EXPERIMENT NO 7

HOUSE PRICE PREDICTION

IMPORTING LIBRARIES

In [25]:
```python
import pandas as pd
import numpy as np

import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.preprocessing import StandardScaler

from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error
from sklearn.model_selection import train_test_split
```

READING THE DATASET

In [3]:
```python
train = pd.read_csv('/DOCUMENTS/COLLEGE/CLASSES/EXPERIMENT_NO_7/train.csv')

test = pd.read_csv('/DOCUMENTS/COLLEGE/CLASSES/EXPERIMENT_NO_7/test.csv')
```

In [4]:
```python
train.head()
```

Out[4]:

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

5 rows × 81 columns

In [5]:
```python
test.head()
```

Out[5]:

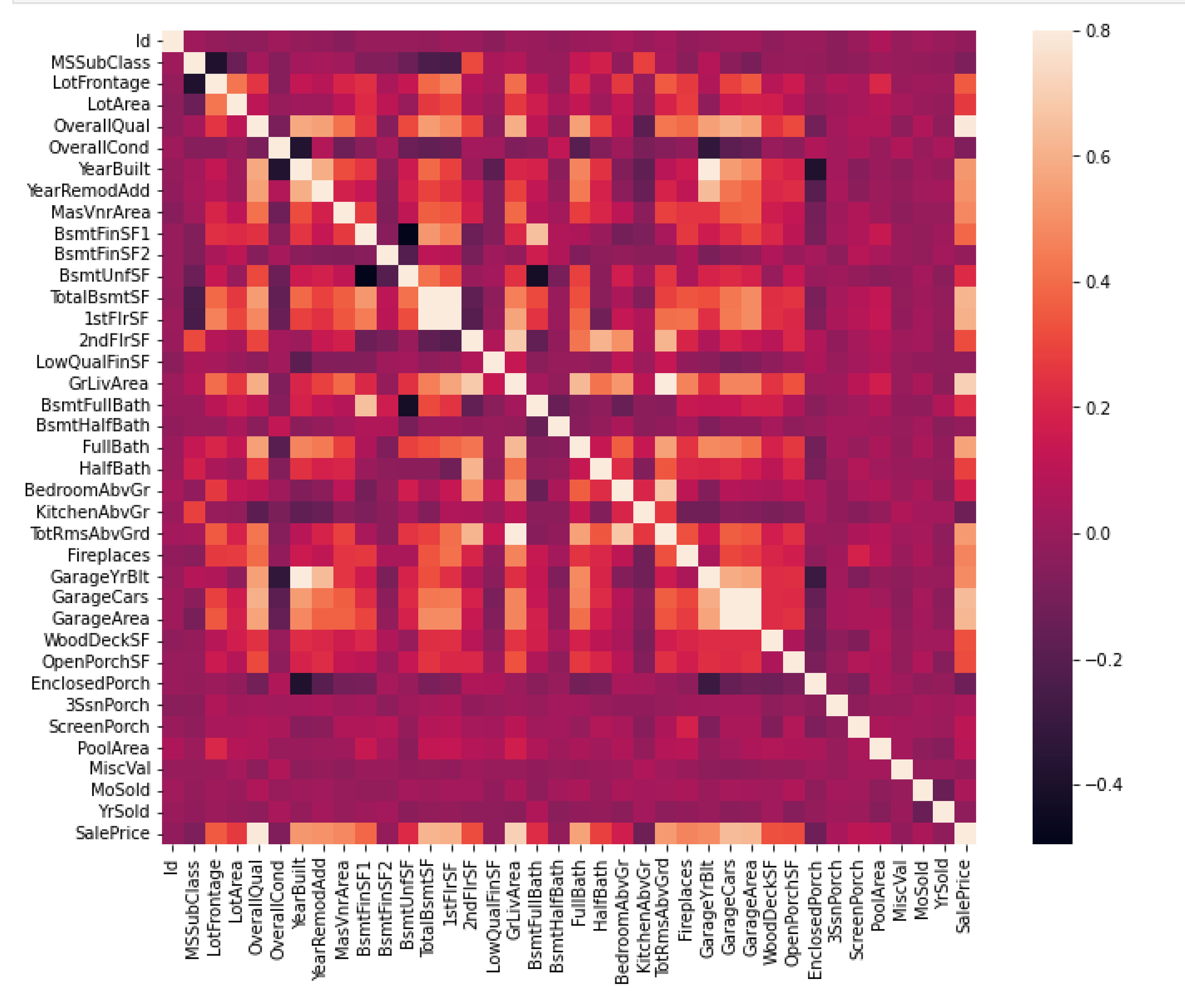
	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	...	ScreenPorch	PoolArea	PoolQC
0	1461	20	RH	80.0	11622	Pave	NaN	Reg	Lvl	AllPub	...	120	0	NaN
1	1462	20	RL	81.0	14267	Pave	NaN	IR1	Lvl	AllPub	...	0	0	NaN
2	1463	60	RL	74.0	13830	Pave	NaN	IR1	Lvl	AllPub	...	0	0	NaN
3	1464	60	RL	78.0	9978	Pave	NaN	IR1	Lvl	AllPub	...	0	0	NaN
4	1465	120	RL	43.0	5005	Pave	NaN	IR1	HLS	AllPub	...	144	0	NaN

5 rows × 80 columns

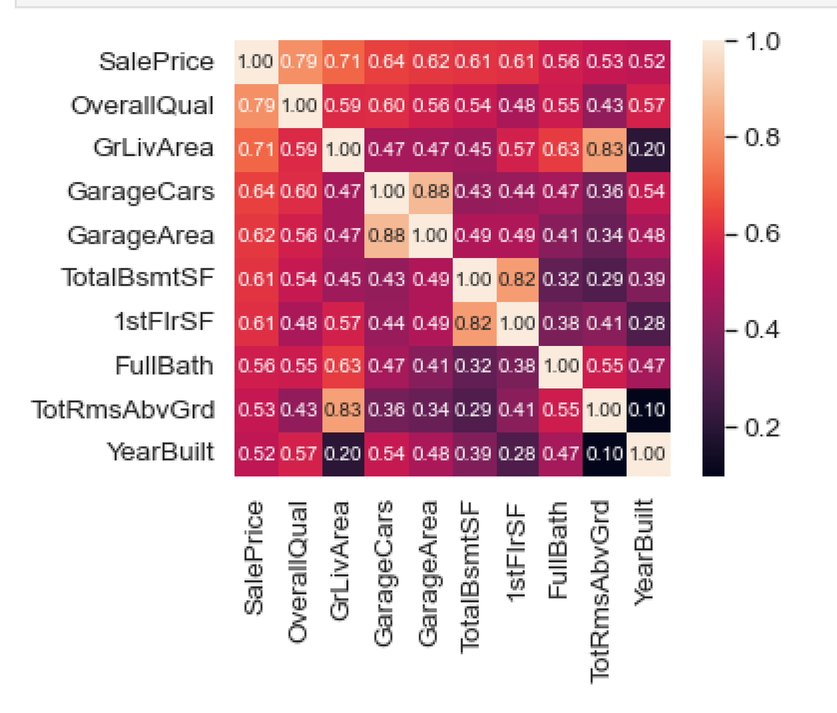
EXPLORATORY DATA ANALYSIS(EDA)

CORRELATION MATRIX

In [11]:
```python
# correlation matrix
corrmat = train.corr()
f, ax = plt.subplots(figsize=(12, 9))
sns.heatmap(corrmat, vmax=.8, square=True)
```

