House Price Prediction

Importing libraries

In [1]:
```
from sklearn.datasets import load_boston
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression

import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np

import statsmodels.api as sm
from statsmodels.stats.outliers_influence import variance_inflation_factor


%matplotlib inline
```

Reading Exploring Dataset

In [183]:
```
data = pd.read_csv('housing.csv')
```

In [184]:
```
# shape
print(data.shape)
```

```
(1460, 81)
```

In [185]:
```
data.dtypes
```

Out[185]:
```
Id                 int64
MSSubClass         int64
MSZoning          object
LotFrontage      float64
LotArea            int64
                  ...
MoSold             int64
YrSold             int64
SaleType          object
SaleCondition     object
SalePrice          int64
Length: 81, dtype: object
```

In [173]:
```
#del data['Id']
```

In [186]:
```
data.dtypes
```

Out[186]:
```
Id                 int64
MSSubClass         int64
MSZoning          object
LotFrontage      float64
LotArea            int64
                  ...
MoSold             int64
YrSold             int64
SaleType          object
SaleCondition     object
SalePrice          int64
Length: 81, dtype: object
```

In [187]:

```
# descriptions
print(data.describe())
```

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

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

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

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

[8 rows x 38 columns]
```

In [188]:

```
# Checking for missing values
data.isna().sum()
```

Out[188]:

```
Id                 0
MSSubClass         0
MSZoning           0
LotFrontage      259
LotArea            0
                ...
MoSold             0
YrSold             0
SaleType           0
SaleCondition      0
SalePrice          0
Length: 81, dtype: int64
```

In [189]:

```
# Checking for missing values another method
pd.isnull(data).any()
```

Out[189]:

```
Id               False
MSSubClass       False
MSZoning         False
LotFrontage       True
LotArea          False
                 ...
MoSold           False
YrSold           False
SaleType         False
SaleCondition    False
SalePrice        False
Length: 81, dtype: bool
```

In [190]:

```
data=data.fillna(" ")
```

In [191]:

```
data.info()
```

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

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

In [192]:
```
data.columns
```

Out[192]:
```
Index(['Id', 'MSSubClass', 'MSZoning', 'LotFrontage', 'LotArea', 'Street',
       'Alley', 'LotShape', 'LandContour', 'Utilities', 'LotConfig',
       'LandSlope', 'Neighborhood', 'Condition1', 'Condition2', 'BldgType',
       'HouseStyle', 'OverallQual', 'OverallCond', 'YearBuilt', 'YearRemodA
dd',
       'RoofStyle', 'RoofMatl', 'Exterior1st', 'Exterior2nd', 'MasVnrType',
       'MasVnrArea', 'ExterQual', 'ExterCond', 'Foundation', 'BsmtQual',
       'BsmtCond', 'BsmtExposure', 'BsmtFinType1', 'BsmtFinSF1',
       'BsmtFinType2', 'BsmtFinSF2', 'BsmtUnfSF', 'TotalBsmtSF', 'Heating',
       'HeatingQC', 'CentralAir', 'Electrical', '1stFlrSF', '2ndFlrSF',
       'LowQualFinSF', 'GrLivArea', 'BsmtFullBath', 'BsmtHalfBath', 'FullBa
th',
       'HalfBath', 'BedroomAbvGr', 'KitchenAbvGr', 'KitchenQual',
       'TotRmsAbvGrd', 'Functional', 'Fireplaces', 'FireplaceQu', 'GarageTy
pe',
       'GarageYrBlt', 'GarageFinish', 'GarageCars', 'GarageArea', 'GarageQu
al',
       'GarageCond', 'PavedDrive', 'WoodDeckSF', 'OpenPorchSF',
       'EnclosedPorch', '3SsnPorch', 'ScreenPorch', 'PoolArea', 'PoolQC',
       'Fence', 'MiscFeature', 'MiscVal', 'MoSold', 'YrSold', 'SaleType',
       'SaleCondition', 'SalePrice'],
      dtype='object')
```

In [193]:
```
columns = ['Id','MSSubClass', 'LotFrontage', 'LotArea', 'Street', 'Alley','Lo
'LandSlope','Neighborhood', 'Condition1', 'Condition2','OverallQual', 'YearRe
'Exterior1st', 'Exterior2nd', 'MasVnrType', 'MasVnrArea','ExterQual', 'ExterC
'BsmtExposure', 'BsmtFinType1', 'BsmtFinSF1', 'BsmtFinType2','BsmtFinSF2', 'B
'HeatingQC','CentralAir', 'Electrical', '1stFlrSF', '2ndFlrSF', 'LowQualFinSF
'BsmtHalfBath','BedroomAbvGr', 'KitchenAbvGr', 'KitchenQual', 'TotRmsAbvGrd',
'GarageType', 'GarageYrBlt','GarageFinish','Fence','GarageArea', 'GarageQual'
'EnclosedPorch', '3SsnPorch','ScreenPorch', 'PoolArea', 'PoolQC',  'MiscFeatu
data.drop(columns, inplace=True, axis=1)
```

In [194]:
```
data.columns
```

Out[194]:
```
Index(['MSZoning', 'BldgType', 'HouseStyle', 'OverallCond', 'YearBuilt',
       'FullBath', 'HalfBath', 'GarageCars', 'PavedDrive', 'SaleCondition',
       'SalePrice'],
      dtype='object')
```

In [195]:
```
data.count()
```

Out[195]:
```
MSZoning         1460
BldgType         1460
HouseStyle       1460
OverallCond      1460
YearBuilt        1460
FullBath         1460
HalfBath         1460
GarageCars       1460
PavedDrive       1460
SaleCondition    1460
SalePrice        1460
dtype: int64
```

In [196]:
```
data.shape
```

Out[196]:
```
(1460, 11)
```

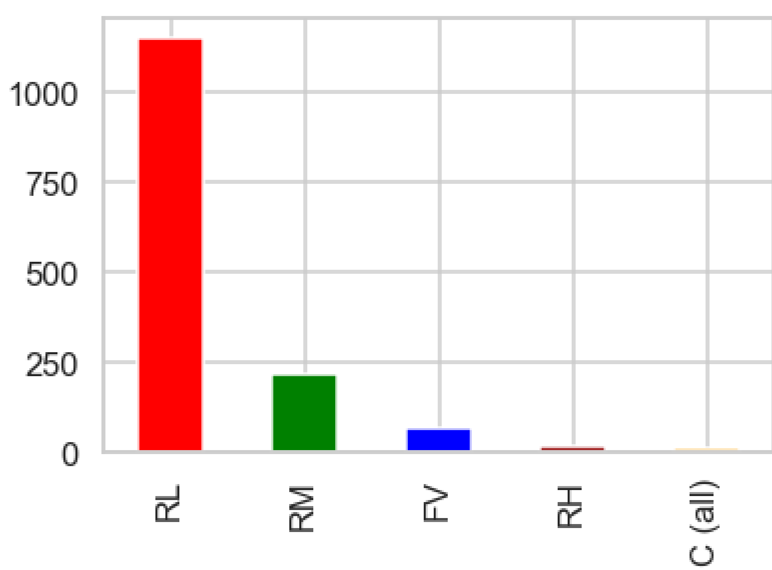
to know unique values in the row

In [197]:
```
data.MSZoning.unique()
```

Out[197]:
```
array(['RL', 'RM', 'C (all)', 'FV', 'RH'], dtype=object)
```

In [198]:
```
#'RL', 'RM', 'C (all)', 'FV', 'RH'
#"red", "green","blue","brown","orange"
data.MSZoning.value_counts().plot(kind="bar", color=["red", "green","blue","b
```

Out[198]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1511dc63cc8>
```

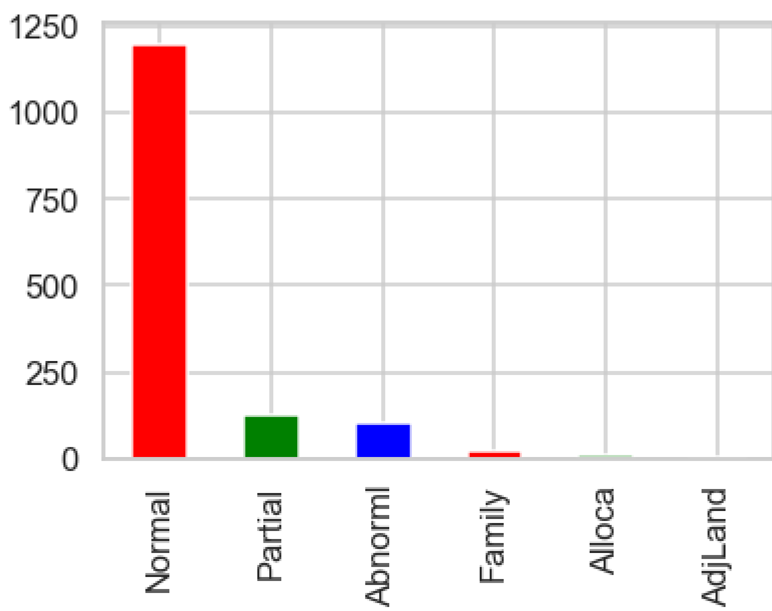


In [199]:
```
data.SaleCondition.unique()
```

Out[199]:
```
array(['Normal', 'Abnorml', 'Partial', 'AdjLand', 'Alloca', 'Family'],
      dtype=object)
```

In [200]:
```
#'Normal', 'Abnorml', 'Partial', 'AdjLand', 'Alloca', 'Family'
#"red", "green","blue"
data.SaleCondition.value_counts().plot(kind="bar", color=["red", "green","blu
```

Out[200]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1511e441c48>
```



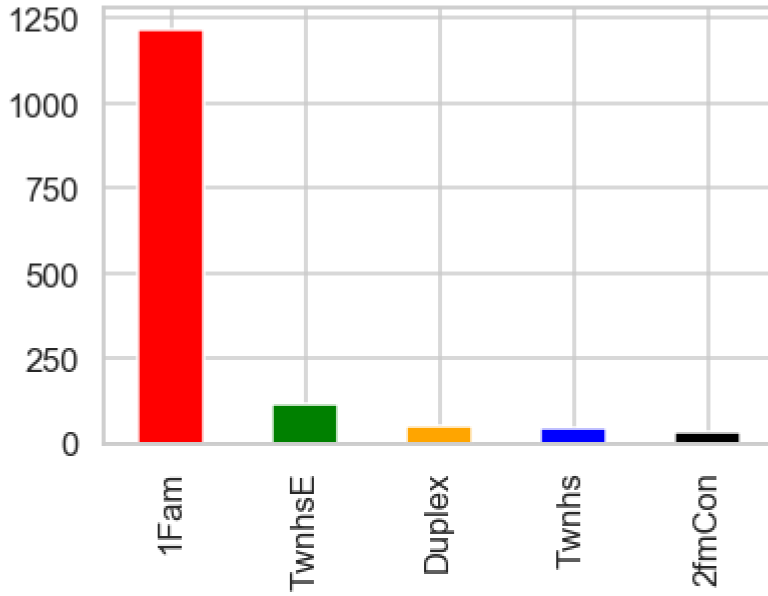
In [201]:
```
data.BldgType.unique()
```

Out[201]:
```
array(['1Fam', '2fmCon', 'Duplex', 'TwnhsE', 'Twnhs'], dtype=object)
```

In [202]:

```
#'1Fam', '2fmCon', 'Duplex', 'TwnhsE', 'Twnhs'
# "red", "green","orange","blue","black"
data.BldgType.value_counts().plot(kind="bar", color=["red", "green","orange",
```

Out[202]: <matplotlib.axes._subplots.AxesSubplot at 0x1511e400508>



In [203]:

