In [1]:
```python
# Import Libraries and Packages

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import statsmodels.api as sm
import statsmodels.formula.api as smf
from sklearn import metrics
```

In [2]:
```python
# Load Data

df = pd.read_csv('DC_Properties.csv')

df.head()
```

```
/opt/anaconda3/lib/python3.8/site-packages/IPython/core/interactiveshell.py:3146: DtypeWarning: Columns (18,19,
20,21,22,23,24,33,34,35,37,43,45) have mixed types.Specify dtype option on import or set low_memory=False.
  has_raised = await self.run_ast_nodes(code_ast.body, cell_name,
```

Out[2]:

	Unnamed: 0	BATHRM	HF_BATHRM	HEAT	AC	NUM_UNITS	ROOMS	BEDRM	AYB	YR_RMDL	...	LONGITUDE	ASSESSMENT_NBI
0	0	4	0	Warm Cool	Y	2.0	8	4	1910.0	1988.0	...	-77.040832	Old Cit
1	1	3	1	Warm Cool	Y	2.0	11	5	1898.0	2007.0	...	-77.040764	Old Cit
2	2	3	1	Hot Water Rad	Y	2.0	9	5	1910.0	2009.0	...	-77.040678	Old Cit
3	3	3	1	Hot Water Rad	Y	2.0	8	5	1900.0	2003.0	...	-77.040629	Old Cit
4	4	2	1	Warm Cool	Y	1.0	11	3	1913.0	2012.0	...	-77.039361	Old Cit

5 rows × 49 columns

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 158957 entries, 0 to 158956
Data columns (total 49 columns):
 #   Column              Non-Null Count   Dtype
---  ------              --------------   -----
 0   Unnamed: 0          158957 non-null  int64
 1   BATHRM              158957 non-null  int64
 2   HF_BATHRM           158957 non-null  int64
 3   HEAT                158957 non-null  object
 4   AC                  158957 non-null  object
 5   NUM_UNITS           106696 non-null  float64
 6   ROOMS               158957 non-null  int64
 7   BEDRM               158957 non-null  int64
 8   AYB                 158686 non-null  float64
 9   YR_RMDL             80928 non-null   float64
 10  EYB                 158957 non-null  int64
 11  STORIES             106652 non-null  float64
 12  SALEDATE            132187 non-null  object
 13  PRICE               98216 non-null   float64
 14  QUALIFIED           158957 non-null  object
 15  SALE_NUM            158957 non-null  int64
 16  GBA                 106696 non-null  float64
 17  BLDG_NUM            158957 non-null  int64
 18  STYLE               106696 non-null  object
 19  STRUCT              106696 non-null  object
 20  GRADE               106696 non-null  object
 21  CNDTN               106696 non-null  object
 22  EXTWALL             106696 non-null  object
 23  ROOF                106696 non-null  object
 24  INTWALL             106696 non-null  object
 25  KITCHENS            106695 non-null  float64
 26  FIREPLACES          158957 non-null  int64
 27  USECODE             158957 non-null  int64
 28  LANDAREA            158957 non-null  int64
 29  GIS_LAST_MOD_DTTM   158957 non-null  object
 30  SOURCE              158957 non-null  object
 31  CMPLX_NUM           52261 non-null   float64
 32  LIVING_GBA          52261 non-null   float64
 33  FULLADDRESS         106040 non-null  object
 34  CITY                106051 non-null  object
 35  STATE               106051 non-null  object
 36  ZIPCODE             158956 non-null  float64
 37  NATIONALGRID        106051 non-null  object
 38  LATITUDE            158956 non-null  float64
 39  LONGITUDE           158956 non-null  float64
 40  ASSESSMENT_NBHD     158956 non-null  object
 41  ASSESSMENT_SUBNBHD  126406 non-null  object
 42  CENSUS_TRACT        158956 non-null  float64
 43  CENSUS_BLOCK        106051 non-null  object
 44  WARD                158956 non-null  object
 45  SQUARE              158957 non-null  object
 46  X                   158720 non-null  float64
 47  Y                   158720 non-null  float64
 48  QUADRANT            158720 non-null  object
dtypes: float64(15), int64(11), object(23)
memory usage: 59.4+ MB
```

In [4]:
```python
# Create new DataFrame with needed columns

studydf = df[['PRICE','BATHRM', 'HF_BATHRM', 'HEAT', 'AC', 'NUM_UNITS', \
         'ROOMS', 'BEDRM', 'AYB', 'EYB', 'STORIES', 'GBA', \
          'STYLE', 'STRUCT', 'GRADE', 'CNDTN', 'EXTWALL', \
         'ROOF', 'INTWALL', 'KITCHENS', 'FIREPLACES', 'LANDAREA', 'WARD']]
studydf.head()
```

Out[4]:

	PRICE	BATHRM	HF_BATHRM	HEAT	AC	NUM_UNITS	ROOMS	BEDRM	AYB	EYB	...	STRUCT	GRADE	CNDTN	EXTWAI
0	1095000.0	4	0	Warm Cool	Y	2.0	8	4	1910.0	1972	...	Row Inside	Very Good	Good	Comm Bri
1	NaN	3	1	Warm Cool	Y	2.0	11	5	1898.0	1972	...	Row Inside	Very Good	Good	Comm Bri
2	2100000.0	3	1	Hot Water Rad	Y	2.0	9	5	1910.0	1984	...	Row Inside	Very Good	Very Good	Comm Bri
3	1602000.0	3	1	Hot Water Rad	Y	2.0	8	5	1900.0	1984	...	Row Inside	Very Good	Good	Comm Bri
4	NaN	2	1	Warm Cool	Y	1.0	11	3	1913.0	1985	...	Semi-Detached	Very Good	Good	Comm Bri

5 rows × 23 columns

In [5]:
```python
# Remove all Rows where PRICE is null

studydf = studydf[studydf['PRICE'].isnull() == False]
studydf.shape
```

Out[5]: (98216, 23)

In [6]:
```python
# Remove the word Ward from WARD column and change to int type
#Code from: https://stackoverflow.com/questions/51778480/remove-certain-string-from-entire-column-in-pandas-dat

studydf['WARD'] = studydf['WARD'].str.replace('Ward', '')
studydf['WARD'] = studydf['WARD'].astype(str).astype(int)

studydf['WARD'].head()
```

Out[6]:
```
0    2
2    2
3    2
5    2
7    2
Name: WARD, dtype: int64
```

Nulls in other columns

Find the percent of null for each column

In [7]:
```python
per_missing = studydf.isnull().sum()*100/len(df.iloc[:,1])
per_missing
```

Out[7]:
```
PRICE          0.000000
BATHRM         0.000000
HF_BATHRM      0.000000
HEAT           0.000000
AC             0.000000
NUM_UNITS     25.362834
ROOMS          0.000000
BEDRM          0.000000
AYB            0.070459
EYB            0.000000
STORIES       25.383594
GBA           25.362834
STYLE         25.362834
STRUCT        25.362834
GRADE         25.362834
CNDTN         25.362834
EXTWALL       25.362834
ROOF          25.362834
INTWALL       25.362834
KITCHENS      25.363463
FIREPLACES     0.000000
LANDAREA       0.000000
WARD           0.000000
dtype: float64
```

11 Columns have 25% Data missing. I will look at the unique values in each column to decide if I can impute a value or need to delete the column.

