Notebook Content:

1. Imports
2. Exploratory Data Analysis
3. Transforming and Engineering Features

In [1]:
```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# visualization tools
import seaborn as sns
import matplotlib.pyplot as plt
plt.style.use('ggplot')

# sci-kit learn tools
from scipy.stats import skew
```

In [2]:
```
# given data imports
Housing_df = pd.read_csv('Housing.csv')
```

In [3]:
```
Housing_df.shape
```

Out[3]:
```
(1460, 81)
```

In [4]:
```
# drop target (dependent variable) from training dataframe
actual_y = Housing_df['SalePrice']
```

Exploratory Data Analysis

In [5]:
```
Housing_df.head()
```

Out[5]:

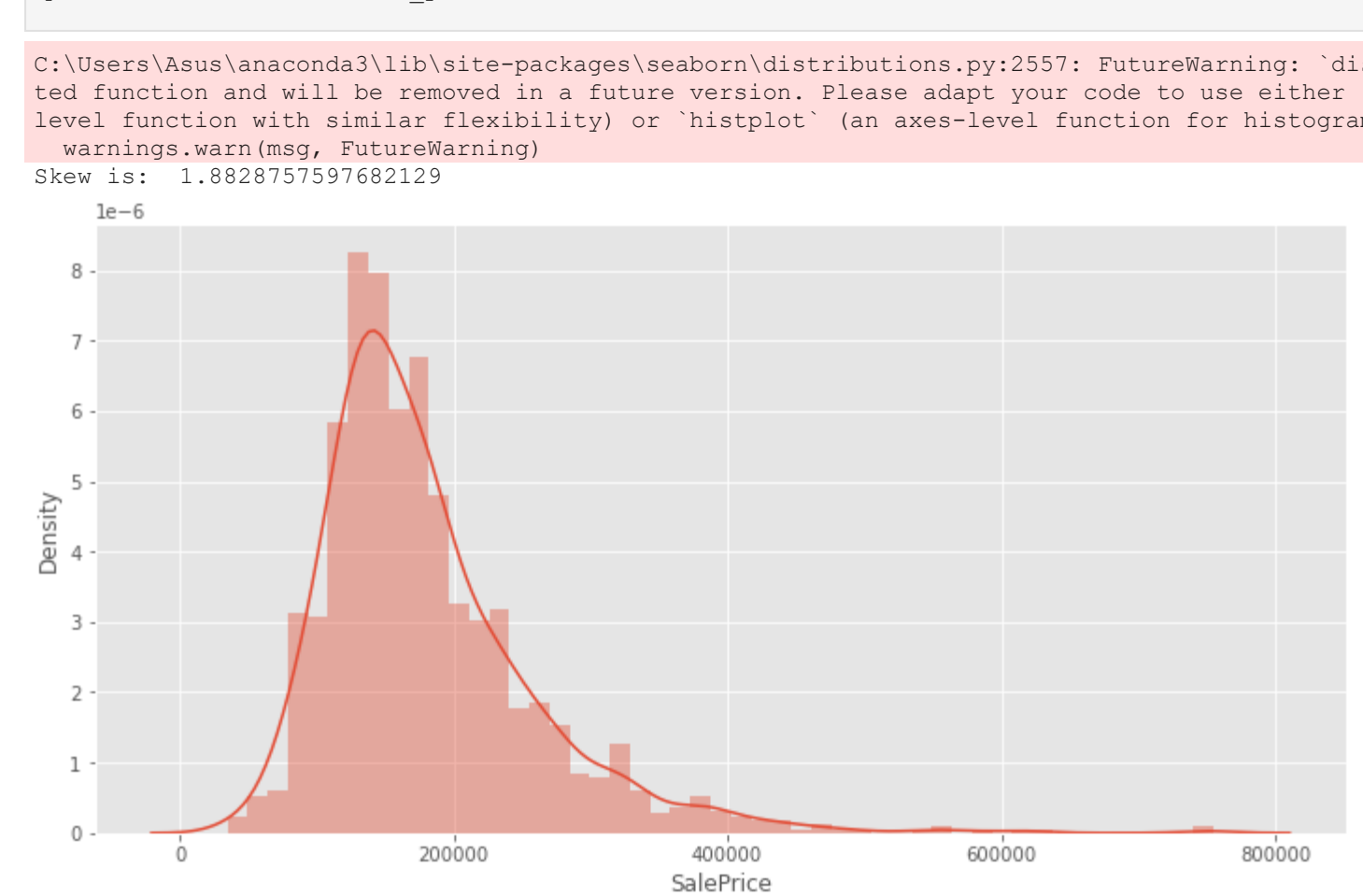
	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	...	PoolArea	PoolQC	Fence	MiscFeature
0	1	60	RL	65.0	8450	Pave	NaN	Reg	Lvl	AllPub	...	0	NaN	NaN	NaN
1	2	20	RL	80.0	9600	Pave	NaN	Reg	Lvl	AllPub	...	0	NaN	NaN	NaN
2	3	60	RL	68.0	11250	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN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

5 rows × 81 columns

> We can see that there exists many qualitative and missing values^
>
> Let's take a look at the skewness of SalePrice to see if a log transformation will be necessary for linear regression.

In [6]:
```
f, ax = plt.subplots(figsize=(12, 6))
sns.distplot(actual_y)
print("Skew is: ", actual_y.skew())
```

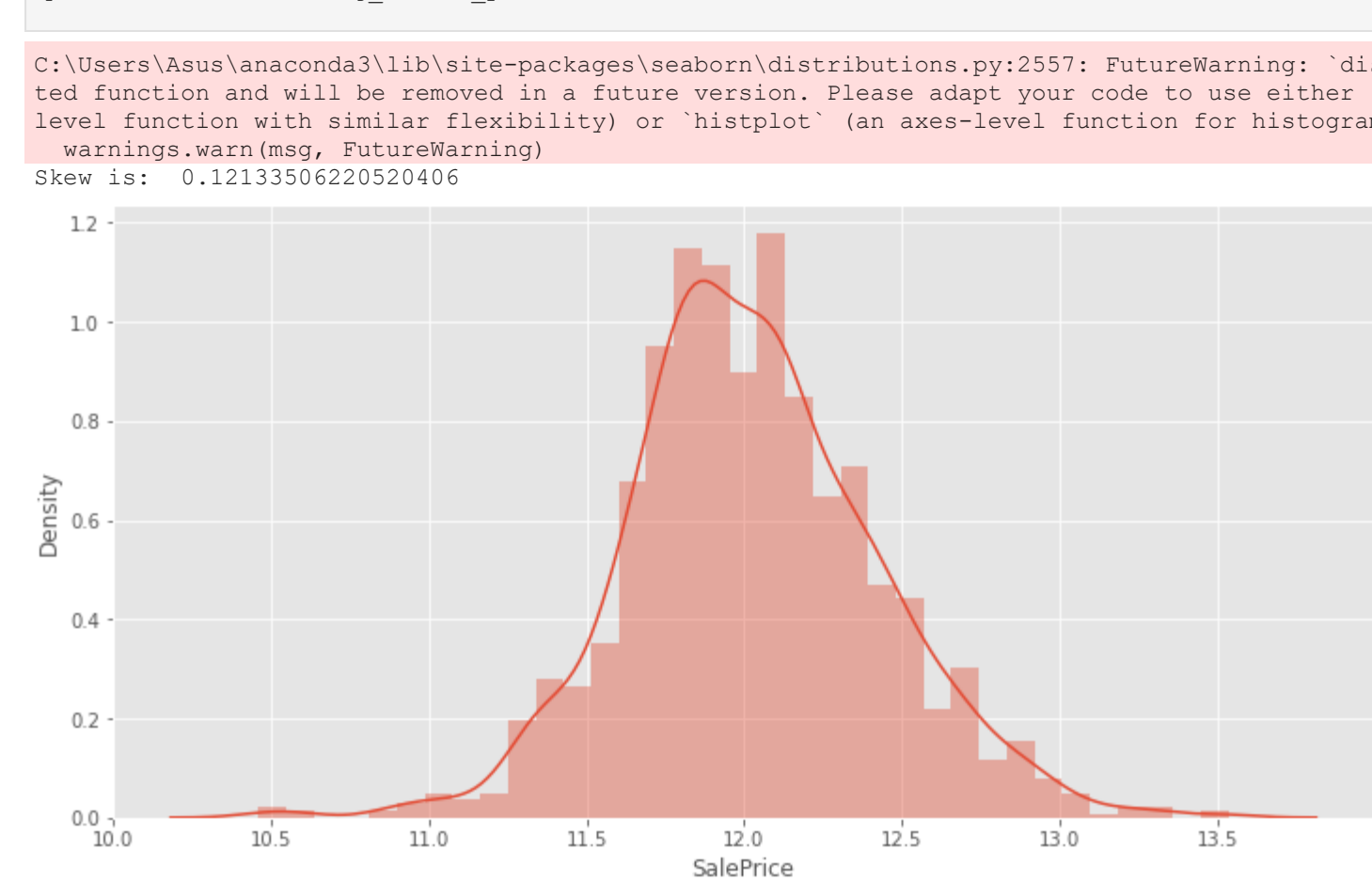
```
C:\Users\Asus\anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
Skew is:  1.8828757597682129
```



> You can see that the data is skewed. We will attempt to log-transform the data to bring the skew number closer to 0.

In [7]:
```
log_actual_y = np.log(actual_y)
f, ax = plt.subplots(figsize=(12, 6))
sns.distplot(log_actual_y)

print("Skew is: ", log_actual_y.skew())
```

```
C:\Users\Asus\anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
Skew is:  0.12133506220520406
```



> A skew value closer to 0 means that we have improved the skewness of the data. You can see from the plot that the logged data resembles a normal distribution!

We will handle quantitative and qualitative features seperately. We will begin with Quantitative features. We will examine correlation between actual SalePrice and quantitative features.

