In [69]:
```
import numpy as np
import pandas as pd

data = pd.read_csv("C:\\Users\\virin\\OneDrive\\Documents\\train.csv")
data.head(5)
```

Out[69]:

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

5 rows × 81 columns

In [53]:
```
data.info()
test1 = data.loc[:,['MSSubClass','LotArea','OverallQual','OverallCond','YearBuilt
'BsmtFinSF2','BsmtUnfSF','TotalBsmtSF']]
test1.head(5)
```

```
<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
```

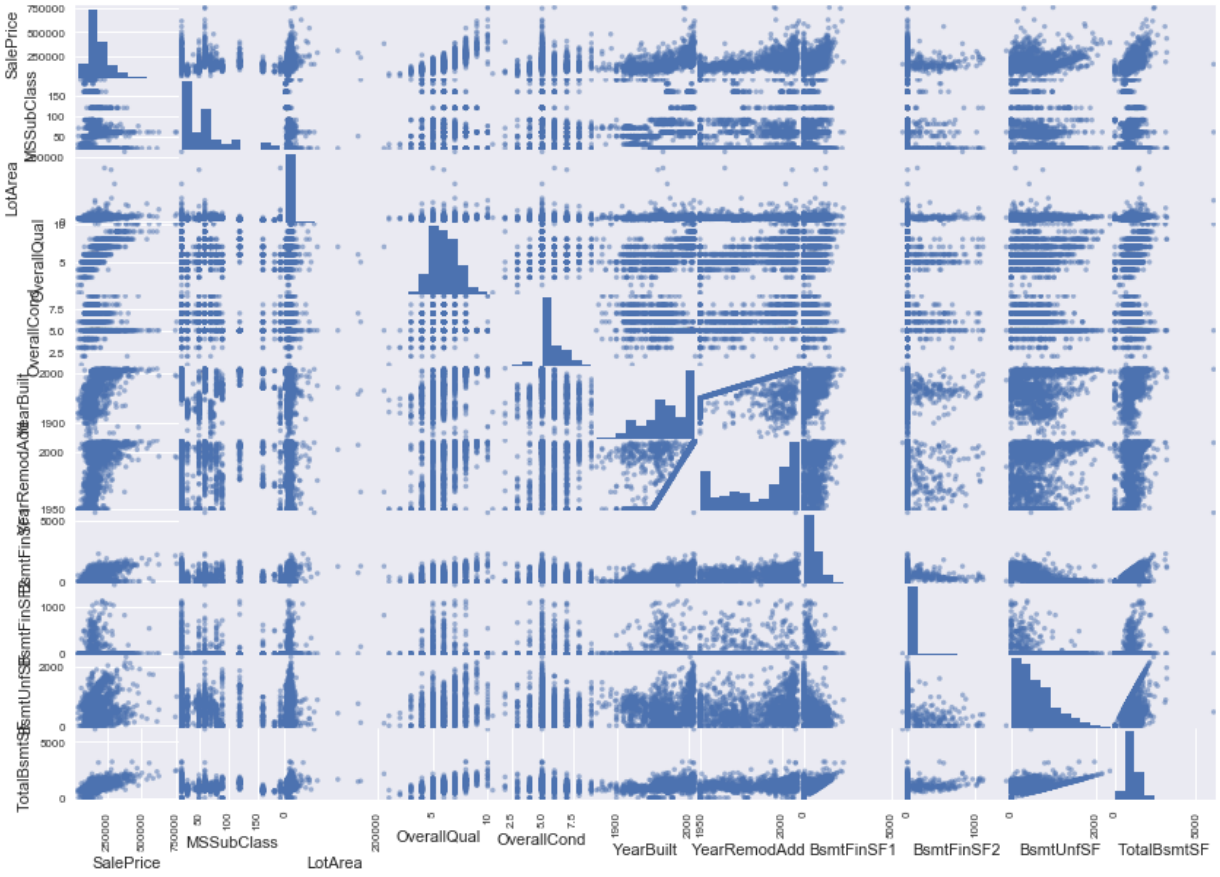
```
FullBath         1460 non-null int64
HalfBath         1460 non-null int64
BedroomAbvGr     1460 non-null int64
KitchenAbvGr     1460 non-null int64
KitchenQual      1460 non-null object
TotRmsAbvGrd     1460 non-null int64
Functional       1460 non-null object
Fireplaces       1460 non-null int64
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
```

Out[53]:

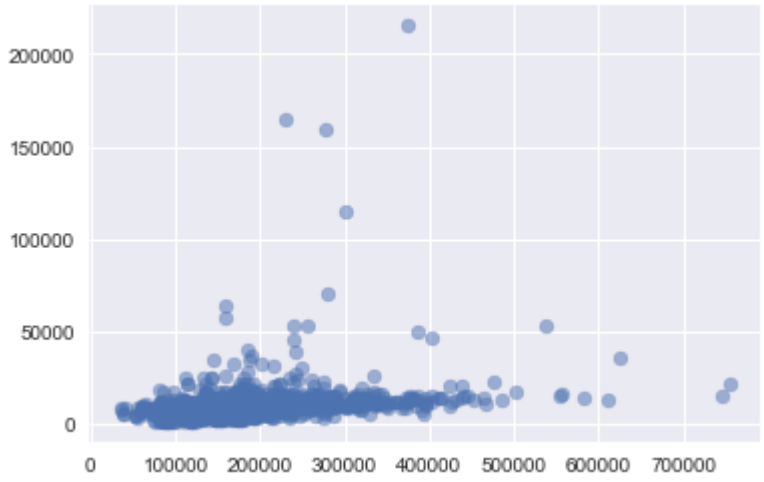
	MSSubClass	LotArea	OverallQual	OverallCond	YearBuilt	YearRemodAdd	BsmtFinSF1	BsmtF
0	60	8450	7	5	2003	2003	706	
1	20	9600	6	8	1976	1976	978	
2	60	11250	7	5	2001	2002	486	
3	70	9550	7	5	1915	1970	216	
4	60	14260	8	5	2000	2000	655	

In [54]:
```
from pandas.tools.plotting import scatter_matrix

attributes = ['SalePrice','MSSubClass','LotArea','OverallQual','OverallCond','Yea
'BsmtFinSF2','BsmtUnfSF','TotalBsmtSF']
scatter_matrix(data[attributes], figsize = (14,10))
plt.show()
```

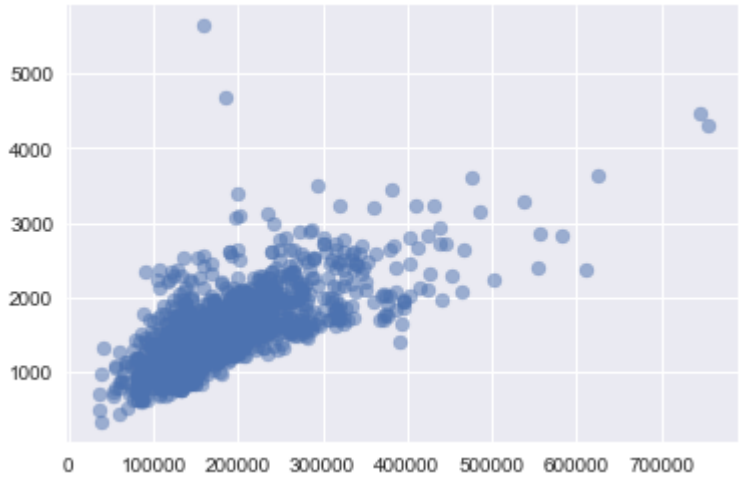
```
C:\Users\virin\Anaconda3\lib\site-packages\ipykernel_launcher.py:5: FutureWarni
ng: 'pandas.tools.plotting.scatter_matrix' is deprecated, import 'pandas.plotti
ng.scatter_matrix' instead.
  """
```



