In [2]:
```
import pandas as pd
import matplotlib.pyplot as plt
data = pd.read_csv("Real_Estate_Sales_2001-2021_GL.csv")
```

```
/tmp/ipykernel_8777/4107135991.py:3: DtypeWarning: Columns (8,9,10,11,12) have mixed
types. Specify dtype option on import or set low_memory=False.
  data = pd.read_csv("Real_Estate_Sales_2001-2021_GL.csv")
```

The dataset that I will be doing analyzing and retrieving data from will be real estate sales from 2001 to 2021. These are the listed sales that took place in September 30 to October 1 of every year from 2001-2021. The dataset includes useful information from which I will be querying from such as the listed year, data recorded for sale, the town it was listed in, the address, the assessed value, and the actual amount it sold for.

Questions to answer using this dataset.

1. List all rows that were sold for more than twice the assessed value, (sales ratio would be less than .5)
2. List the top 10 rows that have the highest rates of single family assessed value
3. List the top 10 rows that have the highest rates of single family actual sale value
4. List the top 10 addresses that have the highest differences among the sale amount and assessed value
5. List the top 10 addresses that have the lowest differences among the sale amount and assessed value
6. List the years and the amount of houses listed that year

The following code will name all of the columns of our dataset.

In [3]:
```
data.columns
```

Out[3]:
```
Index(['Serial Number', 'List Year', 'Date Recorded', 'Town', 'Address',
       'Assessed Value', 'Sale Amount', 'Sales Ratio', 'Property Type',
       'Residential Type', 'Non Use Code', 'Assessor Remarks', 'OPM remarks',
       'Location'],
      dtype='object')
```

I will now drop the non use code, assessor remarks, opm remarks, and location columns from the dataframe since I will not be using any information from them and it will clear up a lot of NaN values.

In [4]:
```
data = data.drop(['Non Use Code','Assessor Remarks','OPM remarks','Location','Prope
```

The following code will show the first 5 of the dataframe.

In [5]:
```
data.head(5)
```

Out[5]:

	Serial Number	List Year	Date Recorded	Town	Address	Assessed Value	Sale Amount	Sales Ratio	Resider T
0	2020348	2020	09/13/2021	Ansonia	230 WAKELEE AVE	150500.0	325000.0	0.4630	
1	20002	2020	10/02/2020	Ashford	390 TURNPIKE RD	253000.0	430000.0	0.5883	Si Fa
2	210317	2021	07/05/2022	Avon	53 COTSWOLD WAY	329730.0	805000.0	0.4096	Si Fa
3	200212	2020	03/09/2021	Avon	5 CHESTNUT DRIVE	130400.0	179900.0	0.7248	Co
4	200243	2020	04/13/2021	Avon	111 NORTHINGTON DRIVE	619290.0	890000.0	0.6958	Si Fa

The following code will show the last 5 of the dataframe.

In [6]:
```
data.tail(5)
```

Out[6]:

	Serial Number	List Year	Date Recorded	Town	Address	Assessed Value	Sale Amount	Sales Ratio	Re
621949	100308	2010	03/01/2011	Danbury	12-16 SCUPPO RD #K-78	129000.0	120000.0	1.075000	
621950	90282	2009	09/28/2010	Orange	254 KAREN DR	276600.0	370000.0	0.747568	
621951	90087	2009	09/10/2010	Preston	127 RIVER RD	157800.0	150000.0	1.052000	
621952	10113	2010	03/15/2011	Madison	33 JEFFERSON PARK ROAD	281300.0	275000.0	1.022909	
621953	10111	2010	12/28/2010	Hamden	115 ELMER AVE	156380.0	23.0	NaN	

The following code will check the number of rows and columns respectively.

In [7]:
```
data.shape
```

Out[7]:
```
(621954, 9)
```

I am going to check for null values in the following code. I will then drop all of the rows with naN values.

