Week 7 :

House Price Prediction

Que : **Data Preprocessing and feature engineering**

Importing Libraries and Dataset

In [ ]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import scipy.stats as stats
from catboost import CatBoostRegressor
from lightgbm import LGBMRegressor
from sklearn.ensemble import RandomForestRegressor, RandomForestClassifier
from sklearn.exceptions import ConvergenceWarning
from sklearn.linear_model import LinearRegression
from sklearn.neighbors import KNeighborsRegressor
from sklearn.svm import SVR
from sklearn.tree import DecisionTreeRegressor
from xgboost import XGBRegressor
from sklearn.preprocessing import LabelEncoder, OneHotEncoder, RobustScaler
from sklearn.metrics import mean_squared_error, f1_score, mean_absolute_error, r2_score
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split, cross_val_score, GridSearchCV
from sklearn.impute import SimpleImputer, KNNImputer
```

In [ ]:
```
pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', None)
pd.set_option('display.width', None)
pd.set_option('display.float_format', lambda x: '%.3f' % x)
```

In [ ]:
```
def load():
    df_train = pd.read_csv("Dataset/train.csv")
    df_test = pd.read_csv("Dataset/test.csv")
    df_merged = pd.concat([df_train, df_test]).reset_index(drop=True)

    return df_merged, df_train, df_test

df, df_train, df_test = load()

df.head(10)
```

Out[ ]:

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities
0	1	60	RL	65.000	8450	Pave	NaN	Reg	Lvl	AllPub
1	2	20	RL	80.000	9600	Pave	NaN	Reg	Lvl	AllPub
2	3	60	RL	68.000	11250	Pave	NaN	IR1	Lvl	AllPub
3	4	70	RL	60.000	9550	Pave	NaN	IR1	Lvl	AllPub
4	5	60	RL	84.000	14260	Pave	NaN	IR1	Lvl	AllPub
5	6	50	RL	85.000	14115	Pave	NaN	IR1	Lvl	AllPub
6	7	20	RL	75.000	10084	Pave	NaN	Reg	Lvl	AllPub
7	8	60	RL	NaN	10382	Pave	NaN	IR1	Lvl	AllPub
8	9	50	RM	51.000	6120	Pave	NaN	Reg	Lvl	AllPub
9	10	190	RL	50.000	7420	Pave	NaN	Reg	Lvl	AllPub

In [ ]:
```
df.tail(10)
```

Out[ ]:

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Uti
2909	2910	180	RM	21.000	1470	Pave	NaN	Reg	Lvl	A
2910	2911	160	RM	21.000	1484	Pave	NaN	Reg	Lvl	A
2911	2912	20	RL	80.000	13384	Pave	NaN	Reg	Lvl	A
2912	2913	160	RM	21.000	1533	Pave	NaN	Reg	Lvl	A
2913	2914	160	RM	21.000	1526	Pave	NaN	Reg	Lvl	A
2914	2915	160	RM	21.000	1936	Pave	NaN	Reg	Lvl	A
2915	2916	160	RM	21.000	1894	Pave	NaN	Reg	Lvl	A
2916	2917	20	RL	160.000	20000	Pave	NaN	Reg	Lvl	A
2917	2918	85	RL	62.000	10441	Pave	NaN	Reg	Lvl	A
2918	2919	60	RL	74.000	9627	Pave	NaN	Reg	Lvl	A

Overview & Preprocessing

In [ ]:
```
def check_df(dataframe, head=5):
    print("##################### Shape #####################")
    print(dataframe.shape)
    print("##################### Types #####################")
    print(dataframe.dtypes)
    print("##################### Duplicated Values #####################")
    print(dataframe.duplicated().sum())
    print("##################### Missing Values #####################")
    print(dataframe.isnull().sum())
    print("##################### Number of Unique Values #####################")
    print(df.nunique())
```

In [ ]:
```
check_df(df)
```

```
##################### Shape #####################
(2919, 81)
##################### Types #####################
Id                int64
MSSubClass        int64
MSZoning         object
LotFrontage     float64
LotArea           int64
Street           object
Alley            object
LotShape         object
LandContour      object
Utilities        object
LotConfig        object
LandSlope        object
Neighborhood     object
Condition1       object
Condition2       object
BldgType         object
HouseStyle       object
OverallQual       int64
OverallCond       int64
YearBuilt         int64
YearRemodAdd      int64
RoofStyle        object
RoofMatl         object
Exterior1st      object
Exterior2nd      object
MasVnrType       object
MasVnrArea      float64
ExterQual        object
ExterCond        object
Foundation       object
BsmtQual         object
BsmtCond         object
BsmtExposure     object
BsmtFinType1     object
BsmtFinSF1      float64
BsmtFinType2     object
BsmtFinSF2      float64
BsmtUnfSF       float64
TotalBsmtSF     float64
Heating          object
HeatingQC        object
CentralAir       object
Electrical       object
1stFlrSF          int64
2ndFlrSF          int64
LowQualFinSF      int64
GrLivArea         int64
BsmtFullBath    float64
BsmtHalfBath    float64
FullBath          int64
HalfBath          int64
BedroomAbvGr      int64
KitchenAbvGr      int64
KitchenQual      object
TotRmsAbvGrd      int64
Functional       object
Fireplaces        int64
FireplaceQu      object
GarageType       object
GarageYrBlt     float64
GarageFinish     object
GarageCars      float64
GarageArea      float64
```

```
GarageQual        object
GarageCond        object
PavedDrive        object
WoodDeckSF         int64
OpenPorchSF        int64
EnclosedPorch      int64
3SsnPorch          int64
ScreenPorch        int64
PoolArea           int64
PoolQC            object
Fence             object
MiscFeature       object
MiscVal            int64
MoSold             int64
YrSold             int64
SaleType          object
SaleCondition     object
SalePrice        float64
dtype: object
###################### Duplicated Values ######################
0
###################### Missing Values ######################
Id                  0
MSSubClass          0
MSZoning            4
LotFrontage       486
LotArea             0
Street              0
Alley            2721
LotShape            0
LandContour         0
Utilities           2
LotConfig           0
LandSlope           0
Neighborhood        0
Condition1          0
Condition2          0
BldgType            0
HouseStyle          0
OverallQual         0
OverallCond         0
YearBuilt           0
YearRemodAdd        0
RoofStyle           0
RoofMatl            0
Exterior1st         1
Exterior2nd         1
MasVnrType       1766
MasVnrArea         23
ExterQual           0
ExterCond           0
Foundation          0
BsmtQual           81
BsmtCond           82
BsmtExposure       82
BsmtFinType1       79
BsmtFinSF1          1
BsmtFinType2       80
BsmtFinSF2          1
BsmtUnfSF           1
TotalBsmtSF         1
Heating             0
HeatingQC           0
CentralAir          0
Electrical          1
1stFlrSF            0
```

```
2ndFlrSF            0
LowQualFinSF        0
GrLivArea           0
BsmtFullBath        2
BsmtHalfBath        2
FullBath            0
HalfBath            0
BedroomAbvGr        0
KitchenAbvGr        0
KitchenQual         1
TotRmsAbvGrd        0
Functional          2
Fireplaces          0
FireplaceQu      1420
GarageType        157
GarageYrBlt       159
GarageFinish      159
GarageCars          1
GarageArea          1
GarageQual        159
GarageCond        159
PavedDrive          0
WoodDeckSF          0
OpenPorchSF         0
EnclosedPorch       0
3SsnPorch           0
ScreenPorch         0
PoolArea            0
PoolQC           2909
Fence            2348
MiscFeature      2814
MiscVal             0
MoSold              0
YrSold              0
SaleType            1
SaleCondition       0
SalePrice        1459
dtype: int64
###################### Number of Unique Values ####################
Id               2919
MSSubClass         16
MSZoning            5
LotFrontage       128
LotArea          1951
Street              2
Alley               2
LotShape            4
LandContour         4
Utilities           2
LotConfig           5
LandSlope           3
Neighborhood       25
Condition1          9
Condition2          8
BldgType            5
HouseStyle          8
OverallQual        10
OverallCond         9
YearBuilt         118
YearRemodAdd       61
RoofStyle           6
RoofMatl            8
Exterior1st        15
Exterior2nd        16
MasVnrType          3
MasVnrArea        444
```

```
ExterQual          4
ExterCond          5
Foundation         6
BsmtQual           4
BsmtCond           4
BsmtExposure       4
BsmtFinType1       6
BsmtFinSF1       991
BsmtFinType2       6
BsmtFinSF2       272
BsmtUnfSF       1135
TotalBsmtSF     1058
Heating            6
HeatingQC          5
CentralAir         2
Electrical         5
1stFlrSF        1083
2ndFlrSF         635
LowQualFinSF      36
GrLivArea       1292
BsmtFullBath       4
BsmtHalfBath       3
FullBath           5
HalfBath           3
BedroomAbvGr       8
KitchenAbvGr       4
KitchenQual        4
TotRmsAbvGrd      14
Functional         7
Fireplaces         5
FireplaceQu        5
GarageType         6
GarageYrBlt      103
GarageFinish       3
GarageCars         6
GarageArea       603
GarageQual         5
GarageCond         5
PavedDrive         3
WoodDeckSF       379
OpenPorchSF      252
EnclosedPorch    183
3SsnPorch         31
ScreenPorch      121
PoolArea          14
PoolQC             3
Fence              4
MiscFeature        4
MiscVal           38
MoSold            12
YrSold             5
SaleType           9
SaleCondition      6
SalePrice        663
dtype: int64
```

In [ ]:
```
df.describe([0, 0.05, 0.50, 0.95, 0.99, 1]).T
```

Out[ ]:

	count	mean	std	min	0%	5%	50%	9
Id	2919.000	1460.000	842.787	1.000	1.000	146.900	1460.000	2773.
MSSubClass	2919.000	57.138	42.518	20.000	20.000	20.000	50.000	160.
LotFrontage	2433.000	69.306	23.345	21.000	21.000	32.000	68.000	107.
LotArea	2919.000	10168.114	7886.996	1300.000	1300.000	3182.000	9453.000	17142.
OverallQual	2919.000	6.089	1.410	1.000	1.000	4.000	6.000	8.
OverallCond	2919.000	5.565	1.113	1.000	1.000	4.000	5.000	8.
YearBuilt	2919.000	1971.313	30.291	1872.000	1872.000	1915.000	1973.000	2007.
YearRemodAdd	2919.000	1984.264	20.894	1950.000	1950.000	1950.000	1993.000	2007.
MasVnrArea	2896.000	102.201	179.334	0.000	0.000	0.000	0.000	466.
BsmtFinSF1	2918.000	441.423	455.611	0.000	0.000	0.000	368.500	1274.
BsmtFinSF2	2918.000	49.582	169.206	0.000	0.000	0.000	0.000	435.
BsmtUnfSF	2918.000	560.772	439.544	0.000	0.000	0.000	467.000	1474.
TotalBsmtSF	2918.000	1051.778	440.766	0.000	0.000	455.250	989.500	1776.
1stFlrSF	2919.000	1159.582	392.362	334.000	334.000	665.900	1082.000	1830.
2ndFlrSF	2919.000	336.484	428.701	0.000	0.000	0.000	0.000	1131.
LowQualFinSF	2919.000	4.694	46.397	0.000	0.000	0.000	0.000	0.
GrLivArea	2919.000	1500.760	506.051	334.000	334.000	861.000	1444.000	2464.
BsmtFullBath	2917.000	0.430	0.525	0.000	0.000	0.000	0.000	1.
BsmtHalfBath	2917.000	0.061	0.246	0.000	0.000	0.000	0.000	1.
FullBath	2919.000	1.568	0.553	0.000	0.000	1.000	2.000	2.
HalfBath	2919.000	0.380	0.503	0.000	0.000	0.000	0.000	1.
BedroomAbvGr	2919.000	2.860	0.823	0.000	0.000	2.000	3.000	4.
KitchenAbvGr	2919.000	1.045	0.214	0.000	0.000	1.000	1.000	1.
TotRmsAbvGrd	2919.000	6.452	1.569	2.000	2.000	4.000	6.000	9.
Fireplaces	2919.000	0.597	0.646	0.000	0.000	0.000	1.000	2.
GarageYrBlt	2760.000	1978.113	25.574	1895.000	1895.000	1928.000	1979.000	2007.
GarageCars	2918.000	1.767	0.762	0.000	0.000	0.000	2.000	3.
GarageArea	2918.000	472.875	215.395	0.000	0.000	0.000	480.000	856.
WoodDeckSF	2919.000	93.710	126.527	0.000	0.000	0.000	0.000	328.
OpenPorchSF	2919.000	47.487	67.575	0.000	0.000	0.000	26.000	183.
EnclosedPorch	2919.000	23.098	64.244	0.000	0.000	0.000	0.000	176.
3SsnPorch	2919.000	2.602	25.188	0.000	0.000	0.000	0.000	0.
ScreenPorch	2919.000	16.062	56.184	0.000	0.000	0.000	0.000	161.
PoolArea	2919.000	2.252	35.664	0.000	0.000	0.000	0.000	0.
MiscVal	2919.000	50.826	567.402	0.000	0.000	0.000	0.000	0.
MoSold	2919.000	6.213	2.715	1.000	1.000	2.000	6.000	11.

	count	mean	std	min	0%	5%	50%	9
YrSold	2919.000	2007.793	1.315	2006.000	2006.000	2006.000	2008.000	2010.(
SalePrice	1460.000	180921.196	79442.503	34900.000	34900.000	88000.000	163000.000	326100.(

Variables Analysis

In [ ]:
```
def grab_col_names(dataframe, cat_th=10, car_th=20):
    # cat_cols, cat_but_car
    cat_cols = [col for col in dataframe.columns if dataframe[col].dtypes == "O"]
    num_but_cat = [col for col in dataframe.columns if dataframe[col].nunique() < cat_th and
                   dataframe[col].dtypes != "O"]
    cat_but_car = [col for col in dataframe.columns if dataframe[col].nunique() > car_th and
                   dataframe[col].dtypes == "O"]
    cat_cols = cat_cols + num_but_cat
    cat_cols = [col for col in cat_cols if col not in cat_but_car]

    # num_cols
    num_cols = [col for col in dataframe.columns if dataframe[col].dtypes != "O"]
    num_cols = [col for col in num_cols if col not in num_but_cat]

    print(f"Observations: {dataframe.shape[0]}")
    print(f"Variables: {dataframe.shape[1]}")
    print(f"cat_cols: {len(cat_cols)}")
    print(f"num_cols: {len(num_cols)}")
    print(f"cat_but_car: {len(cat_but_car)}")
    print(f"num_but_cat: {len(num_but_cat)}")

    return cat_cols, num_cols, cat_but_car
```

In [ ]:
```
cat_cols, num_cols, cat_but_car = grab_col_names(df, car_th=25)
print("##############")
print(f"Cat_Cols : {cat_cols}")
print("##############")
print(f"Num_Cols : {num_cols}")
print("##############")
print(f"Cat_But_Car : {cat_but_car}")
```

```
Observations: 2919
Variables: 81
cat_cols: 53
num_cols: 28
cat_but_car: 0
num_but_cat: 10
##############
Cat_Cols : ['MSZoning', 'Street', 'Alley', 'LotShape', 'LandContour', 'Utilities', 'LotConfi
g', 'LandSlope', 'Neighborhood', 'Condition1', 'Condition2', 'BldgType', 'HouseStyle', 'RoofSt
yle', 'RoofMatl', 'Exterior1st', 'Exterior2nd', 'MasVnrType', 'ExterQual', 'ExterCond', 'Found
ation', 'BsmtQual', 'BsmtCond', 'BsmtExposure', 'BsmtFinType1', 'BsmtFinType2', 'Heating', 'He
atingQC', 'CentralAir', 'Electrical', 'KitchenQual', 'Functional', 'FireplaceQu', 'GarageTyp
e', 'GarageFinish', 'GarageQual', 'GarageCond', 'PavedDrive', 'PoolQC', 'Fence', 'MiscFeatur
e', 'SaleType', 'SaleCondition', 'OverallCond', 'BsmtFullBath', 'BsmtHalfBath', 'FullBath', 'H
alfBath', 'BedroomAbvGr', 'KitchenAbvGr', 'Fireplaces', 'GarageCars', 'YrSold']
##############
Num_Cols : ['Id', 'MSSubClass', 'LotFrontage', 'LotArea', 'OverallQual', 'YearBuilt', 'YearRem
odAdd', 'MasVnrArea', 'BsmtFinSF1', 'BsmtFinSF2', 'BsmtUnfSF', 'TotalBsmtSF', '1stFlrSF', '2nd
FlrSF', 'LowQualFinSF', 'GrLivArea', 'TotRmsAbvGrd', 'GarageYrBlt', 'GarageArea', 'WoodDeckS
F', 'OpenPorchSF', 'EnclosedPorch', '3SsnPorch', 'ScreenPorch', 'PoolArea', 'MiscVal', 'MoSol
d', 'SalePrice']
##############
Cat_But_Car : []
```

In [ ]:
```
num_cols = [col for col in num_cols if col not in ["Id", "SalePrice"]]

print(f"Num_Cols : {num_cols}")
```

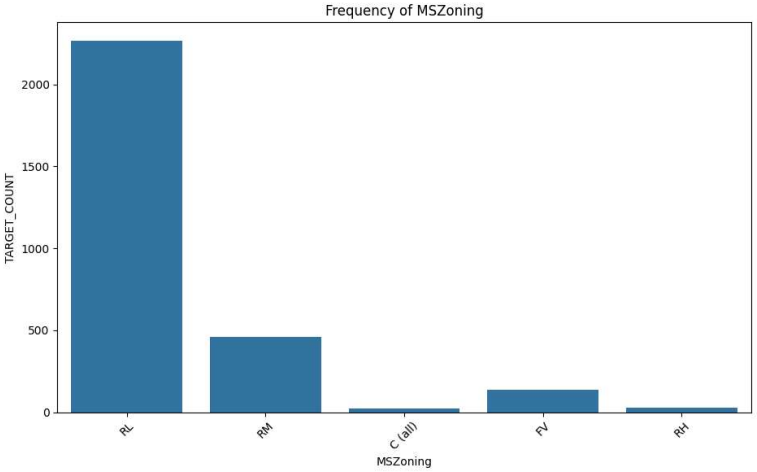
```
Num_Cols : ['MSSubClass', 'LotFrontage', 'LotArea', 'OverallQual', 'YearBuilt', 'YearRemodAd
d', 'MasVnrArea', 'BsmtFinSF1', 'BsmtFinSF2', 'BsmtUnfSF', 'TotalBsmtSF', '1stFlrSF', '2ndFlrS
F', 'LowQualFinSF', 'GrLivArea', 'TotRmsAbvGrd', 'GarageYrBlt', 'GarageArea', 'WoodDeckSF', 'O
penPorchSF', 'EnclosedPorch', '3SsnPorch', 'ScreenPorch', 'PoolArea', 'MiscVal', 'MoSold']
```

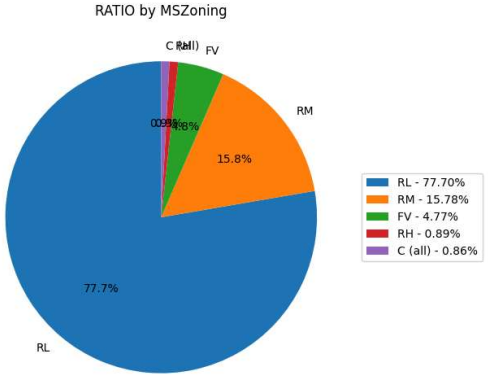
Cat Cols Analysis

In [ ]:
```
def cat_summary(dataframe, col_name, plot=False):

    if dataframe[col_name].dtypes == "bool":
        dataframe[col_name] = dataframe[col_name].astype(int)

        print(pd.DataFrame({col_name: dataframe[col_name].value_counts(),
                            "Ratio": 100 * dataframe[col_name].value_counts() / len(dataframe
        print("##########################################")

        if plot:
            fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(18, 6))

            # TARGET_COUNT
            sns.countplot(x=col_name, data=dataframe, ax=ax1)
            ax1.set_title(f"Frequency of {col_name}")
            ax1.set_ylabel("TARGET_COUNT")
            ax1.tick_params(axis="x", rotation=45)

            # RATIO
            values = dataframe[col_name].value_counts()
            ax2.pie(x=values, labels=values.index, autopct="%1.1f%%", startangle=90)
            ax2.set_title(f"RATIO by {col_name}")
            ax2.legend(labels=[f"{index} - {value/sum(values)*100:.2f}%" for index, value in
                    loc="center left", bbox_to_anchor=(1, 0, 0.5, 1))

            plt.tight_layout()
            plt.show()

    else:
        print(pd.DataFrame({col_name: dataframe[col_name].value_counts(),
                            "Ratio": 100 * dataframe[col_name].value_counts() / len(dataframe
        print("##########################################")

        if plot:
            fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(18, 6))

            # TARGET_COUNT
            sns.countplot(x=col_name, data=dataframe, ax=ax1)
            ax1.set_title(f"Frequency of {col_name}")
            ax1.set_ylabel("TARGET_COUNT")
            ax1.tick_params(axis="x", rotation=45)

            # RATIO
            values = dataframe[col_name].value_counts()
            ax2.pie(x=values, labels=values.index, autopct="%1.1f%%", startangle=90)
            ax2.set_title(f"RATIO by {col_name}")
            ax2.legend(labels=[f"{index} - {value/sum(values)*100:.2f}%" for index, value in
                    loc="center left", bbox_to_anchor=(1, 0, 0.5, 1))

            plt.tight_layout()
            plt.show()
```

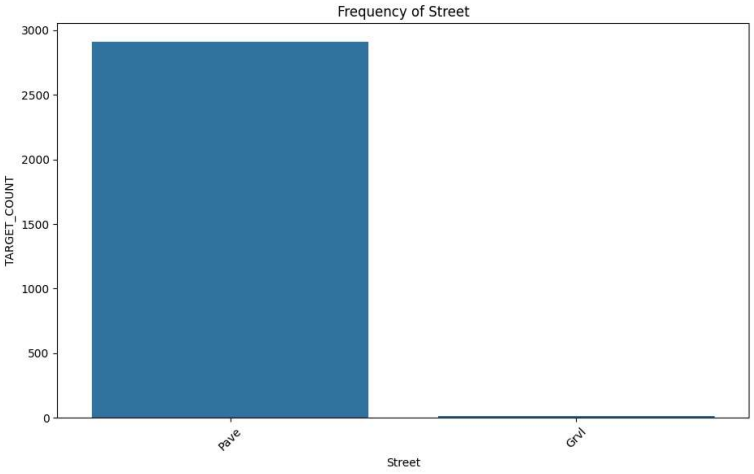
In [ ]:
```
for col in cat_cols:
    cat_summary(df, col, plot=True)
```

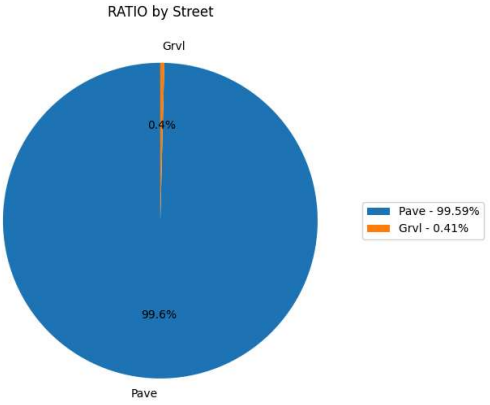
```
          MSZoning  Ratio
MSZoning
RL            2265 77.595
RM             460 15.759
FV             139  4.762
RH              26  0.891
C (all)         25  0.856
##########################################
```



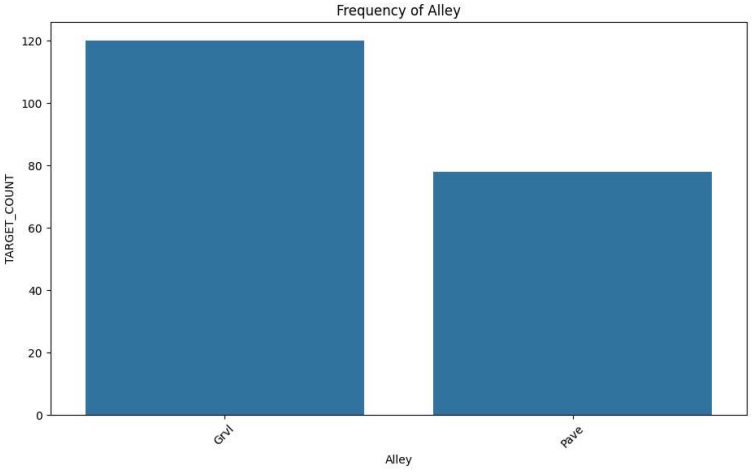


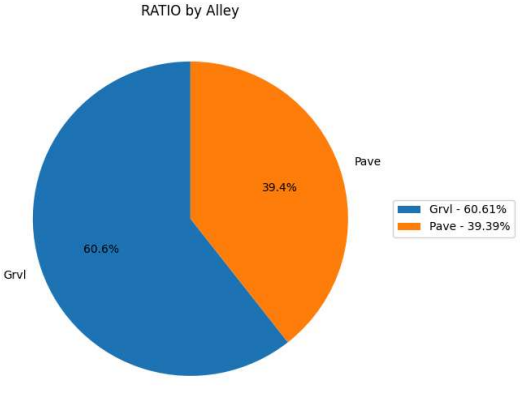

```
        Street  Ratio
Street
Pave      2907 99.589
Grvl        12  0.411
##########################################
```





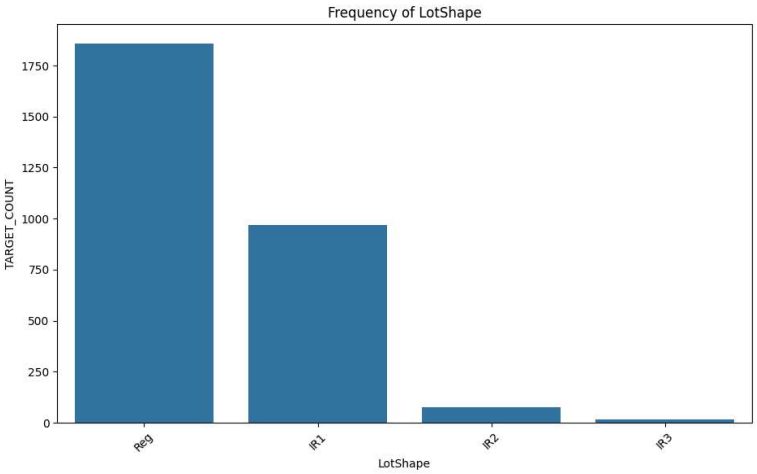