In [8]:
```
Housing_df.dtypes
```

Out[8]:
```
Id                 int64
MSSubClass         int64
MSZoning          object
LotFrontage      float64
LotArea            int64
                  ...   
MoSold             int64
YrSold             int64
SaleType          object
SaleCondition     object
SalePrice          int64
Length: 81, dtype: object
```

In [9]:
```
quant = Housing_df.select_dtypes(include=[np.number])
quant.dtypes
```

Out[9]:
```
Id                 int64
MSSubClass         int64
LotFrontage      float64
LotArea            int64
OverallQual        int64
OverallCond        int64
YearBuilt          int64
YearRemodAdd       int64
MasVnrArea       float64
BsmtFinSF1         int64
BsmtFinSF2         int64
BsmtUnfSF          int64
TotalBsmtSF        int64
1stFlrSF           int64
2ndFlrSF           int64
LowQualFinSF       int64
GrLivArea          int64
BsmtFullBath       int64
BsmtHalfBath       int64
FullBath           int64
HalfBath           int64
BedroomAbvGr       int64
KitchenAbvGr       int64
TotRmsAbvGrd       int64
Fireplaces         int64
GarageYrBlt      float64
GarageCars         int64
GarageArea         int64
WoodDeckSF         int64
OpenPorchSF        int64
EnclosedPorch      int64
3SsnPorch          int64
ScreenPorch        int64
PoolArea           int64
MiscVal            int64
MoSold             int64
YrSold             int64
SalePrice          int64
dtype: object
```

In [10]:
```
quant.head()
```

Out[10]:

	Id	MSSubClass	LotFrontage	LotArea	OverallQual	OverallCond	YearBuilt	YearRemodAdd	MasVnrArea	BsmtFinSF1	...	WoodDeckSF
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	...	298
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	...	192

5 rows × 38 columns

> Examine correlations between SalePrice and target

In [11]:
```
# First and last 5 most correlated values
corr = quant.corr()
corr
```

Out[11]:

	Id	MSSubClass	LotFrontage	LotArea	OverallQual	OverallCond	YearBuilt	YearRemodAdd	MasVnrArea	BsmtFinSF1
Id	1.000000	0.011156	-0.010601	-0.033226	-0.028365	0.012609	-0.012713	-0.021998	-0.050298	-0.005024
MSSubClass	0.011156	1.000000	-0.386347	-0.139781	0.032628	-0.059316	0.027850	0.040581	0.022936	-0.069836
LotFrontage	-0.010601	-0.386347	1.000000	0.426095	0.251646	-0.059213	0.123349	0.088866	0.193458	0.233633
LotArea	-0.033226	-0.139781	0.426095	1.000000	0.105806	-0.005636	0.014228	0.013788	0.104160	0.214103
OverallQual	-0.028365	0.032628	0.251646	0.105806	1.000000	-0.091932	0.572323	0.550684	0.411876	0.239666
OverallCond	0.012609	-0.059316	-0.059213	-0.005636	-0.091932	1.000000	-0.375983	0.073741	-0.128101	-0.046231
YearBuilt	-0.012713	0.027850	0.123349	0.014228	0.572323	-0.375983	1.000000	0.592855	0.315707	0.249503
YearRemodAdd	-0.021998	0.040581	0.088866	0.013788	0.550684	0.073741	0.592855	1.000000	0.179618	0.128451
MasVnrArea	-0.050298	0.022936	0.193458	0.104160	0.411876	-0.128101	0.315707	0.179618	1.000000	0.264736
BsmtFinSF1	-0.005024	-0.069836	0.233633	0.214103	0.239666	-0.046231	0.249503	0.128451	0.264736	1.000000
BsmtFinSF2	-0.005968	-0.065649	0.049900	0.111170	-0.059119	0.040229	-0.049107	-0.067759	-0.072319	-0.050117
BsmtUnfSF	-0.007940	-0.140759	0.132644	-0.002618	0.308159	-0.136841	0.149040	0.181133	0.114442	-0.495251
TotalBsmtSF	-0.015415	-0.238518	0.392075	0.260833	0.537808	-0.171098	0.391452	0.291066	0.363936	0.522396
1stFlrSF	0.010496	-0.251758	0.457181	0.299475	0.476224	-0.144203	0.281986	0.240379	0.344501	0.445863
2ndFlrSF	0.005590	0.307886	0.080177	0.050986	0.295493	0.028942	0.010308	0.140024	0.174561	-0.137079
LowQualFinSF	-0.044230	0.046474	0.038469	0.004779	-0.030429	0.025494	-0.183784	-0.062419	-0.069071	-0.064503
GrLivArea	0.008273	0.074853	0.402797	0.263116	0.593007	-0.079686	0.199010	0.287389	0.390857	0.208171
BsmtFullBath	0.002289	0.003491	0.100949	0.158155	0.111098	-0.054942	0.187599	0.119470	0.085310	0.649212
BsmtHalfBath	-0.020155	-0.002333	-0.007234	0.048046	-0.040150	0.117821	-0.038162	-0.012337	0.026673	0.067418
FullBath	0.005587	0.131608	0.198769	0.126031	0.550600	-0.194149	0.468271	0.439046	0.276833	0.058543
HalfBath	0.006784	0.177354	0.053532	0.014259	0.273458	-0.060769	0.242656	0.183331	0.201444	0.004262
BedroomAbvGr	0.037719	-0.023438	0.263170	0.119690	0.101676	0.012980	-0.070651	-0.040581	0.102821	-0.107355
KitchenAbvGr	0.002951	0.281721	-0.006069	-0.017784	-0.183882	-0.087001	-0.174800	-0.149598	-0.037610	-0.081007
TotRmsAbvGrd	0.027239	0.040380	0.352096	0.190015	0.427452	-0.057583	0.095589	0.191740	0.280682	0.044316
Fireplaces	-0.019772	-0.045569	0.266639	0.271364	0.396765	-0.023820	0.147716	0.112581	0.249070	0.260011
GarageYrBlt	0.000072	0.085072	0.070250	-0.024947	0.547766	-0.324297	0.825667	0.642277	0.252691	0.153484
GarageCars	0.016570	-0.040110	0.285691	0.154871	0.600671	-0.185758	0.537850	0.420622	0.364204	0.224054
GarageArea	0.017634	-0.098672	0.344997	0.180403	0.562022	-0.151521	0.478954	0.371600	0.373066	0.296970
WoodDeckSF	-0.029643	-0.012579	0.088521	0.171698	0.238923	-0.003334	0.224880	0.205726	0.159718	0.204306
OpenPorchSF	-0.000477	-0.006100	0.151972	0.084774	0.308819	-0.032589	0.188686	0.226298	0.125703	0.111761
EnclosedPorch	0.002889	-0.012037	0.010700	-0.018340	-0.113937	0.070356	-0.387268	-0.193919	-0.110204	-0.102303
3SsnPorch	-0.046635	-0.043825	0.070029	0.020423	0.030371	0.025504	0.031355	0.045286	0.018796	0.026451
ScreenPorch	0.001330	-0.026030	0.041383	0.043160	0.064886	0.054811	-0.050364	-0.038740	0.061466	0.062021
PoolArea	0.057044	0.008283	0.206167	0.077672	0.065166	-0.001985	0.004950	0.005829	0.011723	0.140491
MiscVal	-0.006242	-0.007683	0.003368	0.038068	-0.031406	0.068777	-0.034383	-0.010286	-0.029815	0.003571
MoSold	0.021172	-0.013585	0.011200	0.001205	0.070815	-0.003511	0.012398	0.021490	-0.005965	-0.015727
YrSold	0.000712	-0.021407	0.007450	-0.014261	-0.027347	0.043950	-0.013618	0.035743	-0.008201	0.014359
SalePrice	-0.021917	-0.084284	0.351799	0.263843	0.790982	-0.077856	0.522897	0.507101	0.477493	0.386420

38 rows × 38 columns

In [12]:
```
corr['SalePrice'].sort_values(ascending=False)
```

Out[12]:
```
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
```

In [13]:
```
corr['SalePrice'].sort_values(ascending=False)[:5]
```

Out[13]:
```
SalePrice      1.000000
OverallQual    0.790982
GrLivArea      0.708624
GarageCars     0.640409
GarageArea     0.623431
Name: SalePrice, dtype: float64
```

In [14]:
```
print(corr['SalePrice'].sort_values(ascending=False)[-5:])
```

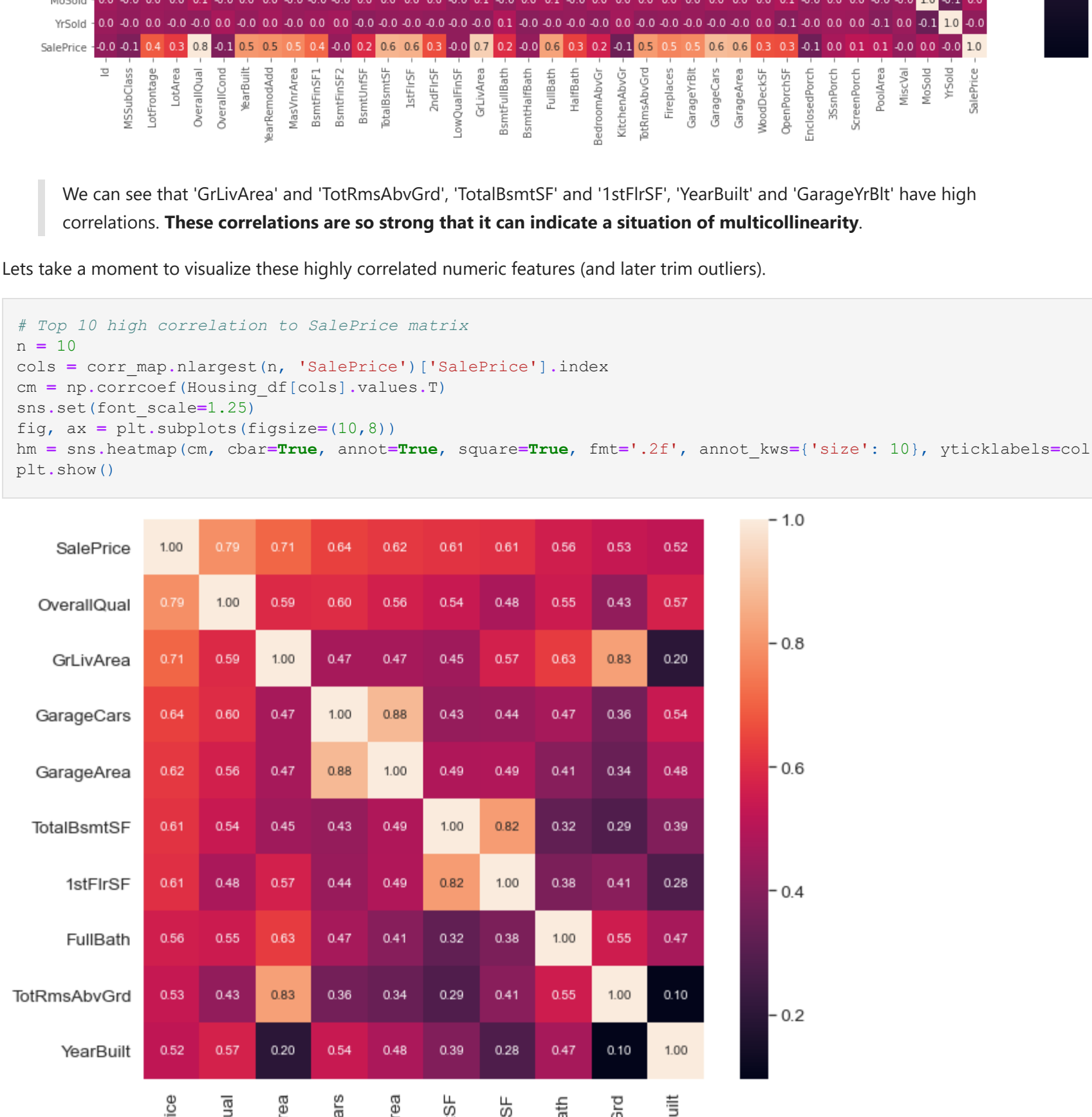
```
YrSold          -0.028923
OverallCond     -0.077856
MSSubClass      -0.084284
EnclosedPorch   -0.128578
KitchenAbvGr    -0.135907
Name: SalePrice, dtype: float64
```

We can now see the top 5 most postiviely correlated features with SalePrice.