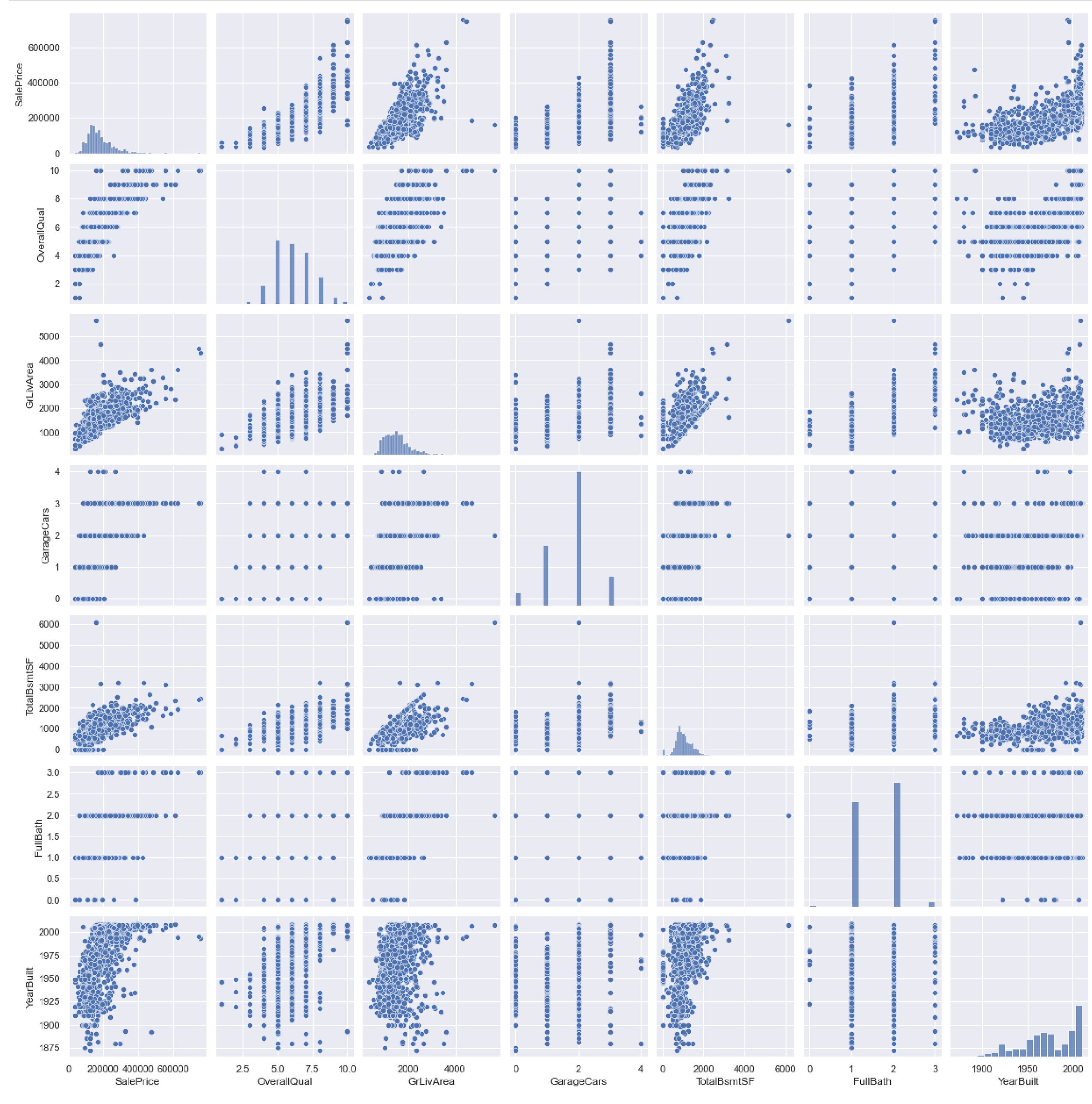


In [13]:
```python
# saleprice correlation matrix
k = 10  # number of variables for heatmap
cols = corrmat.nlargest(k, 'SalePrice')['SalePrice'].index
cm = np.corrcoef(train[cols].values.T)
sns.set(font_scale=1.25)
hm = sns.heatmap(cm,
                 cbar=True,
                 annot=True,
                 square=True,
                 fmt='.2f',
                 annot_kws={'size': 10},
                 yticklabels=cols.values,
                 xticklabels=cols.values)
plt.show()
```



DATA VISUALIZATION

In [16]:
```python
# scatterplot
sns.set()
cols = [
    'SalePrice', 'OverallQual', 'GrLivArea', 'GarageCars', 'TotalBsmtSF',
    'FullBath', 'YearBuilt'
]
sns.pairplot(train[cols], height=2.5)
plt.show()
```



CHECKING THE MISSING DATA

In [26]:
```python
test['SalePrice'] = 0
```

In [18]:
```python
# missing data
total = train.isnull().sum().sort_values(ascending=False)
percent = (train.isnull().sum() /
           train.isnull().count()).sort_values(ascending=False)
missing_data = pd.concat([total, percent], axis=1, keys=['Total', 'Percent'])
missing_data.head(20)
```

Out[18]:

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

In [27]:
```python
data = [train, test]
data = pd.concat(data)
```

In [28]:
```python
percent_missing = data.isnull().sum() * 100 / len(data)
missing_value = pd.DataFrame({
    'column': data.columns,
    'percent_missing': percent_missing
})
missing_value.sort_values('percent_missing', ascending=False)
missing_value.sort_values(by=['percent_missing'],
                          inplace=True,
                          ascending=False)
missing_value.head()
```

Out[28]:

	column	percent_missing
PoolQC	PoolQC	99.657417
MiscFeature	MiscFeature	96.402878
Alley	Alley	93.216855
Fence	Fence	80.438506
FireplaceQu	FireplaceQu	48.646797

In [29]:
```python
dropCols = missing_value["column"][missing_value["percent_missing"] > 47.0]
dropCols
```

Out[29]:
```
PoolQC              PoolQC
MiscFeature    MiscFeature
Alley                Alley
Fence                Fence
FireplaceQu    FireplaceQu
Name: column, dtype: object
```

In [30]:
```python
data.drop(columns=dropCols, inplace=True)
```

In [31]:
```python
data = data.fillna(data.mean())
```

```
C:\Users\HP\AppData\Local\Temp/ipykernel_12768/476127175.py:1: FutureWarning: Dropping of nuisance columns in D
ataFrame reductions (with 'numeric_only=None') is deprecated; in a future version this will raise TypeError.  S
elect only valid columns before calling the reduction.
  data = data.fillna(data.mean())
```

We'll consider that when more than 15% of the data is missing, we should delete the corresponding variable and pretend it never existed.

