In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
from sklearn.preprocessing import PolynomialFeatures
from statsmodels.formula.api import ols as sm_ols
from statsmodels.iolib.summary2 import summary_col
```

In [2]:
```
data=pd.read_csv('input_data2/housing_train.csv')
```

Part 1: EDA

Insert cells as needed below to write a short EDA/data section that summarizes the data for someone who has never opened it before.

- Answer essential questions about the dataset (observation units, time period, sample size, many of the questions above)
- Note any issues you have with the data (variable X has problem Y that needs to get addressed before using it in regressions or a prediction model because Z)
- Present any visual results you think are interesting or important

Answer section

sample basics

- unit of observation: each houses
- time spans: years the house was built; years the house was remodeled; years the house was sold

In [29]:
```
# simple exploration
data.columns
```

Out[29]:
```
Index(['parcel', 'v_MS_SubClass', 'v_MS_Zoning', 'v_Lot_Frontage',
       'v_Lot_Area', 'v_Street', 'v_Alley', 'v_Lot_Shape', 'v_Land_Contour',
       'v_Utilities', 'v_Lot_Config', 'v_Land_Slope', 'v_Neighborhood',
       'v_Condition_1', 'v_Condition_2', 'v_Bldg_Type', 'v_House_Style',
       'v_Overall_Qual', 'v_Overall_Cond', 'v_Year_Built', 'v_Year_Remod/Add',
       'v_Roof_Style', 'v_Roof_Matl', 'v_Exterior_1st', 'v_Exterior_2nd',
       'v_Mas_Vnr_Type', 'v_Mas_Vnr_Area', 'v_Exter_Qual', 'v_Exter_Cond',
       'v_Foundation', 'v_Bsmt_Qual', 'v_Bsmt_Cond', 'v_Bsmt_Exposure',
       'v_BsmtFin_Type_1', 'v_BsmtFin_SF_1', 'v_BsmtFin_Type_2',
       'v_BsmtFin_SF_2', 'v_Bsmt_Unf_SF', 'v_Total_Bsmt_SF', 'v_Heating',
       'v_Heating_QC', 'v_Central_Air', 'v_Electrical', 'v_1st_Flr_SF',
       'v_2nd_Flr_SF', 'v_Low_Qual_Fin_SF', 'v_Gr_Liv_Area',
       'v_Bsmt_Full_Bath', 'v_Bsmt_Half_Bath', 'v_Full_Bath', 'v_Half_Bath',
       'v_Bedroom_AbvGr', 'v_Kitchen_AbvGr', 'v_Kitchen_Qual',
       'v_TotRms_AbvGrd', 'v_Functional', 'v_Fireplaces', 'v_Fireplace_Qu',
       'v_Garage_Type', 'v_Garage_Yr_Blt', 'v_Garage_Finish', 'v_Garage_Cars',
       'v_Garage_Area', 'v_Garage_Qual', 'v_Garage_Cond', 'v_Paved_Drive',
       'v_Wood_Deck_SF', 'v_Open_Porch_SF', 'v_Enclosed_Porch', 'v_3Ssn_Porch',
       'v_Screen_Porch', 'v_Pool_Area', 'v_Pool_QC', 'v_Fence',
       'v_Misc_Feature', 'v_Misc_Val', 'v_Mo_Sold', 'v_Yr_Sold', 'v_Sale_Type',
       'v_Sale_Condition', 'v_SalePrice', 'log_v_Lot_Area', 'log_Sale_Price',
       'log_SalePrice'],
      dtype='object')
```

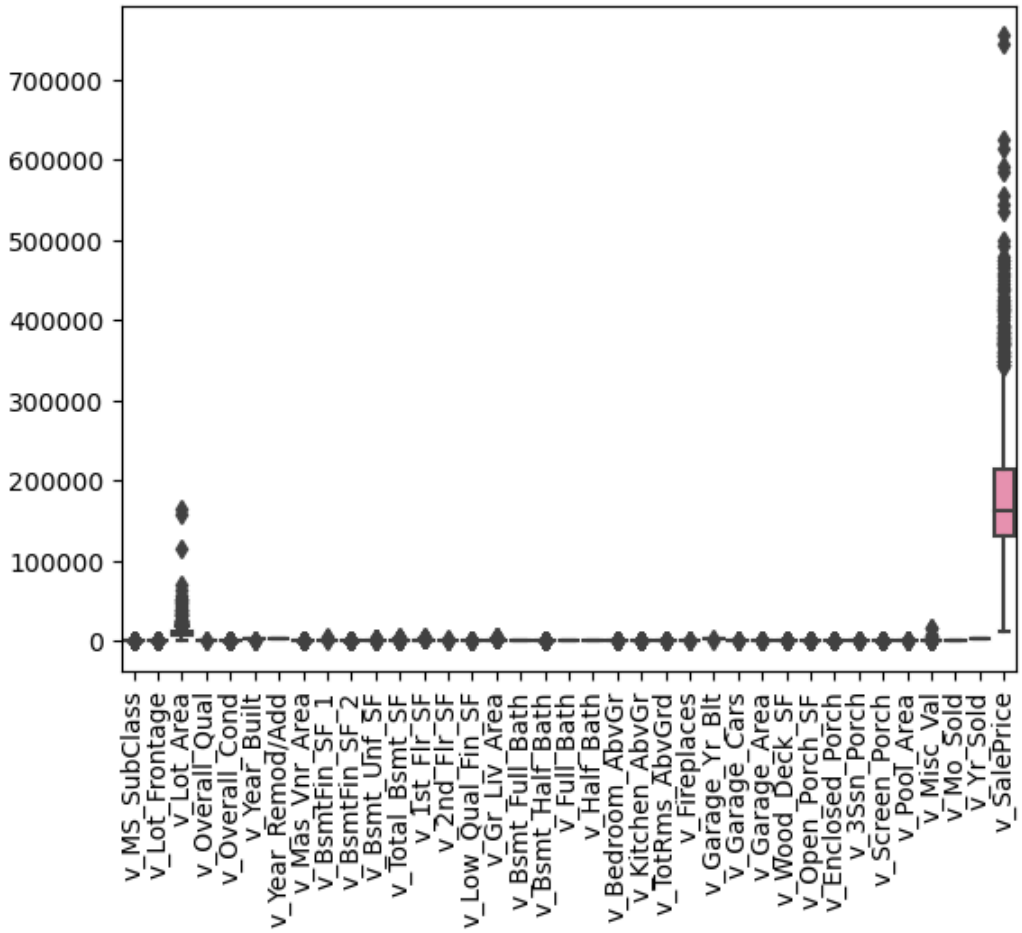
In [30]:
```
data.shape
```

Out[30]: (1941, 84)

outliers, missing values, or data errors

- I searched the excel sheet and didn't find any "NAN", I assumed there's no data errors
- I'll show my work for finding outliers and missing values below

In [3]:
```
# outliers
sns.boxplot(data)
plt.xticks(rotation=90)
plt.show()
```



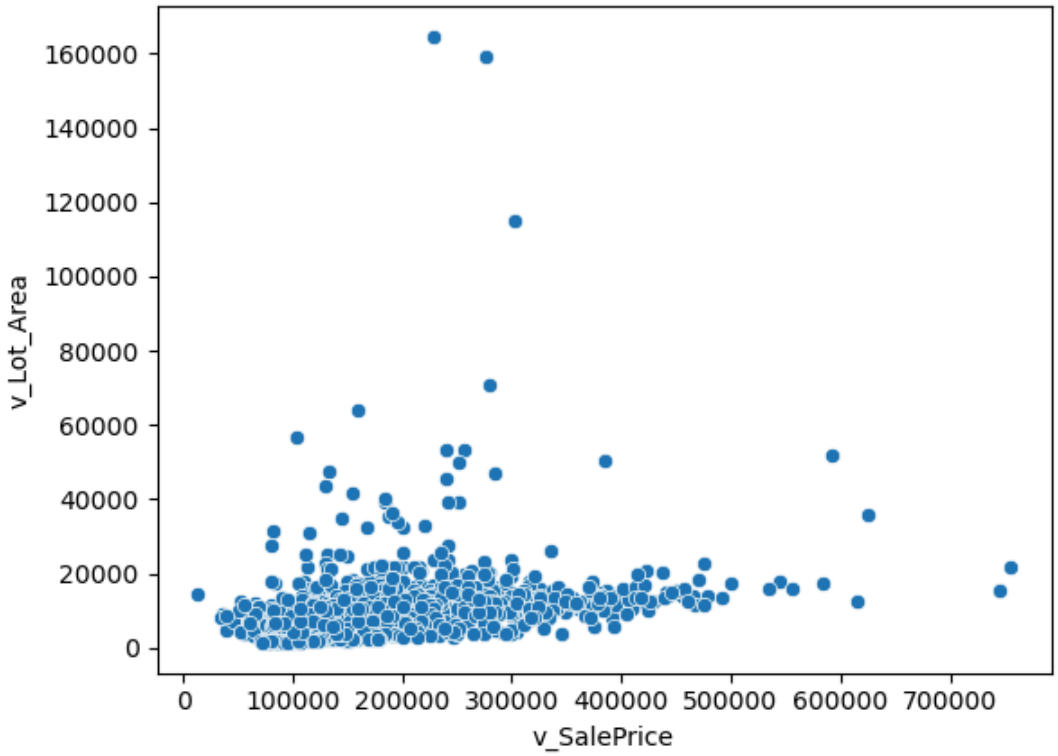
In [4]:
```
# missing values
missing_values = data.isnull()
missing_value_count = data.isnull().sum()
total_missing_value_count = data.isnull().sum().sum()
missing_percentage = (data.isnull().sum() / len(data)) * 100
print("Missing values in each column:")
print(missing_value_count)
print("\nTotal missing values in DataFrame:", total_missing_value_count)
print("\nPercentage of missing values in each column:")
print(missing_percentage)
```

```
Missing values in each column:
parcel                0
v_MS_SubClass         0
v_MS_Zoning           0
v_Lot_Frontage      321
v_Lot_Area            0
                    ...
v_Mo_Sold             0
v_Yr_Sold             0
v_Sale_Type           0
v_Sale_Condition      0
v_SalePrice           0
Length: 81, dtype: int64

Total missing values in DataFrame: 10434

Percentage of missing values in each column:
parcel               0.000000
v_MS_SubClass        0.000000
v_MS_Zoning          0.000000
v_Lot_Frontage      16.537867
v_Lot_Area           0.000000
                        ...
v_Mo_Sold            0.000000
v_Yr_Sold            0.000000
v_Sale_Type          0.000000
v_Sale_Condition     0.000000
v_SalePrice          0.000000
Length: 81, dtype: float64
```

takeaways and findings from the data exploration

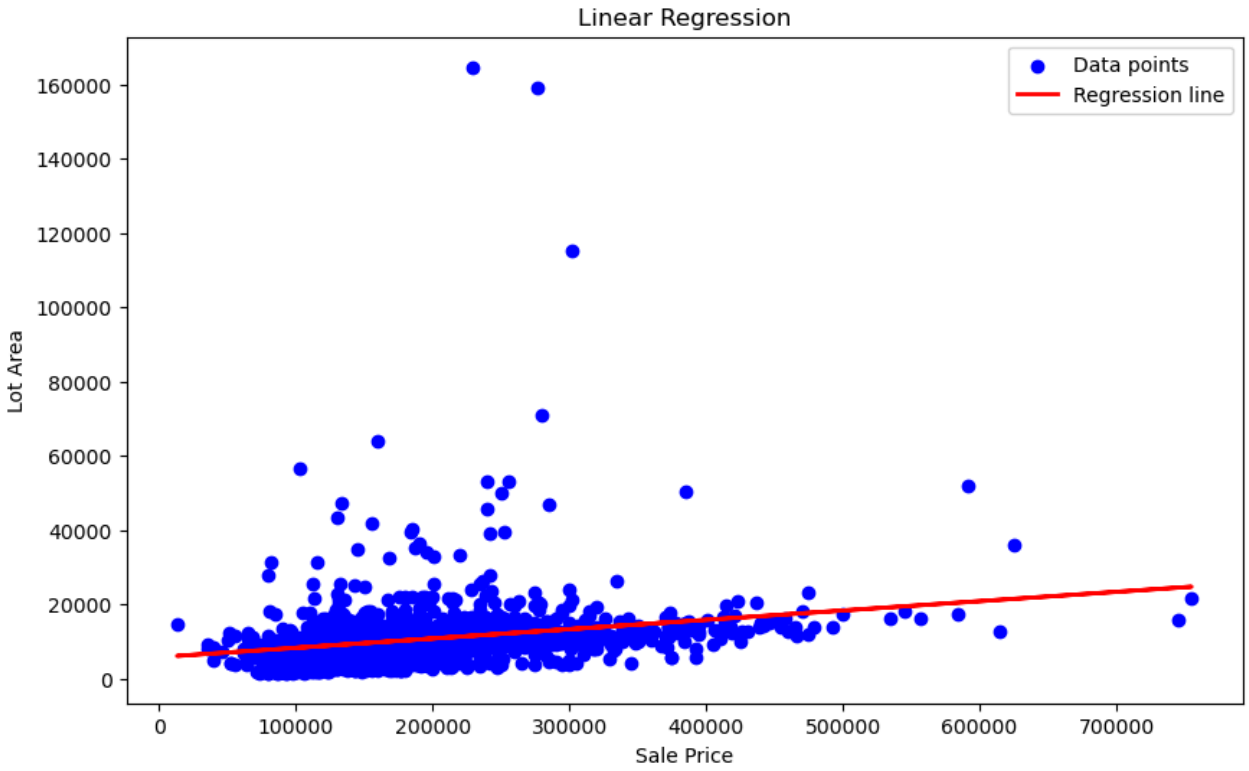
- for the relationship of sale price with continuous variables, I randomly took three variables to make the comparison finding that the relationship between them are majorly linear beside the outliers
- for the categorical data (nominal), I also randomly selected three variables to make the comparison by means of the barplot. The results vary from the selection, I can't really summarize it into conclusive words. But if you just want to know how a certain factor (e.g.: the type of dwelling involved in the sale)is impacting the sale price, this is the go-to solution
- I'll display some of the plots below for better understanding

In [5]:
```
# continuous variables
ax = sns.scatterplot(data = data,
             x='v_SalePrice',y='v_Lot_Area')
```

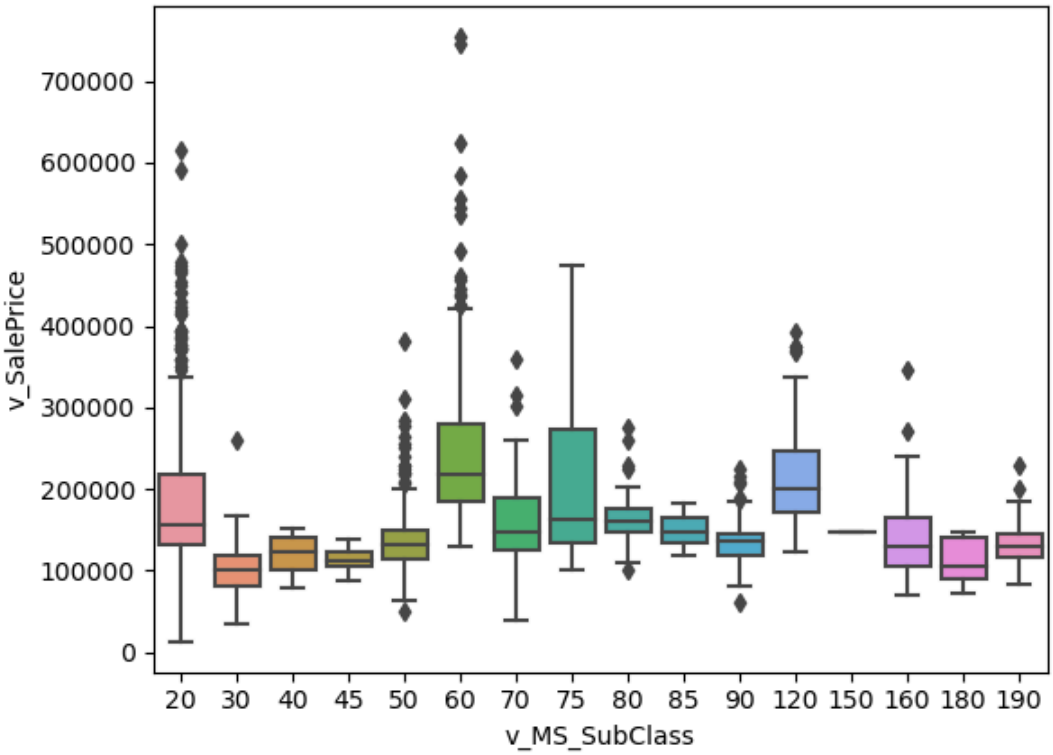


In [6]:
```
X = data['v_SalePrice'].values.reshape(-1, 1)
y = data['v_Lot_Area'].values
model = LinearRegression()
model.fit(X, y)
y_pred = model.predict(X)
print('Slope (Coefficient):', model.coef_[0])
print('Intercept:', model.intercept_)
print('R-squared:', model.score(X, y))
plt.figure(figsize=(10, 6))
plt.scatter(X, y, color='blue', label='Data points')
plt.plot(X, y_pred, color='red', linewidth=2, label='Regression line')
plt.title('Linear Regression')
plt.xlabel('Sale Price')
plt.ylabel('Lot Area')
plt.legend()
plt.show()
```

```
Slope (Coefficient): 0.02513394331576288
Intercept: 5709.557135514661
R-squared: 0.06657818215519129
```



In [7]:
```
# categorical variables
ax = sns.boxplot(data = data,
                 x='v_MS_SubClass',y='v_SalePrice')
```



Part 2: Running Regressions

Run these regressions on the RAW data, even if you found data issues that you think should be addressed.

Insert cells as needed below to run these regressions. Note that i is indexing a given house, and t indexes the year of sale.

1. $\text{Sale Price}_{i,t} = \alpha + \beta_1 * \text{v_Lot_Area}$
2. $\text{Sale Price}_{i,t} = \alpha + \beta_1 * log(\text{v_Lot_Area})$
3. $log(\text{Sale Price}_{i,t}) = \alpha + \beta_1 * \text{v_Lot_Area}$
4. $log(\text{Sale Price}_{i,t}) = \alpha + \beta_1 * log(\text{v_Lot_Area})$
5. $log(\text{Sale Price}_{i,t}) = \alpha + \beta_1 * \text{v_Yr_Sold}$
6. $log(\text{Sale Price}_{i,t}) = \alpha + \beta_1 * (\text{v_Yr_Sold==2007})+ \beta_2 * (\text{v_Yr_Sold==2008})$
7. Choose your own adventure: Pick any five variables from the dataset that you think will generate good R2. Use them in a regression of $log(\text{Sale Price}_{i,t})$
   - Tip: You can transform/create these five variables however you want, even if it creates extra variables. For example: I'd count Model 6 above as only using one variable: `v_Yr_Sold`.
   - I got an R2 of 0.877 with just "5" variables. How close can you get? I won't be shocked if someone beats that!

Bonus formatting trick: Instead of reporting all regressions separately, report all seven regressions in a *single* table using `summary_col`.

In [22]:
```
data['log_v_Lot_Area']=np.log(data['v_Lot_Area'])
data['log_SalePrice']=np.log(data['v_SalePrice'])
```

In [28]:
```
# bonus
results1 = sm_ols("v_SalePrice ~ v_Lot_Area", data=data).fit()
results2 = sm_ols("v_SalePrice ~ log_v_Lot_Area", data=data).fit()
results3 = sm_ols("log_SalePrice ~ v_Lot_Area", data=data).fit()
results4 = sm_ols("log_SalePrice ~ log_v_Lot_Area", data=data).fit()
results5 = sm_ols("log_SalePrice ~ v_Yr_Sold", data=data).fit()
results6 = sm_ols("log_SalePrice ~ C(v_Yr_Sold)", data=data).fit()
results7 = sm_ols("log_SalePrice ~ v_Lot_Area + v_Lot_Frontage + v_Garage_Area + v_Total_Bsmt_SF

results = [results1, results2, results3, results4, results5, results6, results7]
summary_table = summary_col(results, stars=True, float_format='%0.5f',
                            model_names=['Model 1', 'Model 2', 'Model 3(log)','Model 4(log)','Mod
                            info_dict={'R-squared': lambda x: "{:.5f}".format(x.rsquared),
                                       'Adj R-squared' : lambda x: f"{x.rsquared_adj:.5f}",
                                       'No. observations': lambda x: "{:.0f}".format(int(x.nobs))
print('                  y = sale price,')
print(summary_table)
```

```
                  y = sale price,

==============================================================================================
=====================
                         Model 1         Model 2      Model 3(log) Model 4(log) Model 5(log) Mode
l 6(log) Model 7(log)
----------------------------------------------------------------------------------------------
---------------------
C(v_Yr_Sold)[T.2007]                                                                      0.02
559
                                                                                          (0.0
2225)
C(v_Yr_Sold)[T.2008]                                                                      -0.0
1028
                                                                                          (0.0
2285)
Intercept            154789.55021*** -327915.80232*** 11.89407***  9.40505***   22.29321     12.0
2287***  11.18443***
                     (2911.59058)    (30221.34714)    (0.01463)    (0.15108)    (22.93682)   (0.0
1614)    (0.02406)
R-squared            0.06658         0.12840          0.06459      0.13497      0.00010      0.00
144      0.55486
R-squared Adj.       0.06610         0.12795          0.06411      0.13453      -0.00041     0.00
041      0.55348
log_v_Lot_Area                       56028.16996***                0.28826***
                                     (3315.13919)                  (0.01657)
v_Garage_Area
0.00085***

(0.00004)
v_Lot_Area           2.64894***                       0.00001***
0.00000*
                     (0.22525)                        (0.00000)
(0.00000)
v_Lot_Frontage
0.00068*

(0.00037)
v_Pool_Area
-0.00025

(0.00016)
v_Total_Bsmt_SF
0.00034***

(0.00002)
v_Yr_Sold                                                                 -0.00511
                                                                          (0.01143)
R-squared            0.06658         0.12840          0.06459      0.13497      0.00010      0.00
144      0.55486
Adj R-squared        0.06610         0.12795          0.06411      0.13453      -0.00041     0.00
041      0.55348
No. observations     1941            1941             1941         1941         1941         1941
1618
==============================================================================================
=====================
Standard errors in parentheses.
* p<.1, ** p<.05, ***p<.01
```

Part 3: Regression interpretation

Insert cells as needed below to answer these questions. Note that i is indexing a given house, and t indexes the year of sale.

1. If you didn't use the `summary_col` trick, list β_1 for Models 1-6 to make it easier on your graders.
2. Interpret β_1 in Model 2.
3. Interpret β_1 in Model 3.
    - HINT: You might need to print out more decimal places. Show at least 2 non-zero digits.
4. Of models 1-4, which do you think best explains the data and why?
5. Interpret β_1 In Model 5
6. Interpret α in Model 6
7. Interpret β_1 in Model 6
8. Why is the R2 of Model 6 higher than the R2 of Model 5?
9. What variables did you include in Model 7?
10. What is the R2 of your Model 7?
11. Speculate (not graded): Could you use the specification of Model 6 in a predictive regression?
12. Speculate (not graded): Could you use the specification of Model 5 in a predictive regression?

In [10]:
```
# q1
# skip
```

In [11]:
```
# q2
# Beta 1: A 1% increase in lot size, is associated with a 560.28 units decline in sale price
# Beta 1=56028.16996
```

In [ ]:
```
# q3
# Beta 1: A 1 unit increase in lot size, is associated with a 0.001% increase in sale price
# Beta 1=0.00001
```

In [ ]:
```
# q4
# this would be a subjective choice based on the context and the R-squared values
# higher R-squared indicates a better fit to the data
# in this case, Model 4 can best describe the data
```

In [ ]:
```
# q5
# Beta 1: sale price is about 0.511% lower for cases when v_Yr_Sold=1 than when v_Yr_Sold=0
# Beta 1=-0.00511
```

For better understanding (equation 6)

if v_Yr_Sold (original) is	Then [v_Yr_Sold==2007]=	Then [v_Yr_Sold==2008]=
2006	0	0
2007	1	0
2008	0	1

In [ ]:
```
# q6
# intercept=12.02287
# the average value of sale price is 12.02287 for group 0 (because [v_Yr_Sold==2007]=[v_Yr_Sold==
```

In [ ]:
```
# q7
# Beta 1: sale price is about 2.599% higher on average for cases when [v_Yr_Sold==2007]=1 than wh
# Beta 1=0.02559
```

In [ ]:
```
# q8
# Model 6 may have a higher R-squared than Model 5 due to the inclusion of additional variables o
```

In [ ]:
```
# q9
# columns I chose: v_Lot_Area, v_Lot_Frontage, v_Garage_Area, v_Total_Bsmt_SF, v_Pool_Area
# what they represent: Lot size in square feet, Linear feet of street connected to property, Size
```

In [ ]:
```
# q10
# R-squared=0.55486
```

In [ ]:
```
# q11
# Model 6 might be suitable for predictive regression if it includes relevant predictors that cap
# for instance, if the coefficients for the year variables in Model 6 are significant and the yea
# however, the suitability also depends on the stability of the factors affecting sale prices ove
# if the market conditions in 2006-2008 are significantly different from the conditions in the pe
# It's also crucial to validate the model on a separate dataset to ensure it doesn't overfit the
```

In [ ]:
```
# q12
# Model 5 can be used in a predictive regression under similar considerations as for Model 6.
# the key here is the transformation applied to the sale price and lot area into a logarithmic sc
# the log transformation of sale price implies that we're interested in percentage changes rather
# the interpretation of the coefficients in the log-log model (like Model 5) reflects elasticity,
# if a 1% change in lot area is consistently associated with a 01% change in sale price, and if t
# in both cases, the inclusion of variables should be theoretically justified, and the models sho
# additionally, any predictions made would need to be back-transformed from the logarithmic scale
```