```
       Alley  Ratio
Alley
Grvl     120  4.111
Pave      78  2.672
##########################################
```

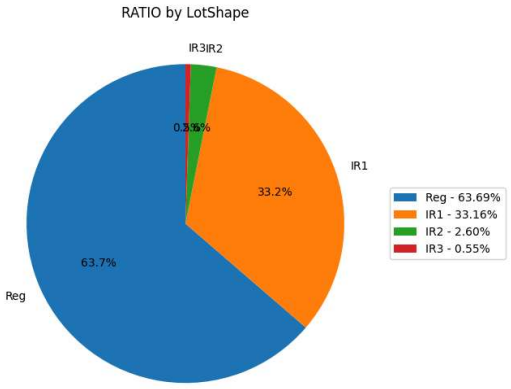




LotShape		Ratio
LotShape		
Reg	1859	63.686
IR1	968	33.162
IR2	76	2.604
IR3	16	0.548

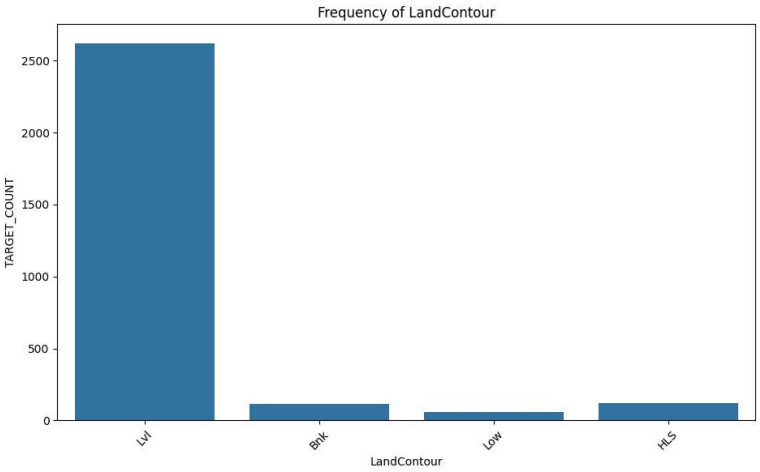
##

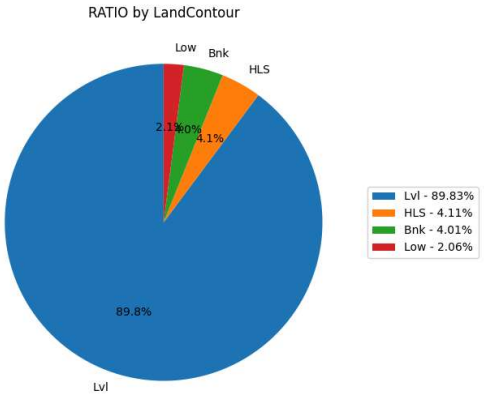




LandContour		Ratio
LandContour		
Lvl	2622	89.825
HLS	120	4.111
Bnk	117	4.008
Low	60	2.055

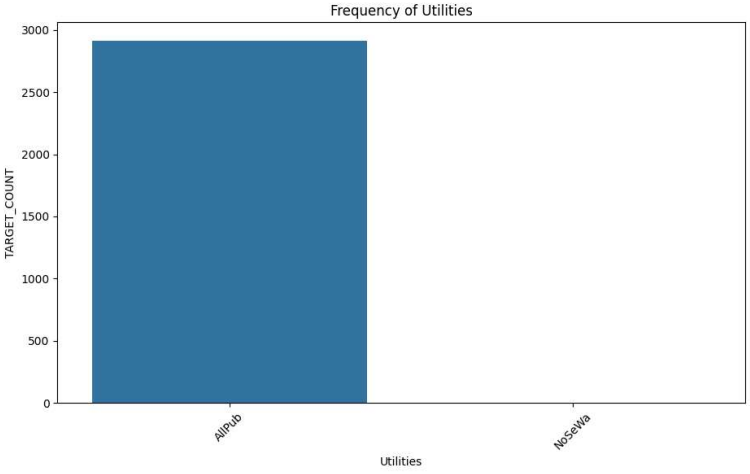
##

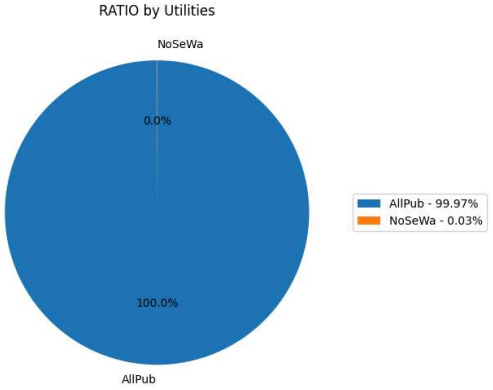




Utilities		Ratio
Utilities		
AllPub	2916	99.897
NoSeWa	1	0.034

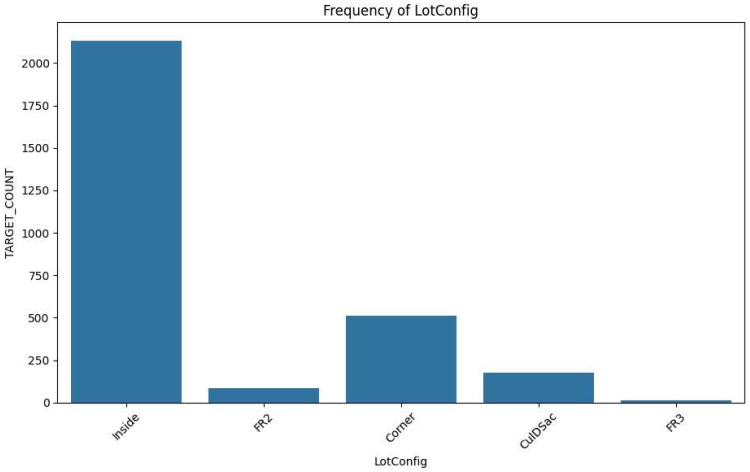
##

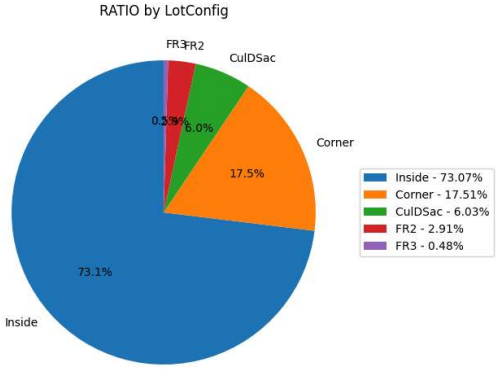




LotConfig		Ratio
LotConfig		
Inside	2133	73.073
Corner	511	17.506
CulDSac	176	6.029
FR2	85	2.912
FR3	14	0.480

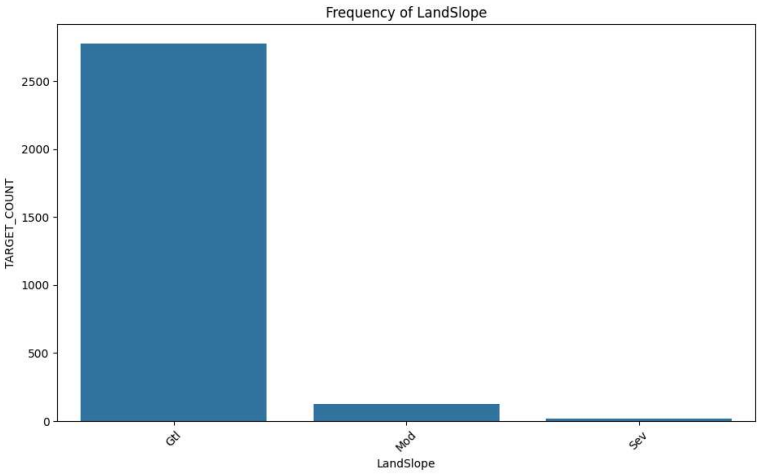
##

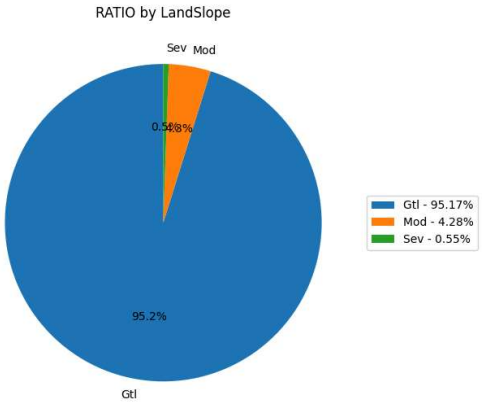




LandSlope		Ratio
LandSlope		
Gtl	2778	95.170
Mod	125	4.282
Sev	16	0.548

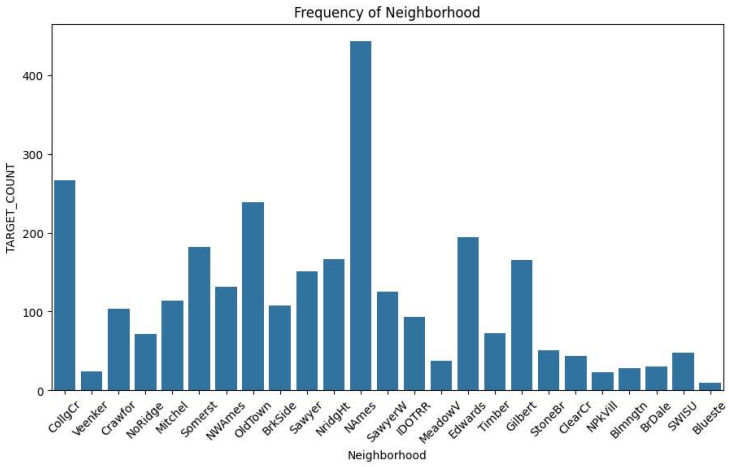
##

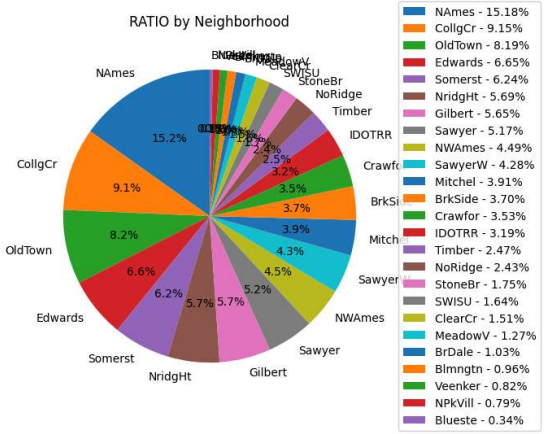




Neighborhood	Neighborhood	Ratio
NAmes	443	15.176
CollgCr	267	9.147
OldTown	239	8.188
Edwards	194	6.646
Somerst	182	6.235
NridgHt	166	5.687
Gilbert	165	5.653
Sawyer	151	5.173
NWAmes	131	4.488
SawyerW	125	4.282
Mitchel	114	3.905
BrkSide	108	3.700
Crawfor	103	3.529
IDOTRR	93	3.186
Timber	72	2.467
NoRidge	71	2.432
StoneBr	51	1.747
SWISU	48	1.644
ClearCr	44	1.507
MeadowV	37	1.268
BrDale	30	1.028
Blmngtn	28	0.959
Veenker	24	0.822
NPkVill	23	0.788
Blueste	10	0.343

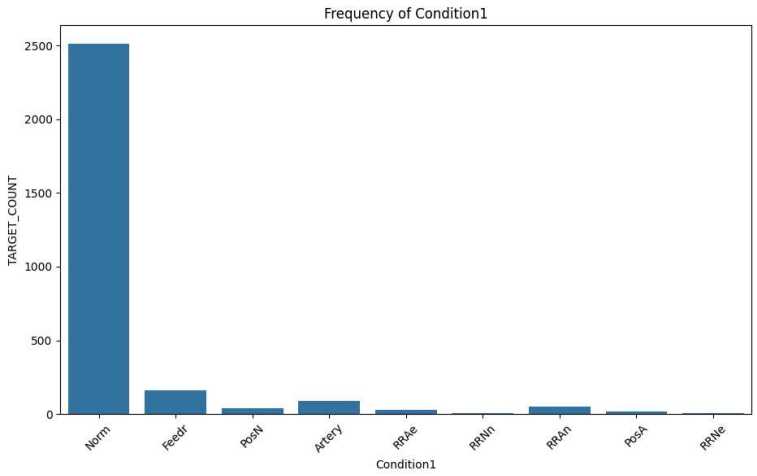
##

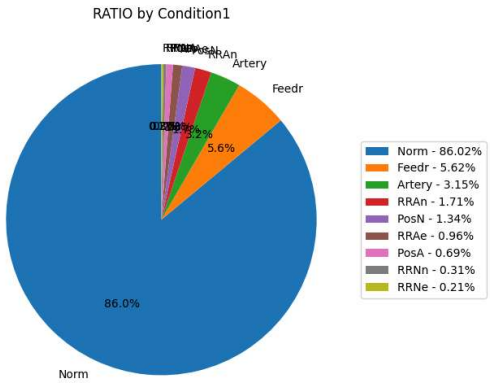




Condition1	Condition1	Ratio
Norm	2511	86.023
Feedr	164	5.618
Artery	92	3.152
RRAn	50	1.713
PosN	39	1.336
RRAe	28	0.959
PosA	20	0.685
RRNn	9	0.308
RRNe	6	0.206

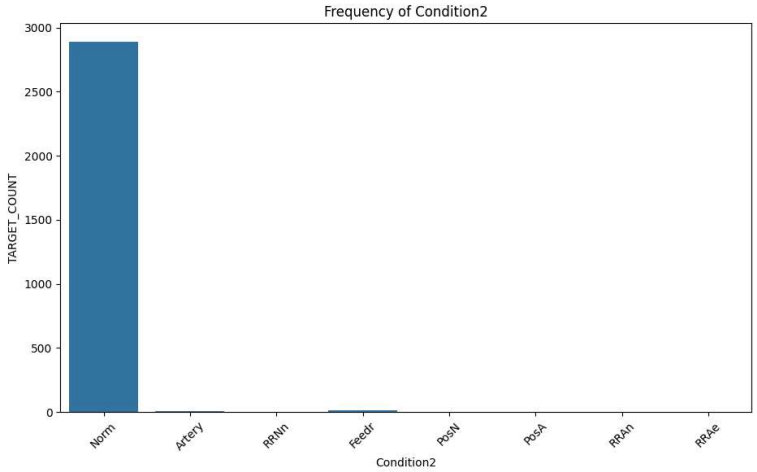
##

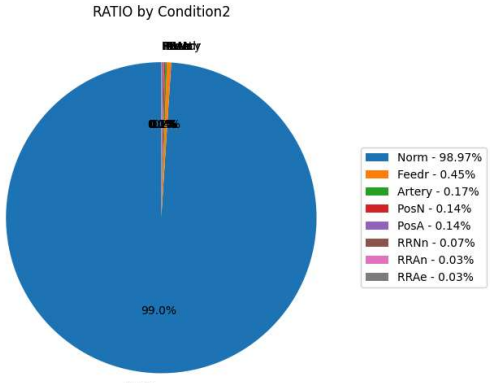




Condition2		Ratio
Condition2		
Norm	2889	98.972
Feedr	13	0.445
Artery	5	0.171
PosN	4	0.137
PosA	4	0.137
RRNn	2	0.069
RRAn	1	0.034
RRAe	1	0.034

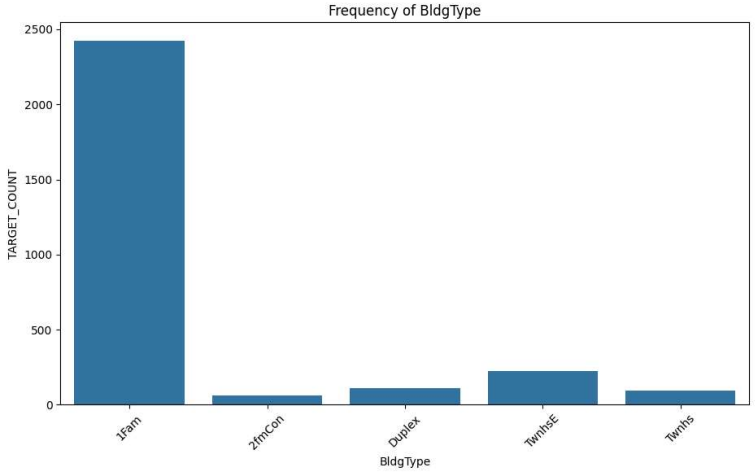
##

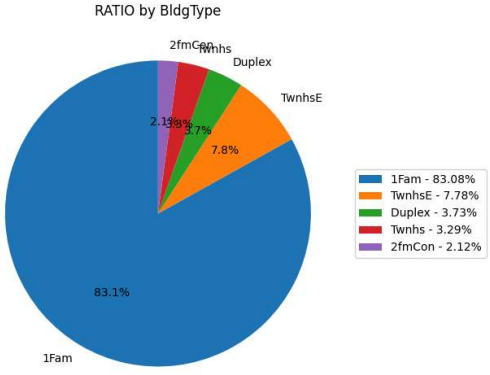




BldgType		Ratio
BldgType		
1Fam	2425	83.076
TwnhsE	227	7.777
Duplex	109	3.734
Twnhs	96	3.289
2fmCon	62	2.124

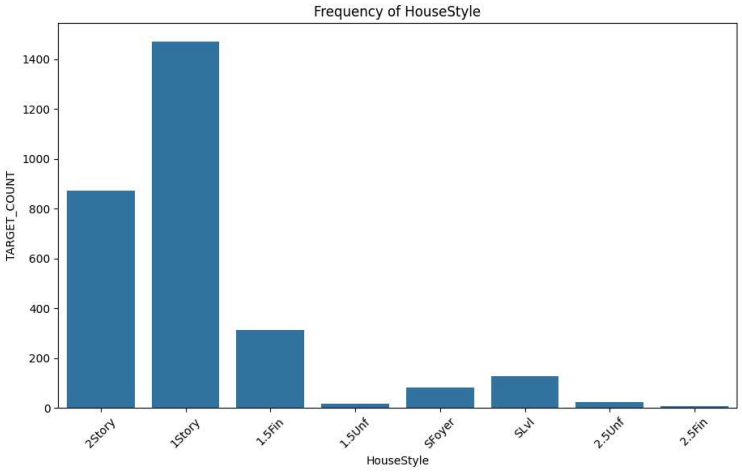
##

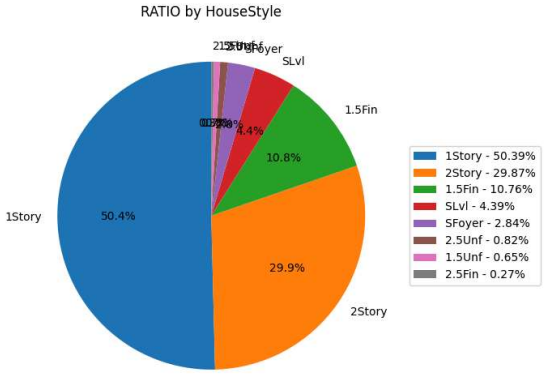




HouseStyle	Ratio
HouseStyle	
1Story	1471 50.394
2Story	872 29.873
1.5Fin	314 10.757
SLvl	128 4.385
SFoyer	83 2.843
2.5Unf	24 0.822
1.5Unf	19 0.651
2.5Fin	8 0.274

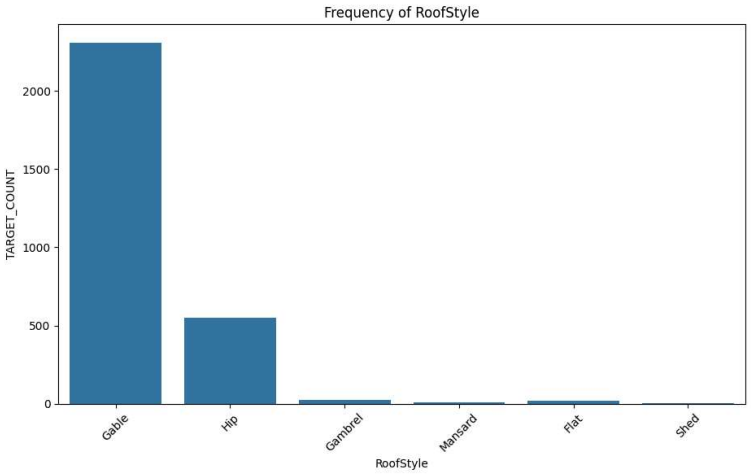
##

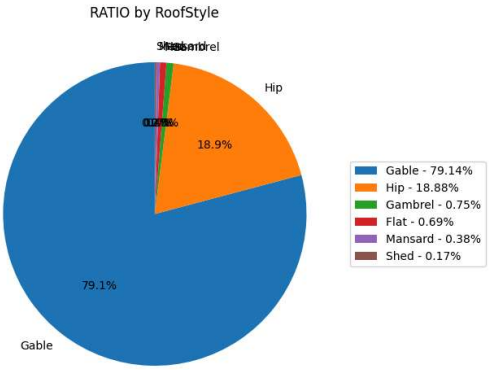




RoofStyle	Ratio
RoofStyle	
Gable	2310 79.137
Hip	551 18.876
Gambrel	22 0.754
Flat	20 0.685
Mansard	11 0.377
Shed	5 0.171

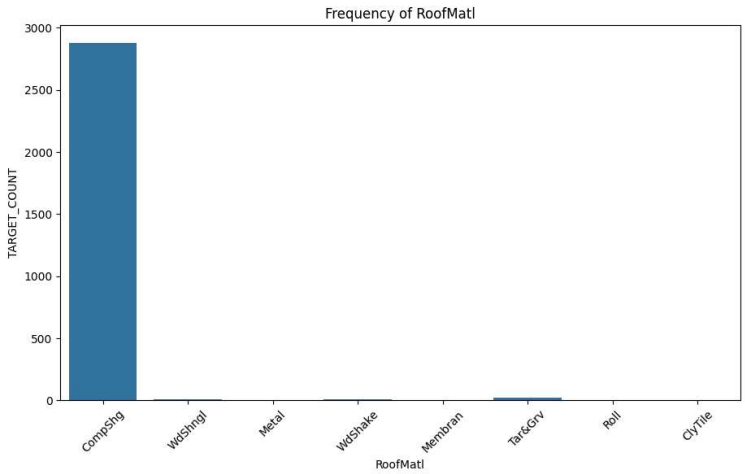
##

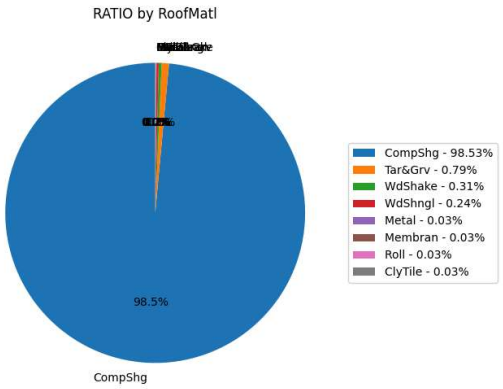




RoofMatl	Ratio
RoofMatl	
CompShg	2876 98.527
Tar&Grv	23 0.788
WdShake	9 0.308
WdShngl	7 0.240
Metal	1 0.034
Membran	1 0.034
Roll	1 0.034
ClyTile	1 0.034

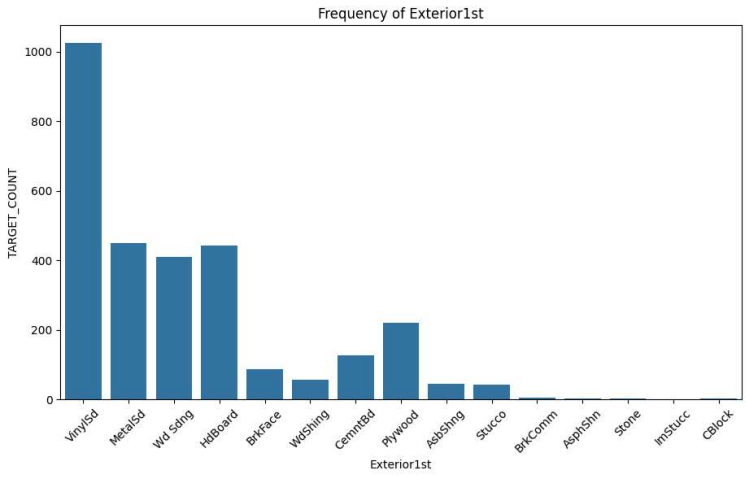
##

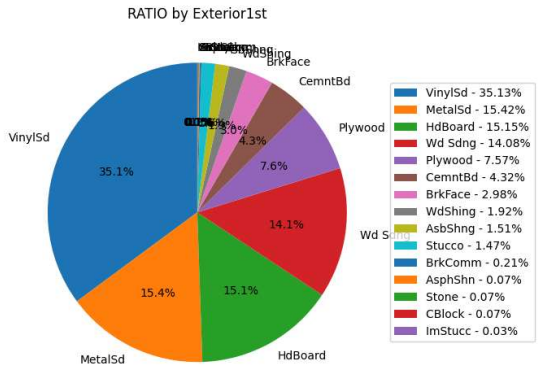




Exterior1st		Ratio
Exterior1st		
VinylSd	1025	35.115
MetalSd	450	15.416
HdBoard	442	15.142
Wd Sdng	411	14.080
Plywood	221	7.571
CemntBd	126	4.317
BrkFace	87	2.980
WdShing	56	1.918
AsbShng	44	1.507
Stucco	43	1.473
BrkComm	6	0.206
AsphShn	2	0.069
Stone	2	0.069
CBlock	2	0.069
ImStucc	1	0.034

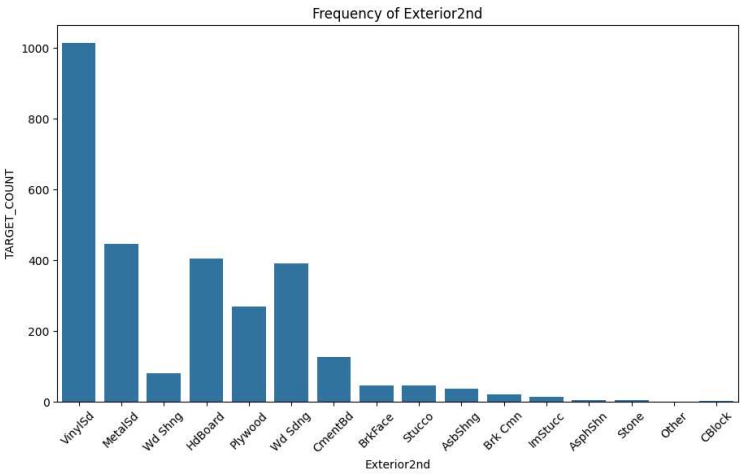
##

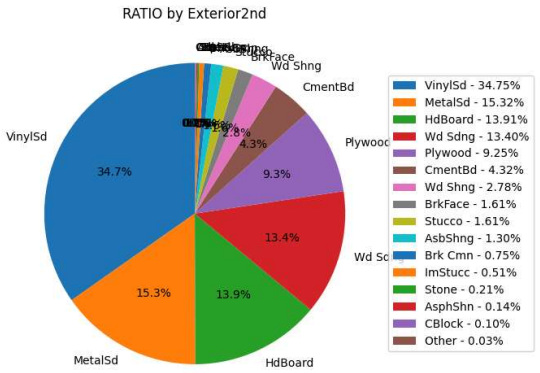




Exterior2nd		Ratio
Exterior2nd		
VinylSd	1014	34.738
MetalSd	447	15.313
HdBoard	406	13.909
Wd Sdng	391	13.395
Plywood	270	9.250
CmentBd	126	4.317
Wd Shng	81	2.775
BrkFace	47	1.610
Stucco	47	1.610
AsbShng	38	1.302
Brk Cmn	22	0.754
ImStucc	15	0.514
Stone	6	0.206
AsphShn	4	0.137
CBlock	3	0.103
Other	1	0.034

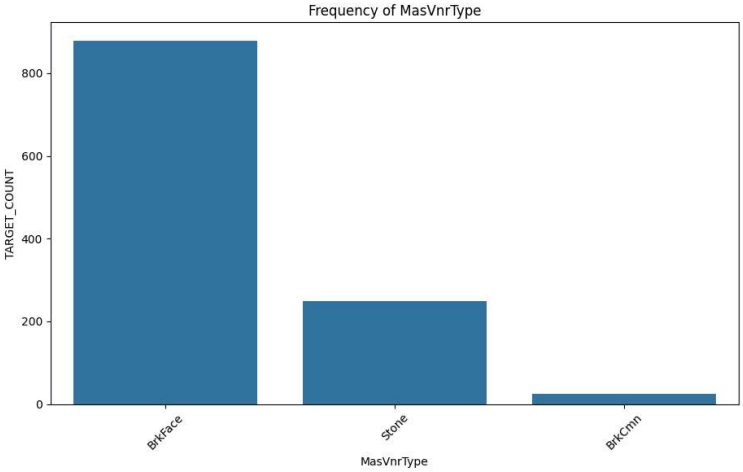
##

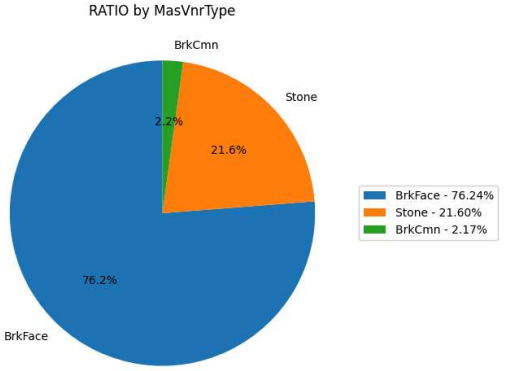




MasVnrType		Ratio
MasVnrType		
BrkFace	879	30.113
Stone	249	8.530
BrkCmn	25	0.856

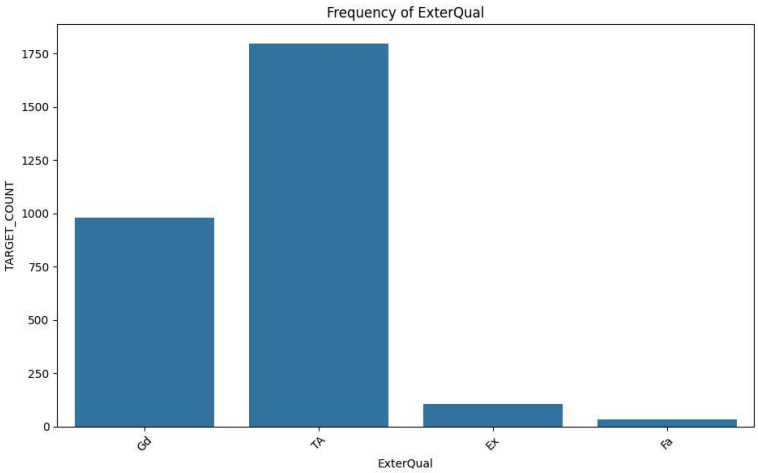
##

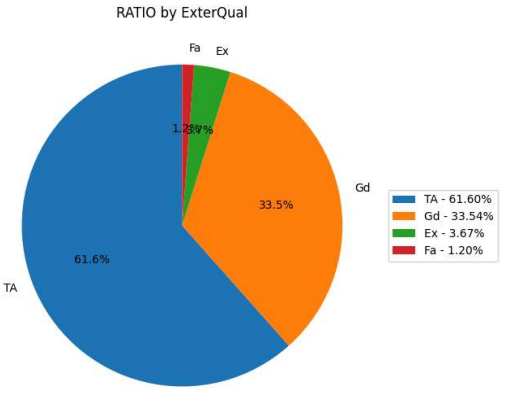




ExterQual		Ratio
ExterQual		
TA	1798	61.596
Gd	979	33.539
Ex	107	3.666
Fa	35	1.199

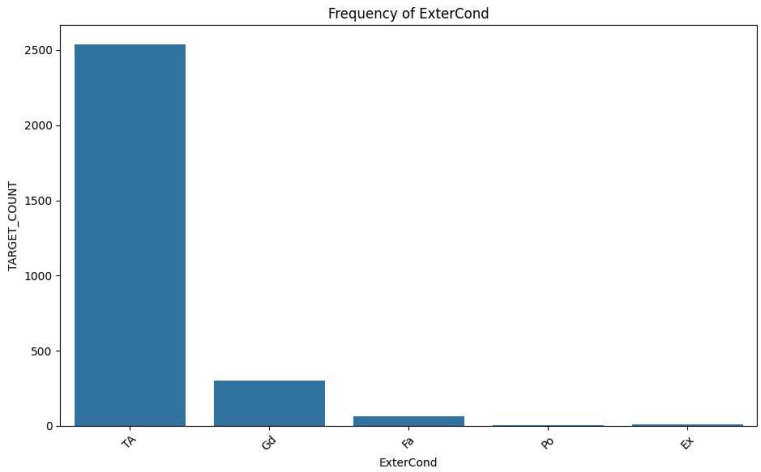
##

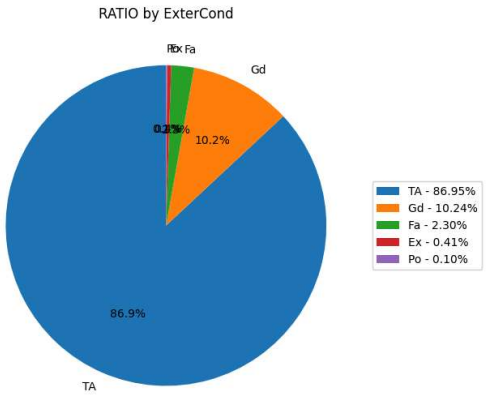




ExterCond		Ratio
ExterCond		
TA	2538	86.948
Gd	299	10.243
Fa	67	2.295
Ex	12	0.411
Po	3	0.103

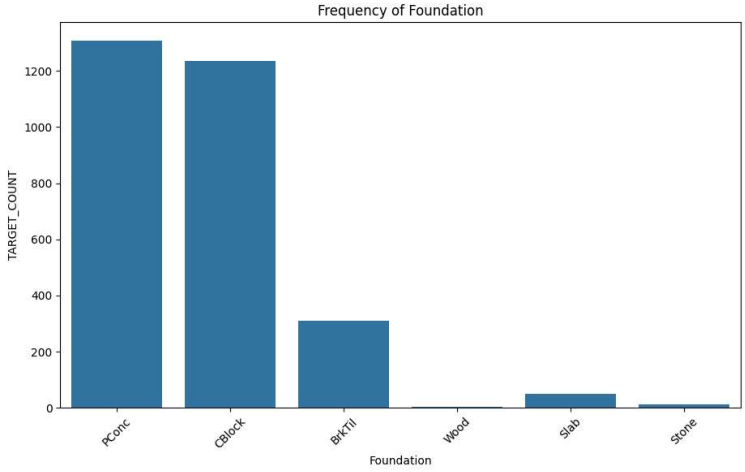
##

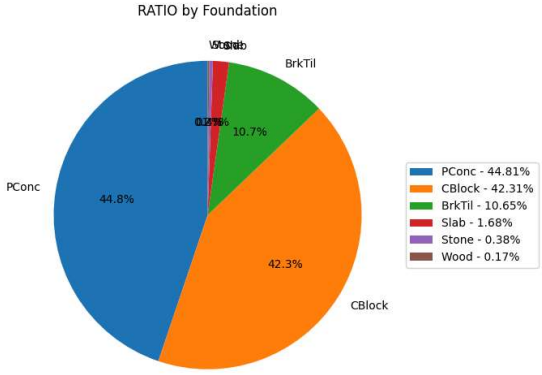




Foundation		Ratio
Foundation		
PConc	1308	44.810
CBlock	1235	42.309
BrkTil	311	10.654
Slab	49	1.679
Stone	11	0.377
Wood	5	0.171

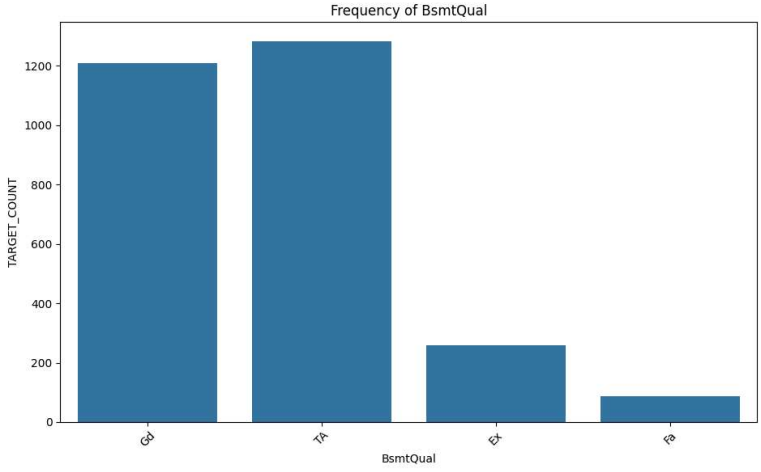
##

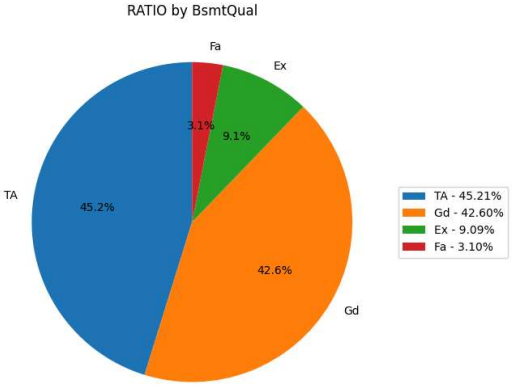




BsmtQual		Ratio
BsmtQual		
TA	1283	43.953
Gd	1209	41.418
Ex	258	8.839
Fa	88	3.015

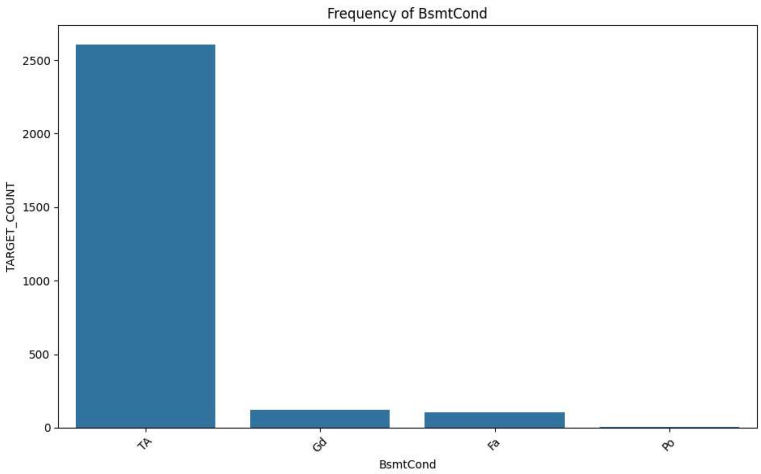
##

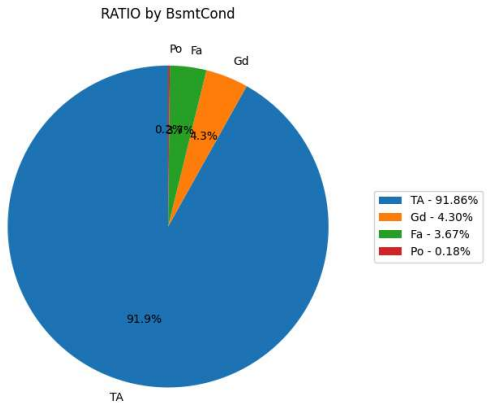




BsmtCond		Ratio
BsmtCond		
TA	2606	89.277
Gd	122	4.180
Fa	104	3.563
Po	5	0.171

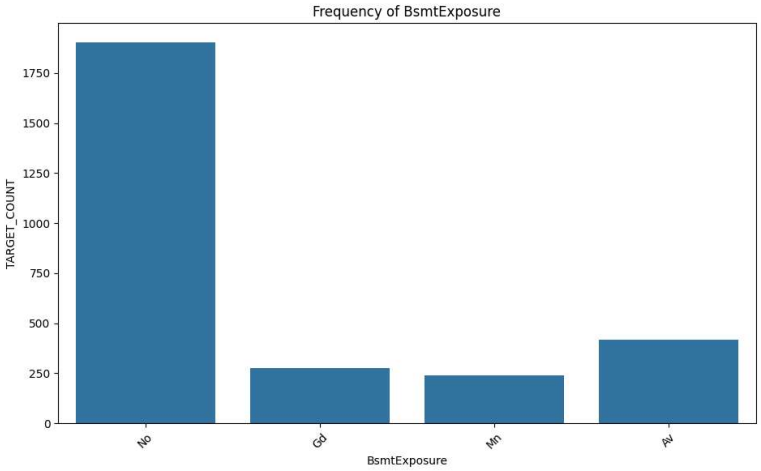
##

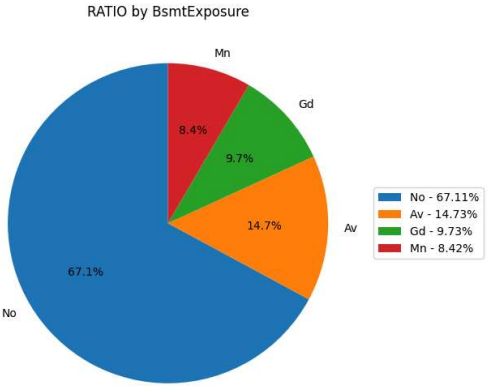




BsmtExposure		Ratio
BsmtExposure		
No	1904	65.228
Av	418	14.320
Gd	276	9.455
Mn	239	8.188

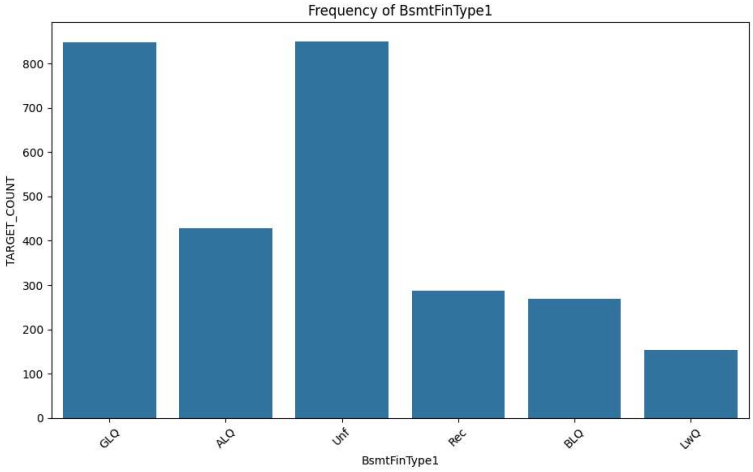
##

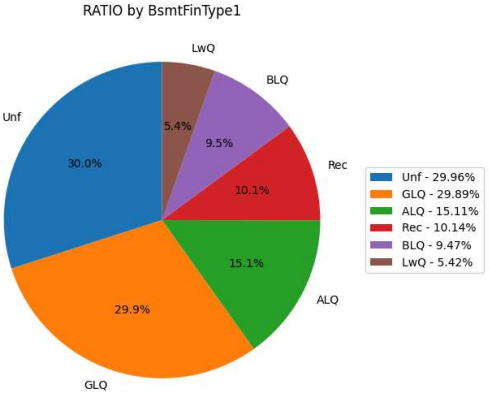




BsmtFinType1		Ratio
BsmtFinType1		
Unf	851	29.154
GLQ	849	29.085
ALQ	429	14.697
Rec	288	9.866
BLQ	269	9.215
LwQ	154	5.276

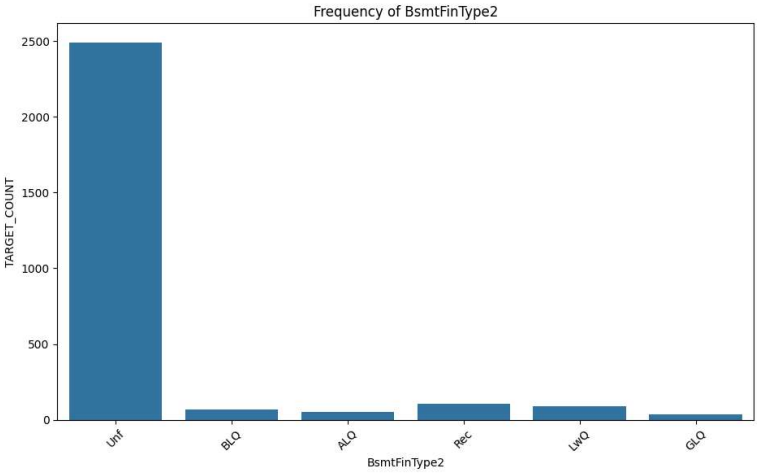
##

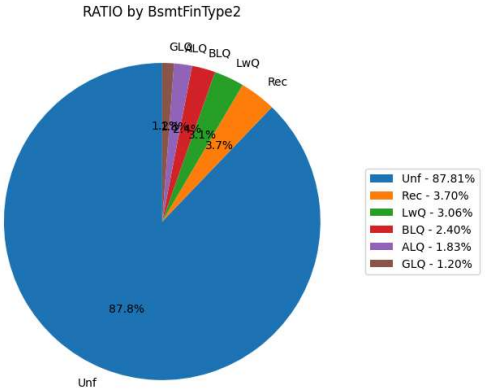




BsmtFinType2		Ratio
BsmtFinType2		
Unf	2493	85.406
Rec	105	3.597
LwQ	87	2.980
BLQ	68	2.330
ALQ	52	1.781
GLQ	34	1.165

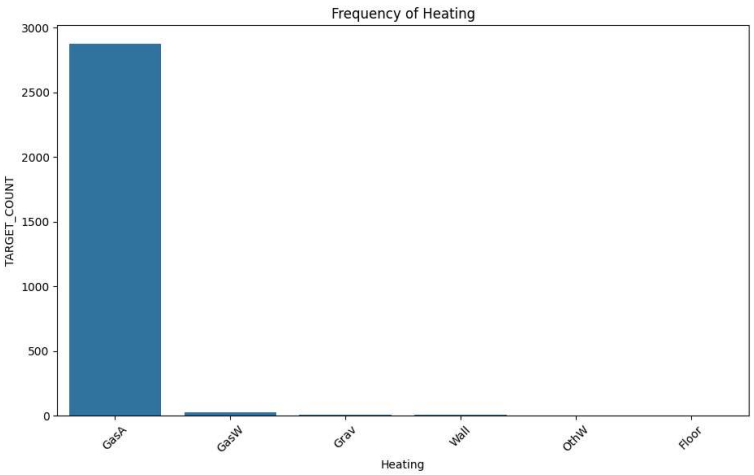
##

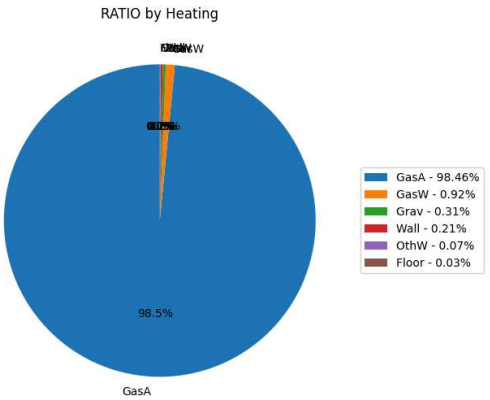




Heating		Ratio
Heating		
GasA	2874	98.458
GasW	27	0.925
Grav	9	0.308
Wall	6	0.206
OthW	2	0.069
Floor	1	0.034

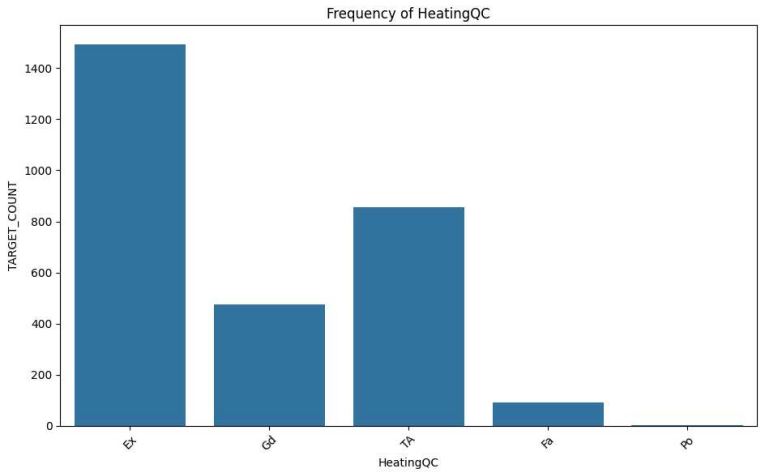
##

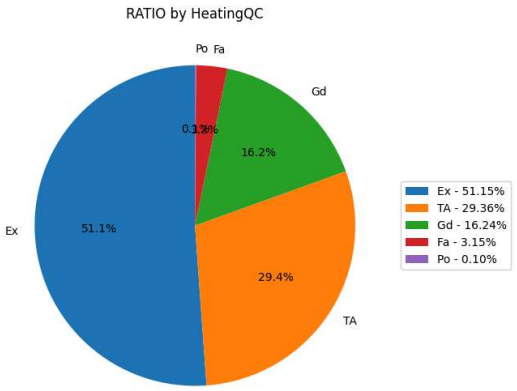




HeatingQC		Ratio
HeatingQC		
Ex	1493	51.148
TA	857	29.359
Gd	474	16.238
Fa	92	3.152
Po	3	0.103

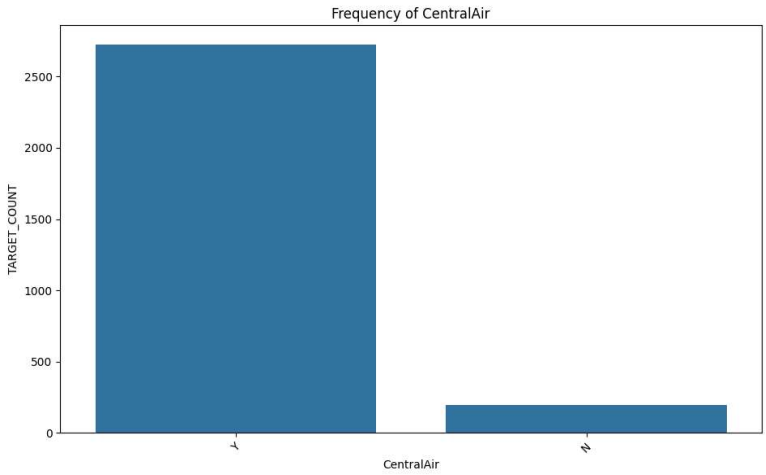
##

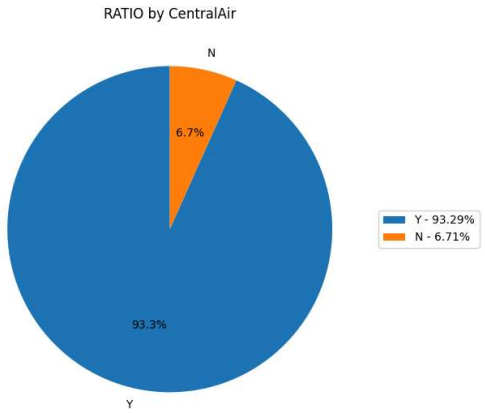




CentralAir		Ratio
CentralAir		
Y	2723	93.285
N	196	6.715

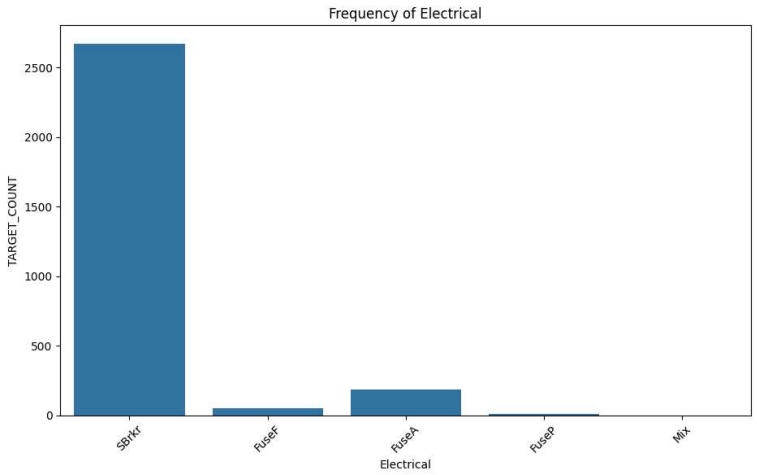
##

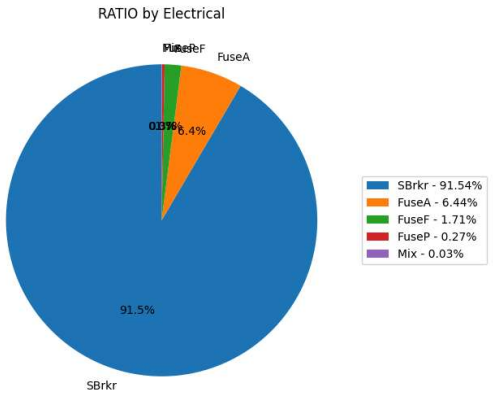




Electrical		Ratio
Electrical		
SBrkr	2671	91.504
FuseA	188	6.441
FuseF	50	1.713
FuseP	8	0.274
Mix	1	0.034

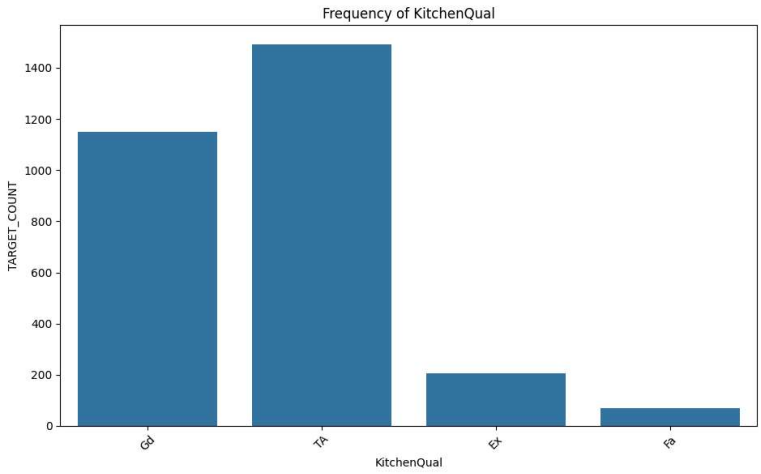
##

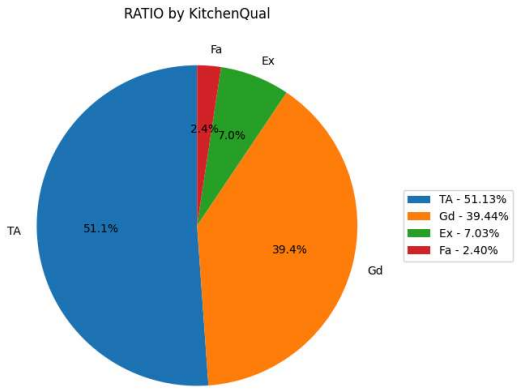




KitchenQual		Ratio
KitchenQual		
TA	1492	51.113
Gd	1151	39.431
Ex	205	7.023
Fa	70	2.398

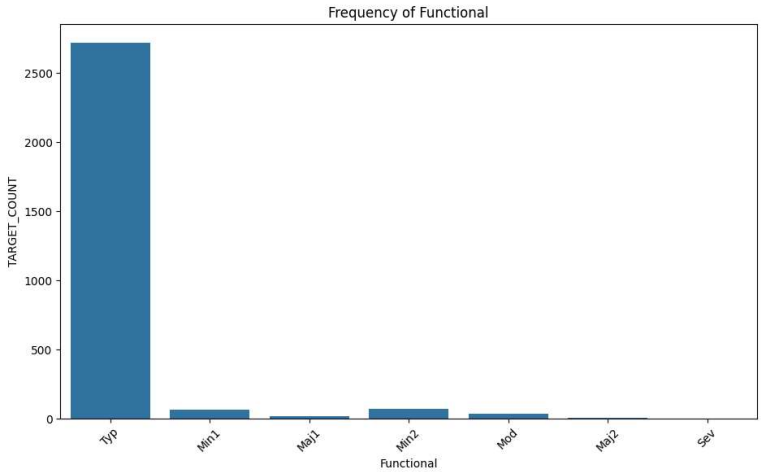
##

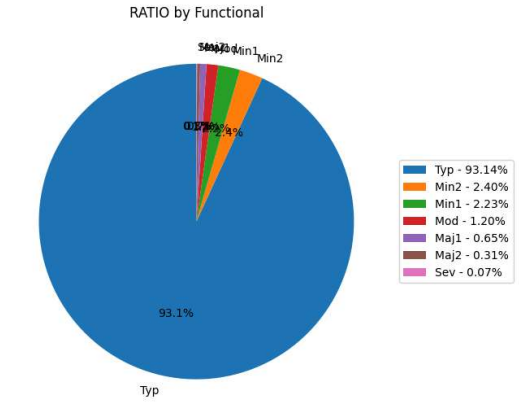




Functional		Ratio
Functional		
Typ	2717	93.080
Min2	70	2.398
Min1	65	2.227
Mod	35	1.199
Maj1	19	0.651
Maj2	9	0.308
Sev	2	0.069

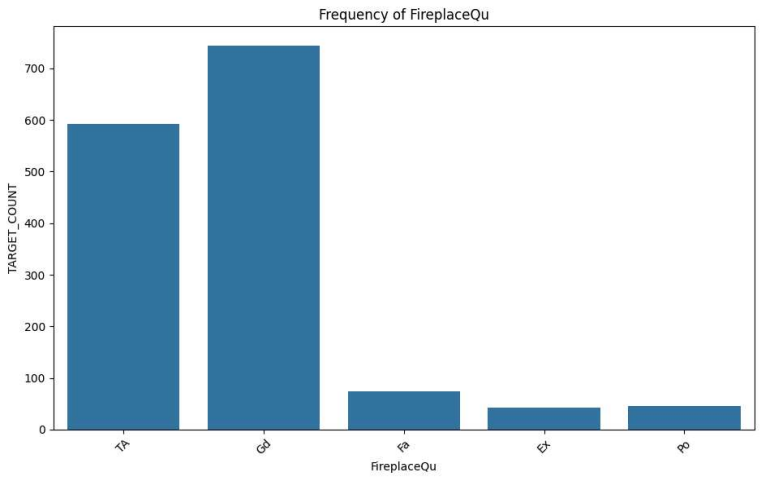
##

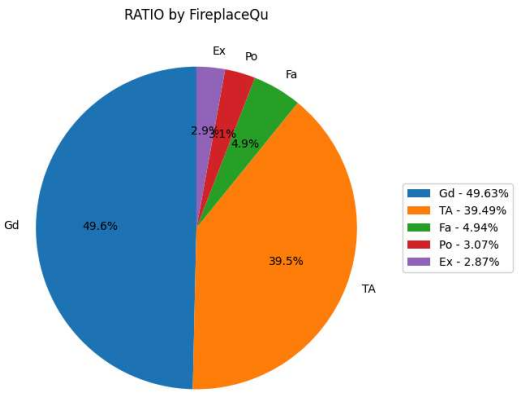




FireplaceQu		Ratio
FireplaceQu		
Gd	744	25.488
TA	592	20.281
Fa	74	2.535
Po	46	1.576
Ex	43	1.473

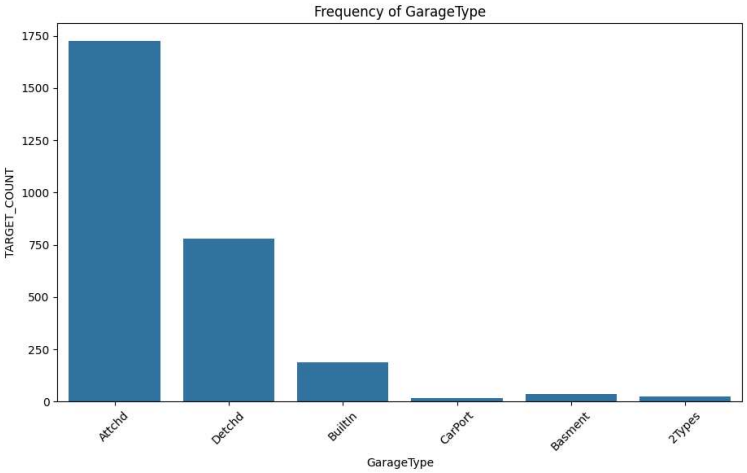
##

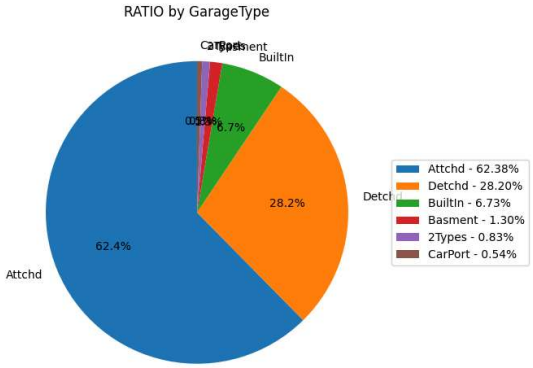




GarageType		Ratio
GarageType		
Attchd	1723	59.027
Detchd	779	26.687
BuiltIn	186	6.372
Basment	36	1.233
2Types	23	0.788
CarPort	15	0.514

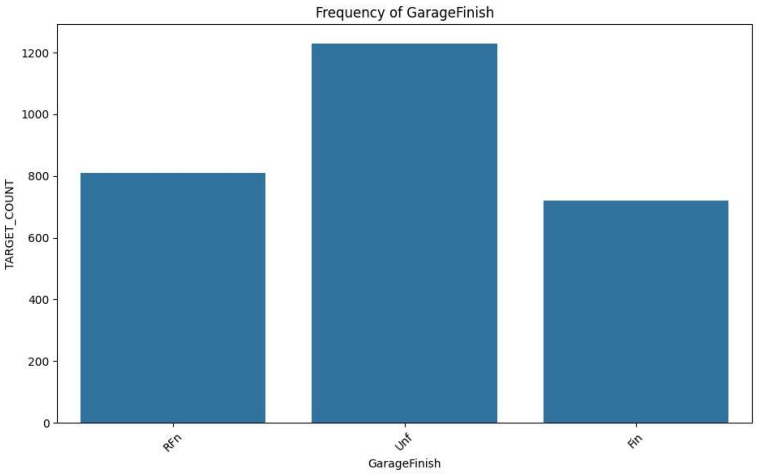
##

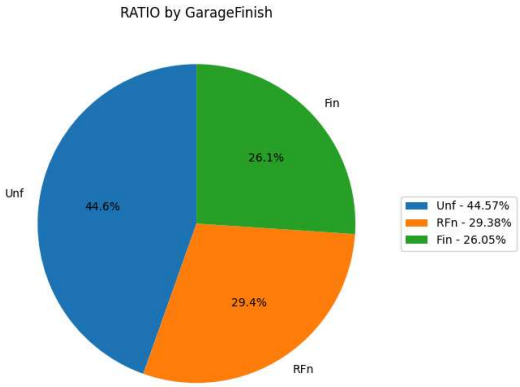




GarageFinish		Ratio
GarageFinish		
Unf	1230	42.138
RFn	811	27.783
Fin	719	24.632

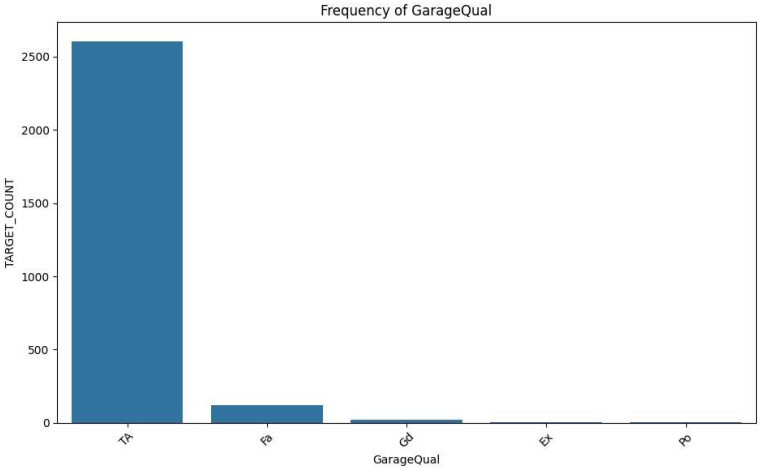
##

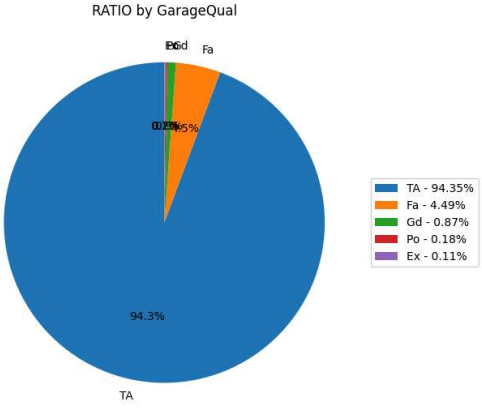




GarageQual		Ratio
GarageQual		
TA	2604	89.209
Fa	124	4.248
Gd	24	0.822
Po	5	0.171
Ex	3	0.103

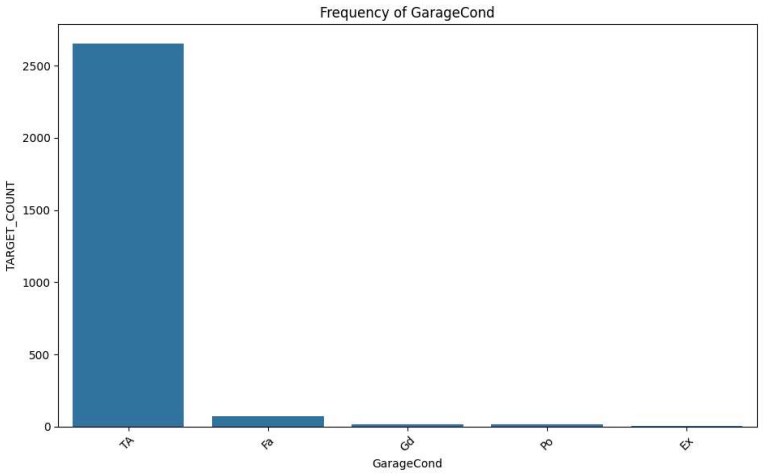
##

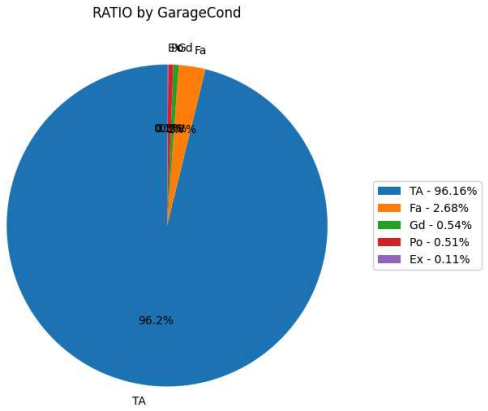




GarageCond	Ratio
GarageCond	
TA	2654 90.922
Fa	74 2.535
Gd	15 0.514
Po	14 0.480
Ex	3 0.103

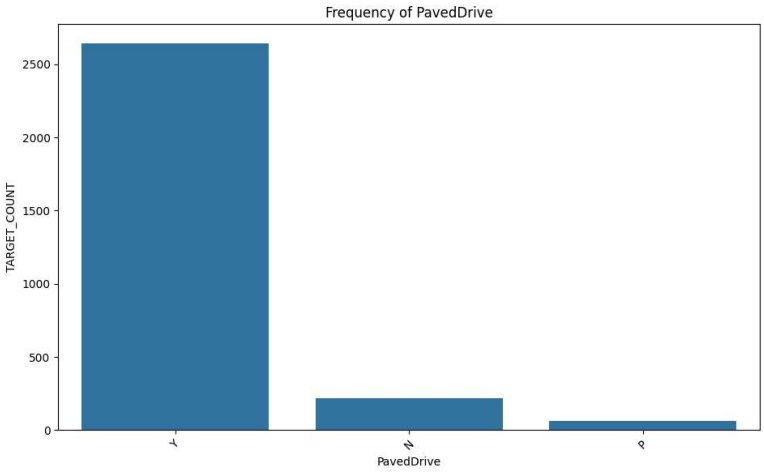
##

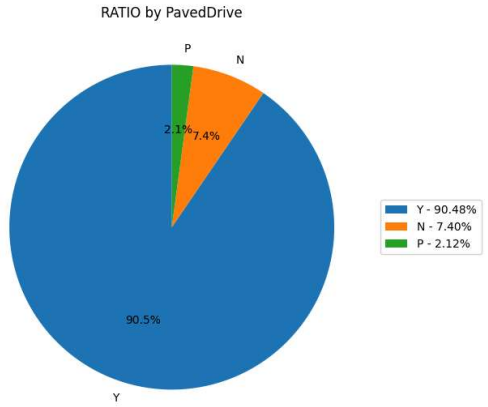




PavedDrive	Ratio
PavedDrive	
Y	2641 90.476
N	216 7.400
P	62 2.124

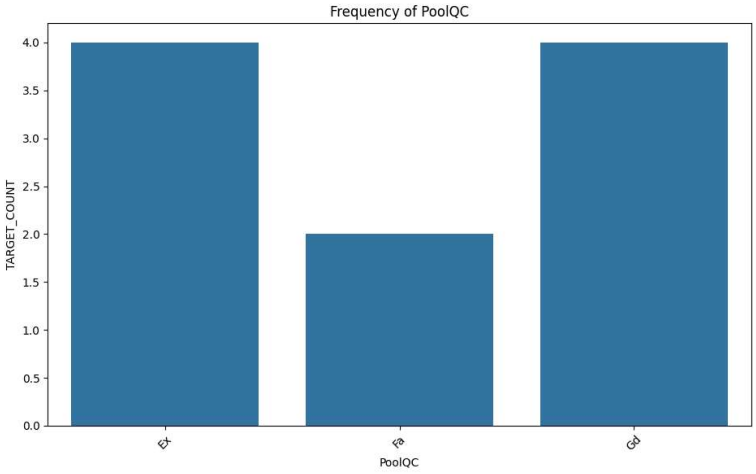
##

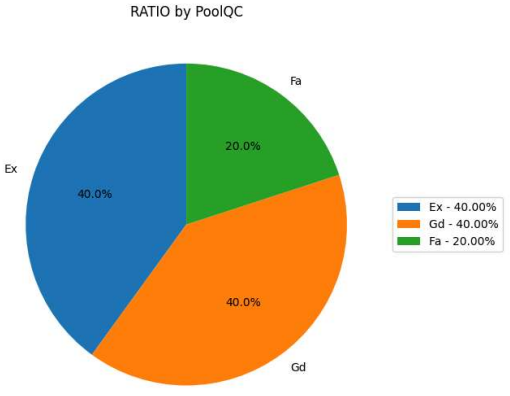




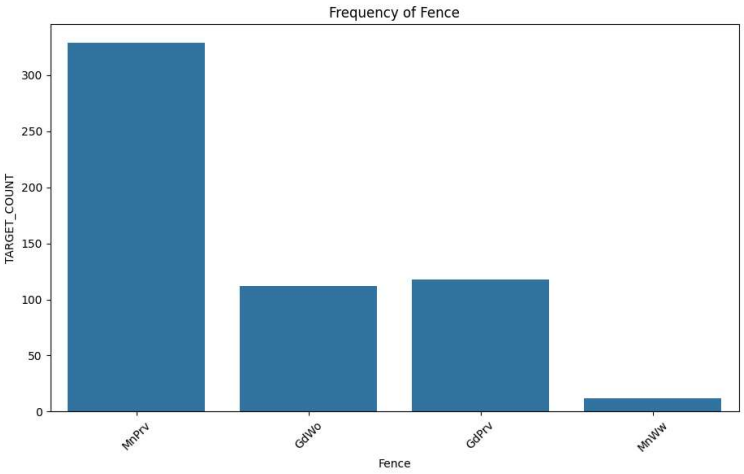
PoolQC	Ratio
PoolQC	
Ex	4 0.137
Gd	4 0.137
Fa	2 0.069

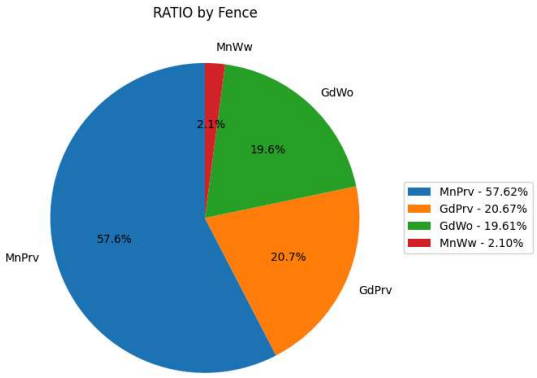
##



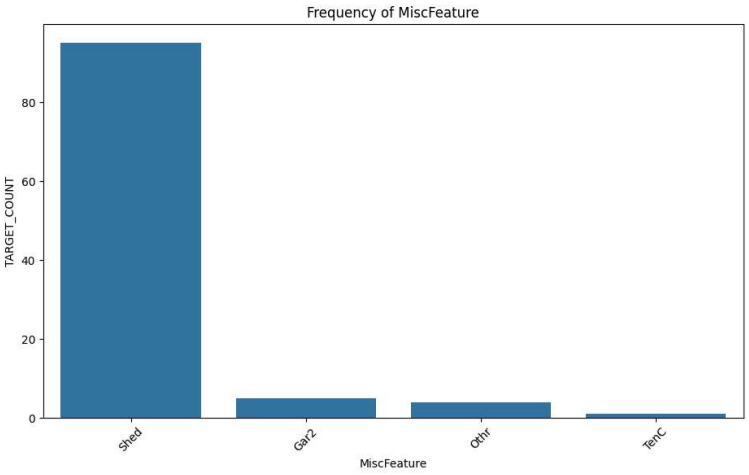


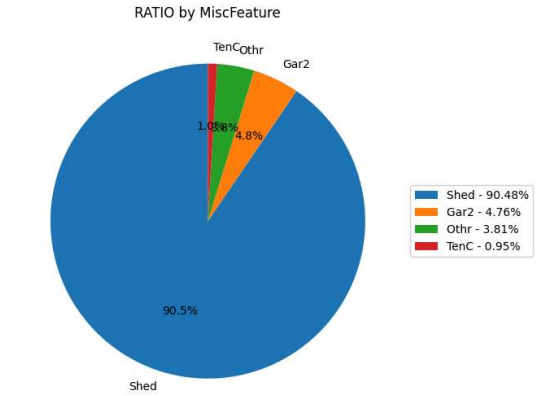
```
       Fence  Ratio
Fence
MnPrv    329 11.271
GdPrv    118  4.042
GdWo     112  3.837
MnWw      12  0.411
########################################
```



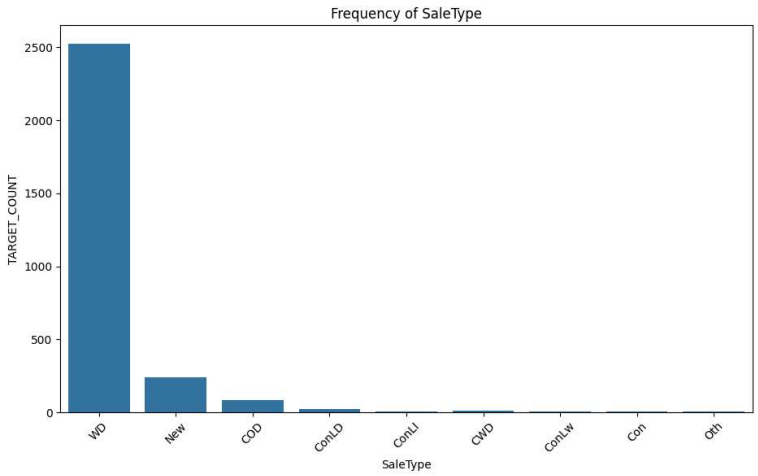


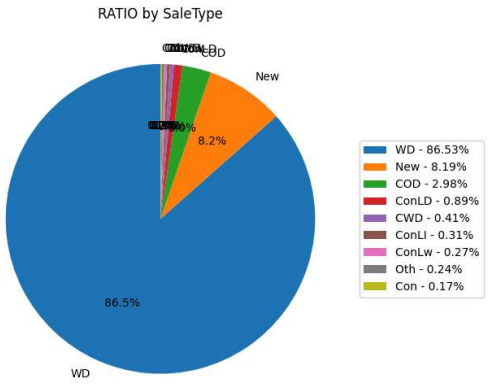

```
             MiscFeature  Ratio
MiscFeature
Shed                  95  3.255
Gar2                   5  0.171
Othr                   4  0.137
TenC                   1  0.034
########################################
```





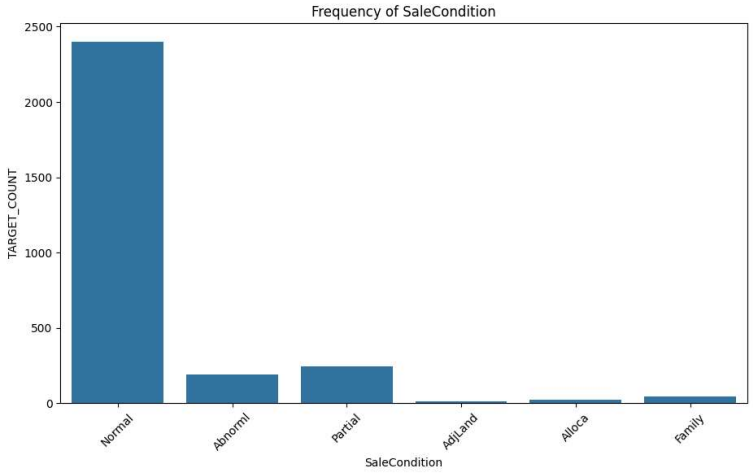

```
           SaleType  Ratio
SaleType
WD             2525 86.502
New             239  8.188
COD              87  2.980
ConLD            26  0.891
CWD              12  0.411
ConLI             9  0.308
ConLw             8  0.274
Oth               7  0.240
Con               5  0.171
########################################
```

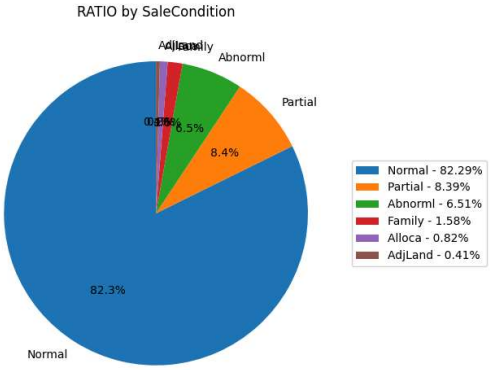




SaleCondition		Ratio
SaleCondition		
Normal	2402	82.288
Partial	245	8.393
Abnorml	190	6.509
Family	46	1.576
Alloca	24	0.822
AdjLand	12	0.411

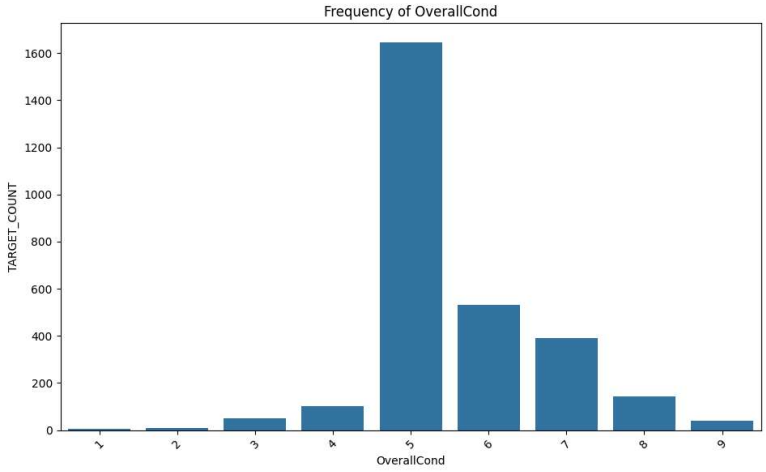
##

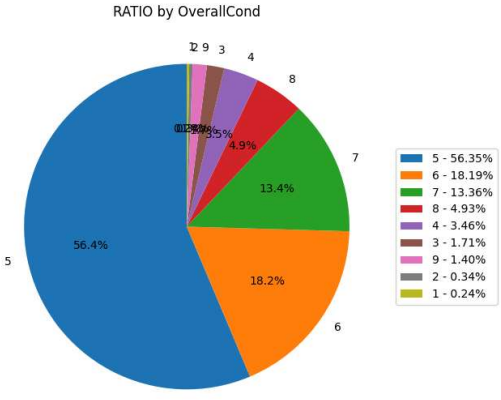




OverallCond		Ratio
OverallCond		
5	1645	56.355
6	531	18.191
7	390	13.361
8	144	4.933
4	101	3.460
3	50	1.713
9	41	1.405
2	10	0.343
1	7	0.240

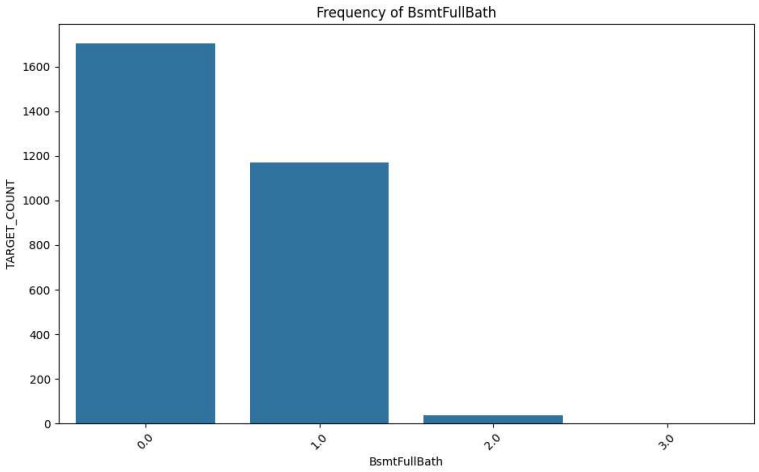
##

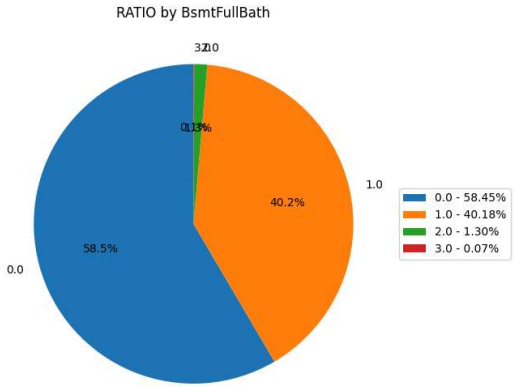




BsmtFullBath		Ratio
0.000	1705	58.410
1.000	1172	40.151
2.000	38	1.302
3.000	2	0.069

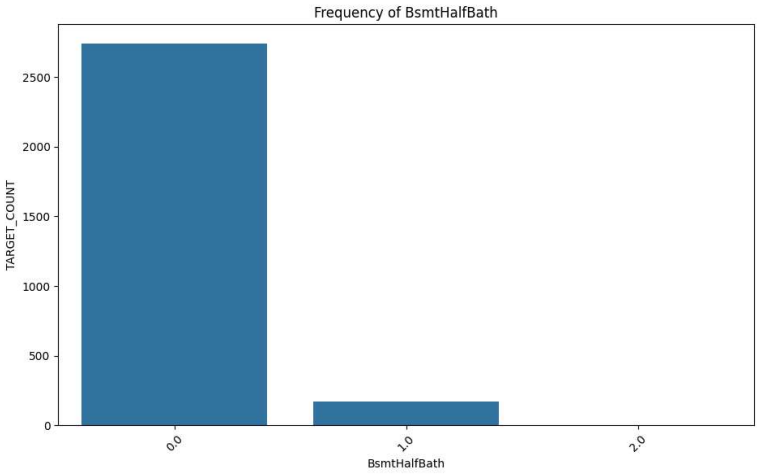
##

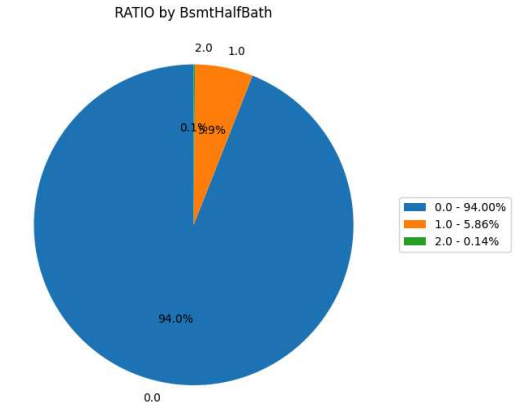




BsmtHalfBath		Ratio
0.000	2742	93.936
1.000	171	5.858
2.000	4	0.137

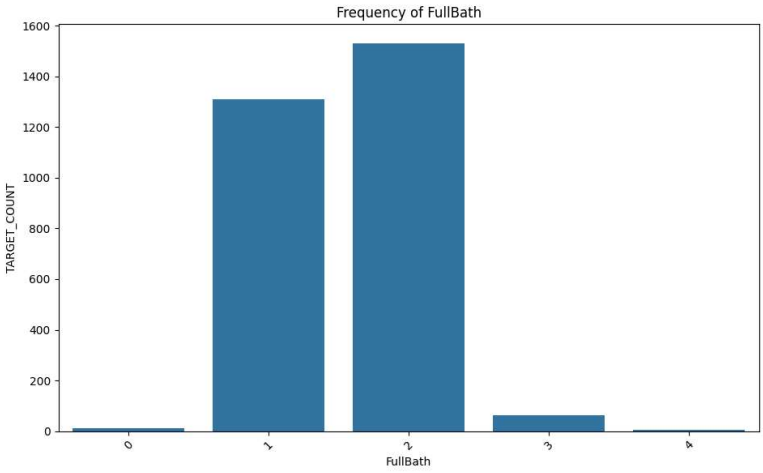
##

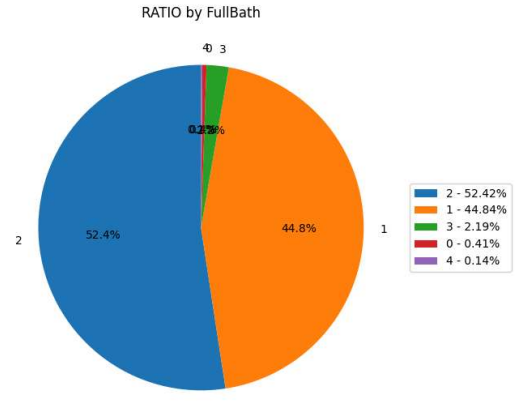




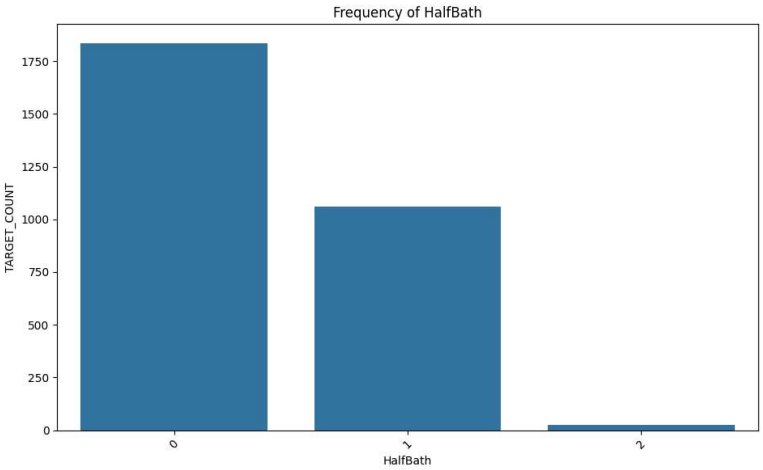
FullBath		Ratio
2	1530	52.415
1	1309	44.844
3	64	2.193
0	12	0.411
4	4	0.137

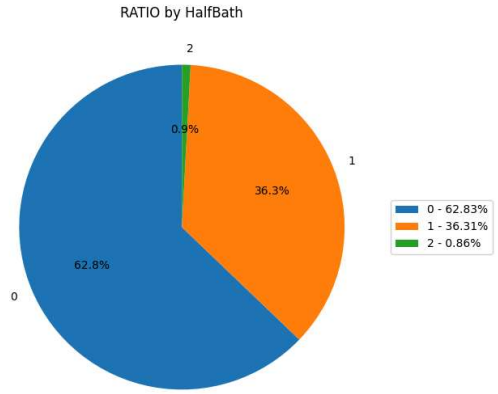
##



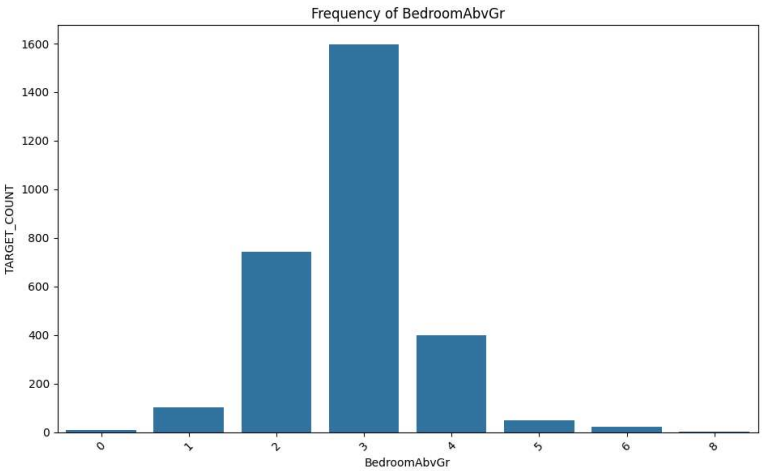


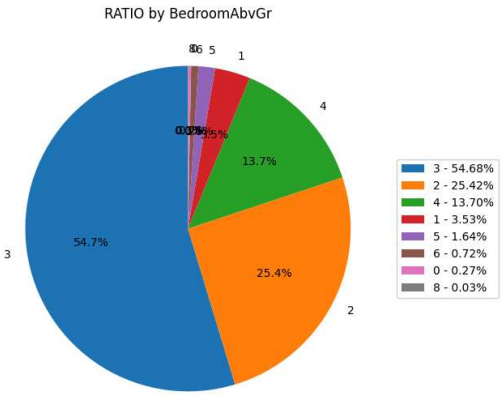
```
          HalfBath  Ratio
HalfBath
0             1834 62.830
1             1060 36.314
2               25  0.856
##########################################
```



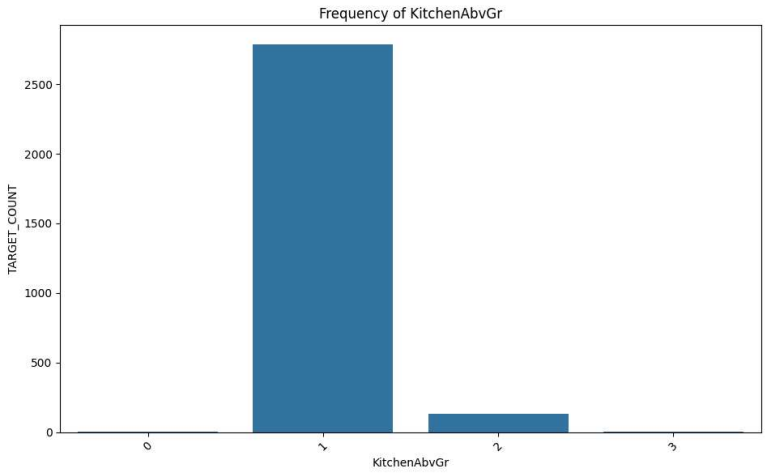


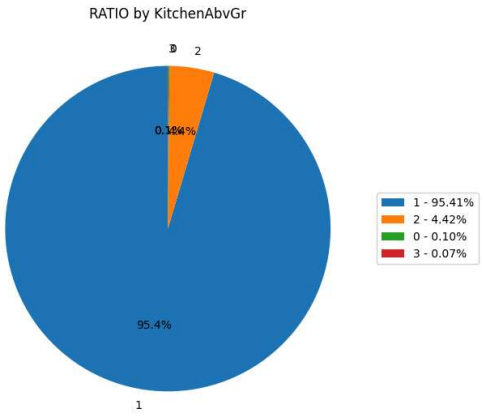

```
              BedroomAbvGr  Ratio
BedroomAbvGr
3                     1596 54.676
2                      742 25.420
4                      400 13.703
1                      103  3.529
5                       48  1.644
6                       21  0.719
0                        8  0.274
8                        1  0.034
##########################################
```





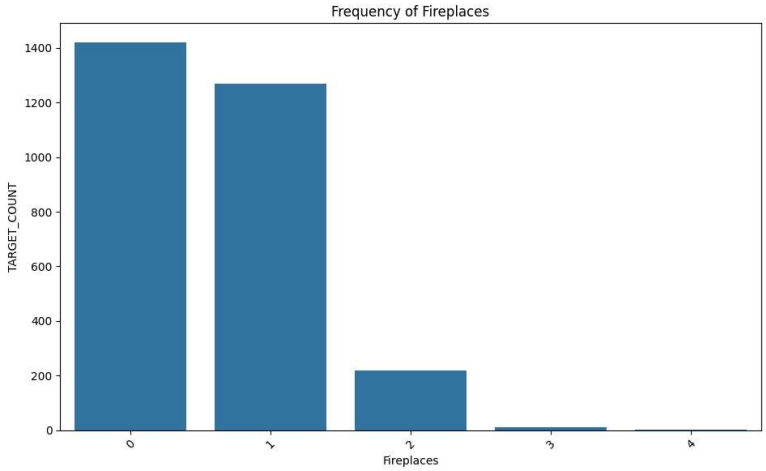

