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

In [2]:
```
import pandas as pd
import numpy as np

df = pd.read_csv('E:\\FIN377\\asgn-06-Shanshan417\\input_data2\\housing_train.csv')
```

In [3]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1941 entries, 0 to 1940
Data columns (total 81 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   parcel             1941 non-null   object 
 1   v_MS_SubClass      1941 non-null   int64  
 2   v_MS_Zoning        1941 non-null   object 
 3   v_Lot_Frontage     1620 non-null   float64
 4   v_Lot_Area         1941 non-null   int64  
 5   v_Street           1941 non-null   object 
 6   v_Alley            136 non-null    object 
 7   v_Lot_Shape        1941 non-null   object 
 8   v_Land_Contour     1941 non-null   object 
 9   v_Utilities        1941 non-null   object 
 10  v_Lot_Config       1941 non-null   object 
 11  v_Land_Slope       1941 non-null   object 
 12  v_Neighborhood     1941 non-null   object 
 13  v_Condition_1      1941 non-null   object 
 14  v_Condition_2      1941 non-null   object 
 15  v_Bldg_Type        1941 non-null   object 
 16  v_House_Style      1941 non-null   object 
 17  v_Overall_Qual     1941 non-null   int64  
 18  v_Overall_Cond     1941 non-null   int64  
 19  v_Year_Built       1941 non-null   int64  
 20  v_Year_Remod/Add   1941 non-null   int64  
 21  v_Roof_Style       1941 non-null   object 
 22  v_Roof_Matl        1941 non-null   object 
 23  v_Exterior_1st     1941 non-null   object 
 24  v_Exterior_2nd     1941 non-null   object 
 25  v_Mas_Vnr_Type     769 non-null    object 
 26  v_Mas_Vnr_Area     1923 non-null   float64
 27  v_Exter_Qual       1941 non-null   object 
 28  v_Exter_Cond       1941 non-null   object 
 29  v_Foundation       1941 non-null   object 
 30  v_Bsmt_Qual        1891 non-null   object 
 31  v_Bsmt_Cond        1891 non-null   object 
 32  v_Bsmt_Exposure    1889 non-null   object 
 33  v_BsmtFin_Type_1   1891 non-null   object 
 34  v_BsmtFin_SF_1     1940 non-null   float64
 35  v_BsmtFin_Type_2   1891 non-null   object 
 36  v_BsmtFin_SF_2     1940 non-null   float64
 37  v_Bsmt_Unf_SF      1940 non-null   float64
 38  v_Total_Bsmt_SF    1940 non-null   float64
 39  v_Heating          1941 non-null   object 
 40  v_Heating_QC       1941 non-null   object 
 41  v_Central_Air      1941 non-null   object 
 42  v_Electrical       1940 non-null   object 
 43  v_1st_Flr_SF       1941 non-null   int64  
 44  v_2nd_Flr_SF       1941 non-null   int64  
 45  v_Low_Qual_Fin_SF  1941 non-null   int64  
 46  v_Gr_Liv_Area      1941 non-null   int64  
 47  v_Bsmt_Full_Bath   1939 non-null   float64
 48  v_Bsmt_Half_Bath   1939 non-null   float64
 49  v_Full_Bath        1941 non-null   int64  
 50  v_Half_Bath        1941 non-null   int64  
```

```
 51  v_Bedroom_AbvGr    1941 non-null   int64
 52  v_Kitchen_AbvGr    1941 non-null   int64
 53  v_Kitchen_Qual     1941 non-null   object
 54  v_TotRms_AbvGrd    1941 non-null   int64
 55  v_Functional       1941 non-null   object
 56  v_Fireplaces       1941 non-null   int64
 57  v_Fireplace_Qu     1001 non-null   object
 58  v_Garage_Type      1836 non-null   object
 59  v_Garage_Yr_Blt    1834 non-null   float64
 60  v_Garage_Finish    1834 non-null   object
 61  v_Garage_Cars      1940 non-null   float64
 62  v_Garage_Area      1940 non-null   float64
 63  v_Garage_Qual      1834 non-null   object
 64  v_Garage_Cond      1834 non-null   object
 65  v_Paved_Drive      1941 non-null   object
 66  v_Wood_Deck_SF     1941 non-null   int64
 67  v_Open_Porch_SF    1941 non-null   int64
 68  v_Enclosed_Porch   1941 non-null   int64
 69  v_3Ssn_Porch       1941 non-null   int64
 70  v_Screen_Porch     1941 non-null   int64
 71  v_Pool_Area        1941 non-null   int64
 72  v_Pool_QC          13 non-null     object
 73  v_Fence            365 non-null    object
 74  v_Misc_Feature     63 non-null     object
 75  v_Misc_Val         1941 non-null   int64
 76  v_Mo_Sold          1941 non-null   int64
 77  v_Yr_Sold          1941 non-null   int64
 78  v_Sale_Type        1941 non-null   object
 79  v_Sale_Condition   1941 non-null   object
 80  v_SalePrice        1941 non-null   int64
dtypes: float64(11), int64(26), object(44)
memory usage: 1.2+ MB
```

In [4]:
```
df.describe()
```

Out[4]:

	v_MS_SubClass	v_Lot_Frontage	v_Lot_Area	v_Overall_Qual	v_Overall_Cond
count	1941.000000	1620.000000	1941.000000	1941.000000	1941.000000
mean	58.088614	69.301235	10284.770222	6.113344	5.568264
std	42.946015	23.978101	7832.295527	1.401594	1.087465
min	20.000000	21.000000	1470.000000	1.000000	1.000000
25%	20.000000	58.000000	7420.000000	5.000000	5.000000
50%	50.000000	68.000000	9450.000000	6.000000	5.000000
75%	70.000000	80.000000	11631.000000	7.000000	6.000000
max	190.000000	313.000000	164660.000000	10.000000	9.000000

8 rows × 37 columns

In [5]:
```
df.isnull().sum()
```

Out[5]:
```
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
```

In [6]:
```
df.nunique().sort_values()
```

Out[6]:
```
v_Central_Air       2
v_Street            2
v_Alley             2
v_Utilities         2
v_Yr_Sold           3
                 ...
v_1st_Flr_SF      901
v_Bsmt_Unf_SF     938
v_Gr_Liv_Area    1045
v_Lot_Area       1413
parcel           1941
Length: 81, dtype: int64
```

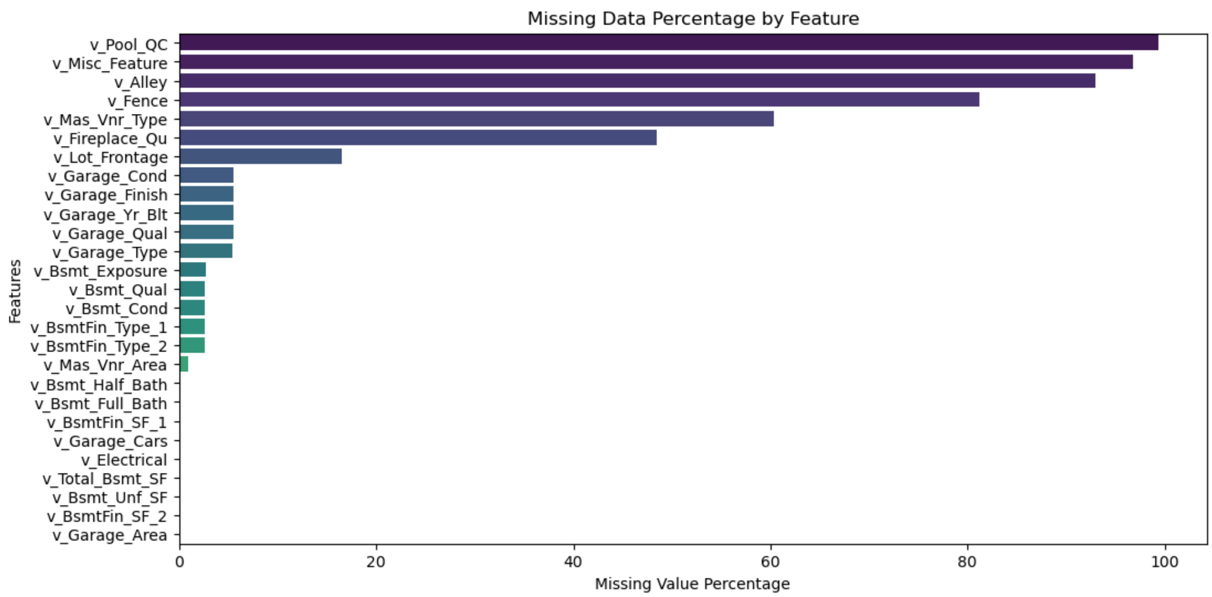
In [7]:
```
missing = df.isnull().mean().sort_values(ascending=False)
print(missing[missing > 0])
```

```
v_Pool_QC           0.993302
v_Misc_Feature      0.967543
v_Alley             0.929933
v_Fence             0.811953
v_Mas_Vnr_Type      0.603812
v_Fireplace_Qu      0.484286
v_Lot_Frontage      0.165379
v_Garage_Cond       0.055126
v_Garage_Finish     0.055126
v_Garage_Yr_Blt     0.055126
v_Garage_Qual       0.055126
v_Garage_Type       0.054096
v_Bsmt_Exposure     0.026790
v_Bsmt_Qual         0.025760
v_Bsmt_Cond         0.025760
v_BsmtFin_Type_1    0.025760
v_BsmtFin_Type_2    0.025760
v_Mas_Vnr_Area      0.009274
v_Bsmt_Half_Bath    0.001030
v_Bsmt_Full_Bath    0.001030
v_BsmtFin_SF_1      0.000515
v_Garage_Cars       0.000515
v_Electrical        0.000515
v_Total_Bsmt_SF     0.000515
v_Bsmt_Unf_SF       0.000515
v_BsmtFin_SF_2      0.000515
v_Garage_Area       0.000515
dtype: float64
```

In [8]:

```
import seaborn as sns
import matplotlib.pyplot as plt

missing_percent = df.isnull().mean().sort_values(ascending=False) * 100
missing_percent = missing_percent[missing_percent > 0]

plt.figure(figsize=(12, 6))
sns.barplot(x=missing_percent.values, y=missing_percent.index, palette="viridis")
plt.xlabel("Missing Value Percentage")
plt.ylabel("Features")
plt.title("Missing Data Percentage by Feature")
plt.show()
```

```
C:\Users\lenovo\AppData\Local\Temp\ipykernel_52060\3277439945.py:8: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.1
4.0. Assign the `y` variable to `hue` and set `legend=False` for the same effect.

  sns.barplot(x=missing_percent.values, y=missing_percent.index, palette="viridis")
```



In [9]:
```
none_fill = [
    'v_Pool_QC', 'v_Misc_Feature', 'v_Alley', 'v_Fence',
    'v_Fireplace_Qu', 'v_Garage_Type', 'v_Garage_Finish',
    'v_Garage_Qual', 'v_Garage_Cond', 'v_Bsmt_Exposure',
    'v_Bsmt_Qual', 'v_Bsmt_Cond', 'v_BsmtFin_Type_1',
    'v_BsmtFin_Type_2', 'v_Electrical'
]
for col in none_fill:
    df[col] = df[col].fillna('None')
```

In [10]:
```
zero_fill = [
    'v_Garage_Yr_Blt', 'v_Mas_Vnr_Area', 'v_Bsmt_Full_Bath',
    'v_Bsmt_Half_Bath', 'v_BsmtFin_SF_1', 'v_BsmtFin_SF_2',
    'v_Bsmt_Unf_SF', 'v_Total_Bsmt_SF', 'v_Garage_Cars',
    'v_Garage_Area'
]
for col in zero_fill:
    df[col] = df[col].fillna(0)
```

In [11]:
```
mode_fill = ['v_Electrical']
for col in mode_fill:
    df[col] = df[col].fillna(df[col].mode()[0])
```

In [12]:
```
from sklearn.ensemble import RandomForestRegressor

features = ['v_Overall_Qual', 'v_Lot_Area', 'v_Year_Built', 'v_Gr_Liv_Area']
lot_data = df[features + ['v_Lot_Frontage']]
train_data = lot_data[lot_data['v_Lot_Frontage'].notnull()]
predict_data = lot_data[lot_data['v_Lot_Frontage'].isnull()]
X_train = train_data[features]
y_train = train_data['v_Lot_Frontage']
model = RandomForestRegressor(random_state=0, n_estimators=100)
model.fit(X_train, y_train)
X_predict = predict_data[features]
predicted_values = model.predict(X_predict)
df.loc[df['v_Lot_Frontage'].isnull(), 'v_Lot_Frontage'] = predicted_values
```

In [13]:
```
missing = df.isnull().mean().sort_values(ascending=False)
print(missing[missing > 0])
```

```
v_Mas_Vnr_Type    0.603812
dtype: float64
```

In [14]:
```
df['v_Mas_Vnr_Type'] = df['v_Mas_Vnr_Type'].fillna('None')
```

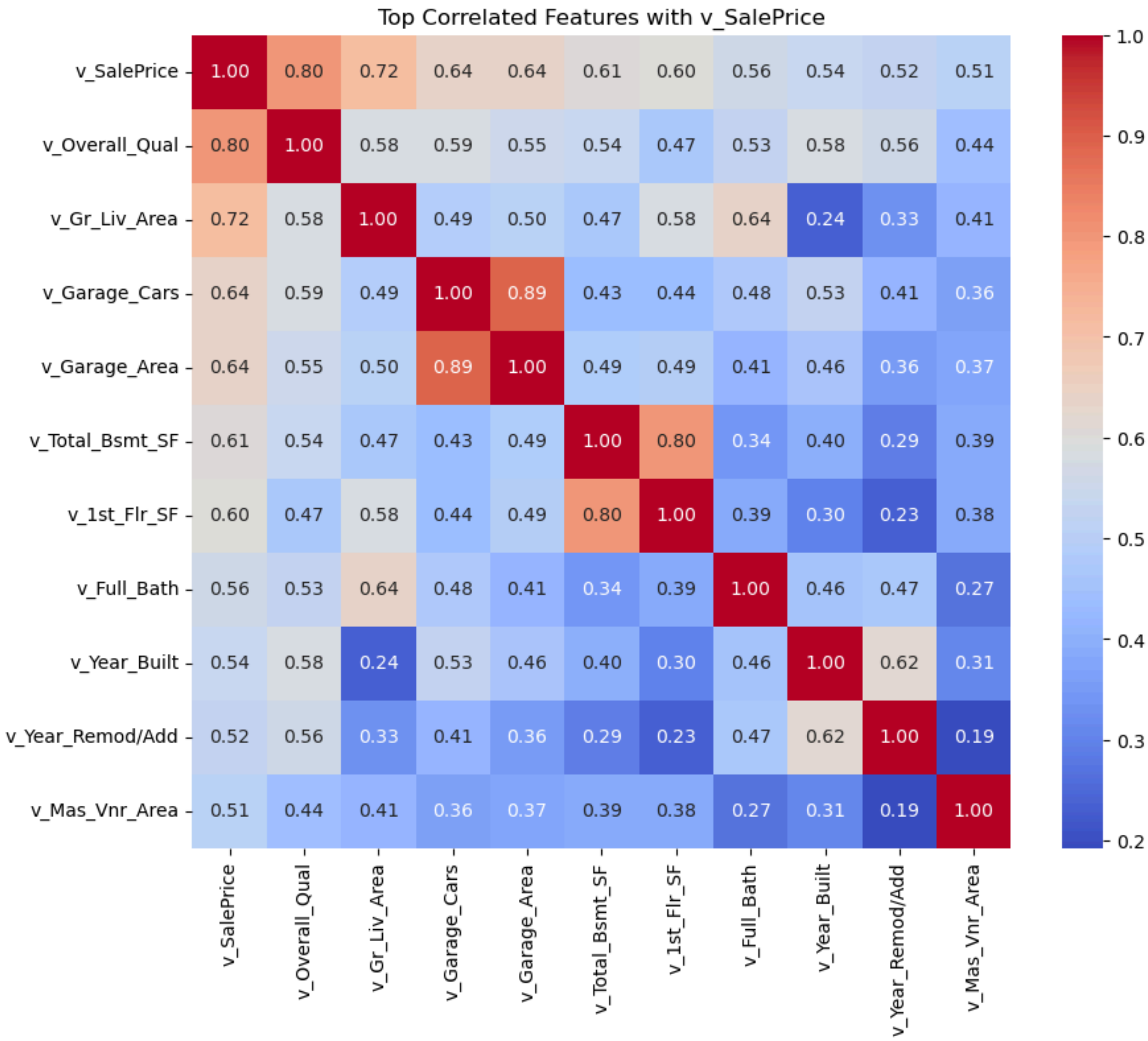
In [15]:
```
missing = df.isnull().mean().sort_values(ascending=False)
print(missing[missing > 0])
```

```
Series([], dtype: float64)
```

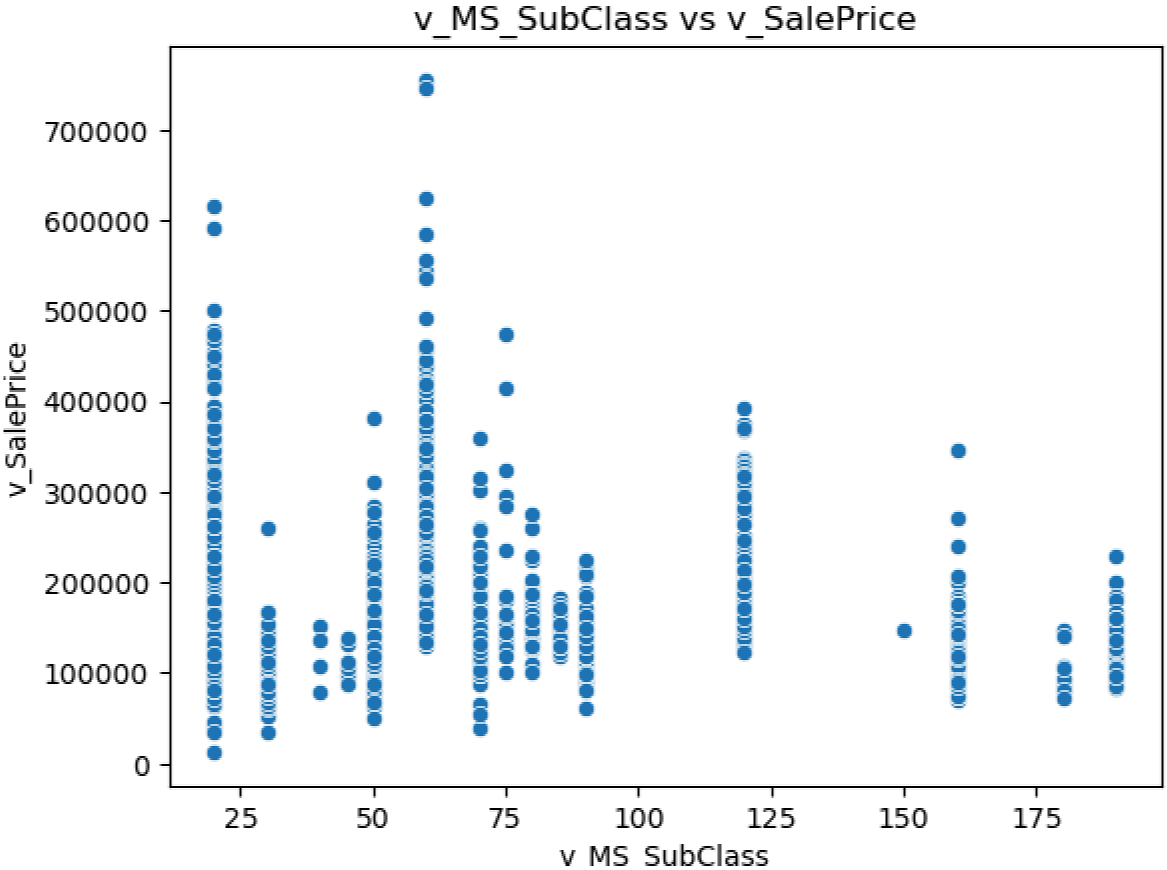
In [16]:
```
numerical = df.select_dtypes(include=['int64', 'float64']).columns
categorical = df.select_dtypes(include=['object']).columns
print("Numerical:", numerical)
print("Categorical:", categorical)
```

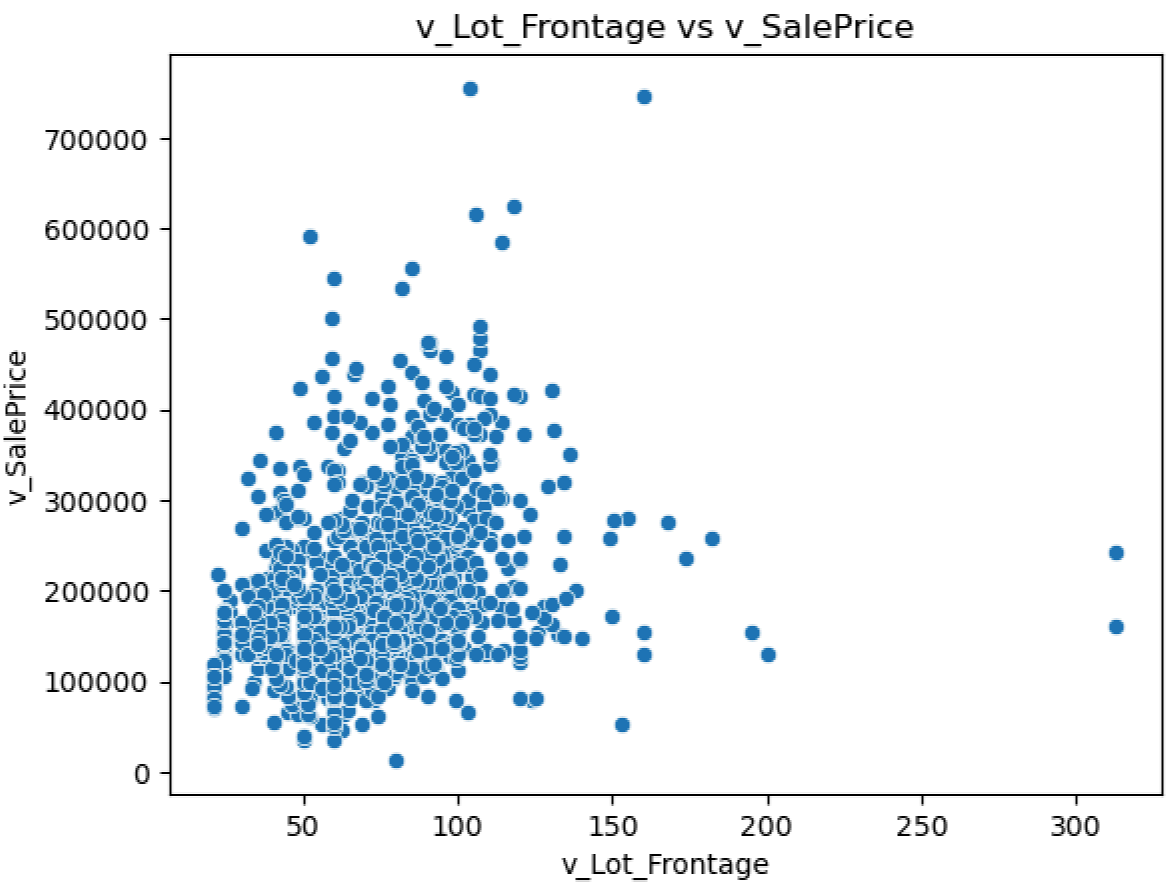
```
Numerical: Index(['v_MS_SubClass', 'v_Lot_Frontage', 'v_Lot_Area', 'v_Overall_Qual',
       'v_Overall_Cond', 'v_Year_Built', 'v_Year_Remod/Add', 'v_Mas_Vnr_Area',
       'v_BsmtFin_SF_1', 'v_BsmtFin_SF_2', 'v_Bsmt_Unf_SF', 'v_Total_Bsmt_SF',
       'v_1st_Flr_SF', 'v_2nd_Flr_SF', 'v_Low_Qual_Fin_SF', '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_TotRms_AbvGrd', 'v_Fireplaces',
       'v_Garage_Yr_Blt', 'v_Garage_Cars', 'v_Garage_Area', 'v_Wood_Deck_SF',
       'v_Open_Porch_SF', 'v_Enclosed_Porch', 'v_3Ssn_Porch', 'v_Screen_Porch',
       'v_Pool_Area', 'v_Misc_Val', 'v_Mo_Sold', 'v_Yr_Sold', 'v_SalePrice'],
      dtype='object')
Categorical: Index(['parcel', 'v_MS_Zoning', '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_Roof_Style', 'v_Roof_Matl', 'v_Exterior_1st',
       'v_Exterior_2nd', 'v_Mas_Vnr_Type', 'v_Exter_Qual', 'v_Exter_Cond',
       'v_Foundation', 'v_Bsmt_Qual', 'v_Bsmt_Cond', 'v_Bsmt_Exposure',
       'v_BsmtFin_Type_1', 'v_BsmtFin_Type_2', 'v_Heating', 'v_Heating_QC',
       'v_Central_Air', 'v_Electrical', 'v_Kitchen_Qual', 'v_Functional',
       'v_Fireplace_Qu', 'v_Garage_Type', 'v_Garage_Finish', 'v_Garage_Qual',
       'v_Garage_Cond', 'v_Paved_Drive', 'v_Pool_QC', 'v_Fence',
       'v_Misc_Feature', 'v_Sale_Type', 'v_Sale_Condition'],
      dtype='object')
```

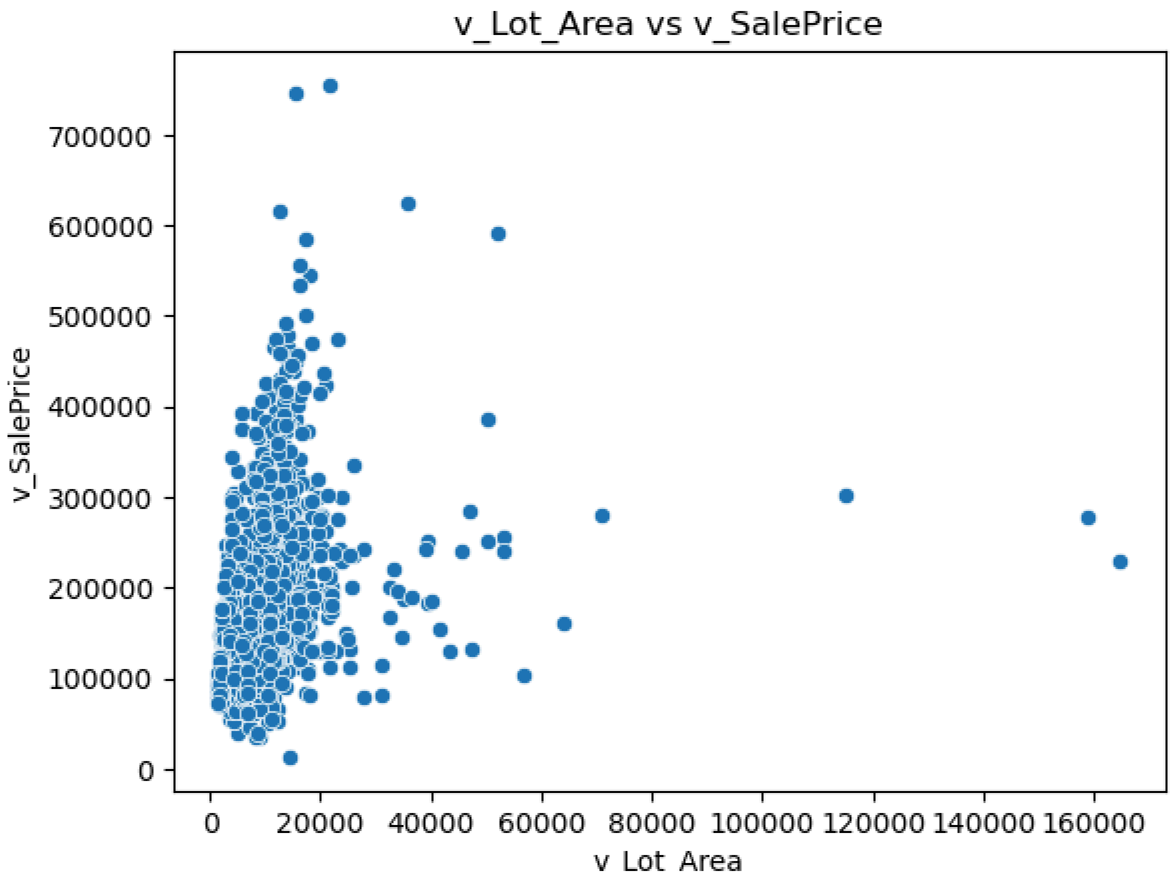
In [17]:
```
corr = df.corr(numeric_only=True)
top_corr = corr['v_SalePrice'].abs().sort_values(ascending=False).head(11)
top_features = top_corr.index