In [8]:
```python
studydf['NUM_UNITS'].value_counts()
```

Out[8]:
```
1.0    49339
2.0     6050
4.0     1862
3.0      590
0.0       51
5.0        5
6.0        3
Name: NUM_UNITS, dtype: int64
```

Fill with 1 as it is the most occuring value

In [9]:
```python
studydf['NUM_UNITS'] = studydf['NUM_UNITS'].fillna(1)
studydf['NUM_UNITS'].value_counts()
```

Out[9]:
```
1.0    89655
2.0     6050
4.0     1862
3.0      590
0.0       51
5.0        5
6.0        3
Name: NUM_UNITS, dtype: int64
```

In [10]:
```python
studydf['STORIES'].value_counts()
```

Out[10]:
```
2.00      43173
3.00       5452
2.50       3363
1.00       2117
2.25       1291
1.50       1002
1.75        584
2.75        266
4.00        237
1.25        213
3.50         63
9.00         15
3.25         14
2.70         14
0.00         12
3.75         10
5.00          9
2.20          8
6.00          5
25.00         4
2.30          3
1.70          3
275.00        2
3.70          1
826.00        1
0.25          1
250.00        1
8.25          1
7.00          1
20.00         1
Name: STORIES, dtype: int64
```

Fill with 2 as it is the most occuring value

In [11]:
```python
studydf['STORIES'] = studydf['STORIES'].fillna(2)
studydf['STORIES'].value_counts()
```

Out[11]:
```
2.00      83522
3.00       5452
2.50       3363
1.00       2117
2.25       1291
1.50       1002
1.75        584
2.75        266
4.00        237
1.25        213
3.50         63
9.00         15
3.25         14
2.70         14
0.00         12
3.75         10
5.00          9
2.20          8
6.00          5
25.00         4
2.30          3
1.70          3
275.00        2
3.70          1
826.00        1
0.25          1
250.00        1
8.25          1
7.00          1
20.00         1
Name: STORIES, dtype: int64
```

Exterior Wall types

Brick is the most common type. I will fill nulls with this type.

In [12]:
```python
studydf['EXTWALL'].value_counts()
```

Out[12]:
```
Common Brick      43369
Brick/Siding       3526
Vinyl Siding       3169
Wood Siding        2273
Stucco             1652
Brick Veneer        593
Shingle             533
Aluminum            461
Face Brick          420
Brick/Stucco        401
Brick/Stone         386
Stone               371
Stone/Siding        214
Stone/Stucco        156
Stone Veneer        148
Hardboard            79
Concrete             40
Concrete Block       35
Stucco Block         28
Metal Siding         28
Default              12
Plywood               4
Adobe                 1
SPlaster              1
Name: EXTWALL, dtype: int64
```

In [13]:
```python
studydf['EXTWALL'] = studydf['EXTWALL'].fillna('Common Brick')
studydf['EXTWALL'].value_counts()
```

Out[13]:
```
Common Brick      83685
Brick/Siding       3526
Vinyl Siding       3169
Wood Siding        2273
Stucco             1652
Brick Veneer        593
Shingle             533
Aluminum            461
Face Brick          420
Brick/Stucco        401
Brick/Stone         386
Stone               371
Stone/Siding        214
Stone/Stucco        156
Stone Veneer        148
Hardboard            79
Concrete             40
Concrete Block       35
Stucco Block         28
Metal Siding         28
Default              12
Plywood               4
Adobe                 1
SPlaster              1
Name: EXTWALL, dtype: int64
```

Interior Wall

Hardwood is the most frequent. I will fill nulls with Hardwood.

In [14]:
```python
studydf['INTWALL'].value_counts()
```

Out[14]:
```
Hardwood         44180
Hardwood/Carp     7511
Wood Floor        4014
Carpet            2014
Lt Concrete         50
Default             41
Ceramic Tile        39
Vinyl Comp          21
Parquet             11
Vinyl Sheet          8
Resiliant            8
Terrazo              3
Name: INTWALL, dtype: int64
```

In [15]:
```python
studydf['INTWALL'] = studydf['INTWALL'].fillna('Hardwood')
studydf['INTWALL'].value_counts()
```

Out[15]:
```
Hardwood         84496
Hardwood/Carp     7511
Wood Floor        4014
Carpet            2014
Lt Concrete         50
Default             41
Ceramic Tile        39
Vinyl Comp          21
Parquet             11
Vinyl Sheet          8
Resiliant            8
Terrazo              3
Name: INTWALL, dtype: int64
```

Kitchen

Most frequent value is 1. Will replace null with 1.

In [16]:
```python
studydf['KITCHENS'].value_counts()
```

Out[16]:
```
1.0     47704
2.0      7656
4.0      1859
3.0       627
0.0        43
5.0         5
6.0         4
44.0        1
Name: KITCHENS, dtype: int64
```

In [17]:
```python
studydf['KITCHENS'] = studydf['KITCHENS'].fillna(1)
studydf['KITCHENS'].value_counts()
```

Out[17]:
```
1.0     88021
2.0      7656
4.0      1859
3.0       627
0.0        43
5.0         5
6.0         4
44.0        1
Name: KITCHENS, dtype: int64
```

Replace Null in GBA with mean

In [18]:
```python
GBAmedian = studydf['GBA'].median()
studydf['GBA'] = studydf['GBA'].fillna(GBAmedian)
```

Structure

There is not one value that stands out above the others. I don't want to get ride of the information the column provides so I will relpace nulls with missing.

In [19]:
```python
studydf['STRUCT'].value_counts()
```

Out[19]:
```
Row Inside       23425
Single           16248
Semi-Detached     8209
Row End           6906
Multi             2843
Town Inside        185
Town End            77
Default              7
Name: STRUCT, dtype: int64
```

In [20]:
```python
studydf['STRUCT'] = studydf['STRUCT'].fillna('Missing')
studydf['STRUCT'].value_counts()
```

Out[20]:
```
Missing          40316
Row Inside       23425
Single           16248
Semi-Detached     8209
Row End           6906
Multi             2843
Town Inside        185
Town End            77
Default              7
Name: STRUCT, dtype: int64
```

Grade

Again, there is not one value that is above the others. I willfill with missing.

In [21]:
```python
studydf['GRADE'].value_counts()
```

Out[21]:
```
Average          19664
Above Average    17321
Good Quality     11752
Very Good         5008
Excellent         1825
Superior          1523
Exceptional-A      473
Exceptional-B      168
Fair Quality        83
Exceptional-C       44
Exceptional-D       32
Low Quality          4
No Data              3
Name: GRADE, dtype: int64
```

In [22]:
```python
studydf['GRADE'] = studydf['GRADE'].fillna('Missing')
studydf['GRADE'].value_counts()
```

Out[22]:
```
Missing          40316
Average          19664
Above Average    17321
Good Quality     11752
Very Good         5008
Excellent         1825
Superior          1523
Exceptional-A      473
Exceptional-B      168
Fair Quality        83
Exceptional-C       44
Exceptional-D       32
Low Quality          4
No Data              3
Name: GRADE, dtype: int64
```

Condition

No one rating has the largest value. I will fill with missing.

In [23]:
```python
studydf['CNDTN'].value_counts()
```

Out[23]:
```
Good         25711
Average      24013
Very Good     6681
Excellent      889
Fair           509
Poor            93
Default          4
Name: CNDTN, dtype: int64
```

In [24]:
```python
studydf['CNDTN'] = studydf['CNDTN'].fillna('Missing')
studydf['CNDTN'].value_counts()
```

Out[24]:
```
Missing      40316
Good         25711
Average      24013
Very Good     6681
Excellent      889
Fair           509
Poor            93
Default          4
Name: CNDTN, dtype: int64
```

Roof Type There is not one type that is most frequent. I will fill with missing.

In [25]:
```python
studydf['ROOF'].value_counts()
```

Out[25]:
```
Built Up          17633
Metal- Sms        16338
Comp Shingle      16170
Slate              5583
Neopren             903
Shake               383
Clay Tile           309
Shingle             248
Metal- Pre          139
Typical              91
Composition Ro       63
Metal- Cpr           20
Concrete Tile        11
Water Proof           4
Wood- FS              4
Concrete              1
Name: ROOF, dtype: int64
```

In [26]:
```python
studydf['ROOF'] = studydf['ROOF'].fillna('Missing')
studydf['ROOF'].value_counts()
```

Out[26]:
```
Missing           40316
Built Up          17633
Metal- Sms        16338
Comp Shingle      16170
Slate              5583
Neopren             903
Shake               383
Clay Tile           309
Shingle             248
Metal- Pre          139
Typical              91
Composition Ro       63
Metal- Cpr           20
Concrete Tile        11
Wood- FS              4
Water Proof           4
Concrete              1
Name: ROOF, dtype: int64
```

Style

This column is mostly redundant to Stories. I will drop it.