> If your dataset has perfectly positive or negative attributes then there is a high chance that the performance of the model will be impacted by a problem called — "Multicollinearity"

Lets take a look at a correlation heatmap.

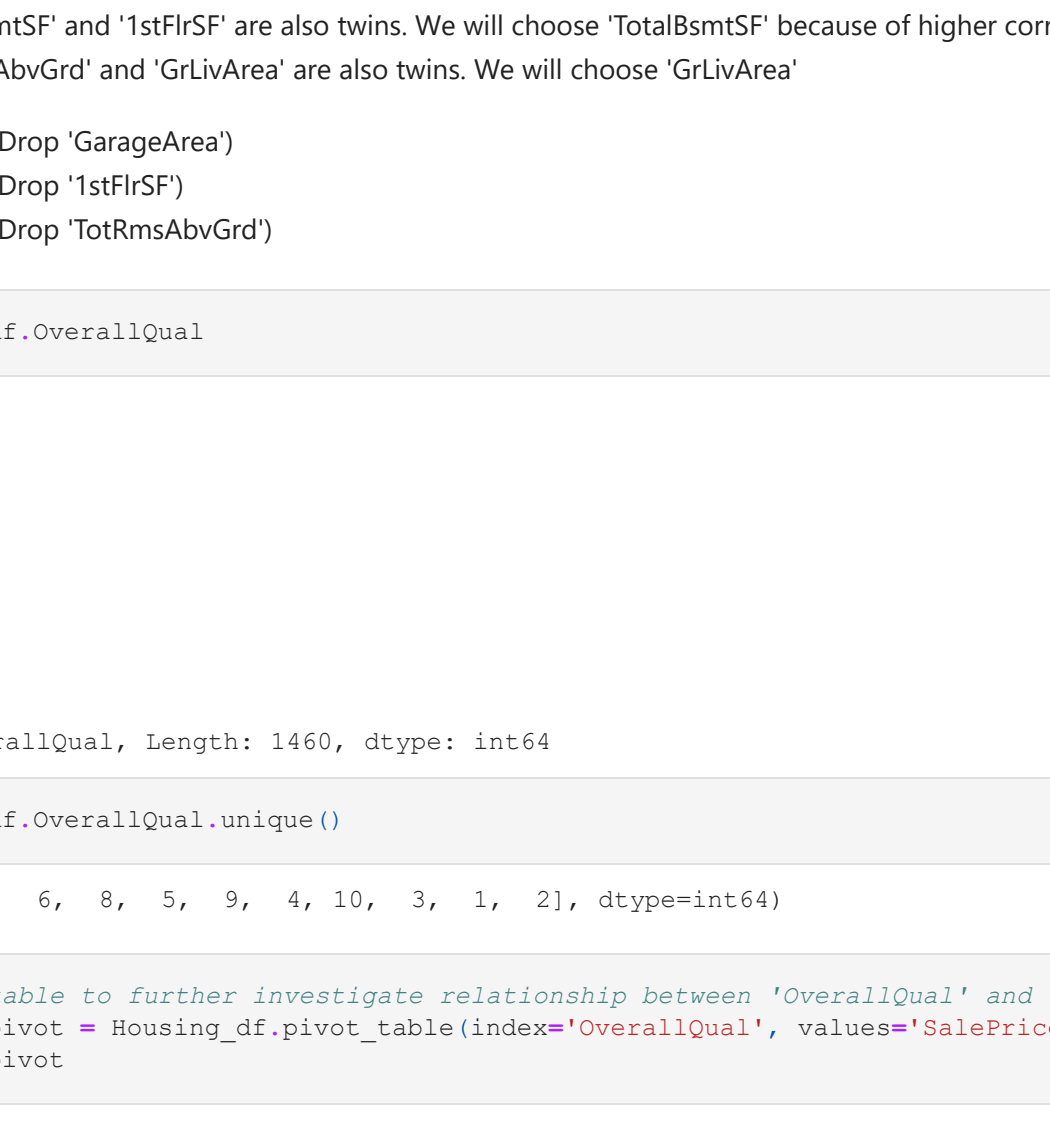
In [15]:
```
corr_map = Housing_df.corr()
fig, ax = plt.subplots(figsize=(20,16))
sns.heatmap(corr_map, vmax=.8, square=True, annot=True, fmt='.1f')
plt.show();
```



> We can see that 'GrLivArea' and 'TotRmsAbvGrd', 'TotalBsmtSF' and '1stFlrSF', 'YearBuilt' and 'GarageYrBlt' have high correlations. **These correlations are so strong that it can indicate a situation of multicollinearity**.

Lets take a moment to visualize these highly correlated numeric features (and later trim outliers).

In [16]:
```
# Top 10 high correlation to SalePrice matrix
n = 10
cols = corr_map.nlargest(n, 'SalePrice')['SalePrice'].index
cm = np.corrcoef(Housing_df[cols].values.T)
sns.set(font_scale=1.25)
fig, ax = plt.subplots(figsize=(10,8))
hm = sns.heatmap(cm, cbar=True, annot=True, square=True, fmt='.2f', annot_kws={'size': 10}, yticklabels=cols.va
plt.show()
```



- 'GarageCars' and 'GarageArea' are like twins. So we just need one. We will choose 'GarageCars' because it has a stronger correlation to 'SalePrice'!
- 'TotalBsmtSF' and '1stFlrSF' are also twins. We will choose 'TotalBsmtSF' because of higher correlation to SP
- 'TotRmsAbvGrd' and 'GrLivArea' are also twins. We will choose 'GrLivArea'

> - (Drop 'GarageArea')
> - (Drop '1stFlrSF')
> - (Drop 'TotRmsAbvGrd')

In [17]:
```
Housing_df.OverallQual
```

Out[17]:
```
0       7
1       6
2       7
3       7
4       8
       ..
1455    6
1456    6
1457    7
1458    5
1459    5
Name: OverallQual, Length: 1460, dtype: int64
```

In [18]:
```
Housing_df.OverallQual.unique()
```

Out[18]:
```
array([ 7,  6,  8,  5,  9,  4, 10,  3,  1,  2], dtype=int64)
```

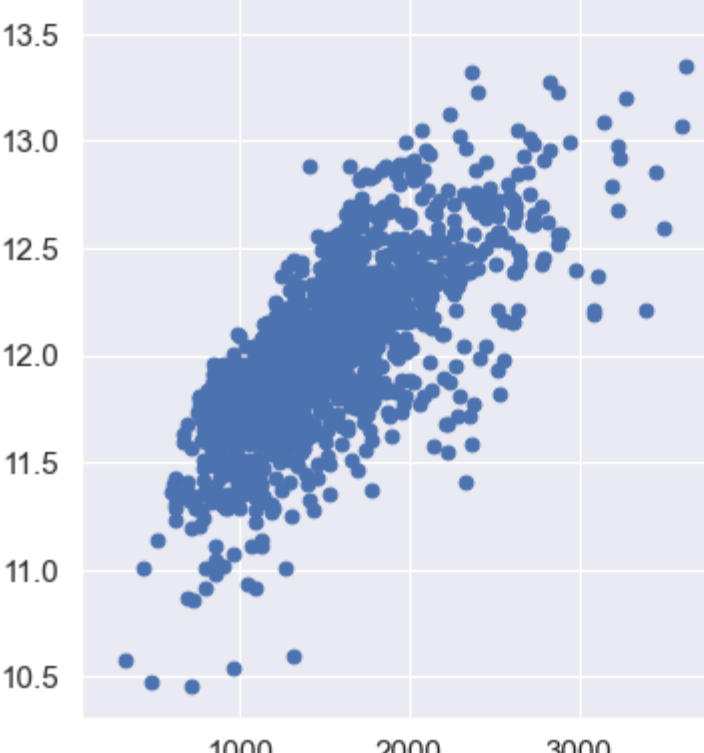
In [19]:
```
# pivot table to further investigate relationship between 'OverallQual' and 'SalePrice'
quality_pivot = Housing_df.pivot_table(index='OverallQual', values='SalePrice', aggfunc=np.median)
quality_pivot
```

Out[19]:

OverallQual	SalePrice
1	50150
2	60000
3	86250
4	108000
5	133000
6	160000
7	200141
8	269750
9	345000
10	432390

In [20]:
```
data = Housing_df[['OverallQual','SalePrice']]
data = data.groupby('OverallQual').max()
data = data.reset_index()
data
#f, ax = plt.subplots(figsize=(8, 4))
plt.plot(data['OverallQual'], data['SalePrice'], linestyle='solid')

#sns.lineplot(x='OverallQual', y = Housing_df.SalePrice, color='green',data=Housing_df)
```

