Importing Modules and Data

In [44]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from matplotlib.pyplot import figure
```

In [2]:
```
filename = r"C:\Users\Hussain\Desktop\Project 2 - Data Cleaning in SQL and EDA in Python\Nashville Housing Data Cleaned
nashville = pd.read_csv(filename)

nashville.head()
```

Out[2]:

	UniqueID	ParcelID	LandUse	SaleDate	SalePrice	LegalReference	SoldAsVacant	OwnerName	Acreage	LandValue	...	TotalValue	YearBuilt
0	2045	007 00 0 125.00	SINGLE FAMILY	2013-04-09	240000	20130412-0036474	No	FRAZIER, CYRENTHA LYNETTE	2.3	50000.0	...	235700.0	1986.0
1	16918	007 00 0 130.00	SINGLE FAMILY	2014-06-10	366000	20140619-0053768	No	BONER, CHARLES & LESLIE	3.5	50000.0	...	319000.0	1998.0
2	54582	007 00 0 138.00	SINGLE FAMILY	2016-09-26	435000	20160927-0101718	No	WILSON, JAMES E. & JOANNE	2.9	50000.0	...	298000.0	1987.0
3	43070	007 00 0 143.00	SINGLE FAMILY	2016-01-29	255000	20160129-0008913	No	BAKER, JAY K. & SUSAN E.	2.6	50000.0	...	197300.0	1985.0
4	22714	007 00 0 149.00	SINGLE FAMILY	2014-10-10	278000	20141015-0095255	No	POST, CHRISTOPHER M. & SAMANTHA C.	2.0	50000.0	...	202300.0	1984.0

5 rows × 21 columns

In [3]:
```
df = pd.DataFrame(nashville)
df.dtypes
```

Out[3]:
```
UniqueID                int64
ParcelID               object
LandUse                object
SaleDate               object
SalePrice              object
LegalReference         object
SoldAsVacant           object
OwnerName              object
Acreage               float64
LandValue             float64
BuildingValue         float64
TotalValue            float64
YearBuilt             float64
Bedrooms              float64
FullBath              float64
HalfBath               object
PropertySplitAddress   object
PropertySplitCity      object
OwnerSplitAddress      object
OwnerSplitCity         object
OwnerSplitState        object
dtype: object
```

Missing Values

In [7]:
```
df.isna().sum()
```

Out[7]:
```
UniqueID                0
ParcelID                0
LandUse                 0
SaleDate                0
SalePrice               0
LegalReference          0
SoldAsVacant            0
OwnerName           31158
Acreage             30404
LandValue           30404
BuildingValue       30404
TotalValue          30404
YearBuilt           32255
Bedrooms            32261
FullBath            32143
HalfBath                0
PropertySplitAddress    0
PropertySplitCity       0
OwnerSplitAddress   30404
OwnerSplitCity      30404
OwnerSplitState     30404
dtype: int64
```

In [8]:
```
# Filling YearBuilt column by finding the most commonly occurring month and filling it in place of the missing values
```

In [9]:
```
year_mode = df['YearBuilt'].mode()[0]
year_mode
```

Out[9]:
```
1950.0
```

In [10]:
```
df['YearBuilt'].fillna(year_mode, inplace = True)
```

In [11]:
```
df.isna().sum()
```

Out[11]:
```
UniqueID                0
ParcelID                0
LandUse                 0
SaleDate                0
SalePrice               0
LegalReference          0
SoldAsVacant            0
OwnerName               31158
Acreage                 30404
LandValue               30404
BuildingValue           30404
TotalValue              30404
YearBuilt               0
Bedrooms                32261
FullBath                32143
HalfBath                0
PropertySplitAddress    0
PropertySplitCity       0
OwnerSplitAddress       30404
OwnerSplitCity          30404
OwnerSplitState         30404
dtype: int64
```

In [12]:
```
pd.set_option('display.max_columns', None)
```

In [13]:
```
df.head()
```

Out[13]:

	UniqueID	ParcelID	LandUse	SaleDate	SalePrice	LegalReference	SoldAsVacant	OwnerName	Acreage	LandValue	BuildingValue	TotalValue
0	2045	007 00 0 125.00	SINGLE FAMILY	2013-04-09	240000	20130412-0036474	No	FRAZIER, CYRENTHA LYNETTE	2.3	50000.0	168200.0	235700.0
1	16918	007 00 0 130.00	SINGLE FAMILY	2014-06-10	366000	20140619-0053768	No	BONER, CHARLES & LESLIE	3.5	50000.0	264100.0	319000.0
2	54582	007 00 0 138.00	SINGLE FAMILY	2016-09-26	435000	20160927-0101718	No	WILSON, JAMES E. & JOANNE	2.9	50000.0	216200.0	298000.0
3	43070	007 00 0 143.00	SINGLE FAMILY	2016-01-29	255000	20160129-0008913	No	BAKER, JAY K. & SUSAN E.	2.6	50000.0	147300.0	197300.0
4	22714	007 00 0 149.00	SINGLE FAMILY	2014-10-10	278000	20141015-0095255	No	POST, CHRISTOPHER M. & SAMANTHA C.	2.0	50000.0	152300.0	202300.0

In [14]:
```
# Dropping missing values for OwnerName column since
# there is nothing that can be done if owners have not fill in their names

