Phase 4 Project Notebook

- Author: Jonathan Holt
- Data Science Flex

1 Overview & Summary

1.1 Business Problem

- FRC Investors is a nationwide real estate investment company. They buy homes all across the country and then sell them through their realty branch, FRC Homes. FRC's executive team is interested in maximizing their efforts and has tasked me with answering the question: **What are the top 5 best zip codes for us to invest in?**

1.2 Criteria:

In my meeting with the executive team, I was given the following directions:

1. FRC Investors is looking to **invest 500,000 per home per zip code**.
2. They want to diversify by investing in **5 different zip codes, in 5 different states**.
3. FRC would like to **maximize their ROI** per home.
4. FRC would like to **minimize risk**. They want "sure thing" investments.
5. FRC is ready to invest now, and **wants forecasts for the next 3 years,**.

Therefore, I classify the "best" zip codes as having:

1. Median house prices around 500,000.
2. Highest forecasted ROI over the next 3 years (2018-2020)
3. Lowest chance of loss. (confidence index lower value)

1.3 The Data

- I will be using the Zillow Median Home Price Research Dataset. I contains the median home prices for every Zip Code in the United States by month from January 1996 - April 2018.
- The data is fairly clean, although many of the Zip Codes don't have data towards the beginning of the dataset. My hypothesis is that these Zip Codes did not exist at that time. Either way, as this is all time series data, I will work with what I've got for each Zip Code.

1.4 Summary of My Process:

1) I searched the data provided for zip codes which have median housing prices near 500,000 at the end of the data (early 2018).

2) I then analyzed the performance of those zip codes over the last 3 years.

3) I ranked my findings based on ROI generated over those last 3 years.

4) I then transformed each of those zip codes into a unique time series for modeling.

5) I used SARIMAX modelling to forecast the future performance of each of my target zip Codes.

6) I used a parameter grid search to determine the optimal parameters for P, D, and Q.

7) I also withheld the last few years of data (usually 20% of dataset) as a test set to validate my model.

8) I validated my model by comparing the Root Mean Squared Error as a judge of model fit.

9) I also did a one-step forecast for each model and compared the prediction to the test set.

10) Once I determined the better performing model, I then fit that model on that zip code's entire time series to get a 3-year forecast.

11) I then analyzed the predicted mean and confidence intervals to determine how safe each zipcode would be to invest in over a three year period, as well as the potentional Return on Investment.

12) I went through my target zip codes in the order of their ranked 3 year ROI from 2016-2018.

13) In order to preserve geographic integrety (per the Criteria listed above), once a Zip Code had been selected as one of the Top 5, I bypassed any future Zip Codes from that state and pursued zip codes from the remaining states.

1.5 Functions

In [405]:
```
#provided by Flatiron
def melt_data(df):
    """
    Takes the zillow_data dataset in wide form or a subset of the zillow_dataset.
    Returns a long-form datetime Ddataframe
    with the datetime column names as the index and the values as the 'values' column.

    If more than one row is passes in the wide-form dataset, the values column
    will be the mean of the values from the datetime columns in all of the rows.
    """
    #melted = pd.melt(df, id_vars=['RegionName', 'RegionID', 'SizeRank', 'City', 'State', 'Metro', '
    melted = pd.melt(df, id_vars=['ZipCode', 'RegionID', 'SizeRank', 'City', 'State', 'Metro', 'Count
    melted['time'] = pd.to_datetime(melted['time'], infer_datetime_format=True)
    melted = melted.dropna(subset=['value'])
    return melted.groupby('time').aggregate({'value':'mean'})
```

executed in 17ms, finished 14:24:59 2022-05-24

In [406]:
```
#function for displaying money in millions.
def display_millions(x, pos):
    return '${:1.1f}M'.format(x*1e-6)
```

executed in 2ms, finished 14:24:59 2022-05-24

2 PreProcessing

2.1 Loading Data

In [407]:
```
import warnings
warnings.filterwarnings('ignore')
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import matplotlib.ticker as mtick
from matplotlib.pylab import rcParams
import seaborn as sns
import statsmodels.api as sm
from statsmodels.tsa.stattools import adfuller
from statsmodels.tsa.seasonal import seasonal_decompose
from statsmodels.tsa.statespace.sarimax import SARIMAX
import itertools
#from sklearn.metrics import mean_squared_error

plt.style.use('fivethirtyeight')
#plt.style.use('ggplot')
pd.set_option('display.max_rows', 1500) #change the amount of rows displayed

pd.options.display.float_format = '{:,.2f}'.format
```

executed in 40ms, finished 14:24:59 2022-05-24

I start by loading the data from the zillow_data.csv provided for this project.

In [408]:
```
ts = pd.read_csv("zillow_data.csv")
ts.head()
```

executed in 347ms, finished 14:25:00 2022-05-24

Out[408]:

	RegionID	RegionName	City	State	Metro	CountyName	SizeRank	1996-04	1996-05	1996-06	...	2017-07	2017-08	20
0	84654	60657	Chicago	IL	Chicago	Cook	1	334,200.00	335,400.00	336,500.00	...	1005500	1007500	10
1	90668	75070	McKinney	TX	Dallas-Fort Worth	Collin	2	235,700.00	236,900.00	236,700.00	...	308000	310000	3
2	91982	77494	Katy	TX	Houston	Harris	3	210,400.00	212,200.00	212,200.00	...	321000	320600	3
3	84616	60614	Chicago	IL	Chicago	Cook	4	498,100.00	500,900.00	503,100.00	...	1289800	1287700	12
4	93144	79936	El Paso	TX	El Paso	El Paso	5	77,300.00	77,300.00	77,300.00	...	119100	119400	1

5 rows × 272 columns

2.2 Changing RegionName to ZipCode

A google search shows that RegionName is the ZipCode for each Region. However, upon sorting, I discovered that any ZipCode beginning with a 0 was ignoring it and displaying as a 4 digit number. I will use the .str().zfill() method to ensure that all RegionNames are displaying the as 5 digits.

In [409]:
```
ts['RegionName'] = ts['RegionName'].astype(str).str.zfill(5)
ts.rename(columns={'RegionName': 'ZipCode'}, inplace=True)
```

executed in 29ms, finished 14:25:00 2022-05-24

2.3 Checking for Null Values

In [410]:
```
#ts.isnull().sum()
```

executed in 1ms, finished 14:25:00 2022-05-24

Analysis

- There are many Null Values.
- For categorical data, 7% of Metro are null (1043 of 14,723).
- Dates from 1996 - mid 2003, also have 7% null values.
- Then it starts to get better. 6% null and decreasing.

Decision: I will keep all of the nulls for now. I will likely care the most about the most recent data, and that is where the records are most complete.

2.4 Metro

- The null values for Metro are because of rural zip codes that aren't close enough to a metro area to be classified as such. I will create a new value for Metro of 'no_metro' to represent these rural zip codes.
- NOTE: for any analysis of Metro areas, 'no_metro' should be excluded as these zip codes are NOT part of the same area.

In [411]:
```
1 ts['Metro'].value_counts().head()
```
executed in 6ms, finished 14:25:00 2022-05-24

Out[411]:
```
New York                          779
Los Angeles-Long Beach-Anaheim    347
Chicago                           325
Philadelphia                      281
Washington                        249
Name: Metro, dtype: int64
```

In [412]:
```
1 ts['Metro'].fillna('no_metro', inplace=True)
```
executed in 4ms, finished 14:25:00 2022-05-24

In [413]:
```
1 ts['Metro'].value_counts().head()
```
executed in 5ms, finished 14:25:00 2022-05-24

Out[413]:
```
no_metro                         1043
New York                          779
Los Angeles-Long Beach-Anaheim    347
Chicago                           325
Philadelphia                      281
Name: Metro, dtype: int64
```

In [414]:
```
1 #ts.isna().sum()
```
executed in 2ms, finished 14:25:00 2022-05-24

- All remaining null values are for the time series data. I will leave them as is for now, and deal with them later on as it will likely depend on how I am grouping the data, etc.

2.5 Changing dtype for all housing data to float.

In [415]:
```
1 ts[ts.columns[7:]] = ts[ts.columns[7:]].astype(float)
```
executed in 102ms, finished 14:25:00 2022-05-24

3 Exploratory Data Analysis

3.1 Creating avg_home_df

- Create Dataset that is just the mean values to do analysis of trends, etc for baseline.
- I can then look for zip codes that have the best performance above and beyond baseline.

In [416]:
```
1 data_values = ts.iloc[:, 7:]
2 data_values['2018-04'].dtype
```
executed in 9ms, finished 14:25:00 2022-05-24

Out[416]:
```
dtype('float64')
```

In [417]:
```
1 avg_home_df = pd.DataFrame(data_values.mean())
2 avg_home_df = avg_home_df.reset_index()
3 avg_home_df['time'] = pd.to_datetime(avg_home_df['index'])
4 avg_home_df.set_index('time', inplace=True)
5 avg_home_df.drop(columns=('index'), inplace=True)
6 avg_home_df.rename(columns={0: 'home_value'}, inplace=True)
```
executed in 68ms, finished 14:25:00 2022-05-24

In [418]:
```
1 avg_home_df.head()
```

executed in 4ms, finished 14:25:00 2022-05-24

Out[418]:

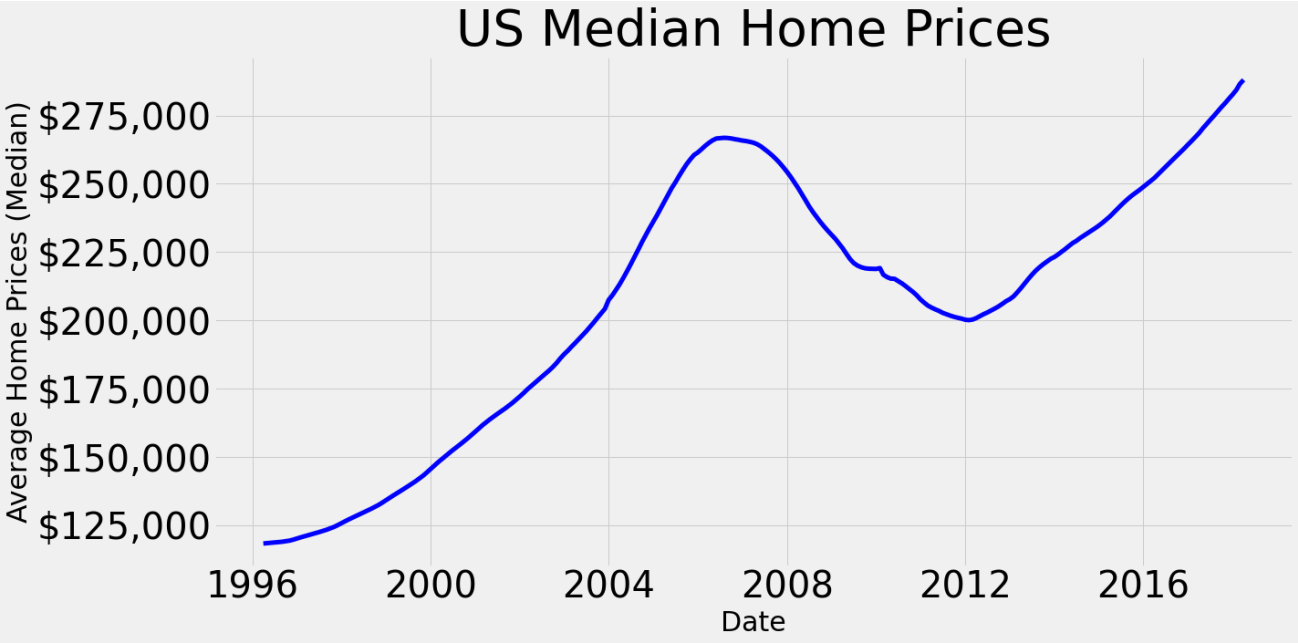
time	home_value
1996-04-01	118,299.12
1996-05-01	118,419.04
1996-06-01	118,537.42
1996-07-01	118,653.07
1996-08-01	118,780.25

3.2 Analysis of Avg_Home_DF

In [419]:

```
fig, ax = plt.subplots(figsize=(20, 10))

p = sns.lineplot(data=avg_home_df, x='time', y='home_value', color='blue', linewidth=5);


p.set_xlabel("Date", fontsize = 30)
p.set_ylabel("Average Home Prices (Median)", fontsize = 30)

#y1 = p.axvline('2008-01', color='red') #housing market crash begins
#y2 = p.axvline('2012-01', color='red') #housing market crash ends
#ax.fill_between(y1, y2, color='yellow')

plt.xticks(fontsize=40)
plt.yticks(fontsize=40)

fmt = '${x:,.0f}'
tick = mtick.StrMethodFormatter(fmt)
ax.yaxis.set_major_formatter(tick)

p.set_title("US Median Home Prices", fontsize = 55)
plt.figsize=(50,25)
plt.savefig('images/us_median_plot_1')

plt.tight_layout()

plt.show();
```

executed in 300ms, finished 14:25:00 2022-05-24



3.3 Analysis:

- Home Sales Values are on an upward trend, however there was a significant dip from aprox 2006 - 2012, before recovering and trending upward again.
- The dip was due to the US Housing Market Crash of 2008. (https://en.wikipedia.org/wiki/United_States_housing_bubble (https://en.wikipedia.org/wiki/United_States_housing_bubble))
- The current trend seems to be at a lower and more stable rate than it was before the crash.

3.4 Testing for Stationarity: Dickey Fuller Test

In [420]:
```
# Perform Dickey-Fuller test:
print ('Results of Dickey-Fuller Test: \n')
#dftest = adfuller(data['#Passengers'])
dftest = adfuller(avg_home_df)

# Extract and display test results in a user friendly manner
dfoutput = pd.Series(dftest[0:4], index=['Test Statistic', 'p-value', '#Lags Used', 'Number of Obser
for key,value in dftest[4].items():
    dfoutput['Critical Value (%s)'%key] = value
print(dfoutput)
```

executed in 24ms, finished 14:25:00 2022-05-24

```
Results of Dickey-Fuller Test:

Test Statistic                 -1.89
p-value                         0.34
#Lags Used                      2.00
Number of Observations Used   262.00
Critical Value (1%)            -3.46
Critical Value (5%)            -2.87
Critical Value (10%)           -2.57
dtype: float64
```

3.5 Analysis:

- The p-value is above .05, therefore I do NOT reject the null hypothesis, and therefore this data is NOT stationary.
- Because I plan on using SARIMAX modeling, I do not need to make this data stationary in order to model.

4 Feature Engineering

- I want to add additional features to the DataSet so that I can determine what data I want to analyze further and ultimately model.
- for example: mean home value as well as ROI over the last 3 years.

In [421]:
```
new_df = ts.copy()
```

executed in 6ms, finished 14:25:00 2022-05-24

In [422]:
```
#making sure that all the data fields were changed to floats
new_df['2018-04'].dtype
```

executed in 3ms, finished 14:25:00 2022-05-24

Out[422]:
```
dtype('float64')
```

In [423]:
```
new_df['mean'] = data_values.apply(lambda x: x.mean(), axis=1)
new_df.head(1)
```

executed in 709ms, finished 14:25:01 2022-05-24

Out[423]:

	RegionID	ZipCode	City	State	Metro	CountyName	SizeRank	1996-04	1996-05	1996-06	...	2017-08	2017-09
0	84654	60657	Chicago	IL	Chicago	Cook	1	334,200.00	335,400.00	336,500.00	...	1,007,500.00	1,007,800.00

1 rows × 273 columns

4.1 Recent Data (last 3 Years) DataFrame

First, I will slice out the last 3 years of data to get metrics on recent performance. This will help me narrow down my target zip codes as I look at recent performance to see which zip codes have performed significantly better than average.

In [424]:
```
1 avg_home_df.describe()
```

executed in 9ms, finished 14:25:01 2022-05-24

Out[424]:

	home_value
count	265.00
mean	206,661.51
std	47,809.38
min	118,299.12
25%	168,653.36
50%	215,113.13
75%	245,934.66
max	288,039.94

RegionID is too easy to mistake for the ZipCode as they are both 5 digit number. Each Zip Code is also unique, so there is no need for an addtional ID field. I will drop RegionID to avoid any confusion.

In [425]:
```
1 new_df.drop(columns=['RegionID'], inplace=True)
```

executed in 62ms, finished 14:25:01 2022-05-24

In [426]:
```
1 new_df.describe()
```

executed in 399ms, finished 14:25:01 2022-05-24

Out[426]:

	SizeRank	1996-04	1996-05	1996-06	1996-07	1996-08	1996-09	1996-10	1996-11	1996-12	..
count	14,723.00	13,684.00	13,684.00	13,684.00	13,684.00	13,684.00	13,684.00	13,684.00	13,684.00	13,684.00	..
mean	7,362.00	118,299.12	118,419.04	118,537.42	118,653.07	118,780.25	118,927.53	119,120.52	119,345.35	119,685.08	..
std	4,250.31	86,002.51	86,155.67	86,309.23	86,467.95	86,650.94	86,872.08	87,151.85	87,479.81	87,912.69	..
min	1.00	11,300.00	11,500.00	11,600.00	11,800.00	11,800.00	12,000.00	12,100.00	12,200.00	12,300.00	..
25%	3,681.50	68,800.00	68,900.00	69,100.00	69,200.00	69,375.00	69,500.00	69,600.00	69,800.00	70,000.00	..
50%	7,362.00	99,500.00	99,500.00	99,700.00	99,700.00	99,800.00	99,900.00	99,950.00	100,100.00	100,300.00	..
75%	11,042.50	143,200.00	143,300.00	143,225.00	143,225.00	143,500.00	143,700.00	143,900.00	144,125.00	144,300.00	..
max	14,723.00	3,676,700.00	3,704,200.00	3,729,600.00	3,754,600.00	3,781,800.00	3,813,500.00	3,849,600.00	3,888,900.00	3,928,800.00	..

8 rows × 267 columns

In [427]:
```
#slicing out the data values for preprocessing
new_df.iloc[:, 6:-41]
```

executed in 13ms, finished 14:25:01 2022-05-24

Out[427]:

	1996-04	1996-05	1996-06	1996-07	1996-08	1996-09	1996-10	1996-11	1996-12	1997-01	...	2014-0
0	334,200.00	335,400.00	336,500.00	337,600.00	338,500.00	339,500.00	340,400.00	341,300.00	342,600.00	344,400.00	...	863,900.0
1	235,700.00	236,900.00	236,700.00	235,400.00	233,300.00	230,600.00	227,300.00	223,400.00	219,600.00	215,800.00	...	234,200.0
2	210,400.00	212,200.00	212,200.00	210,700.00	208,300.00	205,500.00	202,500.00	199,800.00	198,300.00	197,300.00	...	282,100.0
3	498,100.00	500,900.00	503,100.00	504,600.00	505,500.00	505,700.00	505,300.00	504,200.00	503,600.00	503,400.00	...	1,149,900.0
4	77,300.00	77,300.00	77,300.00	77,300.00	77,400.00	77,500.00	77,600.00	77,700.00	77,700.00	77,800.00	...	112,000.0
...	...	...	...	...	...	...	...	...	...	...	...	.
14718	94,600.00	94,300.00	94,000.00	93,700.00	93,400.00	93,200.00	93,000.00	92,900.00	92,700.00	92,600.00	...	187,600.0
14719	92,700.00	92,500.00	92,400.00	92,200.00	92,100.00	91,900.00	91,700.00	91,300.00	90,900.00	90,500.00	...	180,000.0
14720	57,100.00	57,300.00	57,500.00	57,700.00	58,000.00	58,200.00	58,400.00	58,700.00	59,100.00	59,500.00	...	105,100.0
14721	191,100.00	192,400.00	193,700.00	195,000.00	196,300.00	197,700.00	199,100.00	200,700.00	202,600.00	204,900.00	...	528,900.0
14722	176,400.00	176,300.00	176,100.00	176,000.00	175,900.00	175,800.00	175,800.00	176,000.00	176,200.00	176,500.00	...	266,800.0

14723 rows × 225 columns

In [428]:
```
recent_df= new_df.drop(new_df.iloc[:, 6:-41], axis=1)
```

executed in 6ms, finished 14:25:01 2022-05-24

In [429]:
```
recent_df.describe()
```

executed in 69ms, finished 14:25:02 2022-05-24

Out[429]:

	SizeRank	2015-01	2015-02	2015-03	2015-04	2015-05	2015-06	2015-07	2015-08	2(
count	14,723.00	14,723.00	14,723.00	14,723.00	14,723.00	14,723.00	14,723.00	14,723.00	14,723.00	14,
mean	7,362.00	234,691.19	235,760.28	236,836.43	238,009.76	239,354.21	240,744.12	242,103.53	243,424.44	244,
std	4,250.31	307,979.07	312,430.39	316,062.13	317,997.85	319,992.45	323,438.78	327,255.34	330,084.81	331,
min	1.00	13,600.00	13,600.00	13,400.00	13,300.00	13,300.00	13,500.00	13,800.00	14,000.00	14,
25%	3,681.50	110,000.00	110,300.00	111,000.00	111,500.00	112,150.00	112,600.00	113,100.00	113,500.00	114,
50%	7,362.00	162,900.00	163,300.00	163,700.00	164,200.00	165,100.00	165,800.00	166,500.00	167,100.00	167,
75%	11,042.50	260,250.00	261,200.00	262,000.00	263,450.00	264,950.00	266,600.00	268,200.00	269,900.00	271,
max	14,723.00	15,663,600.00	16,271,900.00	16,684,700.00	16,644,000.00	16,659,500.00	17,149,200.00	17,775,200.00	17,965,800.00	17,853,

8 rows × 42 columns

In [430]:
```
recent_df.drop(columns = ['mean'], inplace=True)
recent_df.head(1)
```

executed in 10ms, finished 14:25:02 2022-05-24

Out[430]:

	ZipCode	City	State	Metro	CountyName	SizeRank	2015-01	2015-02	2015-03	2015-04	...	2017-07	2017-08
0	60657	Chicago	IL	Chicago	Cook	1	893,000.00	895,000.00	901,200.00	909,400.00	...	1,005,500.00	1,007,500.00

1 rows × 46 columns

In [431]:
```
recent_df.iloc[:, 6:]
```
executed in 12ms, finished 14:25:02 2022-05-24

Out[431]:

	2015-01	2015-02	2015-03	2015-04	2015-05	2015-06	2015-07	2015-08	2015-09	2015-10
0	893,000.00	895,000.00	901,200.00	909,400.00	915,000.00	916,700.00	917,700.00	919,800.00	925,800.00	937,100.00
1	251,400.00	253,000.00	255,200.00	258,000.00	261,200.00	264,700.00	268,400.00	271,400.00	273,600.00	275,200.00
2	301,700.00	302,400.00	303,600.00	306,200.00	309,100.00	311,900.00	314,100.00	316,300.00	319,000.00	322,000.00
3	1,176,400.00	1,174,600.00	1,178,500.00	1,185,700.00	1,192,900.00	1,198,800.00	1,200,400.00	1,198,900.00	1,200,200.00	1,207,400.00
4	114,700.00	115,000.00	115,000.00	115,200.00	115,600.00	115,900.00	115,600.00	115,400.00	115,400.00	115,500.00
...	...	...	...	...	...	...	...	...	...	...
14718	192,400.00	193,300.00	193,400.00	192,000.00	191,200.00	190,900.00	190,900.00	191,100.00	191,900.00	191,400.00
14719	183,000.00	181,900.00	182,300.00	184,400.00	186,300.00	188,300.00	190,800.00	191,800.00	189,500.00	187,700.00
14720	104,600.00	104,400.00	104,400.00	105,000.00	105,900.00	106,800.00	107,500.00	107,700.00	107,800.00	108,300.00
14721	520,700.00	517,500.00	538,300.00	552,700.00	545,800.00	536,100.00	538,300.00	537,800.00	539,000.00	543,500.00
14722	287,500.00	288,300.00	289,100.00	291,000.00	293,300.00	295,200.00	296,900.00	298,400.00	299,200.00	299,500.00

14723 rows × 40 columns

In [432]:
```
recent_df['3yr_Mean'] = recent_df.iloc[:, 6:].apply(lambda x: x.mean(), axis=1)
recent_df['ROI'] = recent_df.iloc[:, 6:].apply(lambda x: x['2018-04'] - x['2015-01'], axis=1)
recent_df['ROI%'] = recent_df.iloc[:, 6:].apply(lambda x: x['ROI'] / x['2015-01'], axis=1)
```
executed in 1.00s, finished 14:25:03 2022-05-24

In [433]:
```
recent_df.head(1)
```
executed in 9ms, finished 14:25:03 2022-05-24

Out[433]:

	ZipCode	City	State	Metro	CountyName	SizeRank	2015-01	2015-02	2015-03	2015-04	...	2017-10	2017-11
0	60657	Chicago	IL	Chicago	Cook	1	893,000.00	895,000.00	901,200.00	909,400.00	...	1,009,600.00	1,013,300.00

1 rows × 49 columns

Recent_df now contains all data from the last 3 years. I will now sort the data by ROI Percentage Gain

In [434]:
```
recent_df.sort_values('2018-04', ascending=False).head()
```
executed in 13ms, finished 14:25:03 2022-05-24

Out[434]:

	ZipCode	City	State	Metro	CountyName	SizeRank	2015-01	2015-02	2015-03	2015-04	...	2017
272	10021	New York	NY	New York	New York	273	15,663,600.00	16,271,900.00	16,684,700.00	16,644,000.00	...	18,569,400
20	10011	New York	NY	New York	New York	21	10,476,500.00	10,544,400.00	10,569,300.00	10,572,500.00	...	12,050,100
508	10014	New York	NY	New York	New York	509	10,078,900.00	10,143,700.00	10,075,800.00	9,938,600.00	...	9,631,000
21	10128	New York	NY	New York	New York	22	7,384,300.00	7,618,100.00	7,914,200.00	8,244,000.00	...	7,427,300
10237	94027	Atherton	CA	San Francisco	San Mateo	10238	4,846,700.00	4,950,900.00	5,053,600.00	5,140,300.00	...	6,315,400

5 rows × 49 columns

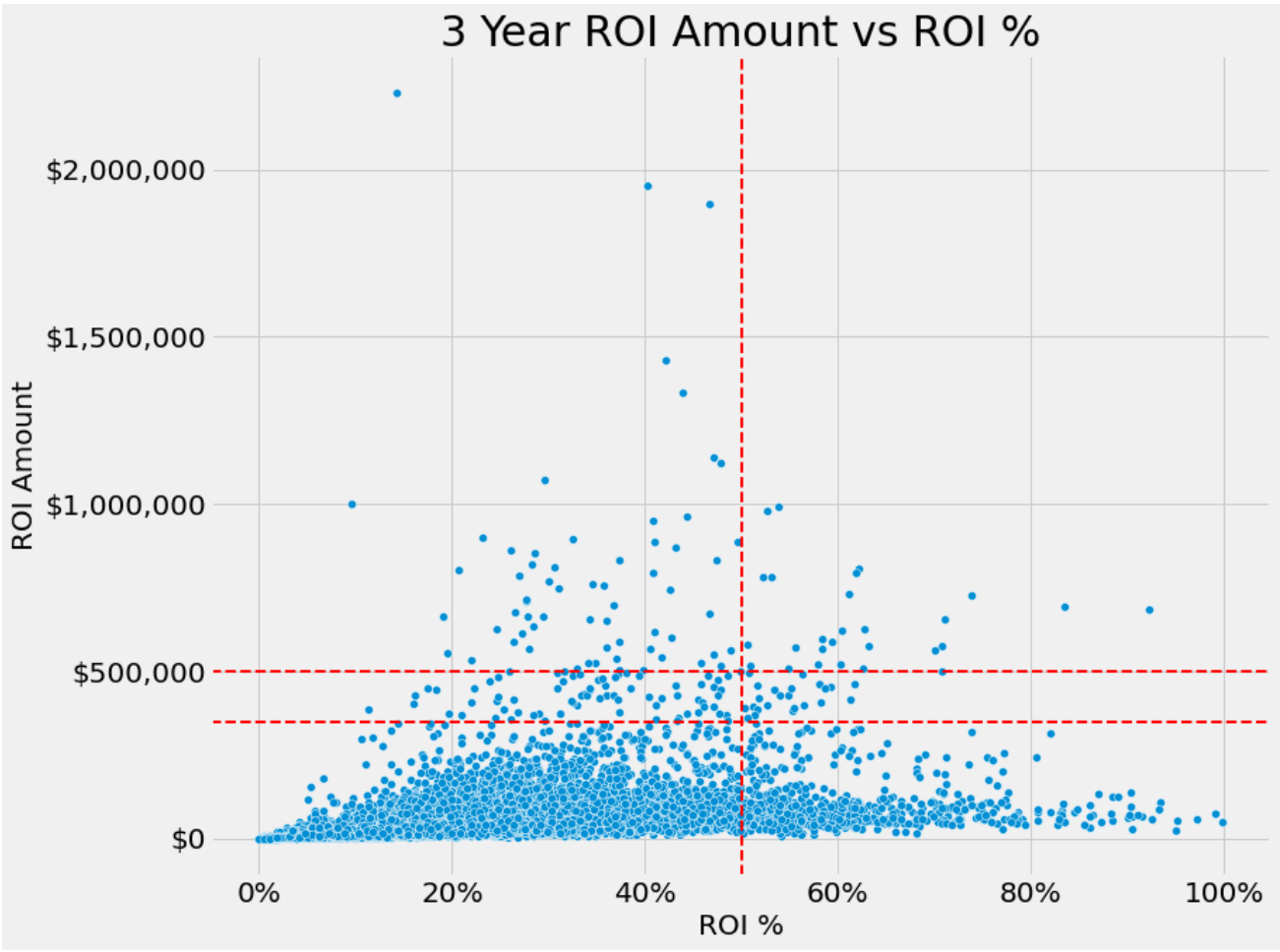
4.2 Scatter Plot

In [435]:
```
#slicing the df so that it shows only postivie ROI and a ROI% less than 1 for my scatter plot.
scatter_plot_df = recent_df.copy()
scatter_plot_df = scatter_plot_df[scatter_plot_df['ROI'] >= 0]
scatter_plot_df = scatter_plot_df[scatter_plot_df['ROI%'] <= 1]
```

executed in 8ms, finished 14:25:03 2022-05-24

In [436]:
```
fig, ax = plt.subplots(figsize=(12, 10))
p = sns.scatterplot(data=scatter_plot_df, x="ROI%", y="ROI");


p.set_xlabel("ROI %", fontsize = 20)
p.set_ylabel("ROI Amount", fontsize = 20)

p.axhline(350000, color='red', ls='--', linewidth=(2))
p.axhline(500000, color='red', ls='--', linewidth=(2))
p.axvline(.5, color='red',  ls='--', linewidth=(2))



plt.xticks(fontsize=20)
plt.yticks(fontsize=20)

fmt = '${x:,.0f}'
tick = mtick.StrMethodFormatter(fmt)
ax.yaxis.set_major_formatter(tick)
ax.xaxis.set_major_formatter(mtick.PercentFormatter(xmax=1, decimals=None, symbol='%', is_latex=Fals
p.set_title("3 Year ROI Amount vs ROI %", fontsize = 30)
plt.figsize=(50,25)
plt.savefig('images/3yr_ROI_scatterplot')


plt.show();
```

executed in 185ms, finished 14:25:03 2022-05-24



Analysis:

- The red lines show where my target zip codes are. Per the criteria that I was given, the client wants to invest aproximately $500,000 per house per zip code. So I have set a range of 350,000 - 500,000 for home value. I also drew a vertical line at 50 Percent ROI as I am interested in the Zip Codes that are performing exceptionally well. The section of the scatterplot between the two horizontal red lines and to the right of the vertical line is where the Target Zip Codes are located.

In [437]:
```
1 top_roi = recent_df[recent_df['ROI%'] >= 0.5]
```
executed in 4ms, finished 14:25:03 2022-05-24

In [438]:
```
1 top_roi.sort_values('2018-04', ascending=False).head(20)
```
executed in 15ms, finished 14:25:03 2022-05-24

Out[438]:

	ZipCode	City	State	Metro	CountyName	SizeRank	2015-01	2015-02	2015-03	2015-04	...	2017-10
11977	98039	Medina	WA	Seattle	King	11978	1,860,100.00	1,878,500.00	1,896,300.00	1,918,800.00	...	2,669,300.00
1791	94040	Mountain View	CA	San Jose	Santa Clara	1792	1,846,600.00	1,875,500.00	1,906,800.00	1,940,700.00	...	2,541,700.00
516	94087	Sunnyvale	CA	San Jose	Santa Clara	517	1,496,100.00	1,519,400.00	1,538,500.00	1,557,600.00	...	1,982,300.00
6345	94041	Mountain View	CA	San Jose	Santa Clara	6346	1,472,900.00	1,499,700.00	1,521,300.00	1,539,900.00	...	1,986,700.00
1125	98004	Bellevue	WA	Seattle	King	1126	1,297,500.00	1,312,000.00	1,324,600.00	1,341,100.00	...	1,943,400.00
2580	94043	Mountain View	CA	San Jose	Santa Clara	2581	1,281,500.00	1,311,800.00	1,343,000.00	1,369,200.00	...	1,812,000.00
582	94086	Sunnyvale	CA	San Jose	Santa Clara	583	1,195,500.00	1,220,900.00	1,239,200.00	1,257,400.00	...	1,695,300.00
5919	94618	Oakland	CA	San Francisco	Alameda	5920	1,146,500.00	1,164,400.00	1,189,300.00	1,212,900.00	...	1,614,900.00
7007	11963	Noyack	NY	New York	Suffolk	7008	986,900.00	989,800.00	988,000.00	984,100.00	...	1,608,000.00
3964	98040	Mercer Island	WA	Seattle	King	3965	1,026,000.00	1,038,200.00	1,053,100.00	1,073,200.00	...	1,545,700.00
117	11211	New York	NY	New York	Kings	118	1,025,700.00	1,019,200.00	1,041,600.00	1,076,800.00	...	1,435,300.00
7770	95130	San Jose	CA	San Jose	Santa Clara	7771	996,200.00	1,011,100.00	1,024,900.00	1,041,200.00	...	1,419,000.00
475	11216	New York	NY	New York	Kings	476	1,027,800.00	1,048,100.00	1,059,800.00	1,071,200.00	...	1,567,700.00
289	95051	Santa Clara	CA	San Jose	Santa Clara	290	922,300.00	938,100.00	950,400.00	962,800.00	...	1,365,400.00
4730	94704	Berkeley	CA	San Francisco	Alameda	4731	987,900.00	1,004,900.00	1,033,900.00	1,053,400.00	...	1,462,300.00
656	95008	Campbell	CA	San Jose	Santa Clara	657	972,500.00	988,600.00	1,002,000.00	1,016,400.00	...	1,351,000.00
1464	94610	Oakland	CA	San Francisco	Alameda	1465	1,013,300.00	1,028,800.00	1,045,500.00	1,061,800.00	...	1,458,600.00
2386	95134	San Jose	CA	San Jose	Santa Clara	2387	831,000.00	863,300.00	896,100.00	922,400.00	...	1,312,600.00
3877	95117	San Jose	CA	San Jose	Santa Clara	3878	910,500.00	922,100.00	930,500.00	939,900.00	...	1,275,900.00
536	95125	San Jose	CA	San Jose	Santa Clara	537	973,900.00	986,300.00	999,200.00	1,015,100.00	...	1,324,600.00

20 rows × 49 columns

4.3 Target Zips

In [439]:
```
1 target_zips = top_roi[top_roi['2018-04'] <= 500000]
```
executed in 3ms, finished 14:25:03 2022-05-24

In [440]:
```
target_zips.head(1)
```
executed in 9ms, finished 14:25:03 2022-05-24

Out[440]:

	ZipCode	City	State	Metro	CountyName	SizeRank	2015-01	2015-02	2015-03	2015-04	...	2017-10	2017-11
17	37211	Nashville	TN	Nashville	Davidson	18	170,600.00	171,700.00	172,800.00	174,400.00	...	248,800.00	251,100.00

1 rows × 49 columns

In [441]:
```
target_zips = target_zips.sort_values('2018-04', ascending=False)
```
executed in 3ms, finished 14:25:03 2022-05-24

In [442]:
```
# dropping unecessary data for ease of reading
target_zips = target_zips.drop(target_zips.iloc[:, 7:-4], axis=1)
```
executed in 3ms, finished 14:25:03 2022-05-24

4.4 List of Target ZipCodes

In [443]:
```
top_zips = target_zips.sort_values('ROI', ascending=False).head(15)
```
executed in 3ms, finished 14:25:03 2022-05-24

In [444]:
```
#slicing out the columns that I don't need.
top_zips.drop(columns=(['SizeRank', 'Metro', 'CountyName']))
```
executed in 8ms, finished 14:25:03 2022-05-24

Out[444]:

	ZipCode	City	State	2015-01	2018-04	3yr_Mean	ROI	ROI%
4678	29403	Charleston	SC	261,400.00	462,700.00	343,962.50	201,300.00	0.77
3781	98146	Burien	WA	292,100.00	491,100.00	391,625.00	199,000.00	0.68
11899	95570	Trinidad	CA	282,300.00	480,600.00	375,162.50	198,300.00	0.70
11136	37046	College Grove	TN	268,900.00	459,800.00	350,822.50	190,900.00	0.71
5021	98043	Mountlake Terrace	WA	287,800.00	474,700.00	366,512.50	186,900.00	0.65
4733	98178	Bryn Mawr-Skyway	WA	270,300.00	455,100.00	365,355.00	184,800.00	0.68
4712	94621	Oakland	CA	231,000.00	405,500.00	315,055.00	174,500.00	0.76
6515	90304	Inglewood	CA	326,100.00	494,300.00	404,962.50	168,200.00	0.52
1921	02128	Boston	MA	323,700.00	489,900.00	398,212.50	166,200.00	0.51
2804	98168	Burien	WA	229,800.00	393,300.00	313,725.00	163,500.00	0.71
4860	02121	Boston	MA	290,100.00	452,700.00	376,882.50	162,600.00	0.56
2155	80204	Denver	CO	210,500.00	371,600.00	291,225.00	161,100.00	0.77
13123	95918	Browns Valley	CA	312,500.00	473,600.00	377,520.00	161,100.00	0.52
319	07087	Union City	NJ	289,500.00	448,900.00	336,135.00	159,400.00	0.55
10589	98359	Olalla	WA	285,700.00	444,600.00	370,297.50	158,900.00	0.56

Analysis:

- Above is the list of Target Zip Codes ranked by ROI Amount. This is the order in which I will analyze and model them.
- Looking at the dataframe There is a lot of potential as these zip codes have all generated a large amount of ROI (at least 50% of their January 2015 value).

4.5 Turning Target Zips into Time Series

Target Zip Codes will now be turned into time series so that I can use them for modeling.

In [445]:
```
data = ts[ts['ZipCode'] == '29403']
melted_data = melt_data(data)
zip_29403_ts = melted_data
```
executed in 26ms, finished 14:25:03 2022-05-24

In [446]:
```
data = ts[ts['ZipCode'] == '98178']
melted_data = melt_data(data)
zip_98178_ts = melted_data
```
executed in 11ms, finished 14:25:03 2022-05-24

In [447]:
```
data = ts[ts['ZipCode'] == '98168']
melted_data = melt_data(data)
zip_98168_ts = melted_data
```
executed in 11ms, finished 14:25:03 2022-05-24

In [448]:
```
data = ts[ts['ZipCode'] == '94621']
melted_data = melt_data(data)
zip_94621_ts = melted_data
```
executed in 10ms, finished 14:25:03 2022-05-24

In [449]:
```
data = ts[ts['ZipCode'] == '98146']
melted_data = melt_data(data)
zip_98146_ts = melted_data
```
executed in 10ms, finished 14:25:03 2022-05-24

In [450]:
```
data = ts[ts['ZipCode'] == '95570']
melted_data = melt_data(data)
zip_95570_ts = melted_data
```
executed in 11ms, finished 14:25:03 2022-05-24

In [451]:
```
data = ts[ts['ZipCode'] == '37046']
melted_data = melt_data(data)
zip_37046_ts = melted_data
```
executed in 10ms, finished 14:25:03 2022-05-24

In [452]:
```
data = ts[ts['ZipCode'] == '98043']
melted_data = melt_data(data)
zip_98043_ts = melted_data
```
executed in 10ms, finished 14:25:03 2022-05-24

In [453]:
```
data = ts[ts['ZipCode'] == '98178']
melted_data = melt_data(data)
zip_98178_ts = melted_data
```
executed in 10ms, finished 14:25:03 2022-05-24

In [454]:
```
data = ts[ts['ZipCode'] == '90304']
melted_data = melt_data(data)
zip_90304_ts = melted_data
```
executed in 11ms, finished 14:25:03 2022-05-24

In [455]:
```
data = ts[ts['ZipCode'] == '02128']
melted_data = melt_data(data)
zip_02128_ts = melted_data
```
executed in 10ms, finished 14:25:03 2022-05-24

In [456]:
```
data = ts[ts['ZipCode'] == '02121']
melted_data = melt_data(data)
zip_02121_ts = melted_data_ts = melted_data
```
executed in 11ms, finished 14:25:03 2022-05-24

In [457]:
```
data = ts[ts['ZipCode'] == '95918']
melted_data = melt_data(data)
zip_95918_ts = melted_data
```
executed in 11ms, finished 14:25:03 2022-05-24

In [458]:
```
data = ts[ts['ZipCode'] == '07087']
melted_data = melt_data(data)
zip_07087_ts = melted_data
```
executed in 10ms, finished 14:25:03 2022-05-24

In [459]:
```
data = ts[ts['ZipCode'] == '90304']
melted_data = melt_data(data)
zip_90304_ts = melted_data
```
executed in 10ms, finished 14:25:03 2022-05-24

In [460]:
```
data = ts[ts['ZipCode'] == '80102']
melted_data = melt_data(data)
zip_80102_ts = melted_data
```
executed in 10ms, finished 14:25:03 2022-05-24

In [461]:
```
data = ts[ts['ZipCode'] == '98359']
melted_data = melt_data(data)
zip_98359_ts = melted_data
```
executed in 10ms, finished 14:25:03 2022-05-24

In [462]:
```
data = ts[ts['ZipCode'] == '84757']
melted_data = melt_data(data)
zip_84757_ts = melted_data
```
executed in 10ms, finished 14:25:03 2022-05-24

In [463]:
```
data = ts[ts['ZipCode'] == '89704']
melted_data = melt_data(data)
zip_89704_ts = melted_data
```
executed in 10ms, finished 14:25:03 2022-05-24

In [464]:
```
data = ts[ts['ZipCode'] == '80204']
melted_data = melt_data(data)
zip_80204_ts = melted_data
```
executed in 10ms, finished 14:25:03 2022-05-24

Modeling

4.6 Param Search Function

In [465]:
```
#from Flatiron SARIMA Lab

def sarimax_param_search(ts):

    # Setting up parameter combinations.
    p = d = q = (0, 1)
    #p = d = q = (0, 1, 2) Alternative parameter combation for more refinement. WARNING: takes a lon

    pdq = [(ar, diff, ma) for ar in p for diff in d for ma in q]
    pdqs = [(c[0], c[1], c[2], 12) for c in pdq]

    # Iterate and try models.
    combo, value = (None, None)
    for pdq_combo in pdq:
        for pdqs_combo in pdqs:
            model = SARIMAX(
                ts,
                order=pdq_combo,
                seasonal_order=pdqs_combo,
                enforce_stationarity=False,
                enforce_invertibility=False
                )
            output = model.fit()

            if value is None or output.aic < value:
                combo, value = ((pdq_combo, pdqs_combo), output.aic)
                print('SARIMA Combos:', (pdq_combo, pdqs_combo),'AIC:', output.aic)

                combo = combo


    return print(f'\n{"*"*20}\nOptimal SARIMA order: {combo}')
```

executed in 4ms, finished 14:25:03 2022-05-24

In [466]:
```
#sarimax_param_search(avg_home_df)
```

executed in 2ms, finished 14:25:03 2022-05-24

In [467]:
```
#AIC: 3400.45
avg_home_df_combo_1 = ((1, 1, 1), (1, 1, 1, 12))
```

executed in 1ms, finished 14:25:03 2022-05-24

In [468]:
```
#advanced combo search (0:2)
#sarimax_param_search(avg_home_df)
```

executed in 1ms, finished 14:25:03 2022-05-24

In [469]:
```
#advanced search combo: AIC: 3112.12
avg_home_df_combo_2 = ((1, 2, 2), (2, 2, 2, 12))
```

executed in 2ms, finished 14:25:03 2022-05-24

5 Modeling

5.1 Avg Home Prices for US

5.1.1 Model #1

In [470]:
```
combo = avg_home_df_combo_1
#combo = avg_home_df_combo_2
```

executed in 2ms, finished 14:25:03 2022-05-24

In [471]:
```
temp_ts = avg_home_df

SPLIT = int(temp_ts.shape[0]*0.8)
train, test = temp_ts[:SPLIT], temp_ts[SPLIT:]
```

executed in 2ms, finished 14:25:03 2022-05-24

In [472]:
```
# Model 1
sarima_model = SARIMAX(
    train,
    order=combo[0],
    seasonal_order=combo[1],
    enforce_stationarity=False,
    enforce_invertibility=False).fit()
sarima_model.summary()
```

executed in 176ms, finished 14:25:03 2022-05-24

Out[472]:

SARIMAX Results

Dep. Variable:	home_value	**No. Observations:**	212
Model:	SARIMAX(1, 1, 1)x(1, 1, 1, 12)	**Log Likelihood**	-1365.256
Date:	Tue, 24 May 2022	**AIC**	2740.511
Time:	14:25:03	**BIC**	2756.613
Sample:	04-01-1996	**HQIC**	2747.037
	- 11-01-2013		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	0.9603	0.029	32.707	0.000	0.903	1.018
ma.L1	-0.2056	0.046	-4.452	0.000	-0.296	-0.115
ar.S.L12	-0.3825	0.058	-6.574	0.000	-0.497	-0.268
ma.S.L12	-0.0199	0.061	-0.324	0.746	-0.140	0.100
sigma2	1.67e+05	9352.934	17.852	0.000	1.49e+05	1.85e+05

Ljung-Box (L1) (Q):	0.79	**Jarque-Bera (JB):**	1266.95
Prob(Q):	0.37	**Prob(JB):**	0.00
Heteroskedasticity (H):	33.12	**Skew:**	-0.34
Prob(H) (two-sided):	0.00	**Kurtosis:**	15.80

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).

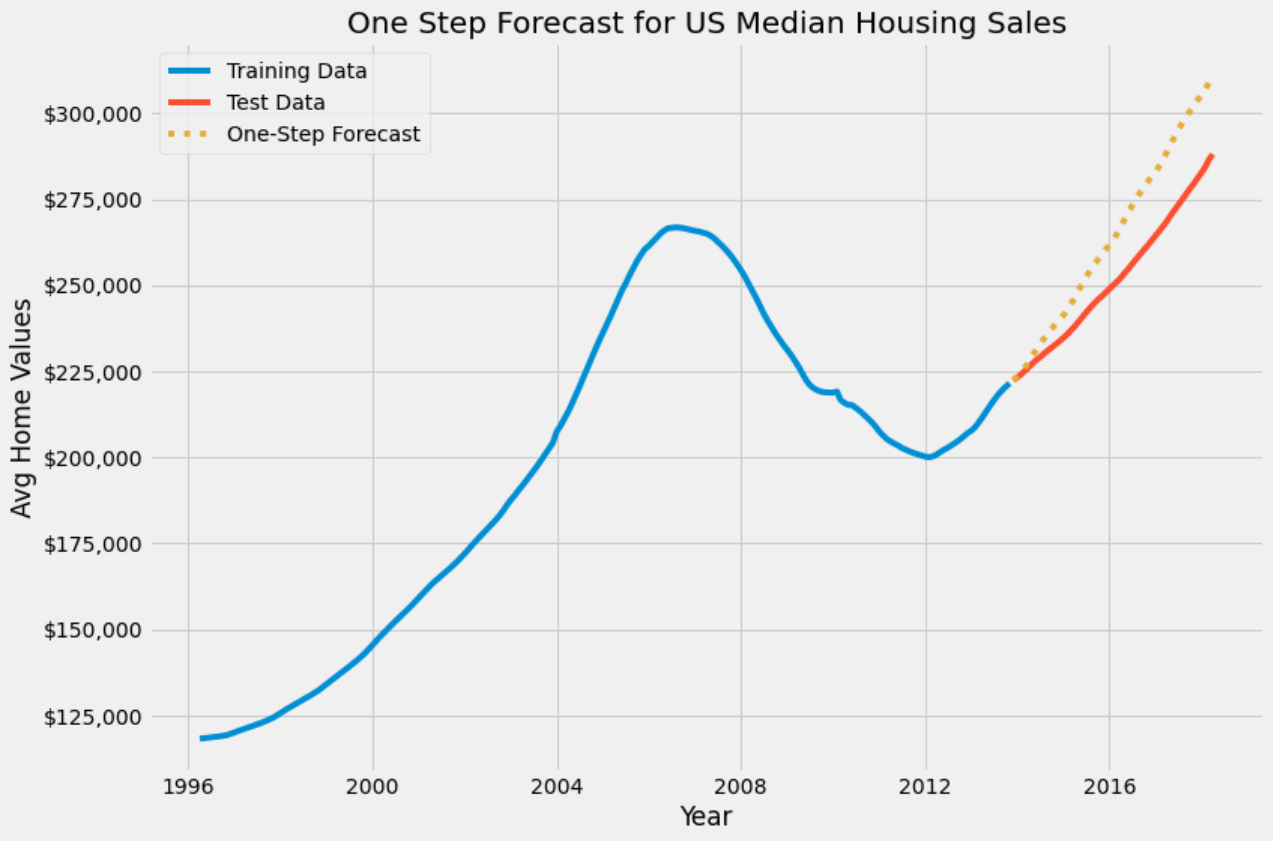
In [473]:
```
avg_home_model_1 = sarima_model
```

executed in 3ms, finished 14:25:03 2022-05-24

In [474]:

```
pred = avg_home_model_1.get_prediction(
    start=test.index.min(),
    end=test.index.max(),
    dynamic=True)

fig, ax = plt.subplots(figsize=(12, 8))
ax.plot(train, label='Training Data')
ax.plot(test, label='Test Data')
ax.plot(pred.predicted_mean, label='One-Step Forecast', ls=':')

ax.set_xlabel('Year')
ax.set_ylabel('Avg Home Values')
ax.set_title("One Step Forecast for US Median Housing Sales", fontsize = 20)
plt.savefig('images/us_mean_one_step_1')


ax.legend()
ax.yaxis.set_major_formatter(tick)
fig.tight_layout()
```

executed in 207ms, finished 14:25:03 2022-05-24



In [475]:
```
y_forecasted = pred.predicted_mean
y_truth = test['home_value']
mse = ((y_forecasted - y_truth) ** 2).mean()
print('The Mean Squared Error is {}'.format(round(mse, 2)))
print('The Root Mean Squared Error is {}'.format(round(np.sqrt(mse), 2)))
```

executed in 3ms, finished 14:25:04 2022-05-24

```
The Mean Squared Error is 225714435.77
The Root Mean Squared Error is 15023.8
```

Analyisis:

- This one-step forecast looks good and the RMSE is low. Still, I will attempt to refine the model and compare model 2 to this model to choose the best one.

5.1.2 Model #2

In [476]:
```
# using parameters from the advanced parameter search
# #advanced search combo: ((1, 2, 2), (2, 2, 2, 12)) AIC: 3112.12
combo = avg_home_df_combo_2
```

executed in 2ms, finished 14:25:04 2022-05-24

In [477]:
```
temp_ts = avg_home_df

SPLIT = int(temp_ts.shape[0]*0.8)
train, test = temp_ts[:SPLIT], temp_ts[SPLIT:]
```

executed in 2ms, finished 14:25:04 2022-05-24

In [478]:
```
# Model 2
sarima_model = SARIMAX(
    train,
    order=combo[0],
    seasonal_order=combo[1],
    enforce_stationarity=False,
    enforce_invertibility=False).fit()
sarima_model.summary()
```

executed in 7.26s, finished 14:25:11 2022-05-24

Out[478]:

SARIMAX Results

Dep. Variable:	home_value	**No. Observations:**	212
Model:	SARIMAX(1, 2, 2)x(2, 2, 2, 12)	**Log Likelihood**	-1182.301
Date:	Tue, 24 May 2022	**AIC**	2380.602
Time:	14:25:11	**BIC**	2405.153
Sample:	04-01-1996	**HQIC**	2390.572
	- 11-01-2013		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	-0.3946	0.112	-3.517	0.000	-0.615	-0.175
ma.L1	0.1510	0.113	1.333	0.183	-0.071	0.373
ma.L2	-0.0122	0.070	-0.175	0.861	-0.150	0.125
ar.S.L12	0.2368	0.071	3.353	0.001	0.098	0.375
ar.S.L24	-0.0501	0.045	-1.114	0.265	-0.138	0.038
ma.S.L12	-1.9123	0.531	-3.603	0.000	-2.953	-0.872
ma.S.L24	0.9653	0.502	1.923	0.054	-0.018	1.949
sigma2	1.196e+05	6.55e+04	1.825	0.068	-8865.671	2.48e+05

Ljung-Box (L1) (Q):	0.08	**Jarque-Bera (JB):**	1363.30
Prob(Q):	0.77	**Prob(JB):**	0.00
Heteroskedasticity (H):	3.24	**Skew:**	-1.13
Prob(H) (two-sided):	0.00	**Kurtosis:**	17.16

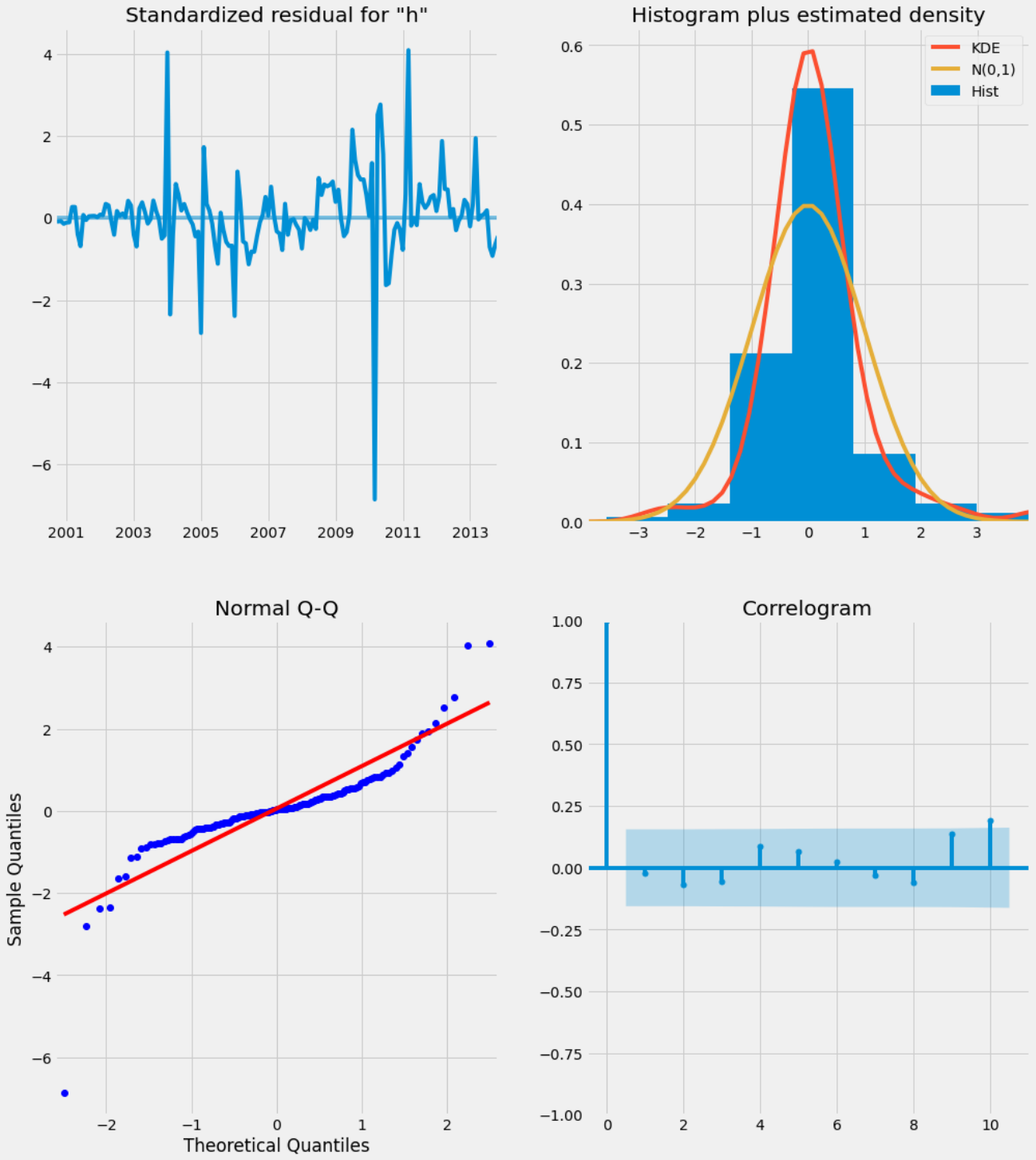
Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).