```
data.HouseStyle.unique()
```

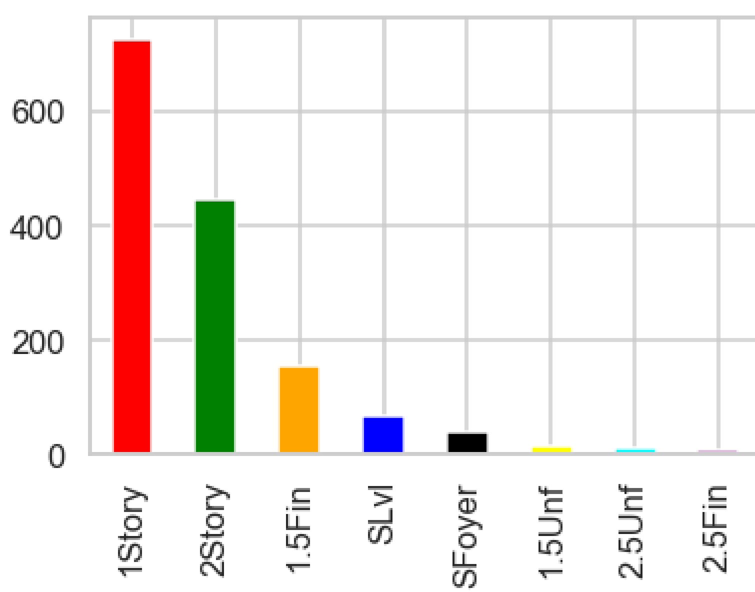
Out[203]:

```
array(['2Story', '1Story', '1.5Fin', '1.5Unf', 'SFoyer', 'SLvl', '2.5Unf',
       '2.5Fin'], dtype=object)
```

In [204]:

```
#'2Story', '1Story', '1.5Fin', '1.5Unf', 'SFoyer', 'SLvl', '2.5Unf','2.5Fin'
#"red", "green","orange","blue","black","yellow","cyan","purple"
data.HouseStyle.value_counts().plot(kind="bar", color=["red", "green","orange
```

Out[204]: <matplotlib.axes._subplots.AxesSubplot at 0x1511f1bca48>

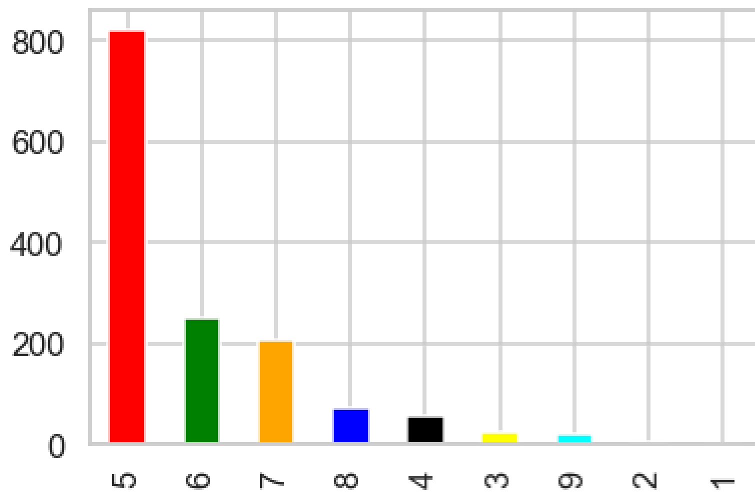


In [205]:
```
data.OverallCond.unique()
```

Out[205]: array([5, 8, 6, 7, 4, 2, 3, 9, 1], dtype=int64)

In [206]:
```
#5, 8, 6, 7, 4, 2, 3, 9, 1
#"red", "green","orange","blue","black","yellow","cyan","purple","brown"
data.OverallCond.value_counts().plot(kind="bar", color=["red", "green","orang
```

Out[206]: <matplotlib.axes._subplots.AxesSubplot at 0x1511e675d48>

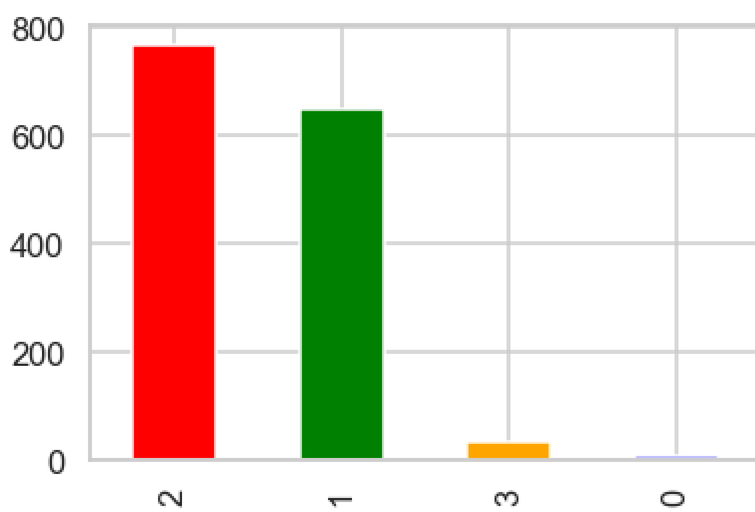


In [207]:
```
data.FullBath.unique()
```

Out[207]: array([2, 1, 3, 0], dtype=int64)

In [208]:
```
#2, 1, 3, 0
#"red", "green","orange","blue"
data.FullBath.value_counts().plot(kind="bar", color=["red", "green","orange",
```

Out[208]: <matplotlib.axes._subplots.AxesSubplot at 0x1511ec163c8>

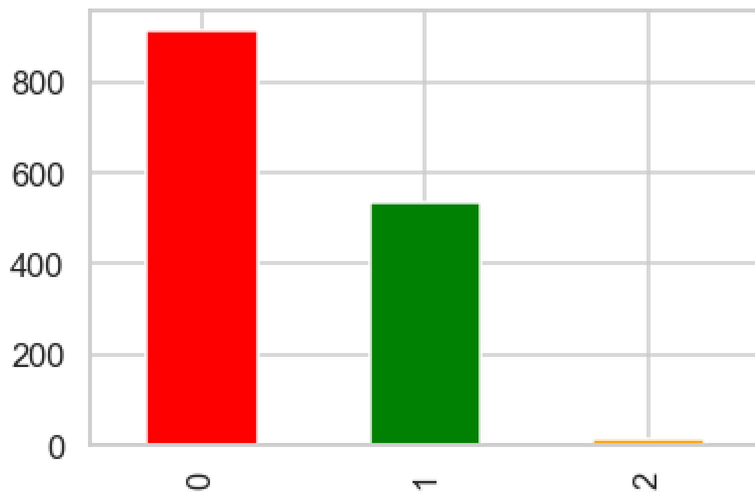


In [209]:
```
data.HalfBath.unique()
```

Out[209]: array([1, 0, 2], dtype=int64)

In [210]:
```
#1, 0, 2
#"red", "green","orange"
data.HalfBath.value_counts().plot(kind="bar", color=["red", "green","orange"]
```

Out[210]: <matplotlib.axes._subplots.AxesSubplot at 0x1511e3f8508>

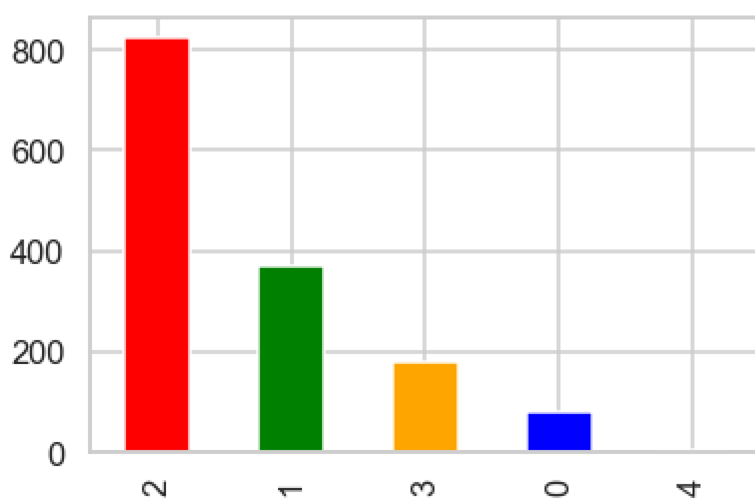


In [211]:
```
data.GarageCars.unique()
```

Out[211]: array([2, 3, 1, 0, 4], dtype=int64)

In [212]:
```
#2, 3, 1, 0, 4
#"red", "green","orange","blue","yellow"
data.GarageCars.value_counts().plot(kind="bar", color=["red", "green","orange
```

Out[212]: <matplotlib.axes._subplots.AxesSubplot at 0x1511f31c0c8>

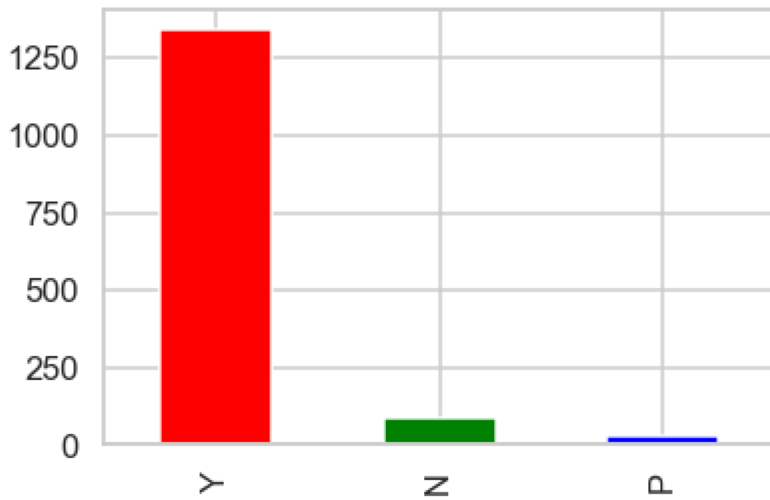


In [213]:
```
data.PavedDrive.unique()
```

Out[213]: array(['Y', 'N', 'P'], dtype=object)

In [214]:
```
#'Y', 'N', 'P'
#"red", "green","blue"
data.PavedDrive.value_counts().plot(kind="bar", color=["red", "green","blue"]
```

Out[214]: <matplotlib.axes._subplots.AxesSubplot at 0x1511df67888>



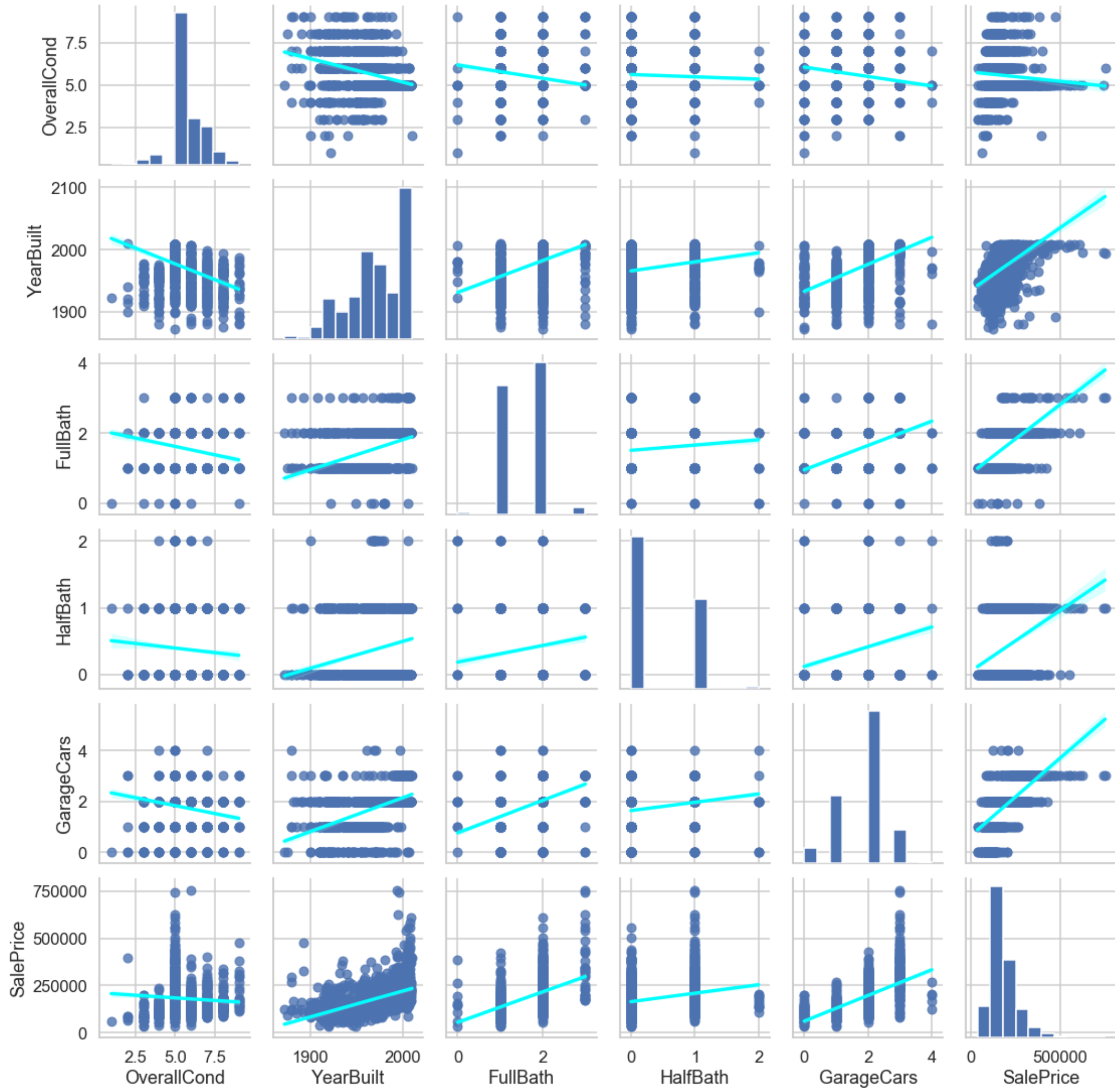
In [215]:
```
data.head(10)
```

Out[215]:

	MSZoning	BldgType	HouseStyle	OverallCond	YearBuilt	FullBath	HalfBath	GarageCars
0	RL	1Fam	2Story	5	2003	2	1	2
1	RL	1Fam	1Story	8	1976	2	0	2
2	RL	1Fam	2Story	5	2001	2	1	2
3	RL	1Fam	2Story	5	1915	1	0	3
4	RL	1Fam	2Story	5	2000	2	1	3
5	RL	1Fam	1.5Fin	5	1993	1	1	2
6	RL	1Fam	1Story	5	2004	2	0	2
7	RL	1Fam	2Story	6	1973	2	1	2
8	RM	1Fam	1.5Fin	5	1931	2	0	2
9	RL	2fmCon	1.5Unf	6	1939	1	0	1

Visualising Data - Histograms, Distributions and Bar Charts

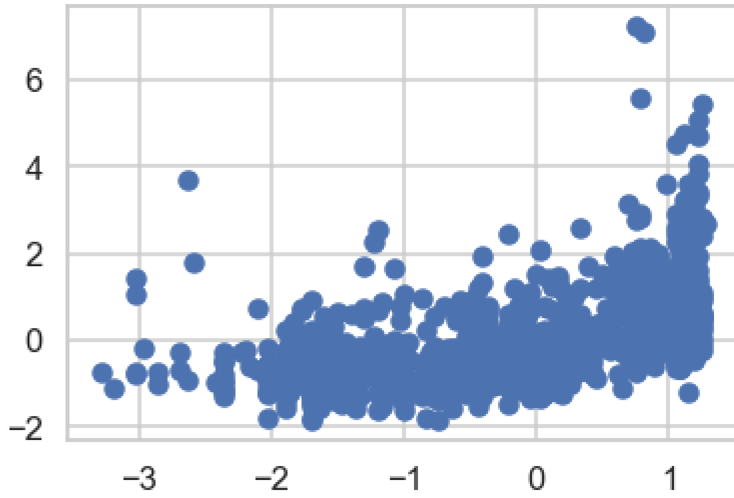
In [216]:
```
%%time
sns.pairplot(data, kind='reg', plot_kws={'line_kws':{'color': 'cyan'}})
plt.show()
```




```
Wall time: 14.4 s
```

In [298]:

```
#scatterplot visualisation
plt.scatter(x=data['YearBuilt'],y=data['SalePrice'])
ax =plt.gca()
ax.get_yaxis().get_major_formatter().set_scientific(False)
plt.draw()
```



In [217]:

```
#counting
data['FullBath'].value_counts()
```

Out[217]:

```
2    768
1    650
3     33
0      9
Name: FullBath, dtype: int64
```

In [218]:
```python
# count by
data['BldgType'].value_counts()

#print(data.groupby('BldgType').size())   #this is another method
```

Out[218]:
```
1Fam      1220
TwnhsE     114
Duplex      52
Twnhs       43
2fmCon      31
Name: BldgType, dtype: int64
```

In [219]:

```
#non_numeric(categorical) and numeric(continous)
non_numeric = []
numeric = []
for column in data.columns:
    print('==============================')
    print(f"{column} : {data[column].unique()}")
    if len(data[column].unique()) <= 10:
        non_numeric.append(column)
    else:
        numeric.append(column)
```

```
==============================
MSZoning : ['RL' 'RM' 'C (all)' 'FV' 'RH']
==============================
BldgType : ['1Fam' '2fmCon' 'Duplex' 'TwnhsE' 'Twnhs']
==============================
HouseStyle : ['2Story' '1Story' '1.5Fin' '1.5Unf' 'SFoyer' 'SLvl' '2.5Un
f' '2.5Fin']
==============================
OverallCond : [5 8 6 7 4 2 3 9 1]
==============================
YearBuilt : [2003 1976 2001 1915 2000 1993 2004 1973 1931 1939 1965 2005
1962 2006
 1960 1929 1970 1967 1958 1930 2002 1968 2007 1951 1957 1927 1920 1966
 1959 1994 1954 1953 1955 1983 1975 1997 1934 1963 1981 1964 1999 1972
 1921 1945 1982 1998 1956 1948 1910 1995 1991 2009 1950 1961 1977 1985
 1979 1885 1919 1990 1969 1935 1988 1971 1952 1936 1923 1924 1984 1926
 1940 1941 1987 1986 2008 1908 1892 1916 1932 1918 1912 1947 1925 1900
 1980 1989 1992 1949 1880 1928 1978 1922 1996 2010 1946 1913 1937 1942
 1938 1974 1893 1914 1906 1890 1898 1904 1882 1875 1911 1917 1872 1905]
==============================
FullBath : [2 1 3 0]
==============================
HalfBath : [1 0 2]
==============================
GarageCars : [2 3 1 0 4]
==============================
PavedDrive : ['Y' 'N' 'P']
==============================
SaleCondition : ['Normal' 'Abnorml' 'Partial' 'AdjLand' 'Alloca' 'Famil
y']
==============================
SalePrice : [208500 181500 223500 140000 250000 143000 307000 200000 129
900 118000
 129500 345000 144000 279500 157000 132000 149000  90000 159000 139000
 325300 139400 230000 154000 256300 134800 306000 207500  68500  40000
 149350 179900 165500 277500 309000 145000 153000 109000  82000 160000
 170000 130250 141000 319900 239686 249700 113000 127000 177000 114500
 110000 385000 130000 180500 172500 196500 438780 124900 158000 101000
 202500 219500 317000 180000 226000  80000 225000 244000 185000 144900
 107400  91000 135750 136500 193500 153500 245000 126500 168500 260000
 174000 164500  85000 123600 109900  98600 163500 133900 204750 214000
  94750  83000 128950 205000 178000 118964 198900 169500 100000 115000
 190000 136900 383970 217000 259500 176000 155000 320000 163990 136000
 153900 181000  84500 128000  87000 150000 150750 220000 171000 231500
 166000 204000 125000 105000 222500 122000 372402 235000  79000 109500
```

```
 269500 254900 162500 412500 103200 152000 127500 325624 183500 228000
 128500 215000 239000 163000 184000 243000 211000 501837 200100 120000
 475000 173000 135000 153337 286000 315000 192000 148500 311872 104000
 274900 171500 112000 143900 277000  98000 186000 252678 156000 161750
 134450 210000 107000 311500 167240 204900  97000 386250 290000 106000
 192500 148000 403000  94500 128200 216500  89500 185500 194500 318000
 262500 110500 241500 137000  76500 276000 151000  73000 175500 179500
 120500 266000 124500 201000 415298 228500 244600 179200 164700  88000
 153575 233230 135900 131000 167000 142500 175000 158500 267000 149900
 295000 305900  82500 360000 165600 119900 375000 188500 270000 187500
 342643 354000 301000 126175 242000 324000 145250 214500  78000 119000
 284000 207000 228950 377426 202900  87500 140200 151500 157500 437154
 318061  95000 105900 177500 134000 280000 198500 147000 165000 162000
 172400 134432 123000  61000 340000 394432 179000 187750 213500  76000
 240000  81000 191000 426000 106500 129000  67000 241000 245500 164990
 108000 258000 168000 339750  60000 222000 181134 149500 126000 142000
 206300 275000 109008 195400  85400  79900 122500 212000 116000  90350
 555000 162900 199900 119500 188000 256000 161000 263435  62383 188700
 124000 178740 146500 187000 440000 251000 132500 208900 380000 297000
  89471 326000 374000 164000  86000 133000 172785  91300  34900 430000
 226700 289000 208300 164900 202665  96500 402861 265000 234000 106250
 184750 315750 446261 200624 107500  39300 111250 272000 248000 213250
 179665 229000 263000 112500 255500 121500 268000 325000 316600 135960
 142600 224500 118500 146000 131500 181900 253293 369900  79500 185900
 451950 138000 319000 114504 194201 217500 221000 359100 313000 261500
  75500 137500 183200 105500 314813 305000 165150 139900 209500  93000
 264561 274000 370878 143250  98300 205950 350000 145500  97500 197900
 402000 423000 230500 173500 103600 257500 372500 159434 285000 227875
 148800 392000 194700 755000 335000 108480 141500  89000 123500 138500
 196000 312500 361919 213000  55000 302000 254000 179540  52000 102776
 189000 130500 159500 341000 103000 236500 131400  93500 239900 299800
 236000 265979 260400 275500 158900 179400 215200 337000 264132 216837
 538000 134900 102000 395000 221500 175900 187100 161500 233000 107900
 160200 146800 269790 143500 485000 582933 227680 135500 159950 144500
  55993 157900 224900 271000 224000 183000 139500 232600 147400 237000
 139950 174900 133500 189950 250580 248900 169000 200500  66500 303477
 132250 328900 122900 154500 118858 142953 611657 125500 255000 154300
 173733  75000  35311 238000 176500 145900 169990 193000 117500 184900
 253000 239799 244400 150900 197500 172000 116500 214900 178900  37900
  99500 182000 167500  85500 178400 336000 159895 255900 117000 395192
 195000 197000 348000 173900 337500 121600 206000 232000 136905 119200
 227000 203000 213490 194000 287000 293077 310000 119750  84000 315500
 262280 278000 139600 556581  84900 176485 200141 185850 328000 167900
 151400  91500 138800 155900  83500 252000  92900 176432 274725 134500
 184100 133700 118400 212900 163900 259000 239500  94000 424870 174500
 116900 201800 218000 235128 108959 233170 245350 625000 171900 154900
 392500 745000 186700 104900 262000 219210 116050 271900 229456  80500
 137900 367294 101800 138887 265900 248328 465000 186500 169900 171750
 294000 165400 301500  99900 128900 183900 378500 381000 185750  68400
 150500 281000 333168 206900 295493 111000 156500  72500  52500 155835
 108500 283463 410000 156932 144152 216000 274300 466500  58500 237500
 377500 246578 281213 137450 193879 282922 257000 223000 274970 182900
 192140 143750  64500 394617 149700 149300 121000 179600  92000 287090
 266500 142125 147500]
```

In [220]:
```
non_numeric
```

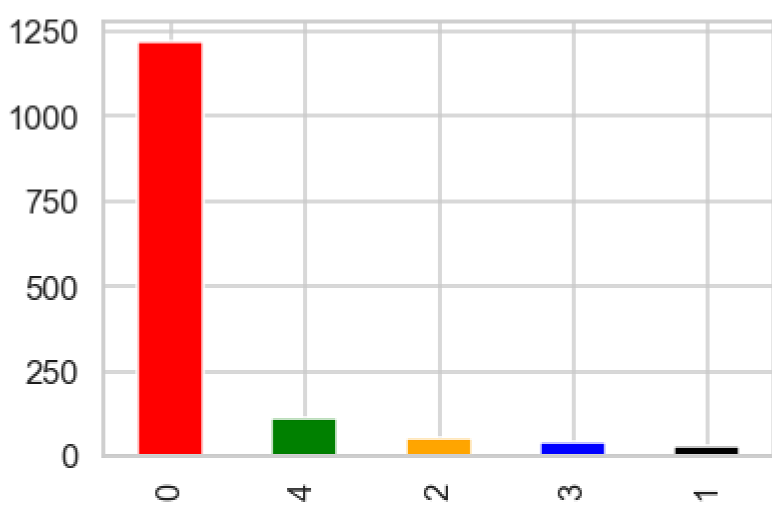
Out[220]:
```
['MSZoning',
 'BldgType',
 'HouseStyle',
 'OverallCond',
 'FullBath',
 'HalfBath',
 'GarageCars',
 'PavedDrive',
 'SaleCondition']
```

In [221]:
```
# Import label encoder
from sklearn import preprocessing

# label_encoder object knows how to understand word labels.
label_encoder = preprocessing.LabelEncoder()

# Encode labels in columns
data['BldgType']= label_encoder.fit_transform(data['BldgType'])

data['BldgType'].unique()
```

Out[221]:
```
array([0, 1, 2, 4, 3])
```

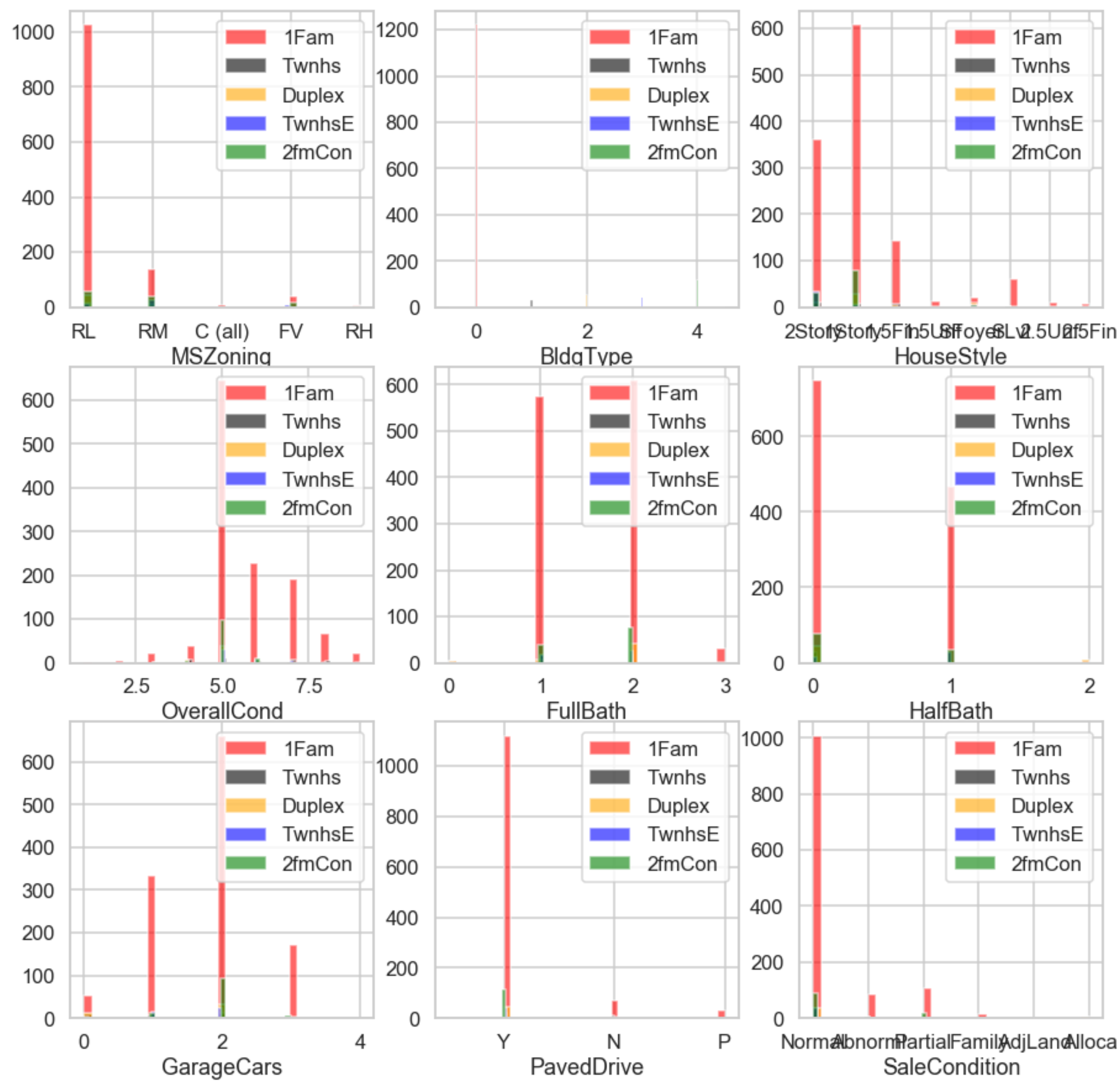
In [222]:
```
#'1Fam', '2fmCon', 'Duplex', 'TwnhsE', 'Twnhs'
# "red", "green","orange","blue","black"
data.BldgType.value_counts().plot(kind="bar", color=["red", "green","orange",
```

Out[222]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1511d6b6388>
```



In [223]:

```
#'1Fam', '2fmCon', 'Duplex', 'TwnhsE', 'Twnhs'
# "red", "green","orange","blue","black"

plt.figure(figsize=(15, 15))

for i, column in enumerate(non_numeric, 1):
    plt.subplot(3, 3, i)
    data[data["BldgType"] == 0][column].hist(bins=35, color='red', label='1Fa
    data[data["BldgType"] == 1][column].hist(bins=35, color='black', label='T
    data[data["BldgType"] == 2][column].hist(bins=35, color='orange', label='
    data[data["BldgType"] == 3][column].hist(bins=35, color='blue', label='Tw
    data[data["BldgType"] == 4][column].hist(bins=35, color='green', label='2

    plt.legend()
    plt.xlabel(column)
```



In [224]:

```
# Import label encoder
from sklearn import preprocessing

# label_encoder object knows how to understand word labels.
label_encoder = preprocessing.LabelEncoder()

# Encode labels in columns
data['MSZoning']= label_encoder.fit_transform(data['MSZoning'])

data['MSZoning'].unique()
```

Out[224]:

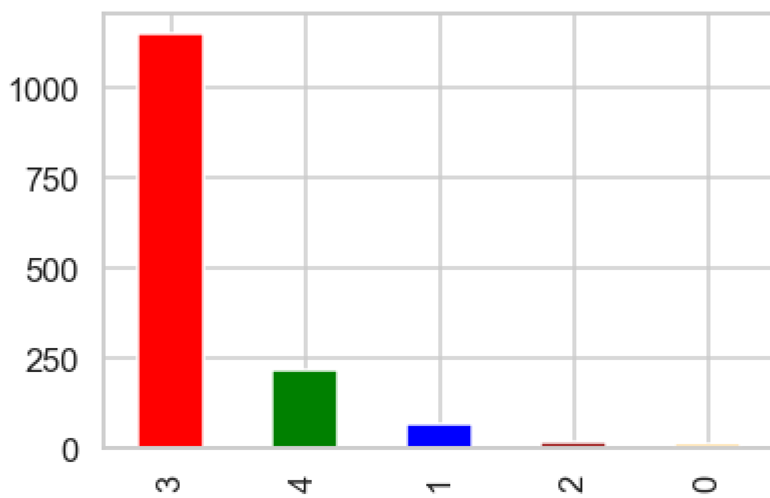
```
array([3, 4, 0, 1, 2])
```

In [225]:

```
#'RL', 'RM', 'C (all)', 'FV', 'RH'
#"red", "green","blue","brown","orange"
data.MSZoning.value_counts().plot(kind="bar", color=["red", "green","blue","b
```

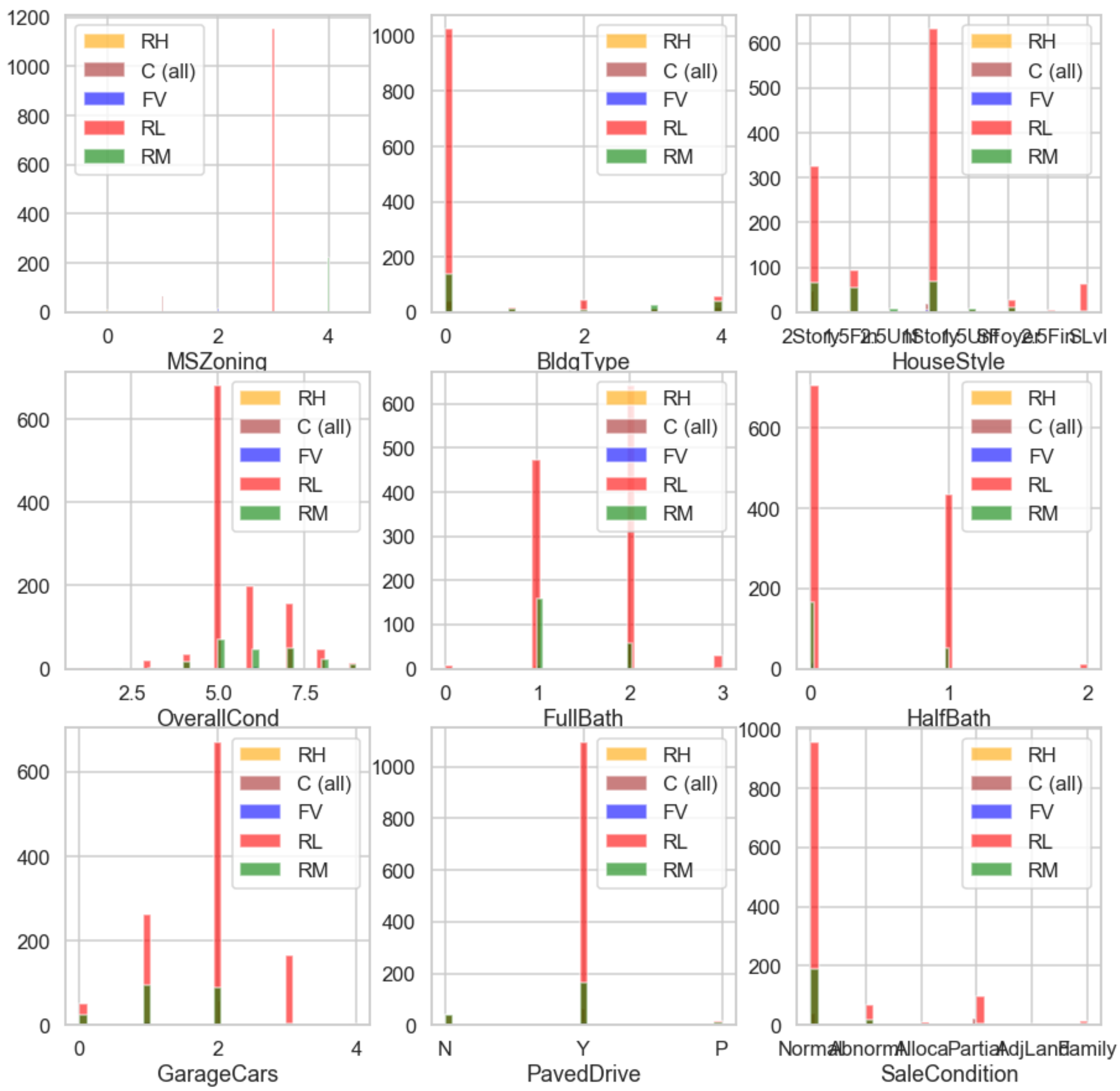
Out[225]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1511f2ee588>
```



In [226]:

```
#'RL'=3, 'RM'=4, 'C (all)'=1, 'FV'=2, 'RH'=0
#"red", "green","blue","brown","orange"

plt.figure(figsize=(15, 15))

for i, column in enumerate(non_numeric, 1):
    plt.subplot(3, 3, i)
    data[data["MSZoning"] == 0][column].hist(bins=35, color='orange', label='
    data[data["MSZoning"] == 1][column].hist(bins=35, color='brown', label='C
    data[data["MSZoning"] == 2][column].hist(bins=35, color='blue', label='FV
    data[data["MSZoning"] == 3][column].hist(bins=35, color='red', label='RL'
    data[data["MSZoning"] == 4][column].hist(bins=35, color='green', label='R

    plt.legend()
    plt.xlabel(column)
```



In [227]:
```
# Import label encoder
from sklearn import preprocessing

# label_encoder object knows how to understand word labels.
label_encoder = preprocessing.LabelEncoder()

# Encode labels in columns
data['BldgType']= label_encoder.fit_transform(data['BldgType'])

data['BldgType'].unique()
```

Out[227]:
```
array([0, 1, 2, 4, 3], dtype=int64)
```

In [228]:
```
# Import label encoder
from sklearn import preprocessing

# label_encoder object knows how to understand word labels.
label_encoder = preprocessing.LabelEncoder()

# Encode labels in columns
data['HouseStyle']= label_encoder.fit_transform(data['HouseStyle'])

data['HouseStyle'].unique()
```

Out[228]:
```
array([5, 2, 0, 1, 6, 7, 4, 3])
```

In [229]:
```
# Import label encoder
from sklearn import preprocessing

# label_encoder object knows how to understand word labels.
label_encoder = preprocessing.LabelEncoder()

# Encode labels in columns
data['OverallCond']= label_encoder.fit_transform(data['OverallCond'])

data['OverallCond'].unique()
```

Out[229]:
```
array([4, 7, 5, 6, 3, 1, 2, 8, 0], dtype=int64)
```

In [230]:
```python
# Import label encoder
from sklearn import preprocessing

# label_encoder object knows how to understand word labels.
label_encoder = preprocessing.LabelEncoder()

# Encode labels in columns
data['FullBath']= label_encoder.fit_transform(data['FullBath'])

data['FullBath'].unique()
```

Out[230]:
```
array([2, 1, 3, 0], dtype=int64)
```

In [231]:
```python
# Import label encoder
from sklearn import preprocessing

# label_encoder object knows how to understand word labels.
label_encoder = preprocessing.LabelEncoder()

# Encode labels in columns
data['HalfBath']= label_encoder.fit_transform(data['HalfBath'])

data['HalfBath'].unique()
```

Out[231]:
```
array([1, 0, 2], dtype=int64)
```

In [232]:
```python
# Import label encoder
from sklearn import preprocessing

# label_encoder object knows how to understand word labels.
label_encoder = preprocessing.LabelEncoder()

# Encode labels in columns
data['GarageCars']= label_encoder.fit_transform(data['GarageCars'])

data['GarageCars'].unique()
```

Out[232]:
```
array([2, 3, 1, 0, 4], dtype=int64)
```

In [233]:
```python
# Import label encoder
from sklearn import preprocessing

# label_encoder object knows how to understand word labels.
label_encoder = preprocessing.LabelEncoder()

# Encode labels in columns
data['PavedDrive']= label_encoder.fit_transform(data['PavedDrive'])

data['PavedDrive'].unique()
```

Out[233]:
```
array([2, 0, 1])
```

In [234]:
```
# Import label encoder
from sklearn import preprocessing

# label_encoder object knows how to understand word labels.
label_encoder = preprocessing.LabelEncoder()

# Encode labels in columns
data['SaleCondition']= label_encoder.fit_transform(data['SaleCondition'])

data['SaleCondition'].unique()
```

Out[234]:
```
array([4, 0, 5, 1, 2, 3])
```

In [235]:
```
numeric
```

Out[235]:
```
['YearBuilt', 'SalePrice']
```

In [236]:
```
data.corr() # Pearson Correlation Coefficients
```

Out[236]:

	MSZoning	BldgType	HouseStyle	OverallCond	YearBuilt	FullBath	HalfBath
MSZoning	1.000000	0.005690	-0.105315	0.186951	-0.308908	-0.198290	-0.133876
BldgType	0.005690	1.000000	0.066552	-0.162040	0.217584	0.070757	-0.007588
HouseStyle	-0.105315	0.066552	1.000000	-0.031329	0.270494	0.237819	0.414705
OverallCond	0.186951	-0.162040	-0.031329	1.000000	-0.375983	-0.194149	-0.060769
YearBuilt	-0.308908	0.217584	0.270494	-0.375983	1.000000	0.468271	0.242656
FullBath	-0.198290	0.070757	0.237819	-0.194149	0.468271	1.000000	0.136381
HalfBath	-0.133876	-0.007588	0.414705	-0.060769	0.242656	0.136381	1.000000
GarageCars	-0.157042	0.007402	0.196761	-0.185758	0.537850	0.469672	0.219178
PavedDrive	-0.100366	0.059390	0.115580	-0.062236	0.427561	0.129435	0.108148
SaleCondition	0.009494	-0.003530	0.022753	0.017758	0.201044	0.143864	0.072135
SalePrice	-0.166872	-0.085591	0.180163	-0.077856	0.522897	0.560664	0.284108

In [237]:

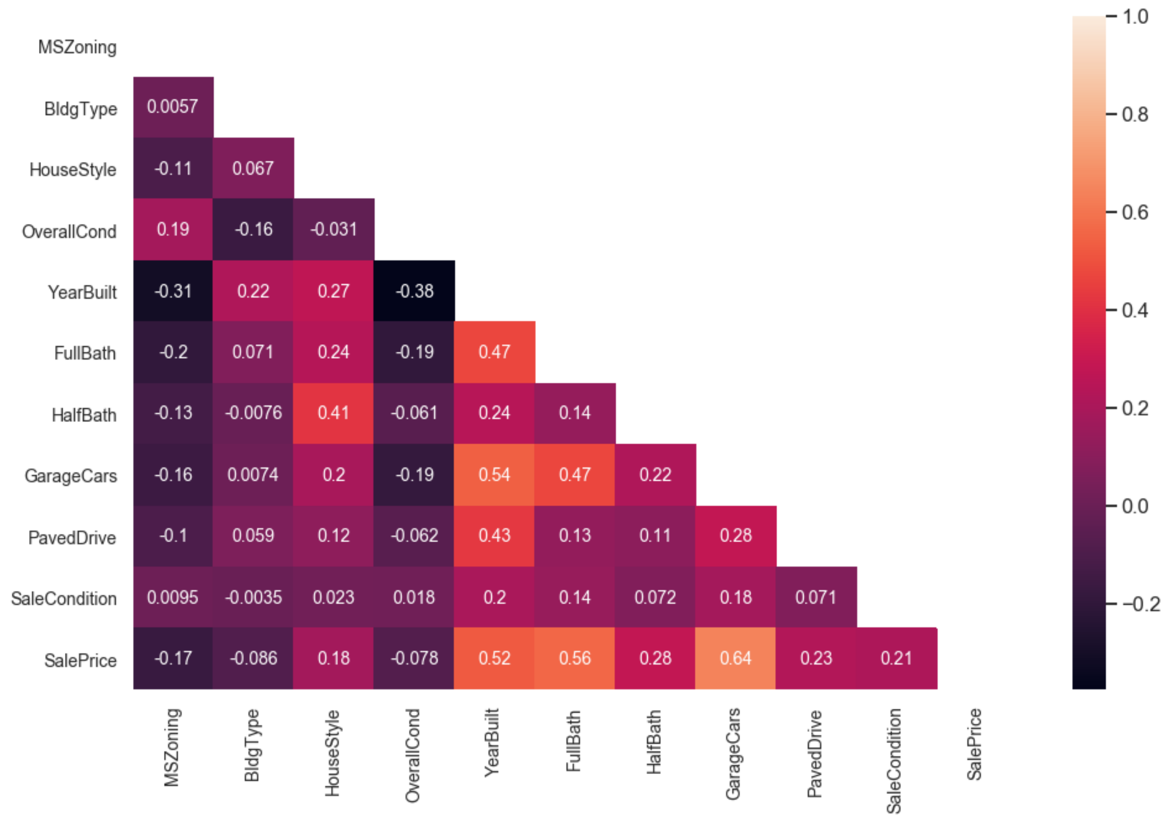
```
mask = np.zeros_like(data.corr())
triangle_indices = np.triu_indices_from(mask)
mask[triangle_indices] = True
mask
```

Out[237]:

```
array([[1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.],
       [0., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.],
       [0., 0., 1., 1., 1., 1., 1., 1., 1., 1., 1.],
       [0., 0., 0., 1., 1., 1., 1., 1., 1., 1., 1.],
       [0., 0., 0., 0., 1., 1., 1., 1., 1., 1., 1.],
       [0., 0., 0., 0., 0., 1., 1., 1., 1., 1., 1.],
       [0., 0., 0., 0., 0., 0., 1., 1., 1., 1., 1.],
       [0., 0., 0., 0., 0., 0., 0., 1., 1., 1., 1.],
       [0., 0., 0., 0., 0., 0., 0., 0., 1., 1., 1.],
       [0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 1.],
       [0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1.]])
```

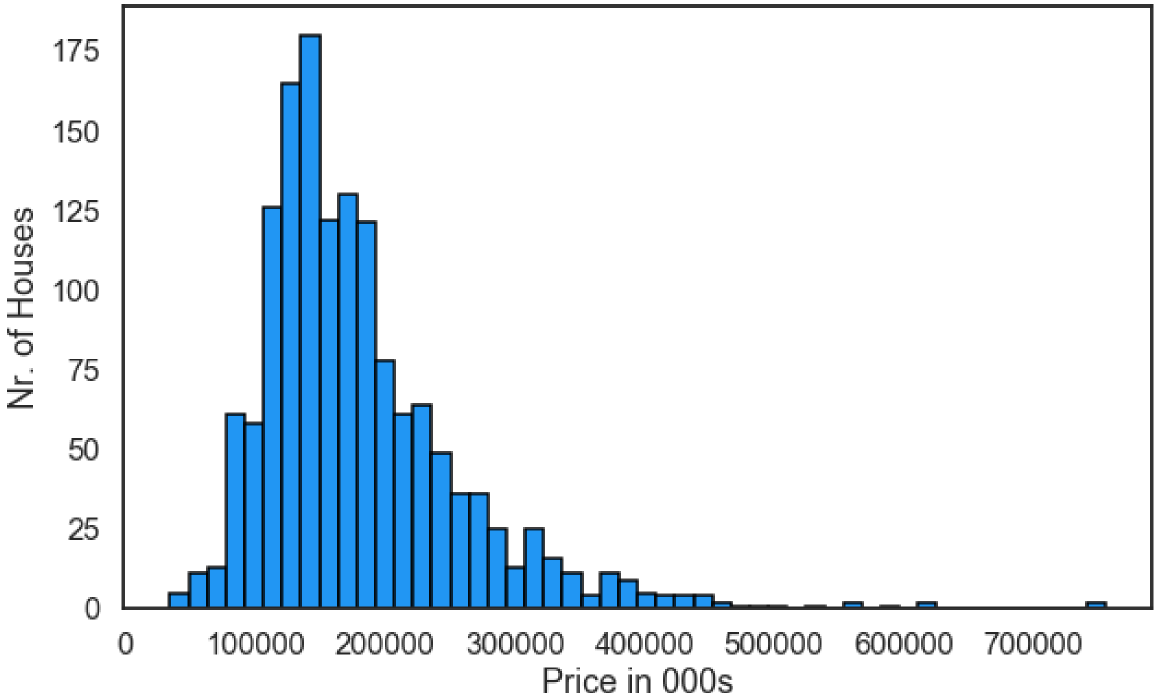
In [238]:

```
plt.figure(figsize=(16,10))
sns.heatmap(data.corr(), mask=mask, annot=True, annot_kws={"size": 14})
sns.set_style('white')
plt.xticks(fontsize=14)
plt.yticks(fontsize=14)
plt.show()
```



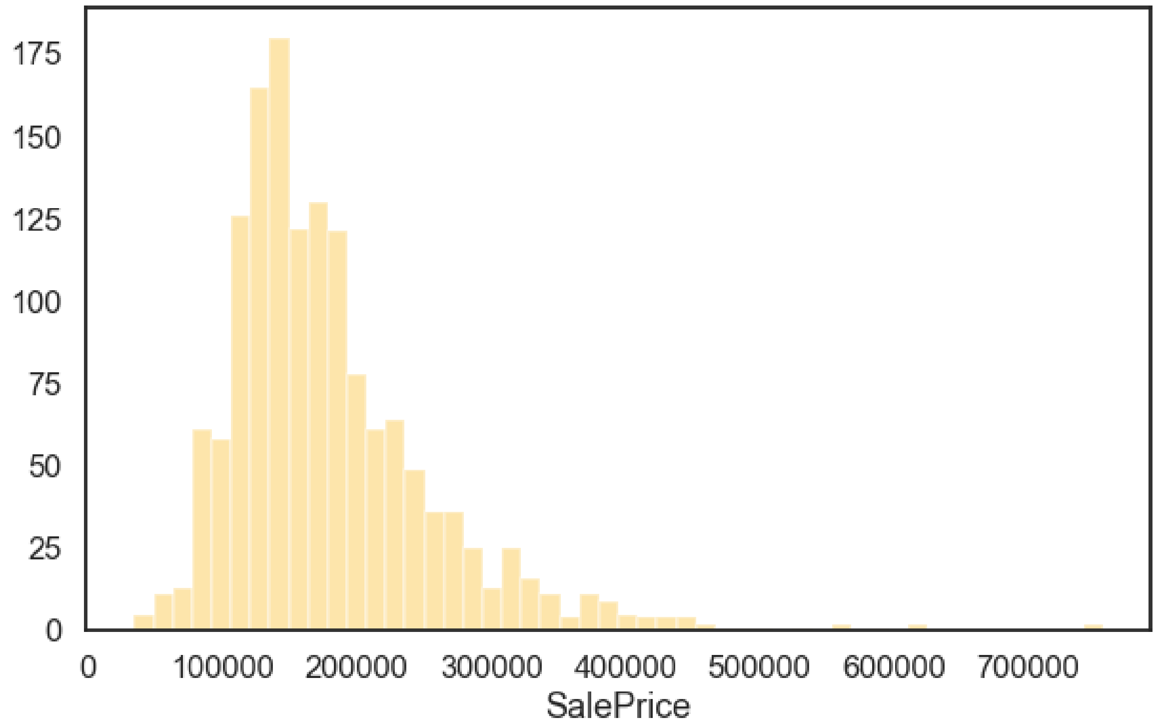
In [239]:

```
plt.figure(figsize=(10, 6))
plt.hist(data['SalePrice'], bins=50, ec='black', color='#2196f3')
plt.xlabel('Price in 000s')
plt.ylabel('Nr. of Houses')
plt.show()
```



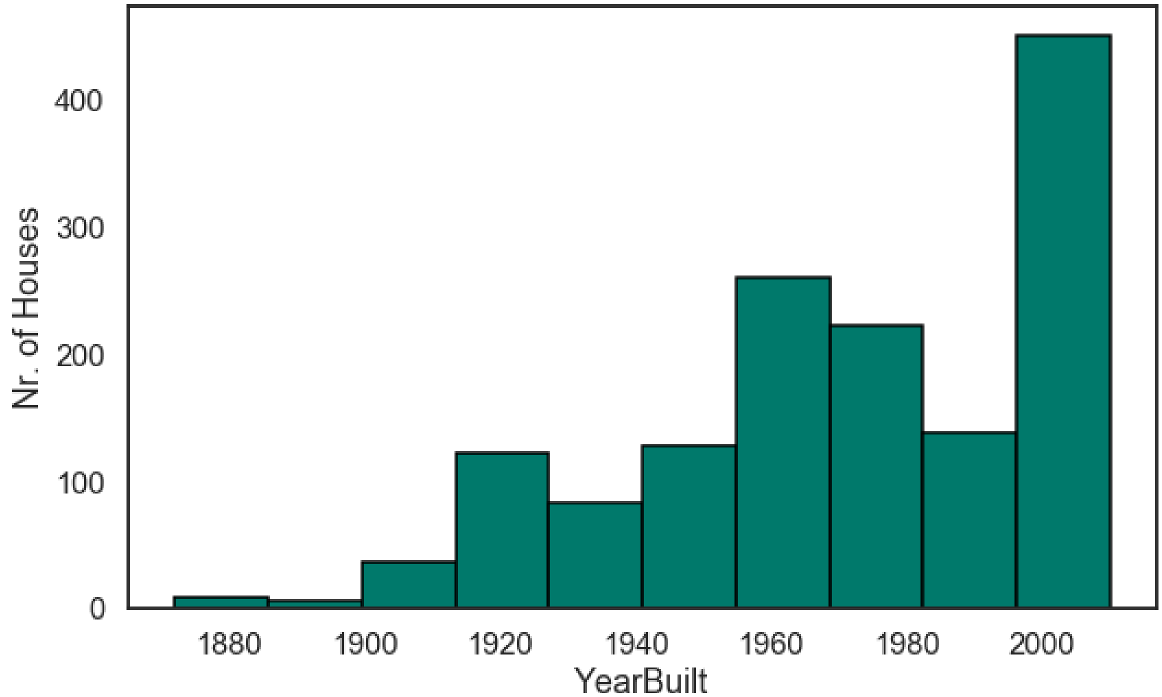
In [240]:

```
plt.figure(figsize=(10, 6))
sns.distplot(data['SalePrice'], bins=50, hist=True, kde=False, color='#fbc02d
plt.show()
```



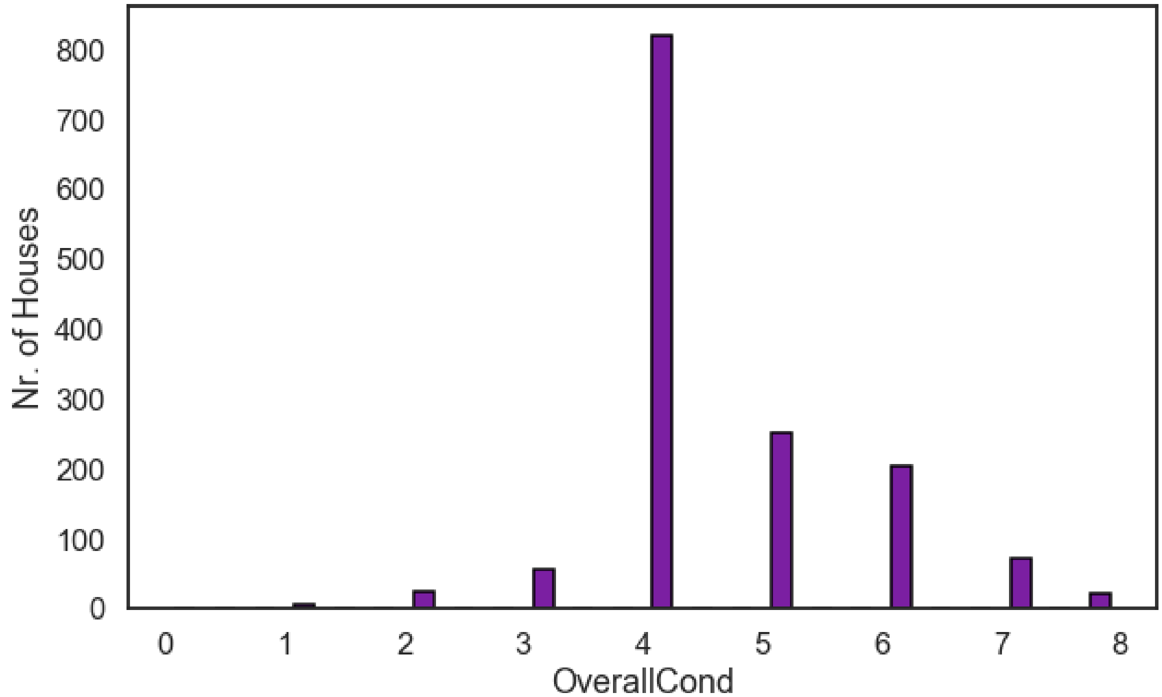
In [241]:

```
plt.figure(figsize=(10, 6))
plt.hist(data['YearBuilt'], ec='black', color='#00796b')
plt.xlabel('YearBuilt')
plt.ylabel('Nr. of Houses')
plt.show()
```

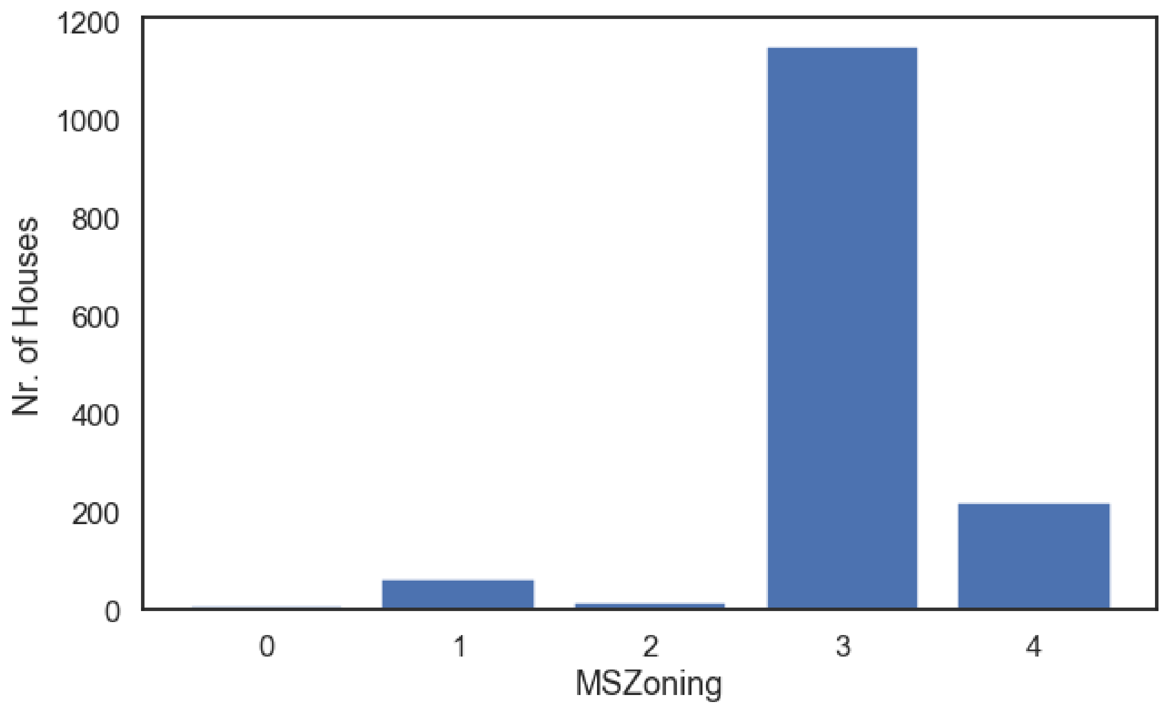


In [242]:

```
plt.figure(figsize=(10, 6))
plt.hist(data['OverallCond'], bins=24, ec='black', color='#7b1fa2', rwidth=0.
plt.xlabel('OverallCond')
plt.ylabel('Nr. of Houses')
plt.show()
```



In [243]:
```
frequency = data['MSZoning'].value_counts()
#type(frequency)
#frequency.index
#frequency.axes[0]
plt.figure(figsize=(10, 6))
plt.xlabel('MSZoning')
plt.ylabel('Nr. of Houses')
plt.bar(frequency.index, height=frequency)
plt.show()
```



Descriptive Statistics

In [244]:
```
data['SalePrice'].mean()
```

Out[244]: 180921.19589041095

In [245]:
```
data['SalePrice'].min()
```

Out[245]: 34900

In [246]:
```
data['SalePrice'].max()
```

Out[246]:
```
755000
```

In [247]:
```
data.min()
```

Out[247]:
```
MSZoning             0
BldgType             0
HouseStyle           0
OverallCond          0
YearBuilt         1872
FullBath             0
HalfBath             0
GarageCars           0
PavedDrive           0
SaleCondition        0
SalePrice        34900
dtype: int64
```

In [248]:
```
data.max()
```

Out[248]:
```
MSZoning              4
BldgType              4
HouseStyle            7
OverallCond           8
YearBuilt          2010
FullBath              3
HalfBath              2
GarageCars            4
PavedDrive            2
SaleCondition         5
SalePrice        755000
dtype: int64
```

In [249]:
```
data.mean()
```

Out[249]:
```
MSZoning              3.028767
BldgType              0.493151
HouseStyle            3.038356
OverallCond           4.575342
YearBuilt          1971.267808
FullBath              1.565068
HalfBath              0.382877
GarageCars            1.767123
PavedDrive            1.856164
SaleCondition         3.770548
SalePrice        180921.195890
dtype: float64
```

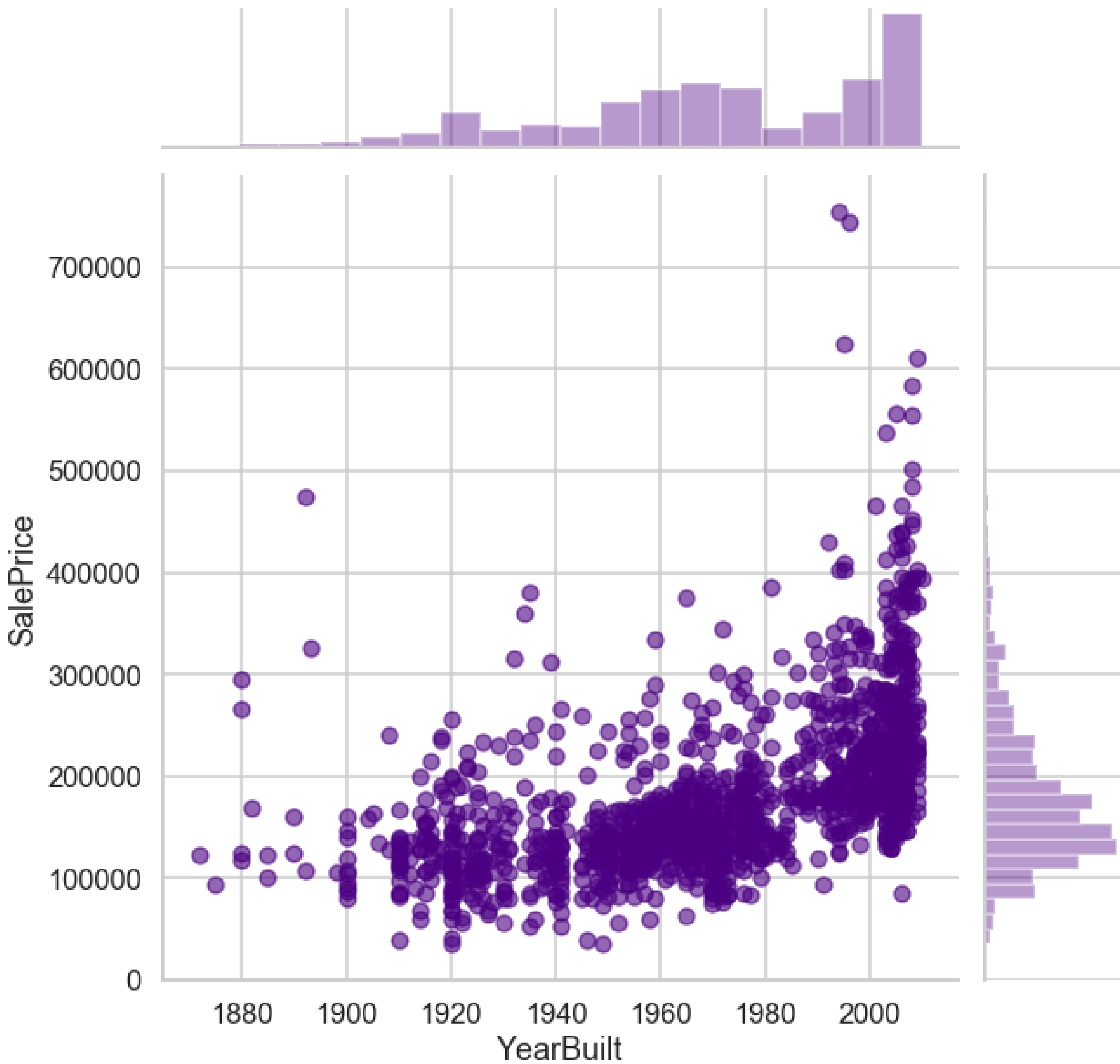
In [250]:
```
data.median()
```

Out[250]:
```
MSZoning              3.0
BldgType              0.0
HouseStyle            2.0
OverallCond           4.0
YearBuilt          1973.0
FullBath              2.0
HalfBath              0.0
GarageCars            2.0
PavedDrive            2.0
SaleCondition         4.0
SalePrice        163000.0
dtype: float64
```

In [252]:

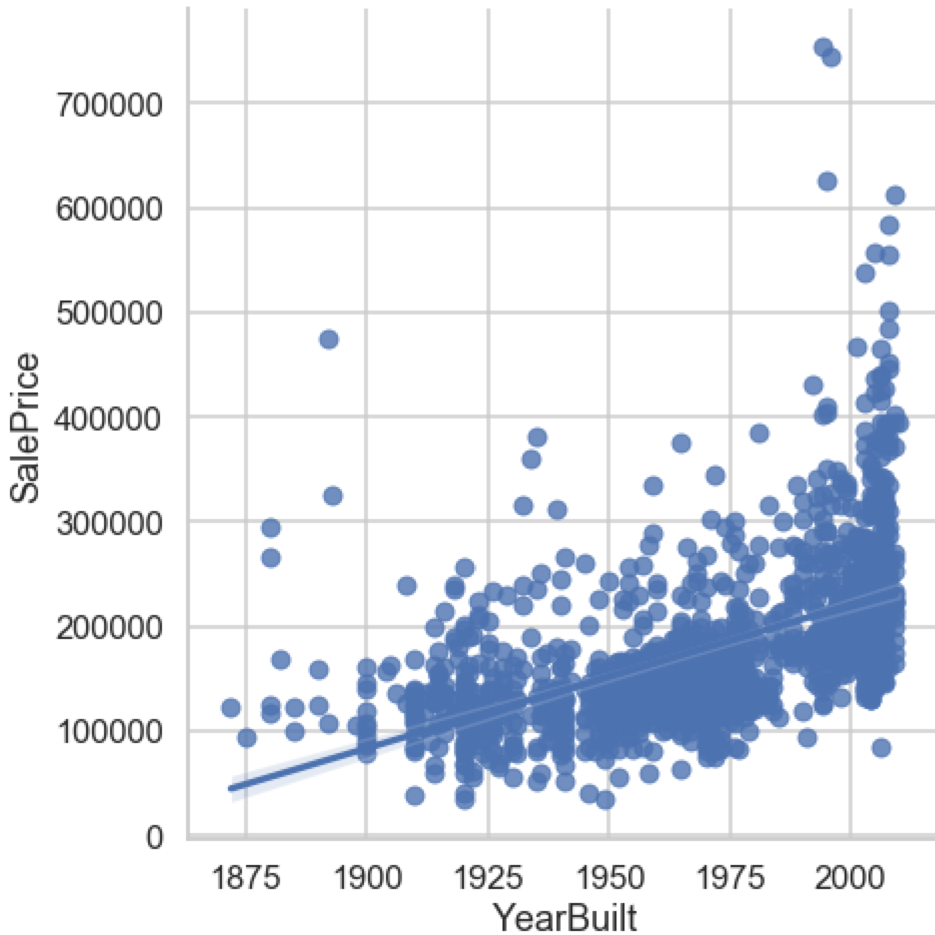
```
sns.set()
sns.set_context('talk')
sns.set_style('whitegrid')
sns.jointplot(x=data['YearBuilt'], y=data['SalePrice'], size=9, color='indigo
plt.show()
```

```
C:\Users\mayam\anaconda3\lib\site-packages\seaborn\axisgrid.py:2272: UserWa
rning: The `size` parameter has been renamed to `height`; please update you
r code.
  warnings.warn(msg, UserWarning)
```



In [253]:
```
sns.lmplot(x='YearBuilt', y='SalePrice', data=data, size=7)
plt.show()
```

```
C:\Users\mayam\anaconda3\lib\site-packages\seaborn\regression.py:574: UserW
arning: The `size` parameter has been renamed to `height`; please update yo
ur code.
  warnings.warn(msg, UserWarning)
```



In [254]:
```
non_numeric
```

Out[254]:
```
['MSZoning',
 'BldgType',
 'HouseStyle',
 'OverallCond',
 'FullBath',
 'HalfBath',
 'GarageCars',
 'PavedDrive',
 'SaleCondition']
```

Training & Test Dataset Split

In [255]:
```
from sklearn.preprocessing import StandardScaler