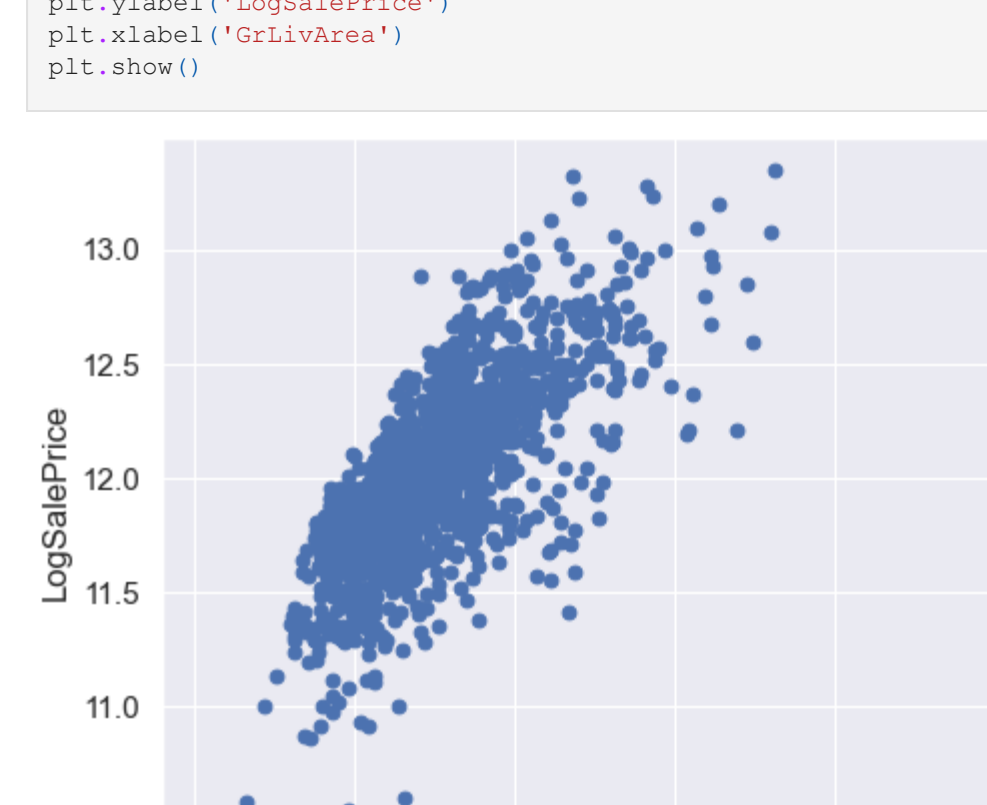
Out[20]:
```
[<matplotlib.lines.Line2D at 0x1d793af1970>]
```



> We can see that as the overall quality of the house increases so does the price of the house. This is an excellent variable for our model.

Now we will take a look at 'GrLivArea'

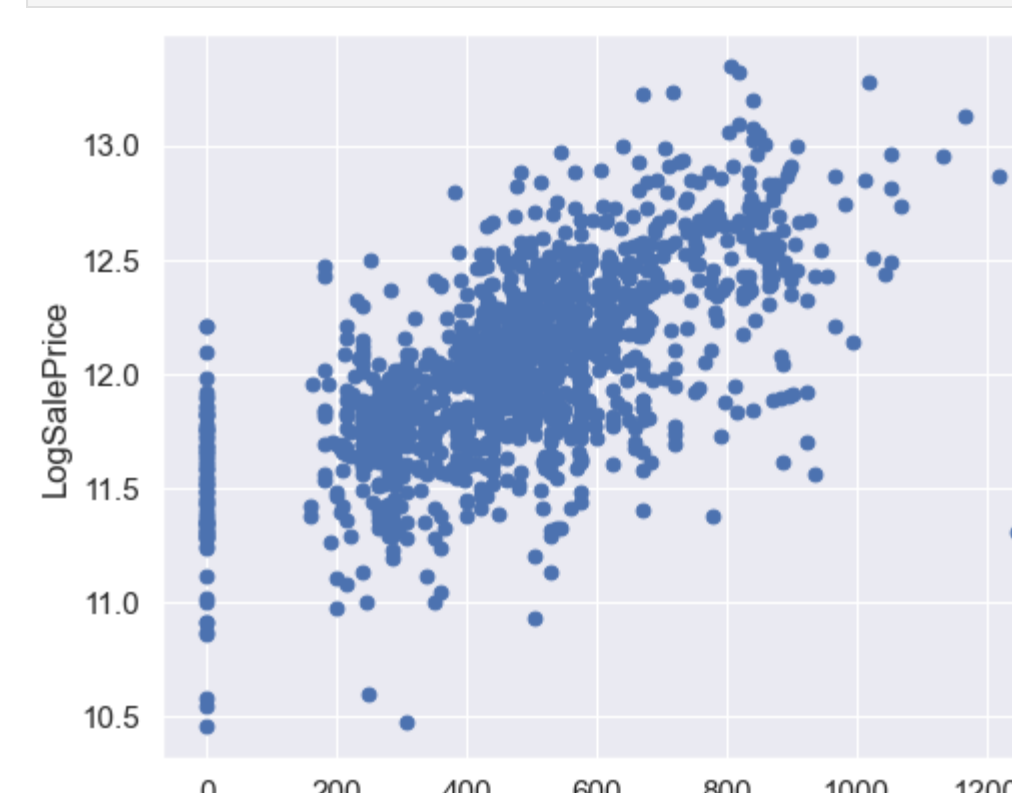
In [21]:
```
plt.figure(figsize=(8, 6), dpi=80)
plt.scatter(x = Housing_df['GrLivArea'], y = log_actual_y)
plt.ylabel('LogSalePrice')
plt.xlabel('GrLivArea')
plt.show()
```



> Outliers can affect a regression model by pulling our estimated regression line further away from the true population regression line. So, we'll remove those observations from our data.

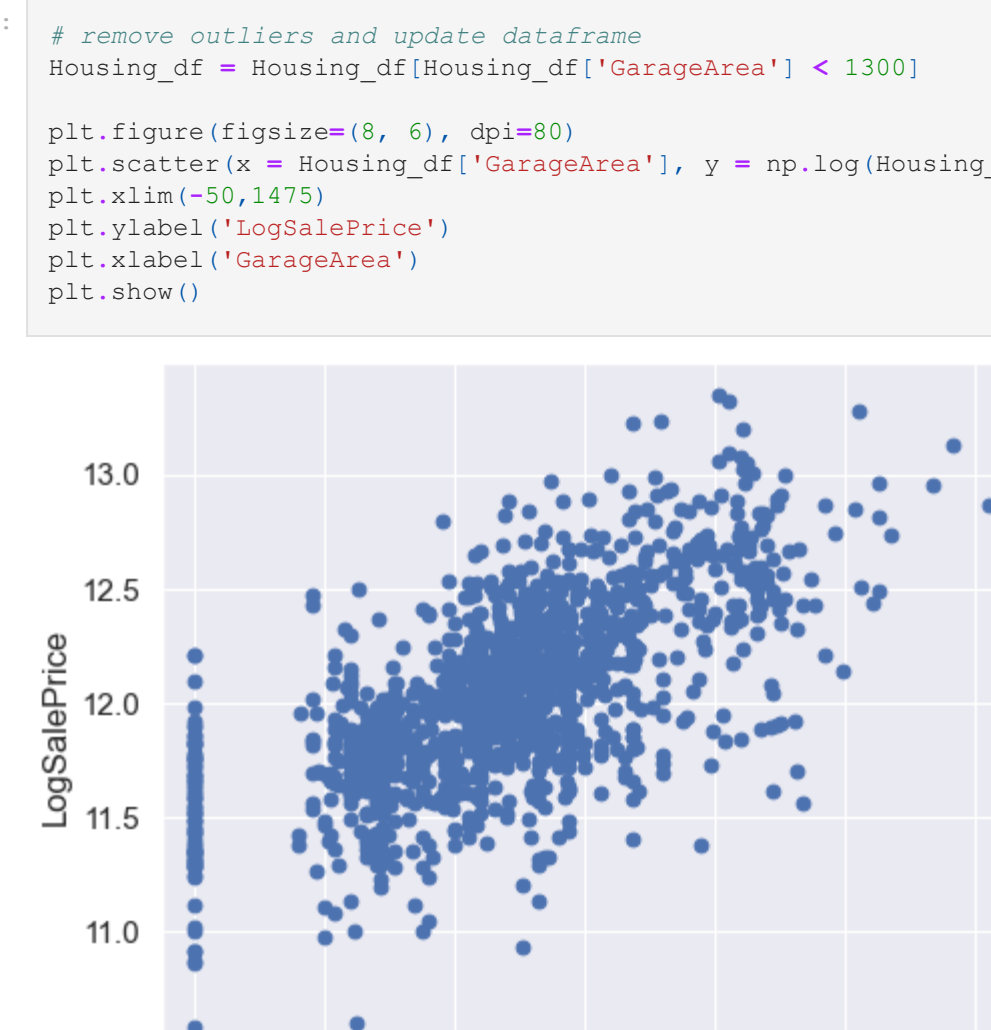
In [22]:
```
# remove outliers
Housing_df = Housing_df[Housing_df['GrLivArea'] < 4000]
```

In [23]:
```
plt.figure(figsize=(8, 6), dpi=80)
plt.scatter(x = Housing_df['GrLivArea'], y = np.log(Housing_df.SalePrice))
plt.xlim(-200,6000) # keeps same scale as first scatter plot
plt.ylabel('LogSalePrice')
plt.xlabel('GrLivArea')
plt.show()
```

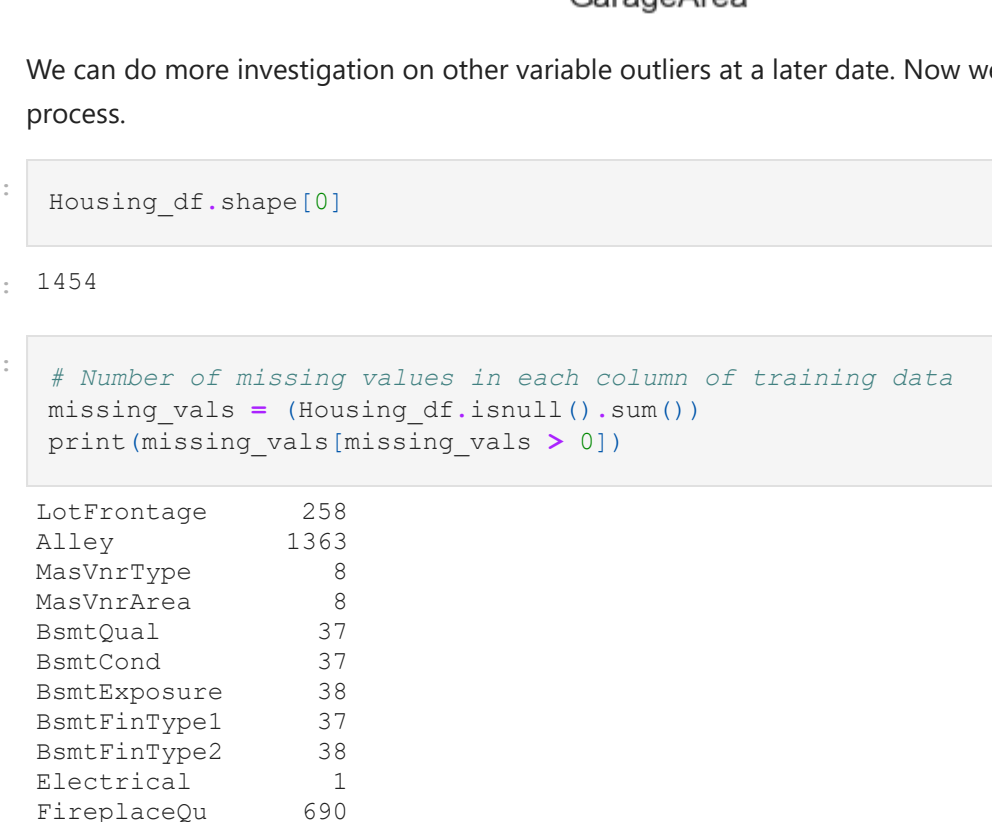


Now we will take a look at garage area.