In [479]:
```
avg_home_model_2 = sarima_model
```

executed in 4ms, finished 14:25:11 2022-05-24

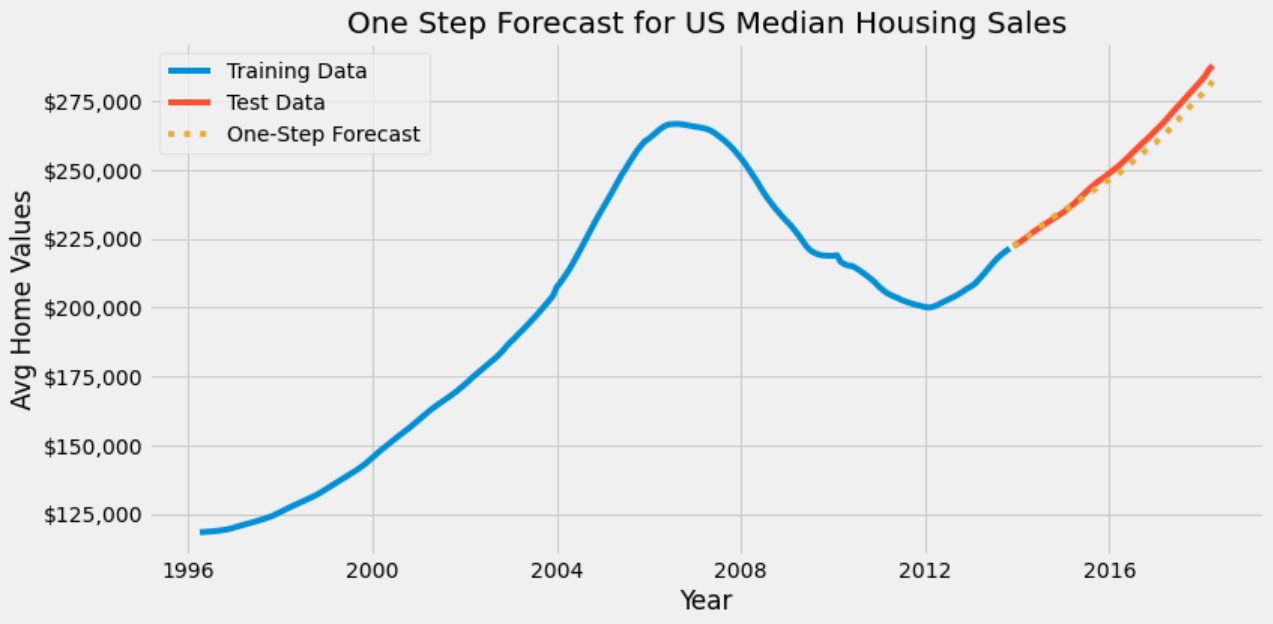
In [480]:
```
avg_home_model_2.plot_diagnostics(figsize=(15, 18))
plt.show()
```

executed in 364ms, finished 14:25:11 2022-05-24



In [481]:
```
pred = avg_home_model_2.get_prediction(
    start=test.index.min(),
    end=test.index.max(),
    dynamic=True)

fig, ax = plt.subplots(figsize=(12, 6))
ax.plot(train, label='Training Data')
ax.plot(test, label='Test Data')
ax.plot(pred.predicted_mean, label='One-Step Forecast', ls=':')

ax.set_xlabel('Year')
ax.set_ylabel('Avg Home Values')
ax.set_title("One Step Forecast for US Median Housing Sales", fontsize = 20)
plt.savefig('images/us_mean_one_step_2')

ax.legend()
ax.yaxis.set_major_formatter(tick)
fig.tight_layout()
```

executed in 199ms, finished 14:25:11 2022-05-24



In [482]:
```
y_forecasted = pred.predicted_mean
y_truth = test['home_value']
mse = ((y_forecasted - y_truth) ** 2).mean()
print('The Mean Squared Error is {}'.format(round(mse, 2)))
print('The Root Mean Squared Error is {}'.format(round(np.sqrt(mse), 2)))
```

executed in 4ms, finished 14:25:11 2022-05-24

```
The Mean Squared Error is 10304769.94
The Root Mean Squared Error is 3210.1
```

Analysis: This model performs so well that I definitely want to check for overfitting. However a look at the plot diagnostics shows that everything is good and that I can proceed with Model #2 for my Forecast.

5.1.3 Prediction

In [483]:
```
#combo = avg_home_df_combo_1
combo = avg_home_df_combo_2
```

executed in 3ms, finished 14:25:11 2022-05-24

In [484]:
```
# Running model again with FULL data

sarima_model = SARIMAX(
    avg_home_df,
    order=combo[0],
    seasonal_order=combo[1],
    enforce_stationarity=False,
    enforce_invertibility=False).fit()
sarima_model.summary()
```

executed in 7.36s, finished 14:25:19 2022-05-24

Out[484]:

SARIMAX Results

Dep. Variable:	home_value	**No. Observations:**	265
Model:	SARIMAX(1, 2, 2)x(2, 2, 2, 12)	**Log Likelihood**	-1548.062
Date:	Tue, 24 May 2022	**AIC**	3112.124
Time:	14:25:19	**BIC**	3138.977
Sample:	04-01-1996	**HQIC**	3122.977
	- 04-01-2018		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	-0.3522	0.099	-3.564	0.000	-0.546	-0.158
ma.L1	0.0909	0.101	0.896	0.370	-0.108	0.290
ma.L2	-0.0412	0.054	-0.767	0.443	-0.146	0.064
ar.S.L12	0.2163	0.046	4.698	0.000	0.126	0.306
ar.S.L24	-0.0567	0.028	-2.049	0.040	-0.111	-0.002
ma.S.L12	-1.8639	0.142	-13.113	0.000	-2.142	-1.585
ma.S.L24	0.8923	0.124	7.169	0.000	0.648	1.136
sigma2	1.021e+05	1.5e+04	6.805	0.000	7.27e+04	1.31e+05

Ljung-Box (L1) (Q):	0.34	**Jarque-Bera (JB):**	2384.79
Prob(Q):	0.56	**Prob(JB):**	0.00
Heteroskedasticity (H):	0.41	**Skew:**	-1.11
Prob(H) (two-sided):	0.00	**Kurtosis:**	19.28

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).

In [485]:
```
avg_home_model_full = sarima_model
```

executed in 4ms, finished 14:25:19 2022-05-24

In [486]:
```
avg_home_df['2018-04']
```

executed in 15ms, finished 14:25:19 2022-05-24

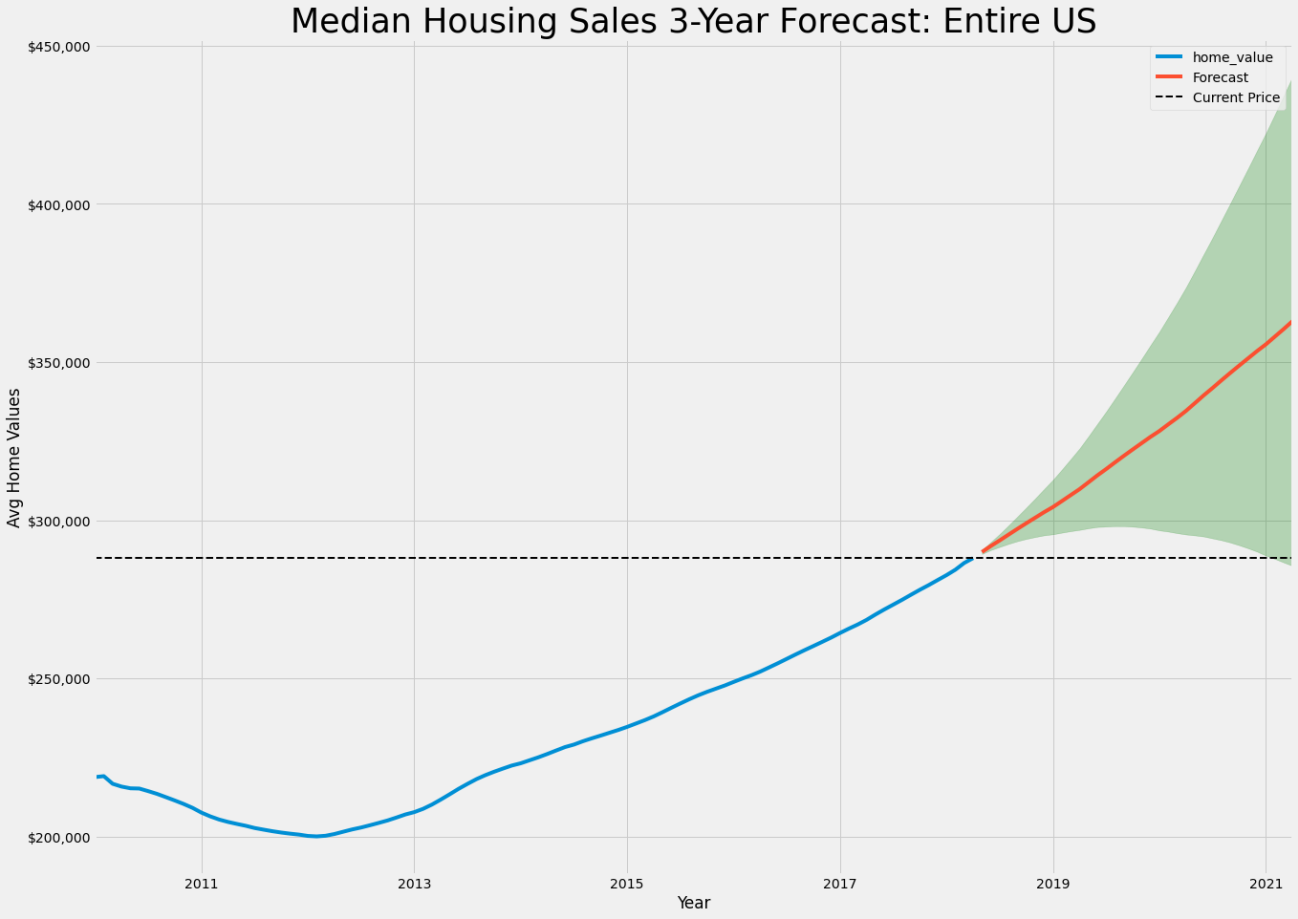
Out[486]:

time	home_value
2018-04-01	288,039.94

In [487]:

```
# Get forecast 2 yrs ahead in future (36 steps)
prediction = avg_home_model_full.get_forecast(steps=36)

# Get confidence intervals of forecasts
pred_conf = prediction.conf_int()

# Plot future predictions with confidence intervals
ax = avg_home_df['2010-01':].plot(label='observed', figsize=(20, 15))
prediction.predicted_mean.plot(ax=ax, label='Forecast')
ax.fill_between(pred_conf.index,
                pred_conf.iloc[:, 0],
                pred_conf.iloc[:, 1], color='green', alpha=0.25)

ax.axhline(288000, ls='--', color='black', linewidth=(2), label='Current Price')

ax.set_xlabel('Year')
ax.set_ylabel('Avg Home Values')
ax.set_title("Median Housing Sales 3-Year Forecast: Entire US", fontsize = 35)
ax.yaxis.set_major_formatter(tick)
plt.savefig('images/us_mean_forecast')

plt.legend()
plt.show()
```

executed in 305ms, finished 14:25:19 2022-05-24



5.1.4 Analysis

- The forecast is for avg home prices in the US to continue on trending up. The confidence interval range shows a high degree of confidence in this for the next 2 years, with the area widening for year 3. That said, even the lower confidence interval shows home values staying at the level that they currently are today.
- **It is safe to assume that average home prices will continue to trend upward.**

5.1.5 Metrics

- capturing metrics to compare with the top zip codes later.

In [488]:
```
analysis_df = pred_conf.resample('Y').mean()
analysis_df['mean'] = analysis_df.apply(lambda x: x.mean(), axis=1)

analysis_df['base'] = 288039
analysis_df['base'] = analysis_df['base'].astype(float)
```

executed in 8ms, finished 14:25:19 2022-05-24

In [489]:
```
analysis_df = analysis_df[['base', 'lower home_value', 'mean', 'upper home_value']]

analysis_df['Min_ROI'] = analysis_df.apply(lambda x: x['lower home_value'] - x['base'], axis=1)
analysis_df['Min_ROI%'] = analysis_df.apply(lambda x: x['Min_ROI'] / x['base'], axis=1)
analysis_df['Forecast_ROI'] = analysis_df.apply(lambda x: x['mean'] - x['base'], axis=1)
analysis_df['Forecast_ROI%'] = analysis_df.apply(lambda x: x['mean'] / x['base'], axis=1)
analysis_df['Max_ROI'] = analysis_df.apply(lambda x: x['upper home_value'] - x['base'], axis=1)
analysis_df['Max_ROI%'] = analysis_df.apply(lambda x: x['upper home_value'] / x['base'], axis=1)
```

executed in 9ms, finished 14:25:19 2022-05-24

In [490]:
```
avg_home_df_metrics = analysis_df
```

executed in 2ms, finished 14:25:19 2022-05-24

In [491]:
```
avg_home_df_metrics
```

executed in 6ms, finished 14:25:19 2022-05-24

Out[491]:

	base	lower home_value	mean	upper home_value	Min_ROI	Min_ROI%	Forecast_ROI	Forecast_ROI%	Max_ROI	Max_ROI%
2018-12-31	288,039.00	292,699.11	296,335.12	299,971.12	4,660.11	0.02	8,296.12	1.03	11,932.12	1.04
2019-12-31	288,039.00	297,332.59	315,184.63	333,036.66	9,293.59	0.03	27,145.63	1.09	44,997.66	1.16
2020-12-31	288,039.00	294,102.00	340,687.59	387,273.19	6,063.00	0.02	52,648.59	1.18	99,234.19	1.34
2021-12-31	288,039.00	287,258.32	359,239.52	431,220.72	-780.68	-0.00	71,200.52	1.25	143,181.72	1.50

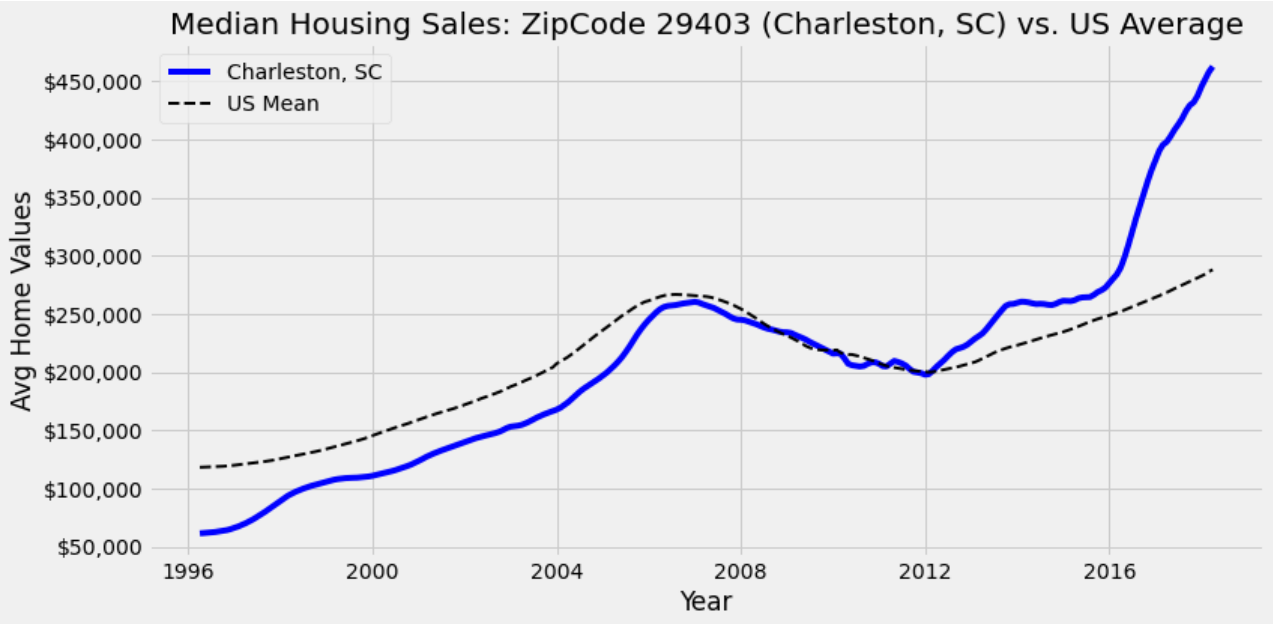
6 Begin Modeling Zip Codes

- I will go through the Zip Codes in order from my List of Target Zip Codes.
- Once I find a Zip Code that I recommed, I will exclude any future Zip Codes from that Zip Code's state in order to preserve that geographic diversity that has been requested by the Client.

7 29403: Charleston SC

In [492]:
```
fig, ax = plt.subplots(figsize=(12, 6))
ax.plot(zip_29403_ts, label='Charleston, SC', color='blue')
line1 = sns.lineplot(data=avg_home_df, x='time', y='home_value', ls='--',
                     color='black', linewidth=(2), label='US Mean');

ax.legend()
ax.set_xlabel('Year')
ax.set_ylabel('Avg Home Values')
ax.set_title("Median Housing Sales: ZipCode 29403 (Charleston, SC) vs. US Average", fontsize = 20)
ax.yaxis.set_major_formatter(tick)

fig.tight_layout()
```

executed in 176ms, finished 14:25:19 2022-05-24



7.1 Model #1

In [493]:
```
#sarimax_param_search(zip_29403_ts)
```

executed in 2ms, finished 14:25:19 2022-05-24

In [494]:
```
# Charlesston SC
# Normal Param Search Results
# AIC: 3900.908
zip_29403_combo_1 = ((1, 1, 1), (1, 1, 1, 12))
# DO NOT DELETE! Results from doing a more advanced parameter search
#zip_29403_combo_2
zip_29403_combo_2 = (((2, 2, 2), (0, 2, 2, 12)))
```

executed in 2ms, finished 14:25:19 2022-05-24

In [495]:
```
combo = zip_29403_combo_1
combo
```

executed in 3ms, finished 14:25:19 2022-05-24

Out[495]:
```
((1, 1, 1), (1, 1, 1, 12))
```

In [496]:
```
# Manually split data.
temp_ts = zip_29403_ts

SPLIT = int(temp_ts.shape[0]*0.80)
train, test = temp_ts[:SPLIT], temp_ts[SPLIT:]

# Model
sarima_model = SARIMAX(
    train,
    order=combo[0],
    seasonal_order=combo[1],
    enforce_stationarity=False,
    enforce_invertibility=False).fit()
sarima_model.summary()
```

executed in 557ms, finished 14:25:20 2022-05-24

Out[496]:

SARIMAX Results

Dep. Variable:	value	**No. Observations:**	212
Model:	SARIMAX(1, 1, 1)x(1, 1, 1, 12)	**Log Likelihood**	-1473.332
Date:	Tue, 24 May 2022	**AIC**	2956.665
Time:	14:25:20	**BIC**	2972.766
Sample:	04-01-1996	**HQIC**	2963.190
	- 11-01-2013		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	0.8600	0.040	21.403	0.000	0.781	0.939
ma.L1	0.6460	0.029	21.943	0.000	0.588	0.704
ar.S.L12	0.0872	0.043	2.013	0.044	0.002	0.172
ma.S.L12	-0.9253	0.058	-16.025	0.000	-1.038	-0.812
sigma2	3.573e+05	2.42e+04	14.770	0.000	3.1e+05	4.05e+05

Ljung-Box (L1) (Q):	0.22	**Jarque-Bera (JB):**	176.17
Prob(Q):	0.64	**Prob(JB):**	0.00
Heteroskedasticity (H):	7.53	**Skew:**	-0.19
Prob(H) (two-sided):	0.00	**Kurtosis:**	7.77

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).

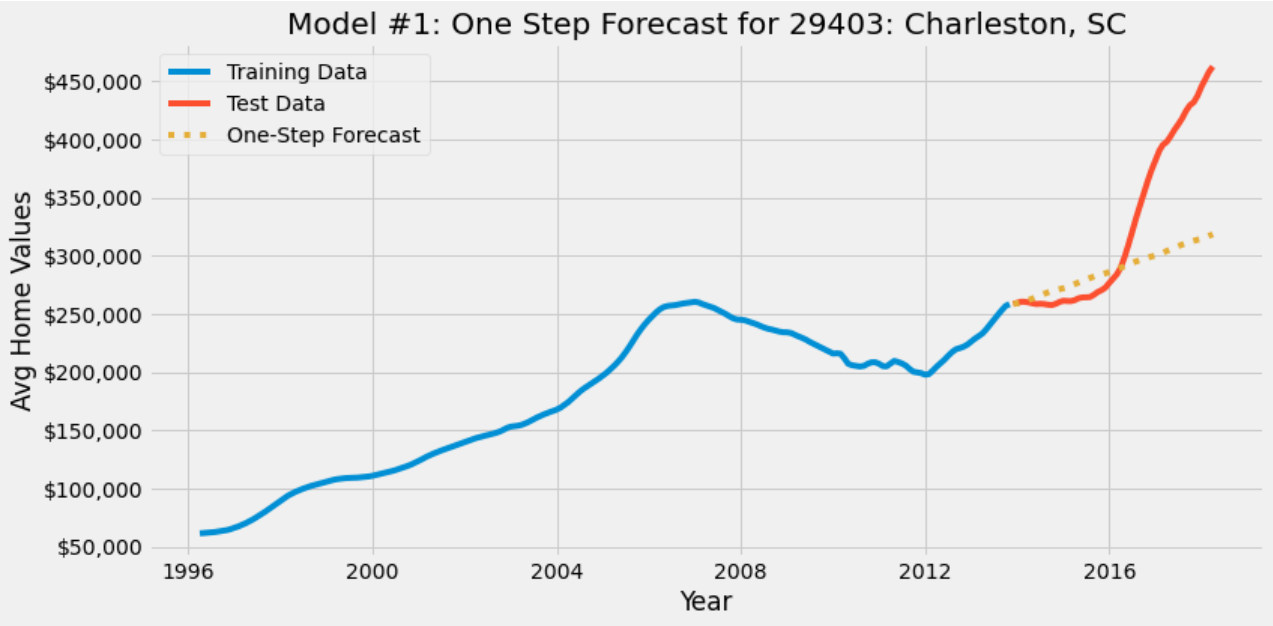
In [497]:
```
zip_29403_model_1 = sarima_model
```

executed in 3ms, finished 14:25:20 2022-05-24

In [498]:

```
pred = zip_29403_model_1.get_prediction(
    start=test.index.min(),
    end=test.index.max(),
    dynamic=False)

fig, ax = plt.subplots(figsize=(12, 6))
ax.plot(train, label='Training Data')
ax.plot(test, label='Test Data')
ax.plot(pred.predicted_mean, label='One-Step Forecast', ls=':')
#line1 = sns.lineplot(data=avg_home_df, x='time', y='home_value',
                     # color='black', linewidth=(2), ls='--', label='US Mean');
ax.set_xlabel('Year')
ax.set_ylabel('Avg Home Values')
ax.set_title("Model #1: One Step Forecast for 29403: Charleston, SC", fontsize = 20)
plt.savefig('images/charleston_one_step_1')

ax.legend()
ax.yaxis.set_major_formatter(tick)
fig.tight_layout()
```

executed in 202ms, finished 14:25:20 2022-05-24



In [499]:
```
y_forecasted = pred.predicted_mean
y_truth = test['value']
mse = ((y_forecasted - y_truth) ** 2).mean()
print('The Mean Squared Error is {}'.format(round(mse, 2)))
print('The Root Mean Squared Error is {}'.format(round(np.sqrt(mse), 2)))
```

executed in 4ms, finished 14:25:20 2022-05-24

```
The Mean Squared Error is 4294194403.4
The Root Mean Squared Error is 65530.1
```

7.2 Model #2

In [500]:
```
combo = zip_29403_combo_2
combo
```

executed in 3ms, finished 14:25:20 2022-05-24

Out[500]:
```
((2, 2, 2), (0, 2, 2, 12))
```

In [501]:
```
# Manually split data.
temp_ts = zip_29403_ts

SPLIT = int(temp_ts.shape[0]*0.80)
train, test = temp_ts[:SPLIT], temp_ts[SPLIT:]

# Model
sarima_model = SARIMAX(
    train,
    order=combo[0],
    seasonal_order=combo[1],
    enforce_stationarity=False,
    enforce_invertibility=False).fit()
sarima_model.summary()
```

executed in 5.94s, finished 14:25:26 2022-05-24

Out[501]:

SARIMAX Results

Dep. Variable:	value	**No. Observations:**	212
Model:	SARIMAX(2, 2, 2)x(0, 2, 2, 12)	**Log Likelihood**	-1306.948
Date:	Tue, 24 May 2022	**AIC**	2627.896
Time:	14:25:26	**BIC**	2649.378
Sample:	04-01-1996	**HQIC**	2636.619
	- 11-01-2013		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	-0.6437	0.055	-11.677	0.000	-0.752	-0.536
ar.L2	-0.3261	0.034	-9.693	0.000	-0.392	-0.260
ma.L1	1.3790	0.041	33.312	0.000	1.298	1.460
ma.L2	0.7858	0.040	19.733	0.000	0.708	0.864
ma.S.L12	-1.5538	0.045	-34.795	0.000	-1.641	-1.466
ma.S.L24	0.6953	0.045	15.568	0.000	0.608	0.783
sigma2	5.26e+05	3.94e+04	13.360	0.000	4.49e+05	6.03e+05

Ljung-Box (L1) (Q):	0.28	**Jarque-Bera (JB):**	24.41
Prob(Q):	0.60	**Prob(JB):**	0.00
Heteroskedasticity (H):	3.77	**Skew:**	-0.06
Prob(H) (two-sided):	0.00	**Kurtosis:**	4.92

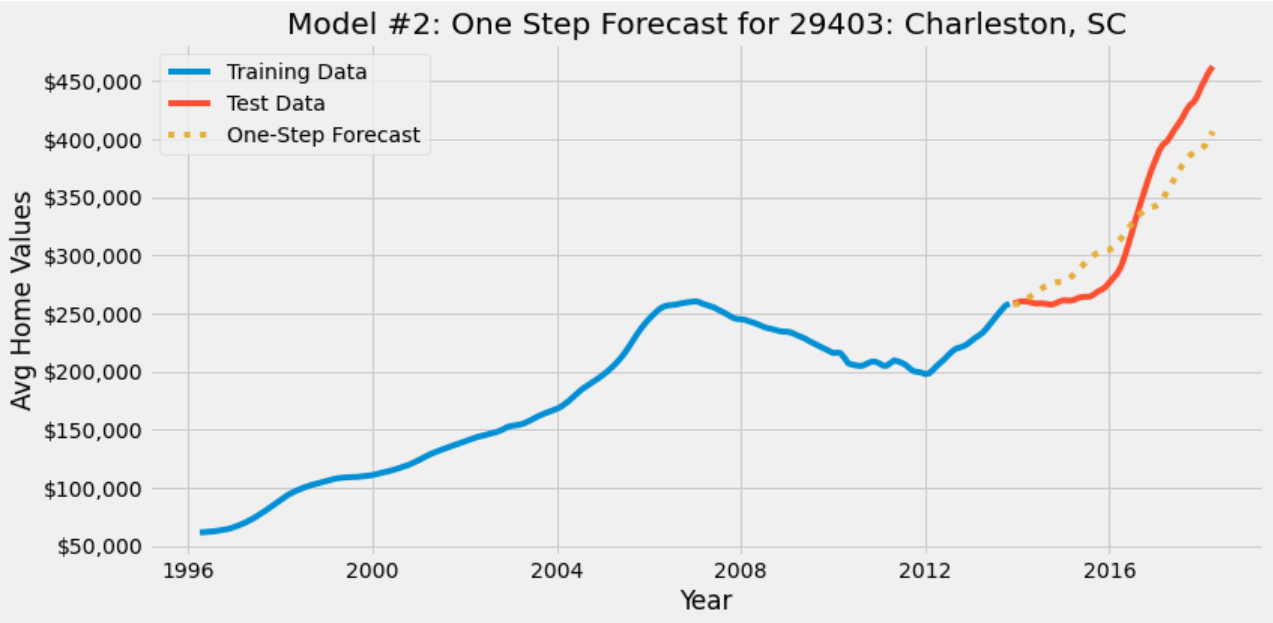
Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).

In [502]:
```
zip_29403_model_2 = sarima_model
```

executed in 2ms, finished 14:25:26 2022-05-24

In [503]:
```
pred = zip_29403_model_2.get_prediction(
    start=test.index.min(),
    end=test.index.max(),
    dynamic=False)

fig, ax = plt.subplots(figsize=(12, 6))
ax.plot(train, label='Training Data')
ax.plot(test, label='Test Data')
ax.plot(pred.predicted_mean, label='One-Step Forecast', ls=':')


ax.set_xlabel('Year')
ax.set_ylabel('Avg Home Values')
ax.set_title("Model #2: One Step Forecast for 29403: Charleston, SC", fontsize = 20)
ax.legend()
plt.savefig('images/charleston_one_step_2')
ax.yaxis.set_major_formatter(tick)
fig.tight_layout()

```

executed in 226ms, finished 14:25:26 2022-05-24



In [504]:
```
y_forecasted = pred.predicted_mean
y_truth = test['value']
mse = ((y_forecasted - y_truth) ** 2).mean()
print('The Mean Squared Error is {}'.format(round(mse, 2)))
print('The Root Mean Squared Error is {}'.format(round(np.sqrt(mse), 2)))
```

executed in 3ms, finished 14:25:26 2022-05-24

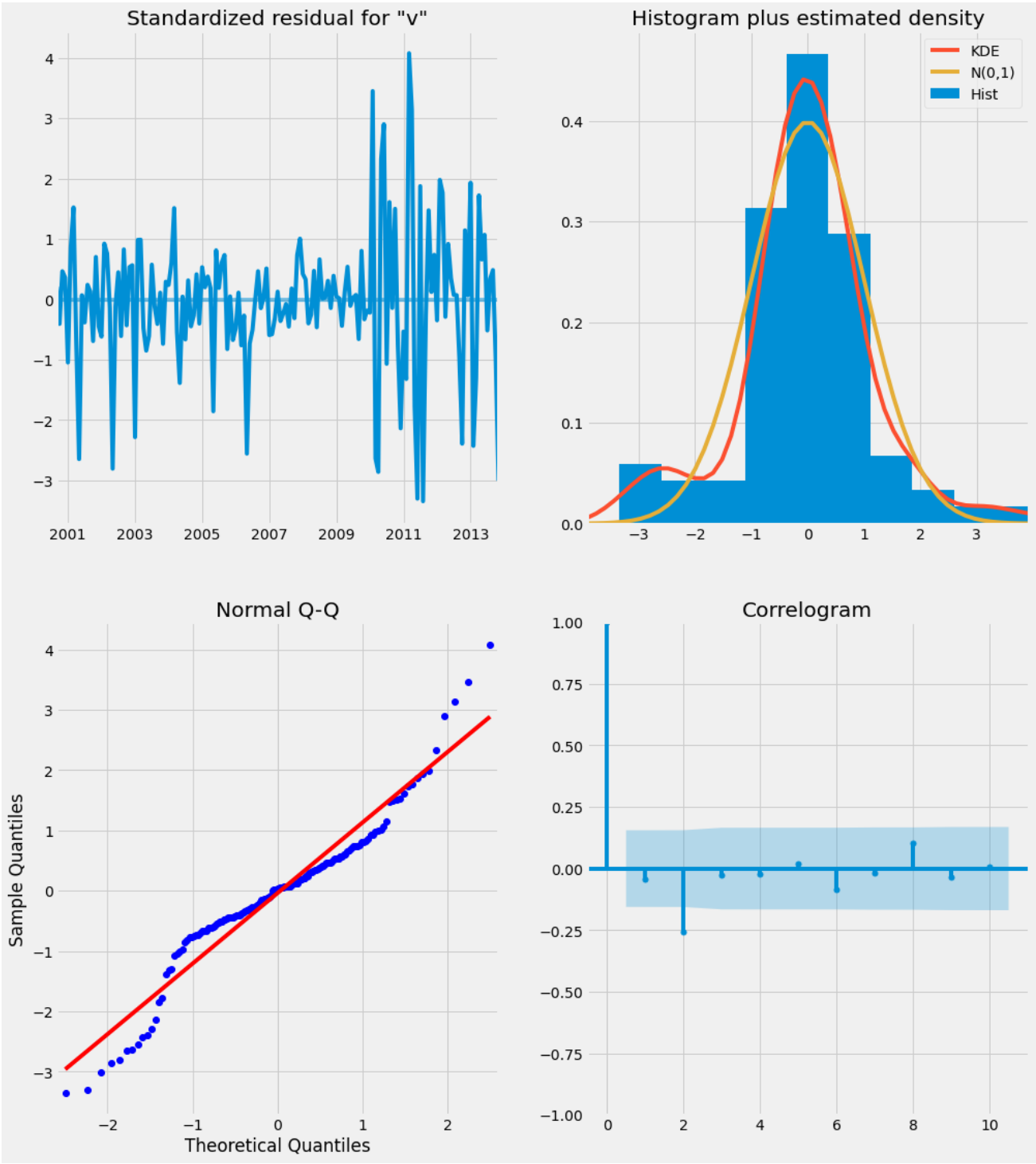
```
The Mean Squared Error is 964280497.2
The Root Mean Squared Error is 31052.87
```

7.3 Model Comparison & Decision:

- Model 2's forecast is a better match for the test data. I will use this to make my predictions.

In [505]:
```
zip_29403_model_2.plot_diagnostics(figsize=(15, 18))
plt.show()
```

executed in 371ms, finished 14:25:27 2022-05-24



- Diagnostics all look good.

7.4 Predictions

In [506]:
```
combo = zip_29403_combo_2
combo
```

executed in 2ms, finished 14:25:27 2022-05-24

Out[506]:
```
((2, 2, 2), (0, 2, 2, 12))
```

In [507]:
```
# Running model again with FULL data

sarima_model = SARIMAX(
    zip_29403_ts,
    order=combo[0],
    seasonal_order=combo[1],
    enforce_stationarity=False,
    enforce_invertibility=False).fit()
sarima_model.summary()
```

executed in 6.48s, finished 14:25:33 2022-05-24

Out[507]:

SARIMAX Results

Dep. Variable:	value	**No. Observations:**	265
Model:	SARIMAX(2, 2, 2)x(0, 2, 2, 12)	**Log Likelihood**	-1760.868
Date:	Tue, 24 May 2022	**AIC**	3535.736
Time:	14:25:33	**BIC**	3559.232
Sample:	04-01-1996	**HQIC**	3545.233
	- 04-01-2018		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	-0.8197	0.069	-11.797	0.000	-0.956	-0.683
ar.L2	-0.3762	0.038	-9.926	0.000	-0.450	-0.302
ma.L1	1.6234	0.051	32.059	0.000	1.524	1.723
ma.L2	0.8922	0.049	18.077	0.000	0.795	0.989
ma.S.L12	-1.6219	0.141	-11.528	0.000	-1.898	-1.346
ma.S.L24	0.6394	0.091	7.011	0.000	0.461	0.818
sigma2	7.392e+05	9.68e+04	7.634	0.000	5.49e+05	9.29e+05

Ljung-Box (L1) (Q):	2.60	**Jarque-Bera (JB):**	62.24
Prob(Q):	0.11	**Prob(JB):**	0.00
Heteroskedasticity (H):	2.62	**Skew:**	0.03
Prob(H) (two-sided):	0.00	**Kurtosis:**	5.65

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).

In [508]:
```
zip_29403_model_full = sarima_model
```

executed in 3ms, finished 14:25:33 2022-05-24

In [509]:
```
1 zip_29403_ts.tail()
```

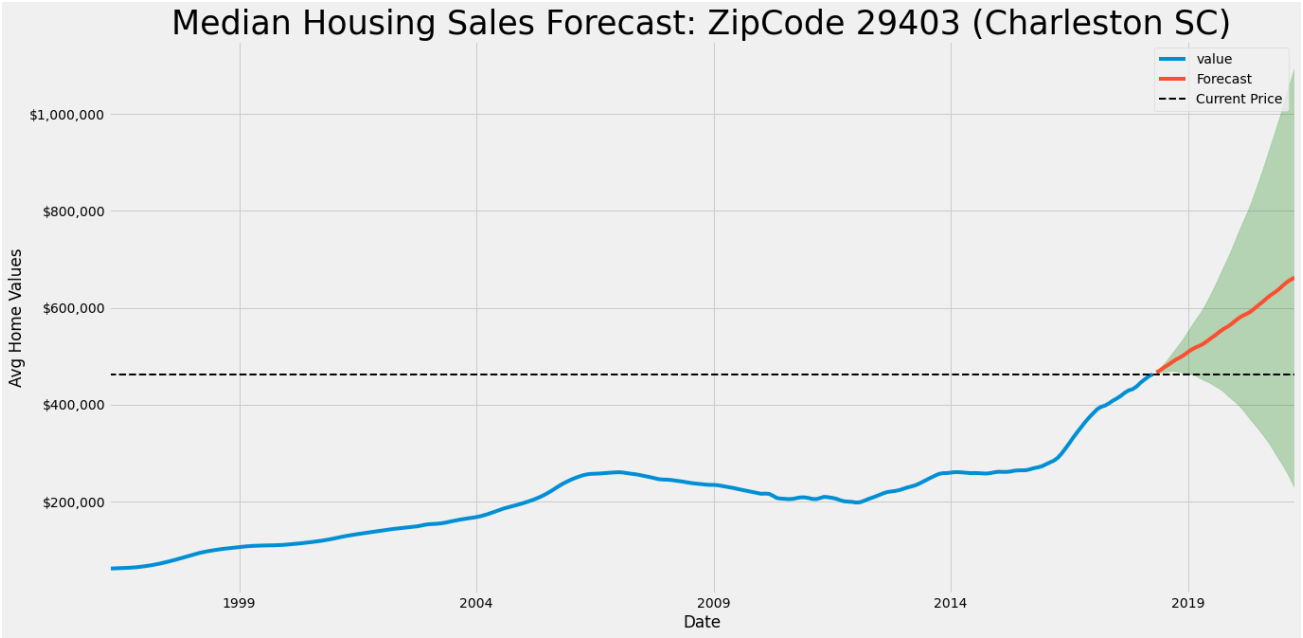
executed in 5ms, finished 14:25:33 2022-05-24

Out[509]:

time	value
2017-12-01	437,500.00
2018-01-01	445,400.00
2018-02-01	451,900.00
2018-03-01	458,000.00
2018-04-01	462,700.00

In [510]:
```
1  # Get forecast 3 yrs ahead in future (36 steps)
2  prediction = zip_29403_model_full.get_forecast(steps=36)
3
4  # Get confidence intervals of forecasts
5  pred_conf = prediction.conf_int()
6
7  # Plot future predictions with confidence intervals
8  ax = zip_29403_ts['1996-01':].plot(label='observed', figsize=(20, 10))
9  prediction.predicted_mean.plot(ax=ax, label='Forecast')
10 ax.fill_between(pred_conf.index,
11                 pred_conf.iloc[:, 0],
12                 pred_conf.iloc[:, 1], color='green', alpha=0.25)
13
14 ax.axhline(462700, ls='--', color='black', linewidth=(2), label='Current Price')
15
16 ax.set_xlabel('Date')
17 ax.set_ylabel('Avg Home Values')
18 ax.set_title("Median Housing Sales Forecast: ZipCode 29403 (Charleston SC)", fontsize = 35)
19 ax.yaxis.set_major_formatter(tick)
20 plt.savefig('images/charleston_forecast')
21
22 plt.legend()
23 plt.show()
```

executed in 280ms, finished 14:25:33 2022-05-24



7.5 Final Analysis:

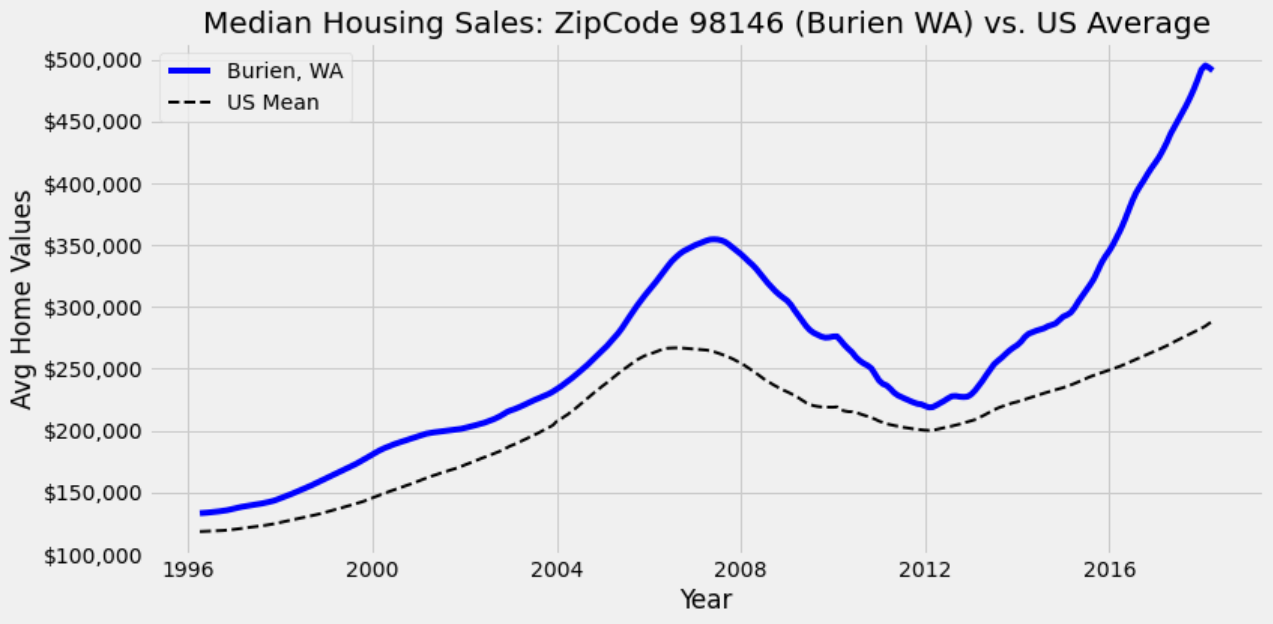
- While 1-2 Years out looks good, there is too much uncertainty in Forecast Year 3 to recommend Charleston as one of the Top 5 Zip Codes.
- That said, if you reduced the investment window to 1-2 years, Charleston is a good choice and therefore should be considered with that caveat.

- **I will recommend this Zip Code as an alternative recommendation with the caveat that it should be re-evaulated during the 1-2 year range.**

8 98146: Burien WA (#1)

In [511]:
```
fig, ax = plt.subplots(figsize=(12, 6))
ax.plot(zip_98146_ts, label='Burien, WA', color='blue')
line1 = sns.lineplot(data=avg_home_df, x='time', y='home_value', ls='--',
                     color='black', linewidth=(2), label='US Mean');

ax.legend()
ax.set_title("Median Housing Sales: ZipCode 98146 (Burien WA) vs. US Average", fontsize = 20)
ax.set_xlabel('Year')
ax.set_ylabel('Avg Home Values')
ax.yaxis.set_major_formatter(tick)
fig.tight_layout()
```

executed in 179ms, finished 14:25:34 2022-05-24



In [512]:
```
#zip_98146_combo_1 = ((1, 1, 1), (1, 1, 1, 12)) AIC: 3862
zip_98146_combo_1 = ((1, 1, 1), (1, 1, 1, 12))
```

executed in 2ms, finished 14:25:34 2022-05-24

In [513]:
```
# SARIMA Combos: ((2, 2, 2), (2, 2, 2, 12)) AIC: 3528.4653033814084
zip_98146_combo_2 = ((2, 2, 2), (2, 2, 2, 12))
zip_98146_combo_2
```

executed in 3ms, finished 14:25:34 2022-05-24

Out[513]:
```
((2, 2, 2), (2, 2, 2, 12))
```

8.1 Model #1

In [514]:
```
combo = zip_98146_combo_1
combo
```

executed in 2ms, finished 14:25:34 2022-05-24

Out[514]:
```
((1, 1, 1), (1, 1, 1, 12))
```

In [515]:
```
# Manually split data.
temp_ts = zip_98146_ts

SPLIT = int(temp_ts.shape[0]*0.80)
train, test = temp_ts[:SPLIT], temp_ts[SPLIT:]

# Model
sarima_model = SARIMAX(
    train,
    order=combo[0],
    seasonal_order=combo[1],
    enforce_stationarity=False,
    enforce_invertibility=False).fit()
sarima_model.summary()
```

executed in 630ms, finished 14:25:34 2022-05-24

Out[515]:

SARIMAX Results

Dep. Variable:	value	**No. Observations:**	212
Model:	SARIMAX(1, 1, 1)x(1, 1, 1, 12)	**Log Likelihood**	-1446.596
Date:	Tue, 24 May 2022	**AIC**	2903.193
Time:	14:25:34	**BIC**	2919.294
Sample:	04-01-1996	**HQIC**	2909.718
	- 11-01-2013		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	0.9574	0.021	45.343	0.000	0.916	0.999
ma.L1	0.3574	0.049	7.227	0.000	0.260	0.454
ar.S.L12	0.1369	0.023	6.077	0.000	0.093	0.181
ma.S.L12	-0.9242	0.060	-15.362	0.000	-1.042	-0.806
sigma2	2.661e+05	1.2e+04	22.231	0.000	2.43e+05	2.9e+05

Ljung-Box (L1) (Q):	4.05	**Jarque-Bera (JB):**	446.91
Prob(Q):	0.04	**Prob(JB):**	0.00
Heteroskedasticity (H):	9.63	**Skew:**	-0.96
Prob(H) (two-sided):	0.00	**Kurtosis:**	10.37

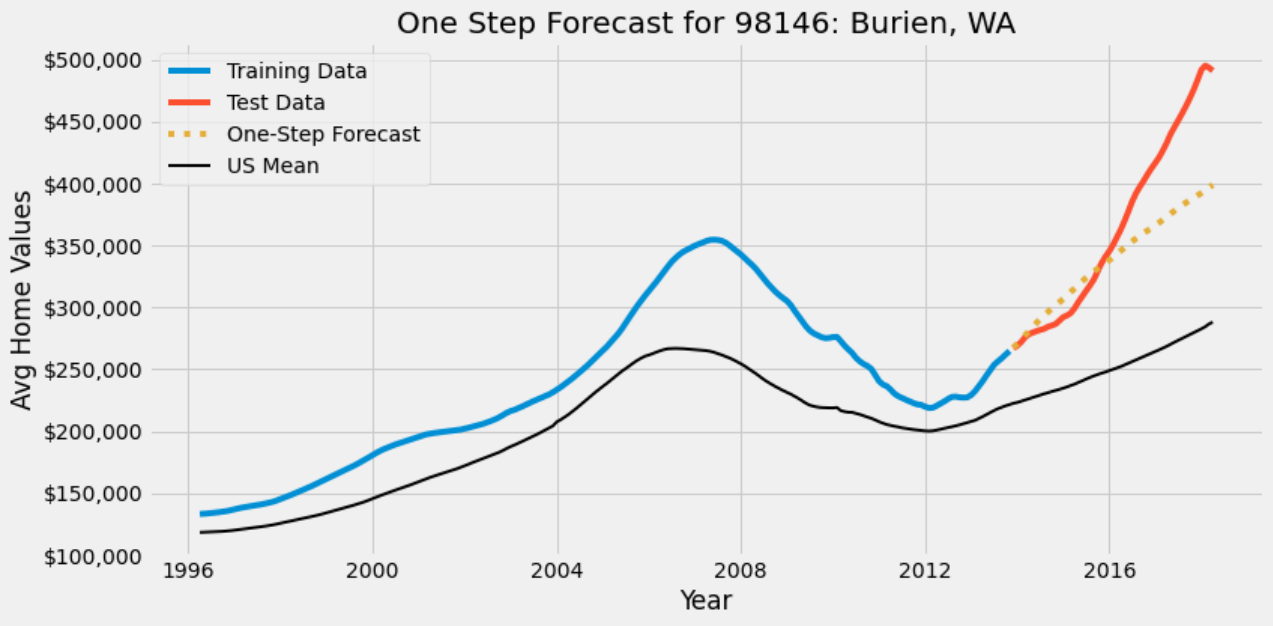
Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).

In [516]:
```
zip_98146_model_1 = sarima_model
```

executed in 2ms, finished 14:25:34 2022-05-24

In [517]:
```
pred = zip_98146_model_1.get_prediction(
    start=test.index.min(),
    end=test.index.max(),
    dynamic=False)

fig, ax = plt.subplots(figsize=(12, 6))
ax.plot(train, label='Training Data')
ax.plot(test, label='Test Data')
ax.plot(pred.predicted_mean, label='One-Step Forecast', ls=':')
line1 = sns.lineplot(data=avg_home_df, x='time', y='home_value',
                     color='black', linewidth=(2), label='US Mean');

ax.set_xlabel('Year')
ax.set_ylabel('Avg Home Values')
ax.set_title("One Step Forecast for 98146: Burien, WA", fontsize = 20)
ax.legend()
plt.savefig('images/burien1_one_step_1')
ax.yaxis.set_major_formatter(tick)
fig.tight_layout()
```

executed in 276ms, finished 14:25:34 2022-05-24



In [518]:
```
y_forecasted = pred.predicted_mean
y_truth = test['value']
mse = ((y_forecasted - y_truth) ** 2).mean()
print('The Mean Squared Error is {}'.format(round(mse, 2)))
print('The Root Mean Squared Error is {}'.format(round(np.sqrt(mse), 2)))
```

executed in 3ms, finished 14:25:34 2022-05-24

```
The Mean Squared Error is 2132788317.74
The Root Mean Squared Error is 46182.12
```

8.2 Model #2

In [519]:
```
combo = zip_98146_combo_2
combo
```

executed in 3ms, finished 14:25:34 2022-05-24

Out[519]: ((2, 2, 2), (2, 2, 2, 12))

In [520]:
```
# Manually split data.
temp_ts = zip_98146_ts

SPLIT = int(temp_ts.shape[0]*0.80)
train, test = temp_ts[:SPLIT], temp_ts[SPLIT:]

# Model
sarima_model = SARIMAX(
    train,
    order=combo[0],
    seasonal_order=combo[1],
    enforce_stationarity=False,
    enforce_invertibility=False).fit()
sarima_model.summary()
```

executed in 7.57s, finished 14:25:42 2022-05-24

Out[520]:

SARIMAX Results

Dep. Variable:	value	**No. Observations:**	212
Model:	SARIMAX(2, 2, 2)x(2, 2, 2, 12)	**Log Likelihood**	-1294.159
Date:	Tue, 24 May 2022	**AIC**	2606.318
Time:	14:25:42	**BIC**	2633.938
Sample:	04-01-1996	**HQIC**	2617.534
	- 11-01-2013		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	1.0489	0.122	8.631	0.000	0.811	1.287
ar.L2	-0.0772	0.113	-0.684	0.494	-0.298	0.144
ma.L1	-0.5791	0.493	-1.175	0.240	-1.545	0.387
ma.L2	-0.4186	0.251	-1.670	0.095	-0.910	0.073
ar.S.L12	-1.2102	0.077	-15.762	0.000	-1.361	-1.060
ar.S.L24	-0.8774	0.119	-7.349	0.000	-1.111	-0.643
ma.S.L12	-0.0233	0.067	-0.348	0.728	-0.154	0.108
ma.S.L24	-0.0289	0.046	-0.623	0.534	-0.120	0.062
sigma2	6.838e+05	3.21e+05	2.130	0.033	5.45e+04	1.31e+06

Ljung-Box (L1) (Q):	0.00	**Jarque-Bera (JB):**	76.40
Prob(Q):	0.98	**Prob(JB):**	0.00
Heteroskedasticity (H):	28.50	**Skew:**	0.11
Prob(H) (two-sided):	0.00	**Kurtosis:**	6.39

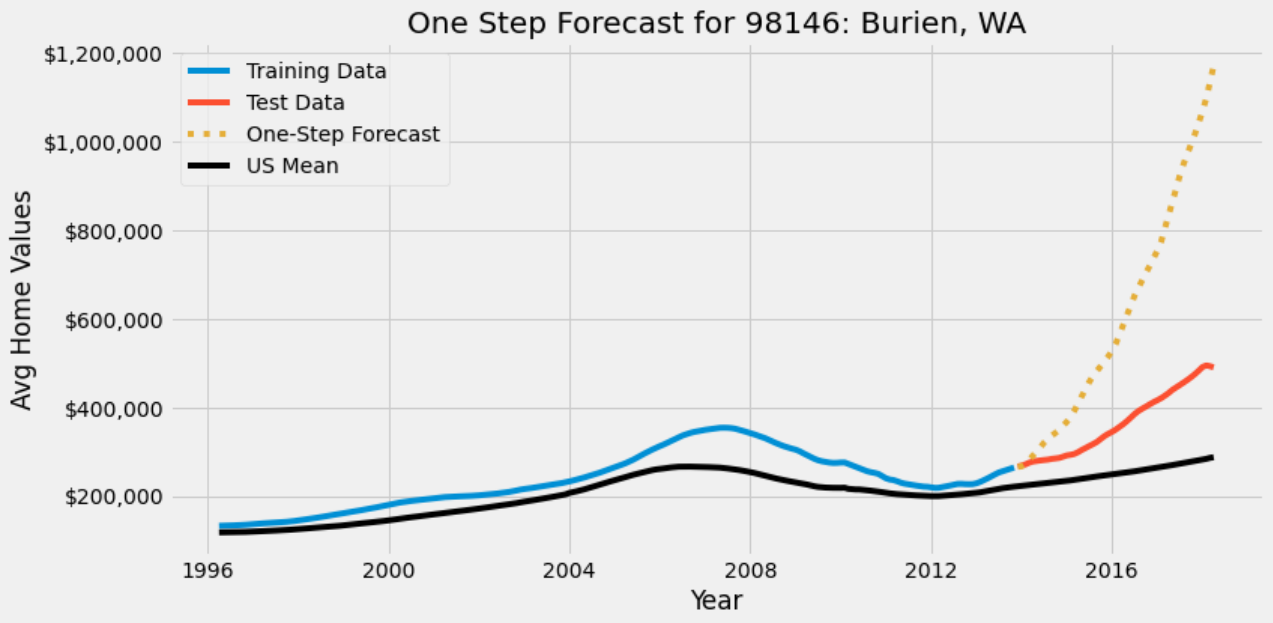
Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).