In [27]:
```python
studydf['STYLE'].value_counts()
```

Out[27]:
```
2 Story             44150
3 Story              5596
2.5 Story Fin        3920
1 Story              2000
1.5 Story Fin        1158
2.5 Story Unfin       380
4 Story               231
Split Level           158
Split Foyer           123
3.5 Story Fin          84
1.5 Story Unfin        54
Default                23
Bi-Level                8
4.5 Story Fin           5
3.5 Story Unfin         5
Vacant                  2
4.5 Story Unfin         2
Outbuildings            1
Name: STYLE, dtype: int64
```

In [28]:
```python
studydf = studydf.drop('STYLE', 1)
```

Check for percent of nulls

In [29]:
```python
per_missing = studydf.isnull().sum()*100/len(df.iloc[:,1])
per_missing
```

Out[29]:
```
PRICE         0.000000
BATHRM        0.000000
HF_BATHRM     0.000000
HEAT          0.000000
AC            0.000000
NUM_UNITS     0.000000
ROOMS         0.000000
BEDRM         0.000000
AYB           0.070459
EYB           0.000000
STORIES       0.000000
GBA           0.000000
STRUCT        0.000000
GRADE         0.000000
CNDTN         0.000000
EXTWALL       0.000000
ROOF          0.000000
INTWALL       0.000000
KITCHENS      0.000000
FIREPLACES    0.000000
LANDAREA      0.000000
WARD          0.000000
dtype: float64
```

There are a small amout of rows in the average year built column that are nulls. I will drop these rows.

In [30]:
```python
studydf['AYB'].isnull().sum()
```

Out[30]: 112

In [31]:
```python
studydf = studydf.dropna(axis = 0, subset=['AYB'])
```

Change EYB and AYB to year only. Remove float from AYB.

Code from:https://stackoverflow.com/questions/66468687/how-can-i-convert-float-column-to-datetime

In [32]:
```python
studydf['EYB']=pd.to_datetime(studydf['EYB'], format='%Y').dt.year
studydf['AYB']=pd.to_datetime(np.int16(studydf['AYB']), format='%Y')
studydf['AYB']=studydf['AYB'].dt.year
studydf.head()
```

Out[32]:

	PRICE	BATHRM	HF_BATHRM	HEAT	AC	NUM_UNITS	ROOMS	BEDRM	AYB	EYB	...	STRUCT	GRADE	CNDTN	EXTWALL
0	1095000.0	4	0	Warm Cool	Y	2.0	8	4	1910	1972	...	Row Inside	Very Good	Good	Common Brick
2	2100000.0	3	1	Hot Water Rad	Y	2.0	9	5	1910	1984	...	Row Inside	Very Good	Very Good	Common Brick
3	1602000.0	3	1	Hot Water Rad	Y	2.0	8	5	1900	1984	...	Row Inside	Very Good	Good	Common Brick
5	1950000.0	3	2	Hot Water Rad	Y	1.0	10	5	1913	1972	...	Row Inside	Very Good	Good	Common Brick
7	1050000.0	3	1	Hot Water Rad	Y	2.0	8	4	1906	1972	...	Row Inside	Very Good	Average	Common Brick

5 rows × 22 columns

Find and Remove all Outliers

In [33]:
```python
studydf.describe()
```

Out[33]:

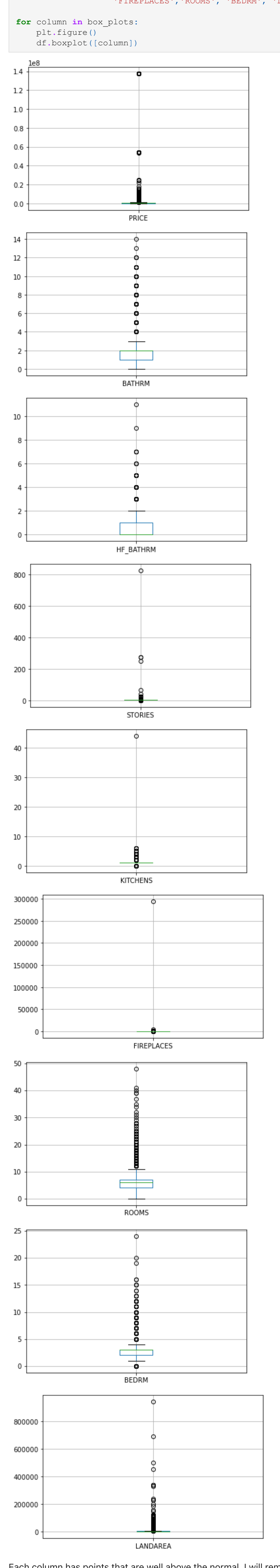
	PRICE	BATHRM	HF_BATHRM	NUM_UNITS	ROOMS	BEDRM	AYB	EYB	ST
count	9.810400e+04	98104.000000	98104.000000	98104.000000	98104.000000	98104.000000	98104.000000	98104.000000	98104.0
mean	9.317931e+05	1.859099	0.444396	1.130290	5.920544	2.605928	1944.903888	1965.771722	2.0
std	7.065297e+06	0.978038	0.576163	0.490895	2.637495	1.393758	36.254098	27.504401	3.0
min	1.000000e+00	0.000000	0.000000	0.000000	0.000000	0.000000	1754.000000	1800.000000	0.0
25%	2.400000e+05	1.000000	0.000000	1.000000	4.000000	2.000000	1917.000000	1954.000000	2.0
50%	3.999990e+05	2.000000	0.000000	1.000000	6.000000	3.000000	1939.000000	1965.000000	2.0
75%	6.520000e+05	2.000000	1.000000	1.000000	7.000000	3.000000	1966.000000	1981.000000	2.0
max	1.374275e+08	12.000000	11.000000	6.000000	31.000000	20.000000	2018.000000	2018.000000	826.0

Stories, Kitchens, and Fireplace max seens outside what is possible. I will create box plots to visualize it.

In [34]:
```python
# Visualize columns to look for outliers
#code from: https://stackoverflow.com/questions/51777217/how-to-plot-a-boxplot-for-each-column-in-a-dataframe

box_plots=studydf[['PRICE','BATHRM', 'HF_BATHRM','STORIES', 'KITCHENS', \
                   'FIREPLACES','ROOMS', 'BEDRM', 'LANDAREA']]

for column in box_plots:
    plt.figure()
    df.boxplot([column])
```



Each column has points that are well above the normal. I will remove these.