```
              KitchenAbvGr  Ratio
KitchenAbvGr
1                     2785 95.409
2                      129  4.419
0                        3  0.103
3                        2  0.069
##########################################
```

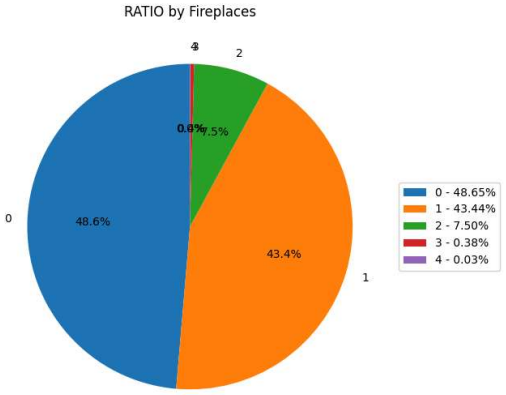




Fireplaces	Ratio
Fireplaces	
0	1420 48.647
1	1268 43.440
2	219 7.503
3	11 0.377
4	1 0.034

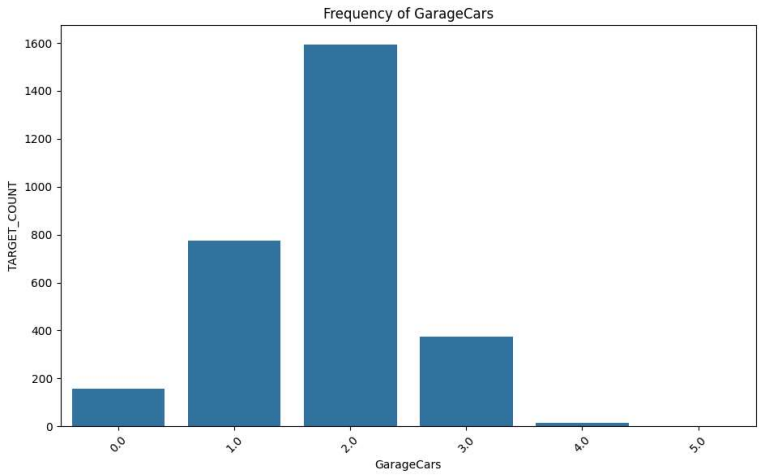
##

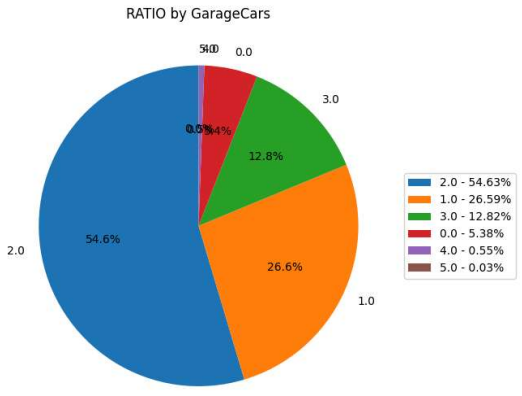




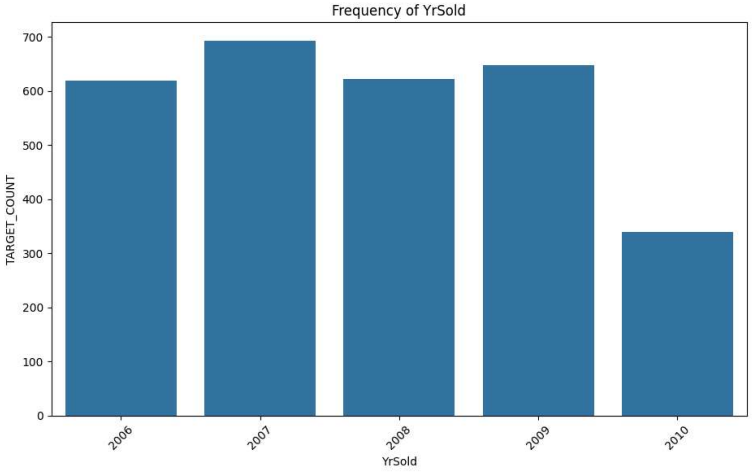
GarageCars	Ratio
GarageCars	
2.000	1594 54.608
1.000	776 26.584
3.000	374 12.813
0.000	157 5.379
4.000	16 0.548
5.000	1 0.034

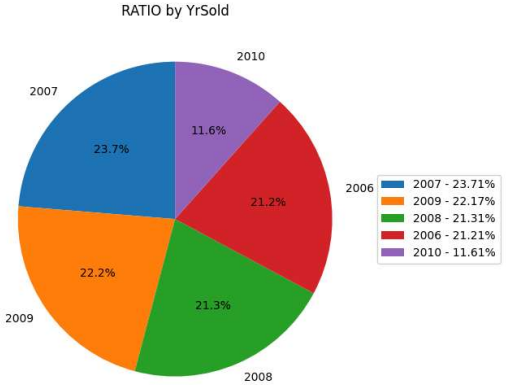
##





```
        YrSold  Ratio
YrSold
2007       692 23.707
2009       647 22.165
2008       622 21.309
2006       619 21.206
2010       339 11.614
##########################################
```



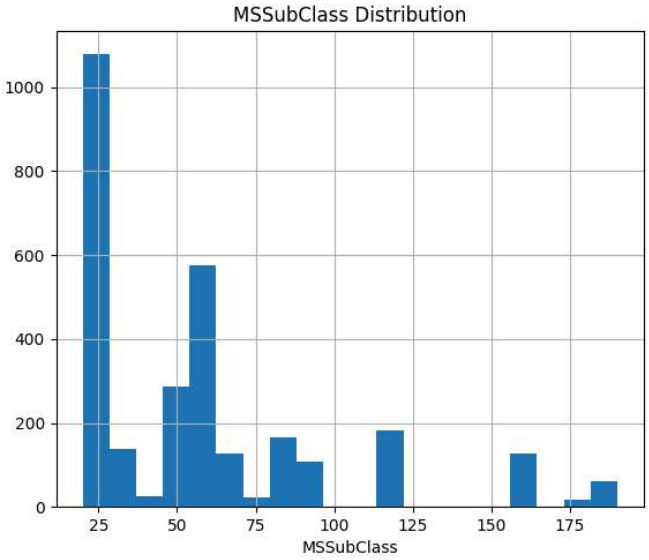


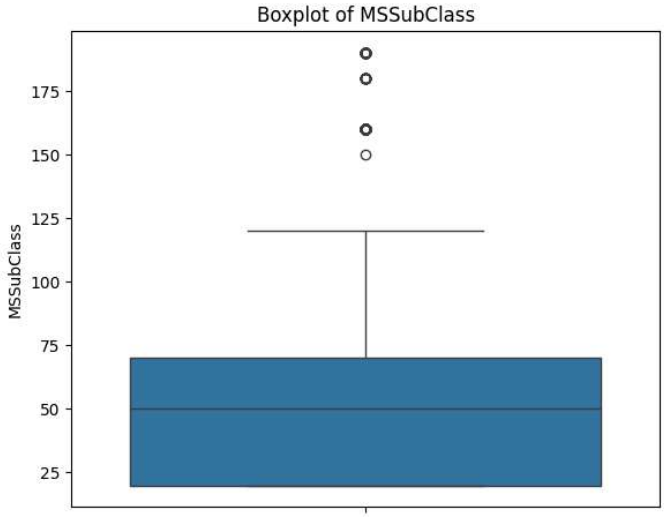
Num Cols Analysis

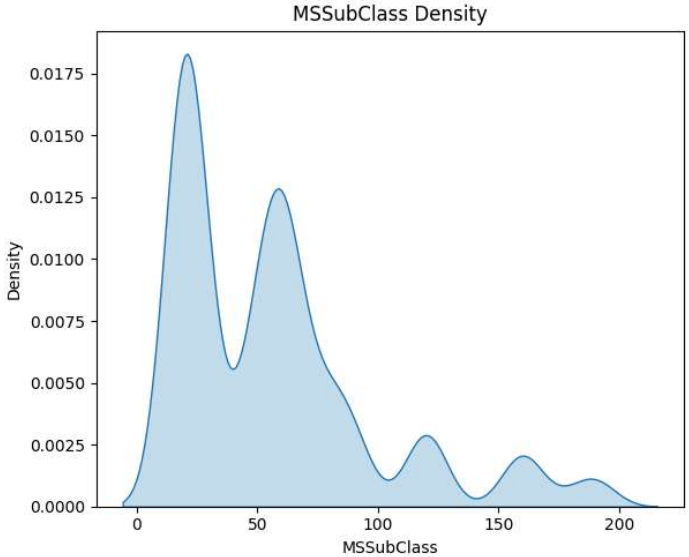
In [ ]:
```python
def num_summary(dataframe, numerical_col, plot=False, hist_bins=20):
    quantiles = [0.05, 0.10, 0.20, 0.30, 0.40, 0.50, 0.60, 0.70, 0.80, 0.90, 0.95, 0.99]
    print(numerical_col)
    print("##########################################")
    print(dataframe[numerical_col].describe(quantiles).T)
    print("##########################################")

    if plot:
        fig, axs = plt.subplots(2, 2, figsize=(12, 10))

        # Histogram
        plt.subplot(2, 2, 1)
        dataframe[numerical_col].hist(bins=hist_bins)
        plt.xlabel(numerical_col)
        plt.title(numerical_col + " Distribution")

        # Boxplot
        plt.subplot(2, 2, 2)
        sns.boxplot(y=numerical_col, data=dataframe)
        plt.title("Boxplot of " + numerical_col)
        plt.xticks(rotation=90)

        # Density Plot
        plt.subplot(2, 2, 3)
        sns.kdeplot(dataframe[numerical_col], fill=True)
        plt.xlabel(numerical_col)
        plt.title(numerical_col + " Density")

        # QQ Plot
        plt.subplot(2, 2, 4)
        stats.probplot(dataframe[numerical_col], dist="norm", plot=plt)
        plt.title(numerical_col + " QQ Plot")

        plt.tight_layout()
        plt.show(block=True)
```

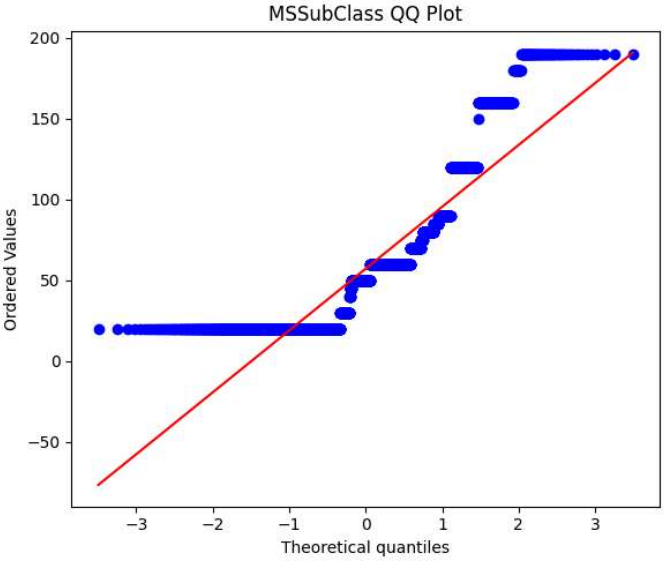
In [ ]:
```python
for col in num_cols:
    num_summary(df, col, plot=True)
```

```
MSSubClass
##########################################
count   2919.000
mean      57.138
std       42.518
min       20.000
5%        20.000
10%       20.000
20%       20.000
30%       20.000
40%       30.000
50%       50.000
60%       60.000
70%       60.000
80%       80.000
90%      120.000
95%      160.000
99%      190.000
max      190.000
Name: MSSubClass, dtype: float64
##########################################
```

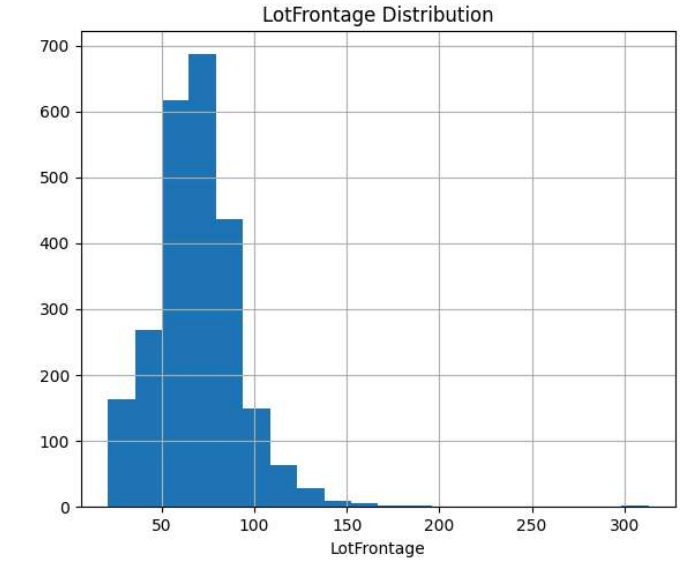


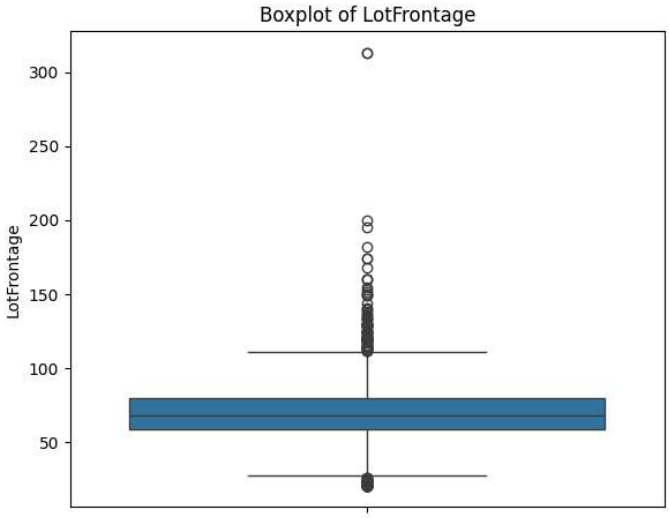


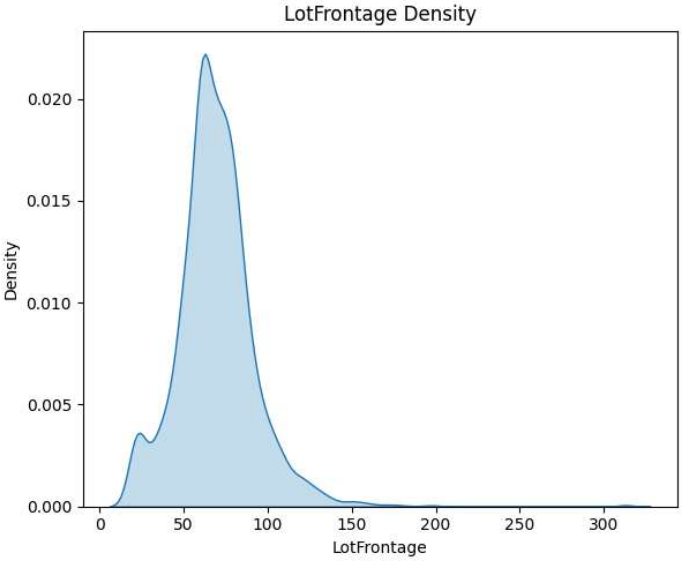


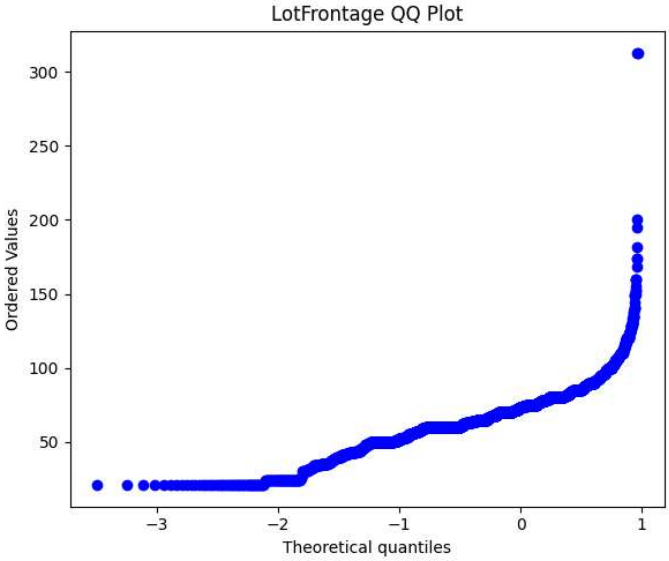


```
LotFrontage
##########################################
count   2433.000
mean      69.306
std       23.345
min       21.000
5%        32.000
10%       43.000
20%       53.000
30%       60.000
40%       63.000
50%       68.000
60%       73.000
70%       78.000
80%       84.000
90%       95.000
95%      107.000
99%      135.680
max      313.000
Name: LotFrontage, dtype: float64
##########################################
```

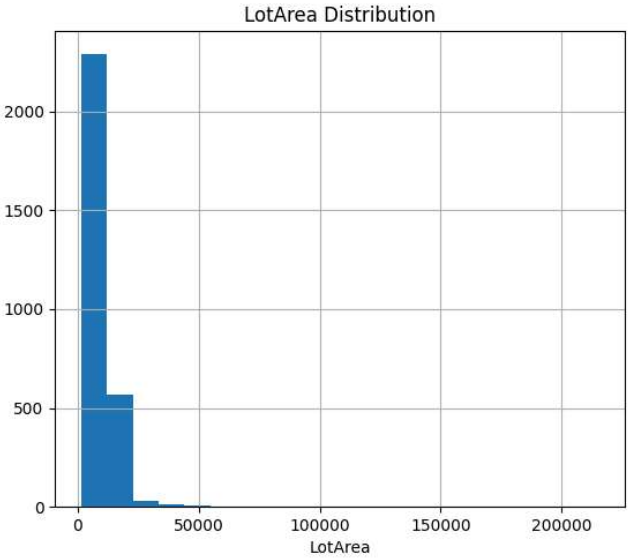


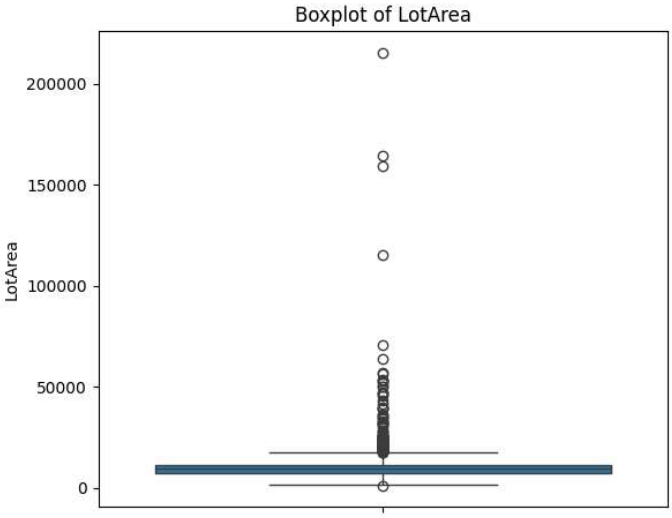


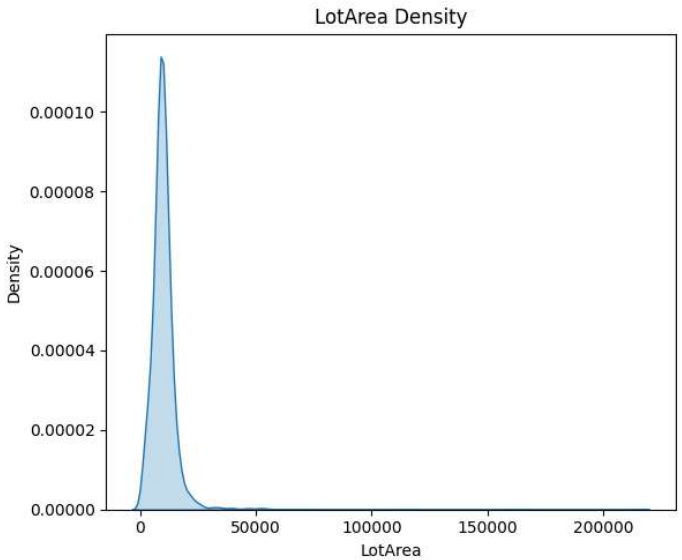


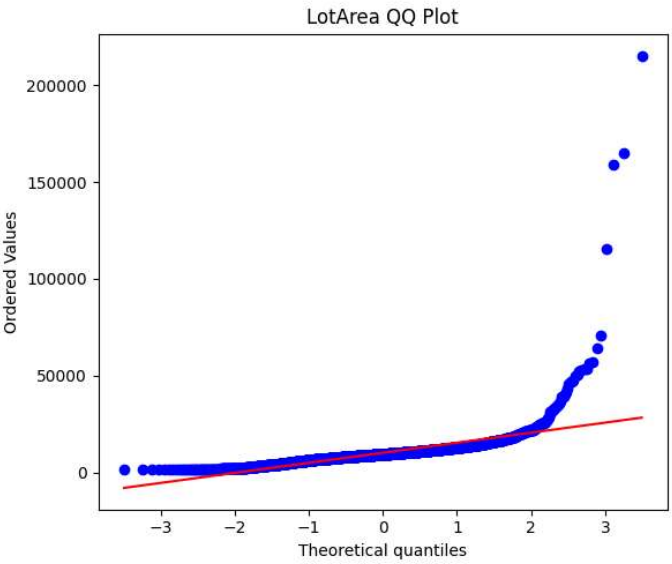


```
LotArea
##########################################
count     2919.000
mean     10168.114
std       7886.996
min       1300.000
5%        3182.000
10%       4922.400
20%       7007.600
30%       7960.400
40%       8741.000
50%       9453.000
60%      10151.600
70%      11001.200
80%      12203.800
90%      14300.600
95%      17142.900
99%      33038.640
max     215245.000
Name: LotArea, dtype: float64
##########################################
```

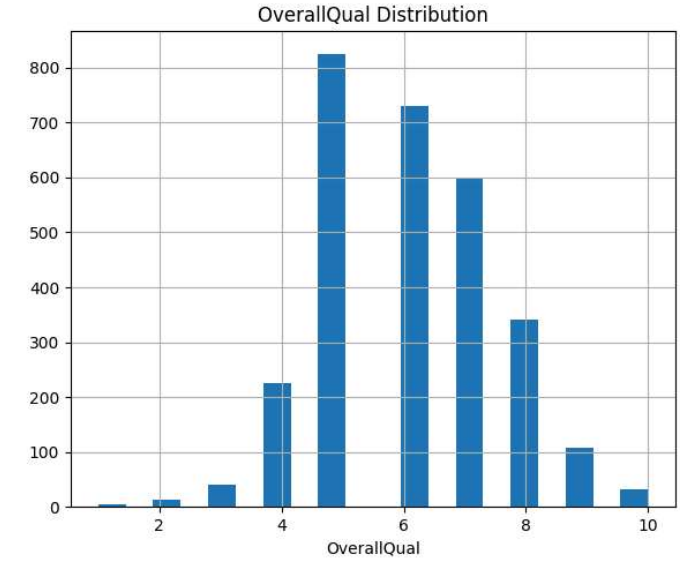


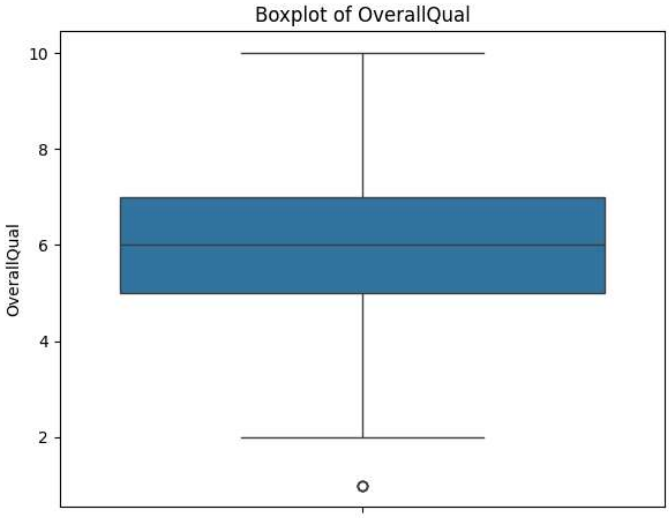


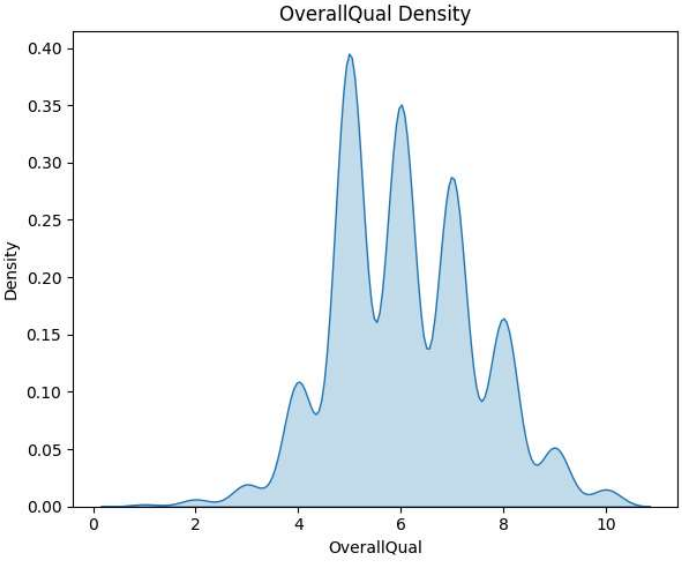


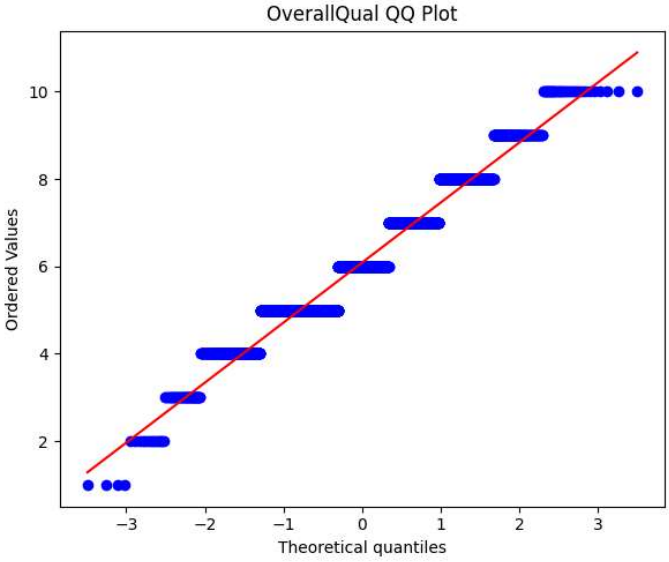


```
OverallQual
##########################################
count   2919.000
mean       6.089
std        1.410
min        1.000
5%         4.000
10%        5.000
20%        5.000
30%        5.000
40%        6.000
50%        6.000
60%        6.000
70%        7.000
80%        7.000
90%        8.000
95%        8.000
99%       10.000
max       10.000
Name: OverallQual, dtype: float64
##########################################
```

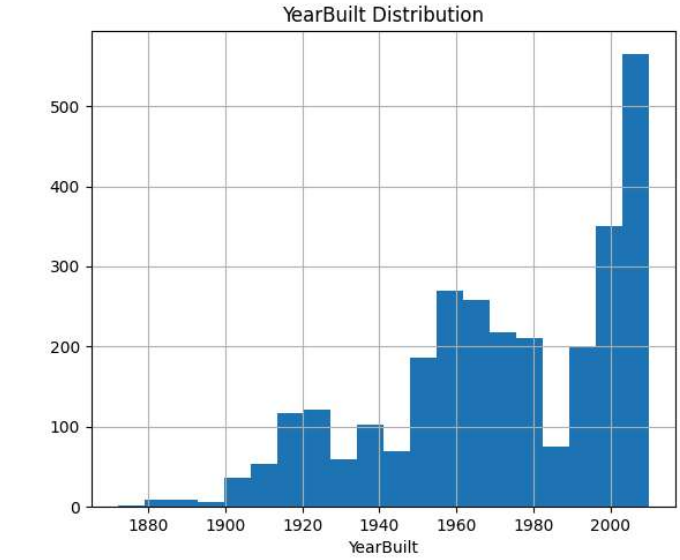


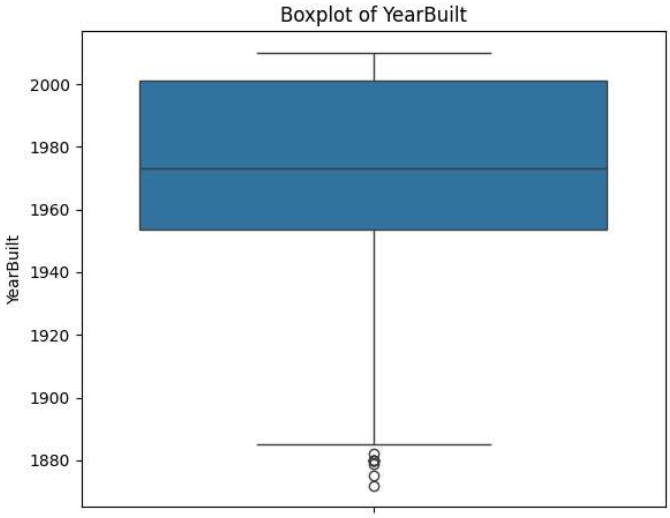


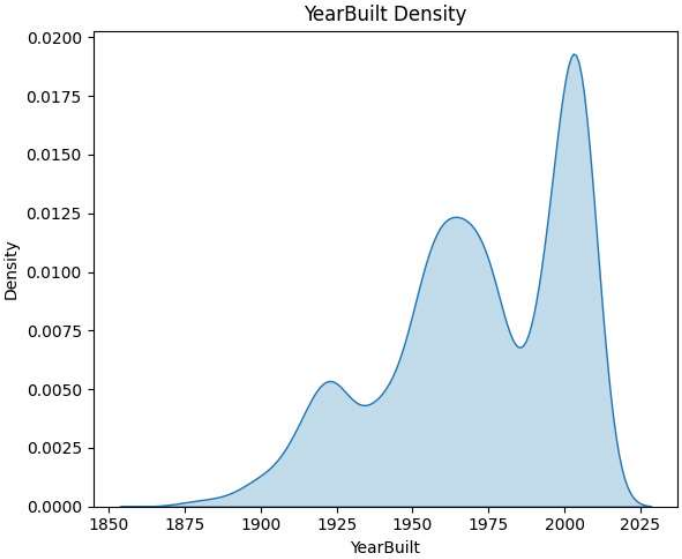


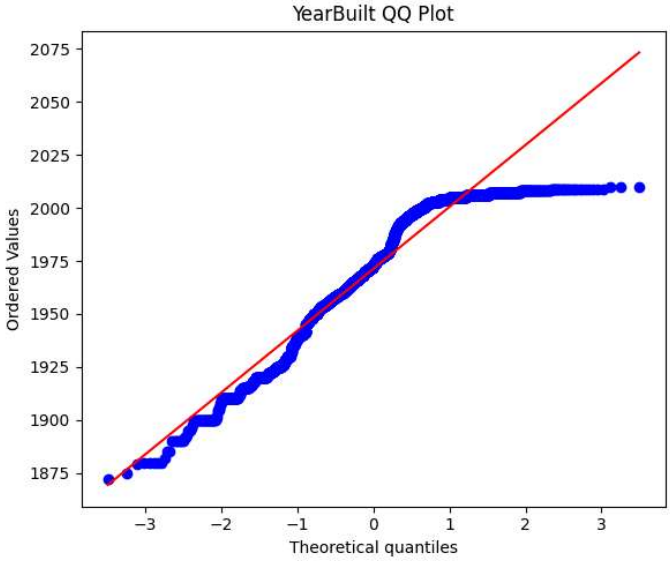


```
YearBuilt
##########################################
count   2919.000
mean    1971.313
std       30.291
min     1872.000
5%      1915.000
10%     1924.000
20%     1947.000
30%     1957.000
40%     1965.000
50%     1973.000
60%     1984.000
70%     1998.000
80%     2003.000
90%     2006.000
95%     2007.000
99%     2008.000
max     2010.000
Name: YearBuilt, dtype: float64
##########################################
```

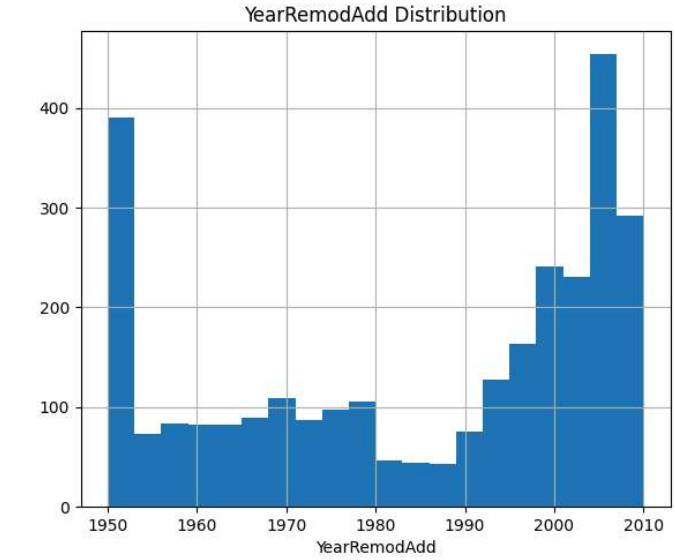


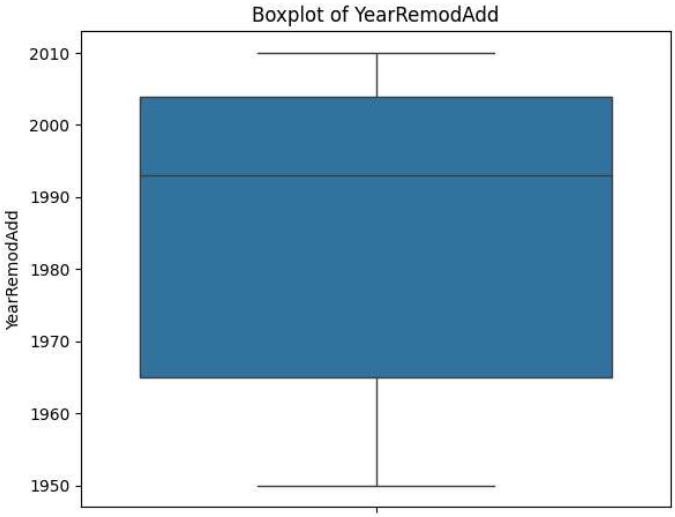


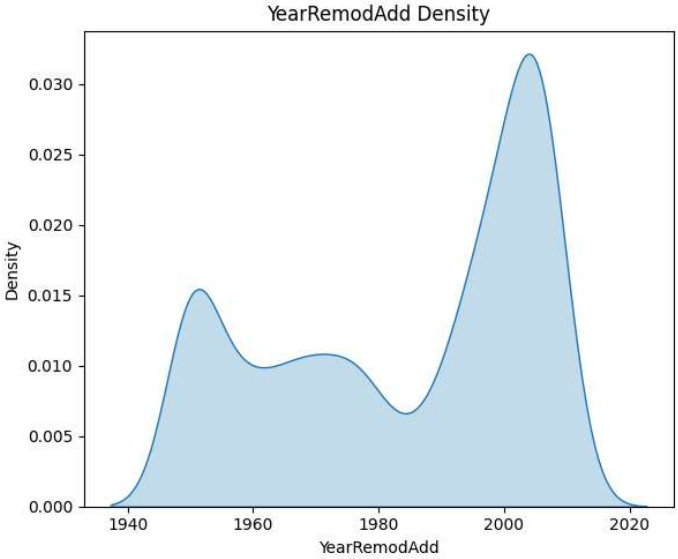


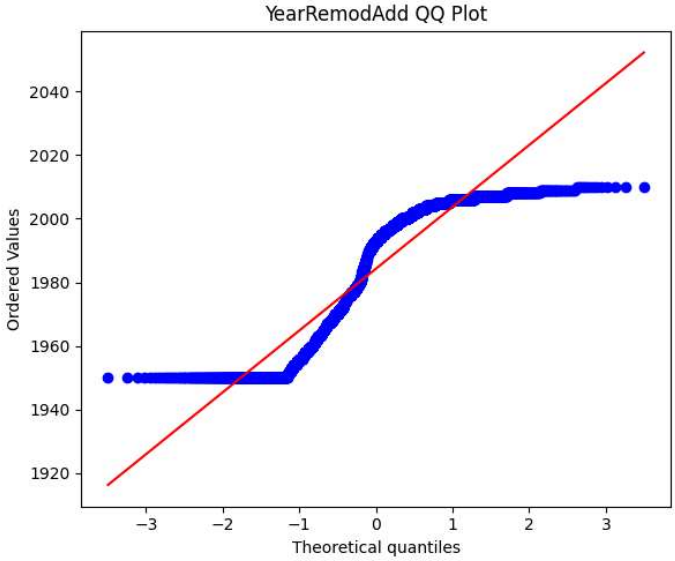


```
YearRemodAdd
##########################################
count   2919.000
mean    1984.264
std       20.894
min     1950.000
5%      1950.000
10%     1950.000
20%     1960.000
30%     1970.000
40%     1978.000
50%     1993.000
60%     1998.000
70%     2002.000
80%     2005.000
90%     2006.200
95%     2007.000
99%     2009.000
max     2010.000
Name: YearRemodAdd, dtype: float64
##########################################
```

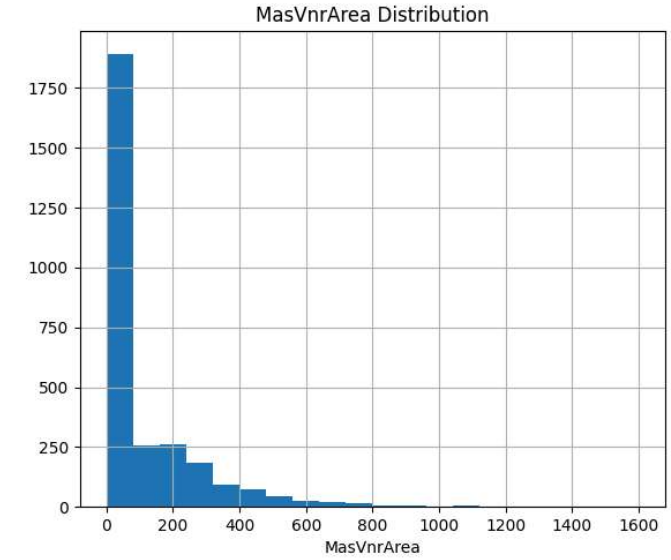


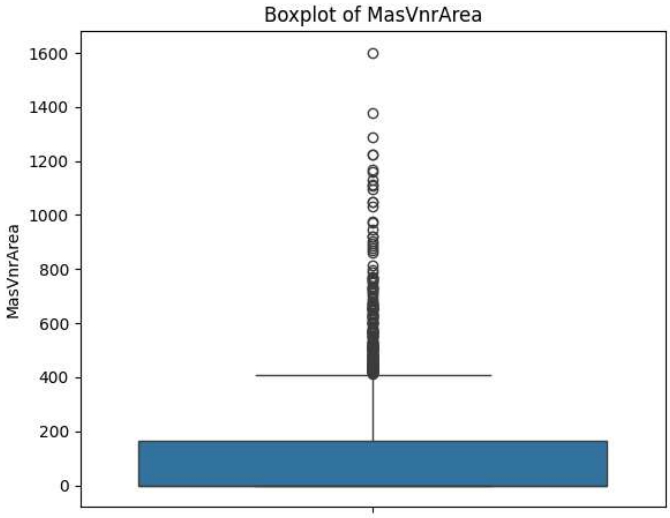


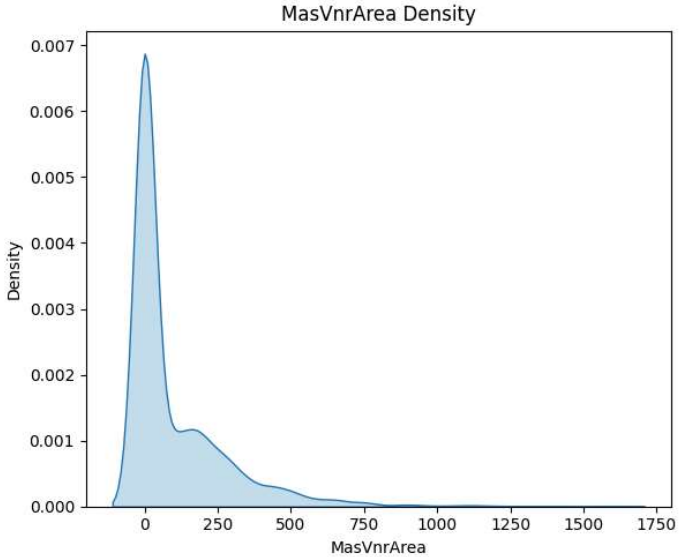


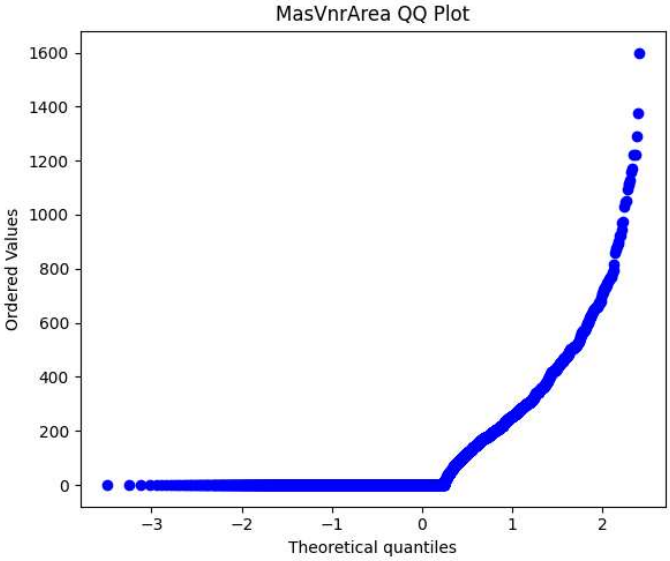


```
MasVnrArea
##########################################
count   2896.000
mean     102.201
std      179.334
min        0.000
5%         0.000
10%        0.000
20%        0.000
30%        0.000
40%        0.000
50%        0.000
60%        0.000
70%      120.000
80%      202.000
90%      325.500
95%      466.500
99%      771.050
max     1600.000
Name: MasVnrArea, dtype: float64
##########################################
```

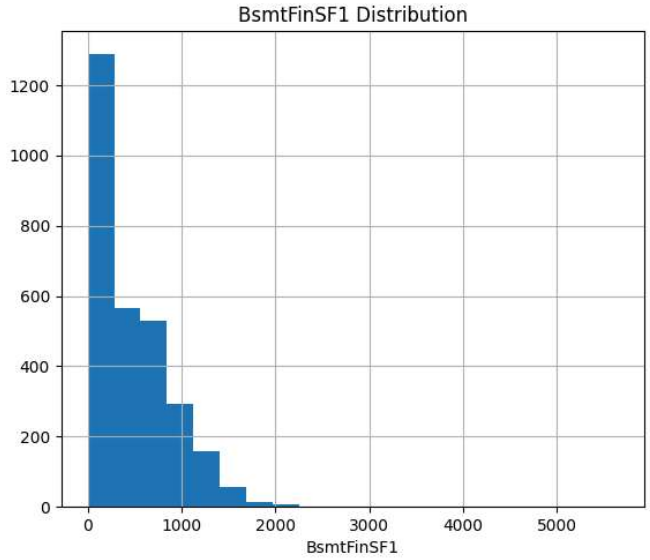


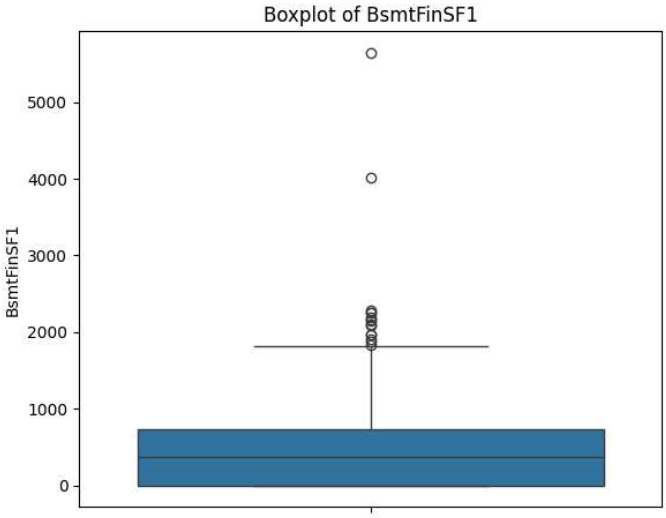


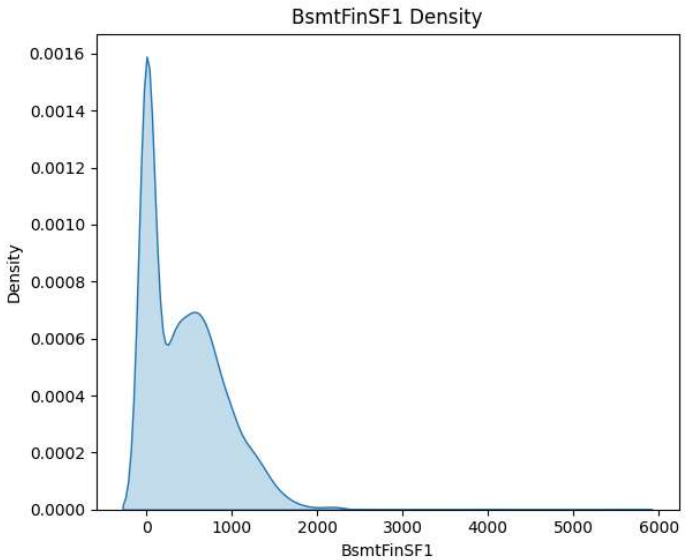


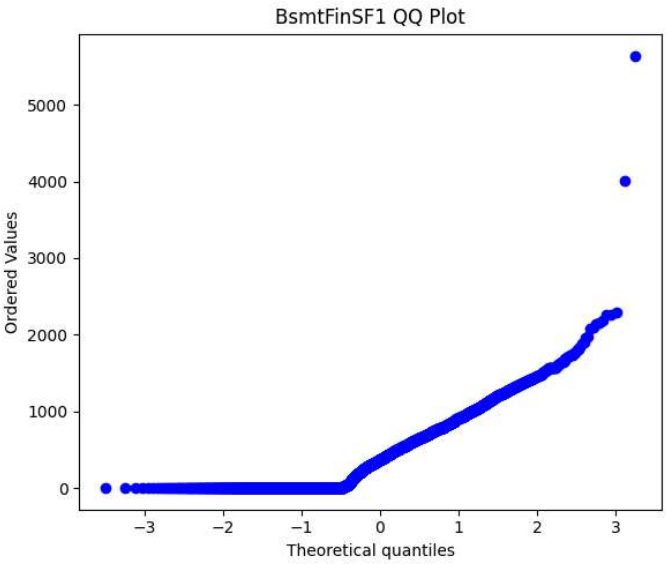


```
BsmtFinSF1
##########################################
count   2918.000
mean     441.423
std      455.611
min        0.000
5%         0.000
10%        0.000
20%        0.000
30%        0.000
40%      202.600
50%      368.500
60%      515.200
70%      656.000
80%      812.000
90%     1056.900
95%     1274.000
99%     1635.320
max     5644.000
Name: BsmtFinSF1, dtype: float64
##########################################
```

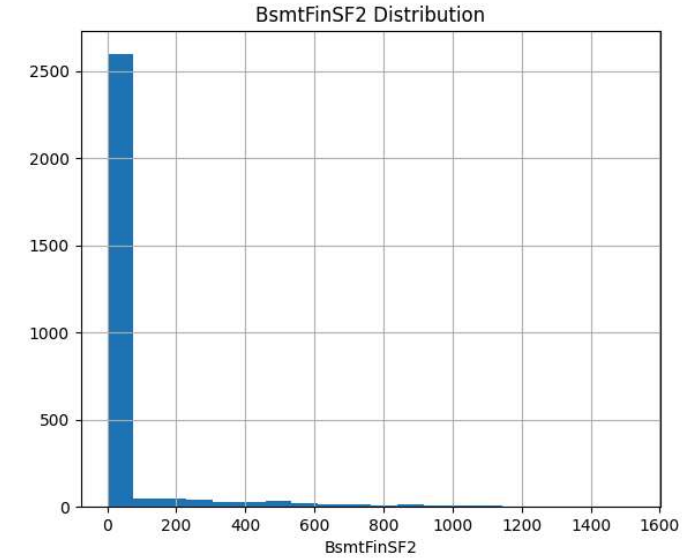


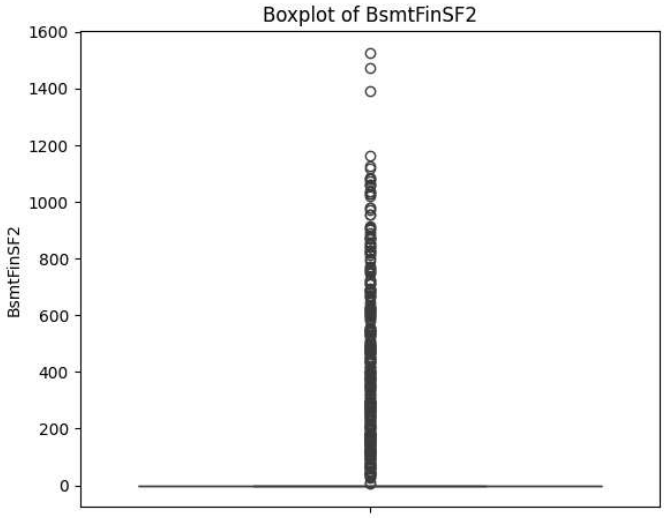


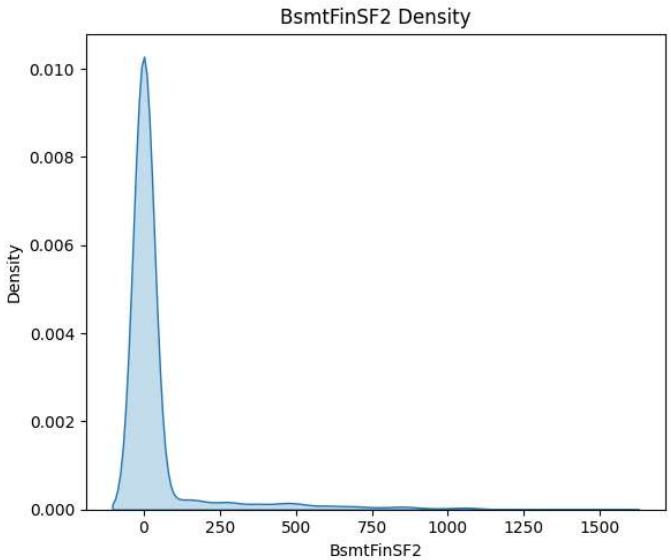


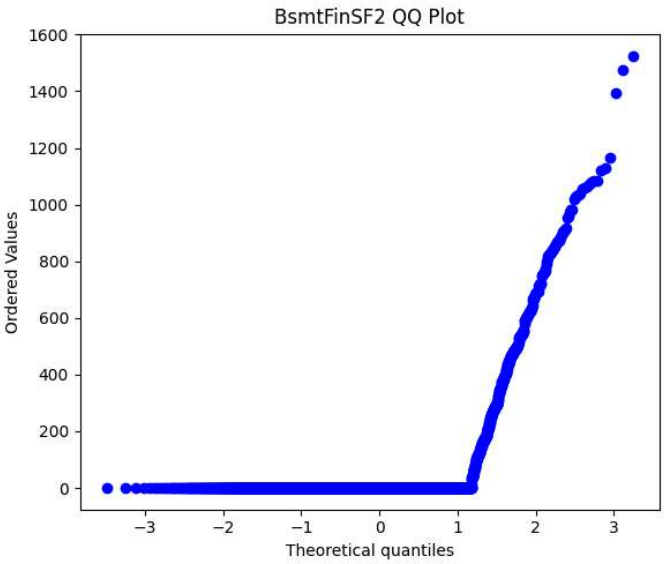


```
BsmtFinSF2
##########################################
count   2918.000
mean      49.582
std      169.206
min        0.000
5%         0.000
10%        0.000
20%        0.000
30%        0.000
40%        0.000
50%        0.000
60%        0.000
70%        0.000
80%        0.000
90%      125.600
95%      435.000
99%      874.660
max     1526.000
Name: BsmtFinSF2, dtype: float64
##########################################
```

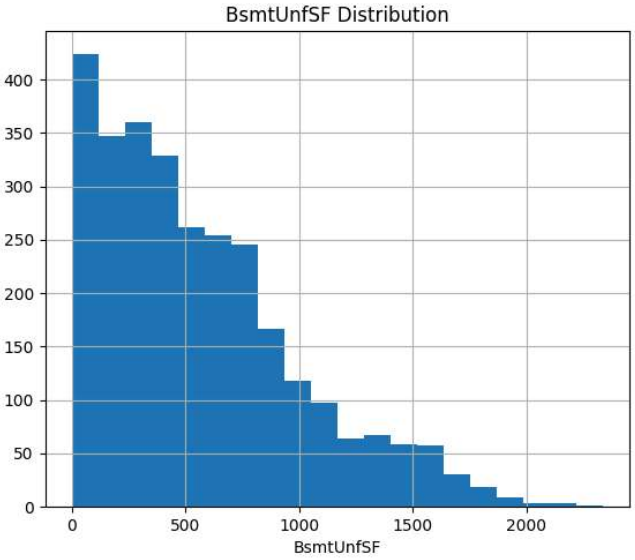


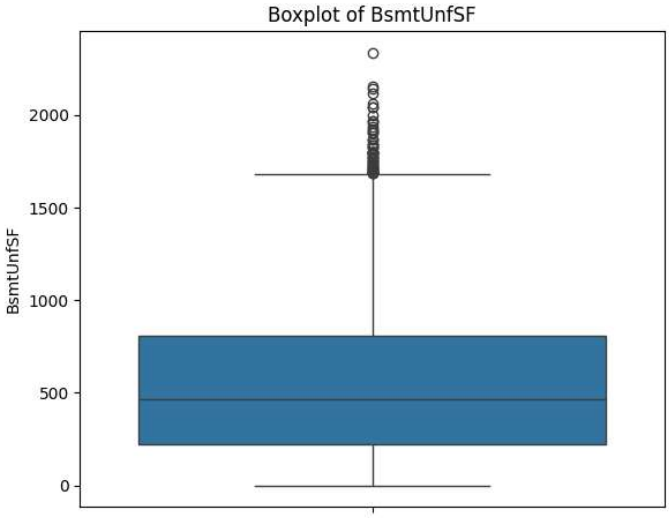


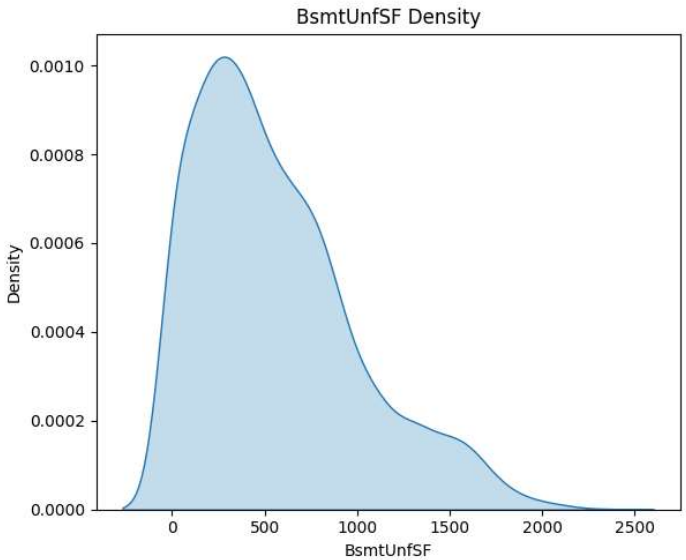


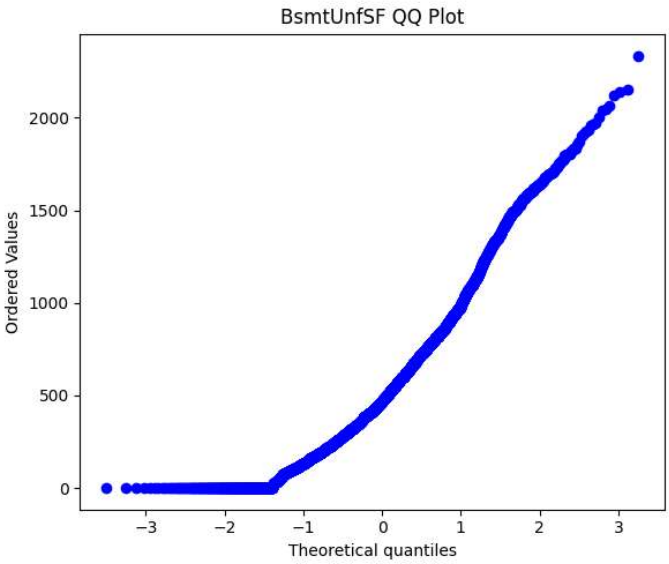


```
BsmtUnfSF
##########################################
count   2918.000
mean     560.772
std      439.544
min        0.000
5%         0.000
10%       56.000
20%      174.000
30%      270.000
40%      365.800
50%      467.000
60%      595.000
70%      732.000
80%      892.600
90%     1212.600
95%     1474.900
99%     1776.490
max     2336.000
Name: BsmtUnfSF, dtype: float64
##########################################
```

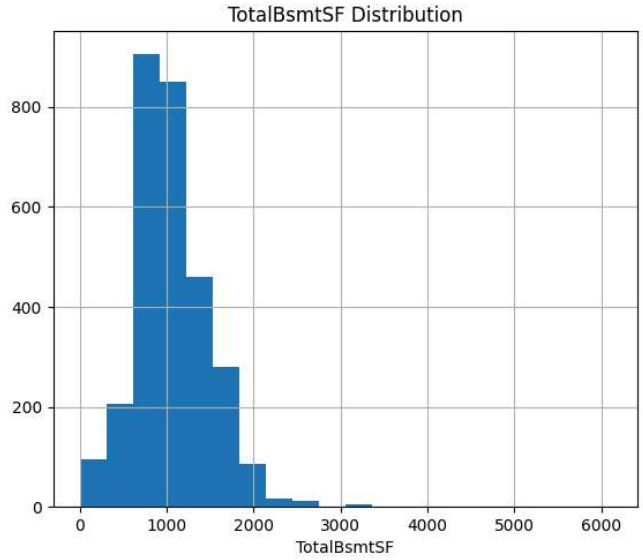


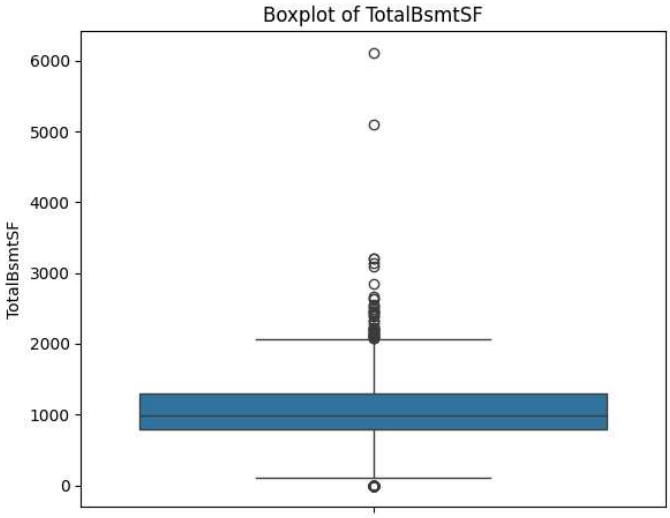


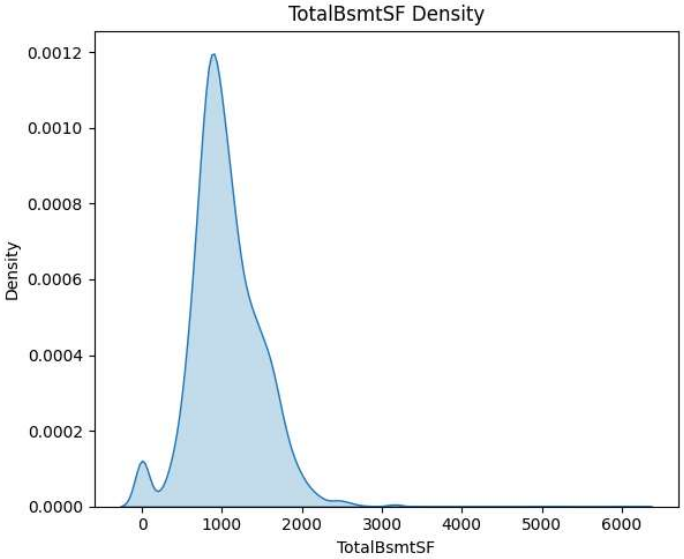


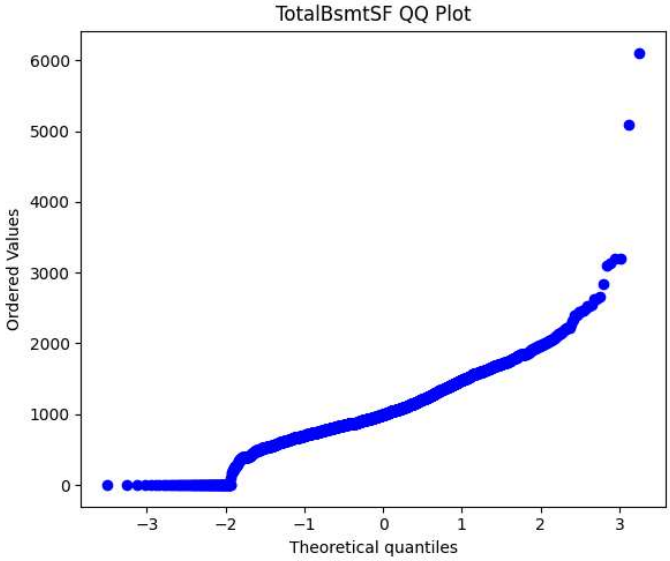


