=800000);
```



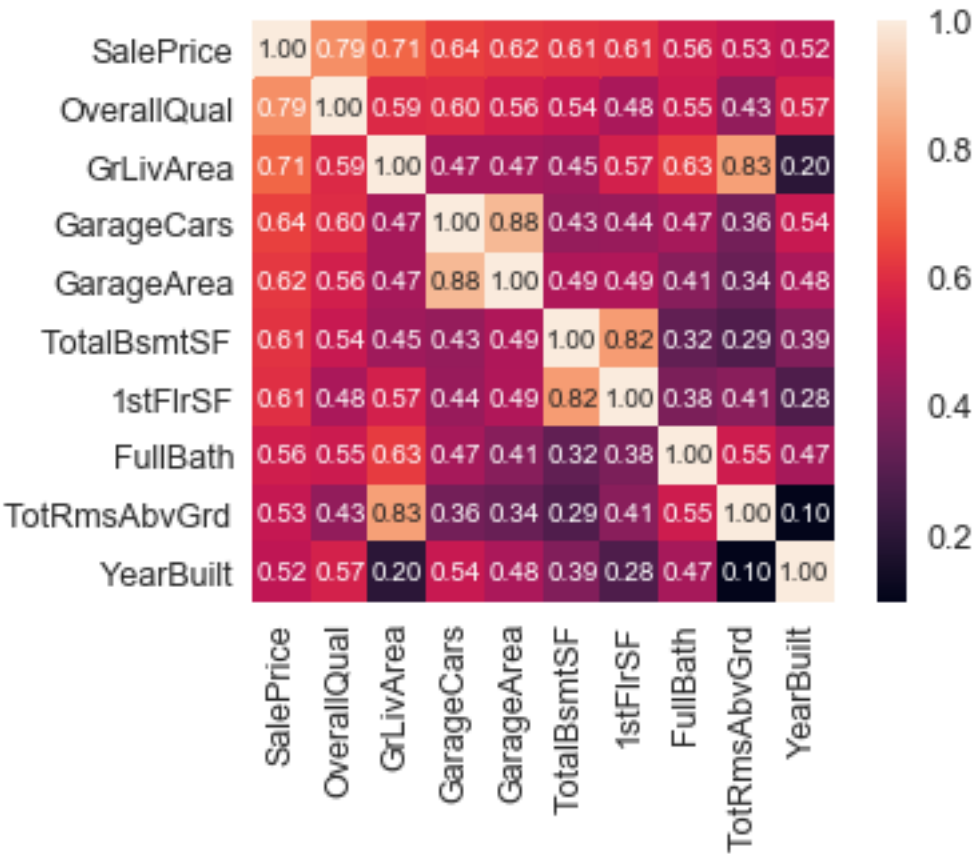
In [67]:
```
var = 'YearBuilt'
data = pd.concat([df_train['SalePrice'], df_train[var]], axis=1)
f, ax = plt.subplots(figsize=(16, 8))
fig = sns.boxplot(x=var, y="SalePrice", data=data)
fig.axis(ymin=0, ymax=800000);
plt.xticks(rotation=90);
```



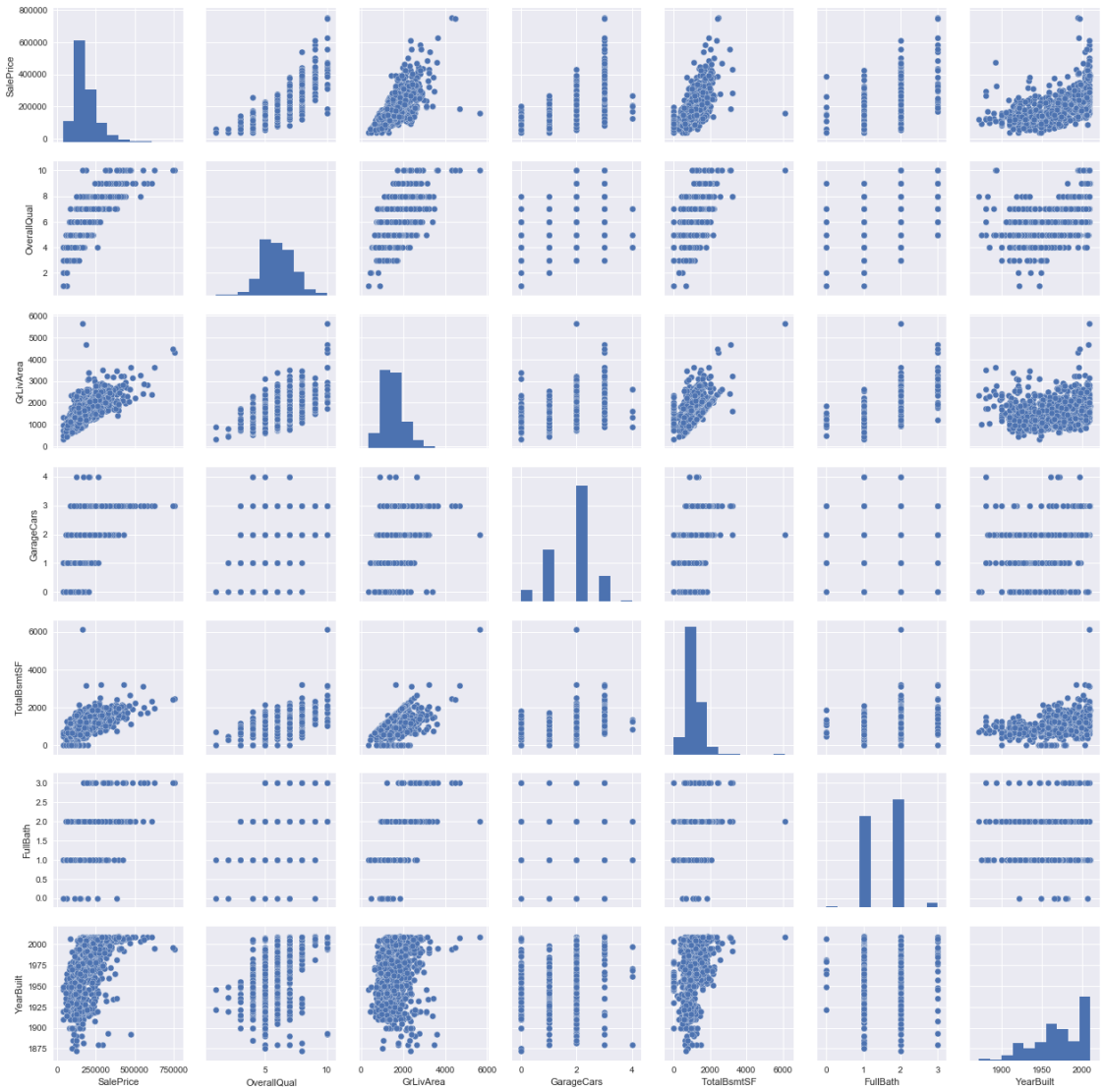
In [68]:
```
#correlation matrix
corrmat = df_train.corr()
f, ax = plt.subplots(figsize=(12, 9))
sns.heatmap(corrmat, vmax=.8, square=True);
```