In [8]:
```
data.isnull().sum()
```

Out[8]:
```
Serial Number            0
List Year                0
Date Recorded            2
Town                     0
Address                 48
Assessed Value           0
Sale Amount              0
Sales Ratio              1
Residential Type    357467
dtype: int64
```

In the following code, I drop all of the null values and it will be inplace which means that the original dataframe will be updated.

In [9]:
```
data.dropna(inplace = True)
data.isnull().sum()
```

Out[9]:
```
Serial Number       0
List Year           0
Date Recorded       0
Town                0
Address             0
Assessed Value      0
Sale Amount         0
Sales Ratio         0
Residential Type    0
dtype: int64
```

I am going to check if there are any duplicated rows in the dataframe and there are not in this case.

In [10]:
```
data.duplicated().sum()
```

Out[10]:
```
0
```

In the following code, I am sorting the values according to the list year from 2006 to 2021.

In [11]:
```
data.sort_values('List Year')
```

Out[11]:

	Serial Number	List Year	Date Recorded	Town	Address	Assessed Value	Sale Amount	Sales Ratio
476158	61828	2006	06/14/2007	Bridgeport	89 ELLSWORTH ST	123649.0	300000.0	0.412163
460728	60201	2006	05/16/2007	Derby	126-128 DERBY AVE	220850.0	65000.0	3.397692
460725	60051	2006	10/31/2006	Glastonbury	222 WILLIAMS ST E UT 116	40300.0	91500.0	0.440437
460723	60358	2006	06/15/2007	Cheshire	50 COPPER BEECH DR	381320.0	750000.0	0.508427
460720	60310	2006	04/03/2007	New Milford	17 CORNWALL DRIVE	244930.0	367000.0	0.667384
...	...	...	...	...	...	...	...	...
89762	21038	2021	10/26/2021	Bethel	14 1/2 TOPSTONE DRIVE	92190.0	210000.0	0.439000
89763	210614	2021	04/28/2022	Fairfield	237 TAUNTON ROAD	414260.0	875000.0	0.473400
89764	210031	2021	10/18/2021	Glastonbury	90 KONGSCUT VALLEY TRL	462200.0	744900.0	0.620400
89753	210272	2021	12/06/2021	Bristol	102 HOLLEY RD	108360.0	195000.0	0.555600
95754	2100830	2021	06/03/2022	Stratford	200 POST OAK ROAD	182840.0	480224.0	0.380700

264486 rows × 9 columns

In the following code, I will set the index for every row as its serial number since that is a unique value that was already assigned to it.

In [12]:
```
data.set_index('Serial Number',inplace = True)
data
```

Out[12]:

Serial Number	List Year	Date Recorded	Town	Address	Assessed Value	Sale Amount	Sales Ratio	Resi
20002	2020	10/02/2020	Ashford	390 TURNPIKE RD	253000.0	430000.0	0.588300	
210317	2021	07/05/2022	Avon	53 COTSWOLD WAY	329730.0	805000.0	0.409600	
200212	2020	03/09/2021	Avon	5 CHESTNUT DRIVE	130400.0	179900.0	0.724800	
200243	2020	04/13/2021	Avon	111 NORTHINGTON DRIVE	619290.0	890000.0	0.695800	
200377	2020	07/02/2021	Avon	70 FAR HILLS DRIVE	862330.0	1447500.0	0.595700	
...	...	...	...	...	...	...	...	
100104	2010	04/01/2011	Watertown	22 PAXTON ST	162700.0	155000.0	1.049677	
100308	2010	03/01/2011	Danbury	12-16 SCUPPO RD #K-78	129000.0	120000.0	1.075000	
90282	2009	09/28/2010	Orange	254 KAREN DR	276600.0	370000.0	0.747568	
90087	2009	09/10/2010	Preston	127 RIVER RD	157800.0	150000.0	1.052000	
10113	2010	03/15/2011	Madison	33 JEFFERSON PARK ROAD	281300.0	275000.0	1.022909	

264486 rows × 8 columns

1. List all rows that were sold for more than twice the assessed value(sales ratio would be less than .5)

The following code checks for sales ratio that is less than 0.5 and then displays a table that has the appropriate columns along with every row that has sales ratio less than 0.5.

In [14]:
```
data[data['Sales Ratio']<.5][['List Year','Address','Town','Sales Ratio']]
```

Out[14]:

Serial Number	List Year	Address	Town	Sales Ratio
210317	2021	53 COTSWOLD WAY	Avon	0.409600
210045	2021	89 LONG MEADOW RD	Bethlehem	0.409100
200086	2020	39 WOODLAND RD	Bethlehem	0.479800
210035	2021	25 QUARRY DOCK RD	Branford	0.464500
211762	2021	38 RIVERVIEW DR	Bridgeport	0.352100
...	...	...	...	...
90792	2009	175 VINE ST	Hartford	0.331169
90076	2009	33-35 LISBON ST	Hartford	0.322667
90588	2009	133 GIRARD AV	Hartford	0.209250
90042	2009	76 VILLAGE ST	Vernon	0.322192
90101	2009	35 YALESVILLE SQUARE	Wallingford	0.435152

68459 rows × 4 columns

2. List the top 10 rows that have the highest rates of single family assessed value

The following code finds only single family in residential types and then displays the appropriate columns which are sorted by the descending order of assessed value. The first 10 are shown since they are the top 10 rows.

In [13]:
```
data[data['Residential Type']=='Single Family'][['List Year','Address','Town','Asse
```

Out[13]:

Serial Number	List Year	Address	Town	Assessed Value	Residential Type
10272	2010	642 NEWHALL ST	Hamden	110670208.0	Single Family
21059	2021	GREAT ISLAND ROAD LOT 22	Darien	35030100.0	Single Family
21058	2021	GREAT ISLAND ROAD LOT 22	Darien	35030100.0	Single Family
20115	2020	22 & 27 CEDAR LANE	Sherman	22235900.0	Single Family
60842	2006	110 FIELD POINT CIR	Greenwich	21119910.0	Single Family
200505	2020	30 JOHN STREET	Greenwich	19964630.0	Single Family
60946	2006	STONYBROOK RD	Stratford	19110000.0	Single Family
90582	2009	124 OLD MILL ROAD, GRW,CT 0683	Greenwich	17003280.0	Single Family
21079	2021	55 OLD QUARRY RD	Ridgefield	16800000.0	Single Family
70084	2007	109 BYRAM SHORE RD	Greenwich	16100000.0	Single Family

3. List the top 10 rows that have the highest rates of single family actual sale value

The following code finds only single family in residential types and then displays the appropriate columns which are sorted by the descending order of sale amount. The first 10 are shown since they are the top 10 rows.

In [14]:
```
data[data['Residential Type']=='Single Family'][['List Year','Address','Town','Sale
```

Out[14]:

Serial Number	List Year	Address	Town	Sale Amount	Residential Type
20200078	2020	224 RIVER ROAD	Willington	318790019.0	Single Family
200505	2020	30 JOHN STREET	Greenwich	45000000.0	Single Family
21079	2021	55 OLD QUARRY RD	Ridgefield	34420750.0	Single Family
90536	2009	30 JOHN STREET,GREENWICH,CT 06	Greenwich	31375000.0	Single Family
60596	2006	897 EAST ST	Middletown	30830293.0	Single Family
70234	2007	30 JOHN ST	Greenwich	30000000.0	Single Family
70252	2007	367 LUKES WOOD RD	New Canaan	30000000.0	Single Family
70084	2007	109 BYRAM SHORE RD	Greenwich	28423600.0	Single Family
60842	2006	110 FIELD POINT CIR	Greenwich	25750000.0	Single Family
21125	2021	746 DANBURY RD	Ridgefield	24345000.0	Single Family

4. List the top 10 addresses that have the highest differences among the sale amount and assessed value

The following code adds a new column 'Sales Difference' by finding the difference between 'Sales Amount' and 'Assessed Value'

In [15]:
```
data['Sales Difference'] = data['Sale Amount'] - data['Assessed Value']
data
```

Out[15]:

Serial Number	List Year	Date Recorded	Town	Address	Assessed Value	Sale Amount	Sales Ratio	Resi
20002	2020	10/02/2020	Ashford	390 TURNPIKE RD	253000.0	430000.0	0.588300	
210317	2021	07/05/2022	Avon	53 COTSWOLD WAY	329730.0	805000.0	0.409600	
200212	2020	03/09/2021	Avon	5 CHESTNUT DRIVE	130400.0	179900.0	0.724800	
200243	2020	04/13/2021	Avon	111 NORTHINGTON DRIVE	619290.0	890000.0	0.695800	
200377	2020	07/02/2021	Avon	70 FAR HILLS DRIVE	862330.0	1447500.0	0.595700	
...	...	...	...	...	...	...	...	
100104	2010	04/01/2011	Watertown	22 PAXTON ST	162700.0	155000.0	1.049677	
100308	2010	03/01/2011	Danbury	12-16 SCUPPO RD #K-78	129000.0	120000.0	1.075000	
90282	2009	09/28/2010	Orange	254 KAREN DR	276600.0	370000.0	0.747568	
90087	2009	09/10/2010	Preston	127 RIVER RD	157800.0	150000.0	1.052000	
10113	2010	03/15/2011	Madison	33 JEFFERSON PARK ROAD	281300.0	275000.0	1.022909	

264486 rows × 9 columns

I will now make a new variable which will have the sales differences sorted in descending order and only the first 10 rows.

In [16]:
```
top_10_address_highsalediff = data.sort_values('Sales Difference', ascending = Fals
top_10_address_highsalediff
```

Out[16]:

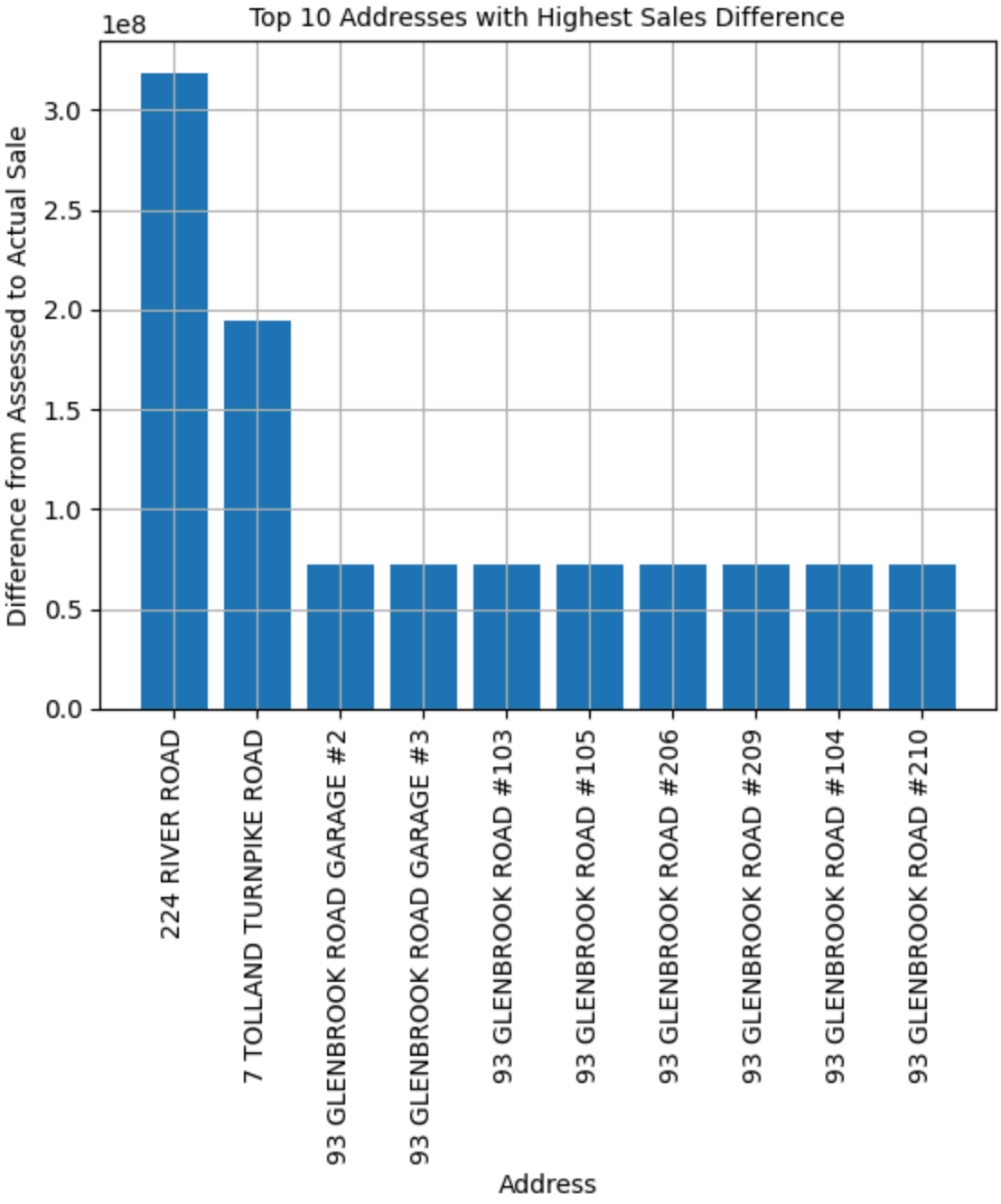
Serial Number	List Year	Date Recorded	Town	Address	Assessed Value	Sale Amount	Sales Ratio	Res
20200078	2020	07/06/2021	Willington	224 RIVER ROAD	223070.0	318790019.0	0.000700	
20200102	2020	09/28/2021	Willington	7 TOLLAND TURNPIKE ROAD	112040.0	194149073.0	0.000577	
2000798	2020	01/15/2021	Stamford	93 GLENBROOK ROAD GARAGE #2	6970.0	72000000.0	0.000000	
2000797	2020	01/15/2021	Stamford	93 GLENBROOK ROAD GARAGE #3	9290.0	72000000.0	0.000100	
2000808	2020	01/15/2021	Stamford	93 GLENBROOK ROAD #103	111240.0	72000000.0	0.001500	
2000810	2020	01/15/2021	Stamford	93 GLENBROOK ROAD #105	111750.0	72000000.0	0.001500	
2000818	2020	01/15/2021	Stamford	93 GLENBROOK ROAD #206	111800.0	72000000.0	0.001500	
2000803	2020	01/15/2021	Stamford	93 GLENBROOK ROAD #209	114140.0	72000000.0	0.001500	
2000809	2020	01/15/2021	Stamford	93 GLENBROOK ROAD #104	114470.0	72000000.0	0.001500	
2000804	2020	01/15/2021	Stamford	93 GLENBROOK ROAD #210	114580.0	72000000.0	0.001500	

I will now make a bar chart which will show the top 10 addresses that had the highest sales differences. It will along with it show the difference on the y-axis and the address on the x-axis.

In [17]:
```
topsalediff = top_10_address_highsalediff['Sales Difference']
addresses = list(top_10_address_highsalediff['Address'])
fig,ax = plt.subplots(nrows=1,ncols=1)
ax.bar(addresses,topsalediff)
```

```
plt.xlabel('Address',fontsize = 10)
plt.ylabel('Difference from Assessed to Actual Sale',fontsize = 10)
plt.xticks(rotation = 'vertical',fontsize = 10)
plt.title('Top 10 Addresses with Highest Sales Difference',fontsize = 10)

ax.grid()
plt.show()
```



5. List the top 10 addresses that have the lowest differences among the sale amount and assessed value

I will now make a new variable which will have the sales differences sorted in ascending order and only the first 10 rows.

In [18]:
```
top_10_address_lowsalediff = data[data['Sales Difference']>0].sort_values('Sales Di
top_10_address_lowsalediff
```

Out[18]:

Serial Number	List Year	Date Recorded	Town	Address	Assessed Value	Sale Amount	Sales Ratio	Reside
900034	2009	06/15/2010	Lisbon	214 RIVER RD	139890.0	139900.0	0.999929	Si Fa
60191	2006	11/27/2006	Bristol	14 FAIRFIELD ST	84980.0	85000.0	0.999765	Si Fa
8408	2008	05/14/2009	Meriden	2-11 NORTHWOOD PL	38980.0	39000.0	0.999487	Co
900542	2009	05/27/2010	Bristol	87 BARBARA RD	126980.0	127000.0	0.999843	Si Fa
100318	2010	05/27/2011	Enfield	37 BRIDGE LA	151970.0	152000.0	0.999803	Si Fa
90231	2009	11/23/2009	Bridgeport	134 SOUNDVIEW AVENUE	124970.0	125000.0	0.999760	Si Fa
210624	2021	04/12/2022	Hamden	2696 WHITNEY AVE	179970.0	180000.0	0.999800	Two Fa
90487	2009	01/25/2010	Bridgeport	223-225 HAWLEY AVENUE	139970.0	140000.0	0.999786	Two Fa
210625	2021	04/12/2022	Hamden	2696 WHITNEY AVE	179970.0	180000.0	0.999800	Two Fa
60292	2006	01/29/2007	West Haven	51 OWNLY	134960.0	135000.0	0.999704	Si Fa

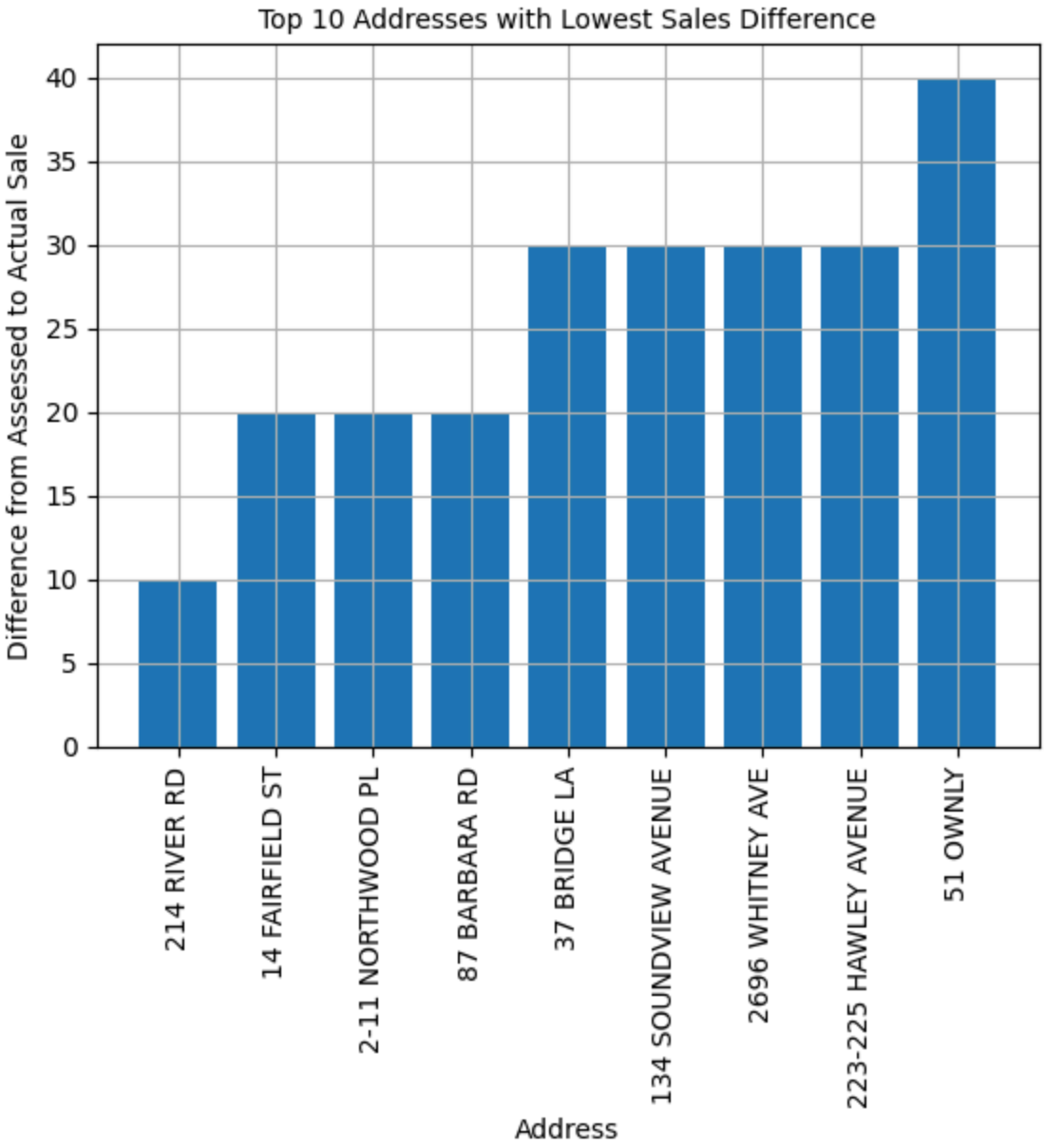
I will now make a bar chart which will show the top 10 addresses that had the lowest sales differences. It will along with it show the difference on the y-axis and the address on the x-axis.

