Summary of EDA of the Ames Housing Dataset

Background

This week, we explored Kaggle's "House Prices: Advanced Regression Techniques" competition to sharpen our Exploratory Data Analysis (EDA) skills. Our focus is on house prices in Ames, Iowa, represented by the dependent variable SalePrice.

Management/Research Question

The research question is: "What factors influence house prices in Ames, Iowa?" Understanding these factors can help buyers, sellers, and real estate professionals make informed decisions.

Requirements and Methods

We conducted EDA on the dataset as follows:

1. **Descriptive Statistics and Visualizations**: Analyzed the marginal distribution of SalePrice.
2. **Missing Data and Outliers**: Investigated missing values and outliers.
3. **Potential Predictors**: Examined three potential predictors of SalePrice with graphs and statistics.
4. **Feature Creation**: Generated a new predictor through feature engineering.
5. **Scaling Methods**: Applied min-max and standard scaling to the dependent variable.

Descriptive Statistics and Visualizations

The dataset contains 1,460 observations and 81 variables. SalePrice ranges from $34,900 to $755,000, with a mean of $180,921. The distribution is right-skewed, suggesting a log transformation for normalization.

Missing Data and Outliers

Nineteen variables have missing values, with Alley, PoolQC, Fence, and MiscFeature having over 50% missingness and subsequently removed. Boxplots identified several outliers, but these values were retained as they were realistic.

Potential Predictors

We investigated the relationships between SalePrice and three predictors:

1. **GrLivArea**: Shows a strong positive correlation with SalePrice, indicating that larger living areas generally command higher prices.
2. **OverallQual**: Another strong predictor; higher quality ratings correlate with higher prices.
3. **GarageCars**: Homes with more garage spaces tend to have higher sale prices.

The correlations are as follows:

- GrLivArea: r = 0.71 r = 0.71
- OverallQual: r = 0.79 r = 0.79
- GarageCars: r = 0.62 r = 0.62

Feature Creation

We created the feature TotalSF, which is the sum of GrLivArea and TotalBsmtSF. This new feature showed a strong association with SalePrice (r = 0.81 r = 0.81).

Scaling Methods

Both min-max and standard scaling were applied to SalePrice:

- **Min-Max Scaling**: Adjusted SalePrice to a 0-1 range.
- **Standard Scaling**: Centered SalePrice around the mean with a standard deviation of 1.

Results and Insights

1. **GrLivArea** and SalePrice have a strong correlation, but two larger homes deviate, indicating potential outliers.
2. **OverallQual** shows a clear trend where higher quality ratings significantly increase the median SalePrice, from $50,150 for quality 1 to $432,390 for quality 10.
3. **GarageCars**: As the number of garage spaces increases, both median and mean SalePrice rise, with homes having 3 garage spaces showing the highest median SalePrice of $295,000.

Conclusion

Our EDA provides valuable insights into the factors influencing house prices in Ames, Iowa. The relationship between GrLivArea, OverallQual, and GarageCars with SalePrice highlights the importance of these features. Feature engineering and scaling techniques further refined our understanding, paving the way for more advanced modeling.

In summary, this EDA revealed insights about factors that may influence the home prices in Ames, Iowa. Our parsimonious regression model that used three predictor variables (TotalSF, YrSinceRemod, GarageCar) accounted for roughly 75% of the variation in Sale Price. Further research and a more sophisticated framework is needed to explain and elucidate the remaining 25% of variation. This baseline regression model can benefit homebuyers and sellers aiming to better understand fair market prices for homes in Ames, Iowa. The lack of ability to study the distribution of the error terms is a limitation of our approach. Other avenues to pursue would include using principal components or clustering techniques on the 80 variables to leverage the dimensionality of the dataset.

Appendix:

Module 1 Assignment 1: House Prices: Advanced Regression Techniques EDA (Kaggle)

Sachin Sharma

MSDS 422

2024/06/23

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import MinMaxScaler
import scipy.stats as st
import seaborn as sns

import warnings

# Ignore all FutureWarnings
warnings.filterwarnings("ignore", category=FutureWarning)
```

Load Train Data

```
train_df = pd.read_csv("train.csv")
train_df.head()
```

```
   Id  MSSubClass MSZoning  LotFrontage  LotArea Street Alley LotShape
\
0   1          60       RL         65.0     8450   Pave   NaN      Reg

1   2          20       RL         80.0     9600   Pave   NaN      Reg

2   3          60       RL         68.0    11250   Pave   NaN      IR1

3   4          70       RL         60.0     9550   Pave   NaN      IR1

4   5          60       RL         84.0    14260   Pave   NaN      IR1


  LandContour Utilities  ... PoolArea PoolQC Fence MiscFeature MiscVal
MoSold  \
0         Lvl    AllPub  ...        0    NaN   NaN         NaN       0
2
1         Lvl    AllPub  ...        0    NaN   NaN         NaN       0
5
2         Lvl    AllPub  ...        0    NaN   NaN         NaN       0
9
3         Lvl    AllPub  ...        0    NaN   NaN         NaN       0
```

```
2
4          Lvl     AllPub  ...        0     NaN    NaN          NaN        0
12

  YrSold  SaleType  SaleCondition  SalePrice
0   2008        WD         Normal     208500
1   2007        WD         Normal     181500
2   2008        WD         Normal     223500
3   2006        WD        Abnorml     140000
4   2008        WD         Normal     250000

[5 rows x 81 columns]
```

```
print('Columns: \n'+", ".join(train_df.columns.tolist()))
```

```
Columns:
Id, MSSubClass, MSZoning, LotFrontage, LotArea, Street, Alley,
LotShape, LandContour, Utilities, LotConfig, LandSlope, Neighborhood,
Condition1, Condition2, BldgType, HouseStyle, OverallQual,
OverallCond, YearBuilt, YearRemodAdd, RoofStyle, RoofMatl,
Exterior1st, Exterior2nd, MasVnrType, MasVnrArea, ExterQual,
ExterCond, Foundation, BsmtQual, BsmtCond, BsmtExposure, BsmtFinType1,
BsmtFinSF1, BsmtFinType2, BsmtFinSF2, BsmtUnfSF, TotalBsmtSF, Heating,
HeatingQC, CentralAir, Electrical, 1stFlrSF, 2ndFlrSF, LowQualFinSF,
GrLivArea, BsmtFullBath, BsmtHalfBath, FullBath, HalfBath,
BedroomAbvGr, KitchenAbvGr, KitchenQual, TotRmsAbvGrd, Functional,
Fireplaces, FireplaceQu, GarageType, GarageYrBlt, GarageFinish,
GarageCars, GarageArea, GarageQual, GarageCond, PavedDrive,
WoodDeckSF, OpenPorchSF, EnclosedPorch, 3SsnPorch, ScreenPorch,
PoolArea, PoolQC, Fence, MiscFeature, MiscVal, MoSold, YrSold,
SaleType, SaleCondition, SalePrice
```

```
train_df.shape
```

```
(1460, 81)
```

EDA

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

The dependent variable is SalePrice. There are 1460 entires in the training data

```
train_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1460 entries, 0 to 1459
```

```
Data columns (total 81 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Id             1460 non-null   int64  
 1   MSSubClass     1460 non-null   int64  
 2   MSZoning       1460 non-null   object 
 3   LotFrontage    1201 non-null   float64
 4   LotArea        1460 non-null   int64  
 5   Street         1460 non-null   object 
 6   Alley          91 non-null     object 
 7   LotShape       1460 non-null   object 
 8   LandContour    1460 non-null   object 
 9   Utilities      1460 non-null   object 
 10  LotConfig      1460 non-null   object 
 11  LandSlope      1460 non-null   object 
 12  Neighborhood   1460 non-null   object 
 13  Condition1     1460 non-null   object 
 14  Condition2     1460 non-null   object 
 15  BldgType       1460 non-null   object 
 16  HouseStyle     1460 non-null   object 
 17  OverallQual    1460 non-null   int64  
 18  OverallCond    1460 non-null   int64  
 19  YearBuilt      1460 non-null   int64  
 20  YearRemodAdd   1460 non-null   int64  
 21  RoofStyle      1460 non-null   object 
 22  RoofMatl       1460 non-null   object 
 23  Exterior1st    1460 non-null   object 
 24  Exterior2nd    1460 non-null   object 
 25  MasVnrType     588 non-null    object 
 26  MasVnrArea     1452 non-null   float64
 27  ExterQual      1460 non-null   object 
 28  ExterCond      1460 non-null   object 
 29  Foundation     1460 non-null   object 
 30  BsmtQual       1423 non-null   object 
 31  BsmtCond       1423 non-null   object 
 32  BsmtExposure   1422 non-null   object 
 33  BsmtFinType1   1423 non-null   object 
 34  BsmtFinSF1     1460 non-null   int64  
 35  BsmtFinType2   1422 non-null   object 
 36  BsmtFinSF2     1460 non-null   int64  
 37  BsmtUnfSF      1460 non-null   int64  
 38  TotalBsmtSF    1460 non-null   int64  
 39  Heating        1460 non-null   object 
 40  HeatingQC      1460 non-null   object 
 41  CentralAir     1460 non-null   object 
 42  Electrical     1459 non-null   object 
 43  1stFlrSF       1460 non-null   int64  
 44  2ndFlrSF       1460 non-null   int64  
 45  LowQualFinSF   1460 non-null   int64  
```