```
In [69]: #saleprice correlation matrix
         k = 10 #number of variables for heatmap
         cols = corrmat.nlargest(k, 'SalePrice')['SalePrice'].index
         cm = np.corrcoef(df_train[cols].values.T)
         sns.set(font_scale=1.25)
         hm = sns.heatmap(cm, cbar=True, annot=True, square=True, fmt='.2f', annot_kws={'size'
         plt.show()
```



In [70]:
```
#scatterplot
sns.set()
cols = ['SalePrice', 'OverallQual', 'GrLivArea', 'GarageCars', 'TotalBsmtSF', 'FullBa
sns.pairplot(df_train[cols], size = 2.5)
plt.show();
```




```
In [71]: #missing data
         total = df_train.isnull().sum().sort_values(ascending=False)
         percent = (df_train.isnull().sum()/df_train.isnull().count()).sort_values(ascending=Fa
         missing_data = pd.concat([total, percent], axis=1, keys=['Total', 'Percent'])
         missing_data.head(20)
```

Out[71]:

	Total	Percent
PoolQC	1453	0.995205
MiscFeature	1406	0.963014
Alley	1369	0.937671
Fence	1179	0.807534
FireplaceQu	690	0.472603
LotFrontage	259	0.177397
GarageCond	81	0.055479

```
GarageType       81  0.055479
GarageYrBlt      81  0.055479
GarageFinish     81  0.055479
GarageQual       81  0.055479
BsmtExposure     38  0.026027
BsmtFinType2     38  0.026027
BsmtFinType1     37  0.025342
BsmtCond         37  0.025342
BsmtQual         37  0.025342
MasVnrArea        8  0.005479
MasVnrType        8  0.005479
Electrical        1  0.000685
Utilities         0  0.000000
```

```
In [72]: #dealing with missing data
         df_train = df_train.drop((missing_data[missing_data['Total'] > 1]).index,1)
         df_train = df_train.drop(df_train.loc[df_train['Electrical'].isnull()].index)
         df_train.isnull().sum().max() #just checking that there's no missing data missing...
```

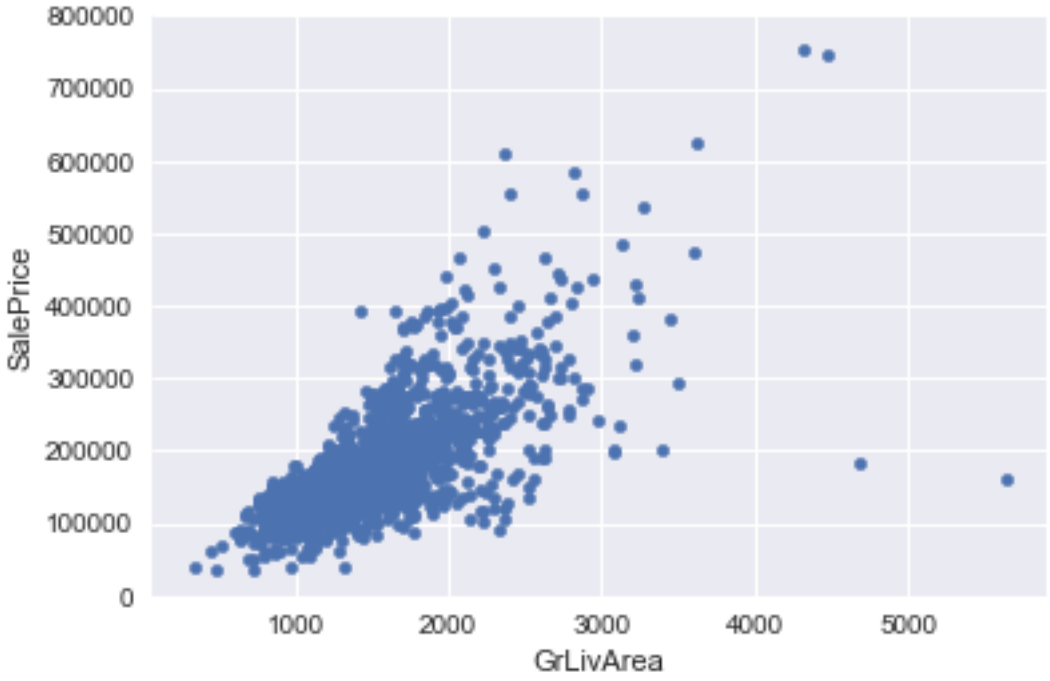
```
Out[72]: 0
```

```
In [73]: #standardizing data
         saleprice_scaled = StandardScaler().fit_transform(df_train['SalePrice'][:,np.newaxis])
         low_range = saleprice_scaled[saleprice_scaled[:,0].argsort()][:10]
         high_range= saleprice_scaled[saleprice_scaled[:,0].argsort()][-10:]
         print('outer range (low) of the distribution:')
         print(low_range)
         print('\nouter range (high) of the distribution:')
         print(high_range)
```