In [35]:
```python
studydf = studydf[studydf['PRICE'] < 1000000]
studydf = studydf[studydf['BATHRM'] < 6]
studydf = studydf[studydf['HF_BATHRM'] < 4]
studydf = studydf[studydf['STORIES'] < 4]
studydf = studydf[studydf['KITCHENS'] < 10]
studydf = studydf[studydf['FIREPLACES'] < 10]
studydf = studydf[studydf['PRICE'] < 6000000]
studydf = studydf[studydf['ROOMS'] < 15]
studydf = studydf[studydf['BEDRM'] < 10]
studydf = studydf[studydf['LANDAREA'] < 200000]
studydf.describe()
```

Out[35]:

	PRICE	BATHRM	HF_BATHRM	NUM_UNITS	ROOMS	BEDRM	AYB	EYB	
count	87934.000000	87934.000000	87934.000000	87934.000000	87934.000000	87934.000000	87934.000000	87934.000000	8793
mean	408595.081231	1.705302	0.405258	1.093809	5.560454	2.438022	1945.386142	1964.360566	
std	232898.178734	0.805314	0.549053	0.391899	2.195054	1.241805	35.850118	27.739682	
min	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	1776.000000	1800.000000	
25%	225000.000000	1.000000	0.000000	1.000000	4.000000	2.000000	1919.000000	1954.000000	
50%	370500.000000	2.000000	0.000000	1.000000	6.000000	2.000000	1939.000000	1964.000000	
75%	567230.500000	2.000000	1.000000	1.000000	7.000000	3.000000	1966.000000	1978.000000	
max	999999.000000	5.000000	3.000000	5.000000	14.000000	9.000000	2018.000000	2018.000000	

Check for Duplicates and drop

In [36]:
```python
studydf.duplicated().sum()
```

Out[36]: 1997

In [37]:
```python
studydf = studydf.drop_duplicates()
studydf.shape
```

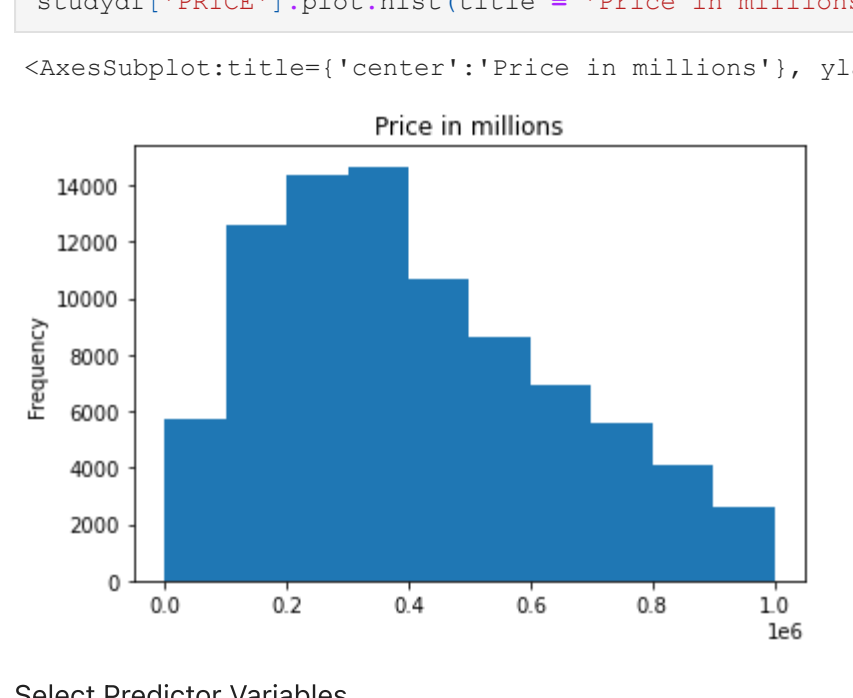
Out[37]: (85937, 22)

Univariate Visulations

Target variable

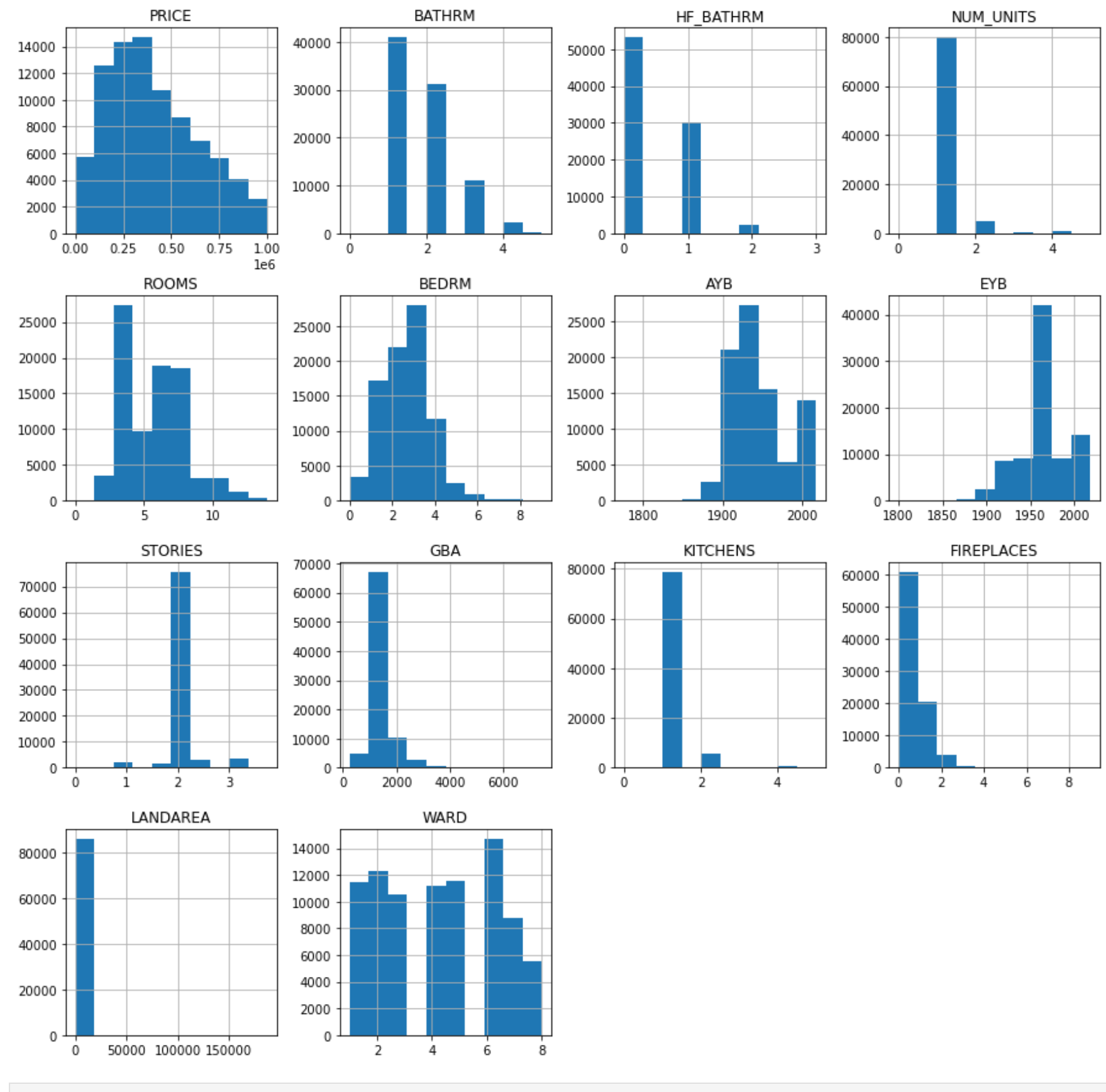
In [38]:
```python
studydf['PRICE'].plot.hist(title = 'Price in millions')
```

Out[38]: <AxesSubplot:title={'center':'Price in millions'}, ylabel='Frequency'>

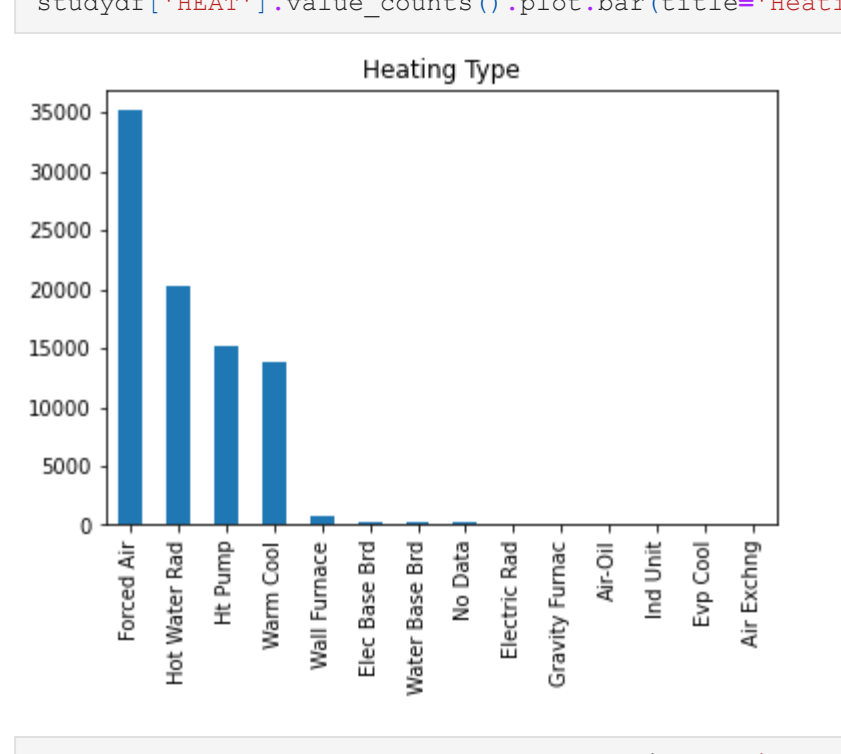


Select Predictor Variables

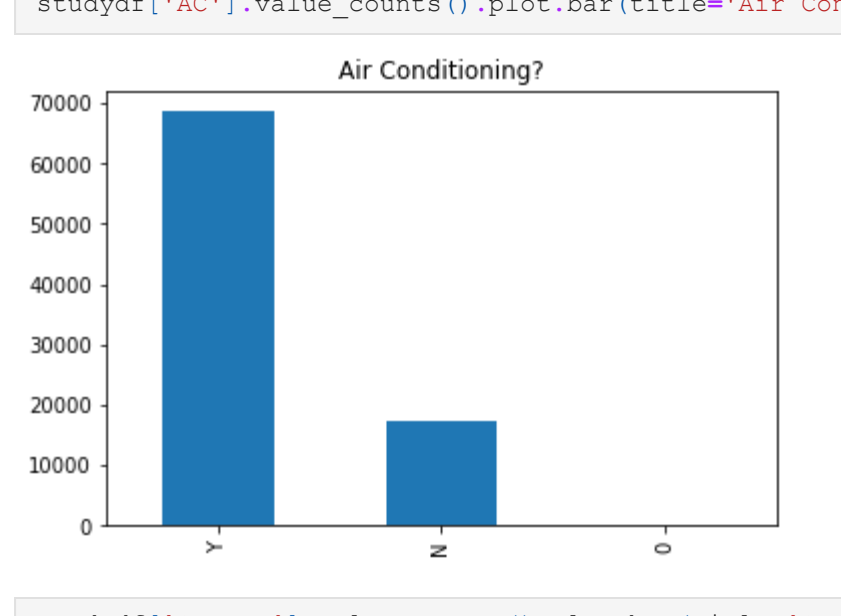
In [39]:
```python
studydf.hist(figsize = (15,15));
```



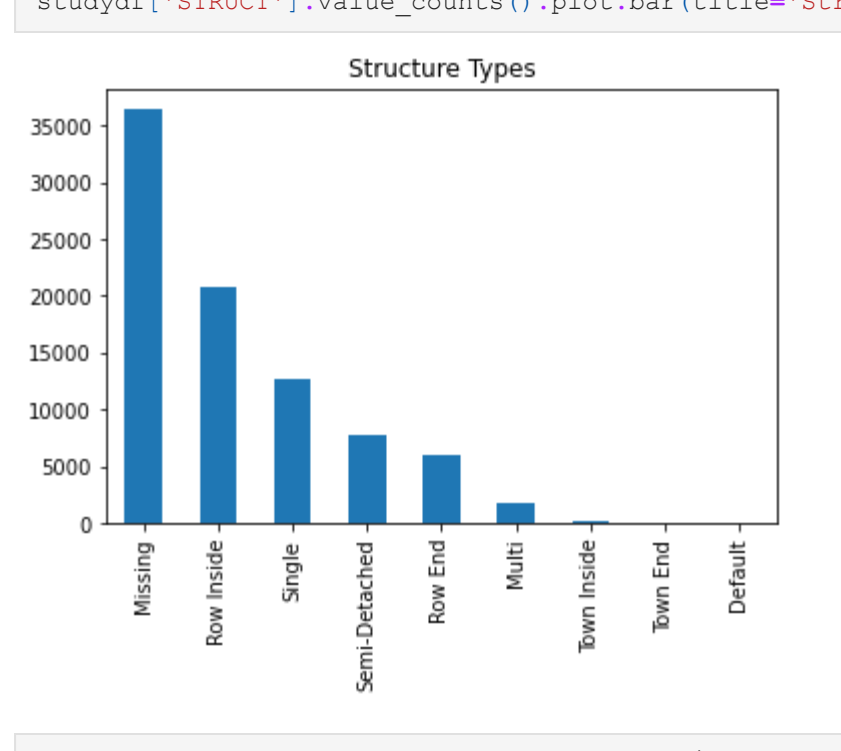
In [40]:
```
studydf['HEAT'].value_counts().plot.bar(title='Heating Type');
```



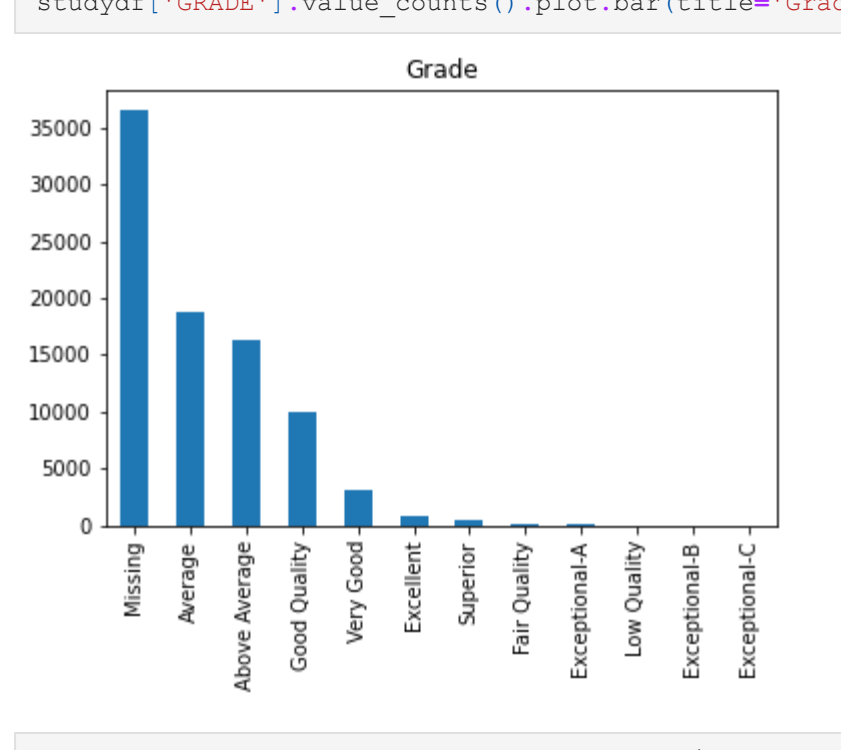