In [24]:
```
plt.figure(figsize=(8, 6), dpi=80)
plt.scatter(x = Housing_df['GarageArea'], y = np.log(Housing_df.SalePrice))
plt.ylabel('LogSalePrice')
plt.xlabel('GarageArea')
plt.show()
```



In [25]:
```
# remove outliers and update dataframe
Housing_df = Housing_df[Housing_df['GarageArea'] < 1300]

plt.figure(figsize=(8, 6), dpi=80)
plt.scatter(x = Housing_df['GarageArea'], y = np.log(Housing_df.SalePrice))
plt.xlim(-50,1475)
plt.ylabel('LogSalePrice')
plt.xlabel('GarageArea')
plt.show()
```



We can do more investigation on other variable outliers at a later date. Now we will take a look at missing values and begin imputation process.

In [26]:
```
Housing_df.shape[0]
```

Out[26]:
```
1454
```

In [27]:
```
# Number of missing values in each column of training data
missing_vals = (Housing_df.isnull().sum())
print(missing_vals[missing_vals > 0])
```

```
LotFrontage      258
Alley           1363
MasVnrType         8
MasVnrArea         8
BsmtQual          37
BsmtCond          37
BsmtExposure      38
BsmtFinType1      37
BsmtFinType2      38
Electrical         1
FireplaceQu      690
GarageType        81
GarageYrBlt       81
GarageFinish      81
GarageQual        81
GarageCond        81
PoolQC          1449
Fence           1174
MiscFeature     1400
dtype: int64
```

We will drop all variables with a high amount of missing values. Why? None of these variables seem to be important or considered when deciding to buy a house (and that's probably why they have so many missing values).

> Drop: 'MiscFeature', 'Fence', 'PoolQC', 'FireplaceQu', 'Alley'

In regards to the Garage-related variables with missing values, we already have a garage variable with a high correlation to SalePrice. That variable alone will do the trick so we will delete all Garage variables with missing data.

> Drop: 'GarageType', 'GarageYrBlt', 'GarageFinish', 'GarageQual', 'GarageCond'

The same logic applies to Bsmt variables. Also MasVnr variables correlate heavily with 'OverallQual' so we will delete those.

We will delete everything except for Electrical.

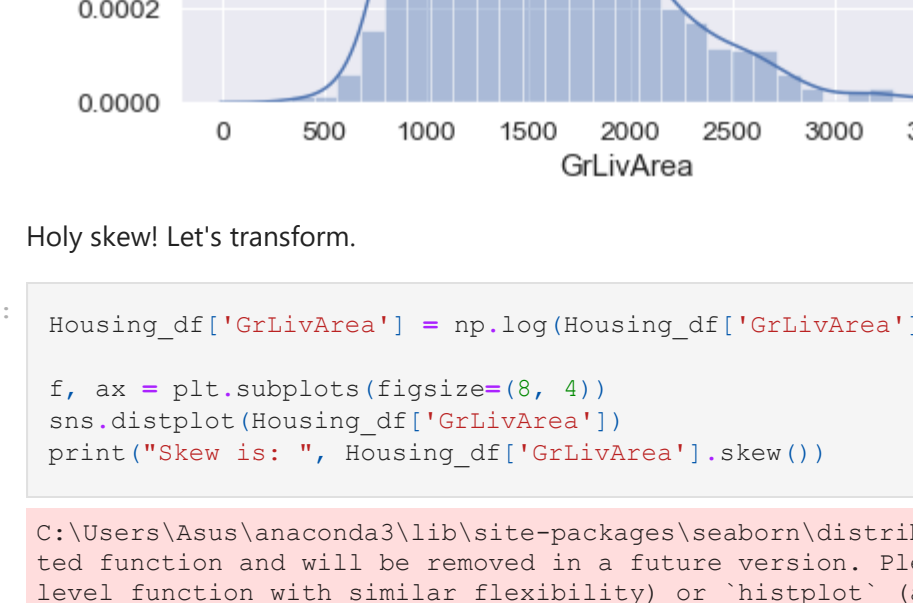
> General intution here was gathered from Comprehensive data exploration with Python

We will now begin an analysis on the normality of some of our very important features. Let's note our most important variables thus far:

> - OverallQual
> - GarageCars (recall all other garage variables have been dropped)
> - TotalBsmtSF (recall all other Bsmt variables have been dropped)
> - GrLivArea

In [28]:
```
# GrLivArea
f, ax = plt.subplots(figsize=(8, 4))
sns.distplot(Housing_df['GrLivArea'])
print("Skew is: ", Housing_df['GrLivArea'].skew())
```

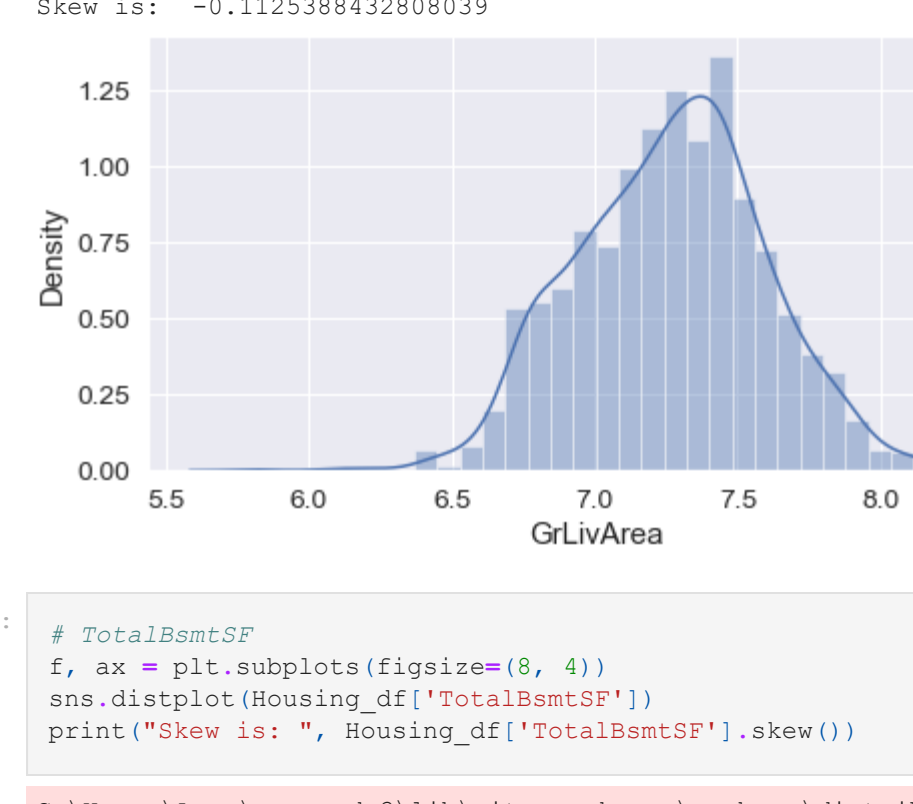
```
Skew is:  0.8375036132286743
C:\Users\Asus\anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



Holy skew! Let's transform.

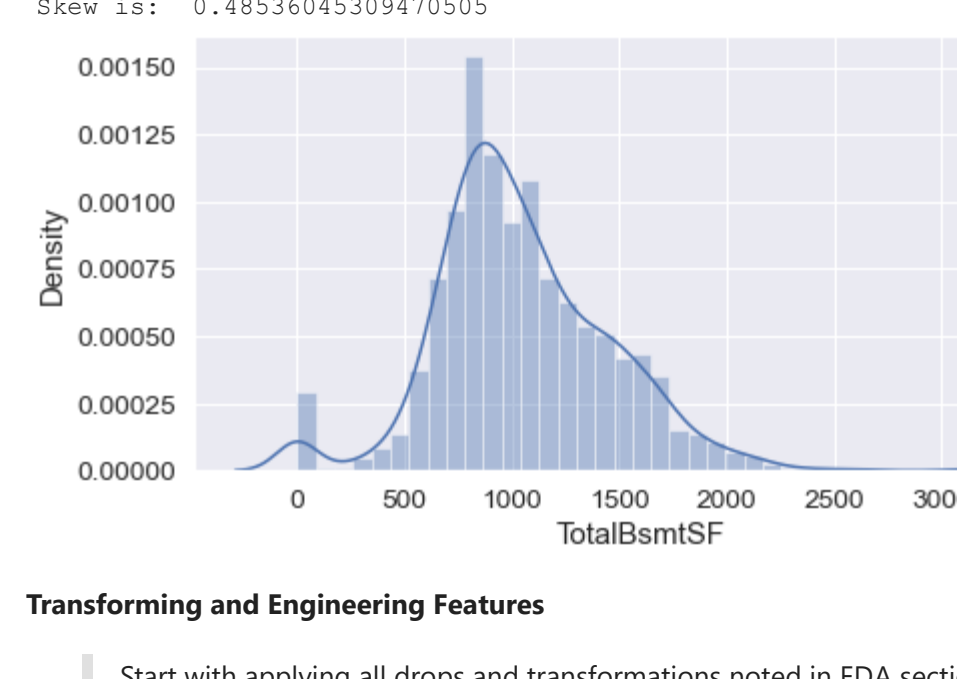
In [29]:
```
Housing_df['GrLivArea'] = np.log(Housing_df['GrLivArea'])

f, ax = plt.subplots(figsize=(8, 4))
sns.distplot(Housing_df['GrLivArea'])
print("Skew is: ", Housing_df['GrLivArea'].skew())
```