In [521]:
```
zip_98146_model_2 = sarima_model
```

executed in 2ms, finished 14:25:42 2022-05-24

In [522]:
```
pred = zip_98146_model_2.get_prediction(
    start=test.index.min(),
    end=test.index.max(),
    dynamic=True)

fig, ax = plt.subplots(figsize=(12, 6))
ax.plot(train, label='Training Data')
ax.plot(test, label='Test Data')
ax.plot(pred.predicted_mean, label='One-Step Forecast', ls=':')
line1 = sns.lineplot(data=avg_home_df, x='time', y='home_value', color='black', label='US Mean');

ax.set_xlabel('Year')
ax.set_ylabel('Avg Home Values')
ax.set_title("One Step Forecast for 98146: Burien, WA", fontsize = 20)
ax.legend()
ax.yaxis.set_major_formatter(tick)
plt.savefig('images/burien1_one_step_1')
fig.tight_layout()
```

executed in 254ms, finished 14:25:42 2022-05-24



In [523]:
```
y_forecasted = pred.predicted_mean
y_truth = test['value']
mse = ((y_forecasted - y_truth) ** 2).mean()
print('The Mean Squared Error is {}'.format(round(mse, 2)))
print('The Root Mean Squared Error is {}'.format(round(np.sqrt(mse), 2)))
```

executed in 4ms, finished 14:25:42 2022-05-24

```
The Mean Squared Error is 97460870085.47
The Root Mean Squared Error is 312187.24
```

8.3 Model Comparison & Decision:

- Model 1 has a significantly lower RSME, and is a much better match for the test data. I will proceed using Model 1.

8.4 Prediction

In [524]:
```
combo = zip_98146_combo_1
#combo = zip_98146_combo_2
combo
```

executed in 3ms, finished 14:25:42 2022-05-24

Out[524]: ((1, 1, 1), (1, 1, 1, 12))

In [525]:
```
# Running model again with FULL data

sarima_model = SARIMAX(
    zip_98146_ts,
    order=combo[0],
    seasonal_order=combo[1],
    enforce_stationarity=False,
    enforce_invertibility=False).fit()
sarima_model.summary()
```

executed in 779ms, finished 14:25:43 2022-05-24

Out[525]:

SARIMAX Results

Dep. Variable:	value	**No. Observations:**	265
Model:	SARIMAX(1, 1, 1)x(1, 1, 1, 12)	**Log Likelihood**	-1926.358
Date:	Tue, 24 May 2022	**AIC**	3862.715
Time:	14:25:43	**BIC**	3880.077
Sample:	04-01-1996	**HQIC**	3869.712
	- 04-01-2018		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	0.9071	0.032	28.389	0.000	0.844	0.970
ma.L1	0.5799	0.034	16.888	0.000	0.513	0.647
ar.S.L12	0.1196	0.060	1.980	0.048	0.001	0.238
ma.S.L12	-0.6561	0.065	-10.137	0.000	-0.783	-0.529
sigma2	5.325e+05	2.47e+04	21.542	0.000	4.84e+05	5.81e+05

Ljung-Box (L1) (Q):	1.59	**Jarque-Bera (JB):**	1013.40
Prob(Q):	0.21	**Prob(JB):**	0.00
Heteroskedasticity (H):	7.37	**Skew:**	-1.58
Prob(H) (two-sided):	0.00	**Kurtosis:**	12.60

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).

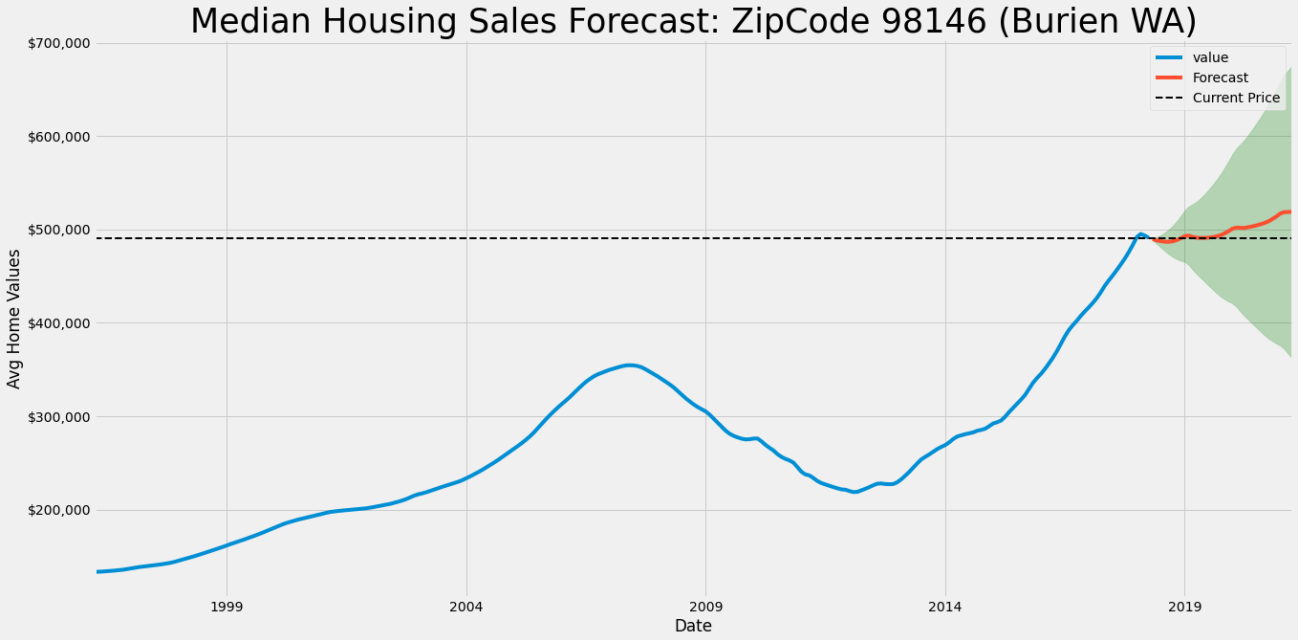
In [526]:
```
zip_98146_model_full = sarima_model
```

executed in 3ms, finished 14:25:43 2022-05-24

In [527]:

```
# Get forecast 3 yrs ahead in future (36 steps)
prediction = zip_98146_model_full.get_forecast(steps=36)

# Get confidence intervals of forecasts
pred_conf = prediction.conf_int()

# Plot future predictions with confidence intervals
ax = zip_98146_ts['1996-01':].plot(label='observed', figsize=(20, 10))
prediction.predicted_mean.plot(ax=ax, label='Forecast')
ax.fill_between(pred_conf.index,
                pred_conf.iloc[:, 0],
                pred_conf.iloc[:, 1], color='green', alpha=0.25)

ax.axhline(491000, ls='--', color='black', linewidth=(2), label='Current Price')

ax.set_xlabel('Date')
ax.set_ylabel('Avg Home Values')
ax.set_title("Median Housing Sales Forecast: ZipCode 98146 (Burien WA)", fontsize = 35)
ax.yaxis.set_major_formatter(tick)
plt.savefig('images/burien1_forecast')

plt.legend()
plt.show()
```

executed in 282ms, finished 14:25:43 2022-05-24



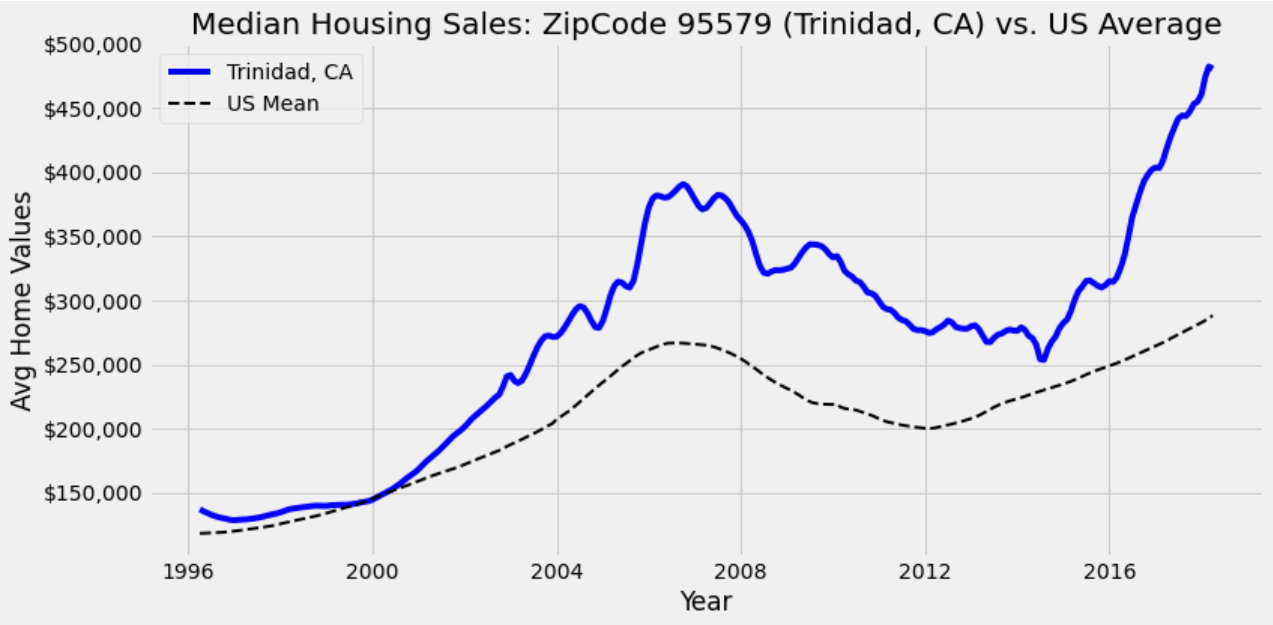
8.5 Analysis:

- There is too much uncertainty in the confidence window. The forecast mean stays around the current value and the upper and lower confidence windows are nearly equally sized, so **investing in this zip code would be more of a 50/50 chance than a low-risk investment. Therefore I do not recommend pursing this zip code.**

9 95570: Trinidad CA

In [528]:
```
fig, ax = plt.subplots(figsize=(12, 6))
ax.plot(zip_95570_ts, label='Trinidad, CA', color='blue')
line1 = sns.lineplot(data=avg_home_df, x='time', y='home_value', ls='--',
                     color='black', linewidth=(2), label='US Mean');

ax.legend()
ax.set_title("Median Housing Sales: ZipCode 95579 (Trinidad, CA) vs. US Average", fontsize = 20)
ax.set_xlabel('Year')
ax.set_ylabel('Avg Home Values')
ax.yaxis.set_major_formatter(tick)
fig.tight_layout()
```

executed in 174ms, finished 14:25:44 2022-05-24



In [529]:
```
#sarimax_param_search(zip_95570_ts)
```

executed in 2ms, finished 14:25:44 2022-05-24

In [530]:
```
#SARIMA Combos: ((1, 1, 1), (1, 1, 1, 12)) AIC: 4470.940175311183
zip_95570_combo_1 = ((1, 1, 1), (1, 1, 1, 12))
zip_95570_combo_1
```

executed in 3ms, finished 14:25:44 2022-05-24

Out[530]:
```
((1, 1, 1), (1, 1, 1, 12))
```

In [531]:
```
#SARIMA Combos: ((2, 2, 2), (2, 2, 2, 12)) AIC: 4087.066104443231
zip_95570_combo_2 = ((2, 2, 2), (2, 2, 2, 12))
zip_95570_combo_2

```

executed in 3ms, finished 14:25:44 2022-05-24

Out[531]:
```
((2, 2, 2), (2, 2, 2, 12))
```

9.1 Model #1

In [532]:
```
combo = zip_95570_combo_1
combo
```

executed in 2ms, finished 14:25:44 2022-05-24

Out[532]:
```
((1, 1, 1), (1, 1, 1, 12))
```

In [533]:
```
# Manually split data.
temp_ts = zip_95570_ts

SPLIT = int(temp_ts.shape[0]*0.80)
train, test = temp_ts[:SPLIT], temp_ts[SPLIT:]

# Model
sarima_model = SARIMAX(
    train,
    order=combo[0],
    seasonal_order=combo[1],
    enforce_stationarity=False,
    enforce_invertibility=False).fit()
sarima_model.summary()
```

executed in 414ms, finished 14:25:44 2022-05-24

Out[533]:

SARIMAX Results

Dep. Variable:	value	**No. Observations:**	212
Model:	SARIMAX(1, 1, 1)x(1, 1, 1, 12)	**Log Likelihood**	-1690.132
Date:	Tue, 24 May 2022	**AIC**	3390.264
Time:	14:25:44	**BIC**	3406.365
Sample:	04-01-1996	**HQIC**	3396.789
	- 11-01-2013		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	0.7030	0.070	10.100	0.000	0.567	0.839
ma.L1	0.8011	0.075	10.722	0.000	0.655	0.948
ar.S.L12	0.2347	0.059	3.967	0.000	0.119	0.351
ma.S.L12	-0.4682	0.057	-8.162	0.000	-0.581	-0.356
sigma2	6.665e+06	7.98e+05	8.356	0.000	5.1e+06	8.23e+06

Ljung-Box (L1) (Q):	5.42	**Jarque-Bera (JB):**	16.58
Prob(Q):	0.02	**Prob(JB):**	0.00
Heteroskedasticity (H):	1.94	**Skew:**	-0.13
Prob(H) (two-sided):	0.01	**Kurtosis:**	4.44

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).

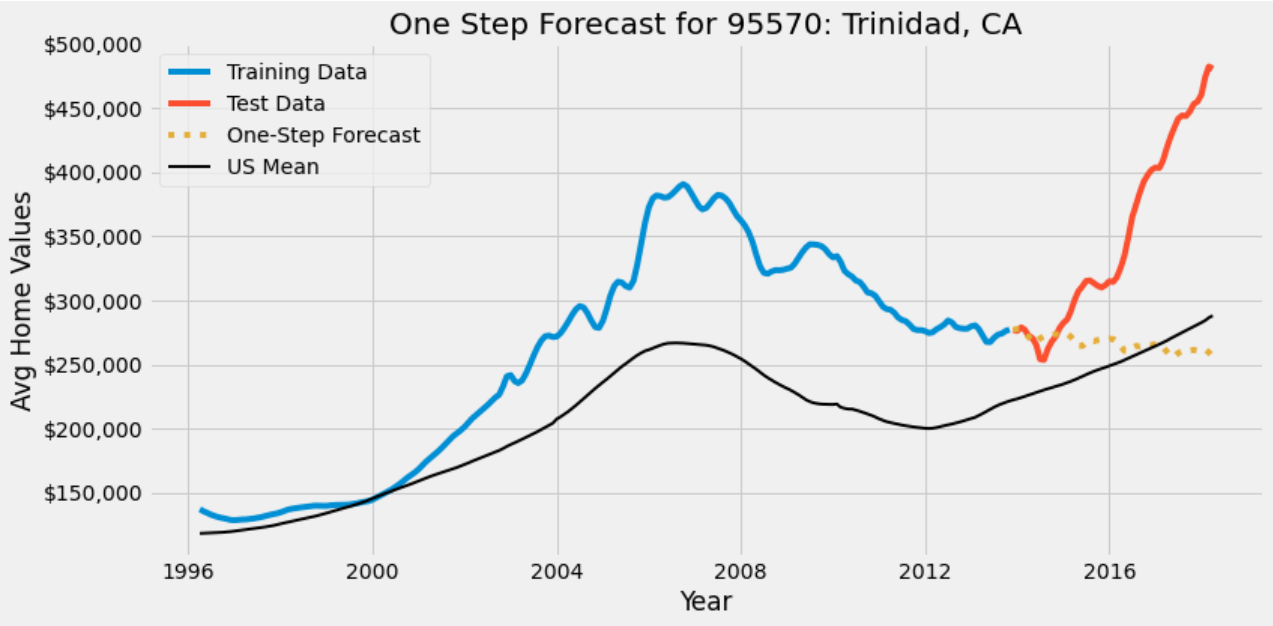
In [534]:
```
zip_95570_model_1 = sarima_model
```

executed in 3ms, finished 14:25:44 2022-05-24

In [535]:

```
pred = zip_95570_model_1.get_prediction(
    start=test.index.min(),
    end=test.index.max(),
    dynamic=False)

fig, ax = plt.subplots(figsize=(12, 6))
ax.plot(train, label='Training Data')
ax.plot(test, label='Test Data')
ax.plot(pred.predicted_mean, label='One-Step Forecast', ls=':')
line1 = sns.lineplot(data=avg_home_df, x='time', y='home_value',
                     color='black', linewidth=(2), label='US Mean');

ax.set_xlabel('Year')
ax.set_ylabel('Avg Home Values')
ax.set_title("One Step Forecast for 95570: Trinidad, CA", fontsize = 20)
ax.legend()
plt.savefig('images/trinidad_one_step_1')
ax.yaxis.set_major_formatter(tick)
fig.tight_layout()
```

executed in 263ms, finished 14:25:44 2022-05-24



9.2 Analysis:

- The forecast doesn't match the test data at all, but looking at the training data's plot, it's easy to see why.
- **It is unreasonable to expect the model to capture the upward trend that the test data shows, so I am going to increase the training group's size and see if that better matches the test data without overfitting on it.**

9.3 Model 1.5 (different train/test split ratio)

In [536]:
```
combo = zip_95570_combo_1
combo
```

executed in 3ms, finished 14:25:44 2022-05-24

Out[536]:
```
((1, 1, 1), (1, 1, 1, 12))
```

In [537]:
```
# Manually split data.
temp_ts = zip_95570_ts

SPLIT = int(temp_ts.shape[0]*0.90)
train, test = temp_ts[:SPLIT], temp_ts[SPLIT:]

# Model
sarima_model = SARIMAX(
    train,
    order=combo[0],
    seasonal_order=combo[1],
    enforce_stationarity=False,
    enforce_invertibility=False).fit()
sarima_model.summary()
```

executed in 475ms, finished 14:25:45 2022-05-24

Out[537]:

SARIMAX Results

Dep. Variable:	value	**No. Observations:**	238
Model:	SARIMAX(1, 1, 1)x(1, 1, 1, 12)	**Log Likelihood**	-1962.616
Date:	Tue, 24 May 2022	**AIC**	3935.232
Time:	14:25:45	**BIC**	3951.991
Sample:	04-01-1996	**HQIC**	3942.006
	- 01-01-2016		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	0.6120	0.060	10.231	0.000	0.495	0.729
ma.L1	0.8132	0.070	11.649	0.000	0.676	0.950
ar.S.L12	0.2694	0.051	5.242	0.000	0.169	0.370
ma.S.L12	-0.5017	0.059	-8.563	0.000	-0.616	-0.387
sigma2	9.258e+06	2.57e-09	3.6e+15	0.000	9.26e+06	9.26e+06

Ljung-Box (L1) (Q):	1.62	**Jarque-Bera (JB):**	57.66
Prob(Q):	0.20	**Prob(JB):**	0.00
Heteroskedasticity (H):	2.61	**Skew:**	-0.12
Prob(H) (two-sided):	0.00	**Kurtosis:**	5.55

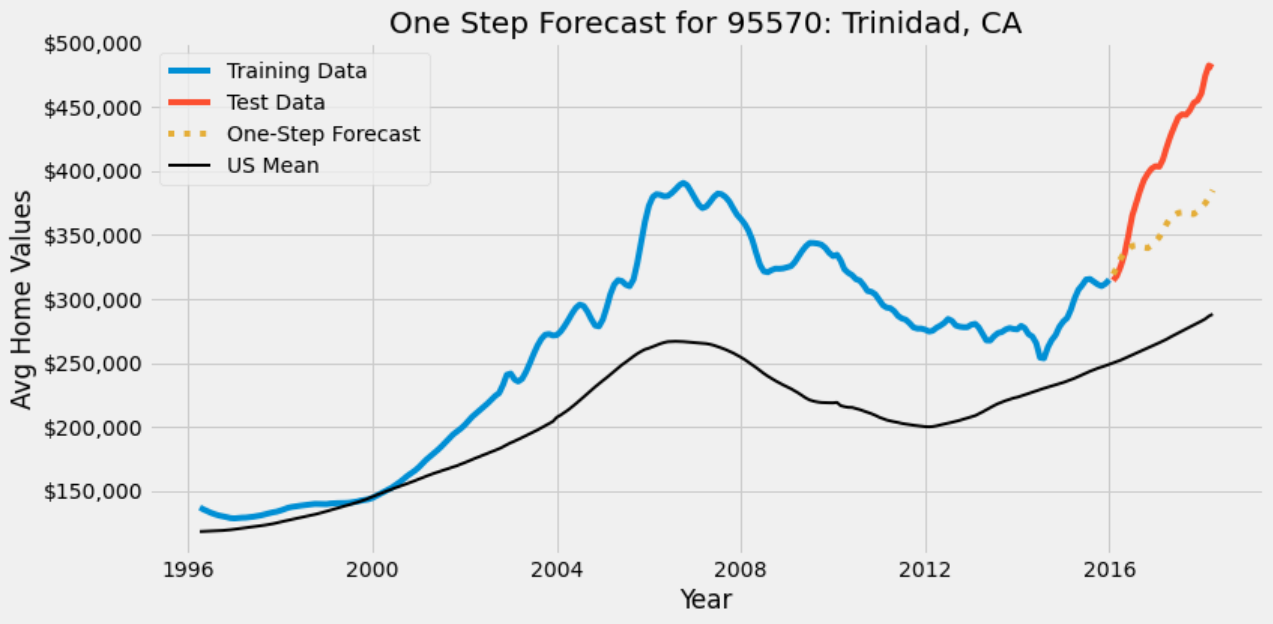
Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
[2] Covariance matrix is singular or near-singular, with condition number 8.04e+30. Standard errors may be unstable.

In [538]:
```
zip_95570_model_1_5 = sarima_model
```

executed in 2ms, finished 14:25:45 2022-05-24

In [539]:
```
pred = zip_95570_model_1_5.get_prediction(
    start=test.index.min(),
    end=test.index.max(),
    dynamic=False)

fig, ax = plt.subplots(figsize=(12, 6))
ax.plot(train, label='Training Data')
ax.plot(test, label='Test Data')
ax.plot(pred.predicted_mean, label='One-Step Forecast', ls=':')
line1 = sns.lineplot(data=avg_home_df, x='time', y='home_value',
                     color='black', linewidth=(2), label='US Mean');

ax.set_xlabel('Year')
ax.set_ylabel('Avg Home Values')
ax.set_title("One Step Forecast for 95570: Trinidad, CA", fontsize = 20)
ax.legend()
plt.savefig('images/us_mean_one_step_2')
ax.yaxis.set_major_formatter(tick)
fig.tight_layout()
```

executed in 266ms, finished 14:25:45 2022-05-24



9.4 Analysis:

- **Increasing the size of the training group definitely helped. I will use the same train_test_split ratio of 90/10 for Model #2.**

In [540]:
```
y_forecasted = pred.predicted_mean
y_truth = test['value']
mse = ((y_forecasted - y_truth) ** 2).mean()
print('The Mean Squared Error is {}'.format(round(mse, 2)))
print('The Root Mean Squared Error is {}'.format(round(np.sqrt(mse), 2)))
```

executed in 4ms, finished 14:25:45 2022-05-24

```
The Mean Squared Error is 4152768083.15
The Root Mean Squared Error is 64441.97
```

9.5 Model 2

In [541]:
```
combo = zip_95570_combo_2
combo
```

executed in 3ms, finished 14:25:45 2022-05-24

Out[541]: ((2, 2, 2), (2, 2, 2, 12))

In [542]:
```
# Manually split data.
temp_ts = zip_95570_ts

SPLIT = int(temp_ts.shape[0]*0.90)
train, test = temp_ts[:SPLIT], temp_ts[SPLIT:]

# Model
sarima_model = SARIMAX(
    train,
    order=combo[0],
    seasonal_order=combo[1],
    enforce_stationarity=False,
    enforce_invertibility=False).fit()
sarima_model.summary()
```

executed in 4.43s, finished 14:25:49 2022-05-24

Out[542]:

SARIMAX Results

Dep. Variable:	value	**No. Observations:**	238
Model:	SARIMAX(2, 2, 2)x(2, 2, 2, 12)	**Log Likelihood**	-1779.848
Date:	Tue, 24 May 2022	**AIC**	3577.696
Time:	14:25:49	**BIC**	3606.679
Sample:	04-01-1996	**HQIC**	3589.442
	- 01-01-2016		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	0.7947	0.283	2.813	0.005	0.241	1.348
ar.L2	-0.1759	0.292	-0.603	0.547	-0.748	0.396
ma.L1	-0.5038	0.282	-1.786	0.074	-1.056	0.049
ma.L2	-0.4581	0.260	-1.759	0.079	-0.969	0.052
ar.S.L12	-0.7903	0.229	-3.458	0.001	-1.238	-0.342
ar.S.L24	-0.4526	0.208	-2.180	0.029	-0.860	-0.046
ma.S.L12	-0.5134	0.212	-2.419	0.016	-0.929	-0.098
ma.S.L24	-0.2086	0.138	-1.509	0.131	-0.480	0.062
sigma2	2.48e+07	5.05e-09	4.91e+15	0.000	2.48e+07	2.48e+07

Ljung-Box (L1) (Q):	1.76	**Jarque-Bera (JB):**	13.36
Prob(Q):	0.18	**Prob(JB):**	0.00
Heteroskedasticity (H):	1.28	**Skew:**	0.03
Prob(H) (two-sided):	0.34	**Kurtosis:**	4.31

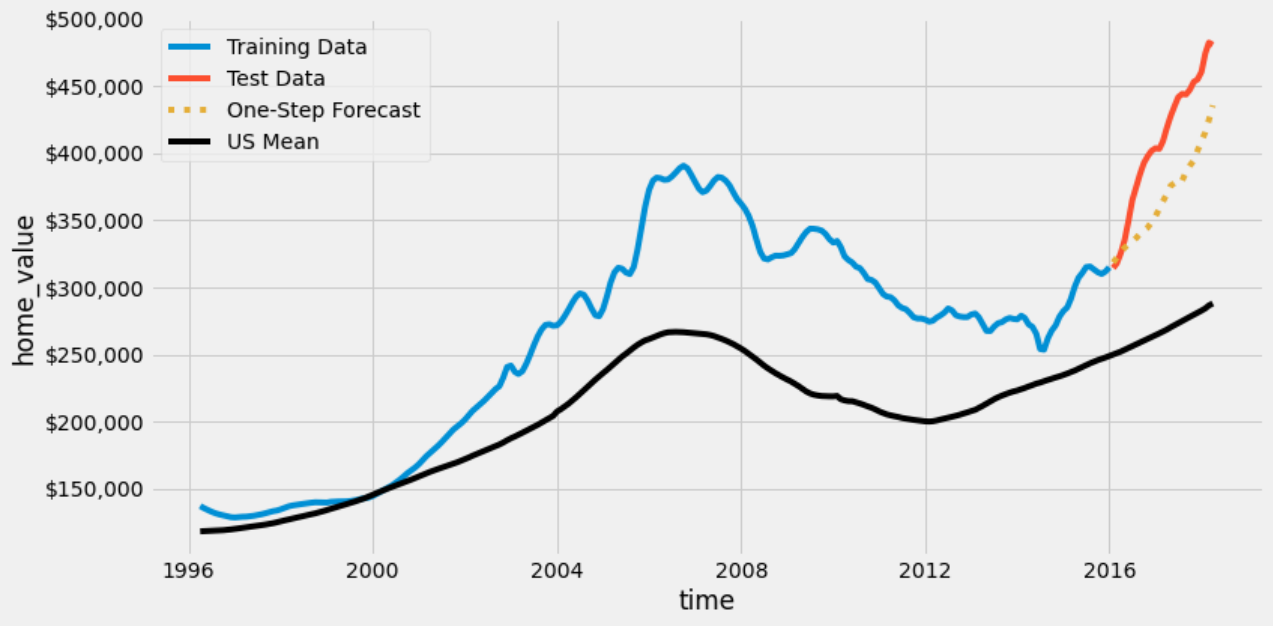
Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
[2] Covariance matrix is singular or near-singular, with condition number 5.41e+31. Standard errors may be unstable.

In [543]:
```
zip_95570_model_2 = sarima_model
```

executed in 3ms, finished 14:25:49 2022-05-24

In [544]:
```
pred = zip_95570_model_2.get_prediction(
    start=test.index.min(),
    end=test.index.max(),
    dynamic=False)

fig, ax = plt.subplots(figsize=(12, 6))
ax.plot(train, label='Training Data')
ax.plot(test, label='Test Data')
ax.plot(pred.predicted_mean, label='One-Step Forecast', ls=':')
line1 = sns.lineplot(data=avg_home_df, x='time', y='home_value', color='black', label='US Mean');
ax.legend()
plt.savefig('images/trinidad_one_step_3')
ax.yaxis.set_major_formatter(tick)
fig.tight_layout()
```

executed in 258ms, finished 14:25:50 2022-05-24



In [545]:
```
y_forecasted = pred.predicted_mean
y_truth = test['value']
mse = ((y_forecasted - y_truth) ** 2).mean()
print('The Mean Squared Error is {}'.format(round(mse, 2)))
print('The Root Mean Squared Error is {}'.format(round(np.sqrt(mse), 2)))
```

executed in 4ms, finished 14:25:50 2022-05-24

```
The Mean Squared Error is 2239631687.88
The Root Mean Squared Error is 47324.75
```

9.6 Model Comparison & Decision

- Model #2 has a lower RMSE and also better reflects the test data. I will use this model to make my predicitions.

In [546]:
```
y_forecasted = pred.predicted_mean
y_truth = test['value']
mse = ((y_forecasted - y_truth) ** 2).mean()
print('The Mean Squared Error is {}'.format(round(mse, 2)))
print('The Root Mean Squared Error is {}'.format(round(np.sqrt(mse), 2)))
```

executed in 4ms, finished 14:25:50 2022-05-24

```
The Mean Squared Error is 2239631687.88
The Root Mean Squared Error is 47324.75
```

9.7 Prediction

In [547]:
```
#combo = zip_95570_combo_1
combo = zip_95570_combo_2
combo
```

executed in 2ms, finished 14:25:50 2022-05-24

Out[547]: ((2, 2, 2), (2, 2, 2, 12))

In [548]:
```
# Running model again with FULL data

sarima_model = SARIMAX(
    zip_95570_ts,
    order=combo[0],
    seasonal_order=combo[1],
    enforce_stationarity=False,
    enforce_invertibility=False).fit()
sarima_model.summary()
```

executed in 8.06s, finished 14:25:58 2022-05-24

Out[548]:

SARIMAX Results

Dep. Variable:	value	**No. Observations:**	265
Model:	SARIMAX(2, 2, 2)x(2, 2, 2, 12)	**Log Likelihood**	-2034.533
Date:	Tue, 24 May 2022	**AIC**	4087.066
Time:	14:25:58	**BIC**	4117.275
Sample:	04-01-1996	**HQIC**	4099.276
	- 04-01-2018		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	0.9138	0.164	5.575	0.000	0.593	1.235
ar.L2	-0.3686	0.167	-2.208	0.027	-0.696	-0.041
ma.L1	-0.7812	0.169	-4.631	0.000	-1.112	-0.451
ma.L2	-0.0437	0.176	-0.249	0.804	-0.388	0.300
ar.S.L12	-0.8435	0.098	-8.575	0.000	-1.036	-0.651
ar.S.L24	-0.4823	0.079	-6.126	0.000	-0.637	-0.328
ma.S.L12	-0.5676	0.091	-6.213	0.000	-0.747	-0.389
ma.S.L24	-0.1973	0.057	-3.473	0.001	-0.309	-0.086
sigma2	1.383e+07	1.35e-08	1.03e+15	0.000	1.38e+07	1.38e+07

Ljung-Box (L1) (Q):	4.68	**Jarque-Bera (JB):**	15.62
Prob(Q):	0.03	**Prob(JB):**	0.00
Heteroskedasticity (H):	2.15	**Skew:**	-0.02
Prob(H) (two-sided):	0.00	**Kurtosis:**	4.33

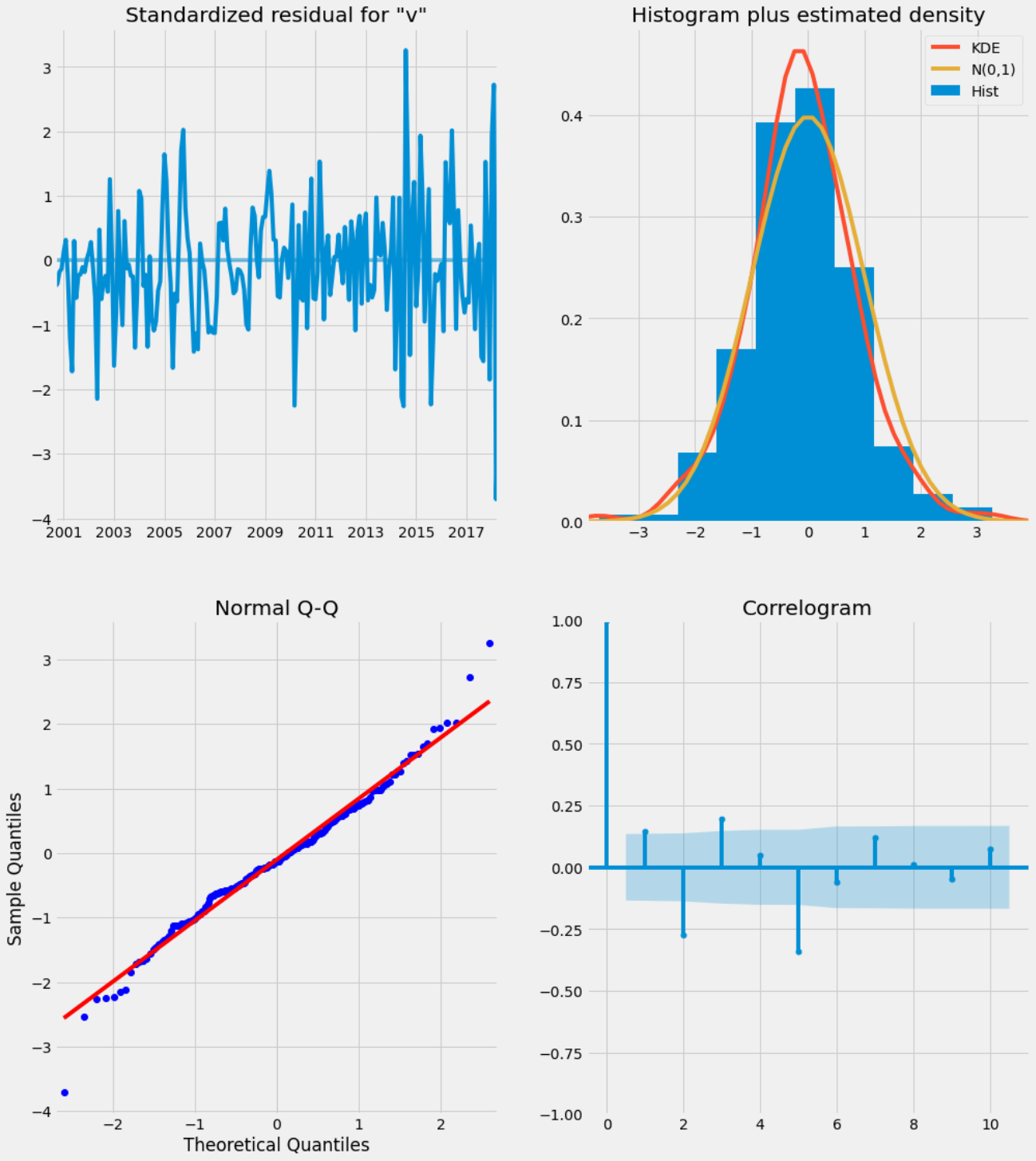
Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
[2] Covariance matrix is singular or near-singular, with condition number 3.15e+30. Standard errors may be unstable.

In [549]:
```
zip_95570_model_full = sarima_model
```

executed in 3ms, finished 14:25:58 2022-05-24

In [550]:
```
zip_95570_model_full.plot_diagnostics(figsize=(15,18))
plt.show()
```

executed in 336ms, finished 14:25:58 2022-05-24



In [551]:
```
1 zip_95570_ts.tail(1)
```

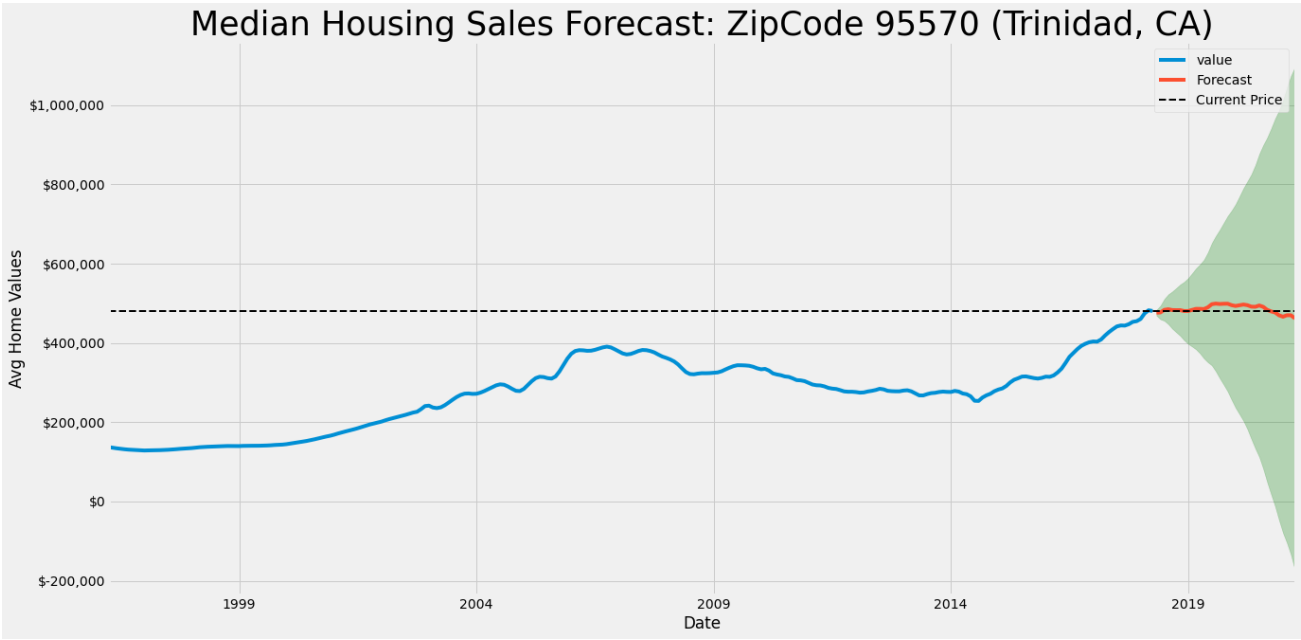
executed in 4ms, finished 14:25:58 2022-05-24

Out[551]:

time	value
2018-04-01	480,600.00

In [552]:
```
1  # Get forecast 3 yrs ahead in future (36 steps)
2  prediction = zip_95570_model_full.get_forecast(steps=36)
3
4  # Get confidence intervals of forecasts
5  pred_conf = prediction.conf_int()
6
7  # Plot future predictions with confidence intervals
8  ax = zip_95570_ts['1996-01':].plot(label='observed', figsize=(20, 10))
9  prediction.predicted_mean.plot(ax=ax, label='Forecast')
10 ax.fill_between(pred_conf.index,
11                 pred_conf.iloc[:, 0],
12                 pred_conf.iloc[:, 1], color='green', alpha=0.25)
13
14 ax.axhline(480600, ls='--', color='black', linewidth=(2), label='Current Price')
15
16 ax.set_xlabel('Date')
17 ax.set_ylabel('Avg Home Values')
18 ax.set_title("Median Housing Sales Forecast: ZipCode 95570 (Trinidad, CA)", fontsize = 35)
19 ax.yaxis.set_major_formatter(tick)
20 plt.savefig('images/trinidad_forecast')
21
22 plt.legend()
23 plt.show()
```

executed in 243ms, finished 14:25:58 2022-05-24



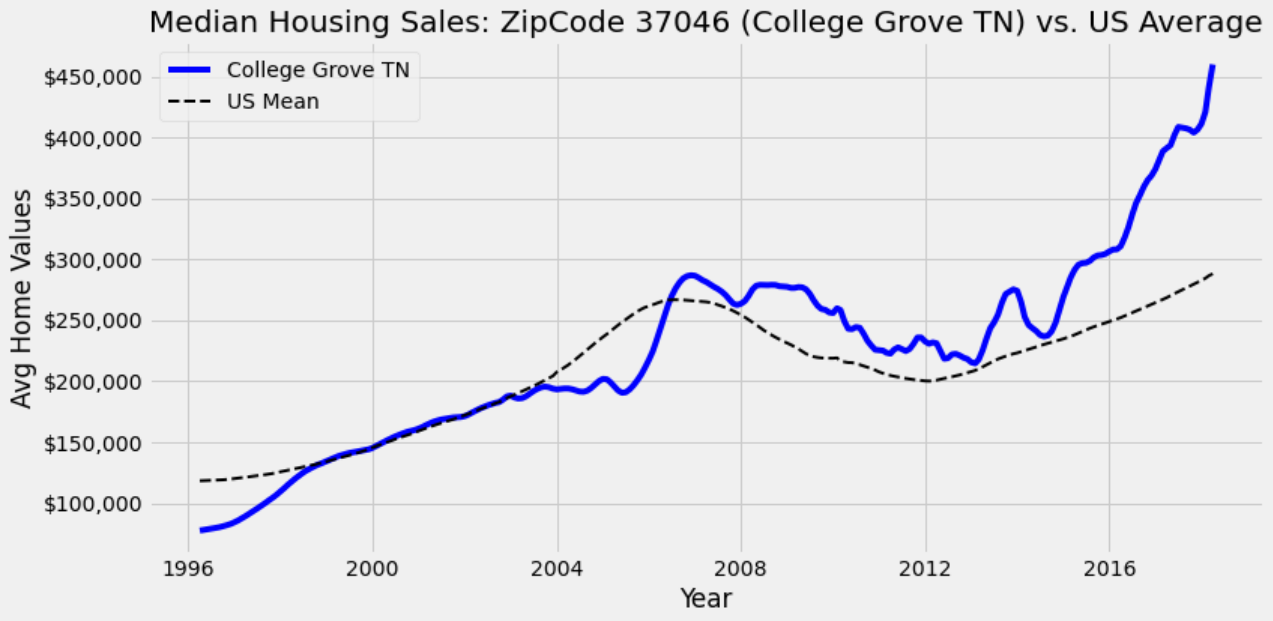
9.8 Analysis:

- Forecast mean stays at the current price. Confidence window is too large. Nearly equal chance for gain and loss.
- There is a lot of noise in the residuals, which is reflected in how wide the confidence window is.
- **I do not recommend investing in this zipcode.**

10 37046: College Grove TN

In [553]:
```
fig, ax = plt.subplots(figsize=(12, 6))
ax.plot(zip_37046_ts, label='College Grove TN', color='blue')
line1 = sns.lineplot(data=avg_home_df, x='time', y='home_value', ls='--',
                     color='black', linewidth=(2), label='US Mean');
ax.legend()
ax.set_title("Median Housing Sales: ZipCode 37046 (College Grove TN) vs. US Average", fontsize = 20)
ax.set_xlabel('Year')
ax.set_ylabel('Avg Home Values')
ax.yaxis.set_major_formatter(tick)
fig.tight_layout()
```

executed in 152ms, finished 14:25:59 2022-05-24



In [554]:
```
#sarimax_param_search(zip_37046_ts)
```

executed in 1ms, finished 14:25:59 2022-05-24

In [555]:
```
#Optimal SARIMA order: ((1, 1, 1), (1, 1, 1, 12)) AIC: 4342.43246819299
zip_37046_combo_1 = ((1, 1, 1), (1, 1, 1, 12))
```

executed in 2ms, finished 14:25:59 2022-05-24

In [556]:
```
# Optimal SARIMA order: ((1, 1, 2), (1, 2, 2, 12)) AIC: 3884.017214422042
zip_37046_combo_2 = ((1, 1, 2), (1, 2, 2, 12))
```

executed in 1ms, finished 14:25:59 2022-05-24

10.1 Model #1

In [557]:
```
combo = zip_37046_combo_1
combo
```

executed in 3ms, finished 14:25:59 2022-05-24

Out[557]:
```
((1, 1, 1), (1, 1, 1, 12))
```

In [558]:
```
# Manually split data.
temp_ts = zip_37046_ts

SPLIT = int(temp_ts.shape[0]*0.80)
train, test = temp_ts[:SPLIT], temp_ts[SPLIT:]

# Model
sarima_model = SARIMAX(
    train,
    order=combo[0],
    seasonal_order=combo[1],
    enforce_stationarity=False,
    enforce_invertibility=False).fit()
sarima_model.summary()
```

executed in 231ms, finished 14:25:59 2022-05-24

Out[558]:

SARIMAX Results

Dep. Variable:	value	**No. Observations:**	212
Model:	SARIMAX(1, 1, 1)x(1, 1, 1, 12)	**Log Likelihood**	-1653.443
Date:	Tue, 24 May 2022	**AIC**	3316.886
Time:	14:25:59	**BIC**	3332.988
Sample:	04-01-1996	**HQIC**	3323.412
	- 11-01-2013		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	0.7207	0.045	15.929	0.000	0.632	0.809
ma.L1	0.4620	0.031	14.760	0.000	0.401	0.523
ar.S.L12	-0.5758	0.081	-7.088	0.000	-0.735	-0.417
ma.S.L12	-0.0671	0.071	-0.943	0.346	-0.207	0.072
sigma2	3.801e+06	3.1e+05	12.254	0.000	3.19e+06	4.41e+06

Ljung-Box (L1) (Q):	5.82	**Jarque-Bera (JB):**	115.83
Prob(Q):	0.02	**Prob(JB):**	0.00
Heteroskedasticity (H):	8.59	**Skew:**	-0.37
Prob(H) (two-sided):	0.00	**Kurtosis:**	6.81

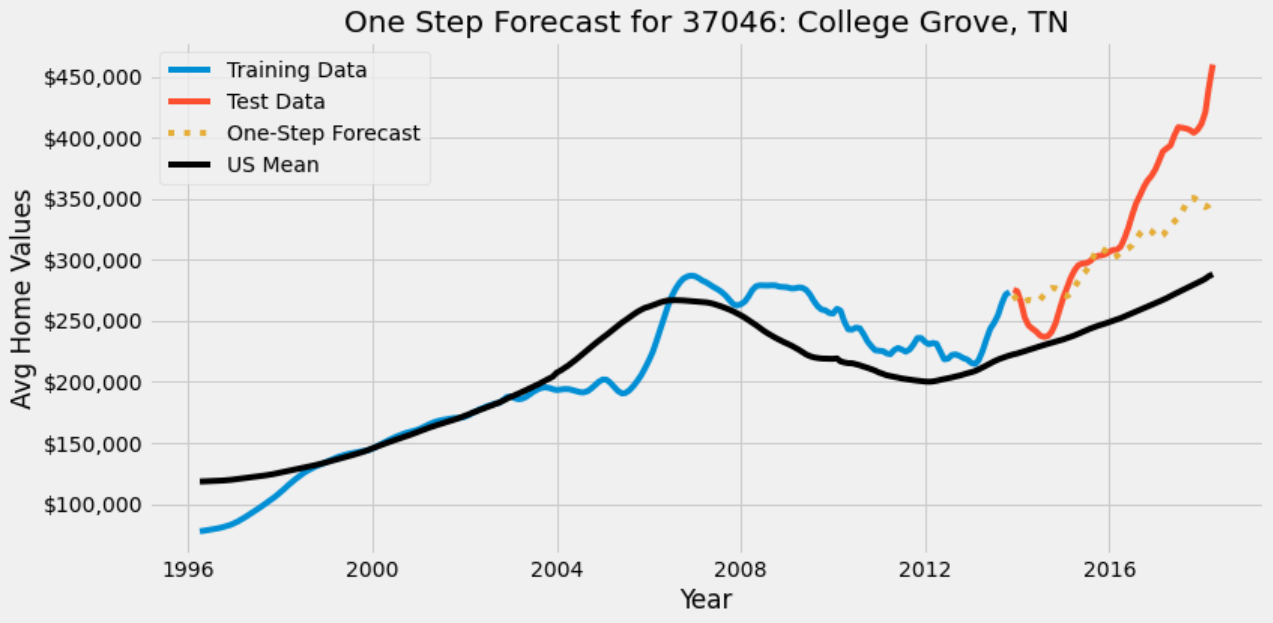
Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).

In [559]:
```
zip_37046_model_1 = sarima_model
```

executed in 2ms, finished 14:25:59 2022-05-24

In [560]:
```
pred = zip_37046_model_1.get_prediction(
    start=test.index.min(),
    end=test.index.max(),
    dynamic=True)

fig, ax = plt.subplots(figsize=(12, 6))
ax.plot(train, label='Training Data')
ax.plot(test, label='Test Data')
ax.plot(pred.predicted_mean, label='One-Step Forecast', ls=':')
line1 = sns.lineplot(data=avg_home_df, x='time', y='home_value', color='black', label='US Mean');

ax.set_xlabel('Year')
ax.set_ylabel('Avg Home Values')
ax.set_title("One Step Forecast for 37046: College Grove, TN", fontsize = 20)
ax.legend()
plt.savefig('images/college_grove_one_step_1')
ax.yaxis.set_major_formatter(tick)
fig.tight_layout()
```

executed in 231ms, finished 14:25:59 2022-05-24



In [561]:
```
y_forecasted = pred.predicted_mean
y_truth = test['value']
mse = ((y_forecasted - y_truth) ** 2).mean()
print('The Mean Squared Error is {}'.format(round(mse, 2)))
print('The Root Mean Squared Error is {}'.format(round(np.sqrt(mse), 2)))
```

executed in 3ms, finished 14:25:59 2022-05-24

```
The Mean Squared Error is 1834575615.05
The Root Mean Squared Error is 42831.95
```

10.2 Model #2

In [562]:
```
combo = zip_37046_combo_2
combo
```

executed in 2ms, finished 14:25:59 2022-05-24

Out[562]:
```
((1, 1, 2), (1, 2, 2, 12))
```

In [563]:
```
# Manually split data.
temp_ts = zip_37046_ts

SPLIT = int(temp_ts.shape[0]*0.80)
train, test = temp_ts[:SPLIT], temp_ts[SPLIT:]

# Model
sarima_model = SARIMAX(
    train,
    order=combo[0],
    seasonal_order=combo[1],
    enforce_stationarity=False,
    enforce_invertibility=False).fit()
sarima_model.summary()
```

executed in 2.56s, finished 14:26:02 2022-05-24

Out[563]:

SARIMAX Results

Dep. Variable:	value	**No. Observations:**	212
Model:	SARIMAX(1, 1, 2)x(1, 2, 2, 12)	**Log Likelihood**	-1485.272
Date:	Tue, 24 May 2022	**AIC**	2984.544
Time:	14:26:02	**BIC**	3006.071
Sample:	04-01-1996	**HQIC**	2993.285
	- 11-01-2013		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	0.3557	0.204	1.744	0.081	-0.044	0.756
ma.L1	1.2094	0.166	7.272	0.000	0.883	1.535
ma.L2	0.5077	0.176	2.888	0.004	0.163	0.852
ar.S.L12	-0.6626	0.300	-2.210	0.027	-1.250	-0.075
ma.S.L12	-0.3840	0.295	-1.302	0.193	-0.962	0.194
ma.S.L24	0.0702	0.284	0.247	0.805	-0.487	0.628
sigma2	1.244e+07	1.85e-08	6.73e+14	0.000	1.24e+07	1.24e+07

Ljung-Box (L1) (Q):	0.00	**Jarque-Bera (JB):**	46.08
Prob(Q):	1.00	**Prob(JB):**	0.00
Heteroskedasticity (H):	8.97	**Skew:**	-0.05
Prob(H) (two-sided):	0.00	**Kurtosis:**	5.63

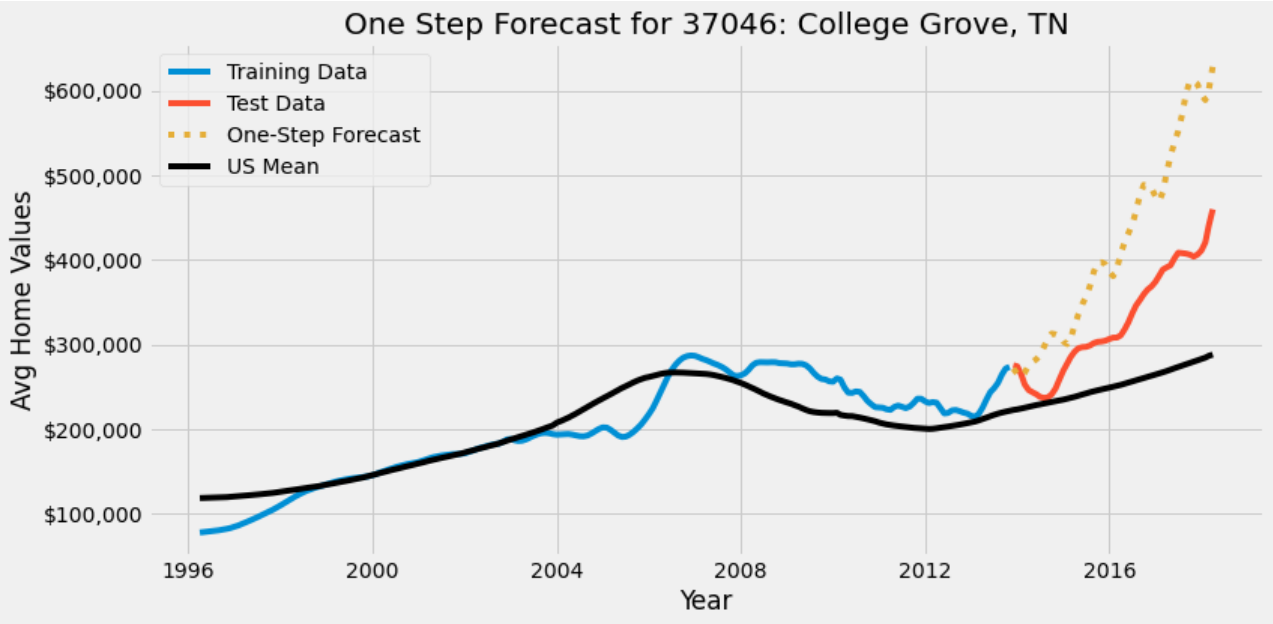
Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
[2] Covariance matrix is singular or near-singular, with condition number 2e+31. Standard errors may be unstable.