In [41]:
```
studydf['AC'].value_counts().plot.bar(title='Air Conditioning?');
```



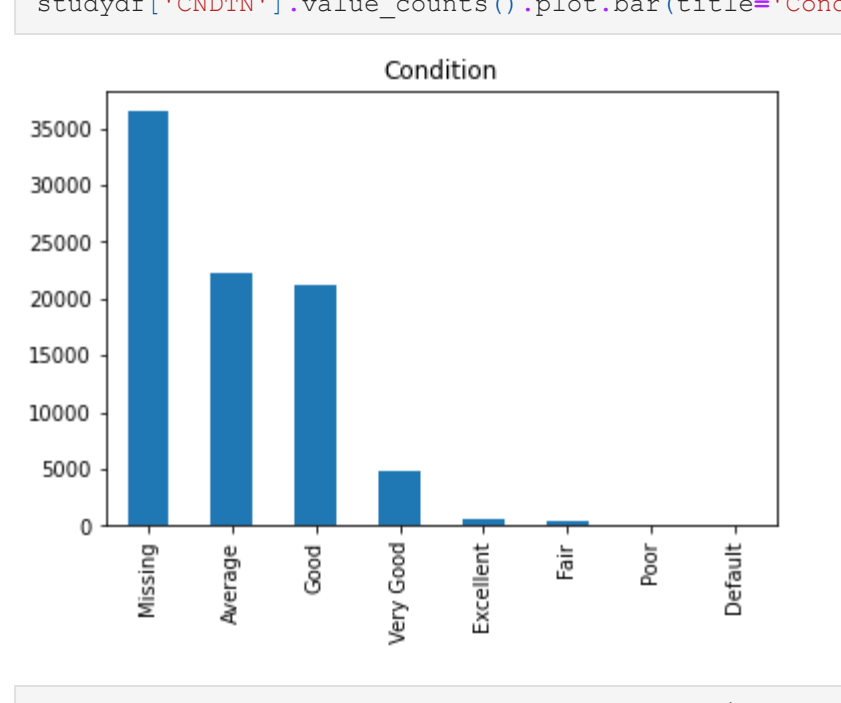
In [42]:
```
studydf['STRUCT'].value_counts().plot.bar(title='Structure Types');
```



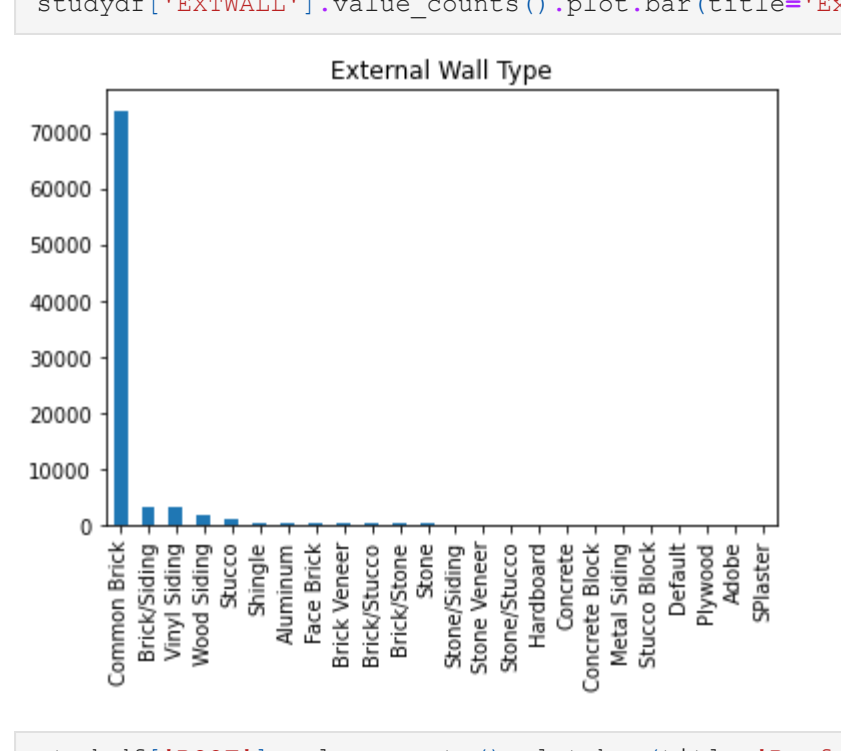
In [43]:
```
studydf['GRADE'].value_counts().plot.bar(title='Grade');
```



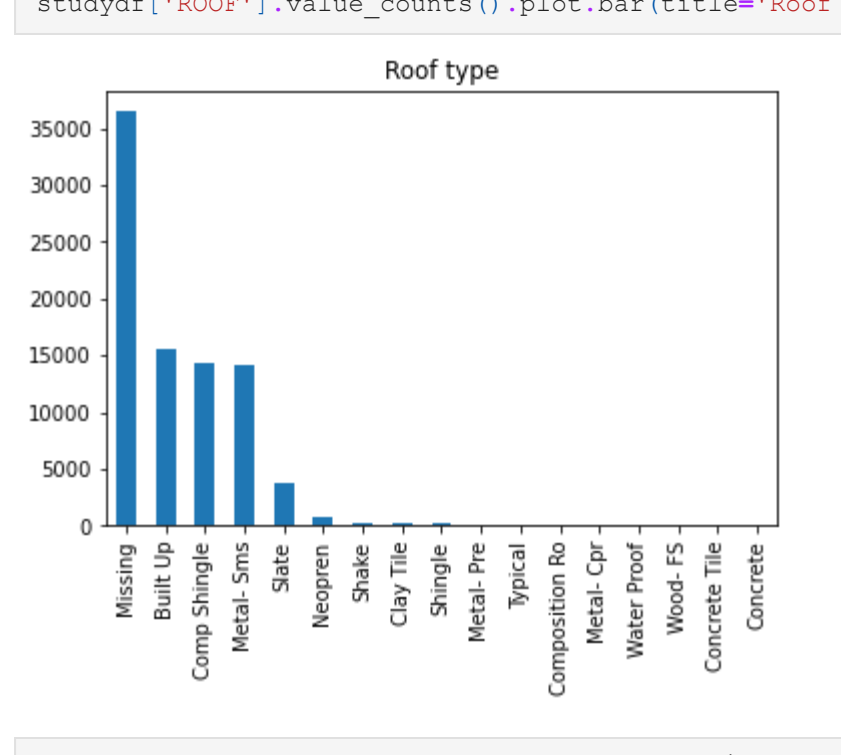
In [44]:
```
studydf['CNDTN'].value_counts().plot.bar(title='Condition');
```



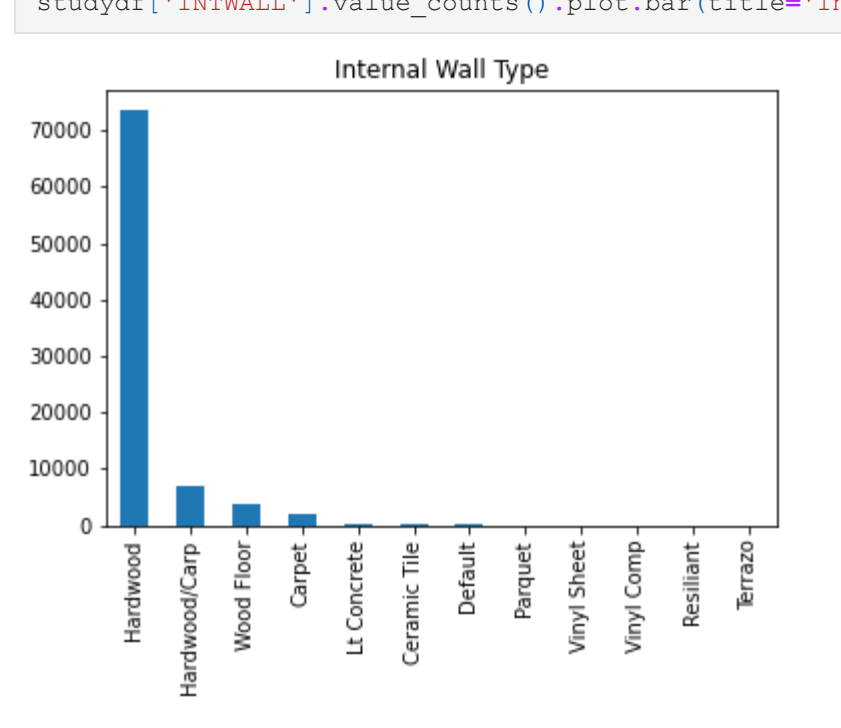
In [45]:
```
studydf['EXTWALL'].value_counts().plot.bar(title='External Wall Type');
```



In [46]:
```
studydf['ROOF'].value_counts().plot.bar(title='Roof type');
```

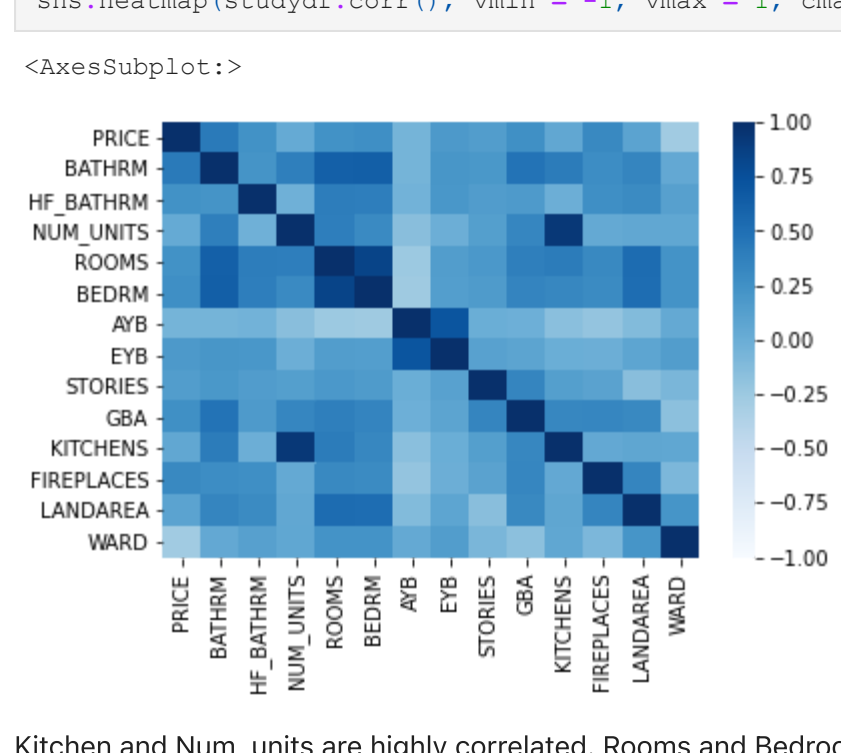


In [47]:
```
studydf['INTWALL'].value_counts().plot.bar(title='Internal Wall Type');
```



In [48]:
```
sns.heatmap(studydf.corr(), vmin = -1, vmax = 1, cmap = 'Blues')
```

Out[48]: <AxesSubplot:>



Kitchen and Num_units are highly correlated. Rooms and Bedrooms are correlated. AYB and EYB are correlated.

Create Dummy Variables

In [49]:
```
study_df = pd.get_dummies(studydf, prefix={'HEAT':'HEAT','AC':'AC','STRUCT':'STRUCT',\
                                            'GRADE':'GRADE','CNDTN':'CNDTN', \
                                             'EXTWALL':'EXTWALL', 'ROOF':'ROOF', \
                                            'INTWALL':'INTWALL'}, \
                          drop_first=True)
study_df.head()
```

Out[49]:

	PRICE	BATHRM	HF_BATHRM	NUM_UNITS	ROOMS	BEDRM	AYB	EYB	STORIES	GBA	...	INTWALL_Default	INTWALL_H
19	592250.0	3	1	2.0	9	3	1908	1969	2.0	1598.0	...	0	
20	33232.0	3	1	1.0	14	5	1880	1987	3.0	3465.0	...	0	
22	907400.0	1	0	1.0	6	3	1880	1967	2.0	1790.0	...	0	
25	20000.0	1	0	1.0	8	4	1880	1967	2.0	2099.0	...	0	
27	703700.0	3	0	4.0	9	3	1900	1967	3.0	2520.0	...	0	

5 rows × 105 columns

In [50]:
```
study_df.shape
```

Out[50]: (85937, 105)

Save Prepared Data

In [51]:
```
study_df.to_csv('dc_prepared_data.csv')
```

Analysis

Split into independent and dependent features

In [52]:
```
y = study_df['PRICE']
X = study_df.drop('PRICE', axis = 1)
```

Add a constant term

In [53]:
```
X = sm.add_constant(X)
```

Build and train model

In [54]:
```
model = sm.OLS(y, X)
results = model.fit()
```

In [55]:
```
results.summary()
```

Out[55]:

OLS Regression Results

Dep. Variable:	PRICE	**R-squared:**	0.445
Model:	OLS	**Adj. R-squared:**	0.445
Method:	Least Squares	**F-statistic:**	682.3
Date:	Sun, 22 May 2022	**Prob (F-statistic):**	0.00
Time:	11:04:55	**Log-Likelihood:**	-1.1589e+06
No. Observations:	85937	**AIC:**	2.318e+06
Df Residuals:	85835	**BIC:**	2.319e+06
Df Model:	101		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-1.167e+06	2.57e+05	-4.542	0.000	-1.67e+06	-6.64e+05
BATHRM	8.504e+04	1164.216	73.048	0.000	8.28e+04	8.73e+04
HF_BATHRM	4.494e+04	1312.525	34.240	0.000	4.24e+04	4.75e+04
NUM_UNITS	-6.496e+04	4353.440	-14.921	0.000	-7.35e+04	-5.64e+04
ROOMS	3189.8637	627.744	5.081	0.000	1959.492	4420.236
BEDRM	2.338e+04	1027.076	22.763	0.000	2.14e+04	2.54e+04
AYB	-1436.2863	39.851	-36.042	0.000	-1514.393	-1358.179
EYB	2259.1784	46.905	48.165	0.000	2167.245	2351.112
STORIES	-2.505e+04	2673.190	-9.371	0.000	-3.03e+04	-1.98e+04
GBA	-53.5925	2.237	-23.963	0.000	-57.976	-49.209
KITCHENS	1.479e+04	4105.785	3.601	0.000	6738.654	2.28e+04
FIREPLACES	3.814e+04	1164.039	32.762	0.000	3.59e+04	4.04e+04
LANDAREA	-2.9952	0.503	-5.959	0.000	-3.980	-2.010
WARD	-2.485e+04	332.787	-74.660	0.000	-2.55e+04	-2.42e+04
HEAT_Air-Oil	3933.4964	4.74e+04	0.083	0.934	-8.9e+04	9.69e+04
HEAT_Elec Base Brd	1.042e+04	4.08e+04	0.255	0.799	-6.96e+04	9.05e+04
HEAT_Electric Rad	3.087e+04	4.35e+04	0.710	0.478	-5.43e+04	1.16e+05
HEAT_Evp Cool	-8.077e+04	5.24e+04	-1.540	0.124	-1.84e+05	2.2e+04
HEAT_Forced Air	3429.0719	3.92e+04	0.087	0.930	-7.34e+04	8.03e+04
HEAT_Gravity Furnac	-1428.2359	4.63e+04	-0.031	0.975	-9.22e+04	8.93e+04
HEAT_Hot Water Rad	1.503e+04	3.92e+04	0.383	0.702	-6.18e+04	9.19e+04
HEAT_Ht Pump	1.97e+04	3.92e+04	0.502	0.616	-5.72e+04	9.66e+04
HEAT_Ind Unit	3962.0647	4.96e+04	0.080	0.936	-9.32e+04	1.01e+05
HEAT_No Data	-3.195e+04	4.13e+04	-0.774	0.439	-1.13e+05	4.89e+04
HEAT_Wall Furnace	-5.775e+04	3.98e+04	-1.452	0.147	-1.36e+05	2.02e+04
HEAT_Warm Cool	-2.906e+04	3.92e+04	-0.741	0.459	-1.06e+05	4.79e+04
HEAT_Water Base Brd	-4.042e+04	4.09e+04	-0.988	0.323	-1.21e+05	3.98e+04
AC_N	-8.469e+04	3.08e+04	-2.751	0.006	-1.45e+05	-2.43e+04
AC_Y	-3.41e+04	3.08e+04	-1.109	0.268	-9.44e+04	2.62e+04
STRUCT_Missing	2316.3559	4.4e+04	0.053	0.958	-8.4e+04	8.86e+04
STRUCT_Multi	-7.783e+04	1.76e+05	-0.442	0.659	-4.23e+05	2.67e+05
STRUCT_Row End	-9.917e+04	1.76e+05	-0.563	0.573	-4.44e+05	2.46e+05
STRUCT_Row Inside	-8.923e+04	1.76e+05	-0.507	0.612	-4.34e+05	2.56e+05
STRUCT_Semi-Detached	-1.143e+05	1.76e+05	-0.649	0.516	-4.59e+05	2.31e+05
STRUCT_Single	-8.933e+04	1.76e+05	-0.508	0.612	-4.34e+05	2.56e+05
STRUCT_Town End	-9.207e+04	1.77e+05	-0.519	0.603	-4.39e+05	2.55e+05
STRUCT_Town Inside	-1.114e+05	1.77e+05	-0.631	0.528	-4.57e+05	2.35e+05
GRADE_Average	-3.146e+04	2037.150	-15.441	0.000	-3.54e+04	-2.75e+04
GRADE_Excellent	1.01e+05	6956.752	14.517	0.000	8.74e+04	1.15e+05
GRADE_Exceptional-A	1.017e+05	2.12e+04	4.802	0.000	6.02e+04	1.43e+05
GRADE_Exceptional-B	2.898e+05	1.01e+05	2.879	0.004	9.25e+04	4.87e+05
GRADE_Exceptional-C	1.703e+05	1.75e+05	0.976	0.329	-1.72e+05	5.12e+05
GRADE_Fair Quality	-4.088e+04	1.97e+04	-2.075	0.038	-7.95e+04	-2256.645
GRADE_Good Quality	8.025e+04	2397.936	33.468	0.000	7.56e+04	8.5e+04
GRADE_Low Quality	-3.952e+04	1.03e+05	-0.384	0.701	-2.41e+05	1.62e+05
GRADE_Missing	2316.3559	4.4e+04	0.053	0.958	-8.4e+04	8.86e+04