```
TotalBsmtSF
##########################################
count   2918.000
mean    1051.778
std      440.766
min        0.000
5%       455.250
10%      600.000
20%      741.000
30%      836.000
40%      911.000
50%      989.500
60%     1089.200
70%     1216.000
80%     1392.000
90%     1614.000
95%     1776.150
99%     2198.300
max     6110.000
Name: TotalBsmtSF, dtype: float64
##########################################
```

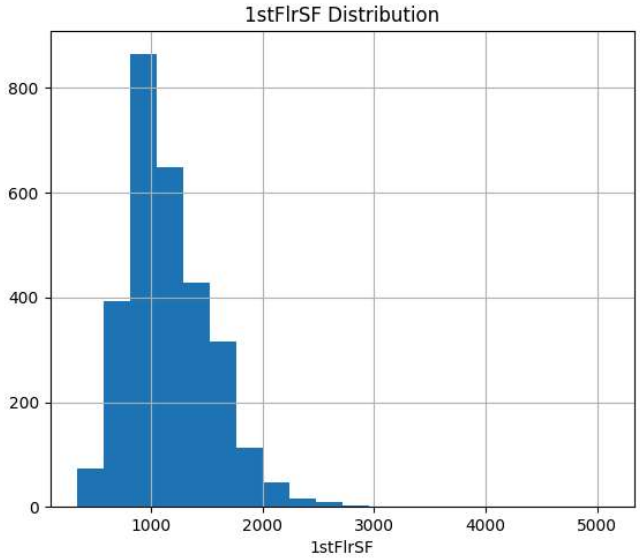


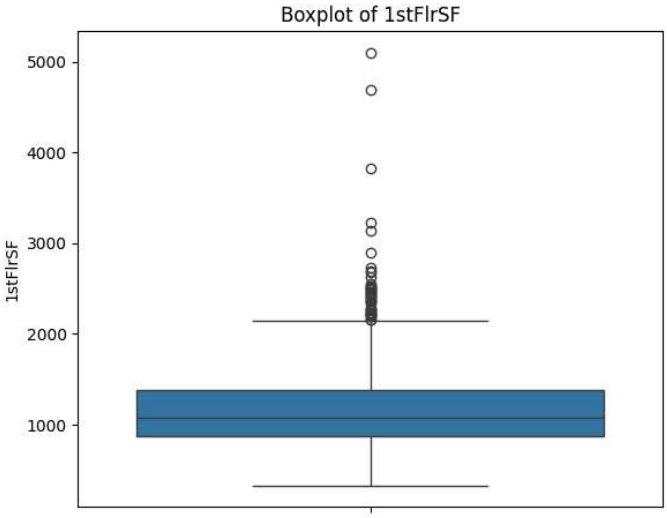


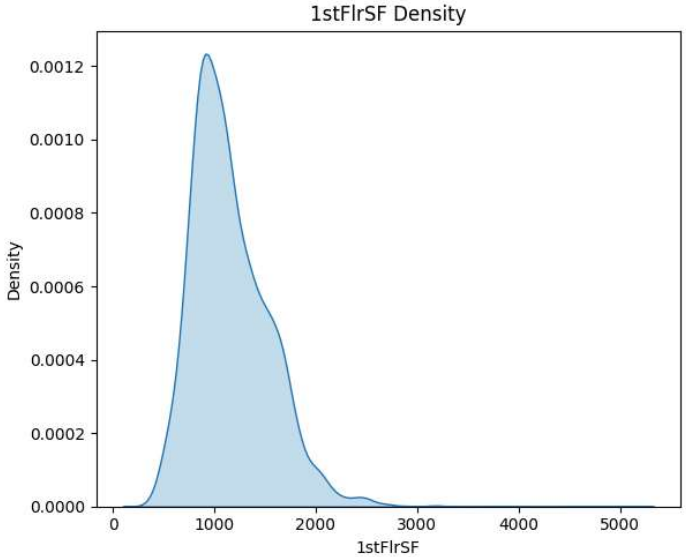


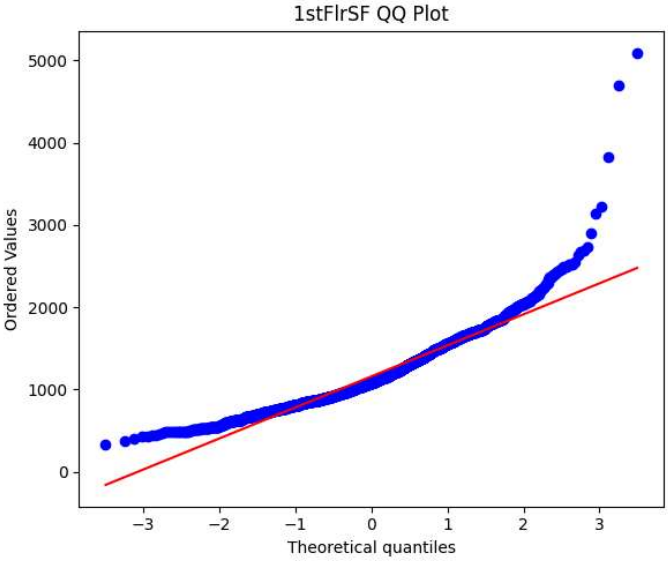


```
1stFlrSF
##########################################
count   2919.000
mean    1159.582
std      392.362
min      334.000
5%       665.900
10%      744.800
20%      847.000
30%      914.000
40%      996.200
50%     1082.000
60%     1180.000
70%     1314.000
80%     1483.400
90%     1675.000
95%     1830.100
99%     2288.020
max     5095.000
Name: 1stFlrSF, dtype: float64
##########################################
```

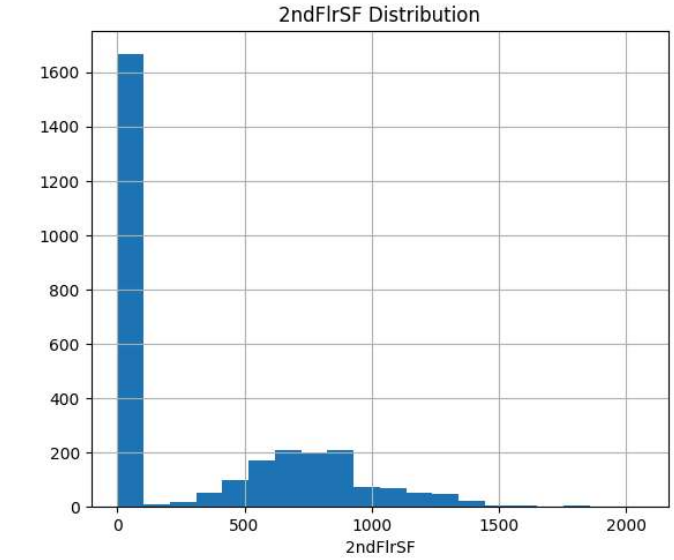


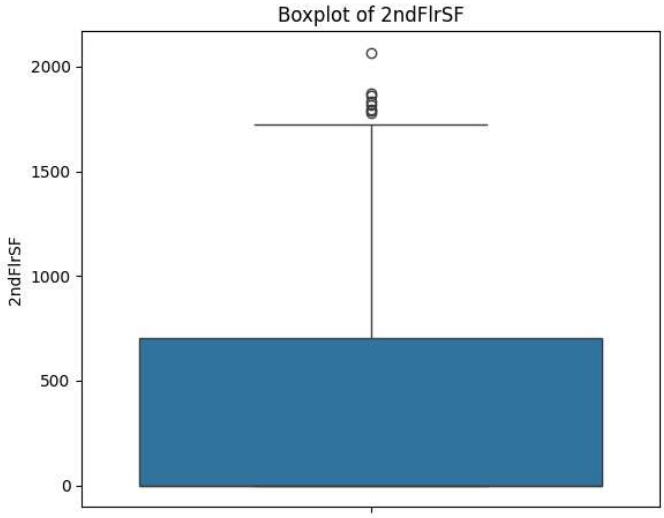


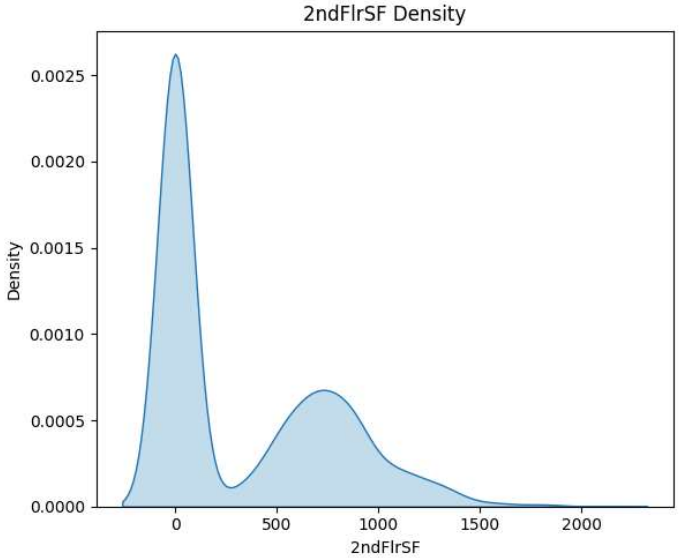


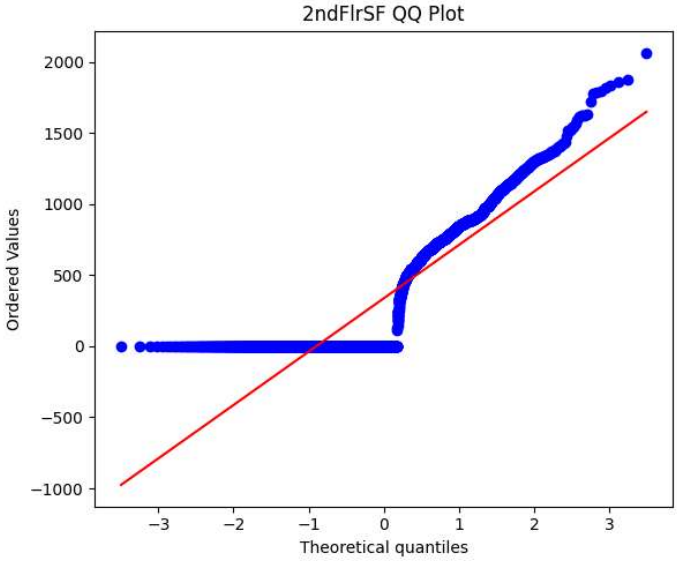


```
2ndFlrSF
##########################################
count   2919.000
mean     336.484
std      428.701
min        0.000
5%         0.000
10%        0.000
20%        0.000
30%        0.000
40%        0.000
50%        0.000
60%      427.400
70%      636.000
80%      770.800
90%      925.000
95%     1131.200
99%     1400.200
max     2065.000
Name: 2ndFlrSF, dtype: float64
##########################################
```

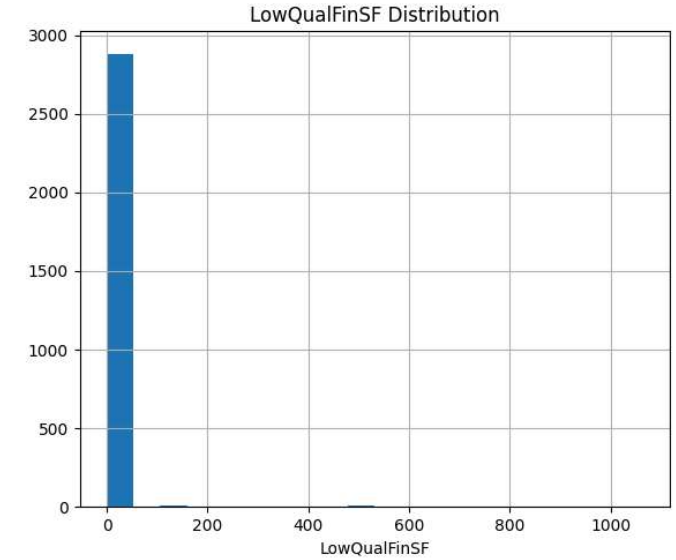


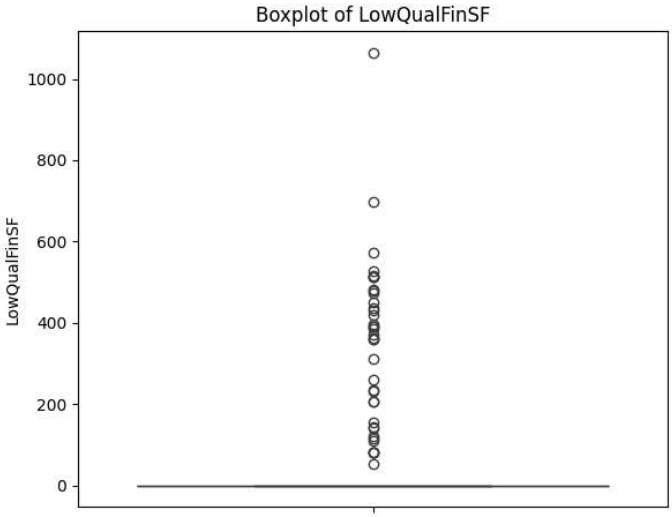


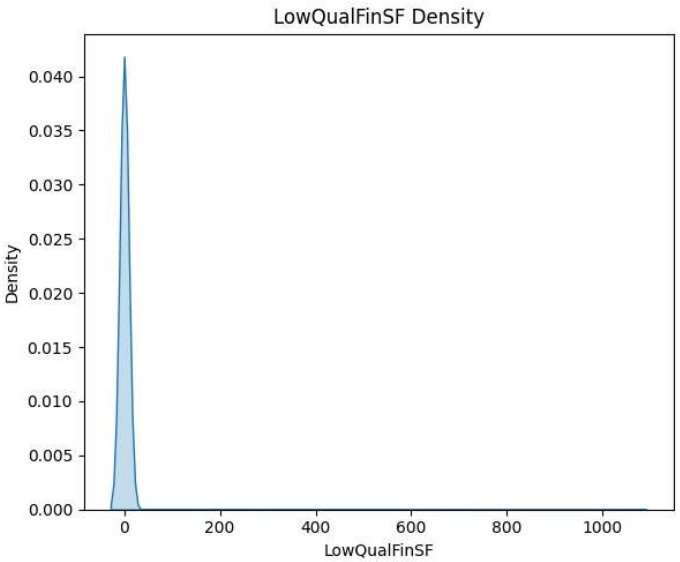


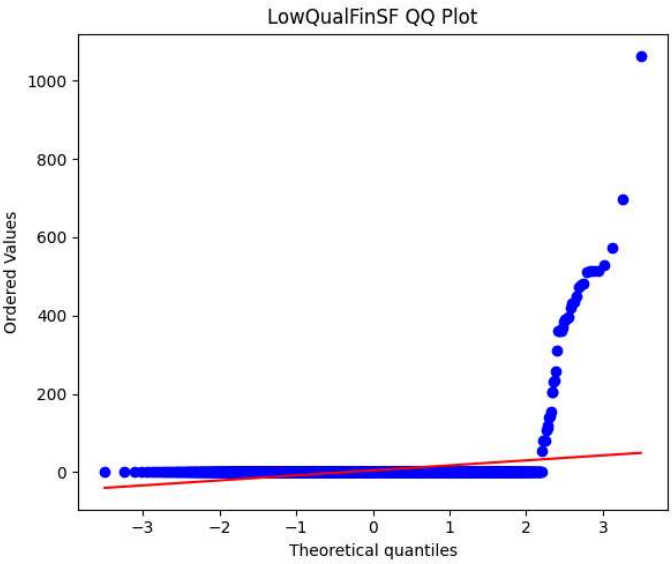


```
LowQualFinSF
##########################################
count   2919.000
mean       4.694
std       46.397
min        0.000
5%         0.000
10%        0.000
20%        0.000
30%        0.000
40%        0.000
50%        0.000
60%        0.000
70%        0.000
80%        0.000
90%        0.000
95%        0.000
99%      153.840
max     1064.000
Name: LowQualFinSF, dtype: float64
##########################################
```

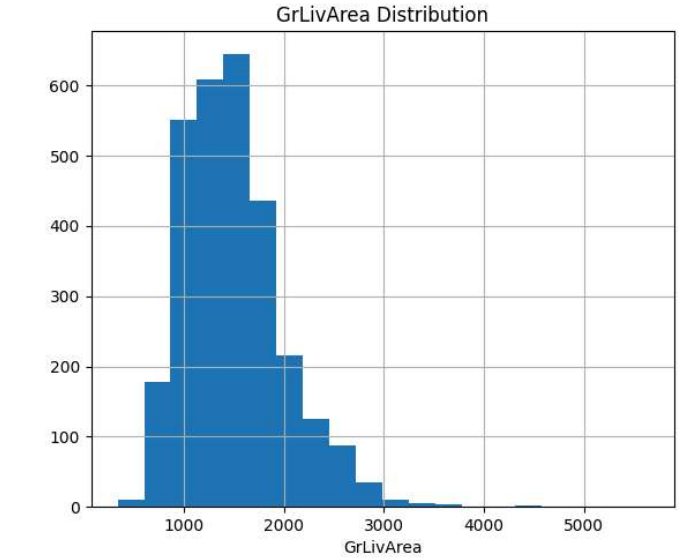


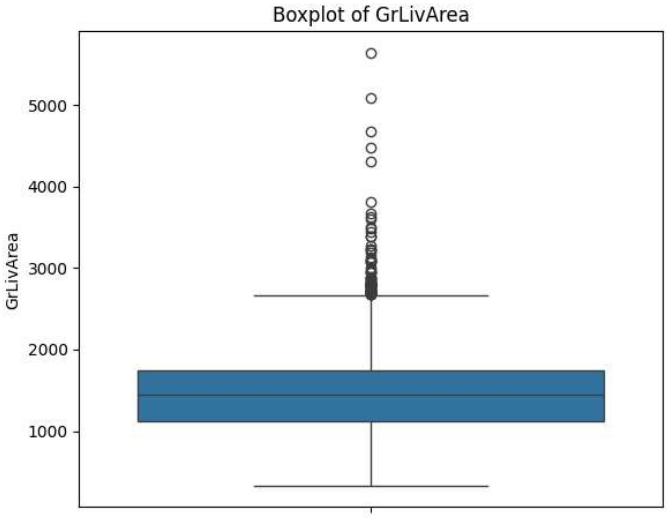


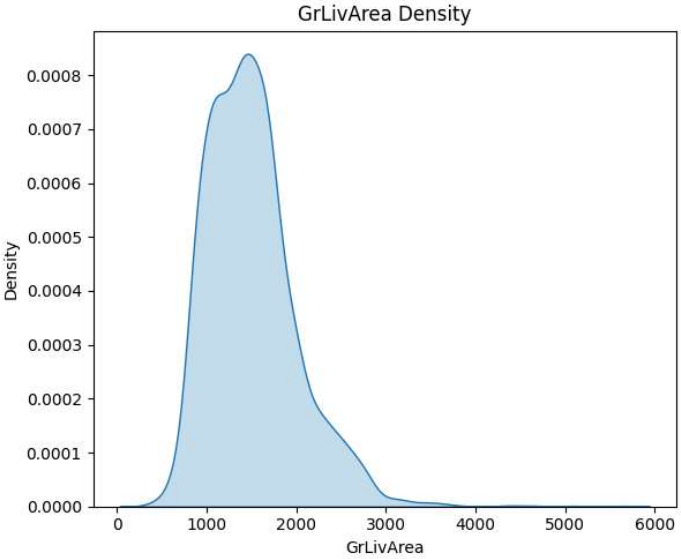


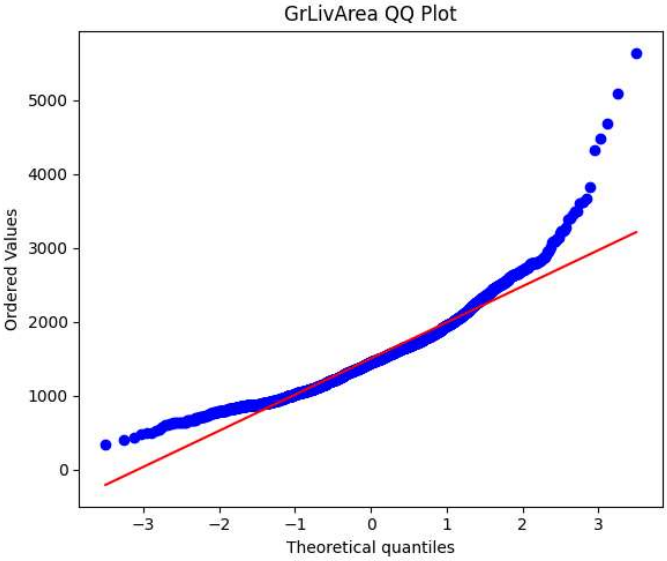


```
GrLivArea
##########################################
count   2919.000
mean    1500.760
std      506.051
min      334.000
5%       861.000
10%      923.800
20%     1064.600
30%     1200.000
40%     1329.200
50%     1444.000
60%     1560.000
70%     1680.000
80%     1838.400
90%     2153.200
95%     2464.200
99%     2935.720
max     5642.000
Name: GrLivArea, dtype: float64
##########################################
```

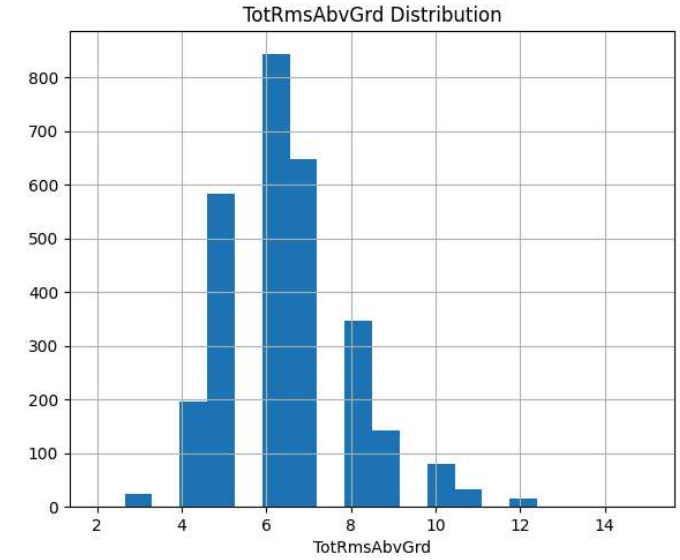


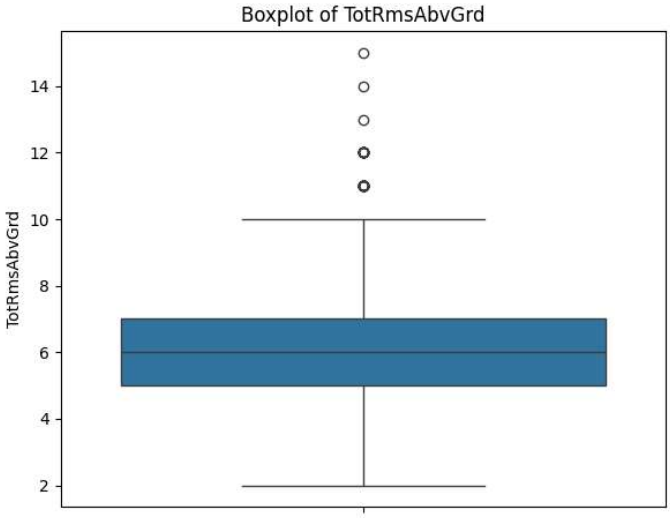


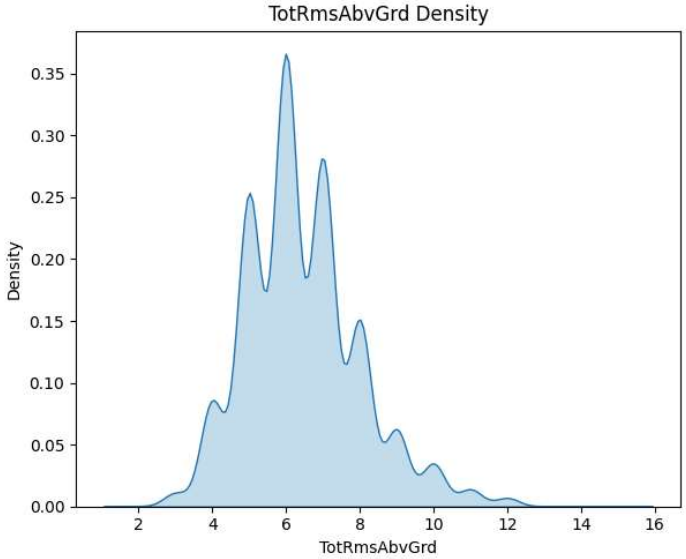


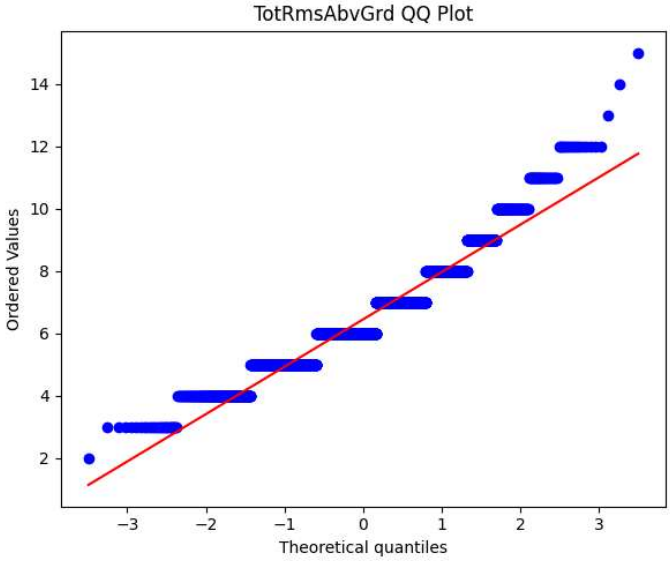


```
TotRmsAbvGrd
##########################################
count   2919.000
mean       6.452
std        1.569
min        2.000
5%         4.000
10%        5.000
20%        5.000
30%        6.000
40%        6.000
50%        6.000
60%        7.000
70%        7.000
80%        8.000
90%        8.000
95%        9.000
99%       11.000
max       15.000
Name: TotRmsAbvGrd, dtype: float64
##########################################
```

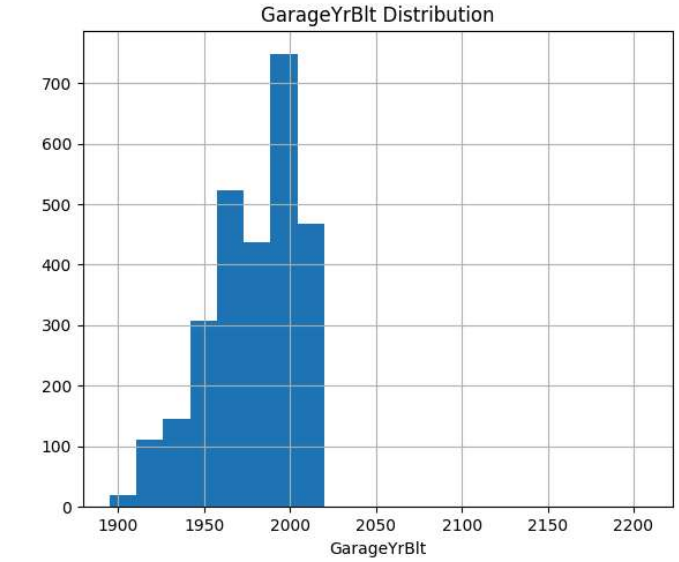


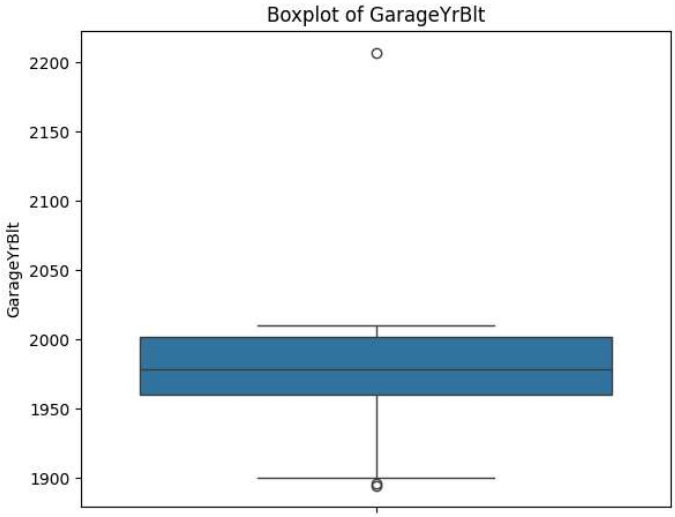


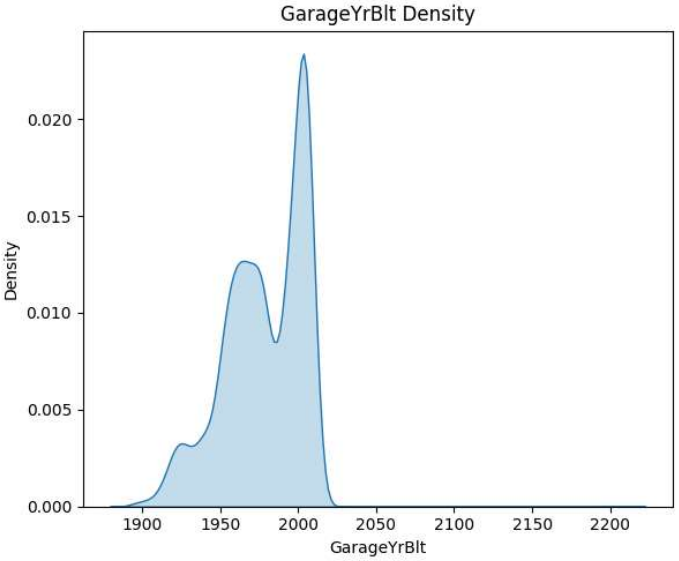


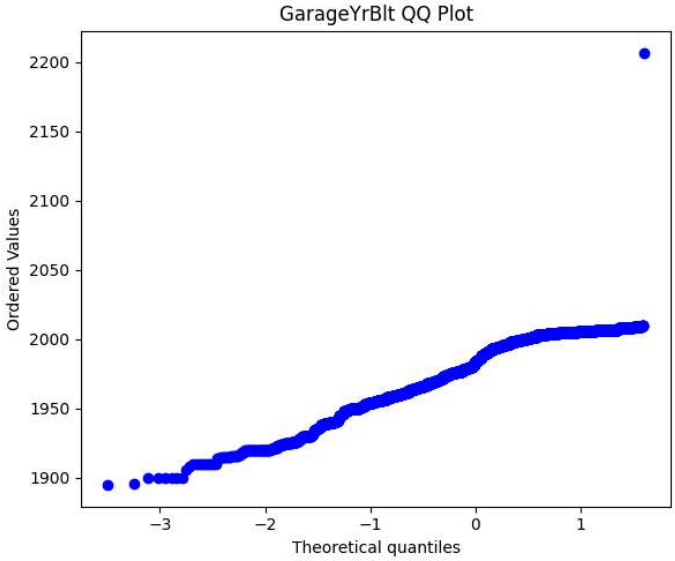


```
GarageYrBlt
##########################################
count   2760.000
mean    1978.113
std       25.574
min     1895.000
5%      1928.000
10%     1941.000
20%     1957.000
30%     1964.000
40%     1972.000
50%     1979.000
60%     1993.000
70%     1999.000
80%     2004.000
90%     2006.000
95%     2007.000
99%     2009.000
max     2207.000
Name: GarageYrBlt, dtype: float64
##########################################
```

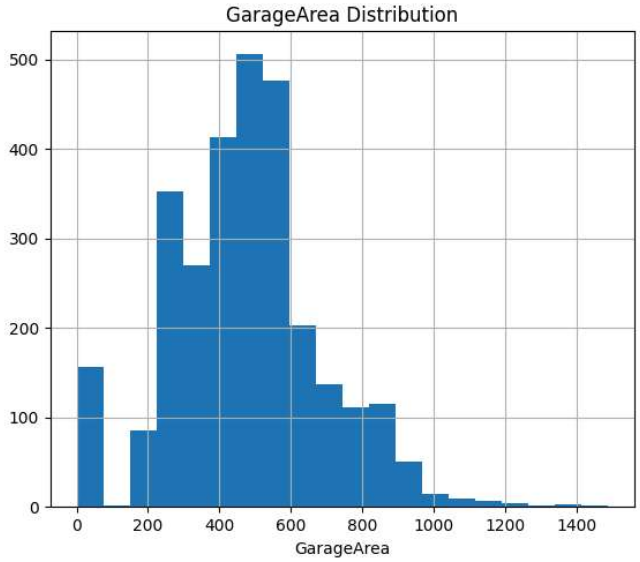


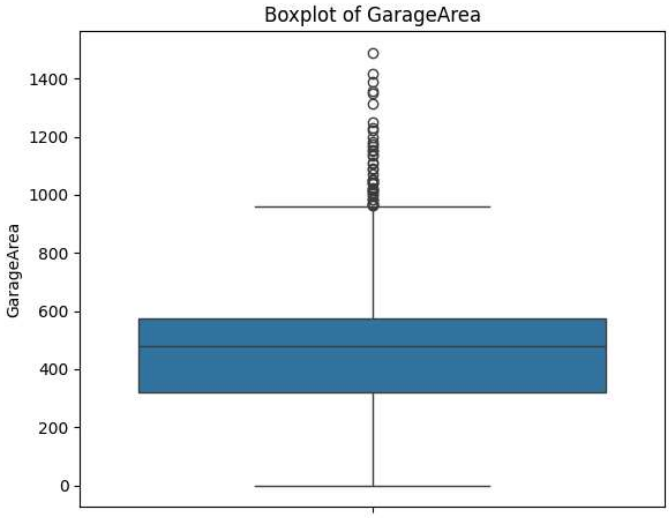


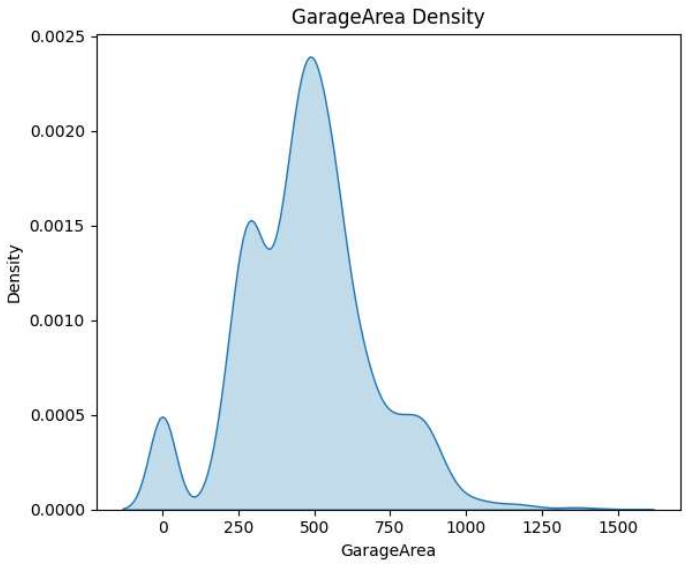


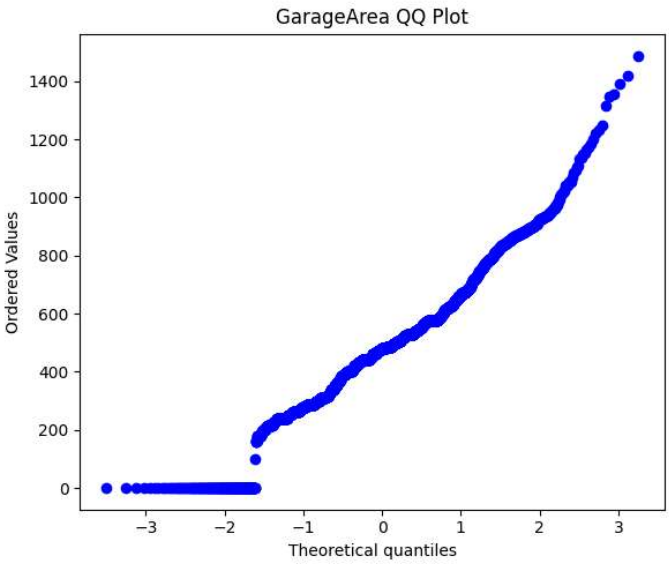


```
GarageArea
##########################################
count   2918.000
mean     472.875
std      215.395
min        0.000
5%         0.000
10%      240.000
20%      296.000
30%      379.000
40%      440.000
50%      480.000
60%      513.000
70%      560.900
80%      621.000
90%      758.000
95%      856.150
99%     1019.490
max     1488.000
Name: GarageArea, dtype: float64
##########################################
```

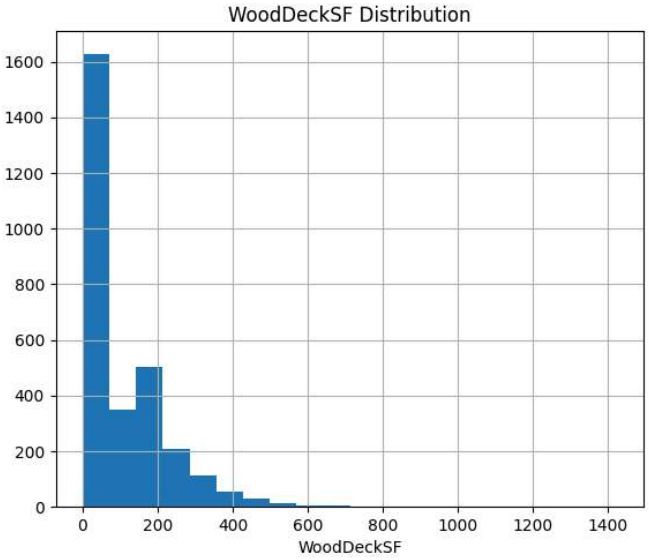


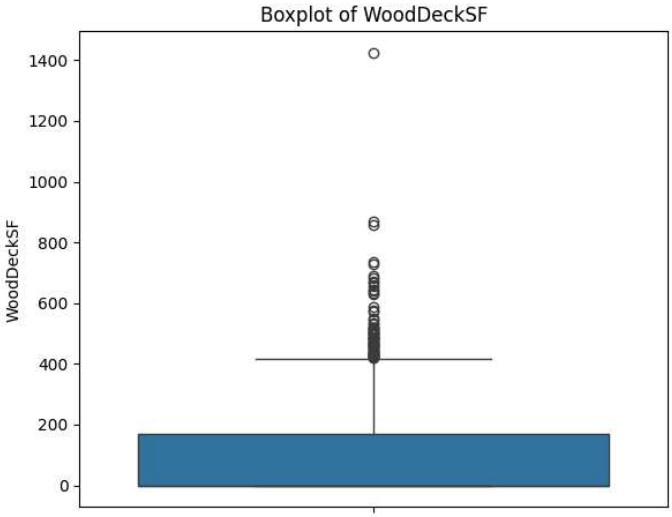


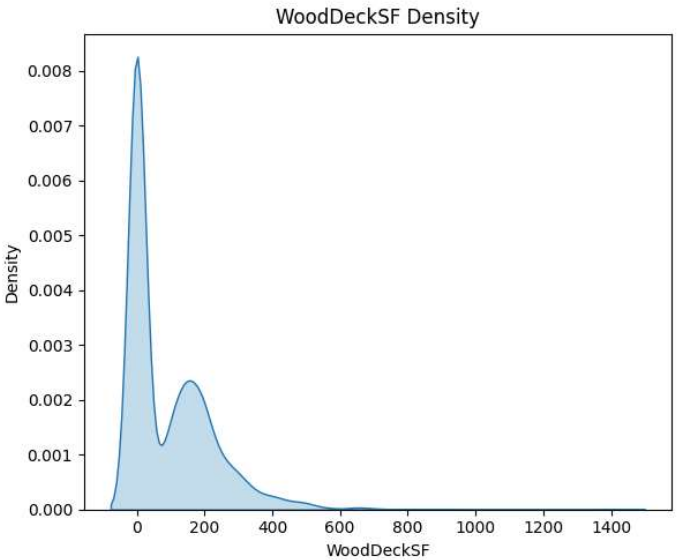


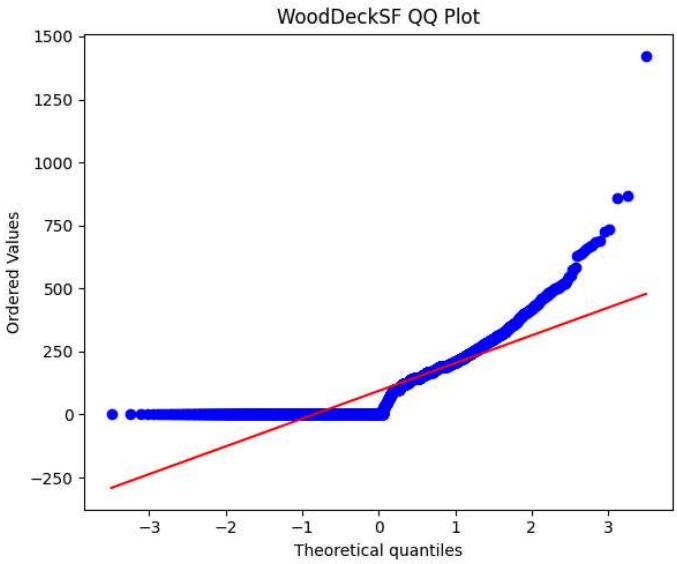


```
WoodDeckSF
##########################################
count   2919.000
mean      93.710
std      126.527
min        0.000
5%         0.000
10%        0.000
20%        0.000
30%        0.000
40%        0.000
50%        0.000
60%      100.000
70%      144.000
80%      192.000
90%      257.000
95%      328.000
99%      500.820
max     1424.000
Name: WoodDeckSF, dtype: float64
##########################################
```

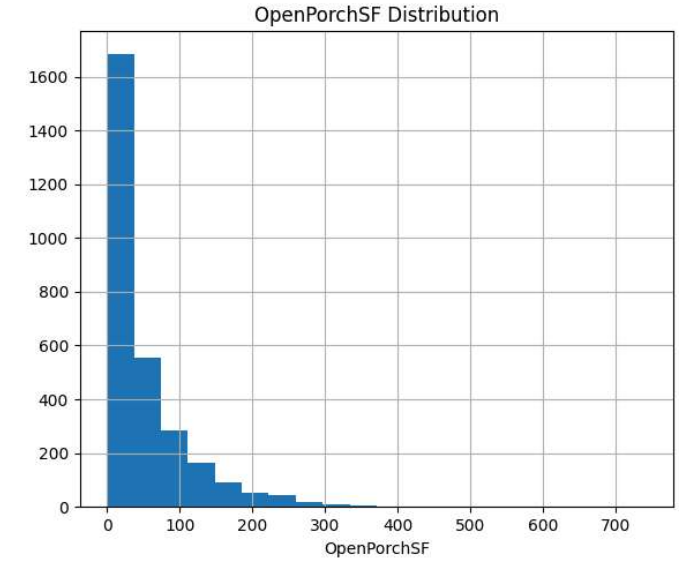


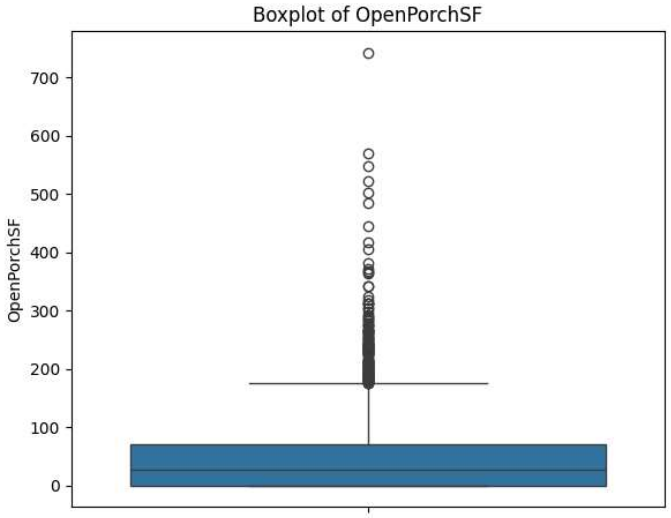


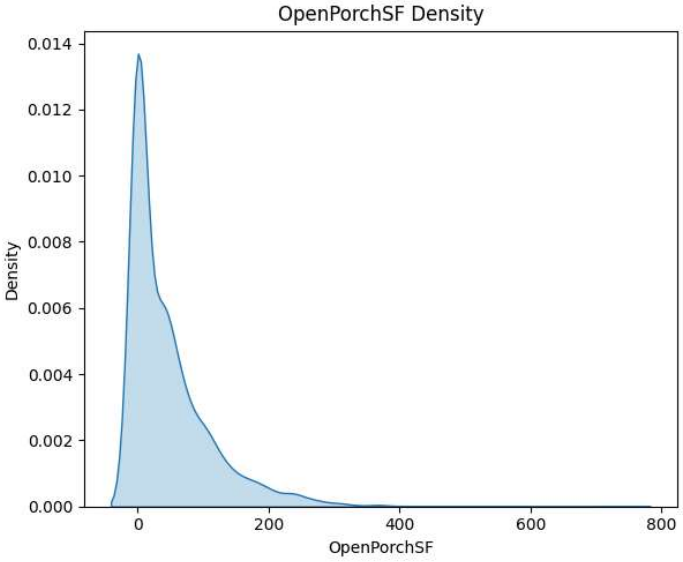


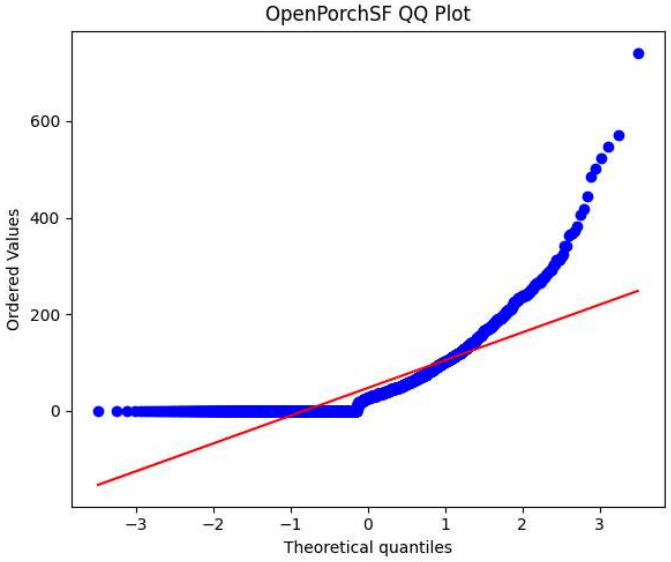


```
OpenPorchSF
##########################################
count   2919.000
mean      47.487
std       67.575
min        0.000
5%         0.000
10%        0.000
20%        0.000
30%        0.000
40%        0.000
50%       26.000
60%       40.000
70%       58.000
80%       85.000
90%      131.200
95%      183.100
99%      284.460
max      742.000
Name: OpenPorchSF, dtype: float64
##########################################
```

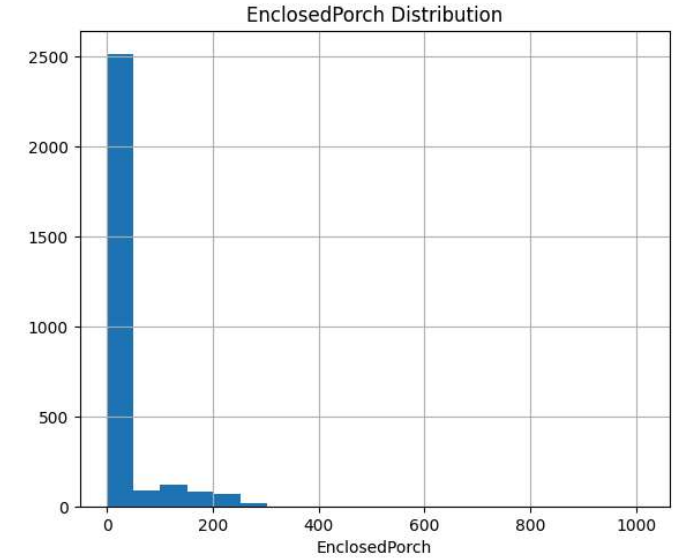


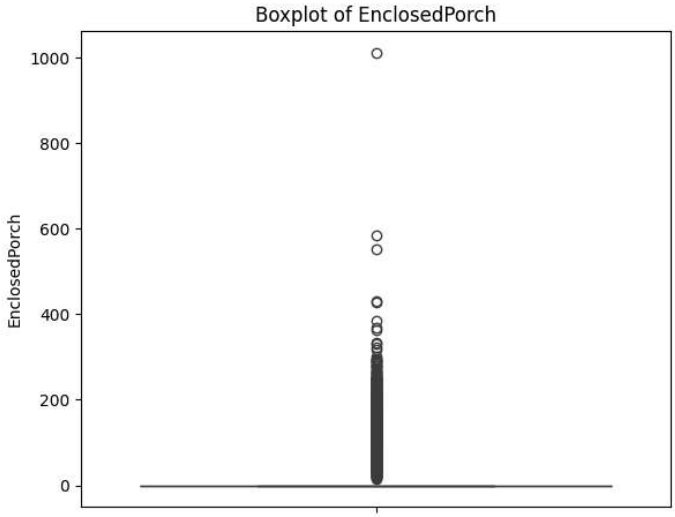


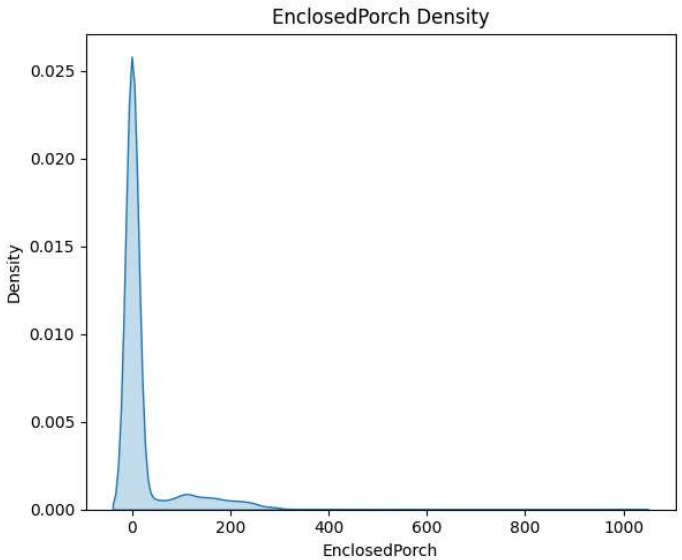


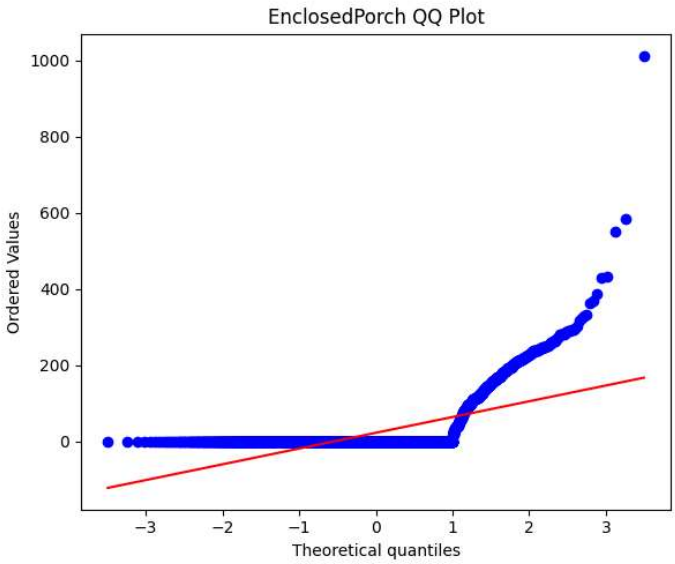


```
EnclosedPorch
##########################################
count   2919.000
mean      23.098
std       64.244
min        0.000
5%         0.000
10%        0.000
20%        0.000
30%        0.000
40%        0.000
50%        0.000
60%        0.000
70%        0.000
80%        0.000
90%      112.000
95%      176.000
99%      264.000
max     1012.000
Name: EnclosedPorch, dtype: float64
##########################################
```

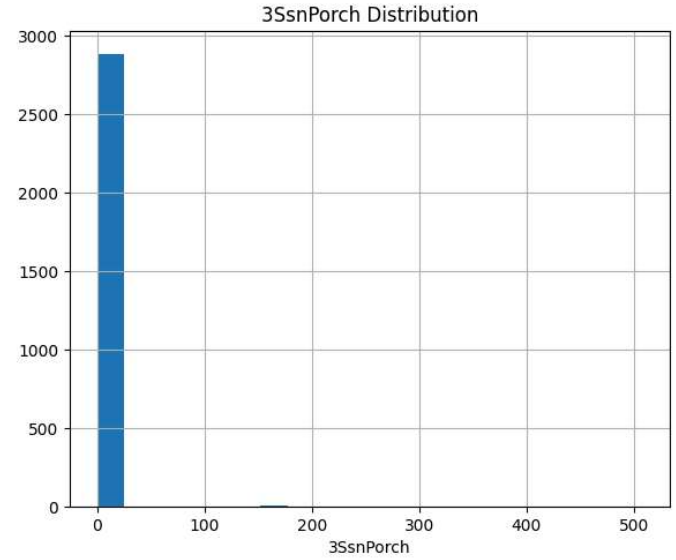


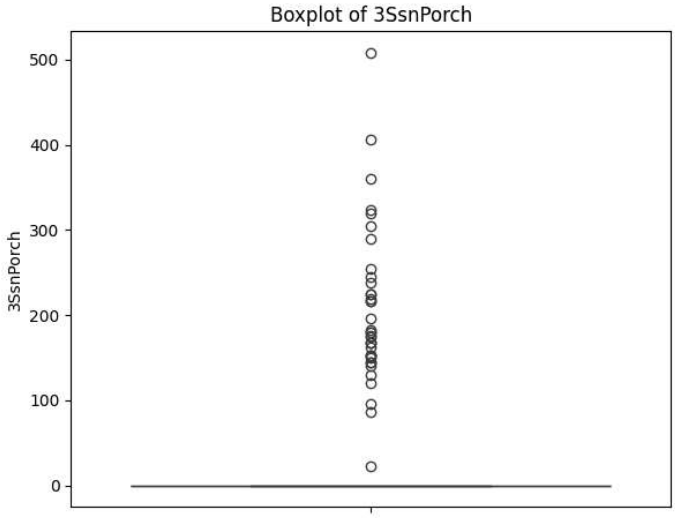


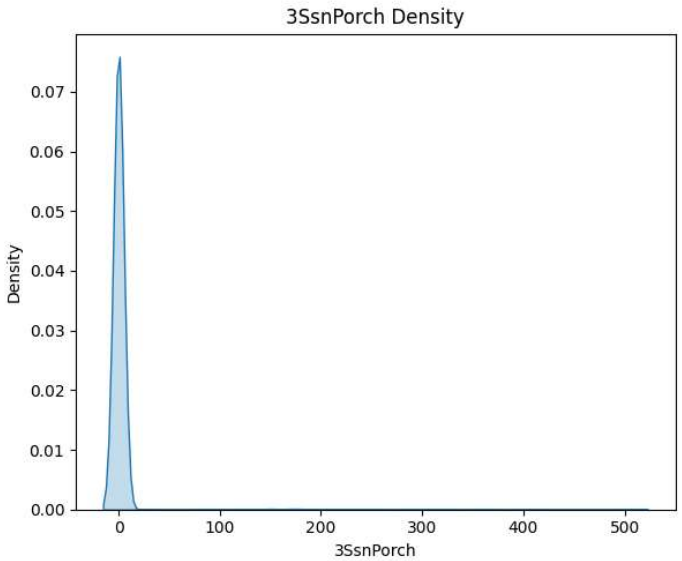


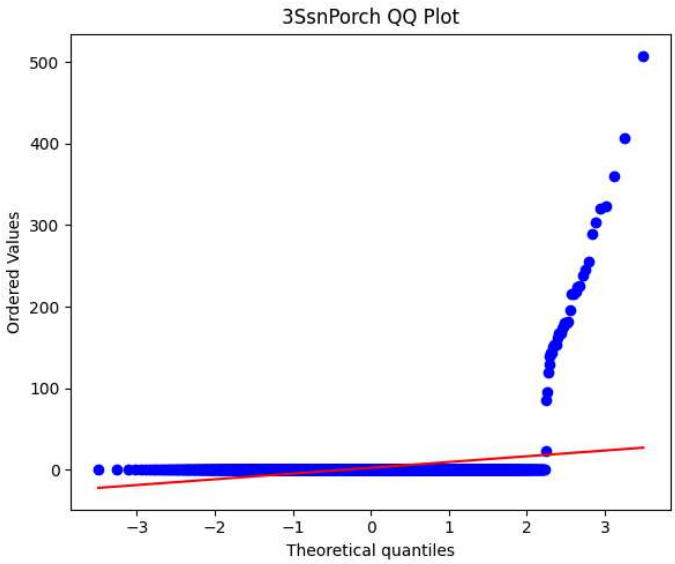


```
3SsnPorch
##########################################
count   2919.000
mean       2.602
std       25.188
min        0.000
5%         0.000
10%        0.000
20%        0.000
30%        0.000
40%        0.000
50%        0.000
60%        0.000
70%        0.000
80%        0.000
90%        0.000
95%        0.000
99%      144.000
max      508.000
Name: 3SsnPorch, dtype: float64
##########################################
```

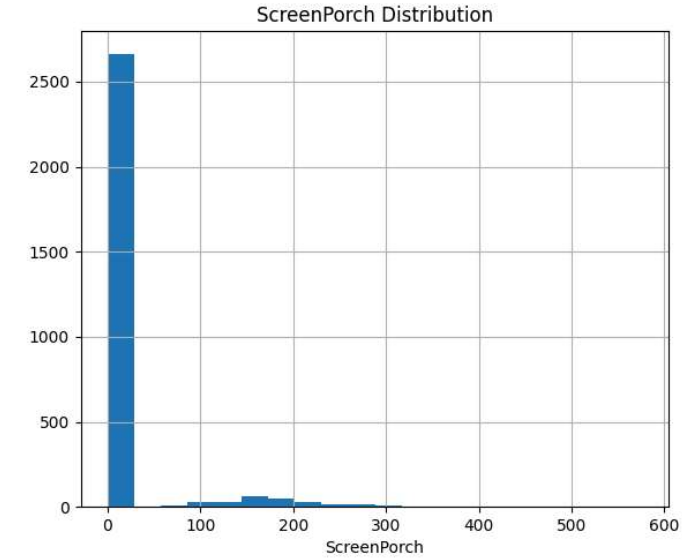


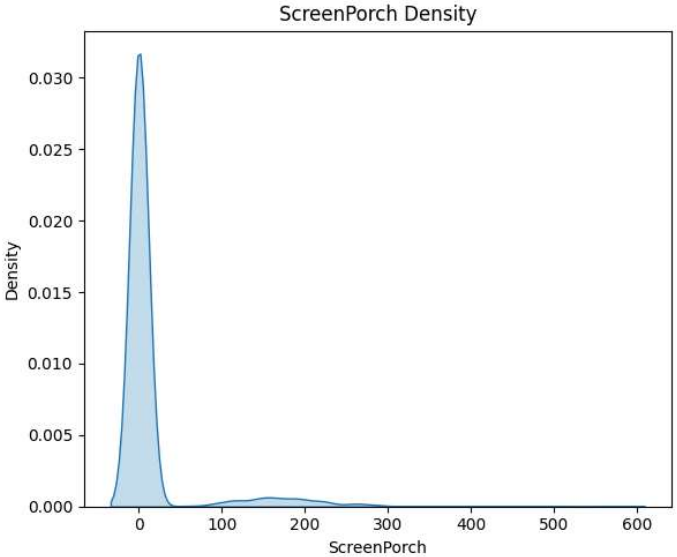


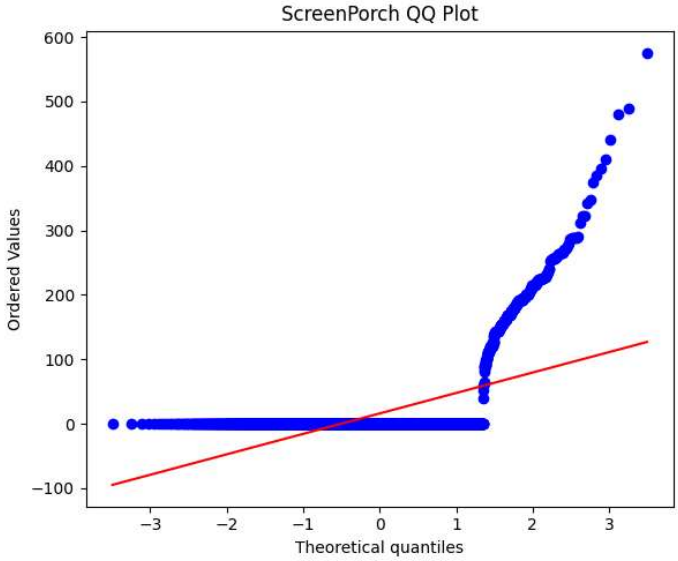




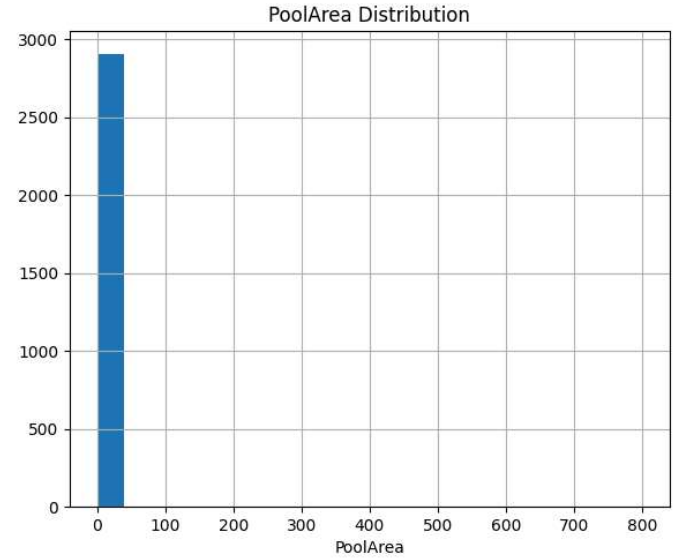
```
ScreenPorch
##########################################
count   2919.000
mean      16.062
std       56.184
min        0.000
5%         0.000
10%        0.000
20%        0.000
30%        0.000
40%        0.000
50%        0.000
60%        0.000
70%        0.000
80%        0.000
90%        0.000
95%      161.000
99%      259.820
max      576.000
Name: ScreenPorch, dtype: float64
##########################################
```

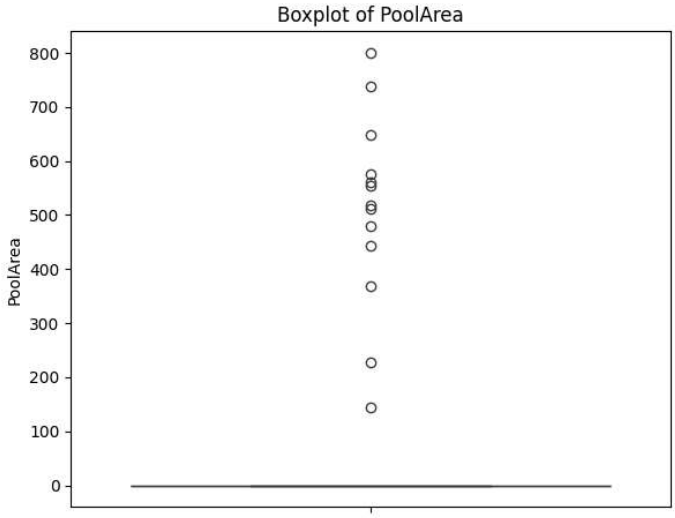


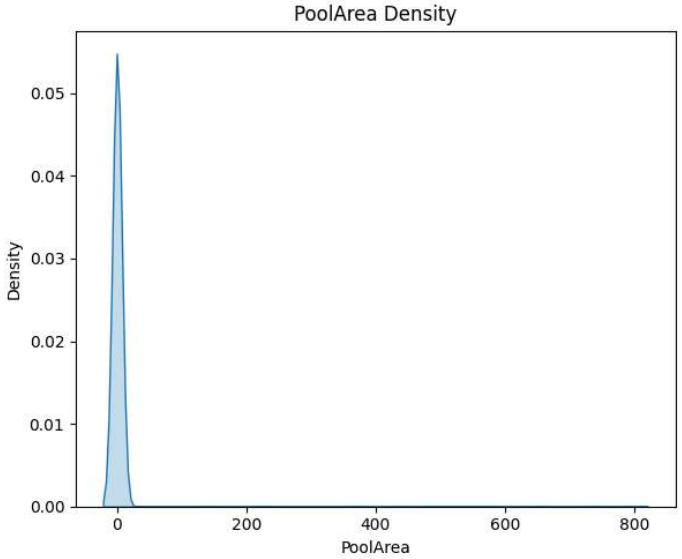


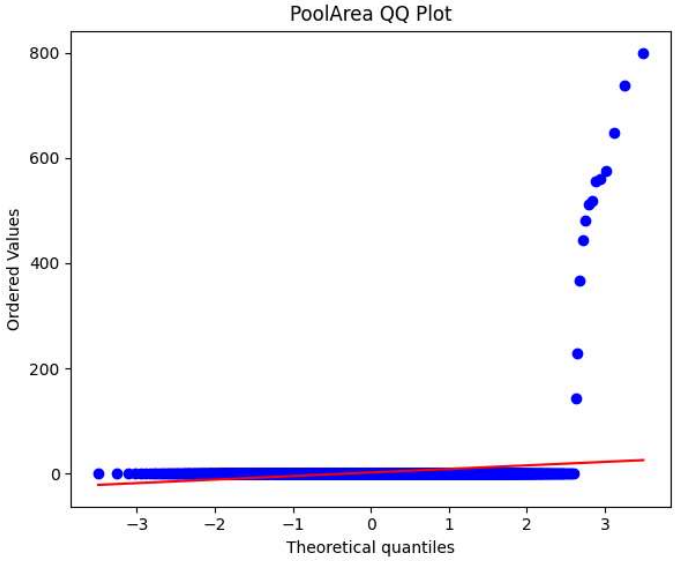


```
PoolArea
##########################################
count   2919.000
mean       2.252
std       35.664
min        0.000
5%         0.000
10%        0.000
20%        0.000
30%        0.000
40%        0.000
50%        0.000
60%        0.000
70%        0.000
80%        0.000
90%        0.000
95%        0.000
99%        0.000
max      800.000
Name: PoolArea, dtype: float64
##########################################
```

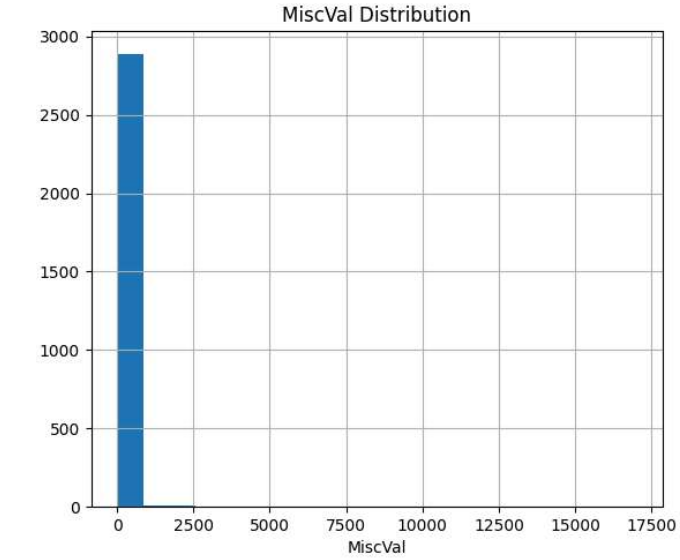


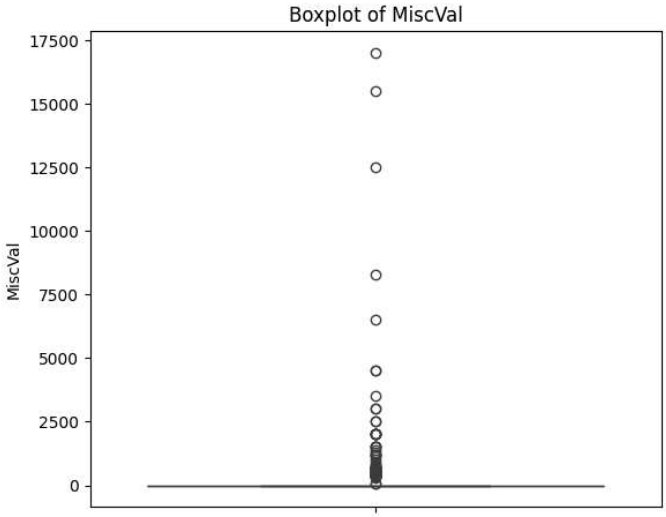


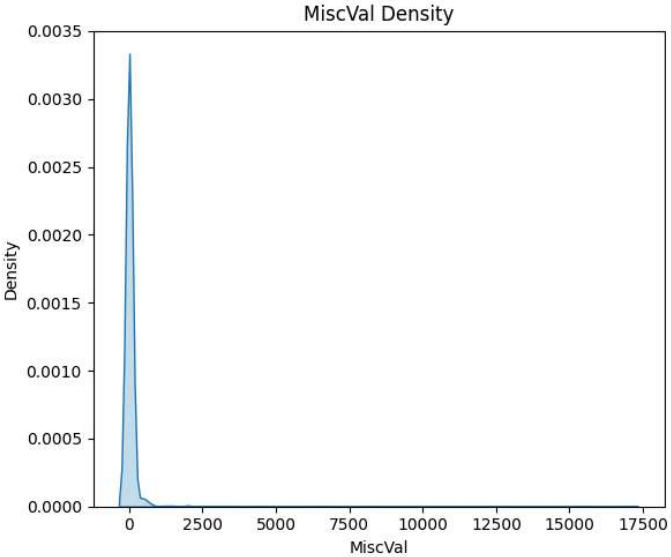


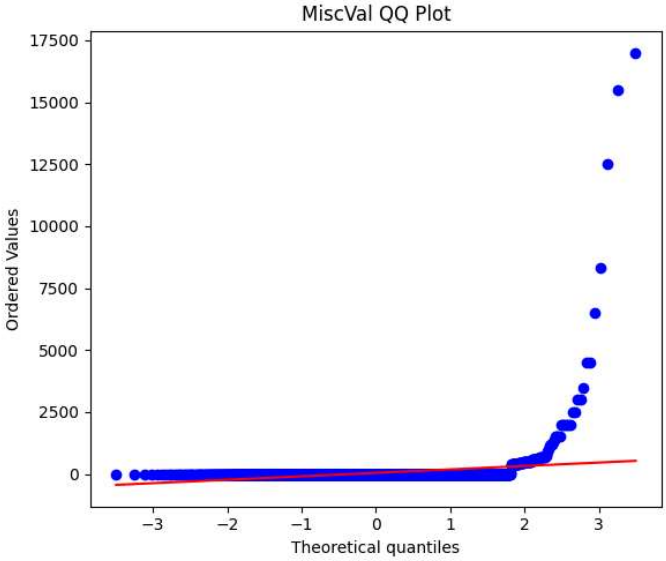


```
MiscVal
##########################################
count    2919.000
mean       50.826
std       567.402
min         0.000
5%          0.000
10%         0.000
20%         0.000
30%         0.000
40%         0.000
50%         0.000
60%         0.000
70%         0.000
80%         0.000
90%         0.000
95%         0.000
99%       982.000
max     17000.000
Name: MiscVal, dtype: float64
##########################################
```

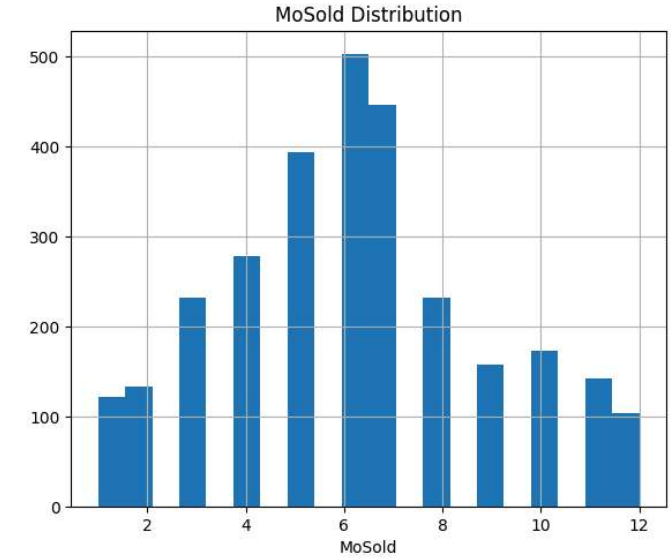


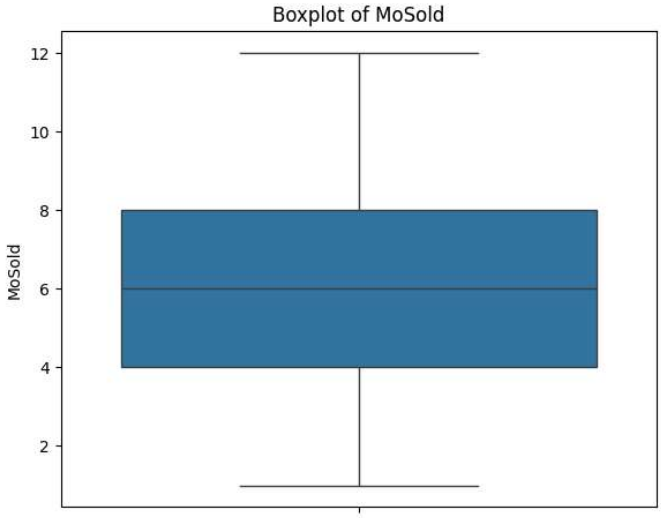


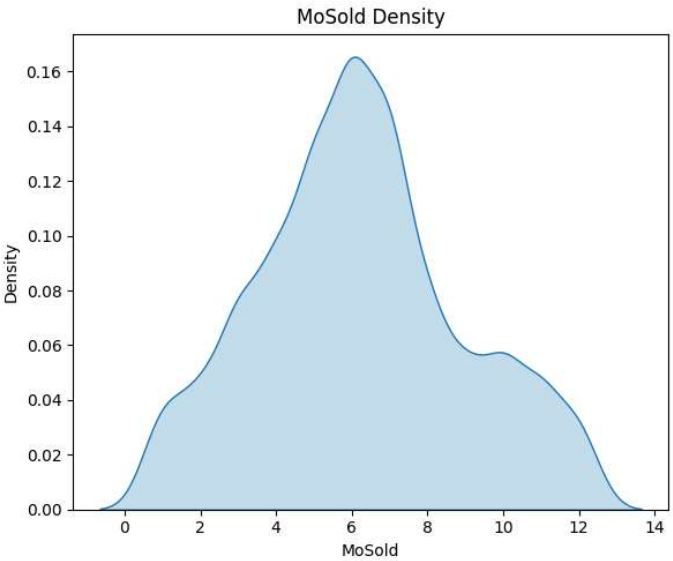


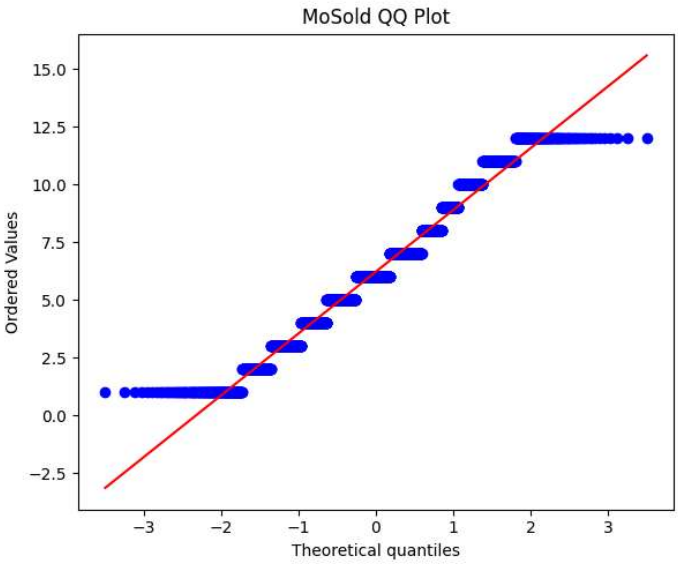


```
MoSold
##########################################
count   2919.000
mean       6.213
std        2.715
min        1.000
5%         2.000
10%        3.000
20%        4.000
30%        5.000
40%        6.000
50%        6.000
60%        7.000
70%        7.000
80%        8.000
90%       10.000
95%       11.000
99%       12.000
max       12.000
Name: MoSold, dtype: float64
##########################################
```







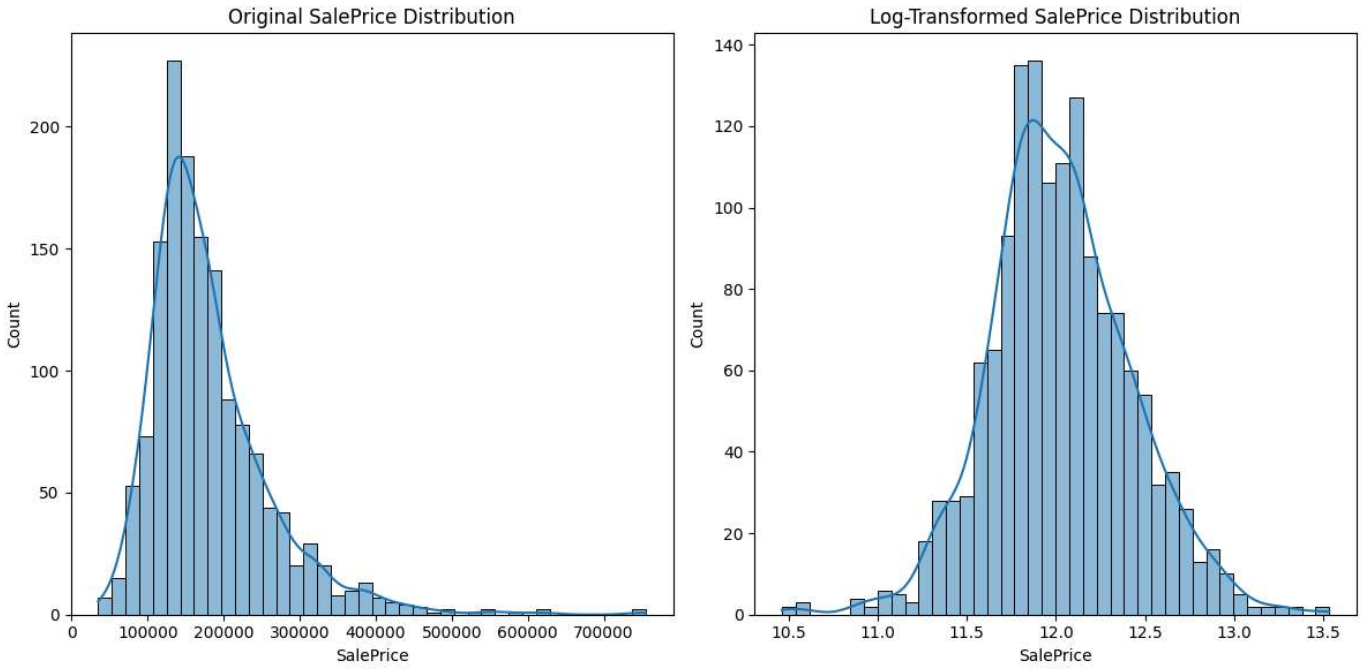


Target Summary

Target Distribution

In [ ]:
```
plt.figure(figsize=(12, 6))

plt.subplot(1, 2, 1)
sns.histplot(df["SalePrice"], bins=40, kde=True)
```

```
plt.title('Original SalePrice Distribution')

plt.subplot(1, 2, 2)
sns.histplot(np.log1p(df['SalePrice']), bins=40, kde=True)
plt.title('Log-Transformed SalePrice Distribution')

plt.tight_layout()
plt.show()
```



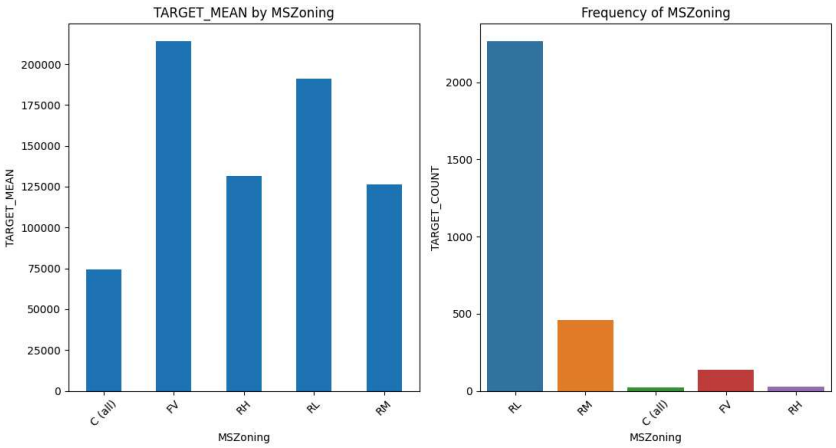
Cat Cols Target Summary

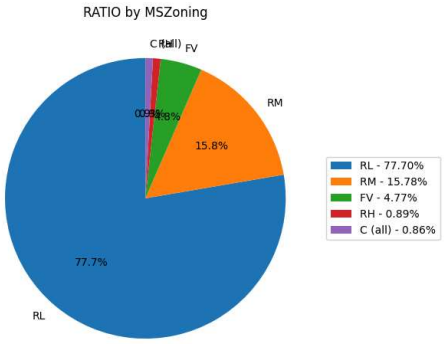
In [ ]:
```
def target_summary_with_cat(dataframe, target, categorical_col, plot=False):
    summary = pd.DataFrame({"TARGET_MEAN": dataframe.groupby(categorical_col)[target].mean(),
                            "TARGET_COUNT": dataframe.groupby(categorical_col)[target].count(
                            "RATIO": 100 * dataframe[categorical_col].value_counts() / len(da

    print(summary, end="\n\n\n")
    print("##########################################")

    if plot:
        fig, (ax1, ax2, ax3) = plt.subplots(1, 3, figsize=(18, 6))

        # TARGET_MEAN
        summary["TARGET_MEAN"].plot(kind="bar", ax=ax1)
        ax1.set_title(f"TARGET_MEAN by {categorical_col}")
        ax1.set_ylabel("TARGET_MEAN")
        ax1.tick_params(axis="x", rotation=45)

        # TARGET_COUNT
        sns.countplot(x=categorical_col, data=dataframe, ax=ax2)
        ax2.set_title(f"Frequency of {categorical_col}")
        ax2.set_ylabel("TARGET_COUNT")
        ax2.tick_params(axis="x", rotation=45)

        # RATIO
        values = dataframe[categorical_col].value_counts()
        ax3.pie(x=values, labels=values.index, autopct="%1.1f%%", startangle=90)
        ax3.set_title(f"RATIO by {categorical_col}")
        ax3.legend(labels=[f"{index} - {value/sum(values)*100:.2f}%" for index, value in zip(
                   loc="center left", bbox_to_anchor=(1, 0, 0.5, 1))

        plt.tight_layout()
        plt.show(block=True)
```

In [ ]:
```
for col in cat_cols:
    target_summary_with_cat(df, "SalePrice", col, plot=True)
```

```
            TARGET_MEAN  TARGET_COUNT  RATIO
MSZoning
C (all)       74528.000            10  0.856
FV           214014.062            65  4.762
RH           131558.375            16  0.891
RL           191004.995          1151 77.595
RM           126316.830           218 15.759
```

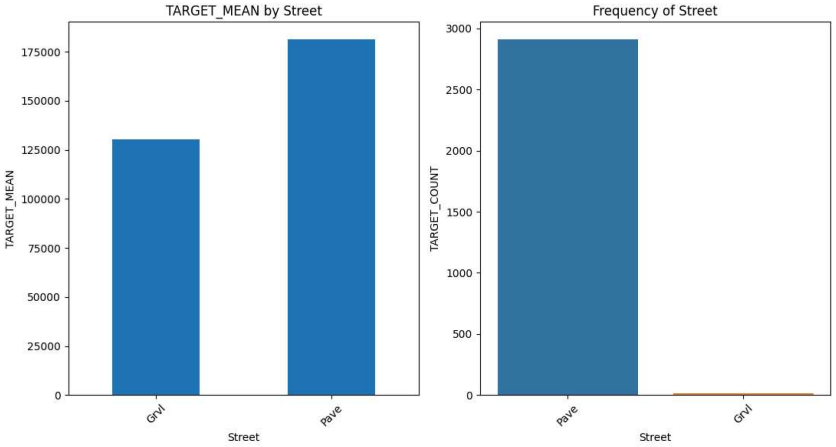
```
##########################################
```

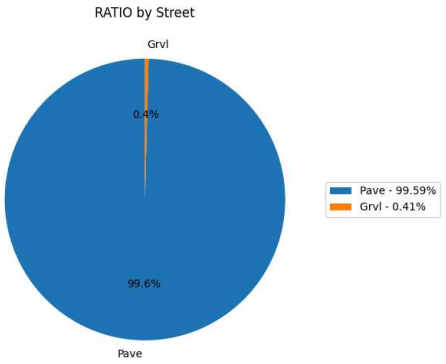





```
          TARGET_MEAN  TARGET_COUNT  RATIO
Street
Grvl       130190.500             6  0.411
Pave       181130.539          1454 99.589
```

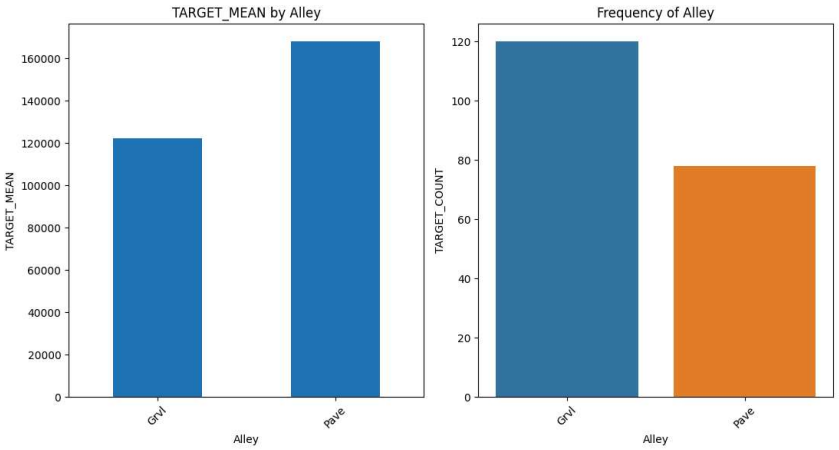
```
##########################################
```

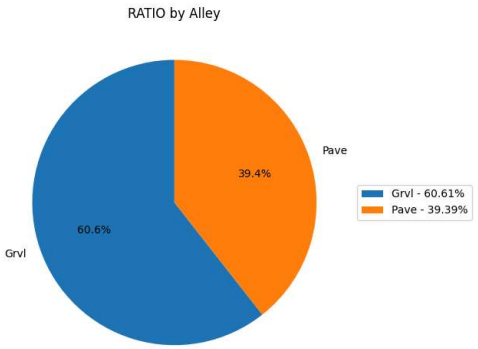





```
        TARGET_MEAN  TARGET_COUNT  RATIO
Alley
Grvl     122219.080            50  4.111
Pave     168000.585            41  2.672
```

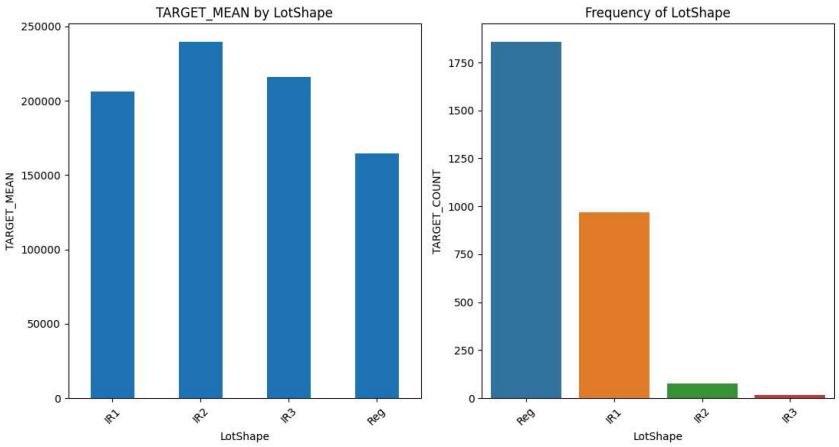
```
##########################################
```

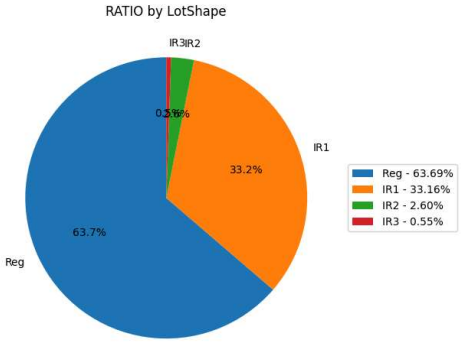




	TARGET_MEAN	TARGET_COUNT	RATIO
LotShape			
IR1	206101.665	484	33.162
IR2	239833.366	41	2.604
IR3	216036.500	10	0.548
Reg	164754.818	925	63.686

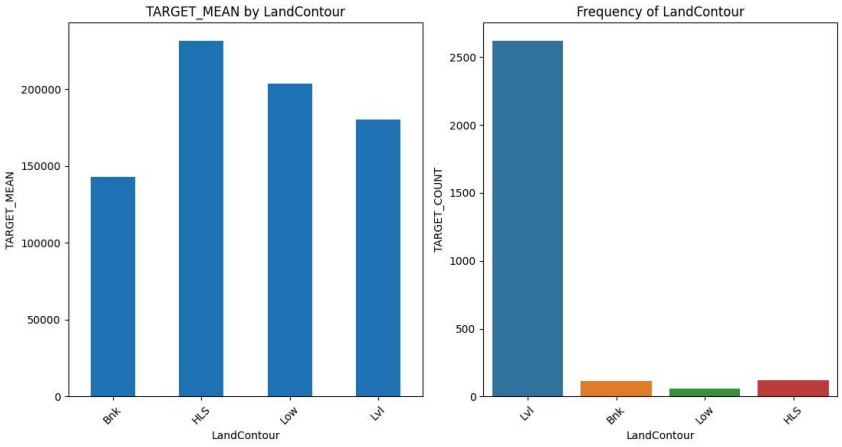
##

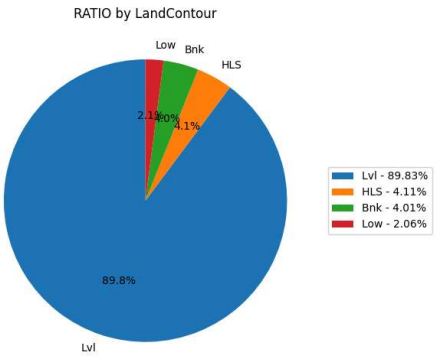




	TARGET_MEAN	TARGET_COUNT	RATIO
LandContour			
Bnk	143104.079	63	4.008
HLS	231533.940	50	4.111
Low	203661.111	36	2.055
Lvl	180183.747	1311	89.825

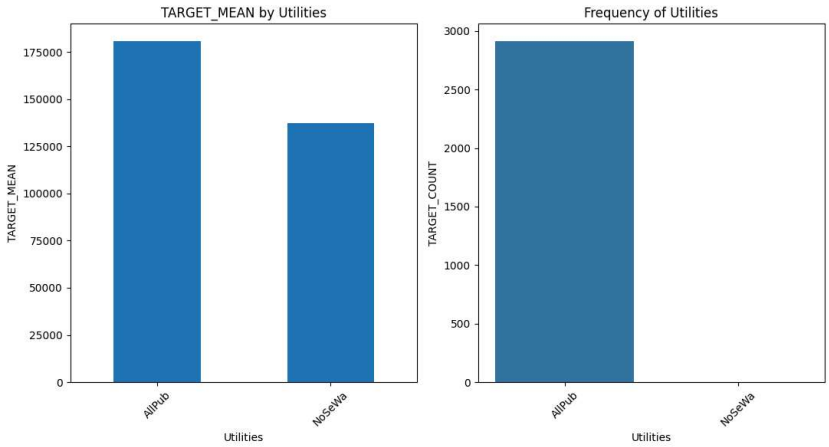
##

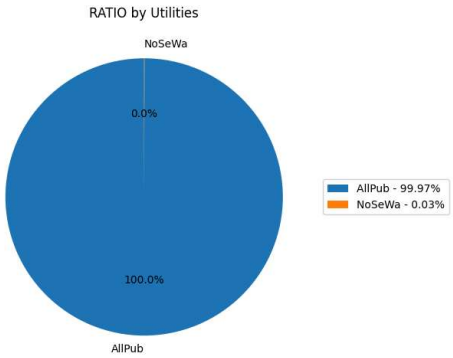




	TARGET_MEAN	TARGET_COUNT	RATIO
Utilities			
AllPub	180950.957	1459	99.897
NoSeWa	137500.000	1	0.034

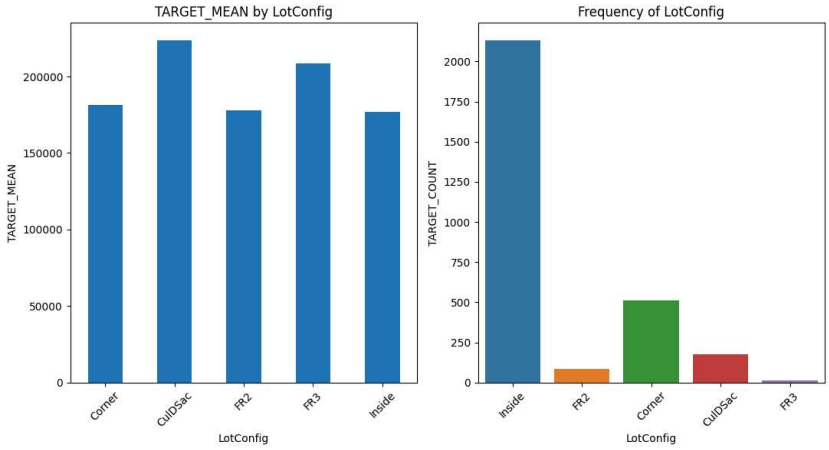
##

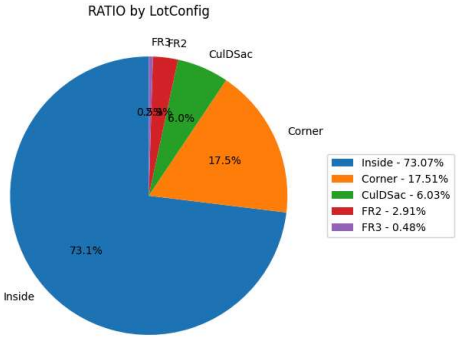




LotConfig	TARGET_MEAN	TARGET_COUNT	RATIO
Corner	181623.426	263	17.506
CulDSac	223854.617	94	6.029
FR2	177934.574	47	2.912
FR3	208475.000	4	0.480
Inside	176938.048	1052	73.073

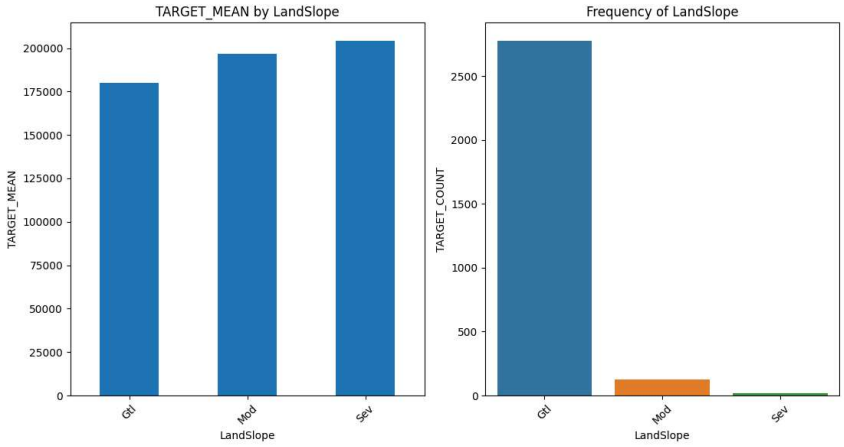
##

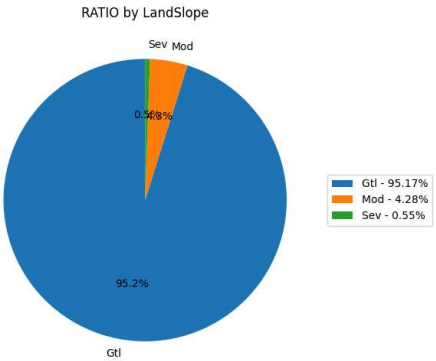




LandSlope	TARGET_MEAN	TARGET_COUNT	RATIO
Gtl	179956.800	1382	95.170
Mod	196734.138	65	4.282
Sev	204379.231	13	0.548

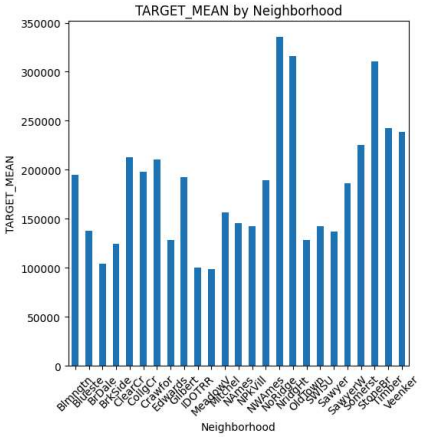
##

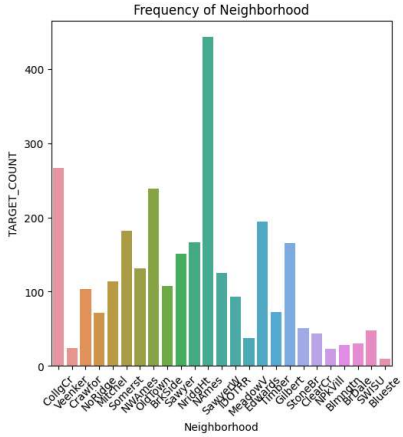


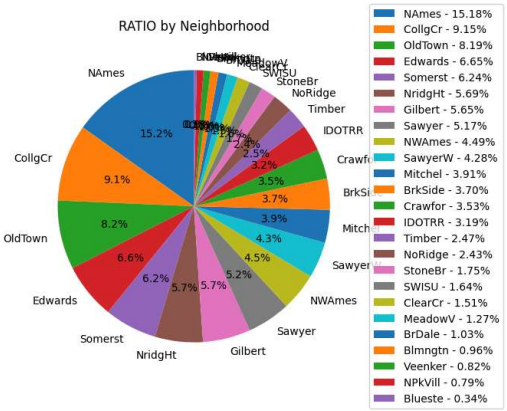


Neighborhood	TARGET_MEAN	TARGET_COUNT	RATIO
Blmngtn	194870.882	17	0.959
Blueste	137500.000	2	0.343
BrDale	104493.750	16	1.028
BrkSide	124834.052	58	3.700
ClearCr	212565.429	28	1.507
CollgCr	197965.773	150	9.147
Crawfor	210624.725	51	3.529
Edwards	128219.700	100	6.646
Gilbert	192854.506	79	5.653
IDOTRR	100123.784	37	3.186
MeadowV	98576.471	17	1.268
Mitchel	156270.122	49	3.905
NAmes	145847.080	225	15.176
NPkVill	142694.444	9	0.788
NWAmes	189050.068	73	4.488
NoRidge	335295.317	41	2.432
NridgHt	316270.623	77	5.687
OldTown	128225.301	113	8.188
SWISU	142591.360	25	1.644
Sawyer	136793.135	74	5.173
SawyerW	186555.797	59	4.282
Somerst	225379.837	86	6.235
StoneBr	310499.000	25	1.747
Timber	242247.447	38	2.467
Veenker	238772.727	11	0.822

##

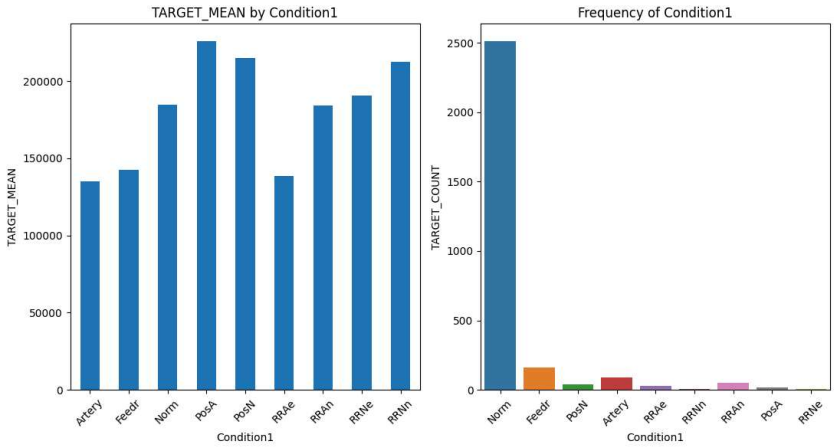


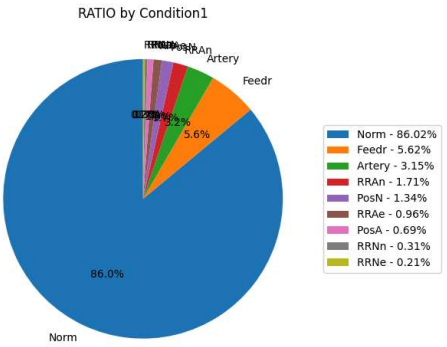




Condition1	TARGET_MEAN	TARGET_COUNT	RATIO
Artery	135091.667	48	3.152
Feedr	142475.481	81	5.618
Norm	184495.492	1260	86.023
PosA	225875.000	8	0.685
PosN	215184.211	19	1.336
RRAe	138400.000	11	0.959
RRAn	184396.615	26	1.713
RRNe	190750.000	2	0.206
RRNn	212400.000	5	0.308

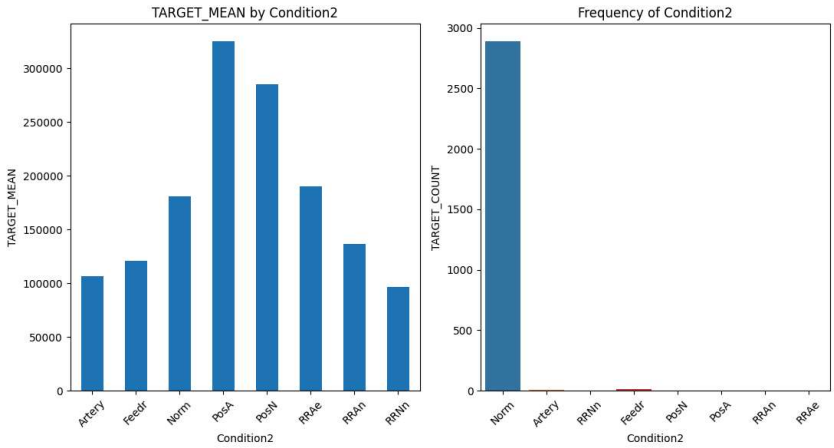
##

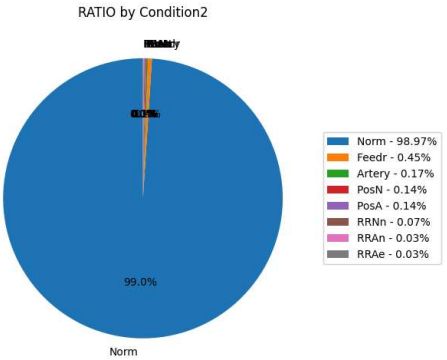




Condition2	TARGET_MEAN	TARGET_COUNT	RATIO
Artery	106500.000	2	0.171
Feedr	121166.667	6	0.445
Norm	181169.406	1445	98.972
PosA	325000.000	1	0.137
PosN	284875.000	2	0.137
RRAe	190000.000	1	0.034
RRAn	136905.000	1	0.034
RRNn	96750.000	2	0.069

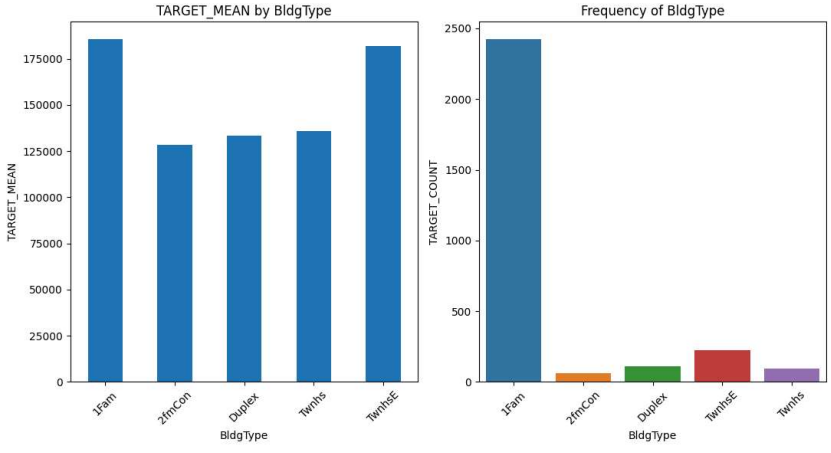
##

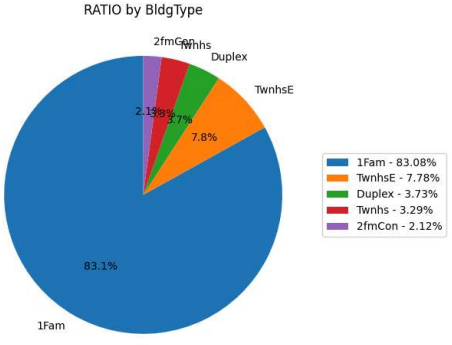




BldgType	TARGET_MEAN	TARGET_COUNT	RATIO
1Fam	185763.807	1220	83.076
2fmCon	128432.258	31	2.124
Duplex	133541.077	52	3.734
Twnhs	135911.628	43	3.289
TwnhsE	181959.342	114	7.777

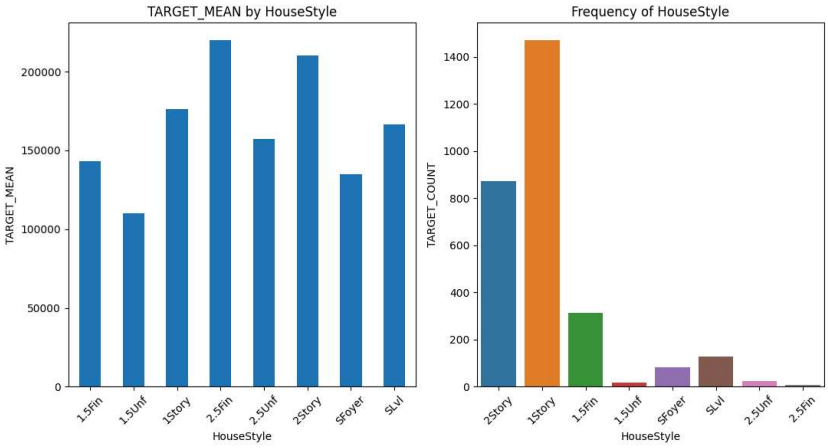
##

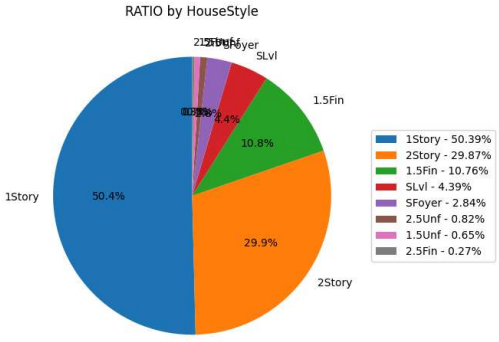




HouseStyle	TARGET_MEAN	TARGET_COUNT	RATIO
1.5Fin	143116.740	154	10.757
1.5Unf	110150.000	14	0.651
1Story	175985.478	726	50.394
2.5Fin	220000.000	8	0.274
2.5Unf	157354.545	11	0.822
2Story	210051.764	445	29.873
SFoyer	135074.486	37	2.843
SLvl	166703.385	65	4.385

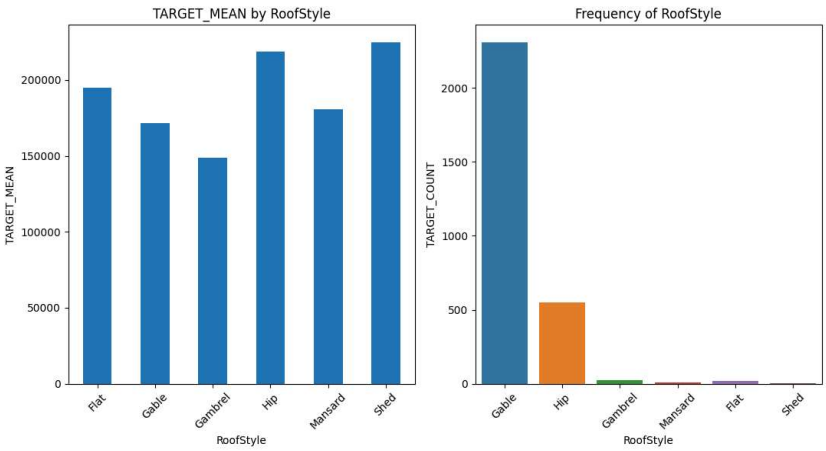
##

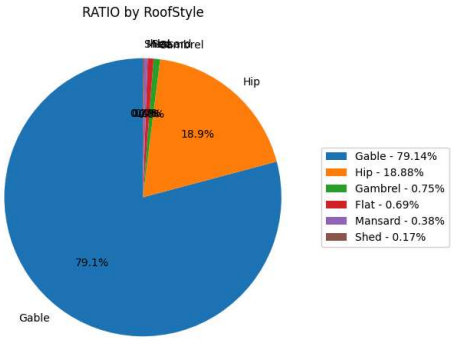




RoofStyle	TARGET_MEAN	TARGET_COUNT	RATIO
Flat	194690.000	13	0.685
Gable	171483.956	1141	79.137
Gambrel	148909.091	11	0.754
Hip	218876.934	286	18.876
Mansard	180568.429	7	0.377
Shed	225000.000	2	0.171

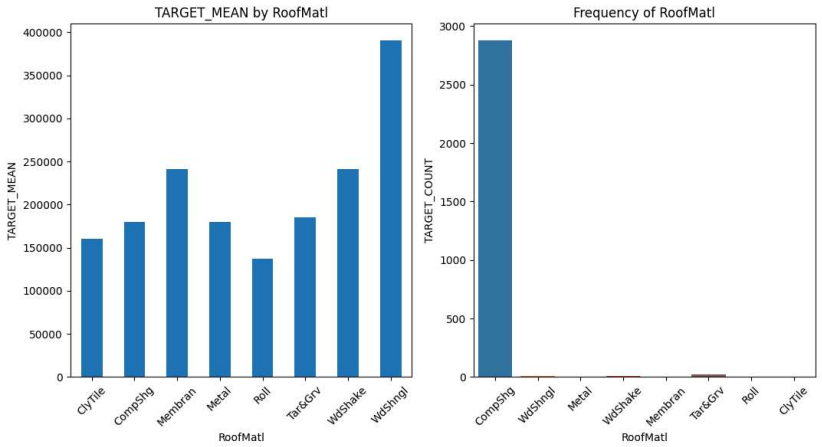
##

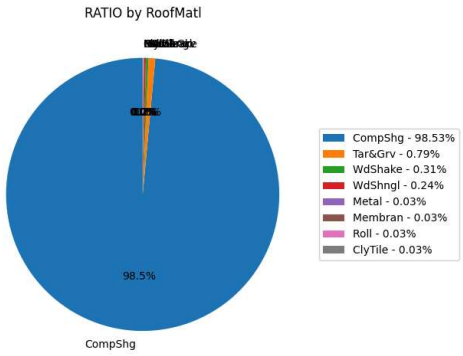




RoofMatl	TARGET_MEAN	TARGET_COUNT	RATIO
ClyTile	160000.000	1	0.034
CompShg	179803.679	1434	98.527
Membran	241500.000	1	0.034
Metal	180000.000	1	0.034
Roll	137000.000	1	0.034
Tar&Grv	185406.364	11	0.788
WdShake	241400.000	5	0.308
WdShngl	390250.000	6	0.240

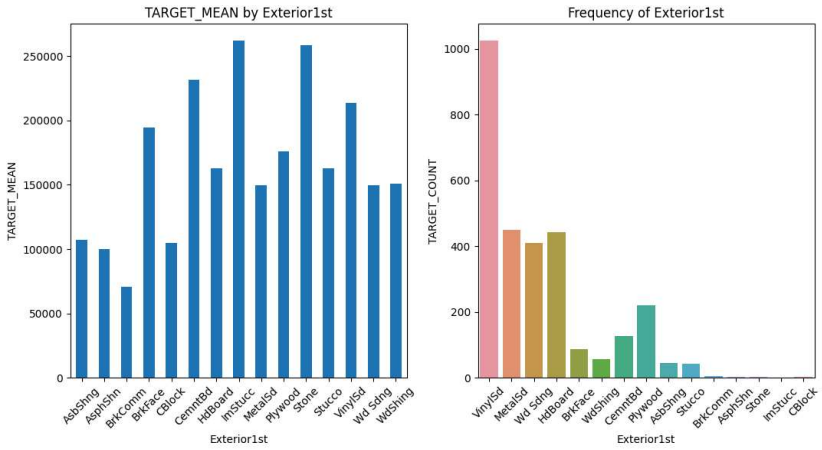
##

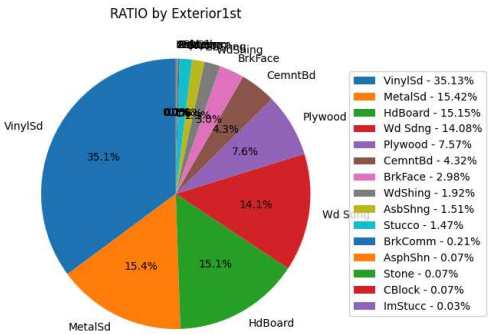




Exterior1st	TARGET_MEAN	TARGET_COUNT	RATIO
AsbShng	107385.550	20	1.507
AsphShn	100000.000	1	0.069
BrkComm	71000.000	2	0.206
BrkFace	194573.000	50	2.980
CBlock	105000.000	1	0.069
CemntBd	231690.656	61	4.317
HdBoard	163077.450	222	15.142
ImStucc	262000.000	1	0.034
MetalSd	149422.177	220	15.416
Plywood	175942.380	108	7.571
Stone	258500.000	2	0.069
Stucco	162990.000	25	1.473
VinylSd	213732.901	515	35.115
Wd Sdng	149841.646	206	14.080
WdShing	150655.077	26	1.918

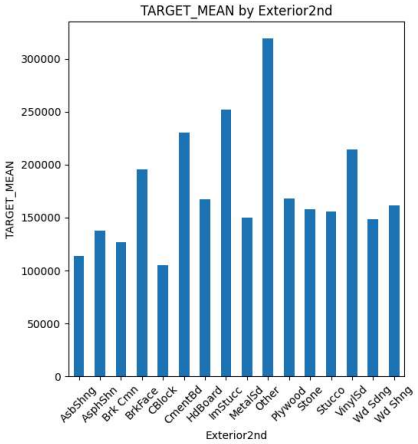
##

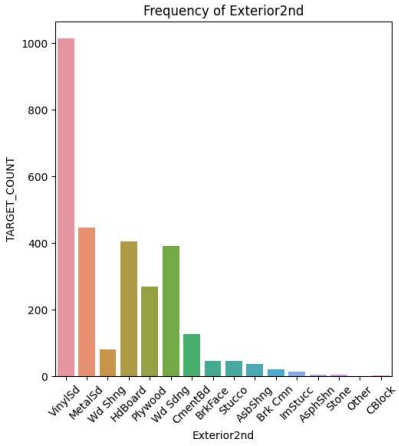


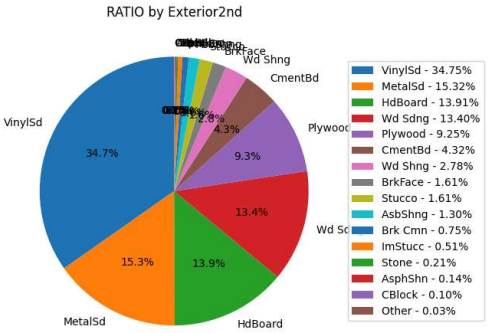


	TARGET_MEAN	TARGET_COUNT	RATIO
Exterior2nd			
AsbShng	114060.550	20	1.302
AsphShn	138000.000	3	0.137
Brk Cmn	126714.286	7	0.754
BrkFace	195818.000	25	1.610
CBlock	105000.000	1	0.103
CmentBd	230093.833	60	4.317
HdBoard	167661.565	207	13.909
ImStucc	252070.000	10	0.514
MetalSd	149803.173	214	15.313
Other	319000.000	1	0.034
Plywood	168112.387	142	9.250
Stone	158224.800	5	0.206
Stucco	155905.154	26	1.610
VinylSd	214432.460	504	34.738
Wd Sdng	148386.066	197	13.395
Wd Shng	161328.947	38	2.775

##

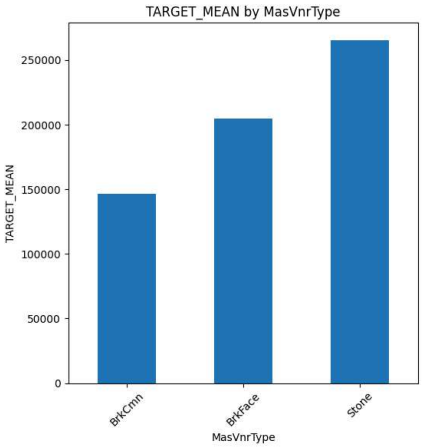


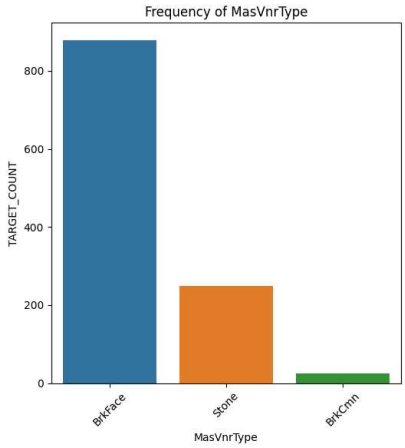


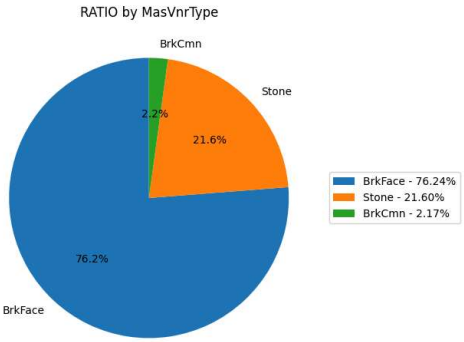


	TARGET_MEAN	TARGET_COUNT	RATIO
MasVnrType			
BrkCmn	146318.067	15	0.856
BrkFace	204691.872	445	30.113
Stone	265583.625	128	8.530

##

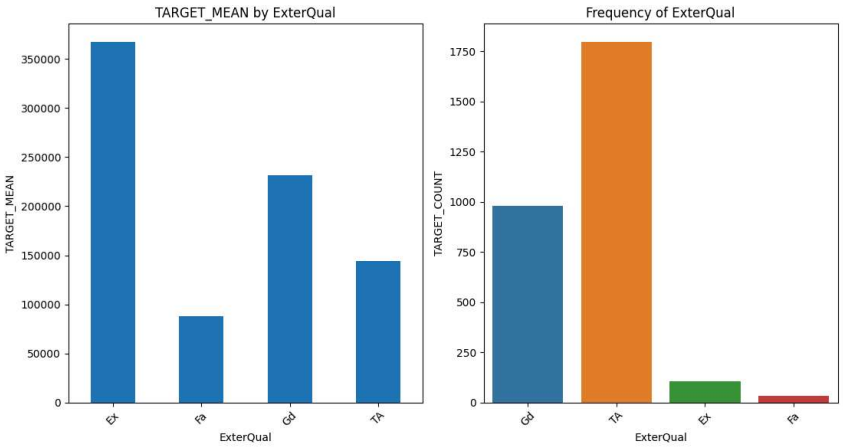


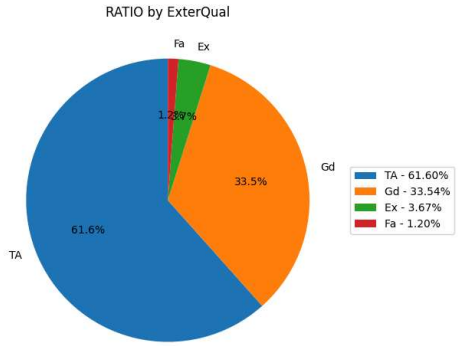




	TARGET_MEAN	TARGET_COUNT	RATIO
ExterQual			
Ex	367360.962	52	3.666
Fa	87985.214	14	1.199
Gd	231633.510	488	33.539
TA	144341.313	906	61.596

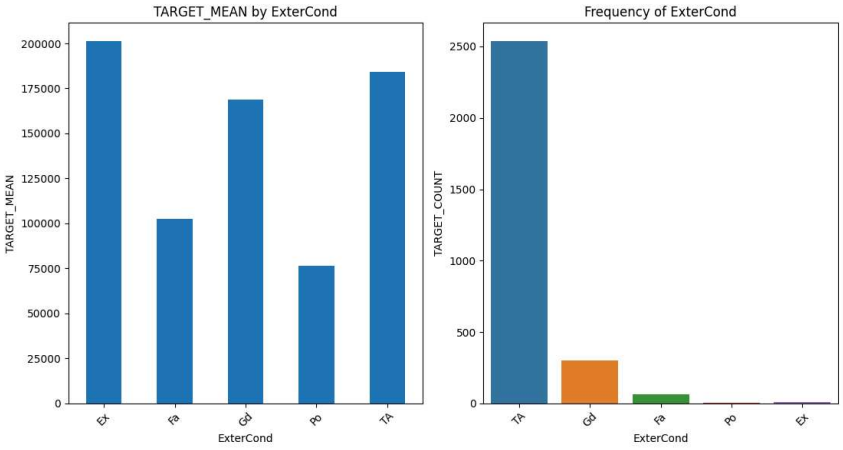
##

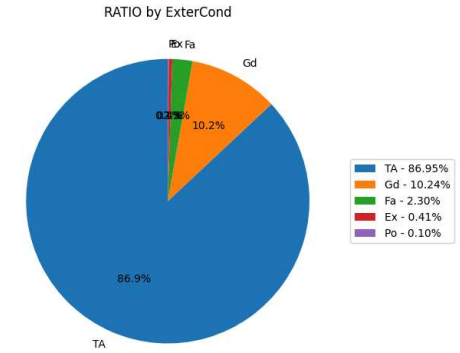




ExterCond	TARGET_MEAN	TARGET_COUNT	RATIO
Ex	201333.333	3	0.411
Fa	102595.143	28	2.295
Gd	168897.568	146	10.243
Po	76500.000	1	0.103
TA	184034.896	1282	86.948

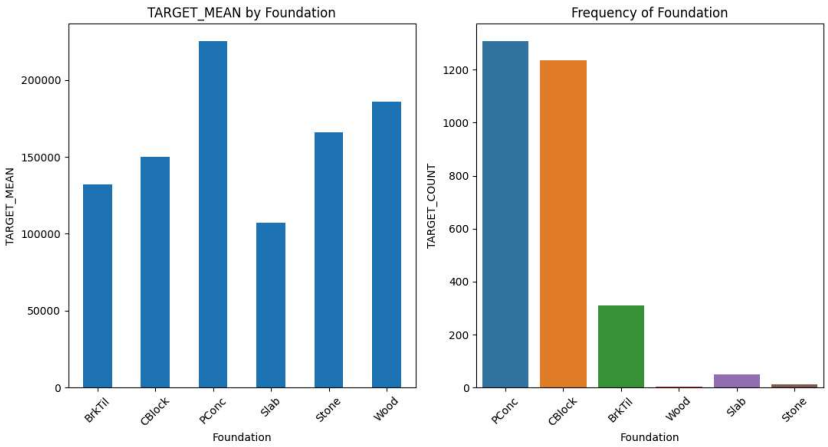
##

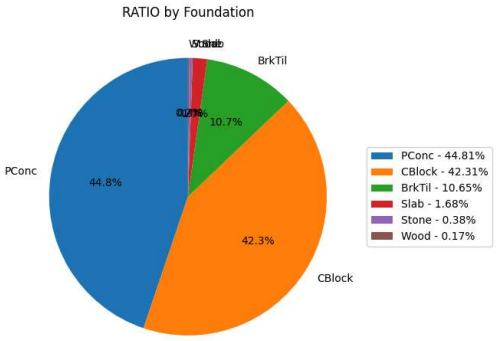




Foundation	TARGET_MEAN	TARGET_COUNT	RATIO
BrkTil	132291.075	146	10.654
CBlock	149805.715	634	42.309
PConc	225230.442	647	44.810
Slab	107365.625	24	1.679
Stone	165959.167	6	0.377
Wood	185666.667	3	0.171

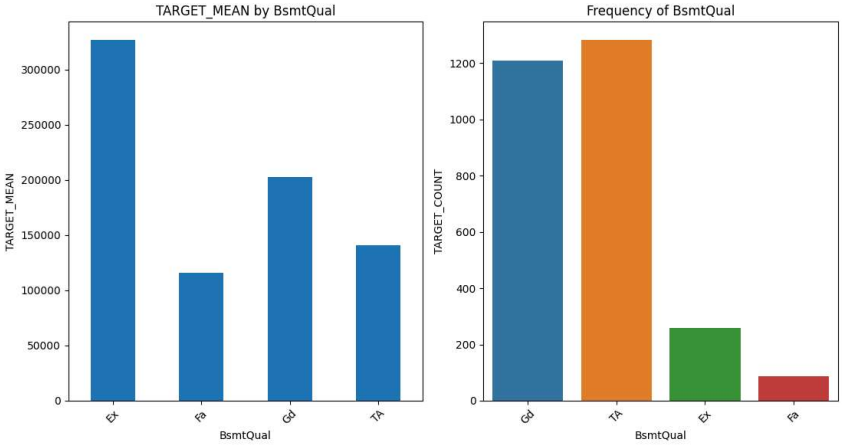
##

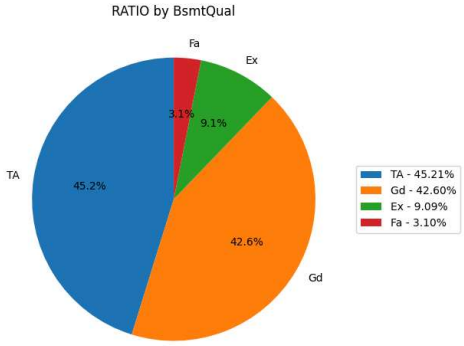




	TARGET_MEAN	TARGET_COUNT	RATIO
BsmtQual			
Ex	327041.041	121	8.839
Fa	115692.029	35	3.015
Gd	202688.479	618	41.418
TA	140759.818	649	43.953

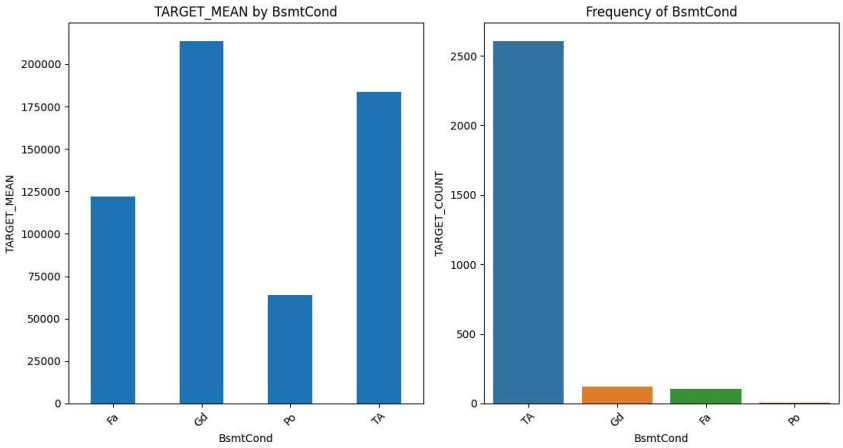
##

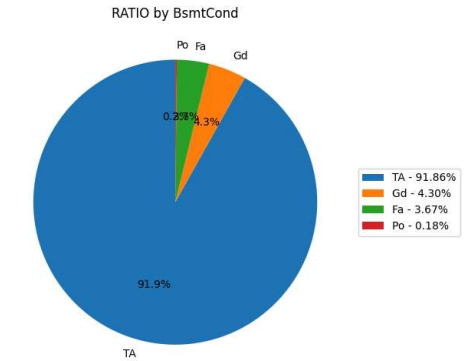




	TARGET_MEAN	TARGET_COUNT	RATIO
BsmtCond			
Fa	121809.533	45	3.563
Gd	213599.908	65	4.180
Po	64000.000	2	0.171
TA	183632.621	1311	89.277

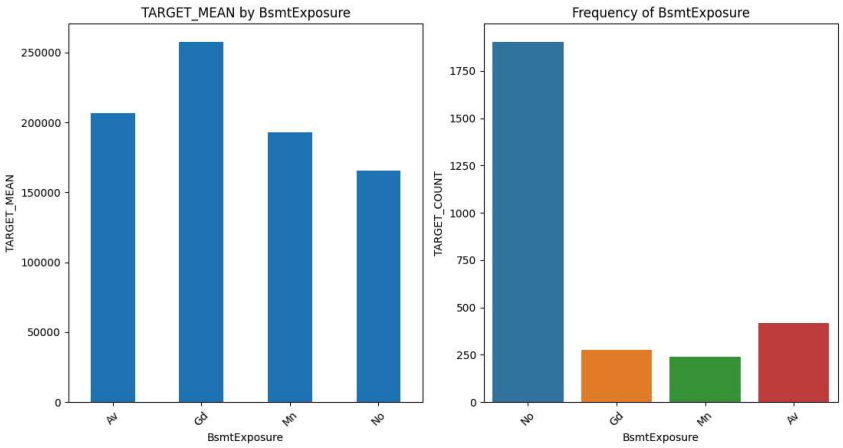
##

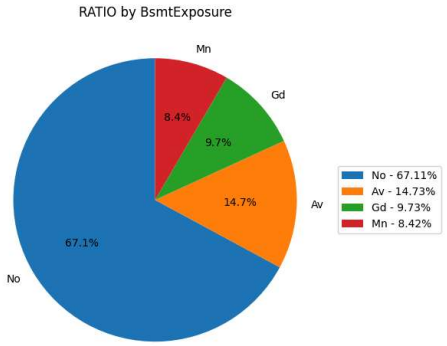




	TARGET_MEAN	TARGET_COUNT	RATIO
BsmtExposure			
Av	206643.421	221	14.320
Gd	257689.806	134	9.455
Mn	192789.658	114	8.188
No	165652.296	953	65.228

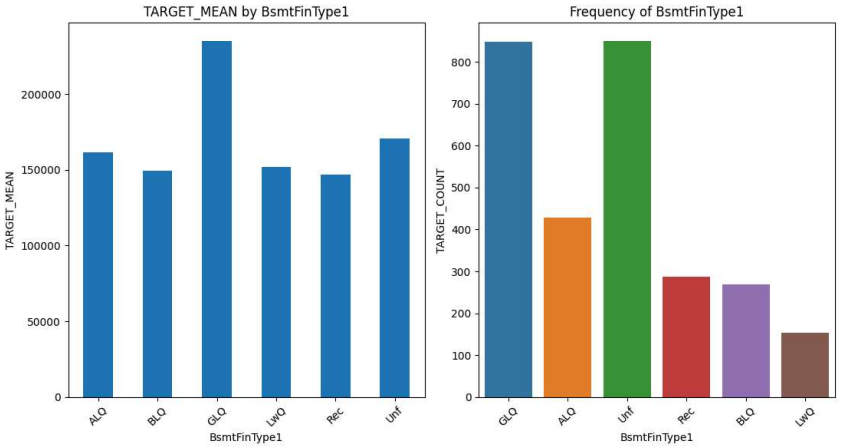
##

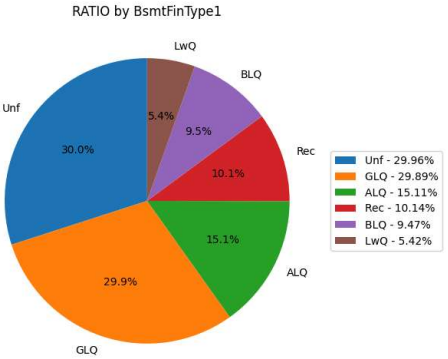




	TARGET_MEAN	TARGET_COUNT	RATIO
BsmtFinType1			
ALQ	161573.068	220	14.697
BLQ	149493.655	148	9.215
GLQ	235413.720	418	29.085
LwQ	151852.703	74	5.276
Rec	146889.248	133	9.866
Unf	170670.577	430	29.154

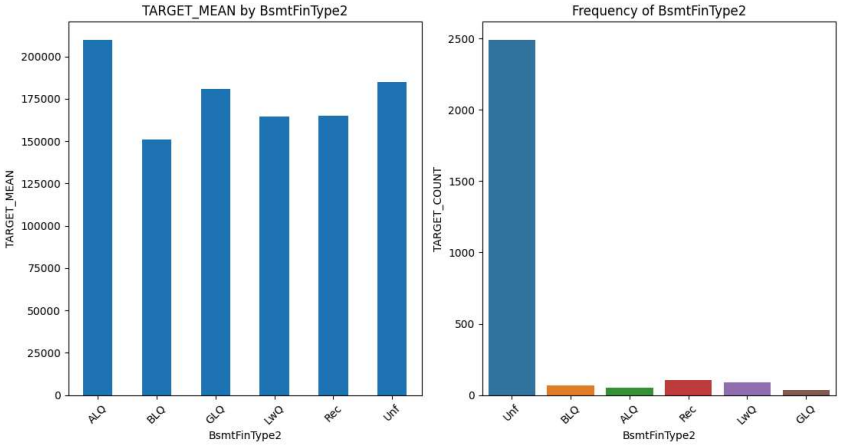
##

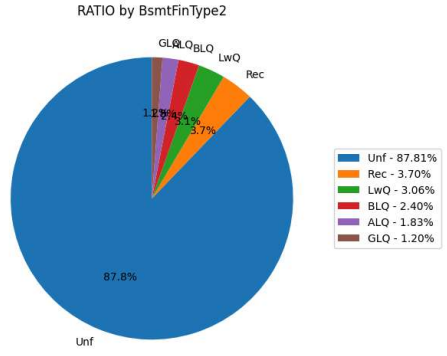




	TARGET_MEAN	TARGET_COUNT	RATIO
BsmtFinType2			
ALQ	209942.105	19	1.781
BLQ	151101.000	33	2.330
GLQ	180982.143	14	1.165
LwQ	164364.130	46	2.980
Rec	164917.130	54	3.597
Unf	184694.690	1256	85.406

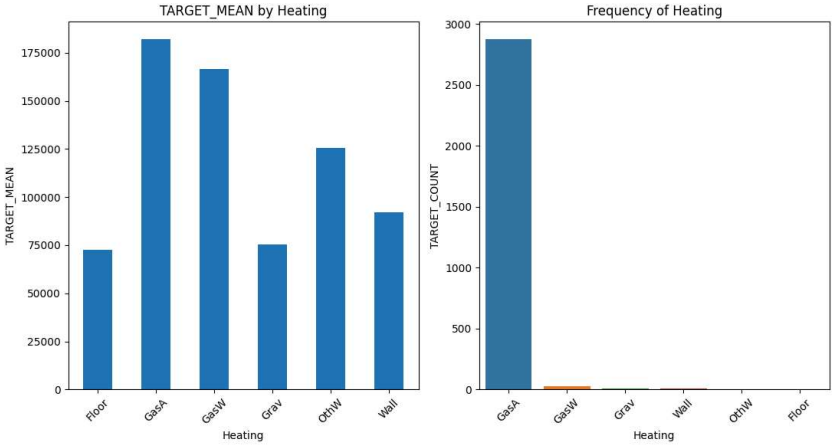
##

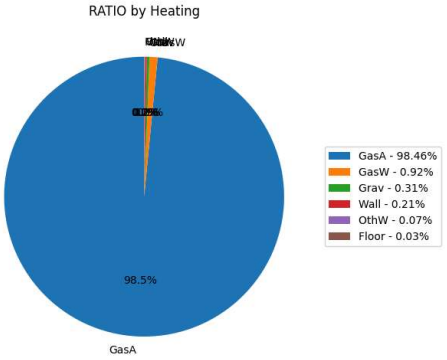




Heating	TARGET_MEAN	TARGET_COUNT	RATIO
Floor	72500.000	1	0.034
GasA	182021.195	1428	98.458
GasW	166632.167	18	0.925
Grav	75271.429	7	0.308
OthW	125750.000	2	0.069
Wall	92100.000	4	0.206

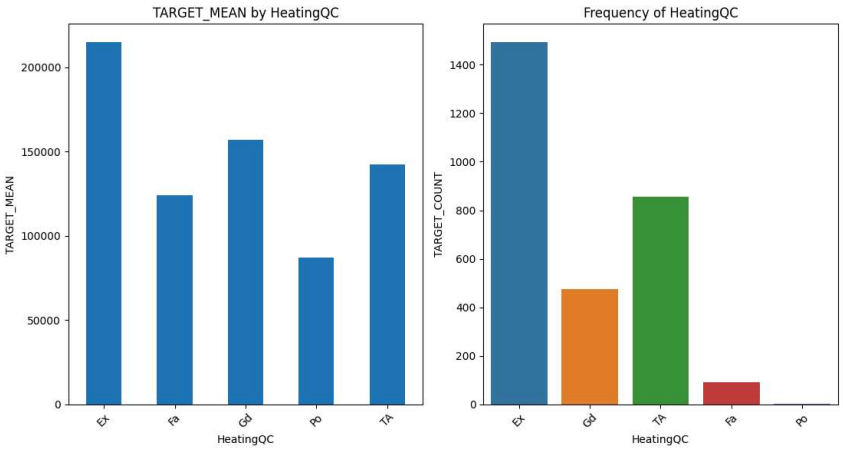
##

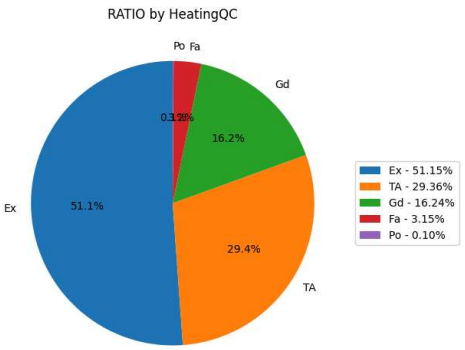




HeatingQC	TARGET_MEAN	TARGET_COUNT	RATIO
Ex	214914.429	741	51.148
Fa	123919.490	49	3.152
Gd	156858.871	241	16.238
Po	87000.000	1	0.103
TA	142362.876	428	29.359

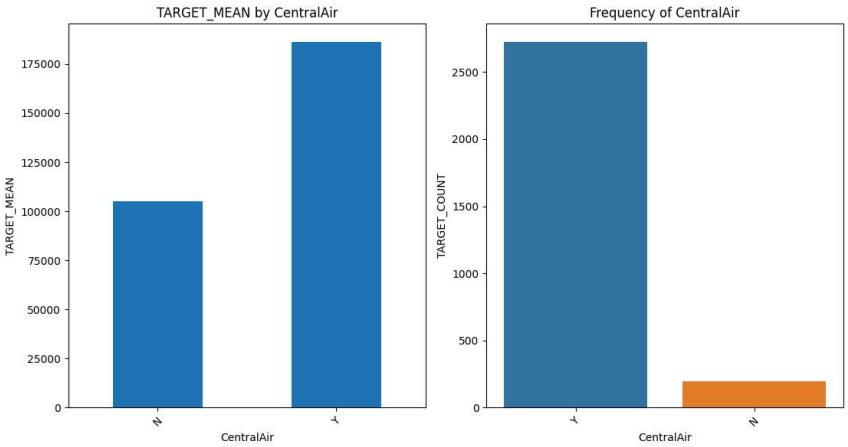
##

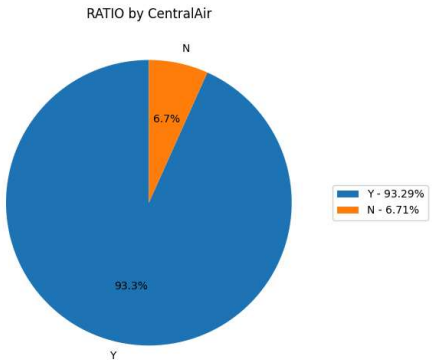




CentralAir	TARGET_MEAN	TARGET_COUNT	RATIO
N	105264.074	95	6.715
Y	186186.710	1365	93.285

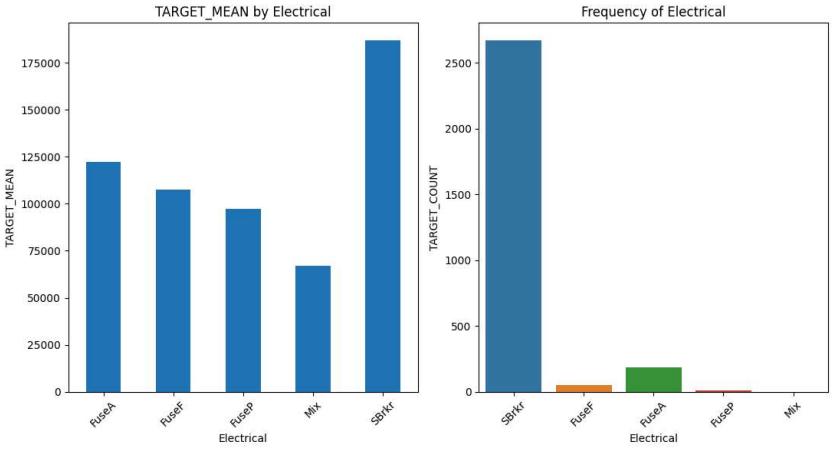
##

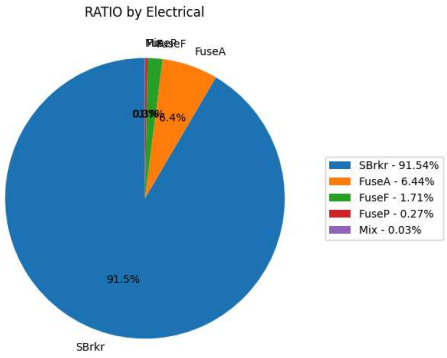




Electrical	TARGET_MEAN	TARGET_COUNT	RATIO
FuseA	122196.894	94	6.441
FuseF	107675.444	27	1.713
FuseP	97333.333	3	0.274
Mix	67000.000	1	0.034
SBrkr	186825.113	1334	91.504

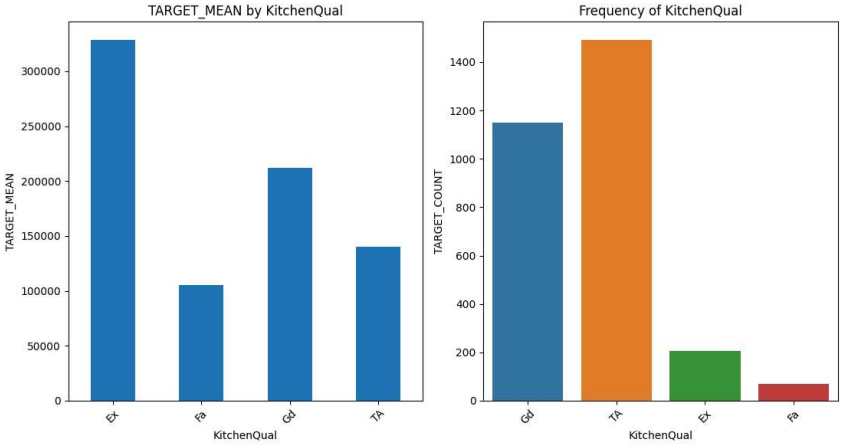
##

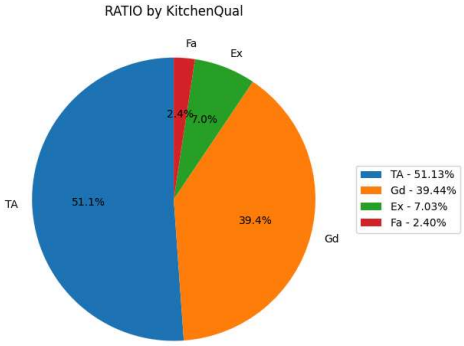




KitchenQual	TARGET_MEAN	TARGET_COUNT	RATIO
Ex	328554.670	100	7.023
Fa	105565.205	39	2.398
Gd	212116.024	586	39.431
TA	139962.512	735	51.113

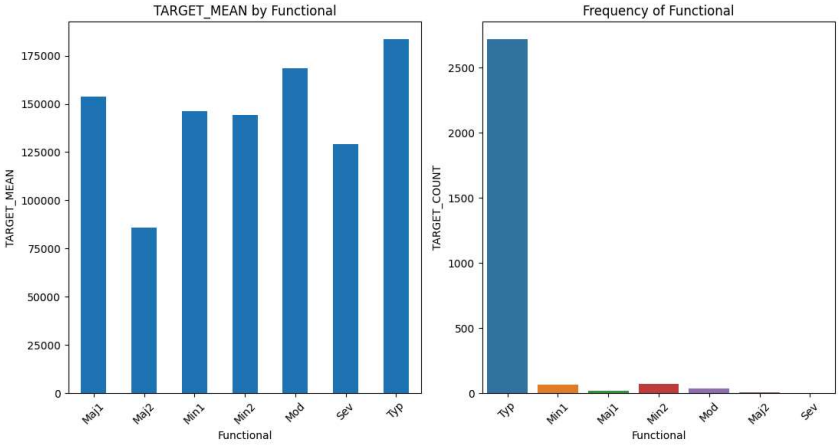
##

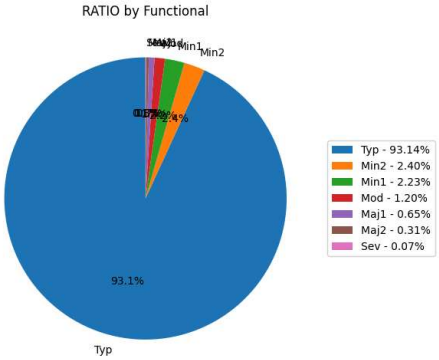




	TARGET_MEAN	TARGET_COUNT	RATIO
Functional			
Maj1	153948.143	14	0.651
Maj2	85800.000	5	0.308
Min1	146385.484	31	2.227
Min2	144240.647	34	2.398
Mod	168393.333	15	1.199
Sev	129000.000	1	0.069
Typ	183429.147	1360	93.080

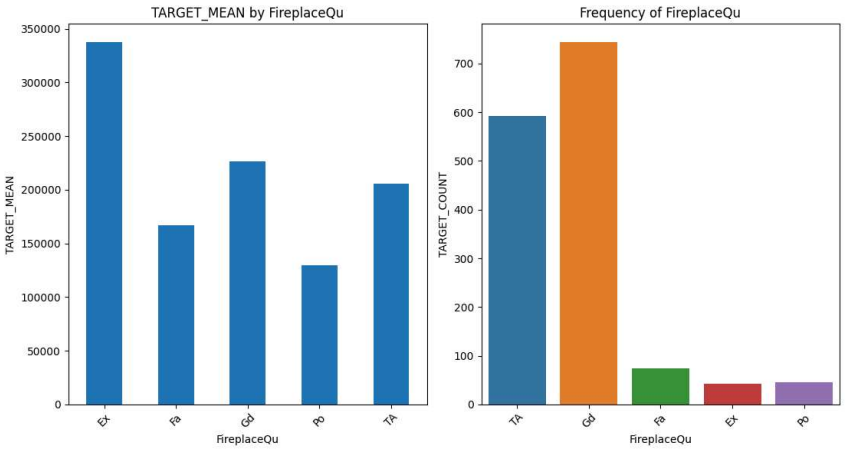
##

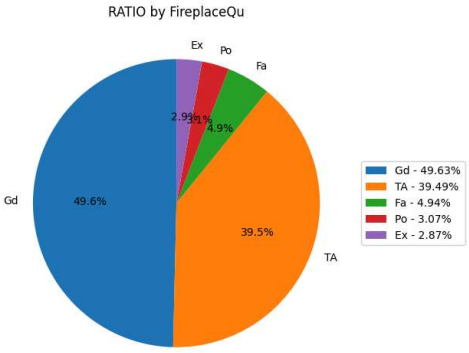




	TARGET_MEAN	TARGET_COUNT	RATIO
FireplaceQu			
Ex	337712.500	24	1.473
Fa	167298.485	33	2.535
Gd	226351.416	380	25.488
Po	129764.150	20	1.576
TA	205723.489	313	20.281

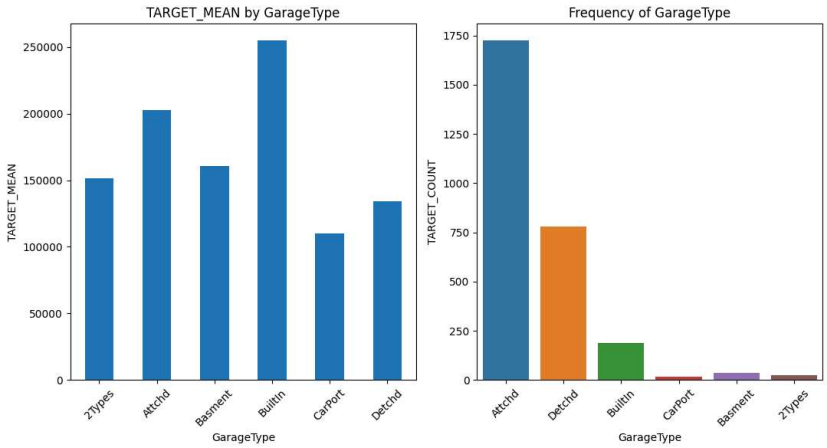
##

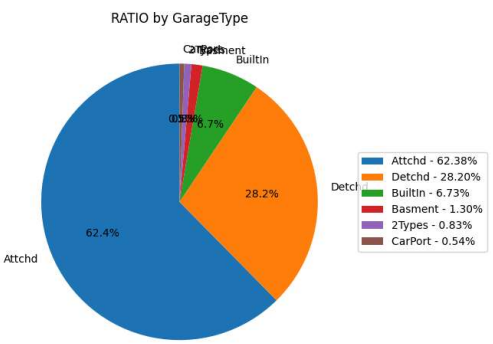




	TARGET_MEAN	TARGET_COUNT	RATIO
GarageType			
2Types	151283.333	6	0.788
Attchd	202892.656	870	59.027
Basment	160570.684	19	1.233
BuiltIn	254751.739	88	6.372
CarPort	109962.111	9	0.514
Detchd	134091.163	387	26.687

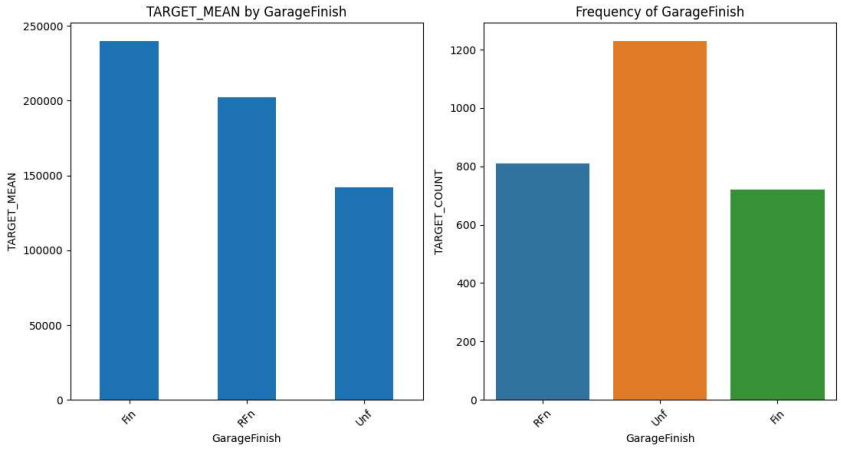
##

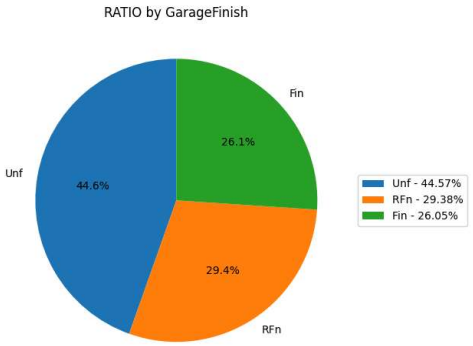




GarageFinish	TARGET_MEAN	TARGET_COUNT	RATIO
Fin	240052.690	352	24.632
RFn	202068.870	422	27.783
Unf	142156.423	605	42.138

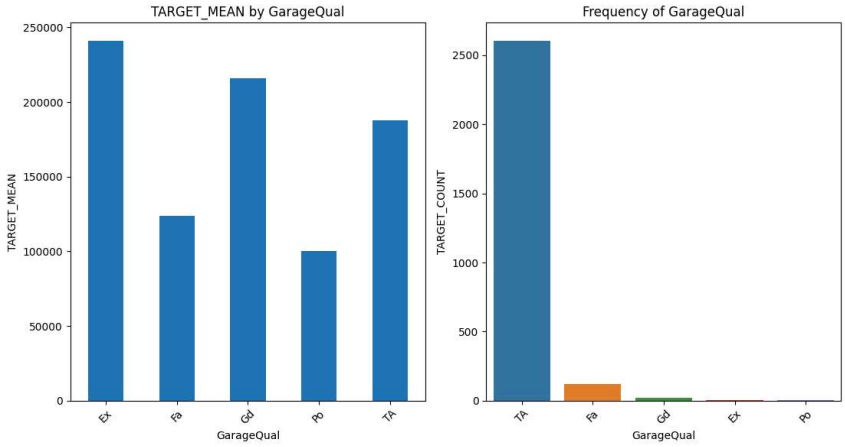
##

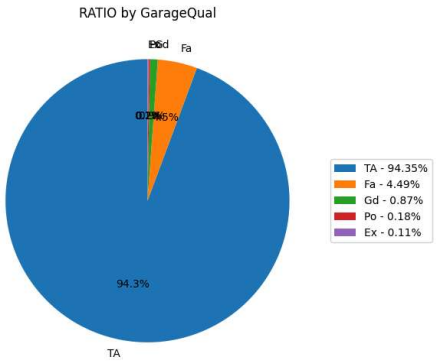




GarageQual	TARGET_MEAN	TARGET_COUNT	RATIO
Ex	241000.000	3	0.103
Fa	123573.354	48	4.248
Gd	215860.714	14	0.822
Po	100166.667	3	0.171
TA	187489.836	1311	89.209

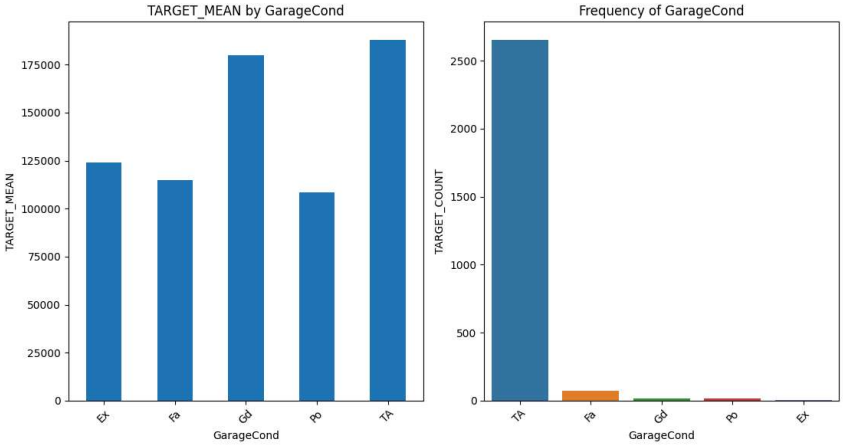
##

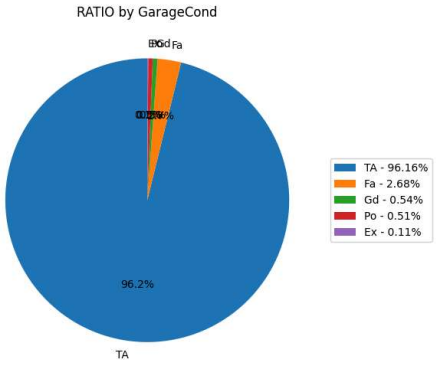




	TARGET_MEAN	TARGET_COUNT	RATIO
GarageCond			
Ex	124000.000	2	0.103
Fa	114654.029	35	2.535
Gd	179930.000	9	0.514
Po	108500.000	7	0.480
TA	187885.735	1326	90.922

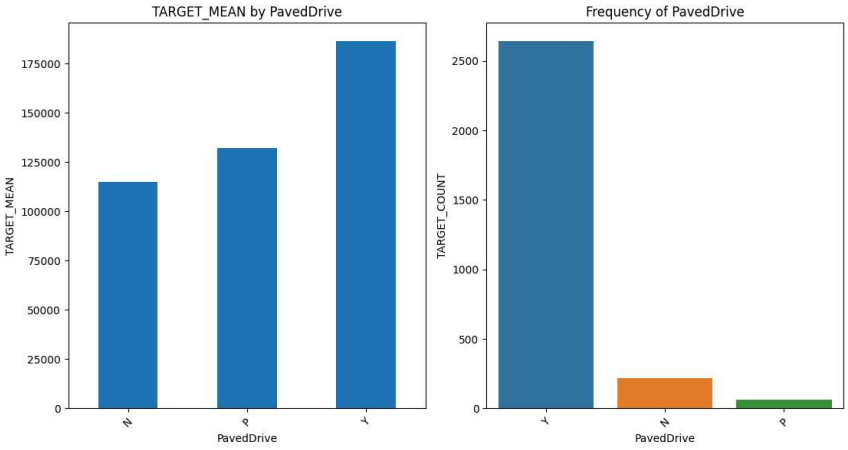
##

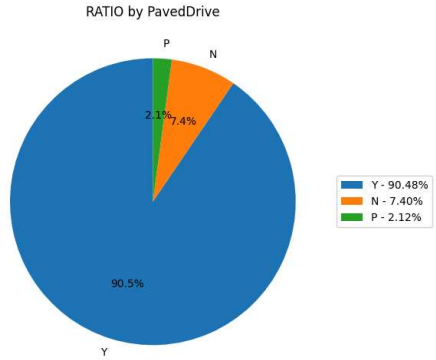




	TARGET_MEAN	TARGET_COUNT	RATIO
PavedDrive			
N	115039.122	90	7.400
P	132330.000	30	2.124
Y	186433.974	1340	90.476

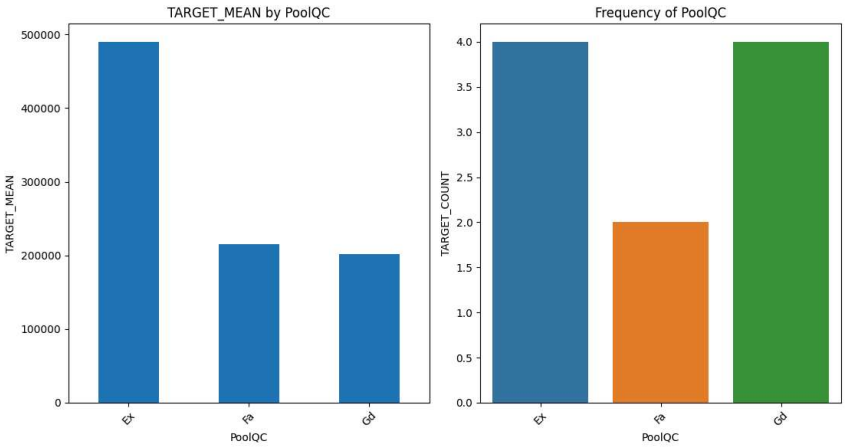
##

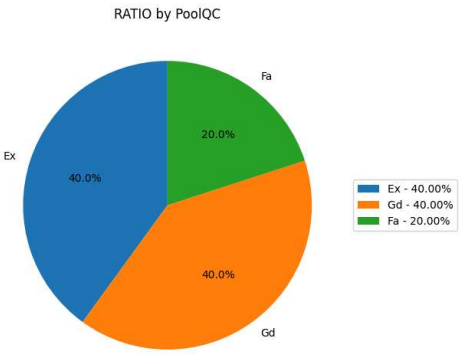




	TARGET_MEAN	TARGET_COUNT	RATIO
PoolQC			
Ex	490000.000	2	0.137
Fa	215500.000	2	0.069
Gd	201990.000	3	0.137

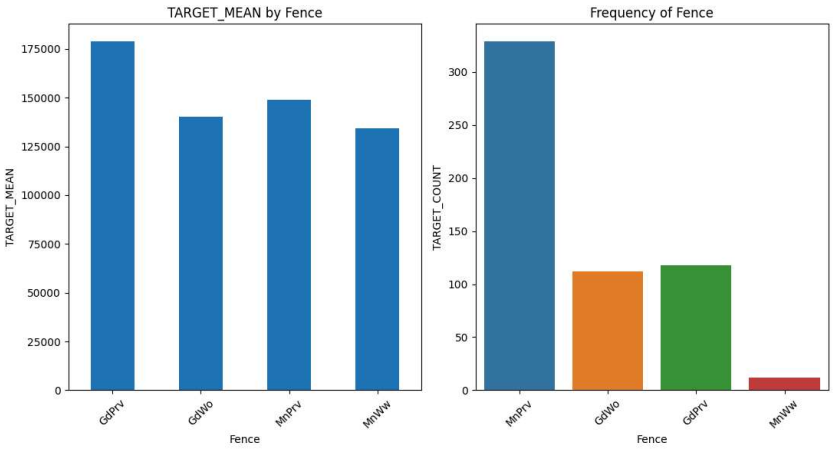
##

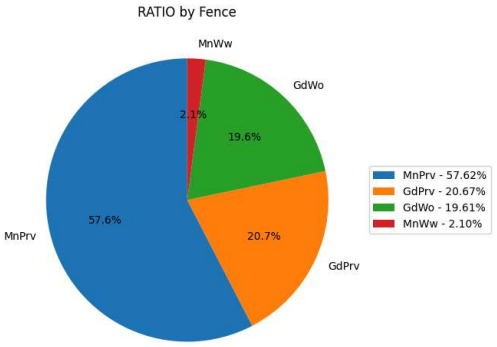




	TARGET_MEAN	TARGET_COUNT	RATIO
Fence			
GdPrv	178927.458	59	4.042
GdWo	140379.315	54	3.837
MnPrv	148751.089	157	11.271
MnWw	134286.364	11	0.411

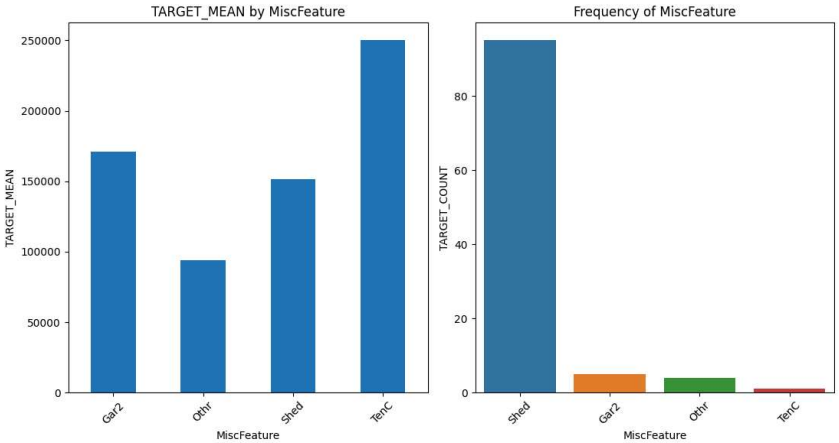
##

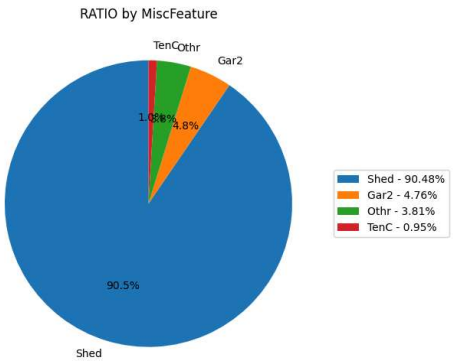




	TARGET_MEAN	TARGET_COUNT	RATIO
MiscFeature			
Gar2	170750.000	2	0.171
Othr	94000.000	2	0.137
Shed	151187.612	49	3.255
TenC	250000.000	1	0.034

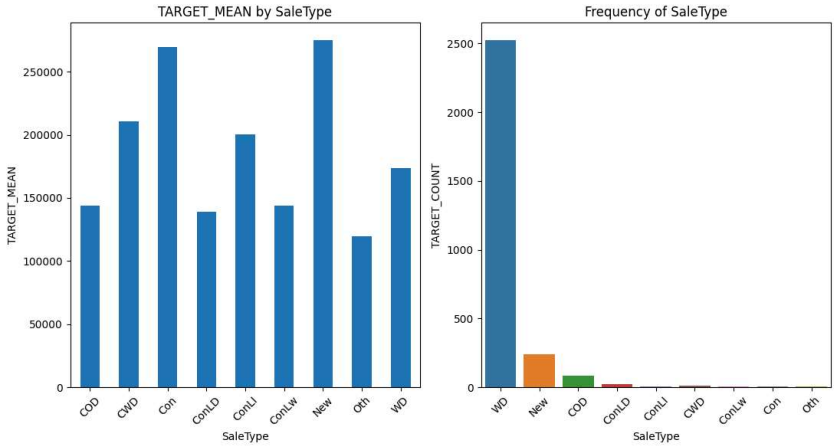
##

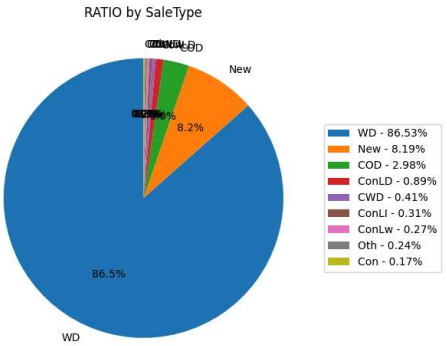




SaleType	TARGET_MEAN	TARGET_COUNT	RATIO
COD	143973.256	43	2.980
CWD	210600.000	4	0.411
Con	269600.000	2	0.171
ConLD	138780.889	9	0.891
ConLI	200390.000	5	0.308
ConLw	143700.000	5	0.274
New	274945.418	122	8.188
Oth	119850.000	3	0.240
WD	173401.837	1267	86.502

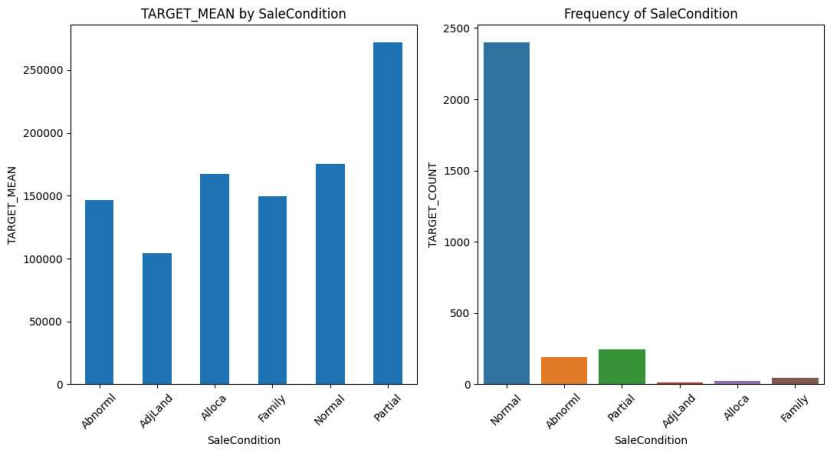
##

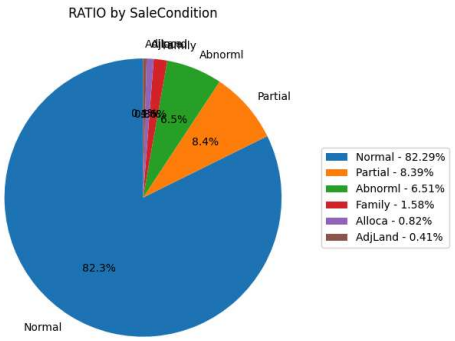




SaleCondition	TARGET_MEAN	TARGET_COUNT	RATIO
Abnorml	146526.624	101	6.509
AdjLand	104125.000	4	0.411
Alloca	167377.417	12	0.822
Family	149600.000	20	1.576
Normal	175202.220	1198	82.288
Partial	272291.752	125	8.393

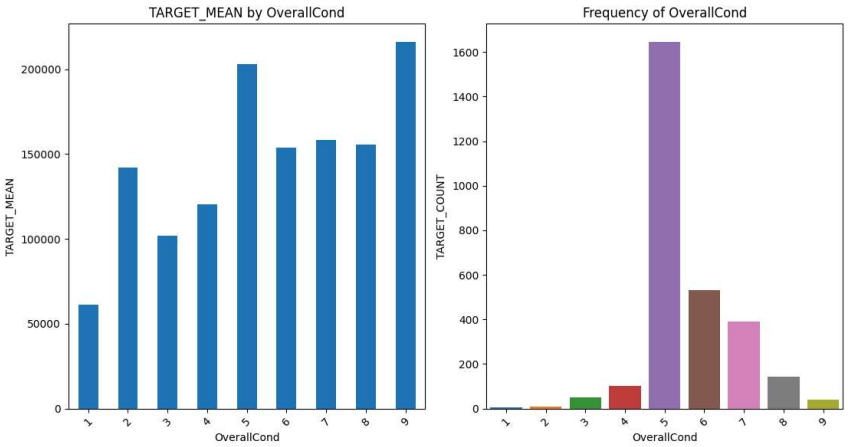
##

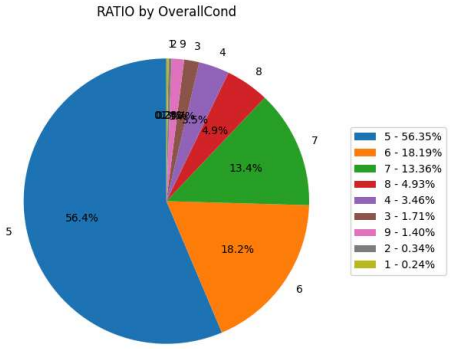




OverallCond	TARGET_MEAN	TARGET_COUNT	RATIO
1	61000.000	1	0.240
2	141986.400	5	0.343
3	101929.400	25	1.713
4	120438.439	57	3.460
5	203146.915	821	56.355
6	153961.591	252	18.191
7	158145.488	205	13.361
8	155651.736	72	4.933
9	216004.545	22	1.405

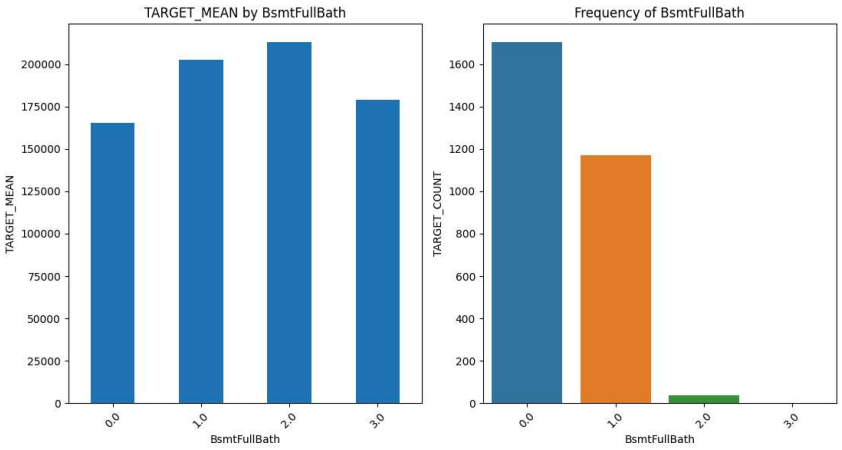
##

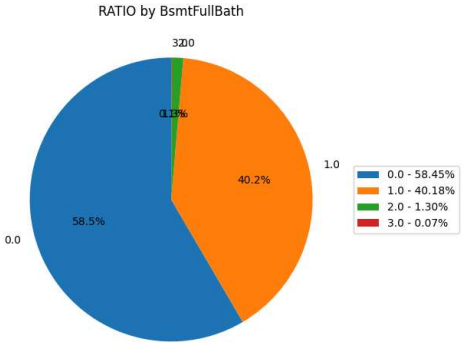




	TARGET_MEAN	TARGET_COUNT	RATIO
BsmtFullBath			
0.000	165521.640	856	58.410
1.000	202522.918	588	40.151
2.000	213063.067	15	1.302
3.000	179000.000	1	0.069

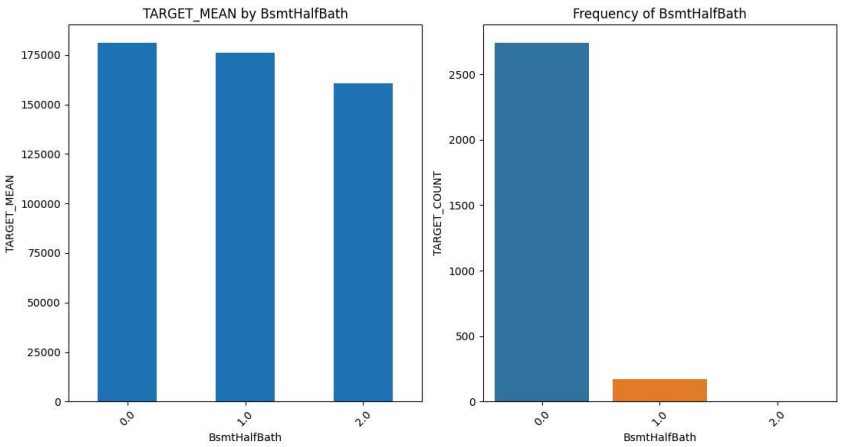
##

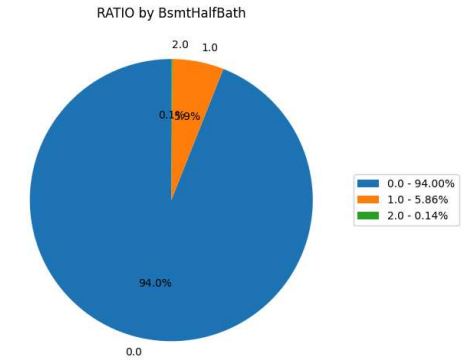




	TARGET_MEAN	TARGET_COUNT	RATIO
BsmtHalfBath			
0.000	181230.330	1378	93.936
1.000	176098.125	80	5.858
2.000	160850.500	2	0.137

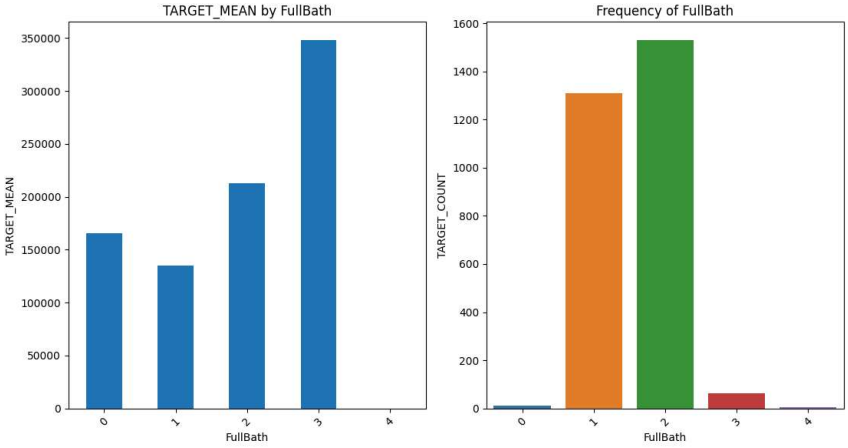
##

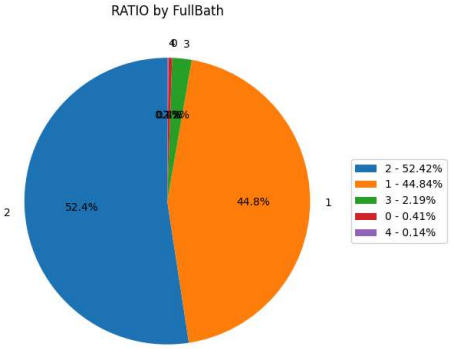




FullBath	TARGET_MEAN	TARGET_COUNT	RATIO
0	165200.889	9	0.411
1	134751.440	650	44.844
2	213009.826	768	52.415
3	347822.909	33	2.193
4	NaN	0	0.137

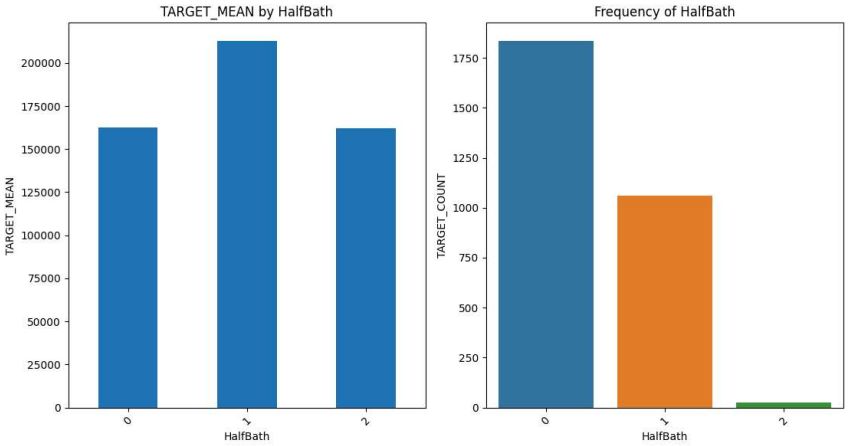
##

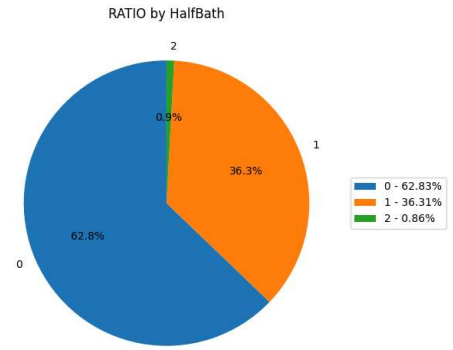




HalfBath	TARGET_MEAN	TARGET_COUNT	RATIO
0	162534.885	913	62.830
1	212721.961	535	36.314
2	162028.917	12	0.856

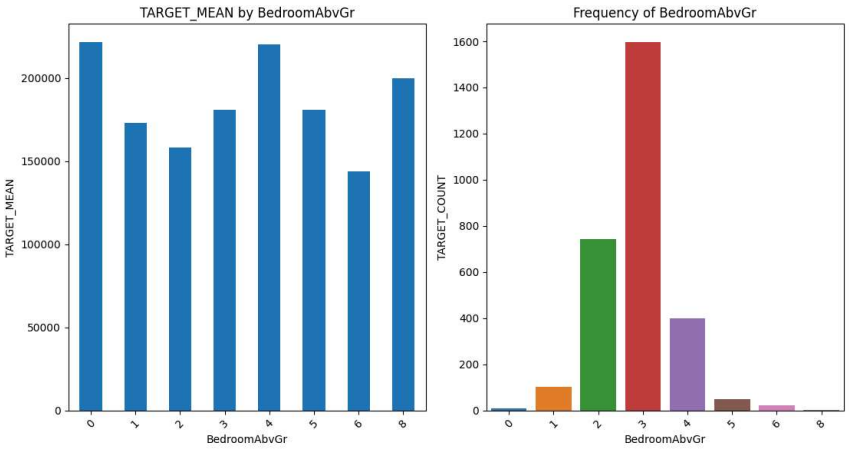
##

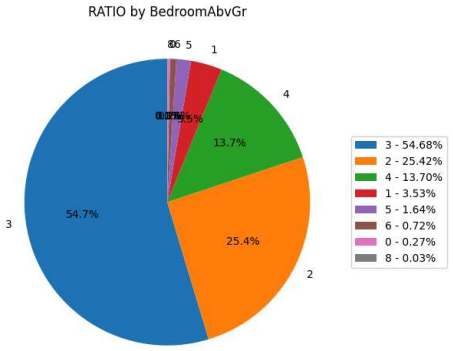




BedroomAbvGr	TARGET_MEAN	TARGET_COUNT	RATIO
0	221493.167	6	0.274
1	173162.420	50	3.529
2	158197.659	358	25.420
3	181056.871	804	54.676
4	220421.254	213	13.703
5	180819.048	21	1.644
6	143779.000	7	0.719
8	200000.000	1	0.034

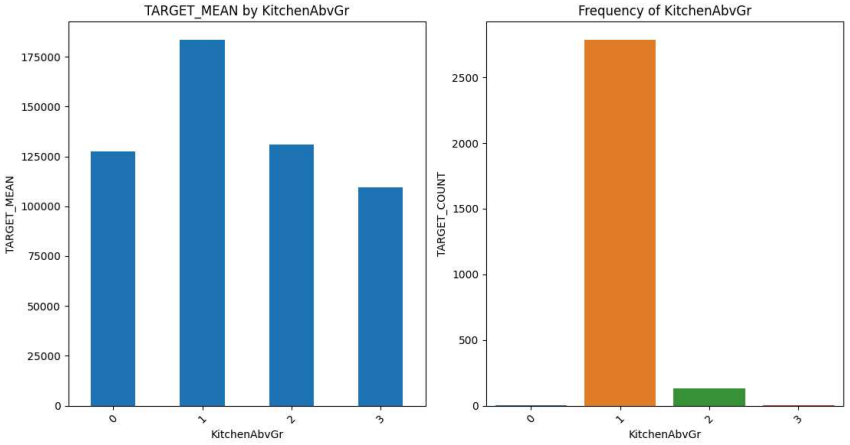
##

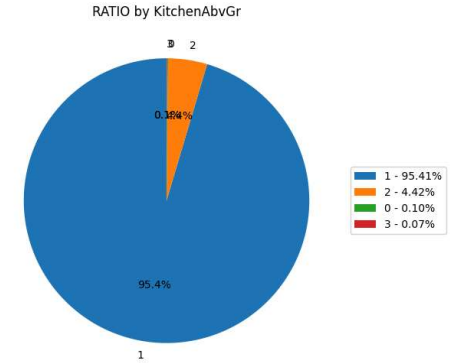




KitchenAbvGr	TARGET_MEAN	TARGET_COUNT	RATIO
0	127500.000	1	0.103
1	183388.790	1392	95.409
2	131096.154	65	4.419
3	109500.000	2	0.069

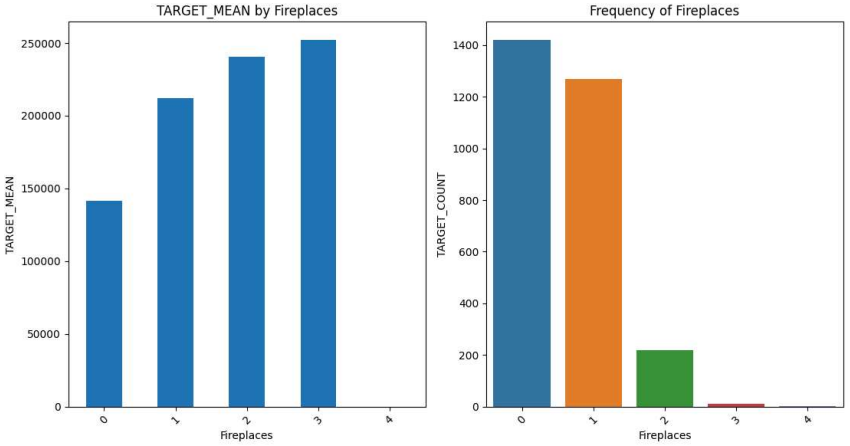
##

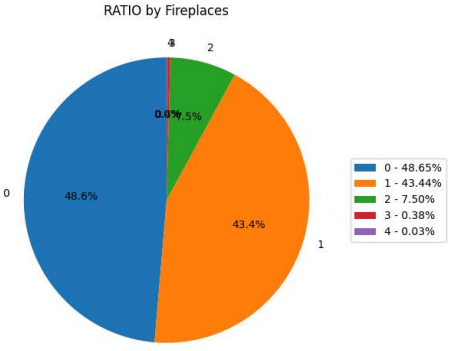




Fireplaces	TARGET_MEAN	TARGET_COUNT	RATIO
0	141331.483	690	48.647
1	211843.909	650	43.440
2	240588.539	115	7.503
3	252000.000	5	0.377
4	NaN	0	0.034

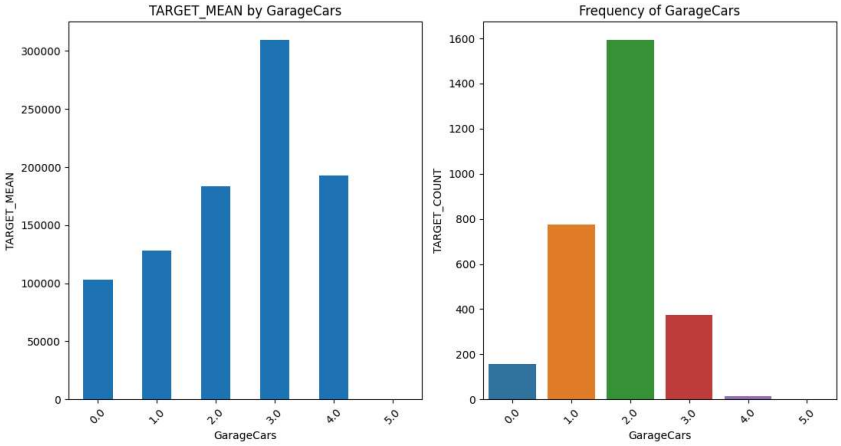
##

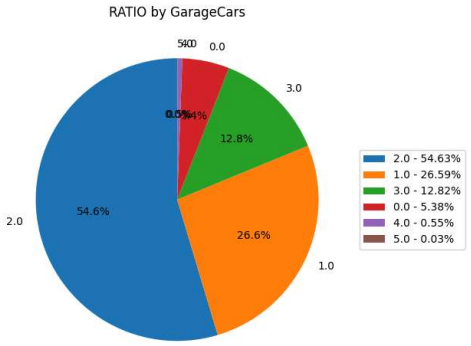




```
            TARGET_MEAN  TARGET_COUNT  RATIO
GarageCars
0.000         103317.284            81  5.379
1.000         128116.688           369 26.584
2.000         183851.664           824 54.608
3.000         309636.122           181 12.813
4.000         192655.800             5  0.548
5.000                NaN             0  0.034
```

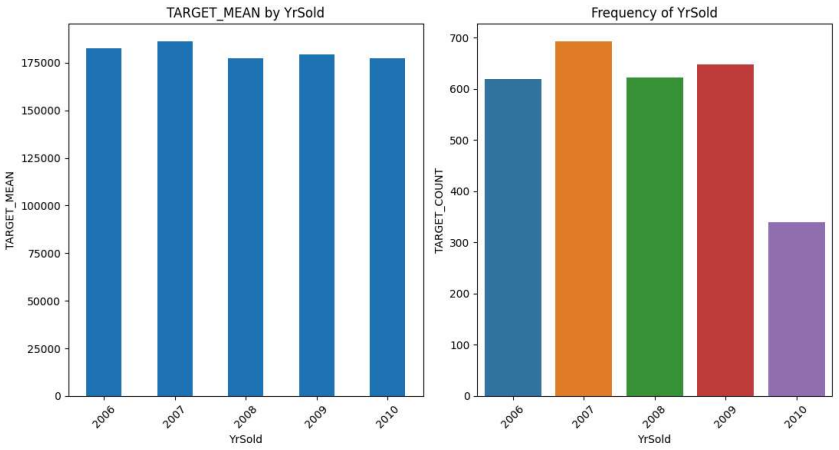
```
##########################################
```

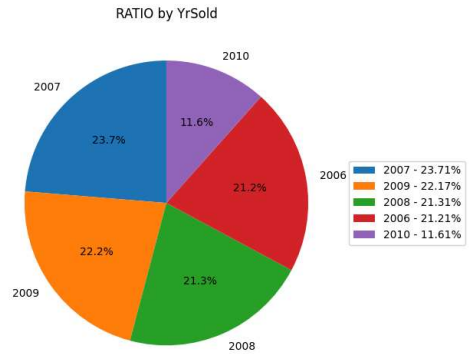





```
         TARGET_MEAN  TARGET_COUNT  RATIO
YrSold
2006     182549.459           314 21.206
2007     186063.152           329 23.707
2008     177360.839           304 21.309
2009     179432.104           338 22.165
2010     177393.674           175 11.614
```

```
##########################################
```





Num Cols Target Summary

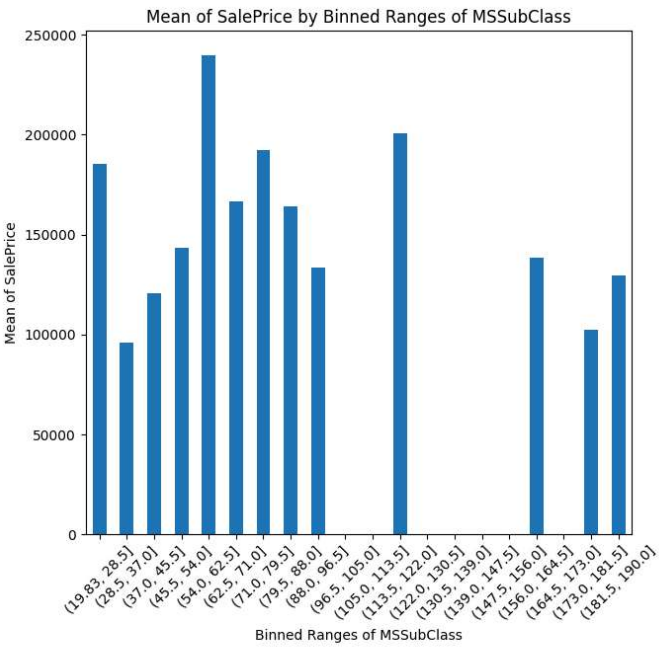
In [ ]:
```python
def target_summary_with_num(dataframe, target, numerical_col, bins=10):
    df = dataframe.copy()

    summary_df = df.groupby(target).agg({numerical_col: "mean"})

    df["binned"] = pd.cut(df[numerical_col], bins=bins)
    binned_summary_df = df.groupby("binned").agg({target: "mean"})

    return binned_summary_df, numerical_col
```

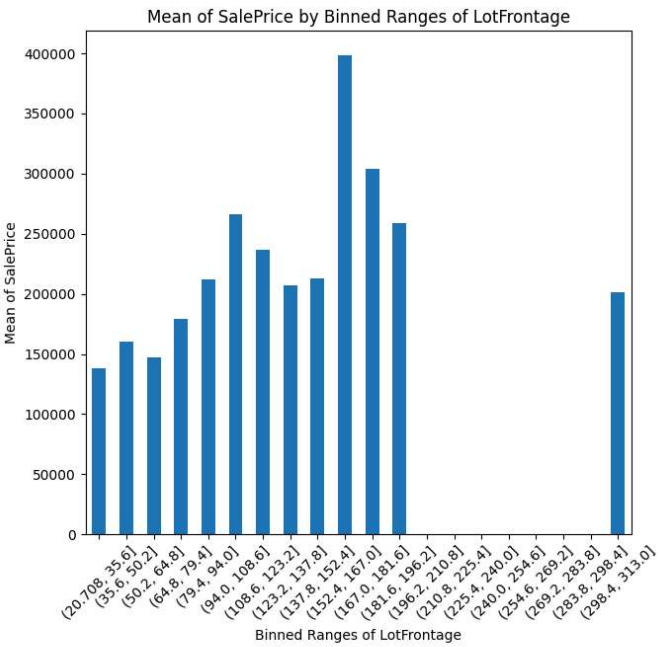
In [ ]:
```python
fig, axes = plt.subplots(nrows=(len(num_cols) + 1) // 2, ncols=2, figsize=(14, 7 * ((len(num_

for idx, col in enumerate(num_cols):
    binned_summary_df, numerical_col = target_summary_with_num(df, "SalePrice", col, bins=20)
```

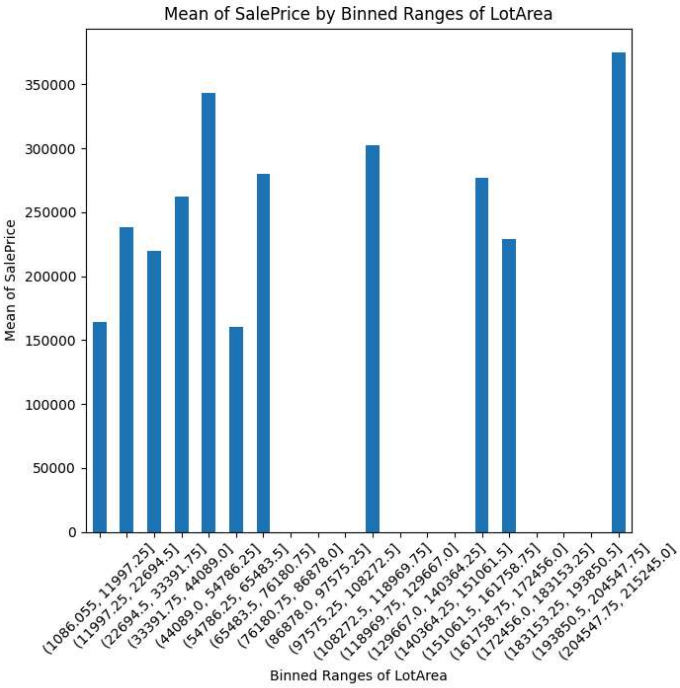
```
    ax = axes[idx // 2, idx % 2]
    binned_summary_df.plot(kind="bar", legend=False, ax=ax)
    ax.set_title(f"Mean of SalePrice by Binned Ranges of {numerical_col}")
    ax.set_xlabel(f"Binned Ranges of {numerical_col}")
    ax.set_ylabel(f"Mean of SalePrice")
    ax.set_xticklabels(binned_summary_df.index, rotation=45)

plt.tight_layout()
plt.show()
```

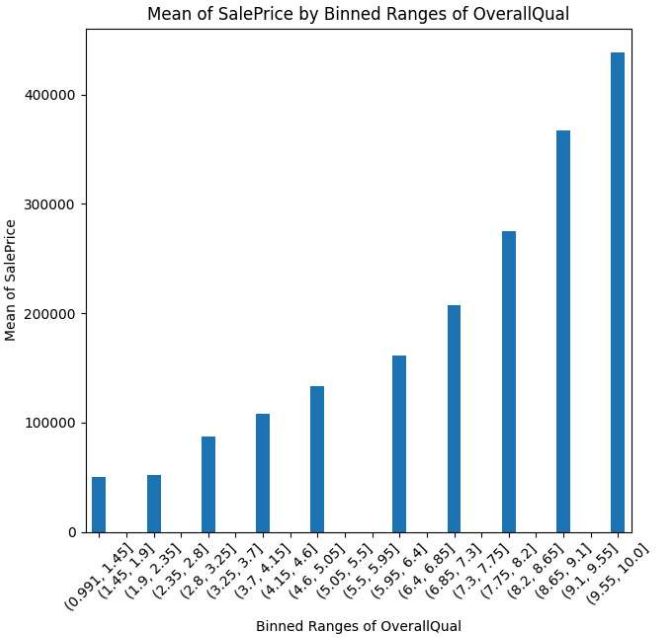
```
C:\Users\Prasanna Pandhare\AppData\Local\Temp\ipykernel_18976\1484173475.py:7: FutureWarning: The default of observed=False is deprecated and will be changed to True in a future version of pandas. Pass observed=False to retain current behavior or observed=True to adopt the future de fault and silence this warning.
  binned_summary_df = df.groupby("binned").agg({target: "mean"})
C:\Users\Prasanna Pandhare\AppData\Local\Temp\ipykernel_18976\1484173475.py:7: FutureWarning: The default of observed=False is deprecated and will be changed to True in a future version of pandas. Pass observed=False to retain current behavior or observed=True to adopt the future de fault and silence this warning.
  binned_summary_df = df.groupby("binned").agg({target: "mean"})
C:\Users\Prasanna Pandhare\AppData\Local\Temp\ipykernel_18976\1484173475.py:7: FutureWarning: The default of observed=False is deprecated and will be changed to True in a future version of pandas. Pass observed=False to retain current behavior or observed=True to adopt the future de fault and silence this warning.
  binned_summary_df = df.groupby("binned").agg({target: "mean"})
C:\Users\Prasanna Pandhare\AppData\Local\Temp\ipykernel_18976\1484173475.py:7: FutureWarning: The default of observed=False is deprecated and will be changed to True in a future version of pandas. Pass observed=False to retain current behavior or observed=True to adopt the future de fault and silence this warning.
  binned_summary_df = df.groupby("binned").agg({target: "mean"})
C:\Users\Prasanna Pandhare\AppData\Local\Temp\ipykernel_18976\1484173475.py:7: FutureWarning: The default of observed=False is deprecated and will be changed to True in a future version of pandas. Pass observed=False to retain current behavior or observed=True to adopt the future de fault and silence this warning.
  binned_summary_df = df.groupby("binned").agg({target: "mean"})
C:\Users\Prasanna Pandhare\AppData\Local\Temp\ipykernel_18976\1484173475.py:7: FutureWarning: The default of observed=False is deprecated and will be changed to True in a future version of pandas. Pass observed=False to retain current behavior or observed=True to adopt the future de fault and silence this warning.
  binned_summary_df = df.groupby("binned").agg({target: "mean"})
C:\Users\Prasanna Pandhare\AppData\Local\Temp\ipykernel_18976\1484173475.py:7: FutureWarning: The default of observed=False is deprecated and will be changed to True in a future version of pandas. Pass observed=False to retain current behavior or observed=True to adopt the future de fault and silence this warning.
  binned_summary_df = df.groupby("binned").agg({target: "mean"})
C:\Users\Prasanna Pandhare\AppData\Local\Temp\ipykernel_18976\1484173475.py:7: FutureWarning: The default of observed=False is deprecated and will be changed to True in a future version of pandas. Pass observed=False to retain current behavior or observed=True to adopt the future de fault and silence this warning.
  binned_summary_df = df.groupby("binned").agg({target: "mean"})
C:\Users\Prasanna Pandhare\AppData\Local\Temp\ipykernel_18976\1484173475.py:7: FutureWarning: The default of observed=False is deprecated and will be changed to True in a future version of pandas. Pass observed=False to retain current behavior or observed=True to adopt the future de fault and silence this warning.
  binned_summary_df = df.groupby("binned").agg({target: "mean"})
C:\Users\Prasanna Pandhare\AppData\Local\Temp\ipykernel_18976\1484173475.py:7: FutureWarning: The default of observed=False is deprecated and will be changed to True in a future version of pandas. Pass observed=False to retain current behavior or observed=True to adopt the future de fault and silence this warning.
  binned_summary_df = df.groupby("binned").agg({target: "mean"})
C:\Users\Prasanna Pandhare\AppData\Local\Temp\ipykernel_18976\1484173475.py:7: FutureWarning: The default of observed=False is deprecated and will be changed to True in a future version of pandas. Pass observed=False to retain current behavior or observed=True to adopt the future de fault and silence this warning.
  binned_summary_df = df.groupby("binned").agg({target: "mean"})
C:\Users\Prasanna Pandhare\AppData\Local\Temp\ipykernel_18976\1484173475.py:7: FutureWarning: The default of observed=False is deprecated and will be changed to True in a future version of pandas. Pass observed=False to retain current behavior or observed=True to adopt the future de fault and silence this warning.
  binned_summary_df = df.groupby("binned").agg({target: "mean"})
C:\Users\Prasanna Pandhare\AppData\Local\Temp\ipykernel_18976\1484173475.py:7: FutureWarning: The default of observed=False is deprecated and will be changed to True in a future version of pandas. Pass observed=False to retain current behavior or observed=True to adopt the future de fault and silence this warning.
  binned_summary_df = df.groupby("binned").agg({target: "mean"})
```

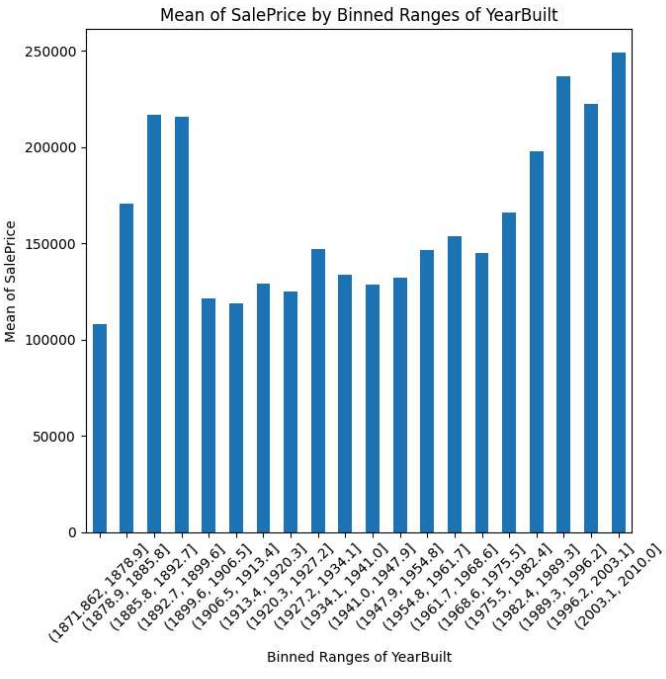
```
C:\Users\Prasanna Pandhare\AppData\Local\Temp\ipykernel_18976\1484173475.py:7: FutureWarning: 
The default of observed=False is deprecated and will be changed to True in a future version of 
pandas. Pass observed=False to retain current behavior or observed=True to adopt the future de
fault and silence this warning.
  binned_summary_df = df.groupby("binned").agg({target: "mean"})
C:\Users\Prasanna Pandhare\AppData\Local\Temp\ipykernel_18976\1484173475.py:7: FutureWarning: 
The default of observed=False is deprecated and will be changed to True in a future version of 
pandas. Pass observed=False to retain current behavior or observed=True to adopt the future de
fault and silence this warning.
  binned_summary_df = df.groupby("binned").agg({target: "mean"})
C:\Users\Prasanna Pandhare\AppData\Local\Temp\ipykernel_18976\1484173475.py:7: FutureWarning: 
The default of observed=False is deprecated and will be changed to True in a future version of 
pandas. Pass observed=False to retain current behavior or observed=True to adopt the future de
fault and silence this warning.
  binned_summary_df = df.groupby("binned").agg({target: "mean"})
C:\Users\Prasanna Pandhare\AppData\Local\Temp\ipykernel_18976\1484173475.py:7: FutureWarning: 
The default of observed=False is deprecated and will be changed to True in a future version of 
pandas. Pass observed=False to retain current behavior or observed=True to adopt the future de
fault and silence this warning.
  binned_summary_df = df.groupby("binned").agg({target: "mean"})
C:\Users\Prasanna Pandhare\AppData\Local\Temp\ipykernel_18976\1484173475.py:7: FutureWarning: 
The default of observed=False is deprecated and will be changed to True in a future version of 
pandas. Pass observed=False to retain current behavior or observed=True to adopt the future de
fault and silence this warning.
  binned_summary_df = df.groupby("binned").agg({target: "mean"})
C:\Users\Prasanna Pandhare\AppData\Local\Temp\ipykernel_18976\1484173475.py:7: FutureWarning: 
The default of observed=False is deprecated and will be changed to True in a future version of 
pandas. Pass observed=False to retain current behavior or observed=True to adopt the future de
fault and silence this warning.
  binned_summary_df = df.groupby("binned").agg({target: "mean"})
C:\Users\Prasanna Pandhare\AppData\Local\Temp\ipykernel_18976\1484173475.py:7: FutureWarning: 
The default of observed=False is deprecated and will be changed to True in a future version of 
pandas. Pass observed=False to retain current behavior or observed=True to adopt the future de
fault and silence this warning.
  binned_summary_df = df.groupby("binned").agg({target: "mean"})
C:\Users\Prasanna Pandhare\AppData\Local\Temp\ipykernel_18976\1484173475.py:7: FutureWarning: 
The default of observed=False is deprecated and will be changed to True in a future version of 
pandas. Pass observed=False to retain current behavior or observed=True to adopt the future de
fault and silence this warning.
  binned_summary_df = df.groupby("binned").agg({target: "mean"})
C:\Users\Prasanna Pandhare\AppData\Local\Temp\ipykernel_18976\1484173475.py:7: FutureWarning: 
The default of observed=False is deprecated and will be changed to True in a future version of 
pandas. Pass observed=False to retain current behavior or observed=True to adopt the future de
fault and silence this warning.
  binned_summary_df = df.groupby("binned").agg({target: "mean"})
C:\Users\Prasanna Pandhare\AppData\Local\Temp\ipykernel_18976\1484173475.py:7: FutureWarning: 
The default of observed=False is deprecated and will be changed to True in a future version of 
pandas. Pass observed=False to retain current behavior or observed=True to adopt the future de
fault and silence this warning.
  binned_summary_df = df.groupby("binned").agg({target: "mean"})
C:\Users\Prasanna Pandhare\AppData\Local\Temp\ipykernel_18976\1484173475.py:7: FutureWarning: 
The default of observed=False is deprecated and will be changed to True in a future version of 
pandas. Pass observed=False to retain current behavior or observed=True to adopt the future de
fault and silence this warning.
  binned_summary_df = df.groupby("binned").agg({target: "mean"})
C:\Users\Prasanna Pandhare\AppData\Local\Temp\ipykernel_18976\1484173475.py:7: FutureWarning: 
The default of observed=False is deprecated and will be changed to True in a future version of 
pandas. Pass observed=False to retain current behavior or observed=True to adopt the future de
fault and silence this warning.
  binned_summary_df = df.groupby("binned").agg({target: "mean"})
C:\Users\Prasanna Pandhare\AppData\Local\Temp\ipykernel_18976\1484173475.py:7: FutureWarning: 
The default of observed=False is deprecated and will be changed to True in a future version of 
pandas. Pass observed=False to retain current behavior or observed=True to adopt the future de
fault and silence this warning.
  binned_summary_df = df.groupby("binned").agg({target: "mean"})
```

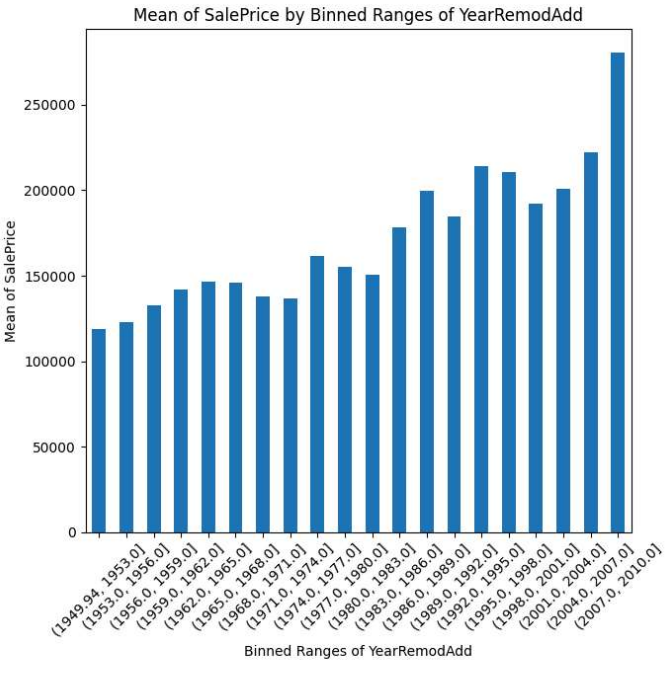

Mean of SalePrice by Binned Ranges of MSSubClass
Mean of SalePrice
Binned Ranges of MSSubClass


Mean of SalePrice by Binned Ranges of LotFrontage
Mean of SalePrice
Binned Ranges of LotFrontage


Mean of SalePrice by Binned Ranges of LotArea
Mean of SalePrice
Binned Ranges of LotArea

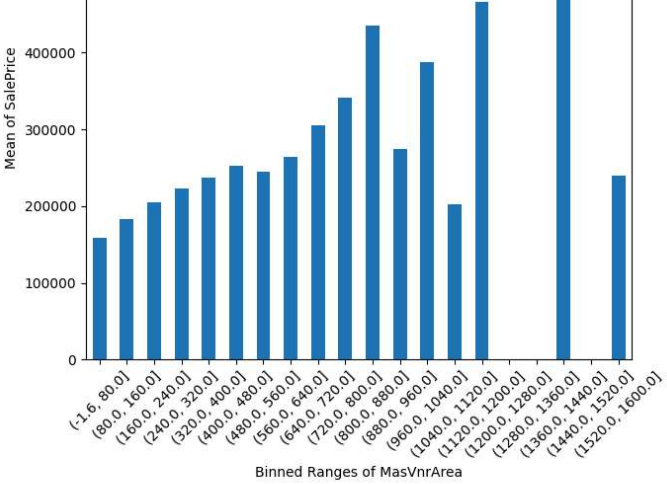

Mean of SalePrice by Binned Ranges of OverallQual
Mean of SalePrice
Binned Ranges of OverallQual

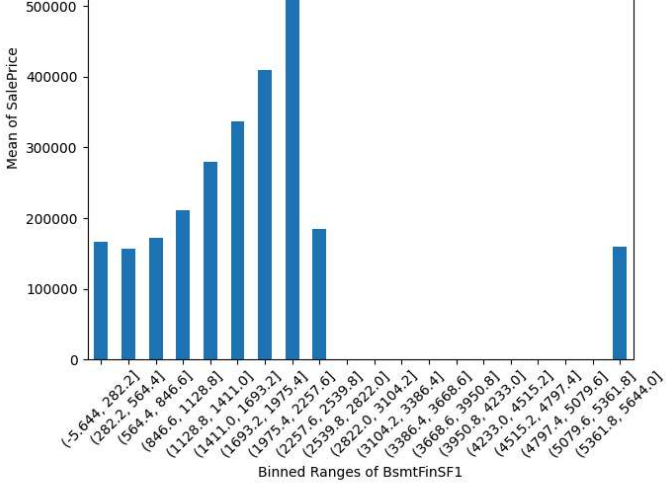

Mean of SalePrice by Binned Ranges of YearBuilt
Mean of SalePrice
Binned Ranges of YearBuilt

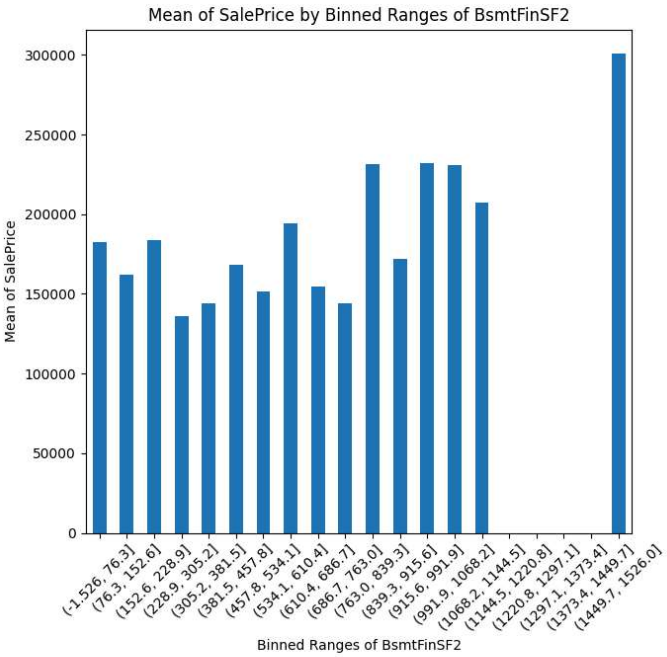

Mean of SalePrice by Binned Ranges of YearRemodAdd
Mean of SalePrice
Binned Ranges of YearRemodAdd

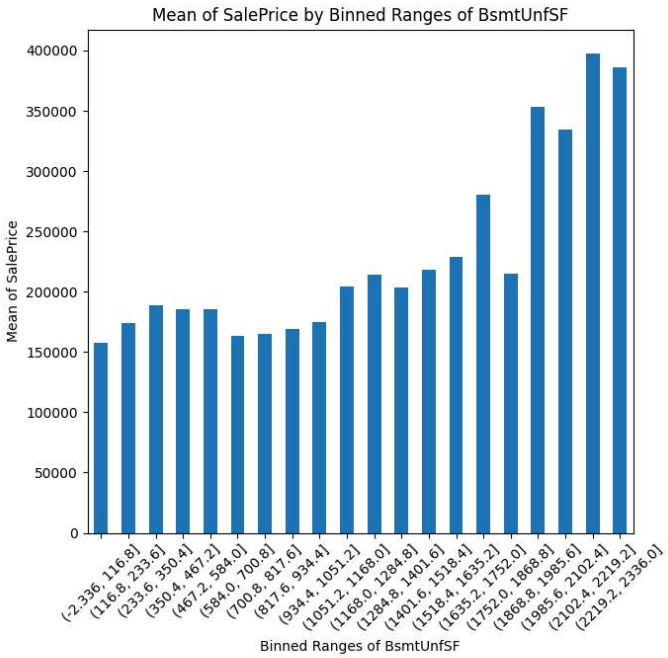

Mean of SalePrice by Binned Ranges of MasVnrArea

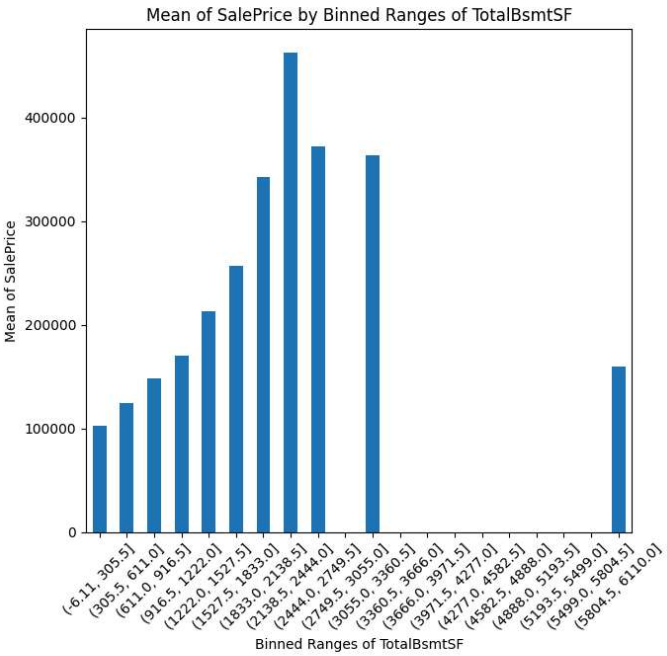

Mean of SalePrice by Binned Ranges of BsmtFinSF1

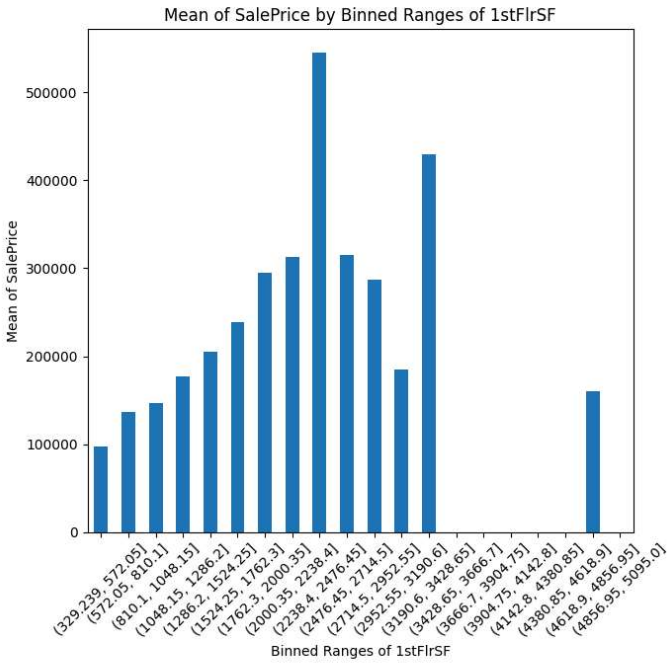

Mean of SalePrice
Binned Ranges of MasVnrArea


Mean of SalePrice
Binned Ranges of BsmtFinSF1


Mean of SalePrice by Binned Ranges of BsmtFinSF2
Mean of SalePrice
Binned Ranges of BsmtFinSF2

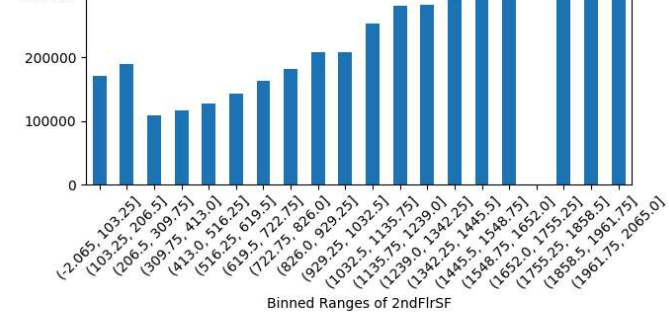

Mean of SalePrice by Binned Ranges of BsmtUnfSF
Mean of SalePrice
Binned Ranges of BsmtUnfSF

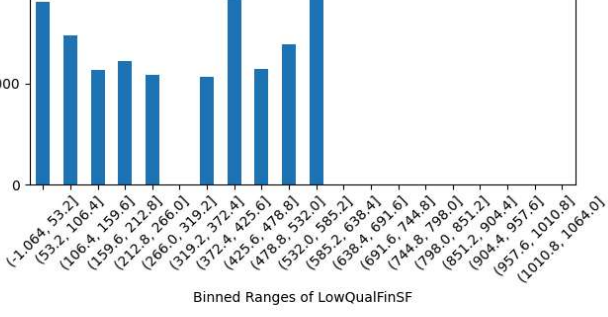

Mean of SalePrice by Binned Ranges of TotalBsmtSF
Mean of SalePrice
Binned Ranges of TotalBsmtSF

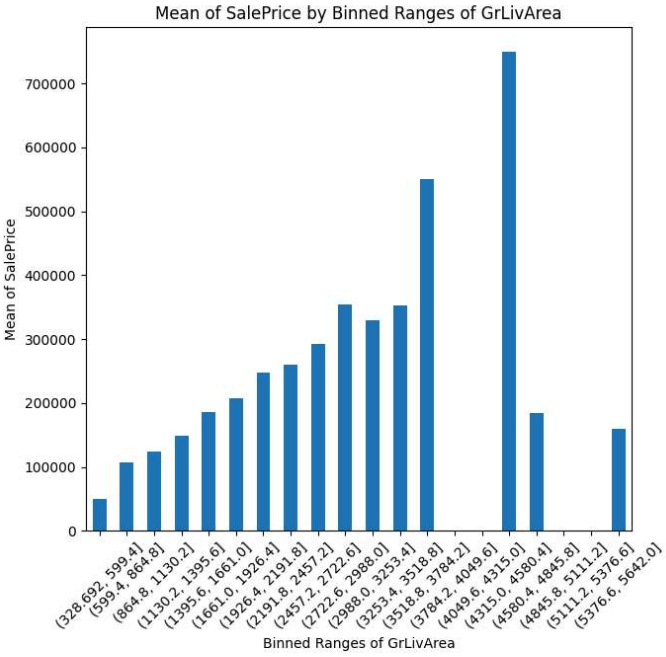

Mean of SalePrice by Binned Ranges of 1stFlrSF
Mean of SalePrice
Binned Ranges of 1stFlrSF

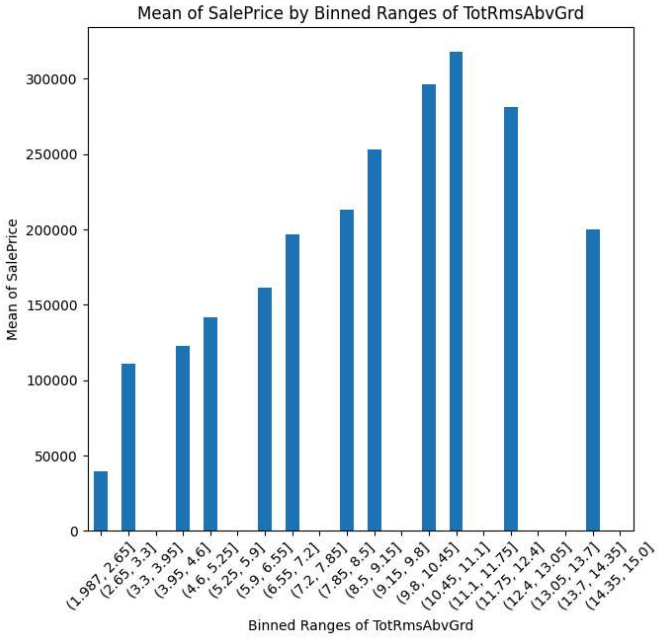

Mean of SalePrice by Binned Ranges of 2ndFlrSF
Mean of SalePrice

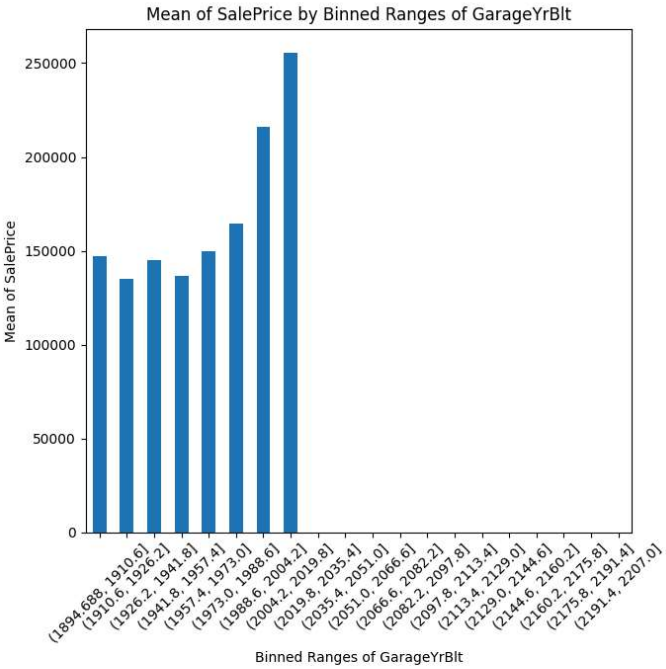

Mean of SalePrice by Binned Ranges of LowQualFinSF
Mean of SalePrice

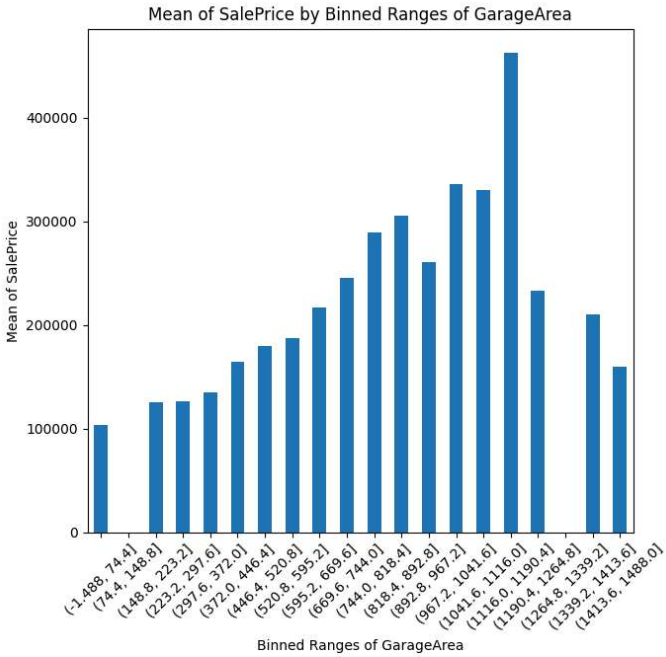

Binned Ranges of 2ndFlrSF

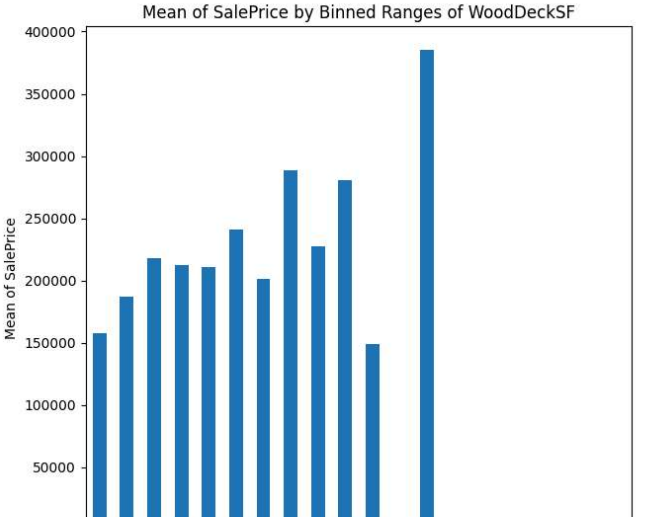

Binned Ranges of LowQualFinSF

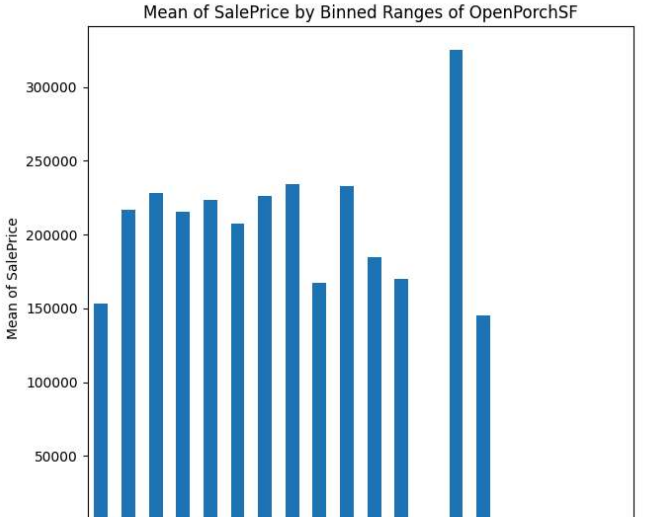

Mean of SalePrice by Binned Ranges of GrLivArea
Mean of SalePrice
Binned Ranges of GrLivArea

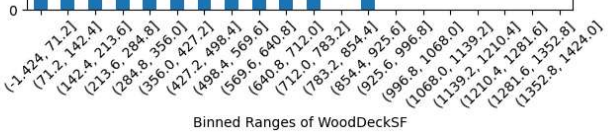

Mean of SalePrice by Binned Ranges of TotRmsAbvGrd
Mean of SalePrice
Binned Ranges of TotRmsAbvGrd

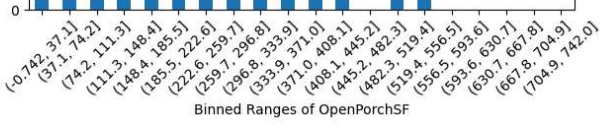

Mean of SalePrice by Binned Ranges of GarageYrBlt
Mean of SalePrice
Binned Ranges of GarageYrBlt

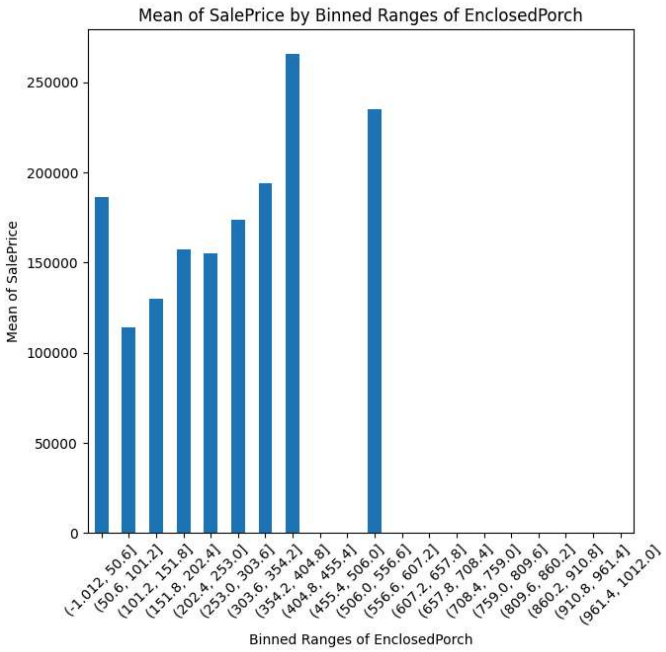

Mean of SalePrice by Binned Ranges of GarageArea
Mean of SalePrice
Binned Ranges of GarageArea

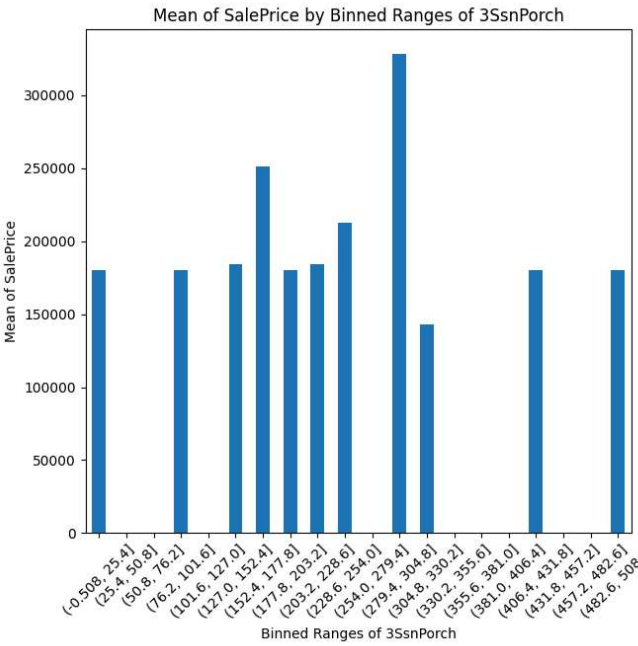

Mean of SalePrice by Binned Ranges of WoodDeckSF
Mean of SalePrice

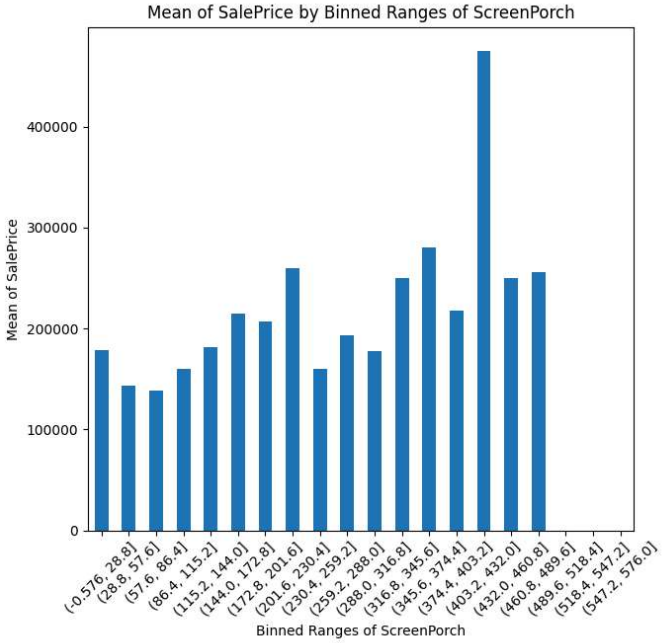

Mean of SalePrice by Binned Ranges of OpenPorchSF
Mean of SalePrice

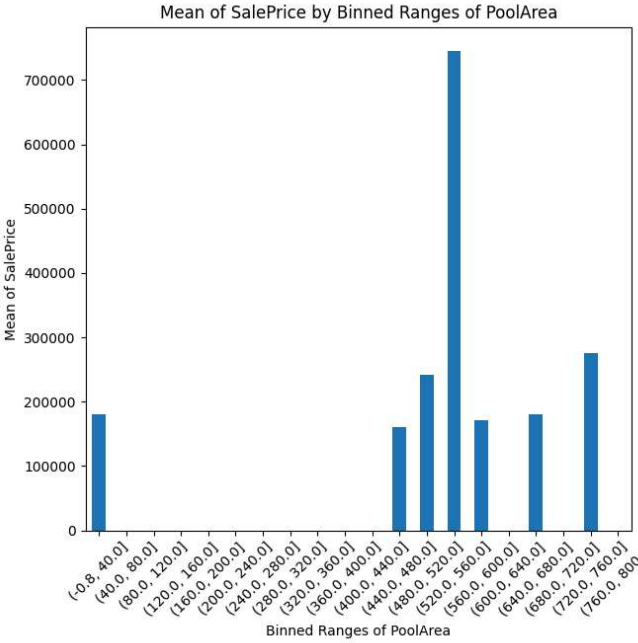

Binned Ranges of WoodDeckSF

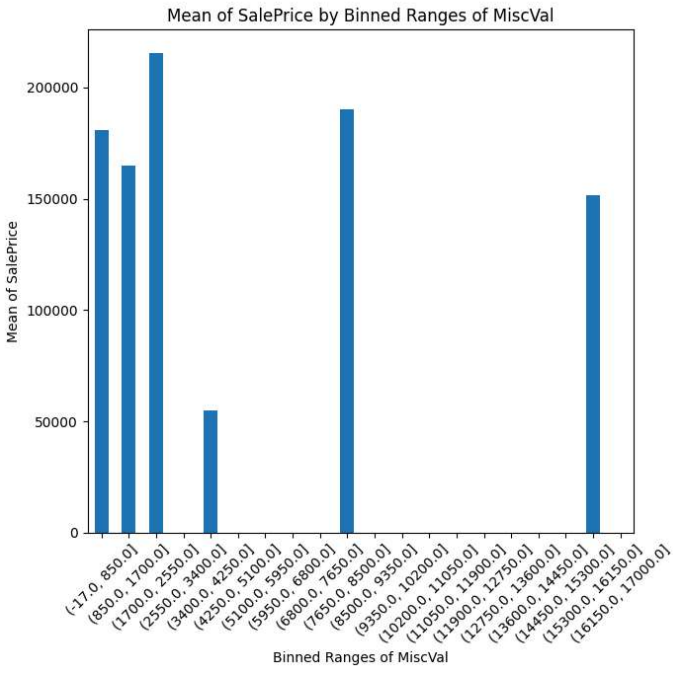

Binned Ranges of OpenPorchSF

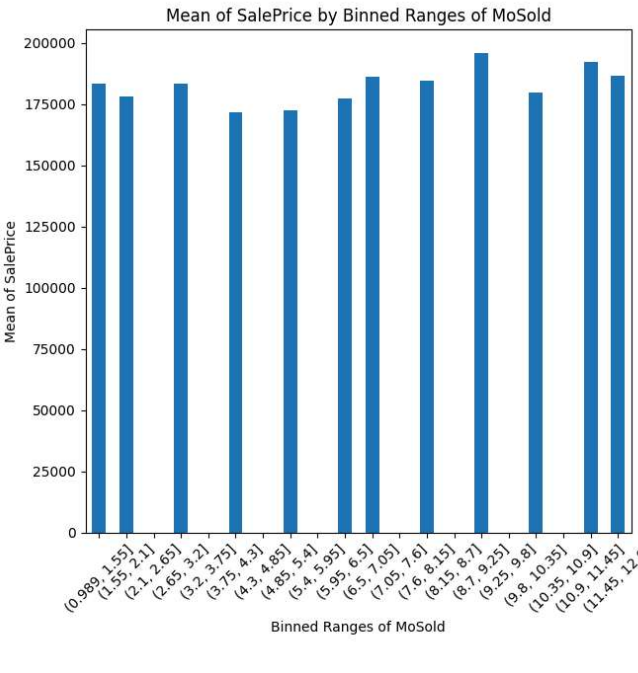

Mean of SalePrice by Binned Ranges of EnclosedPorch
Mean of SalePrice
Binned Ranges of EnclosedPorch


Mean of SalePrice by Binned Ranges of 3SsnPorch
Mean of SalePrice
Binned Ranges of 3SsnPorch


Mean of SalePrice by Binned Ranges of ScreenPorch
Mean of SalePrice
Binned Ranges of ScreenPorch


Mean of SalePrice by Binned Ranges of PoolArea
Mean of SalePrice
Binned Ranges of PoolArea

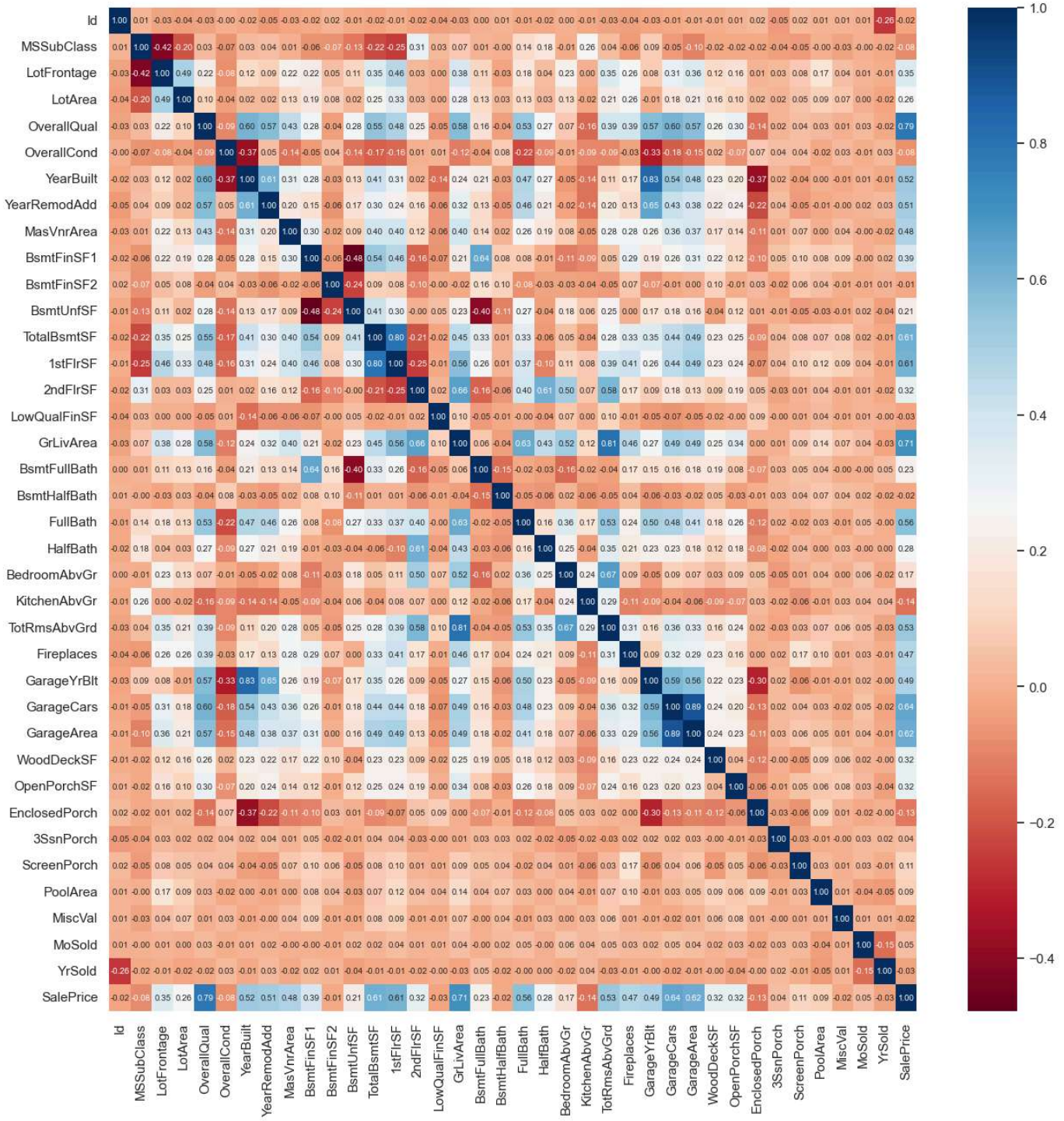

Mean of SalePrice by Binned Ranges of MiscVal
Mean of SalePrice
Binned Ranges of MiscVal


Mean of SalePrice by Binned Ranges of MoSold
Mean of SalePrice
Binned Ranges of MoSold

Correlation Analysis

In [ ]:
```
def high_correlated_cols(dataframe, plot=False, corr_th=0.90):
    corr = dataframe.corr(numeric_only=True)
    cor_matrix = corr.abs()
    upper_triangle_matrix = cor_matrix.where(np.triu(np.ones(cor_matrix.shape), k=1).astype(b
    drop_list = [col for col in upper_triangle_matrix.columns if any(upper_triangle_matrix[co
    if plot:
        sns.set_theme(rc={"figure.figsize": (15, 15)})
        sns.heatmap(corr, annot=True, fmt=".2f", cmap="RdBu", annot_kws={"size": 7})
        plt.show()
    return drop_list
```

In [ ]:
```
high_correlated_cols(df, plot=True)
```



Out[ ]:
```
[]
```

Outlier Analysis

In [ ]:
```python
def outlier_thresholds(dataframe, col_name, q1=0.05, q3=0.95):
    quartile1 = dataframe[col_name].quantile(q1)
    quartile3 = dataframe[col_name].quantile(q3)
    interquantile_range = quartile3 - quartile1
    up_limit = quartile3 + 1.5 * interquantile_range
    low_limit = quartile1 - 1.5 * interquantile_range
    return low_limit, up_limit

def check_outlier(dataframe, col_name):
    low_limit, up_limit = outlier_thresholds(dataframe, col_name)
    if dataframe[(dataframe[col_name] > up_limit) | (dataframe[col_name] < low_limit)].shape[
        return True
    else:
        return False
```

In [ ]:
```python
for col in num_cols:
    print(col, check_outlier(df, col))
```

```
MSSubClass False
LotFrontage True
LotArea True
OverallQual False
YearBuilt False
YearRemodAdd False
MasVnrArea True
BsmtFinSF1 True
BsmtFinSF2 True
BsmtUnfSF False
TotalBsmtSF True
1stFlrSF True
2ndFlrSF False
LowQualFinSF True
GrLivArea True
TotRmsAbvGrd False
GarageYrBlt True
GarageArea False
WoodDeckSF True
OpenPorchSF True
EnclosedPorch True
3SsnPorch True
ScreenPorch True
PoolArea True
MiscVal True
MoSold False
```

In [ ]:
```python
def replace_with_thresholds(dataframe, variable):
    low_limit, up_limit = outlier_thresholds(dataframe, variable)
    dataframe.loc[(dataframe[variable] < low_limit), variable] = low_limit
    dataframe.loc[(dataframe[variable] > up_limit), variable] = up_limit
```

In [ ]:
```python
for col in num_cols:
    replace_with_thresholds(df, col)
```

```
C:\Users\Prasanna Pandhare\AppData\Local\Temp\ipykernel_18976\130684917.py:3: FutureWarning: S
etting an item of incompatible dtype is deprecated and will raise an error in a future version
of pandas. Value '-17759.35' has dtype incompatible with int64, please explicitly cast to a co
mpatible dtype first.
  dataframe.loc[(dataframe[variable] < low_limit), variable] = low_limit
C:\Users\Prasanna Pandhare\AppData\Local\Temp\ipykernel_18976\130684917.py:3: FutureWarning: S
etting an item of incompatible dtype is deprecated and will raise an error in a future version
of pandas. Value '1864.5' has dtype incompatible with int64, please explicitly cast to a compa
tible dtype first.
  dataframe.loc[(dataframe[variable] < low_limit), variable] = low_limit
C:\Users\Prasanna Pandhare\AppData\Local\Temp\ipykernel_18976\130684917.py:3: FutureWarning: S
etting an item of incompatible dtype is deprecated and will raise an error in a future version
of pandas. Value '-1080.3999999999996' has dtype incompatible with int64, please explicitly ca
st to a compatible dtype first.
  dataframe.loc[(dataframe[variable] < low_limit), variable] = low_limit
C:\Users\Prasanna Pandhare\AppData\Local\Temp\ipykernel_18976\130684917.py:3: FutureWarning: S
etting an item of incompatible dtype is deprecated and will raise an error in a future version
of pandas. Value '-1696.7999999999997' has dtype incompatible with int64, please explicitly ca
st to a compatible dtype first.
  dataframe.loc[(dataframe[variable] < low_limit), variable] = low_limit
C:\Users\Prasanna Pandhare\AppData\Local\Temp\ipykernel_18976\130684917.py:3: FutureWarning: S
etting an item of incompatible dtype is deprecated and will raise an error in a future version
of pandas. Value '-1543.7999999999997' has dtype incompatible with int64, please explicitly ca
st to a compatible dtype first.
  dataframe.loc[(dataframe[variable] < low_limit), variable] = low_limit
C:\Users\Prasanna Pandhare\AppData\Local\Temp\ipykernel_18976\130684917.py:3: FutureWarning: S
etting an item of incompatible dtype is deprecated and will raise an error in a future version
of pandas. Value '-3.5' has dtype incompatible with int64, please explicitly cast to a compati
ble dtype first.
  dataframe.loc[(dataframe[variable] < low_limit), variable] = low_limit
C:\Users\Prasanna Pandhare\AppData\Local\Temp\ipykernel_18976\130684917.py:3: FutureWarning: S
etting an item of incompatible dtype is deprecated and will raise an error in a future version
of pandas. Value '-274.64999999999986' has dtype incompatible with int64, please explicitly ca
st to a compatible dtype first.
  dataframe.loc[(dataframe[variable] < low_limit), variable] = low_limit
C:\Users\Prasanna Pandhare\AppData\Local\Temp\ipykernel_18976\130684917.py:3: FutureWarning: S
etting an item of incompatible dtype is deprecated and will raise an error in a future version
of pandas. Value '-241.5' has dtype incompatible with int64, please explicitly cast to a compa
tible dtype first.
  dataframe.loc[(dataframe[variable] < low_limit), variable] = low_limit
C:\Users\Prasanna Pandhare\AppData\Local\Temp\ipykernel_18976\130684917.py:3: FutureWarning: S
etting an item of incompatible dtype is deprecated and will raise an error in a future version
of pandas. Value '-11.5' has dtype incompatible with int64, please explicitly cast to a compat
ible dtype first.
  dataframe.loc[(dataframe[variable] < low_limit), variable] = low_limit
```

In [ ]:
```
for col in num_cols:
    print(col, check_outlier(df, col))
```

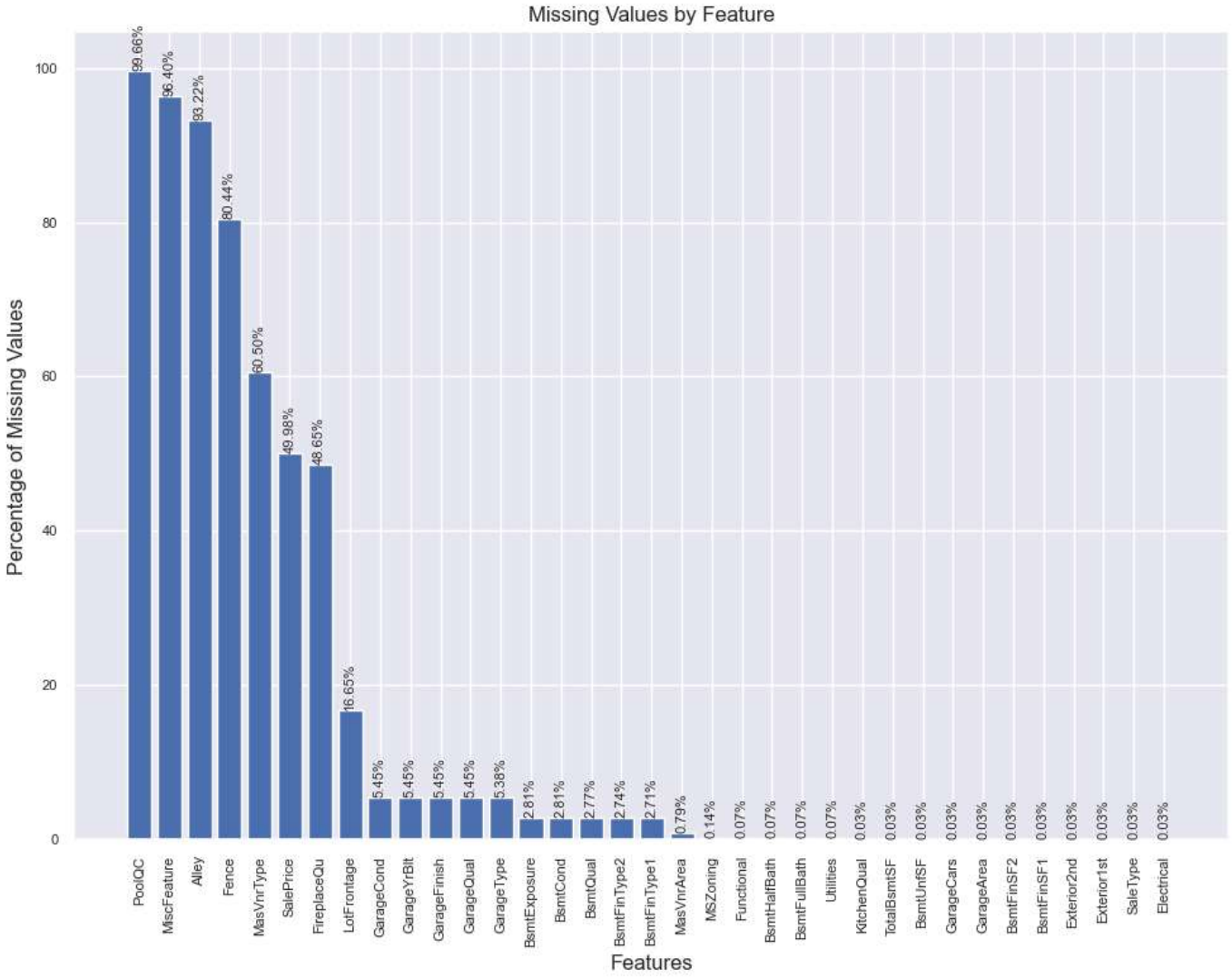
```
MSSubClass False
LotFrontage False
LotArea False
OverallQual False
YearBuilt False
YearRemodAdd False
MasVnrArea False
BsmtFinSF1 False
BsmtFinSF2 False
BsmtUnfSF False
TotalBsmtSF False
1stFlrSF False
2ndFlrSF False
LowQualFinSF False
GrLivArea False
TotRmsAbvGrd False
GarageYrBlt False
GarageArea False
WoodDeckSF False
OpenPorchSF False
EnclosedPorch False
3SsnPorch False
ScreenPorch False
PoolArea False
MiscVal False
MoSold False
```

Missing Values Analysis

In [ ]:
```python
def missing_values_table(dataframe, na_name=False, plot=False):
    na_columns = [col for col in dataframe.columns if dataframe[col].isnull().sum() > 0]
    n_miss = dataframe[na_columns].isnull().sum().sort_values(ascending=False)
    ratio = (dataframe[na_columns].isnull().sum() / dataframe.shape[0] * 100).sort_values(asc
    missing_df = pd.concat([n_miss, np.round(ratio, 2)], axis=1, keys=["n_miss", "ratio"])
    print(missing_df, end="\n")
    print("#########################################")

    if plot:
        plt.figure(figsize=(10, 8))
        bars = plt.bar(missing_df.index, missing_df["ratio"])
        plt.xlabel("Features")
        plt.ylabel("Percentage of Missing Values")
        plt.title("Missing Values by Feature")

        for bar in bars:
            yval = bar.get_height()
            plt.text(bar.get_x() + bar.get_width() / 2, yval, f"{yval:.2f}%", ha="center", va

        plt.xticks(rotation=90, fontsize=8)
        plt.yticks(fontsize=8)
        plt.grid(True)
        plt.tight_layout()
        plt.show()

    if na_name:
        return na_columns
```

In [ ]:
```python
missing_values_table(df, plot=True)
```

```
              n_miss  ratio
PoolQC          2909 99.660
MiscFeature     2814 96.400
Alley           2721 93.220
Fence           2348 80.440
MasVnrType      1766 60.500
SalePrice       1459 49.980
FireplaceQu     1420 48.650
LotFrontage      486 16.650
GarageCond       159  5.450
GarageYrBlt      159  5.450
GarageFinish     159  5.450
GarageQual       159  5.450
GarageType       157  5.380
BsmtExposure      82  2.810
BsmtCond          82  2.810
BsmtQual          81  2.770
BsmtFinType2      80  2.740
BsmtFinType1      79  2.710
MasVnrArea        23  0.790
MSZoning           4  0.140
Functional         2  0.070
BsmtHalfBath       2  0.070
BsmtFullBath       2  0.070
Utilities          2  0.070
KitchenQual        1  0.030
TotalBsmtSF        1  0.030
BsmtUnfSF          1  0.030
GarageCars         1  0.030
GarageArea         1  0.030
BsmtFinSF2         1  0.030
BsmtFinSF1         1  0.030
Exterior2nd        1  0.030
Exterior1st        1  0.030
SaleType           1  0.030
Electrical         1  0.030
##########################################
```



In [ ]:
```
no_cols = ["Alley","BsmtQual","BsmtCond","BsmtExposure","BsmtFinType1","BsmtFinType2","Firepl
           "GarageType","GarageFinish","GarageQual","GarageCond","PoolQC","Fence","MiscFeatur

for col in no_cols:
    df[col].fillna("No",inplace=True)
```

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

Out[ ]:
```
MSZoning            4
Street              0
Alley               0
LotShape            0
LandContour         0
Utilities           2
LotConfig           0
LandSlope           0
Neighborhood        0
Condition1          0
Condition2          0
BldgType            0
HouseStyle          0
RoofStyle           0
RoofMatl            0
Exterior1st         1
Exterior2nd         1
MasVnrType       1766
ExterQual           0
ExterCond           0
Foundation          0
BsmtQual            0
BsmtCond            0
BsmtExposure        0
BsmtFinType1        0
BsmtFinType2        0
Heating             0
HeatingQC           0
CentralAir          0
Electrical          1
KitchenQual         1
Functional          2
FireplaceQu         0
GarageType          0
GarageFinish        0
GarageQual          0
GarageCond          0
PavedDrive          0
PoolQC              0
Fence               0
MiscFeature         0
SaleType            1
SaleCondition       0
OverallCond         0
BsmtFullBath        2
BsmtHalfBath        2
FullBath            0
HalfBath            0
BedroomAbvGr        0
KitchenAbvGr        0
Fireplaces          0
GarageCars          1
YrSold              0
dtype: int64
```

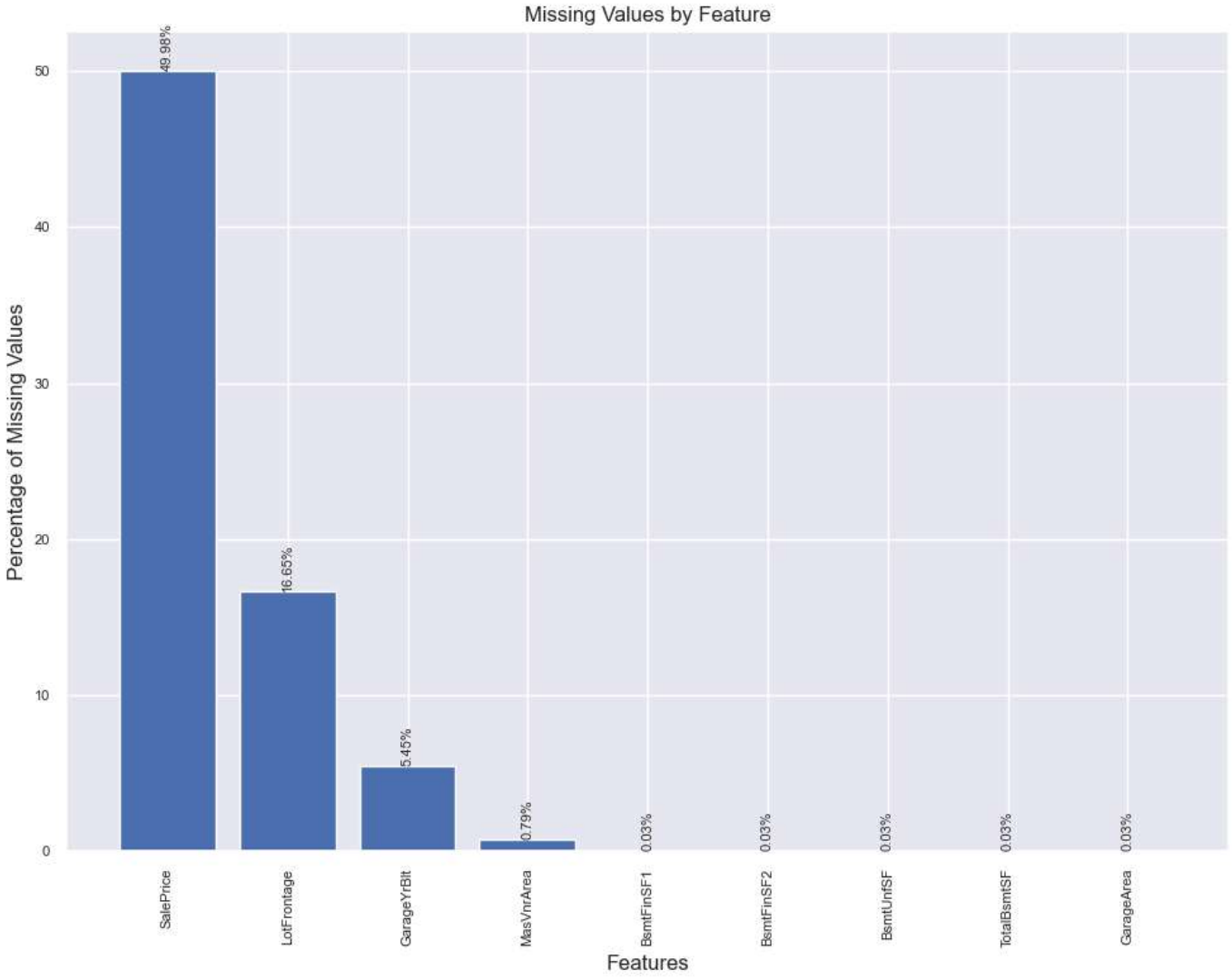
In [ ]:
```
for col in cat_cols:
    df[col].fillna(df[col].mode()[0], inplace=True)
```

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

Out[ ]:
```
MSZoning         0
Street           0
Alley            0
LotShape         0
LandContour      0
Utilities        0
LotConfig        0
LandSlope        0
Neighborhood     0
Condition1       0
Condition2       0
BldgType         0
HouseStyle       0
RoofStyle        0
RoofMatl         0
Exterior1st      0
Exterior2nd      0
MasVnrType       0
ExterQual        0
ExterCond        0
Foundation       0
BsmtQual         0
BsmtCond         0
BsmtExposure     0
BsmtFinType1     0
BsmtFinType2     0
Heating          0
HeatingQC        0
CentralAir       0
Electrical       0
KitchenQual      0
Functional       0
FireplaceQu      0
GarageType       0
GarageFinish     0
GarageQual       0
GarageCond       0
PavedDrive       0
PoolQC           0
Fence            0
MiscFeature      0
SaleType         0
SaleCondition    0
OverallCond      0
BsmtFullBath     0
BsmtHalfBath     0
FullBath         0
HalfBath         0
BedroomAbvGr     0
KitchenAbvGr     0
Fireplaces       0
GarageCars       0
YrSold           0
dtype: int64
```

In [ ]:
```
missing_values_table(df, plot=True)
```

```
             n_miss  ratio
SalePrice      1459 49.980
LotFrontage     486 16.650
GarageYrBlt     159  5.450
MasVnrArea       23  0.790
BsmtFinSF1        1  0.030
BsmtFinSF2        1  0.030
BsmtUnfSF         1  0.030
TotalBsmtSF       1  0.030
GarageArea        1  0.030
########################################
```



Imputer Selection

In [ ]:
```python
def select_imputer(dataframe, target_column, additional_column, random_state=42):
    droplist = [target_column, additional_column]
    X = dataframe.drop(droplist, axis=1)
    y = dataframe[target_column]
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=ran

    impute_methods = {"mean": SimpleImputer(strategy="mean"),
                      "median": SimpleImputer(strategy="median"),
                      "most_frequent": SimpleImputer(strategy="most_frequent"),
                      "knn": KNNImputer(n_neighbors=5)}

    results = {}

    numeric_cols = X.select_dtypes(include=["int64", "float64"]).columns
    categorical_cols = X.select_dtypes(include=["object", "category"]).columns

    encoder = OneHotEncoder(handle_unknown='ignore', sparse_output=False)
```

```
    if dataframe[target_column].dtype not in ["int64", "float64"] or dataframe[target_column]
        for method_name, imputer in impute_methods.items():
            X_train_numeric = X_train[numeric_cols]
            X_test_numeric = X_test[numeric_cols]

            X_train_numeric_imputed = imputer.fit_transform(X_train_numeric)
            X_test_numeric_imputed = imputer.transform(X_test_numeric)

            X_train_categorical_imputed = X_train[categorical_cols].fillna(X_train[categorica
            X_test_categorical_imputed = X_test[categorical_cols].fillna(X_test[categorical_c

            X_train_categorical_encoded = encoder.fit_transform(X_train_categorical_imputed)
            X_test_categorical_encoded = encoder.transform(X_test_categorical_imputed)

            X_train_imputed = np.hstack([X_train_numeric_imputed, X_train_categorical_encoded
            X_test_imputed = np.hstack([X_test_numeric_imputed, X_test_categorical_encoded])

            model = RandomForestClassifier(random_state=42)
            model.fit(X_train_imputed, y_train)

            y_pred = model.predict(X_test_imputed)
            f1 = f1_score(y_test, y_pred)

            results[method_name] = f1

        for method, f1 in results.items():
            print(f"Impute method: {method}, F1-Score: {f1:.4f}")

    else:
        for method_name, imputer in impute_methods.items():
            X_train_numeric = X_train[numeric_cols]
            X_test_numeric = X_test[numeric_cols]

            X_train_numeric_imputed = imputer.fit_transform(X_train_numeric)
            X_test_numeric_imputed = imputer.transform(X_test_numeric)

            X_train_categorical_imputed = X_train[categorical_cols].fillna(X_train[categorica
            X_test_categorical_imputed = X_test[categorical_cols].fillna(X_test[categorical_c

            X_train_categorical_encoded = encoder.fit_transform(X_train_categorical_imputed)
            X_test_categorical_encoded = encoder.transform(X_test_categorical_imputed)

            X_train_imputed = np.hstack([X_train_numeric_imputed, X_train_categorical_encoded
            X_test_imputed = np.hstack([X_test_numeric_imputed, X_test_categorical_encoded])

            model = RandomForestRegressor(random_state=42)
            model.fit(X_train_imputed, y_train)

            y_pred = model.predict(X_test_imputed)
            rmse = np.sqrt(mean_squared_error(y_test, y_pred))

            results[method_name] = rmse

        for method, rmse in results.items():
            print(f"Impute method: {method}, RMSE: {rmse:.4f}")
```

In [ ]:

```
df_ = df.copy()

train_df_ = df_[df_["SalePrice"].notnull()]
test_df_ = df_[df_["SalePrice"].isnull()]

select_imputer(train_df_, "SalePrice", "Id")
```

```
Impute method: mean, RMSE: 28755.0587
Impute method: median, RMSE: 29293.7540
Impute method: most_frequent, RMSE: 29074.9876
Impute method: knn, RMSE: 28872.6535
```

> **Selected Imputation Method:** Mean

In [ ]:
```python
variables_with_na = missing_values_table(df, na_name=True, plot=False)
```

```
             n_miss  ratio
SalePrice      1459 49.980
LotFrontage     486 16.650
GarageYrBlt     159  5.450
MasVnrArea       23  0.790
BsmtFinSF1        1  0.030
BsmtFinSF2        1  0.030
BsmtUnfSF         1  0.030
TotalBsmtSF       1  0.030
GarageArea        1  0.030
##########################################
```

In [ ]:
```python
variables_with_na = [col for col in variables_with_na if "SalePrice" not in col]

variables_with_na
```

Out[ ]:
```
['LotFrontage',
 'MasVnrArea',
 'BsmtFinSF1',
 'BsmtFinSF2',
 'BsmtUnfSF',
 'TotalBsmtSF',
 'GarageYrBlt',
 'GarageArea']
```

In [ ]:
```python
for col in variables_with_na:
    df[col].fillna(df[col].mean(), inplace=True)
```

In [ ]:
```python
missing_values_table(df, plot=False)
```

```
           n_miss  ratio
SalePrice    1459 49.980
##########################################
```

> Only test set SalePrice Features missing. It is correct.

Feature Engineering

Base ML Model

In [ ]:
```python
dff = df.copy()

def label_encoder(dataframe, binary_col):
    labelencoder = LabelEncoder()
    dataframe[binary_col] = labelencoder.fit_transform(dataframe[binary_col])
    return dataframe

binary_cols = [col for col in df.columns if df[col].dtypes == "O" and len(df[col].unique()) =

for col in binary_cols:
    label_encoder(dff, col)

def one_hot_encoder(dataframe, categorical_cols, drop_first=False):
```

```
    dataframe = pd.get_dummies(dataframe, columns=categorical_cols, drop_first=drop_first, dty
    return dataframe

dff = one_hot_encoder(dff, cat_cols, drop_first=True)
dff.head()
```

Out[ ]:

	Id	MSSubClass	LotFrontage	LotArea	OverallQual	YearBuilt	YearRemodAdd	MasVnrArea	Bsm
0	1	60	65.000	8450.000	7	2003	2003.000	196.000	
1	2	20	80.000	9600.000	6	1976	1976.000	0.000	
2	3	60	68.000	11250.000	7	2001	2002.000	162.000	
3	4	70	60.000	9550.000	7	1915	1970.000	0.000	
4	5	60	84.000	14260.000	8	2000	2000.000	350.000	

In [ ]:
```
train_df = dff[dff["SalePrice"].notnull()]
test_df = dff[dff["SalePrice"].isnull()]

y = train_df["SalePrice"]
X = train_df.drop(["Id", "SalePrice"], axis=1)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20)

models = [("LR", LinearRegression()),
          ("KNN", KNeighborsRegressor()),
          ("CART", DecisionTreeRegressor()),
          ("RF", RandomForestRegressor()),
          ("SVM", SVR()),
          ("XGB", XGBRegressor()),
          ("LightGBM", LGBMRegressor(force_row_wise=True, verbose=-1)),
          ("CatBoost", CatBoostRegressor(verbose=False))]

for name, model in models:
    rmse = np.mean(np.sqrt(-cross_val_score(model, X, y, cv=5, scoring="neg_mean_squared_erro

    print(f"RMSE: {round(rmse, 4)} ({name})")
```

```
RMSE: 54327.5416 (LR)
RMSE: 45980.3326 (KNN)
RMSE: 40567.6175 (CART)
RMSE: 29701.3368 (RF)
RMSE: 81138.8683 (SVM)
RMSE: 28576.9793 (XGB)
RMSE: 29189.1229 (LightGBM)
RMSE: 25192.5422 (CatBoost)
```

> **Selected Model:** CatBoost

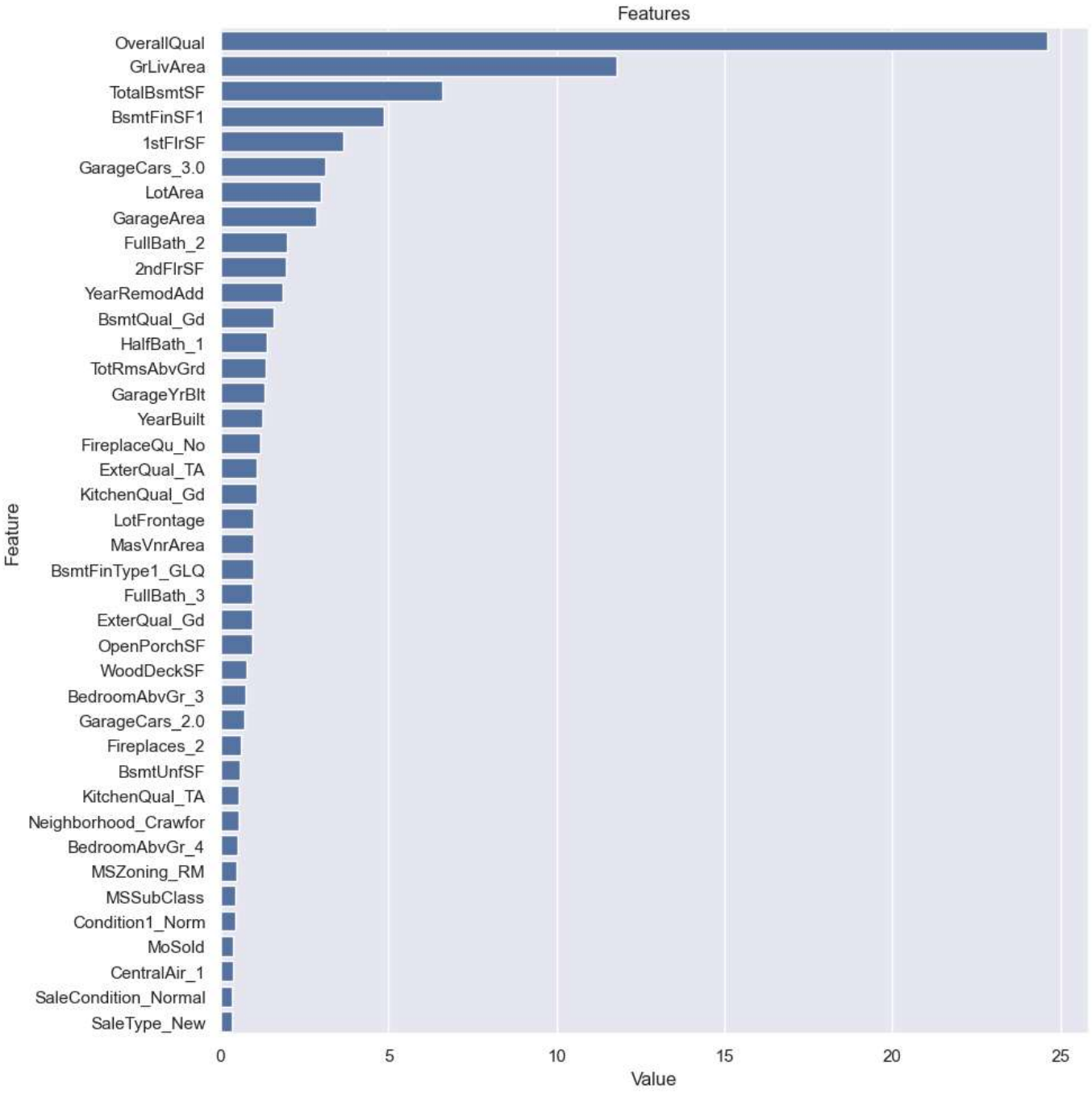
In [ ]:
```
selected_model = CatBoostRegressor(verbose=False, random_state=42).fit(X_train, y_train)
print(f"RMSE: {round(rmse, 4)} (CatBoost)")
```

```
RMSE: 25192.5422 (CatBoost)
```

Feature Importance

In [ ]:
```
def plot_importance(model, features, dataframe, save=False):
    num = len(dataframe)
    feature_imp = pd.DataFrame({"Value": model.feature_importances_, "Feature": features.colu
```

```
    plt.figure(figsize=(10, 10))
    sns.set_theme(font_scale=1)
    sns.barplot(x="Value", y="Feature", data=feature_imp.sort_values(by="Value", ascending=Fa
    plt.title("Features")
    plt.tight_layout()
    plt.show()
    if save:
        plt.savefig("importances.png")

plot_importance(selected_model, X_train, dff)
```



Feature Extraction

In [ ]:

```
def engineered_features(df):
    df_new = df.copy()

    # 1. Age of the house
    df_new["NEW_HouseAge"] = df_new["YrSold"] - df_new["YearBuilt"]

    # 2. Total square footage
    df_new["NEW_TotalSF"] = df_new["TotalBsmtSF"] + df_new["1stFlrSF"] + df_new["2ndFlrSF"]

    # 3. Total bathrooms
    df_new["NEW_TotalBathrooms"] = df_new["FullBath"] + 0.5 * df_new["HalfBath"] + df_new["Bs
```

```python
# 4. House quality score
df_new["NEW_QualityScore"] = df_new["OverallQual"] * df_new["OverallCond"]

# 5. Garage area per car
df_new["NEW_GarageAreaPerCar"] = df_new["GarageArea"] / (df_new["GarageCars"] + 1)  # Add

# 6. Total porch area
df_new["NEW_TotalPorchArea"] = df_new["OpenPorchSF"] + df_new["EnclosedPorch"] + df_new["

# 7. Has pool
df_new["NEW_HasPool"] = (df_new["PoolArea"] > 0).astype(int)

# 8. Has fireplace
df_new["NEW_HasFireplace"] = (df_new["Fireplaces"] > 0).astype(int)

# 9. Remodel age
df_new["NEW_ReModelAge"] = df_new["YrSold"] - df_new["YearRemodAdd"]

# 10. Total outdoor area
df_new["NEW_TotalOutdoorSF"] = df_new["LotArea"] - df_new["1stFlrSF"]

# 11. Living area ratio
df_new["NEW_LivingAreaRatio"] = df_new["GrLivArea"] / df_new["LotArea"]

# 12. Is new construction
df_new["NEW_IsNewConstruction"] = (df_new["YearBuilt"] == df_new["YrSold"]).astype(int)

# 13. Age when sold category
df_new["NEW_AgeSoldCategory"] = pd.cut(df_new["NEW_HouseAge"], bins=[0, 5, 10, 20, 30, 50

# 14. Total rooms above ground to total area ratio
df_new["NEW_RoomsPerArea"] = df_new["TotRmsAbvGrd"] / df_new["GrLivArea"]

# 15. Masonry veneer area percentage
df_new["NEW_MasonryVeneerPercentage"] = df_new["MasVnrArea"] / df_new["NEW_TotalSF"]

# 16. Overall material and finish quality category
df_new["NEW_OverallQualCategory"] = pd.cut(df_new["OverallQual"], bins=[0, 3, 5, 7, 10],

# 17. Z-score normalization for key metrics
key_metrics = ["GrLivArea", "TotalBsmtSF", "GarageArea", "LotArea"]

for metric in key_metrics:
    df_new[f"NEW_{metric}ZScore"] = stats.zscore(df_new[metric])

# 18. Total value of miscellaneous features
df_new["NEW_TotalMiscValue"] = df_new["MiscVal"] + df_new["PoolArea"] * 100  # Assuming $

# 19. Total SF to lot area ratio
df_new["NEW_TotalSFtoLotRatio"] = df_new["NEW_TotalSF"] / df_new["LotArea"]

# 20. Age since remodel category
df_new["NEW_RemodelAgeCategory"] = pd.cut(df_new["NEW_ReModelAge"], bins=[-1, 0, 5, 10, 2

# 21. Kitchen and bathroom score
df_new["NEW_KitchenBathScore"] = df_new["KitchenQual"].map({"Ex": 5, "Gd": 4, "TA": 3, "F

# 22. Exterior quality score
df_new["NEW_ExteriorScore"] = df_new["ExterQual"].map({"Ex": 5, "Gd": 4, "TA": 3, "Fa": 2

# 23. Total quality score (combination of various quality indicators)
df_new["NEW_TotalQualityScore"] = df_new["OverallQual"] + df_new["OverallCond"] + df_new[

# 24. Has second floor
df_new["NEW_Has2ndFloor"] = (df_new["2ndFlrSF"] > 0).astype(int)
```

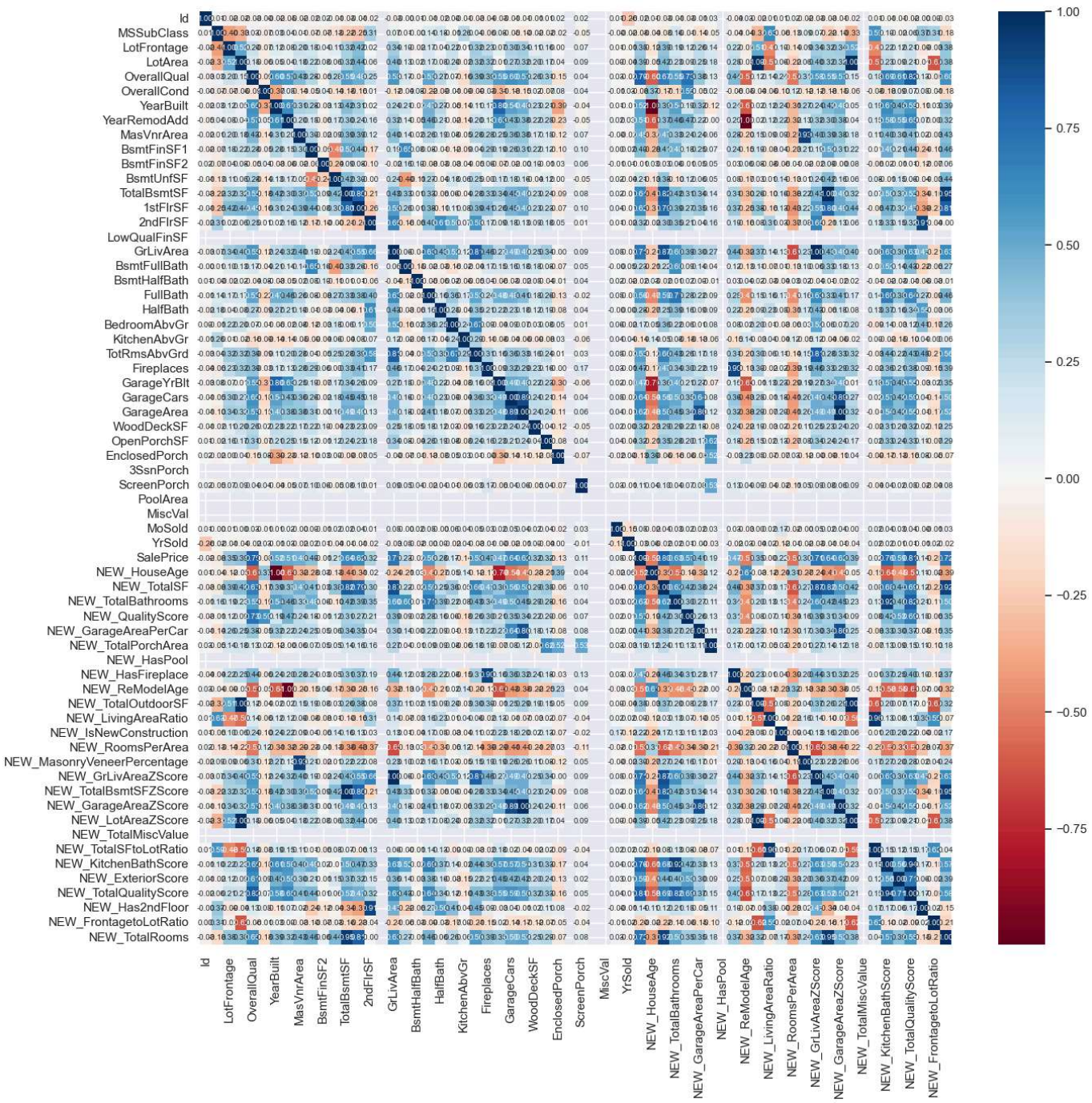
```
    # 25. Lot frontage to lot area ratio
    df_new["NEW_FrontagetoLotRatio"] = df_new["LotFrontage"] / df_new["LotArea"]

    # 26. Total number of rooms (including basement)
    df_new["NEW_TotalRooms"] = df_new["TotRmsAbvGrd"] + df_new["TotalBsmtSF"] // 100  # Assum

    return df_new
```

In [ ]:
```
df = engineered_features(df)

df.head()
```

Out[ ]:

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities
0	1	60	RL	65.000	8450.000	Pave	No	Reg	Lvl	AllPub
1	2	20	RL	80.000	9600.000	Pave	No	Reg	Lvl	AllPub
2	3	60	RL	68.000	11250.000	Pave	No	IR1	Lvl	AllPub
3	4	70	RL	60.000	9550.000	Pave	No	IR1	Lvl	AllPub
4	5	60	RL	84.000	14260.000	Pave	No	IR1	Lvl	AllPub

In [ ]:
```
drop_list = high_correlated_cols(df, plot=True, corr_th=0.90)

drop_list
```



Out[ ]:
```
['NEW_HouseAge',
 'NEW_ReModelAge',
 'NEW_TotalOutdoorSF',
 'NEW_MasonryVeneerPercentage',
 'NEW_GrLivAreaZScore',
 'NEW_TotalBsmtSFZScore',
 'NEW_GarageAreaZScore',
 'NEW_LotAreaZScore',
 'NEW_TotalSFtoLotRatio',
 'NEW_KitchenBathScore',
 'NEW_TotalQualityScore',
 'NEW_Has2ndFloor',
 'NEW_TotalRooms']
```

In [ ]:
```
df.drop(columns=drop_list, inplace=True)

df.head()
```

Out[ ]:

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities
0	1	60	RL	65.000	8450.000	Pave	No	Reg	Lvl	AllPub
1	2	20	RL	80.000	9600.000	Pave	No	Reg	Lvl	AllPub
2	3	60	RL	68.000	11250.000	Pave	No	IR1	Lvl	AllPub
3	4	70	RL	60.000	9550.000	Pave	No	IR1	Lvl	AllPub
4	5	60	RL	84.000	14260.000	Pave	No	IR1	Lvl	AllPub

Encoding

In [ ]:
```
cat_cols, num_cols, cat_but_car = grab_col_names(df, car_th=25)
print("###############")
print(f"Cat_Cols : {cat_cols}")
print("###############")
print(f"Num_Cols : {num_cols}")
print("###############")
print(f"Cat_But_Car : {cat_but_car}")
```

```
Observations: 2919
Variables: 97
cat_cols: 65
num_cols: 32
cat_but_car: 0
num_but_cat: 22
#############
Cat_Cols : ['MSZoning', 'Street', 'Alley', 'LotShape', 'LandContour', 'Utilities', 'LotConfi
g', 'LandSlope', 'Neighborhood', 'Condition1', 'Condition2', 'BldgType', 'HouseStyle', 'RoofSt
yle', 'RoofMatl', 'Exterior1st', 'Exterior2nd', 'MasVnrType', 'ExterQual', 'ExterCond', 'Found
ation', 'BsmtQual', 'BsmtCond', 'BsmtExposure', 'BsmtFinType1', 'BsmtFinType2', 'Heating', 'He
atingQC', 'CentralAir', 'Electrical', 'KitchenQual', 'Functional', 'FireplaceQu', 'GarageTyp
e', 'GarageFinish', 'GarageQual', 'GarageCond', 'PavedDrive', 'PoolQC', 'Fence', 'MiscFeatur
e', 'SaleType', 'SaleCondition', 'OverallCond', 'LowQualFinSF', 'BsmtFullBath', 'BsmtHalfBat
h', 'FullBath', 'HalfBath', 'BedroomAbvGr', 'KitchenAbvGr', 'Fireplaces', 'GarageCars', '3SsnP
orch', 'PoolArea', 'MiscVal', 'YrSold', 'NEW_HasPool', 'NEW_HasFireplace', 'NEW_IsNewConstruct
ion', 'NEW_AgeSoldCategory', 'NEW_OverallQualCategory', 'NEW_TotalMiscValue', 'NEW_RemodelAgeC
ategory', 'NEW_ExteriorScore']
#############
Num_Cols : ['Id', 'MSSubClass', 'LotFrontage', 'LotArea', 'OverallQual', 'YearBuilt', 'YearRem
odAdd', 'MasVnrArea', 'BsmtFinSF1', 'BsmtFinSF2', 'BsmtUnfSF', 'TotalBsmtSF', '1stFlrSF', '2nd
FlrSF', 'GrLivArea', 'TotRmsAbvGrd', 'GarageYrBlt', 'GarageArea', 'WoodDeckSF', 'OpenPorchSF',
'EnclosedPorch', 'ScreenPorch', 'MoSold', 'SalePrice', 'NEW_TotalSF', 'NEW_TotalBathrooms', 'N
EW_QualityScore', 'NEW_GarageAreaPerCar', 'NEW_TotalPorchArea', 'NEW_LivingAreaRatio', 'NEW_Ro
omsPerArea', 'NEW_FrontagetoLotRatio']
#############
Cat_But_Car : []
```

In [ ]:
```
num_cols = [col for col in num_cols if col not in ["Id", "SalePrice"]]

print(f"Num_Cols : {num_cols}")
```

```
Num_Cols : ['MSSubClass', 'LotFrontage', 'LotArea', 'OverallQual', 'YearBuilt', 'YearRemodAd
d', 'MasVnrArea', 'BsmtFinSF1', 'BsmtFinSF2', 'BsmtUnfSF', 'TotalBsmtSF', '1stFlrSF', '2ndFlrS
F', 'GrLivArea', 'TotRmsAbvGrd', 'GarageYrBlt', 'GarageArea', 'WoodDeckSF', 'OpenPorchSF', 'En
closedPorch', 'ScreenPorch', 'MoSold', 'NEW_TotalSF', 'NEW_TotalBathrooms', 'NEW_QualityScor
e', 'NEW_GarageAreaPerCar', 'NEW_TotalPorchArea', 'NEW_LivingAreaRatio', 'NEW_RoomsPerArea',
'NEW_FrontagetoLotRatio']
```

Label Encoding

In [ ]:
```
def label_encoder(dataframe, binary_col):
    labelencoder = LabelEncoder()
    dataframe[binary_col] = labelencoder.fit_transform(dataframe[binary_col])
    return dataframe
```

In [ ]:
```
binary_cols = [col for col in df.columns if df[col].dtypes == "O" and df[col].nunique() == 2]

for col in binary_cols:
    df = label_encoder(df, col)
```

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

Out[ ]:

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities
0	1	60	RL	65.000	8450.000	1	No	Reg	Lvl	
1	2	20	RL	80.000	9600.000	1	No	Reg	Lvl	
2	3	60	RL	68.000	11250.000	1	No	IR1	Lvl	
3	4	70	RL	60.000	9550.000	1	No	IR1	Lvl	
4	5	60	RL	84.000	14260.000	1	No	IR1	Lvl	

One-Hot Encoding

In [ ]:
```
cat_cols = [col for col in cat_cols if col not in binary_cols]
```

In [ ]:
```
def one_hot_encoder(dataframe, categorical_cols, drop_first=False):
    dataframe = pd.get_dummies(dataframe, columns=categorical_cols, dtype=int, drop_first=dro
    return dataframe
```

In [ ]:
```
df = one_hot_encoder(df, cat_cols)

df.head()
```

Out[ ]:

	Id	MSSubClass	LotFrontage	LotArea	Street	Utilities	OverallQual	YearBuilt	YearRemodAdd	M
0	1	60	65.000	8450.000	1	0	7	2003	2003.000	
1	2	20	80.000	9600.000	1	0	6	1976	1976.000	
2	3	60	68.000	11250.000	1	0	7	2001	2002.000	
3	4	70	60.000	9550.000	1	0	7	1915	1970.000	
4	5	60	84.000	14260.000	1	0	8	2000	2000.000	

Deleting Useless Cols

In [ ]:
```
useless_cols = [col for col in df.columns if df[col].nunique() == 2 and
                (df[col].value_counts() / len(df) < 0.01).any(axis=None)]

useless_cols
```

Out[ ]:
```
['Street',
 'Utilities',
 'MSZoning_C (all)',
 'MSZoning_RH',
 'LotShape_IR3',
 'LotConfig_FR3',
 'LandSlope_Sev',
 'Neighborhood_Blmngtn',
 'Neighborhood_Blueste',
 'Neighborhood_NPkVill',
 'Neighborhood_Veenker',
 'Condition1_PosA',
 'Condition1_RRAe',
 'Condition1_RRNe',
 'Condition1_RRNn',
 'Condition2_Artery',
 'Condition2_Feedr',
 'Condition2_PosA',
 'Condition2_PosN',
 'Condition2_RRAe',
 'Condition2_RRAn',
 'Condition2_RRNn',
 'HouseStyle_1.5Unf',
 'HouseStyle_2.5Fin',
 'HouseStyle_2.5Unf',
 'RoofStyle_Flat',
 'RoofStyle_Gambrel',
 'RoofStyle_Mansard',
 'RoofStyle_Shed',
 'RoofMatl_ClyTile',
 'RoofMatl_Membran',
 'RoofMatl_Metal',
 'RoofMatl_Roll',
 'RoofMatl_Tar&Grv',
 'RoofMatl_WdShake',
 'RoofMatl_WdShngl',
 'Exterior1st_AsphShn',
 'Exterior1st_BrkComm',
 'Exterior1st_CBlock',
 'Exterior1st_ImStucc',
 'Exterior1st_Stone',
 'Exterior2nd_AsphShn',
 'Exterior2nd_Brk Cmn',
 'Exterior2nd_CBlock',
 'Exterior2nd_ImStucc',
 'Exterior2nd_Other',
 'Exterior2nd_Stone',
 'MasVnrType_BrkCmn',
 'ExterCond_Ex',
 'ExterCond_Po',
 'Foundation_Stone',
 'Foundation_Wood',
 'BsmtCond_Po',
 'Heating_Floor',
 'Heating_GasW',
 'Heating_Grav',
 'Heating_OthW',
 'Heating_Wall',
 'HeatingQC_Po',
 'Electrical_FuseP',
 'Electrical_Mix',
 'Functional_Maj1',
 'Functional_Maj2',
 'Functional_Sev',
 'GarageType_2Types',
```

```
 'GarageType_CarPort',
 'GarageQual_Ex',
 'GarageQual_Gd',
 'GarageQual_Po',
 'GarageCond_Ex',
 'GarageCond_Gd',
 'GarageCond_Po',
 'PoolQC_Ex',
 'PoolQC_Fa',
 'PoolQC_Gd',
 'PoolQC_No',
 'Fence_MnWw',
 'MiscFeature_Gar2',
 'MiscFeature_Othr',
 'MiscFeature_TenC',
 'SaleType_CWD',
 'SaleType_Con',
 'SaleType_ConLD',
 'SaleType_ConLI',
 'SaleType_ConLw',
 'SaleType_Oth',
 'SaleCondition_AdjLand',
 'SaleCondition_Alloca',
 'OverallCond_1',
 'OverallCond_2',
 'BsmtFullBath_3.0',
 'BsmtHalfBath_2.0',
 'FullBath_0',
 'FullBath_4',
 'HalfBath_2',
 'BedroomAbvGr_0',
 'BedroomAbvGr_6',
 'BedroomAbvGr_8',
 'KitchenAbvGr_0',
 'KitchenAbvGr_3',
 'Fireplaces_3',
 'Fireplaces_4',
 'GarageCars_4.0',
 'GarageCars_5.0',
 'NEW_ExteriorScore_3',
 'NEW_ExteriorScore_4',
 'NEW_ExteriorScore_9',
 'NEW_ExteriorScore_10']
```

In [ ]:
```
df.drop(columns=useless_cols, axis=1, inplace=True)

df.head()
```

Out[ ]:

	Id	MSSubClass	LotFrontage	LotArea	OverallQual	YearBuilt	YearRemodAdd	MasVnrArea	Bsm
0	1	60	65.000	8450.000	7	2003	2003.000	196.000	
1	2	20	80.000	9600.000	6	1976	1976.000	0.000	
2	3	60	68.000	11250.000	7	2001	2002.000	162.000	
3	4	70	60.000	9550.000	7	1915	1970.000	0.000	
4	5	60	84.000	14260.000	8	2000	2000.000	350.000	

Standardization

In [ ]:
```
cat_cols, num_cols, cat_but_car = grab_col_names(df, car_th=25)

num_cols = [col for col in num_cols if col not in ["Id", "SalePrice"]]

print(f"Num_Cols : {num_cols}")
```

```
Observations: 2919
Variables: 272
cat_cols: 240
num_cols: 32
cat_but_car: 0
num_but_cat: 240
Num_Cols : ['MSSubClass', 'LotFrontage', 'LotArea', 'OverallQual', 'YearBuilt', 'YearRemodAd
d', 'MasVnrArea', 'BsmtFinSF1', 'BsmtFinSF2', 'BsmtUnfSF', 'TotalBsmtSF', '1stFlrSF', '2ndFlrS
F', 'GrLivArea', 'TotRmsAbvGrd', 'GarageYrBlt', 'GarageArea', 'WoodDeckSF', 'OpenPorchSF', 'En
closedPorch', 'ScreenPorch', 'MoSold', 'NEW_TotalSF', 'NEW_TotalBathrooms', 'NEW_QualityScor
e', 'NEW_GarageAreaPerCar', 'NEW_TotalPorchArea', 'NEW_LivingAreaRatio', 'NEW_RoomsPerArea',
'NEW_FrontagetoLotRatio']
```

In [ ]:
```
scaler = RobustScaler()
df[num_cols] = scaler.fit_transform(df[num_cols])
```

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

Out[ ]:

	Id	MSSubClass	LotFrontage	LotArea	OverallQual	YearBuilt	YearRemodAdd	MasVnrArea	BsmtF
0	1	0.200	-0.235	-0.245	0.500	0.632	0.256	1.199	
1	2	-0.600	0.598	0.036	0.000	0.063	-0.436	0.000	
2	3	0.200	-0.068	0.439	0.500	0.589	0.231	0.991	
3	4	0.400	-0.513	0.024	0.500	-1.221	-0.590	0.000	-
4	5	0.200	0.821	1.175	1.000	0.568	0.179	2.141	

ML Model

Target Mean & Std

In [ ]:
```
print(f"Sale Price Mean:{df['SalePrice'].mean()}")
print(f"Sale Price Std:{df['SalePrice'].std()}")
```

```
Sale Price Mean:180921.19589041095
Sale Price Std:79442.50288288662
```

Train-Test Split

In [ ]:
```
train_df = df[df["SalePrice"].notnull()]
test_df = df[df["SalePrice"].isnull()]

y = np.log1p(train_df["SalePrice"])
X = train_df.drop(["Id", "SalePrice"], axis=1)
```

Model Selection

In [ ]:
```
catboost = CatBoostRegressor(verbose=False, random_state=42)
```

Hyperparameter Optimization

In [ ]:
```
catboost_params = {"iterations": [200, 500],
                   "learning_rate": [0.01, 0.1],
                   "depth": [3, 6]}
```

In [ ]:
```
catboost_best = GridSearchCV(catboost,
                             catboost_params,
                             cv=3,
                             n_jobs=-1,
                             verbose=True).fit(X, y)
```

```
Fitting 3 folds for each of 8 candidates, totalling 24 fits
```

In [ ]:
```
catboost_best.best_params_
```

Out[ ]:
```
{'depth': 6, 'iterations': 500, 'learning_rate': 0.1}
```

In [ ]:
```
final_model = catboost.set_params(**catboost_best.best_params_).fit(X, y)
```

In [ ]:
```
def rmse_scorer(estimator, X, y):
    y_pred = estimator.predict(X)
    y_pred = np.expm1(y_pred)
    y_true = np.expm1(y)
    return -np.sqrt(mean_squared_error(y_true, y_pred))
```

In [ ]:
```
scores = cross_val_score(final_model, X, y, cv=5, scoring=rmse_scorer)

print(f"Cross-validation RMSE scores: {-scores}")
print(f"Mean RMSE: {-np.mean(scores)}")
```

```
Cross-validation RMSE scores: [22520.37096204 26880.8831089  28033.94004759 19938.39449019
 30089.16514826]
Mean RMSE: 25492.550751397615
```

In [ ]:
```
def calculate_metrics(y_true, y_pred):
    mae = mean_absolute_error(y_true, y_pred)
    mse = mean_squared_error(y_true, y_pred)
    rmse = np.sqrt(mse)
    r2 = r2_score(y_true, y_pred)
    return mae, mse, rmse, r2

def evaluate_percentiles(y_true, y_pred):
    percentiles = [5, 25, 50, 75, 95, 100]
    results = {}
    for percentile in percentiles:
        threshold = np.percentile(y_true, percentile)
        indices = y_true <= threshold
        filtered_y_true = y_true[indices]
        filtered_y_pred = y_pred[indices]
        mae, mse, rmse, r2 = calculate_metrics(filtered_y_true, filtered_y_pred)
        results[percentile] = (mae, mse, rmse, r2)
    return results

def print_results(results):
    for percentile, metrics in results.items():
        mae, mse, rmse, r2 = metrics
        print(f"Performance for {percentile}th Percentile:")
        print(f"  Mean Absolute Error (MAE): {mae}")
        print(f"  Mean Squared Error (MSE): {mse}")
        print(f"  Root Mean Squared Error (RMSE): {rmse}")
        print(f"  R-squared (R²): {r2}")
```

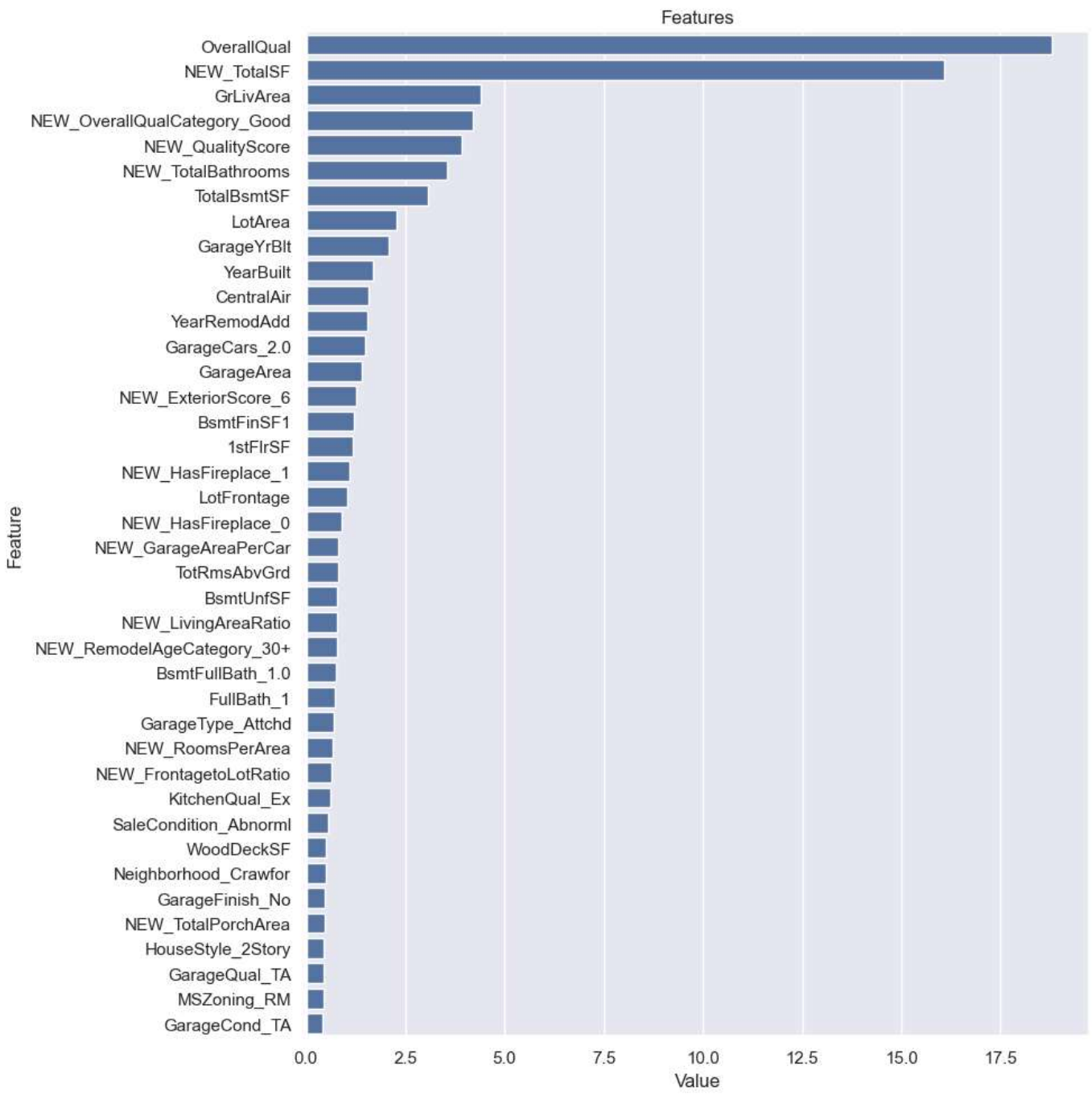
In [ ]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20)

catboost_ = CatBoostRegressor(verbose=False, random_state=42)
```

```
final_model_ = catboost_.set_params(**catboost_best.best_params_).fit(X_train, y_train)

y_pred = final_model_.predict(X_test)
y_pred = np.expm1(y_pred)
y_true = np.expm1(y_test)

results = evaluate_percentiles(y_true, y_pred)

print_results(results)
```

```
Performance for 5th Percentile:
  Mean Absolute Error (MAE): 14274.5055866363
  Mean Squared Error (MSE): 432709022.3548505
  Root Mean Squared Error (RMSE): 20801.659125051792
  R-squared (R²): -1.9099734221238647
Performance for 25th Percentile:
  Mean Absolute Error (MAE): 10154.529781887373
  Mean Squared Error (MSE): 187545178.44694406
  Root Mean Squared Error (RMSE): 13694.713521901218
  R-squared (R²): 0.48386321478056116
Performance for 50th Percentile:
  Mean Absolute Error (MAE): 11729.708259999812
  Mean Squared Error (MSE): 501488017.39365536
  Root Mean Squared Error (RMSE): 22393.928136744016
  R-squared (R²): 0.059011257189576116
Performance for 75th Percentile:
  Mean Absolute Error (MAE): 11505.18079756062
  Mean Squared Error (MSE): 401182000.2433673
  Root Mean Squared Error (RMSE): 20029.528208207186
  R-squared (R²): 0.655832450508359
Performance for 95th Percentile:
  Mean Absolute Error (MAE): 13026.59516822559
  Mean Squared Error (MSE): 431890209.97390646
  Root Mean Squared Error (RMSE): 20781.96838545152
  R-squared (R²): 0.8721833488221937
Performance for 100th Percentile:
  Mean Absolute Error (MAE): 15351.371303393416
  Mean Squared Error (MSE): 768464817.1379036
  Root Mean Squared Error (RMSE): 27721.197974436524
  R-squared (R²): 0.8951764494601102
```

Feature Importance

In [ ]:
```
def plot_importance(model, features, dataframe, save=False):
    num = len(dataframe)
    feature_imp = pd.DataFrame({"Value": model.feature_importances_, "Feature": features.colu
    plt.figure(figsize=(10, 10))
    sns.set_theme(font_scale=1)
    sns.barplot(x="Value", y="Feature", data=feature_imp.sort_values(by="Value", ascending=Fa
    plt.title("Features")
    plt.tight_layout()
    plt.show()
    if save:
        plt.savefig("importances.png")

plot_importance(final_model, X, df)
```


Features
OverallQual
NEW_TotalSF
GrLivArea
NEW_OverallQualCategory_Good
NEW_QualityScore
NEW_TotalBathrooms
TotalBsmtSF
LotArea
GarageYrBlt
YearBuilt
CentralAir
YearRemodAdd
GarageCars_2.0
GarageArea
NEW_ExteriorScore_6
BsmtFinSF1
1stFlrSF
NEW_HasFireplace_1
LotFrontage
NEW_HasFireplace_0
NEW_GarageAreaPerCar
TotRmsAbvGrd
BsmtUnfSF
NEW_LivingAreaRatio
NEW_RemodelAgeCategory_30+
BsmtFullBath_1.0
FullBath_1
GarageType_Attchd
NEW_RoomsPerArea
NEW_FrontagetoLotRatio
KitchenQual_Ex
SaleCondition_Abnorml
WoodDeckSF
Neighborhood_Crawfor
GarageFinish_No
NEW_TotalPorchArea
HouseStyle_2Story
GarageQual_TA
MSZoning_RM
GarageCond_TA
Feature
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
Value