In [564]:
```
zip_37046_model_2 = sarima_model
```

executed in 2ms, finished 14:26:02 2022-05-24

In [565]:
```
pred = zip_37046_model_2.get_prediction(
    start=test.index.min(),
    end=test.index.max(),
    dynamic=True)

fig, ax = plt.subplots(figsize=(12, 6))
ax.plot(train, label='Training Data')
ax.plot(test, label='Test Data')
ax.plot(pred.predicted_mean, label='One-Step Forecast', ls=':')
line1 = sns.lineplot(data=avg_home_df, x='time', y='home_value', color='black', label='US Mean');
ax.set_xlabel('Year')
ax.set_ylabel('Avg Home Values')
ax.set_title("One Step Forecast for 37046: College Grove, TN", fontsize = 20)
ax.legend()
plt.savefig('images/us_mean_one_step_2')
ax.yaxis.set_major_formatter(tick)
fig.tight_layout()
```

executed in 234ms, finished 14:26:02 2022-05-24



In [566]:
```
y_forecasted = pred.predicted_mean
y_truth = test['value']
mse = ((y_forecasted - y_truth) ** 2).mean()
print('The Mean Squared Error is {}'.format(round(mse, 2)))
print('The Root Mean Squared Error is {}'.format(round(np.sqrt(mse), 2)))
```

executed in 3ms, finished 14:26:02 2022-05-24

```
The Mean Squared Error is 11513817677.28
The Root Mean Squared Error is 107302.46
```

10.3 Model Comparison & Decision:

- Model 1 has a lower RSME and it's one-step forecast better matches the test data. I will proceed with Model #1 for my Forecast.

10.4 Prediction

In [567]:
```
combo = zip_37046_combo_1
#combo = zip_37046_combo_2
combo
```

executed in 2ms, finished 14:26:02 2022-05-24

Out[567]:
```
((1, 1, 1), (1, 1, 1, 12))
```

In [568]:
```
# Running model again with FULL data

sarima_model = SARIMAX(
    zip_37046_ts,
    order=combo[0],
    seasonal_order=combo[1],
    enforce_stationarity=False,
    enforce_invertibility=False).fit()
sarima_model.summary()
```

executed in 447ms, finished 14:26:02 2022-05-24

Out[568]:

SARIMAX Results

Dep. Variable:	value	**No. Observations:**	265
Model:	SARIMAX(1, 1, 1)x(1, 1, 1, 12)	**Log Likelihood**	-2166.216
Date:	Tue, 24 May 2022	**AIC**	4342.432
Time:	14:26:02	**BIC**	4359.794
Sample:	04-01-1996	**HQIC**	4349.429
	- 04-01-2018		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	0.7166	0.057	12.507	0.000	0.604	0.829
ma.L1	0.7790	0.037	21.255	0.000	0.707	0.851
ar.S.L12	0.2480	0.086	2.898	0.004	0.080	0.416
ma.S.L12	-0.6107	0.068	-8.928	0.000	-0.745	-0.477
sigma2	5.939e+06	5.34e+05	11.114	0.000	4.89e+06	6.99e+06

Ljung-Box (L1) (Q):	1.19	**Jarque-Bera (JB):**	31.81
Prob(Q):	0.28	**Prob(JB):**	0.00
Heteroskedasticity (H):	7.74	**Skew:**	0.07
Prob(H) (two-sided):	0.00	**Kurtosis:**	4.79

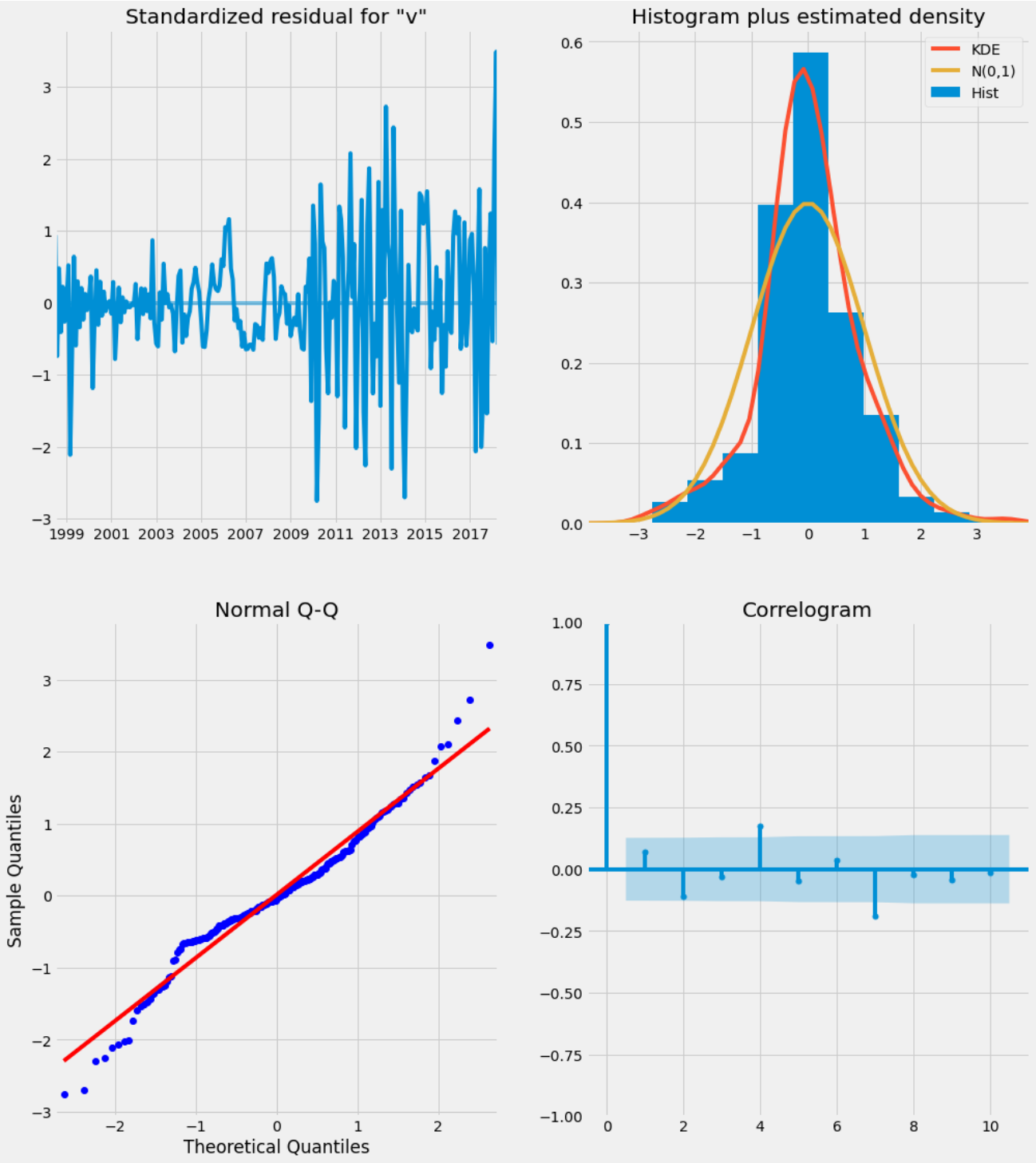
Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).

In [569]:
```
zip_37046_model_full = sarima_model
```

executed in 2ms, finished 14:26:02 2022-05-24

In [570]:
```
1 zip_37046_model_full.plot_diagnostics(figsize=(15, 18))
2 plt.show()
```

executed in 380ms, finished 14:26:03 2022-05-24



In [571]:
```
1 zip_37046_ts.tail(1)
```

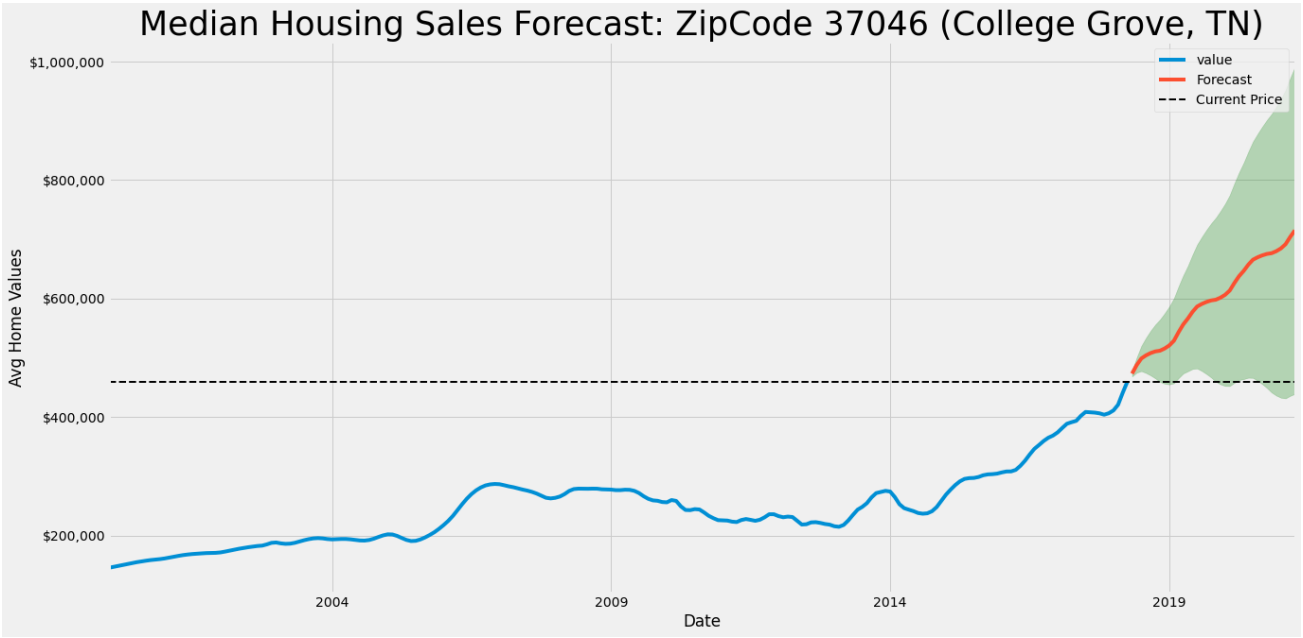
executed in 4ms, finished 14:26:03 2022-05-24

Out[571]:

time	value
2018-04-01	459,800.00

In [572]:
```
1  # Get forecast 3 yrs ahead in future (36 steps)
2  prediction = zip_37046_model_full.get_forecast(steps=36)
3
4  # Get confidence intervals of forecasts
5  pred_conf = prediction.conf_int()
6
7  # Plot future predictions with confidence intervals
8  ax = zip_37046_ts['2000-01':].plot(label='observed', figsize=(20, 10))
9  prediction.predicted_mean.plot(ax=ax, label='Forecast')
10 ax.fill_between(pred_conf.index,
11                 pred_conf.iloc[:, 0],
12                 pred_conf.iloc[:, 1], color='green', alpha=0.25)
13
14 ax.axhline(459800, ls='--', color='black', linewidth=(2), label='Current Price')
15
16 ax.set_xlabel('Date')
17 ax.set_ylabel('Avg Home Values')
18 ax.set_title("Median Housing Sales Forecast: ZipCode 37046 (College Grove, TN)", fontsize = 35)
19 ax.yaxis.set_major_formatter(tick)
20 plt.savefig('images/college_grove_forecast')
21
22 plt.legend()
23 plt.show()
```

executed in 294ms, finished 14:26:03 2022-05-24



10.5 Analysis:

- Potential loss forecast for the next three years is minimal. Model confidence interval lower value for year 3 is 439,002, which would just be a loss of 20,000.
- **While potential loss is low, potential gain is high. I recommend this as a zipcode to invest in.**

11 Metrics

- saving forecast metrics from my 5 recommended Zip Codes for my Final Analysis.

In [573]:
```
1 zip_37046_ts.tail(1)
```
executed in 4ms, finished 14:26:03 2022-05-24

Out[573]:

	value
time	
2018-04-01	459,800.00

In [574]:
```
1 analysis_df = pred_conf.resample('Y').mean()
2 analysis_df['mean'] = analysis_df.apply(lambda x: x.mean(), axis=1)
3
4 analysis_df['base'] = 459800
5 analysis_df['base'] = analysis_df['base'].astype(float)
```
executed in 5ms, finished 14:26:03 2022-05-24

In [575]:
```
1 analysis_df = analysis_df[['base', 'lower value', 'mean', 'upper value']]
2
3 analysis_df['Min_ROI'] = analysis_df.apply(lambda x: x['lower value'] - x['base'], axis=1)
4 analysis_df['Min_ROI%'] = analysis_df.apply(lambda x: x['Min_ROI'] / x['base'], axis=1)
5 analysis_df['Forecast_ROI'] = analysis_df.apply(lambda x: x['mean'] - x['base'], axis=1)
6 analysis_df['Forecast_ROI%'] = analysis_df.apply(lambda x: x['mean'] / x['base'], axis=1)
7 analysis_df['Max_ROI'] = analysis_df.apply(lambda x: x['upper value'] - x['base'], axis=1)
8 analysis_df['Max_ROI%'] = analysis_df.apply(lambda x: x['upper value'] / x['base'], axis=1)
```
executed in 10ms, finished 14:26:03 2022-05-24

In [576]:
```
1 zip_37046_metrics = analysis_df
```
executed in 2ms, finished 14:26:03 2022-05-24

In [577]:
```
1 zip_37046_metrics
```
executed in 6ms, finished 14:26:03 2022-05-24

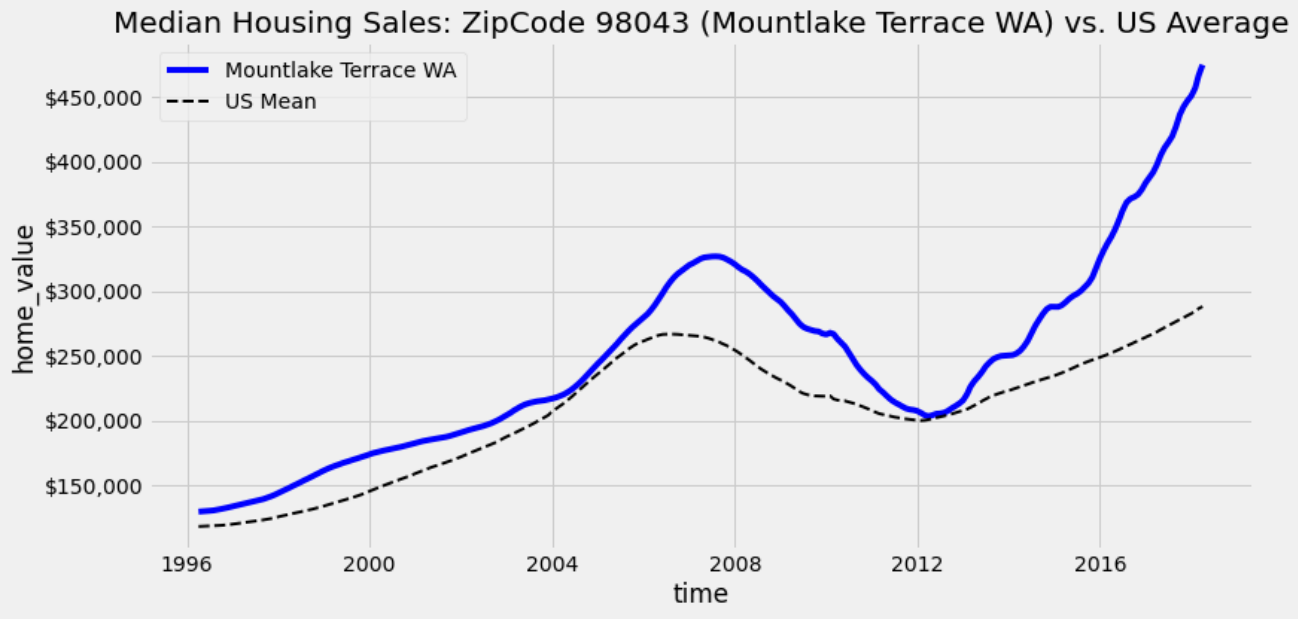
Out[577]:

	base	lower value	mean	upper value	Min_ROI	Min_ROI%	Forecast_ROI	Forecast_ROI%	Max_ROI	Max_ROI%
2018-12-31	459,800.00	468,257.69	501,253.86	534,250.03	8,457.69	0.02	41,453.86	1.09	74,450.03	1.16
2019-12-31	459,800.00	468,612.24	571,721.86	674,831.47	8,812.24	0.02	111,921.86	1.24	215,031.47	1.47
2020-12-31	459,800.00	455,392.58	652,309.15	849,225.72	-4,407.42	-0.01	192,509.15	1.42	389,425.72	1.85
2021-12-31	459,800.00	434,655.38	698,855.46	963,055.53	-25,144.62	-0.05	239,055.46	1.52	503,255.53	2.09

12 98043: Mountlake Terrace WA

In [578]:
```
fig, ax = plt.subplots(figsize=(12, 6))
ax.plot(zip_98043_ts, label='Mountlake Terrace WA', color='blue')
line1 = sns.lineplot(data=avg_home_df, x='time', y='home_value', ls='--',
                     color='black', linewidth=(2), label='US Mean');
ax.legend()
ax.set_title("Median Housing Sales: ZipCode 98043 (Mountlake Terrace WA) vs. US Average", fontsize =
ax.yaxis.set_major_formatter(tick)
fig.tight_layout()
```

executed in 153ms, finished 14:26:03 2022-05-24



In [579]:
```
#sarimax_param_search(zip_98043_ts)
```

executed in 2ms, finished 14:26:03 2022-05-24

In [580]:
```
#SARIMA Combos: ((1, 1, 1), (1, 1, 1, 12)) AIC: 3916.8599638578153
zip_98403_combo_1 = ((1, 1, 1), (1, 1, 1, 12))
```

executed in 2ms, finished 14:26:03 2022-05-24

In [581]:
```
#SARIMA Combos: ((2, 2, 2), (2, 2, 2, 12)) AIC: 3577.2395803210907
zip_98403_combo_2 = ((2, 2, 2), (2, 2, 2, 12))
```

executed in 2ms, finished 14:26:03 2022-05-24

12.1 Model #1

In [582]:
```
combo = zip_98403_combo_1
#combo = zip_98403_combo_2
combo
```

executed in 3ms, finished 14:26:03 2022-05-24

Out[582]: ((1, 1, 1), (1, 1, 1, 12))

In [583]:
```
# Manually split data.
temp_ts = zip_98043_ts

SPLIT = int(temp_ts.shape[0]*0.8)
train, test = temp_ts[:SPLIT], temp_ts[SPLIT:]

# Model
sarima_model = SARIMAX(
    train,
    order=combo[0],
    seasonal_order=combo[1],
    enforce_stationarity=False,
    enforce_invertibility=False).fit()
sarima_model.summary()
```

executed in 378ms, finished 14:26:04 2022-05-24

Out[583]:

SARIMAX Results

Dep. Variable:	value	**No. Observations:**	212
Model:	SARIMAX(1, 1, 1)x(1, 1, 1, 12)	**Log Likelihood**	-1470.060
Date:	Tue, 24 May 2022	**AIC**	2950.121
Time:	14:26:04	**BIC**	2966.222
Sample:	04-01-1996	**HQIC**	2956.646
	- 11-01-2013		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	0.9355	0.018	52.004	0.000	0.900	0.971
ma.L1	0.2389	0.023	10.276	0.000	0.193	0.285
ar.S.L12	-0.5693	0.051	-11.109	0.000	-0.670	-0.469
ma.S.L12	-0.0208	0.036	-0.573	0.566	-0.092	0.050
sigma2	4.569e+05	2.75e+04	16.591	0.000	4.03e+05	5.11e+05

Ljung-Box (L1) (Q):	2.18	**Jarque-Bera (JB):**	190.87
Prob(Q):	0.14	**Prob(JB):**	0.00
Heteroskedasticity (H):	21.30	**Skew:**	0.10
Prob(H) (two-sided):	0.00	**Kurtosis:**	7.97

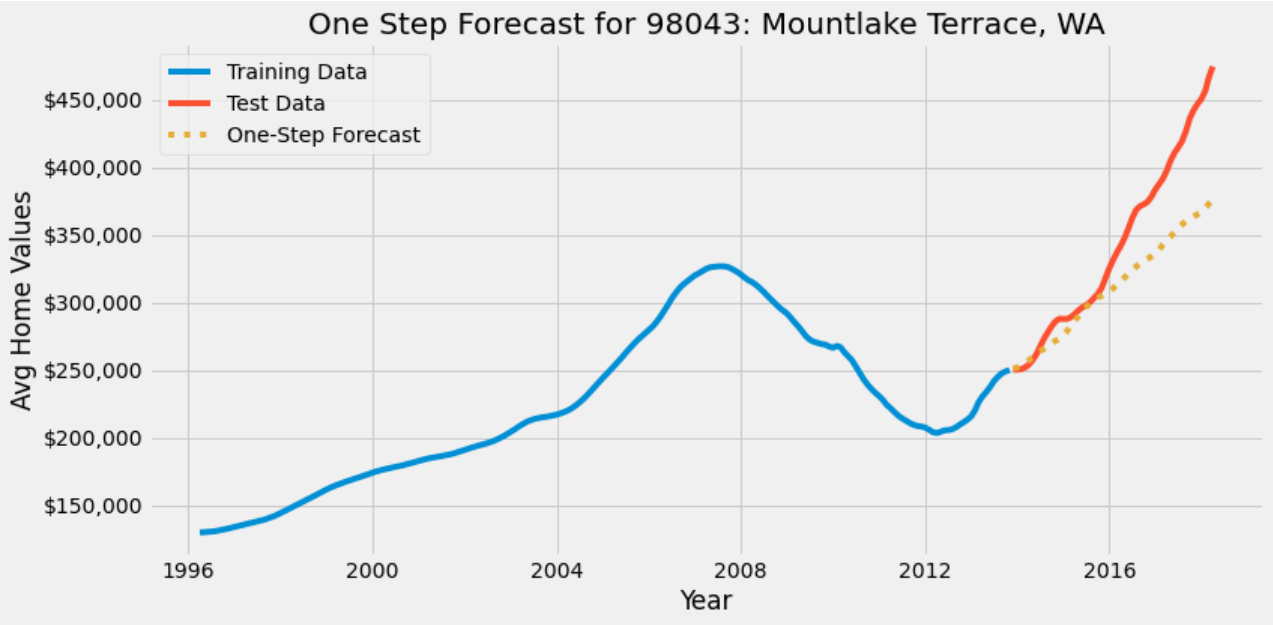
Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).

In [584]:
```
zip_98043_model_1 = sarima_model
```

executed in 2ms, finished 14:26:04 2022-05-24

In [585]:
```
pred = zip_98043_model_1.get_prediction(
    start=test.index.min(),
    end=test.index.max(),
    dynamic=True)

fig, ax = plt.subplots(figsize=(12, 6))
ax.plot(train, label='Training Data')
ax.plot(test, label='Test Data')
ax.plot(pred.predicted_mean, label='One-Step Forecast', ls=':')
#line1 = sns.lineplot(data=avg_home_df, x='time', y='home_value', color='black', label='US Mean');
ax.set_xlabel('Year')
ax.set_ylabel('Avg Home Values')
ax.set_title("One Step Forecast for 98043: Mountlake Terrace, WA", fontsize = 20)
ax.legend()
plt.savefig('images/mountlake_one_step_1')
ax.yaxis.set_major_formatter(tick)
fig.tight_layout()
```

executed in 179ms, finished 14:26:04 2022-05-24



In [586]:
```
y_forecasted = pred.predicted_mean
y_truth = test['value']
mse = ((y_forecasted - y_truth) ** 2).mean()
print('The Mean Squared Error is {}'.format(round(mse, 2)))
print('The Root Mean Squared Error is {}'.format(round(np.sqrt(mse), 2)))
```

executed in 2ms, finished 14:26:04 2022-05-24

```
The Mean Squared Error is 1799294145.98
The Root Mean Squared Error is 42418.09
```

12.2 Model #2

In [587]:
```
#combo = zip_98403_combo_1
combo = zip_98403_combo_2
combo
```

executed in 2ms, finished 14:26:04 2022-05-24

Out[587]: ((2, 2, 2), (2, 2, 2, 12))

In [588]:
```
# Manually split data.
temp_ts = zip_98043_ts

SPLIT = int(temp_ts.shape[0]*0.8)
train, test = temp_ts[:SPLIT], temp_ts[SPLIT:]

# Model
sarima_model = SARIMAX(
    train,
    order=combo[0],
    seasonal_order=combo[1],
    enforce_stationarity=False,
    enforce_invertibility=False).fit()
sarima_model.summary()
```

executed in 6.40s, finished 14:26:10 2022-05-24

Out[588]:

SARIMAX Results

Dep. Variable:	value	**No. Observations:**	212
Model:	SARIMAX(2, 2, 2)x(2, 2, 2, 12)	**Log Likelihood**	-1312.585
Date:	Tue, 24 May 2022	**AIC**	2643.170
Time:	14:26:10	**BIC**	2670.790
Sample:	04-01-1996	**HQIC**	2654.386
	- 11-01-2013		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	-0.7604	0.177	-4.303	0.000	-1.107	-0.414
ar.L2	-0.4977	0.126	-3.947	0.000	-0.745	-0.251
ma.L1	1.2923	0.178	7.251	0.000	0.943	1.642
ma.L2	0.6167	0.135	4.576	0.000	0.353	0.881
ar.S.L12	-0.1668	0.105	-1.588	0.112	-0.373	0.039
ar.S.L24	0.2869	0.101	2.827	0.005	0.088	0.486
ma.S.L12	-0.8459	0.102	-8.314	0.000	-1.045	-0.647
ma.S.L24	0.1526	0.142	1.079	0.281	-0.125	0.430
sigma2	1.301e+06	1.88e+05	6.915	0.000	9.32e+05	1.67e+06

Ljung-Box (L1) (Q):	0.55	**Jarque-Bera (JB):**	18.96
Prob(Q):	0.46	**Prob(JB):**	0.00
Heteroskedasticity (H):	4.82	**Skew:**	0.01
Prob(H) (two-sided):	0.00	**Kurtosis:**	4.69

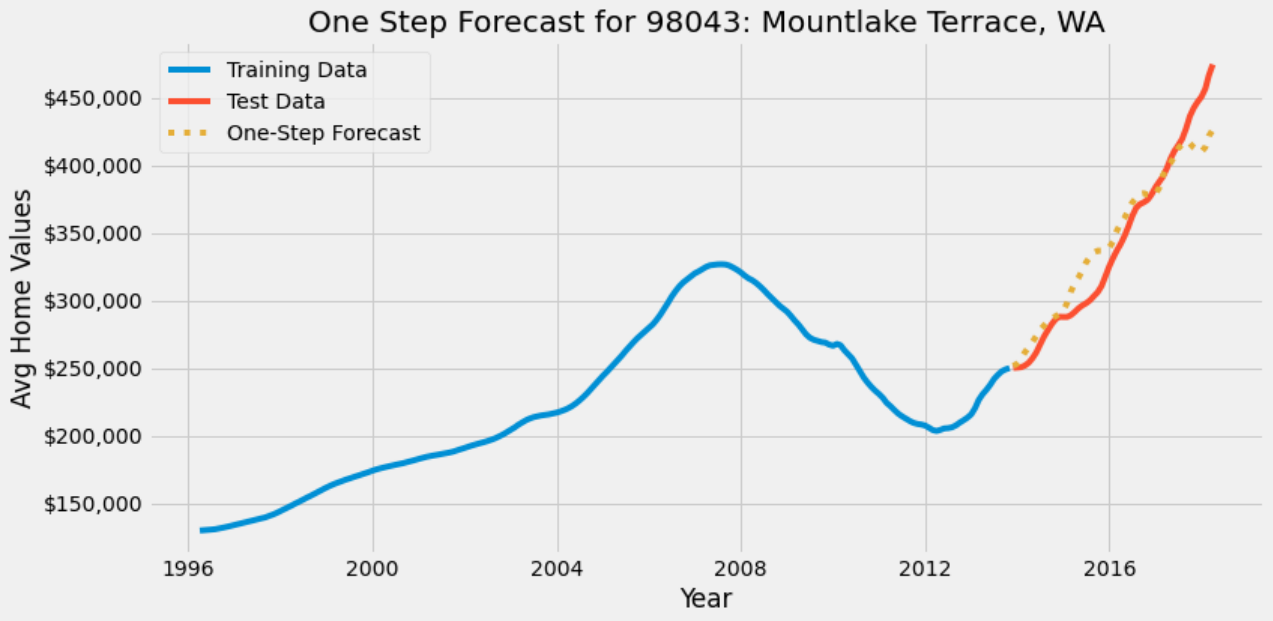
Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).

In [589]:
```
zip_98043_model_2 = sarima_model
```

executed in 3ms, finished 14:26:10 2022-05-24

In [590]:
```
pred = zip_98043_model_2.get_prediction(
    start=test.index.min(),
    end=test.index.max(),
    dynamic=True)

fig, ax = plt.subplots(figsize=(12, 6))
ax.plot(train, label='Training Data')
ax.plot(test, label='Test Data')
ax.plot(pred.predicted_mean, label='One-Step Forecast', ls=':')
#line1 = sns.lineplot(data=avg_home_df, x='time', y='home_value', color='black', label='US Mean');

ax.set_xlabel('Year')
ax.set_ylabel('Avg Home Values')
ax.set_title("One Step Forecast for 98043: Mountlake Terrace, WA", fontsize = 20)
ax.legend()
plt.savefig('images/mountlake_one_step_2')
ax.yaxis.set_major_formatter(tick)
fig.tight_layout()
```

executed in 196ms, finished 14:26:10 2022-05-24



In [591]:
```
y_forecasted = pred.predicted_mean
y_truth = test['value']
mse = ((y_forecasted - y_truth) ** 2).mean()
print('The Mean Squared Error is {}'.format(round(mse, 2)))
print('The Root Mean Squared Error is {}'.format(round(np.sqrt(mse), 2)))
```

executed in 3ms, finished 14:26:10 2022-05-24

```
The Mean Squared Error is 374885110.5
The Root Mean Squared Error is 19361.95
```

12.3 Model Comparison & Decision:

- Model 2 has a better RSME and also better models the test data. I will use Model #2 for my predictions.

12.4 Prediction

In [592]:
```
#combo = zip_98403_combo_1
combo = zip_98403_combo_2
combo
```

executed in 2ms, finished 14:26:10 2022-05-24

Out[592]:
```
((2, 2, 2), (2, 2, 2, 12))
```

In [593]:
```
# Running model again with FULL data

sarima_model = SARIMAX(
    zip_98043_ts,
    order=combo[0],
    seasonal_order=combo[1],
    enforce_stationarity=False,
    enforce_invertibility=False).fit()
sarima_model.summary()
```

executed in 7.32s, finished 14:26:18 2022-05-24

Out[593]:

SARIMAX Results

Dep. Variable:	value	**No. Observations:**	265
Model:	SARIMAX(2, 2, 2)x(2, 2, 2, 12)	**Log Likelihood**	-1779.620
Date:	Tue, 24 May 2022	**AIC**	3577.240
Time:	14:26:18	**BIC**	3607.449
Sample:	04-01-1996	**HQIC**	3589.449
	- 04-01-2018		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	0.4393	0.091	4.813	0.000	0.260	0.618
ar.L2	-0.5197	0.060	-8.698	0.000	-0.637	-0.403
ma.L1	0.1085	0.109	0.992	0.321	-0.106	0.323
ma.L2	-0.0419	0.077	-0.547	0.584	-0.192	0.108
ar.S.L12	-1.2029	0.061	-19.814	0.000	-1.322	-1.084
ar.S.L24	-0.7024	0.064	-10.907	0.000	-0.829	-0.576
ma.S.L12	-0.0171	0.051	-0.336	0.737	-0.117	0.082
ma.S.L24	-0.0537	0.041	-1.309	0.191	-0.134	0.027
sigma2	1.127e+06	8.73e+04	12.917	0.000	9.56e+05	1.3e+06

Ljung-Box (L1) (Q):	0.01	**Jarque-Bera (JB):**	30.06
Prob(Q):	0.92	**Prob(JB):**	0.00
Heteroskedasticity (H):	15.08	**Skew:**	0.16
Prob(H) (two-sided):	0.00	**Kurtosis:**	4.82

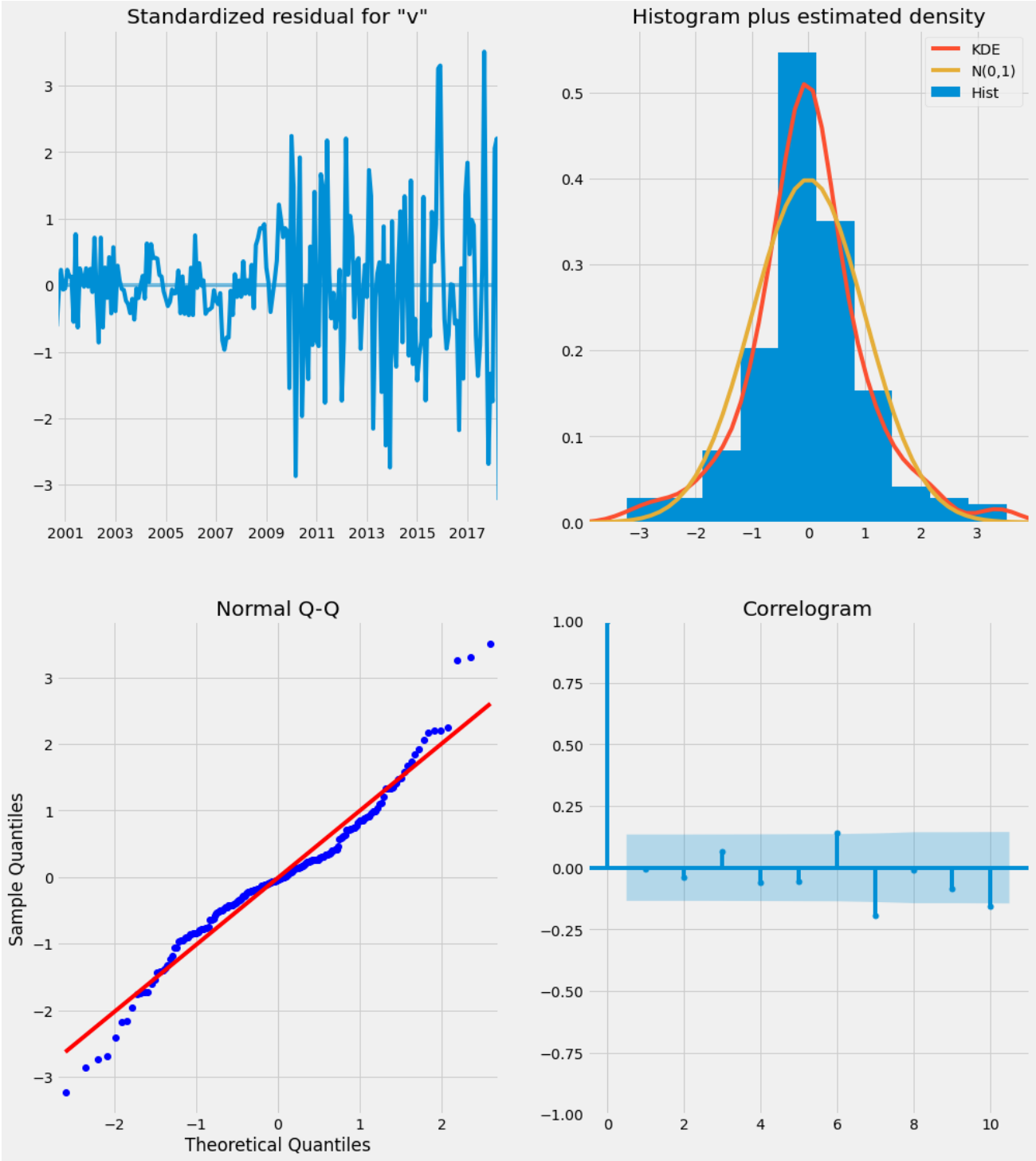
Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).

In [594]:
```
zip_98043_model_full = sarima_model
```

executed in 3ms, finished 14:26:18 2022-05-24

In [595]:
```
1 zip_98043_model_full.plot_diagnostics(figsize=(15,18))
2 plt.show()
```

executed in 364ms, finished 14:26:18 2022-05-24



In [596]:
```
zip_98043_ts.tail(1)
```

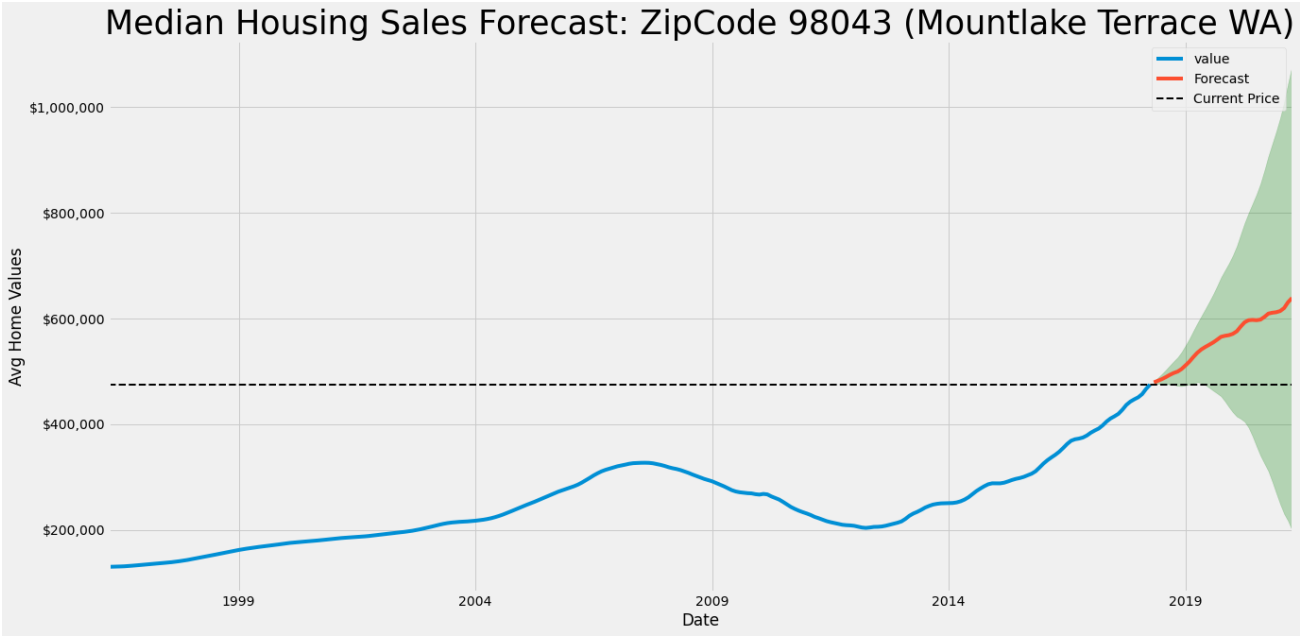
executed in 4ms, finished 14:26:18 2022-05-24

Out[596]:

time	value
2018-04-01	474,700.00

In [597]:
```
# Get forecast 3 yrs ahead in future (36 steps)
prediction = zip_98043_model_full.get_forecast(steps=36)

# Get confidence intervals of forecasts
pred_conf = prediction.conf_int()

# Plot future predictions with confidence intervals
ax = zip_98043_ts['1996-01':].plot(label='observed', figsize=(20, 10))
prediction.predicted_mean.plot(ax=ax, label='Forecast')
ax.fill_between(pred_conf.index,
                pred_conf.iloc[:, 0],
                pred_conf.iloc[:, 1], color='green', alpha=0.25)

ax.axhline(474700, ls='--', color='black', linewidth=(2), label='Current Price')

ax.set_xlabel('Date')
ax.set_ylabel('Avg Home Values')
ax.set_title("Median Housing Sales Forecast: ZipCode 98043 (Mountlake Terrace WA)", fontsize = 35)
ax.yaxis.set_major_formatter(tick)
plt.savefig('images/mountlake_forecast')

plt.legend()
plt.show()
```

executed in 255ms, finished 14:26:18 2022-05-24



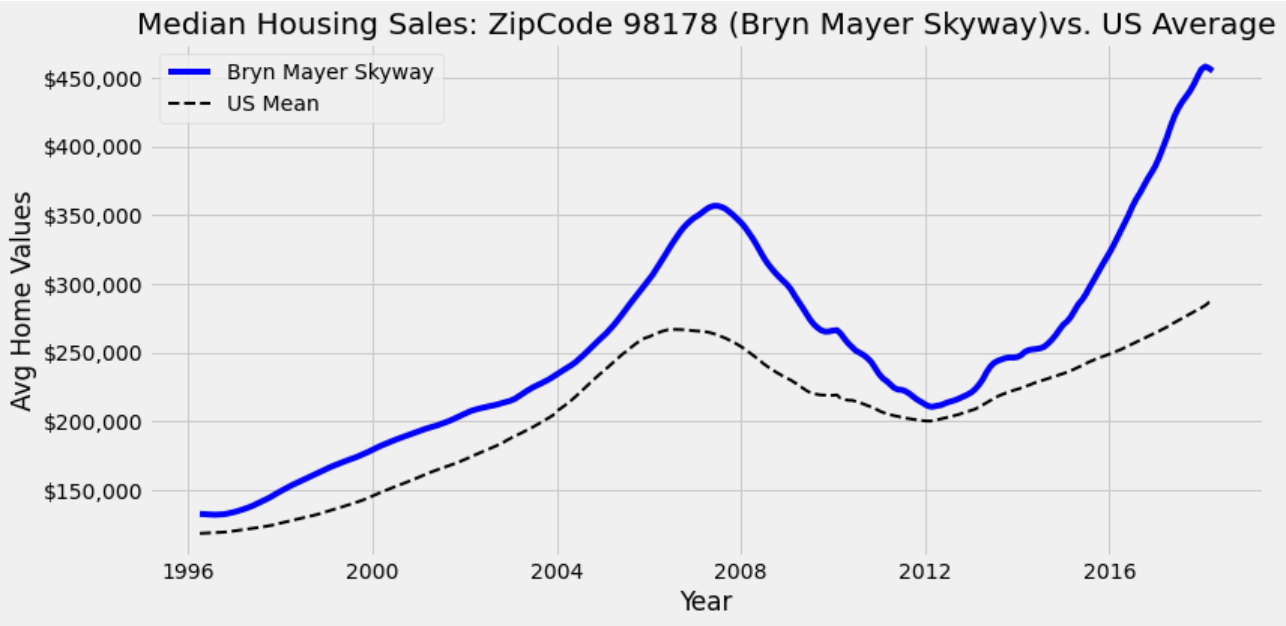
12.5 Analysis:

- No losses forecast for yr1, and minimal for year 2. Then the confidence window greatly widens and has a lot of uncertainty.
- **I recommend this as one of my alternative recommendations for investment with montoring between year 1-2.**

13 98178: Bryn Mewer-Skyway WA

In [598]:
```
fig, ax = plt.subplots(figsize=(12, 6))
ax.plot(zip_98178_ts, label='Bryn Mayer Skyway', color='blue')
line1 = sns.lineplot(data=avg_home_df, x='time', y='home_value', ls='--',
                     color='black', linewidth=(2), label='US Mean');

ax.legend()
ax.set_title("Median Housing Sales: ZipCode 98178 (Bryn Mayer Skyway)vs. US Average", fontsize = 20)
ax.set_xlabel('Year')
ax.set_ylabel('Avg Home Values')
ax.yaxis.set_major_formatter(tick)
fig.tight_layout()
```

executed in 176ms, finished 14:26:19 2022-05-24



In [599]:
```
#sarimax_param_search(zip_98178_ts)
```

executed in 2ms, finished 14:26:19 2022-05-24

In [600]:
```
#SARIMA Combos: ((1, 1, 1), (1, 1, 1, 12)) AIC: 3762.954984096742
zip_98178_combo_1 = ((1, 1, 1), (1, 1, 1, 12))
```

executed in 2ms, finished 14:26:19 2022-05-24

13.1 Model #1

In [601]:
```
combo = zip_98178_combo_1
combo
```

executed in 2ms, finished 14:26:19 2022-05-24

Out[601]:
```
((1, 1, 1), (1, 1, 1, 12))
```

In [602]:
```
# Manually split data.
temp_ts = zip_98178_ts

SPLIT = int(temp_ts.shape[0]*0.80)
train, test = temp_ts[:SPLIT], temp_ts[SPLIT:]

# Model
sarima_model = SARIMAX(
    train,
    order=combo[0],
    seasonal_order=combo[1],
    enforce_stationarity=False,
    enforce_invertibility=False).fit()
sarima_model.summary()
```

executed in 174ms, finished 14:26:19 2022-05-24

Out[602]:

SARIMAX Results

Dep. Variable:	value	**No. Observations:**	212
Model:	SARIMAX(1, 1, 1)x(1, 1, 1, 12)	**Log Likelihood**	-1438.599
Date:	Tue, 24 May 2022	**AIC**	2887.198
Time:	14:26:19	**BIC**	2903.300
Sample:	04-01-1996	**HQIC**	2893.723
	- 11-01-2013		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	0.9440	0.021	45.477	0.000	0.903	0.985
ma.L1	0.3234	0.037	8.635	0.000	0.250	0.397
ar.S.L12	-0.2794	0.073	-3.831	0.000	-0.422	-0.136
ma.S.L12	-0.0162	0.055	-0.295	0.768	-0.124	0.091
sigma2	3.356e+05	2.35e+04	14.272	0.000	2.9e+05	3.82e+05

Ljung-Box (L1) (Q):	8.18	**Jarque-Bera (JB):**	228.73
Prob(Q):	0.00	**Prob(JB):**	0.00
Heteroskedasticity (H):	14.61	**Skew:**	0.14
Prob(H) (two-sided):	0.00	**Kurtosis:**	8.44

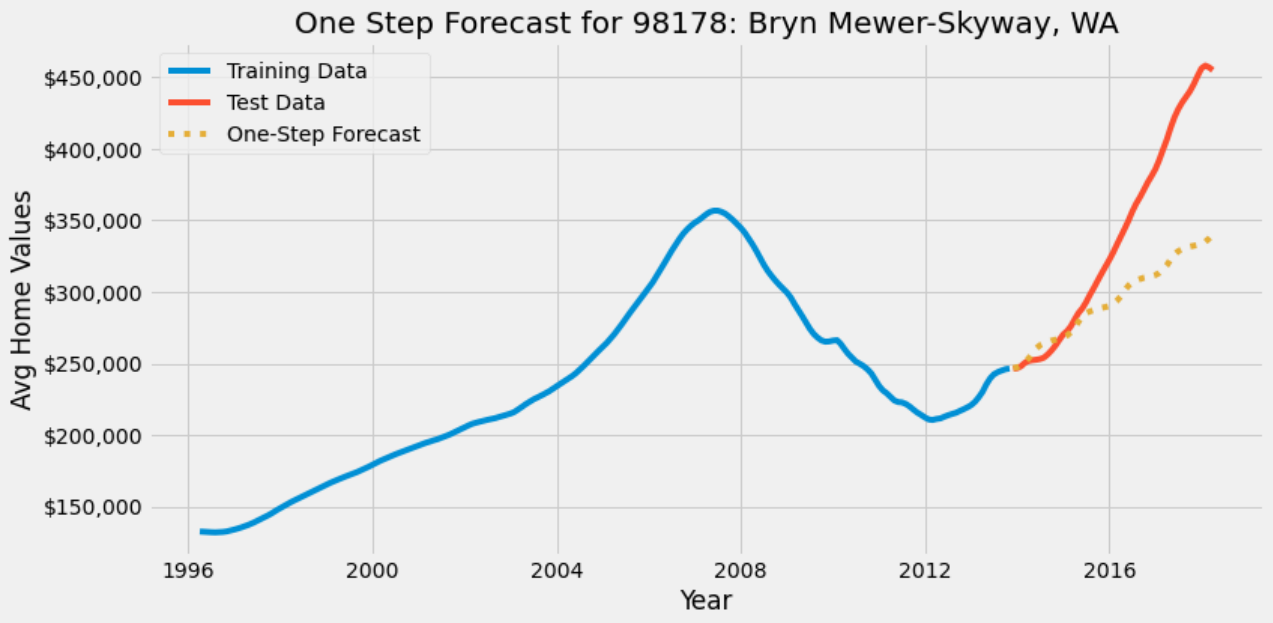
Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).

In [603]:
```
zip_98178_model_1 = sarima_model
```

executed in 4ms, finished 14:26:19 2022-05-24

In [604]:
```
pred = zip_98178_model_1.get_prediction(
    start=test.index.min(),
    end=test.index.max(),
    dynamic=True)

fig, ax = plt.subplots(figsize=(12, 6))
ax.plot(train, label='Training Data')
ax.plot(test, label='Test Data')
ax.plot(pred.predicted_mean, label='One-Step Forecast', ls=':')
#line1 = sns.lineplot(data=avg_home_df, x='time', y='home_value', color='black', label='US Mean');

ax.set_xlabel('Year')
ax.set_ylabel('Avg Home Values')
ax.set_title("One Step Forecast for 98178: Bryn Mewer-Skyway, WA", fontsize = 20)
ax.legend()
plt.savefig('images/bryn_mewer_one_step_1')
ax.yaxis.set_major_formatter(tick)
fig.tight_layout()
```

executed in 186ms, finished 14:26:19 2022-05-24



In [605]:
```
y_forecasted = pred.predicted_mean
y_truth = test['value']
mse = ((y_forecasted - y_truth) ** 2).mean()
print('The Mean Squared Error is {}'.format(round(mse, 2)))
print('The Root Mean Squared Error is {}'.format(round(np.sqrt(mse), 2)))
```

executed in 4ms, finished 14:26:19 2022-05-24

```
The Mean Squared Error is 3883551680.49
The Root Mean Squared Error is 62318.15
```

13.2 Model #2

In [606]:
```
#sarimax_param_search(zip_98178_ts)
```

executed in 2ms, finished 14:26:19 2022-05-24

In [607]:
```
#Optimal SARIMA order: ((2, 2, 2), (1, 2, 2, 12)) AIC: 3436.031197456773
zip_98178_combo_2 = ((2, 2, 2), (1, 2, 2, 12))
```

executed in 2ms, finished 14:26:19 2022-05-24

In [608]:
```
combo = zip_98178_combo_2
combo
```

executed in 3ms, finished 14:26:19 2022-05-24

Out[608]:
```
((2, 2, 2), (1, 2, 2, 12))
```

In [609]:
```
# Manually split data.
temp_ts = zip_98178_ts

SPLIT = int(temp_ts.shape[0]*0.80)
train, test = temp_ts[:SPLIT], temp_ts[SPLIT:]

# Model
sarima_model = SARIMAX(
    train,
    order=combo[0],
    seasonal_order=combo[1],
    enforce_stationarity=False,
    enforce_invertibility=False).fit()
sarima_model.summary()
```

executed in 6.23s, finished 14:26:25 2022-05-24

Out[609]:

SARIMAX Results

Dep. Variable:	value	**No. Observations:**	212
Model:	SARIMAX(2, 2, 2)x(1, 2, 2, 12)	**Log Likelihood**	-1288.019
Date:	Tue, 24 May 2022	**AIC**	2592.037
Time:	14:26:25	**BIC**	2616.588
Sample:	04-01-1996	**HQIC**	2602.007
	- 11-01-2013		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	-0.5286	0.340	-1.553	0.120	-1.196	0.138
ar.L2	0.0074	0.129	0.057	0.954	-0.245	0.260
ma.L1	1.2894	0.311	4.151	0.000	0.681	1.898
ma.L2	0.4993	0.153	3.272	0.001	0.200	0.798
ar.S.L12	-0.6354	0.105	-6.030	0.000	-0.842	-0.429
ma.S.L12	-0.3484	0.105	-3.317	0.001	-0.554	-0.143
ma.S.L24	0.1183	0.075	1.572	0.116	-0.029	0.266
sigma2	7.849e+05	7.01e+04	11.192	0.000	6.47e+05	9.22e+05

Ljung-Box (L1) (Q):	2.96	**Jarque-Bera (JB):**	165.74
Prob(Q):	0.09	**Prob(JB):**	0.00
Heteroskedasticity (H):	15.72	**Skew:**	-0.67
Prob(H) (two-sided):	0.00	**Kurtosis:**	7.82

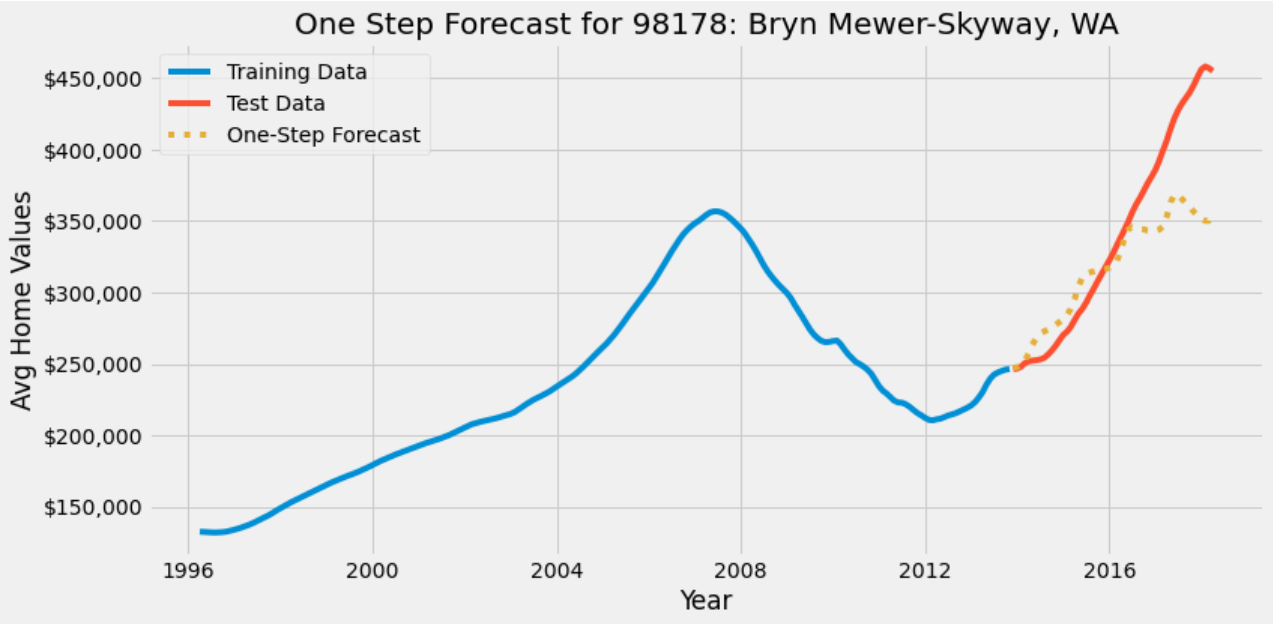
Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).