df = df[~df['OwnerName'].isnull()].copy()

df.isnull().sum()
```

Out[14]:
```
UniqueID                0
ParcelID                0
LandUse                 0
SaleDate                0
SalePrice               0
LegalReference          0
SoldAsVacant            0
OwnerName               0
Acreage                 0
LandValue               0
BuildingValue           0
TotalValue              0
YearBuilt               0
Bedrooms             1408
FullBath             1298
HalfBath                0
PropertySplitAddress    0
PropertySplitCity       0
OwnerSplitAddress       0
OwnerSplitCity          0
OwnerSplitState         0
dtype: int64
```

In [15]:
```
# Dropping missing values for Bedrooms and FullBath columns since
# using Measures of Central Tendency to fill in the values may distort our analysis

df = df[~df['Bedrooms'].isnull()].copy()
df = df[~df['FullBath'].isnull()].copy()

df.isnull().sum()
```

Out[15]:
```
UniqueID                0
ParcelID                0
LandUse                 0
SaleDate                0
SalePrice               0
LegalReference          0
SoldAsVacant            0
OwnerName               0
Acreage                 0
LandValue               0
BuildingValue           0
TotalValue              0
YearBuilt               0
Bedrooms                0
FullBath                0
HalfBath                0
PropertySplitAddress    0
PropertySplitCity       0
OwnerSplitAddress       0
OwnerSplitCity          0
OwnerSplitState         0
dtype: int64
```

Converting Data Types

In [23]:
```
df['Acreage'] = df['Acreage'].astype(int)
df['LandValue'] = df['LandValue'].astype(int)
df['BuildingValue'] = df['BuildingValue'].astype(int)
df['TotalValue'] = df['TotalValue'].astype(int)
df['YearBuilt'] = df['YearBuilt'].astype(int)

df.dtypes
```

Out[23]:
```
UniqueID                int64
ParcelID               object
LandUse                object
SaleDate               object
SalePrice              object
LegalReference         object
SoldAsVacant           object
OwnerName              object
Acreage                 int32
LandValue               int32
BuildingValue           int32
TotalValue              int32
YearBuilt               int32
Bedrooms              float64
FullBath              float64
HalfBath               object
PropertySplitAddress   object
PropertySplitCity      object
OwnerSplitAddress      object
OwnerSplitCity         object
OwnerSplitState        object
dtype: object
```

In [29]:
```
df['SalePrice'] = df['SalePrice'].str.replace(",","")
df['SalePrice'] = df['SalePrice'].str.replace("$","")
df['SalePrice'] = df['SalePrice'].astype(int)
df.dtypes
```

```
<ipython-input-29-74262e52d198>:2: FutureWarning: The default value of regex will change from True to False in a future
version. In addition, single character regular expressions will*not* be treated as literal strings when regex=True.
  df['SalePrice'] = df['SalePrice'].str.replace("$","")
```

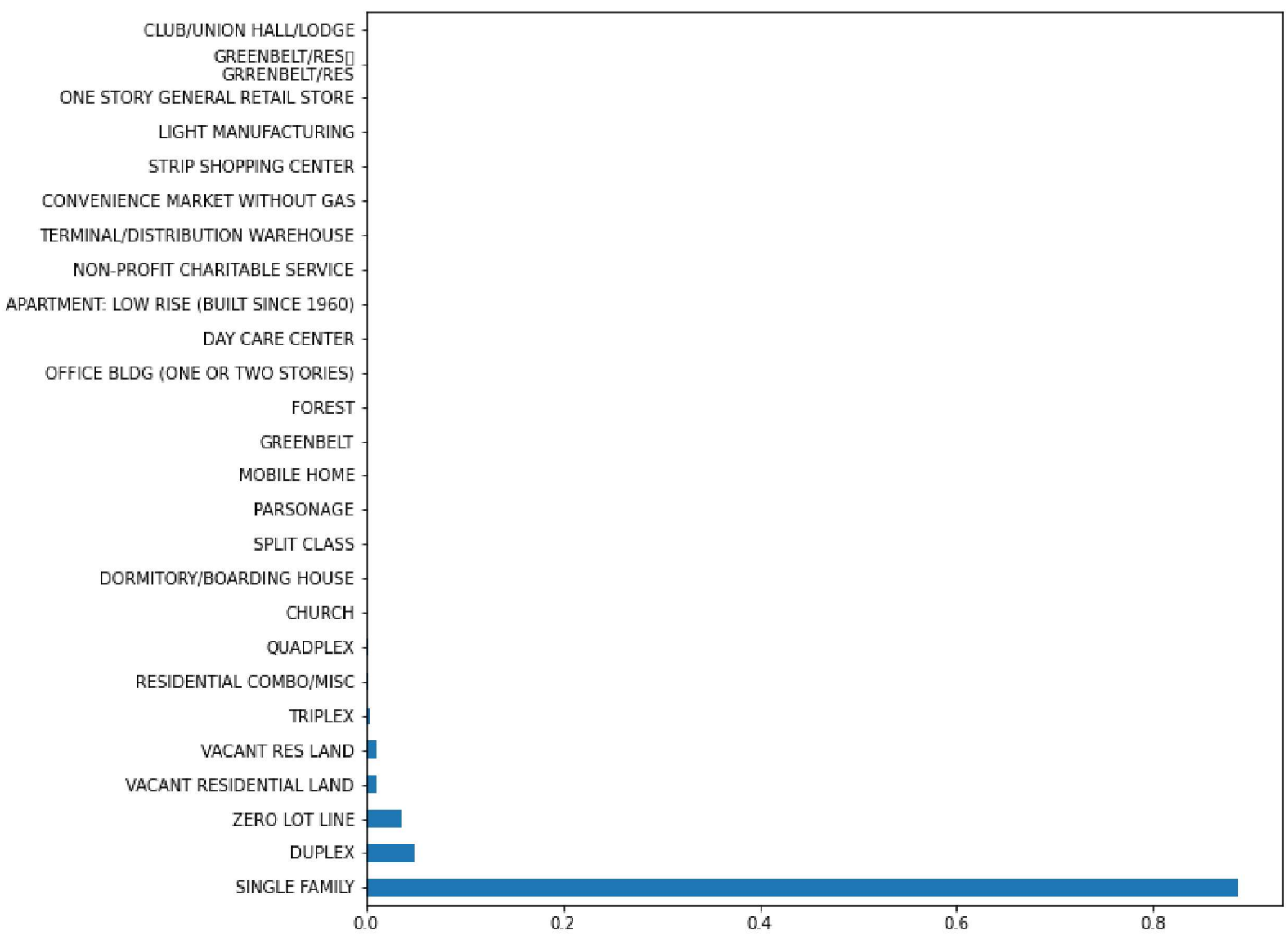
Out[29]:
```
UniqueID                 int64
ParcelID                object
LandUse                 object
SaleDate                object
SalePrice                int32
LegalReference          object
SoldAsVacant            object
OwnerName               object
Acreage                  int32
LandValue                int32
BuildingValue            int32
TotalValue               int32
YearBuilt                int32
Bedrooms               float64
FullBath               float64
HalfBath                object
PropertySplitAddress    object
PropertySplitCity       object
OwnerSplitAddress       object
OwnerSplitCity          object
OwnerSplitState         object
dtype: object
```

Univariate and Multivariate Analysis

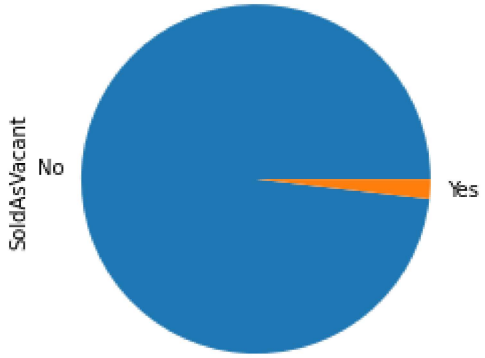
1. Univariate Analysis

In [18]:
```
# Percentage of each land use category
LandUse_percent = df['LandUse'].value_counts(normalize = True)

# Plotting the bar chart of percentage land use categories
plt.figure(figsize = (10,10))

LandUse_percent.plot.barh()
plt.show()
```

```
C:\Users\Hussain\anaconda3\lib\site-packages\matplotlib\backends\backend_agg.py:238: RuntimeWarning: Glyph 13 missing f
rom current font.
  font.set_text(s, 0.0, flags=flags)
C:\Users\Hussain\anaconda3\lib\site-packages\matplotlib\backends\backend_agg.py:201: RuntimeWarning: Glyph 13 missing f
rom current font.
  font.set_text(s, 0, flags=flags)
```


CLUB/UNION HALL/LODGE
GREENBELT/RES
GRRENBELT/RES
ONE STORY GENERAL RETAIL STORE
LIGHT MANUFACTURING
STRIP SHOPPING CENTER
CONVENIENCE MARKET WITHOUT GAS
TERMINAL/DISTRIBUTION WAREHOUSE
NON-PROFIT CHARITABLE SERVICE
APARTMENT: LOW RISE (BUILT SINCE 1960)
DAY CARE CENTER
OFFICE BLDG (ONE OR TWO STORIES)
FOREST
GREENBELT
MOBILE HOME
PARSONAGE
SPLIT CLASS
DORMITORY/BOARDING HOUSE
CHURCH
QUADPLEX
RESIDENTIAL COMBO/MISC
TRIPLEX
VACANT RES LAND
VACANT RESIDENTIAL LAND
ZERO LOT LINE
DUPLEX
SINGLE FAMILY
0.0
0.2
0.4
0.6
0.8

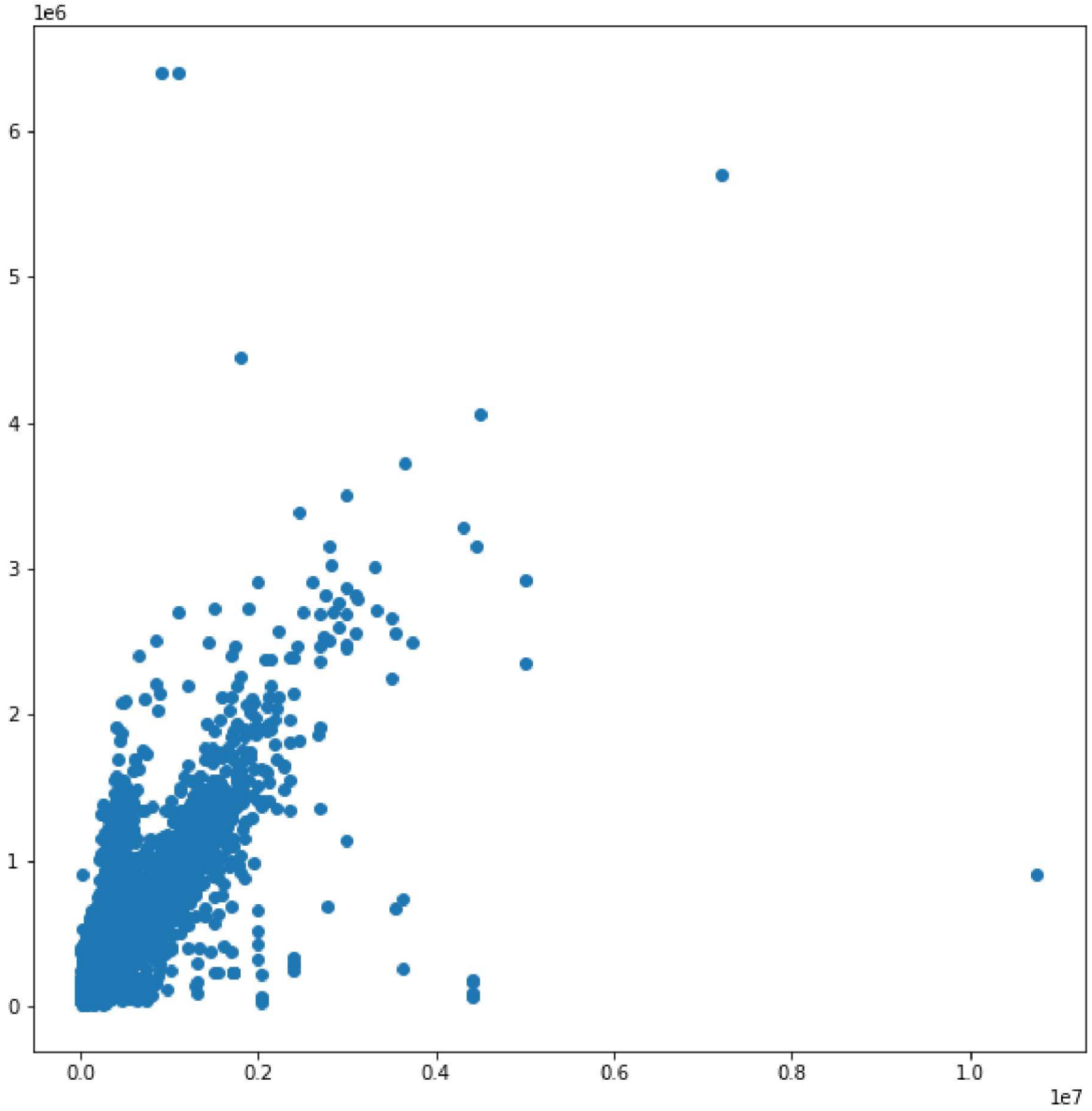
In [19]:
```
# Percentage of each SoldAsVacant category
SoldAsVacant_percentage = df['SoldAsVacant'].value_counts(normalize = True)

# Plotting the Pie Chart of SoldAsVacant category
SoldAsVacant_percentage.plot.pie()
plt.show()
```



2. Bivariate Analysis

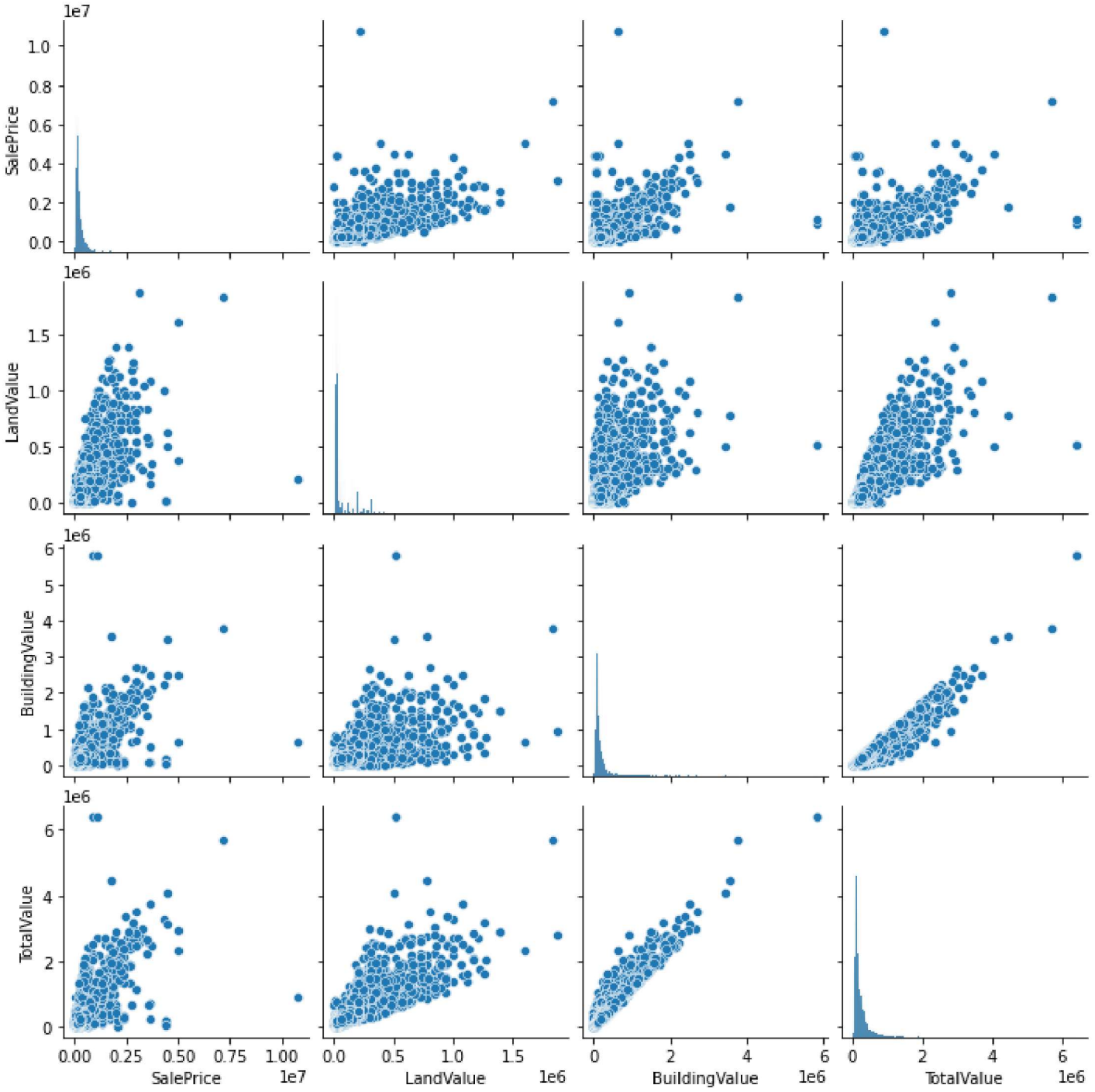
(i) Numeric-Numeric Analysis

In [30]:
```
# Plotting the Scatter Plot of SalePrice and TotalValue variables
plt.figure(figsize = (10,10))
plt.scatter(df['SalePrice'],df['TotalValue'])
plt.show()
```

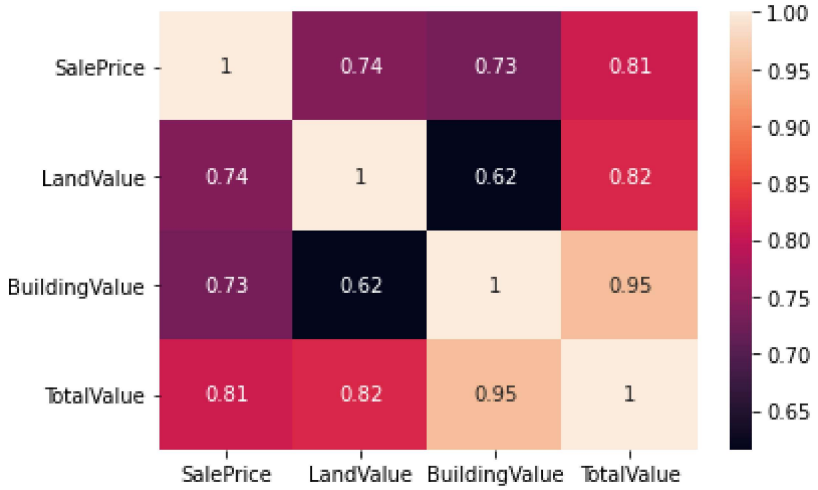


In [31]:
```
# Plotting the pair plot of SalePrice, LandValue, BuildingValue, and TotalValue variables
plt.figure(figsize = (10,10))
sns.pairplot(data = df, vars = ['SalePrice', 'LandValue', 'BuildingValue', 'TotalValue'])
plt.show()
```

```
<Figure size 720x720 with 0 Axes>
```


SalePrice
LandValue
BuildingValue
TotalValue
1e7
1e6
0.00 0.25 0.50 0.75 1.00
0.0 0.5 1.0 1.5
0 2 4 6

In [52]:
```
# Creating a Matrix using SalePrice, LandValue, BuildingValue, and TotalValue variables as rows and columns
corr_matrix = df[['SalePrice', 'LandValue', 'BuildingValue', 'TotalValue']].corr()
corr_matrix

# Plotting the Correlation Matrix of SalePrice, LandValue, BuildingValue, and TotalValue variables
sns.heatmap(corr_matrix, annot = True)
plt.show()
```



(ii) Numeric - Categorical Analysis

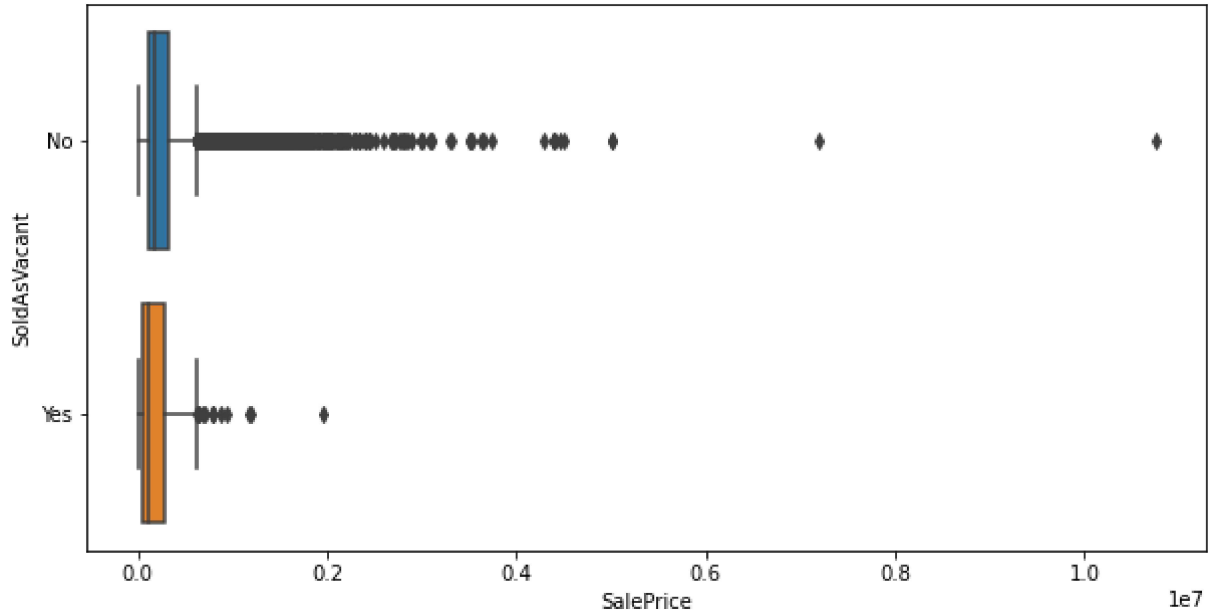
In [33]:
```
# Grouping by SoldAsVacant to find the mean of the SalePrice with response to No and Yes separately
df.groupby('SoldAsVacant')['SalePrice'].mean()
```

Out[33]:
```
SoldAsVacant
No     276710.633979
Yes    184566.226328
Name: SalePrice, dtype: float64
```

In [34]:
```
# Grouping by SoldAsVacant to find the median of the SalePrice with response to No and Yes separately
df.groupby('SoldAsVacant')['SalePrice'].median()
```

Out[34]:
```
SoldAsVacant
No     185000
Yes    106062
Name: SalePrice, dtype: int32
```

In [41]:
```
# Plotting the box plot of SalePrice for No and Yes responses
plt.figure(figsize = (10,5))
sns.boxplot(y = df['SoldAsVacant'], x = df['SalePrice'])
plt.show()
```



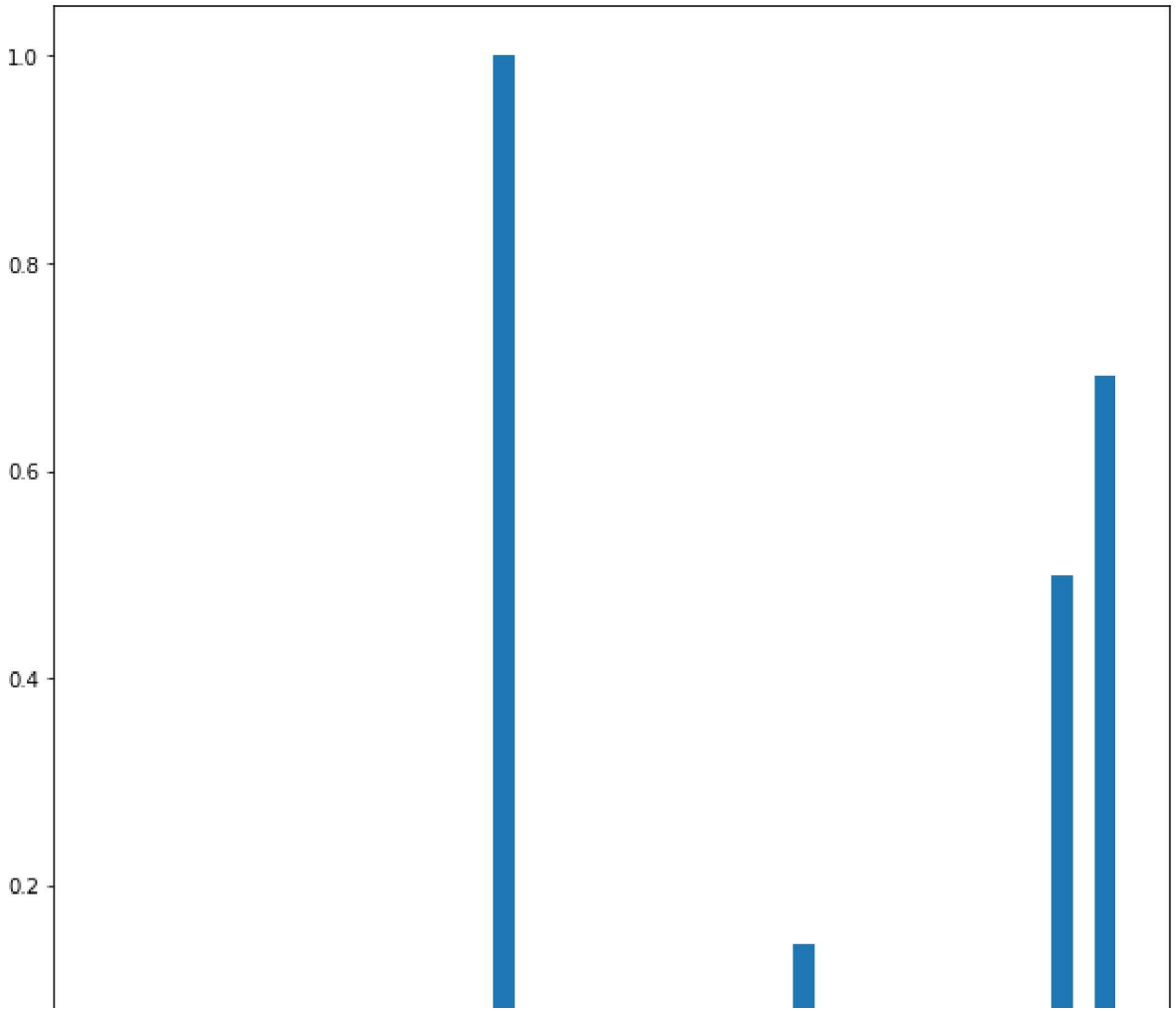
(iii) Categorical - Categorical Analysis

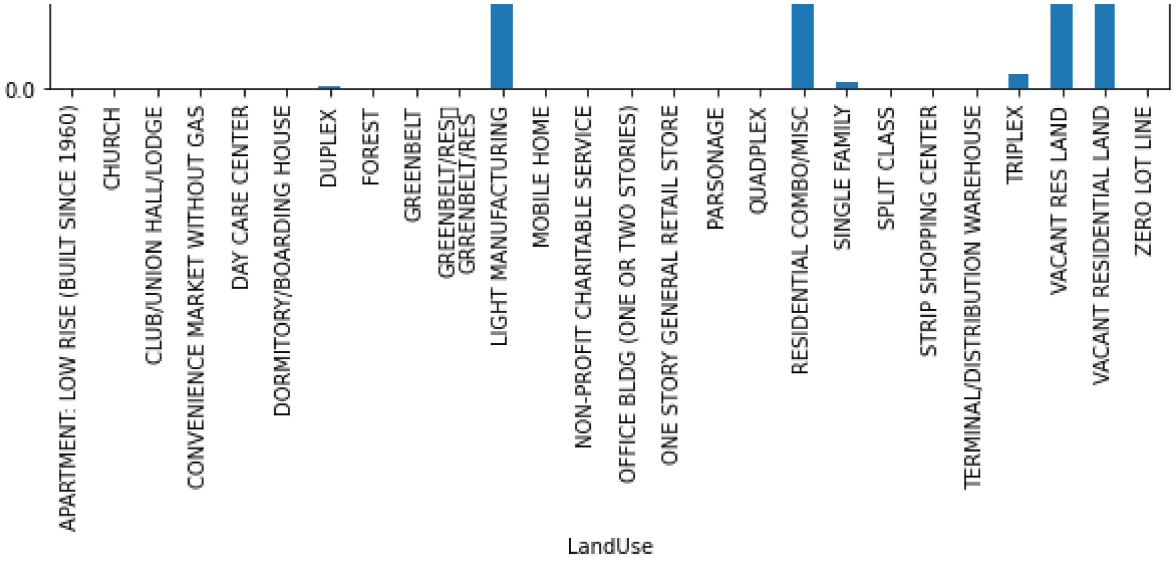
In [42]:
```
df['response_rate'] = np.where(df['SoldAsVacant'] == 'Yes', 1, 0)
df['response_rate']
```

Out[42]:
```
0        0
1        0
2        0
3        0
4        0
        ..
54106    0
54107    0
54108    0
54109    0
54110    0
Name: response_rate, Length: 23806, dtype: int32
```

In [43]:
```
# Plotting the bar chart of LandUse with average value of response_rate
plt.figure(figsize = (10,10))
df.groupby('LandUse')['response_rate'].mean().plot.bar()
plt.show()
```

