Zillow Home Value Index Data

Obtaining the Data

In [1]:
```
#import libraries & modules
import pandas as pd
import numpy as np
import warnings
warnings.filterwarnings('ignore')


pd.set_option('display.max_colwidth', None)
pd.set_option('display.max_rows', None)
warnings.filterwarnings('ignore')
warnings.simplefilter('ignore')
%matplotlib inline
pd.set_option('display.max_columns', None)
```

Zillow Data

Data from Zillow's Home Value Index (ZHVI) is being used. Zillow Home Value Index is a measure of a homes typical value and market changes across a given region and type of housing. By measuring monthly changes in property across different housing types and geographies Zillow is able to capture how the market price changes and not just the changes in the kinds of markets or property types that sell on a month to month basis. The ZHVI dollar amount is representative of the "typical home value for a region" and not the "median home value".

We will be looking at three different datasets from Zillow:

- **Bottom Tier Homes**: typical value for homes within the 5th to 35th percentile range for a given region.
- **Middle Tier Homes**: typical value for homes within the 35th to 65th percentile range for a given region.
- **Top Tier Homes**: typical value for homes within the 65th to 95th percentile range for a given region.

Column Name	Description
SizeRank	Numerical rank of size of cities, ranked 0 through 30,132
City (RegionName)	Name of the city
State	State in which the city is located
1/31/2000 through 4/30/2023	refers to the typical home value for the city for January 2000 through April 2023.
before	home value six months before the hurricane
after	home value six months after the hurricane

Column Name	Description
percent	percent change in home value from six months before hurricane to six months after hurricane
increase	If the percentage change in the value of a home was in the 75th percentile six months after a hurricane it was considered to be in the category 1, if not then 0.

Citations

Olsen S. Zillow Home Value Index Methodology, 2023 Revision: What's Changed? Zillow. Published February 11, 2023. https://www.zillow.com/research/methodology-neural-zhvi-32128/ (https://www.zillow.com/research/methodology-neural-zhvi-32128/)
ZHVI User Guide. Zillow Research. https://www.zillow.com/research/zhvi-user-guide/ (https://www.zillow.com/research/zhvi-user-guide/)

Obtaining the Data

In [2]:
```
#opening the datasets
#bottom tier
bottom = pd.read_csv(r'data\zillow_bottom_tier.csv')
#top tier
top = pd.read_csv(r'data\zillow_top_tier.csv')
#middle tier
middle = pd.read_csv(r'data\zillow_middle_tier.csv')
```

In [3]:
```
#taking a look
bottom.head()
```

Out[3]:

	SizeRank	RegionName	State	1/31/2000	2/29/2000	3/31/2000	4/30/2000	
0	0	New York	NY	73546.94209	73946.76925	74343.34862	75095.91259	
1	1	Los Angeles	CA	112089.32630	112263.53370	112762.71010	113750.27700	1
2	2	Houston	TX	48111.19222	48108.47093	48052.63010	48041.43903	
3	3	Chicago	IL	29056.60823	29052.14635	29125.11876	29292.67884	
4	4	San Antonio	TX	45948.02234	45991.78734	46030.67980	46100.98692	

In [4]:
```
#taking a look
middle.head()
```

Out[4]:

	SizeRank	RegionName	State	1/31/2000	2/29/2000	3/31/2000	4/30/2000	
0	0	New York	NY	131748.37630	132455.14810	133172.63100	134560.11470	
1	1	Los Angeles	CA	215492.28720	215796.93350	216730.57720	218588.28870	2
2	2	Houston	TX	98322.10168	98295.78363	98158.99868	98115.03948	
3	3	Chicago	IL	121417.32980	121451.25850	121760.20900	122543.59260	
4	4	San Antonio	TX	97194.61919	97285.78750	97355.59314	97480.82108	

In [5]:
```
#taking a look
top.head()
```

Out[5]:

	SizeRank	RegionName	State	1/31/2000	2/29/2000	3/31/2000	4/30/2000	5/3
0	0	New York	NY	363570.5618	365652.0674	367773.4539	371912.8236	3761
1	1	Los Angeles	CA	430196.1809	430835.8446	432742.7602	436562.7946	4412
2	2	Houston	TX	216952.5626	216916.4350	216579.8438	216509.7298	2165
3	3	Chicago	IL	310440.8686	310673.7785	311639.3970	313902.1714	3168
4	4	San Antonio	TX	174586.0000	174788.3493	174971.2252	175326.2611	1745

Data Scrubbing

Currently these datasets contain home values for all 50 states, for the purpose of this project we are only interested in data from Florida. Let's use a function to limit the dataset to Florida and then removie the 'State' column, which is just a unique identifier. We will also rename RegionName to City for clarity.

In [6]:
```
#function to limit dataset to just Florida and drop 'State' column
def clean(df):
    #limiting the dataset to just the state of Florida
    df = df[df["State"] == 'FL']

    #State is not needed because we are just looking at Florida
    df = df.drop(['State'], axis = 1)

    #renaming RegionName to City
    df.rename(columns={"RegionName": "City"}, inplace=True)
    return df

```

In [7]:
```
#applying functions to each dataset
middle = clean(middle)
top = clean(top)
bottom = clean(bottom)
```

In [8]:
```
#checking it out
middle.head()
```

Out[8]:

	SizeRank	City	1/31/2000	2/29/2000	3/31/2000	4/30/2000	5/31
12	12	Jacksonville	88967.23677	89139.14852	89327.04785	89739.28754	90194.5
16	16	Orlando	110587.77770	110861.49990	111260.38600	111975.56300	112743.2
20	20	Miami	118032.42490	118517.95820	118949.87390	119812.76850	120585.
50	50	Tampa	90917.05524	91209.23749	91533.74884	92260.11010	93010.7
72	73	Naples	167101.42240	167175.11190	167806.44150	168945.21810	170492.3

In [9]:
```
#taking a look
top.head()
```

Out[9]:

	SizeRank	City	1/31/2000	2/29/2000	3/31/2000	4/30/2000	5/31/2000
12	12	Jacksonville	162694.8807	162998.2202	163385.0538	164173.9853	165065.0763
16	16	Orlando	198438.0210	198863.8577	199547.7750	200752.6874	202079.8852
20	20	Miami	268968.0970	270033.1253	270924.4218	272821.5998	274535.7826
50	50	Tampa	213722.8511	214604.6181	215440.3645	217163.2621	218819.5140
72	73	Naples	378274.3016	378470.3032	380066.5185	382966.1783	386824.4212

In [10]:
```
#taking a look
bottom.head()
```

Out[10]:

	SizeRank	City	1/31/2000	2/29/2000	3/31/2000	4/30/2000	5/31/2000
12	12	Jacksonville	34345.37809	34401.37835	34463.49336	34614.56979	34789.20541
16	16	Orlando	54399.11437	54514.30500	54714.68168	55081.25437	55486.76470
20	20	Miami	52834.56992	53017.65767	53177.81077	53521.19432	53839.20505
50	50	Tampa	32442.30379	32592.93576	32737.85510	33051.73181	33342.68522
72	73	Naples	83105.88309	83134.46159	83478.57626	84091.82543	84915.45576

Data Exploration

The datasets are missing some values, however, it is less than 10% of the data we will be using for the purpose of this project. We will drop any missing values. The Bottom Tier Housing Set has 573 entries, three different data types, and housing values starting at a mean value of 61,812USD and ending with home values of about 302,307USD. The Middle Tier Housing set has 582 entries, three different data types, and mean home value starting at 127,165USD and ending at 478,743USD. The Top Tier Housing set has 596 entries, three different data types, and mean home value starting at 261,626USD and ending at 814,351USD.

Exploring Bottom Tier Housing Values

In [11]:
```
#getting descriptive stats
bottom.describe()
```

Out[11]:

	SizeRank	1/31/2000	2/29/2000	3/31/2000	4/30/2000	5/31
count	573.000000	467.000000	469.000000	469.000000	469.000000	469.0(
mean	7580.900524	61812.228839	61790.052626	61980.488757	62393.028522	62831.0
std	7365.234662	51594.521249	51869.512645	52065.858044	52549.738161	52958.4(
min	12.000000	3009.483002	3014.873264	3021.220070	3036.350104	3057.0(
25%	1741.000000	33338.807730	33239.761190	33402.911120	33524.466130	33807.1
50%	5170.000000	53246.145590	53017.657670	53177.810770	53521.194320	53798.1
75%	10986.000000	75217.550390	75219.130980	75424.110130	75850.551130	76271.5(
max	28699.000000	734329.642300	739839.486700	743865.469600	751602.320300	755145.6

In [12]:
```
#looking at data types
bottom.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 573 entries, 12 to 21441
Columns: 282 entries, SizeRank to 4/30/2023
dtypes: float64(280), int64(1), object(1)
memory usage: 1.2+ MB
```

In [13]:
```
#checking for missing values
bottom.isna().sum()
```

```
11/30/2021      6
12/31/2021      6
1/31/2022       6
2/28/2022       3
3/31/2022       3
4/30/2022       3
5/31/2022       3
6/30/2022       3
7/31/2022       3
8/31/2022       3
9/30/2022       3
10/31/2022      3
11/30/2022      3
12/31/2022      3
1/31/2023       4
2/28/2023       4
3/31/2023       4
4/30/2023       1
dtype: int64
```

In [14]:
```
#dropping missing values
bottom.dropna(inplace=True)

bottom.isna().sum()
```

```
1/31/2005     0
2/28/2005     0
3/31/2005     0
4/30/2005     0
5/31/2005     0
6/30/2005     0
7/31/2005     0
8/31/2005     0
9/30/2005     0
10/31/2005    0
11/30/2005    0
12/31/2005    0
1/31/2006     0
2/28/2006     0
3/31/2006     0
4/30/2006     0
5/31/2006     0
6/30/2006     0
7/31/2006     0
8/31/2006     0
```

Exploring Housing Values in the 35th to 65th Percentile Range

In [15]:
```
#getting descriptive stats
middle.describe()
```

Out[15]:

	SizeRank	1/31/2000	2/29/2000	3/31/2000	4/30/2000	5/31/2000
count	582.000000	4.740000e+02	4.760000e+02	4.770000e+02	4.770000e+02	4.770000e+02
mean	7703.982818	1.271650e+05	1.273793e+05	1.277044e+05	1.285548e+05	1.294463e+05
std	7404.384329	1.106724e+05	1.112648e+05	1.119233e+05	1.136379e+05	1.151120e+05
min	12.000000	2.617646e+04	2.622625e+04	2.635335e+04	2.652310e+04	2.668611e+04
25%	1788.500000	7.760322e+04	7.768362e+04	7.795715e+04	7.824740e+04	7.880663e+04
50%	5290.500000	1.027824e+05	1.030046e+05	1.032534e+05	1.037273e+05	1.041574e+05
75%	11450.250000	1.420007e+05	1.419802e+05	1.424877e+05	1.430154e+05	1.440219e+05
max	28699.000000	1.531128e+06	1.547405e+06	1.569073e+06	1.608019e+06	1.634701e+06

In [16]:
```
#looking at data types
middle.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 582 entries, 12 to 22142
Columns: 282 entries, SizeRank to 4/30/2023
dtypes: float64(280), int64(1), object(1)
memory usage: 1.3+ MB
```

In [17]:
```
#checking for missing values
middle.isna().sum()
```

Out[17]:
```
SizeRank        0
City            0
1/31/2000     108
2/29/2000     106
3/31/2000     105
4/30/2000     105
5/31/2000     105
6/30/2000     105
7/31/2000     105
8/31/2000     105
9/30/2000     105
10/31/2000    105
11/30/2000    104
12/31/2000    104
1/31/2001     104
2/28/2001     104
3/31/2001     104
4/30/2001     104
5/31/2001     104
```

In [18]:
```
#dropping missing values
middle.dropna(inplace=True)

middle.isna().sum()
```

```
10/31/2001    0
11/30/2001    0
12/31/2001    0
1/31/2002     0
2/28/2002     0
3/31/2002     0
4/30/2002     0
5/31/2002     0
6/30/2002     0
7/31/2002     0
8/31/2002     0
9/30/2002     0
10/31/2002    0
11/30/2002    0
12/31/2002    0
1/31/2003     0
2/28/2003     0
3/31/2003     0
4/30/2003     0
5/31/2003     0
```

Exploring Housing Values in the 65th to 100th Percentile Range

In [19]:
```
#checking descriptive stats
top.describe()
```

Out[19]:

	SizeRank	1/31/2000	2/29/2000	3/31/2000	4/30/2000	5/31/2000
count	586.000000	4.780000e+02	4.800000e+02	4.800000e+02	4.800000e+02	4.800000e+02
mean	7787.126280	2.616267e+05	2.619208e+05	2.627467e+05	2.645054e+05	2.663321e+05
std	7462.052386	3.380758e+05	3.398479e+05	3.426333e+05	3.482321e+05	3.533410e+05
min	12.000000	4.025377e+04	4.030378e+04	4.036054e+04	4.047724e+04	4.060978e+04
25%	1797.500000	1.474623e+05	1.476534e+05	1.479823e+05	1.489066e+05	1.496993e+05
50%	5316.500000	1.899119e+05	1.907133e+05	1.912005e+05	1.918216e+05	1.920236e+05
75%	11717.500000	2.571210e+05	2.552917e+05	2.560419e+05	2.574097e+05	2.591469e+05
max	28699.000000	4.430545e+06	4.478852e+06	4.560440e+06	4.700627e+06	4.810938e+06

In [20]:
```
#checking datatypes
top.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 586 entries, 12 to 22156
Columns: 282 entries, SizeRank to 4/30/2023
dtypes: float64(280), int64(1), object(1)
memory usage: 1.3+ MB
```

In [21]:
```
#checking for missing values
top.isna().sum()
```

```
10/31/2021      2
11/30/2021      2
12/31/2021      2
1/31/2022       2
2/28/2022       2
3/31/2022       2
4/30/2022       2
5/31/2022       2
6/30/2022       2
7/31/2022       2
8/31/2022       2
9/30/2022       2
10/31/2022      2
11/30/2022      2
12/31/2022      2
1/31/2023       2
2/28/2023       2
3/31/2023       2
4/30/2023       0
dtype: int64
```

In [22]:
```
#dropping missing values
top.dropna(inplace=True)

top.isna().sum()
```

```
11/30/2021    0
12/31/2021    0
1/31/2022     0
2/28/2022     0
3/31/2022     0
4/30/2022     0
5/31/2022     0
6/30/2022     0
7/31/2022     0
8/31/2022     0
9/30/2022     0
10/31/2022    0
11/30/2022    0
12/31/2022    0
1/31/2023     0
2/28/2023     0
3/31/2023     0
4/30/2023     0
dtype: int64
```

6 Months Before and After

Since this was a classification problem the target variable has to be in the form of a class. Using home value data from six months before and after each hurricane I engineered a column that contained two categories, increased significantly (coded as 1) and did not increase significantly (coded as 0). If the percentage change in the value of a home was in the 75th percentile six months after a hurricane it was considered to be in the category 1, if not then 0. The dates we will look at six months before and after each hurricane are as follows:

Hurricane	6 Months Before	6 Months After
Charley	2/2004	2/2005
Dennis	1/2005	1/2006
Matthew	4/2016	4/2017
Irma	3/2017	3/2018
Michael	4/2018	4/2019
Ian	3/2022	3/2023

Graphing Home Value

Let's graph home value to visualize how how home prices have changed six months before and six months after each hurricane. As we can see from the graphs below, home value has always increased.

In [23]:
```
#creating dataframes to plot home value change overtime
#we only need date and value columns for this
bottom_graph = bottom.drop(['SizeRank', 'City'], axis = 1)
middle_graph = middle.drop(['SizeRank', 'City'], axis = 1)
top_graph = top.drop(['SizeRank', 'City'], axis = 1)
```

In [24]:
```
#creating function to melt data
#this will put date along the index
def melt_data(df):
    melted = pd.melt(df, var_name='Date')
    melted['Date'] = pd.to_datetime(melted['Date'], infer_datetime_for
    melted = melted.dropna(subset=['value'])
    #grouping by mean of ZHVI
    melted = melted.groupby('Date').mean('value')
    return melted
```

In [25]:
```
#applying the function to our dataframes
bottom_graph = melt_data(bottom_graph)
middle_graph = melt_data(middle_graph)
top_graph = melt_data(top_graph)
```

Graphing Bottom Tier Housing

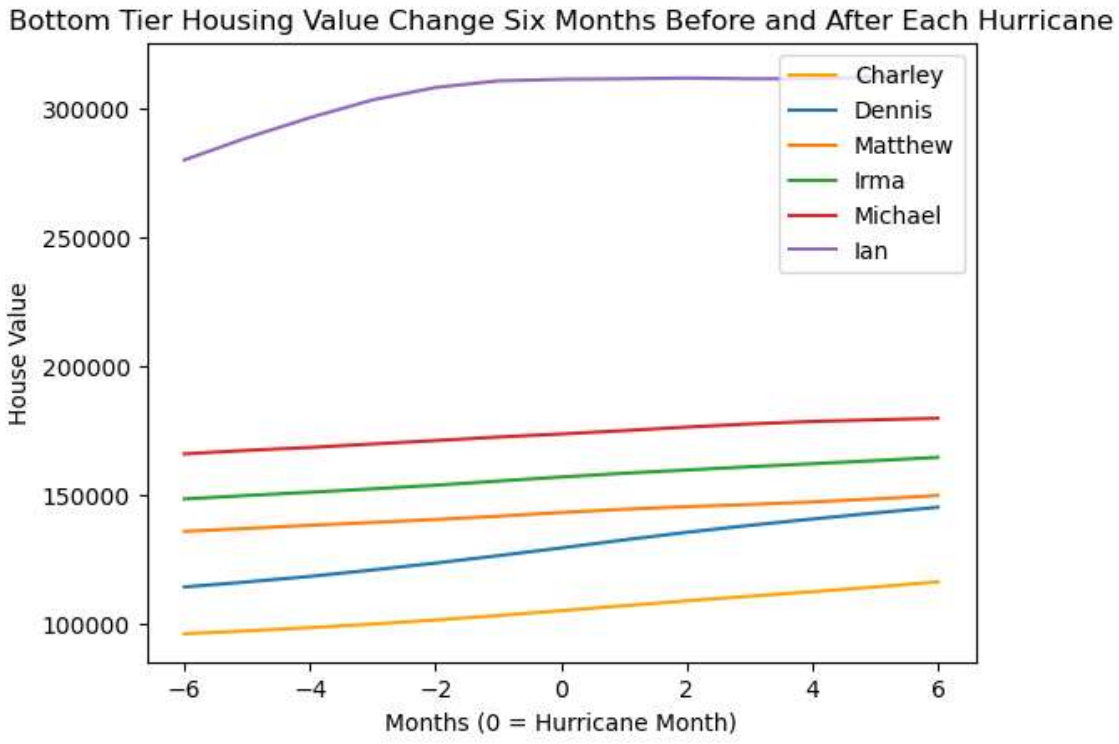
In [26]:
```
#Creating dataframes for each hurricane
#Values are from six months before and after the hurricane
charley_graph = bottom_graph.iloc[49:62]
dennis_graph = bottom_graph.iloc[60:73]
matthew_graph = bottom_graph.iloc[195:208]
irma_graph = bottom_graph.iloc[206:219]
michael_graph = bottom_graph.iloc[219:232]
ian_graph = bottom_graph.iloc[266:279]
```

In [27]:
```
#writing a function to change date to month number
def month(df):
    df['month'] = np.repeat(np.arange(-6, 7), 1)
    df.set_index('month', inplace = True)
    return df
```

In [28]:
```
#applying the function to our dataframes
graph_list = [charley_graph, dennis_graph, matthew_graph, irma_graph,
graph_list = [df.pipe(month) for df in graph_list]
```

In [29]:

```
import matplotlib.pyplot as plt
# Visualizing The Value Change of Homes Six Months Before and After Ea

# to set the plot size
plt.figure()

# using plot method to plot values.
# in plot method we set the label and color of the curve
charley_graph['value'].plot(label='Charley', color='orange')
dennis_graph['value'].plot(label='Dennis')
matthew_graph['value'].plot(label='Matthew')
irma_graph['value'].plot(label='Irma')
michael_graph['value'].plot(label='Michael')
ian_graph['value'].plot(label='Ian')

# adding title to the plot
plt.title('Bottom Tier Housing Value Change Six Months Before and Afte

# adding label to the x-axis
plt.xlabel('Months (0 = Hurricane Month)')

# adding label to the y-axis
plt.ylabel('House Value')

# adding legend to the curve
plt.legend();
```



Graphing Middle Tier Housing

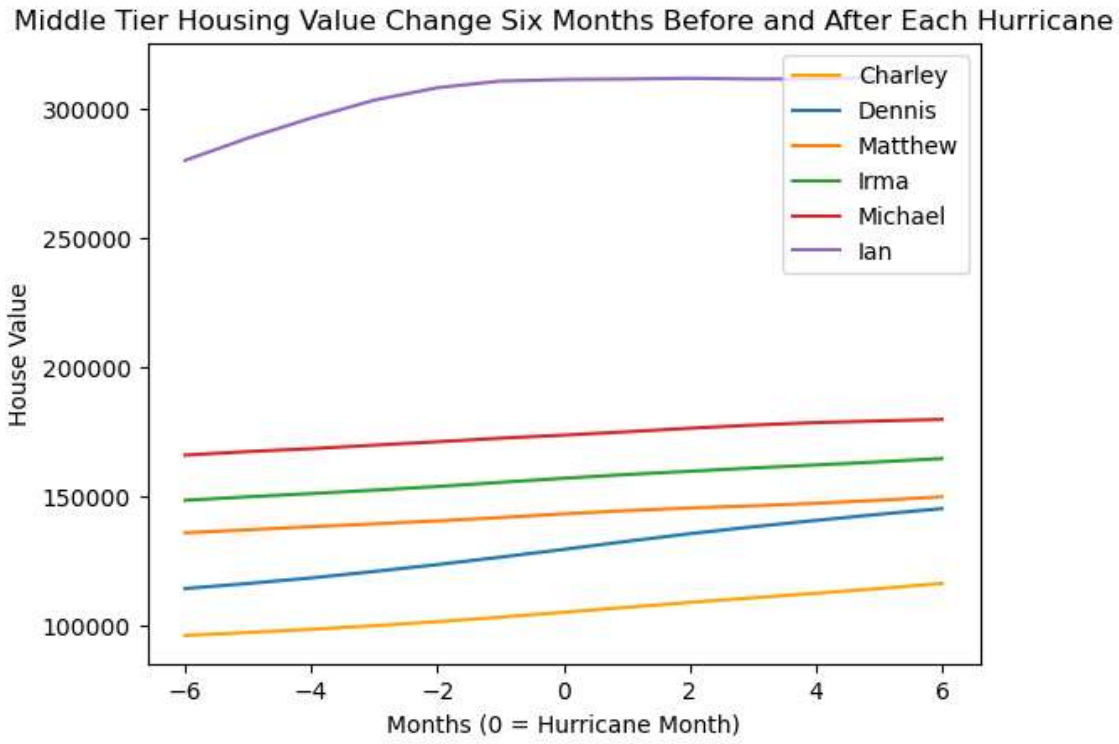
In [30]:
```
#Creating dataframes for each hurricane
#Values are from six months before and after the hurricane
charley_graph = bottom_graph.iloc[49:62]
dennis_graph = bottom_graph.iloc[60:73]
matthew_graph = bottom_graph.iloc[195:208]
irma_graph = bottom_graph.iloc[206:219]
michael_graph = bottom_graph.iloc[219:232]
ian_graph = bottom_graph.iloc[266:279]
```

In [31]:
```
def month(df):
    df['month'] = np.repeat(np.arange(-6, 7), 1)
    df.set_index('month', inplace = True)
    return
```

In [32]:
```
#applying the function to our dataframes
graph_list = [charley_graph, dennis_graph, matthew_graph, irma_graph,
graph_list = [df.pipe(month) for df in graph_list]
```

In [33]:

```
import matplotlib.pyplot as plt
# Visualizing The Value Change of Homes Six Months Before and After Ea

# to set the plot size
plt.figure()

# using plot method to plot values.
# in plot method we set the label and color of the curve.
charley_graph['value'].plot(label='Charley', color='orange')
dennis_graph['value'].plot(label='Dennis')
matthew_graph['value'].plot(label='Matthew')
irma_graph['value'].plot(label='Irma')
michael_graph['value'].plot(label='Michael')
ian_graph['value'].plot(label='Ian')

# adding title to the plot
plt.title('Middle Tier Housing Value Change Six Months Before and Afte

# adding label to the x-axis
plt.xlabel('Months (0 = Hurricane Month)')

# adding label to the y-axis
plt.ylabel('House Value')

# adding legend to the curve
plt.legend();
```



Graphing Top Tier Housing

In [34]:
```
#Creating dataframes for each hurricane
#Values are from six months before and after the hurricane
charley_graph = bottom_graph.iloc[49:62]
dennis_graph = bottom_graph.iloc[60:73]
matthew_graph = bottom_graph.iloc[195:208]
irma_graph = bottom_graph.iloc[206:219]
michael_graph = bottom_graph.iloc[219:232]
ian_graph = bottom_graph.iloc[266:279]
```

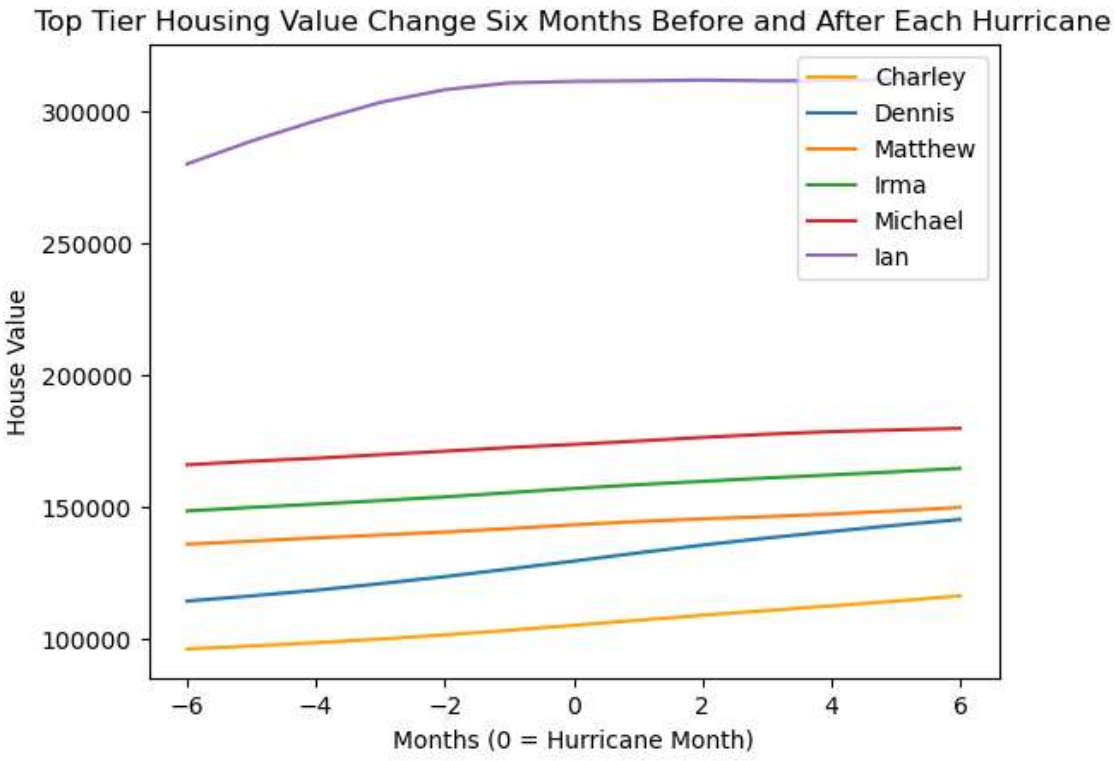
In [35]:
```
def month(df):
    df['month'] = np.repeat(np.arange(-6, 7), 1)
    df.set_index('month', inplace = True)
    return
```

In [36]:
```
#applying the function to our dataframes
graph_list = [charley_graph, dennis_graph, matthew_graph, irma_graph,
graph_list = [df.pipe(month) for df in graph_list]
```

In [37]:
```
graph_list[0]
```

In [38]:

```
import matplotlib.pyplot as plt
# Visualizing The Value Change of Homes Six Months Before and After Ea

# to set the plot size
fig= plt.figure()

# using plot method to plot values.
# in plot method we set the label and color of the curve.
charley_graph['value'].plot(label='Charley', color='orange')
dennis_graph['value'].plot(label='Dennis')
matthew_graph['value'].plot(label='Matthew')
irma_graph['value'].plot(label='Irma')
michael_graph['value'].plot(label='Michael')
ian_graph['value'].plot(label='Ian')

# adding title to the plot
plt.title('Top Tier Housing Value Change Six Months Before and After E

# adding label to the x-axis
plt.xlabel('Months (0 = Hurricane Month)')

# adding label to the y-axis
plt.ylabel('House Value')

# adding legend to the curve
plt.legend();

```



Writing Functions

Let's write a functions to make scrubbing and engineering our three dataframes simpler. This function will:

- rename the date columns six months before and after the hurricane
- give us the percent change in home value six months before and after the hurricane
- create a column telling us if the percent increase was more than 75%
- and convert the dataframe from wide to long format

In [39]:

```
#writing function to rename
#Renaming the columns 6 months before and after each hurricane
#b = before a = after
def renaming(df):


    #Charley 2/29/2004 2/28/2005
    df['before_charley'] = df['2/29/2004']
    df['after_charley'] = df['2/28/2005']

    #Dennis 1/31/2005 1/31/2006
    df['before_dennis'] = df['1/31/2005']
    df['after_dennis'] = df['1/31/2006']

    #Matthew 4/30/2016 4/30/2017
    df['before_matthew'] = df['4/30/2016']
    df['after_matthew'] = df['4/30/2017']

    #Irma 3/31/2017 3/31/2018
    df['before_irma'] = df['3/31/2017']
    df['after_irma'] = df['3/31/2018']

    #Michael 4/30/2018 4/30/2019
    df['before_michael'] = df['4/30/2018']
    df['after_michael'] = df['4/30/2019']

    #Ian 3/30/2022 - 3/30/2023
    df['before_ian'] = df['4/30/2018']
    df['after_ian'] = df['4/30/2019']



    #dropping all unecessary date columns
    df.drop(df.iloc[:, 2:282], inplace=True, axis=1)
    return df
```

In [40]:

```
#applying the function to our dataframes
df_list = [bottom, middle, top]
df_list = [df.pipe(renaming) for df in df_list]
```

In [41]:
```
#writing function to find percent change in home value
#percent = (after - before/before)*100
def percent(df):

    #finding percent change for Hurricane Charley
    df['percent_charley'] = (df['after_charley'] - df['before_charley'

    #finding percent change for Hurricane Dennis
    df['percent_dennis'] = (df['after_dennis'] - df['before_dennis'])/

    #finding percent change for Hurricane Matthew
    df['percent_matthew'] = (df['after_matthew'] - df['before_matthew'

    #finding percent change for Hurricane Irma
    df['percent_irma'] = (df['after_irma'] - df['before_irma'])/df['be

    #finding percent change for Hurricane Michael
    df['percent_michael'] = (df['after_michael'] - df['before_michael'

    #finding percent change for Hurricane Ian
    df['percent_ian'] = (df['after_ian'] - df['before_ian'])/df['befor

    return df
```

In [42]:
```
#applying the function to our dataframes
df_list = [bottom, middle, top]
df_list = [df.pipe(percent) for df in df_list]
```

In [43]:
```
#checking it out the dataframe
bottom.head()
```

Out[43]:

	SizeRank	City	before_charley	after_charley	before_dennis	after_dennis	befor
12	12	Jacksonville	46528.30349	52803.32790	52229.68266	61480.05435	61
16	16	Orlando	75863.27537	88560.22345	87103.38849	114235.08580	102
20	20	Miami	86752.55847	106338.70750	104500.57250	132340.36560	14
50	50	Tampa	51585.41060	61309.89329	60275.09126	77788.70274	79
83	84	Saint Petersburg	47796.13229	57880.05754	56710.46260	72152.72440	68

In [44]:

```
#function to create a boolean column for home value increase
def bool(df):

    #using cutoff to create boolean value
    #cutting off at 75%
    #1 = True (increase of more than 75%)
    #0 = False (no increase of more than 75%)

    #creating bool column for charley
    df['increase_charley'] = np.where(df['percent_charley'] >= (df['pe

    #creating bool column for dennis
    df['increase_dennis'] = np.where(df['percent_dennis'] >= (df['perc

    #creating bool column for matthew
    df['increase_matthew'] = np.where(df['percent_matthew'] >= (df['pe

    #creating bool column for irma
    df['increase_irma'] = np.where(df['percent_irma'] >= (df['percent_

    #creating bool column for michael
    df['increase_michael'] = np.where(df['percent_michael'] >= (df['pe

     #creating bool column for ian
    df['increase_ian'] = np.where(df['percent_ian'] >= (df['percent_ia

    #Using pandas wide_to_long (https://pandas.pydata.org/docs/referen
    #j = hurricane name
    #i = city
    #stubname = name of variables

    df = pd.wide_to_long(df, stubnames = ['before', 'after', 'percent'

    #reseting index
    df.reset_index(inplace=True)

    return df
```

In [45]:

```
#applying the function to our dataframes
bottom = bool(bottom)
middle = bool(middle)
top = bool(top)
```

In [46]:
```
#checking it out
bottom.head()
```

Out[46]:

	City	HurricaneName	SizeRank	before	after	percent	increase
0	Jacksonville	charley	12	46528.30349	52803.32790	13.486467	0
1	Orlando	charley	16	75863.27537	88560.22345	16.736620	0
2	Miami	charley	20	86752.55847	106338.70750	22.577028	0
3	Tampa	charley	50	51585.41060	61309.89329	18.851227	0
4	Saint Petersburg	charley	84	47796.13229	57880.05754	21.097785	0

In [47]:
```
#checking it out
middle.head()
```

Out[47]:

	City	HurricaneName	SizeRank	before	after	percent	increase
0	Jacksonville	charley	12	120287.1799	136338.2043	13.343919	0
1	Orlando	charley	16	153628.1167	178133.7990	15.951300	0
2	Miami	charley	20	196585.3564	242294.9810	23.251795	1
3	Tampa	charley	50	134130.4031	158405.6253	18.098225	0
4	Saint Petersburg	charley	84	112809.7002	134746.5950	19.445930	0

In [48]:
```
#checking it out
top.head()
```

Out[48]:

	City	HurricaneName	SizeRank	before	after	percent	increase
0	Jacksonville	charley	12	219711.2386	249137.7656	13.393273	0
1	Orlando	charley	16	268593.0990	311622.3096	16.020222	0
2	Miami	charley	20	438933.0461	531656.5319	21.124745	0
3	Tampa	charley	50	314461.3881	368173.9723	17.080820	0
4	Saint Petersburg	charley	84	245925.9512	292718.6866	19.027165	0

Checking for null values

In [49]:
```
1 bottom.isna().sum()
```

Out[49]:
```
City            0
HurricaneName   0
SizeRank        0
before          0
after           0
percent         0
increase        0
dtype: int64
```

In [50]:
```
1 bottom.dropna(inplace=True)
2 bottom.isna().sum()
```

Out[50]:
```
City            0
HurricaneName   0
SizeRank        0
before          0
after           0
percent         0
increase        0
dtype: int64
```

In [51]:
```
1 middle.isna().sum()
```

Out[51]:
```
City            0
HurricaneName   0
SizeRank        0
before          0
after           0
percent         0
increase        0
dtype: int64
```

In [52]:
```
1 middle.dropna(inplace=True)
2 middle.isna().sum()
```

Out[52]:
```
City            0
HurricaneName   0
SizeRank        0
before          0
after           0
percent         0
increase        0
dtype: int64
```

In [53]:
```
1 top.isna().sum()
```

Out[53]:
```
City            0
HurricaneName   0
SizeRank        0
before          0
after           0
percent         0
increase        0
dtype: int64
```

In [54]:
```
top.dropna(inplace=True)
top.isna().sum()
```

Out[54]:
```
City            0
HurricaneName   0
SizeRank        0
before          0
after           0
percent         0
increase        0
dtype: int64
```

Saving the Datasets

Later on we will need to merge these dataset with hurricane data. Let's save it.

In [55]:
```
#saving the bottom dataset as bottom_housing.csv
bottom.to_csv(r'data\bottom_housing.csv', index=False)
#saving the middle dataset as middle_housing.csv
middle.to_csv(r'data\middle_housing.csv', index=False)
#saving the top dataset as top_housing.csv
top.to_csv(r'data\top_housing.csv', index=False)
```