In [55]:
```
plt.scatter(x = data['SalePrice'], y = data['LotArea'], alpha = 0.5)
plt.show()
```



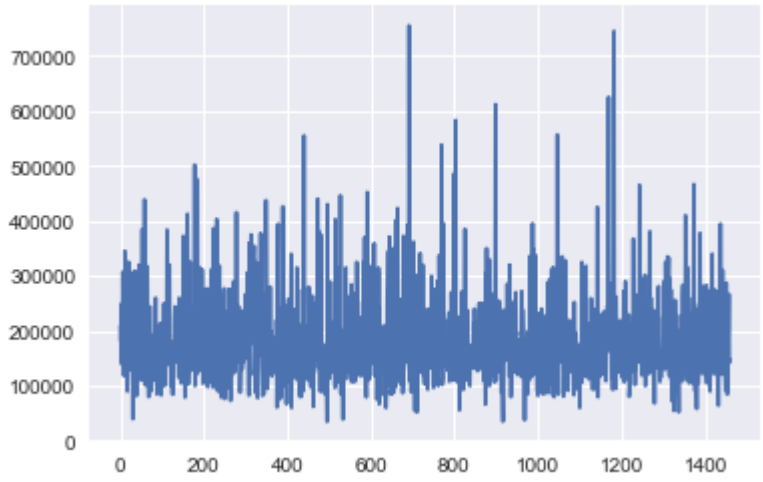
In [56]:
```
plt.scatter(x = data['SalePrice'], y = data['GrLivArea'], alpha = 0.5)
plt.show()
```



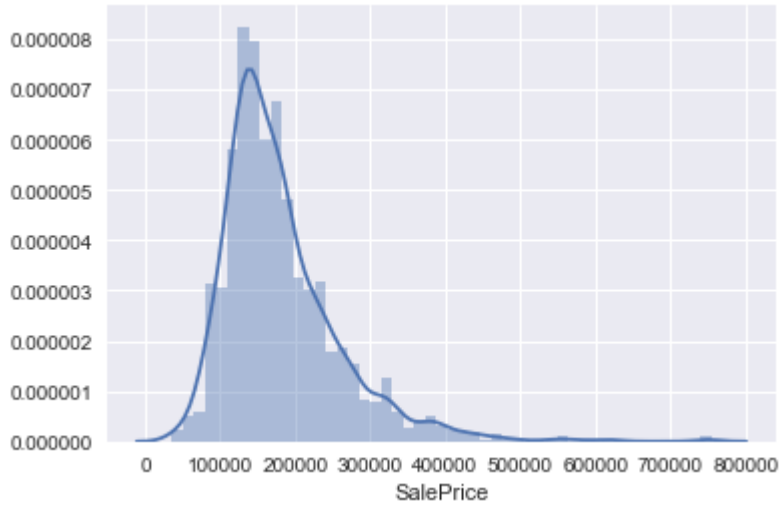
In [57]:
```
data['SalePrice'].describe()
```

Out[57]:
```
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
```

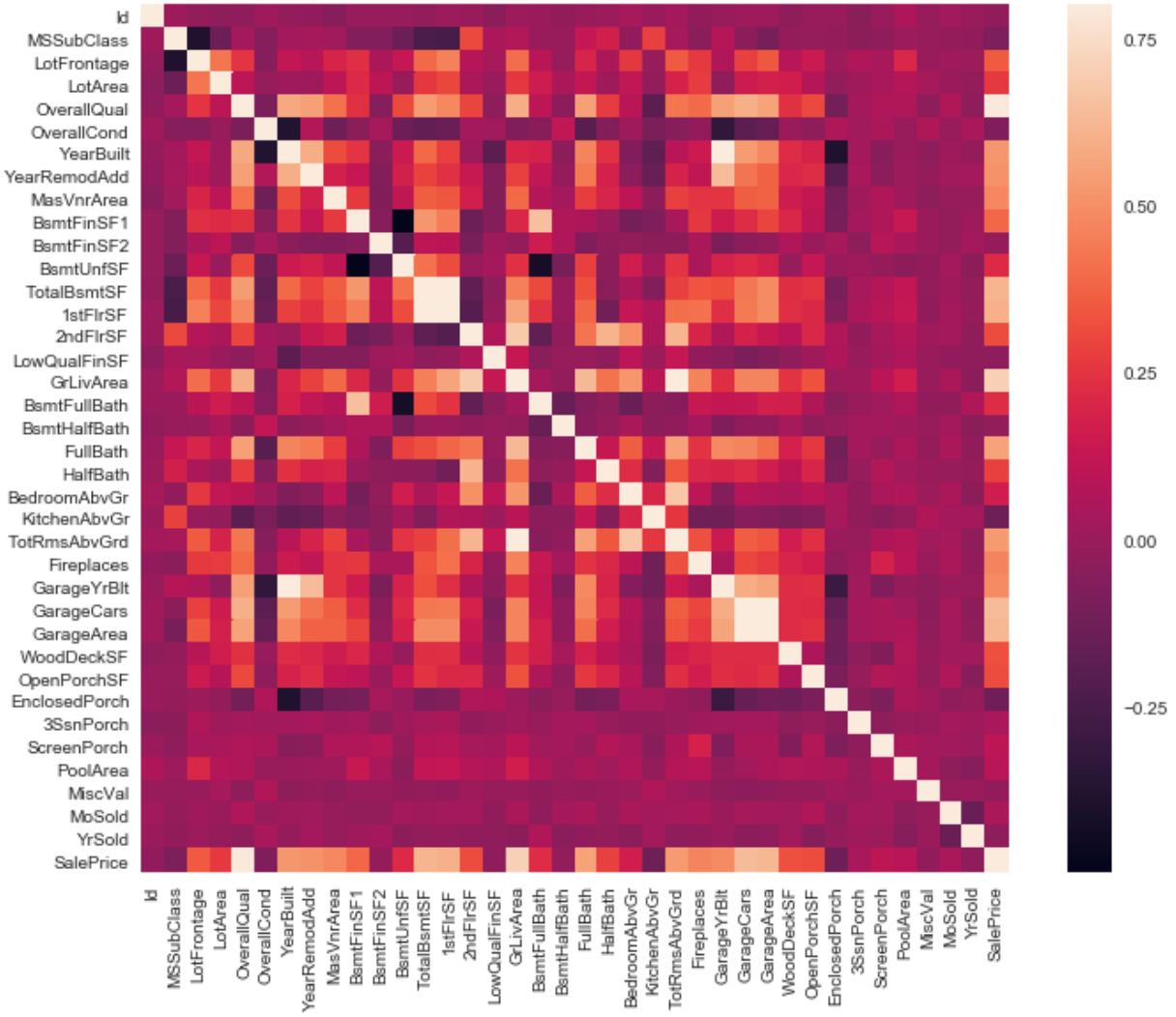
In [58]:
```
import matplotlib.pyplot as plt
plt.plot(data['SalePrice'])
plt.show()
```