```
C:\Users\Asus\anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
Skew is:  -0.1125368432808039
```



In [30]:
```
# TotalBsmtSF
f, ax = plt.subplots(figsize=(8, 4))
sns.distplot(Housing_df['TotalBsmtSF'])
print("Skew is: ", Housing_df['TotalBsmtSF'].skew())
```

```
C:\Users\Asus\anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
Skew is:  0.48536045309470505
```



Transforming and Engineering Features

> Start with applying all drops and transformations noted in EDA section. For this section I reference: Regularized Linear Models

In [31]:
```
Housing_df.shape
```

Out[31]:
```
(1454, 81)
```

In [32]:
```
# we will begin by applying log transformation to skewed numeric features
num_data = Housing_df.dtypes[Housing_df.dtypes != "object"].index

skew_data = Housing_df[num_data].apply(lambda x: skew(x.dropna()))
skew_data = skew_data[skew_data > 0.75]
skew_data = skew_data.index
skew_data
Housing_df[skew_data] = np.log1p(Housing_df[skew_data])
```

For the sake of working quick, we will encode all qualitative variables with dummy representations. At a later point we will re-visit qualitative variables with a more granular approach.

In [33]:
```
Housing_df.shape
```

Out[33]:
```
(1454, 81)
```

In [34]:
```
missing_vals[missing_vals > 1]/len(Housing_df) *100
```

Out[34]:
```
LotFrontage     17.744154
Alley           93.741403
MasVnrType       0.550206
MasVnrArea       0.550206
BsmtQual         2.544704
BsmtCond         2.544704
BsmtExposure     2.613480
BsmtFinType1     2.544704
BsmtFinType2     2.613480
FireplaceQu     47.455296
GarageType       5.570839
GarageYrBlt      5.570839
GarageFinish     5.570839
GarageQual       5.570839
GarageCond       5.570839
PoolQC          99.656121
Fence           80.742779
MiscFeature     96.286107
dtype: float64
```

In [35]:
```
Housing_df.drop(columns=['MiscFeature','Fence','PoolQC','Alley','FireplaceQu'],inplace= True)
```

In [36]:
```
missing_vals = Housing_df.isnull().sum()
missing_vals[missing_vals > 0]
```

Out[36]:
```
LotFrontage     258
MasVnrType        8
MasVnrArea        8
BsmtQual         37
BsmtCond         37
BsmtExposure     38
BsmtFinType1     37
BsmtFinType2     38
GarageType       81
GarageYrBlt      81
GarageFinish     81
GarageQual       81
GarageCond       81
dtype: int64
```

In [37]:
```
Housing_df[["LotFrontage","MasVnrType","MasVnrArea",
"BsmtQual","BsmtCond",
"BsmtExposure",
"BsmtFinType1",
"BsmtFinType2",
"GarageType",
"GarageYrBlt","GarageFinish",
"GarageQual",
"GarageCond"]].head()
```

Out[37]:

	LotFrontage	MasVnrType	MasVnrArea	BsmtQual	BsmtCond	BsmtExposure	BsmtFinType1	BsmtFinType2	GarageType	GarageYrBlt	Garag
0	4.189655	BrkFace	5.283204	Gd	TA	No	GLQ	Unf	Attchd	2003.0	
1	4.394449	None	0.000000	Gd	TA	Gd	ALQ	Unf	Attchd	1976.0	
2	4.234107	BrkFace	5.093750	Gd	TA	Mn	GLQ	Unf	Attchd	2001.0	
3	4.110874	None	0.000000	TA	Gd	No	ALQ	Unf	Detchd	1998.0	
4	4.442651	BrkFace	5.860786	Gd	TA	Av	GLQ	Unf	Attchd	2000.0	

In [40]:
```
Housing_df['GarageYrBlt'] = Housing_df['GarageYrBlt'].astype('object')
```

In [41]:
```
# drop all features with missing values, noted above : keep electrical
Housing_df_num = Housing_df.select_dtypes(exclude='object')
```

In [42]:
```
Housing_df_num.isnull().sum() >0 # Only LotFrontage
```

Out[42]:
```
Id               False
MSSubClass       False
LotFrontage       True
LotArea          False
OverallQual      False
OverallCond      False
YearBuilt        False
YearRemodAdd     False
MasVnrArea        True
BsmtFinSF1       False
BsmtFinSF2       False
BsmtUnfSF        False
TotalBsmtSF      False
1stFlrSF         False
2ndFlrSF         False
LowQualFinSF     False
GrLivArea        False
BsmtFullBath     False
BsmtHalfBath     False
FullBath         False
HalfBath         False
BedroomAbvGr     False
KitchenAbvGr     False
TotRmsAbvGrd     False
Fireplaces       False
GarageCars       False
GarageArea       False
WoodDeckSF       False
OpenPorchSF      False
EnclosedPorch    False
3SsnPorch        False
ScreenPorch      False
PoolArea         False
MiscVal          False
MoSold           False
YrSold           False
SalePrice        False
dtype: bool
```

In [43]:
```
Housing_df_num['LotFrontage'] = Housing_df_num['LotFrontage'].fillna(Housing_df_num['LotFrontage'].median())
```

```
<ipython-input-43-44c4e709cc6d>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#ret
urning-a-view-versus-a-copy
  Housing_df_num['LotFrontage'] = Housing_df_num['LotFrontage'].fillna(Housing_df_num['LotFrontage'].median())
```

In [44]:
```
Housing_df_num['MasVnrArea'] = Housing_df_num['MasVnrArea'].fillna(Housing_df_num['MasVnrArea'].median())
```

```
<ipython-input-44-f09af01d3b90>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#ret
urning-a-view-versus-a-copy
  Housing_df_num['MasVnrArea'] = Housing_df_num['MasVnrArea'].fillna(Housing_df_num['MasVnrArea'].median())
```

In [45]:
```
Housing_df_num.isnull().sum()
```

Out[45]:
```
Id               0
MSSubClass       0
LotFrontage      0
LotArea          0
OverallQual      0
OverallCond      0
YearBuilt        0
YearRemodAdd     0
MasVnrArea       0
BsmtFinSF1       0
BsmtFinSF2       0
BsmtUnfSF        0
TotalBsmtSF      0
1stFlrSF         0
2ndFlrSF         0
LowQualFinSF     0
GrLivArea        0
BsmtFullBath     0
BsmtHalfBath     0
FullBath         0
HalfBath         0
BedroomAbvGr     0
KitchenAbvGr     0
TotRmsAbvGrd     0
Fireplaces       0
GarageCars       0
GarageArea       0
WoodDeckSF       0
OpenPorchSF      0
EnclosedPorch    0
3SsnPorch        0
ScreenPorch      0
PoolArea         0
MiscVal          0
MoSold           0
YrSold           0
SalePrice        0
dtype: int64
```

In [46]:
```
Housing_df_num.isnull().sum().sum()
```

Out[46]: 0

In [47]:
```
Housing_df_cat = Housing_df.select_dtypes(include='object')
```

In [48]:
```
Housing_df_cat.head()
```

Out[48]:

	MSZoning	Street	LotShape	LandContour	Utilities	LotConfig	LandSlope	Neighborhood	Condition1	Condition2	...	KitchenQual	Functi
0	RL	Pave	Reg	Lvl	AllPub	Inside	Gtl	CollgCr	Norm	Norm	...	Gd	
1	RL	Pave	Reg	Lvl	AllPub	FR2	Gtl	Veenker	Feedr	Norm	...	TA	
2	RL	Pave	IR1	Lvl	AllPub	Inside	Gtl	CollgCr	Norm	Norm	...	Gd	
3	RL	Pave	IR1	Lvl	AllPub	Corner	Gtl	Crawfor	Norm	Norm	...	Gd	
4	RL	Pave	IR1	Lvl	AllPub	FR2	Gtl	NoRidge	Norm	Norm	...	Gd	

5 rows × 39 columns

In [49]:
```
X = Housing_df_cat[Housing_df_cat['BsmtQual'].isnull()]
```

In [50]:
```
X[['BsmtExposure','BsmtCond','MasVnrType']].head()
```

Out[50]:

	BsmtExposure	BsmtCond	MasVnrType
17	NaN	NaN	None
39	NaN	NaN	None
90	NaN	NaN	None
102	NaN	NaN	None
156	NaN	NaN	None

In [51]:
```
#Housing_df_cat = Housing_df.select_dtypes(include='object')
Housing_df_cat_miss = Housing_df_cat.isnull().sum()
```

In [52]:
```
Housing_df_cat_miss[Housing_df_cat_miss>0]
```

Out[52]:
```
MasVnrType       8
BsmtQual        37
BsmtCond        37
BsmtExposure    38
BsmtFinType1    37
BsmtFinType2    38
GarageType      81
GarageYrBlt     81
GarageFinish    81
GarageQual      81
GarageCond      81
dtype: int64
```

In [53]:
```
Housing_df_cat = Housing_df_cat.fillna('None')
```

In [54]:
```
print(Housing_df_cat.isnull().sum().sum())
```

```
0
```

In [55]:
```
print(Housing_df_num.shape)
Housing_df_cat.shape
```

```
(1454, 37)
```

Out[55]: (1454, 39)

In [56]:
```
all_data = pd.concat([Housing_df_cat,Housing_df_num],axis=1)
all_data.shape
```

Out[56]: (1454, 76)

In [57]:
```
# drop variables noted in EDA section
drop_me = ['GarageArea', '1stFlrSF', 'TotRmsAbvGrd']
all_data = all_data.drop(drop_me, axis=1)
```

In [58]:
```
all_data = pd.get_dummies(all_data)
all_data.shape
```

Out[58]: (1454, 375)

In [60]:
```
# quick look under the hood
all_data.head()
```

Out[60]:

	Id	MSSubClass	LotFrontage	LotArea	OverallQual	OverallCond	YearBuilt	YearRemodAdd	MasVnrArea	BsmtFinSF1	...	SaleType_ConLw
0	1	4.110874	4.189655	9.042040	7	5	2003	2003	5.283204	706	...	0
1	2	3.044522	4.394449	9.169623	6	8	1976	1976	0.000000	978	...	0
2	3	4.110874	4.234107	9.328212	7	5	2001	2002	5.093750	486	...	0
3	4	4.262680	4.110874	9.164401	7	5	1915	1970	0.000000	216	...	0
4	5	4.110874	4.442651	9.565284	8	5	2000	2000	5.860786	655	...	0

5 rows × 375 columns

In [61]:
```
# split concatenated data into train and test dataframes
from sklearn.model_selection import train_test_split
y = all_data['SalePrice']
X = all_data.loc[:, all_data.columns != 'SalePrice']
print(X.shape)
```