plt.figure(figsize=(10, 8))
sns.heatmap(df[top_features].corr(), annot=True, cmap='coolwarm', fmt='.2f')
plt.title('Top Correlated Features with v_SalePrice')
plt.show()
```

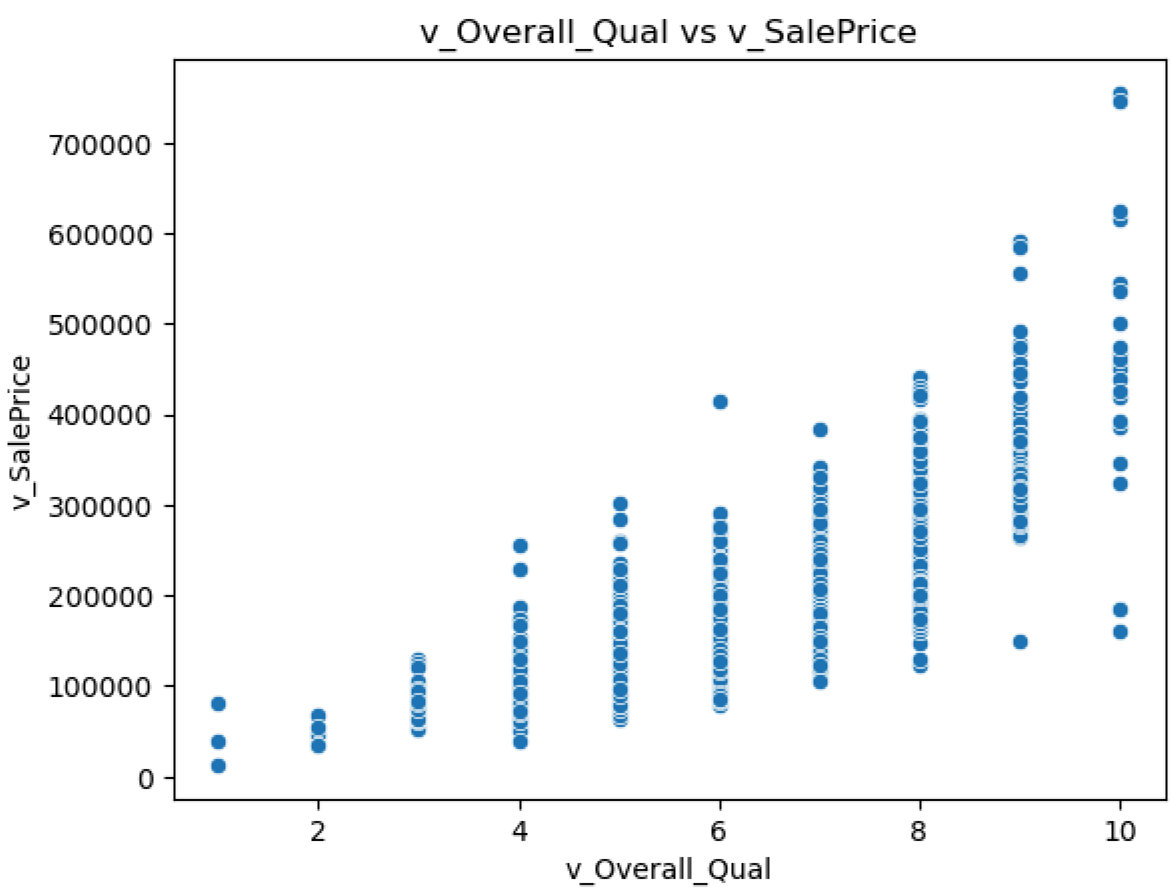


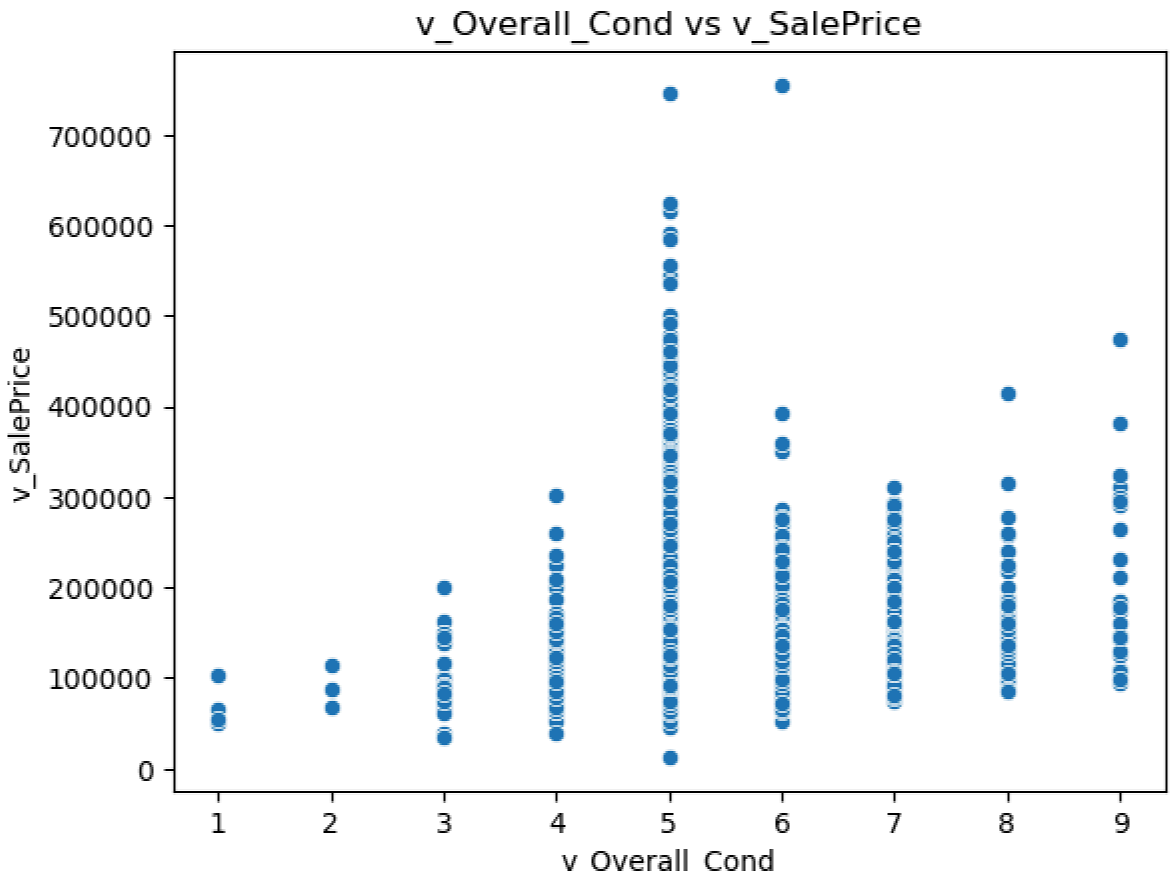
In [18]:
```
for col in numerical:
    if col != 'v_SalePrice':
        sns.scatterplot(x=df[col], y=df['v_SalePrice'])
        plt.title(f'{col} vs v_SalePrice')
        plt.show()
```


v_MS_SubClass vs v_SalePrice
v_SalePrice
0
100000
200000
300000
400000
500000
600000
700000
25
50
75
100
125
150
175
v_MS_SubClass

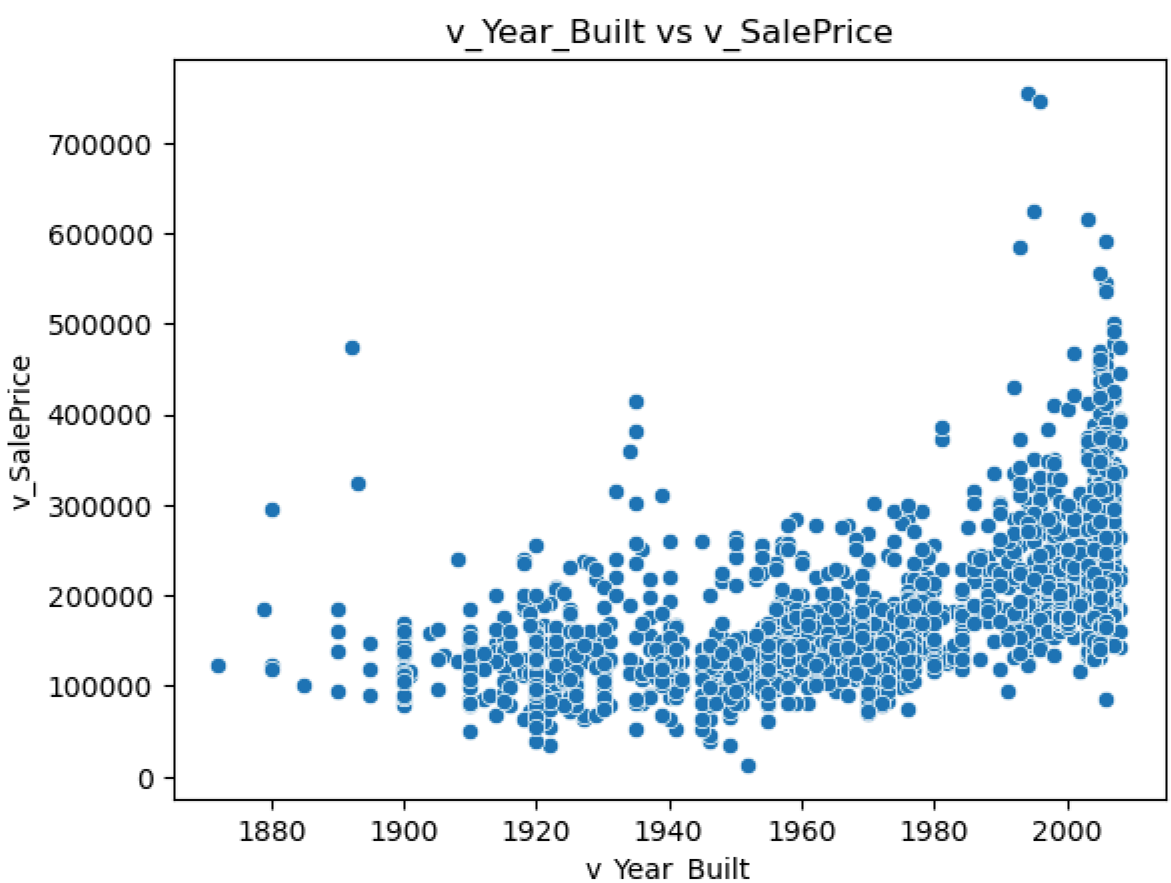

v_Lot_Frontage vs v_SalePrice
v_SalePrice
0
100000
200000
300000
400000
500000
600000
700000
50
100
150
200
250
300
v_Lot_Frontage


v_Lot_Area vs v_SalePrice
v_SalePrice
v_Lot_Area
0
100000
200000
300000
400000
500000
600000
700000
20000
40000
60000
80000
100000
120000
140000
160000

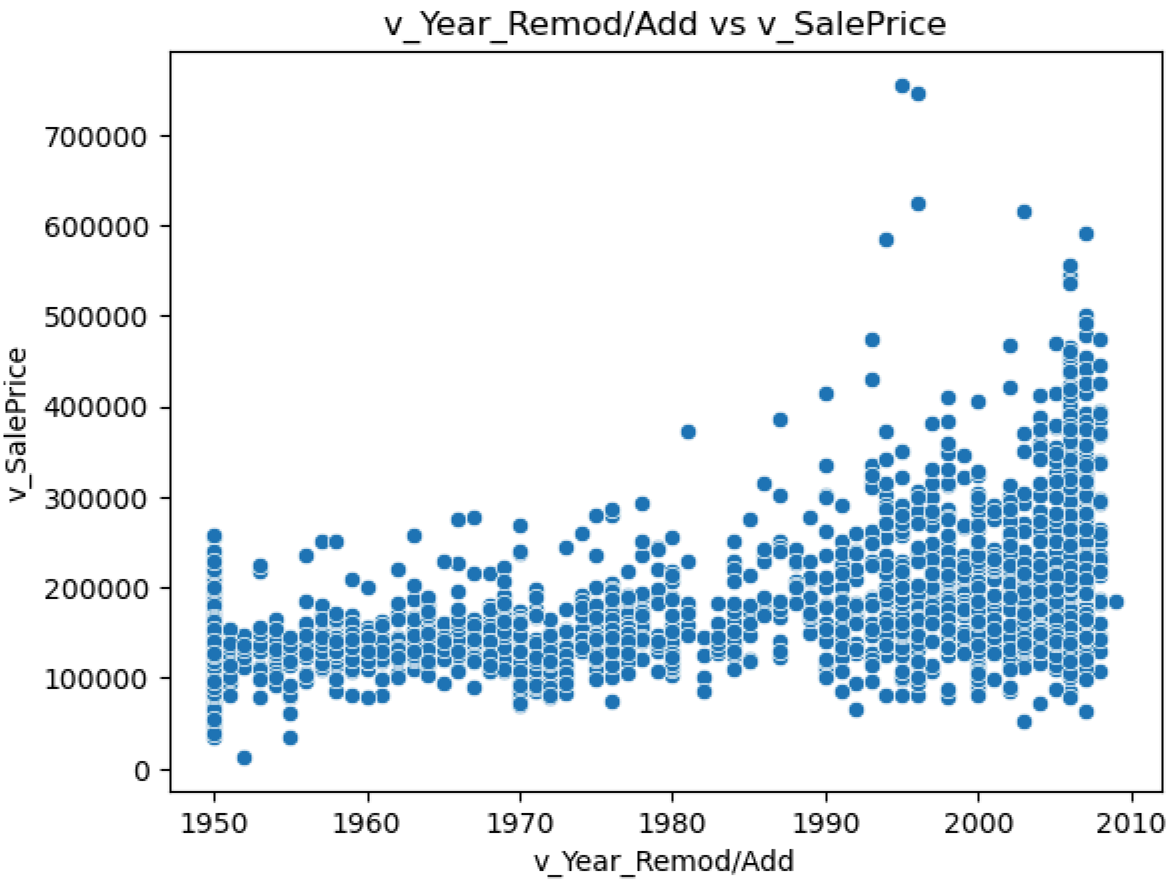

v_Overall_Qual vs v_SalePrice
v_SalePrice
v_Overall_Qual
0
100000
200000
300000
400000
500000
600000
700000
2
4
6
8
10

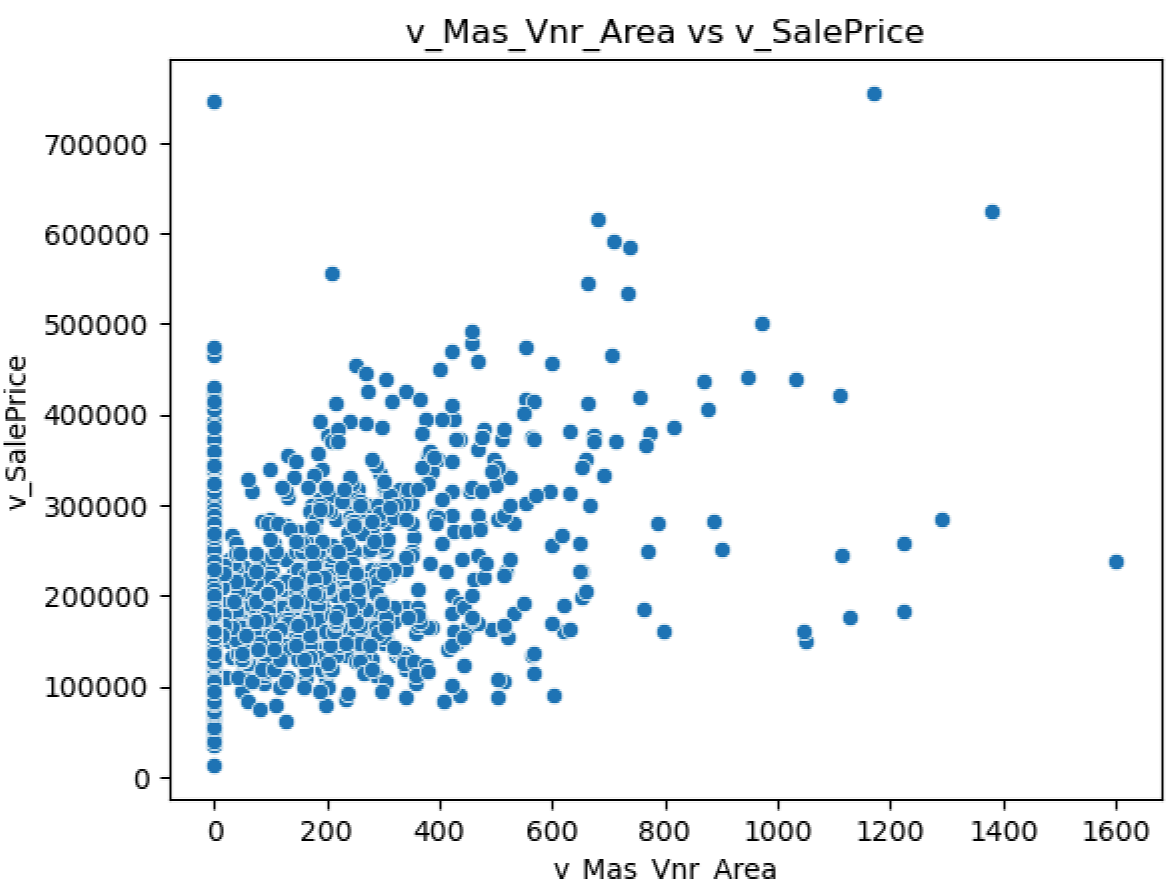


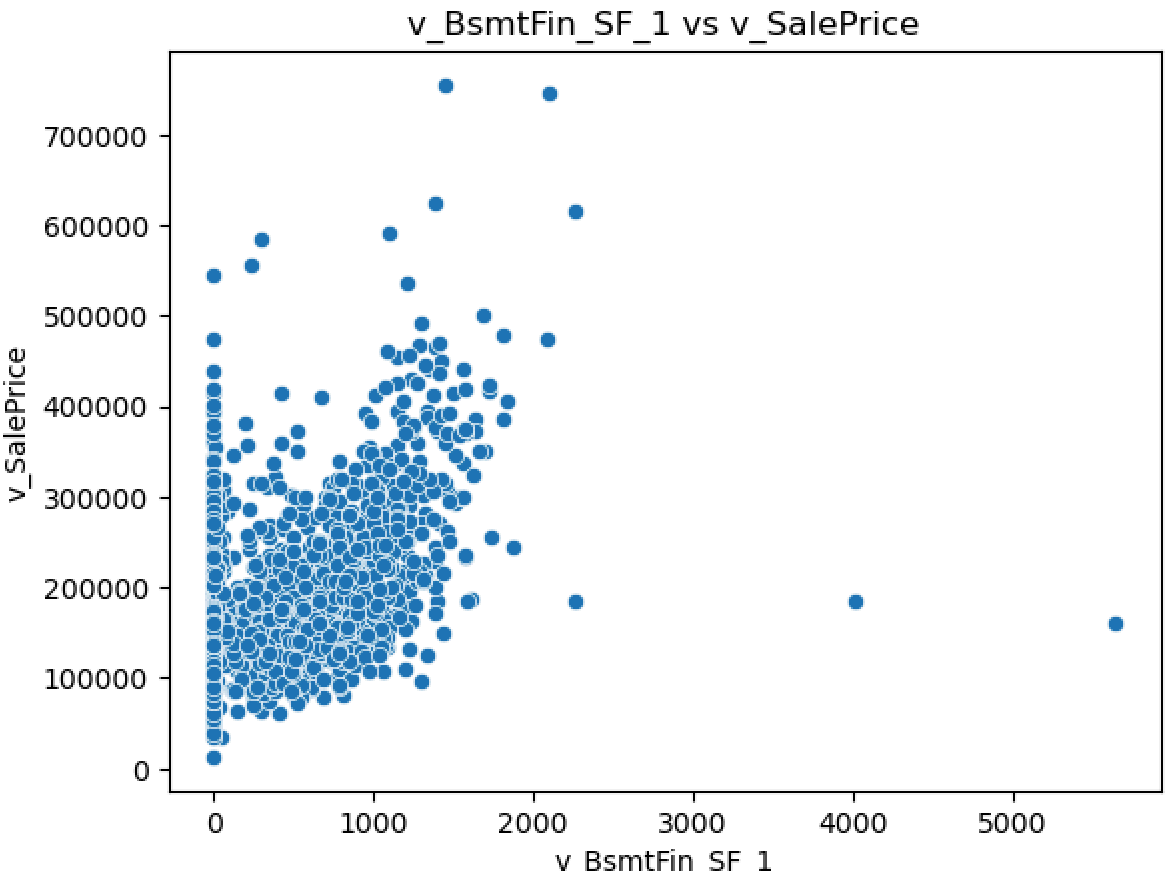
v_Overall_Cond vs v_SalePrice
v_SalePrice
v_Overall_Cond

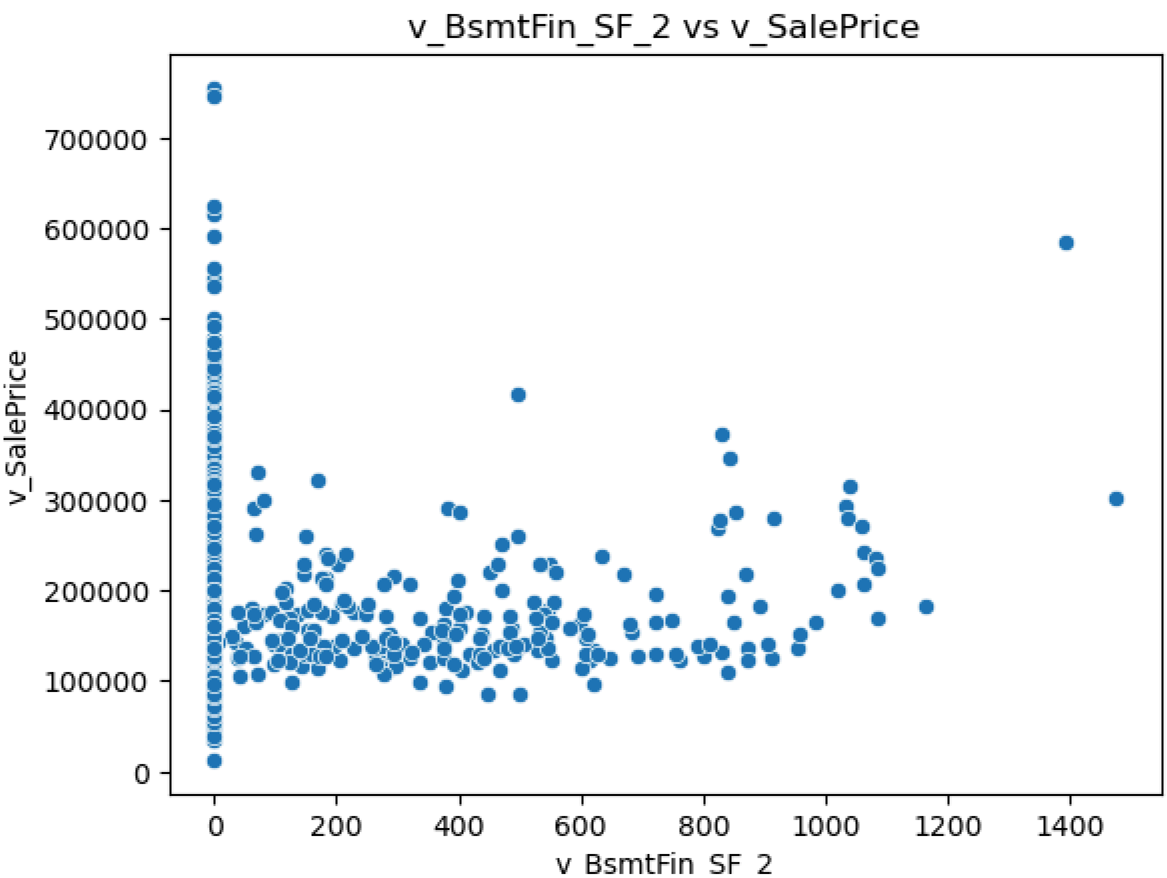


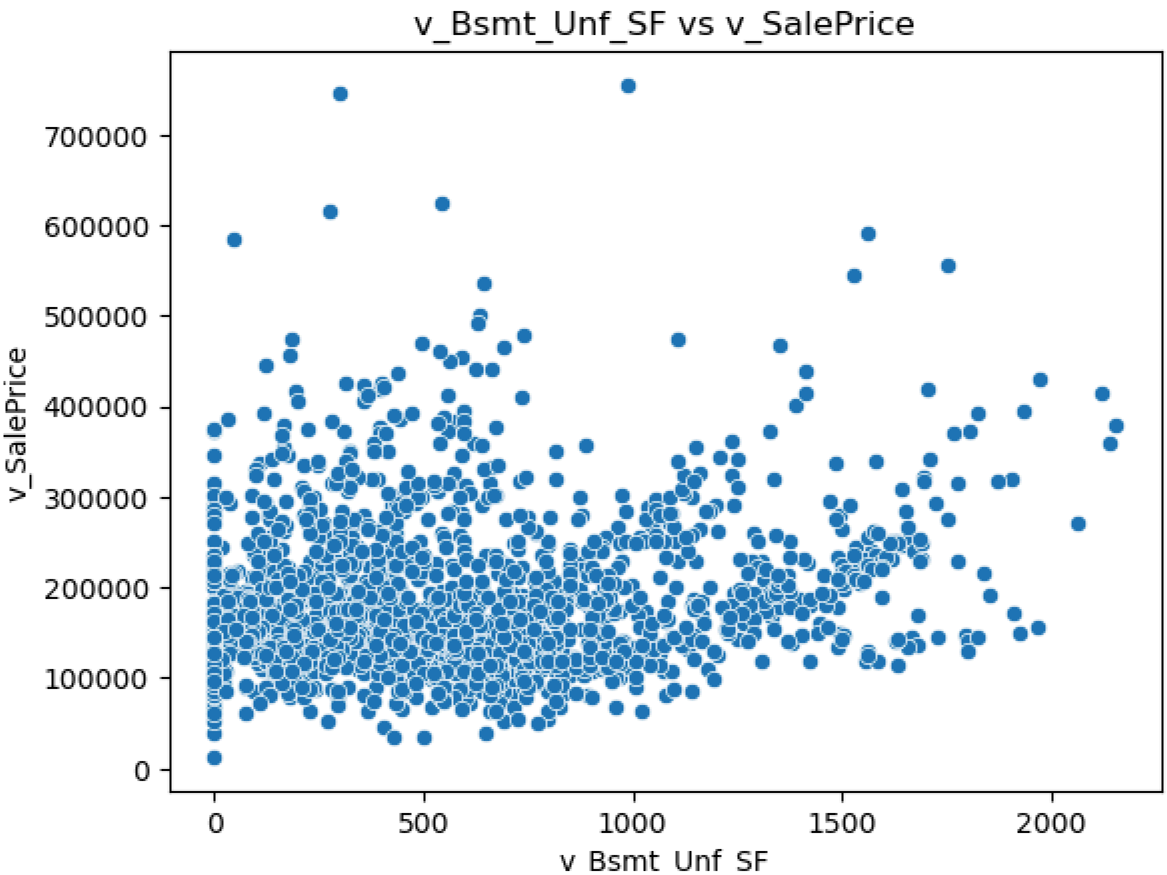
v_Year_Built vs v_SalePrice
v_SalePrice
v_Year_Built

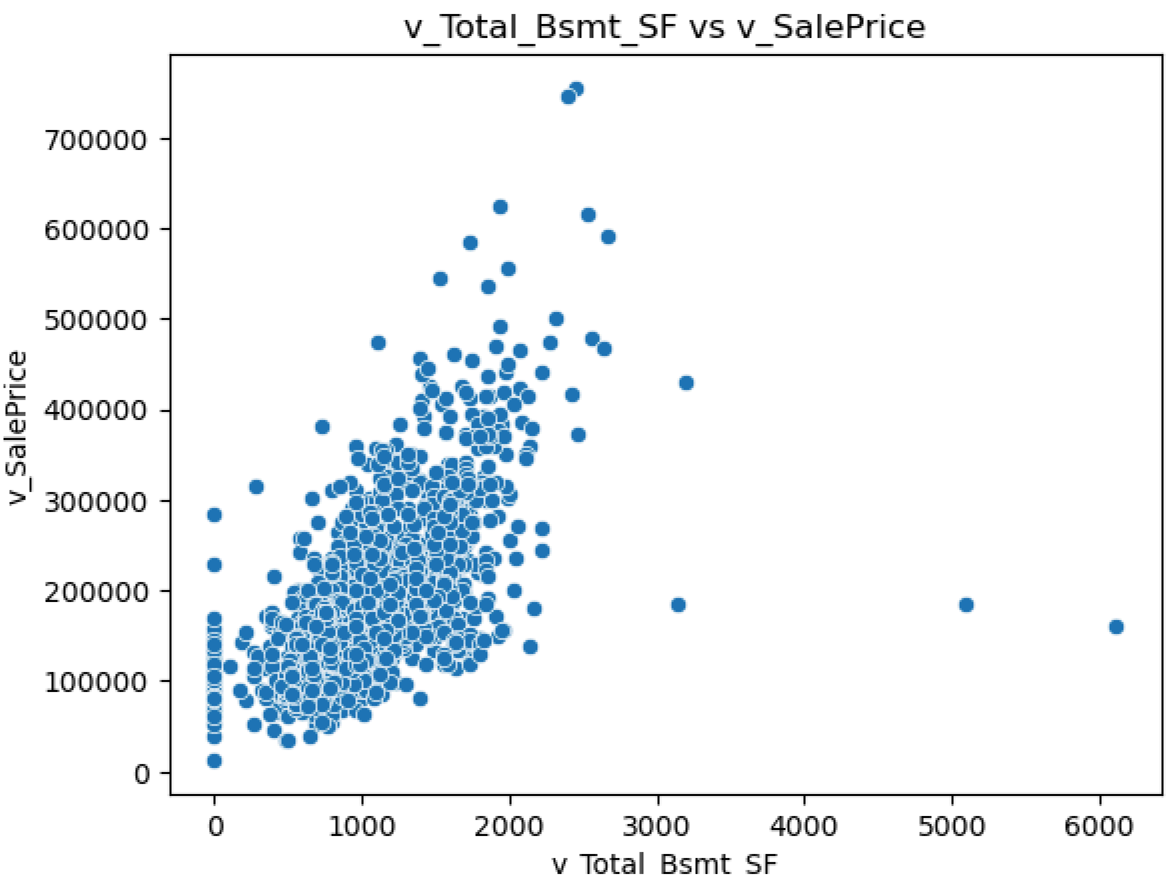

v_Year_Remod/Add vs v_SalePrice
v_SalePrice
v_Year_Remod/Add
0
100000
200000
300000
400000
500000
600000
700000
1950
1960
1970
1980
1990
2000
2010

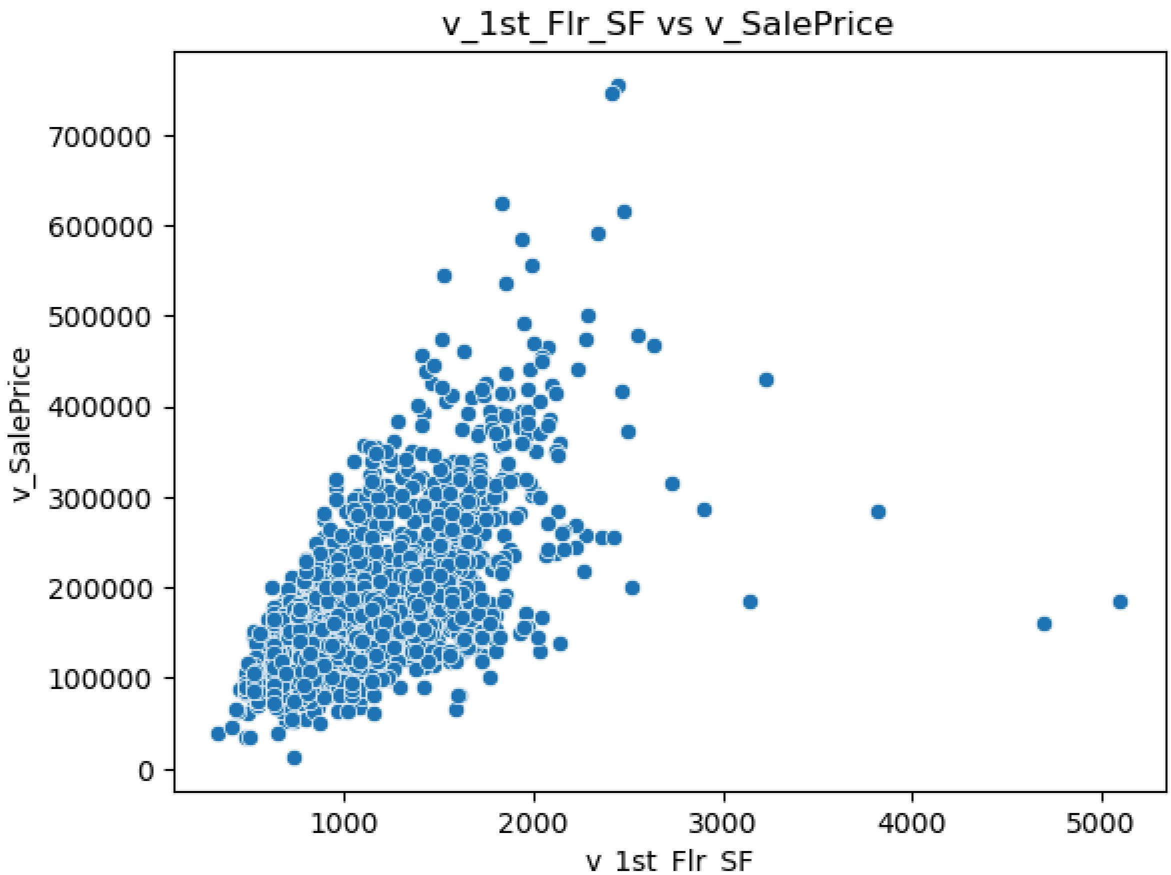

v_Mas_Vnr_Area vs v_SalePrice
v_SalePrice
v_Mas_Vnr_Area
0
100000
200000
300000
400000
500000
600000
700000
0
200
400
600
800
1000
1200
1400
1600

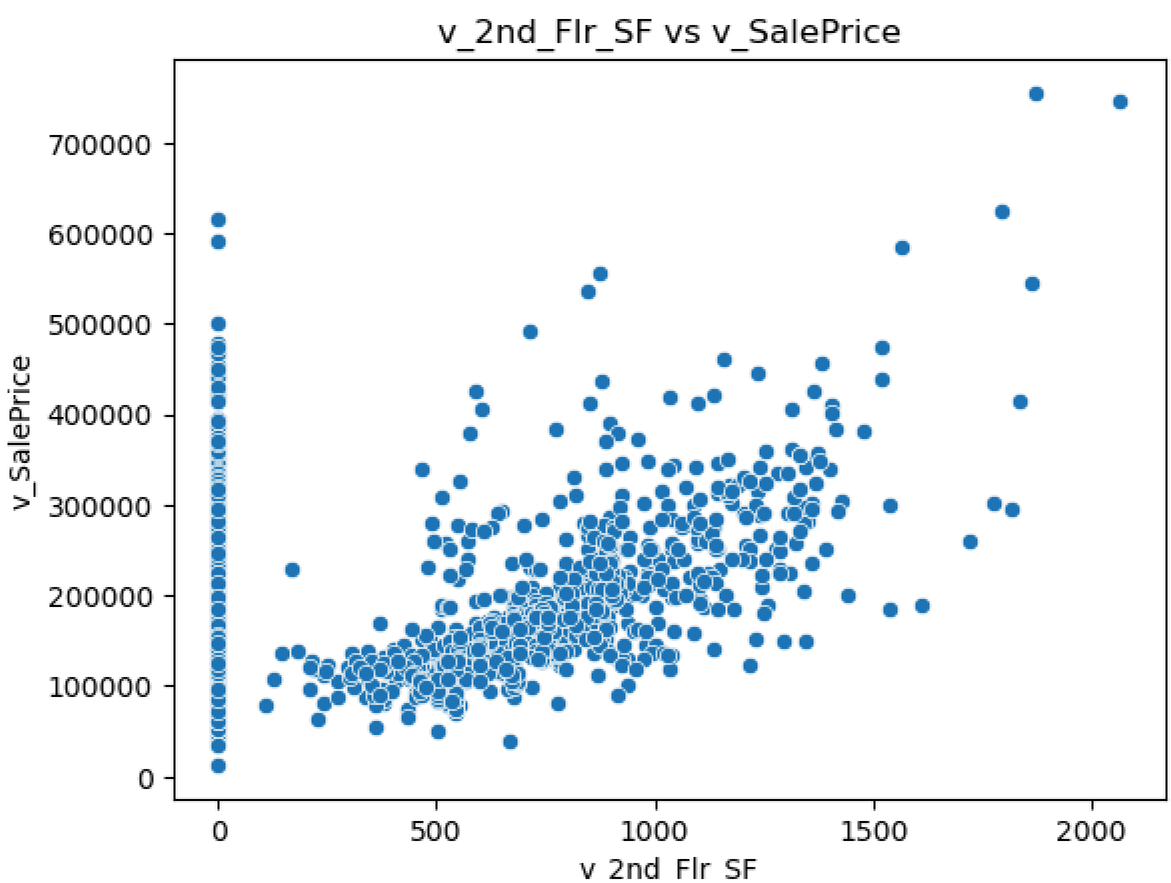

v_BsmtFin_SF_1 vs v_SalePrice
v_SalePrice
v_BsmtFin_SF_1
0
100000
200000
300000
400000
500000
600000
700000
0
1000
2000
3000
4000
5000

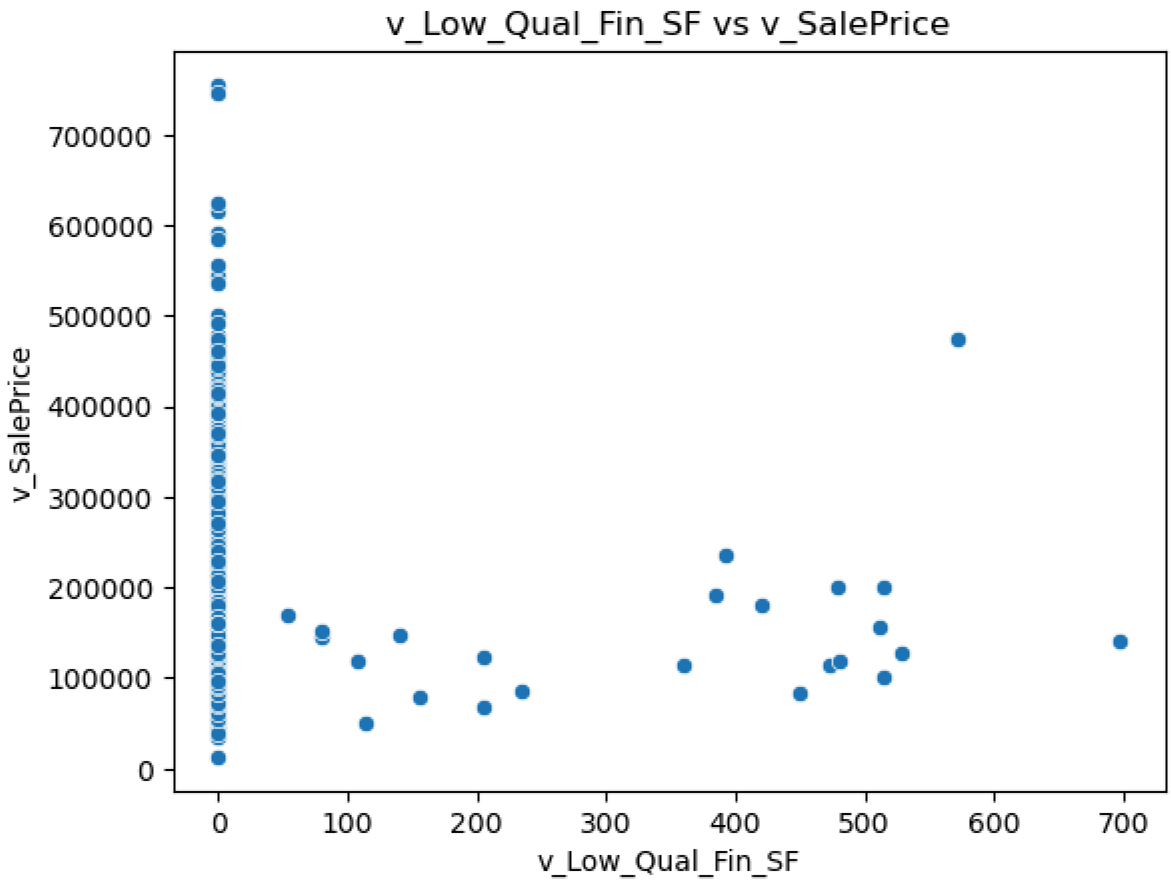

v_BsmtFin_SF_2 vs v_SalePrice
v_SalePrice
v_BsmtFin_SF_2
0
100000
200000
300000
400000
500000
600000
700000
0
200
400
600
800
1000
1200
1400

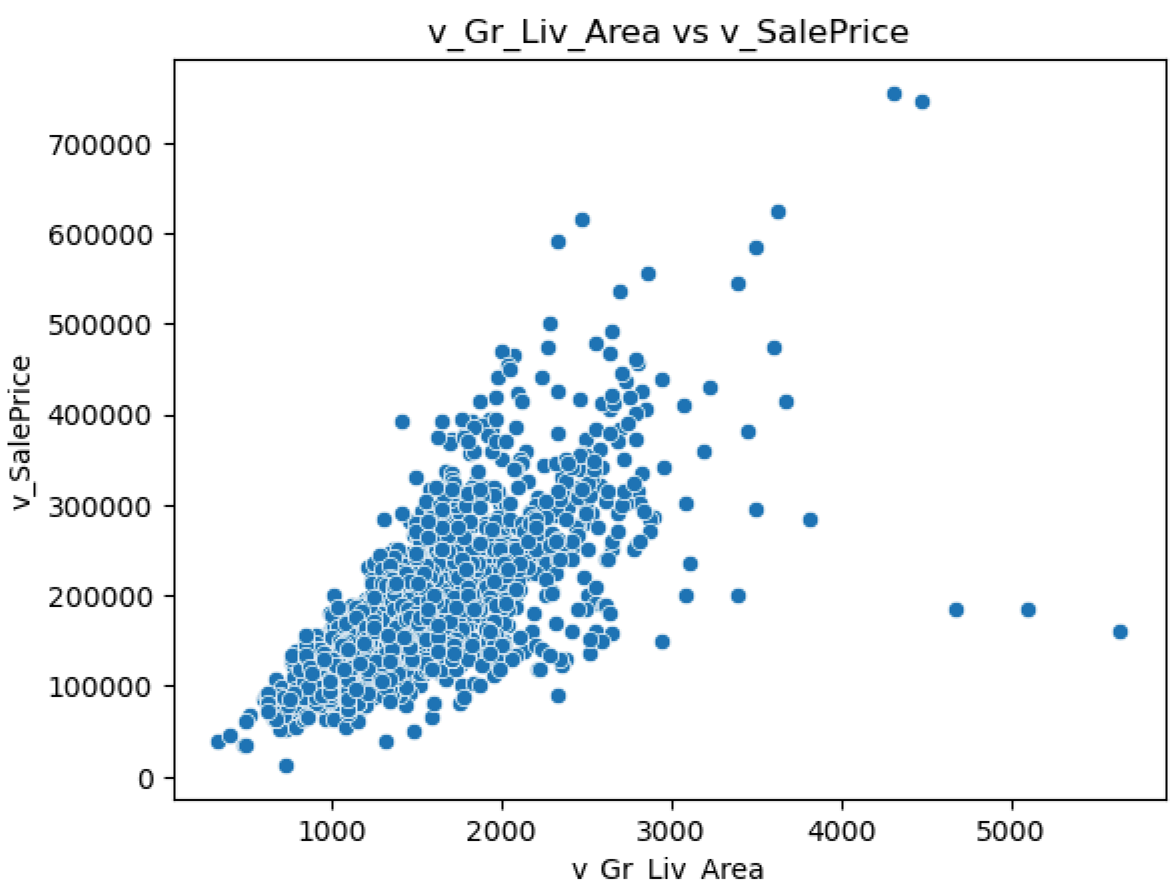

v_Bsmt_Unf_SF vs v_SalePrice
v_SalePrice
v_Bsmt_Unf_SF
0
100000
200000
300000
400000
500000
600000
700000
0
500
1000
1500
2000

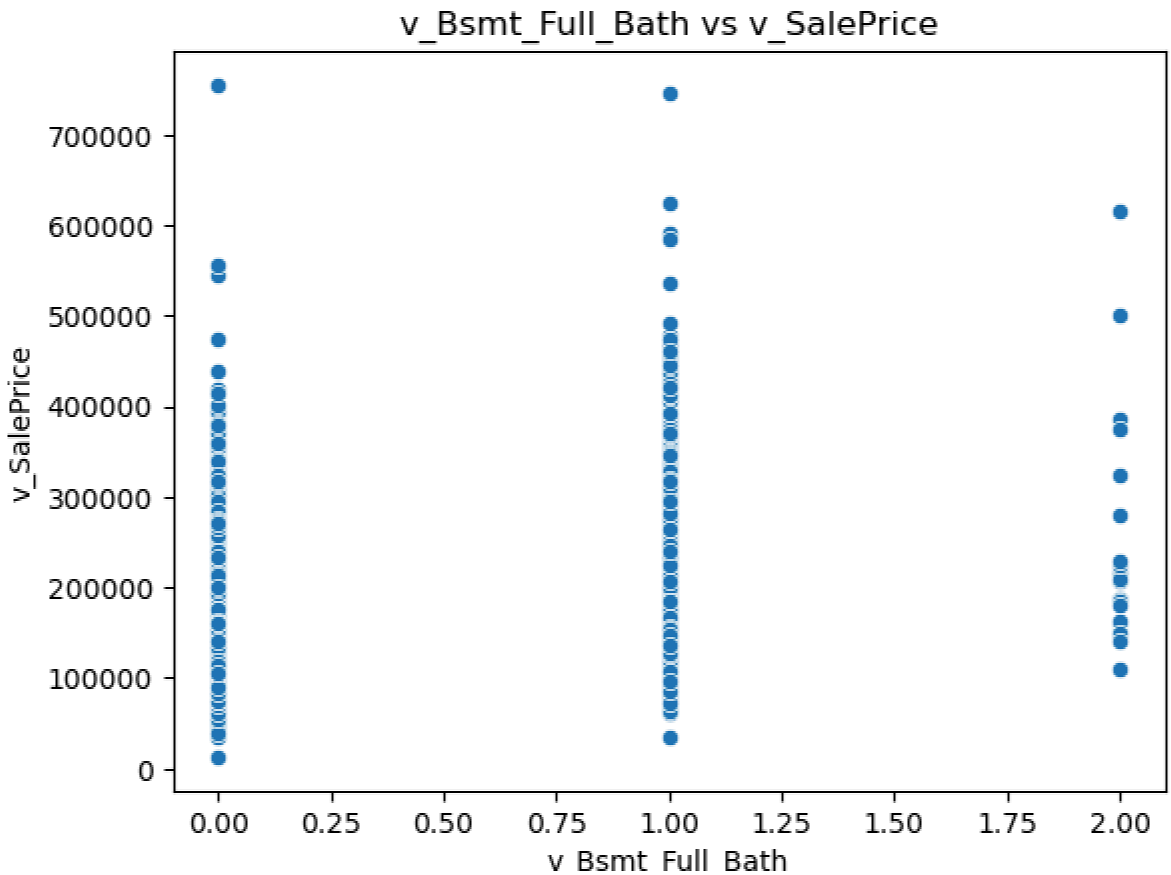

v_Total_Bsmt_SF vs v_SalePrice
v_SalePrice
v_Total_Bsmt_SF
0
100000
200000
300000
400000
500000
600000
700000
0
1000
2000
3000
4000
5000
6000

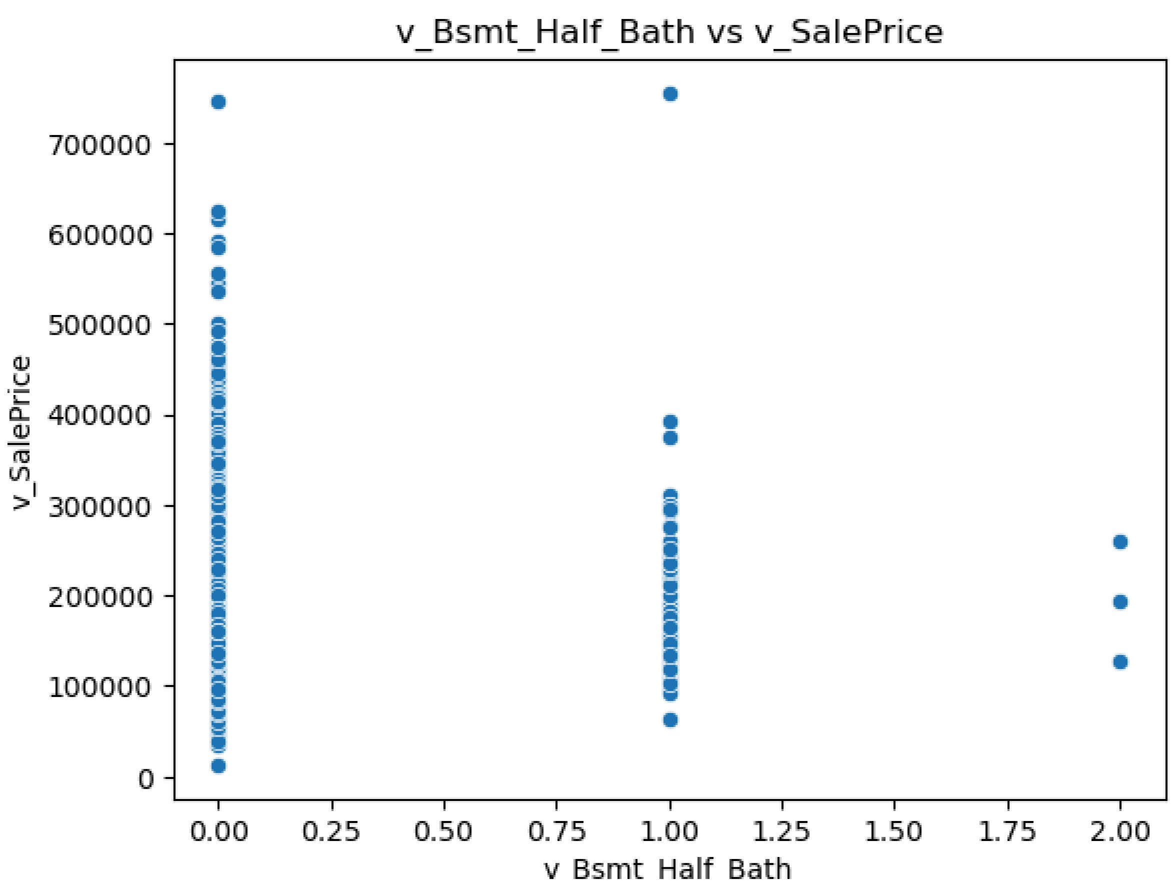

v_1st_Flr_SF vs v_SalePrice
v_SalePrice
0
100000
200000
300000
400000
500000
600000
700000
1000
2000
3000
4000
5000
v_1st_Flr_SF

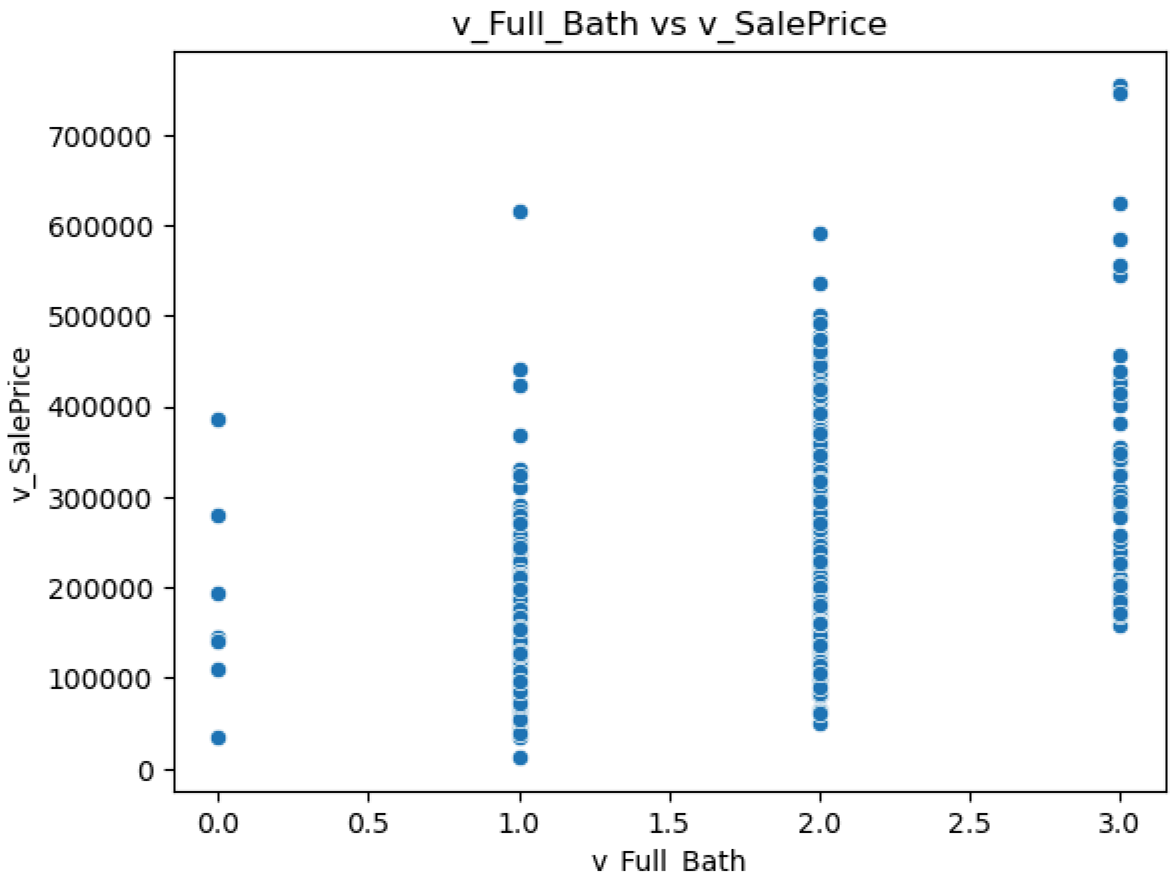

v_2nd_Flr_SF vs v_SalePrice
v_SalePrice
0
100000
200000
300000
400000
500000
600000
700000
0
500
1000
1500
2000
v_2nd_Flr_SF