In [19]:
```
topsalediff = top_10_address_lowsalediff['Sales Difference']
addresses = list(top_10_address_lowsalediff['Address'])
fig,ax = plt.subplots(nrows=1,ncols=1)
ax.bar(addresses,topsalediff)

plt.xlabel('Address',fontsize = 10)
plt.ylabel('Difference from Assessed to Actual Sale',fontsize = 10)
plt.xticks(rotation = 'vertical',fontsize = 10)
```

```
plt.title('Top 10 Addresses with Lowest Sales Difference',fontsize = 10)

ax.grid()
plt.show()
```



6. List the years and the amount of houses listed that year

In the following code, I make a new variable in which I find the counted values of the list year and sort them by descending order. I then create a new variable to create a dataframe for the past variable and rename the column to counted. In the end, this creates a new dataframe which has the list year and the amount of time it shows up.

In [20]:
```
top_list_year = data['List Year'].value_counts().sort_values(ascending=False)
top_list_year1 = pd.DataFrame(top_list_year,columns = ['List Year'])
top_list_year1=top_list_year1.rename(columns={'List Year':'Counted'})
top_list_year1
```

Out[20]:

	Counted
2020	60728
2021	51371
2006	43290
2009	38769
2007	31827
2008	30104
2010	8135
2019	57
2018	41
2017	40
2016	31
2015	24
2012	20
2011	17
2014	17
2013	15

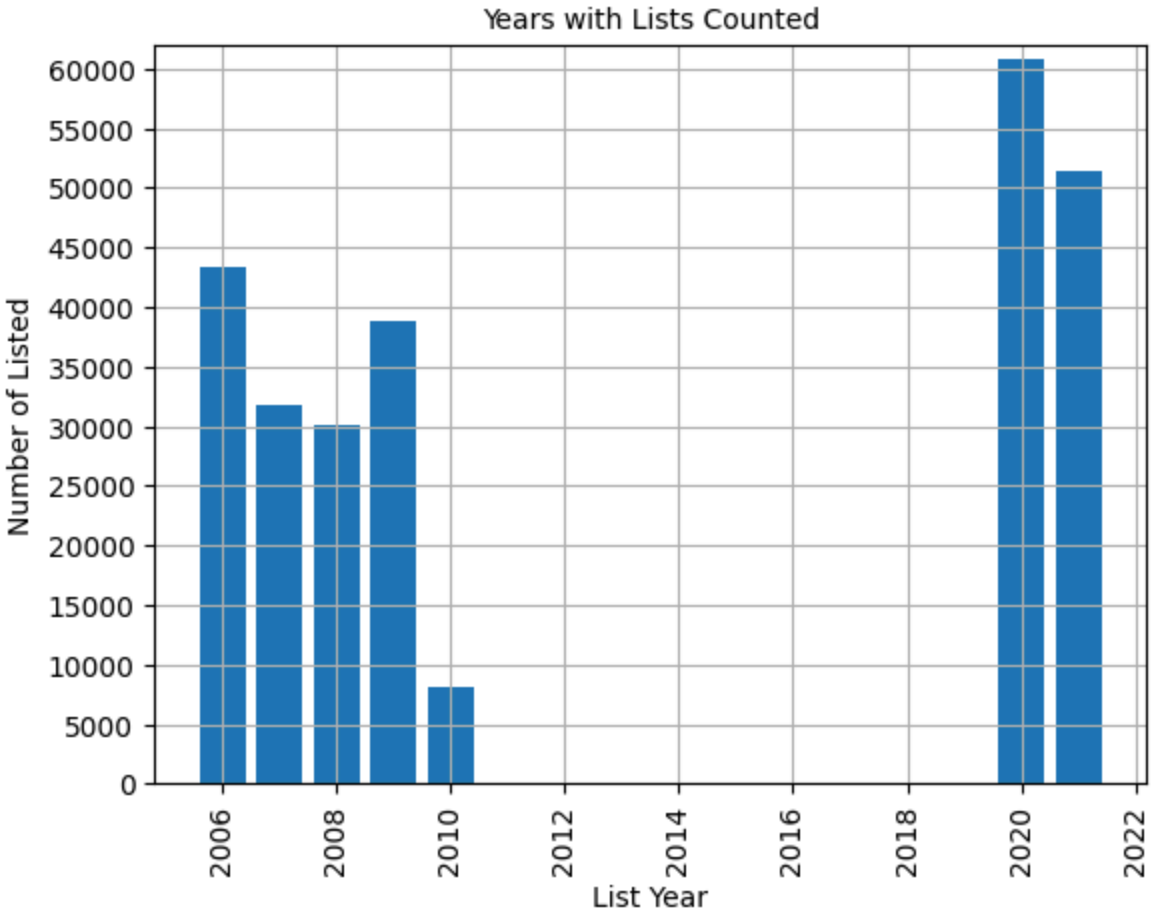
I will now make a bar chart which will show list years on the x-axis and the amount of listed on the y-axis. This will show all of the years and the list years to give an overview of the whole dataset.

In [21]:
```
import matplotlib.ticker as ticker

toplisted = top_list_year1['Counted']
listyear = list(top_list_year1.index)
fig,ax = plt.subplots(nrows=1,ncols=1)
ax.bar(listyear,toplisted)

plt.xlabel('List Year',fontsize = 10)
plt.ylabel('Number of Listed',fontsize = 10)
plt.xticks(rotation = 'vertical',fontsize = 10)
plt.title('Years with Lists Counted',fontsize = 10)
ax.yaxis.set_major_locator(ticker.MultipleLocator(5000))
plt.ylim(0, 62000)


ax.grid()
plt.show()
```


Years with Lists Counted
Number of Listed
List Year
0
5000
10000
15000
20000
25000
30000
35000
40000
45000
50000
55000
60000
2006
2008
2010
2012
2014
2016
2018
2020
2022