```
(1454, 374)
```

In [62]:
```
X_train, X_test, y_train, y_test = train_test_split( X, y, test_size=0.3, random_state=42)
```

In [63]:
```
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(1017, 374)
(437, 374)
(1017,)
(437,)
```

Build the Model

We will attempt to apply the following models:

- Linear Regression
- Decision Tree
- Random Forests

In [64]:
```
X_train.head()
```

Out[64]:

	Id	MSSubClass	LotFrontage	LotArea	OverallQual	OverallCond	YearBuilt	YearRemodAdd	MasVnrArea	BsmtFinSF1	...	SaleType_C
464	465	3.044522	4.110874	9.039671	5	5	1978	1978	4.919981	616	...	
471	472	4.110874	4.532599	9.388738	7	6	1977	1977	0.000000	0	...	
1333	1334	3.931826	4.110874	8.881975	5	6	1938	1995	0.000000	0	...	
435	436	4.110874	3.784190	9.275004	7	6	1996	1996	0.000000	385	...	
1210	1211	4.110874	4.262680	9.325364	6	5	1992	1992	0.000000	0	...	

5 rows × 374 columns

In [65]:
```
y_train.head()
```

Out[65]:
```
464     11.728045
471     12.154785
1333    11.686879
435     12.264346
1210    12.149509
Name: SalePrice, dtype: float64
```

In [66]:
```
# Linear Regression
from sklearn import linear_model

linear_model = linear_model.LinearRegression()
lr_model = linear_model.fit(X_train, y_train)
y_pred_lr = lr_model.predict(X_test)
```

In [67]:
```
y_pred_lr
```

Out[67]:
```
array([ 1.20955316e+01,  1.23154814e+01,  1.17524551e+01,  1.19645058e+01,
        1.27121321e+01,  1.16236263e+01,  1.24001939e+01,  1.16364958e+01,
        1.20001366e+01,  1.25657689e+01,  1.16436152e+01,  1.17050129e+01,
        1.23356441e+01,  1.22850611e+01,  1.20533642e+01,  1.11890757e+01,
        1.21532837e+01,  1.21775063e+01,  1.18907407e+01,  1.22894803e+01,
        1.18606076e+01,  1.24288014e+01,  1.20530901e+01,  1.27606907e+01,
        1.21974186e+01,  1.18958778e+01, -3.75576329e+01,  1.15836265e+01,
        1.19127726e+01,  1.19045187e+01,  1.19300686e+01,  1.32180371e+01,
        1.18016928e+01,  1.18002746e+01,  1.17206225e+01,  1.18486879e+01,
        1.20011911e+01,  1.53997214e+01,  1.24366086e+01,  1.19219511e+01,
        1.16704554e+01,  1.23481948e+01,  1.17611648e+01,  1.13950014e+01,
        1.16916407e+01,  1.22122999e+01,  1.18948212e+01,  1.16424246e+01,
        1.20380074e+01,  1.22636276e+01,  1.16822087e+01,  1.24124402e+01,
        1.17965286e+01,  1.15778811e+01,  1.20557457e+01,  1.20273763e+01,
        1.17978138e+01,  1.15304859e+01,  1.15693998e+01,  1.16477115e+01,
        1.21089339e+01,  1.17923954e+01,  1.26107880e+01,  1.12353714e+01,
        1.24158050e+01,  1.21787973e+01,  1.28535807e+01,  1.15502395e+01,
        1.19823576e+01,  1.15724616e+01,  1.20000342e+01,  1.20236944e+01,
        1.16193324e+01,  1.20509484e+01,  1.17954933e+01,  1.16366832e+01,
        1.17819868e+01,  1.20803821e+01,  1.19094756e+01,  1.26332679e+01,
        1.19413669e+01,  1.27463105e+01,  1.22308158e+01,  1.25547097e+01,
        1.16414566e+01,  1.27508734e+01,  1.18763778e+01,  1.17799522e+01,
        1.17658905e+01,  1.21090931e+01,  1.14766208e+01,  1.17723698e+01,
        1.18034657e+01,  1.24899809e+01,  1.16854071e+01,  1.24589050e+01,
        1.23021917e+01,  1.19785545e+01,  1.22470061e+01,  1.18455764e+01,
        1.21435241e+01,  1.20251909e+01,  1.25161427e+01,  1.22960265e+01,
        1.20252717e+01, -5.85149639e+00,  1.25070921e+01,  1.16167470e+01,
        1.18447577e+01,  1.15185858e+01,  1.03700647e+01,  1.14020691e+01,
        1.18857801e+01,  1.22211576e+01,  1.19136709e+01, -6.37411760e+00,
        1.27617614e+01,  1.22070559e+01,  1.15460767e+01,  1.22574904e+01,
        1.22425968e+01,  1.18217632e+01,  1.23322288e+01,  1.14295362e+01,
        1.17548949e+01,  1.21778426e+01,  1.22478787e+01,  1.21049249e+01,
        1.16604019e+01,  1.19774716e+01,  1.16040497e+01,  1.19219763e+01,
        1.21664916e+01,  1.17691536e+01,  1.18736897e+01, -7.94006356e+00,
        1.19485769e+01,  1.20805818e+01,  1.19400113e+01,  1.15863213e+01,
        1.19502514e+01,  1.21155206e+01,  1.15701975e+01,  1.22079005e+01,
        1.18836176e+01,  1.19232802e+01,  1.23292878e+01,  1.19232850e+01,
        1.23458813e+01,  1.19133849e+01,  1.22633330e+01,  1.16779657e+01,
        1.21370357e+01,  1.18678135e+01,  1.13133935e+01,  1.19857888e+01,
        1.13001461e+01,  1.18357017e+01,  1.16737085e+01,  1.18360066e+01,
        1.16134419e+01,  1.14502865e+01,  1.23840870e+01,  1.26073796e+01,
        1.20160966e+01,  1.27666733e+01,  1.28165722e+01,  1.15986725e+01,
        1.21677239e+01,  1.25658759e+01,  1.21568652e+01,  1.21423138e+01,
        1.16763567e+01,  1.18762937e+01,  1.18583986e+01,  1.28045962e+01,
        1.17352757e+01,  1.19052893e+01,  1.16510642e+01,  1.25285155e+01,
        1.21600299e+01,  1.19861828e+01,  1.17906816e+01,  1.18541588e+01,
        1.20802704e+01,  1.19032974e+01,  1.23599019e+01,  1.21836835e+01,
        1.26950200e+01,  1.18932631e+01,  1.11920055e+01,  1.16527312e+01,
        1.21749201e+01,  1.19964854e+01,  1.19256130e+01,  1.18168865e+01,
        1.21256970e+01,  1.17005409e+01,  1.16381726e+01,  1.24149761e+01,
        1.21105554e+01,  1.17617770e+01,  1.23450383e+01,  1.25001723e+01,
        1.16344944e+01,  1.20066242e+01,  1.18761805e+01,  1.19588560e+01,
        1.17886726e+01,  1.21562108e+01,  1.16186677e+01,  1.17218114e+01,
        1.22302850e+01,  1.20925004e+01, -5.85144298e+00,  1.25573936e+01,
        1.21036860e+01,  1.16675814e+01,  1.26741844e+01,  1.21920035e+01,
        1.21625940e+01,  1.21536104e+01,  1.13724649e+01,  1.19880265e+01,
        1.21068494e+01,  1.26564191e+01,  1.17876548e+01,  1.18449045e+01,
        1.24312627e+01,  1.18058874e+01,  1.24124148e+01,  1.20302665e+01,
        1.23032810e+01,  1.23632476e+01,  1.21593616e+01,  1.30789862e+01,
        1.23113254e+01,  1.18876692e+01,  1.18438539e+01,  1.22356082e+01,
        1.18373423e+01,  1.17825463e+01,  1.22281945e+01,  1.17028259e+01,
        1.19055416e+01,  1.20158262e+01,  1.18453619e+01,  5.99972135e+01,
        1.25977331e+01,  1.22695145e+01,  1.22556610e+01,  1.23197685e+01,
        1.19028110e+01,  1.20606518e+01,  1.17580846e+01,  1.23272986e+01,
        1.22678800e+01,  1.22950894e+01,  1.19564390e+01,  1.15371735e+01,
        1.19658446e+01,  1.20005166e+01,  1.14988244e+01, -1.71317019e+01,
        1.23664158e+01,  1.17264946e+01,  1.14931209e+01,  1.20460209e+01,
        1.23797658e+01,  1.22189524e+01,  1.19888879e+01,  1.16862395e+01,
        1.21955479e+01,  1.24121416e+01,  1.21343951e+01,  1.14059971e+01,
        1.18365130e+01,  1.18691255e+01,  1.16068178e+01,  1.28780553e+01,
       -5.78971963e+00,  1.16312336e+01,  1.31137810e+01,  1.17878884e+01,
        1.21100515e+01,  1.23233070e+01,  1.17183228e+01,  1.21146372e+01,
        1.22082339e+01,  1.22328624e+01,  1.24032810e+01,  1.26547455e+01,
        1.24350580e+01,  1.14935457e+01,  1.21790277e+01,  1.31298889e+01,
        1.17946810e+01,  1.22373164e+01,  1.19126359e+01,  1.23728196e+01,
        1.20413279e+01,  1.15966163e+01,  1.19958817e+01,  1.21073854e+01,
        1.19341491e+01,  1.18572225e+01,  1.17061038e+01,  1.20415976e+01,
        1.16919046e+01,  1.18502883e+01,  1.19126129e+01,  1.15876116e+01,
        1.23495451e+01,  1.22245446e+01,  1.19957451e+01,  1.11877006e+01,
        1.16672463e+01,  1.21243335e+01,  1.20561996e+01,  1.14672704e+01,
        1.17333346e+01,  1.16225146e+01,  1.15930532e+01,  1.23772055e+01,
        1.13858263e+01,  1.20330763e+01,  1.16543932e+01,  1.20132953e+01,
        1.21216337e+01,  1.15979746e+01,  1.22546169e+01,  1.17820400e+01,
        1.22276486e+01,  1.18591345e+01,  1.20305441e+01,  1.28988777e+01,
        1.23074747e+01,  1.19350898e+01,  1.16925998e+01,  1.23443536e+01,
        1.21891312e+01,  1.20997384e+01,  1.22617446e+01,  1.19962529e+01,
        1.18337605e+01,  1.16902293e+01,  1.21697760e+01,  1.16762065e+01,
        1.20605024e+01,  1.19543874e+01,  1.16220560e+01,  1.23251675e+01,
        1.18418270e+01,  1.27463248e+01,  1.20041751e+01,  1.24697196e+01,
        1.19764452e+01,  1.16933268e+01,  1.24281022e+01,  1.21067805e+01,
        1.18582195e+01,  1.21040838e+01,  1.24107616e+01,  1.13714232e+01,
        1.18503734e+01,  1.20213651e+01,  1.19023036e+01,  1.22539327e+01,
        1.22974801e+01,  1.20290733e+01,  1.18961520e+01,  1.24025817e+01,
        1.18620212e+01,  1.21447551e+01,  1.20029706e+01,  1.16319359e+01,
        1.17265342e+01,  1.17436551e+01,  1.20031195e+01,  1.21719937e+01,
        1.17585208e+01,  1.17595707e+01,  1.17810624e+01,  1.24175851e+01,
        1.20902405e+01,  1.19600568e+01,  1.17859049e+01,  1.17039524e+01,
        1.17270855e+01,  1.16600366e+01,  1.21443088e+01,  8.42940330e+00,
        1.21909144e+01,  1.19945322e+01,  1.21463857e+01,  1.16075473e+01,
        1.20925159e+01])
```

In [70]:
```
combined_df = pd.concat([y_test, pd.DataFrame(y_pred_lr)], axis=1)
```

In [71]:
```
combined_df.head(10)
```

Out[71]:

	SalePrice	0
0	12.247699	11.706462
1	12.109016	12.178569
2	12.317171	11.712342
3	11.849405	12.290642
4	12.429220	12.101144
5	11.870607	11.555565
6	12.634606	11.627747
7	12.206073	11.628404
8	11.774528	12.379262
9	11.678448	11.766366

In [72]:
```
from sklearn.metrics import mean_squared_error
print(mean_squared_error(y_test, y_pred_lr))
```