```
 46  GrLivArea      1460 non-null   int64
 47  BsmtFullBath   1460 non-null   int64
 48  BsmtHalfBath   1460 non-null   int64
 49  FullBath       1460 non-null   int64
 50  HalfBath       1460 non-null   int64
 51  BedroomAbvGr   1460 non-null   int64
 52  KitchenAbvGr   1460 non-null   int64
 53  KitchenQual    1460 non-null   object
 54  TotRmsAbvGrd   1460 non-null   int64
 55  Functional     1460 non-null   object
 56  Fireplaces     1460 non-null   int64
 57  FireplaceQu    770 non-null    object
 58  GarageType     1379 non-null   object
 59  GarageYrBlt    1379 non-null   float64
 60  GarageFinish   1379 non-null   object
 61  GarageCars     1460 non-null   int64
 62  GarageArea     1460 non-null   int64
 63  GarageQual     1379 non-null   object
 64  GarageCond     1379 non-null   object
 65  PavedDrive     1460 non-null   object
 66  WoodDeckSF     1460 non-null   int64
 67  OpenPorchSF    1460 non-null   int64
 68  EnclosedPorch  1460 non-null   int64
 69  3SsnPorch      1460 non-null   int64
 70  ScreenPorch    1460 non-null   int64
 71  PoolArea       1460 non-null   int64
 72  PoolQC         7 non-null      object
 73  Fence          281 non-null    object
 74  MiscFeature    54 non-null     object
 75  MiscVal        1460 non-null   int64
 76  MoSold         1460 non-null   int64
 77  YrSold         1460 non-null   int64
 78  SaleType       1460 non-null   object
 79  SaleCondition  1460 non-null   object
 80  SalePrice      1460 non-null   int64
dtypes: float64(3), int64(35), object(43)
memory usage: 924.0+ KB
```

```
train_df.describe()
```

```
                Id   MSSubClass  LotFrontage        LotArea
OverallQual  \
count  1460.000000  1460.000000  1201.000000    1460.000000
1460.000000
mean    730.500000    56.897260    70.049958   10516.828082
6.099315
std     421.610009    42.300571    24.284752    9981.264932
1.382997
min       1.000000    20.000000    21.000000    1300.000000
1.000000
```

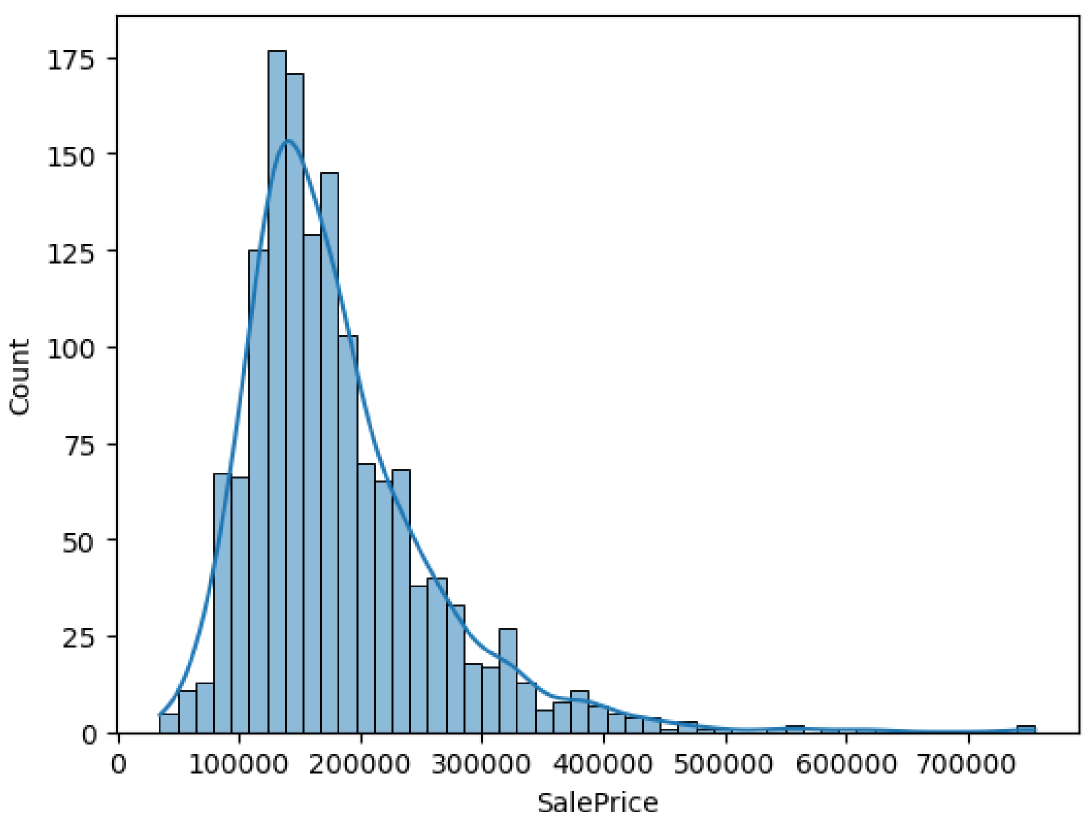
```
25%     365.750000    20.000000    59.000000    7553.500000
5.000000
50%     730.500000    50.000000    69.000000    9478.500000
6.000000
75%    1095.250000    70.000000    80.000000   11601.500000
7.000000
max    1460.000000   190.000000   313.000000  215245.000000
10.000000

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

        WoodDeckSF  OpenPorchSF  EnclosedPorch    3SsnPorch
ScreenPorch  \
count  1460.000000  1460.000000    1460.000000  1460.000000
1460.000000
mean     94.244521    46.660274      21.954110     3.409589
15.060959
std     125.338794    66.256028      61.119149    29.317331
55.757415
min       0.000000     0.000000       0.000000     0.000000
0.000000
25%       0.000000     0.000000       0.000000     0.000000
0.000000
50%       0.000000    25.000000       0.000000     0.000000
0.000000
75%     168.000000    68.000000       0.000000     0.000000
0.000000
max     857.000000   547.000000     552.000000   508.000000
480.000000

          PoolArea      MiscVal       MoSold       YrSold
SalePrice
count  1460.000000   1460.000000  1460.000000  1460.000000
```

```
1460.000000
mean       2.758904     43.489041      6.321918   2007.815753
180921.195890
std       40.177307    496.123024      2.703626      1.328095
79442.502883
min        0.000000      0.000000      1.000000   2006.000000
34900.000000
25%        0.000000      0.000000      5.000000   2007.000000
129975.000000
50%        0.000000      0.000000      6.000000   2008.000000
163000.000000
75%        0.000000      0.000000      8.000000   2009.000000
214000.000000
max      738.000000  15500.000000     12.000000   2010.000000
755000.000000

[8 rows x 38 columns]
```

Distribution of SalePrice:

```
sns.histplot(data=train_df, kde=True, x='SalePrice')
```

```
<Axes: xlabel='SalePrice', ylabel='Count'>
```



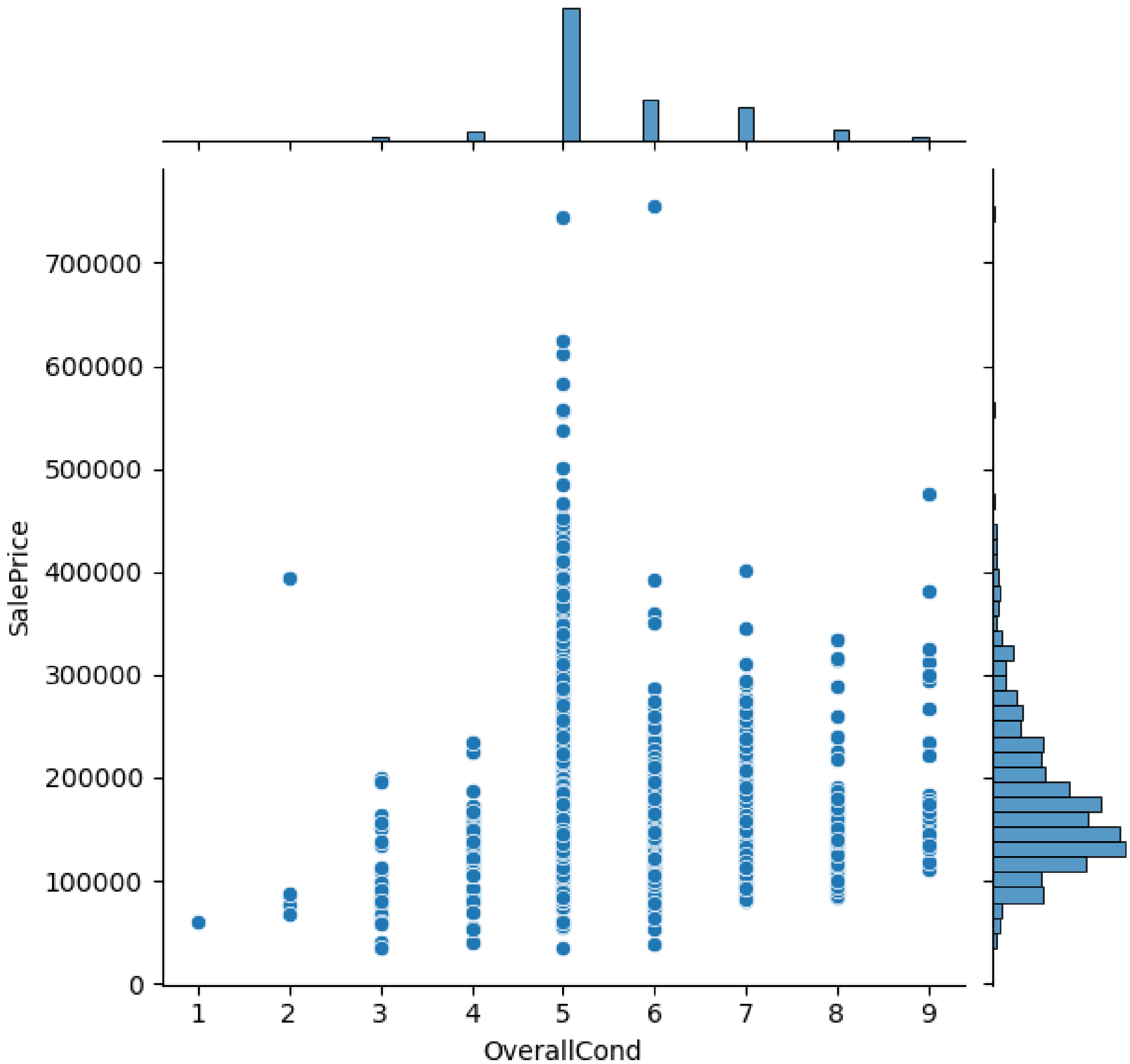
```
train_df.SalePrice.describe()
```

```
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
```

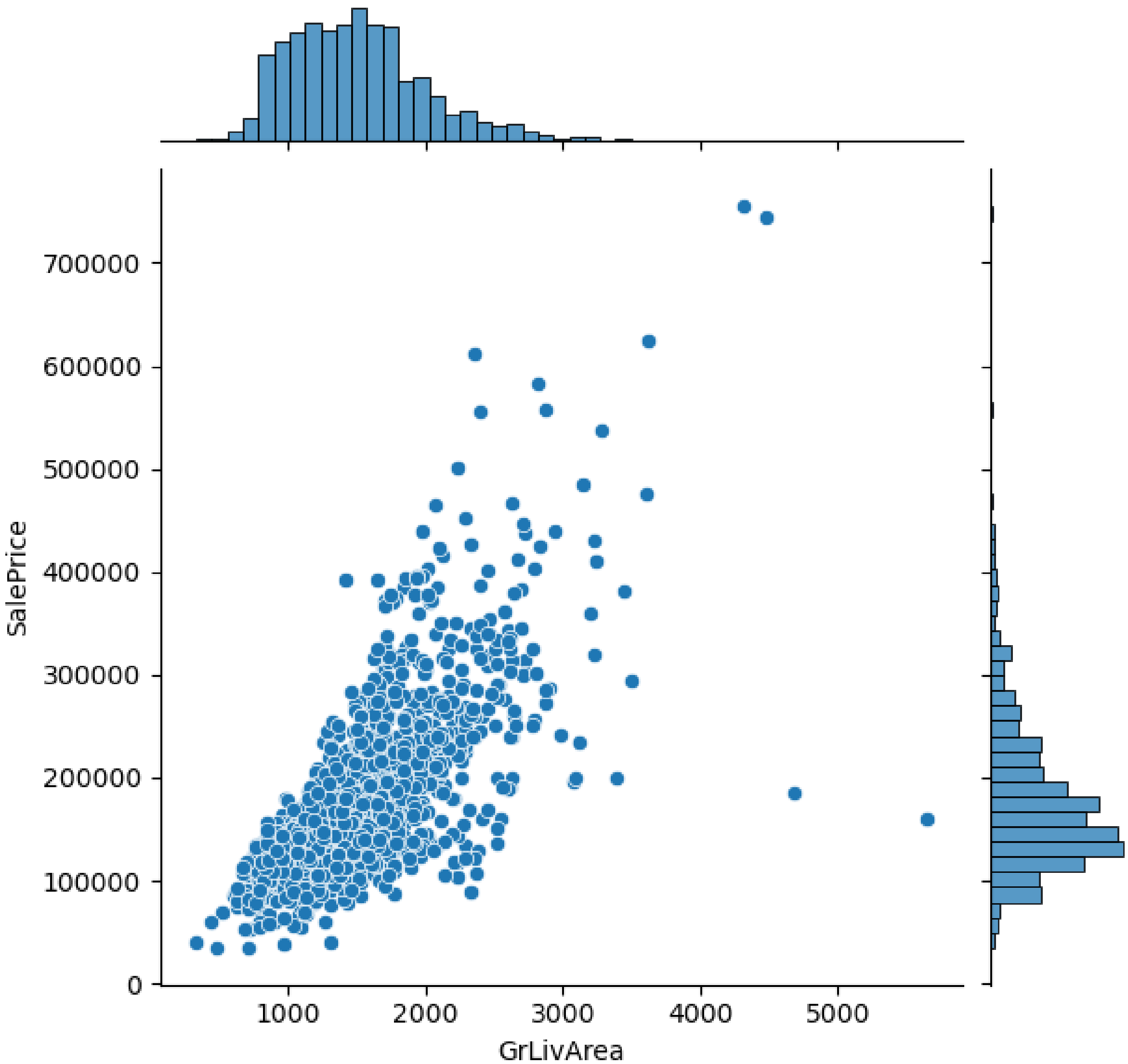
The median SalePrice is \$163000 while the mean is ~\$180921.20.

Joint Plots of SalePrice in respect to other features like OverallCond, GrLivingArea:

```
sns.jointplot(x="OverallCond",
y="SalePrice",edgecolor="white",data=train_df);
sns.jointplot(x="GrLivArea",
y="SalePrice",edgecolor="white",data=train_df);
```



700000
600000
500000
400000
300000
200000
100000
0
SalePrice
1
2
3
4
5
6
7
8
9
OverallCond



Qualitative/categorial and Quantitative/numerical data:

```
# Get a list of qualitative columns in the training set, where column
type is 'object'
qual_cols = train_df.select_dtypes(include='object').columns.tolist()
print('Qualitative Columns: \n' + ", ".join(qual_cols))

# Get a list of quantitative columns in the training set, where column
type is not 'object'
quant_cols = train_df.select_dtypes(exclude='object').columns.tolist()
print('\nQuantitative Columns: \n' + ", ".join(quant_cols))

Qualitative Columns:
MSZoning, Street, Alley, LotShape, LandContour, Utilities, LotConfig,
LandSlope, Neighborhood, Condition1, Condition2, BldgType, HouseStyle,
RoofStyle, RoofMatl, Exterior1st, Exterior2nd, MasVnrType, ExterQual,
```

```
ExterCond, Foundation, BsmtQual, BsmtCond, BsmtExposure, BsmtFinType1,
BsmtFinType2, Heating, HeatingQC, CentralAir, Electrical, KitchenQual,
Functional, FireplaceQu, GarageType, GarageFinish, GarageQual,
GarageCond, PavedDrive, PoolQC, Fence, MiscFeature, SaleType,
SaleCondition

Quantitative Columns:
Id, MSSubClass, LotFrontage, LotArea, OverallQual, OverallCond,
YearBuilt, YearRemodAdd, MasVnrArea, BsmtFinSF1, BsmtFinSF2,
BsmtUnfSF, TotalBsmtSF, 1stFlrSF, 2ndFlrSF, LowQualFinSF, GrLivArea,
BsmtFullBath, BsmtHalfBath, FullBath, HalfBath, BedroomAbvGr,
KitchenAbvGr, TotRmsAbvGrd, Fireplaces, GarageYrBlt, GarageCars,
GarageArea, WoodDeckSF, OpenPorchSF, EnclosedPorch, 3SsnPorch,
ScreenPorch, PoolArea, MiscVal, MoSold, YrSold, SalePrice
```

2. Investigate missing data and outliers.

```python
nullTotals = train_df.isnull().sum().sort_values(ascending = False)
percentageOfNull = (train_df.isnull().sum() /
train_df.isnull().count()).sort_values(ascending = False)
emptyVals = pd.concat([nullTotals, percentageOfNull], axis=1,
keys=['Total Missing Values', 'Percentage Null'])
emptyVals.head(20)
```

	Total Missing Values	Percentage Null
PoolQC	1453	0.995205
MiscFeature	1406	0.963014
Alley	1369	0.937671
Fence	1179	0.807534
MasVnrType	872	0.597260
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
Electrical	1	0.000685
Id	0	0.000000

```python
num_na_features = train_df.isna().sum()
num_na_features = num_na_features[num_na_features > 0]
num_na_features.sort_values(inplace=True)
num_na_features.describe()
```

```
count      19.000000
mean      412.052632
std       551.246917
min         1.000000
25%        37.500000
50%        81.000000
75%       781.000000
max      1453.000000
dtype: float64
```

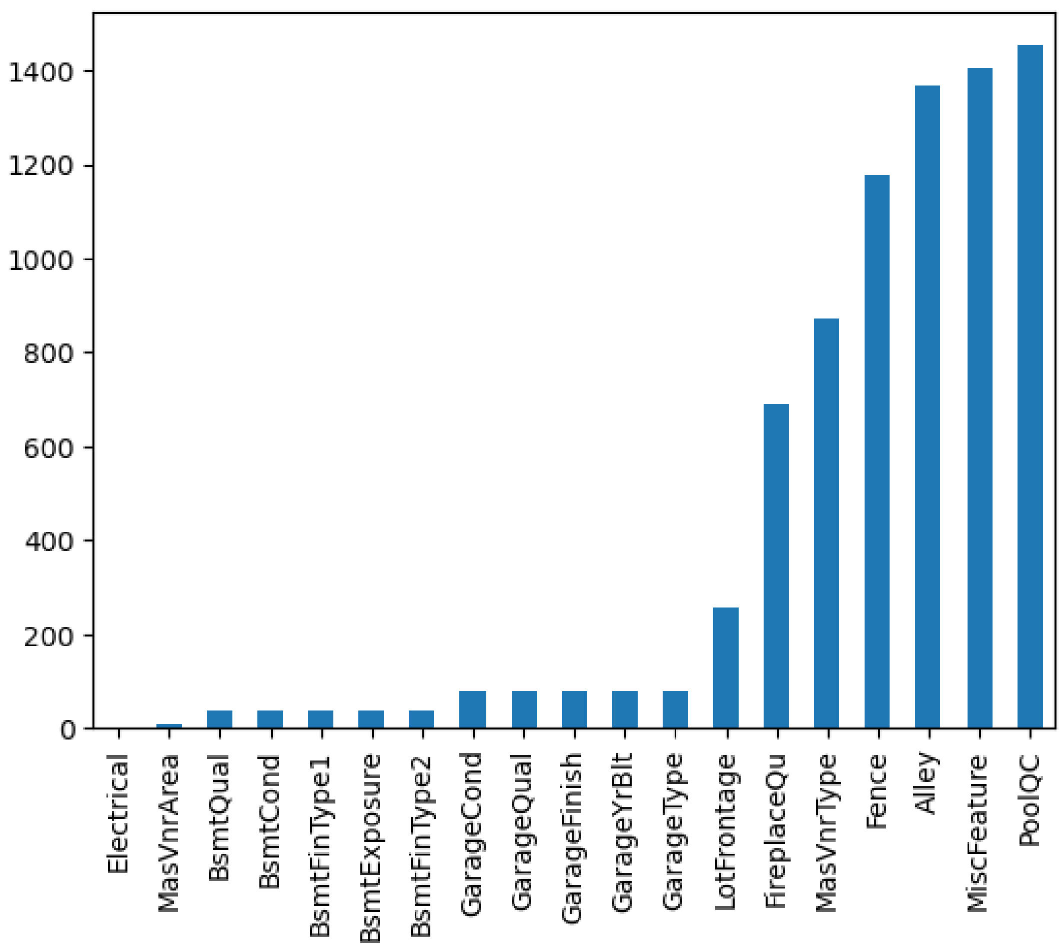
```
num_na_features.size
```

```
19
```

Let's graph a bar chart of null values.

```
num_na_features.plot.bar()
```

```
<Axes: >
```



There are 19 features that contain null values.

Most listings/rows do not contain 'PoolQC', 'MiscFeature', 'Alley', 'Fence', 'FireplaceQC', etc features.

Almost half of the homes do not list 'FireplaceQC' (690)

```
train_df[train_df['Fireplaces'] == 0].FireplaceQu.isnull().sum()
```

```
690
```

All but 7 of the 1453 listings/rows do not have 'PoolQc'

```
len(train_df[train_df['PoolArea'] != 0])
```

```
7
```

1369 null in Alley column

```
train_df['Alley'].isnull().sum()
```

```
1369
```

Let's examine the category outliers with respect to the dependent variable, SalePrice. The total number of outliers within SalePrice is 61.

```
 # get a list of qualitative columns in the training set, where colume
type = 'object'
qual_cols = [f for f in train_df.columns if train_df.dtypes[f] ==
'object']
qual_cols.append('SalePrice') # add the quantitative dependent
varialble (SalePrice) to the list
train_qual_df = train_df[qual_cols] # get a dataframe of qualitative
columns mapped to dependent variable (SalePrice)
train_qual_df = train_qual_df.fillna('NONE') # fill any null cells in
the dataframe with NONE

Q1 = train_df['SalePrice'].quantile(0.25)
Q3 = train_df['SalePrice'].quantile(0.75)
IQR = Q3 - Q1
totalOutliers = ((train_df['SalePrice'] < (Q1 - 1.5 * IQR)) |
(train_df['SalePrice'] > (Q3 + 1.5 * IQR))).sum()
print("IQR value: {}\nTotal amount of outliers within SalePrice:
{}".format(IQR, totalOutliers))
```

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

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

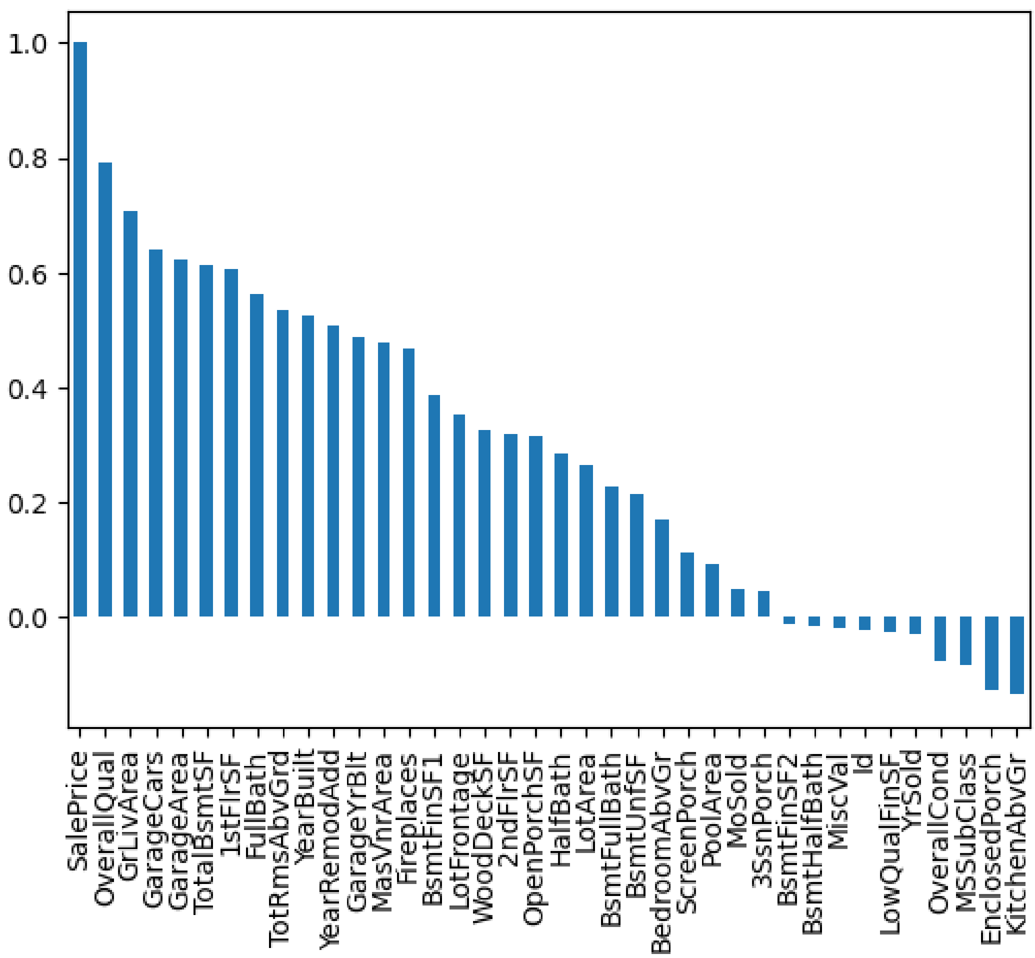
```python
# Calculate correlation matrix for numeric columns only
corr_matrix = train_df.corr(method='pearson', min_periods=30,
numeric_only=True)
corr_matrix_sorted =
corr_matrix['SalePrice'].sort_values(ascending=False)
print(corr_matrix_sorted)
```

```
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
```

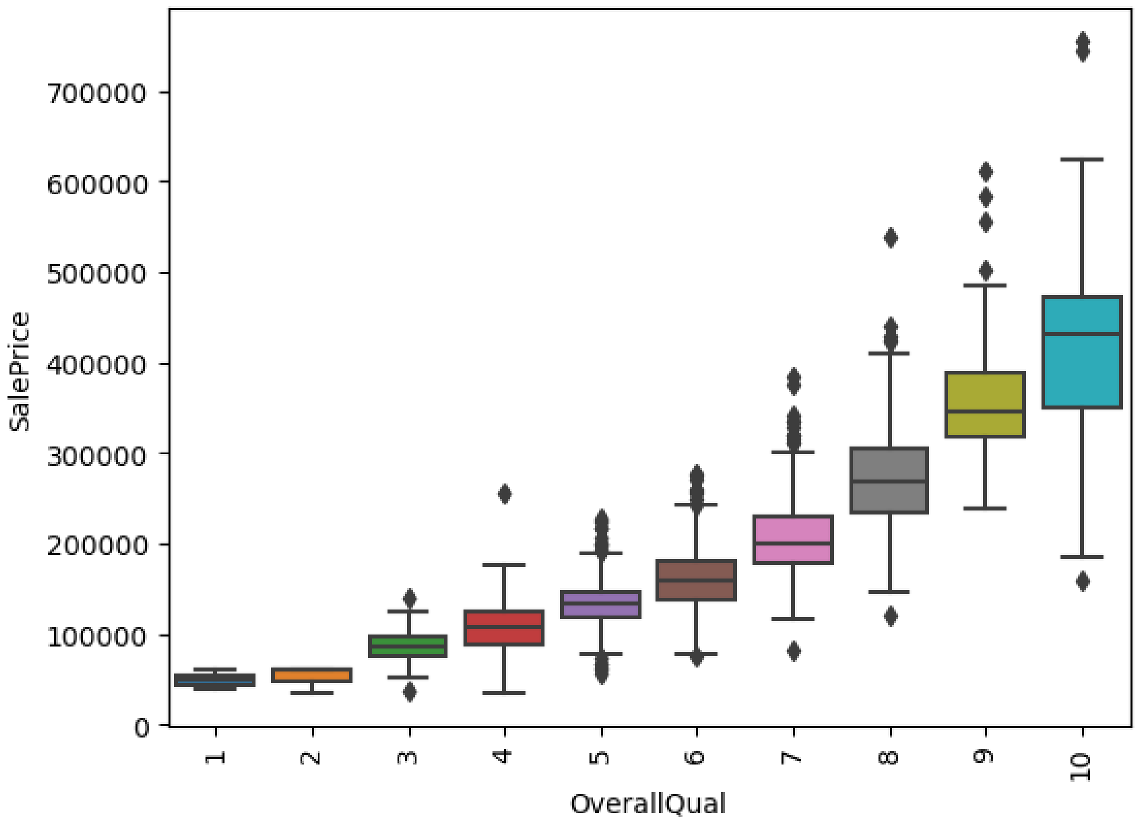
Let's create a bar chart of quantitative correlations.

```
corr_matrix_sorted.plot(kind='bar')

<Axes: >
```



Predictor 1: 'OverallQual'

```
plot = sns.boxplot(data=train_df, x='OverallQual', y='SalePrice')
plot.set_xticklabels(plot.get_xticklabels(), rotation=90)
plt.show()
```



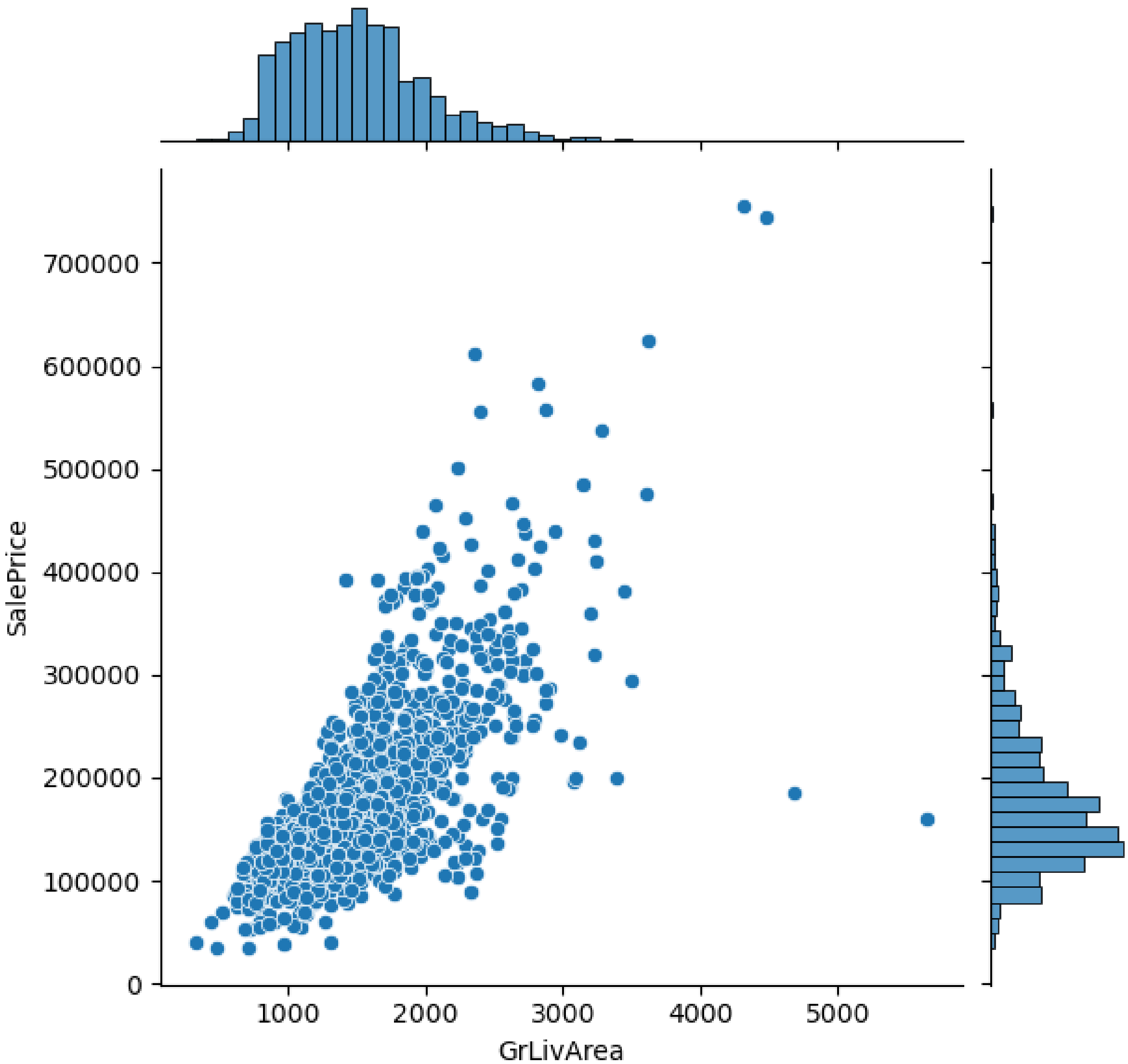

```
train_df.groupby(['OverallQual']).SalePrice.median().sort_values(ascen
ding=False)
```

```
OverallQual
10    432390.0
9     345000.0
8     269750.0
7     200141.0
6     160000.0
5     133000.0
4     108000.0
3      86250.0
2      60000.0
1      50150.0
Name: SalePrice, dtype: float64
```

The data indicates that as the `OverallQual` rating increases, the median `SalePrice` of homes rises significantly, with top-quality homes (rating 10) having a median price of 432,390 and the lowest quality homes (rating 1) having a median price of 50,150.

Predictor 2: 'GrLivArea'

```
sns.jointplot(x="GrLivArea", y="SalePrice", data=train_df);
```




```
train_df.groupby('GrLivArea').SalePrice.describe()
```

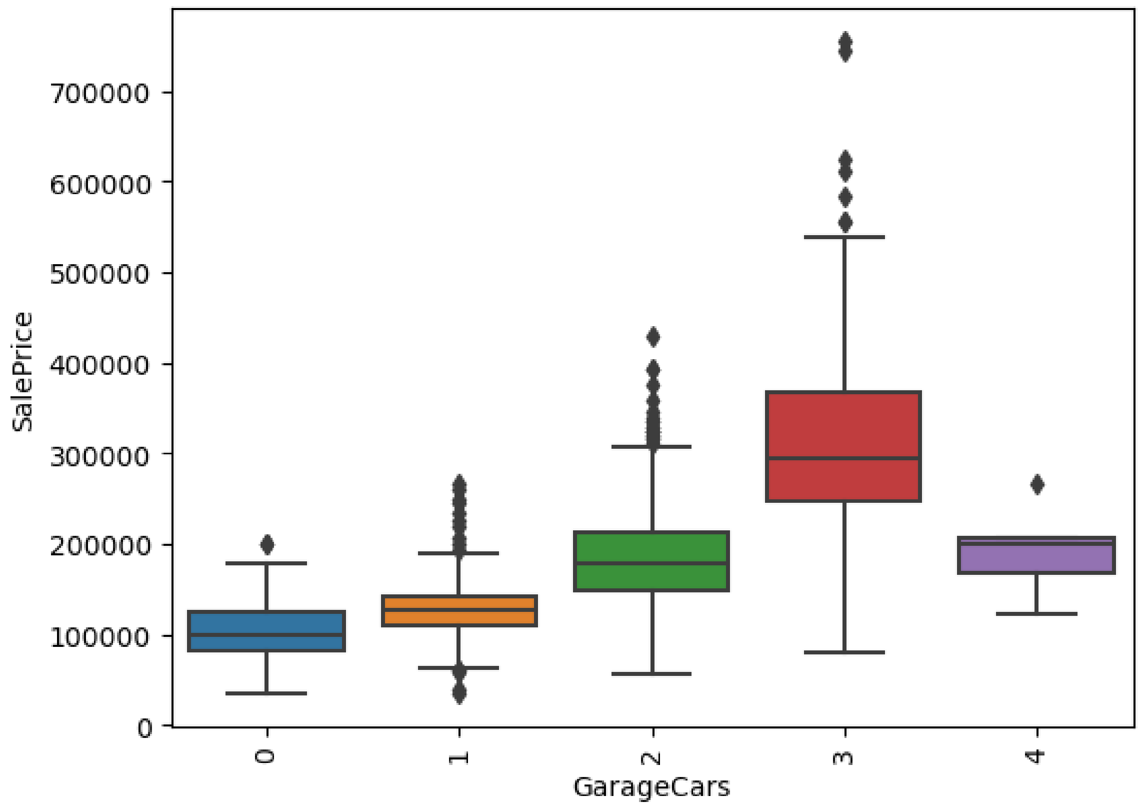
```
            count     mean  std     min      25%      50%
75%  \
GrLivArea

334           1.0  39300.0  NaN  39300.0  39300.0  39300.0
39300.0
438           1.0  60000.0  NaN  60000.0  60000.0  60000.0
60000.0
480           1.0  35311.0  NaN  35311.0  35311.0  35311.0
35311.0
520           1.0  68500.0  NaN  68500.0  68500.0  68500.0
68500.0
605           1.0  86000.0  NaN  86000.0  86000.0  86000.0
```

```
86000.0
...            ...       ...  ...       ...       ...       ...       ..
.
3627           1.0  625000.0  NaN  625000.0  625000.0  625000.0
625000.0
4316           1.0  755000.0  NaN  755000.0  755000.0  755000.0
755000.0
4476           1.0  745000.0  NaN  745000.0  745000.0  745000.0
745000.0
4676           1.0  184750.0  NaN  184750.0  184750.0  184750.0
184750.0
5642           1.0  160000.0  NaN  160000.0  160000.0  160000.0
160000.0

                max
GrLivArea
334         39300.0
438         60000.0
480         35311.0
520         68500.0
605         86000.0
...             ...
3627       625000.0
4316       755000.0
4476       745000.0
4676       184750.0
5642       160000.0

[861 rows x 8 columns]
```

Based on the observation that GrLivingArea (above ground living area) correlates well with SalePrice, with the exception of two larger homes that have a SalePrice just over the median or average, it indicates that while overall there is a strong positive relationship between living area and price, some larger homes may be underpriced or not following the general trend.

Predictor 3: 'GarageCars'

```
plot = sns.boxplot(data=train_df, x='GarageCars', y='SalePrice')
plot.set_xticklabels(plot.get_xticklabels(),rotation = 90)
plt.show()
```



The summary statistics show that as the number of GarageCars increases, the median and mean SalePrice generally rise, with homes having 3 GarageCars having the highest median and mean SalePrice among the categories.

```
train_df.groupby('GarageCars').SalePrice.describe()
```

```
            count          mean            std       min       25%
50%  \
GarageCars

0            81.0  103317.283951   32815.023389   34900.0   82500.0
100000.0
1           369.0  128116.688347   30412.386890   35311.0  110000.0
128000.0
2           824.0  183851.663835   51617.144258   55993.0  148000.0
177750.0
3           181.0  309636.121547  106832.925939   81000.0  246578.0
295000.0
4             5.0  192655.800000   52621.839745  123000.0  168000.0
200000.0

                 75%       max
GarageCars
0           124000.0  200500.0
1           142000.0  266500.0
```

```
2            213000.0  430000.0
3            367294.0  755000.0
4            206300.0  265979.0
```

The median SalePrice increases as the number of GarageCars increases. Homes with 3 GarageCars have the highest median SalePrice of 295,000, followed by those with 4 GarageCars at 200,000. Homes with 2 GarageCars have a median SalePrice of 177,750, while those with 1 GarageCar have a median SalePrice of 128,000. Homes without a garage (0 GarageCars) have the lowest median SalePrice at 100,000

```
train_df.groupby(['GarageCars']).SalePrice.median().sort_values(ascend
ing=False)
```

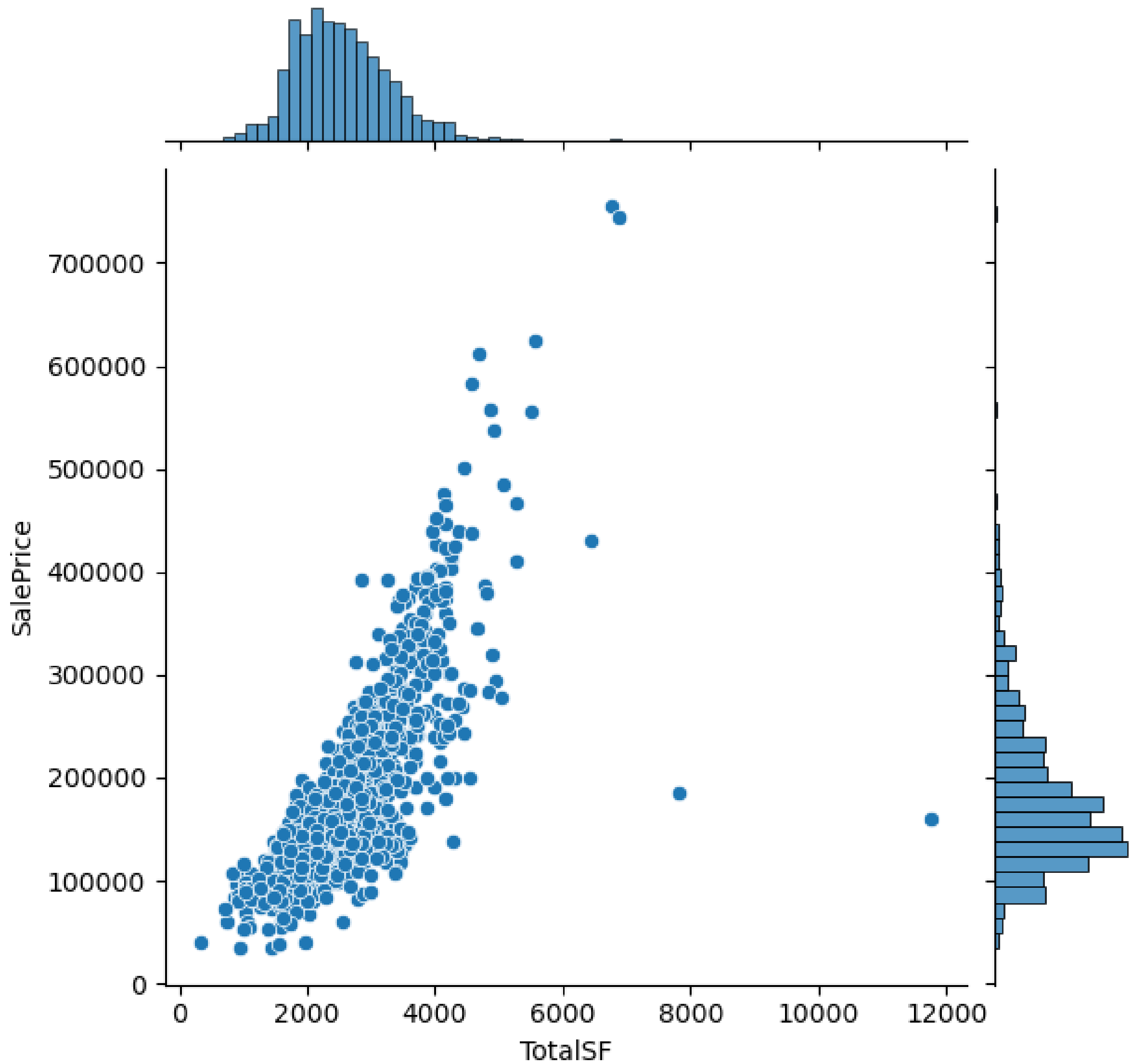
```
GarageCars
3    295000.0
4    200000.0
2    177750.0
1    128000.0
0    100000.0
Name: SalePrice, dtype: float64
```

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

New features may enable us to create more accurate prediction models for home sale prices. Let's create new attribute 'TotalSF' which is TotalBsmtSF, 1stFlrSF, and 2ndFlrSF combined.