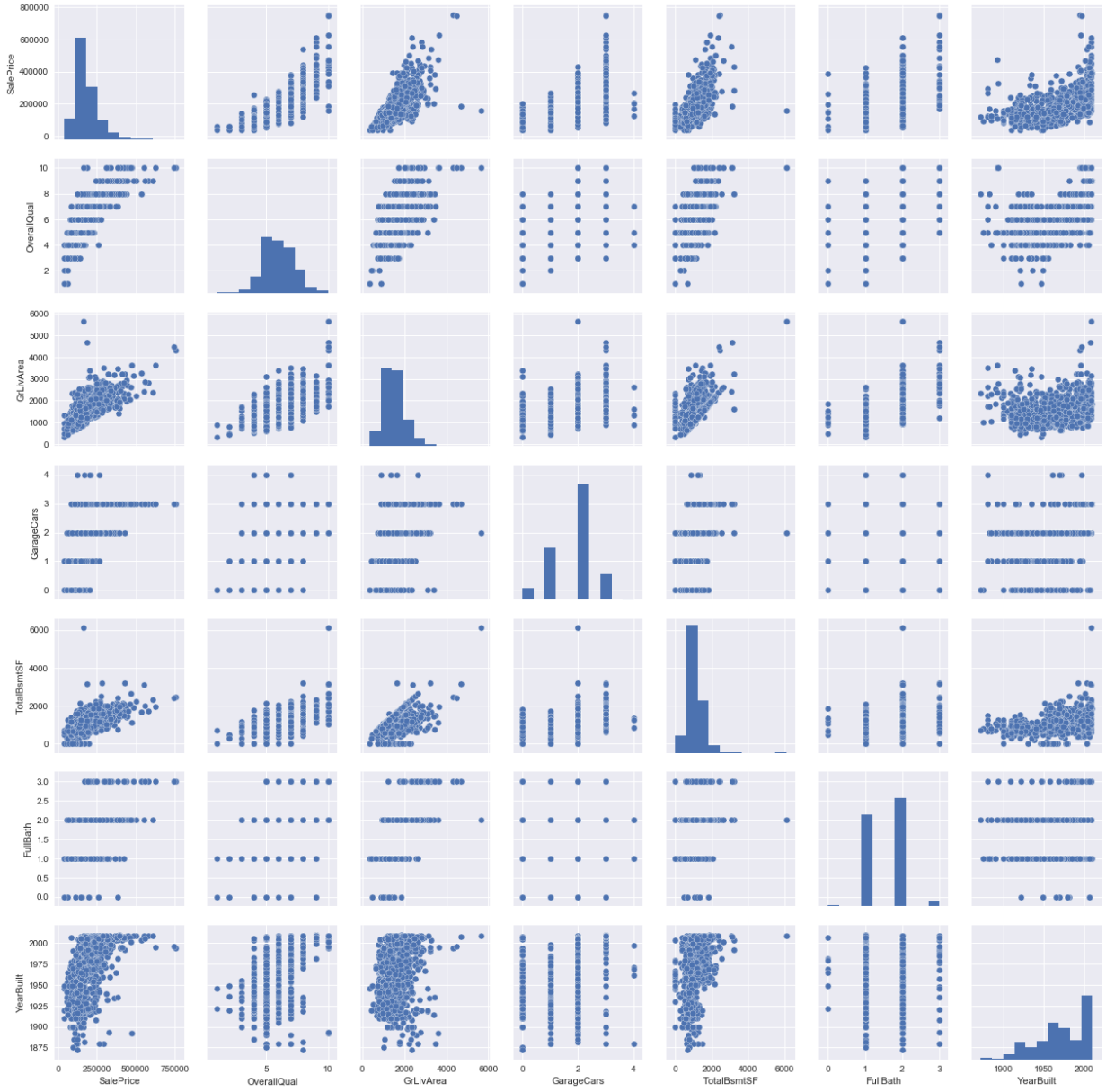
In [59]:
```
import seaborn as sns
#Right skewed data
#mean<median
sns.distplot(data['SalePrice']);
plt.show()
```



In [60]:
```
#correlation matrix
corrmat = data.corr()
f, ax = plt.subplots(figsize=(12, 9))
sns.heatmap(corrmat, vmax=.8, square=True);
plt.show()
```

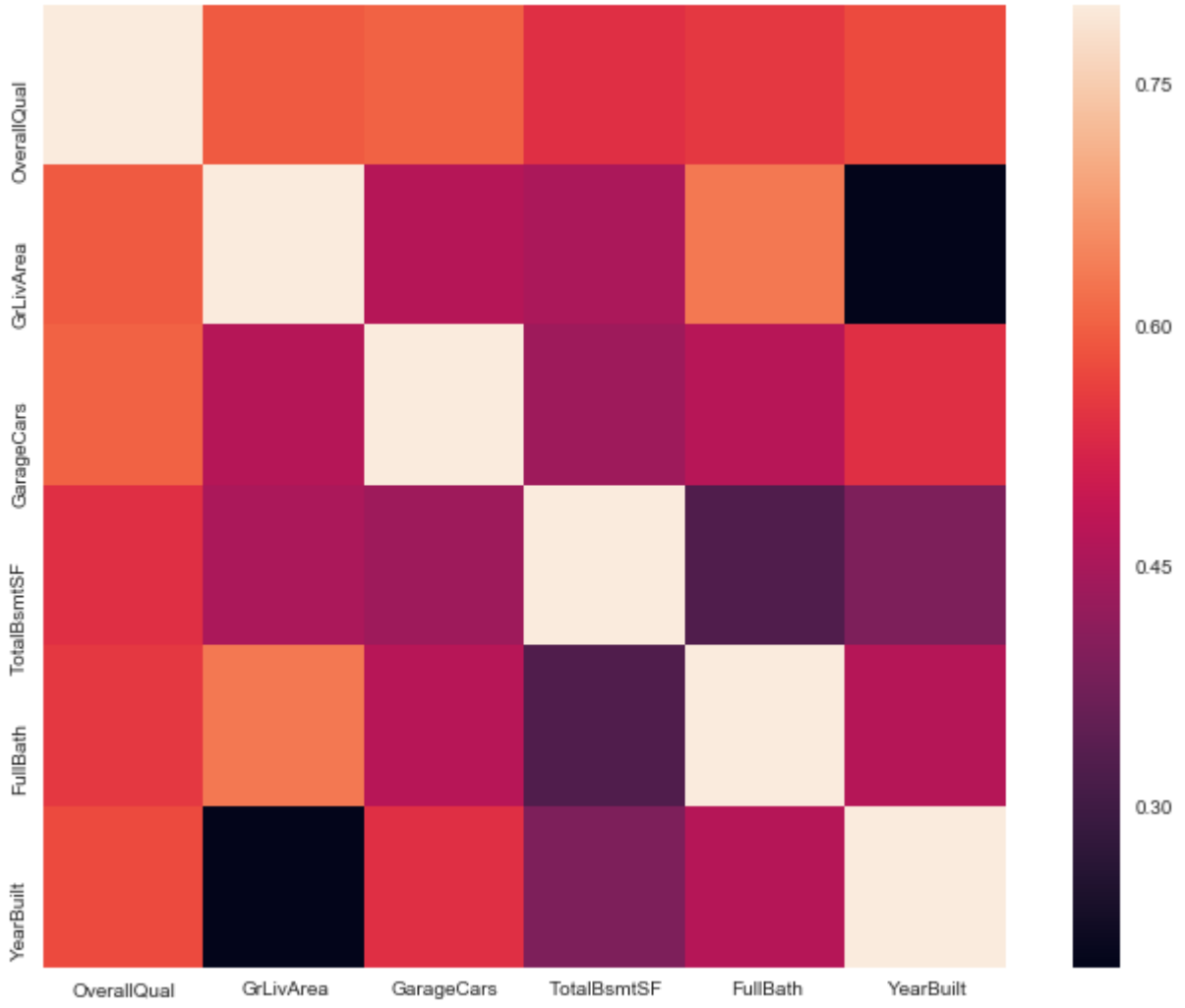


In [61]:
```
#the highly correlated variables are selected
#scatterplot
sns.set()
cols = ['SalePrice', 'OverallQual', 'GrLivArea', 'GarageCars', 'TotalBsmtSF', 'Fu
sns.pairplot(data[cols], size = 2.5)
plt.show();
```



In [62]:
```
#checking inter correlation
sns.set()
data_inter = data.loc[:,['OverallQual', 'GrLivArea', 'GarageCars', 'TotalBsmtSF',
sns.pairplot(data_inter, size = 2.5)
plt.show();
```

In [63]:
```
#no inter correlation between the variables is observed
corrmat = data_inter.corr()
f, ax = plt.subplots(figsize=(12, 9))
sns.heatmap(corrmat, vmax=.8, square=True);
plt.show()
```



In [70]:
```
#missing data
total = data.isnull().sum().sort_values(ascending=False)
percent = ((data.isnull().sum()/data.isnull().count())*100).sort_values(ascending=
missing_data = pd.concat([total, percent], axis=1, keys=['Total', 'Percent'])
missing_data.head(20)
```

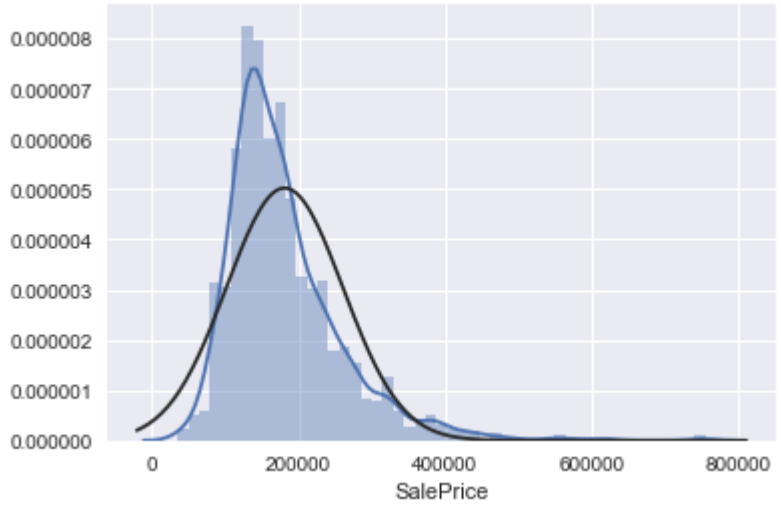
Out[70]:

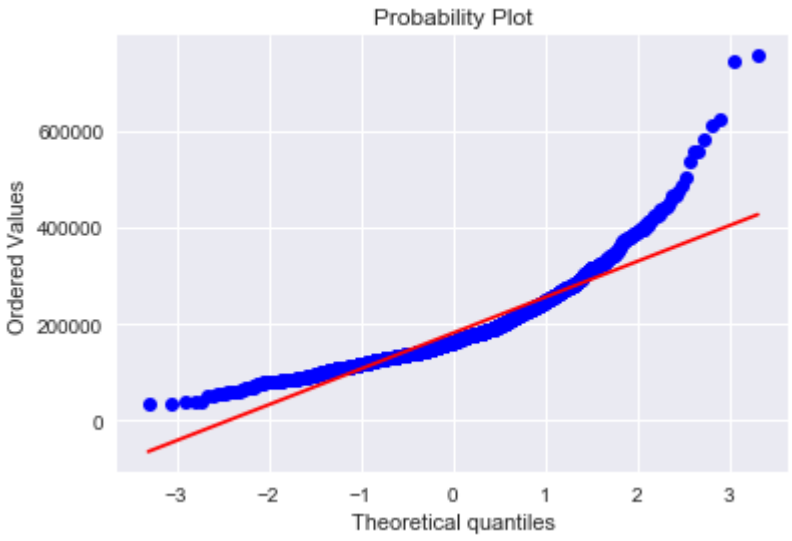
	Total	Percent
PoolQC	1453	99.520548
MiscFeature	1406	96.301370
Alley	1369	93.767123
Fence	1179	80.753425
FireplaceQu	690	47.260274
LotFrontage	259	17.739726
GarageCond	81	5.547945
GarageType	81	5.547945
GarageYrBlt	81	5.547945
GarageFinish	81	5.547945
GarageQual	81	5.547945
BsmtExposure	38	2.602740
BsmtFinType2	38	2.602740
BsmtFinType1	37	2.534247
BsmtCond	37	2.534247
BsmtQual	37	2.534247
MasVnrArea	8	0.547945
MasVnrType	8	0.547945
Electrical	1	0.068493
Utilities	0	0.000000

In [71]:
```
#dealing with missing data
data = data.drop((missing_data[missing_data['Total'] > 1]).index,1)
data = data.drop(data.loc[data['Electrical'].isnull()].index)
data.isnull().sum().max() #just checking that there's no missing data missing...
```

Out[71]: 0

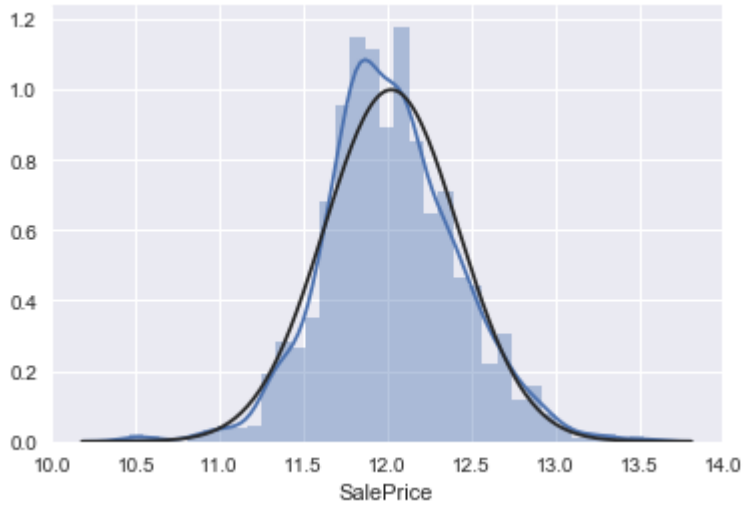
In [77]:
```
#histogram and normal probability plot
from scipy.stats import norm
from sklearn.preprocessing import StandardScaler
from scipy import stats
sns.distplot(data['SalePrice'], fit = norm);
fig = plt.figure()
res = stats.probplot(data['SalePrice'], plot=plt)
plt.show()
```