Checking Outliers

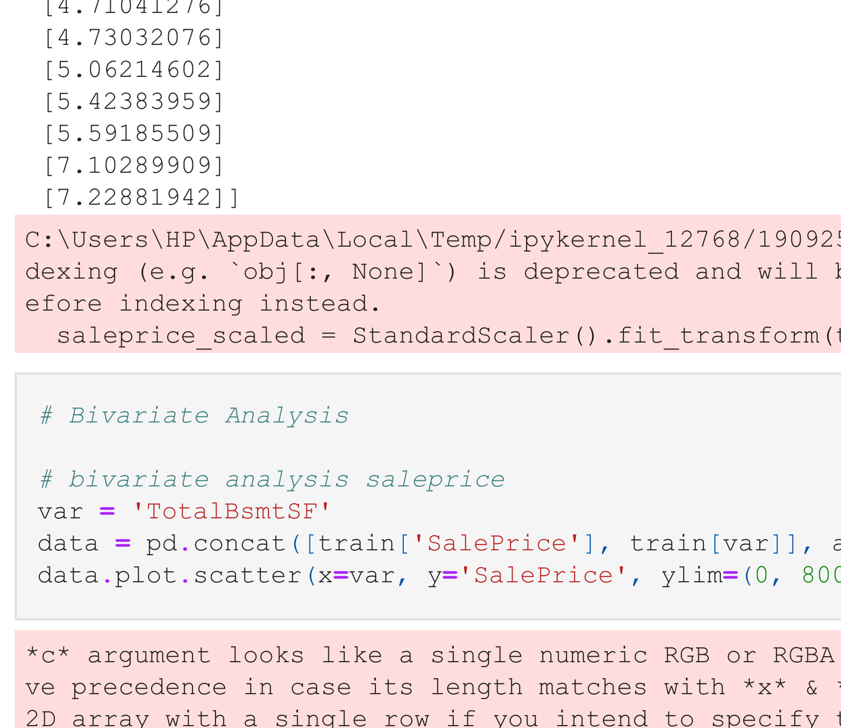
In [21]:
```python
# Univariate Analysis

# standardizing data
saleprice_scaled = StandardScaler().fit_transform(
    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:
[[-1.83870376]
 [-1.83352844]
 [-1.80092766]
 [-1.78329881]
 [-1.77448439]
 [-1.62337999]
 [-1.61708398]
 [-1.58560389]
 [-1.58560389]
 [-1.5731    ]]

outer range (high) of the distribution:
[[3.82897043]
 [4.04098249]
 [4.49634819]
 [4.71041276]
 [4.73032076]
 [5.06214602]
 [5.42383959]
 [5.59185509]
 [7.10289909]
 [7.22881942]]
```

```
C:\Users\HP\AppData\Local\Temp/ipykernel_12768/1909256962.py:4: FutureWarning: Support for multi-dimensional in
dexing (e.g. `obj[:, None]`) is deprecated and will be removed in a future version.  Convert to a numpy array b
efore indexing instead.
  saleprice_scaled = StandardScaler().fit_transform(train['SalePrice'][:,np.newaxis]);
```

In [23]:
```python
# Bivariate Analysis

# bivariate analysis saleprice
var = 'TotalBsmtSF'
data = pd.concat([train['SalePrice'], train[var]], axis=1)
data.plot.scatter(x=var, y='SalePrice', ylim=(0, 800000))
```

```
*c* argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value-mapping will ha
ve precedence in case its length matches with *x* & *y*.  Please use the *color* keyword-argument or provide a
2D array with a single row if you intend to specify the same RGB or RGBA value for all points.
```



MODEL BUILDING

In [61]:
```python
for i in data.columns:
    if (data[i].dtype == 'O'):
        data.drop(columns=[i], inplace=True)

data.drop(columns=['Id'], inplace=True)
```

In [73]:
```python
X_train, X_test, y_train, y_test = train_test_split(
    data.drop(columns=['SalePrice']), data['SalePrice'], test_size=.2)
```

Multiple Linear Regression

In [43]:
```python
from sklearn.linear_model import LinearRegression

multi_reg = LinearRegression()
multi_reg.fit(x_train, y_train)
print('Training Accuracy : ', multi_reg.score(x_train, y_train) * 100, '%')
```

```
Training Accuracy :  83.98680290654255 %
```

Lasso Regression

In [43]:
```python
from sklearn.linear_model import LinearRegression

multi_reg = LinearRegression()
multi_reg.fit(x_train, y_train)
print('Training Accuracy : ', multi_reg.score(x_train, y_train) * 100, '%')
```

```
Training Accuracy :  83.98680290654255 %
```

Random Forest Regressor

In [43]:
```python
from sklearn.linear_model import LinearRegression

multi_reg = LinearRegression()
multi_reg.fit(x_train, y_train)
print('Training Accuracy : ', multi_reg.score(x_train, y_train) * 100, '%')
```

```
Training Accuracy :  83.98680290654255 %
```

----x----x----x----x----x----x----x----x----x----x----x----x----x----x----x----x----x----x---