```
C:\Users\Hussain\anaconda3\lib\site-packages\matplotlib\backends\backend_agg.py:238: RuntimeWarning: Glyph 13 missing f
rom current font.
  font.set_text(s, 0.0, flags=flags)
C:\Users\Hussain\anaconda3\lib\site-packages\matplotlib\backends\backend_agg.py:201: RuntimeWarning: Glyph 13 missing f
rom current font.
  font.set_text(s, 0, flags=flags)
```




0.0
APARTMENT: LOW RISE (BUILT SINCE 1960)
CHURCH
CLUB/UNION HALL/LODGE
CONVENIENCE MARKET WITHOUT GAS
DAY CARE CENTER
DORMITORY/BOARDING HOUSE
DUPLEX
FOREST
GREENBELT
GREENBELT/RES]
GRRENBELT/RES
LIGHT MANUFACTURING
MOBILE HOME
NON-PROFIT CHARITABLE SERVICE
OFFICE BLDG (ONE OR TWO STORIES)
ONE STORY GENERAL RETAIL STORE
PARSONAGE
QUADPLEX
RESIDENTIAL COMBO/MISC
SINGLE FAMILY
SPLIT CLASS
STRIP SHOPPING CENTER
TERMINAL/DISTRIBUTION WAREHOUSE
TRIPLEX
VACANT RES LAND
VACANT RESIDENTIAL LAND
ZERO LOT LINE
LandUse

In [ ]: