Dealing with Outliers in House Prices Data

In this notebook, I will use a house price data set to deal with outliers. In particular, I will...

- Create boxplots
- Identify the type and volume of skewness in the data.
- Handle outliers in the data set to make the data symmetrical

In [15]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import PowerTransformer, StandardScaler
from scipy import stats
```

In [4]:
```
df = pd.read_csv("train.csv")
```

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

Out[5]:

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContou
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

5 rows × 81 columns

In [6]:
```
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
```

In [7]:

```python
df.describe()
```

Out[7]:

	Id	MSSubClass	LotFrontage	LotArea	OverallQual	OverallCond
count	1460.000000	1460.000000	1201.000000	1460.000000	1460.000000	1460.000000
mean	730.500000	56.897260	70.049958	10516.828082	6.099315	5.575342
std	421.610009	42.300571	24.284752	9981.264932	1.382997	1.112799
min	1.000000	20.000000	21.000000	1300.000000	1.000000	1.000000
25%	365.750000	20.000000	59.000000	7553.500000	5.000000	5.000000
50%	730.500000	50.000000	69.000000	9478.500000	6.000000	5.000000
75%	1095.250000	70.000000	80.000000	11601.500000	7.000000	6.000000
max	1460.000000	190.000000	313.000000	215245.000000	10.000000	9.000000

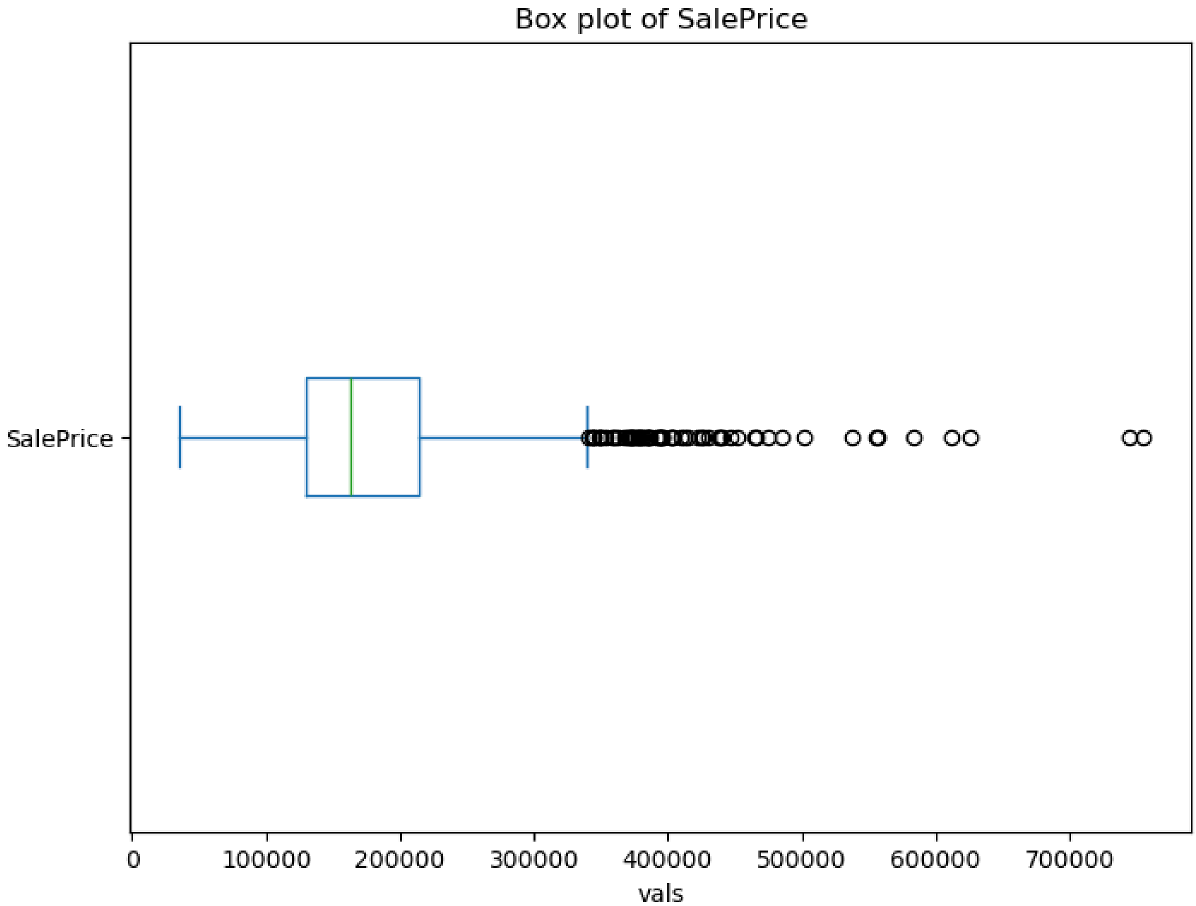
8 rows × 38 columns

In [13]:
```
#Check for duplicate values
duplicates = df.duplicated().sum()
print(duplicates)

#Check for missing values
missing = df.isnull().sum().sort_values(ascending=False)
missing = missing[missing > 0]
print(missing)
```

```
0
PoolQC          1453
MiscFeature     1406
Alley           1369
Fence           1179
MasVnrType       872
FireplaceQu      690
LotFrontage      259
GarageQual        81
GarageFinish      81
GarageType        81
GarageYrBlt       81
GarageCond        81
BsmtFinType2      38
BsmtExposure      38
BsmtCond          37
BsmtQual          37
BsmtFinType1      37
MasVnrArea         8
Electrical         1
dtype: int64
```

In [29]:
```
# Making a box plot of Sale Price
plt.figure(figsize=(8,6))
df['SalePrice'].plot(kind='box', vert=False)
plt.title('Box plot of SalePrice')
plt.xlabel('vals')
plt.show()
```



From this Box plot on Sale price, we can infer a few things...

- The box covers the middle 50% range, with the bottom being the 25%, the top being the 75%, and the green line being the median.
- The box plot is likely right-skewed because the whiskers to the right is longer than the whiskers to the left. But to be sure, we must calculate this.
- Their are a lot of dots to the right that are past these whiskers. These are considered outliers.

In [34]:
```
# Right-skewed means that the data is clustered mostly on the left
# Left-skewed means that the data is clustered mostly on the right
# We can determine if the box plot is right or left-skewed by comparing the mean an
# If mean > median, then the distribution is right-skewed
# If mean < median, then the distribution is left-skewed
print(df['SalePrice'].mean())

print(df['SalePrice'].median())
```

```
180921.19589041095
163000.0
```

In [24]:
```
# Here we can use the following code to get a skewness value
# If skewness > 0, then right-skewed, if skewness < 0, then left-skewed.
# 0 - 0.5 means fairly symmetrical, 0.5 - 1, means moderately skewed, and above 1 m
```

```
skew_vals = df[numeric_cols].skew().sort_values(ascending=False)
print("Skewness:\n", skew_vals)
```

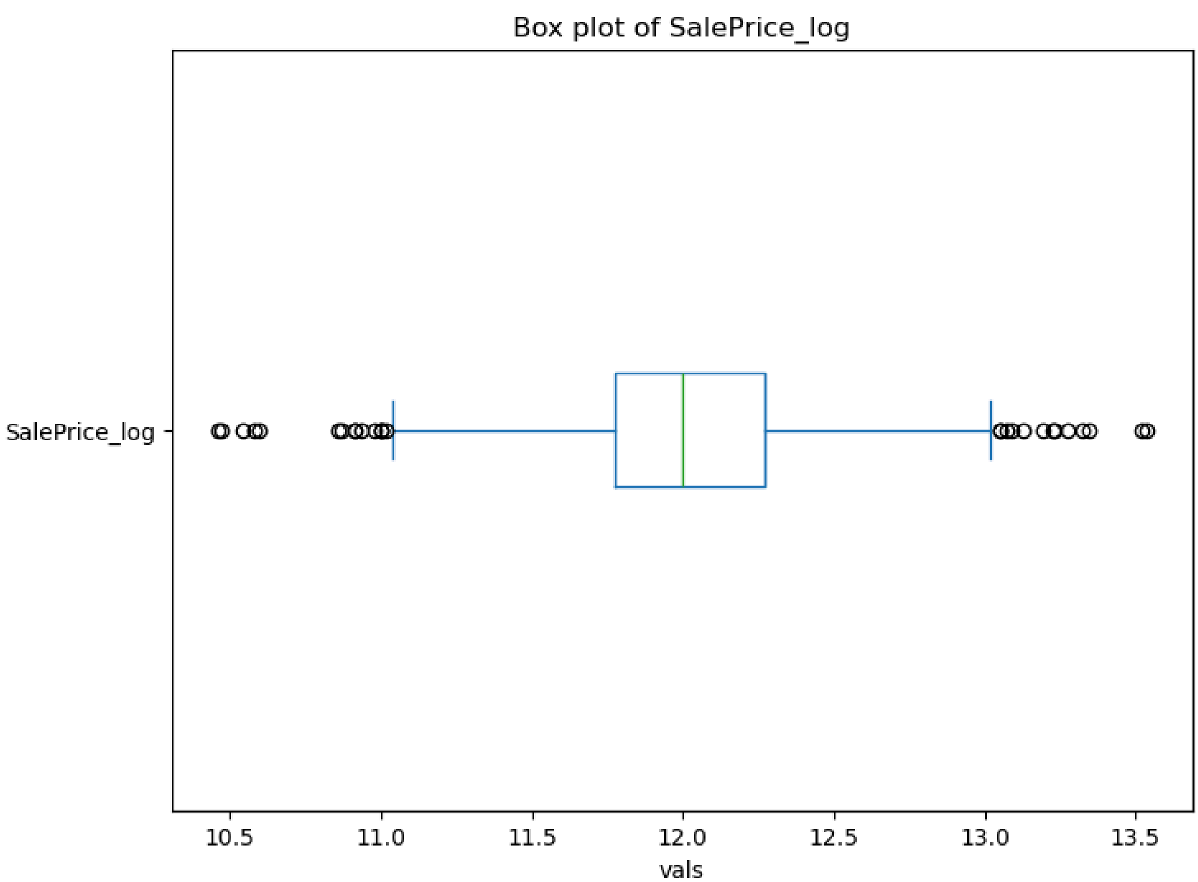
```
Skewness:
 SalePrice    1.882876
dtype: float64
```

In this case, we got a skewness value of 1.88. This means that these outliers on the right side of our data are lifting the mean higher than the median. We want our data to be more symmetrical because it facilitates statistical analysis and machine learning models. Many statistical models assume normality, and reducing the influence of outliers will help improve model performance.

When dealing with right-skewed data, one common transformation to implement is log transform. This transformation compresses large values (outliers), much more than small ones, and works well if skewness is higher than 1 like our in this case.

In [35]:
```
# Creating a new column with SalePrice after the log transformation
df['SalePrice_log'] = np.log1p(df['SalePrice'])
```

In [38]:
```
plt.figure(figsize=(8,6))
df['SalePrice_log'].plot(kind='box', vert=False)
plt.title('Box plot of SalePrice_log')
plt.xlabel('vals')
plt.show()
```



In [40]:
```
# Finding the skewness value of the transformed column.
skew_value = df['SalePrice_log'].skew()
print("Skewness:", skew_value)
```

```
Skewness: 0.12134661989685333
```

Now we can see that the skewness is 0.12. This means that the data is basically symetrical. This balanced distribution is better suited for statistical analysis and predictive modeling.