In [610]:
```
zip_98178_model_2 = sarima_model
```

executed in 4ms, finished 14:26:25 2022-05-24

In [611]:
```
pred = zip_98178_model_2.get_prediction(
    start=test.index.min(),
    end=test.index.max(),
    dynamic=True)

fig, ax = plt.subplots(figsize=(12, 6))
ax.plot(train, label='Training Data')
ax.plot(test, label='Test Data')
ax.plot(pred.predicted_mean, label='One-Step Forecast', ls=':')
#line1 = sns.lineplot(data=avg_home_df, x='time', y='home_value', color='black', label='US Mean');
ax.set_xlabel('Year')
ax.set_ylabel('Avg Home Values')
ax.set_title("One Step Forecast for 98178: Bryn Mewer-Skyway, WA", fontsize = 20)
ax.legend()
plt.savefig('images/bryn_mewer_one_step_2')
ax.yaxis.set_major_formatter(tick)
fig.tight_layout()
```

executed in 221ms, finished 14:26:25 2022-05-24



In [612]:
```
y_forecasted = pred.predicted_mean
y_truth = test['value']
mse = ((y_forecasted - y_truth) ** 2).mean()
print('The Mean Squared Error is {}'.format(round(mse, 2)))
print('The Root Mean Squared Error is {}'.format(round(np.sqrt(mse), 2)))
```

executed in 4ms, finished 14:26:25 2022-05-24

```
The Mean Squared Error is 1987691995.51
The Root Mean Squared Error is 44583.54
```

13.3 Model Comparison & Decision:

- Model 2 has a lower RSME and is a closer match for the test data.

13.4 Prediction

In [613]:
```
#combo = zip_98178_combo_1
combo = zip_98178_combo_2
combo
```

executed in 3ms, finished 14:26:25 2022-05-24

Out[613]: ((2, 2, 2), (1, 2, 2, 12))

In [614]:
```
# Running model again with FULL data

sarima_model = SARIMAX(
    zip_98178_ts,
    order=combo[0],
    seasonal_order=combo[1],
    enforce_stationarity=False,
    enforce_invertibility=False).fit()
sarima_model.summary()
```

executed in 8.47s, finished 14:26:34 2022-05-24

Out[614]:

SARIMAX Results

Dep. Variable:	value	**No. Observations:**	265
Model:	SARIMAX(2, 2, 2)x(1, 2, 2, 12)	**Log Likelihood**	-1710.016
Date:	Tue, 24 May 2022	**AIC**	3436.031
Time:	14:26:34	**BIC**	3462.884
Sample:	04-01-1996	**HQIC**	3446.884
	- 04-01-2018		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	0.2103	0.167	1.257	0.209	-0.118	0.538
ar.L2	-0.0241	0.045	-0.535	0.593	-0.112	0.064
ma.L1	0.3129	0.140	2.231	0.026	0.038	0.588
ma.L2	0.0462	0.089	0.517	0.605	-0.129	0.221
ar.S.L12	0.2883	0.023	12.636	0.000	0.244	0.333
ma.S.L12	-1.9804	0.061	-32.216	0.000	-2.101	-1.860
ma.S.L24	1.0102	0.072	14.071	0.000	0.870	1.151
sigma2	3.942e+05	3.09e-07	1.28e+12	0.000	3.94e+05	3.94e+05

Ljung-Box (L1) (Q):	2.79	**Jarque-Bera (JB):**	160.90
Prob(Q):	0.09	**Prob(JB):**	0.00
Heteroskedasticity (H):	7.25	**Skew:**	-0.95
Prob(H) (two-sided):	0.00	**Kurtosis:**	6.82

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
[2] Covariance matrix is singular or near-singular, with condition number 1.13e+28. Standard errors may be unstable.

In [615]:
```
zip_98178_model_full = sarima_model
```

executed in 5ms, finished 14:26:34 2022-05-24

In [616]:
```
1 zip_98178_ts.tail(1)
```

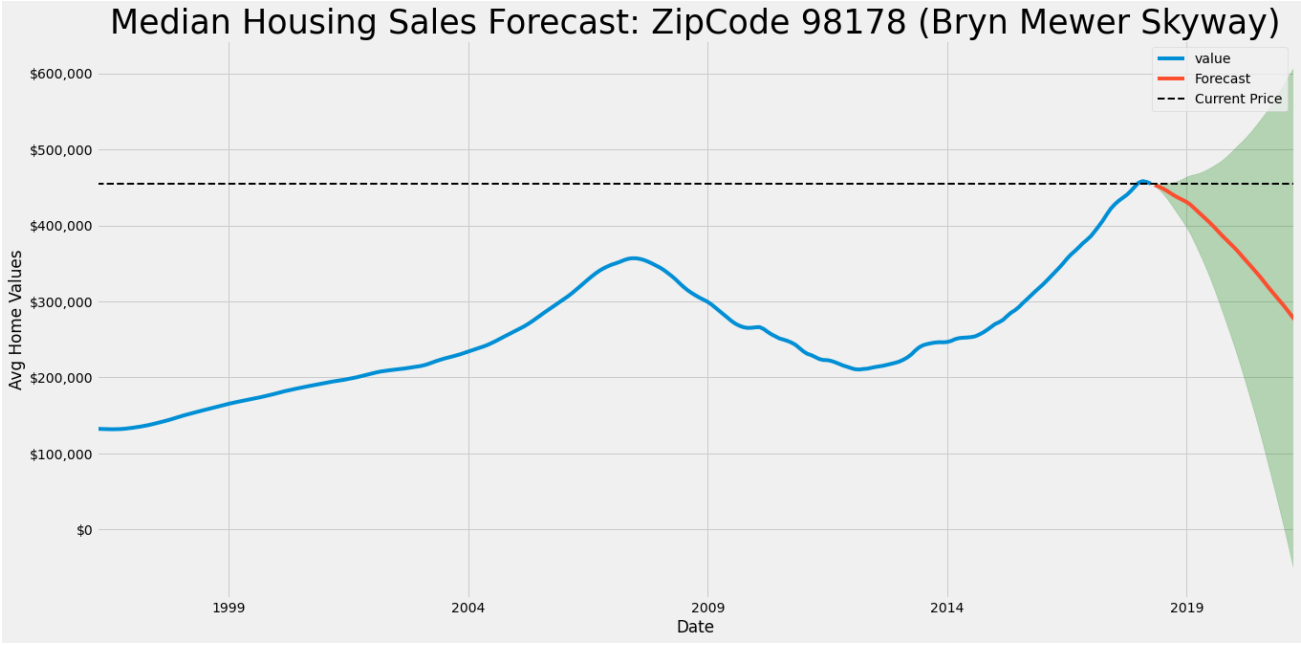
executed in 5ms, finished 14:26:34 2022-05-24

Out[616]:

time	value
2018-04-01	455,100.00

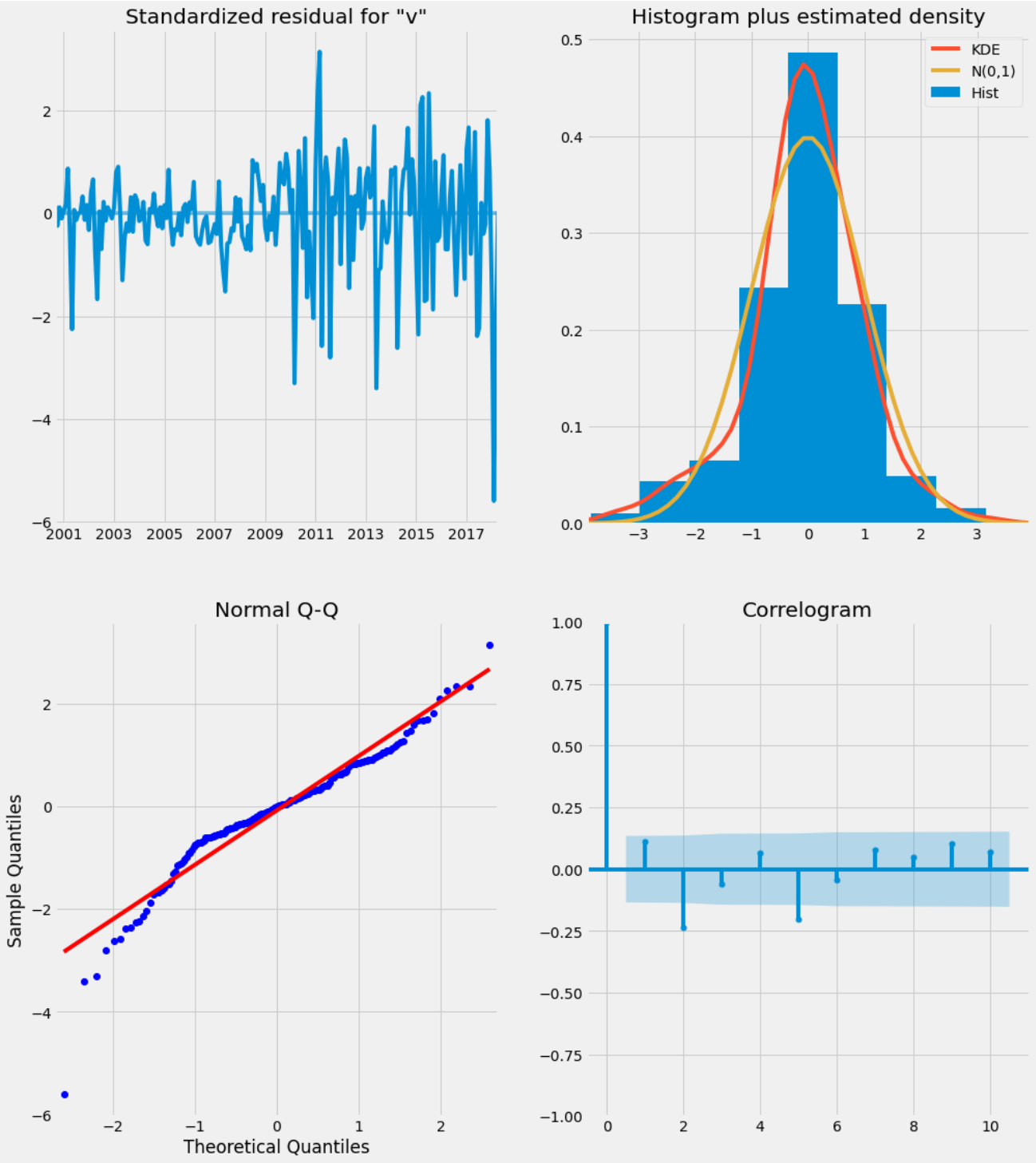
In [617]:
```
1  # Get forecast 3 yrs ahead in future (36 steps)
2  prediction = zip_98178_model_full.get_forecast(steps=36)
3
4  # Get confidence intervals of forecasts
5  pred_conf = prediction.conf_int()
6
7  # Plot future predictions with confidence intervals
8  ax = zip_98178_ts['1996-01':].plot(label='observed', figsize=(20, 10))
9  prediction.predicted_mean.plot(ax=ax, label='Forecast')
10 ax.fill_between(pred_conf.index,
11                 pred_conf.iloc[:, 0],
12                 pred_conf.iloc[:, 1], color='green', alpha=0.25)
13
14 ax.axhline(455100, ls='--', color='black', linewidth=(2), label='Current Price')
15
16 ax.set_xlabel('Date')
17 ax.set_ylabel('Avg Home Values')
18 ax.set_title("Median Housing Sales Forecast: ZipCode 98178 (Bryn Mewer Skyway)", fontsize = 35)
19 ax.yaxis.set_major_formatter(tick)
20 plt.savefig('images/bryn_mewer_forecast')
21
22 plt.legend()
23 plt.show()
```

executed in 313ms, finished 14:26:34 2022-05-24



In [618]:
```
zip_98178_model_full.plot_diagnostics(figsize=(15,18))
plt.show()
```

executed in 360ms, finished 14:26:35 2022-05-24



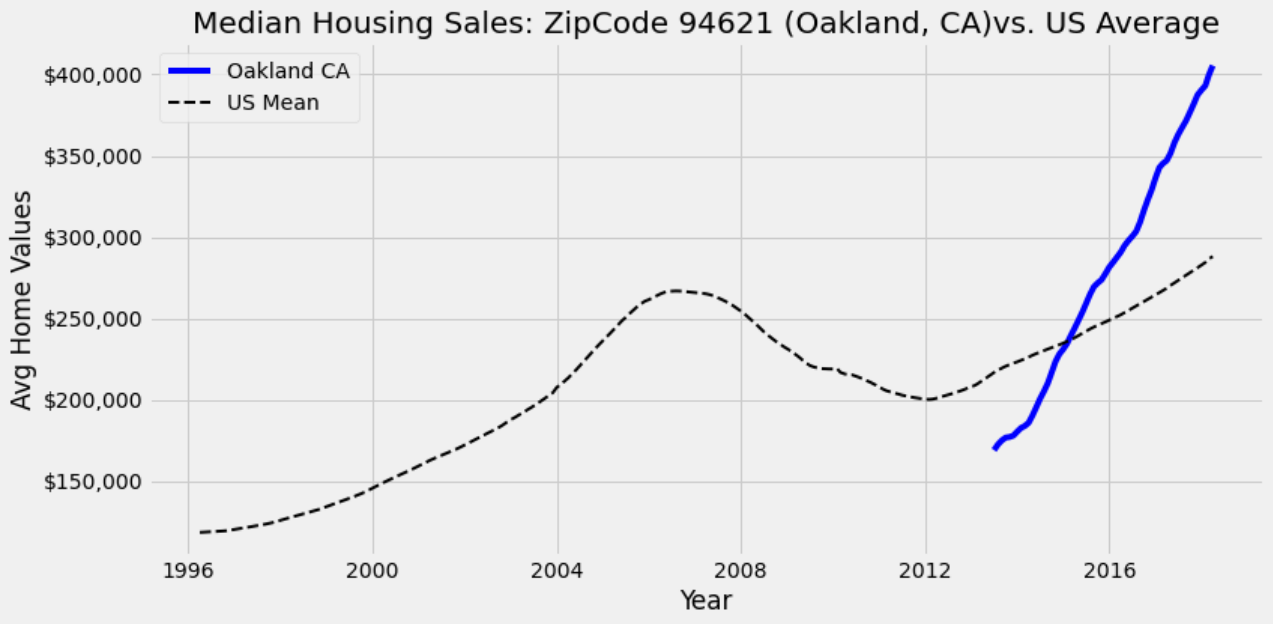
13.5 Analysis:

- Model forecast mean predicts increasing losses.
- The confidence interval doesn't show a lot of optimism for profits.
- **I do not recommend investing in this zipcode in any form.**

14 94621: Oakland CA

In [619]:
```
fig, ax = plt.subplots(figsize=(12, 6))
ax.plot(zip_94621_ts, label='Oakland CA', color='blue')
line1 = sns.lineplot(data=avg_home_df, x='time', y='home_value', ls='--',
                     color='black', linewidth=(2), label='US Mean');

ax.legend()
ax.set_title("Median Housing Sales: ZipCode 94621 (Oakland, CA)vs. US Average", fontsize = 20)
ax.set_xlabel('Year')
ax.set_ylabel('Avg Home Values')
ax.yaxis.set_major_formatter(tick)
fig.tight_layout()
```

executed in 173ms, finished 14:26:35 2022-05-24



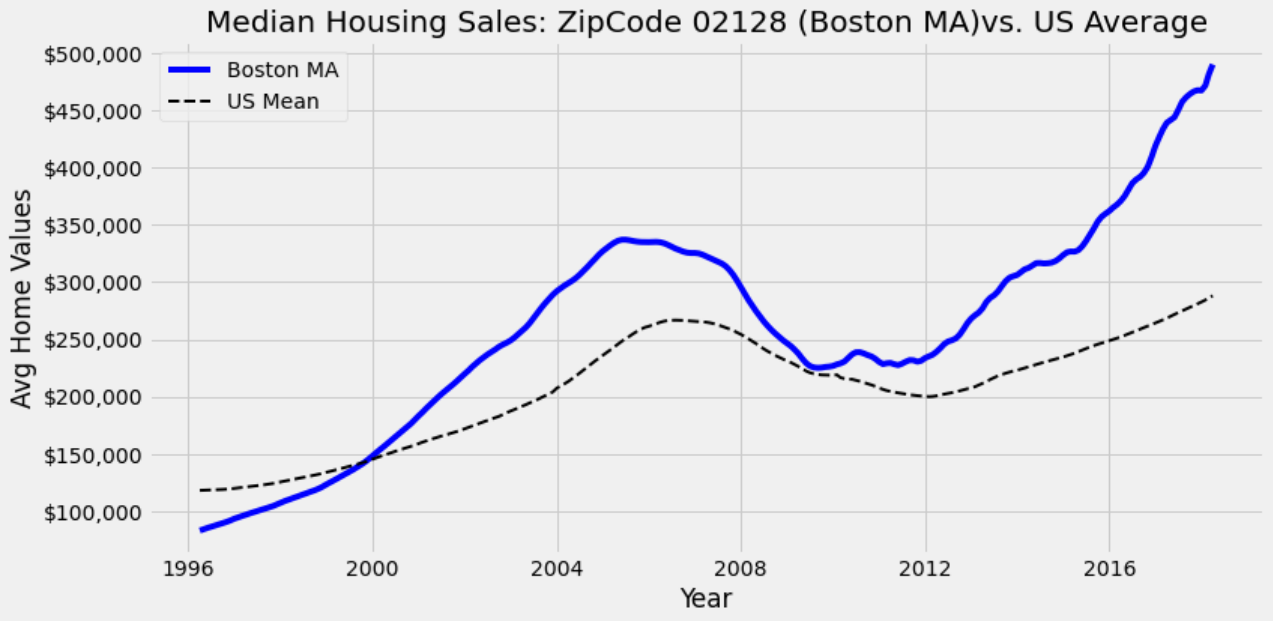
14.1 Analysis: SKIP. Not Enough Data to Model

- This must be a new ZipCode. There isn't enough data for me to be comfortable modeling, so I am going to skip this Zip Code and move on to the next one.
- Since I am looking for a maximum of one Zip Code per state, this isn't a big deal.
- **Skip this Zip Code and move on to the next one.**

15 02128 Boston MA

In [620]:
```
fig, ax = plt.subplots(figsize=(12, 6))
ax.plot(zip_02128_ts, label='Boston MA', color='blue')
line1 = sns.lineplot(data=avg_home_df, x='time', y='home_value', ls='--',
                     color='black', linewidth=(2), label='US Mean');

ax.legend()
ax.set_title("Median Housing Sales: ZipCode 02128 (Boston MA)vs. US Average", fontsize = 20)
ax.set_xlabel('Year')
ax.set_ylabel('Avg Home Values')
ax.yaxis.set_major_formatter(tick)
fig.tight_layout()
```

executed in 177ms, finished 14:26:35 2022-05-24



In [621]:
```
#sarimax_param_search(zip_02128_ts)
```

executed in 1ms, finished 14:26:35 2022-05-24

In [622]:
```
#SARIMA Combos: ((1, 1, 1), (1, 1, 1, 12)) AIC: 3948.4693734854354
zip_02128_combo_1 = ((1, 1, 1), (1, 1, 1, 12))
```

executed in 2ms, finished 14:26:35 2022-05-24

15.1 Model #1

In [623]:
```
combo = zip_02128_combo_1
#combo = zip_98403_combo_2
combo
```

executed in 3ms, finished 14:26:35 2022-05-24

Out[623]:
```
((1, 1, 1), (1, 1, 1, 12))
```

In [624]:
```
# Manually split data.
temp_ts = zip_02128_ts

SPLIT = int(temp_ts.shape[0]*0.80)
train, test = temp_ts[:SPLIT], temp_ts[SPLIT:]

# Model
sarima_model = SARIMAX(
    train,
    order=combo[0],
    seasonal_order=combo[1],
    enforce_stationarity=False,
    enforce_invertibility=False).fit()
sarima_model.summary()
```

executed in 634ms, finished 14:26:36 2022-05-24

Out[624]:

SARIMAX Results

Dep. Variable:	value	**No. Observations:**	212
Model:	SARIMAX(1, 1, 1)x(1, 1, 1, 12)	**Log Likelihood**	-1457.125
Date:	Tue, 24 May 2022	**AIC**	2924.251
Time:	14:26:36	**BIC**	2940.353
Sample:	04-01-1996	**HQIC**	2930.777
	- 11-01-2013		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	0.9244	0.025	36.850	0.000	0.875	0.974
ma.L1	0.7032	0.035	19.912	0.000	0.634	0.772
ar.S.L12	0.0726	0.036	2.042	0.041	0.003	0.142
ma.S.L12	-0.8359	0.054	-15.382	0.000	-0.942	-0.729
sigma2	3.09e+05	2.58e+04	11.997	0.000	2.59e+05	3.6e+05

Ljung-Box (L1) (Q):	1.26	**Jarque-Bera (JB):**	64.63
Prob(Q):	0.26	**Prob(JB):**	0.00
Heteroskedasticity (H):	7.25	**Skew:**	0.24
Prob(H) (two-sided):	0.00	**Kurtosis:**	5.85

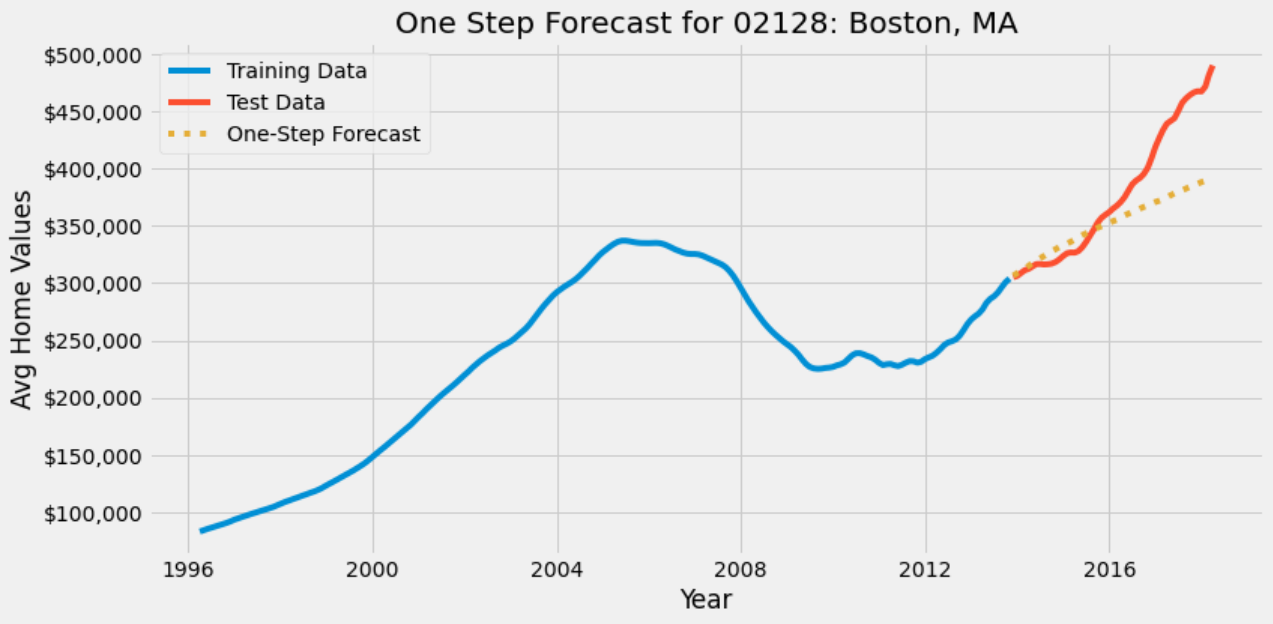
Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).

In [625]:
```
zip_02128_model_1 = sarima_model
```

executed in 3ms, finished 14:26:36 2022-05-24

In [626]:
```
pred = zip_02128_model_1.get_prediction(
    start=test.index.min(),
    end=test.index.max(),
    dynamic=True)

fig, ax = plt.subplots(figsize=(12, 6))
ax.plot(train, label='Training Data')
ax.plot(test, label='Test Data')
ax.plot(pred.predicted_mean, label='One-Step Forecast', ls=':')
#line1 = sns.lineplot(data=avg_home_df, x='time', y='home_value', color='black', label='US Mean');
ax.set_xlabel('Year')
ax.set_ylabel('Avg Home Values')
ax.set_title("One Step Forecast for 02128: Boston, MA", fontsize = 20)
ax.legend()
plt.savefig('images/boston_one_step_1')
ax.yaxis.set_major_formatter(tick)
fig.tight_layout()
```

executed in 211ms, finished 14:26:36 2022-05-24



In [627]:
```
y_forecasted = pred.predicted_mean
y_truth = test['value']
mse = ((y_forecasted - y_truth) ** 2).mean()
print('The Mean Squared Error is {}'.format(round(mse, 2)))
print('The Root Mean Squared Error is {}'.format(round(np.sqrt(mse), 2)))
```

executed in 4ms, finished 14:26:36 2022-05-24

```
The Mean Squared Error is 1829510970.86
The Root Mean Squared Error is 42772.78
```

15.2 Model #2

In [628]:
```
#SARIMA Combos: ((2, 2, 2), (0, 2, 2, 12)) AIC: 3572.582171627717
zip_02128_combo_2 = ((2, 2, 2), (0, 2, 2, 12))
```

executed in 2ms, finished 14:26:36 2022-05-24

In [629]:
```
combo = zip_02128_combo_2
combo
```

executed in 3ms, finished 14:26:36 2022-05-24

Out[629]:
```
((2, 2, 2), (0, 2, 2, 12))
```

In [630]:
```
# Manually split data.
temp_ts = zip_02128_ts

SPLIT = int(temp_ts.shape[0]*0.80)
train, test = temp_ts[:SPLIT], temp_ts[SPLIT:]

# Model
sarima_model = SARIMAX(
    train,
    order=combo[0],
    seasonal_order=combo[1],
    enforce_stationarity=False,
    enforce_invertibility=False).fit()
sarima_model.summary()
```

executed in 5.95s, finished 14:26:42 2022-05-24

Out[630]:

SARIMAX Results

Dep. Variable:	value	**No. Observations:**	212
Model:	SARIMAX(2, 2, 2)x(0, 2, 2, 12)	**Log Likelihood**	-1322.603
Date:	Tue, 24 May 2022	**AIC**	2659.205
Time:	14:26:42	**BIC**	2680.688
Sample:	04-01-1996	**HQIC**	2667.929
	- 11-01-2013		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	-0.8227	0.180	-4.565	0.000	-1.176	-0.469
ar.L2	-0.2955	0.097	-3.058	0.002	-0.485	-0.106
ma.L1	1.7453	0.081	21.455	0.000	1.586	1.905
ma.L2	0.9452	0.084	11.271	0.000	0.781	1.110
ma.S.L12	-0.9155	0.160	-5.707	0.000	-1.230	-0.601
ma.S.L24	0.2140	0.133	1.605	0.109	-0.047	0.475
sigma2	1.58e+06	2.65e+05	5.964	0.000	1.06e+06	2.1e+06

Ljung-Box (L1) (Q):	1.29	**Jarque-Bera (JB):**	58.28
Prob(Q):	0.26	**Prob(JB):**	0.00
Heteroskedasticity (H):	5.62	**Skew:**	0.26
Prob(H) (two-sided):	0.00	**Kurtosis:**	5.92

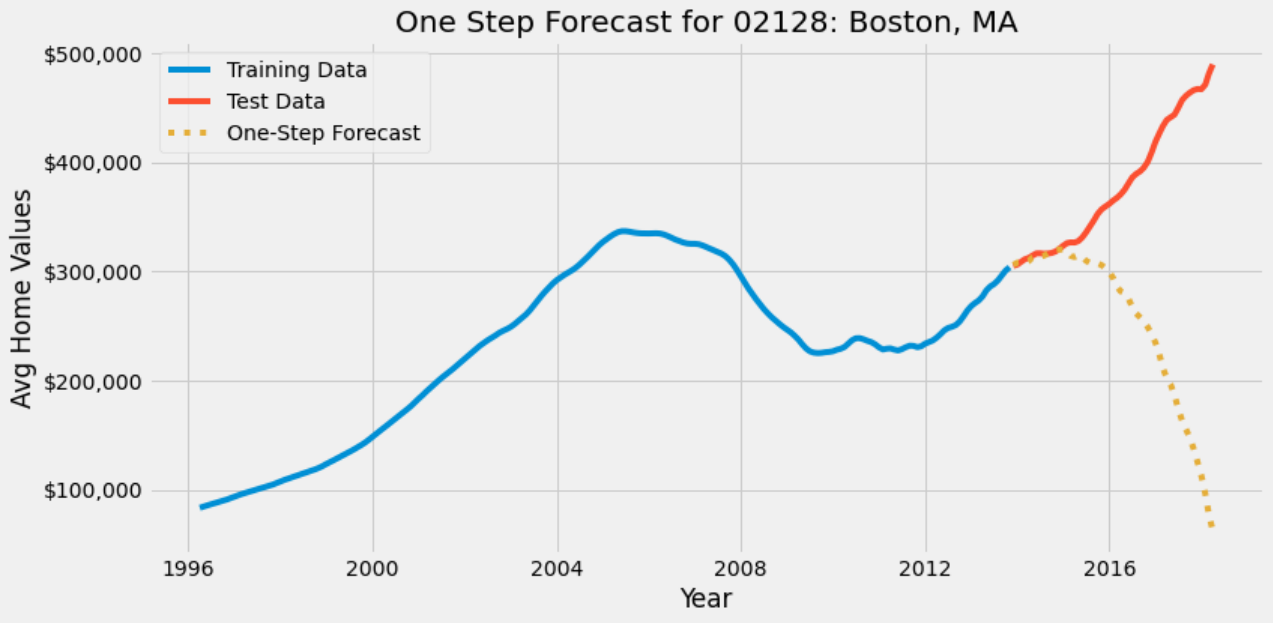
Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).

In [631]:
```
zip_02128_model_2 = sarima_model
```

executed in 2ms, finished 14:26:42 2022-05-24

In [632]:
```
pred = zip_02128_model_2.get_prediction(
    start=test.index.min(),
    end=test.index.max(),
    dynamic=True)

fig, ax = plt.subplots(figsize=(12, 6))
ax.plot(train, label='Training Data')
ax.plot(test, label='Test Data')
ax.plot(pred.predicted_mean, label='One-Step Forecast', ls=':')
#line1 = sns.lineplot(data=avg_home_df, x='time', y='home_value', color='black', label='US Mean');

ax.set_xlabel('Year')
ax.set_ylabel('Avg Home Values')
ax.set_title("One Step Forecast for 02128: Boston, MA", fontsize = 20)
ax.legend()
plt.savefig('images/boston_one_step_2')
ax.yaxis.set_major_formatter(tick)
fig.tight_layout()
```

executed in 207ms, finished 14:26:42 2022-05-24



In [633]:
```
y_forecasted = pred.predicted_mean
y_truth = test['value']
mse = ((y_forecasted - y_truth) ** 2).mean()
print('The Mean Squared Error is {}'.format(round(mse, 2)))
print('The Root Mean Squared Error is {}'.format(round(np.sqrt(mse), 2)))
```

executed in 3ms, finished 14:26:42 2022-05-24

```
The Mean Squared Error is 31606888091.33
The Root Mean Squared Error is 177783.26
```

15.3 Model Comparison & Decision:

- Model 2's Forecast doesn't follow the Test data at all, and it has a significantly higher RMSE so **I am rejecting Model #2 and sticking with model #1 for Forecasting.**

15.4 Boston Prediction

In [634]:
```
zip_02128_combo_1 = ((1, 1, 1), (1, 1, 1, 12))
```

executed in 2ms, finished 14:26:42 2022-05-24

In [635]:
```
combo = zip_02128_combo_1
```
executed in 2ms, finished 14:26:42 2022-05-24

In [636]:
```
combo
```
executed in 2ms, finished 14:26:42 2022-05-24

Out[636]:
```
((1, 1, 1), (1, 1, 1, 12))
```

In [637]:
```
# Running model again with FULL data

sarima_model = SARIMAX(
    zip_02128_ts,
    order=combo[0],
    seasonal_order=combo[1],
    enforce_stationarity=False,
    enforce_invertibility=False).fit()
sarima_model.summary()
```
executed in 786ms, finished 14:26:43 2022-05-24

Out[637]:

SARIMAX Results

Dep. Variable:	value	**No. Observations:**	265
Model:	SARIMAX(1, 1, 1)x(1, 1, 1, 12)	**Log Likelihood**	-1969.235
Date:	Tue, 24 May 2022	**AIC**	3948.469
Time:	14:26:43	**BIC**	3965.831
Sample:	04-01-1996	**HQIC**	3955.466
	- 04-01-2018		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	0.7926	0.032	25.117	0.000	0.731	0.854
ma.L1	0.7739	0.028	27.686	0.000	0.719	0.829
ar.S.L12	0.0897	0.062	1.435	0.151	-0.033	0.212
ma.S.L12	-0.6286	0.049	-12.761	0.000	-0.725	-0.532
sigma2	8.071e+05	4.64e+04	17.388	0.000	7.16e+05	8.98e+05

Ljung-Box (L1) (Q):	1.53	**Jarque-Bera (JB):**	230.85
Prob(Q):	0.22	**Prob(JB):**	0.00
Heteroskedasticity (H):	10.55	**Skew:**	1.01
Prob(H) (two-sided):	0.00	**Kurtosis:**	7.38

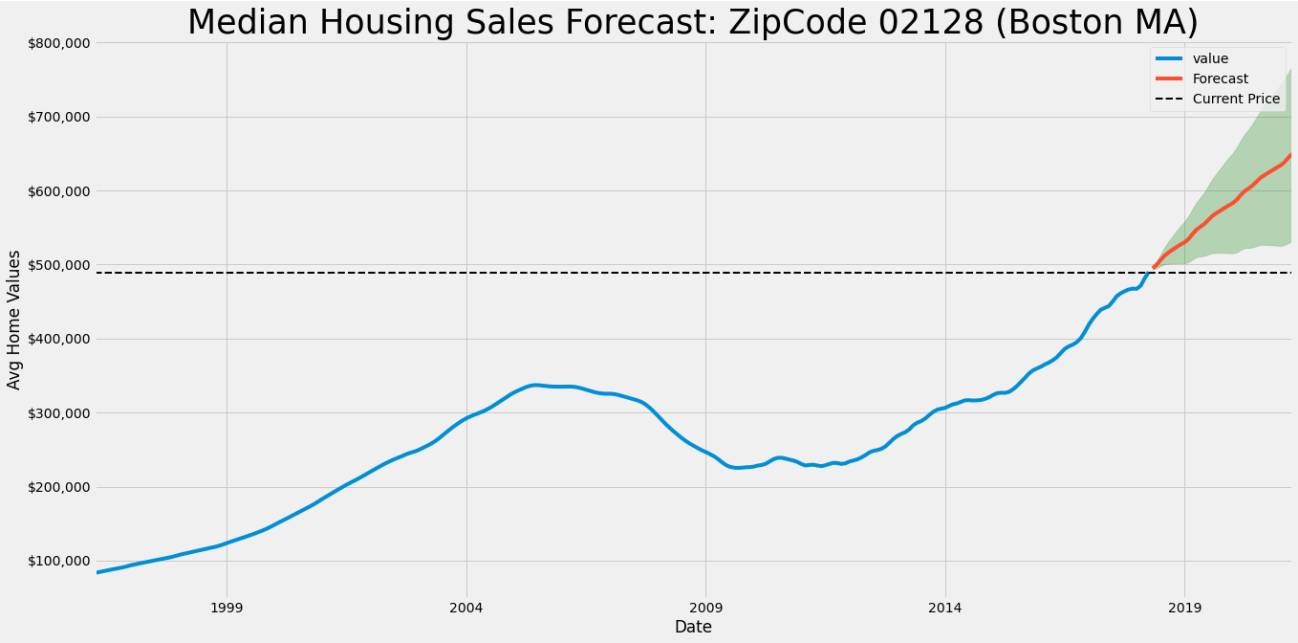
Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).

In [638]:
```
zip_02128_model_full = sarima_model
```
executed in 3ms, finished 14:26:43 2022-05-24

In [639]:

```
# Get forecast 3 yrs ahead in future (36 steps)
prediction = zip_02128_model_full.get_forecast(steps=36)

# Get confidence intervals of forecasts
pred_conf = prediction.conf_int()

# Plot future predictions with confidence intervals
ax = zip_02128_ts['1996-01':].plot(label='observed', figsize=(20, 10))
prediction.predicted_mean.plot(ax=ax, label='Forecast')
ax.fill_between(pred_conf.index,
                pred_conf.iloc[:, 0],
                pred_conf.iloc[:, 1], color='green', alpha=0.25)

ax.axhline(489400, ls='--', color='black', linewidth=(2), label='Current Price')

ax.set_xlabel('Date')
ax.set_ylabel('Avg Home Values')
ax.set_title("Median Housing Sales Forecast: ZipCode 02128 (Boston MA)", fontsize = 35)
ax.yaxis.set_major_formatter(tick)
plt.savefig('images/boston_forecast')

plt.legend()
plt.show()
```

executed in 273ms, finished 14:26:43 2022-05-24



15.5 Analysis:

- The Confidence window is narrow and even the lower forecasted values are gains.
- **This Zip Code is a strong recommendation for investment!**

15.6 Metrics

- capturing metrics to compare with the top zip codes later.

In [640]:
```
analysis_df = pred_conf.resample('Y').mean()
```
executed in 4ms, finished 14:26:43 2022-05-24

In [641]:
```
analysis_df['mean'] = analysis_df.apply(lambda x: x.mean(), axis=1)
```
executed in 4ms, finished 14:26:43 2022-05-24

In [642]:
```
analysis_df = pred_conf.resample('Y').mean()
analysis_df['mean'] = analysis_df.apply(lambda x: x.mean(), axis=1)

analysis_df['base'] = 489900
analysis_df['base'] = analysis_df['base'].astype(float)

analysis_df = analysis_df[['base', 'lower value', 'mean', 'upper value']]

analysis_df['Min_ROI'] = analysis_df.apply(lambda x: x['lower value'] - x['base'], axis=1)
analysis_df['Min_ROI%'] = analysis_df.apply(lambda x: x['Min_ROI'] / x['base'], axis=1)
analysis_df['Forecast_ROI'] = analysis_df.apply(lambda x: x['mean'] - x['base'], axis=1)
analysis_df['Forecast_ROI%'] = analysis_df.apply(lambda x: x['mean'] / x['base'], axis=1)
analysis_df['Max_ROI'] = analysis_df.apply(lambda x: x['upper value'] - x['base'], axis=1)
analysis_df['Max_ROI%'] = analysis_df.apply(lambda x: x['upper value'] / x['base'], axis=1)
```
executed in 14ms, finished 14:26:43 2022-05-24

In [643]:
```
zip_02128_ts.tail(1)
```
executed in 4ms, finished 14:26:43 2022-05-24

Out[643]:

time	value
2018-04-01	489,900.00

In [644]:
```
zip_02128_metrics = analysis_df
```
executed in 2ms, finished 14:26:43 2022-05-24

In [645]:
```
zip_02128_metrics
```
executed in 6ms, finished 14:26:43 2022-05-24

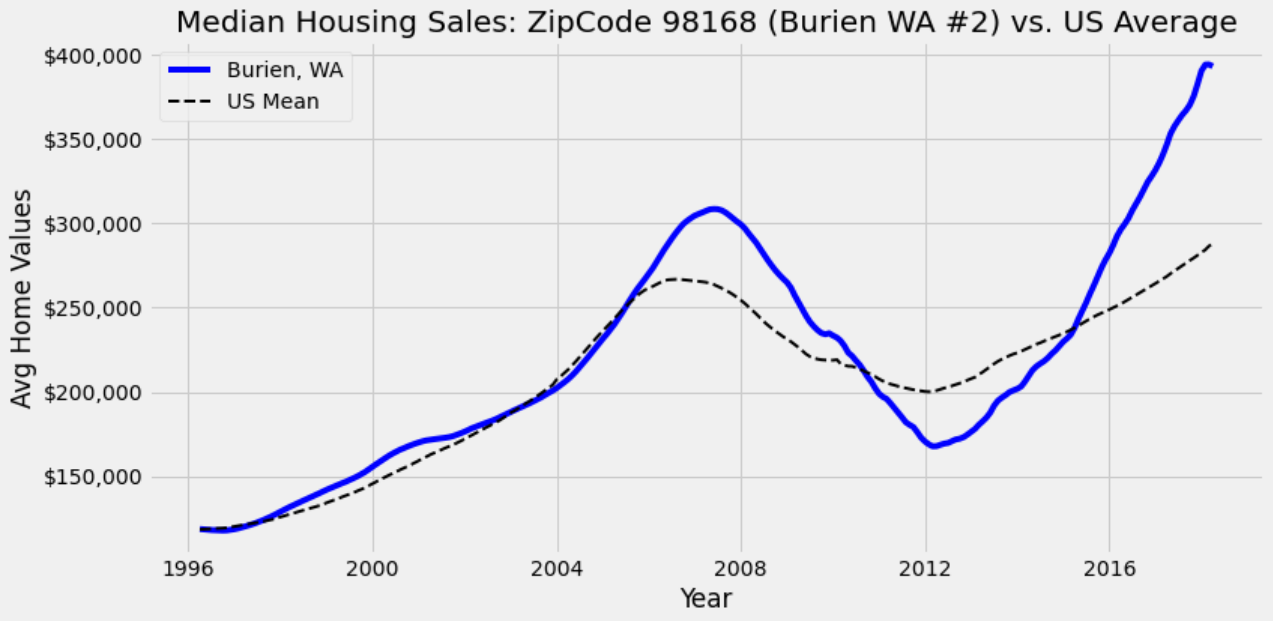
Out[645]:

	base	lower value	mean	upper value	Min_ROI	Min_ROI%	Forecast_ROI	Forecast_ROI%	Max_ROI	Max_ROI%
2018-12-31	489,900.00	498,942.37	512,514.32	526,086.27	9,042.37	0.02	22,614.32	1.05	36,186.27	1.07
2019-12-31	489,900.00	511,264.16	557,030.40	602,796.64	21,364.16	0.04	67,130.40	1.14	112,896.64	1.23
2020-12-31	489,900.00	522,757.43	608,620.65	694,483.87	32,857.43	0.07	118,720.65	1.24	204,583.87	1.42
2021-12-31	489,900.00	527,880.09	641,545.46	755,210.84	37,980.09	0.08	151,645.46	1.31	265,310.84	1.54

16 98168: Burien WA #2

In [646]:
```
fig, ax = plt.subplots(figsize=(12, 6))
ax.plot(zip_98168_ts, label='Burien, WA', color='blue')


line1 = sns.lineplot(data=avg_home_df, x='time', y='home_value', ls='--',
                     color='black', linewidth=(2), label='US Mean');
ax.set_xlabel('Year')
ax.set_ylabel('Avg Home Values')
ax.set_title("Median Housing Sales: ZipCode 98168 (Burien WA #2) vs. US Average", fontsize = 20)
ax.yaxis.set_major_formatter(tick)
fig.tight_layout()
```

executed in 165ms, finished 14:26:43 2022-05-24



In [647]:
```
#sarimax_param_search(zip_98168_ts)
```

executed in 2ms, finished 14:26:43 2022-05-24

In [648]:
```
#SARIMA Combos: ((1, 1, 1), (0, 1, 1, 12)) AIC: 3853.8875856880986
zip_98168_combo_1 = ((1, 1, 1), (0, 1, 1, 12))
```

executed in 2ms, finished 14:26:43 2022-05-24

In [649]:
```
combo= zip_98168_combo_1
combo
```

executed in 2ms, finished 14:26:43 2022-05-24

Out[649]:
```
((1, 1, 1), (0, 1, 1, 12))
```

16.1 Model #1

In [650]:
```
# Manually split data.
temp_ts = zip_98168_ts

SPLIT = int(temp_ts.shape[0]*0.80)
train, test = temp_ts[:SPLIT], temp_ts[SPLIT:]

# Model
sarima_model = SARIMAX(
    train,
    order=combo[0],
    seasonal_order=combo[1],
    enforce_stationarity=False,
    enforce_invertibility=False).fit()
sarima_model.summary()
```

executed in 560ms, finished 14:26:44 2022-05-24

Out[650]:

SARIMAX Results

Dep. Variable:	value	**No. Observations:**	212
Model:	SARIMAX(1, 1, 1)x(0, 1, 1, 12)	**Log Likelihood**	-1450.847
Date:	Tue, 24 May 2022	**AIC**	2909.694
Time:	14:26:44	**BIC**	2922.576
Sample:	04-01-1996	**HQIC**	2914.915
	- 11-01-2013		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	0.9410	0.025	37.320	0.000	0.892	0.990
ma.L1	0.1632	0.048	3.367	0.001	0.068	0.258
ma.S.L12	-0.4180	0.033	-12.680	0.000	-0.483	-0.353
sigma2	3.382e+05	2.16e+04	15.683	0.000	2.96e+05	3.81e+05

Ljung-Box (L1) (Q):	0.54	**Jarque-Bera (JB):**	117.06
Prob(Q):	0.46	**Prob(JB):**	0.00
Heteroskedasticity (H):	11.64	**Skew:**	0.28
Prob(H) (two-sided):	0.00	**Kurtosis:**	6.86

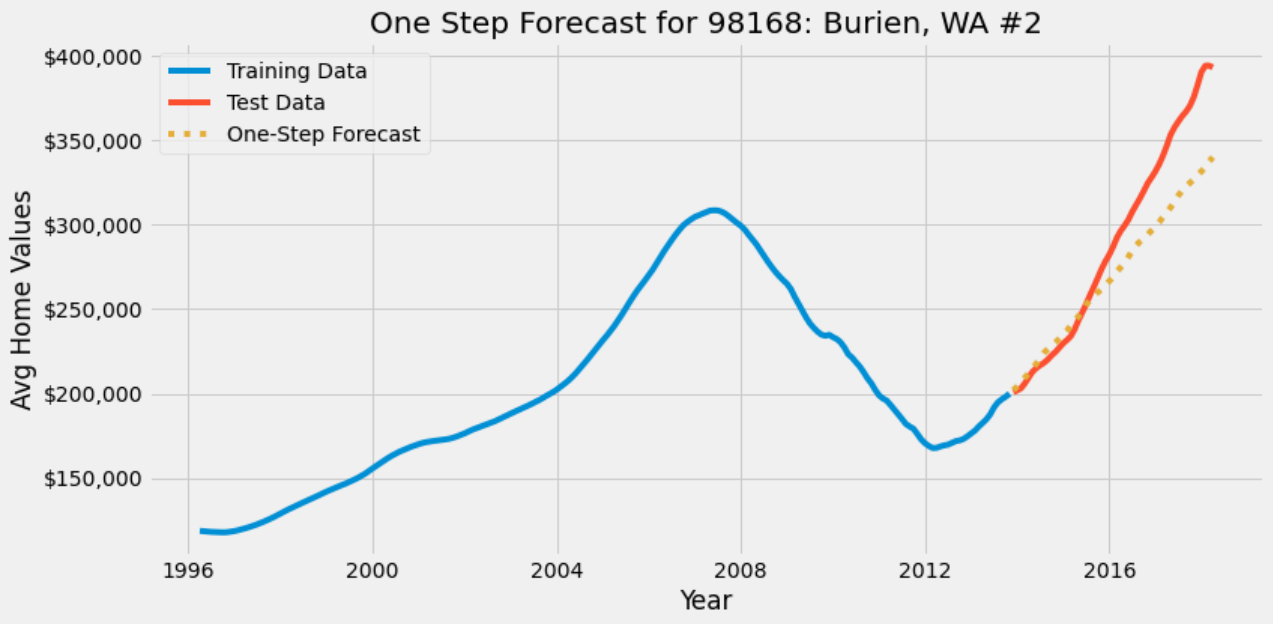
Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).

In [651]:
```
zip_98168_model_1 = sarima_model
```

executed in 25ms, finished 14:26:44 2022-05-24

In [652]:
```
pred = zip_98168_model_1.get_prediction(
    start=test.index.min(),
    end=test.index.max(),
    dynamic=True)

fig, ax = plt.subplots(figsize=(12, 6))
ax.plot(train, label='Training Data')
ax.plot(test, label='Test Data')
ax.plot(pred.predicted_mean, label='One-Step Forecast', ls=':')
#line1 = sns.lineplot(data=avg_home_df, x='time', y='home_value', color='black', label='US Mean');

ax.set_xlabel('Year')
ax.set_ylabel('Avg Home Values')
ax.set_title("One Step Forecast for 98168: Burien, WA #2", fontsize = 20)
ax.legend()
plt.savefig('images/burien2_one_step_1')
ax.yaxis.set_major_formatter(tick)
fig.tight_layout()
```

executed in 296ms, finished 14:26:44 2022-05-24



In [653]:
```
y_forecasted = pred.predicted_mean
y_truth = test['value']
mse = ((y_forecasted - y_truth) ** 2).mean()
print('The Mean Squared Error is {}'.format(round(mse, 2)))
print('The Root Mean Squared Error is {}'.format(round(np.sqrt(mse), 2)))
```

executed in 18ms, finished 14:26:44 2022-05-24

```
The Mean Squared Error is 809682182.79
The Root Mean Squared Error is 28454.91
```

16.2 Model #2

In [654]:
```
#sarimax_param_search(zip_98168_ts)
```
executed in 8ms, finished 14:26:44 2022-05-24

In [655]:
```
#SARIMA Combos: ((0, 2, 2), (1, 2, 2, 12)) AIC: 3499.2603303107535
zip_98168_combo_2 = ((0, 2, 2), (1, 2, 2, 12))
```
executed in 3ms, finished 14:26:44 2022-05-24

In [656]:
```
combo = zip_98168_combo_2
combo
```
executed in 3ms, finished 14:26:44 2022-05-24

Out[656]:
```
((0, 2, 2), (1, 2, 2, 12))
```

In [657]:
```
# Manually split data.
temp_ts = zip_98168_ts

SPLIT = int(temp_ts.shape[0]*0.80)
train, test = temp_ts[:SPLIT], temp_ts[SPLIT:]

# Model
sarima_model = SARIMAX(
    train,
    order=combo[0],
    seasonal_order=combo[1],
    enforce_stationarity=False,
    enforce_invertibility=False).fit()
sarima_model.summary()
```
executed in 4.50s, finished 14:26:49 2022-05-24

Out[657]:

SARIMAX Results

Dep. Variable:	value	**No. Observations:**	212
Model:	SARIMAX(0, 2, 2)x(1, 2, 2, 12)	**Log Likelihood**	-1262.835
Date:	Tue, 24 May 2022	**AIC**	2537.669
Time:	14:26:49	**BIC**	2556.083
Sample:	04-01-1996	**HQIC**	2545.147
	- 11-01-2013		
Covariance Type:	opg		

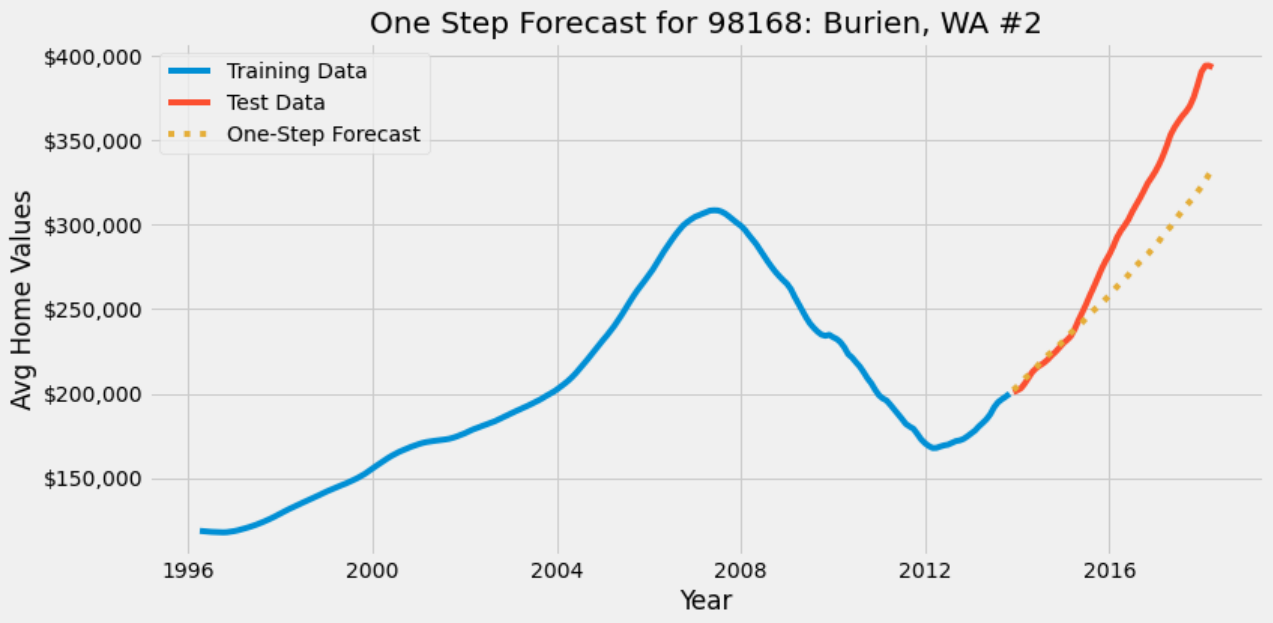
	coef	std err	z	P>|z|	[0.025	0.975]
ma.L1	0.1044	0.044	2.377	0.017	0.018	0.191
ma.L2	-0.0411	0.049	-0.838	0.402	-0.137	0.055
ar.S.L12	0.2963	0.026	11.200	0.000	0.244	0.348
ma.S.L12	-1.9371	0.745	-2.600	0.009	-3.397	-0.477
ma.S.L24	0.9855	0.751	1.312	0.189	-0.486	2.457
sigma2	2.97e+05	2.23e+05	1.333	0.183	-1.4e+05	7.34e+05

Ljung-Box (L1) (Q):	0.13	**Jarque-Bera (JB):**	44.67
Prob(Q):	0.72	**Prob(JB):**	0.00
Heteroskedasticity (H):	16.65	**Skew:**	0.00
Prob(H) (two-sided):	0.00	**Kurtosis:**	5.60

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).

In [658]:
```
zip_98168_model_2 = sarima_model
```
executed in 4ms, finished 14:26:49 2022-05-24

In [659]:
```
pred = zip_98168_model_2.get_prediction(
    start=test.index.min(),
    end=test.index.max(),
    dynamic=True)

fig, ax = plt.subplots(figsize=(12, 6))
ax.plot(train, label='Training Data')
ax.plot(test, label='Test Data')
ax.plot(pred.predicted_mean, label='One-Step Forecast', ls=':')
#line1 = sns.lineplot(data=avg_home_df, x='time', y='home_value', color='black', label='US Mean');

ax.set_xlabel('Year')
ax.set_ylabel('Avg Home Values')
ax.set_title("One Step Forecast for 98168: Burien, WA #2", fontsize = 20)
ax.legend()
ax.legend()
plt.savefig('images/burien2_one_step_2')
ax.yaxis.set_major_formatter(tick)
fig.tight_layout()
```
executed in 213ms, finished 14:26:49 2022-05-24



In [660]:
```
y_forecasted = pred.predicted_mean
y_truth = test['value']
mse = ((y_forecasted - y_truth) ** 2).mean()
print('The Mean Squared Error is {}'.format(round(mse, 2)))
print('The Root Mean Squared Error is {}'.format(round(np.sqrt(mse), 2)))
```
executed in 5ms, finished 14:26:49 2022-05-24

```
The Mean Squared Error is 1282920819.61
The Root Mean Squared Error is 35817.88
```

16.3 Model Comparison & Decision:

- The one-step forecast for both models is very similar, but Model 1 has a lower RMSE so **I will use Model #1 for forecasting.**

16.4 Prediction

In [661]:
```
combo = zip_98168_combo_1
#combo = zip_98168_combo_2
combo
```

executed in 4ms, finished 14:26:49 2022-05-24

Out[661]: ((1, 1, 1), (0, 1, 1, 12))

In [662]:
```
# Running model again with FULL data

sarima_model = SARIMAX(
    zip_98168_ts,
    order=combo[0],
    seasonal_order=combo[1],
    enforce_stationarity=False,
    enforce_invertibility=False).fit()
sarima_model.summary()
```

executed in 570ms, finished 14:26:50 2022-05-24

Out[662]:

SARIMAX Results

Dep. Variable:	value	**No. Observations:**	265
Model:	SARIMAX(1, 1, 1)x(0, 1, 1, 12)	**Log Likelihood**	-1922.944
Date:	Tue, 24 May 2022	**AIC**	3853.888
Time:	14:26:50	**BIC**	3867.777
Sample:	04-01-1996	**HQIC**	3859.485
	- 04-01-2018		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	0.8779	0.032	27.687	0.000	0.816	0.940
ma.L1	0.4338	0.038	11.534	0.000	0.360	0.507
ma.S.L12	-0.4225	0.032	-13.139	0.000	-0.485	-0.359
sigma2	5.41e+05	3.37e+04	16.055	0.000	4.75e+05	6.07e+05

Ljung-Box (L1) (Q):	0.01	**Jarque-Bera (JB):**	225.76
Prob(Q):	0.91	**Prob(JB):**	0.00
Heteroskedasticity (H):	7.39	**Skew:**	-0.53
Prob(H) (two-sided):	0.00	**Kurtosis:**	7.65

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).

In [663]:
```
zip_98168_model_full = sarima_model
```

executed in 4ms, finished 14:26:50 2022-05-24

In [664]:
```
zip_98168_ts.tail(1)
```

executed in 4ms, finished 14:26:50 2022-05-24

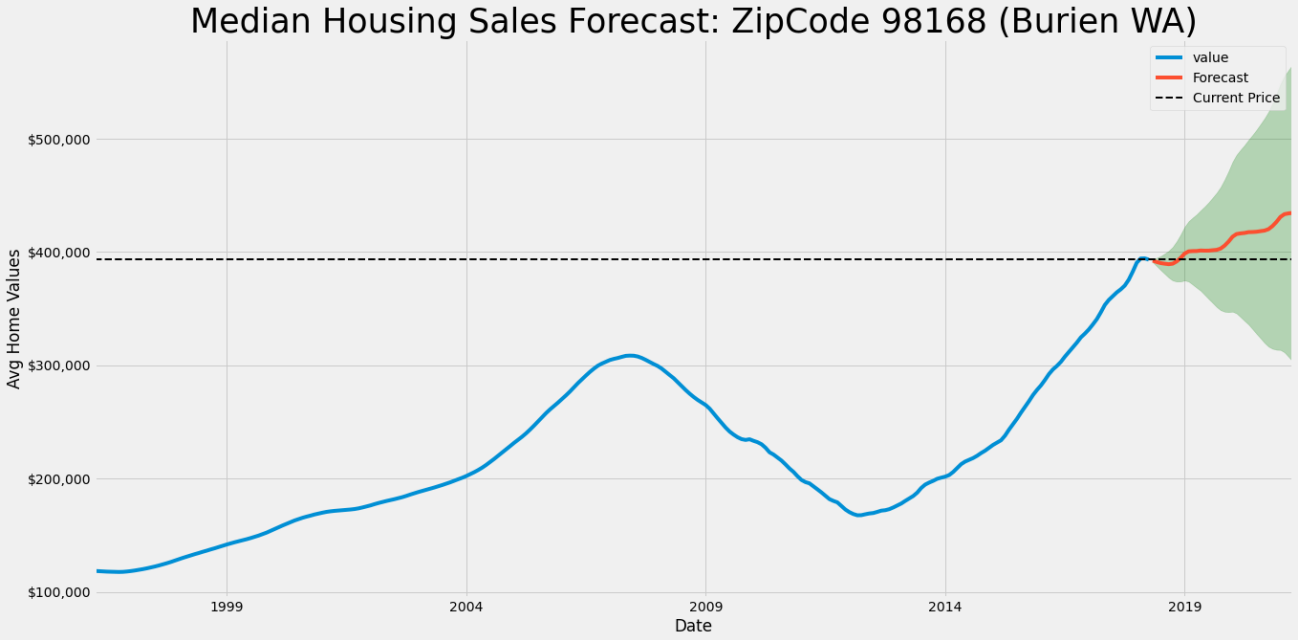
Out[664]:

time	value
2018-04-01	393,300.00

In [665]:

```
# Get forecast 3 yrs ahead in future (36 steps)
prediction = zip_98168_model_full.get_forecast(steps=36)

# Get confidence intervals of forecasts
pred_conf = prediction.conf_int()

# Plot future predictions with confidence intervals
ax = zip_98168_ts['1996-01':].plot(label='observed', figsize=(20, 10))
prediction.predicted_mean.plot(ax=ax, label='Forecast')
ax.fill_between(pred_conf.index,
                pred_conf.iloc[:, 0],
                pred_conf.iloc[:, 1], color='green', alpha=0.25)

ax.axhline(393300, ls='--', color='black', linewidth=(2), label='Current Price')

ax.set_xlabel('Date')
ax.set_ylabel('Avg Home Values')
ax.set_title("Median Housing Sales Forecast: ZipCode 98168 (Burien WA)", fontsize = 35)
ax.yaxis.set_major_formatter(tick)
plt.savefig('images/burien2_forecast')

plt.legend()
plt.show()
```

executed in 277ms, finished 14:26:50 2022-05-24



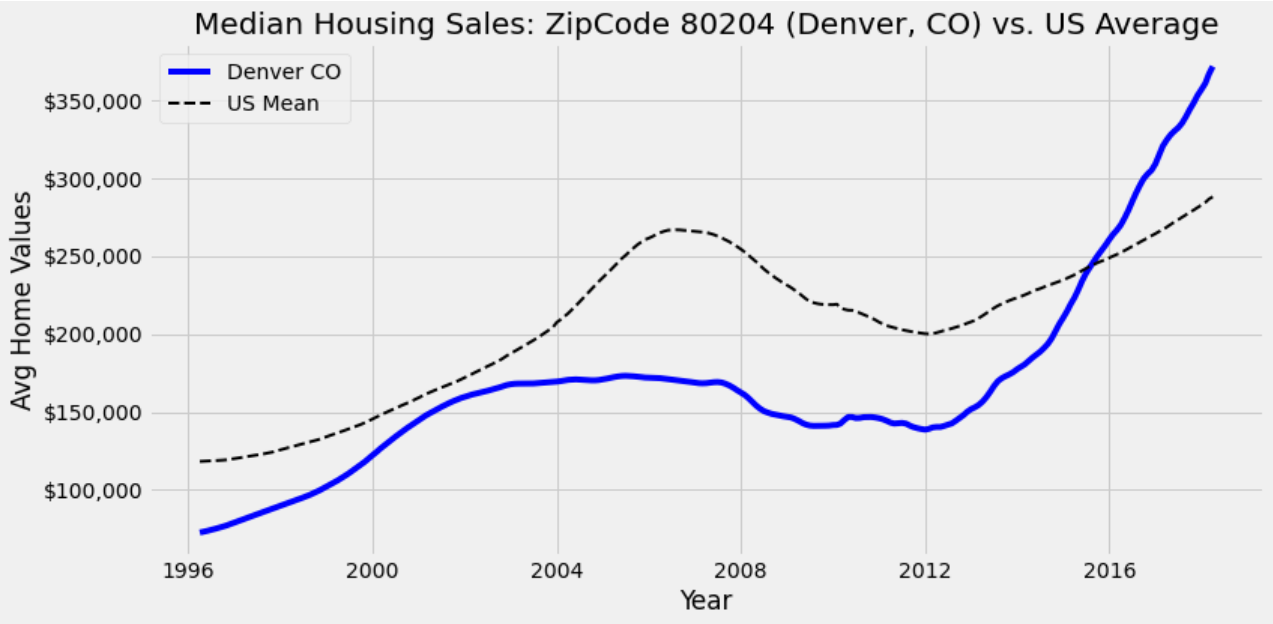
16.5 Analysis:

- The Confidence window is too wide and is spread close to the current price. There is too much uncertainty to recommend investing.
- **I do not recommend investing in this Zip Code.**

17 80204 Denver CO

In [666]:
```
fig, ax = plt.subplots(figsize=(12, 6))
ax.plot(zip_80204_ts, label='Denver CO', color='blue')

line1 = sns.lineplot(data=avg_home_df, x='time', y='home_value', ls='--',
                     color='black', linewidth=(2), label='US Mean');

ax.set_xlabel('Year')
ax.set_ylabel('Avg Home Values')
ax.set_title("Median Housing Sales: ZipCode 80204 (Denver, CO) vs. US Average", fontsize = 20)
ax.yaxis.set_major_formatter(tick)
fig.tight_layout()
```

executed in 168ms, finished 14:26:50 2022-05-24



In [667]:
```
#sarimax_param_search(zip_80204_ts)
```

executed in 2ms, finished 14:26:50 2022-05-24

In [668]:
```
#SARIMA Combos: ((1, 1, 1), (1, 1, 1, 12)) AIC: 3579.819829788861
zip_80204_combo_1 = ((1, 1, 1), (1, 1, 1, 12))
```

executed in 2ms, finished 14:26:50 2022-05-24

17.1 Model #1

In [669]:
```
combo = zip_80204_combo_1
combo
```

executed in 2ms, finished 14:26:50 2022-05-24

Out[669]:
```
((1, 1, 1), (1, 1, 1, 12))
```

In [670]:
```
# Manually split data.
temp_ts = zip_80204_ts

SPLIT = int(temp_ts.shape[0]*0.8)
train, test = temp_ts[:SPLIT], temp_ts[SPLIT:]

# Model
sarima_model = SARIMAX(
    train,
    order=combo[0],
    seasonal_order=combo[1],
    enforce_stationarity=False,
    enforce_invertibility=False).fit()
sarima_model.summary()
```

executed in 668ms, finished 14:26:51 2022-05-24

Out[670]:

SARIMAX Results

Dep. Variable:	value	**No. Observations:**	212
Model:	SARIMAX(1, 1, 1)x(1, 1, 1, 12)	**Log Likelihood**	-1326.153
Date:	Tue, 24 May 2022	**AIC**	2662.307
Time:	14:26:51	**BIC**	2678.409
Sample:	04-01-1996	**HQIC**	2668.833
	- 11-01-2013		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	0.9167	0.026	35.390	0.000	0.866	0.967
ma.L1	0.6154	0.042	14.824	0.000	0.534	0.697
ar.S.L12	0.0415	0.020	2.094	0.036	0.003	0.080
ma.S.L12	-0.9960	1.012	-0.984	0.325	-2.980	0.988
sigma2	7.961e+04	8.07e+04	0.986	0.324	-7.86e+04	2.38e+05

Ljung-Box (L1) (Q):	0.26	**Jarque-Bera (JB):**	98.02
Prob(Q):	0.61	**Prob(JB):**	0.00
Heteroskedasticity (H):	10.25	**Skew:**	0.29
Prob(H) (two-sided):	0.00	**Kurtosis:**	6.52

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).

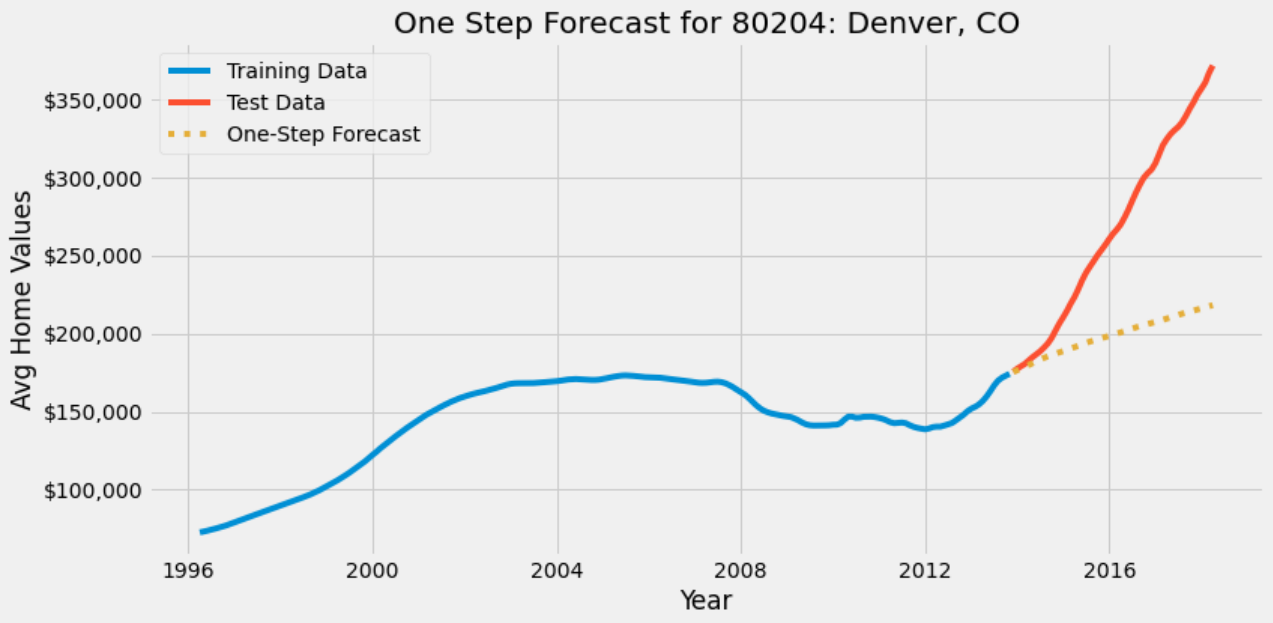
In [671]:
```
zip_80204_model_1 = sarima_model
```

executed in 4ms, finished 14:26:51 2022-05-24

In [672]:

```
pred = zip_80204_model_1.get_prediction(
    start=test.index.min(),
    end=test.index.max(),
    dynamic=False)

fig, ax = plt.subplots(figsize=(12, 6))
ax.plot(train, label='Training Data')
ax.plot(test, label='Test Data')
ax.plot(pred.predicted_mean, label='One-Step Forecast', ls=':')
#line1 = sns.lineplot(data=avg_home_df, x='time', y='home_value', color='black', label='US Mean');
ax.set_xlabel('Year')
ax.set_ylabel('Avg Home Values')
ax.set_title("One Step Forecast for 80204: Denver, CO", fontsize = 20)
ax.legend()
plt.savefig('images/denver_one_step_1')

ax.legend()
ax.yaxis.set_major_formatter(tick)
fig.tight_layout()
```

executed in 207ms, finished 14:26:51 2022-05-24



In [673]:
```
y_forecasted = pred.predicted_mean
y_truth = test['value']
mse = ((y_forecasted - y_truth) ** 2).mean()
print('The Mean Squared Error is {}'.format(round(mse, 2)))
print('The Root Mean Squared Error is {}'.format(round(np.sqrt(mse), 2)))
```

executed in 4ms, finished 14:26:51 2022-05-24

```
The Mean Squared Error is 6851819866.21
The Root Mean Squared Error is 82775.72
```

17.1.1 Decision: Changing Train Test Split

- A visual inspection of the observed data shows that the train test split needs to be adjusted so the model can better learn the trend at the end of this time series. I will just need to be careful not to overfit.

17.2 Model #1.5 (different Train_Test_Split)

In [674]:
```
combo = zip_80204_combo_1
combo
```

executed in 3ms, finished 14:26:51 2022-05-24

Out[674]:
```
((1, 1, 1), (1, 1, 1, 12))
```

In [675]:
```
# Manually split data.
temp_ts = zip_80204_ts

SPLIT = int(temp_ts.shape[0]*0.85)
train, test = temp_ts[:SPLIT], temp_ts[SPLIT:]

# Model
sarima_model = SARIMAX(
    train,
    order=combo[0],
    seasonal_order=combo[1],
    enforce_stationarity=False,
    enforce_invertibility=False).fit()
sarima_model.summary()
```

executed in 691ms, finished 14:26:52 2022-05-24

Out[675]:

SARIMAX Results

Dep. Variable:	value	**No. Observations:**	225
Model:	SARIMAX(1, 1, 1)x(1, 1, 1, 12)	**Log Likelihood**	-1433.605
Date:	Tue, 24 May 2022	**AIC**	2877.211
Time:	14:26:52	**BIC**	2893.652
Sample:	04-01-1996	**HQIC**	2883.866
	- 12-01-2014		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	0.9355	0.021	44.384	0.000	0.894	0.977
ma.L1	0.6188	0.042	14.624	0.000	0.536	0.702
ar.S.L12	0.0440	0.021	2.051	0.040	0.002	0.086
ma.S.L12	-0.9935	0.684	-1.451	0.147	-2.335	0.348
sigma2	9.221e+04	6.29e+04	1.467	0.142	-3.1e+04	2.15e+05

Ljung-Box (L1) (Q):	0.09	**Jarque-Bera (JB):**	60.27
Prob(Q):	0.77	**Prob(JB):**	0.00
Heteroskedasticity (H):	11.11	**Skew:**	0.35
Prob(H) (two-sided):	0.00	**Kurtosis:**	5.61

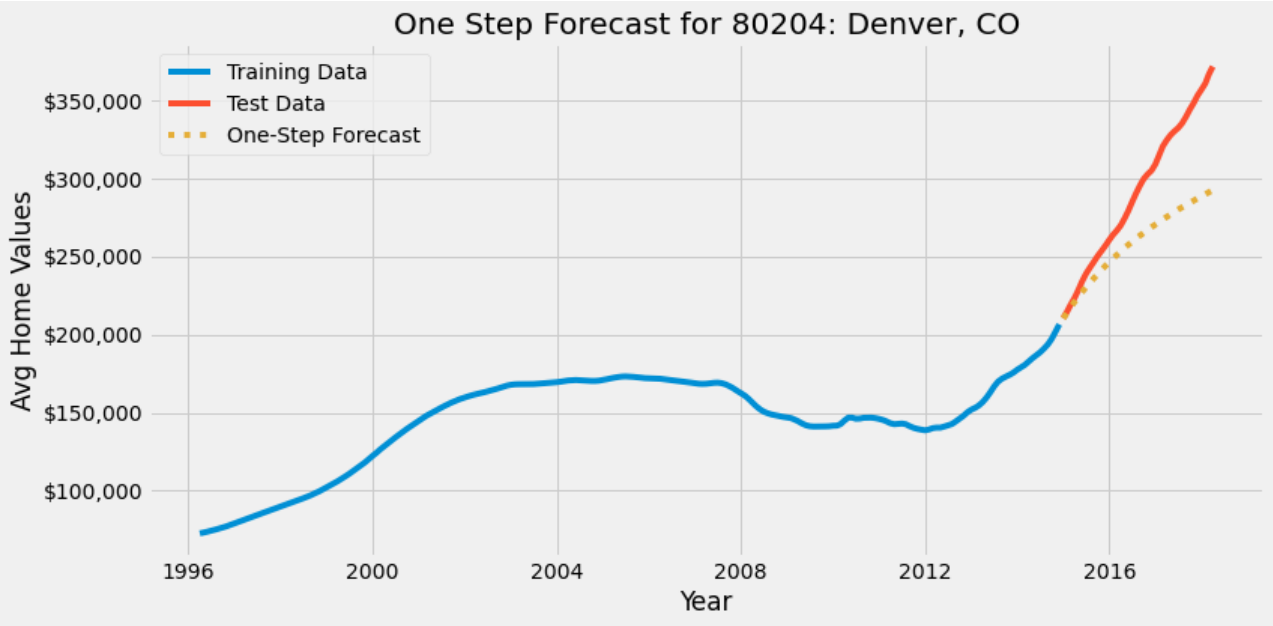
Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).

In [676]:
```
zip_80204_model_1_5 = sarima_model
```

executed in 4ms, finished 14:26:52 2022-05-24

In [677]:
```
pred = zip_80204_model_1_5.get_prediction(
    start=test.index.min(),
    end=test.index.max(),
    dynamic=False)

fig, ax = plt.subplots(figsize=(12, 6))
ax.plot(train, label='Training Data')
ax.plot(test, label='Test Data')
ax.plot(pred.predicted_mean, label='One-Step Forecast', ls=':')
#line1 = sns.lineplot(data=avg_home_df, x='time', y='home_value', color='black', label='US Mean');
ax.set_xlabel('Year')
ax.set_ylabel('Avg Home Values')
ax.set_title("One Step Forecast for 80204: Denver, CO", fontsize = 20)
ax.legend()
plt.savefig('images/denver_one_step_2')
ax.yaxis.set_major_formatter(tick)
fig.tight_layout()
```

executed in 204ms, finished 14:26:52 2022-05-24



In [678]:
```
y_forecasted = pred.predicted_mean
y_truth = test['value']
mse = ((y_forecasted - y_truth) ** 2).mean()
print('The Mean Squared Error is {}'.format(round(mse, 2)))
print('The Root Mean Squared Error is {}'.format(round(np.sqrt(mse), 2)))
```

executed in 4ms, finished 14:26:52 2022-05-24

```
The Mean Squared Error is 1611073051.99
The Root Mean Squared Error is 40138.17
```

17.3 Model #2

In [679]:
```
1 #sarimax_param_search(zip_80204_ts)
```
executed in 2ms, finished 14:26:52 2022-05-24

In [680]:
```
1 # SARIMA Combos: ((1, 2, 2), (2, 2, 2, 12)) AIC: 3270.679349948866
2 zip_80204_combo_2 = ((1, 2, 2), (2, 2, 2, 12))
```
executed in 2ms, finished 14:26:52 2022-05-24

In [681]:
```
1 combo = zip_80204_combo_2
2 combo
```
executed in 2ms, finished 14:26:52 2022-05-24

Out[681]:
```
((1, 2, 2), (2, 2, 2, 12))
```

In [682]:
```
# Manually split data.
temp_ts = zip_80204_ts

SPLIT = int(temp_ts.shape[0]*0.85)
train, test = temp_ts[:SPLIT], temp_ts[SPLIT:]

# Model
sarima_model = SARIMAX(
    train,
    order=combo[0],
    seasonal_order=combo[1],
    enforce_stationarity=False,
    enforce_invertibility=False).fit()
sarima_model.summary()
```

executed in 6.31s, finished 14:26:58 2022-05-24

Out[682]:

SARIMAX Results

Dep. Variable:	value	**No. Observations:**	225
Model:	SARIMAX(1, 2, 2)x(2, 2, 2, 12)	**Log Likelihood**	-1307.665
Date:	Tue, 24 May 2022	**AIC**	2631.329
Time:	14:26:58	**BIC**	2656.509
Sample:	04-01-1996	**HQIC**	2641.545
	- 12-01-2014		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	-0.4353	0.170	-2.553	0.011	-0.769	-0.101
ma.L1	1.1774	0.163	7.203	0.000	0.857	1.498
ma.L2	0.4448	0.101	4.421	0.000	0.248	0.642
ar.S.L12	-0.0868	0.064	-1.354	0.176	-0.212	0.039
ar.S.L24	0.1593	0.049	3.240	0.001	0.063	0.256
ma.S.L12	-1.2285	0.098	-12.497	0.000	-1.421	-1.036
ma.S.L24	0.3226	0.108	2.978	0.003	0.110	0.535
sigma2	1.734e+05	1.79e+04	9.707	0.000	1.38e+05	2.08e+05

Ljung-Box (L1) (Q):	1.57	**Jarque-Bera (JB):**	25.04
Prob(Q):	0.21	**Prob(JB):**	0.00
Heteroskedasticity (H):	9.83	**Skew:**	0.16
Prob(H) (two-sided):	0.00	**Kurtosis:**	4.84

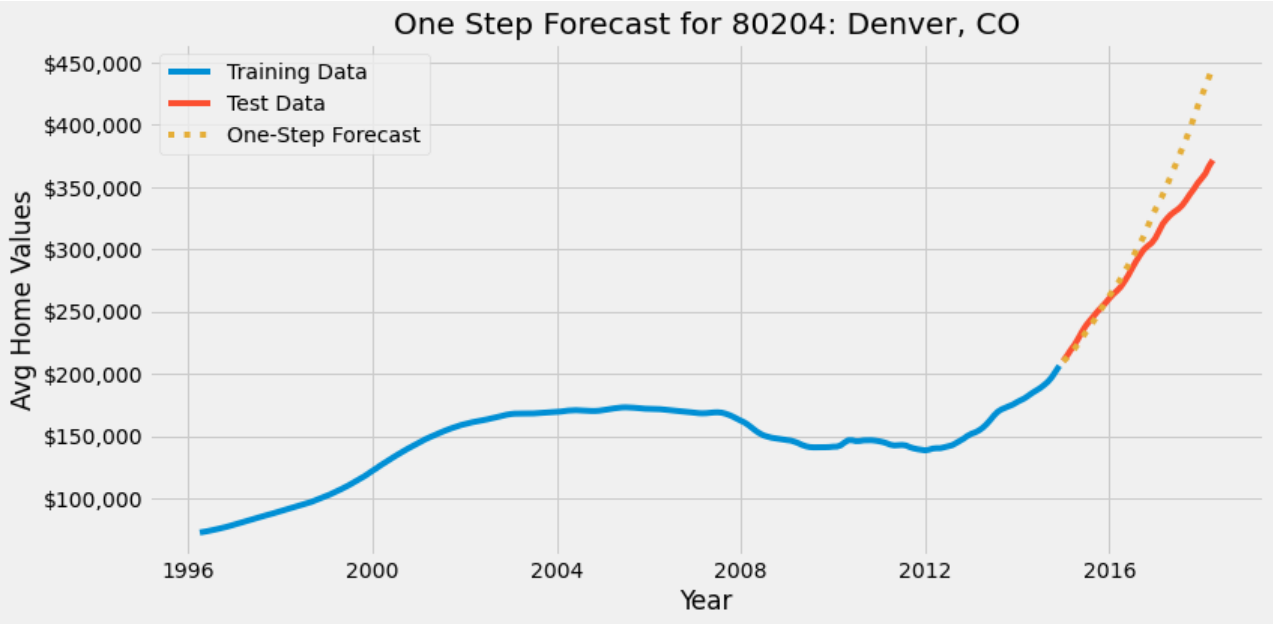
Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).

In [683]:
```
zip_80204_model_2 = sarima_model
```

executed in 2ms, finished 14:26:58 2022-05-24

In [684]:
```
pred = zip_80204_model_2.get_prediction(
    start=test.index.min(),
    end=test.index.max(),
    dynamic=False)

fig, ax = plt.subplots(figsize=(12, 6))
ax.plot(train, label='Training Data')
ax.plot(test, label='Test Data')
ax.plot(pred.predicted_mean, label='One-Step Forecast', ls=':')
#line1 = sns.lineplot(data=avg_home_df, x='time', y='home_value', color='black', label='US Mean');
ax.set_xlabel('Year')
ax.set_ylabel('Avg Home Values')
ax.set_title("One Step Forecast for 80204: Denver, CO", fontsize = 20)
ax.legend()
plt.savefig('images/denver_one_step_3')
ax.yaxis.set_major_formatter(tick)
fig.tight_layout()
```

executed in 213ms, finished 14:26:58 2022-05-24



In [685]:
```
y_forecasted = pred.predicted_mean
y_truth = test['value']
mse = ((y_forecasted - y_truth) ** 2).mean()
print('The Mean Squared Error is {}'.format(round(mse, 2)))
print('The Root Mean Squared Error is {}'.format(round(np.sqrt(mse), 2)))
```

executed in 4ms, finished 14:26:58 2022-05-24

```
The Mean Squared Error is 1043987906.65
The Root Mean Squared Error is 32310.8
```

17.4 Decision:

- Model 1.5 has a slightly better RMSE, and Model 2 seems to be overfit as it is predicting a straight increase. **I will use Model 1.5 for Forecasting**

In [ ]:
```
1
```

17.5 Prediction

In [686]:
```
1 combo = zip_80204_combo_1
2 #combo = zip_80204_combo_2
3 combo
```

executed in 2ms, finished 14:26:58 2022-05-24

Out[686]:
```
((1, 1, 1), (1, 1, 1, 12))
```

In [687]:
```
1 # Model
2 sarima_model = SARIMAX(
3     zip_80204_ts,
4     order=combo[0],
5     seasonal_order=combo[1],
6     enforce_stationarity=False,
7     enforce_invertibility=False).fit()
8 sarima_model.summary()
```

executed in 758ms, finished 14:26:59 2022-05-24

Out[687]:

SARIMAX Results

Dep. Variable:	value	**No. Observations:**	265
Model:	SARIMAX(1, 1, 1)x(1, 1, 1, 12)	**Log Likelihood**	-1784.910
Date:	Tue, 24 May 2022	**AIC**	3579.820
Time:	14:26:59	**BIC**	3597.181
Sample:	04-01-1996	**HQIC**	3586.817
	- 04-01-2018		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	0.9278	0.017	53.915	0.000	0.894	0.962
ma.L1	0.7078	0.030	23.751	0.000	0.649	0.766
ar.S.L12	0.0554	0.024	2.311	0.021	0.008	0.102
ma.S.L12	-0.9874	0.208	-4.748	0.000	-1.395	-0.580
sigma2	1.505e+05	3.1e+04	4.854	0.000	8.98e+04	2.11e+05

Ljung-Box (L1) (Q):	0.07	**Jarque-Bera (JB):**	54.85
Prob(Q):	0.79	**Prob(JB):**	0.00
Heteroskedasticity (H):	8.10	**Skew:**	0.40
Prob(H) (two-sided):	0.00	**Kurtosis:**	5.21

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).

In [688]:
```
1 zip_80204_model_full = sarima_model
```

executed in 3ms, finished 14:26:59 2022-05-24

In [689]:
```
zip_80204_ts.tail(1)
```

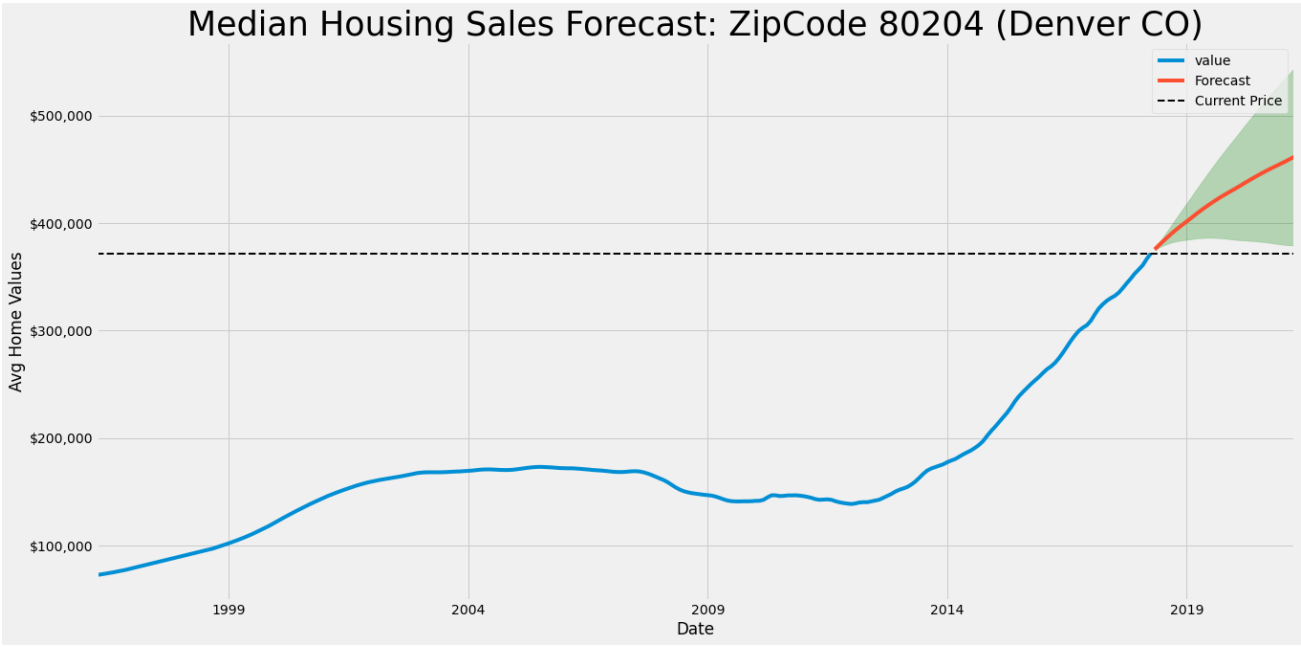
executed in 4ms, finished 14:26:59 2022-05-24

Out[689]:

time	value
2018-04-01	371,600.00

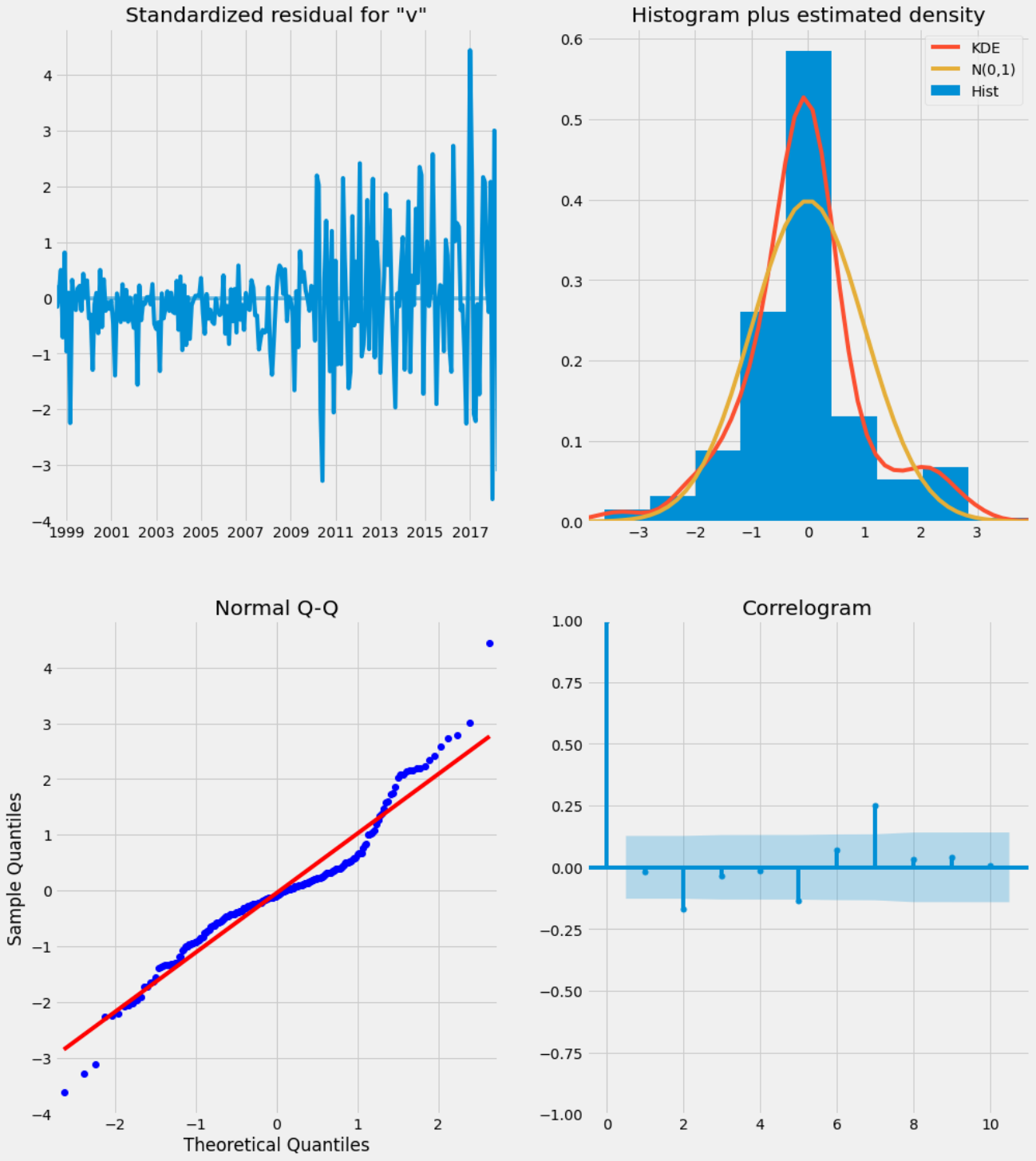
In [690]:
```
# Get forecast 3 yrs ahead in future (36 steps)
prediction = zip_80204_model_full.get_forecast(steps=36)

# Get confidence intervals of forecasts
pred_conf = prediction.conf_int()

# Plot future predictions with confidence intervals
ax = zip_80204_ts['1996-01':].plot(label='observed', figsize=(20, 10))
prediction.predicted_mean.plot(ax=ax, label='Forecast')
ax.fill_between(pred_conf.index,
                pred_conf.iloc[:, 0],
                pred_conf.iloc[:, 1], color='green', alpha=0.25)

ax.axhline(371600, ls='--', color='black', linewidth=(2), label='Current Price')

ax.set_xlabel('Date')
ax.set_ylabel('Avg Home Values')
ax.set_title("Median Housing Sales Forecast: ZipCode 80204 (Denver CO)", fontsize = 35)
ax.yaxis.set_major_formatter(tick)
plt.savefig('images/denver_forecast')

plt.legend()
plt.show()
```

executed in 265ms, finished 14:26:59 2022-05-24



In [691]:
```
zip_80204_model_full.plot_diagnostics(figsize=(15, 18))
plt.show()
```

executed in 456ms, finished 14:27:00 2022-05-24



17.6 Analysis:

- While this zip code is a little lower than the exact price range that I was given, it is a strong investment, with even the lower confidence value making a slight profit.
- **I recommend to invest in this Zip Code for the full 3 years.**

17.7 Metrics

- capturing metrics to compare with the top zip codes later.

In [692]:
```
zip_80204_ts.tail(1)
```
executed in 4ms, finished 14:27:00 2022-05-24

Out[692]:

time	value
2018-04-01	371,600.00

In [693]:
```
analysis_df = pred_conf.resample('Y').mean()
analysis_df['mean'] = analysis_df.apply(lambda x: x.mean(), axis=1)

analysis_df['base'] = 371600
analysis_df['base'] = analysis_df['base'].astype(float)
```
executed in 6ms, finished 14:27:00 2022-05-24

In [694]:
```
analysis_df = analysis_df[['base', 'lower value', 'mean', 'upper value']]

analysis_df['Min_ROI'] = analysis_df.apply(lambda x: x['lower value'] - x['base'], axis=1)
analysis_df['Min_ROI%'] = analysis_df.apply(lambda x: x['Min_ROI'] / x['base'], axis=1)
analysis_df['Forecast_ROI'] = analysis_df.apply(lambda x: x['mean'] - x['base'], axis=1)
analysis_df['Forecast_ROI%'] = analysis_df.apply(lambda x: x['mean'] / x['base'], axis=1)
analysis_df['Max_ROI'] = analysis_df.apply(lambda x: x['upper value'] - x['base'], axis=1)
analysis_df['Max_ROI%'] = analysis_df.apply(lambda x: x['upper value'] / x['base'], axis=1)
```
executed in 10ms, finished 14:27:00 2022-05-24

In [695]:
```
zip_80204_metrics = analysis_df
```
executed in 1ms, finished 14:27:00 2022-05-24

In [696]:
```
zip_80204_metrics
```
executed in 6ms, finished 14:27:00 2022-05-24

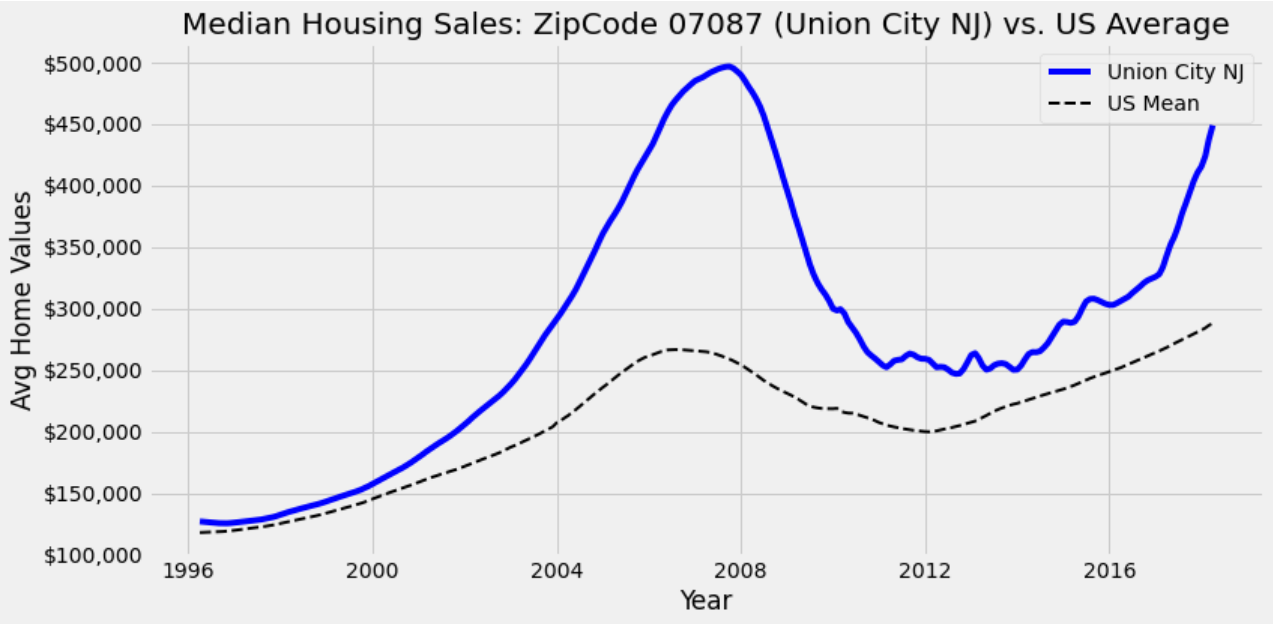
Out[696]:

	base	lower value	mean	upper value	Min_ROI	Min_ROI%	Forecast_ROI	Forecast_ROI%	Max_ROI	Max_ROI%
2018-12-31	371,600.00	380,320.70	387,502.07	394,683.43	8,720.70	0.02	15,902.07	1.04	23,083.43	1.06
2019-12-31	371,600.00	385,601.55	416,261.60	446,921.66	14,001.55	0.04	44,661.60	1.12	75,321.66	1.20
2020-12-31	371,600.00	382,699.74	443,296.34	503,892.94	11,099.74	0.03	71,696.34	1.19	132,292.94	1.36
2021-12-31	371,600.00	379,421.36	458,806.66	538,191.96	7,821.36	0.02	87,206.66	1.23	166,591.96	1.45

18 07087 Union City NJ

In [697]:
```
fig, ax = plt.subplots(figsize=(12, 6))
ax.plot(zip_07087_ts, label='Union City NJ', color='blue')

\
line1 = sns.lineplot(data=avg_home_df, x='time', y='home_value', ls='--',
                     color='black', linewidth=(2), label='US Mean');

ax.set_xlabel('Year')
ax.set_ylabel('Avg Home Values')
ax.set_title("Median Housing Sales: ZipCode 07087 (Union City NJ) vs. US Average", fontsize = 20)
ax.yaxis.set_major_formatter(tick)
fig.tight_layout()
```

executed in 173ms, finished 14:27:00 2022-05-24



In [698]:
```
#SARIMA Combos: ((1, 1, 1), (1, 1, 1, 12)) AIC: 4153.772501773967
zip_07087_combo_1 = ((1,1,1), (1,1,1,12))
```

executed in 2ms, finished 14:27:00 2022-05-24

18.1 Model #1

In [699]:
```
combo = zip_07087_combo_1
combo
```

executed in 2ms, finished 14:27:00 2022-05-24

Out[699]: ((1, 1, 1), (1, 1, 1, 12))

In [700]:
```
# Manually split data.
temp_ts = zip_07087_ts

SPLIT = int(temp_ts.shape[0]*0.8)
train, test = temp_ts[:SPLIT], temp_ts[SPLIT:]

# Model
sarima_model = SARIMAX(
    train,
    order=combo[0],
    seasonal_order=combo[1],
    enforce_stationarity=False,
    enforce_invertibility=False).fit()
sarima_model.summary()
```

executed in 257ms, finished 14:27:00 2022-05-24

Out[700]:

SARIMAX Results

Dep. Variable:	value	**No. Observations:**	212
Model:	SARIMAX(1, 1, 1)x(1, 1, 1, 12)	**Log Likelihood**	-1584.993
Date:	Tue, 24 May 2022	**AIC**	3179.986
Time:	14:27:00	**BIC**	3196.088
Sample:	04-01-1996	**HQIC**	3186.512
	- 11-01-2013		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	0.8630	0.043	20.172	0.000	0.779	0.947
ma.L1	0.6437	0.048	13.547	0.000	0.551	0.737
ar.S.L12	0.1807	0.087	2.069	0.039	0.009	0.352
ma.S.L12	-0.4898	0.070	-7.020	0.000	-0.627	-0.353
sigma2	1.852e+06	1.25e+05	14.865	0.000	1.61e+06	2.1e+06

Ljung-Box (L1) (Q):	0.51	**Jarque-Bera (JB):**	329.34
Prob(Q):	0.47	**Prob(JB):**	0.00
Heteroskedasticity (H):	15.27	**Skew:**	1.11
Prob(H) (two-sided):	0.00	**Kurtosis:**	9.15

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).

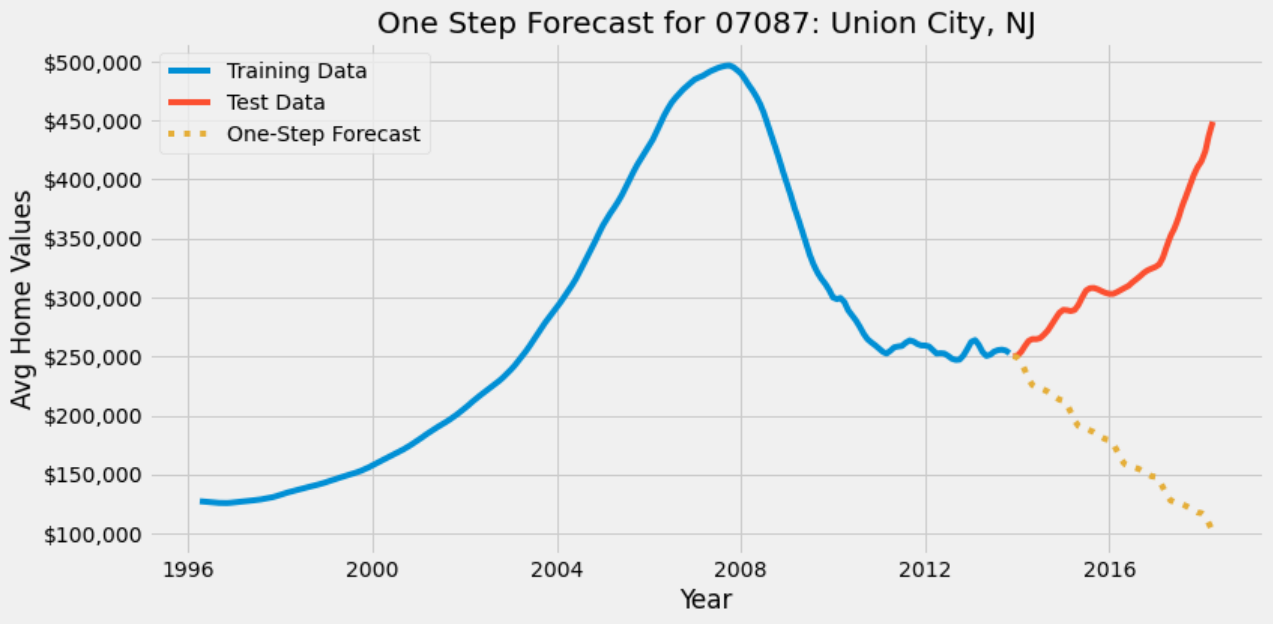
In [701]:
```
zip_07087_model_1 = sarima_model
```

executed in 2ms, finished 14:27:00 2022-05-24

In [702]:

```
pred = zip_07087_model_1.get_prediction(
    start=test.index.min(),
    end=test.index.max(),
    dynamic=False)

fig, ax = plt.subplots(figsize=(12, 6))
ax.plot(train, label='Training Data')
ax.plot(test, label='Test Data')
ax.plot(pred.predicted_mean, label='One-Step Forecast', ls=':')
#line1 = sns.lineplot(data=avg_home_df, x='time', y='home_value', color='black', label='US Mean');
ax.set_xlabel('Year')
ax.set_ylabel('Avg Home Values')
ax.set_title("One Step Forecast for 07087: Union City, NJ", fontsize = 20)
ax.legend()
plt.savefig('images/union_city_one_step_1')
ax.yaxis.set_major_formatter(tick)
fig.tight_layout()
```

executed in 213ms, finished 14:27:01 2022-05-24



18.2 Analysis:

- It is not surprising that the one-step forecast continues to drop so much due to the downward trend of the training data. As I know that the price did go up, I am going to modify my train_test_split to 90% so that the model can get some of the increasing trend.
- I will need to make sure that it isn't overfitting though.

In [703]:
```
y_forecasted = pred.predicted_mean
y_truth = test['value']
mse = ((y_forecasted - y_truth) ** 2).mean()
print('The Mean Squared Error is {}'.format(round(mse, 2)))
print('The Root Mean Squared Error is {}'.format(round(np.sqrt(mse), 2)))
```

executed in 4ms, finished 14:27:01 2022-05-24

```
The Mean Squared Error is 29128557428.9
The Root Mean Squared Error is 170670.9
```

18.3 Model 1.5 (with new train-test split)

In [704]:
```
combo = zip_07087_combo_1
combo
```

executed in 3ms, finished 14:27:01 2022-05-24

Out[704]:
```
((1, 1, 1), (1, 1, 1, 12))
```

In [705]:
```
# Manually split data.
temp_ts = zip_07087_ts

SPLIT = int(temp_ts.shape[0]*0.9)
train, test = temp_ts[:SPLIT], temp_ts[SPLIT:]

# Model
sarima_model = SARIMAX(
    train,
    order=combo[0],
    seasonal_order=combo[1],
    enforce_stationarity=False,
    enforce_invertibility=False).fit()
sarima_model.summary()
```

executed in 288ms, finished 14:27:01 2022-05-24

Out[705]:

SARIMAX Results

Dep. Variable:	value	**No. Observations:**	238
Model:	SARIMAX(1, 1, 1)x(1, 1, 1, 12)	**Log Likelihood**	-1829.401
Date:	Tue, 24 May 2022	**AIC**	3668.801
Time:	14:27:01	**BIC**	3685.560
Sample:	04-01-1996	**HQIC**	3675.575
	- 01-01-2016		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	0.8345	0.041	20.194	0.000	0.753	0.915
ma.L1	0.6587	0.047	14.106	0.000	0.567	0.750
ar.S.L12	0.1982	0.063	3.127	0.002	0.074	0.322
ma.S.L12	-0.5505	0.048	-11.373	0.000	-0.645	-0.456
sigma2	2.099e+06	1.54e+05	13.670	0.000	1.8e+06	2.4e+06

Ljung-Box (L1) (Q):	3.32	**Jarque-Bera (JB):**	144.67
Prob(Q):	0.07	**Prob(JB):**	0.00
Heteroskedasticity (H):	9.62	**Skew:**	0.62
Prob(H) (two-sided):	0.00	**Kurtosis:**	6.86

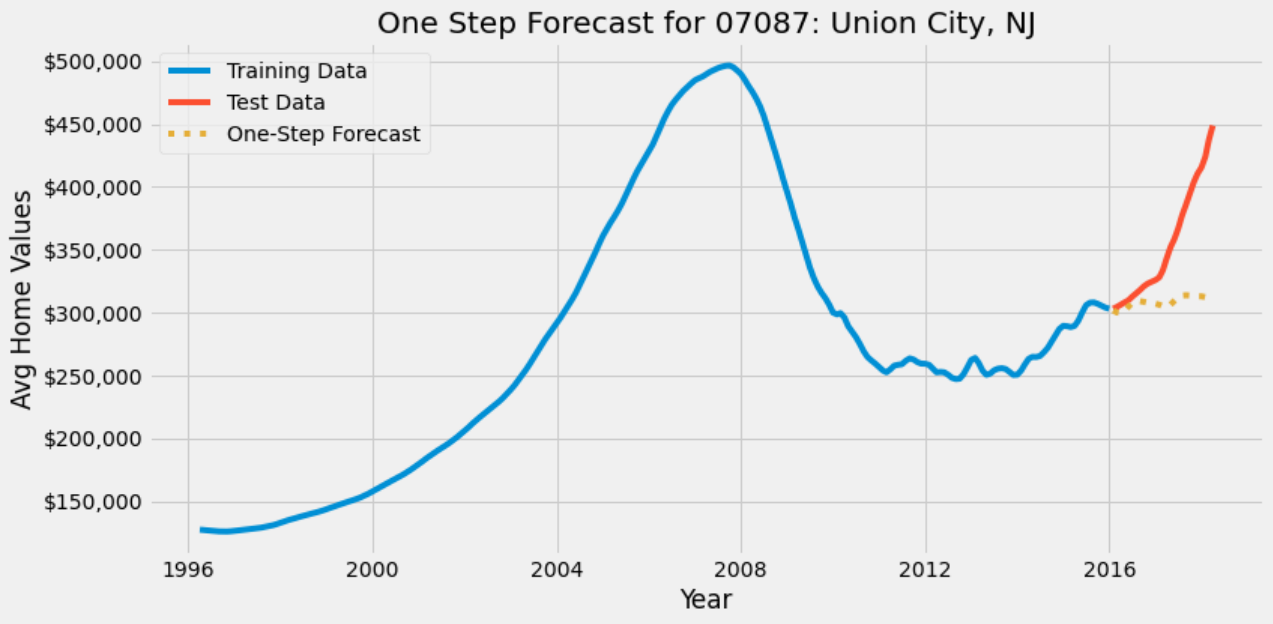
Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).