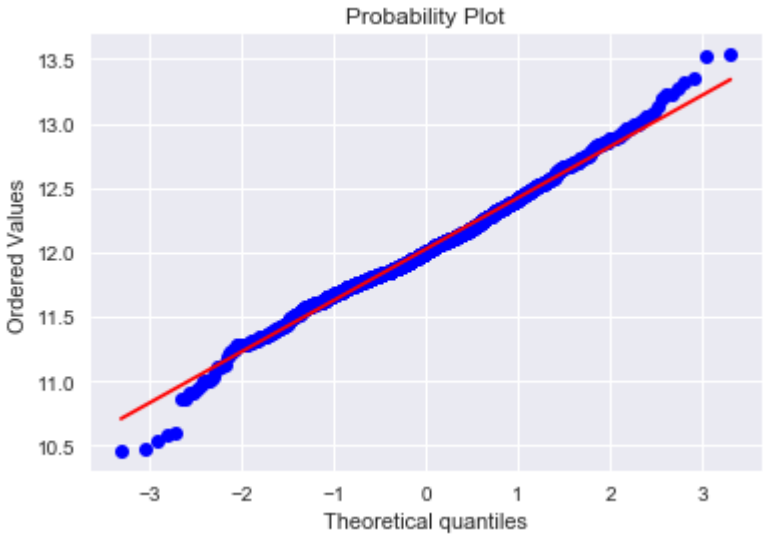




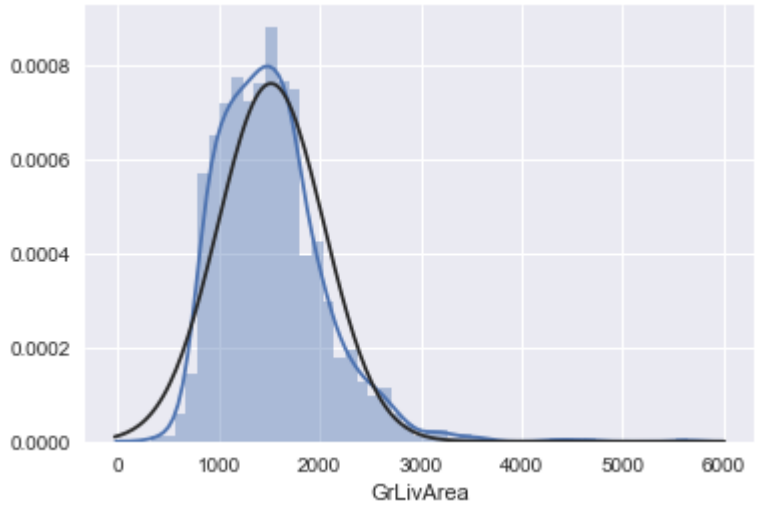
In [83]:
```
#applying log transformation
data['SalePrice'] = np.log(data['SalePrice'])
```

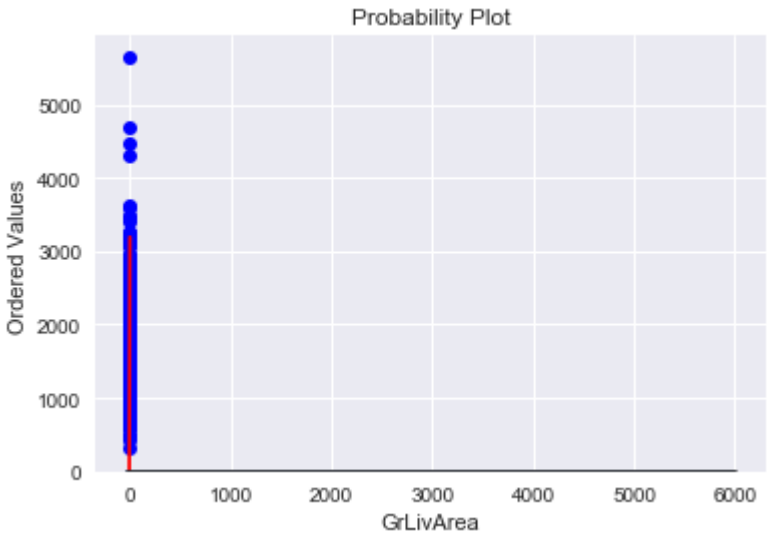
In [84]:
```
sns.distplot(data['SalePrice'], fit = norm);
fig = plt.figure()
res = stats.probplot(data['SalePrice'], plot=plt)
plt.show()
```

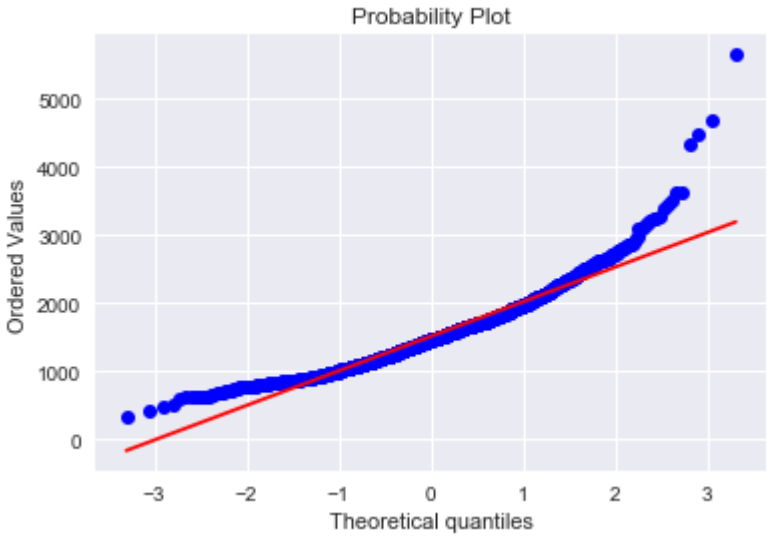




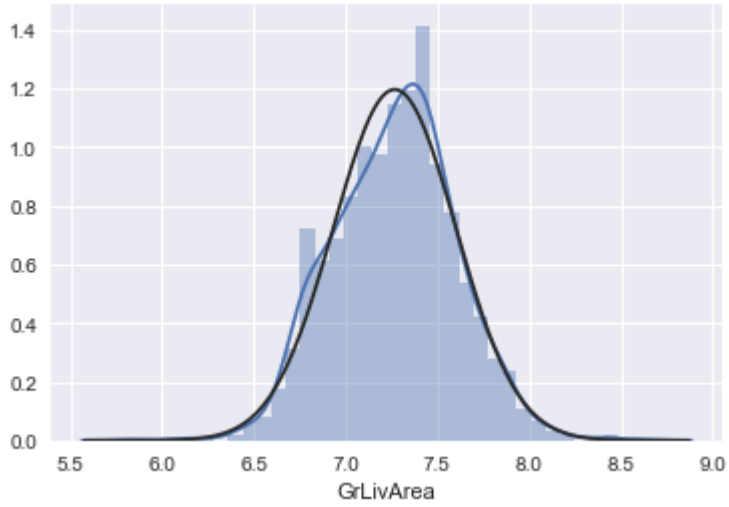
In [87]:
```
#histogram and normal probability plot
sns.distplot(data['GrLivArea'], fit=norm);
fig = plt.figure()
res = stats.probplot(data['GrLivArea'], plot=plt)
plt.show()
```

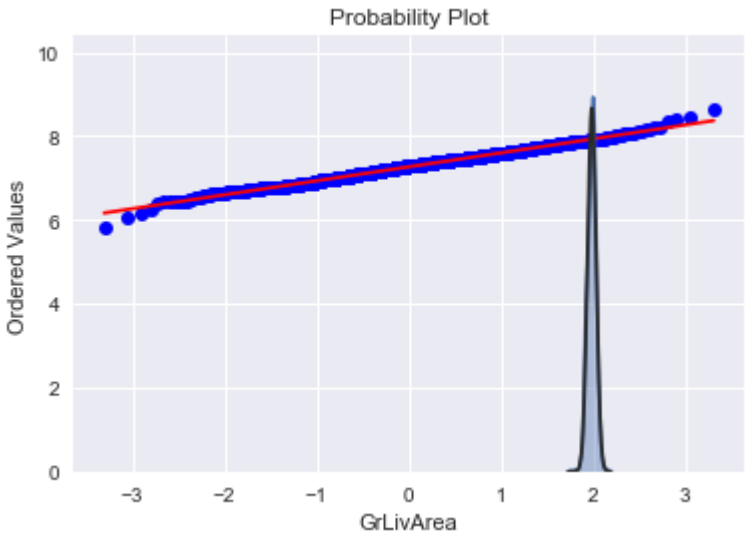


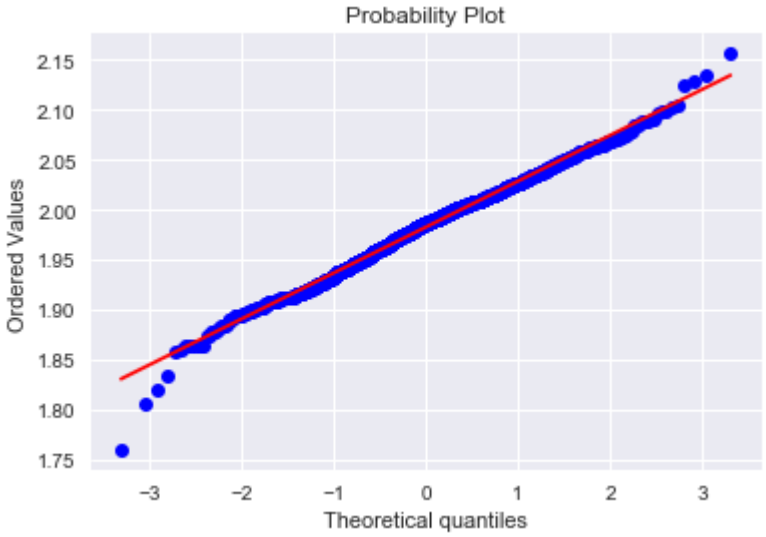




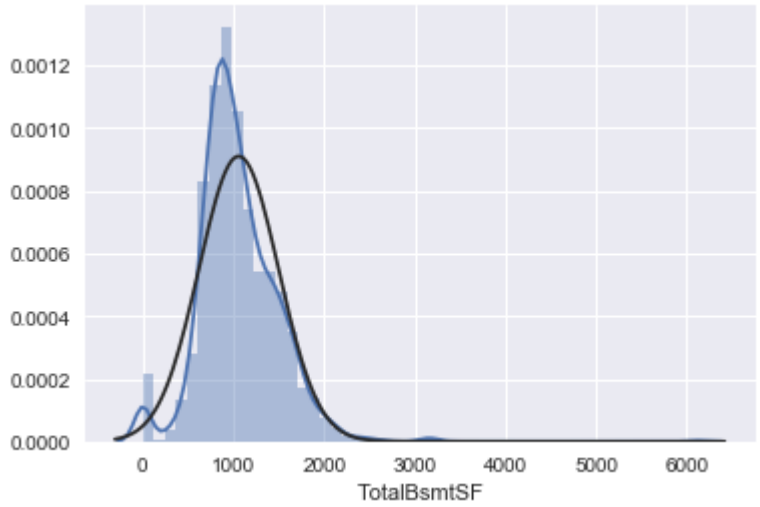
In [89]:
```
#data transformation
data['GrLivArea'] = np.log(data['GrLivArea'])

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

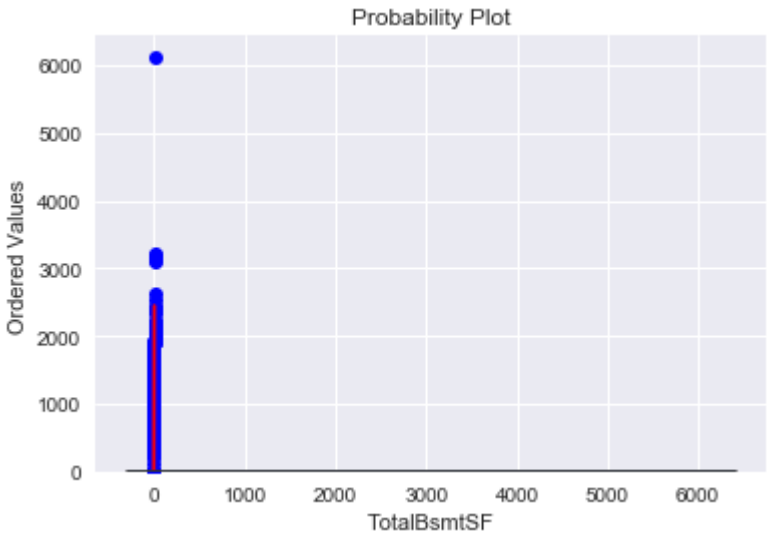


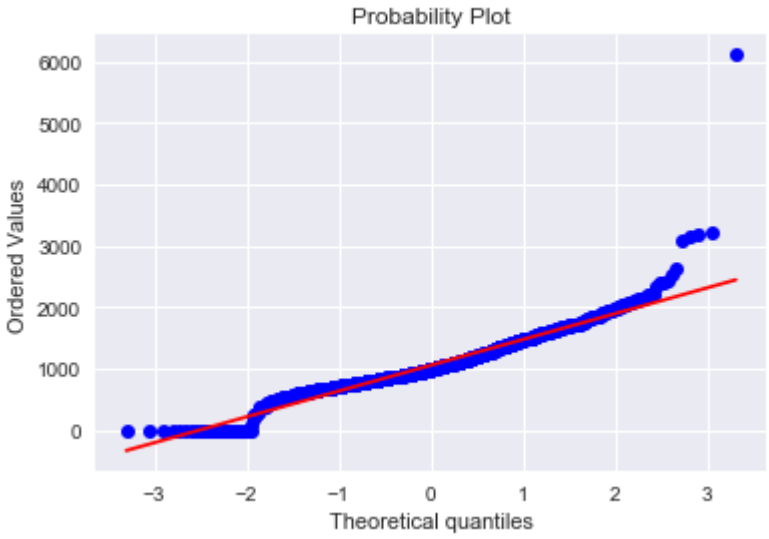



Probability Plot
Ordered Values
Theoretical quantiles
2.15
2.10
2.05
2.00
1.95
1.90
1.85
1.80
1.75
−3
−2
−1
0
1
2
3

In [91]:
```
#histogram and normal probability plot
sns.distplot(data['TotalBsmtSF'], fit=norm);
fig = plt.figure()
res = stats.probplot(data['TotalBsmtSF'], plot=plt)
plt.show()
```







In [82]:
```
data['HasBsmt'] = pd.Series(len(data['TotalBsmtSF']), index=df_train.index)
data['HasBsmt'] = 0
data.loc[data['TotalBsmtSF']>0,'HasBsmt'] = 1
```

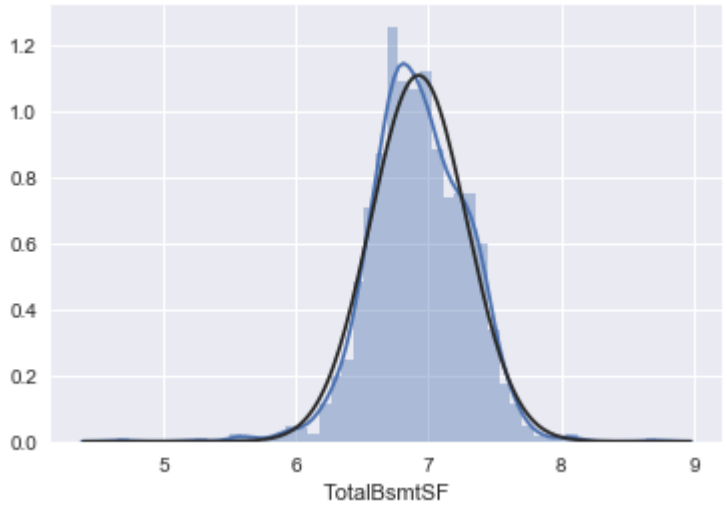
Out[82]:
```
0       1
1       1
2       1
3       1
4       1
5       1
6       1
7       1
8       1
9       1
10      1
11      1
12      1
13      1
14      1
15      1
16      1
17      0
18      1
19      1
20      1
21      1
22      1
23      1
24      1
25      1
26      1
27      1
28      1
29      1
       ..
1430    1
1431    1
1432    1
1433    1
1434    1
1435    1
1436    1
1437    1
1438    1
1439    1
1440    1
1441    1
1442    1
1443    1
1444    1
1445    1
1446    1
1447    1
1448    1
1449    1
1450    1
```

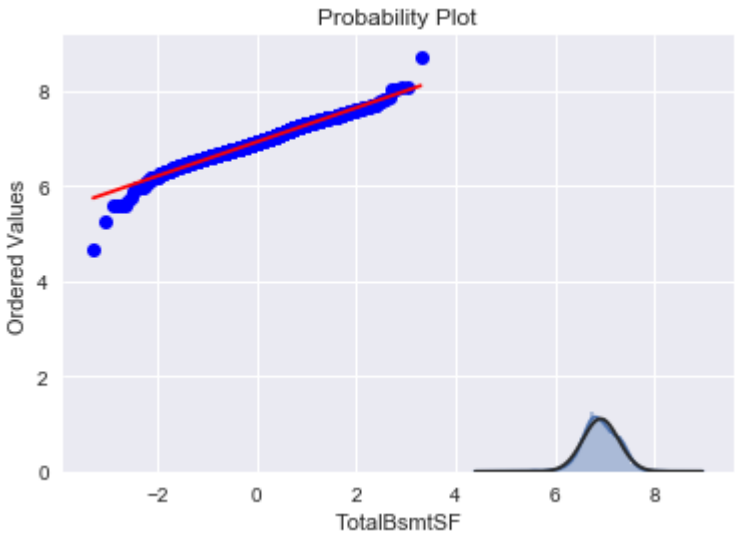
```
1451    1
1452    1
1453    1
1454    1
1455    1
1456    1
1457    1
1458    1
1459    1
Name: HasBsmt, Length: 1459, dtype: int64
```

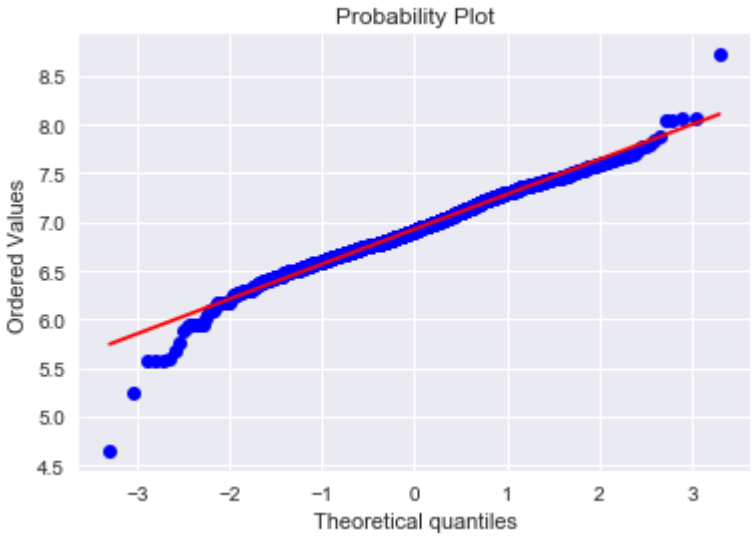
In [92]:
```
#transform data
data.loc[data['HasBsmt']==1,'TotalBsmtSF'] = np.log(data['TotalBsmtSF'])
```

```
C:\Users\virin\Anaconda3\lib\site-packages\ipykernel_launcher.py:2: RuntimeWarn
ing: divide by zero encountered in log
```

In [94]:
```
#histogram and normal probability plot
sns.distplot(data[data['TotalBsmtSF']>0]['TotalBsmtSF'], fit=norm);
fig = plt.figure()
res = stats.probplot(data[data['TotalBsmtSF']>0]['TotalBsmtSF'], plot=plt)
plt.show()
```







In [95]:
```
#convert categorical variable into dummy
data = pd.get_dummies(data)
data.head(10)
```

Out[95]:

	Id	MSSubClass	LotArea	OverallQual	OverallCond	YearBuilt	YearRemodAdd	BsmtFinSF1	Bs
0	1	60	8450	7	5	2003	2003	706	
1	2	20	9600	6	8	1976	1976	978	
2	3	60	11250	7	5	2001	2002	486	
3	4	70	9550	7	5	1915	1970	216	
4	5	60	14260	8	5	2000	2000	655	
5	6	50	14115	5	5	1993	1995	732	
6	7	20	10084	8	5	2004	2005	1369	
7	8	60	10382	7	6	1973	1973	859	
8	9	50	6120	7	5	1931	1950	0	
9	10	190	7420	5	6	1939	1950	851	

10 rows × 223 columns

In [116]:
```
import numpy as np
from sklearn import datasets
from sklearn.linear_model import Ridge
from sklearn.model_selection import GridSearchCV
clf = RandomForestRegressor(random_state =42, max_features = 'auto')
param_grid = {'max_depth' : [10,20,30,40,50],
              'n_estimators' :[10,100,200,300]}
validator = GridSearchCV(clf, param_grid= param_grid)
validator.fit(x_train,y_train)
for i in max_depth:
    print(validator.best_score_)

print(validator.best_estimator_.n_estimators)
print(validator.best_estimator_.max_depth)
print(validator.best_estimator_.max_features)
```

```
0.829000944687
0.829000944687
300
10
auto
```

In [125]:
```
from sklearn import model_selection
for cv in np.arange(2, 12, 2):
    GS = model_selection.GridSearchCV(
        cv=cv, estimator=RandomForestRegressor(random_state=42),
        param_grid={'max_depth' : [10,20,30,40,50],
                    'n_estimators' :[10,100,200,300]}
        )
    GS.fit(x_train, y_train)
    print(cv, GS.best_score_,GS.best_estimator_.max_depth,GS.best_estimator_.n_es
```

```
2 0.83425233276 10 RandomForestRegressor(bootstrap=True, criterion='mse', max_d
epth=10,
           max_features='auto', max_leaf_nodes=None,
           min_impurity_decrease=0.0, min_impurity_split=None,
           min_samples_leaf=1, min_samples_split=2,
           min_weight_fraction_leaf=0.0, n_estimators=300, n_jobs=1,
           oob_score=False, random_state=42, verbose=0, warm_start=False)
4 0.831039483983 10 RandomForestRegressor(bootstrap=True, criterion='mse', max_
depth=10,
           max_features='auto', max_leaf_nodes=None,
           min_impurity_decrease=0.0, min_impurity_split=None,
           min_samples_leaf=1, min_samples_split=2,
           min_weight_fraction_leaf=0.0, n_estimators=300, n_jobs=1,
           oob_score=False, random_state=42, verbose=0, warm_start=False)
6 0.831593268234 10 RandomForestRegressor(bootstrap=True, criterion='mse', max_
depth=10,
           max_features='auto', max_leaf_nodes=None,
           min_impurity_decrease=0.0, min_impurity_split=None,
           min_samples_leaf=1, min_samples_split=2,
           min_weight_fraction_leaf=0.0, n_estimators=300, n_jobs=1,
           oob_score=False, random_state=42, verbose=0, warm_start=False)
8 0.831951522831 10 RandomForestRegressor(bootstrap=True, criterion='mse', max_
depth=10,
           max_features='auto', max_leaf_nodes=None,
           min_impurity_decrease=0.0, min_impurity_split=None,
           min_samples_leaf=1, min_samples_split=2,
           min_weight_fraction_leaf=0.0, n_estimators=300, n_jobs=1,
           oob_score=False, random_state=42, verbose=0, warm_start=False)
10 0.829060593681 10 RandomForestRegressor(bootstrap=True, criterion='mse', max
_depth=10,
           max_features='auto', max_leaf_nodes=None,
           min_impurity_decrease=0.0, min_impurity_split=None,
           min_samples_leaf=1, min_samples_split=2,
           min_weight_fraction_leaf=0.0, n_estimators=100, n_jobs=1,
           oob_score=False, random_state=42, verbose=0, warm_start=False)
```

In [132]:
```
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split
from sklearn import preprocessing
from sklearn.metrics import mean_squared_error

y_train = (data["SalePrice"])
x_train = data.loc[:,['OverallQual', 'GrLivArea', 'GarageCars', 'TotalBsmtSF', 'F
x = x_train.values #returns a numpy array
min_max_scaler = preprocessing.MinMaxScaler()
x_scaled = min_max_scaler.fit_transform(x)
x_train = pd.DataFrame(x_scaled)

tree_reg = RandomForestRegressor(max_depth = 30, random_state=42, n_estimators =
tree_reg.fit(x_train,y_train)
tree_pred = tree_reg.predict(x_train)
tree_pred = pd.DataFrame(tree_pred)
mat = pd.concat([tree_pred,y_train], axis =1)
print(mat)
dec_mse = mean_squared_error(tree_pred, y_train)
rmse = np.sqrt(dec_mse)
rmse
```

```
              0  SalePrice
0     12.181467  12.247694
1     12.075451  12.109011
2     12.281178  12.317167
3     11.940430  11.849398
4     12.484539  12.429216
5     11.868170  11.870600
6     12.562950  12.634603
7     12.263832  12.206073
8     11.932748  11.774520
9     11.648572  11.678440
10    11.783054  11.771436
11    12.742856  12.751300
12    11.800299  11.877569
13    12.457670  12.540758
14    11.954887  11.964001
15    11.853259  11.790557
16    11.921819  11.911702
17    11.470390  11.407565
18    11.965298  11.976659
19    11.857918  11.842229
20    12.674504  12.692503
21    11.823628  11.845103
22    12.361737  12.345835
23    11.824561  11.774520
24    11.906183  11.944708
25    12.450367  12.454104
26    11.803750  11.811547
27    12.650978  12.631340
28    12.105803  12.242887
29    11.160344  11.134589
...         ...        ...
1430  11.866198  12.165980
1431  11.171037  11.875831
1432  12.174334  11.074421
```

```
1433  12.010596  12.136187
1434  12.069521  11.982929
1435  11.612483  12.066811
1436  12.779190  11.699405
1437  11.898818  12.885671
1438  12.132275  11.916389
1439  12.135791  12.190959
1440  11.898458  12.160029
1441  12.672757  11.913713
1442  11.655390  12.644328
1443  12.120834  11.703546
1444  11.753428  12.098487
1445  11.920762  11.767568
1446  12.400151  11.969717
1447  11.593975  12.388394
1448  11.473034  11.626254
1449  11.850328  11.429544
1450  12.507410  11.820410
1451  11.838413  12.567551
1452  11.543214  11.884489
1453  12.115119  11.344507
1454  12.079915  12.128111
1455  12.274108  12.072541
1456  12.464539  12.254863
1457  11.829097  12.493130
1458  11.911618  11.864462
1459        NaN  11.901583

[1460 rows x 2 columns]
```

Out[132]: 0.061898635910699447

In [232]:
```
#testing on test dataset
df_test = pd.read_csv("C:\\Users\\virin\\OneDrive\\Documents\\test.csv")
df_test.head(5)
```

Out[232]:

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Util
0	1461	20	RH	80.0	11622	Pave	NaN	Reg	Lvl	Al
1	1462	20	RL	81.0	14267	Pave	NaN	IR1	Lvl	Al
2	1463	60	RL	74.0	13830	Pave	NaN	IR1	Lvl	Al
3	1464	60	RL	78.0	9978	Pave	NaN	IR1	Lvl	Al
4	1465	120	RL	43.0	5005	Pave	NaN	IR1	HLS	Al

5 rows × 80 columns

In [233]:
```
#missing data
total = df_test.isnull().sum().sort_values(ascending=False)
percent = ((df_test.isnull().sum()/df_test.isnull().count())*100).sort_values(asc
missing_data = pd.concat([total, percent], axis=1, keys=['Total', 'Percent'])
missing_data.head(5)
```

Out[233]:

	Total	Percent
PoolQC	1456	99.794380
MiscFeature	1408	96.504455
Alley	1352	92.666210
Fence	1169	80.123372
FireplaceQu	730	50.034270

In [234]:
```
#dealing with missing data
df_test = df_test.drop((missing_data[missing_data['Total'] > 1]).index,1)
df_test.head(10)
df_test.columns.get_loc("TotalBsmtSF")
```

Out[234]: 27

In [235]:
```
from sklearn.preprocessing import Imputer
import numpy as np

imputer = Imputer(strategy = "median")
x =df_test.iloc[:, np.r_[43,27]]


df_test= df_test.fillna((x.median()), inplace=True)

#df_test = df_test.drop(df_test.loc[df_test['GarageCars'].isnull()].index)
#df_test = df_test.drop(df_test.loc[df_test['TotalBsmtSF'].isnull()].index)
df_test.isnull().sum().max() #just checking that there's no missing data missing.
```

Out[235]: 1

In [236]:
```
#applying log transformation
df_test['GrLivArea'] = np.log(df_test['GrLivArea'])
```

In [237]:
```
df_test['HasBsmt'] = pd.Series(len(df_test['TotalBsmtSF']), index=df_test.index)
df_test['HasBsmt'] = 0
df_test.loc[df_test['TotalBsmtSF']>0,'HasBsmt'] = 1
```

In [238]:
```
#transform data
df_test.loc[df_test['HasBsmt']==1,'TotalBsmtSF'] = np.log(df_test['TotalBsmtSF'])
```

```
C:\Users\virin\Anaconda3\lib\site-packages\ipykernel_launcher.py:2: RuntimeWarn
ing: divide by zero encountered in log
```

In [239]:
```
#convert categorical variable into dummy
df_test = pd.get_dummies(df_test)
df_test.head(10)
```

Out[239]:

	Id	MSSubClass	LotArea	OverallQual	OverallCond	YearBuilt	YearRemodAdd	BsmtFinSF1
0	1461	20	11622	5	6	1961	1961	468.0
1	1462	20	14267	6	6	1958	1958	923.0
2	1463	60	13830	5	5	1997	1998	791.0
3	1464	60	9978	6	6	1998	1998	602.0
4	1465	120	5005	8	5	1992	1992	263.0
5	1466	60	10000	6	5	1993	1994	0.0
6	1467	20	7980	6	7	1992	2007	935.0
7	1468	60	8402	6	5	1998	1998	0.0
8	1469	20	10176	7	5	1990	1990	637.0
9	1470	20	8400	4	5	1970	1970	804.0

10 rows × 192 columns

In [240]:
```
import math
df_test1 = df_test.loc[:,['OverallQual', 'GrLivArea', 'GarageCars', 'TotalBsmtSF'
tree_pred = pd.DataFrame(tree_reg.predict(df_test1), columns = ["SalePrice"])
#tree_pred = list(tree_pred)
tree_pred = np.e**(tree_pred)
tree_pred.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1459 entries, 0 to 1458
Data columns (total 1 columns):
SalePrice    1459 non-null float64
dtypes: float64(1)
memory usage: 11.5 KB
```

In [241]:
```
test_data_id = pd.DataFrame(df_test["Id"])
test_data_id.head(5)
```

Out[241]:

	Id
0	1461
1	1462
2	1463
3	1464
4	1465

In [242]:
```
output_data=pd.concat([test_data_id, tree_pred], axis = 1)
```

In [244]:
```
output_data.to_csv('output_data_1.csv', index = False)
```

In [ ]:
```

```