s_sc = StandardScaler()
col_to_scale = ['YearBuilt', 'SalePrice']
data[col_to_scale] = s_sc.fit_transform(data[col_to_scale])
```

In [256]:
```
data.head()
```

Out[256]:

	MSZoning	BldgType	HouseStyle	OverallCond	YearBuilt	FullBath	HalfBath	GarageCars
0	3	0	5	4	1.050994	2	1	2
1	3	0	2	7	0.156734	2	0	2
2	3	0	5	4	0.984752	2	1	2
3	3	0	5	4	-1.863632	1	0	3
4	3	0	5	4	0.951632	2	1	3

In [257]:
```
data.describe()
```

Out[257]:

	MSZoning	BldgType	HouseStyle	OverallCond	YearBuilt	FullBath	H
count	1460.000000	1460.000000	1460.000000	1460.000000	1.460000e+03	1460.000000	1460.
mean	3.028767	0.493151	3.038356	4.575342	1.032983e-15	1.565068	0.
std	0.632017	1.198277	1.911305	1.112799	1.000343e+00	0.550916	0.
min	0.000000	0.000000	0.000000	0.000000	-3.287824e+00	0.000000	0.
25%	3.000000	0.000000	2.000000	4.000000	-5.719226e-01	1.000000	0.
50%	3.000000	0.000000	2.000000	4.000000	5.737148e-02	2.000000	0.
75%	3.000000	0.000000	5.000000	5.000000	9.516316e-01	2.000000	1.
max	4.000000	4.000000	7.000000	8.000000	1.282839e+00	3.000000	2.

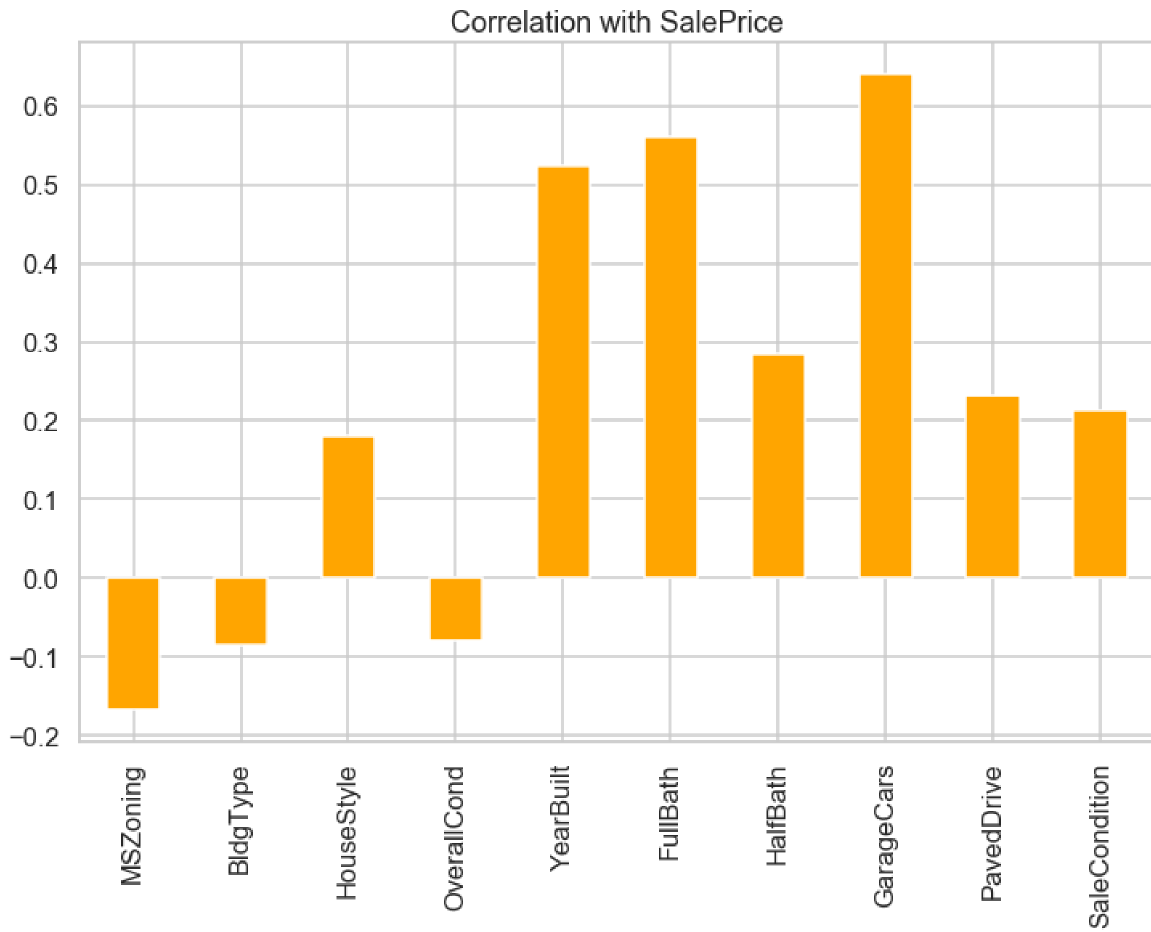
In [258]:
```
data.corr() # Pearson Correlation Coefficients
```

Out[258]:

	MSZoning	BldgType	HouseStyle	OverallCond	YearBuilt	FullBath	HalfBath
MSZoning	1.000000	0.005690	-0.105315	0.186951	-0.308908	-0.198290	-0.133876
BldgType	0.005690	1.000000	0.066552	-0.162040	0.217584	0.070757	-0.007588
HouseStyle	-0.105315	0.066552	1.000000	-0.031329	0.270494	0.237819	0.414705
OverallCond	0.186951	-0.162040	-0.031329	1.000000	-0.375983	-0.194149	-0.060769
YearBuilt	-0.308908	0.217584	0.270494	-0.375983	1.000000	0.468271	0.242656
FullBath	-0.198290	0.070757	0.237819	-0.194149	0.468271	1.000000	0.136381
HalfBath	-0.133876	-0.007588	0.414705	-0.060769	0.242656	0.136381	1.000000
GarageCars	-0.157042	0.007402	0.196761	-0.185758	0.537850	0.469672	0.219178
PavedDrive	-0.100366	0.059390	0.115580	-0.062236	0.427561	0.129435	0.108148
SaleCondition	0.009494	-0.003530	0.022753	0.017758	0.201044	0.143864	0.072135
SalePrice	-0.166872	-0.085591	0.180163	-0.077856	0.522897	0.560664	0.284108

In [259]:
```
data.drop('SalePrice', axis=1).corrwith(data.SalePrice).plot(kind='bar', gric
                                                            title="Correlation with Sa
```

Out[259]: <matplotlib.axes._subplots.AxesSubplot at 0x1511d245688>



In [260]:
```
data['SalePrice'].corr(data['YearBuilt'])
```

Out[260]: 0.5228973328794971

In [263]:
```
data['SalePrice'].corr(data['MSZoning'])
```
Out[263]: -0.16687220265320626

In [264]:
```
data['SalePrice'].corr(data['OverallCond'])
```
Out[264]: -0.077855894048678

In [265]:
```
data['SalePrice'].corr(data['BldgType'])
```
Out[265]: -0.08559060818352934

In [266]:
```
data['SalePrice'].corr(data['HouseStyle'])
```
Out[266]: 0.1801626233439912

In [267]:
```
data['SalePrice'].corr(data['FullBath'])
```
Out[267]: 0.5606637627484456

In [268]:
```
data['SalePrice'].corr(data['HalfBath'])
```
Out[268]: 0.28410767559478295

In [269]:
```
data['SalePrice'].corr(data['GarageCars'])
```
Out[269]: 0.6404091972583532

In [270]:
```
data['SalePrice'].corr(data['PavedDrive'])
```
Out[270]: 0.2313569522572269

In [271]:
```
data['SalePrice'].corr(data['SaleCondition'])
```
Out[271]: 0.21309202967780422

In [ ]:

machine learning algorithms

In [315]:

```
from sklearn.dummy import DummyRegressor
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import r2_score
from sklearn.model_selection import train_test_split
features =  list(data.drop(['SalePrice'],axis=1))
y = data.SalePrice
X = data[features]
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2,random_sta
dummy_median = DummyRegressor(strategy='mean')
dummy_regressor = dummy_median.fit(X_train,y_train)
dummy_predicts = dummy_regressor.predict(X_test)
print("Model Accuracy:", dummy_regressor.score(X_test,y_test)*100)
```

```
Model Accuracy: -0.034758835279324884
```

In [316]:

```
print('$',mean_absolute_error(y_test,dummy_predicts))
```

```
$ 0.713958955762855
```

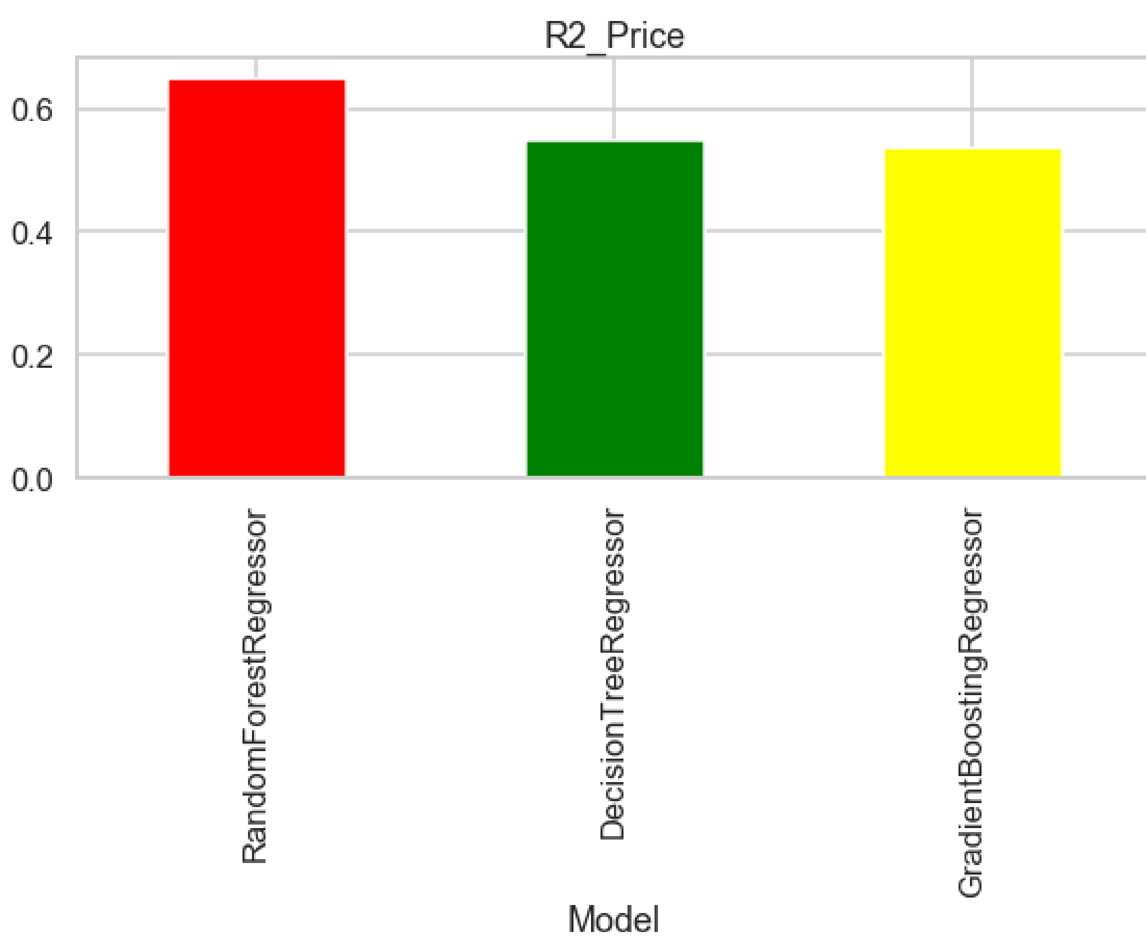
In [317]:

```
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import GradientBoostingRegressor
models = [RandomForestRegressor(n_estimators=200,criterion='mse',max_depth=20
learning_mods = pd.DataFrame()
temp = {}
```

In [318]:

```
#run through models
for model in models:
    print(model)
    m = str(model)
    temp['Model'] = m[:m.index('(')]
    model.fit(X_train, y_train)
    temp['R2_Price'] = r2_score(y_test, model.predict(X_test))
    print('score on training',model.score(X_train, y_train))
    print('r2 score',r2_score(y_test, model.predict(X_test)))
    learning_mods = learning_mods.append([temp])
learning_mods.set_index('Model', inplace=True)

fig, axes = plt.subplots(ncols=1, figsize=(10, 4))
learning_mods.R2_Price.plot(ax=axes, kind='bar', title='R2_Price',color=["rec
plt.show()
```

```
RandomForestRegressor(max_depth=20, n_estimators=200, random_state=100)
score on training 0.8939792839462513
r2 score 0.6492521593041091
DecisionTreeRegressor(max_depth=11, random_state=100)
score on training 0.8855854581536768
r2 score 0.5477859297372936
GradientBoostingRegressor(max_depth=12, n_estimators=200)
score on training 0.9269106478982981
r2 score 0.534382761138374
```



In [319]:

```
regressionTree_imp = model.feature_importances_
plt.figure(figsize=(16,6))
plt.yscale('log',nonposy='clip')
plt.bar(range(len(regressionTree_imp)),regressionTree_imp,align='center',colo
'purple','yellow','black','pink','cyan'])
plt.xticks(range(len(regressionTree_imp)),features,rotation='vertical')
plt.title('Feature Importance')
plt.ylabel('Importance')
plt.show()
```