In [706]:
```
zip_07087_model_1_5 = sarima_model
```

executed in 2ms, finished 14:27:01 2022-05-24

In [707]:
```
pred = zip_07087_model_1_5.get_prediction(
    start=test.index.min(),
    end=test.index.max(),
    dynamic=False)

fig, ax = plt.subplots(figsize=(12, 6))
ax.plot(train, label='Training Data')
ax.plot(test, label='Test Data')
ax.plot(pred.predicted_mean, label='One-Step Forecast', ls=':')
#line1 = sns.lineplot(data=avg_home_df, x='time', y='home_value', color='black', label='US Mean');
ax.set_xlabel('Year')
ax.set_ylabel('Avg Home Values')
ax.set_title("One Step Forecast for 07087: Union City, NJ", fontsize = 20)
ax.legend()
plt.savefig('images/union_city_one_step_2')
ax.yaxis.set_major_formatter(tick)
fig.tight_layout()
```

executed in 206ms, finished 14:27:01 2022-05-24



- As expected, the one step forecast has evened out a bit.

In [708]:
```
y_forecasted = pred.predicted_mean
y_truth = test['value']
mse = ((y_forecasted - y_truth) ** 2).mean()
print('The Mean Squared Error is {}'.format(round(mse, 2)))
print('The Root Mean Squared Error is {}'.format(round(np.sqrt(mse), 2)))
```

executed in 4ms, finished 14:27:01 2022-05-24

```
The Mean Squared Error is 3820762507.36
The Root Mean Squared Error is 61812.32
```

18.4 Model #2

In [709]:
```
#SARIMA Combos: ((1, 1, 2), (1, 2, 2, 12)) AIC: 3771.3550338175914
zip_07087_combo_2 = ((1, 1, 2), (1, 2, 2, 12))
```

executed in 2ms, finished 14:27:01 2022-05-24

In [710]:
```
combo = zip_07087_combo_2
combo
```
executed in 3ms, finished 14:27:01 2022-05-24

Out[710]:
```
((1, 1, 2), (1, 2, 2, 12))
```

In [711]:
```
# Manually split data.
temp_ts = zip_07087_ts

SPLIT = int(temp_ts.shape[0]*0.90)
train, test = temp_ts[:SPLIT], temp_ts[SPLIT:]

# Model
sarima_model = SARIMAX(
    train,
    order=combo[0],
    seasonal_order=combo[1],
    enforce_stationarity=False,
    enforce_invertibility=False).fit()
sarima_model.summary()
```
executed in 5.70s, finished 14:27:07 2022-05-24

Out[711]:

SARIMAX Results

Dep. Variable:	value	**No. Observations:**	238
Model:	SARIMAX(1, 1, 2)x(1, 2, 2, 12)	**Log Likelihood**	-1638.292
Date:	Tue, 24 May 2022	**AIC**	3290.584
Time:	14:27:07	**BIC**	3313.164
Sample:	04-01-1996	**HQIC**	3299.734
	- 01-01-2016		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	0.5706	0.072	7.874	0.000	0.429	0.713
ma.L1	1.2943	0.060	21.563	0.000	1.177	1.412
ma.L2	0.6085	0.052	11.613	0.000	0.506	0.711
ar.S.L12	0.1514	0.042	3.582	0.000	0.069	0.234
ma.S.L12	-1.2616	0.067	-18.726	0.000	-1.394	-1.130
ma.S.L24	0.3513	0.059	5.991	0.000	0.236	0.466
sigma2	2.286e+06	2.04e+05	11.203	0.000	1.89e+06	2.69e+06

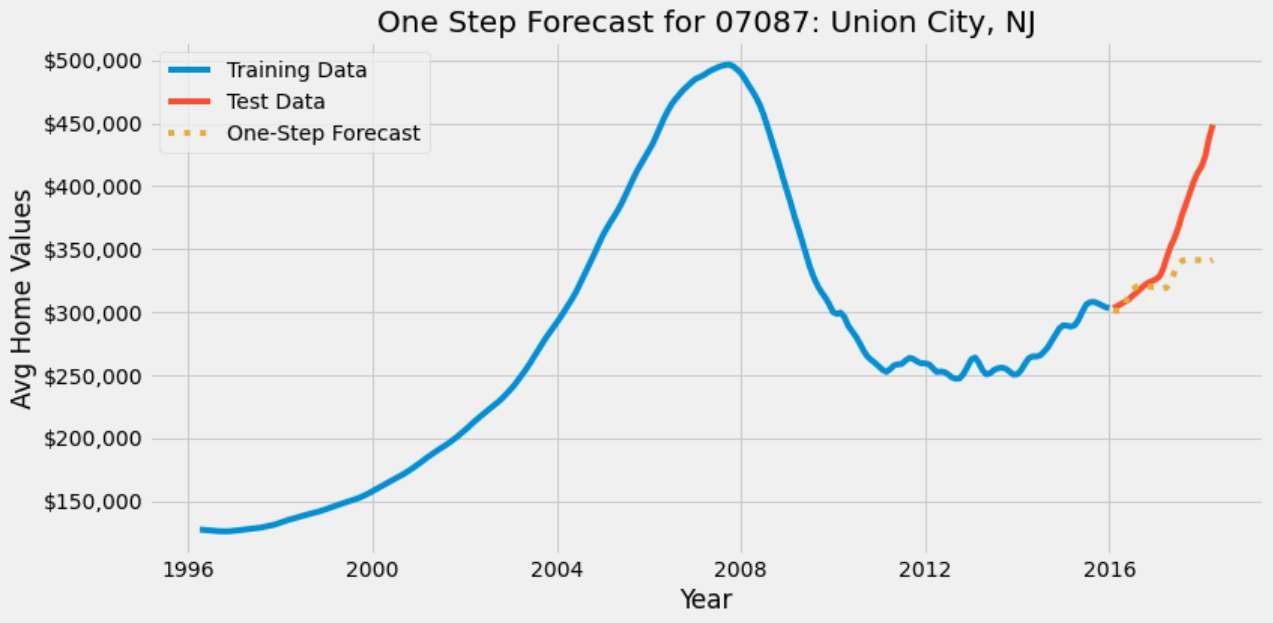
Ljung-Box (L1) (Q):	2.27	**Jarque-Bera (JB):**	72.48
Prob(Q):	0.13	**Prob(JB):**	0.00
Heteroskedasticity (H):	4.49	**Skew:**	0.86
Prob(H) (two-sided):	0.00	**Kurtosis:**	5.52

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).

In [712]:
```
zip_07087_model_2 = sarima_model
```
executed in 4ms, finished 14:27:07 2022-05-24

In [713]:
```
pred = zip_07087_model_2.get_prediction(
    start=test.index.min(),
    end=test.index.max(),
    dynamic=False)

fig, ax = plt.subplots(figsize=(12, 6))
ax.plot(train, label='Training Data')
ax.plot(test, label='Test Data')
ax.plot(pred.predicted_mean, label='One-Step Forecast', ls=':')

#line1 = sns.lineplot(data=avg_home_df, x='time', y='home_value', color='black', label='US Mean');
ax.set_xlabel('Year')
ax.set_ylabel('Avg Home Values')
ax.set_title("One Step Forecast for 07087: Union City, NJ", fontsize = 20)
plt.savefig('images/union_city_one_step_3')


ax.legend()
ax.yaxis.set_major_formatter(tick)
fig.tight_layout()
```

executed in 196ms, finished 14:27:07 2022-05-24



In [714]:
```
y_forecasted = pred.predicted_mean
y_truth = test['value']
mse = ((y_forecasted - y_truth) ** 2).mean()
print('The Mean Squared Error is {}'.format(round(mse, 2)))
print('The Root Mean Squared Error is {}'.format(round(np.sqrt(mse), 2)))
```

executed in 3ms, finished 14:27:07 2022-05-24

```
The Mean Squared Error is 1912316377.95
The Root Mean Squared Error is 43730.04
```

18.5 Model Comparison & Decision:

- Model 2 has a lower RSME and doesn't appear to be overfit, so I will use it for my forecast.

18.6 Prediction

In [715]:
```
#combo = zip_07087_combo_1
combo = zip_07087_combo_2
combo
```

executed in 4ms, finished 14:27:07 2022-05-24

Out[715]:
```
((1, 1, 2), (1, 2, 2, 12))
```

In [716]:
```
# Running model again with FULL data

sarima_model = SARIMAX(
    zip_07087_ts,
    order=combo[0],
    seasonal_order=combo[1],
    enforce_stationarity=False,
    enforce_invertibility=False).fit()
sarima_model.summary()
```

executed in 6.90s, finished 14:27:14 2022-05-24

Out[716]:

SARIMAX Results

Dep. Variable:	value	**No. Observations:**	265
Model:	SARIMAX(1, 1, 2)x(1, 2, 2, 12)	**Log Likelihood**	-1878.678
Date:	Tue, 24 May 2022	**AIC**	3771.355
Time:	14:27:14	**BIC**	3794.884
Sample:	04-01-1996	**HQIC**	3780.864
	- 04-01-2018		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	0.5457	0.070	7.850	0.000	0.409	0.682
ma.L1	1.3168	0.057	23.206	0.000	1.206	1.428
ma.L2	0.6283	0.050	12.505	0.000	0.530	0.727
ar.S.L12	0.1490	0.041	3.663	0.000	0.069	0.229
ma.S.L12	-1.2606	0.066	-19.079	0.000	-1.390	-1.131
ma.S.L24	0.3543	0.059	5.998	0.000	0.239	0.470
sigma2	2.392e+06	2.04e+05	11.749	0.000	1.99e+06	2.79e+06

Ljung-Box (L1) (Q):	2.14	**Jarque-Bera (JB):**	47.60
Prob(Q):	0.14	**Prob(JB):**	0.00
Heteroskedasticity (H):	3.76	**Skew:**	0.69
Prob(H) (two-sided):	0.00	**Kurtosis:**	4.87

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).

In [717]:
```
zip_07087_model_full = sarima_model
```

executed in 3ms, finished 14:27:14 2022-05-24

In [718]:
```
zip_07087_ts.tail(1)
```

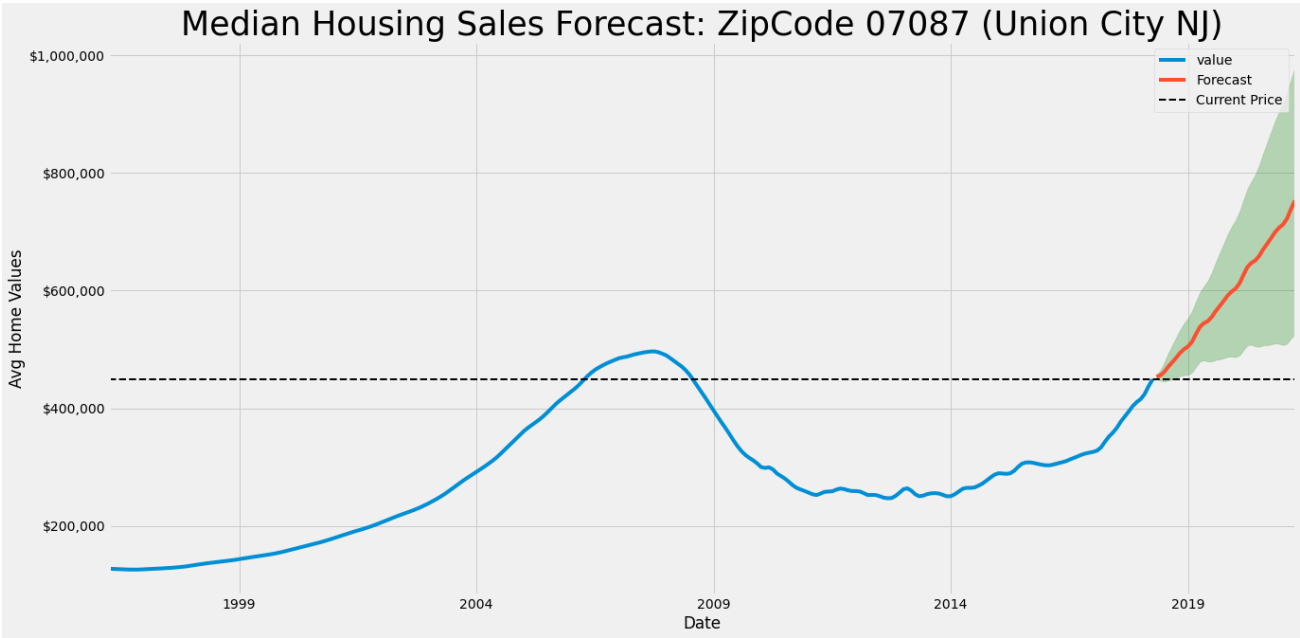
executed in 4ms, finished 14:27:14 2022-05-24

Out[718]:

time	value
2018-04-01	448,900.00

In [719]:
```
# Get forecast 3 yrs ahead in future (36 steps)
prediction = zip_07087_model_full.get_forecast(steps=36)

# Get confidence intervals of forecasts
pred_conf = prediction.conf_int()

# Plot future predictions with confidence intervals
ax = zip_07087_ts['1996-01':].plot(label='observed', figsize=(20, 10))
prediction.predicted_mean.plot(ax=ax, label='Forecast')
ax.fill_between(pred_conf.index,
                pred_conf.iloc[:, 0],
                pred_conf.iloc[:, 1], color='green', alpha=0.25)

ax.axhline(448900, ls='--', color='black', linewidth=(2), label='Current Price')

ax.set_xlabel('Date')
ax.set_ylabel('Avg Home Values')
ax.set_title("Median Housing Sales Forecast: ZipCode 07087 (Union City NJ)", fontsize = 35)
ax.yaxis.set_major_formatter(tick)
plt.savefig('images/union_city_forecast')

plt.legend()
plt.show()
```

executed in 278ms, finished 14:27:14 2022-05-24



18.7 Analysis:

- My model forecasts minimial losses in the lower value confidence index, with the potential for high gains.
- **I recommend this Zip Code.**

18.8 Metrics

- capturing metrics to compare with the top zip codes later.

In [720]:
```
zip_07087_ts.tail(1)
```

executed in 5ms, finished 14:27:14 2022-05-24

Out[720]:

	value
time	
2018-04-01	448,900.00

In [721]:
```
analysis_df = pred_conf.resample('Y').mean()
analysis_df['mean'] = analysis_df.apply(lambda x: x.mean(), axis=1)

analysis_df['base'] = 448900
analysis_df['base'] = analysis_df['base'].astype(float)
```

executed in 6ms, finished 14:27:14 2022-05-24

In [722]:
```
analysis_df = analysis_df[['base', 'lower value', 'mean', 'upper value']]

analysis_df['Min_ROI'] = analysis_df.apply(lambda x: x['lower value'] - x['base'], axis=1)
analysis_df['Min_ROI%'] = analysis_df.apply(lambda x: x['Min_ROI'] / x['base'], axis=1)
analysis_df['Forecast_ROI'] = analysis_df.apply(lambda x: x['mean'] - x['base'], axis=1)
analysis_df['Forecast_ROI%'] = analysis_df.apply(lambda x: x['mean'] / x['base'], axis=1)
analysis_df['Max_ROI'] = analysis_df.apply(lambda x: x['upper value'] - x['base'], axis=1)
analysis_df['Max_ROI%'] = analysis_df.apply(lambda x: x['upper value'] / x['base'], axis=1)
```

executed in 10ms, finished 14:27:14 2022-05-24

In [723]:
```
zip_07087_metrics = analysis_df
```

executed in 2ms, finished 14:27:14 2022-05-24

In [724]:
```
zip_07087_metrics
```

executed in 7ms, finished 14:27:14 2022-05-24

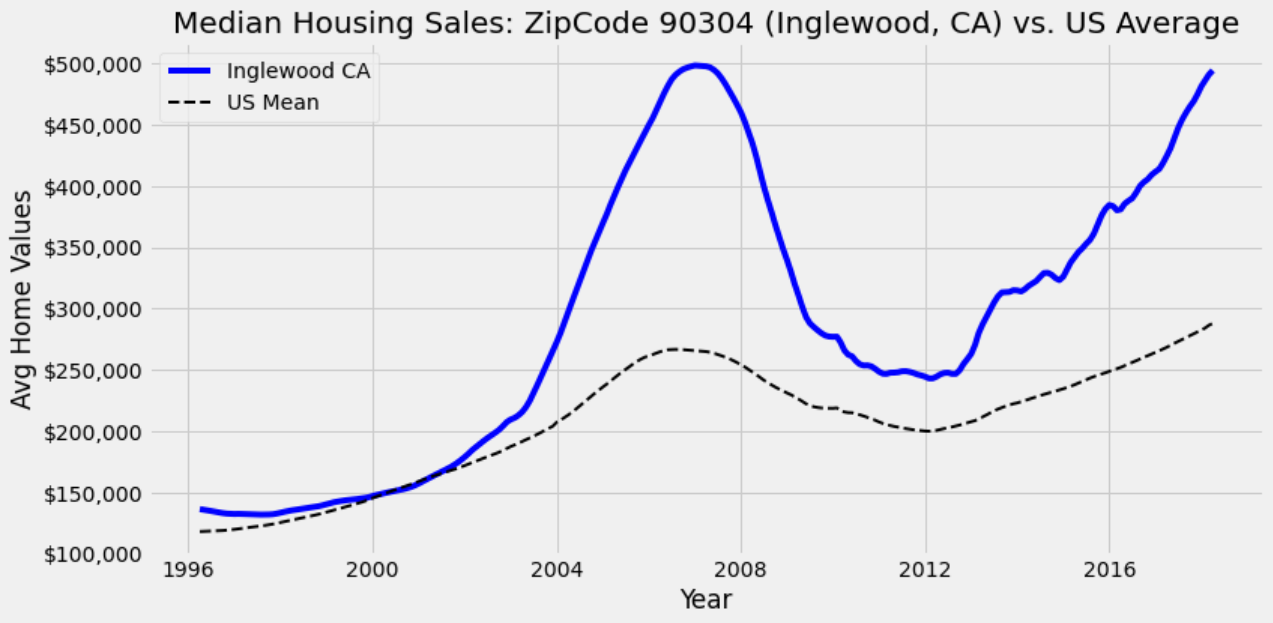
Out[724]:

	base	lower value	mean	upper value	Min_ROI	Min_ROI%	Forecast_ROI	Forecast_ROI%	Max_ROI	Max_ROI%
2018-12-31	448,900.00	450,172.27	475,146.70	500,121.13	1,272.27	0.00	26,246.70	1.06	51,221.13	1.11
2019-12-31	448,900.00	477,799.93	553,457.43	629,114.93	28,899.93	0.06	104,557.43	1.23	180,214.93	1.40
2020-12-31	448,900.00	503,549.54	656,978.95	810,408.37	54,649.54	0.12	208,078.95	1.46	361,508.37	1.81
2021-12-31	448,900.00	515,409.16	731,392.17	947,375.18	66,509.16	0.15	282,492.17	1.63	498,475.18	2.11

19 90304: Inglewood CA

In [725]:
```
fig, ax = plt.subplots(figsize=(12, 6))
ax.plot(zip_90304_ts, label='Inglewood CA', color='blue')
line1 = sns.lineplot(data=avg_home_df, x='time', y='home_value', ls='--',
                     color='black', linewidth=(2), label='US Mean');

ax.set_xlabel('Year')
ax.set_ylabel('Avg Home Values')
ax.set_title("Median Housing Sales: ZipCode 90304 (Inglewood, CA) vs. US Average", fontsize = 20)
ax.yaxis.set_major_formatter(tick)
fig.tight_layout()
```

executed in 176ms, finished 14:27:14 2022-05-24



In [726]:
```
#SARIMA Combos: ((1, 1, 1), (1, 1, 1, 12)) AIC: 4153.772501773967
zip_90304_combo_1 = ((1, 1, 1), (1, 1, 1, 12))
```

executed in 2ms, finished 14:27:14 2022-05-24

In [727]:
```
#SARIMA Combos: ((2, 2, 2), (0, 2, 2, 12)) AIC: 3738.2085627898796
zip_90304_combo_2 = ((2, 2, 2), (0, 2, 2, 12))
```

executed in 2ms, finished 14:27:14 2022-05-24

19.1 Model #1

In [728]:
```
combo = zip_90304_combo_1
combo
```

executed in 2ms, finished 14:27:14 2022-05-24

Out[728]:
```
((1, 1, 1), (1, 1, 1, 12))
```

In [729]:
```
# Manually split data.
temp_ts = zip_90304_ts

SPLIT = int(temp_ts.shape[0]*0.80)
train, test = temp_ts[:SPLIT], temp_ts[SPLIT:]

# Model
sarima_model = SARIMAX(
    train,
    order=combo[0],
    seasonal_order=combo[1],
    enforce_stationarity=False,
    enforce_invertibility=False).fit()
sarima_model.summary()
```

executed in 379ms, finished 14:27:15 2022-05-24

Out[729]:

SARIMAX Results

Dep. Variable:	value	**No. Observations:**	212
Model:	SARIMAX(1, 1, 1)x(1, 1, 1, 12)	**Log Likelihood**	-1550.936
Date:	Tue, 24 May 2022	**AIC**	3111.872
Time:	14:27:15	**BIC**	3127.974
Sample:	04-01-1996	**HQIC**	3118.397
	- 11-01-2013		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	0.9327	0.036	26.076	0.000	0.863	1.003
ma.L1	0.6061	0.056	10.838	0.000	0.496	0.716
ar.S.L12	0.1775	0.110	1.607	0.108	-0.039	0.394
ma.S.L12	-0.5168	0.088	-5.899	0.000	-0.688	-0.345
sigma2	1.345e+06	1.19e+05	11.288	0.000	1.11e+06	1.58e+06

Ljung-Box (L1) (Q):	0.19	**Jarque-Bera (JB):**	258.39
Prob(Q):	0.66	**Prob(JB):**	0.00
Heteroskedasticity (H):	9.01	**Skew:**	-0.72
Prob(H) (two-sided):	0.00	**Kurtosis:**	8.61

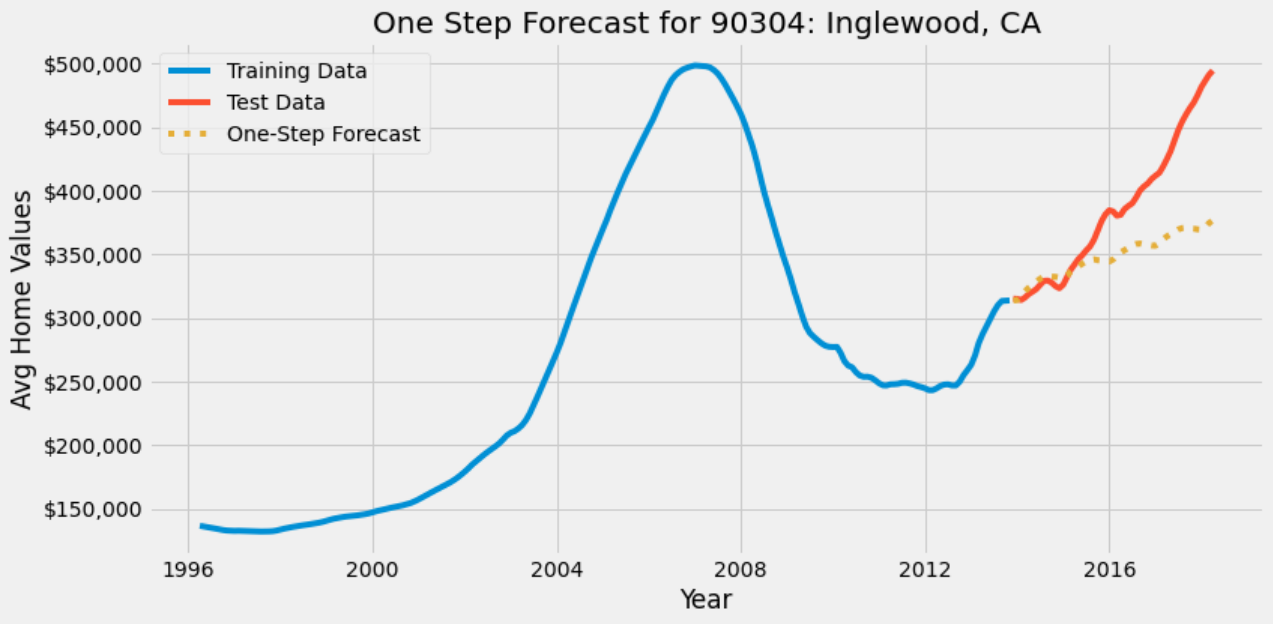
Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).

In [730]:
```
zip_90304_model_1 = sarima_model
```

executed in 5ms, finished 14:27:15 2022-05-24

In [731]:
```
pred = zip_90304_model_1.get_prediction(
    start=test.index.min(),
    end=test.index.max(),
    dynamic=False)

fig, ax = plt.subplots(figsize=(12, 6))
ax.plot(train, label='Training Data')
ax.plot(test, label='Test Data')
ax.plot(pred.predicted_mean, label='One-Step Forecast', ls=':')
#line1 = sns.lineplot(data=avg_home_df, x='time', y='home_value', color='black', label='US Mean');
ax.set_xlabel('Year')
ax.set_ylabel('Avg Home Values')
ax.set_title("One Step Forecast for 90304: Inglewood, CA", fontsize = 20)
ax.legend()
plt.savefig('images/inglewood_one_step_1')
ax.yaxis.set_major_formatter(tick)
fig.tight_layout()
```

executed in 221ms, finished 14:27:15 2022-05-24



In [732]:
```
y_forecasted = pred.predicted_mean
y_truth = test['value']
mse = ((y_forecasted - y_truth) ** 2).mean()
print('The Mean Squared Error is {}'.format(round(mse, 2)))
print('The Root Mean Squared Error is {}'.format(round(np.sqrt(mse), 2)))
```

executed in 4ms, finished 14:27:15 2022-05-24

```
The Mean Squared Error is 2802962940.02
The Root Mean Squared Error is 52943.02
```

19.2 Model #2

In [733]:
```
#SARIMA Combos: ((2, 2, 2), (0, 2, 2, 12)) AIC: 3738.2085627898796
zip_90304_combo_2 = ((2, 2, 2), (0, 2, 2, 12))
```
executed in 2ms, finished 14:27:15 2022-05-24

In [734]:
```
combo = zip_90304_combo_2
combo
```
executed in 3ms, finished 14:27:15 2022-05-24

Out[734]: ((2, 2, 2), (0, 2, 2, 12))

In [735]:
```
# Manually split data.
temp_ts = zip_90304_ts

SPLIT = int(temp_ts.shape[0]*0.80)
train, test = temp_ts[:SPLIT], temp_ts[SPLIT:]

# Model
sarima_model = SARIMAX(
    train,
    order=combo[0],
    seasonal_order=combo[1],
    enforce_stationarity=False,
    enforce_invertibility=False).fit()
sarima_model.summary()
```
executed in 5.75s, finished 14:27:21 2022-05-24

Out[735]:

SARIMAX Results

Dep. Variable:	value	**No. Observations:**	212
Model:	SARIMAX(2, 2, 2)x(0, 2, 2, 12)	**Log Likelihood**	-1354.020
Date:	Tue, 24 May 2022	**AIC**	2722.041
Time:	14:27:21	**BIC**	2743.523
Sample:	04-01-1996	**HQIC**	2730.765
	- 11-01-2013		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	-0.8043	0.058	-13.902	0.000	-0.918	-0.691
ar.L2	-0.3452	0.040	-8.718	0.000	-0.423	-0.268
ma.L1	1.7510	0.061	28.622	0.000	1.631	1.871
ma.L2	1.0338	0.076	13.565	0.000	0.884	1.183
ma.S.L12	-1.1472	0.098	-11.749	0.000	-1.339	-0.956
ma.S.L24	0.0970	0.043	2.267	0.023	0.013	0.181
sigma2	8.613e+05	8.45e-08	1.02e+13	0.000	8.61e+05	8.61e+05

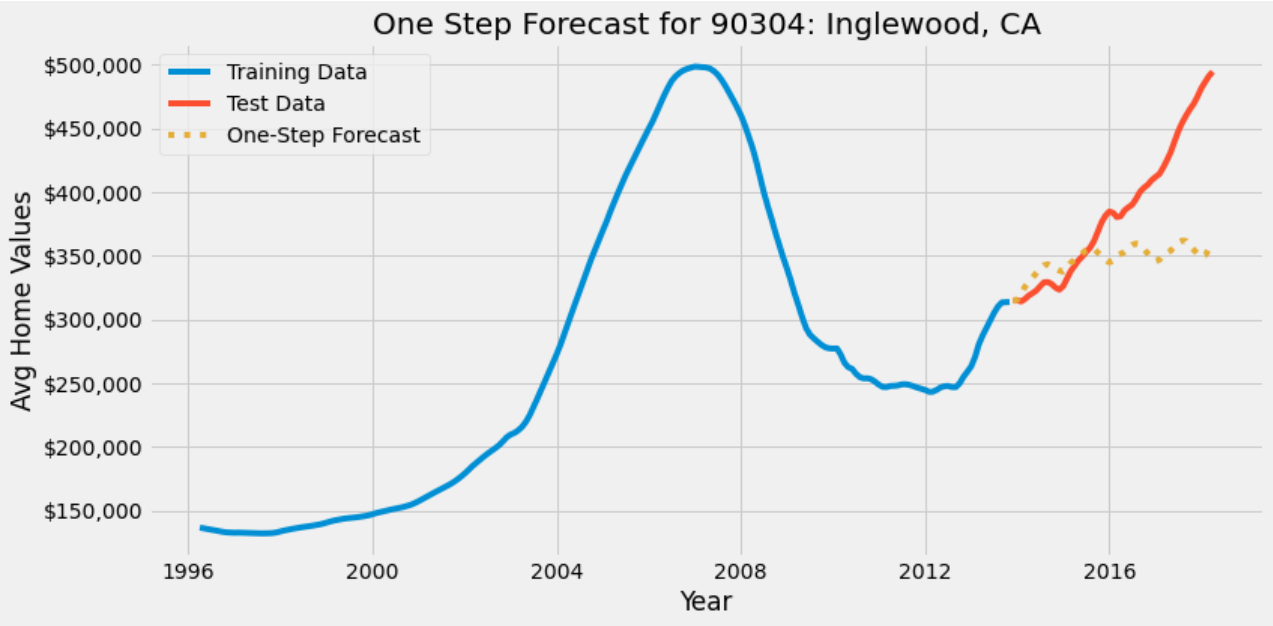
Ljung-Box (L1) (Q):	1.27	**Jarque-Bera (JB):**	482.11
Prob(Q):	0.26	**Prob(JB):**	0.00
Heteroskedasticity (H):	5.33	**Skew:**	0.13
Prob(H) (two-sided):	0.00	**Kurtosis:**	11.53

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
[2] Covariance matrix is singular or near-singular, with condition number 2.32e+28. Standard errors may be unstable.

In [736]:
```
zip_90304_model_2 = sarima_model
```
executed in 3ms, finished 14:27:21 2022-05-24

In [737]:
```
pred = zip_90304_model_2.get_prediction(
    start=test.index.min(),
    end=test.index.max(),
    dynamic=False)

fig, ax = plt.subplots(figsize=(12, 6))
ax.plot(train, label='Training Data')
ax.plot(test, label='Test Data')
ax.plot(pred.predicted_mean, label='One-Step Forecast', ls=':')
#line1 = sns.lineplot(data=avg_home_df, x='time', y='home_value', color='black', label='US Mean');
ax.set_xlabel('Year')
ax.set_ylabel('Avg Home Values')
ax.set_title("One Step Forecast for 90304: Inglewood, CA", fontsize = 20)
ax.legend()
plt.savefig('images/inglewood_one_step_2')
ax.yaxis.set_major_formatter(tick)
fig.tight_layout()
```

executed in 225ms, finished 14:27:21 2022-05-24



In [738]:
```
y_forecasted = pred.predicted_mean
y_truth = test['value']
mse = ((y_forecasted - y_truth) ** 2).mean()
print('The Mean Squared Error is {}'.format(round(mse, 2)))
print('The Root Mean Squared Error is {}'.format(round(np.sqrt(mse), 2)))
```

executed in 4ms, finished 14:27:21 2022-05-24

```
The Mean Squared Error is 3669561267.0
The Root Mean Squared Error is 60576.9
```

19.3 Model Comparison & Decision

- Model 1 has a slightly lower RMSE and better matches the test data, so I will use that one for my predictions.

19.4 Prediction

In [739]:
```
combo = zip_90304_combo_1
#combo = zip_90304_combo_2
combo
```

executed in 4ms, finished 14:27:21 2022-05-24

Out[739]: ((1, 1, 1), (1, 1, 1, 12))

In [740]:
```
# Running model again with FULL data

sarima_model = SARIMAX(
    zip_90304_ts,
    order=combo[0],
    seasonal_order=combo[1],
    enforce_stationarity=False,
    enforce_invertibility=False).fit()
sarima_model.summary()
```

executed in 557ms, finished 14:27:22 2022-05-24

Out[740]:

SARIMAX Results

Dep. Variable:	value	**No. Observations:**	265
Model:	SARIMAX(1, 1, 1)x(1, 1, 1, 12)	**Log Likelihood**	-2065.138
Date:	Tue, 24 May 2022	**AIC**	4140.276
Time:	14:27:22	**BIC**	4157.637
Sample:	04-01-1996	**HQIC**	4147.272
	- 04-01-2018		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	0.9149	0.031	29.201	0.000	0.853	0.976
ma.L1	0.3350	0.027	12.426	0.000	0.282	0.388
ar.S.L12	-0.4683	0.063	-7.447	0.000	-0.592	-0.345
ma.S.L12	-0.0416	0.038	-1.086	0.277	-0.117	0.033
sigma2	2.319e+06	1.9e+05	12.184	0.000	1.95e+06	2.69e+06

Ljung-Box (L1) (Q):	6.10	**Jarque-Bera (JB):**	46.21
Prob(Q):	0.01	**Prob(JB):**	0.00
Heteroskedasticity (H):	8.46	**Skew:**	0.24
Prob(H) (two-sided):	0.00	**Kurtosis:**	5.10

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).

In [741]:
```
zip_90304_model_full = sarima_model
```

executed in 3ms, finished 14:27:22 2022-05-24

In [742]:
```
zip_90304_ts.tail(1)
```

executed in 5ms, finished 14:27:22 2022-05-24

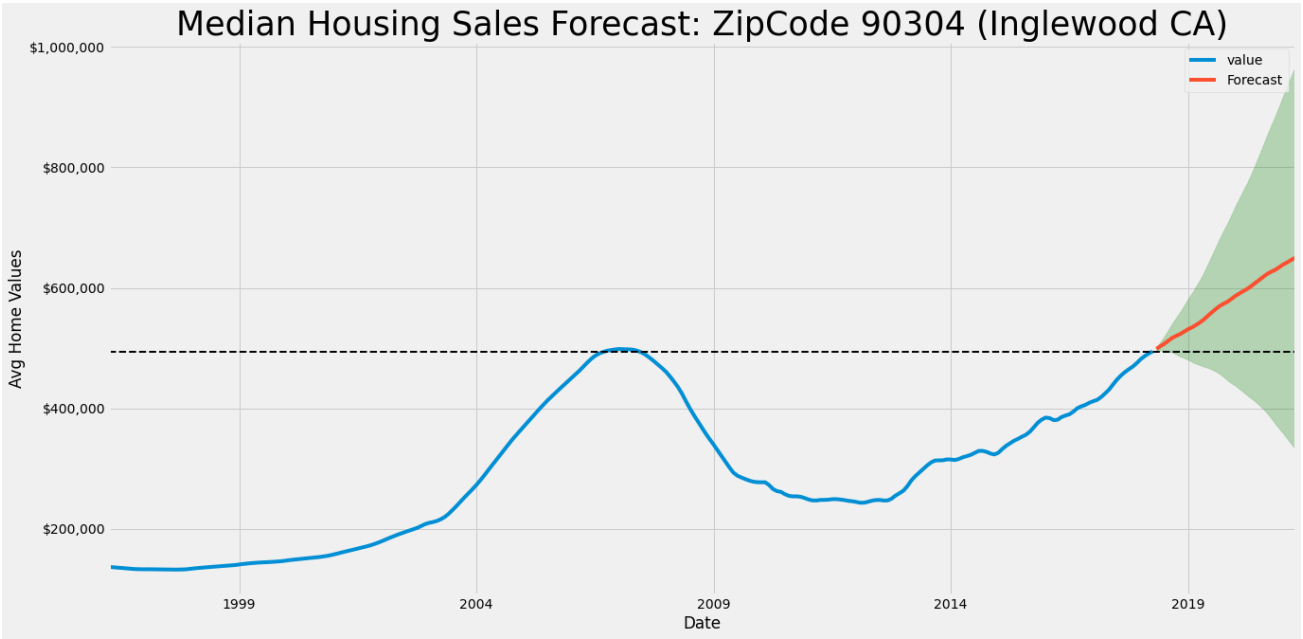
Out[742]:

time	value
2018-04-01	494,300.00

In [743]:

```
# Get forecast 3 yrs ahead in future (36 steps)
prediction = zip_90304_model_full.get_forecast(steps=36)

# Get confidence intervals of forecasts
pred_conf = prediction.conf_int()

# Plot future predictions with confidence intervals
ax = zip_90304_ts['1996-01':].plot(label='observed', figsize=(20, 10))
prediction.predicted_mean.plot(ax=ax, label='Forecast')
ax.fill_between(pred_conf.index,
                pred_conf.iloc[:, 0],
                pred_conf.iloc[:, 1], color='green', alpha=0.25)

ax.axhline(494300, ls='--', color='black', linewidth=(2))

ax.set_xlabel('Date')
ax.set_ylabel('Avg Home Values')
ax.set_title("Median Housing Sales Forecast: ZipCode 90304 (Inglewood CA)", fontsize = 35)
ax.yaxis.set_major_formatter(tick)
plt.savefig('images/inglewood_forecast')

plt.legend()
plt.show()
```

executed in 272ms, finished 14:27:22 2022-05-24



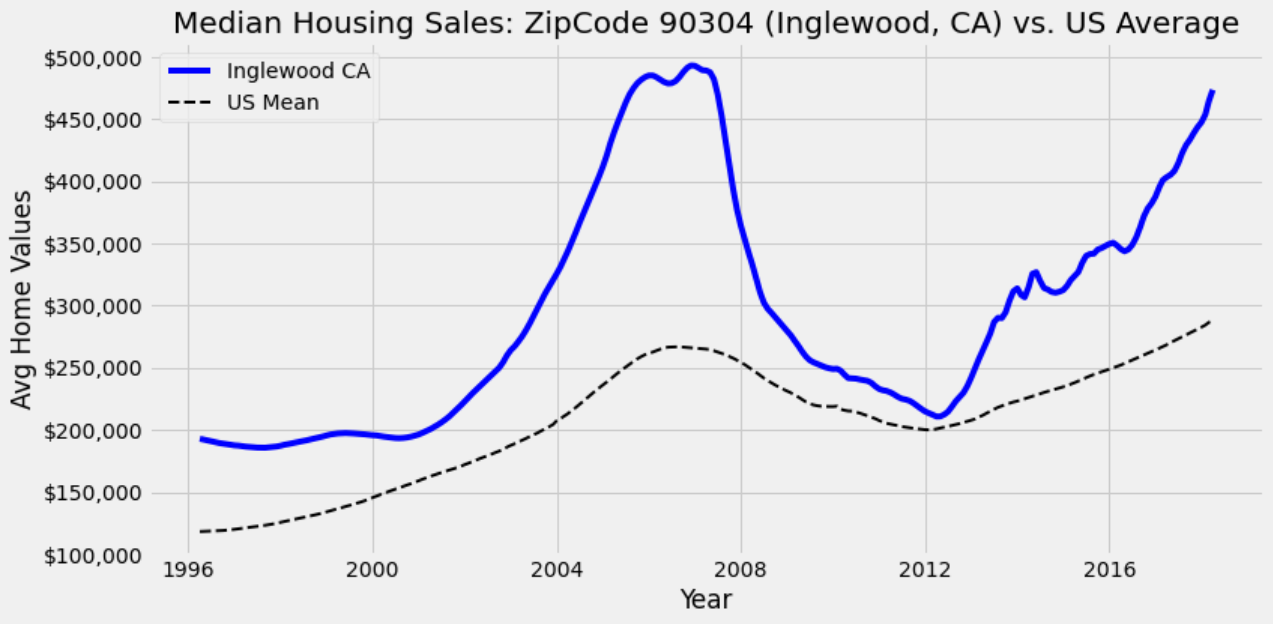
19.5 Analysis:

- The lower confidence level predicts a loss, so I do not recommend investing in this zip code.
- That said, most of the confidence index is above the current value, so if the company is okay with taking on a little more risk, this would be a good investment with high potential for large ROI.

20 95918: Brown's Valley CA

In [744]:
```
fig, ax = plt.subplots(figsize=(12, 6))
ax.plot(zip_95918_ts, label='Inglewood CA', color='blue')
line1 = sns.lineplot(data=avg_home_df, x='time', y='home_value', ls='--',
                     color='black', linewidth=(2), label='US Mean');

ax.set_xlabel('Year')
ax.set_ylabel('Avg Home Values')
ax.set_title("Median Housing Sales: ZipCode 90304 (Inglewood, CA) vs. US Average", fontsize = 20)
ax.yaxis.set_major_formatter(tick)
fig.tight_layout()
```

executed in 177ms, finished 14:27:22 2022-05-24



In [745]:
```
#SARIMA Combos: ((1, 1, 1), (1, 1, 1, 12)) AIC: 4153.772501773967
zip_95918_combo_1 = ((1, 1, 1), (1, 1, 1, 12))
```

executed in 3ms, finished 14:27:22 2022-05-24

20.1 Model #1

In [746]:
```
combo = zip_95918_combo_1
combo
```

executed in 3ms, finished 14:27:22 2022-05-24

Out[746]:
```
((1, 1, 1), (1, 1, 1, 12))
```

In [747]:
```
# Manually split data.
temp_ts = zip_95918_ts

SPLIT = int(temp_ts.shape[0]*0.80)
train, test = temp_ts[:SPLIT], temp_ts[SPLIT:]

# Model
sarima_model = SARIMAX(
    train,
    order=combo[0],
    seasonal_order=combo[1],
    enforce_stationarity=False,
    enforce_invertibility=False).fit()
sarima_model.summary()
```

executed in 310ms, finished 14:27:22 2022-05-24

Out[747]:

SARIMAX Results

Dep. Variable:	value	**No. Observations:**	212
Model:	SARIMAX(1, 1, 1)x(1, 1, 1, 12)	**Log Likelihood**	-1666.400
Date:	Tue, 24 May 2022	**AIC**	3342.800
Time:	14:27:22	**BIC**	3358.901
Sample:	04-01-1996	**HQIC**	3349.325
	- 11-01-2013		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	0.9467	0.027	35.606	0.000	0.895	0.999
ma.L1	0.0347	0.032	1.101	0.271	-0.027	0.097
ar.S.L12	-0.7033	0.017	-42.028	0.000	-0.736	-0.670
ma.S.L12	1.6258	0.067	24.231	0.000	1.494	1.757
sigma2	1.971e+06	1.91e+05	10.330	0.000	1.6e+06	2.34e+06

Ljung-Box (L1) (Q):	59.18	**Jarque-Bera (JB):**	191.69
Prob(Q):	0.00	**Prob(JB):**	0.00
Heteroskedasticity (H):	1.92	**Skew:**	-0.04
Prob(H) (two-sided):	0.01	**Kurtosis:**	7.99

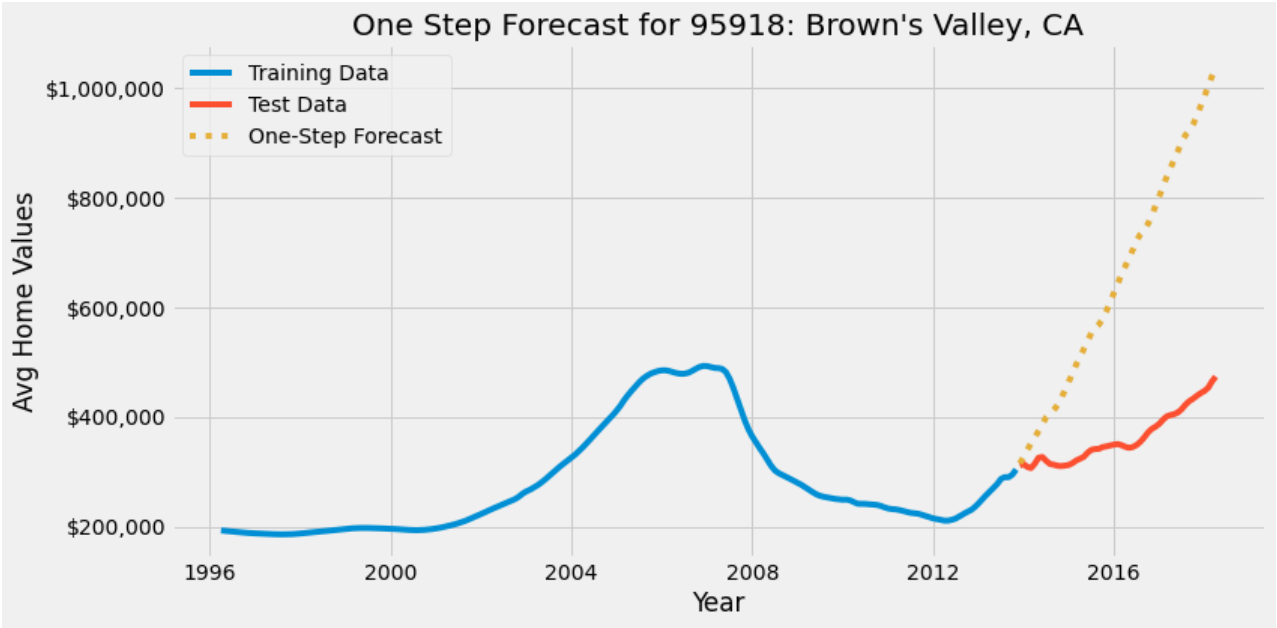
Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).

In [748]:
```
zip_95918_model_1 = sarima_model
```

executed in 4ms, finished 14:27:22 2022-05-24

In [749]:
```
pred = zip_95918_model_1.get_prediction(
    start=test.index.min(),
    end=test.index.max(),
    dynamic=False)

fig, ax = plt.subplots(figsize=(12, 6))
ax.plot(train, label='Training Data')
ax.plot(test, label='Test Data')
ax.plot(pred.predicted_mean, label='One-Step Forecast', ls=':')
#line1 = sns.lineplot(data=avg_home_df, x='time', y='home_value', color='black', label='US Mean');
ax.set_xlabel('Year')
ax.set_ylabel('Avg Home Values')
ax.set_title("One Step Forecast for 95918: Brown's Valley, CA", fontsize = 20)
ax.legend()
plt.savefig('images/brownsvalley_one_step_1')

ax.yaxis.set_major_formatter(tick)
fig.tight_layout()
```

executed in 215ms, finished 14:27:23 2022-05-24



In [750]:
```
y_forecasted = pred.predicted_mean
y_truth = test['value']
mse = ((y_forecasted - y_truth) ** 2).mean()
print('The Mean Squared Error is {}'.format(round(mse, 2)))
print('The Root Mean Squared Error is {}'.format(round(np.sqrt(mse), 2)))
```

executed in 4ms, finished 14:27:23 2022-05-24

```
The Mean Squared Error is 113915320918.45
The Root Mean Squared Error is 337513.44
```

In [ ]:
```
1
```

20.2 Model #2

In [751]:
```
1 #SARIMA Combos: ((2, 2, 0), (2, 2, 1, 12)) AIC: 3945.7191825608493
2 zip_95918_combo_2 = ((2, 2, 0), (2, 2, 1, 12))
```
executed in 2ms, finished 14:27:23 2022-05-24

In [752]:
```
1 combo = zip_95918_combo_2
2 combo
```
executed in 2ms, finished 14:27:23 2022-05-24

Out[752]:
```
((2, 2, 0), (2, 2, 1, 12))
```

In [753]:
```
1  # Manually split data.
2  temp_ts = zip_95918_ts
3
4  SPLIT = int(temp_ts.shape[0]*0.80)
5  train, test = temp_ts[:SPLIT], temp_ts[SPLIT:]
6
7  # Model
8  sarima_model = SARIMAX(
9      train,
10     order=combo[0],
11     seasonal_order=combo[1],
12     enforce_stationarity=False,
13     enforce_invertibility=False).fit()
14 sarima_model.summary()
```
executed in 2.17s, finished 14:27:25 2022-05-24

Out[753]:

SARIMAX Results

Dep. Variable:	value	**No. Observations:**	212
Model:	SARIMAX(2, 2, 0)x(2, 2, [1], 12)	**Log Likelihood**	-1410.123
Date:	Tue, 24 May 2022	**AIC**	2832.245
Time:	14:27:25	**BIC**	2850.696
Sample:	04-01-1996	**HQIC**	2839.738
	- 11-01-2013		
Covariance Type:	opg		

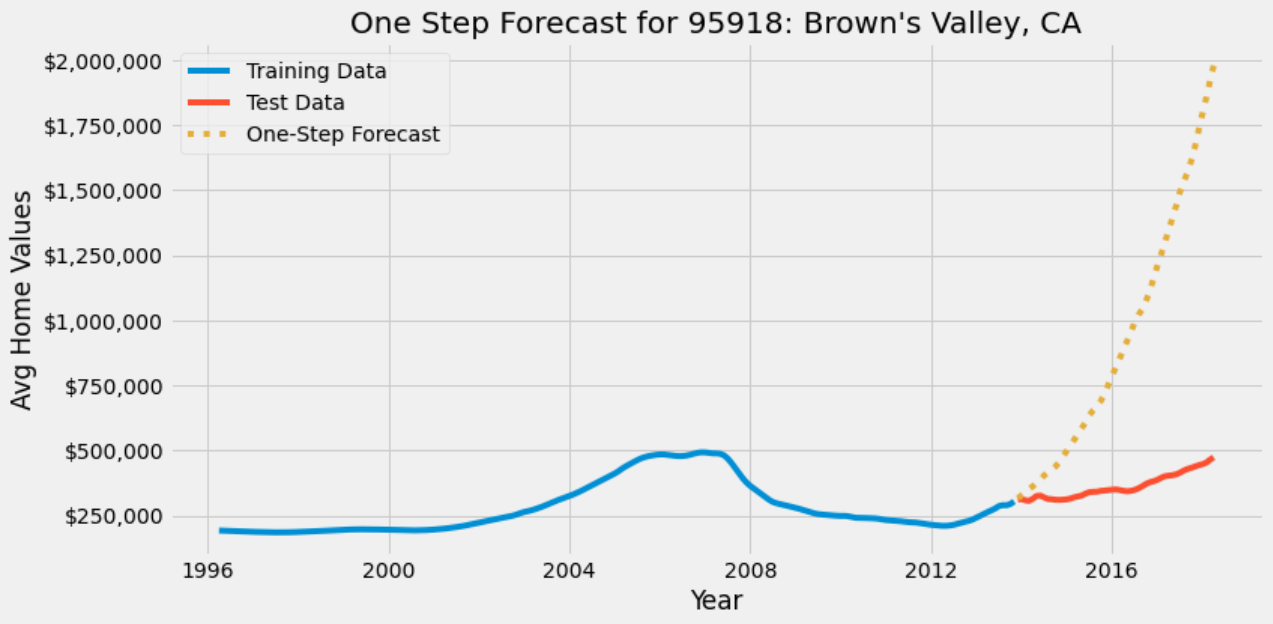
	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	0.9010	0.094	9.615	0.000	0.717	1.085
ar.L2	-0.4662	0.092	-5.044	0.000	-0.647	-0.285
ar.S.L12	-1.1460	0.156	-7.338	0.000	-1.452	-0.840
ar.S.L24	-0.5736	0.161	-3.560	0.000	-0.889	-0.258
ma.S.L12	-0.0198	0.102	-0.194	0.846	-0.220	0.181
sigma2	4.224e+06	4.73e+05	8.923	0.000	3.3e+06	5.15e+06

Ljung-Box (L1) (Q):	1.90	**Jarque-Bera (JB):**	464.10
Prob(Q):	0.17	**Prob(JB):**	0.00
Heteroskedasticity (H):	12.83	**Skew:**	-0.72
Prob(H) (two-sided):	0.00	**Kurtosis:**	11.22

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).