```
outer range (low) of the distribution:
[[-1.83820775]
 [-1.83303414]
 [-1.80044422]
 [-1.78282123]
 [-1.77400974]
 [-1.62295562]
 [-1.6166617 ]
 [-1.58519209]
 [-1.58519209]
 [-1.57269236]]

outer range (high) of the distribution:
[[ 3.82758058]
 [ 4.0395221 ]
 [ 4.49473628]
 [ 4.70872962]
 [ 4.728631  ]
 [ 5.06034585]
```

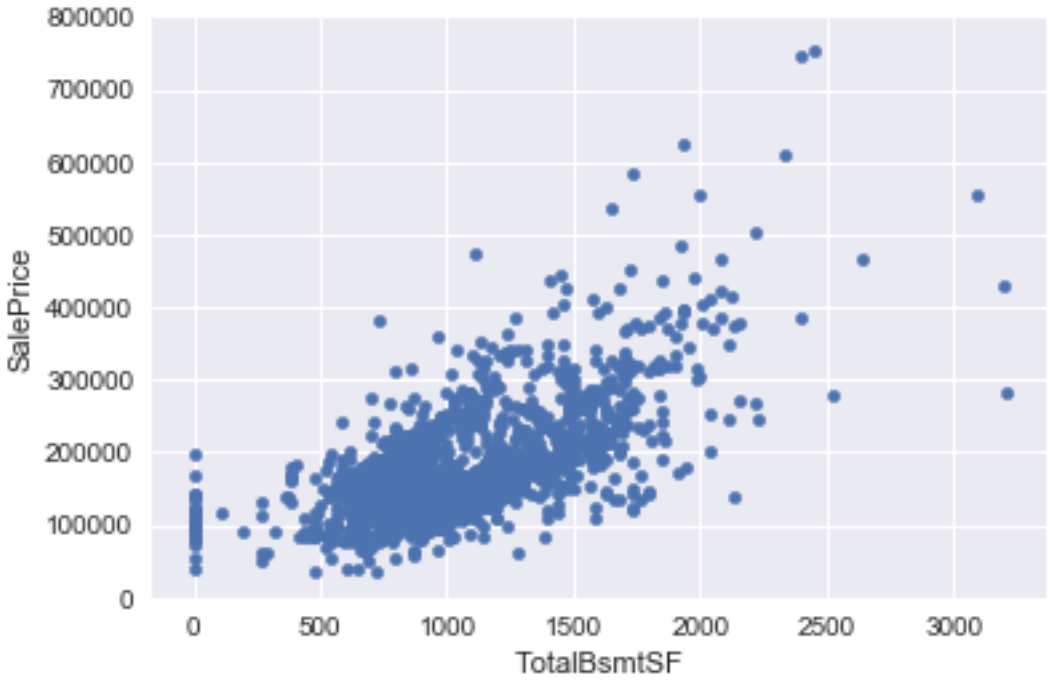
```
 [ 5.42191907]
 [ 5.58987866]
 [ 7.10041987]
 [ 7.22629831]]
```

In [74]:
```
#bivariate analysis saleprice/grlivarea
var = 'GrLivArea'
data = pd.concat([df_train['SalePrice'], df_train[var]], axis=1)
data.plot.scatter(x=var, y='SalePrice', ylim=(0,800000));
```

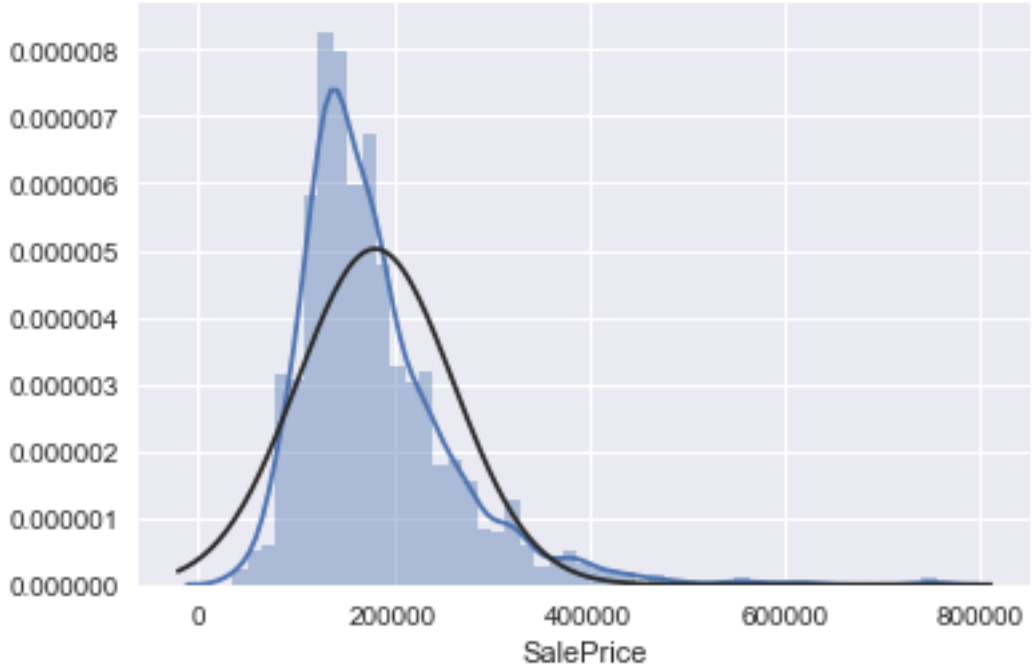


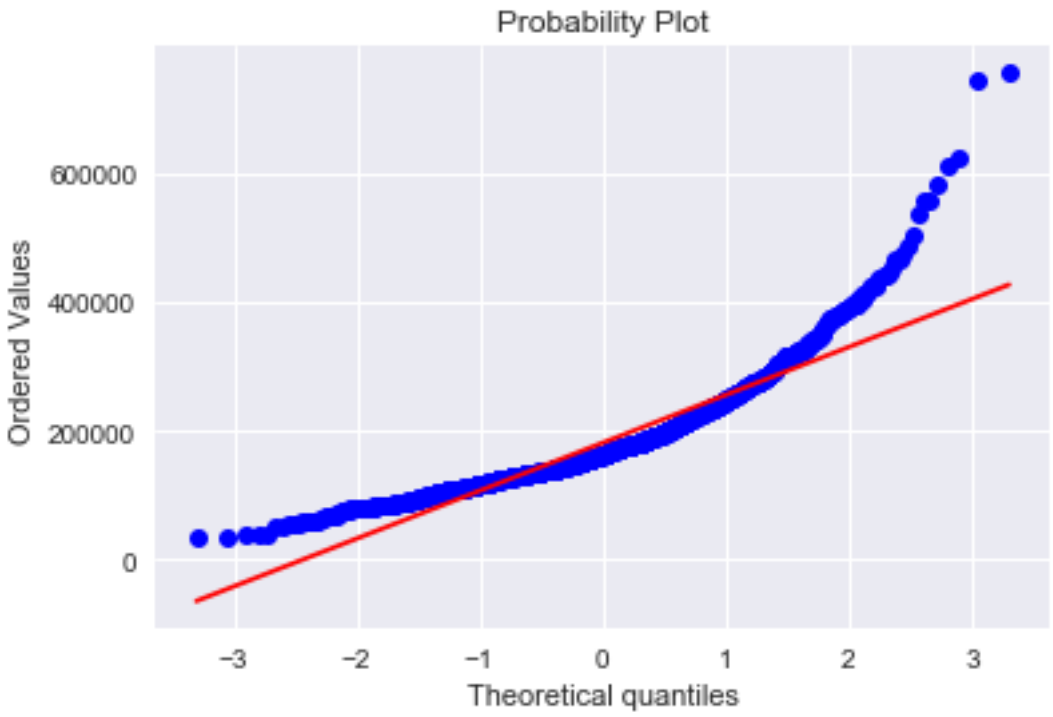
In [75]:
```
#deleting points
df_train.sort_values(by = 'GrLivArea', ascending = False)[:2]
df_train = df_train.drop(df_train[df_train['Id'] == 1299].index)
df_train = df_train.drop(df_train[df_train['Id'] == 524].index)
```

In [76]:
```
#bivariate analysis saleprice/grlivarea
var = 'TotalBsmtSF'
data = pd.concat([df_train['SalePrice'], df_train[var]], axis=1)
data.plot.scatter(x=var, y='SalePrice', ylim=(0,800000));
```

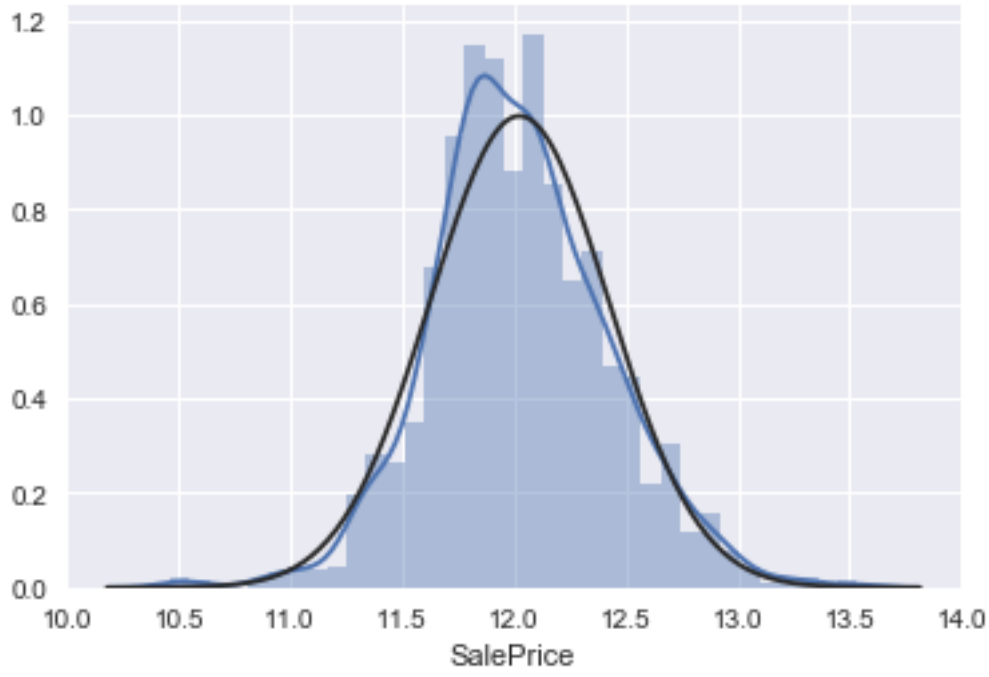


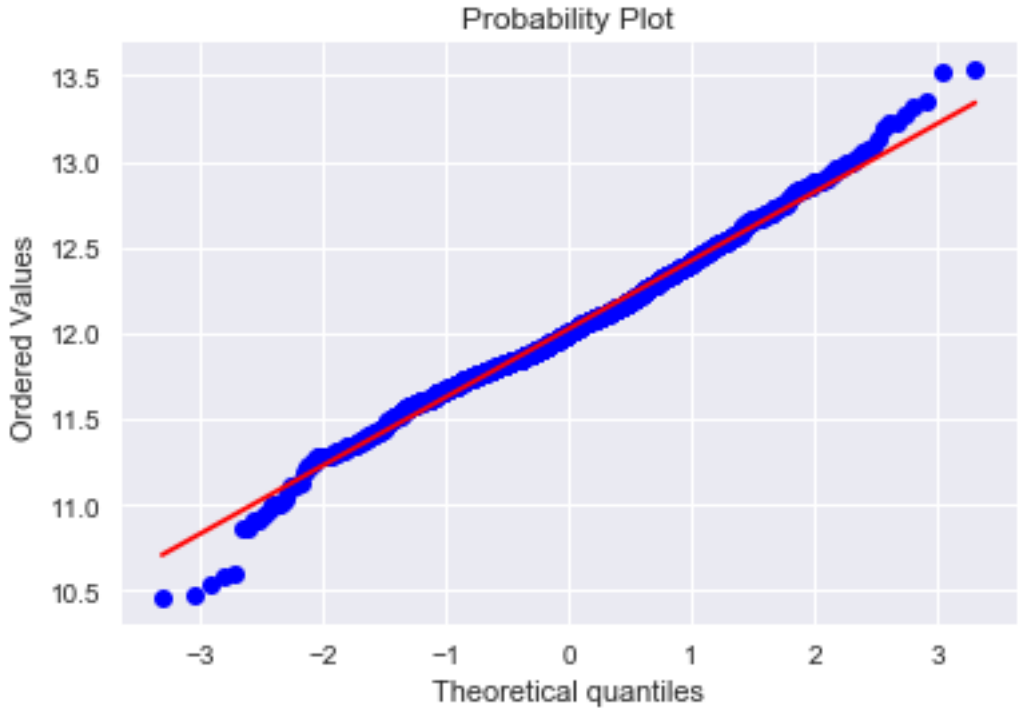
In [77]:
```
#histogram and normal probability plot
sns.distplot(df_train['SalePrice'], fit=norm);
fig = plt.figure()
res = stats.probplot(df_train['SalePrice'], plot=plt)
```





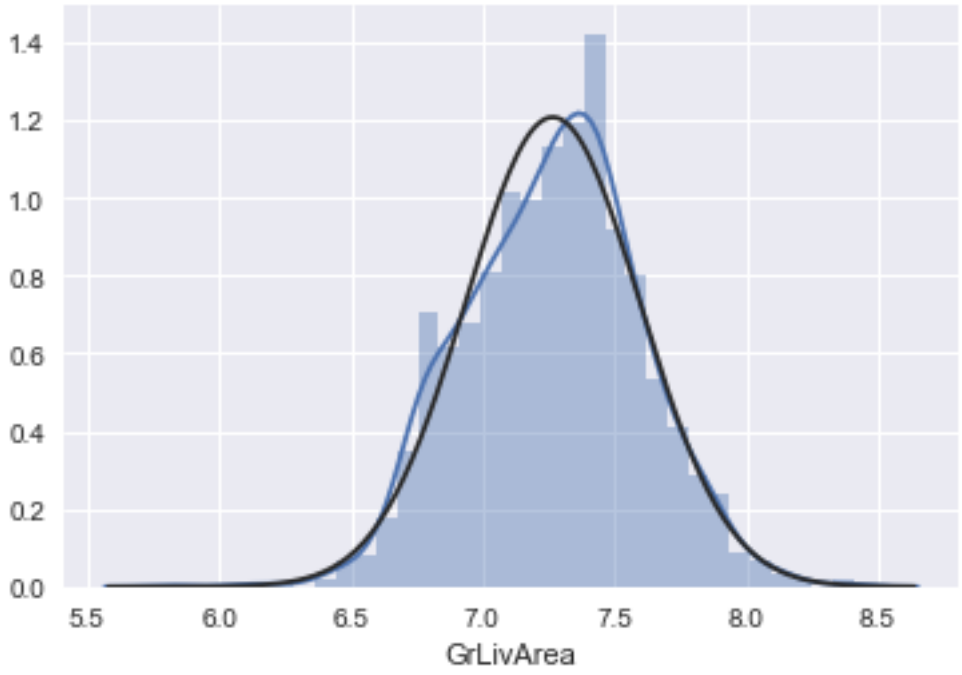

```
In [78]: #applying log transformation
         df_train['SalePrice'] = np.log(df_train['SalePrice'])

In [79]: #transformed histogram and normal probability plot
         sns.distplot(df_train['SalePrice'], fit=norm);
         fig = plt.figure()
         res = stats.probplot(df_train['SalePrice'], plot=plt)
```

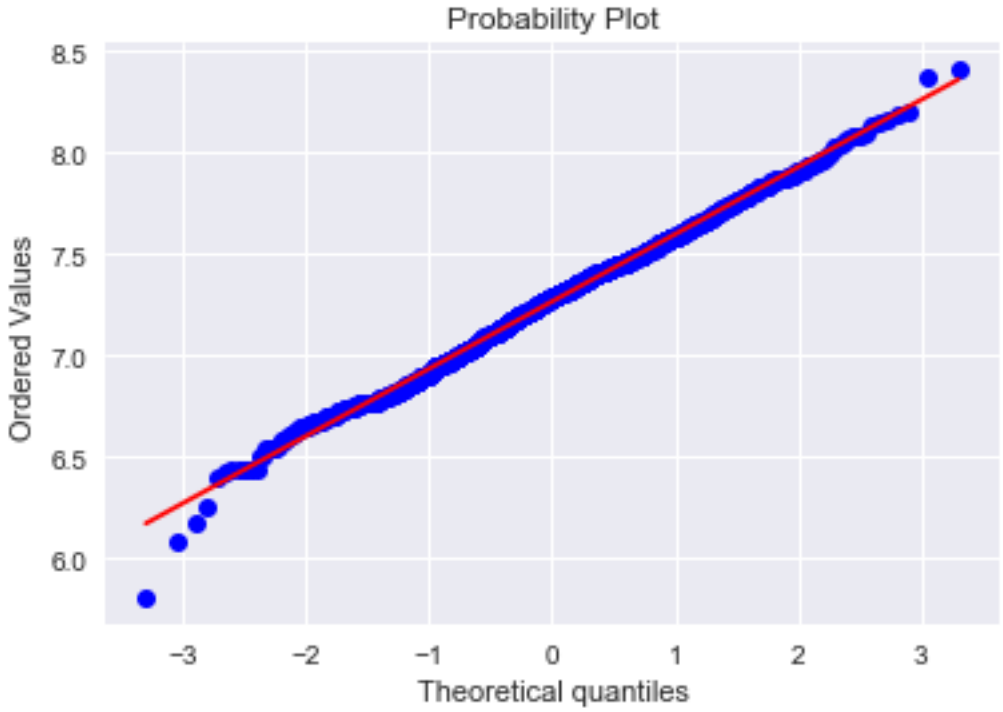




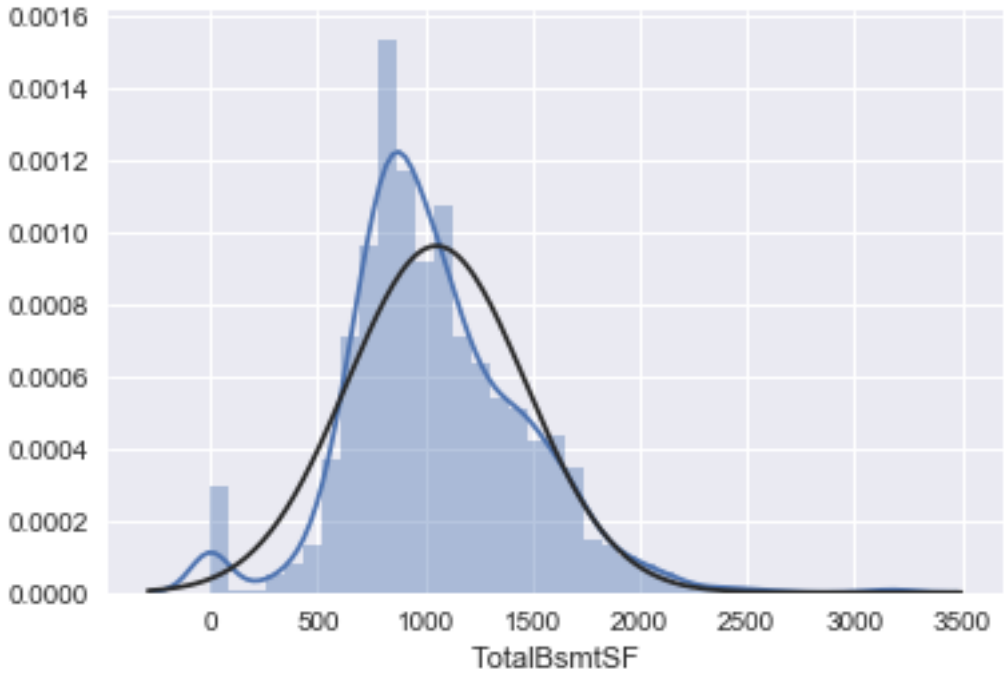

```
In [80]: #data transformation
         df_train['GrLivArea'] = np.log(df_train['GrLivArea'])
```

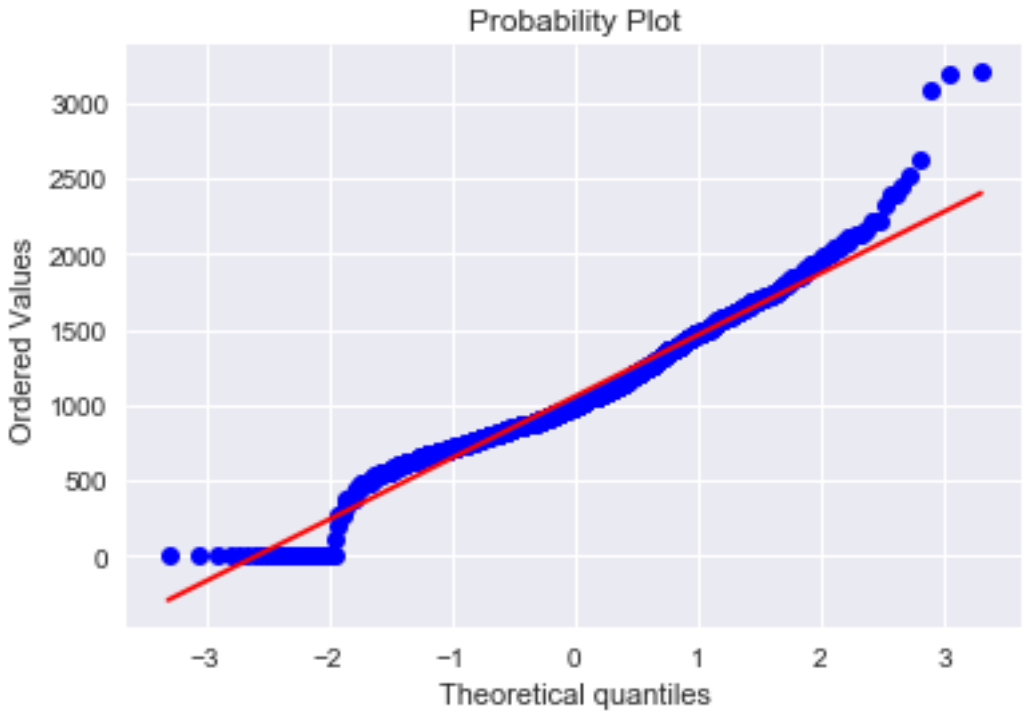
In [81]:
```
#transformed histogram and normal probability plot
sns.distplot(df_train['GrLivArea'], fit=norm);
fig = plt.figure()
res = stats.probplot(df_train['GrLivArea'], plot=plt)
```





In [82]:
```
#histogram and normal probability plot
sns.distplot(df_train['TotalBsmtSF'], fit=norm);
fig = plt.figure()
res = stats.probplot(df_train['TotalBsmtSF'], plot=plt)
```

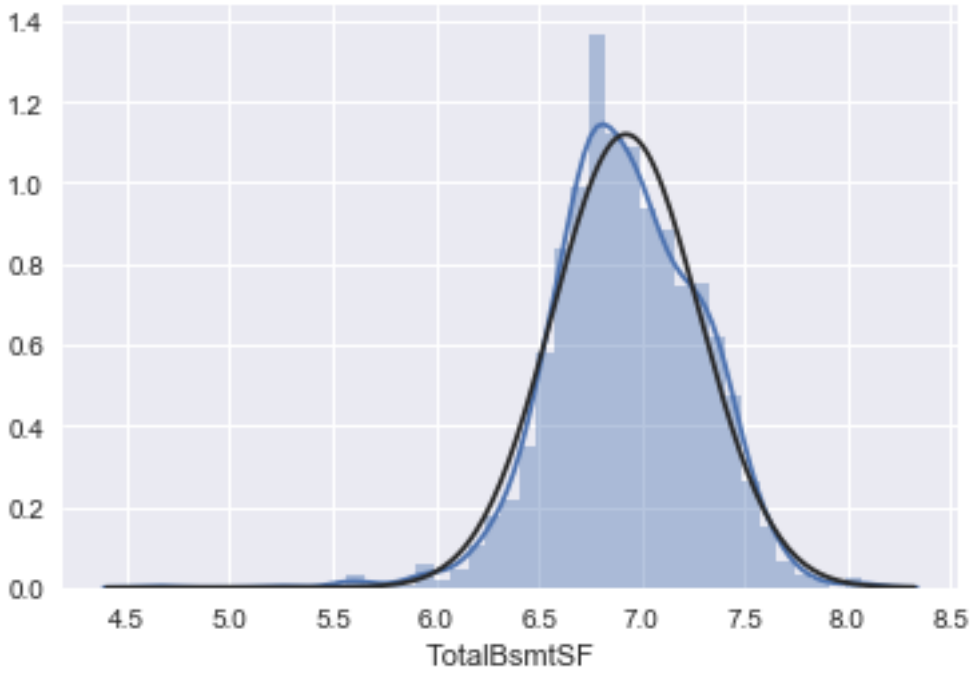


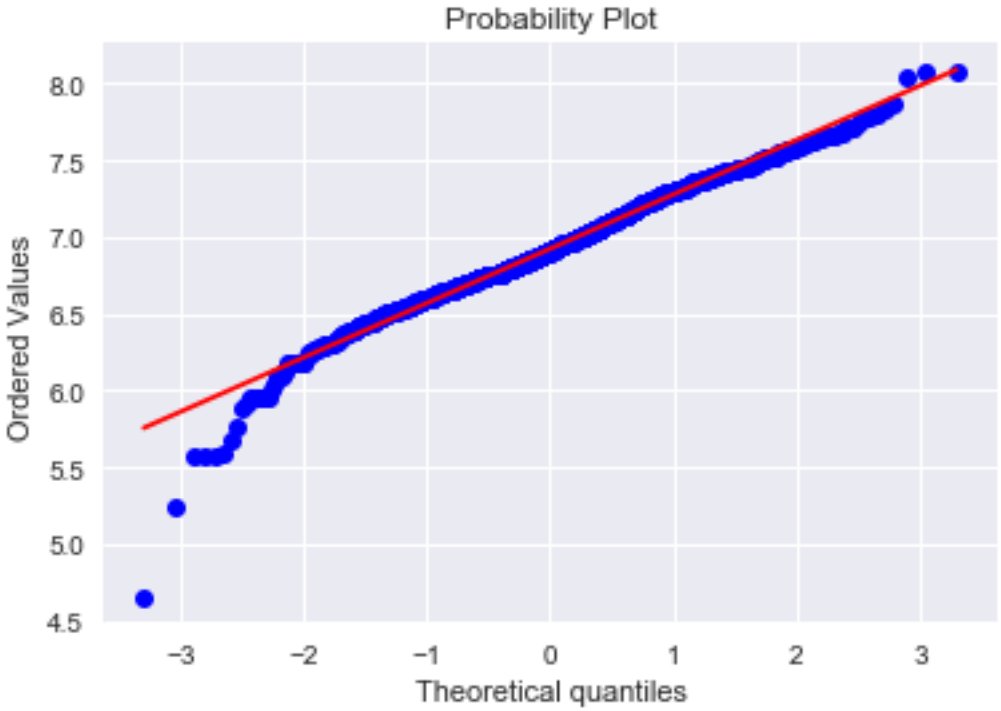


In [83]:
```
#create column for new variable (one is enough because it's a binary categorical feat
#if area>0 it gets 1, for area==0 it gets 0
df_train['HasBsmt'] = pd.Series(len(df_train['TotalBsmtSF']), index=df_train.index)
df_train['HasBsmt'] = 0
df_train.loc[df_train['TotalBsmtSF']>0,'HasBsmt'] = 1
```

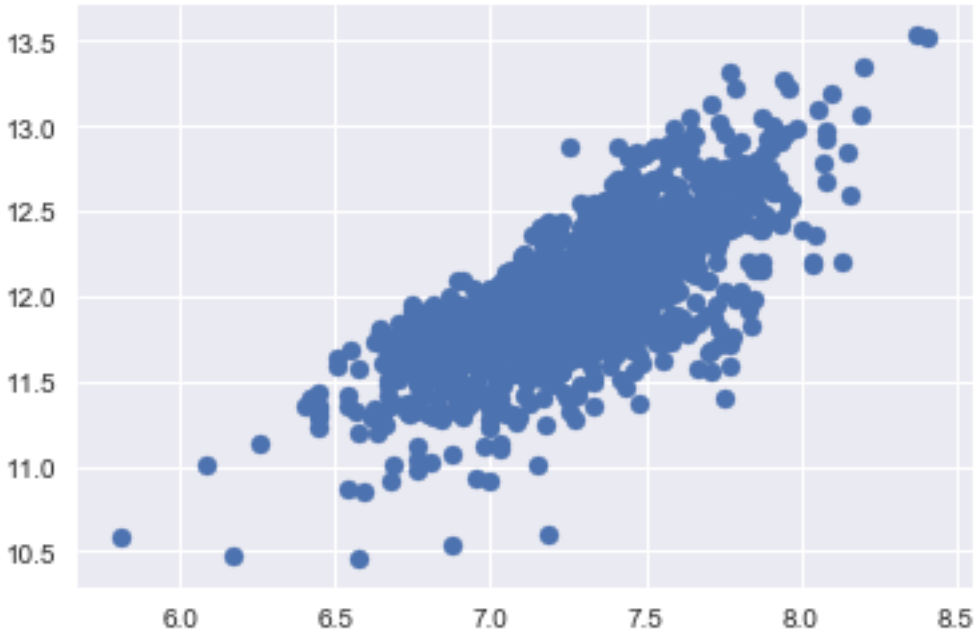
In [84]:
```
#transform data
df_train.loc[df_train['HasBsmt']==1,'TotalBsmtSF'] = np.log(df_train['TotalBsmtSF'])
```

In [85]:
```
#histogram and normal probability plot
sns.distplot(df_train[df_train['TotalBsmtSF']>0]['TotalBsmtSF'], fit=norm);
fig = plt.figure()
res = stats.probplot(df_train[df_train['TotalBsmtSF']>0]['TotalBsmtSF'], plot=plt)
```

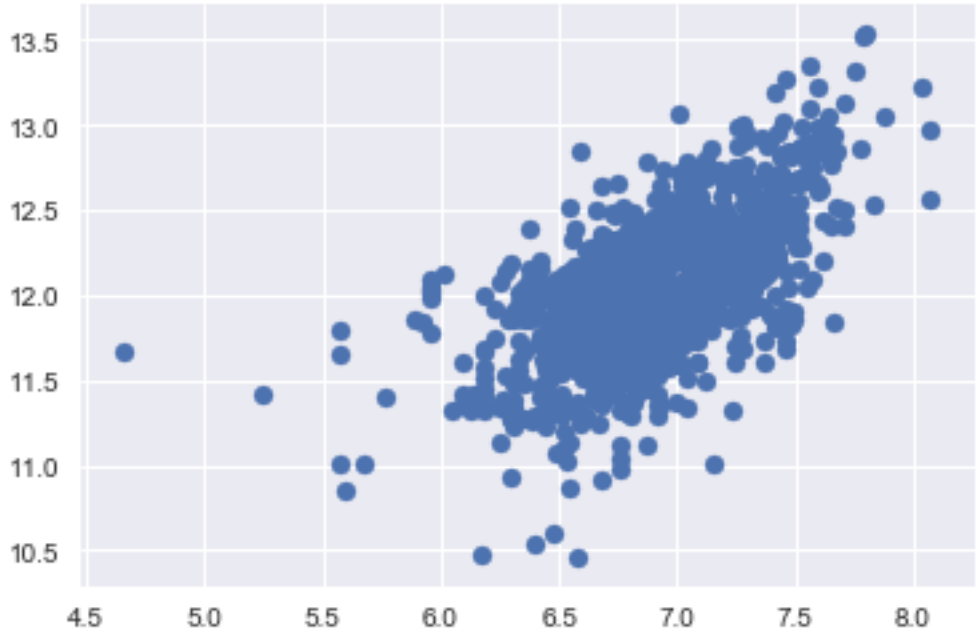




In [86]:
```
#scatter plot
plt.scatter(df_train['GrLivArea'], df_train['SalePrice']);
```



In [87]:
```
#scatter plot
plt.scatter(df_train[df_train['TotalBsmtSF']>0]['TotalBsmtSF'], df_train[df_train['To
```

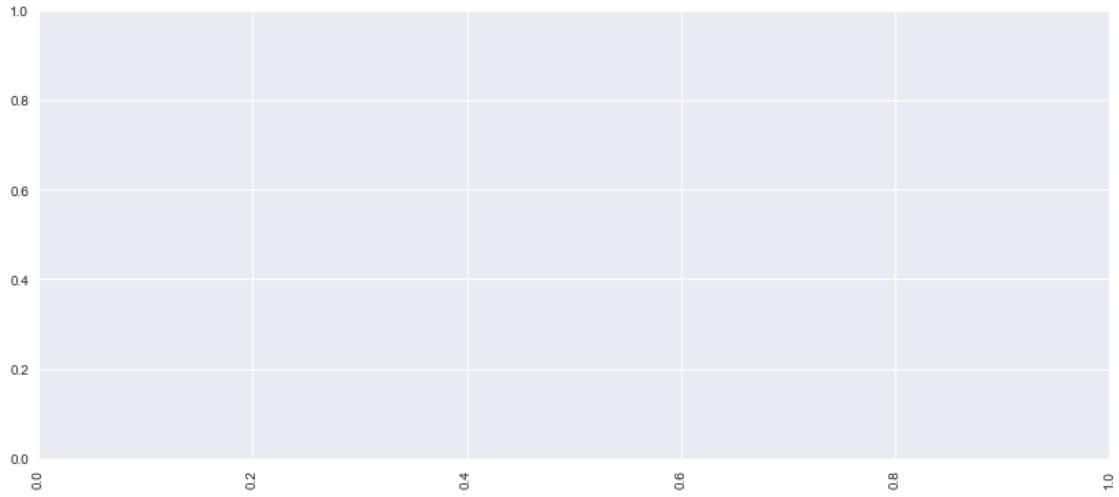


In [88]:
```
#convert categorical variable into dummy
df_train = pd.get_dummies(df_train)
```

```
In [89]: df_train.head()
```

```
Out[89]:    Id  MSSubClass  LotArea  OverallQual  OverallCond  YearBuilt  YearRemodAdd  \
0   1          60     8450            7            5       2003          2003
1   2          20     9600            6            8       1976          1976
2   3          60    11250            7            5       2001          2002
3   4          70     9550            7            5       1915          1970
4   5          60    14260            8            5       2000          2000

   BsmtFinSF1  BsmtFinSF2  BsmtUnfSF          ...            SaleType_ConLw  \
0         706           0        150          ...                         0
1         978           0        284          ...                         0
2         486           0        434          ...                         0
3         216           0        540          ...                         0
4         655           0        490          ...                         0

   SaleType_New  SaleType_Oth  SaleType_WD  SaleCondition_Abnorml  \
0             0             0            1                      0
1             0             0            1                      0
2             0             0            1                      0
3             0             0            1                      1
4             0             0            1                      0

   SaleCondition_AdjLand  SaleCondition_Alloca  SaleCondition_Family  \
0                      0                     0                     0
1                      0                     0                     0
2                      0                     0                     0
3                      0                     0                     0
4                      0                     0                     0

   SaleCondition_Normal  SaleCondition_Partial
0                     1                     0
1                     1                     0
2                     1                     0
3                     0                     0
4                     1                     0

[5 rows x 222 columns]
```

```
In [90]: test = pd.read_csv('test.csv')
         no_missing_col = [c for c in test.columns if test[c].isnull().sum() ==0]
         missing_col = [c for c in test.columns if test[c].isnull().sum() >0]

         missing = test[missing_col].isnull().sum()
         plt.figure(figsize=(14,6))
         plt.xticks(rotation=90);
```




```
In [91]: print('Number of rows and columns in train dataset:', df_train.shape)
         print('Number of rows and columns in test dataset:', test.shape)
```

```
Number of rows and columns in train dataset: (1457, 222)
Number of rows and columns in test dataset: (1459, 80)
```

```
In [92]: def Numeric_plot(df,column = '', title='',ncols=2,trans_func = None):
             """ Histogram plot Box plot of Numeric variable"""

             # Box plot
             trace1 = go.Box(y = df[column],name='Box')

             # Histogram
             trace2 = go.Histogram(x = df[column], name = 'x')

             fig = tools.make_subplots(rows=1, cols=ncols)
             fig.append_trace(trace1, 1,1)
             fig.append_trace(trace2, 1,2)
             fig['layout'].update(height=300, title=title)
             fig['layout']['yaxis1'].update(title= column)

             # Histogram after transformation
             if trans_func != None:
                 tmp = df[column].apply(trans_func)
                 trace3 = go.Histogram(x = tmp, name = trans_func+'(x)')
                 fig.append_trace(trace3, 1,3)

             py.iplot(fig)
```

```
In [99]: # Run this only once
         map_value = {20: '1-STORY 1946 & NEWER ALL STYLES',
```

```
                    30: '1-STORY 1945 & OLDER',
                    40: '1-STORY W/FINISHED ATTIC ALL AGES',
                    45: '1-1/2 STORY - UNFINISHED ALL AGES',
                    50: '1-1/2 STORY FINISHED ALL AGES',
                    60: '2-STORY 1946 & NEWER',
                    70: '2-STORY 1945 & OLDER',
                    75: '2-1/2 STORY ALL AGES',
                    80: 'PLIT OR MULTI-LEVEL',
                    85: 'SPLIT FOYER',
                    90: 'DUPLEX - ALL STYLES AND AGES',
                    120: '1-STORY PUD (Planned Unit Development) - 1946 & NEWER',
                    150: '1-1/2 STORY PUD - ALL AGES',
                    160: '2-STORY PUD - 1946 & NEWER',
                    180: 'PUD - MULTILEVEL - INCL SPLIT LEV/FOYER',
                    190: '2 FAMILY CONVERSION - ALL STYLES AND AGES'}

        df_train['MSSubClass'] = df_train['MSSubClass'].map(map_value)
        test['MSSubClass'] = test['MSSubClass'].map(map_value)

In [100]: def Regression_plot(df,column=''):
              """Regression plot: with pearsonr correlation value """
              cor = round(df[['SalePrice',column]].corr().iloc[0,1], 3)
              sns.jointplot(x= df[column], y = df['SalePrice'], kind= 'reg',
                            label = 'r: '+str(cor),color='blue')
              plt.legend()
              #plt.title('Regression plot ')

In [102]: from sklearn.preprocessing import scale
          from sklearn.model_selection import train_test_split
          from sklearn.linear_model import Ridge, RidgeCV, Lasso, LassoCV
          from sklearn.metrics import mean_squared_error


          ridge = Ridge(normalize = True)
          coefs = []
```