```
train_df['TotalSF'] = train_df['TotalBsmtSF'] + train_df['1stFlrSF'] +
train_df['2ndFlrSF']
sns.jointplot(x="TotalSF", y="SalePrice", data=train_df)
```

```
<seaborn.axisgrid.JointGrid at 0x152a28cd0>
```




```
train_df_test = train_df[['TotalBsmtSF', '1stFlrSF', '2ndFlrSF',
'TotalSF', 'GrLivArea', 'SalePrice']].copy()
train_df_test
```

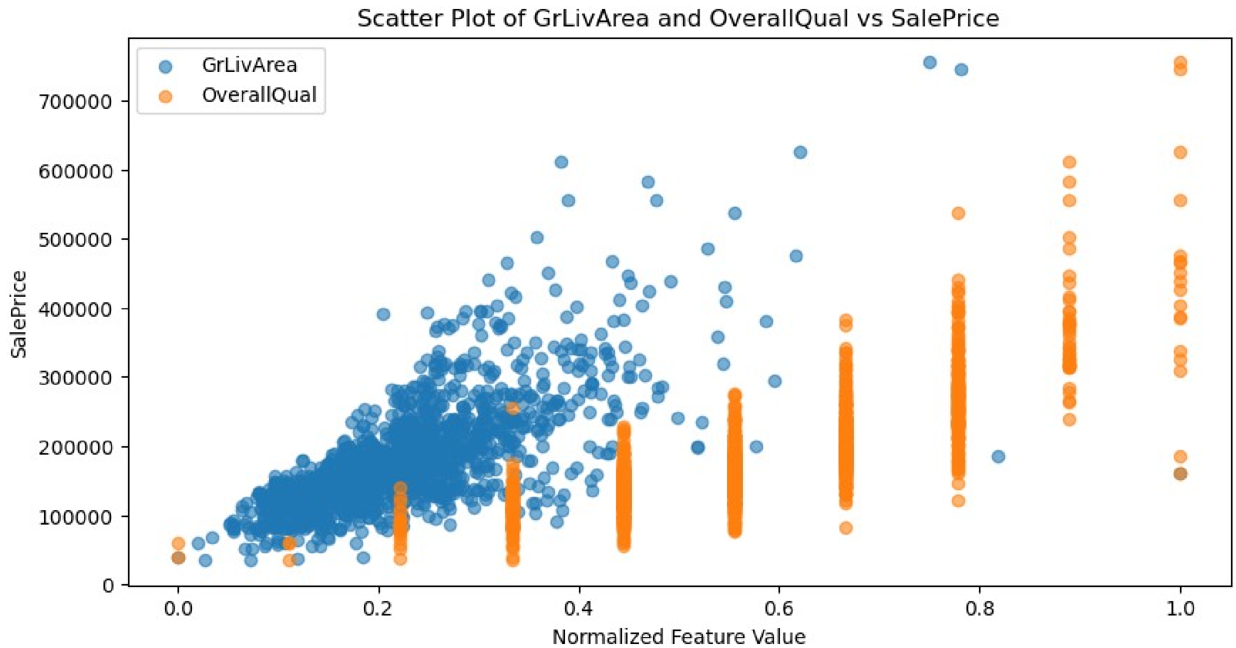
	TotalBsmtSF	1stFlrSF	2ndFlrSF	TotalSF	GrLivArea	SalePrice
0	856	856	854	2566	1710	208500
1	1262	1262	0	2524	1262	181500
2	920	920	866	2706	1786	223500
3	756	961	756	2473	1717	140000
4	1145	1145	1053	3343	2198	250000
...	...	...	...	...	...	...
1455	953	953	694	2600	1647	175000
1456	1542	2073	0	3615	2073	210000
1457	1152	1188	1152	3492	2340	266500
1458	1078	1078	0	2156	1078	142125

```
1459         1256      1256            0      2512          1256      147500

[1460 rows x 6 columns]
```

5. Using the dependent variable, perform both min-max and standard scaling in Python.

The focus of this section appears to be exclusively on scaling the dependent variable, not the independent variables. MinMax Scaling: comparing GrLivArea and OverallQual in respect to the dependent variable (SalePrice)

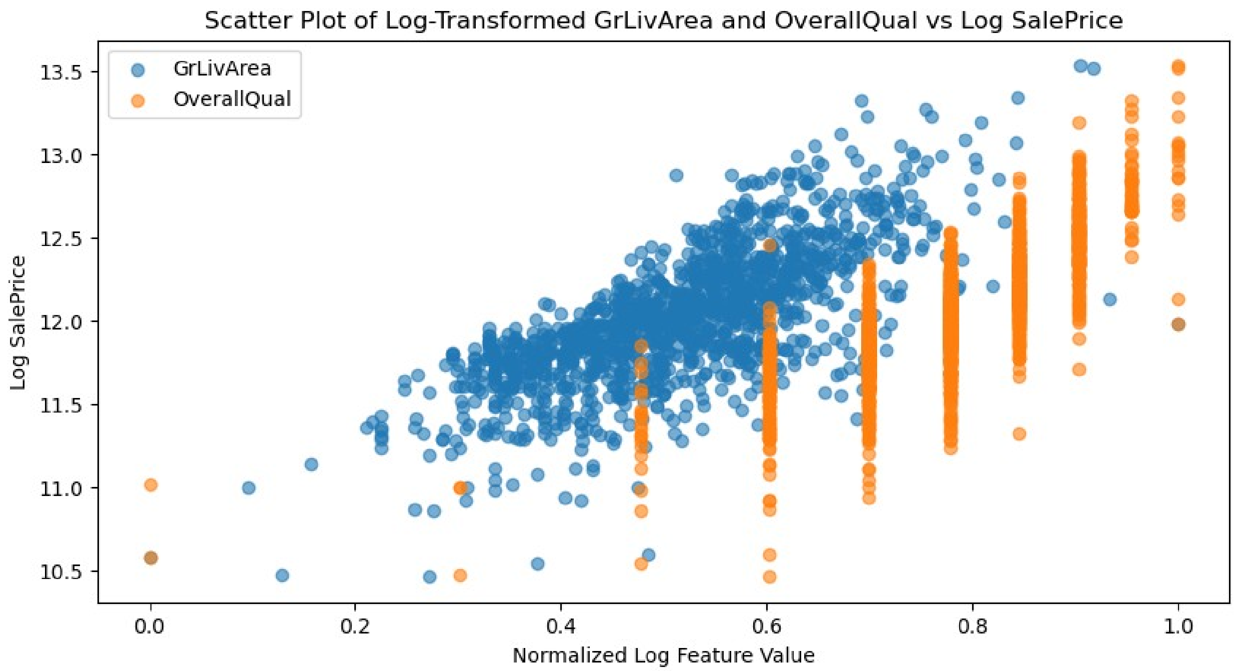
```
features = ['GrLivArea', 'OverallQual']
features_df = train_df[features]

# Scale the features using MinMaxScaler
scaler = MinMaxScaler()
minmax_features = scaler.fit_transform(features_df)
sale_price = train_df["SalePrice"]

# Create a scatter plot
fig, ax = plt.subplots(figsize=(10, 5))

# Scatter plot for GrLivArea
ax.scatter(x=minmax_features[:, 0], y=sale_price, label='GrLivArea',
alpha=0.6)

# Scatter plot for OverallQual
ax.scatter(x=minmax_features[:, 1], y=sale_price, label='OverallQual',
alpha=0.6)

# Add legend
ax.legend()

# Add labels and title
ax.set_xlabel('Normalized Feature Value')
ax.set_ylabel('SalePrice')
ax.set_title('Scatter Plot of GrLivArea and OverallQual vs SalePrice')

# Display the plot
plt.show()
```



MinMax scaling combined with logarithmic transformation helps align the distribution of TotalSF more closely with OverallQual's general distribution, thereby reducing the impact of outliers.

```
features = ['GrLivArea', 'OverallQual']
features_df = train_df[features]

# Log-transform the features
features_df_log = np.log(features_df)

# Scale the log-transformed features using MinMaxScaler
scaler = MinMaxScaler()
minmax_features = scaler.fit_transform(features_df_log)

# Log-transform the SalePrice
sale_price_log = np.log(train_df["SalePrice"])

# Create a scatter plot
fig, ax = plt.subplots(figsize=(10, 5))

# Scatter plot for GrLivArea
ax.scatter(x=minmax_features[:, 0], y=sale_price_log,
label='GrLivArea', alpha=0.6)

# Scatter plot for OverallQual
ax.scatter(x=minmax_features[:, 1], y=sale_price_log,
label='OverallQual', alpha=0.6)

# Add legend
ax.legend()
```

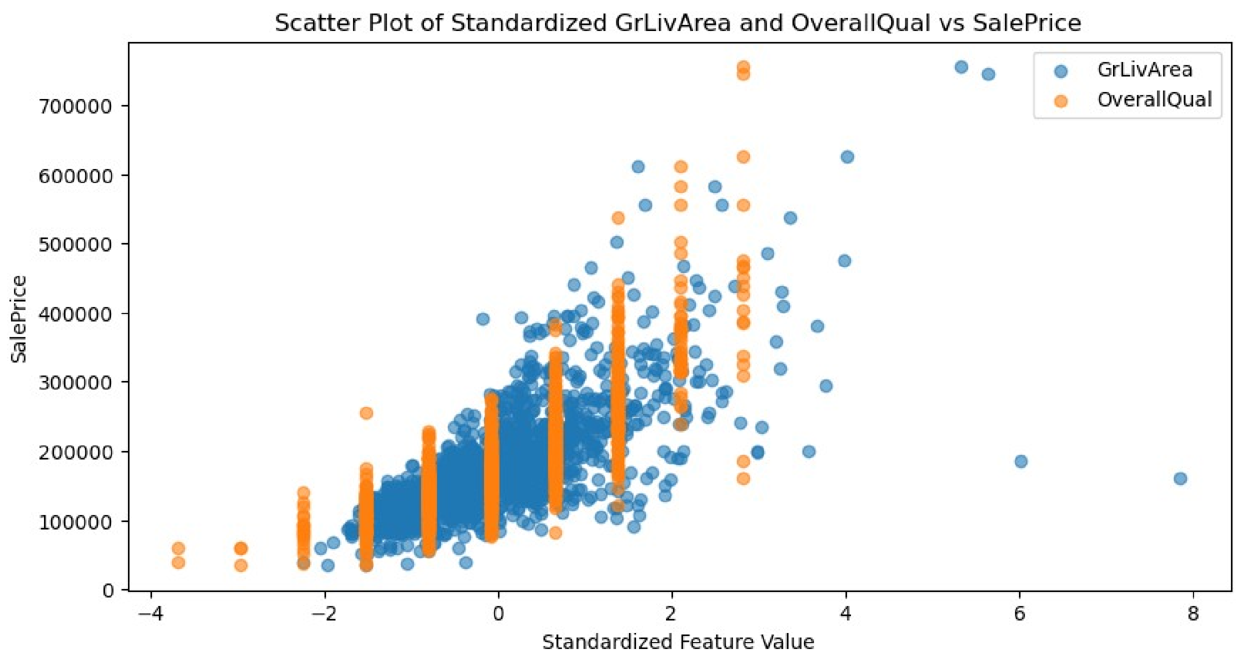
```
# Add labels and title
ax.set_xlabel('Normalized Log Feature Value')
ax.set_ylabel('Log SalePrice')
ax.set_title('Scatter Plot of Log-Transformed GrLivArea and
OverallQual vs Log SalePrice')

# Display the plot
plt.show()
```



Standard Scaling: comparing TotalSF and OverallQual in respect to the SalePrice

```
features = ['GrLivArea', 'OverallQual']
features_df = train_df[features]

# Standardize the features using StandardScaler
scaler = StandardScaler()
standardized_features = scaler.fit_transform(features_df)

# SalePrice
sale_price = train_df["SalePrice"]

# Create a scatter plot
fig, ax = plt.subplots(figsize=(10, 5))

# Plot each feature against SalePrice
for idx, val in enumerate(features):
    ax.scatter(x=standardized_features[:, idx], y=sale_price,
label=val, alpha=0.6)
```

```
# Add legend
ax.legend()

# Add labels and title
ax.set_xlabel('Standardized Feature Value')
ax.set_ylabel('SalePrice')
ax.set_title('Scatter Plot of Standardized GrLivArea and OverallQual
vs SalePrice')

# Display the plot
plt.show()
```

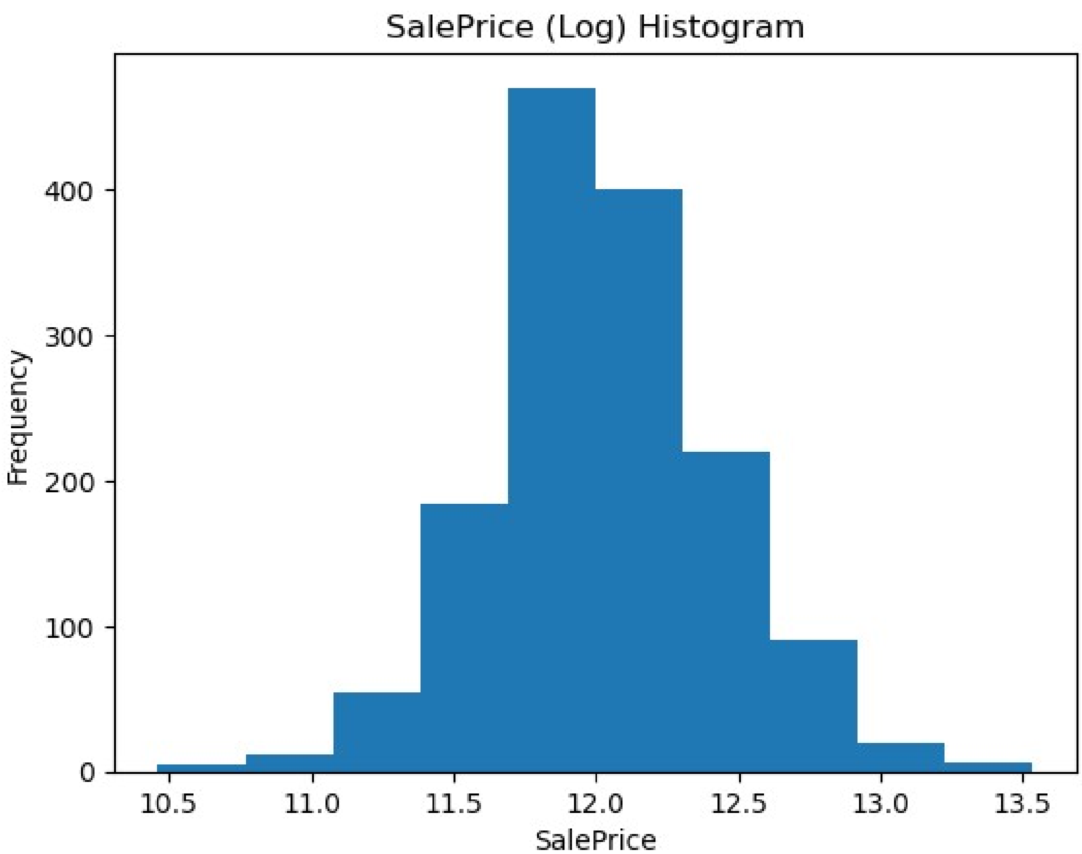


Standard scaling combined with logarithmic transformation doesn't have as dramatic an effect as MinMax scaling with logarithmic transformation because the distributions of the features were already fairly similar before transformation.

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

```
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
```

```python
log_transformed = np.log1p(train_df['SalePrice'])
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
```

(N/A)