In [754]:
```
zip_95918_model_2 = sarima_model
```
executed in 4ms, finished 14:27:25 2022-05-24

In [755]:
```
pred = zip_95918_model_2.get_prediction(
    start=test.index.min(),
    end=test.index.max(),
    dynamic=False)

fig, ax = plt.subplots(figsize=(12, 6))
ax.plot(train, label='Training Data')
ax.plot(test, label='Test Data')
ax.plot(pred.predicted_mean, label='One-Step Forecast', ls=':')
#line1 = sns.lineplot(data=avg_home_df, x='time', y='home_value', color='black', label='US Mean');
ax.set_xlabel('Year')
ax.set_ylabel('Avg Home Values')
ax.set_title("One Step Forecast for 95918: Brown's Valley, CA", fontsize = 20)
ax.legend()
plt.savefig('images/browns_valley_one_step_2')
ax.yaxis.set_major_formatter(tick)
fig.tight_layout()
```
executed in 214ms, finished 14:27:25 2022-05-24



In [756]:
```
y_forecasted = pred.predicted_mean
y_truth = test['value']
mse = ((y_forecasted - y_truth) ** 2).mean()
print('The Mean Squared Error is {}'.format(round(mse, 2)))
print('The Root Mean Squared Error is {}'.format(round(np.sqrt(mse), 2)))
```
executed in 3ms, finished 14:27:25 2022-05-24

```
The Mean Squared Error is 515246780918.21
The Root Mean Squared Error is 717806.92
```

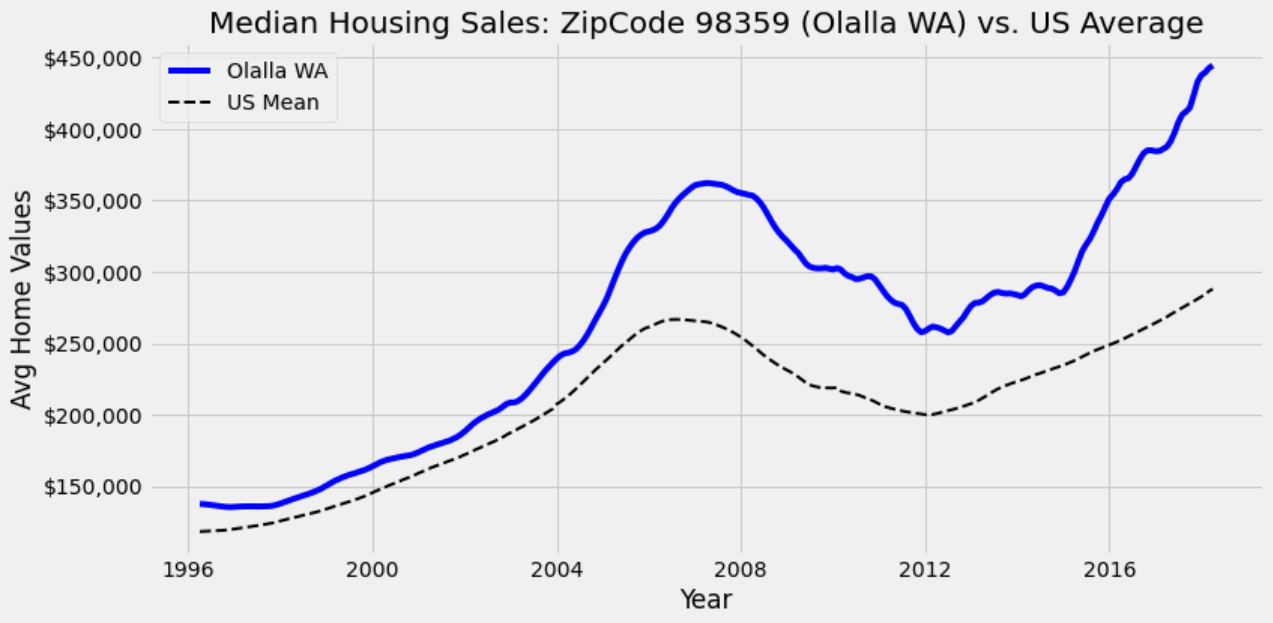
20.3 Model Comparison & Decision:

- Both models seem to be overfit on the data and are projecting an upward trend that doesn't seem to match the known data.
- Both also have high RSMEs relative to the other models that I have evaluated.
- I do not recommend this Zip Code, and don't even trust my models enough to run a prediction.
- **Skip and move on to the next Zip Code**

21 98359: Olalla WA

In [757]:
```
fig, ax = plt.subplots(figsize=(12, 6))
ax.plot(zip_98359_ts, label='Olalla WA', color='blue')
line1 = sns.lineplot(data=avg_home_df, x='time', y='home_value', ls='--',
                     color='black', linewidth=(2), label='US Mean');

ax.set_xlabel('Year')
ax.set_ylabel('Avg Home Values')
ax.set_title("Median Housing Sales: ZipCode 98359 (Olalla WA) vs. US Average", fontsize = 20)
ax.yaxis.set_major_formatter(tick)
fig.tight_layout()
```

executed in 168ms, finished 14:27:25 2022-05-24



In [758]:
```
#sarimax_param_search(zip_98359_ts)
```

executed in 1ms, finished 14:27:25 2022-05-24

In [759]:
```
#Optimal SARIMA Combo: ((1, 1, 1), (0, 1, 1, 12)) AIC: 4034.182214906454
zip_98359_combo_1 = ((1, 1, 1), (0, 1, 1, 12))
```

executed in 2ms, finished 14:27:25 2022-05-24

21.1 Model #1

In [760]:
```
combo = zip_98359_combo_1
combo
```

executed in 3ms, finished 14:27:25 2022-05-24

Out[760]:
```
((1, 1, 1), (0, 1, 1, 12))
```

In [761]:
```
# Manually split data.
temp_ts = zip_80102_ts

SPLIT = int(temp_ts.shape[0]*0.80)
train, test = temp_ts[:SPLIT], temp_ts[SPLIT:]

# Model
sarima_model = SARIMAX(
    train,
    order=combo[0],
    seasonal_order=combo[1],
    enforce_stationarity=False,
    enforce_invertibility=False).fit()
sarima_model.summary()
```

executed in 411ms, finished 14:27:26 2022-05-24

Out[761]:

SARIMAX Results

Dep. Variable:	value	**No. Observations:**	212
Model:	SARIMAX(1, 1, 1)x(0, 1, 1, 12)	**Log Likelihood**	-1525.927
Date:	Tue, 24 May 2022	**AIC**	3059.853
Time:	14:27:26	**BIC**	3072.734
Sample:	04-01-1996	**HQIC**	3065.074
	- 11-01-2013		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	0.7912	0.039	20.334	0.000	0.715	0.867
ma.L1	0.6517	0.042	15.401	0.000	0.569	0.735
ma.S.L12	-0.4040	0.035	-11.581	0.000	-0.472	-0.336
sigma2	7.04e+05	5.8e+04	12.139	0.000	5.9e+05	8.18e+05

Ljung-Box (L1) (Q):	3.68	**Jarque-Bera (JB):**	5.99
Prob(Q):	0.06	**Prob(JB):**	0.05
Heteroskedasticity (H):	4.61	**Skew:**	-0.00
Prob(H) (two-sided):	0.00	**Kurtosis:**	3.88

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).

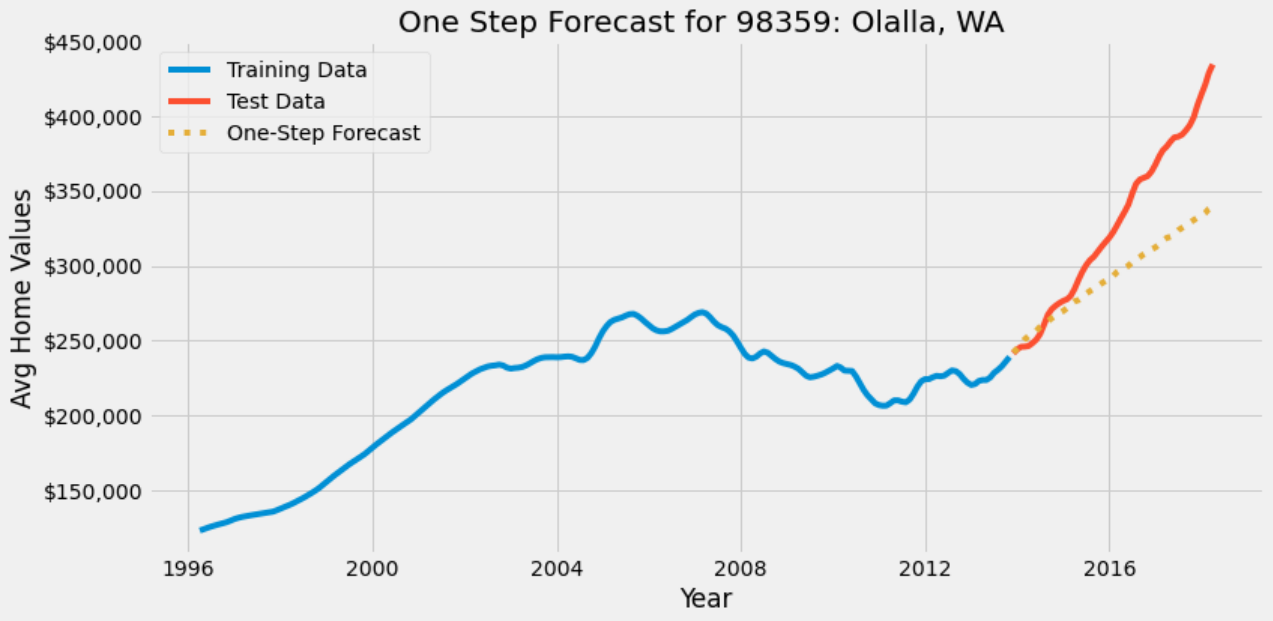
In [762]:
```
zip_98359_model_1 = sarima_model
```

executed in 2ms, finished 14:27:26 2022-05-24

In [763]:

```
pred = zip_98359_model_1.get_prediction(
    start=test.index.min(),
    end=test.index.max(),
    dynamic=False)

fig, ax = plt.subplots(figsize=(12, 6))
ax.plot(train, label='Training Data')
ax.plot(test, label='Test Data')
ax.plot(pred.predicted_mean, label='One-Step Forecast', ls=':')
#line1 = sns.lineplot(data=avg_home_df, x='time', y='home_value', color='black', label='US Mean');

ax.set_xlabel('Year')
ax.set_ylabel('Avg Home Values')
ax.set_title("One Step Forecast for 98359: Olalla, WA ", fontsize = 20)
ax.legend()
plt.savefig('images/olalla_one_step_1')
ax.yaxis.set_major_formatter(tick)
fig.tight_layout()
```

executed in 210ms, finished 14:27:26 2022-05-24



In [764]:
```
y_forecasted = pred.predicted_mean
y_truth = test['value']
mse = ((y_forecasted - y_truth) ** 2).mean()
print('The Mean Squared Error is {}'.format(round(mse, 2)))
print('The Root Mean Squared Error is {}'.format(round(np.sqrt(mse), 2)))
```

executed in 4ms, finished 14:27:26 2022-05-24

```
The Mean Squared Error is 1997230432.36
The Root Mean Squared Error is 44690.38
```

21.2 Model #2

In [765]:
```
#sarimax_param_search(zip_98359_ts)
```

executed in 1ms, finished 14:27:26 2022-05-24

In [766]:
```
#SARIMA Combos: ((2, 2, 2), (2, 2, 2, 12)) AIC: 3663.638994204836
zip_98359_combo_2 = ((2, 2, 2), (2, 2, 2, 12))
```

executed in 2ms, finished 14:27:26 2022-05-24

In [767]:
```
combo = zip_98359_combo_2
combo
```

executed in 3ms, finished 14:27:26 2022-05-24

Out[767]:
```
((2, 2, 2), (2, 2, 2, 12))
```

In [768]:
```
# Manually split data.
temp_ts = zip_98359_ts

SPLIT = int(temp_ts.shape[0]*0.80)
train, test = temp_ts[:SPLIT], temp_ts[SPLIT:]

# Model
sarima_model = SARIMAX(
    train,
    order=combo[0],
    seasonal_order=combo[1],
    enforce_stationarity=False,
    enforce_invertibility=False).fit()
sarima_model.summary()
```

executed in 6.16s, finished 14:27:32 2022-05-24

Out[768]:

SARIMAX Results

Dep. Variable:	value	**No. Observations:**	212
Model:	SARIMAX(2, 2, 2)x(2, 2, 2, 12)	**Log Likelihood**	-1330.031
Date:	Tue, 24 May 2022	**AIC**	2678.062
Time:	14:27:32	**BIC**	2705.682
Sample:	04-01-1996	**HQIC**	2689.278
	- 11-01-2013		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	1.1509	0.103	11.181	0.000	0.949	1.353
ar.L2	-0.4443	0.080	-5.536	0.000	-0.602	-0.287
ma.L1	-0.4194	0.107	-3.929	0.000	-0.629	-0.210
ma.L2	-0.2772	0.085	-3.266	0.001	-0.444	-0.111
ar.S.L12	-1.0968	0.149	-7.363	0.000	-1.389	-0.805
ar.S.L24	-0.4728	0.099	-4.788	0.000	-0.666	-0.279
ma.S.L12	-0.0801	0.110	-0.729	0.466	-0.295	0.135
ma.S.L24	0.0151	0.132	0.114	0.909	-0.244	0.275
sigma2	1.047e+06	8.59e+04	12.180	0.000	8.78e+05	1.22e+06

Ljung-Box (L1) (Q):	0.85	**Jarque-Bera (JB):**	48.66
Prob(Q):	0.36	**Prob(JB):**	0.00
Heteroskedasticity (H):	7.09	**Skew:**	-0.08
Prob(H) (two-sided):	0.00	**Kurtosis:**	5.71

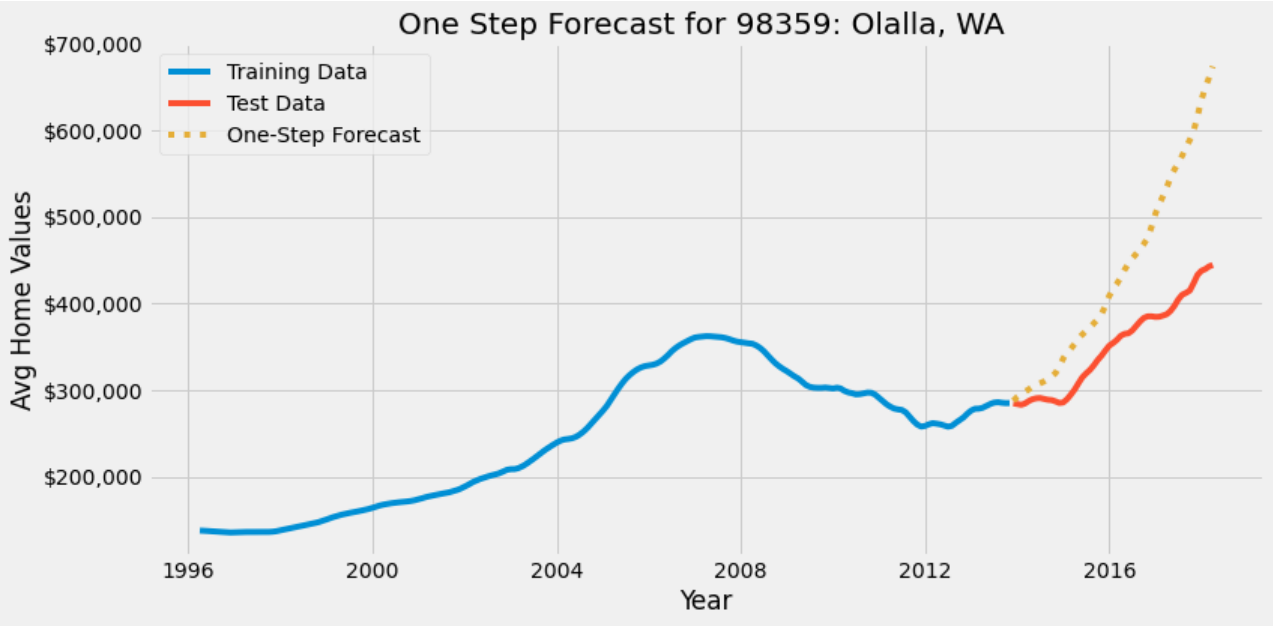
Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).

In [769]:
```
zip_98359_model_2 = sarima_model
```

executed in 4ms, finished 14:27:32 2022-05-24

In [770]:
```
pred = zip_98359_model_2.get_prediction(
    start=test.index.min(),
    end=test.index.max(),
    dynamic=False)

fig, ax = plt.subplots(figsize=(12, 6))
ax.plot(train, label='Training Data')
ax.plot(test, label='Test Data')
ax.plot(pred.predicted_mean, label='One-Step Forecast', ls=':')
#line1 = sns.lineplot(data=avg_home_df, x='time', y='home_value', color='black', label='US Mean');

ax.set_xlabel('Year')
ax.set_ylabel('Avg Home Values')
ax.set_title("One Step Forecast for 98359: Olalla, WA ", fontsize = 20)
ax.legend()
plt.savefig('images/olalla_one_step_2')
ax.yaxis.set_major_formatter(tick)
fig.tight_layout()
```

executed in 214ms, finished 14:27:32 2022-05-24



In [771]:
```
y_forecasted = pred.predicted_mean
y_truth = test['value']
mse = ((y_forecasted - y_truth) ** 2).mean()
print('The Mean Squared Error is {}'.format(round(mse, 2)))
print('The Root Mean Squared Error is {}'.format(round(np.sqrt(mse), 2)))
```

executed in 4ms, finished 14:27:32 2022-05-24

```
The Mean Squared Error is 11100001106.79
The Root Mean Squared Error is 105356.54
```

21.3 Model Comparison & Decision:

- Model 1 has a significantly lower RMSE, and the one-step forecast for Model 2 seems overfit, so **I will use that Model #1 for predictions.**

21.4 Prediction

In [772]:
```
combo = zip_98359_combo_1
#combo = zip_98359_combo_2
combo
```

executed in 3ms, finished 14:27:32 2022-05-24

Out[772]:
```
((1, 1, 1), (0, 1, 1, 12))
```

In [773]:
```
# Running model again with FULL data

sarima_model = SARIMAX(
    zip_98359_ts,
    order=combo[0],
    seasonal_order=combo[1],
    enforce_stationarity=False,
    enforce_invertibility=False).fit()
sarima_model.summary()
```

executed in 550ms, finished 14:27:33 2022-05-24

Out[773]:

SARIMAX Results

Dep. Variable:	value	**No. Observations:**	265
Model:	SARIMAX(1, 1, 1)x(0, 1, 1, 12)	**Log Likelihood**	-2013.091
Date:	Tue, 24 May 2022	**AIC**	4034.182
Time:	14:27:33	**BIC**	4048.071
Sample:	04-01-1996	**HQIC**	4039.780
	- 04-01-2018		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	0.7581	0.039	19.286	0.000	0.681	0.835
ma.L1	0.6919	0.039	17.688	0.000	0.615	0.769
ma.S.L12	-0.3228	0.032	-10.096	0.000	-0.385	-0.260
sigma2	1.14e+06	5.84e+04	19.517	0.000	1.03e+06	1.25e+06

Ljung-Box (L1) (Q):	1.86	**Jarque-Bera (JB):**	227.44
Prob(Q):	0.17	**Prob(JB):**	0.00
Heteroskedasticity (H):	6.19	**Skew:**	1.02
Prob(H) (two-sided):	0.00	**Kurtosis:**	7.33

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).

In [774]:
```
zip_98359_model_full = sarima_model
```

executed in 4ms, finished 14:27:33 2022-05-24

In [775]:
```
zip_98359_ts.tail(1)
```

executed in 4ms, finished 14:27:33 2022-05-24

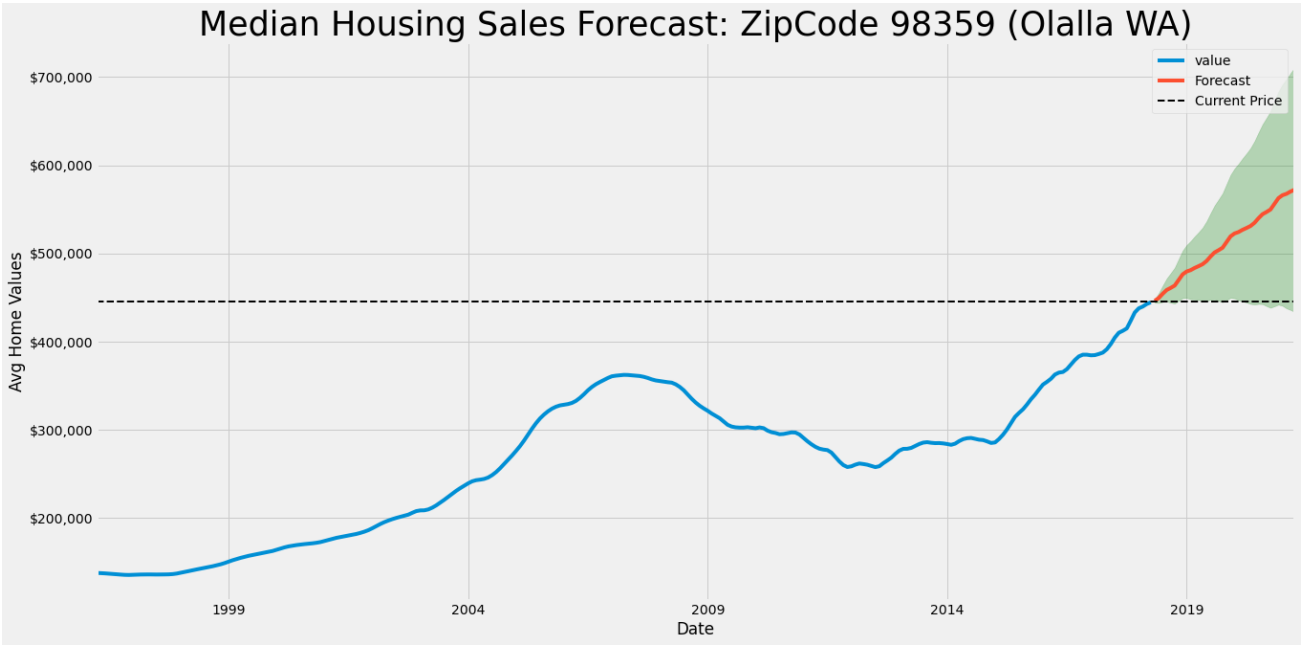
Out[775]:

time	value
2018-04-01	444,600.00

In [776]:

```
# Get forecast 3 yrs ahead in future (36 steps)
prediction = zip_98359_model_full.get_forecast(steps=36)

# Get confidence intervals of forecasts
pred_conf = prediction.conf_int()

# Plot future predictions with confidence intervals
ax = zip_98359_ts['1996-01':].plot(label='observed', figsize=(20, 10))

prediction.predicted_mean.plot(ax=ax, label='Forecast')
ax.fill_between(pred_conf.index,
                pred_conf.iloc[:, 0],
                pred_conf.iloc[:, 1], color='green', alpha=0.25)

ax.axhline(446000, ls='--', color='black', linewidth=(2), label='Current Price')

ax.set_xlabel('Date')
ax.set_ylabel('Avg Home Values')
ax.set_title("Median Housing Sales Forecast: ZipCode 98359 (Olalla WA)", fontsize = 35)
ax.yaxis.set_major_formatter(tick)
plt.savefig('images/olalla_forecast')

plt.legend()
plt.show()
```

executed in 271ms, finished 14:27:33 2022-05-24



21.5 Analysis:

- Confidence interval low value has minimal losses, but basically breaking even, while there is a high chance of gains and a relatively narrow confidence window.
- **I highly recommend investing in this zip code!**

21.6 Metrics

- capturing metrics to compare with the top zip codes later.

In [777]:
```
zip_98359_ts.tail(1)
```
executed in 4ms, finished 14:27:33 2022-05-24

Out[777]:

time	value
2018-04-01	444,600.00

In [778]:
```
analysis_df = pred_conf.resample('Y').mean()
analysis_df['mean'] = analysis_df.apply(lambda x: x.mean(), axis=1)

analysis_df['base'] = 444600
analysis_df['base'] = analysis_df['base'].astype(float)
```
executed in 6ms, finished 14:27:33 2022-05-24

In [779]:
```
analysis_df = analysis_df[['base', 'lower value', 'mean', 'upper value']]

analysis_df['Min_ROI'] = analysis_df.apply(lambda x: x['lower value'] - x['base'], axis=1)
analysis_df['Min_ROI%'] = analysis_df.apply(lambda x: x['Min_ROI'] / x['base'], axis=1)
analysis_df['Forecast_ROI'] = analysis_df.apply(lambda x: x['mean'] - x['base'], axis=1)
analysis_df['Forecast_ROI%'] = analysis_df.apply(lambda x: x['mean'] / x['base'], axis=1)
analysis_df['Max_ROI'] = analysis_df.apply(lambda x: x['upper value'] - x['base'], axis=1)
analysis_df['Max_ROI%'] = analysis_df.apply(lambda x: x['upper value'] / x['base'], axis=1)
```
executed in 10ms, finished 14:27:33 2022-05-24

In [780]:
```
zip_98359_metrics = analysis_df
```
executed in 2ms, finished 14:27:33 2022-05-24

In [781]:
```
zip_98359_metrics
```
executed in 6ms, finished 14:27:33 2022-05-24

Out[781]:

	base	lower value	mean	upper value	Min_ROI	Min_ROI%	Forecast_ROI	Forecast_ROI%	Max_ROI	Max_ROI%
2018-12-31	444,600.00	445,224.01	459,933.74	474,643.47	624.01	0.00	15,333.74	1.03	30,043.47	1.07
2019-12-31	444,600.00	447,379.66	495,806.15	544,232.64	2,779.66	0.01	51,206.15	1.12	99,632.64	1.22
2020-12-31	444,600.00	443,098.27	539,090.73	635,083.20	-1,501.73	-0.00	94,490.73	1.21	190,483.20	1.43
2021-12-31	444,600.00	437,052.81	569,029.36	701,005.90	-7,547.19	-0.02	124,429.36	1.28	256,405.90	1.58

22 Final Analysis of Recommended Zip Codes

22.1 Avg Home DF

In [782]:
```
avg_home_df_metrics
```
executed in 5ms, finished 14:27:33 2022-05-24

Out[782]:

	base	lower home_value	mean	upper home_value	Min_ROI	Min_ROI%	Forecast_ROI	Forecast_ROI%	Max_ROI	Max_ROI%
2018-12-31	288,039.00	292,699.11	296,335.12	299,971.12	4,660.11	0.02	8,296.12	1.03	11,932.12	1.04
2019-12-31	288,039.00	297,332.59	315,184.63	333,036.66	9,293.59	0.03	27,145.63	1.09	44,997.66	1.16
2020-12-31	288,039.00	294,102.00	340,687.59	387,273.19	6,063.00	0.02	52,648.59	1.18	99,234.19	1.34
2021-12-31	288,039.00	287,258.32	359,239.52	431,220.72	-780.68	-0.00	71,200.52	1.25	143,181.72	1.50

22.2 37046:College Grove TN

In [783]:
```
1 zip_37046_metrics
```
executed in 7ms, finished 14:27:33 2022-05-24

Out[783]:

	base	lower value	mean	upper value	Min_ROI	Min_ROI%	Forecast_ROI	Forecast_ROI%	Max_ROI	Max_ROI%
2018-12-31	459,800.00	468,257.69	501,253.86	534,250.03	8,457.69	0.02	41,453.86	1.09	74,450.03	1.16
2019-12-31	459,800.00	468,612.24	571,721.86	674,831.47	8,812.24	0.02	111,921.86	1.24	215,031.47	1.47
2020-12-31	459,800.00	455,392.58	652,309.15	849,225.72	-4,407.42	-0.01	192,509.15	1.42	389,425.72	1.85
2021-12-31	459,800.00	434,655.38	698,855.46	963,055.53	-25,144.62	-0.05	239,055.46	1.52	503,255.53	2.09

22.3 02128: Boston MA

In [784]:
```
1 zip_02128_metrics
```
executed in 6ms, finished 14:27:33 2022-05-24

Out[784]:

	base	lower value	mean	upper value	Min_ROI	Min_ROI%	Forecast_ROI	Forecast_ROI%	Max_ROI	Max_ROI%
2018-12-31	489,900.00	498,942.37	512,514.32	526,086.27	9,042.37	0.02	22,614.32	1.05	36,186.27	1.07
2019-12-31	489,900.00	511,264.16	557,030.40	602,796.64	21,364.16	0.04	67,130.40	1.14	112,896.64	1.23
2020-12-31	489,900.00	522,757.43	608,620.65	694,483.87	32,857.43	0.07	118,720.65	1.24	204,583.87	1.42
2021-12-31	489,900.00	527,880.09	641,545.46	755,210.84	37,980.09	0.08	151,645.46	1.31	265,310.84	1.54

22.4 80204: Denver, CO

In [785]:
```
1 zip_80204_metrics
```
executed in 6ms, finished 14:27:33 2022-05-24

Out[785]:

	base	lower value	mean	upper value	Min_ROI	Min_ROI%	Forecast_ROI	Forecast_ROI%	Max_ROI	Max_ROI%
2018-12-31	371,600.00	380,320.70	387,502.07	394,683.43	8,720.70	0.02	15,902.07	1.04	23,083.43	1.06
2019-12-31	371,600.00	385,601.55	416,261.60	446,921.66	14,001.55	0.04	44,661.60	1.12	75,321.66	1.20
2020-12-31	371,600.00	382,699.74	443,296.34	503,892.94	11,099.74	0.03	71,696.34	1.19	132,292.94	1.36
2021-12-31	371,600.00	379,421.36	458,806.66	538,191.96	7,821.36	0.02	87,206.66	1.23	166,591.96	1.45

22.5 07087: Union City, NJ

In [786]:
```
1 zip_07087_metrics
```
executed in 6ms, finished 14:27:33 2022-05-24

Out[786]:

	base	lower value	mean	upper value	Min_ROI	Min_ROI%	Forecast_ROI	Forecast_ROI%	Max_ROI	Max_ROI%
2018-12-31	448,900.00	450,172.27	475,146.70	500,121.13	1,272.27	0.00	26,246.70	1.06	51,221.13	1.11
2019-12-31	448,900.00	477,799.93	553,457.43	629,114.93	28,899.93	0.06	104,557.43	1.23	180,214.93	1.40
2020-12-31	448,900.00	503,549.54	656,978.95	810,408.37	54,649.54	0.12	208,078.95	1.46	361,508.37	1.81
2021-12-31	448,900.00	515,409.16	731,392.17	947,375.18	66,509.16	0.15	282,492.17	1.63	498,475.18	2.11

22.6 98359: Olalla, WA

In [787]:
```
1 zip_98359_metrics
```

executed in 6ms, finished 14:27:33 2022-05-24

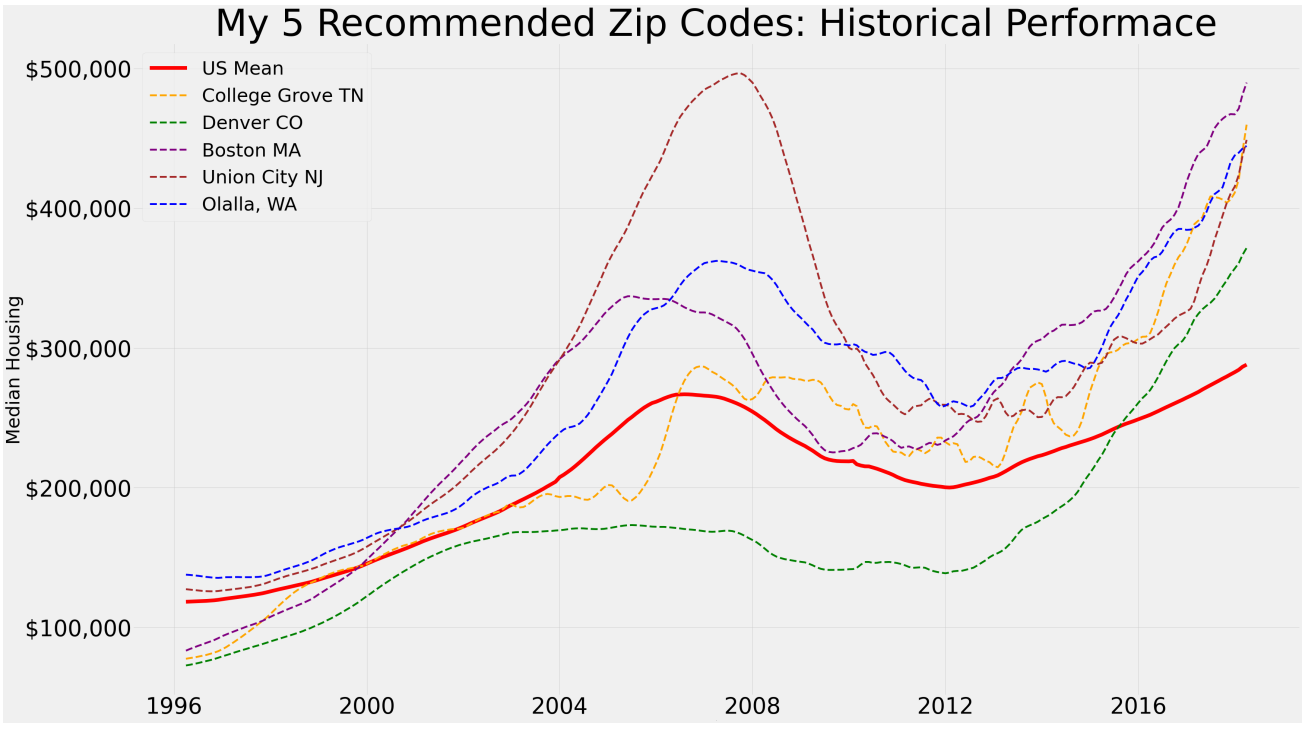
Out[787]:

	base	lower value	mean	upper value	Min_ROI	Min_ROI%	Forecast_ROI	Forecast_ROI%	Max_ROI	Max_ROI%
2018-12-31	444,600.00	445,224.01	459,933.74	474,643.47	624.01	0.00	15,333.74	1.03	30,043.47	1.07
2019-12-31	444,600.00	447,379.66	495,806.15	544,232.64	2,779.66	0.01	51,206.15	1.12	99,632.64	1.22
2020-12-31	444,600.00	443,098.27	539,090.73	635,083.20	-1,501.73	-0.00	94,490.73	1.21	190,483.20	1.43
2021-12-31	444,600.00	437,052.81	569,029.36	701,005.90	-7,547.19	-0.02	124,429.36	1.28	256,405.90	1.58

In [788]:

```
fig, ax = plt.subplots(figsize=(50, 30))

p = sns.lineplot(data=avg_home_df, x='time', y='home_value', color='red', label='US Mean',
                 linewidth = 10);
p = sns.lineplot(data=zip_37046_ts, x='time', y='value', color='orange', label='College Grove TN'
                 , linewidth = 5, ls='--');
p = sns.lineplot(data=zip_80204_ts, x='time', y='value', color='green', label='Denver CO',
                 linewidth = 5, ls='--');
p = sns.lineplot(data=zip_02128_ts, x='time', y='value', color='purple', label='Boston MA',
                linewidth = 5, ls='--');
p = sns.lineplot(data=zip_07087_ts, x='time', y='value', color='brown', label='Union City NJ',
                linewidth = 5, ls='--');
p = sns.lineplot(data=zip_98359_ts, x='time', y='value', color='blue', label='Olalla, WA',
                linewidth = 5, ls='--');


p.set_xlabel("Date", fontsize = 0)
p.set_ylabel("Median Housing", fontsize = 50)

#y1 = p.axvline('2008-01', color='red') #housing market crash begins
#y2 = p.axvline('2012-01', color='red') #housing market crash ends
#ax.fill_between(y1, y2)

plt.xticks(fontsize=60)
plt.yticks(fontsize=60)

fmt = '${x:,.0f}'
tick = mtick.StrMethodFormatter(fmt)
ax.yaxis.set_major_formatter(tick)

ax.legend(fontsize=50)
plt.savefig('images/top_zips_lineplot')

p.set_title("My 5 Recommended Zip Codes: Historical Performace", fontsize = 100)
plt.figsize=(50,25)

plt.show();
```

executed in 1.05s, finished 14:27:34 2022-05-24



22.7 Analysis:

- All 5 Zip Codes are have an upward trend that is much steeper than the national average.
- They are currently at the price point that the client wants to invest in.

- While the **best** time to invest would have been in 2012 at the bottom of the real estate market crash, **these zip codes are forecast to continue trending upward, and if they don't, will lose very little value, making them great low-risk, high-potential areas for FRC to invest in.**

22.8 Metrics DataFrame

- Creating Metrics DataFrame to analyze the 3 Year Forecast.
- I will slice out just Forecast Year 3 (12-01-2020) to compare as that is the length of time that the Client has requested.

In [789]:
```
df = zip_37046_metrics.reset_index()
df.iloc[:, 0] = df.iloc[:, 0].astype(str)
df.iloc[:, 0] = ['2018', '2019', 'College Grove, TN', '2021']
df = df.drop([0,1,3], axis=0)
college_grove = df
college_grove

```

executed in 8ms, finished 14:27:34 2022-05-24

Out[789]:

	index	base	lower value	mean	upper value	Min_ROI	Min_ROI%	Forecast_ROI	Forecast_ROI%	Max_ROI	Max_ROI%
2	College Grove, TN	459,800.00	455,392.58	652,309.15	849,225.72	-4,407.42	-0.01	192,509.15	1.42	389,425.72	1.85

In [790]:
```
boston = zip_02128_metrics.reset_index()
boston.iloc[:, 0] = boston.iloc[:, 0].astype(str)
boston.iloc[:, 0] = ['2018', '2019', 'Boston, MA', '2021']
boston = boston.drop([0,1,3], axis=0)
boston
```

executed in 8ms, finished 14:27:34 2022-05-24

Out[790]:

	index	base	lower value	mean	upper value	Min_ROI	Min_ROI%	Forecast_ROI	Forecast_ROI%	Max_ROI	Max_ROI%
2	Boston, MA	489,900.00	522,757.43	608,620.65	694,483.87	32,857.43	0.07	118,720.65	1.24	204,583.87	1.42

In [791]:
```
df = zip_80204_metrics.reset_index()
df.iloc[:, 0] = df.iloc[:, 0].astype(str)
df.iloc[:, 0] = ['2018', '2019', 'Denver, CO', '2021']
df = df.drop([0,1,3], axis=0)
denver = df
denver

```

executed in 8ms, finished 14:27:34 2022-05-24

Out[791]:

	index	base	lower value	mean	upper value	Min_ROI	Min_ROI%	Forecast_ROI	Forecast_ROI%	Max_ROI	Max_ROI%
2	Denver, CO	371,600.00	382,699.74	443,296.34	503,892.94	11,099.74	0.03	71,696.34	1.19	132,292.94	1.36

In [792]:
```
df = zip_07087_metrics.reset_index()
df.iloc[:, 0] = df.iloc[:, 0].astype(str)
df.iloc[:, 0] = ['2018', '2019', 'Union City, NJ', '2021']
df = df.drop([0,1,3], axis=0)
union_city = df
union_city
```

executed in 8ms, finished 14:27:34 2022-05-24

Out[792]:

	index	base	lower value	mean	upper value	Min_ROI	Min_ROI%	Forecast_ROI	Forecast_ROI%	Max_ROI	Max_ROI%
2	Union City, NJ	448,900.00	503,549.54	656,978.95	810,408.37	54,649.54	0.12	208,078.95	1.46	361,508.37	1.81

In [793]:
```
df = zip_98359_metrics.reset_index()
df.iloc[:, 0] = df.iloc[:, 0].astype(str)
df.iloc[:, 0] = ['2018', '2019', 'Olalla, WA', '2021']
df = df.drop([0,1,3], axis=0)
olalla = df
olalla
```

executed in 8ms, finished 14:27:34 2022-05-24

Out[793]:

	index	base	lower value	mean	upper value	Min_ROI	Min_ROI%	Forecast_ROI	Forecast_ROI%	Max_ROI	Max_ROI%
2	Olalla, WA	444,600.00	443,098.27	539,090.73	635,083.20	-1,501.73	-0.00	94,490.73	1.21	190,483.20	1.43

In [794]:
```
#creating forecast_df to compare all of the forecast metrics
forecast_df = college_grove.append(boston)
forecast_df = forecast_df.append(denver)
forecast_df = forecast_df.append(union_city)
forecast_df = forecast_df.append(olalla)

```

executed in 4ms, finished 14:27:34 2022-05-24

In [795]:
```
forecast_df = forecast_df.reset_index()
forecast_df.drop(columns=('level_0'), inplace=True)
forecast_df
```

executed in 7ms, finished 14:27:34 2022-05-24

Out[795]:

	index	base	lower value	mean	upper value	Min_ROI	Min_ROI%	Forecast_ROI	Forecast_ROI%	Max_ROI	Max_ROI%
0	College Grove, TN	459,800.00	455,392.58	652,309.15	849,225.72	-4,407.42	-0.01	192,509.15	1.42	389,425.72	1.85
1	Boston, MA	489,900.00	522,757.43	608,620.65	694,483.87	32,857.43	0.07	118,720.65	1.24	204,583.87	1.42
2	Denver, CO	371,600.00	382,699.74	443,296.34	503,892.94	11,099.74	0.03	71,696.34	1.19	132,292.94	1.36
3	Union City, NJ	448,900.00	503,549.54	656,978.95	810,408.37	54,649.54	0.12	208,078.95	1.46	361,508.37	1.81
4	Olalla, WA	444,600.00	443,098.27	539,090.73	635,083.20	-1,501.73	-0.00	94,490.73	1.21	190,483.20	1.43

In [796]:
```
forecast_roi = forecast_df.sort_values('Forecast_ROI', ascending=False)
forecast_roi
```

executed in 7ms, finished 14:27:34 2022-05-24

Out[796]:

	index	base	lower value	mean	upper value	Min_ROI	Min_ROI%	Forecast_ROI	Forecast_ROI%	Max_ROI	Max_ROI%
3	Union City, NJ	448,900.00	503,549.54	656,978.95	810,408.37	54,649.54	0.12	208,078.95	1.46	361,508.37	1.81
0	College Grove, TN	459,800.00	455,392.58	652,309.15	849,225.72	-4,407.42	-0.01	192,509.15	1.42	389,425.72	1.85
1	Boston, MA	489,900.00	522,757.43	608,620.65	694,483.87	32,857.43	0.07	118,720.65	1.24	204,583.87	1.42
4	Olalla, WA	444,600.00	443,098.27	539,090.73	635,083.20	-1,501.73	-0.00	94,490.73	1.21	190,483.20	1.43
2	Denver, CO	371,600.00	382,699.74	443,296.34	503,892.94	11,099.74	0.03	71,696.34	1.19	132,292.94	1.36

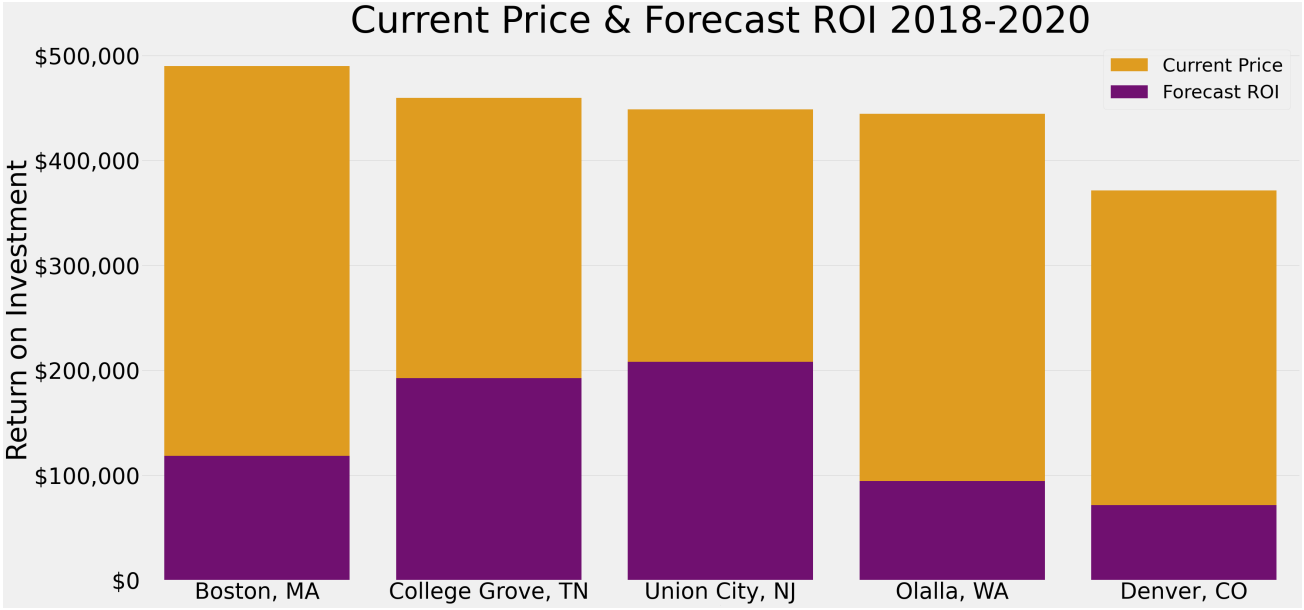
In [797]:
```
# sorted by base price for barplot visualization.
forecast_base = forecast_df.sort_values('base', ascending=False)
```

executed in 2ms, finished 14:27:34 2022-05-24

In [798]:

```
fig, ax = plt.subplots(figsize=(50, 25))

p = sns.barplot(data = forecast_base, x = "index", y = "base", color='orange', label='Current Price'
p = sns.barplot(data = forecast_base, x = "index", y = "Forecast_ROI", color='purple', label='Forecas


p.set_xlabel("City", fontsize = 0)
p.set_ylabel("Return on Investment", fontsize = 75)


plt.xticks(fontsize=60)
plt.yticks(fontsize=60)

fmt = '${x:,.0f}'
tick = mtick.StrMethodFormatter(fmt)
ax.yaxis.set_major_formatter(tick)

ax.legend(fontsize=50)
plt.savefig('images/top_zips_forecast_roi')

p.set_title("Current Price & Forecast ROI 2018-2020", fontsize = 100)
plt.figsize=(50,25)

plt.show();
```

executed in 533ms, finished 14:27:35 2022-05-24



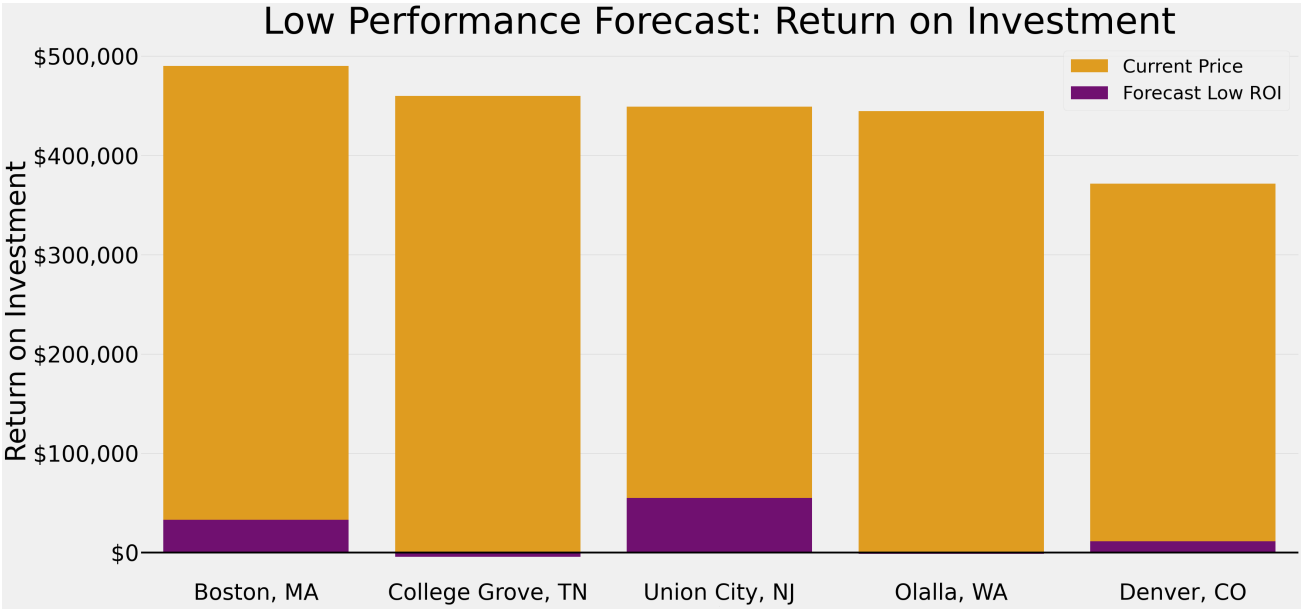
Analysis:

- The Mean Forecast ROI for all 5 Zip Codes is good, especially considering that this is just for a 3 year investment.
- These gains will of course be mulitplied per house invested in.

In [799]:

```
fig, ax = plt.subplots(figsize=(50, 25))

p = sns.barplot(data = forecast_base, x = "index", y = "base", color='orange', label='Current Price'
p = sns.barplot(data = forecast_base, x = "index", y = "Min_ROI", color="purple", label='Forecast Lo

p.set_xlabel("City", fontsize = 0)
p.set_ylabel("Return on Investment", fontsize = 75)

ax.axhline(0, ls='-', color='black', linewidth=(5))


plt.xticks(fontsize=60)
plt.yticks(fontsize=60)


fmt = '${x:,.0f}'
tick = mtick.StrMethodFormatter(fmt)
ax.yaxis.set_major_formatter(tick)

ax.legend(fontsize=50)

p.set_title("Low Performance Forecast: Return on Investment", fontsize = 100)
plt.figsize=(50,25)
plt.savefig('images/top_zips_low_roi')

plt.show();
```

executed in 566ms, finished 14:27:35 2022-05-24

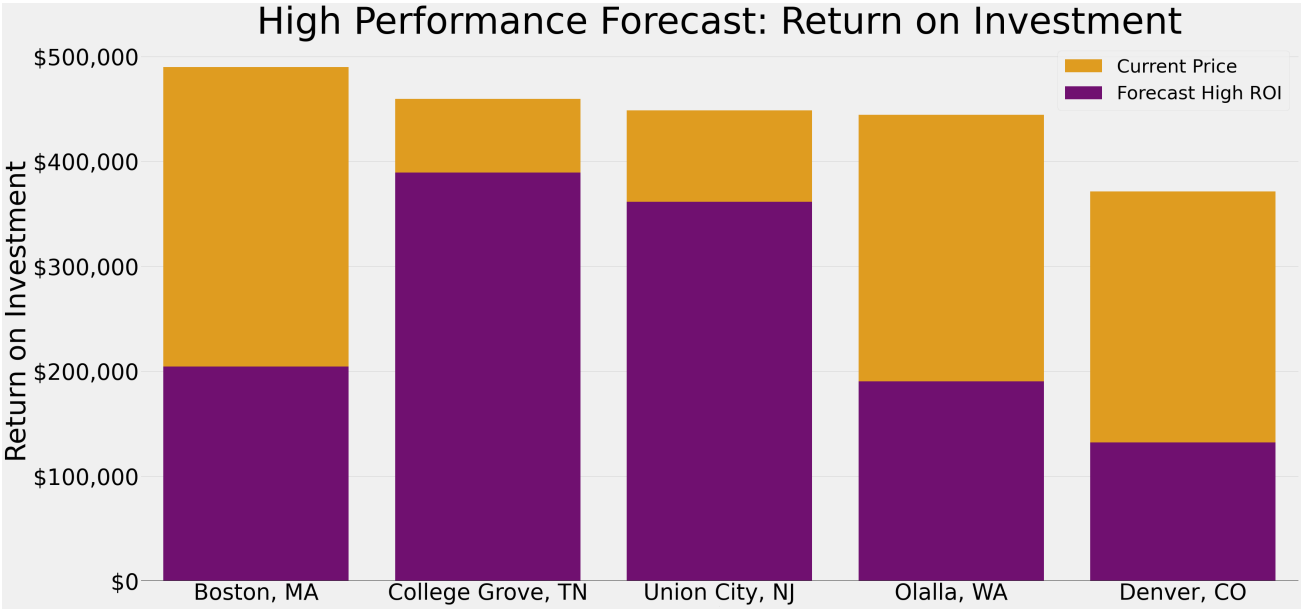


Analysis:

- The low forecast (low value confidence index) for 3 of the zip codes still returns a minimal ROI.
- The two which do lose money are very minimal, and barely show up on this barplot.
- **All 5 of these Zip Codes should be considered to be incredibly low risk.**

In [800]:
```
fig, ax = plt.subplots(figsize=(50, 25))

p = sns.barplot(data = forecast_base, x = "index", y = "base", color='orange', label='Current Price'
p = sns.barplot(data = forecast_base, x = "index", y = "Max_ROI", color="purple", label='Forecast Hi

p.set_xlabel("City", fontsize = 0)
p.set_ylabel("Return on Investment", fontsize = 75)

ax.axhline(0, ls='-', color='black', linewidth=(5))


plt.xticks(fontsize=60)
plt.yticks(fontsize=60)

fmt = '${x:,.0f}'
tick = mtick.StrMethodFormatter(fmt)
ax.yaxis.set_major_formatter(tick)

ax.legend(fontsize=50)

p.set_title("High Performance Forecast: Return on Investment", fontsize = 100)
plt.figsize=(50,25)
plt.savefig('images/top_zips_high_roi')

plt.show();
```

executed in 553ms, finished 14:27:36 2022-05-24



Analysis:

- The High Performance ROI forecast (upper value confidence interval), is the absolute best scenario that the model forecasts as a possibility. As such, these ROI should not be expected. However, this shows that there is significant ROI potential beyond the Mean Forecast.

23 Recommendations

23.1 Recommendation #1: Top 5 ZipCodes to Invest in:

- 1. 37046 College Grove, TN
- 2. 02128 Boston, MA
- 3. 80204 Denver, CO
- 4. 07087 Union City, NJ
- 5. 98359 Olalla, WA

These Zip Codes completely meet the criteria that I was given:

- **Cost of Entry:** Average Home Values around 500,000
  - (currently 370,000 - 489,000)
- **Geographic Diversity:** Each is in a different State.
- **High ROI:** Each is forecasted to have a high return on investment over the next 3 years.

-- **Mean Forecast Values for Forecast Year 3:** 1.2% ROI - 1.6% ROI.

-- **Mean Forecast ROI Amounts for Forecast Year 3:** 87,000 - 282,000.

- **Low Risk:** Each has been forecast with a low risk of losing money over the next 3 years.

-- **Maximum Forecast Loss at Forecast Year 3 :** -5% ROI

- **Timing:** All forecasts were for 3 years.

23.2 Recommendation #2: ZipCodes for Shorter investment

ZipCodes that showed a good return with low risk for 1 or 2 years.

- 1. 29403 Charleston SC
- 2. 98043 Mountlake Terrace WA

These Zip Codes still show a lot of potental, but need to be watched more closely. They all show great potentional for one or even two years of investment, but then the third year forecast becomes more uncertain.

I am in confident in recommending both of these Zip Codes during that shorter investment window, and then running models again as more data comes in.

23.3 Recommendation #3: Model more Zip Codes.

I was limited by time and resources available to me. **I recommend running this model on more zipcodes in more states, and at other price points to find more great investment opportunities.**

23.4 Recommendation #4: ACT NOW!

- I used the most recent data that I had access to, but these recommended zip codes are all rapidly increasing in value. **The longer you wait, the higher the cost of entry is going to be.**

24 Conclusion

- Analyzing the past performance over the last 3 years proved to be a good way to identify zip codes that have a high probability to continue performing well 2-3 years into the future.
- While not all Zip Codes that I identified and targetted ended up being recommended, **I only had to model 15 Zip Codes in order to get my 5 "best" Zip Codes to recommend.** It also led to me discovering 2-3 Zip Codes that nearly met all criteria and became a strong alternative recommendation.
- The approach taken throughout this entire process is replicatable and can and should be done often to monitor zip codes where FRC is focusing as well as identifying new markets to enter into or increase their presence in. **If FRC continues to do this, they will have a disctinct competetive advantage in the markets where they have a presence and will be able to increase the value of their investments for both the Company and it's shareholders.**