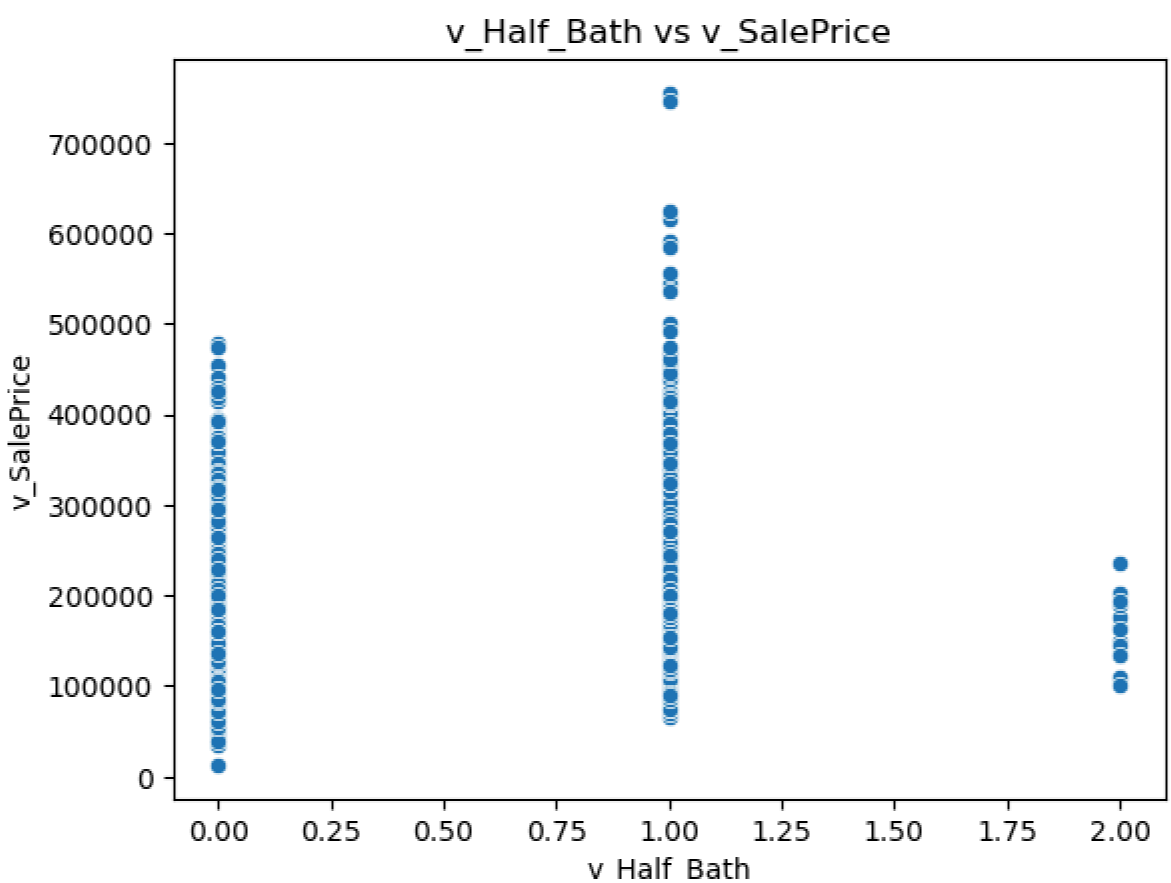

v_Low_Qual_Fin_SF vs v_SalePrice
v_SalePrice
0
100000
200000
300000
400000
500000
600000
700000
0
100
200
300
400
500
600
700
v_Low_Qual_Fin_SF

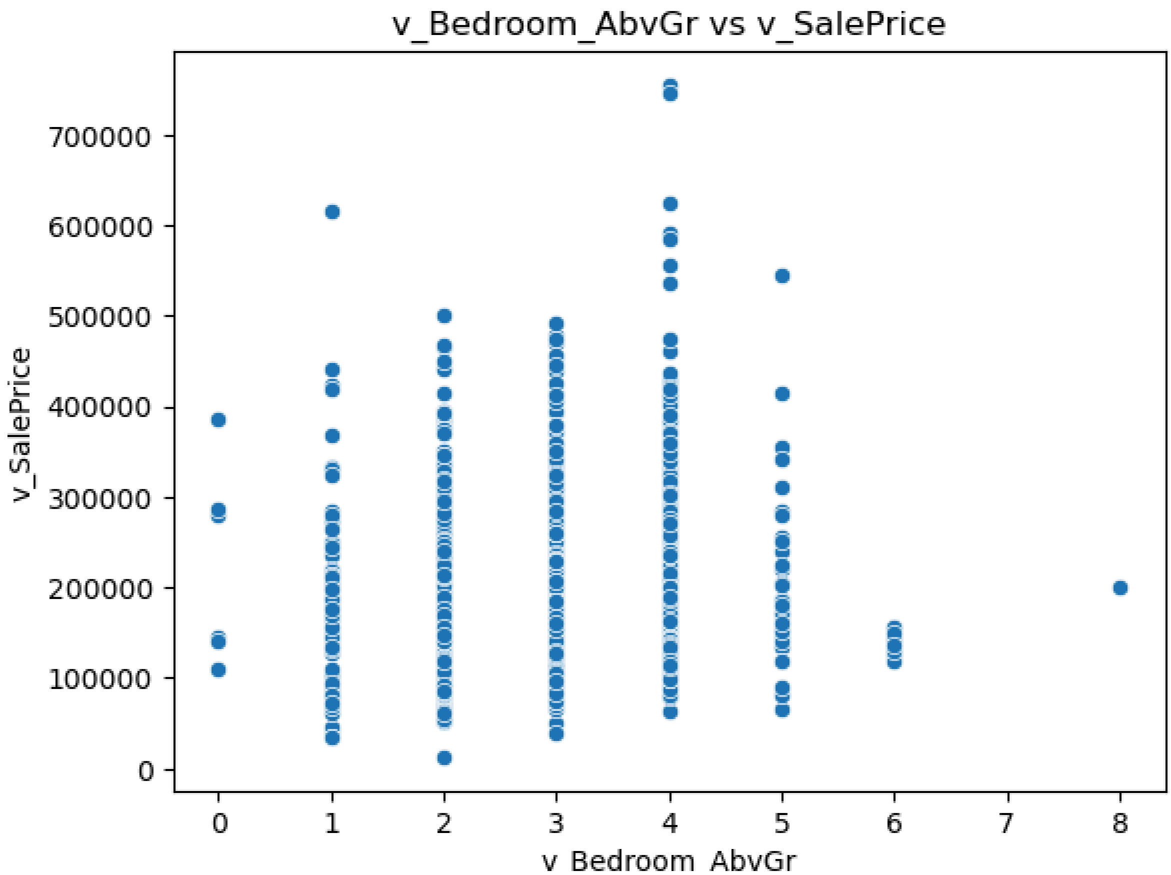

v_Gr_Liv_Area vs v_SalePrice
v_SalePrice
0
100000
200000
300000
400000
500000
600000
700000
1000
2000
3000
4000
5000
v_Gr_Liv_Area

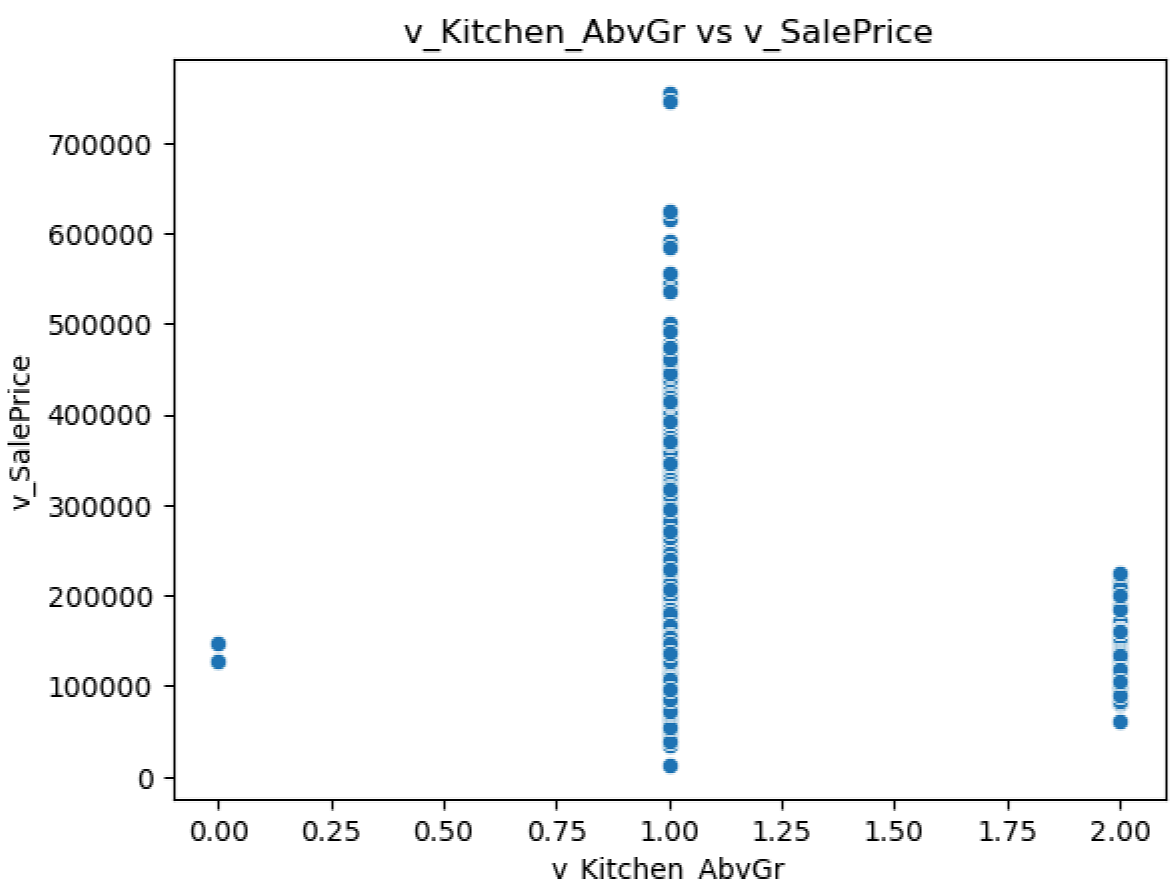

v_Bsmt_Full_Bath vs v_SalePrice
v_SalePrice
0
100000
200000
300000
400000
500000
600000
700000
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
v_Bsmt_Full_Bath

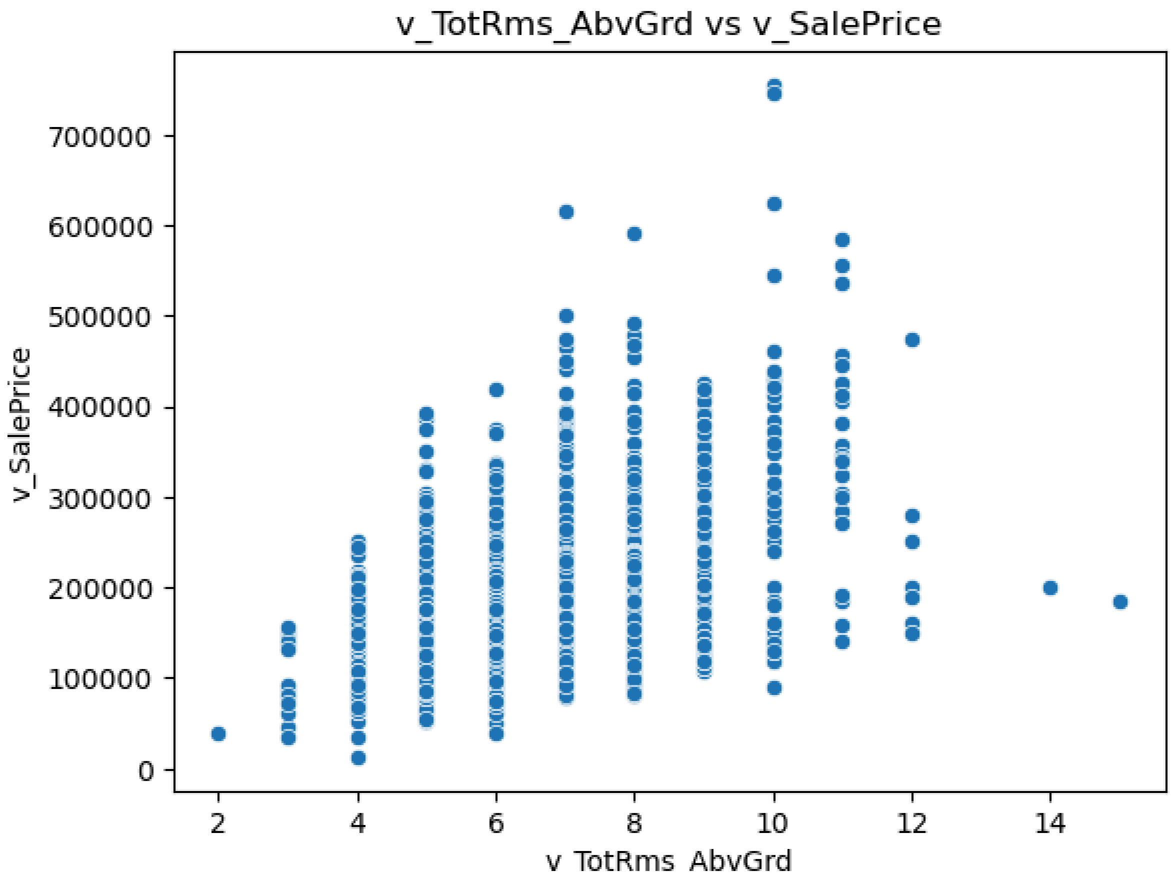

v_Bsmt_Half_Bath vs v_SalePrice
v_SalePrice
0
100000
200000
300000
400000
500000
600000
700000
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
v_Bsmt_Half_Bath

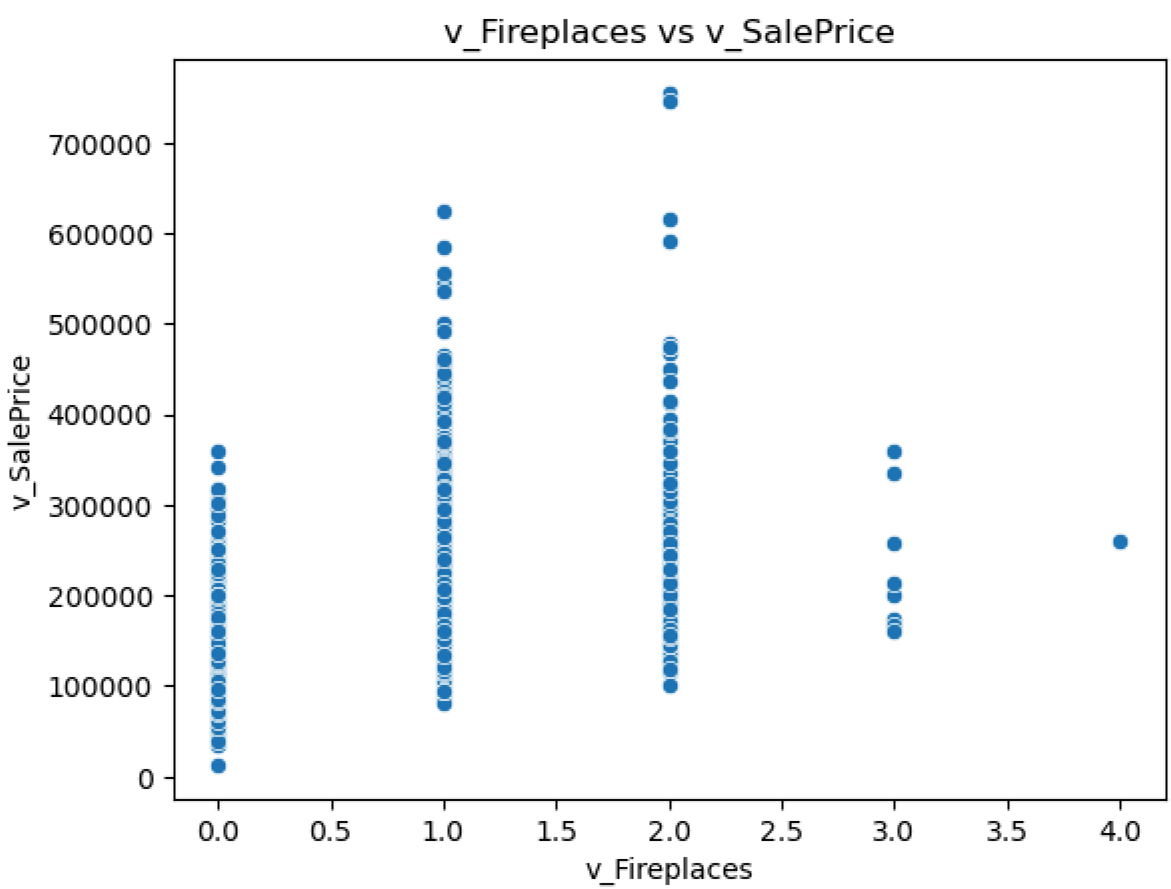

v_Full_Bath vs v_SalePrice
700000
600000
500000
400000
300000
200000
100000
0
v_SalePrice
0.0
0.5
1.0
1.5
2.0
2.5
3.0
v_Full_Bath

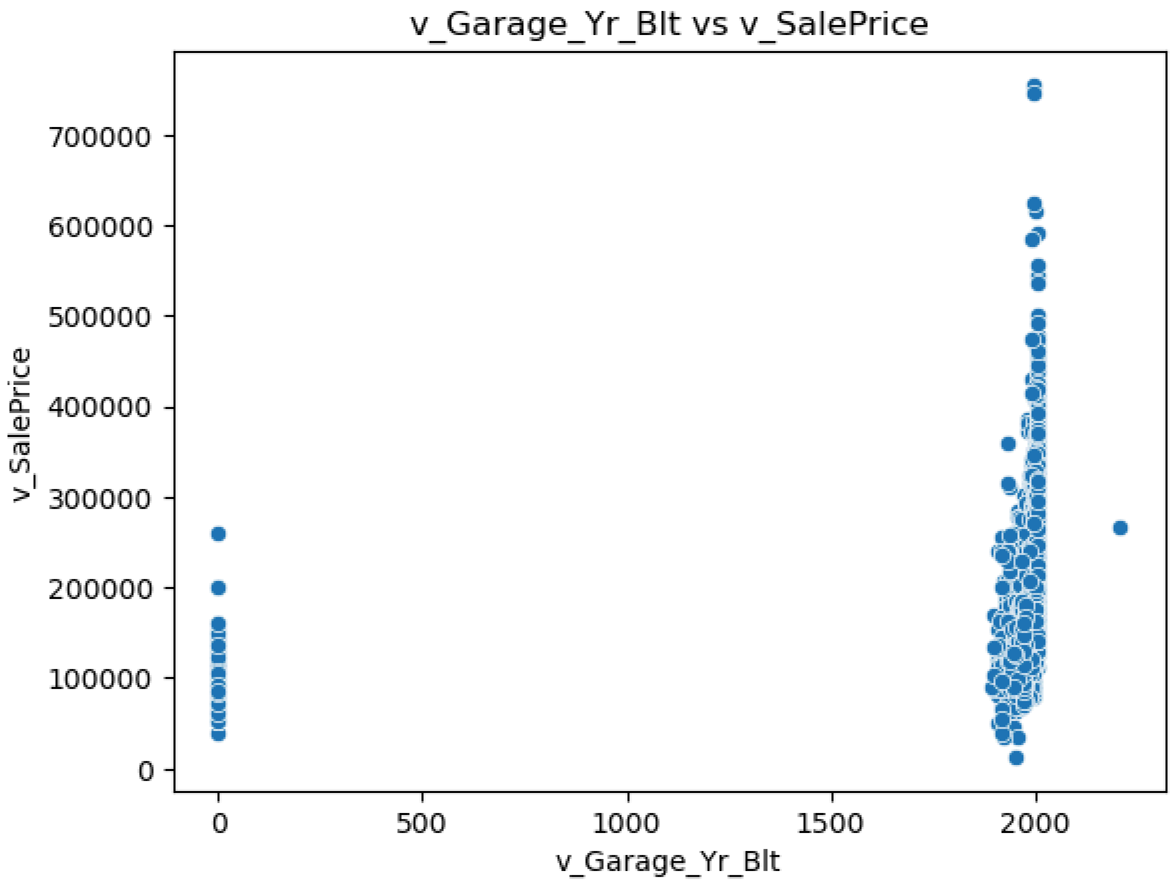

v_Half_Bath vs v_SalePrice
700000
600000
500000
400000
300000
200000
100000
0
v_SalePrice
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
v_Half_Bath

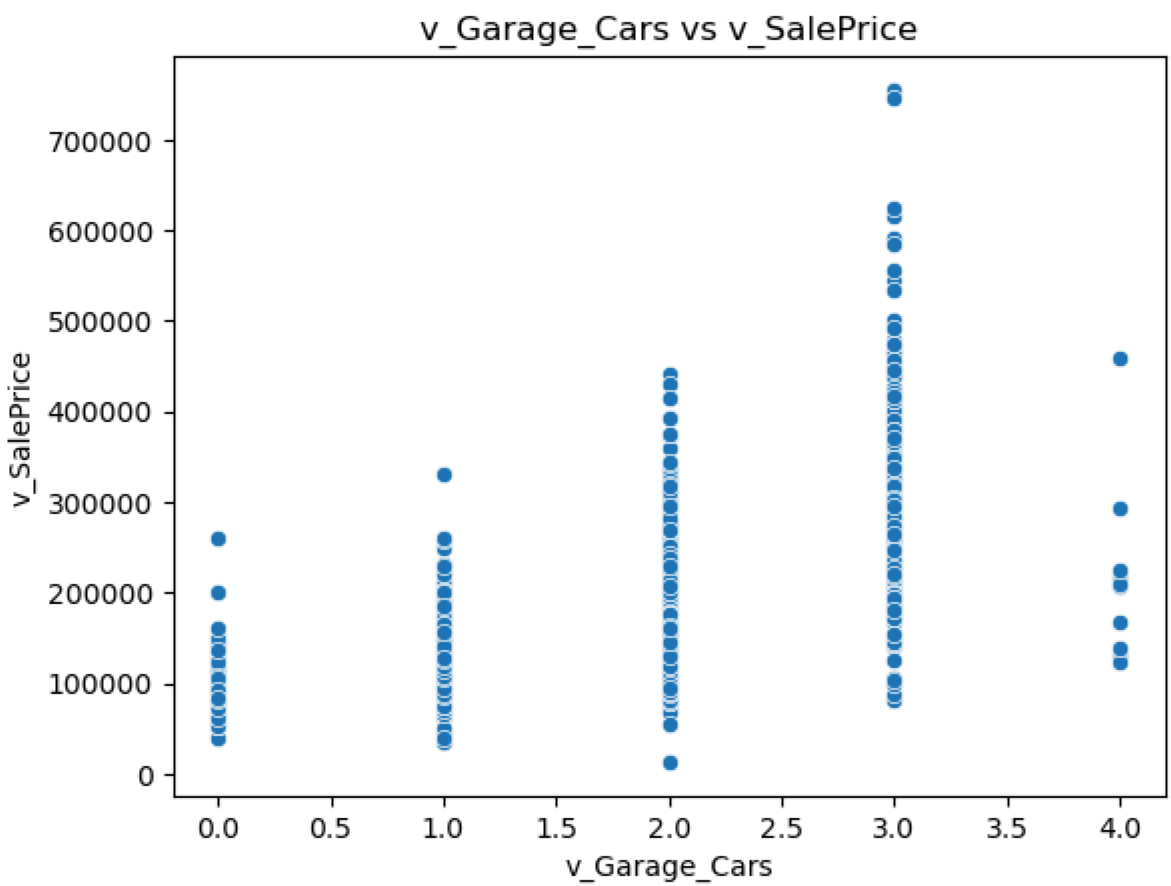

v_Bedroom_AbvGr vs v_SalePrice
v_SalePrice
0
100000
200000
300000
400000
500000
600000
700000
0
1
2
3
4
5
6
7
8
v_Bedroom_AbvGr

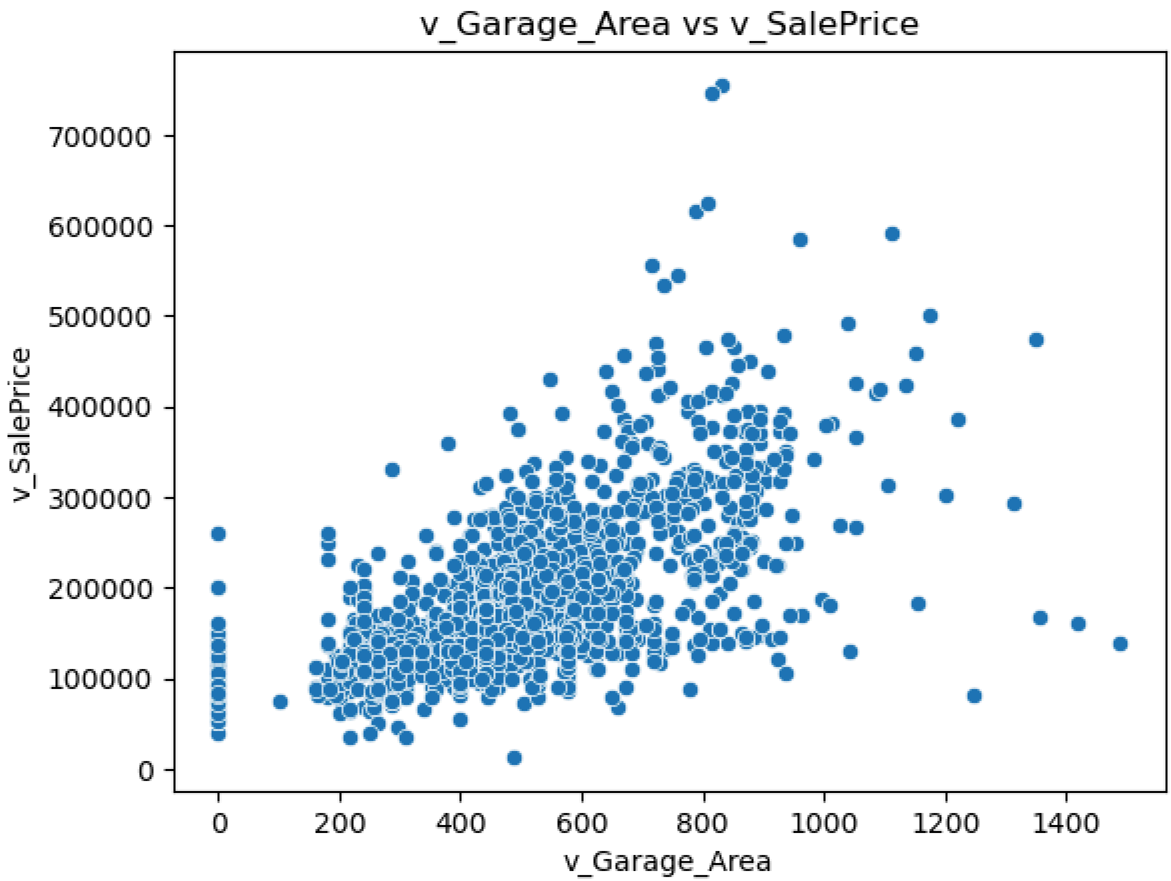

v_Kitchen_AbvGr vs v_SalePrice
v_SalePrice
0
100000
200000
300000
400000
500000
600000
700000
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
v_Kitchen_AbvGr

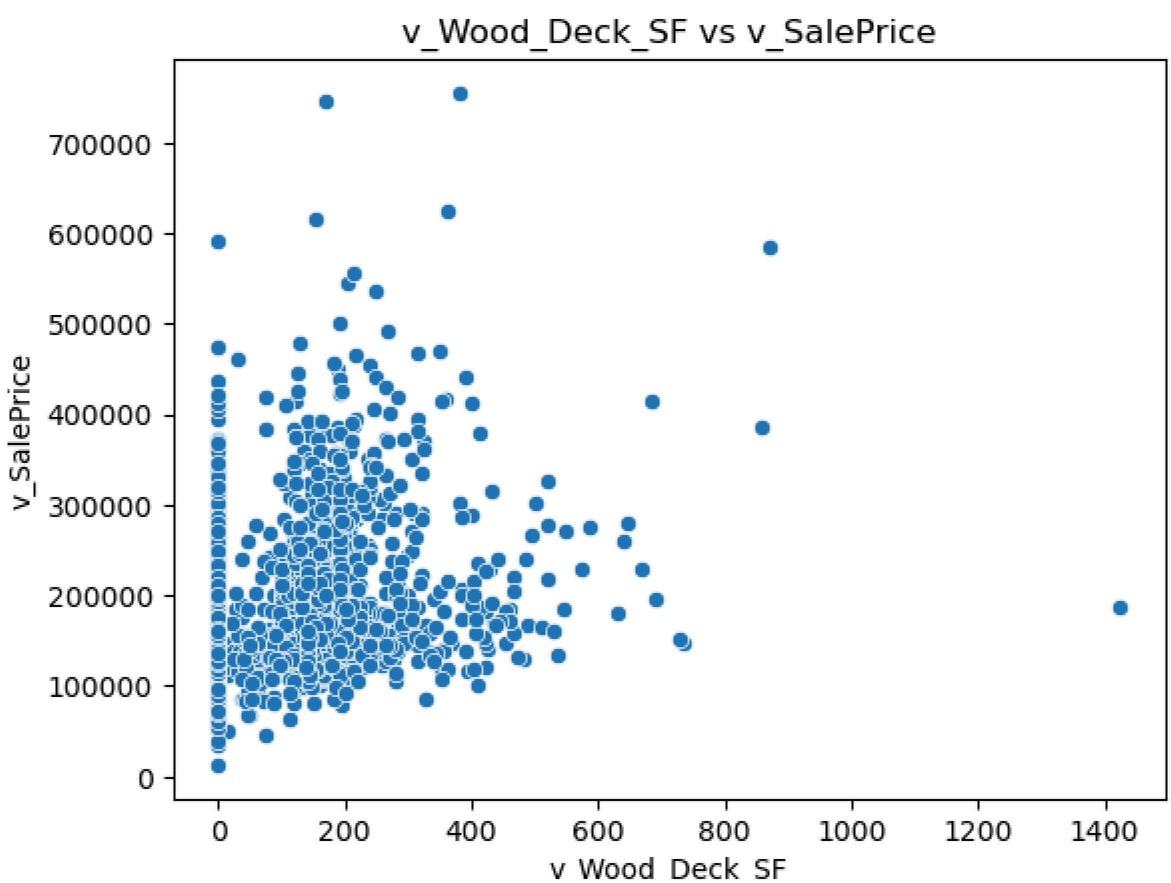

v_TotRms_AbvGrd vs v_SalePrice
v_SalePrice
v_TotRms_AbvGrd
0
100000
200000
300000
400000
500000
600000
700000
2
4
6
8
10
12
14

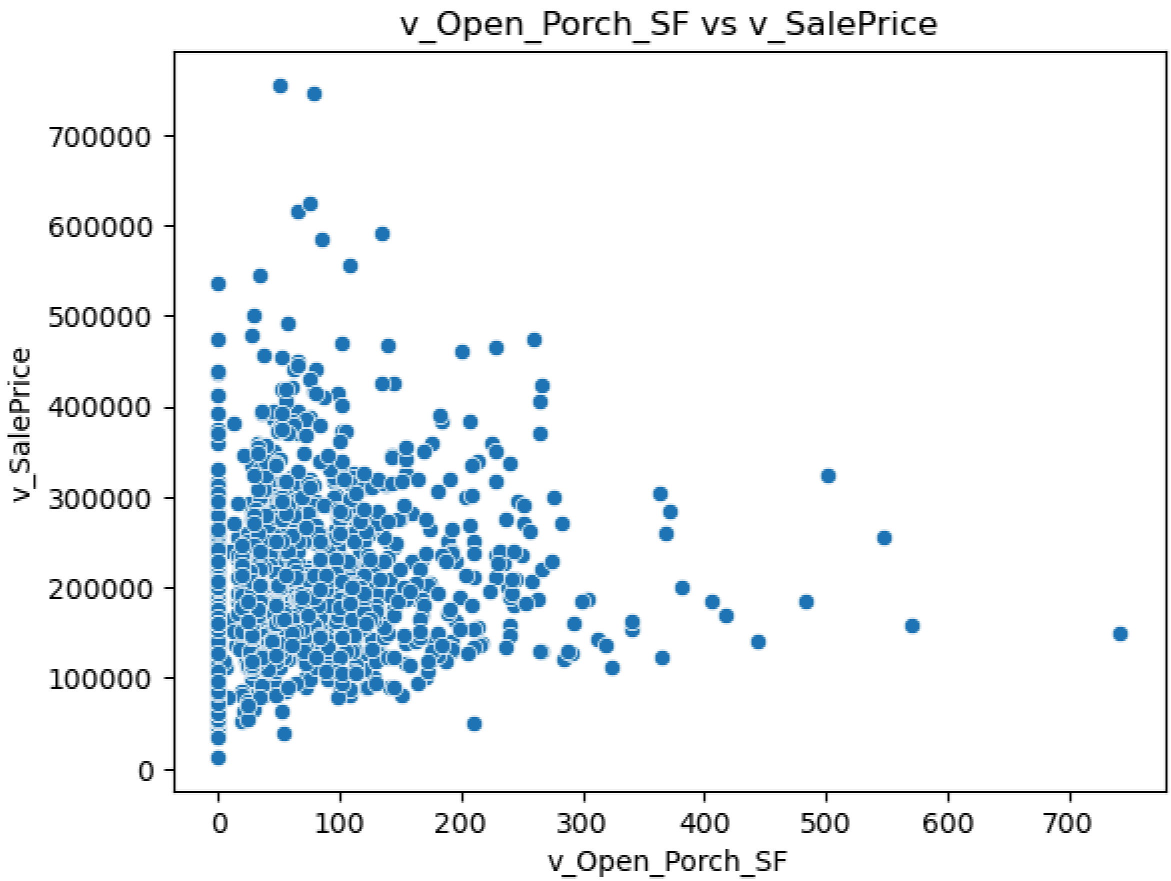

v_Fireplaces vs v_SalePrice
v_SalePrice
v_Fireplaces
0
100000
200000
300000
400000
500000
600000
700000
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0


v_Garage_Yr_Blt vs v_SalePrice
v_SalePrice
v_Garage_Yr_Blt
0
100000
200000
300000
400000
500000
600000
700000
0
500
1000
1500
2000

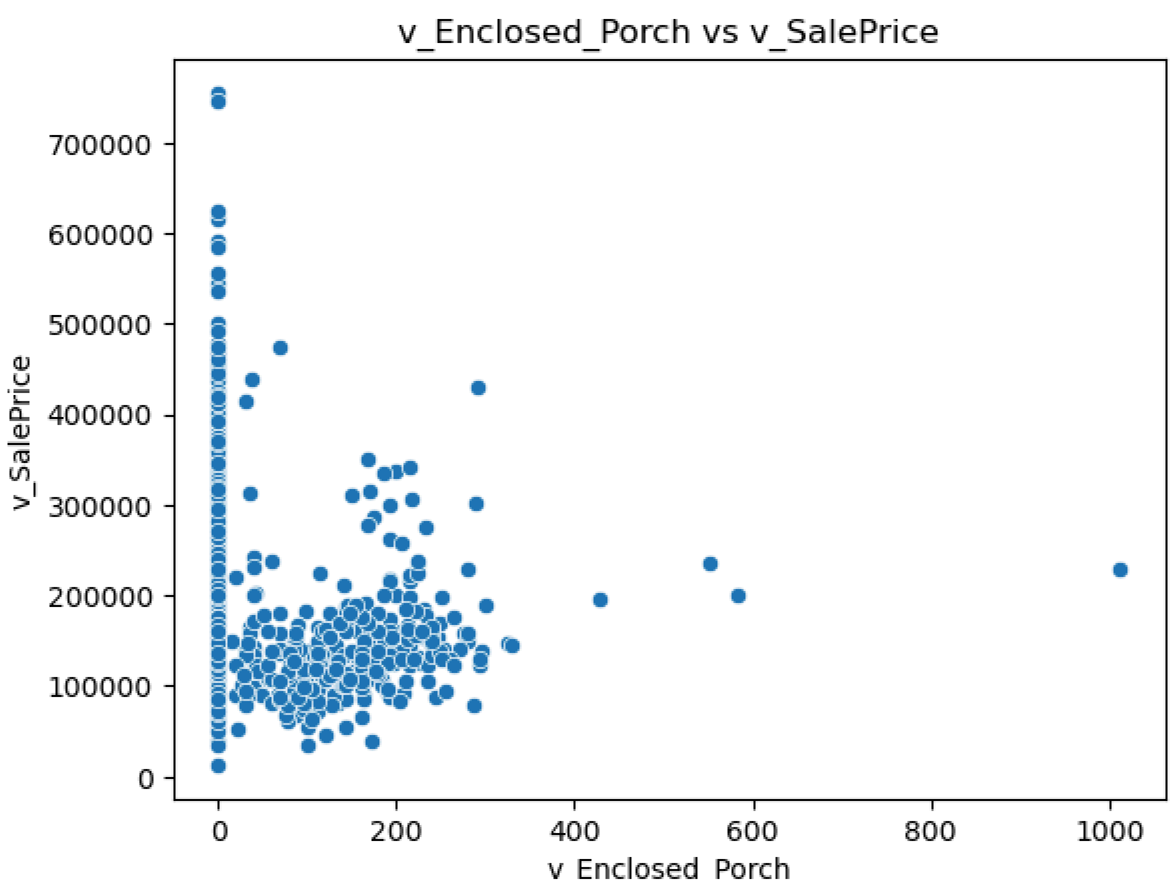

v_Garage_Cars vs v_SalePrice
v_SalePrice
v_Garage_Cars
0
100000
200000
300000
400000
500000
600000
700000
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0


v_Garage_Area vs v_SalePrice
v_SalePrice
0
100000
200000
300000
400000
500000
600000
700000
0
200
400
600
800
1000
1200
1400
v_Garage_Area

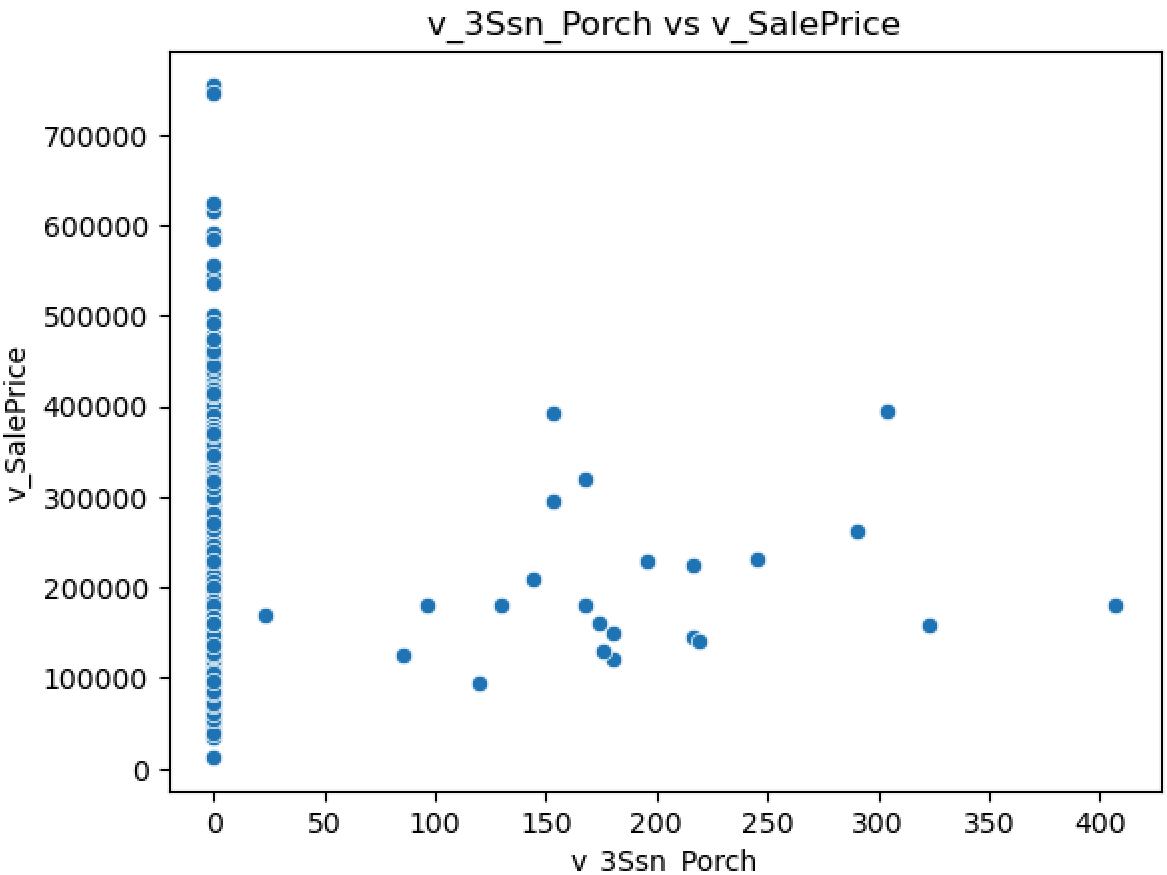

v_Wood_Deck_SF vs v_SalePrice
v_SalePrice
0
100000
200000
300000
400000
500000
600000
700000
0
200
400
600
800
1000
1200
1400
v_Wood_Deck_SF



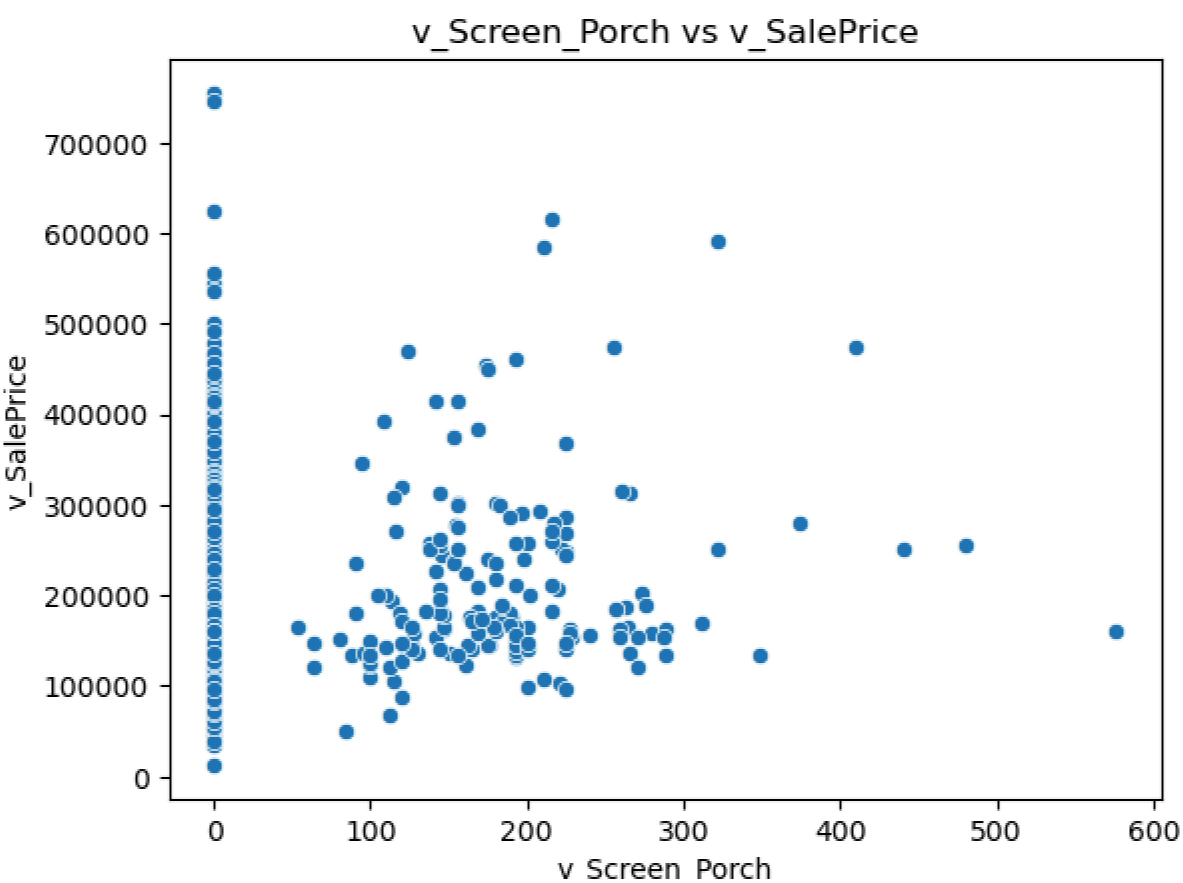
v_Open_Porch_SF vs v_SalePrice
v_SalePrice
0
100000
200000
300000
400000
500000
600000
700000
0
100
200
300
400
500
600
700
v_Open_Porch_SF



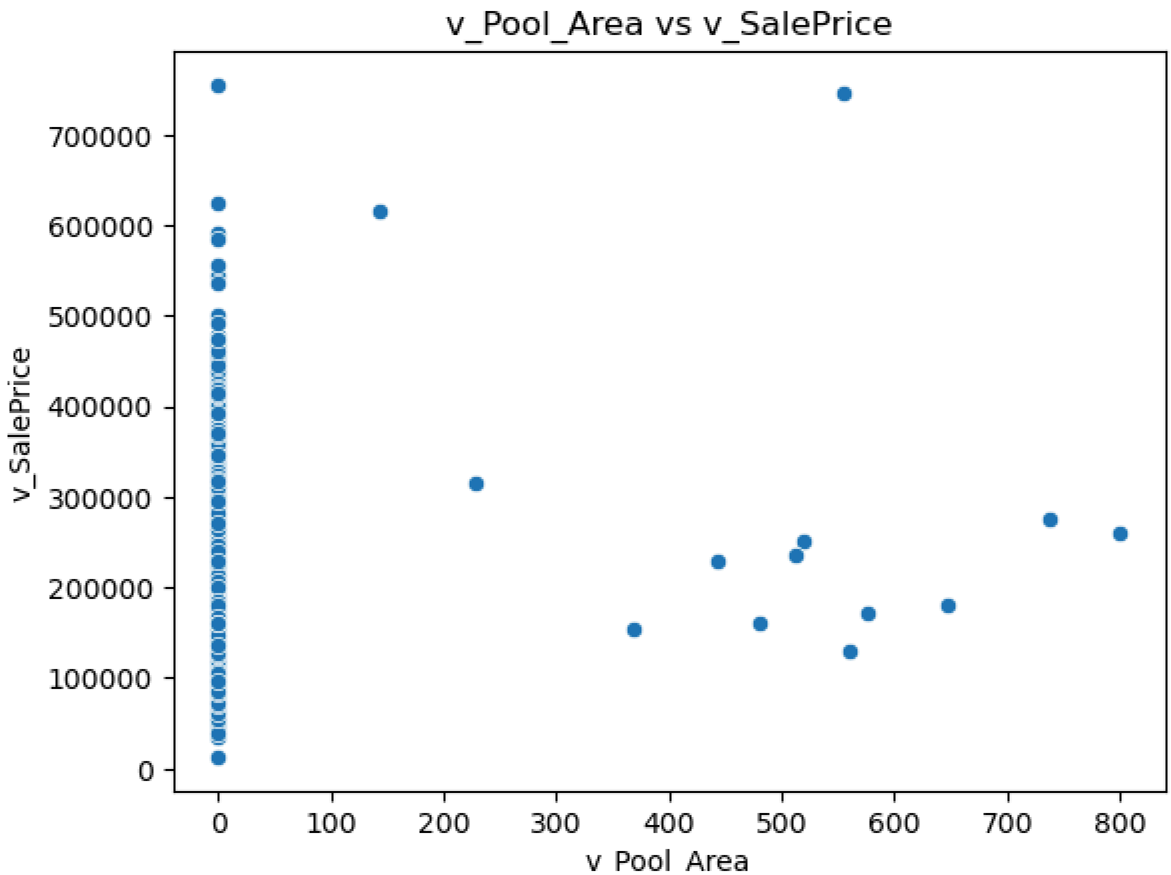
v_Enclosed_Porch vs v_SalePrice
v_SalePrice
0
100000
200000
300000
400000
500000
600000
700000
0
200
400
600
800
1000
v_Enclosed_Porch

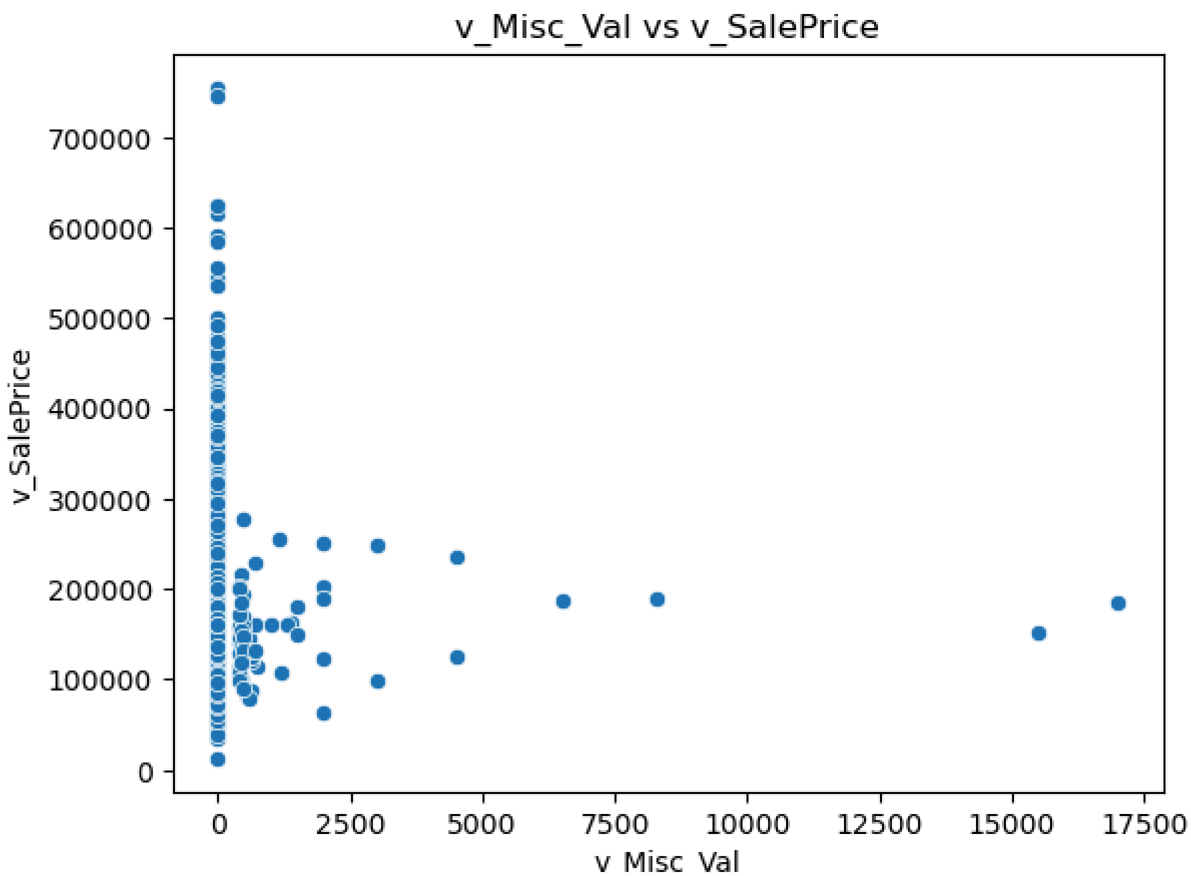


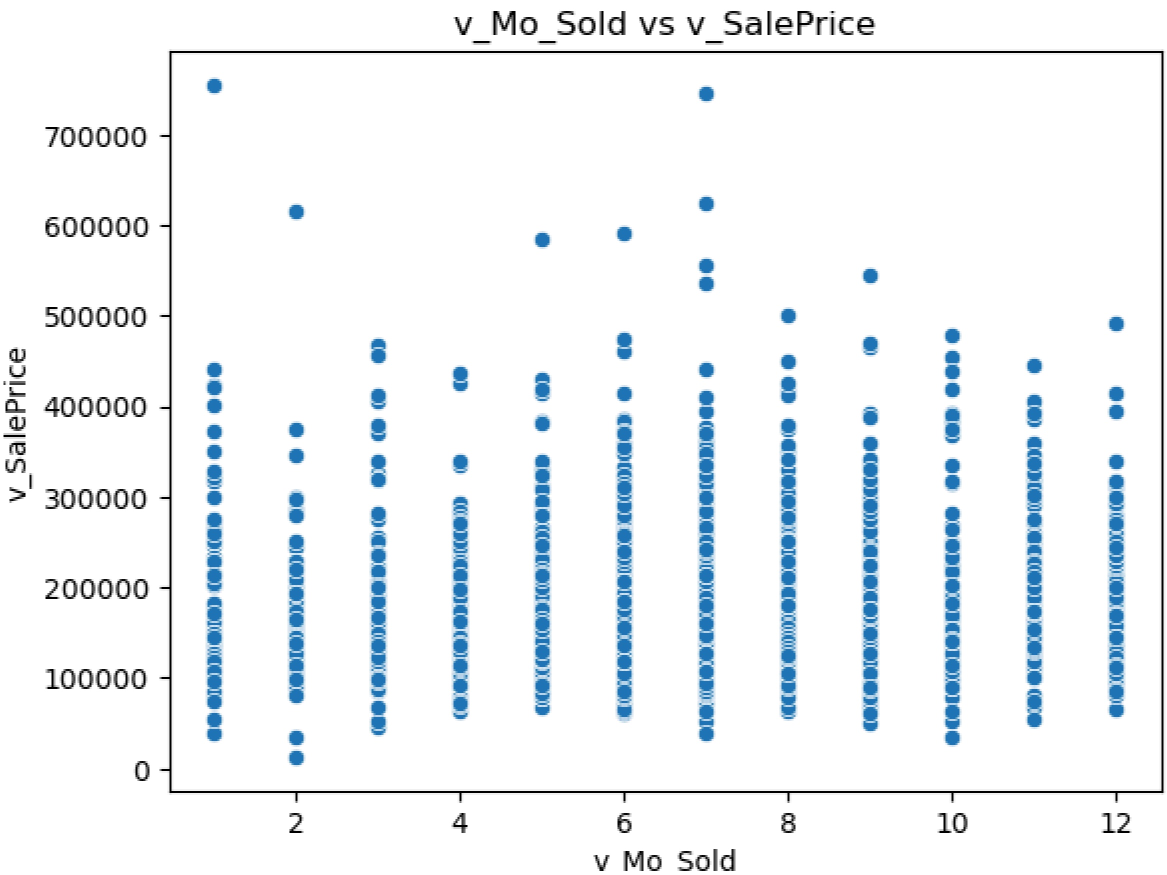
v_3Ssn_Porch vs v_SalePrice
v_SalePrice
v_3Ssn_Porch
0
100000
200000
300000
400000
500000
600000
700000
0
50
100
150
200
250
300
350
400

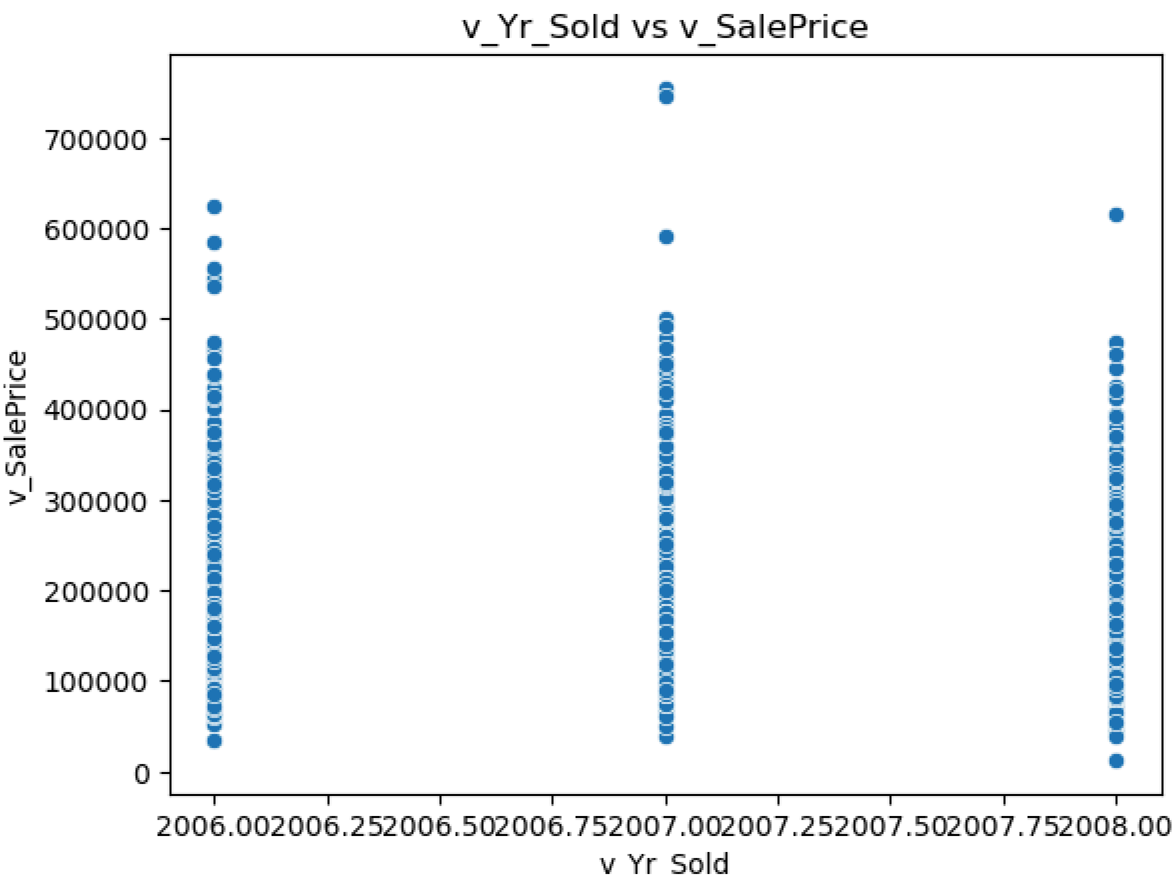


v_Screen_Porch vs v_SalePrice
v_SalePrice
v_Screen_Porch
0
100000
200000
300000
400000
500000
600000
700000
0
100
200
300
400
500
600

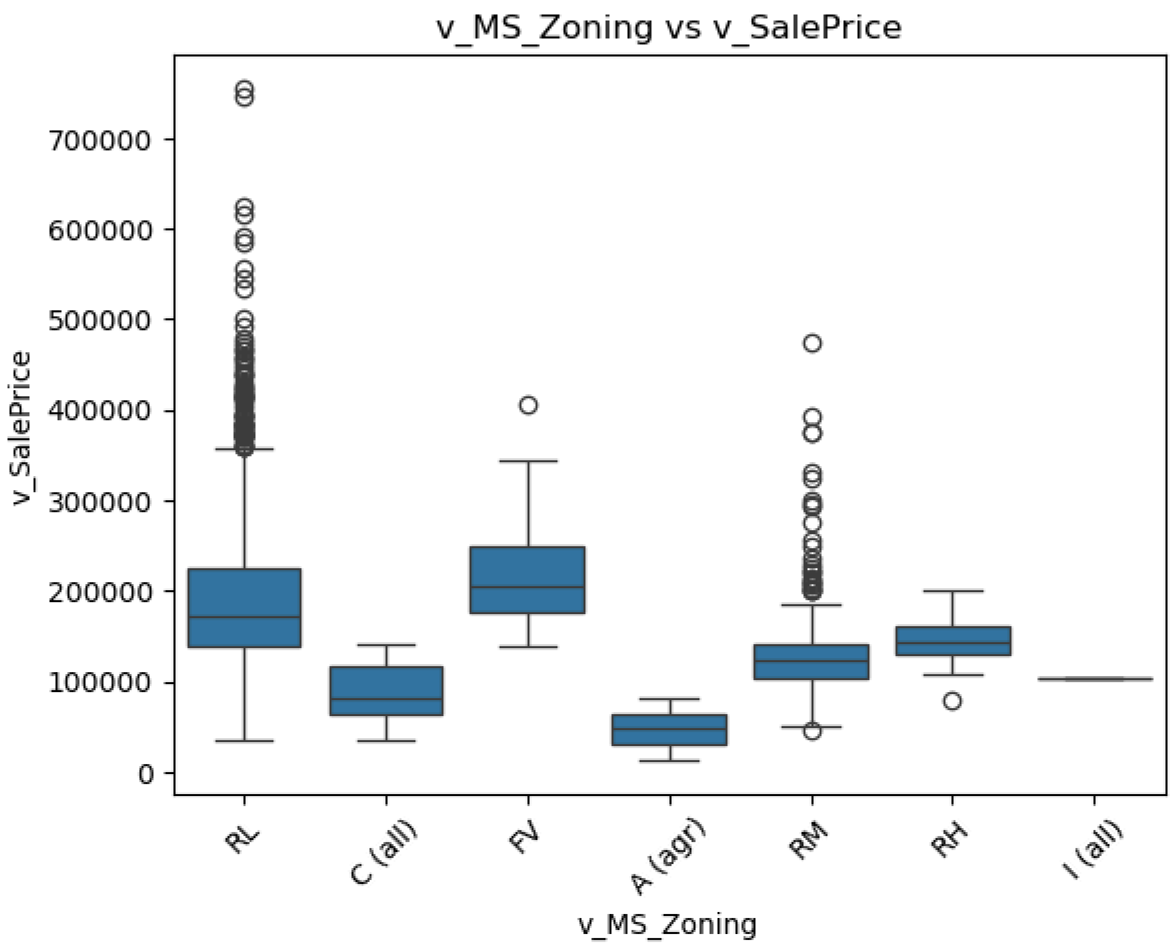

v_Pool_Area vs v_SalePrice
v_SalePrice
v_Pool_Area
0
100000
200000
300000
400000
500000
600000
700000
0
100
200
300
400
500
600
700
800

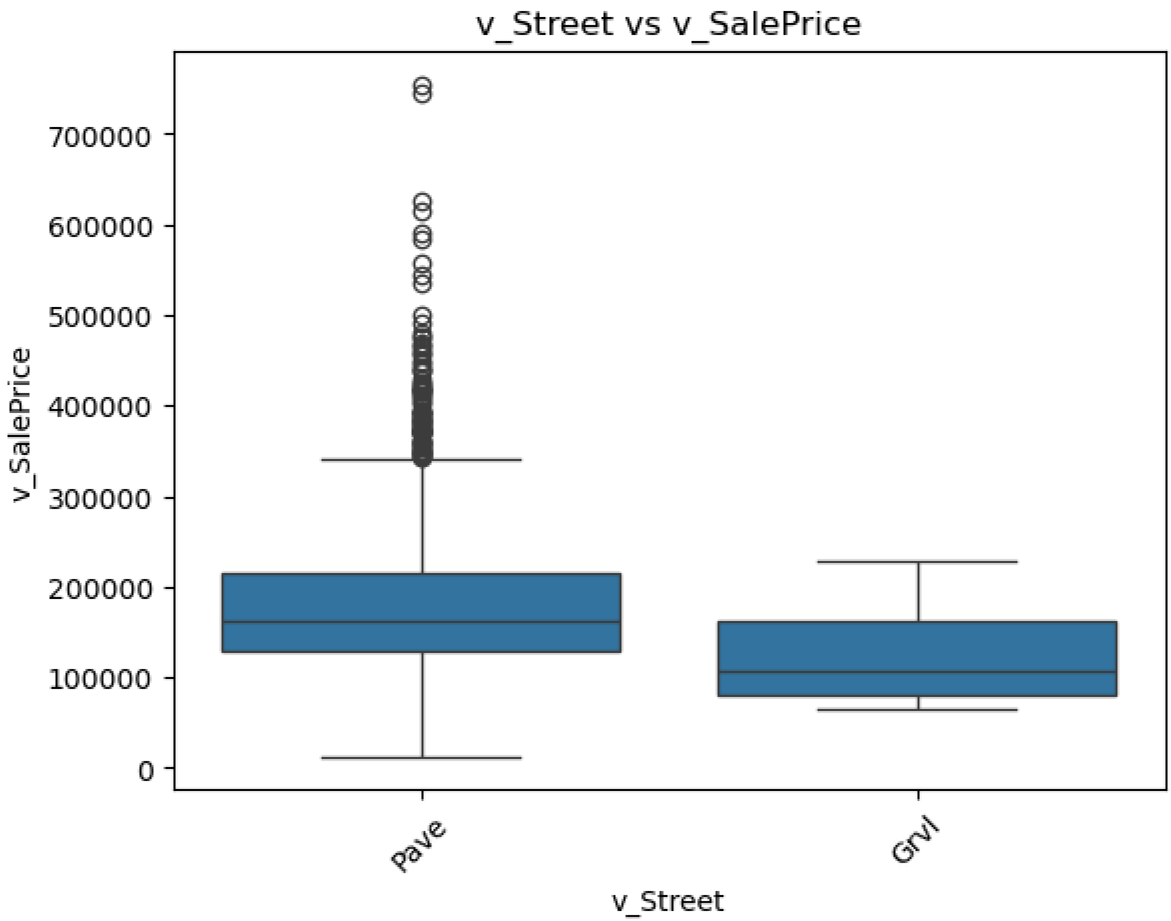

v_Misc_Val vs v_SalePrice
v_SalePrice
v_Misc_Val
0
100000
200000
300000
400000
500000
600000
700000
0
2500
5000
7500
10000
12500
15000
17500


v_Mo_Sold vs v_SalePrice
v_SalePrice
v_Mo_Sold
0
100000
200000
300000
400000
500000
600000
700000
2
4
6
8
10
12

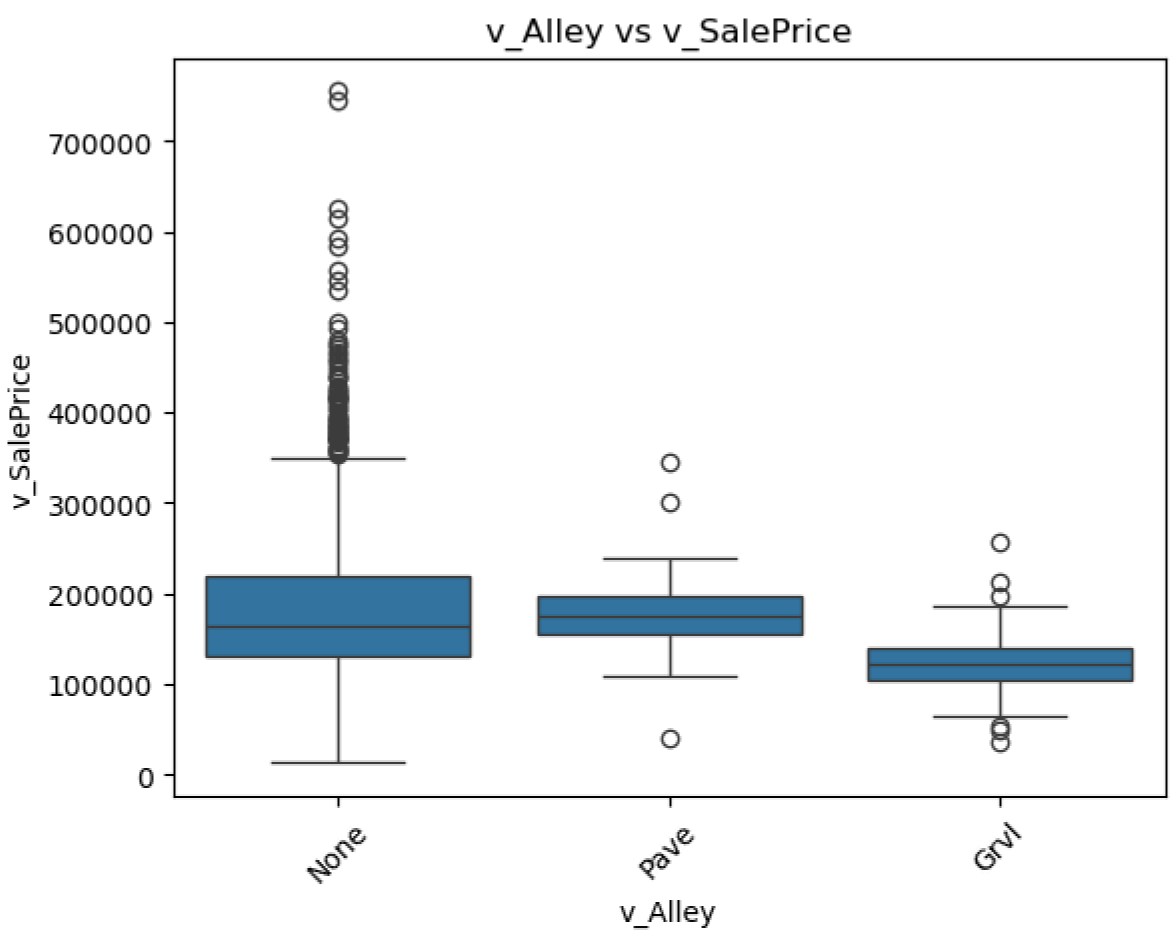

v_Yr_Sold vs v_SalePrice
v_SalePrice
v_Yr_Sold
0
100000
200000
300000
400000
500000
600000
700000
2006.00
2006.25
2006.50
2006.75
2007.00
2007.25
2007.50
2007.75
2008.00

In [19]:
```
for col in categorical:
    if df[col].nunique() < 10:
        sns.boxplot(x=df[col], y=df['v_SalePrice'])
        plt.title(f'{col} vs v_SalePrice')
        plt.xticks(rotation=45)
        plt.show()
```

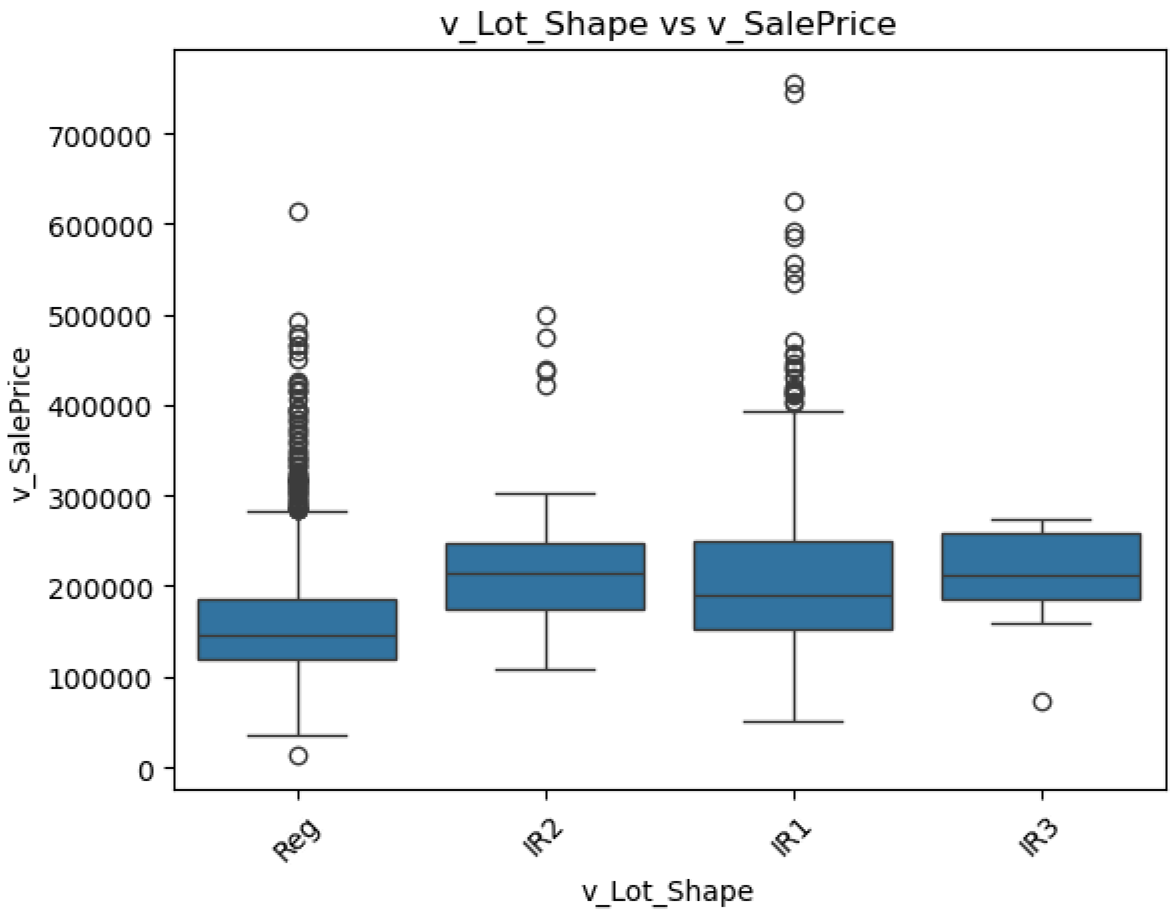


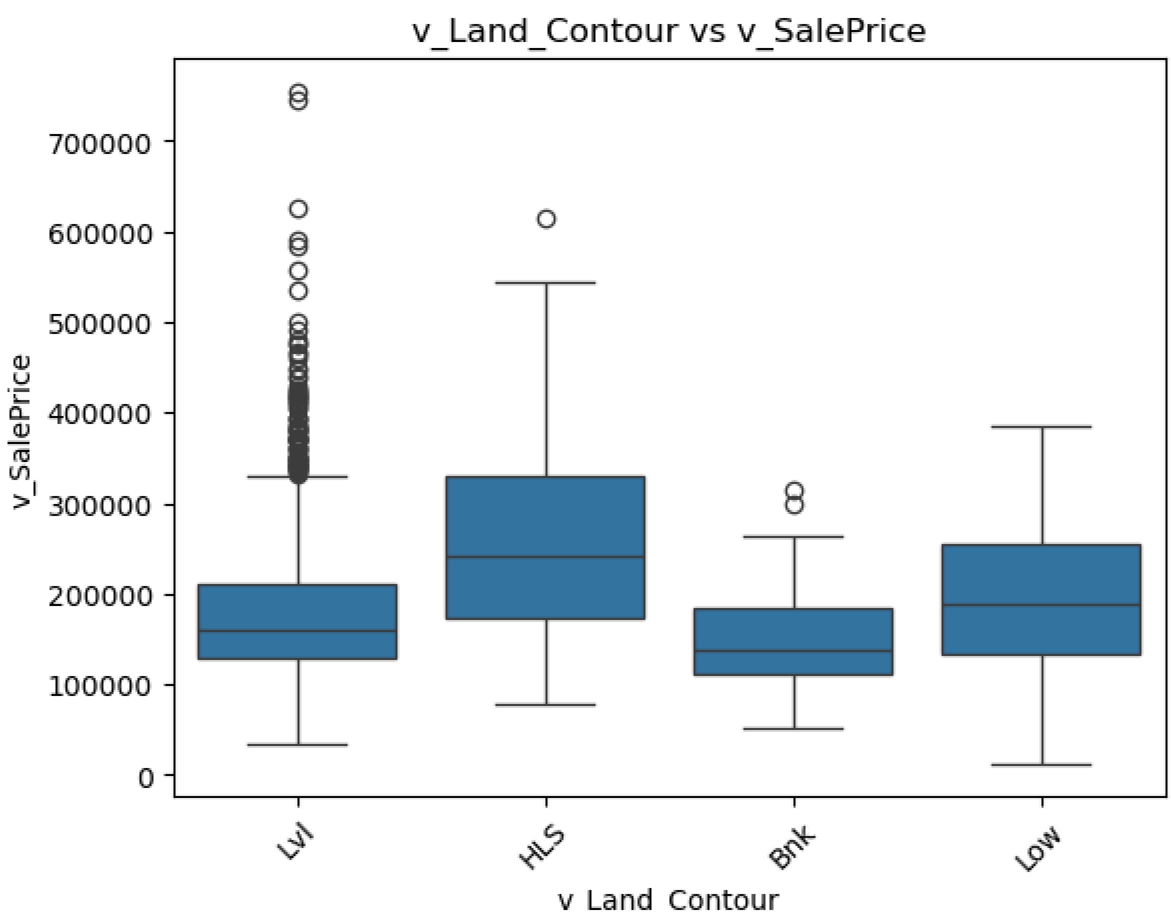


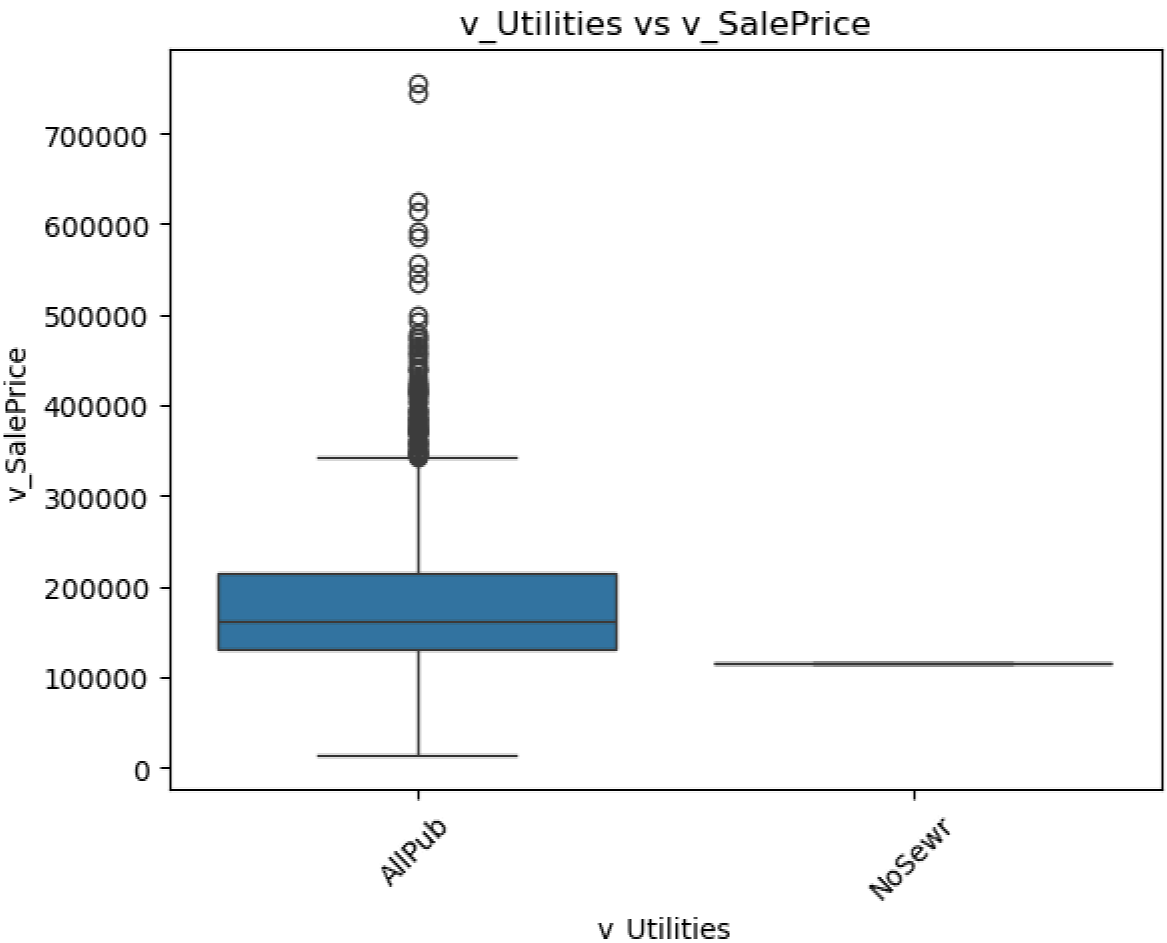
v_Street vs v_SalePrice
v_SalePrice
700000
600000
500000
400000
300000
200000
100000
0
Pave
Grvl
v_Street

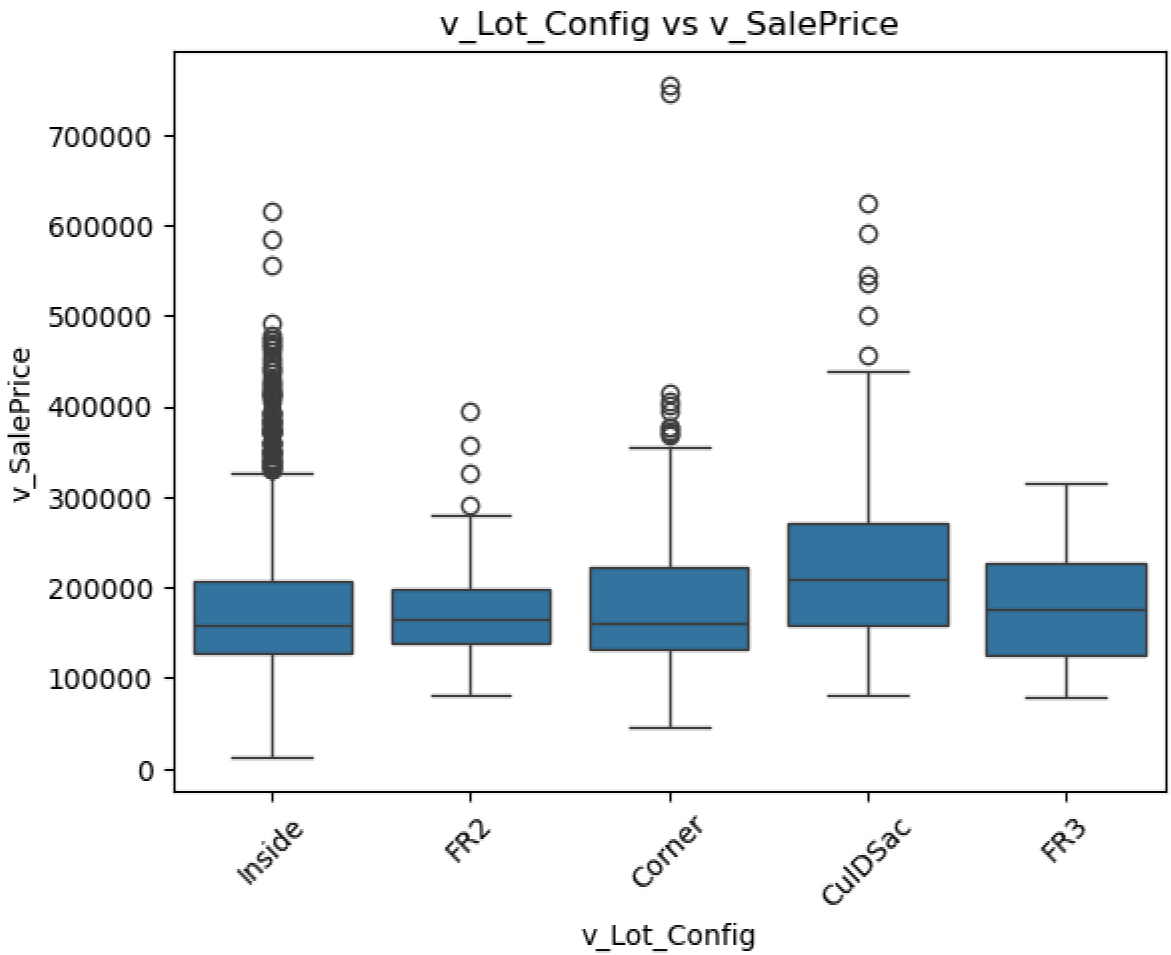


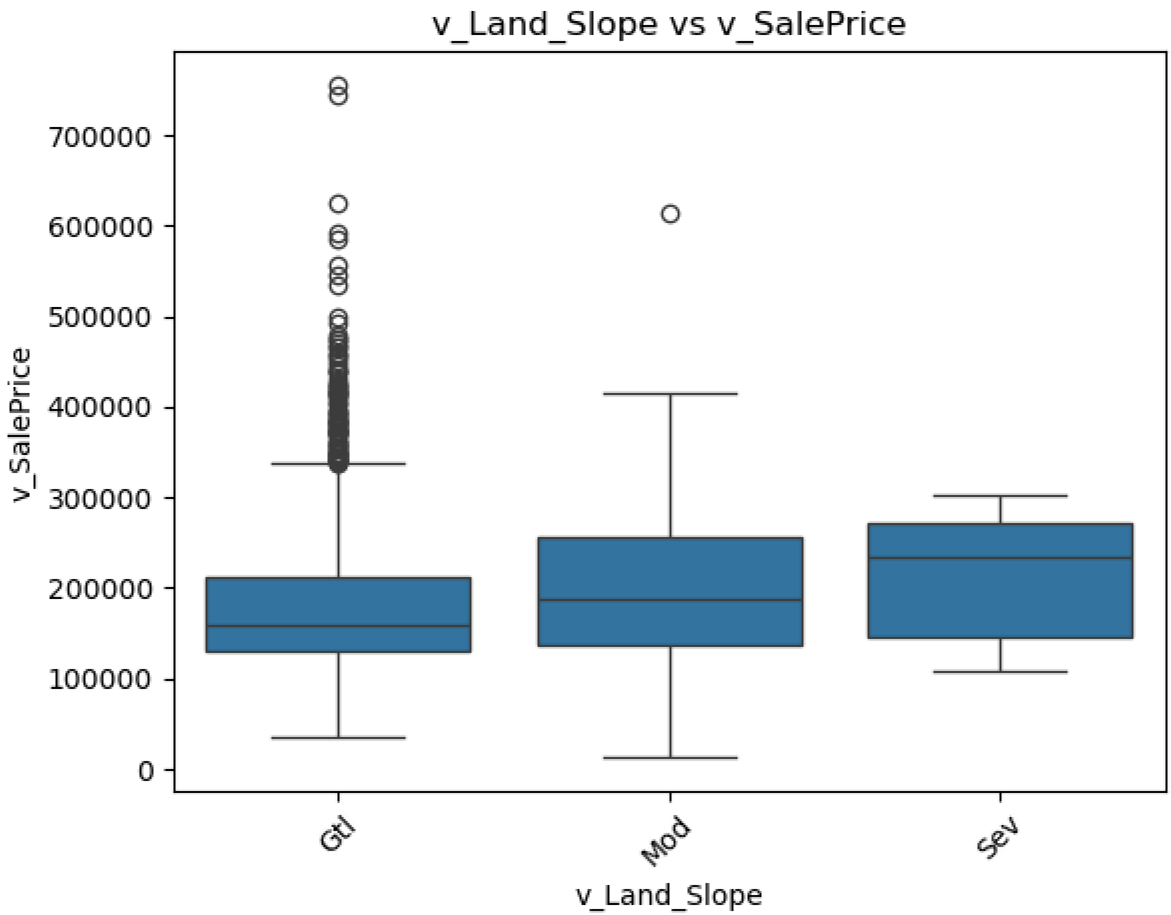
v_Alley vs v_SalePrice
v_SalePrice
700000
600000
500000
400000
300000
200000
100000
0
None
Pave
Grvl
v_Alley

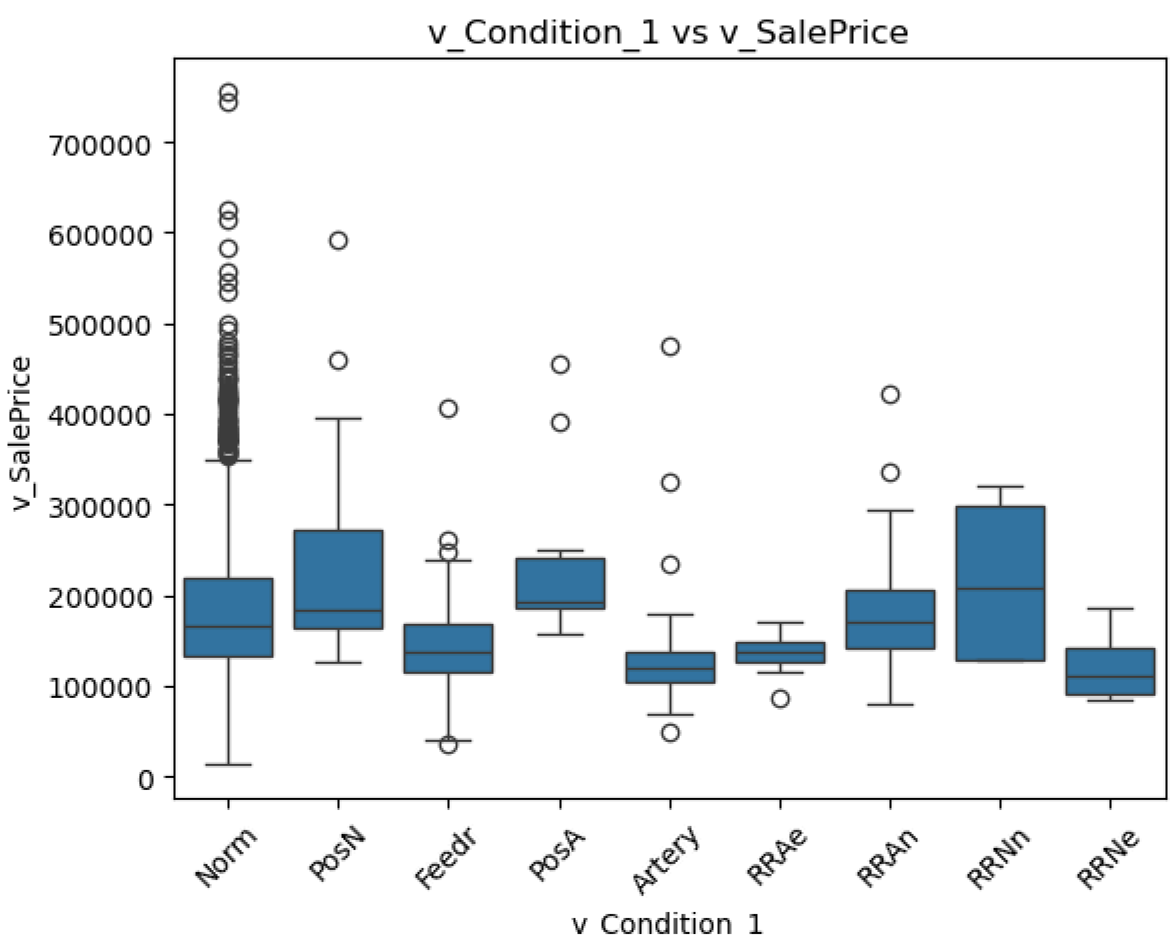

v_Lot_Shape vs v_SalePrice
v_SalePrice
700000
600000
500000
400000
300000
200000
100000
0
Reg
IR2
IR1
IR3
v_Lot_Shape

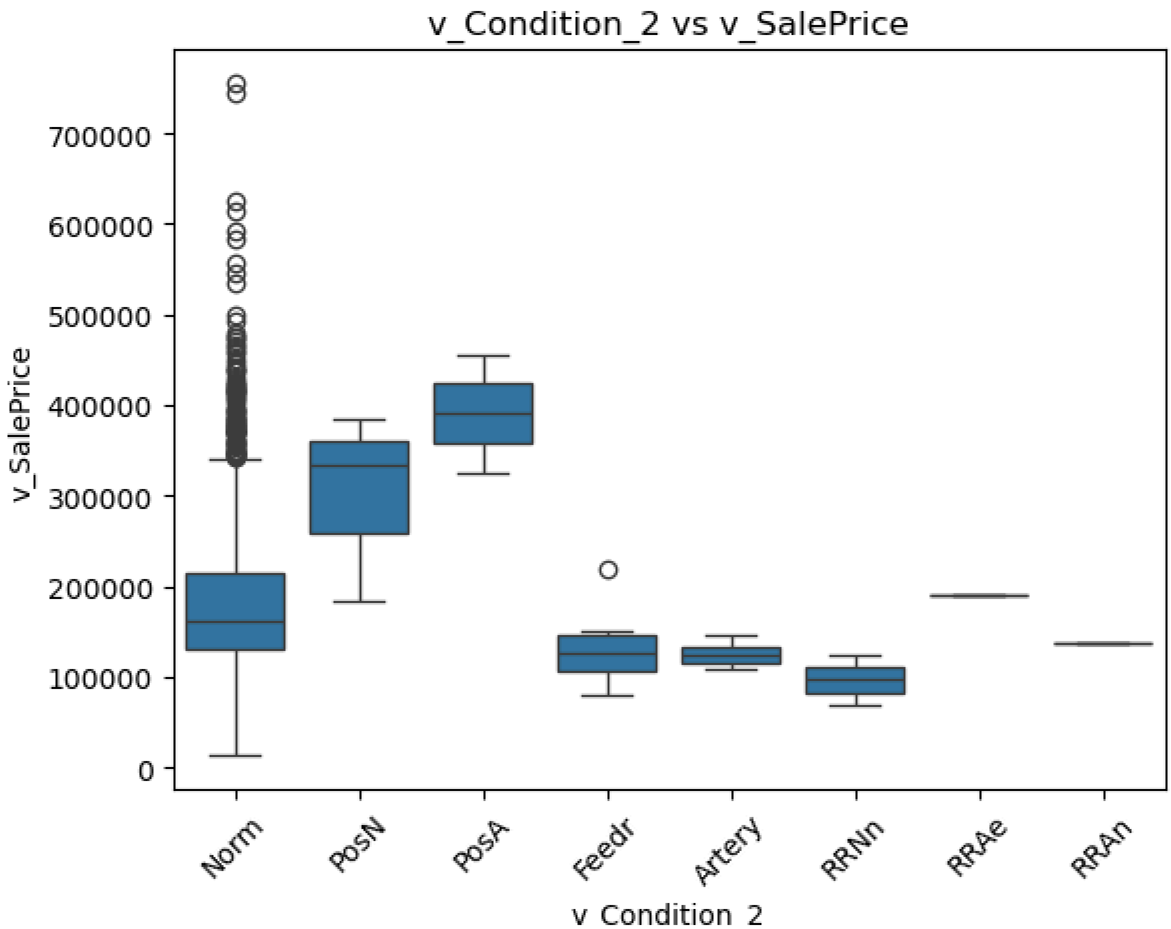

v_Land_Contour vs v_SalePrice
v_SalePrice
700000
600000
500000
400000
300000
200000
100000
0
Lvl
HLS
Bnk
Low
v_Land_Contour

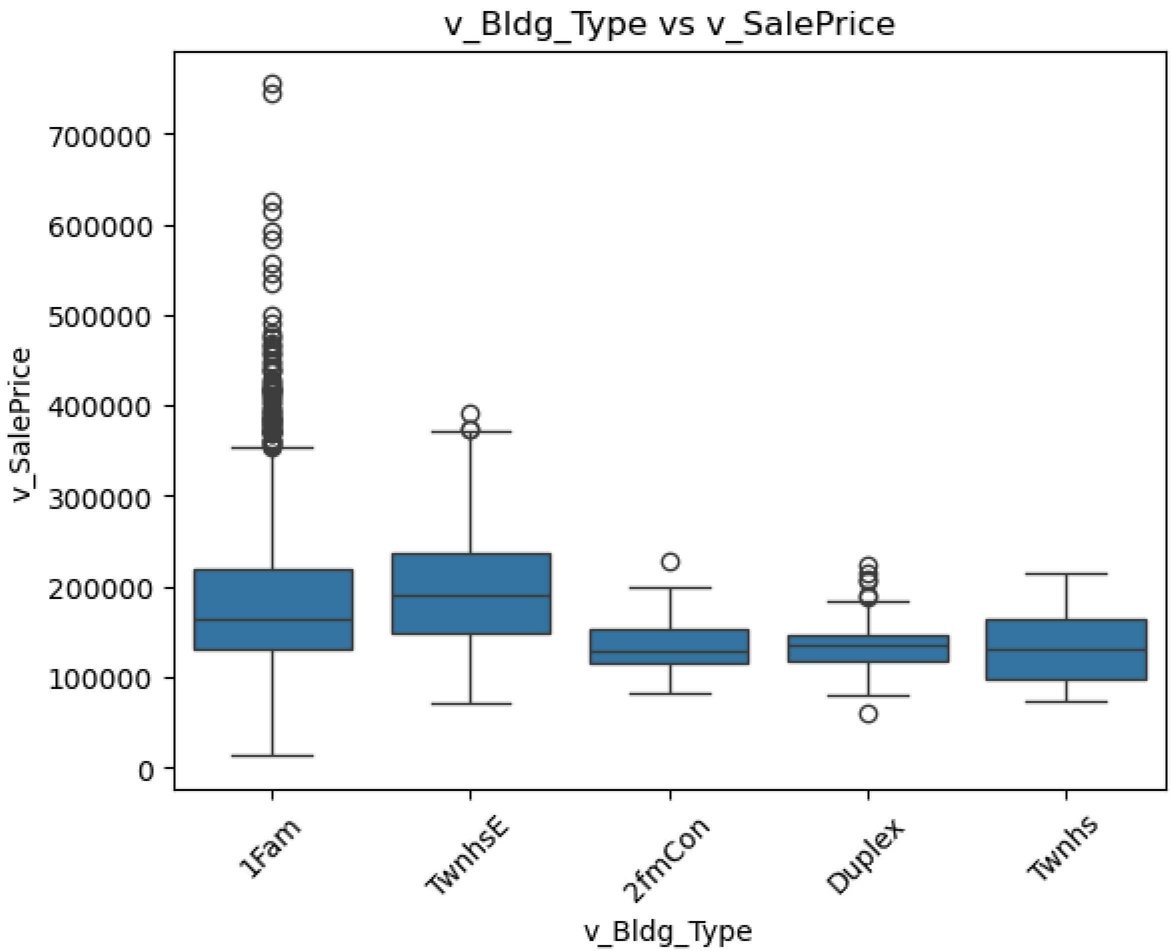

v_Utilities vs v_SalePrice
700000
600000
500000
400000
300000
200000
100000
0
v_SalePrice
AllPub
NoSewr
v_Utilities

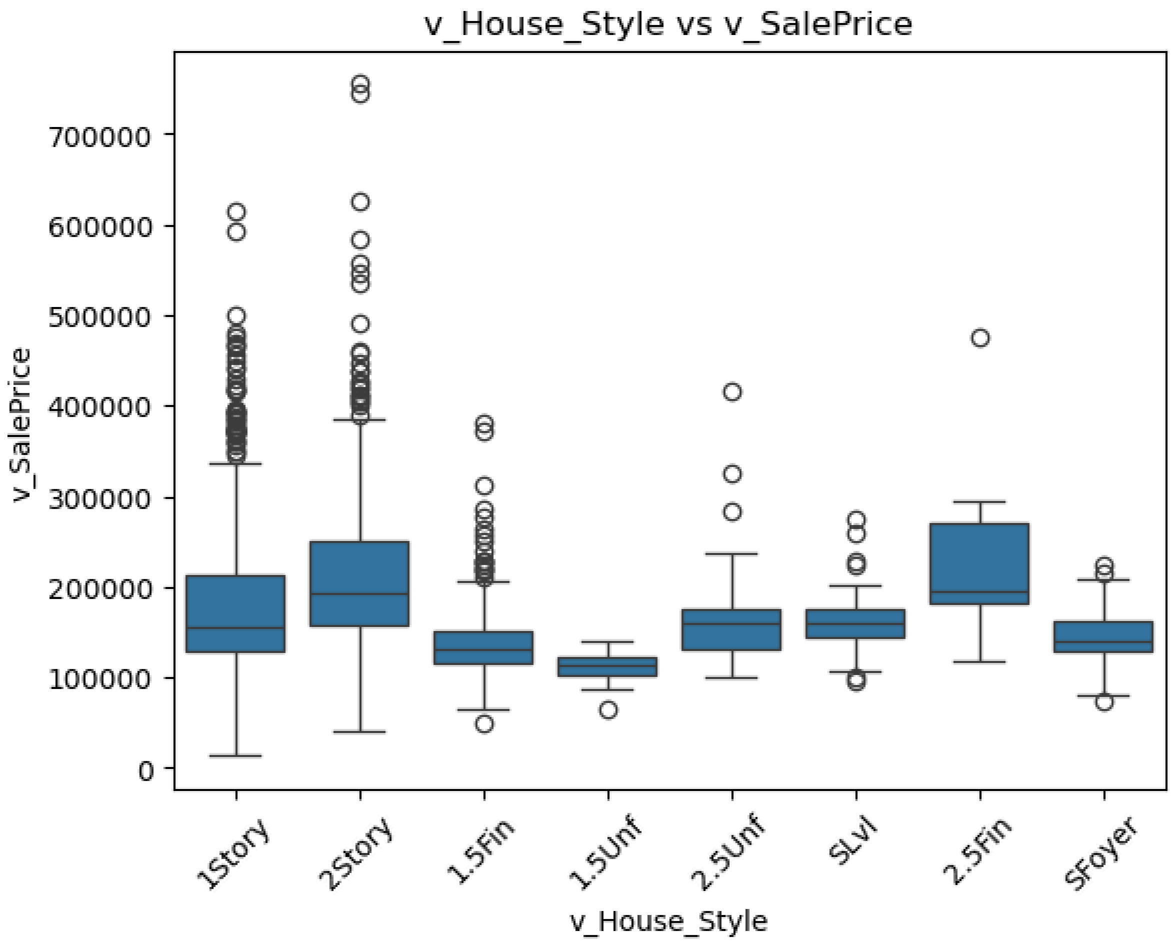

v_Lot_Config vs v_SalePrice
v_SalePrice
700000
600000
500000
400000
300000
200000
100000
0
Inside
FR2
Corner
CulDSac
FR3
v_Lot_Config


v_Land_Slope vs v_SalePrice
v_SalePrice
0
100000
200000
300000
400000
500000
600000
700000
Gtl
Mod
Sev
v_Land_Slope

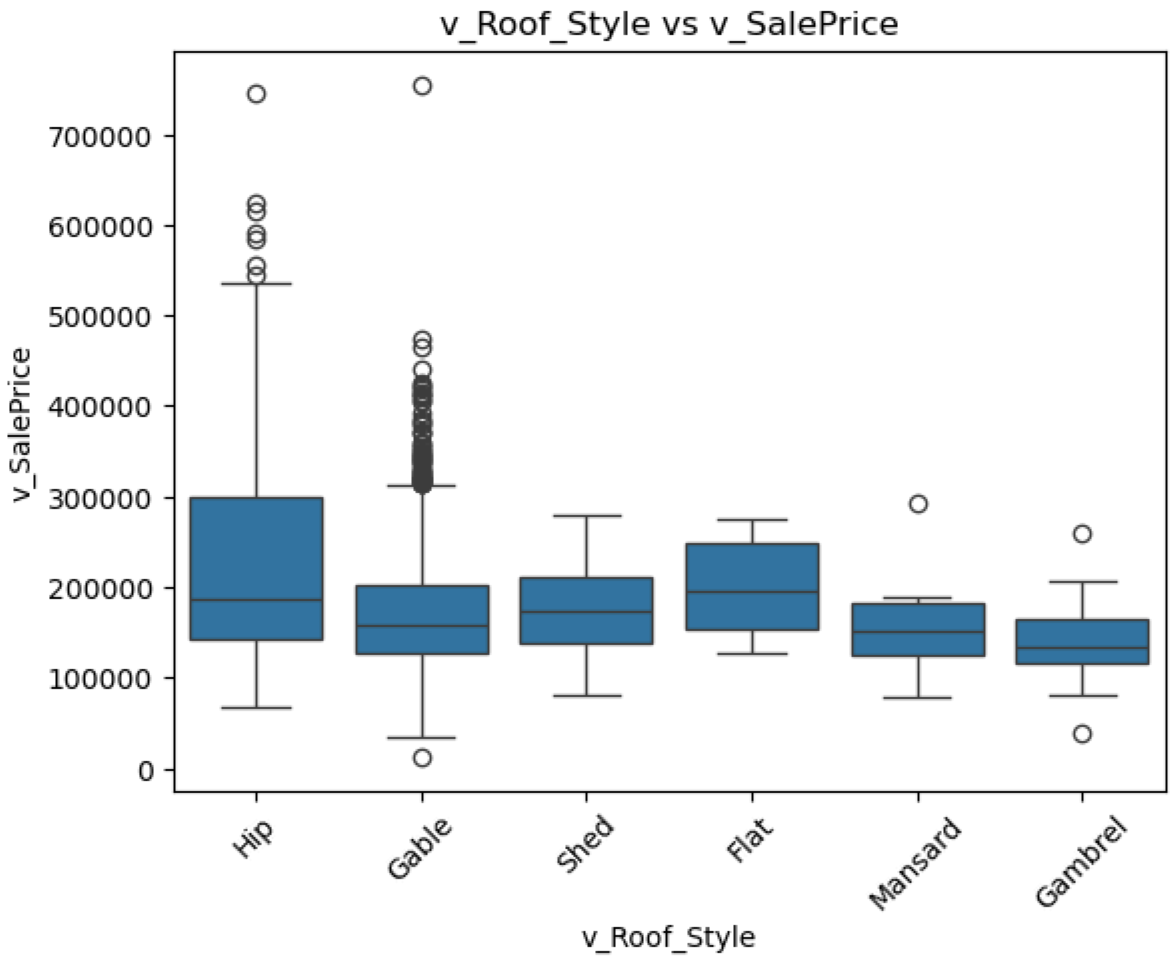

v_Condition_1 vs v_SalePrice
v_SalePrice
0
100000
200000
300000
400000
500000
600000
700000
Norm
PosN
Feedr
PosA
Artery
RRAe
RRAn
RRNn
RRNe
v_Condition_1


v_Condition_2 vs v_SalePrice
v_SalePrice
700000
600000
500000
400000
300000
200000
100000
0
Norm
PosN
PosA
Feedr
Artery
RRNn
RRAe
RRAn
v_Condition_2

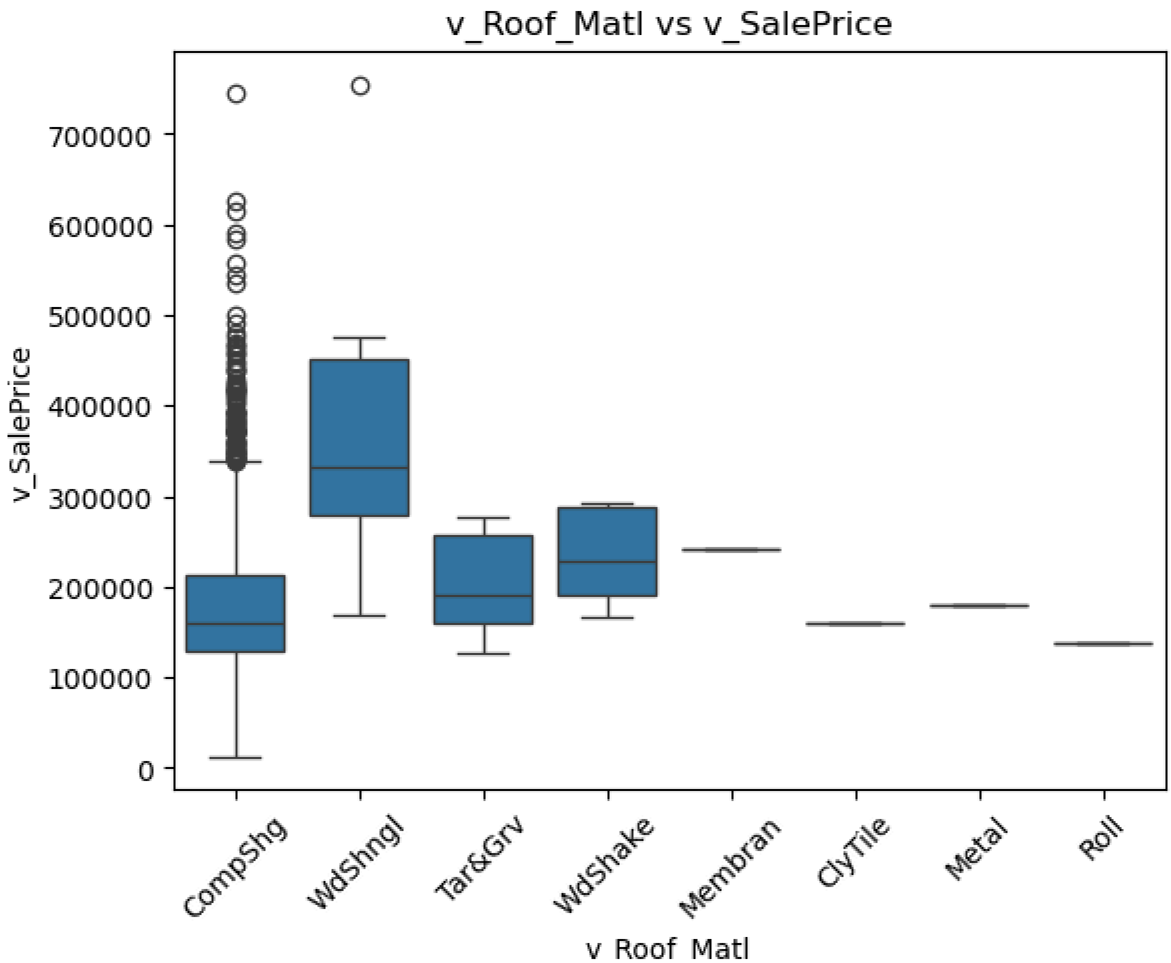

v_Bldg_Type vs v_SalePrice
v_SalePrice
700000
600000
500000
400000
300000
200000
100000
0
1Fam
TwnhsE
2fmCon
Duplex
Twnhs
v_Bldg_Type

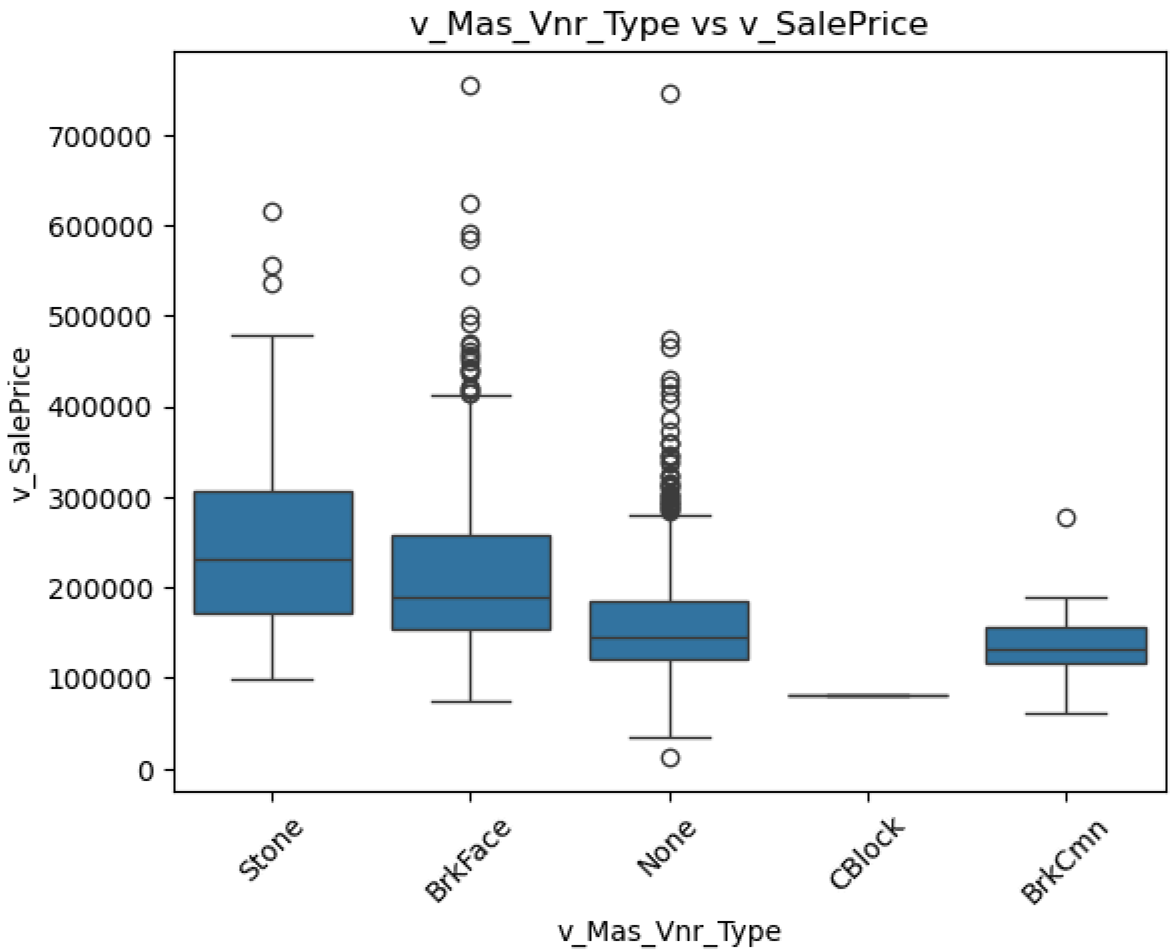


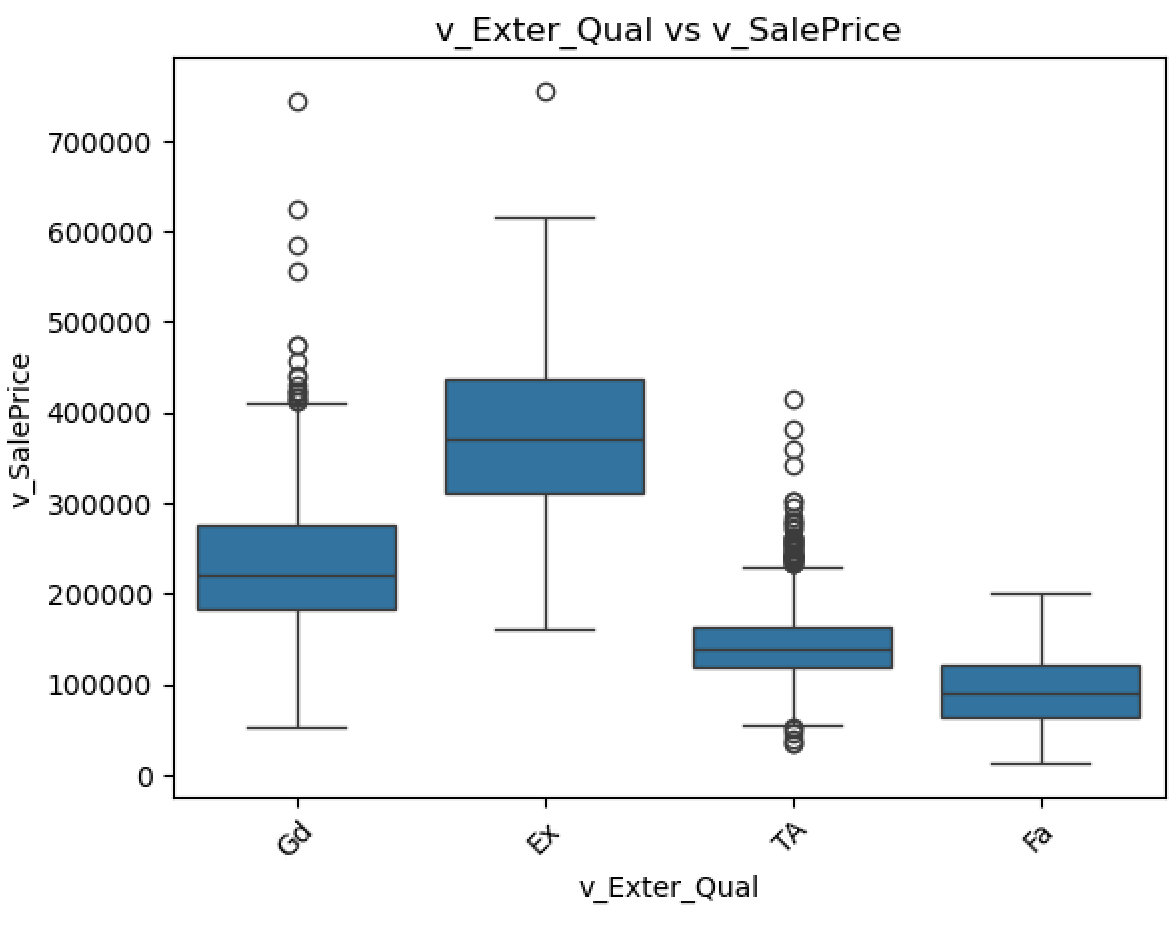
v_House_Style vs v_SalePrice
v_SalePrice
0
100000
200000
300000
400000
500000
600000
700000
1Story
2Story
1.5Fin
1.5Unf
2.5Unf
SLvl
2.5Fin
SFoyer
v_House_Style

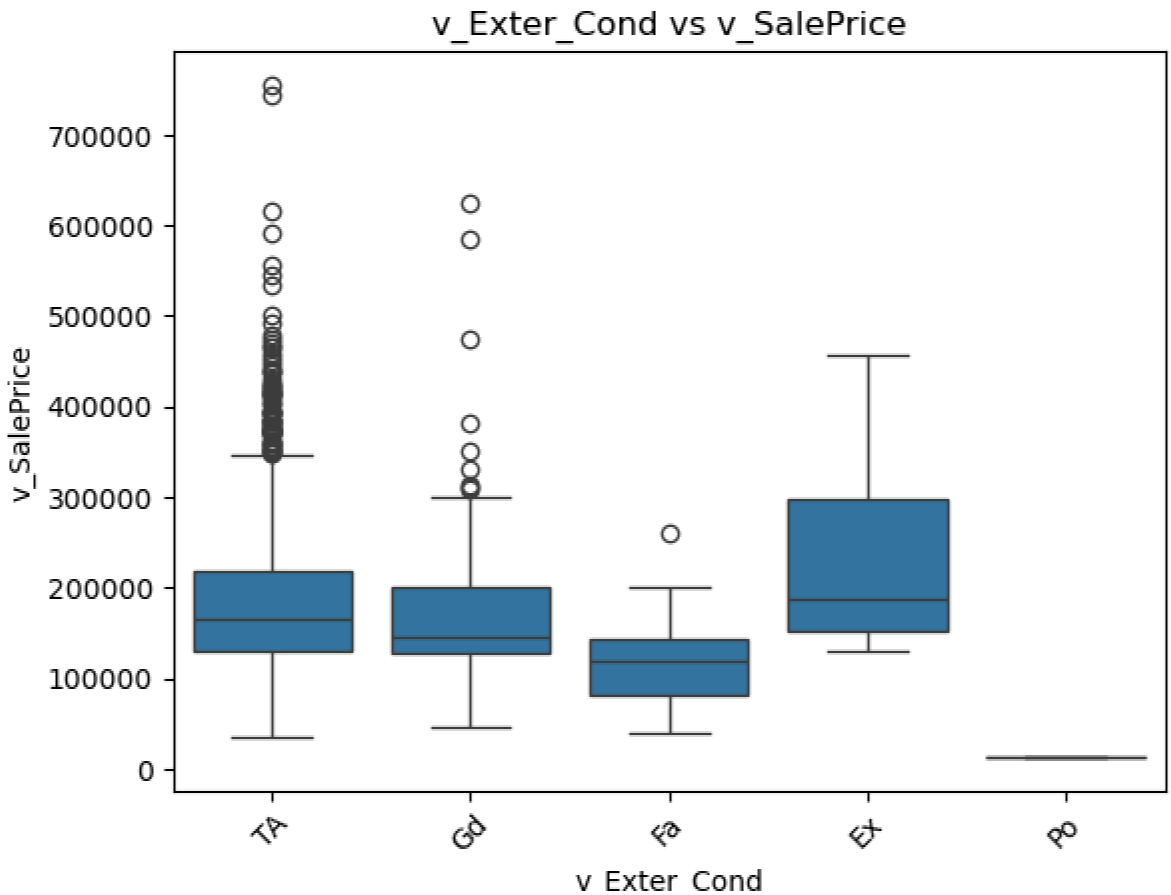


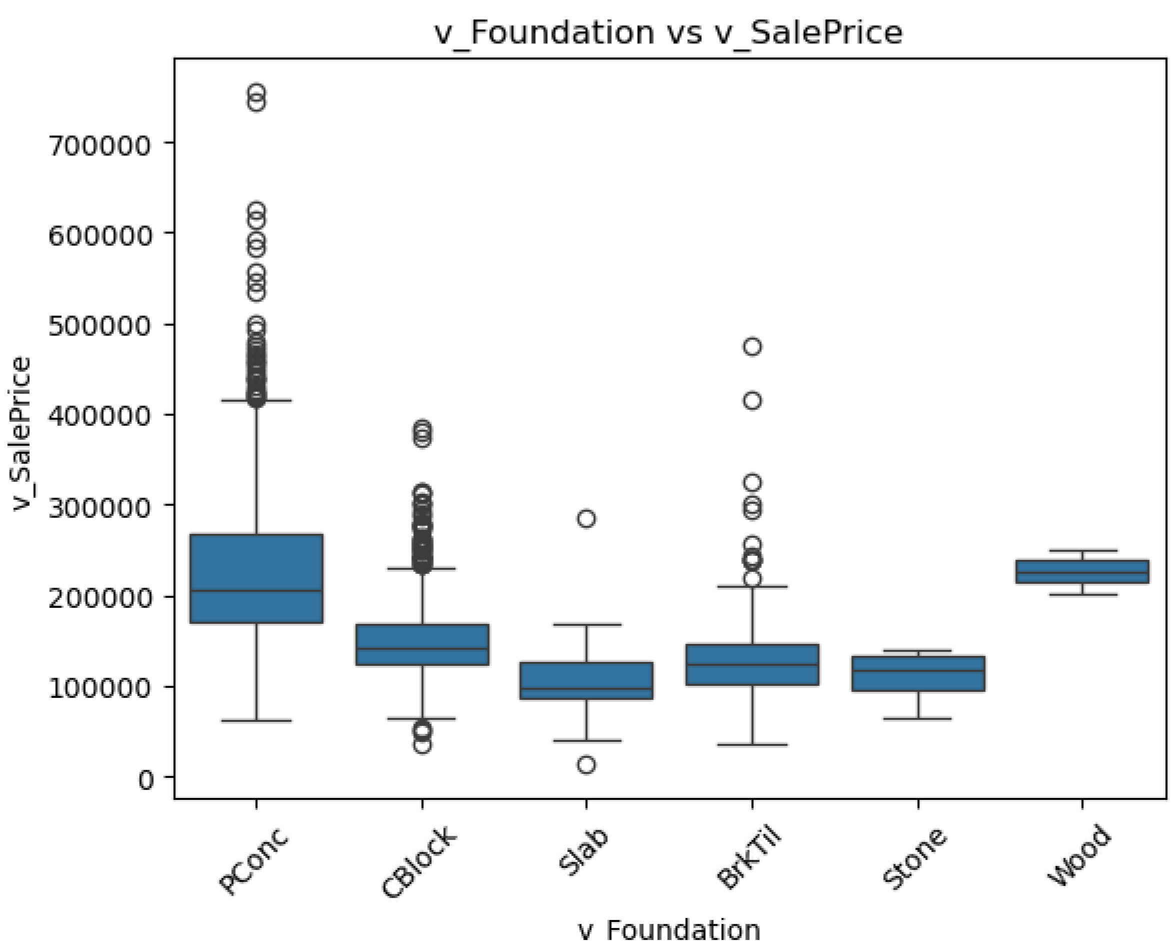
v_Roof_Style vs v_SalePrice
700000
600000
500000
400000
300000
200000
100000
0
v_SalePrice
Hip
Gable
Shed
Flat
Mansard
Gambrel
v_Roof_Style

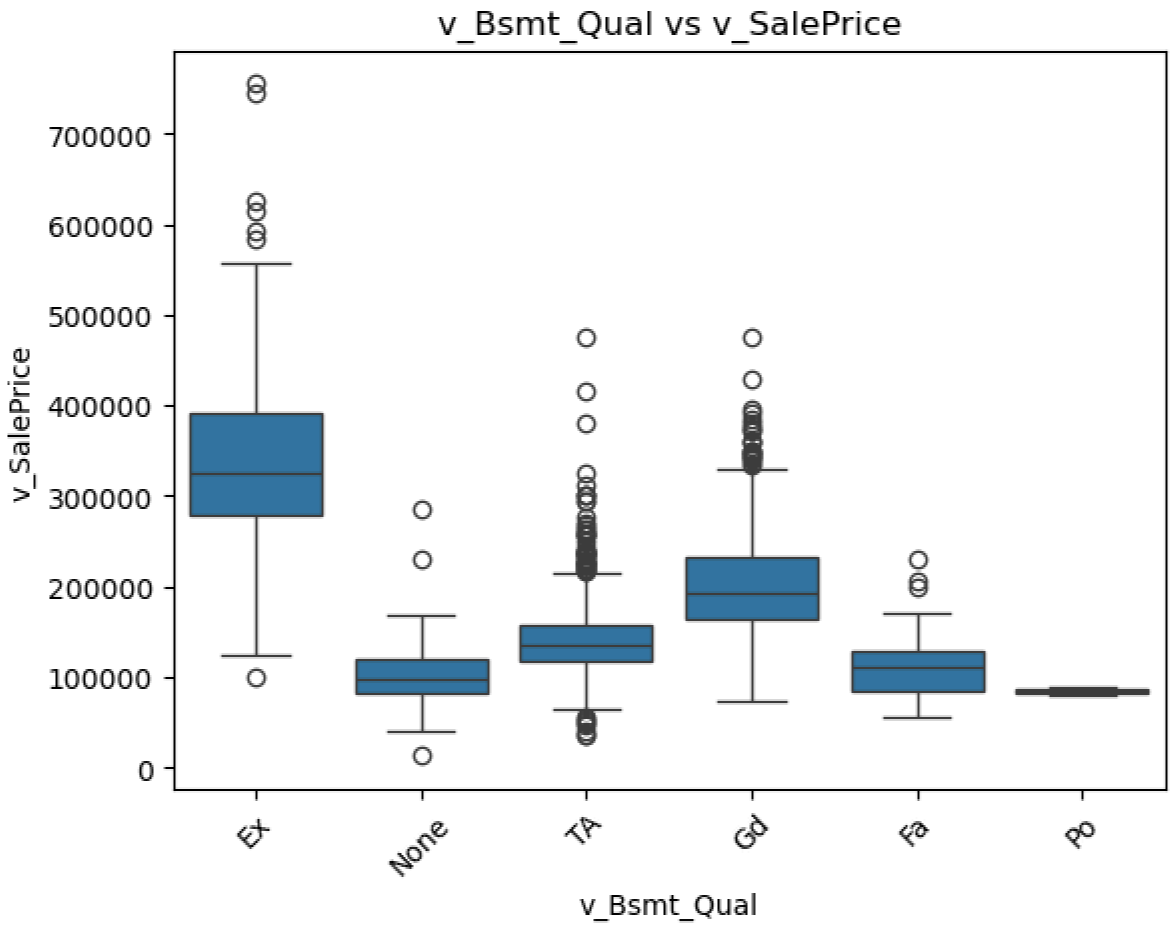

v_Roof_Matl vs v_SalePrice
v_SalePrice
700000
600000
500000
400000
300000
200000
100000
0
CompShg
WdShngl
Tar&Grv
WdShake
Membran
ClyTile
Metal
Roll
v_Roof_Matl

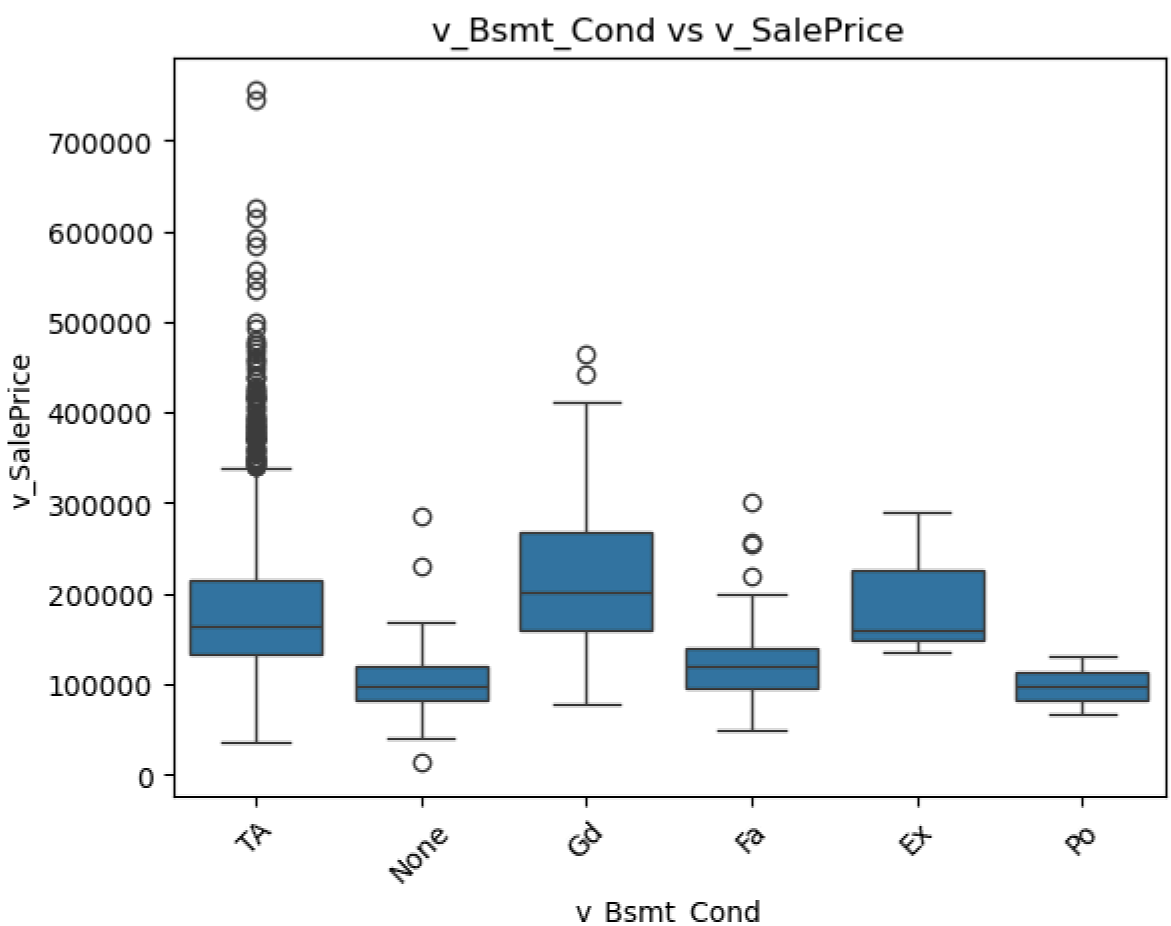

v_Mas_Vnr_Type vs v_SalePrice
v_SalePrice
700000
600000
500000
400000
300000
200000
100000
0
Stone
BrkFace
None
CBlock
BrkCmn
v_Mas_Vnr_Type

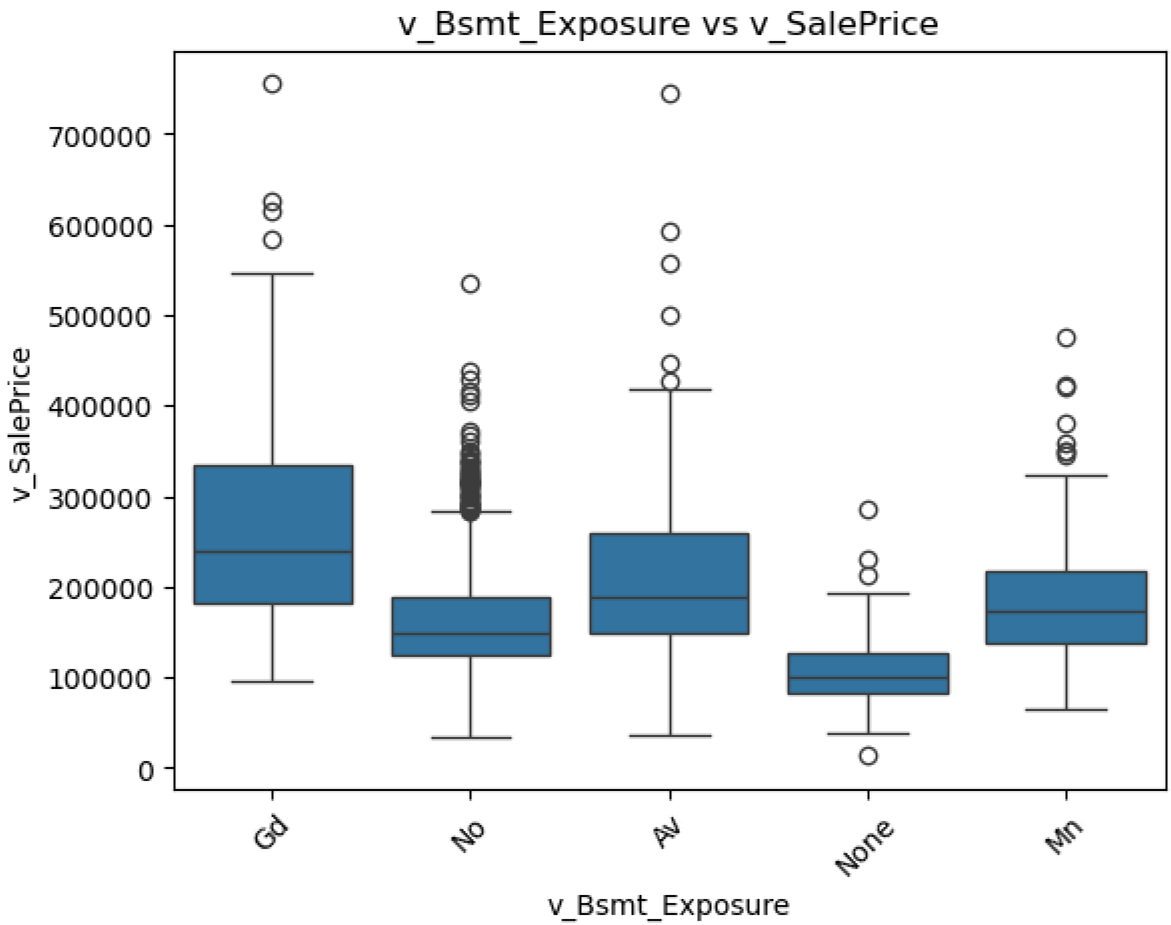

v_Exter_Qual vs v_SalePrice
v_SalePrice
700000
600000
500000
400000
300000
200000
100000
0
Gd
Ex
TA
Fa
v_Exter_Qual

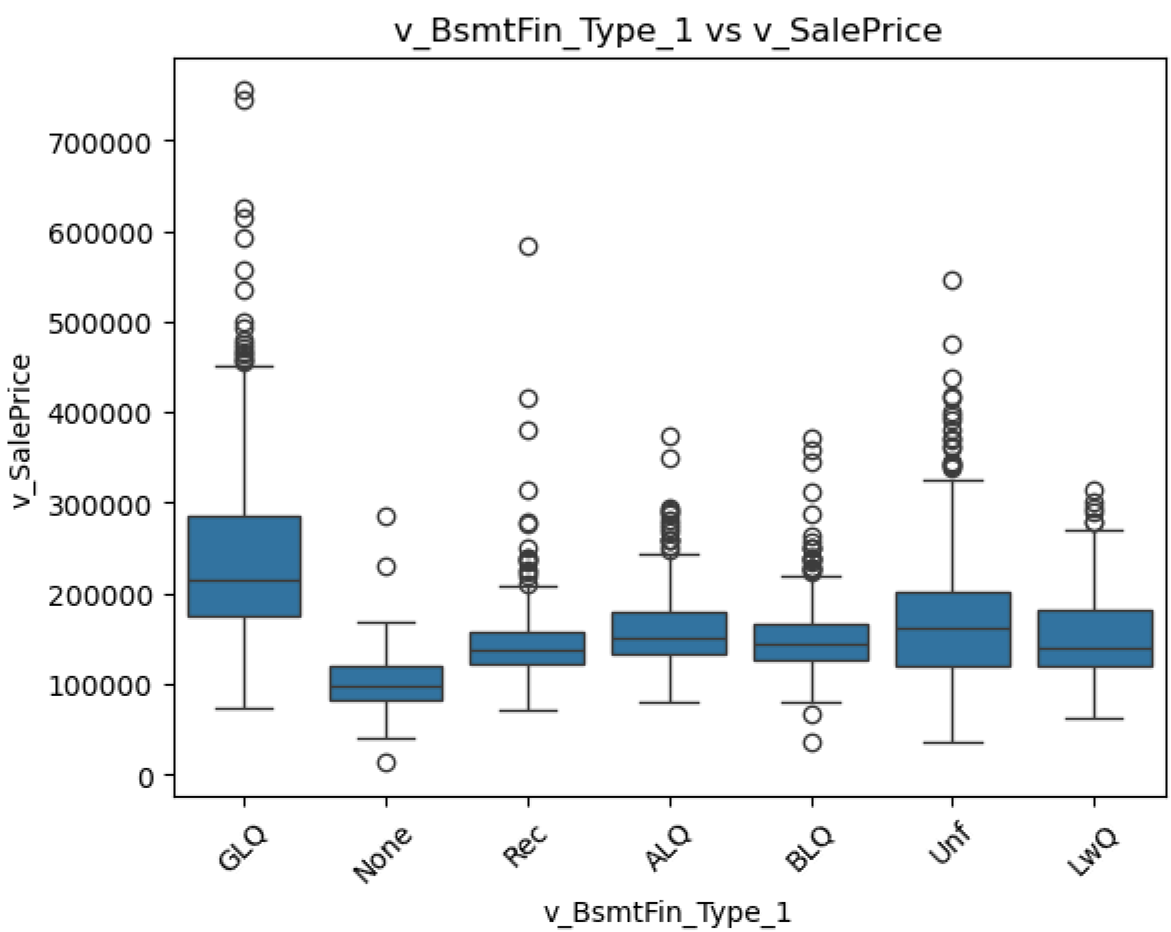

v_Exter_Cond vs v_SalePrice
v_SalePrice
700000
600000
500000
400000
300000
200000
100000
0
TA
Gd
Fa
Ex
Po
v_Exter_Cond

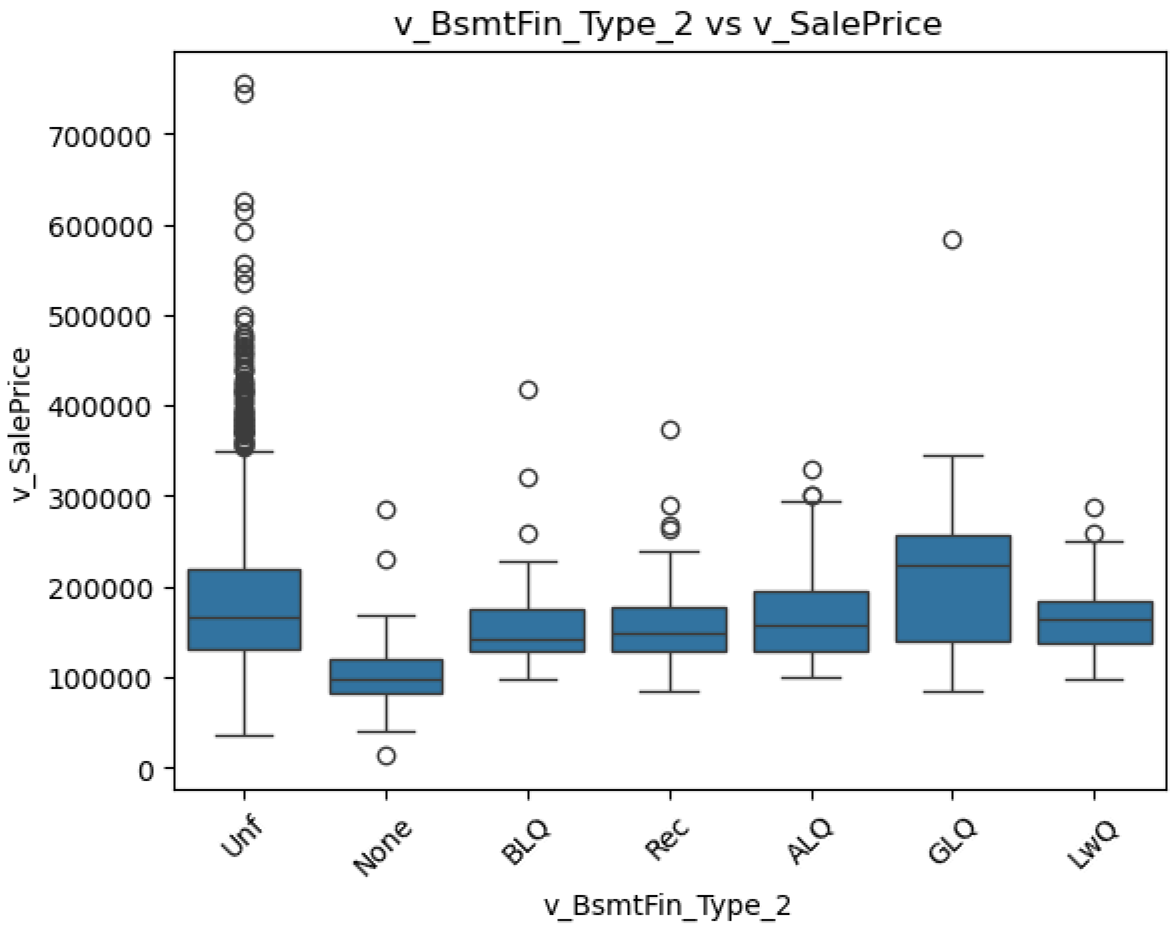

v_Foundation vs v_SalePrice
v_SalePrice
700000
600000
500000
400000
300000
200000
100000
0
PConc
CBlock
Slab
BrkTil
Stone
Wood
v_Foundation


v_Bsmt_Qual vs v_SalePrice
v_SalePrice
0
100000
200000
300000
400000
500000
600000
700000
Ex
None
TA
Gd
Fa
Po
v_Bsmt_Qual

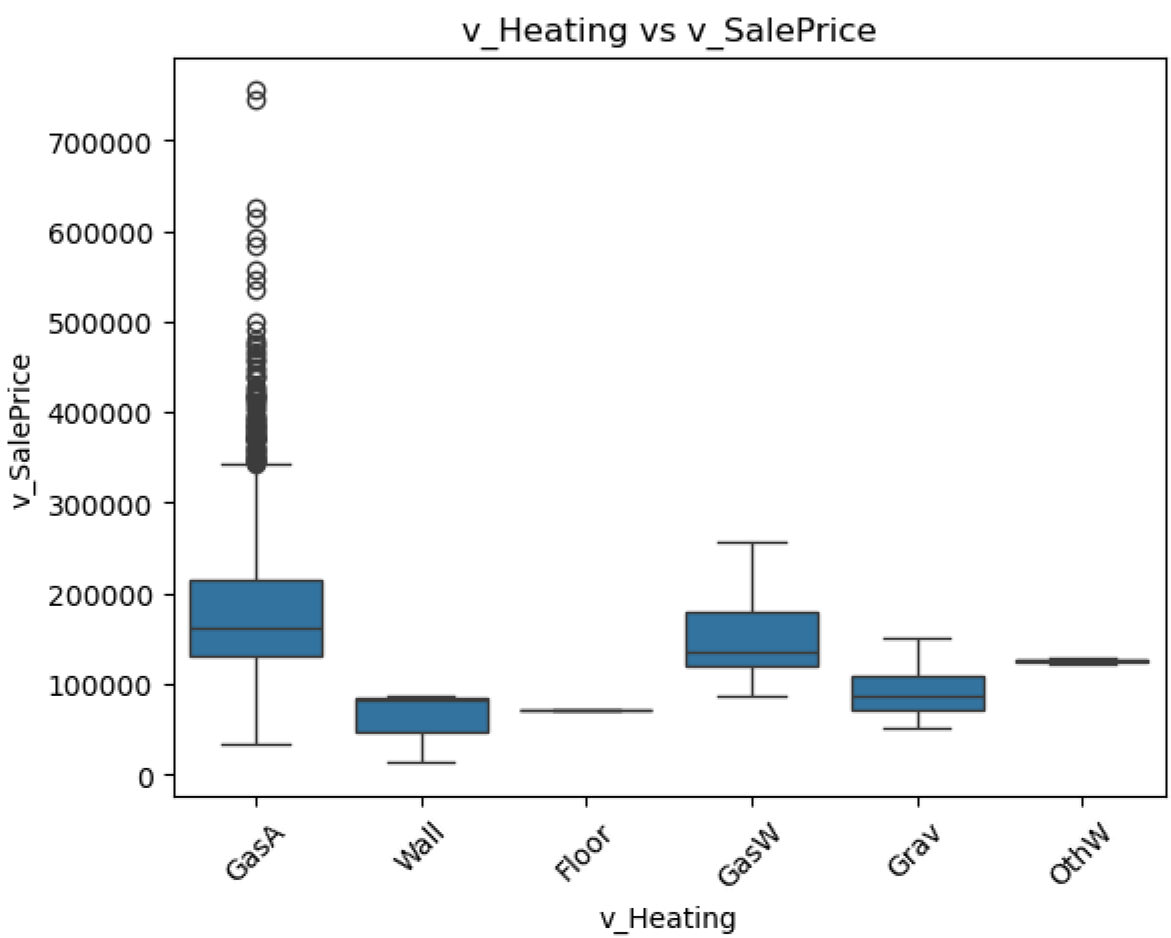

v_Bsmt_Cond vs v_SalePrice
v_SalePrice
0
100000
200000
300000
400000
500000
600000
700000
TA
None
Gd
Fa
Ex
Po
v_Bsmt_Cond


v_Bsmt_Exposure vs v_SalePrice
v_SalePrice
0
100000
200000
300000
400000
500000
600000
700000
Gd
No
Av
None
Mn
v_Bsmt_Exposure

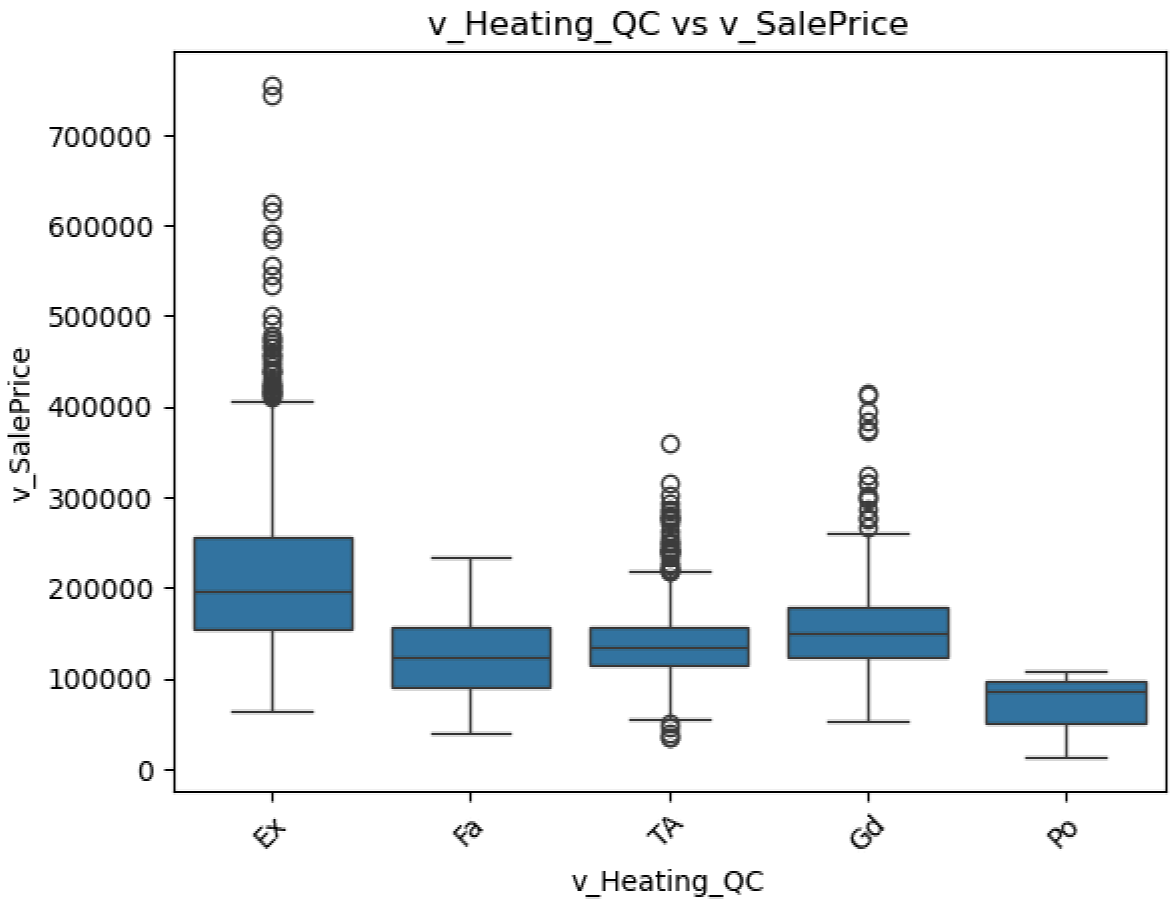

v_BsmtFin_Type_1 vs v_SalePrice
v_SalePrice
0
100000
200000
300000
400000
500000
600000
700000
GLQ
None
Rec
ALQ
BLQ
Unf
LwQ
v_BsmtFin_Type_1



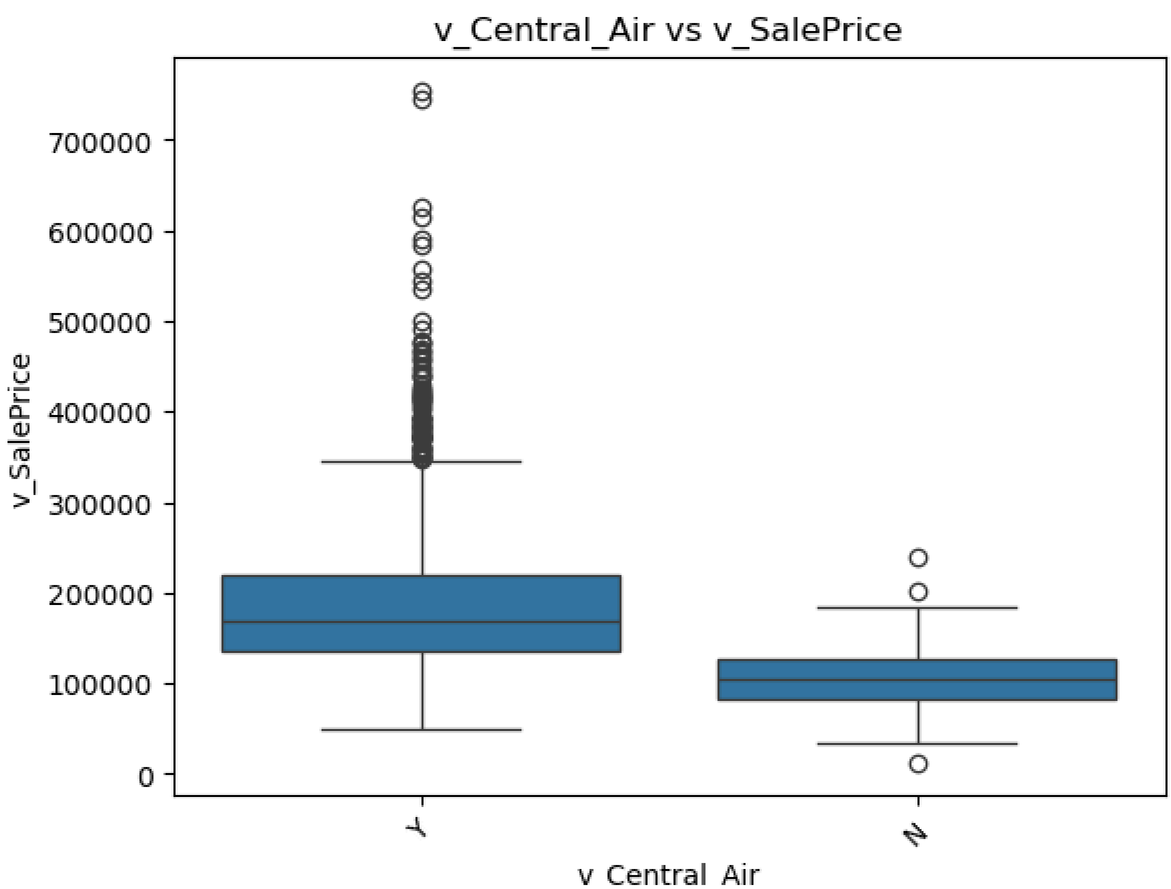
v_BsmtFin_Type_2 vs v_SalePrice
v_SalePrice
0
100000
200000
300000
400000
500000
600000
700000
Unf
None
BLQ
Rec
ALQ
GLQ
LwQ
v_BsmtFin_Type_2



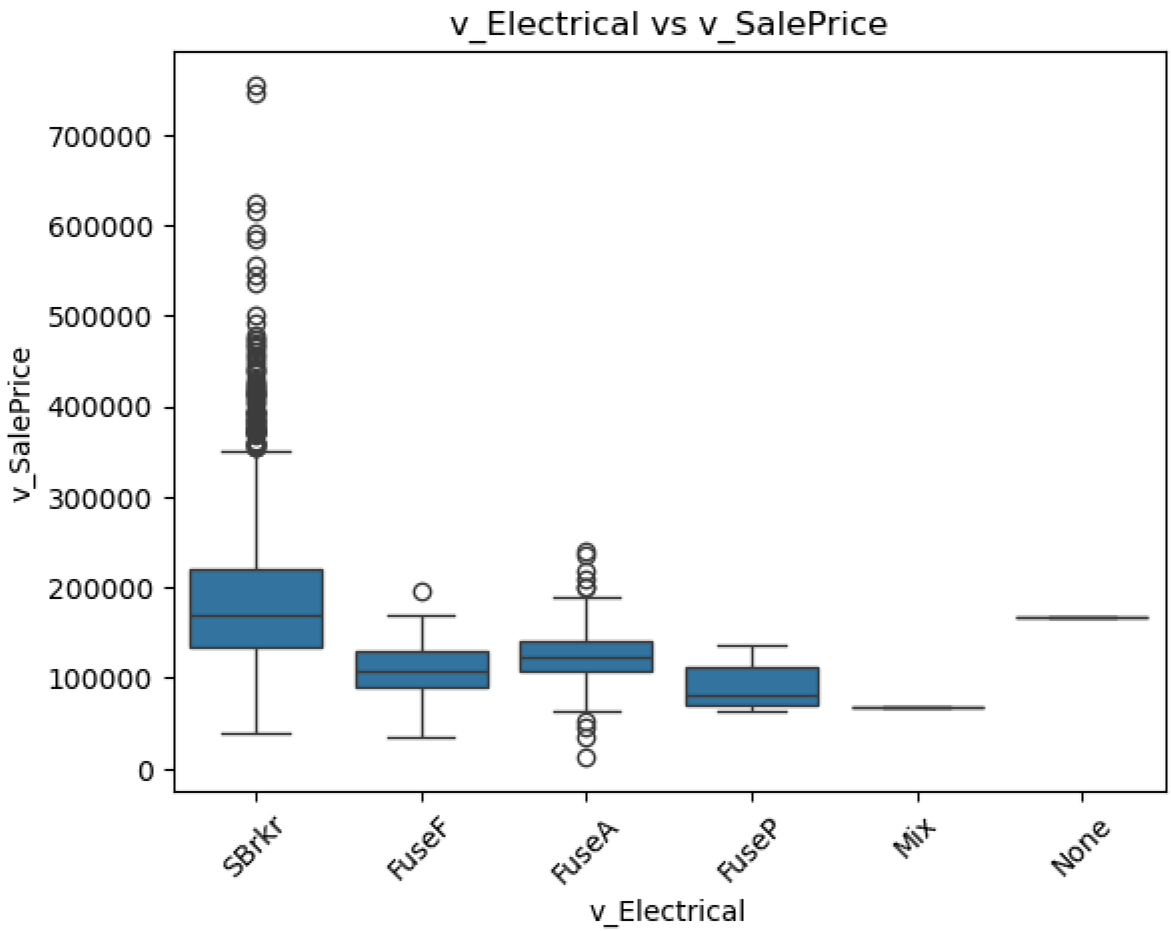
v_Heating vs v_SalePrice
v_SalePrice
0
100000
200000
300000
400000
500000
600000
700000
GasA
Wall
Floor
GasW
Grav
OthW
v_Heating

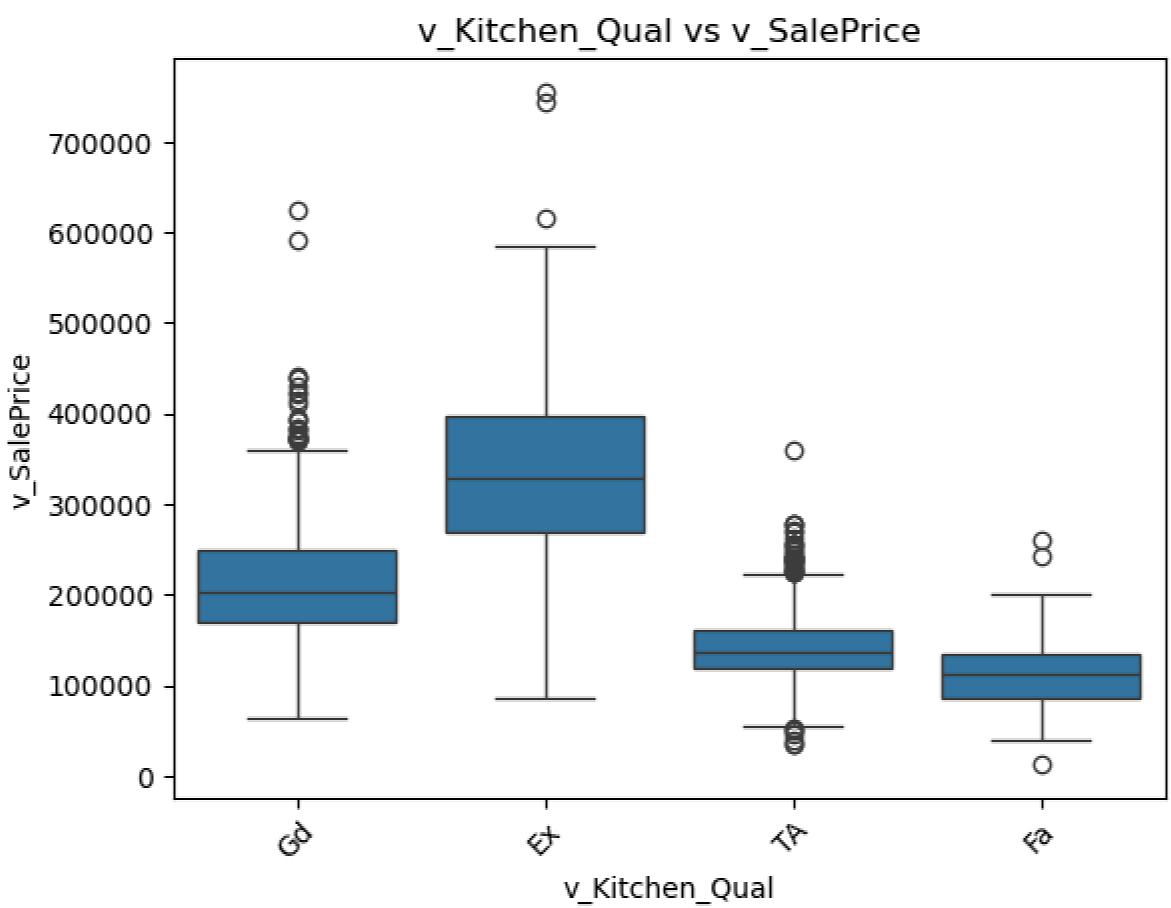


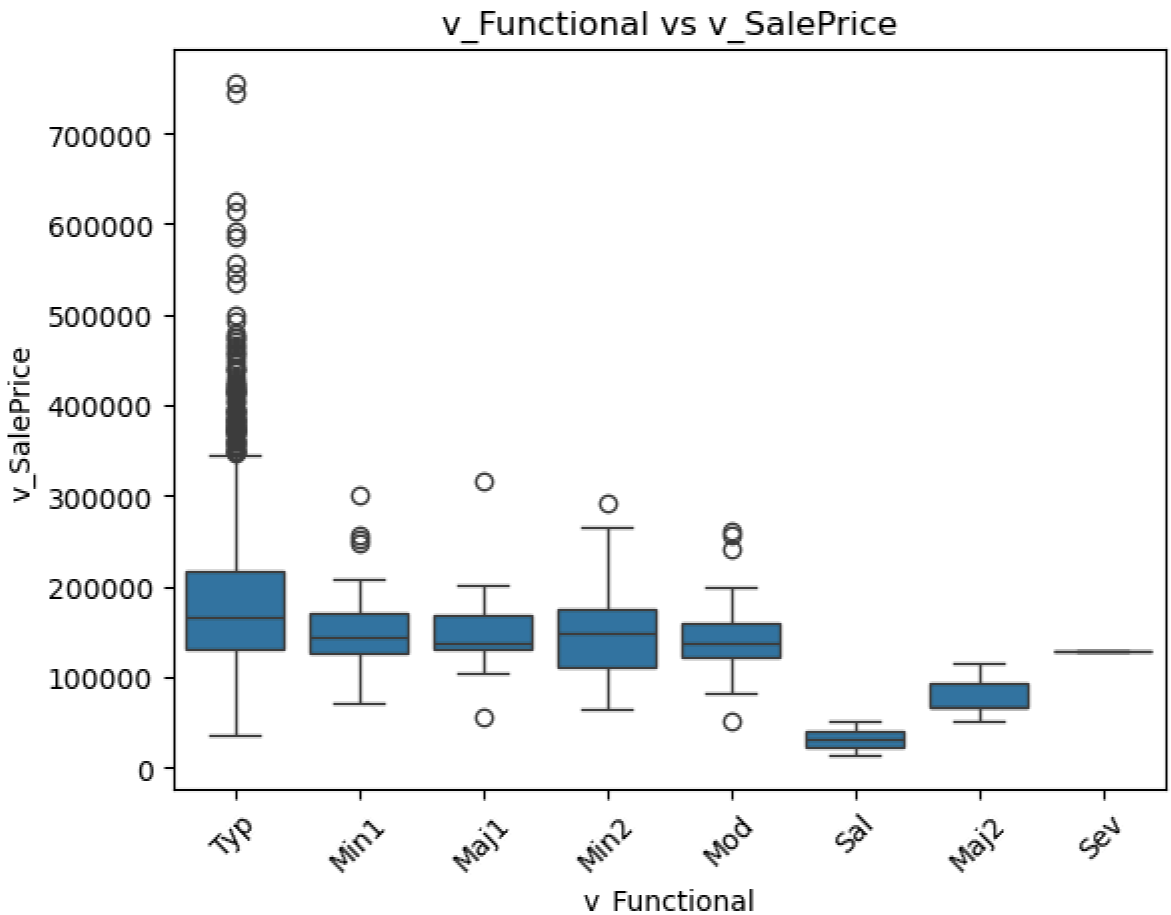
v_Heating_QC vs v_SalePrice
v_SalePrice
0
100000
200000
300000
400000
500000
600000
700000
Ex
Fa
TA
Gd
Po
v_Heating_QC

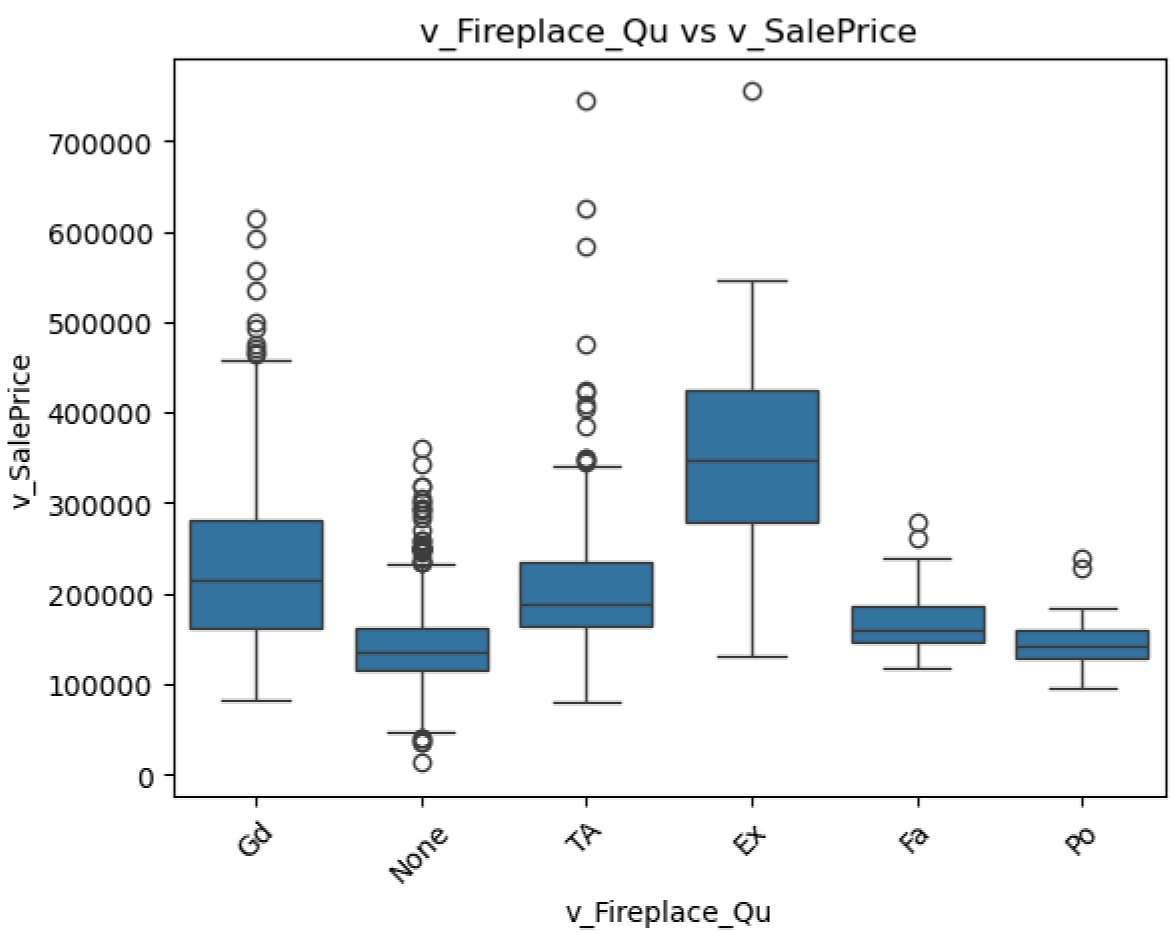


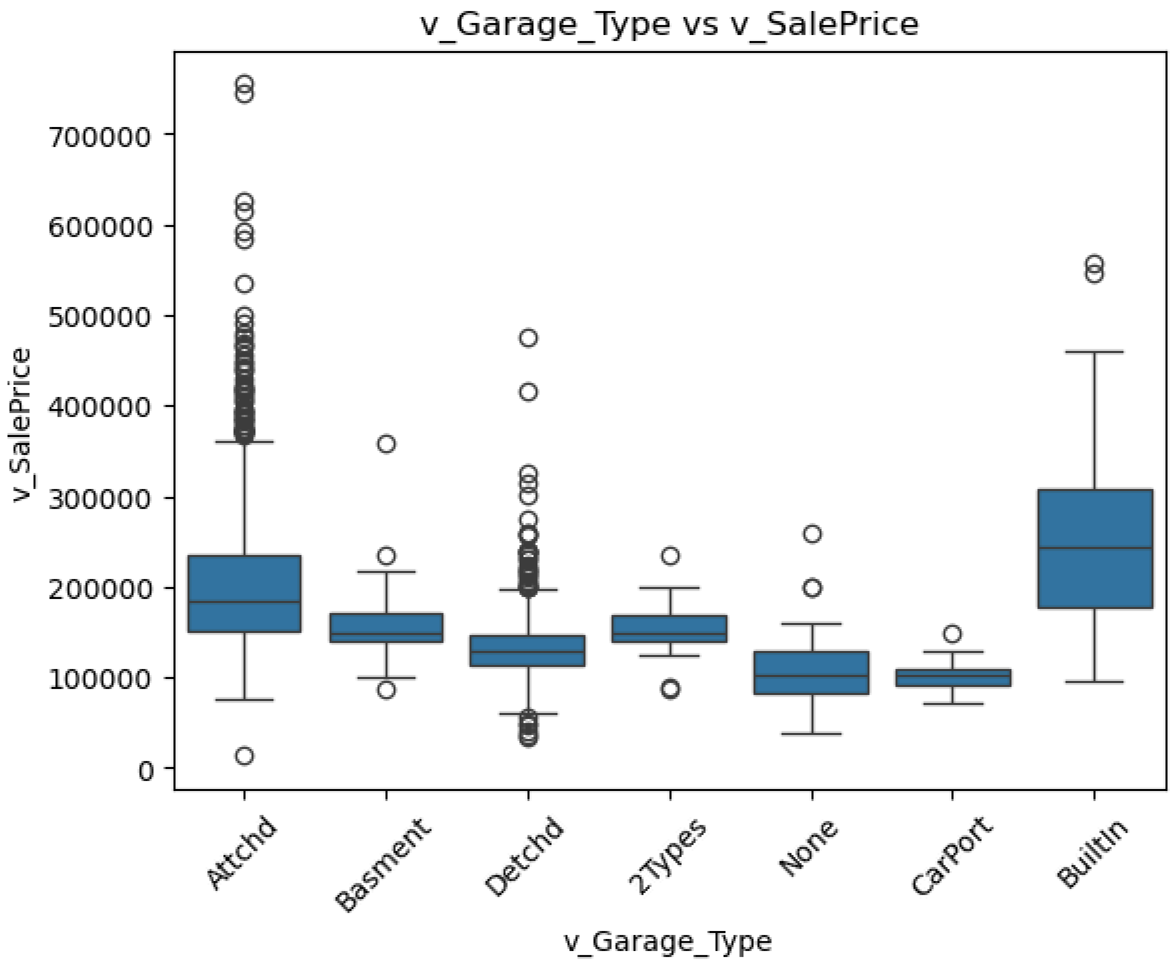
v_Central_Air vs v_SalePrice
v_SalePrice
0
100000
200000
300000
400000
500000
600000
700000
Y
N
v_Central_Air


v_Electrical vs v_SalePrice
v_SalePrice
700000
600000
500000
400000
300000
200000
100000
0
SBrkr
FuseF
FuseA
FuseP
Mix
None
v_Electrical

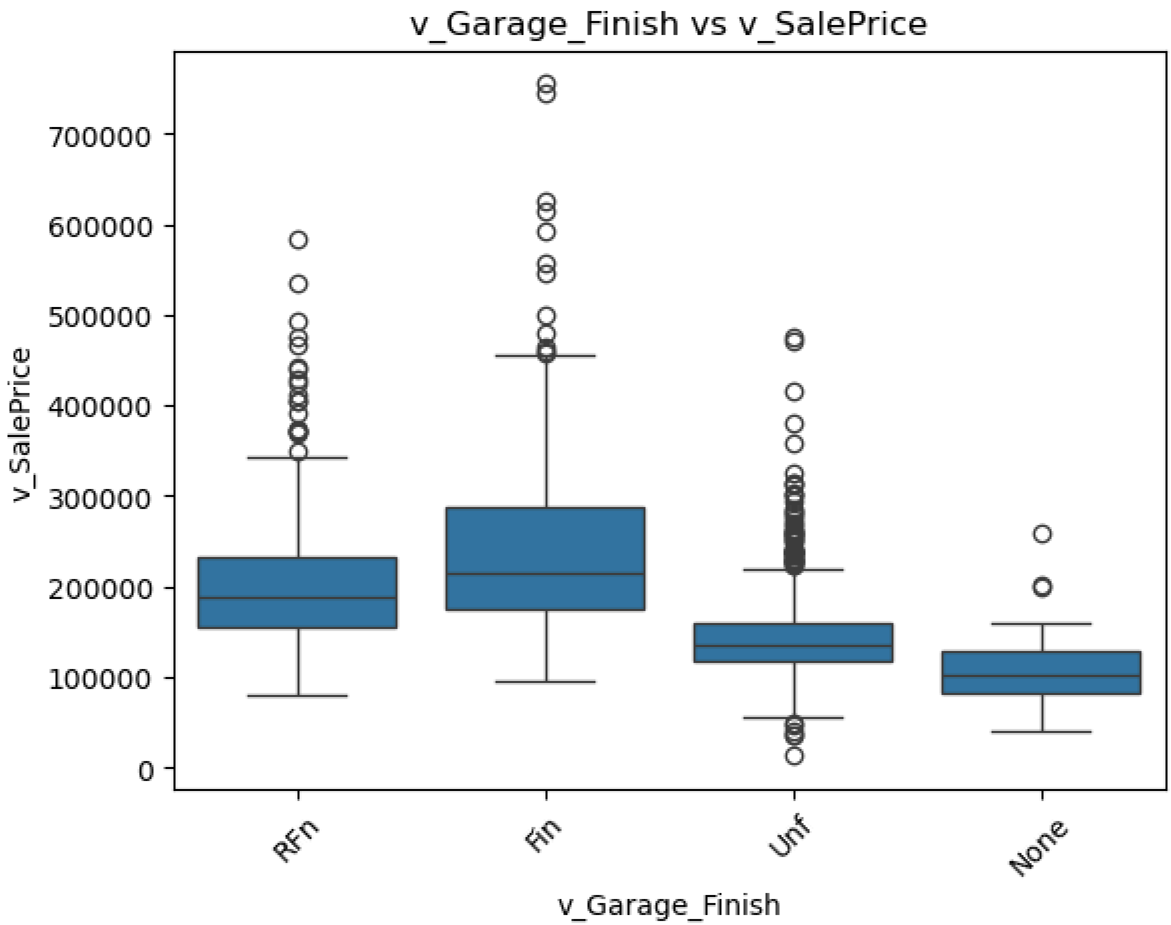

v_Kitchen_Qual vs v_SalePrice
v_SalePrice
700000
600000
500000
400000
300000
200000
100000
0
Gd
Ex
TA
Fa
v_Kitchen_Qual

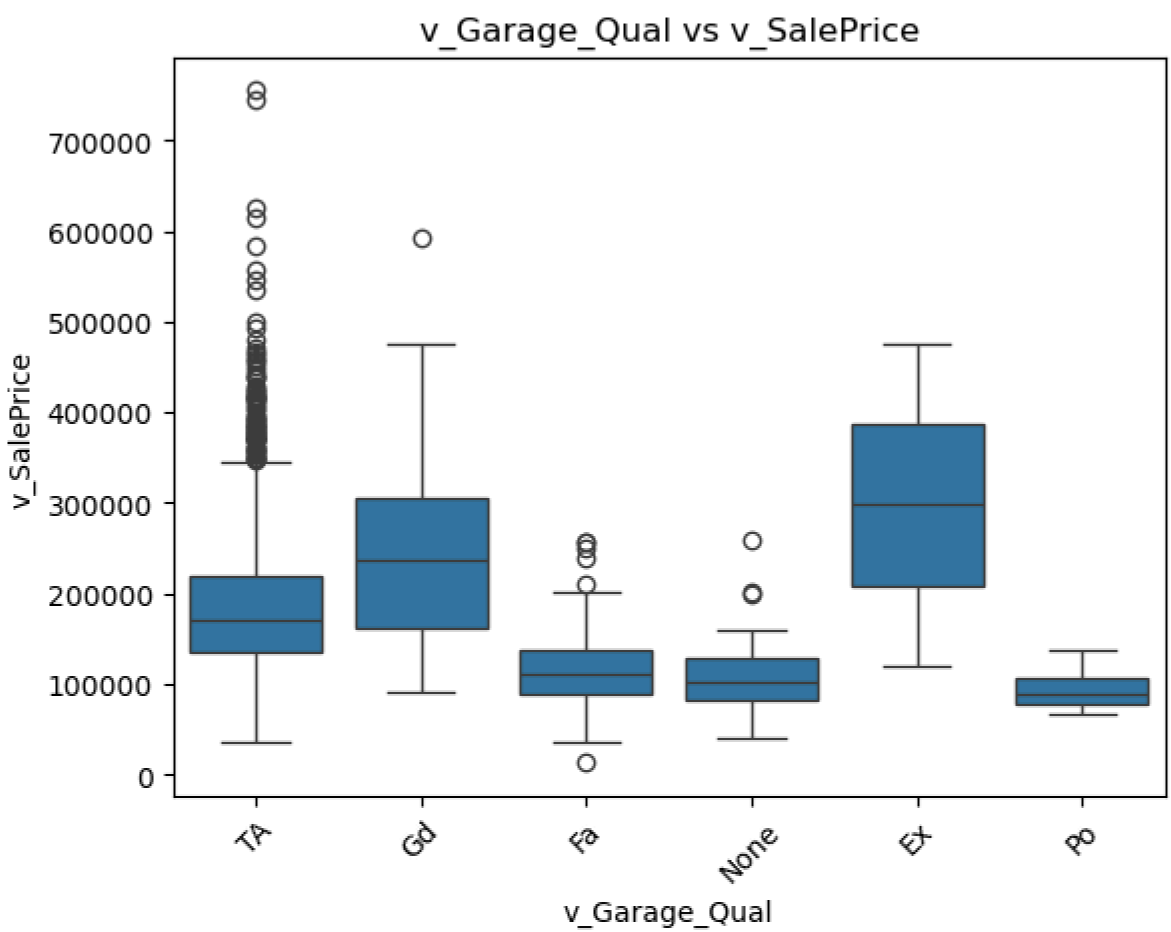

v_Functional vs v_SalePrice
v_SalePrice
700000
600000
500000
400000
300000
200000
100000
0
Typ
Min1
Maj1
Min2
Mod
Sal
Maj2
Sev
v_Functional

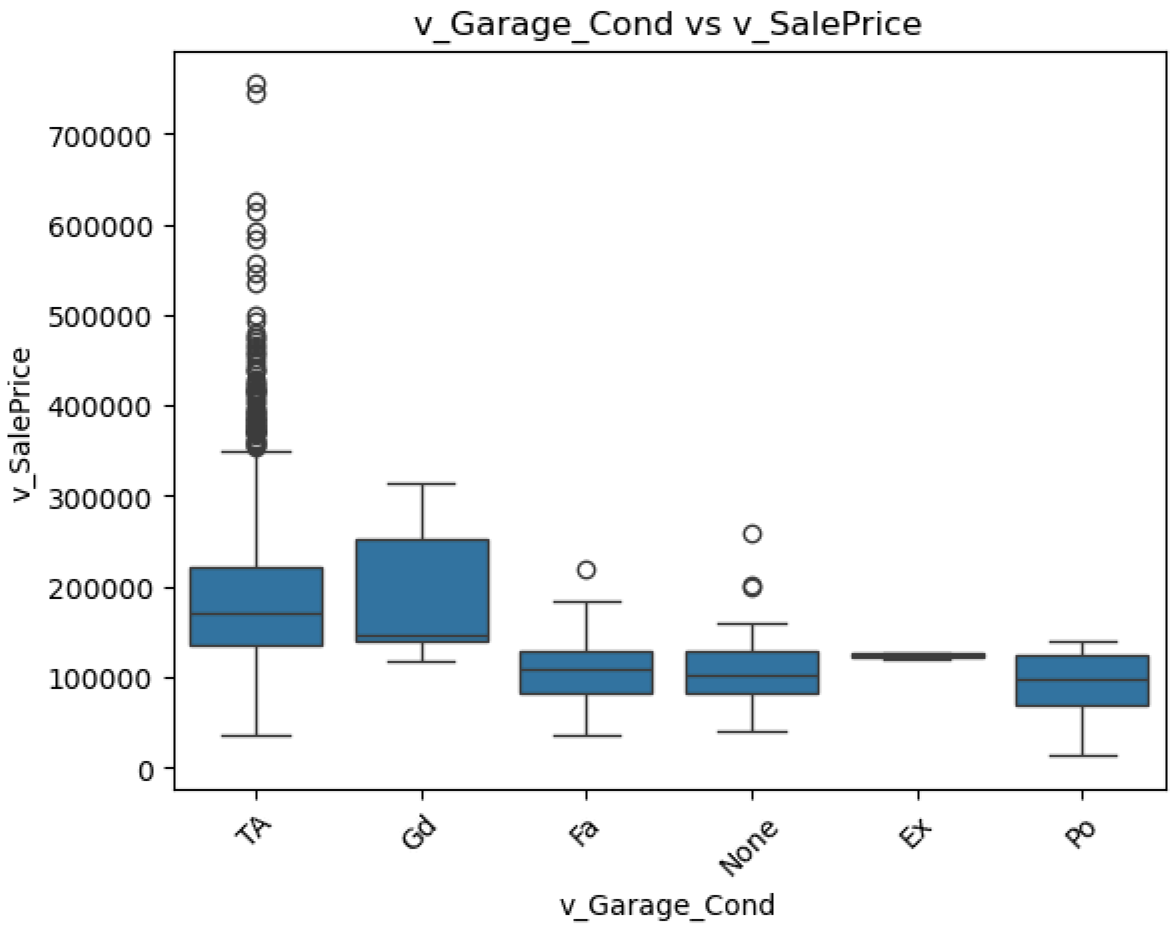

v_Fireplace_Qu vs v_SalePrice
v_SalePrice
700000
600000
500000
400000
300000
200000
100000
0
Gd
None
TA
Ex
Fa
Po
v_Fireplace_Qu

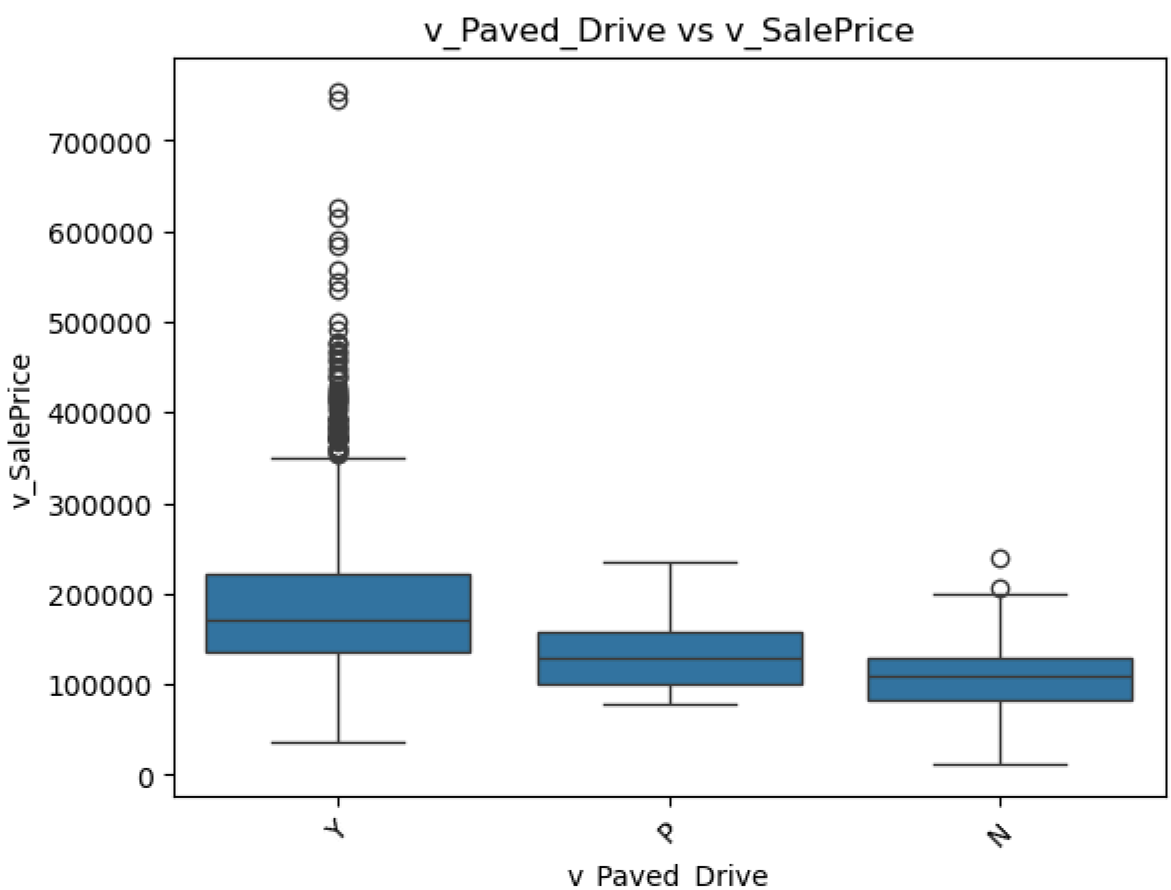


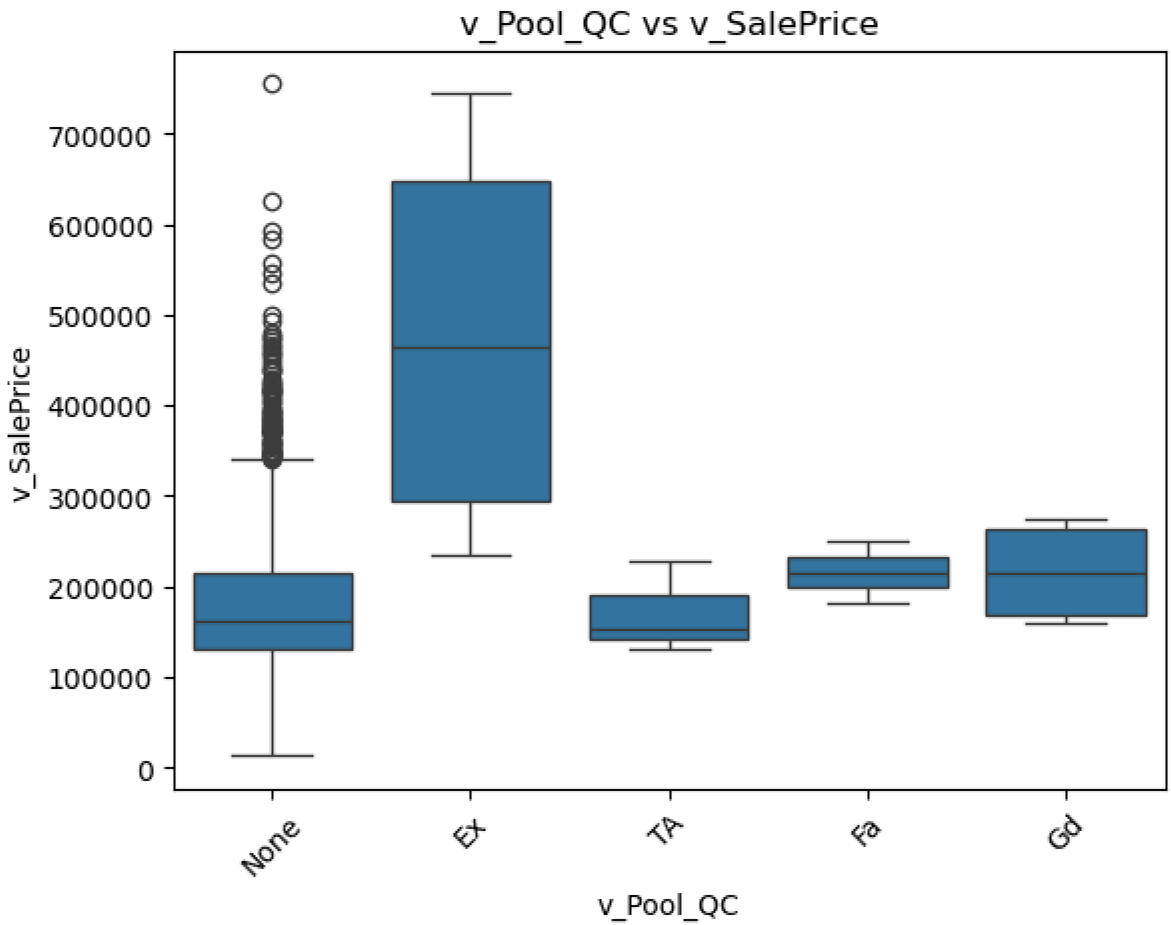
v_Garage_Type vs v_SalePrice
v_SalePrice
700000
600000
500000
400000
300000
200000
100000
0
Attchd
Basment
Detchd
2Types
None
CarPort
BuiltIn
v_Garage_Type

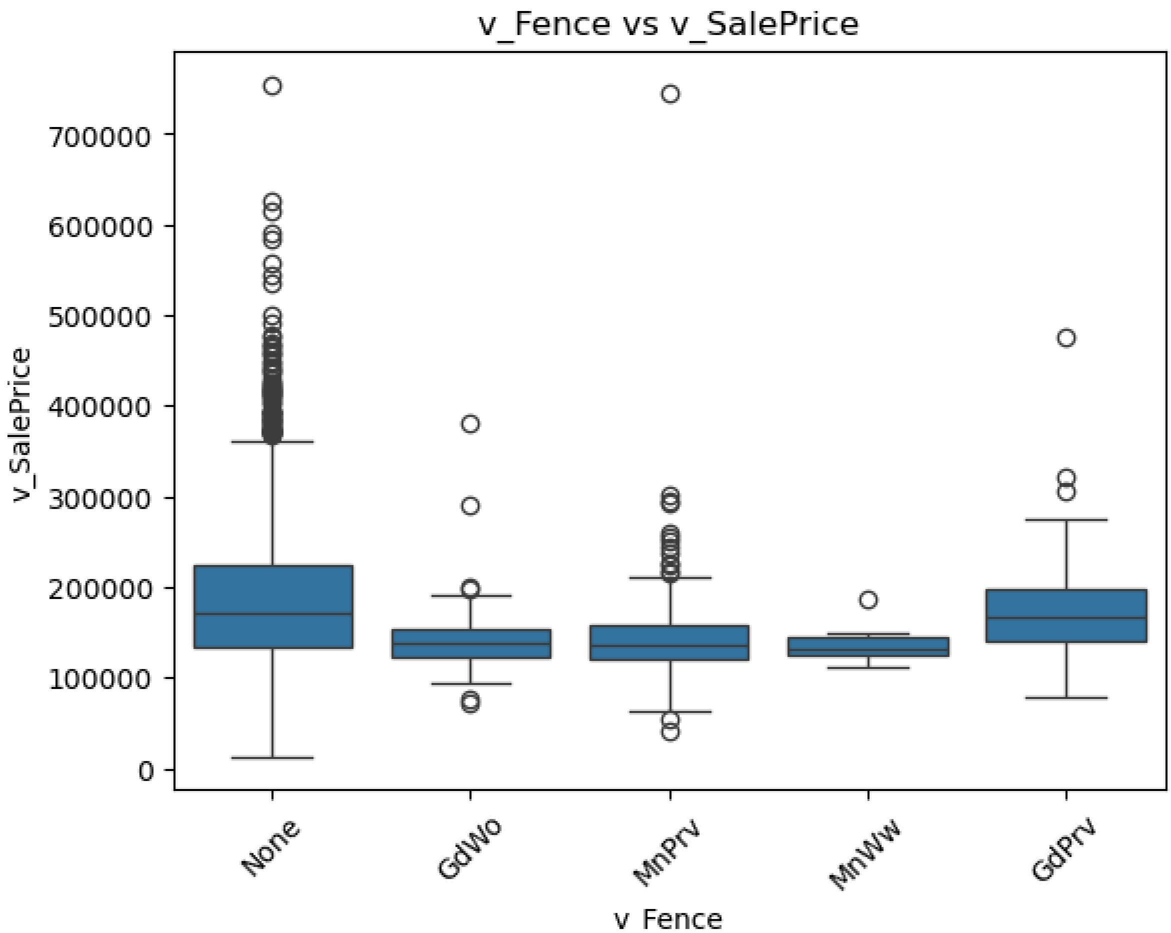

v_Garage_Finish vs v_SalePrice
v_SalePrice
0
100000
200000
300000
400000
500000
600000
700000
RFn
Fin
Unf
None
v_Garage_Finish


v_Garage_Qual vs v_SalePrice
v_SalePrice
0
100000
200000
300000
400000
500000
600000
700000
TA
Gd
Fa
None
Ex
Po
v_Garage_Qual

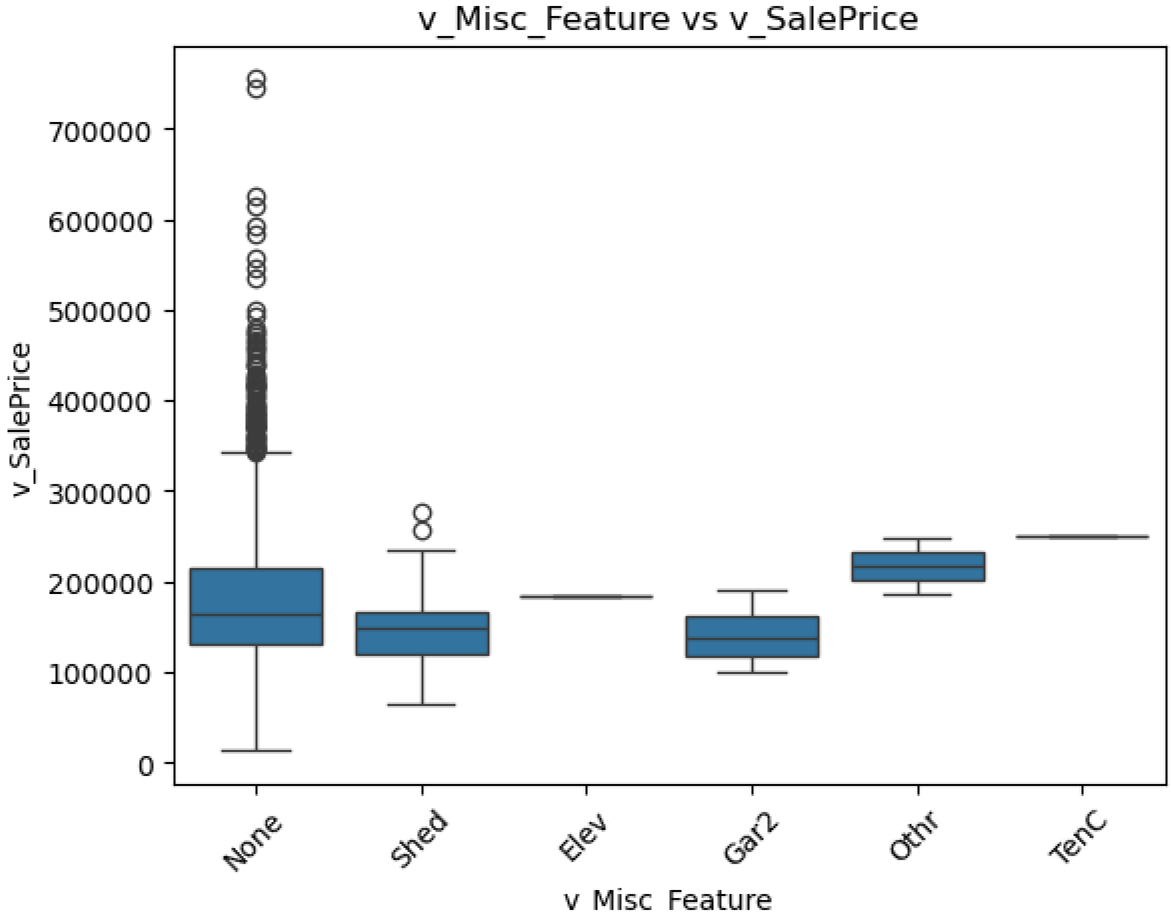

v_Garage_Cond vs v_SalePrice
v_SalePrice
700000
600000
500000
400000
300000
200000
100000
0
TA
Gd
Fa
None
Ex
Po
v_Garage_Cond

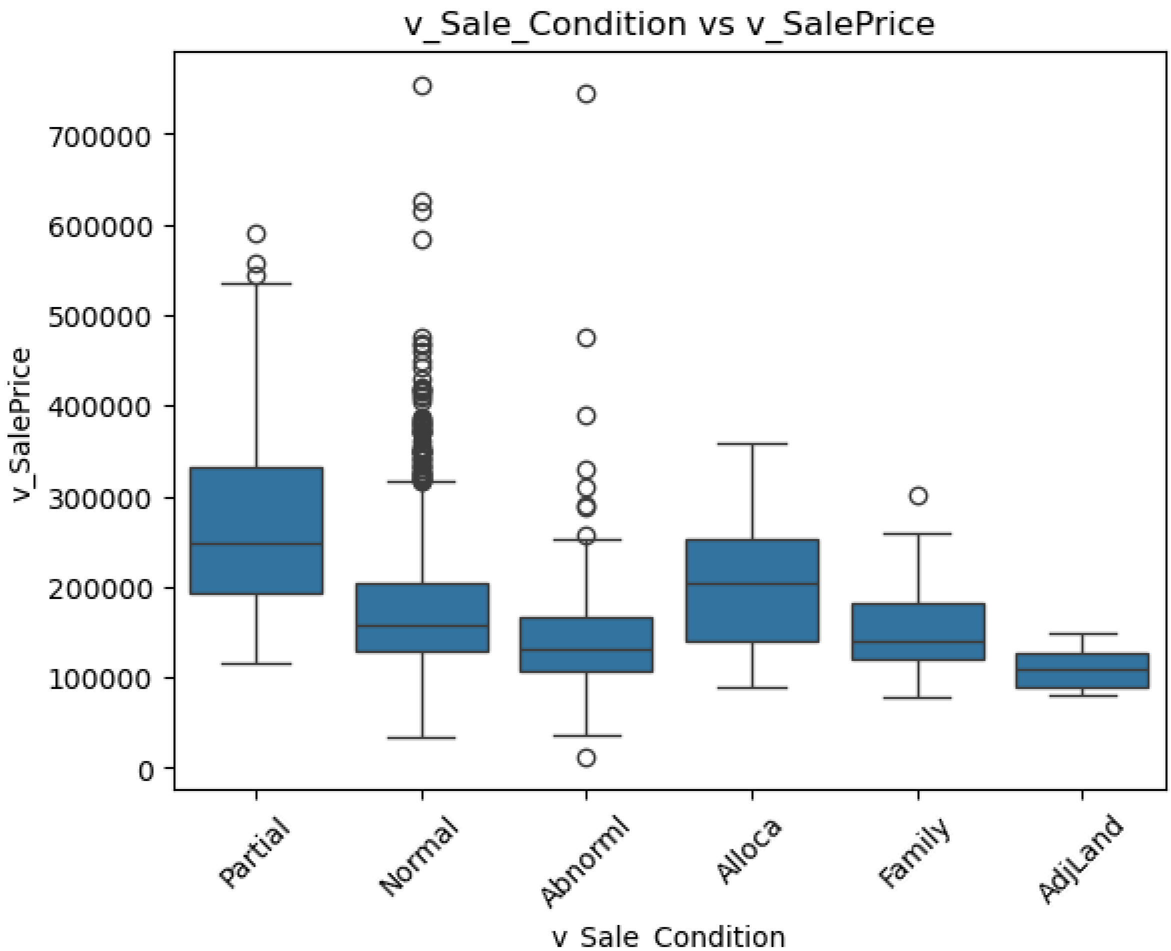

v_Paved_Drive vs v_SalePrice
v_SalePrice
700000
600000
500000
400000
300000
200000
100000
0
Y
P
N
v_Paved_Drive


v_Pool_QC vs v_SalePrice
v_SalePrice
0
100000
200000
300000
400000
500000
600000
700000
None
Ex
TA
Fa
Gd
v_Pool_QC



v_Fence vs v_SalePrice
700000
600000
500000
400000
300000
200000
100000
0
v_SalePrice
None
GdWo
MnPrv
MnWw
GdPrv
v_Fence


v_Misc_Feature vs v_SalePrice
v_SalePrice
700000
600000
500000
400000
300000
200000
100000
0
None
Shed
Elev
Gar2
Othr
TenC
v_Misc_Feature



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.

Note: If you are using VS Code, these might not display correctly. Add a "\" in front of the underscores in the variable names, so `\text{v_Lot_Area}` becomes `\text{v\_Lot\_Area}`.

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? One student in five years has beat that.

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

In [21]:
```
df = (
    df
    .assign(
        l_Lot_Area = np.log(df['v_Lot_Area']),
        l_SalePrice = np.log(df['v_SalePrice']),
        Yr_Sold = df['v_Yr_Sold'],
        const = 1
    )
)

df['Qual_bins'] = pd.cut(
    df['v_Overall_Qual'],
    bins=[0, 4, 6, 8, 10],
    labels=['Low', 'Medium', 'High', 'Very High']
)
```

In [22]:
```
import statsmodels.formula.api as smf
from statsmodels.iolib.summary2 import summary_col

df['log_SalePrice'] = np.log(df['v_SalePrice'])
df['log_Lot_Area'] = np.log(df['v_Lot_Area'])

reg1 = smf.ols('v_SalePrice ~ v_Lot_Area', data=df).fit()
reg2 = smf.ols('v_SalePrice ~ log_Lot_Area', data=df).fit()
reg3 = smf.ols('log_SalePrice ~ v_Lot_Area', data=df).fit()
reg4 = smf.ols('log_SalePrice ~ log_Lot_Area', data=df).fit()
reg5 = smf.ols('log_SalePrice ~ v_Yr_Sold', data=df).fit()
reg6 = smf.ols('log_SalePrice ~ C(v_Yr_Sold)', data=df).fit()
reg7 = smf.ols('np.log(v_SalePrice) ~ v_Overall_Qual + v_Gr_Liv_Area + v_Garage_Car

summary_col([reg1, reg2, reg3, reg4, reg5, reg6, reg7],
            stars=True,
            model_names=['SP~Area', 'SP~log(Area)', 'log(SP)~Area', 'log(SP)~log(Ar
```

Out[22]:

	SP~Area	SP~log(Area)	log(SP)~Area	log(SP)~log(Area)	lo
Intercept	154789.5502***	-327915.8023***	11.8941***	9.4051***	
	(2911.5906)	(30221.3471)	(0.0146)	(0.1511)	
v_Lot_Area	2.6489***		0.0000***		
	(0.2252)		(0.0000)		
log_Lot_Area		56028.1700***		0.2883***	
		(3315.1392)		(0.0166)	
v_Yr_Sold					
C(v_Yr_Sold)[T.2007]					
C(v_Yr_Sold)[T.2008]					
v_Overall_Qual					
v_Gr_Liv_Area					
v_Garage_Cars					
v_Total_Bsmt_SF					
v_1st_Flr_SF					
R-squared	0.0666	0.1284	0.0646	0.1350	
R-squared Adj.	0.0661	0.1279	0.0641	0.1345	

Standard errors in parentheses.
* p<.1, ** p<.05, ***p<.01

1. $\text{Sale Price}_{i,t} = 154789.55 + 2.6489 * \text{v\_Lot\_Area}$
2. $\text{Sale Price}_{i,t} = -327915.80 + 56028.17 * log(\text{v\_Lot\_Area})$
3. $log(\text{Sale Price}_{i,t}) = 11.8941 + 0.00 * \text{v\_Lot\_Area}$
4. $log(\text{Sale Price}_{i,t}) = 9.4051 + 0.2883 * log(\text{v\_Lot\_Area})$

5. $log(\text{Sale Price}_{i,t}) = 22.2932 - 0.0051 * \text{v\_Yr\_Sold}$
6. $log(\text{Sale Price}_{i,t}) = 12.0229 + 0.0256 * (\text{v\_Yr\_Sold==2007}) - 0.0103 * (\text{v\_}$
7. $log(\text{Sale Price}_{i,t}) = 10.5440 + 0.1383 \cdot v\_Overall\_Qual + 0.0002 \cdot v\_Gr\_Liv\_$
   $\cdot v\_1st\_Flr\_SF)$

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 [25]:
```
print("Model 1 β1:", reg1.params[1])
print("Model 2 β1:", reg2.params[1])
print("Model 3 β1:", reg3.params[1])
print("Model 4 β1:", reg4.params[1])
print("Model 5 β1:", reg5.params[1])
print("Model 6 β1:", reg6.params['C(v_Yr_Sold)[T.2007]'])
print("Model 6 β2:", reg6.params['C(v_Yr_Sold)[T.2008]'])
print("Model 7 β1 - β5:")
print(reg7.params[1:6])
```