```
0.006372769175265204
```

The smaller the mean squared error, the closer you are to finding the line of best fit.

In [74]:
```
from sklearn.tree import DecisionTreeRegressor
DT = DecisionTreeRegressor()
DT.fit(X_train, y_train)
Y_pred_dt =DT.predict(X_test)
print(mean_squared_error(y_train,DT.predict(X_train)))
print(mean_squared_error(y_test,DT.predict(X_test)))
```

```
0.0
0.04224470454455363
```

In [75]:
```
from sklearn.ensemble import RandomForestRegressor

rf_model = RandomForestRegressor(random_state=42, max_depth = 6, n_jobs = 5)
rf_model.fit(X_train, y_train)
rf_predict = rf_model.predict(X_test)
```

In [77]:
```
print(mean_squared_error(y_test,rf_predict))
print(mean_squared_error(y_train,rf_model.predict(X_train)))
```

```
0.010481864919806414
0.018016617454384974
```

In [78]:
```
rf_predict
```

Out[78]:
```
array([12.11176966, 11.87451613, 11.80267713, 11.76749491, 12.58522703,
       11.89102584, 11.20577322, 11.75407433, 11.05409399, 11.58202762,
       11.79817339, 11.70415184, 11.21903305, 11.19922363, 11.88197655,
       11.69856323, 12.20590614, 11.82995575, 11.37092031, 12.23520845,
       11.01125788, 12.36619364, 11.05460226, 11.69639218, 12.17464217,
       11.84669989, 11.64483385, 11.89569894, 11.95426851, 11.84639316,
       11.79292007, 12.05173832, 11.85590156, 11.64585015, 11.80444474,
       11.44047345, 11.60426517, 11.99905288, 11.28294769, 11.91610395,
       12.14157776, 12.37419157, 11.82515115, 11.86449021, 11.82459436,
       12.26314171, 11.76544064, 11.78056279, 11.93263217, 12.09395672,
       11.84305986, 11.70020581, 11.75723508, 11.78800115, 11.15382143,
       11.61610003, 11.66986593, 11.89561906, 11.80249903, 11.70965515,
       11.48159066, 11.66036617, 11.41810986, 11.17593153, 11.72005828,
       11.51500564, 11.80004391, 11.80011818, 11.69258099, 12.00894423,
       12.16580567, 12.32116908, 12.41184681, 11.44519899, 11.73621896,
       11.34591206, 11.63500731, 11.72254936, 11.25531687, 12.37029713,
       11.06105053, 11.54046113, 11.46866614, 11.22079656, 11.2557661 ,
       11.45911063, 12.01068498, 11.31516391, 11.13274649, 11.72277061,
       11.75862906, 11.20115588, 11.08906063, 11.31236593, 11.82569099,
       12.58468775, 12.04701006, 12.22047851, 11.31167755, 11.57403613,
       11.66104153, 11.70418848, 12.30064773, 11.10962559, 11.25091234,
       11.35155437, 11.96617614, 12.05134658, 12.69860084, 11.79476824,
       12.06573163, 11.96113779, 11.77567003, 11.75773586, 11.71340904,
       ...
       11.72622737, 12.20636242, 12.08292841, 11.75334242, 12.16391201,
       11.77750717, 11.80417925, 11.67304384, 11.17001607, 11.75730518,
       11.98123863, 11.95142951, 12.25182517, 11.86880421, 12.20541523,
       11.84134309, 12.13023624, 11.97080612, 11.36123837, 11.89569874,
       12.30857036, 11.83503009, 12.15152343, 11.8783536 , 11.76237734,
       11.6175871 , 12.21970116, 11.71816352, 11.70539317, 12.70823236,
       11.7856898 , 11.56417501, 12.22748023, 11.38905787, 11.99749692,
       12.09033126, 11.70919239, 11.99417453, 11.82725853, 11.64148788,
       12.22682679, 11.84794229, 11.8446958 , 11.71774981, 12.05798231,
       11.93134561, 11.82200152, 12.63938783, 12.07125652, 11.59851474,
       11.54748393, 11.79758836, 12.58573039, 11.80318522, 11.44064124,
       12.20398671, 11.64656843, 11.50367794, 12.27782159, 11.78406081,
       12.24860804, 11.79406118, 11.65203979, 11.56224823, 11.83630685,
       11.6393938 , 11.6953309 , 11.85055333, 11.62950757, 11.84381868,
       12.46582738, 12.4250004 , 11.7102196 , 11.76440187, 12.11297127,
       11.4836168 , 12.08171499, 12.57792646, 11.54062135, 11.74386024,
       12.16963517, 12.08659665, 12.00593817, 12.66006825, 11.75647028,
       11.81376329, 12.70551714, 12.12647083, 12.26296055, 11.6644956 ,
       12.36425691, 12.10326668, 11.76493449, 12.65247899, 11.71191168,
       11.79973253, 11.37459325, 11.79309128, 12.48943769, 11.04380799,
       11.43333546, 12.36544902, 12.24852318, 11.8330373 , 11.71490812,
       11.83657275, 11.25692852, 11.5765431 , 11.8754343 , 11.56905661,
       12.07644427, 11.77008   , 11.34593517, 12.15384976, 11.88604061,
       11.81497191, 12.62057436, 12.23028862, 12.25651132, 12.29031284,
       11.96671996, 11.97197558, 11.74686601, 12.26424215, 11.45801711,
       12.17485304, 12.00780518, 11.56535053, 11.55824478, 12.05853005,
       11.41632527, 11.29073383, 12.33186079, 11.52711931, 11.62294416,
       12.06714359, 11.59971577, 12.09982964, 12.35435907, 11.82183223,
       11.75768089, 11.569885  , 12.14716633, 11.87417441, 11.84483708,
       11.87948885, 12.16041335, 11.94378063, 11.71281797, 12.67808023,
       11.70074461, 11.72387298, 11.87454614, 11.72099489, 11.54387477,
       12.14646156, 11.47650797, 11.81347022, 11.11976001, 11.42595672,
       12.06709117, 11.35645101, 11.70345823, 12.09915776, 12.06758456,
       11.20356993, 11.98937847, 12.2339382 , 11.1744921 , 11.42785706,
       12.18364157, 11.11095305, 11.88585755, 12.09456628, 11.62842518,
       12.06319873, 12.16382701, 11.95364157, 11.84478893, 11.33280388,
       11.83171309, 11.80320456, 12.07920311, 11.59700222, 11.71257157,
       11.78635152, 11.74633242, 12.36952459, 11.68701619, 12.26520727,
       11.91700762, 12.10043357, 12.12645151, 12.36593278, 11.42912855,
       11.82617262, 11.61915411, 11.41884535, 11.52382143, 11.61023321,
       11.84888646, 11.80073513, 12.10718104, 12.31978024, 12.63589636,
       11.63973499, 11.68410799, 11.79862534, 11.58413981, 12.27782192,
       12.07466609, 11.82819246, 11.37806601, 12.04401072, 11.7914414 ,
       11.57949084, 11.7662055 , 11.74427297, 11.50709305, 12.26335003,
       11.72343652, 11.88495395, 12.51412985, 11.47966861, 12.08249846,
       11.18972032, 11.56789325, 12.09140974, 11.82951364, 11.75779851,
       11.84802287, 11.9665469 , 12.23330985, 11.71137723, 11.83288969,
       12.16243992, 12.44429404, 12.13395453, 11.8479999 , 12.32947808,
       11.10312749, 11.74073676, 11.83710406, 11.81441868, 12.16344577,
       11.1223808 , 11.56720531, 12.10298721, 11.10735794, 11.78112217,
       11.86558475, 12.02695435, 11.8384551 , 11.8329249 , 11.46893656,
       11.83661899, 11.97454344, 11.83664886, 11.69533707, 12.2923727 ,
       11.90443165, 11.89068505, 12.32521942, 11.61026757, 12.0490132 ,
       11.87429706, 11.82744498, 12.37615781, 11.83194306, 11.95076146,
       11.50587063, 11.60540568, 12.65462957, 12.0546323 , 11.17068192,
       11.63456053, 12.29253658, 11.81986603, 12.35016091, 11.84277504,
       11.88940313, 12.04287946, 12.60313648, 11.82969414, 11.78988142,
       11.85516633, 11.96144177, 12.07585139, 11.96187713, 11.8345284 ,
       12.62305552, 11.82394688, 11.7377175 , 11.92318745, 12.13181381,
       12.48720738, 12.08233089])
```

In [ ]: