In [112…

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import matplotlib.pyplot as plt
import scipy.stats as stats
import math
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import MinMaxScaler

df_train = pd.read_csv('train_house_price.csv')
```

In [113…

```
dependentVariable = df_train['SalePrice']
df_train
```

Out[113…

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour
0	1	60	RL	65.0	8450	Pave	NaN	Reg	Lv
1	2	20	RL	80.0	9600	Pave	NaN	Reg	Lv
2	3	60	RL	68.0	11250	Pave	NaN	IR1	Lv
3	4	70	RL	60.0	9550	Pave	NaN	IR1	Lv
4	5	60	RL	84.0	14260	Pave	NaN	IR1	Lv
...	...	...	...	...	...	...	...	...	..
1455	1456	60	RL	62.0	7917	Pave	NaN	Reg	Lv
1456	1457	20	RL	85.0	13175	Pave	NaN	Reg	Lv
1457	1458	70	RL	66.0	9042	Pave	NaN	Reg	Lv
1458	1459	20	RL	68.0	9717	Pave	NaN	Reg	Lv
1459	1460	20	RL	75.0	9937	Pave	NaN	Reg	Lv

1460 rows × 81 columns

Provide appropriate descriptive statistics and visualizations to help understand the marginal distribution of the dependent variable.

In [101…

```
df_train.describe()
```

Out[101…

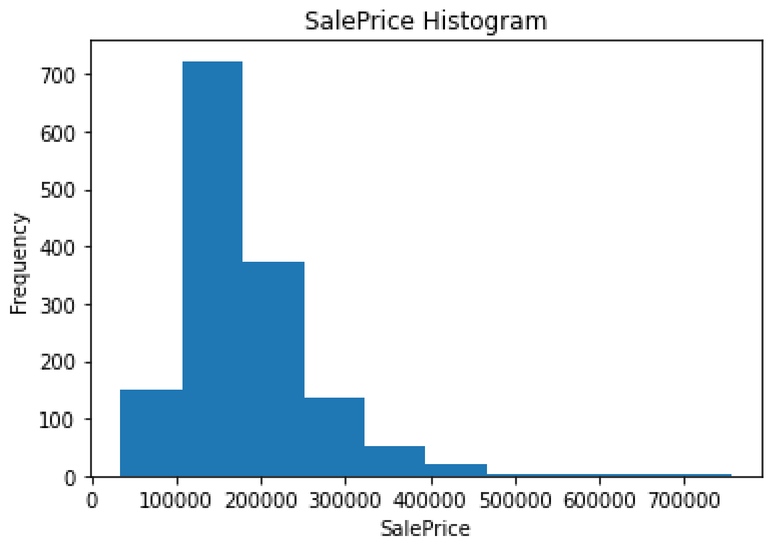
	Id	MSSubClass	LotFrontage	LotArea	OverallQual	OverallCond	Ye
count	1460.000000	1460.000000	1201.000000	1460.000000	1460.000000	1460.000000	1460.(
mean	730.500000	56.897260	70.049958	10516.828082	6.099315	5.575342	1971.:
std	421.610009	42.300571	24.284752	9981.264932	1.382997	1.112799	30.:
min	1.000000	20.000000	21.000000	1300.000000	1.000000	1.000000	1872.(
25%	365.750000	20.000000	59.000000	7553.500000	5.000000	5.000000	1954.(
50%	730.500000	50.000000	69.000000	9478.500000	6.000000	5.000000	1973.(

	Id	MSSubClass	LotFrontage	LotArea	OverallQual	OverallCond	Ye
75%	1095.250000	70.000000	80.000000	11601.500000	7.000000	6.000000	2000.(
max	1460.000000	190.000000	313.000000	215245.000000	10.000000	9.000000	2010.(

8 rows × 38 columns

In [102…

```
plt.hist(dependentVariable)
plt.xlabel('SalePrice')
plt.ylabel('Frequency')
plt.title('SalePrice Histogram')
plt.show()
```



Investigate missing data and outliers.

In [118…

```
nullTotals = df_train.isnull().sum().sort_values(ascending = False)
percentageOfNull = (df_train.isnull().sum() / df_train.isnull().count()).sort_va
emptyVals = pd.concat([nullTotals, percentageOfNull], axis=1, keys=['Total Missi
emptyVals.head(20)
```

Out[118…

	Total Missing Values	Percentage of Feature Specific Data that is Null
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

	Total Missing Values	Percentage of Feature Specific Data that is Null
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 [119…

```
quartile1 = dependentVariable.quantile(0.25)
quartile3 = dependentVariable.quantile(0.75)
IQR = quartile3 - quartile1
totalOutliers = ((dependentVariable < (Q1 - 1.5 * IQR)) | (dependentVariable > (
print("IQR value: {}\nTotal amount of outliers within SalePrice: {}".format(IQR,
```

```
IQR value: 84025.0
Total amount of outliers within SalePrice: 61
```

Investigate at least three potential predictors of the dependent variable and provide appropriate graphs / statistics to demonstrate the relationships.

In [120…

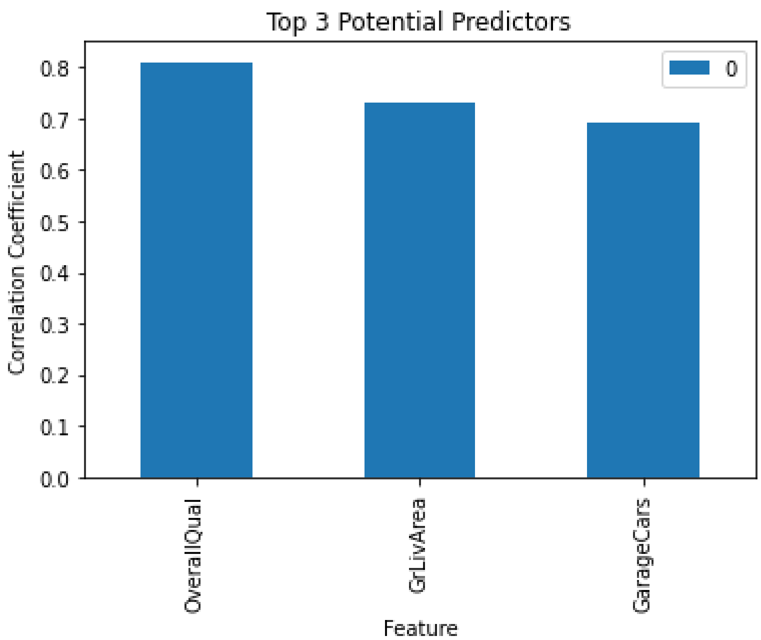
```
correlations = df_train.corr(method='spearman')['SalePrice'].sort_values(ascendi
correlations_abs = correlations.abs()
print('\nTop 10 correlations (absolute):\n', correlations_abs.head(11))
```

```
Top 10 correlations (absolute):
 SalePrice       1.000000
OverallQual     0.809829
GrLivArea       0.731310
GarageCars      0.690711
YearBuilt       0.652682
GarageArea      0.649379
FullBath        0.635957
TotalBsmtSF     0.602725
GarageYrBlt     0.593788
1stFlrSF        0.575408
YearRemodAdd    0.571159
Name: SalePrice, dtype: float64
```

In [129…

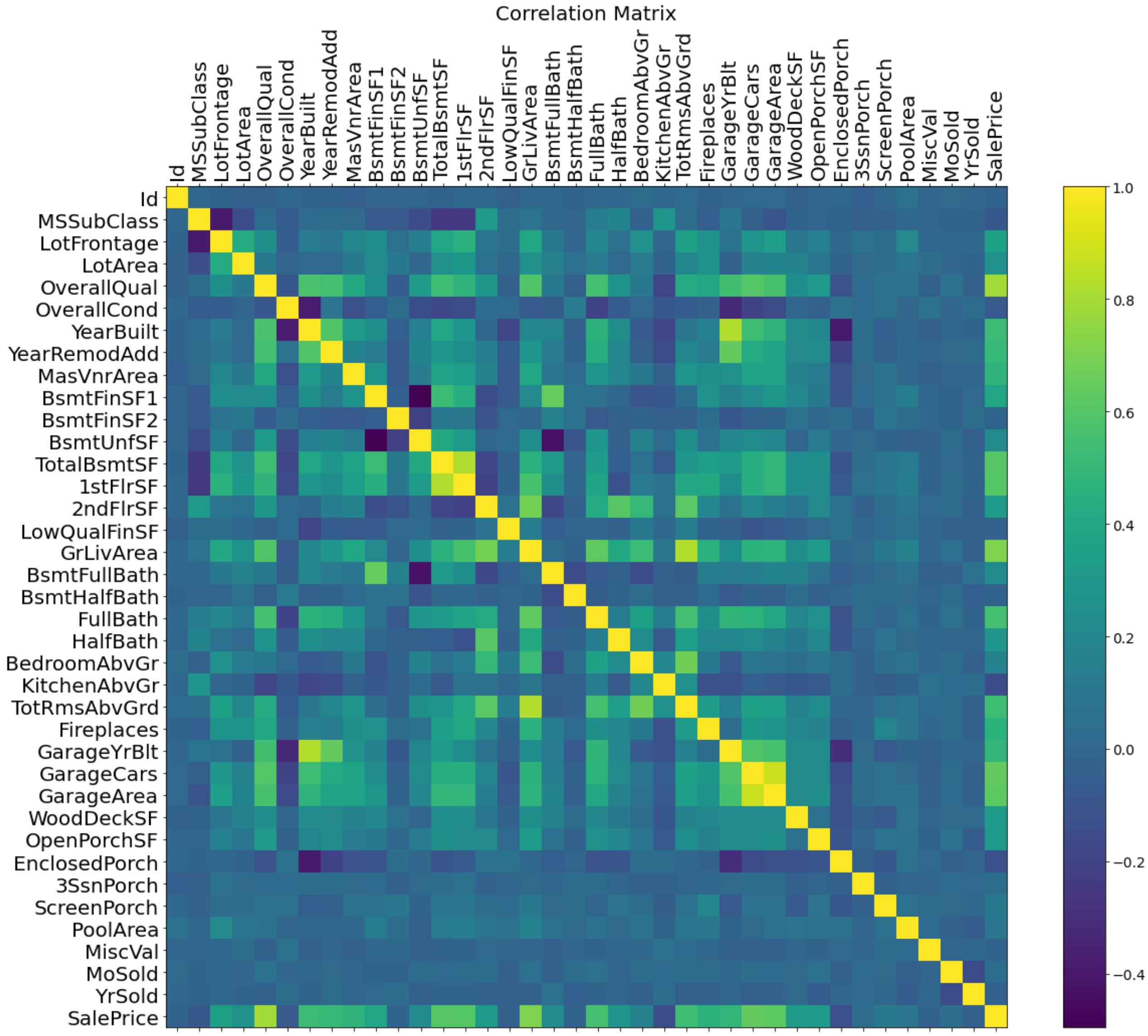
```
top3corr = pd.DataFrame([correlations_abs['OverallQual'], correlations_abs['GrLi

top3corrplot = top3corr.plot(kind='bar')
x_labels = ['OverallQual', 'GrLivArea', 'GarageCars']
top3corrplot.set_title("Top 3 Potential Predictors")
```

```
top3corrplot.set_xlabel("Feature")
top3corrplot.set_ylabel("Correlation Coefficient")
top3corrplot.set_xticklabels(x_labels)
plt.show()
```



In [106…

```
# https://stackoverflow.com/questions/29432629/plot-correlation-matrix-using-pan
# plotting correlation coefficiant via pandas, seeing the relationships between
f = plt.figure(figsize=(20, 15))

plt.matshow(df_train.corr(), fignum=f.number)
plt.xticks(range(df_train.select_dtypes(['number']).shape[1]), df_train.select_d
plt.yticks(range(df_train.select_dtypes(['number']).shape[1]), df_train.select_d

cb = plt.colorbar()
cb.ax.tick_params(labelsize=14)
plt.title('Correlation Matrix', fontsize=20)
# Look only at the bottom row of SalePrice to see the relationship between the o
plt.show()
```



Engage in feature creation by splitting, merging, or otherwise generating a new predictor.

In [107…

```
df_train = df_train.drop(['PoolQC','MiscFeature','Alley','Fence'],axis = 1)
df_train
```

Out[107…

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	LotShape	LandContour	Utilit
0	1	60	RL	65.0	8450	Pave	Reg	Lvl	AllP
1	2	20	RL	80.0	9600	Pave	Reg	Lvl	AllP
2	3	60	RL	68.0	11250	Pave	IR1	Lvl	AllP
3	4	70	RL	60.0	9550	Pave	IR1	Lvl	AllP
4	5	60	RL	84.0	14260	Pave	IR1	Lvl	AllP
...	...	...	...	...	...	...	...	...	...
1455	1456	60	RL	62.0	7917	Pave	Reg	Lvl	AllP
1456	1457	20	RL	85.0	13175	Pave	Reg	Lvl	AllP

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	LotShape	LandContour	Utilit
1457	1458	70	RL	66.0	9042	Pave	Reg	Lvl	AllI
1458	1459	20	RL	68.0	9717	Pave	Reg	Lvl	AllI
1459	1460	20	RL	75.0	9937	Pave	Reg	Lvl	AllI

1460 rows × 77 columns

In [109…

```
related_garage_features = ['GarageYrBlt','GarageCond','GarageQual','GarageType',
df_train2 = df_train.copy()
for cols in related_garage_features:
    if df_train2[cols].dtype == object:
        df_train2.loc[df_train2[cols].isnull(), cols] = 'None'
    else:
        df_train2.loc[df_train2[cols].isnull(), cols] = 0

df_train2
```

Out[109…

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	LotShape	LandContour	Utilit
0	1	60	RL	65.0	8450	Pave	Reg	Lvl	AllI
1	2	20	RL	80.0	9600	Pave	Reg	Lvl	AllI
2	3	60	RL	68.0	11250	Pave	IR1	Lvl	AllI
3	4	70	RL	60.0	9550	Pave	IR1	Lvl	AllI
4	5	60	RL	84.0	14260	Pave	IR1	Lvl	AllI
...	...	...	...	...	...	...	...	...	
1455	1456	60	RL	62.0	7917	Pave	Reg	Lvl	AllI
1456	1457	20	RL	85.0	13175	Pave	Reg	Lvl	AllI
1457	1458	70	RL	66.0	9042	Pave	Reg	Lvl	AllI
1458	1459	20	RL	68.0	9717	Pave	Reg	Lvl	AllI
1459	1460	20	RL	75.0	9937	Pave	Reg	Lvl	AllI

1460 rows × 77 columns

In [114…

```
# OverallQual and OverallCond both had high correlation with eachother as well a
df_train3 = df_train.copy()
df_train3['QualityAndConditionFactor'] = df_train3['OverallQual']*df_train3['Ove
df_train3['LivingLotAreaRatio'] = df_train3.GrLivArea / df_train3.LotArea
df_train3
```

Out[114…

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour
0	1	60	RL	65.0	8450	Pave	NaN	Reg	Lv
1	2	20	RL	80.0	9600	Pave	NaN	Reg	Lv
2	3	60	RL	68.0	11250	Pave	NaN	IR1	Lv

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour
3	4	70	RL	60.0	9550	Pave	NaN	IR1	Lv
4	5	60	RL	84.0	14260	Pave	NaN	IR1	Lv
...	...	...	...	...	...	...	...	...	..
1455	1456	60	RL	62.0	7917	Pave	NaN	Reg	Lv
1456	1457	20	RL	85.0	13175	Pave	NaN	Reg	Lv
1457	1458	70	RL	66.0	9042	Pave	NaN	Reg	Lv
1458	1459	20	RL	68.0	9717	Pave	NaN	Reg	Lv
1459	1460	20	RL	75.0	9937	Pave	NaN	Reg	Lv

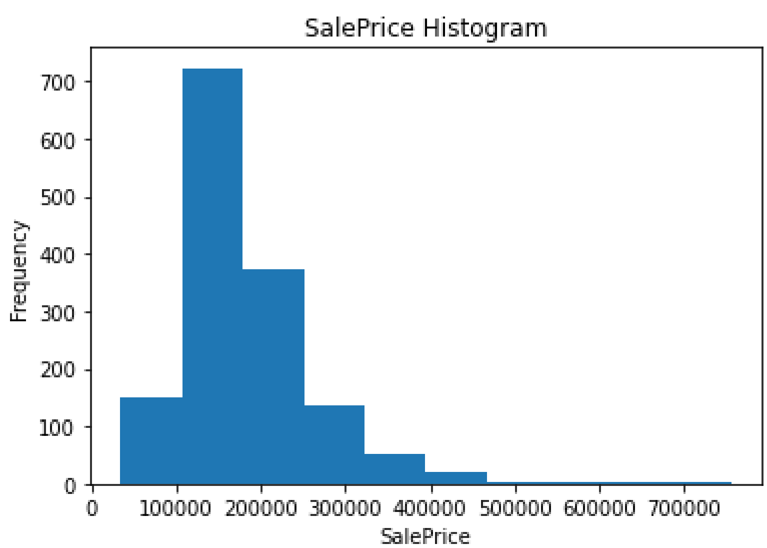
1460 rows × 83 columns

Using the dependent variable, or some other continuously valued variable, perform both min-max and standard scaling in Python.

In [60]:

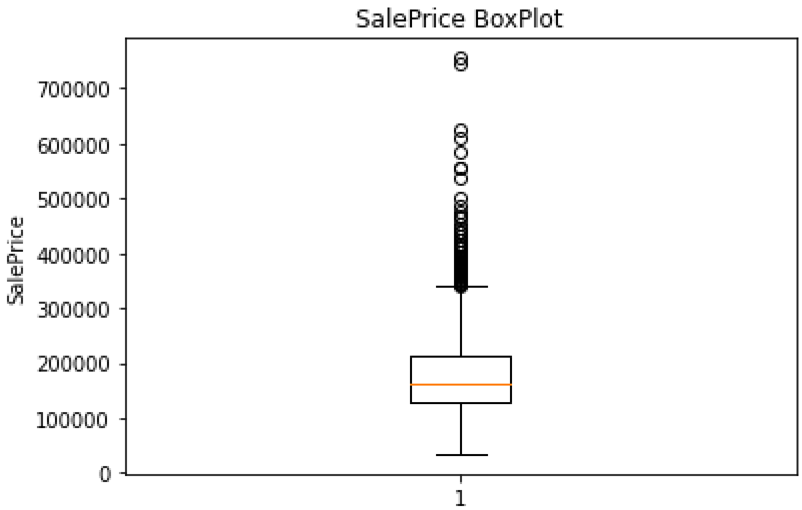
```
# Plotting the SalePrice without any transformations
plt.hist(dependentVariable)
plt.xlabel('SalePrice')
plt.ylabel('Frequency')
print("Skewness: %f" % dependentVariable.skew())
plt.title('SalePrice Histogram')
plt.show()

plt.boxplot(dependentVariable)
plt.ylabel('SalePrice')
plt.title('SalePrice BoxPlot')
```

```
Skewness: 1.882876
```



Out[60]:

```
Text(0.5, 1.0, 'SalePrice BoxPlot')
```



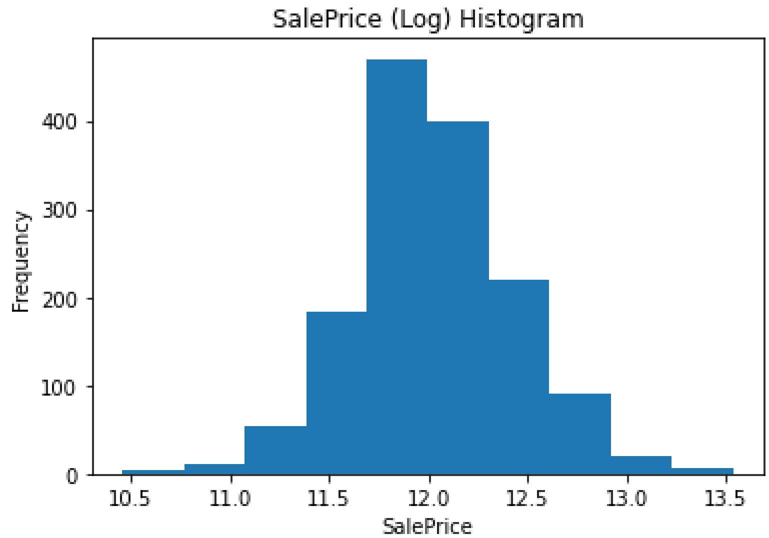
In [117…

```
# General Min max data and general statistics
print("SalePrice Statistics\n")
print("Min house price: ${:,}".format(np.min(dependentVariable)))
print("Median house price ${:,}".format(np.median(dependentVariable)))
print("Max house price: ${:,}".format(np.max(dependentVariable)))
print("Mean house price: ${:,}".format(np.mean(dependentVariable)))
print("Standard deviation of prices: ${:,}".format(np.std(dependentVariable)))
```

```
SalePrice Statistics

Min house price: $34,900
Median house price $163,000.0
Max house price: $755,000
Mean house price: $180,921.19589041095
Standard deviation of prices: $79,415.29188606751
```

In [62]:

```
log_transformed = np.log1p(dependentVariable)
plt.hist(log_transformed)
plt.title('SalePrice (Log) Histogram')
plt.xlabel('SalePrice')
plt.ylabel('Frequency')
plt.show()

print("Skewness: %f" % log_transformed.skew())
```



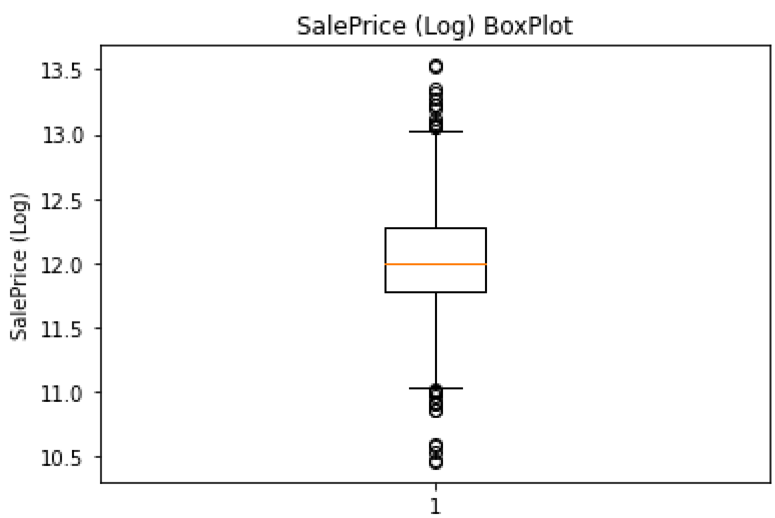

```
Skewness: 0.121347
```

In [64]:

```
plt.boxplot(log_transformed)
plt.ylabel('SalePrice (Log)')
plt.title('SalePrice (Log) BoxPlot')
```

Out[64]:

```
Text(0.5, 1.0, 'SalePrice (Log) BoxPlot')
```



In [57]:

```
quartile1 = log_transformed.quantile(0.25)
quartile3 = log_transformed.quantile(0.75)
IQR2 = quartile3 - quartile1
totalOutliers2 = ((log_transformed < (quartile1 - 1.5 * IQR2)) | (log_transforme
print("IQR value: {}\nTotal amount of outliers within SalePrice: {}".format(IQR,
```

```
IQR value: 0.49863092538878107
Total amount of outliers within SalePrice: 28
```

In [90]:

```
scaleMinMax = pd.get_dummies(df_train.drop(["SalePrice"],axis=1))
scalerMinMax = MinMaxScaler(feature_range=(0, 1))
scaleMinMax[scaleMinMax.columns] = scalerMinMax.fit_transform(scaleMinMax[scaleM
scaleMinMax
```

Out[90]:

	Id	MSSubClass	LotFrontage	LotArea	OverallQual	OverallCond	YearBuilt	YearRe

	Id	MSSubClass	LotFrontage	LotArea	OverallQual	OverallCond	YearBuilt	YearRe
0	0.000000	0.235294	0.150685	0.033420	0.666667	0.500	0.949275	(
1	0.000685	0.000000	0.202055	0.038795	0.555556	0.875	0.753623	(
2	0.001371	0.235294	0.160959	0.046507	0.666667	0.500	0.934783	(
3	0.002056	0.294118	0.133562	0.038561	0.666667	0.500	0.311594	(
4	0.002742	0.235294	0.215753	0.060576	0.777778	0.500	0.927536	(
...	...	...	...	...	...	...	...	
1455	0.997258	0.235294	0.140411	0.030929	0.555556	0.500	0.920290	(
1456	0.997944	0.000000	0.219178	0.055505	0.555556	0.625	0.768116	(
1457	0.998629	0.294118	0.154110	0.036187	0.666667	1.000	0.500000	(
1458	0.999315	0.000000	0.160959	0.039342	0.444444	0.625	0.565217	
1459	1.000000	0.000000	0.184932	0.040370	0.444444	0.625	0.673913	(

1460 rows × 289 columns

In [87]:

```
scaleStandard = pd.get_dummies(df_train.drop(["SalePrice"],axis=1))
scaler=StandardScaler()
scaleStandard[scaleStandard.columns] = scaler.fit_transform(scaleStandard[scaleS
scaleStandard
```

Out[87]:

	Id	MSSubClass	LotFrontage	LotArea	OverallQual	OverallCond	YearBuilt	Yea
0	-1.730865	0.073375	-0.208034	-0.207142	0.651479	-0.517200	1.050994	
1	-1.728492	-0.872563	0.409895	-0.091886	-0.071836	2.179628	0.156734	
2	-1.726120	0.073375	-0.084449	0.073480	0.651479	-0.517200	0.984752	
3	-1.723747	0.309859	-0.414011	-0.096897	0.651479	-0.517200	-1.863632	
4	-1.721374	0.073375	0.574676	0.375148	1.374795	-0.517200	0.951632	
...	...	...	...	...	...	...	...	
1455	1.721374	0.073375	-0.331620	-0.260560	-0.071836	-0.517200	0.918511	
1456	1.723747	-0.872563	0.615871	0.266407	-0.071836	0.381743	0.222975	
1457	1.726120	0.309859	-0.166839	-0.147810	0.651479	3.078570	-1.002492	
1458	1.728492	-0.872563	-0.084449	-0.080160	-0.795151	0.381743	-0.704406	
1459	1.730865	-0.872563	0.203918	-0.058112	-0.795151	0.381743	-0.207594	

1460 rows × 289 columns