```
Model 1 β1: 2.648935000718191
Model 2 β1: 56028.16996046537
Model 3 β1: 1.3092338465836504e-05
Model 4 β1: 0.28826331962293017
Model 5 β1: -0.005114348195977281
Model 6 β1: 0.025590319971647263
Model 6 β2: -0.010281565074487964
Model 7 β1 - β5:
v_Overall_Qual     0.138264
v_Gr_Liv_Area      0.000191
v_Garage_Cars      0.105987
v_Total_Bsmt_SF    0.000098
v_1st_Flr_SF       0.000052
dtype: float64
```

```
C:\Users\lenovo\AppData\Local\Temp\ipykernel_52060\2003613305.py:1: FutureWarning: S
eries.__getitem__ treating keys as positions is deprecated. In a future version, int
eger keys will always be treated as labels (consistent with DataFrame behavior). To
access a value by position, use `ser.iloc[pos]`
  print("Model 1 β1:", reg1.params[1])
C:\Users\lenovo\AppData\Local\Temp\ipykernel_52060\2003613305.py:2: FutureWarning: S
eries.__getitem__ treating keys as positions is deprecated. In a future version, int
eger keys will always be treated as labels (consistent with DataFrame behavior). To
access a value by position, use `ser.iloc[pos]`
  print("Model 2 β1:", reg2.params[1])
C:\Users\lenovo\AppData\Local\Temp\ipykernel_52060\2003613305.py:3: FutureWarning: S
eries.__getitem__ treating keys as positions is deprecated. In a future version, int
eger keys will always be treated as labels (consistent with DataFrame behavior). To
access a value by position, use `ser.iloc[pos]`
  print("Model 3 β1:", reg3.params[1])
C:\Users\lenovo\AppData\Local\Temp\ipykernel_52060\2003613305.py:4: FutureWarning: S
eries.__getitem__ treating keys as positions is deprecated. In a future version, int
eger keys will always be treated as labels (consistent with DataFrame behavior). To
access a value by position, use `ser.iloc[pos]`
  print("Model 4 β1:", reg4.params[1])
C:\Users\lenovo\AppData\Local\Temp\ipykernel_52060\2003613305.py:5: FutureWarning: S
eries.__getitem__ treating keys as positions is deprecated. In a future version, int
eger keys will always be treated as labels (consistent with DataFrame behavior). To
access a value by position, use `ser.iloc[pos]`
  print("Model 5 β1:", reg5.params[1])
```

2. If v_Lot_Area goes up by 1%, the house price increases by about $560.

3. If v_Lot_Area increases by 1 unit, the house price goes up by about 0.00139%.

4. I think Model 4 is the best because it uses the log form and has the highest R-squared among the four models.

5. It means that for every year increase, the log of SalePrice decreases by about 0.0051.

6. α represents the log average of house sale prices in 2006 (with 2006 as the baseline year), which means the average sale price in 2006 was approximately $165,500.

7. β_1 represents the average percentage change in house prices in 2007 compared to 2006. That means house prices in 2007 were on average about 2.56% higher than in 2006.

8. Model 6 has a higher R^2 because it considers the effects of 2007 and 2008 separately, making it more detailed. Model 5 only treats the year as a continuous variable and doesn't capture the specific differences between years. So Model 6 explains the changes in housing prices better.

9. In model 7, I included: v_Overall_Qual, v_Gr_Liv_Area, v_Garage_Cars, v_Total_Bsmt_SF, v_1st_Flr_SF

10. The R^2 in Model 7 is 0.8024

11. I think Model 6 is not suitable for prediction because it uses two dummy variables for the years 2007 and 2008.

12. I think Model 5 is not suitable for prediction because its R-squared is 0.0014 and its adjusted R-squared is negative. This means the model has no predictive power.

In [ ]:
```

```