GRADE_Superior	1.02e+05	8191.472	12.448	0.000	8.59e+04	1.18e+05
GRADE_Very Good	1.162e+05	3811.649	30.494	0.000	1.09e+05	1.24e+05
CNDTN_Default	2.522e+05	1.74e+05	1.448	0.148	-8.92e+04	5.94e+05
CNDTN_Excellent	1.213e+05	8030.690	15.099	0.000	1.06e+05	1.37e+05
CNDTN_Fair	1.499e+04	8291.135	1.808	0.071	-1256.619	3.12e+04
CNDTN_Good	7.12e+04	1872.133	38.032	0.000	6.75e+04	7.49e+04
CNDTN_Missing	2316.3559	4.4e+04	0.053	0.958	-8.4e+04	8.86e+04
CNDTN_Poor	6.7e+04	1.99e+04	3.366	0.001	2.8e+04	1.06e+05
CNDTN_Very Good	1.283e+05	3233.116	39.676	0.000	1.22e+05	1.35e+05
EXTWALL_Aluminum	-1.244e+04	1.74e+05	-0.071	0.943	-3.54e+05	3.29e+05
EXTWALL_Brick Veneer	-3.656e+04	1.74e+05	-0.210	0.834	-3.78e+05	3.05e+05
EXTWALL_Brick/Siding	-5.048e+04	1.74e+05	-0.290	0.772	-3.92e+05	2.91e+05
EXTWALL_Brick/Stone	-3.511e+04	1.74e+05	-0.201	0.841	-3.77e+05	3.07e+05
EXTWALL_Brick/Stucco	8269.5580	1.74e+05	0.047	0.962	-3.34e+05	3.5e+05
EXTWALL_Common Brick	-1.579e+04	1.74e+05	-0.091	0.928	-3.57e+05	3.26e+05
EXTWALL_Concrete	-2.762e+04	1.77e+05	-0.156	0.876	-3.74e+05	3.19e+05
EXTWALL_Concrete Block	-7.189e+04	1.77e+05	-0.406	0.685	-4.19e+05	2.75e+05
EXTWALL_Default	-1.306e+05	1.85e+05	-0.707	0.480	-4.93e+05	2.31e+05
EXTWALL_Face Brick	-7.612e+04	1.74e+05	-0.436	0.662	-4.18e+05	2.66e+05
EXTWALL_Hardboard	4979.6343	1.76e+05	0.028	0.977	-3.39e+05	3.49e+05
EXTWALL_Metal Siding	-1.4e+04	1.78e+05	-0.079	0.937	-3.62e+05	3.34e+05
EXTWALL_Plywood	2.356e+04	1.95e+05	0.121	0.904	-3.58e+05	4.06e+05
EXTWALL_SPlaster	-4.095e+04	2.48e+05	-0.165	0.869	-5.27e+05	4.45e+05
EXTWALL_Shingle	-1.426e+04	1.74e+05	-0.082	0.935	-3.56e+05	3.27e+05
EXTWALL_Stone	-4.413e+04	1.75e+05	-0.253	0.800	-3.86e+05	2.98e+05
EXTWALL_Stone Veneer	-1.026e+04	1.75e+05	-0.059	0.953	-3.53e+05	3.32e+05
EXTWALL_Stone/Siding	-2.219e+04	1.75e+05	-0.127	0.899	-3.65e+05	3.2e+05
EXTWALL_Stone/Stucco	-1364.5345	1.75e+05	-0.008	0.994	-3.44e+05	3.42e+05
EXTWALL_Stucco	-2.019e+04	1.74e+05	-0.116	0.908	-3.62e+05	3.21e+05
EXTWALL_Stucco Block	-8938.4839	1.79e+05	-0.050	0.960	-3.59e+05	3.41e+05
EXTWALL_Vinyl Siding	-6.916e+04	1.74e+05	-0.397	0.691	-4.11e+05	2.72e+05
EXTWALL_Wood Siding	1.018e+04	1.74e+05	0.058	0.953	-3.31e+05	3.52e+05
ROOF_Clay Tile	-4.841e+04	1.29e+04	-3.759	0.000	-7.37e+04	-2.32e+04
ROOF_Comp Shingle	-3.662e+04	2799.982	-13.080	0.000	-4.21e+04	-3.11e+04
ROOF_Composition Ro	-7.597e+04	2.58e+04	-2.950	0.003	-1.26e+05	-2.55e+04
ROOF_Concrete	-8.183e+04	1.74e+05	-0.470	0.638	-4.23e+05	2.6e+05
ROOF_Concrete Tile	-6.146e+04	1.23e+05	-0.498	0.618	-3.03e+05	1.8e+05
ROOF_Metal- Cpr	1.35e+04	6.6e+04	0.205	0.838	-1.16e+05	1.43e+05
ROOF_Metal- Pre	-2.187e+04	1.61e+04	-1.354	0.176	-5.35e+04	9778.547
ROOF_Metal- Sms	-3765.6527	2172.476	-1.733	0.083	-8023.687	492.381
ROOF_Missing	2316.3559	4.4e+04	0.053	0.958	-8.4e+04	8.86e+04
ROOF_Neopren	8.59e+04	6852.442	12.535	0.000	7.25e+04	9.93e+04
ROOF_Shake	-2.423e+04	1.06e+04	-2.277	0.023	-4.51e+04	-3373.978
ROOF_Shingle	-3838.3736	1.33e+04	-0.288	0.773	-3e+04	2.23e+04
ROOF_Slate	-2295.5416	4005.275	-0.573	0.567	-1.01e+04	5554.763
ROOF_Typical	-1.798e+04	1.99e+04	-0.904	0.366	-5.7e+04	2.1e+04
ROOF_Water Proof	-7.398e+04	8.71e+04	-0.850	0.396	-2.45e+05	9.67e+04
ROOF_Wood- FS	1.979e+05	1.01e+05	1.967	0.049	684.064	3.95e+05
INTWALL_Ceramic Tile	5.605e+04	2.98e+04	1.883	0.060	-2286.934	1.14e+05
INTWALL_Default	8.763e+04	3.23e+04	2.714	0.007	2.43e+04	1.51e+05
INTWALL_Hardwood	6.577e+04	4346.585	15.130	0.000	5.72e+04	7.43e+04
INTWALL_Hardwood/Carp	2.786e+04	4691.132	5.939	0.000	1.87e+04	3.71e+04
INTWALL_Lt Concrete	2.591e+04	2.65e+04	0.979	0.328	-2.6e+04	7.78e+04
INTWALL_Parquet	1.97e+05	6.17e+04	3.193	0.001	7.61e+04	3.18e+05
INTWALL_Resiliant	-2.581e+04	7.81e+04	-0.331	0.741	-1.79e+05	1.27e+05
INTWALL_Terrazo	3.516e+05	1.74e+05	2.017	0.044	1e+04	6.93e+05
INTWALL_Vinyl Comp	1.583e+05	7.29e+04	2.170	0.030	1.53e+04	3.01e+05
INTWALL_Vinyl Sheet	1.304e+05	6.6e+04	1.975	0.048	959.916	2.6e+05
INTWALL_Wood Floor	4.835e+04	5156.224	9.377	0.000	3.82e+04	5.85e+04

Omnibus:	772.452	**Durbin-Watson:**	1.506
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	986.070
Skew:	0.149	**Prob(JB):**	7.55e-215
Kurtosis:	3.432	**Cond. No.**	1.38e+16

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The smallest eigenvalue is 6.78e-21. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.

In [56]:
```
pred_y = results.predict()
```

In [57]:
```
metrics.mean_absolute_error(y, pred_y)
```

Out[57]: 135808.04005241138

In [58]:
```
residuals = y - pred_y
ax1 = sns.residplot(x = y, y = residuals, scatter_kws={'alpha': 0.08})
ax1.set(xlabel = 'Price', ylabel = 'Residuals');
```