

Table of Contents

[1 Overview](#)

[2 Business Problem](#)

[3 Data Understanding](#)

[4 EDA](#)

[4.1 Number of Zip Codes per City in SA Metro](#)

[4.2 Reshape from Wide to Long Format](#)

[4.3 San Antonio vs US Median Home Prices](#)

[4.4 Average Housing Prices In San Antonio Metro Over Time](#)

[4.5 Zipcodes with Highest 5 Year ROI](#)

[4.6 Zip Codes with Highest 22 Year ROI](#)

[4.7 Seasonality EDA](#)

[4.8 Discussion on EDA](#)

[5 SARIMAX Modeling](#)

[5.1 Create Time Series for Each Zip Code in Our Top 10](#)

[5.2 Seasonal Decomposition](#)

[5.3 Stationarity Check](#)

[5.4 Autocorrelation and Partial Autocorrelation](#)

[5.5 Gridsearch for SARIMAX Model](#)

[5.6 Best Parameters SARIMAX Model](#)

[5.6.1 Arena District](#)

[5.6.2 Dignowity Hill](#)

[5.6.3 Gardendale / Brady Gardens](#)

[5.6.4 Kingsbury](#)

[5.6.5 Medina](#)

[5.6.6 Denver Heights / Highland Park](#)

[5.6.7 Inner Westside](#)

[5.6.8 Fisher](#)

[5.6.9 Canyon Lake](#)

[5.6.10 Eastside](#)

[6 Interpret Results](#)

[6.1 Projections for 3 and 5 year ROI](#)

[6.2 Barplots of ROI Projections and Errors](#)

[6.3 Top 5 Zip Codes](#)

[6.4 Average Housing Prices of our Top 5 Zip Codes from 2000 - 2022](#)

[6.5 Map Visualization of Top 5 Zip Codes](#)

[7 Recommendations](#)

[7.1 Inner Westside](#)

[7.2 Denver Heights](#)

[7.3 Eastside](#)

[7.4 Arena District](#)

[7.5 Dignowity Hill](#)

[8 Future Work](#)

Time Series Analysis of the San Antonio Metro Area

Author: Melody Bass



1 Overview

San Antonio is the seventh-most populous city in the United States. According to Wikipedia, it was the fastest-growing city among those in the top ten, at roughly 2% growth per year. [1]

(https://en.wikipedia.org/wiki/San_Antonio) According to recent data from DCInno.com, San

Antonio, over the last 4 years, was among the nation's fastest-growing cities for college-educated millennials, and was at the top in Texas. [2]

(<https://www.forbes.com/sites/scottbeyer/2016/04/30/when-it-comes-to-millennial-appeal-sanantonio-is-creeping-on-austin/?sh=6d8c68727c57>).

San Antonio has been one of the hottest real estate markets in the country for many years. It is one of the best places to visit and has one of the fastest-growing economies in the country, driven by sectors such as military, tourism, health care, and insurance. The U.S. Armed Forces have numerous facilities in and around San Antonio; including Fort Sam Houston, Lackland Air Force Base, Randolph Air Force Base, Kelly Air Force Base, Camp Bullis, and Camp Stanley. [1]

(https://en.wikipedia.org/wiki/San_Antonio).

The aim of this project is to provide a detailed analysis of the real estate market in the San Antonio metropolitan area using time series modeling and price forecasting. Utilizing median monthly home sales prices contained in this [dataset \(./data/zillow_2022.csv\)](#) prepared by Zillow and my clients

specific needs, I will recommend the top 5 San Antonio metro zip codes to invest in which provide the highest return on investment (ROI) and with the lowest associated risk.

▼ 2 Business Problem

EXPI Investments, a new real estate investment firm has hired me to identify the top 5 neighborhoods in the SA metro area that would give them the highest return on investment upon selling in 3 - 5 years.

They want to start with short term investments in one of the most lucrative real estate markets in the country. The company is most interested in safe investments to ensure they will have cash flow to reinvest when the time comes.

▼ 3 Data Understanding

The dataset used for this analysis contains median monthly housing sales prices for 265 zip codes over the period of January 2000 through June 2022 as reported by Zillow and can be found [here](#) ([./data/zillow_2022.csv](#)). This is the most up to date information that is available on [Zillow](#) (<https://www.zillow.com/research/data/>).

Each row represents a unique zip code. Each record contains location info and median housing sales prices for each month.

There are 27280 rows and 277 variables:

- *RegionID* - Unique index
- *RegionName* - Unique Zip Code
- *City* - City in which the zip code is located
- *State* - State in which the zip code is located
- *Metro* - Metropolitan Area in which the zip code is located
- *CountyName* - County in which the zip code is located
- *SizeRank* - Numerical rank of size of zip code, ranked 1 through 34430
- *2000-01-31* through *2022-06-30* - median monthly housing sales prices from January 2000 through June 2022, that is 269 data points of monthly data for each zip code

Our business case is based solely in the San Antonio metro area, so I will filter the dataset down to only include the zip codes contained in the SA metro area.

```
In [1]: ▾ 1 # Basics
2 import pandas as pd
3 import numpy as np
4 import itertools
5
6 # Visualization
7 import matplotlib.pyplot as plt
8 import plotly.express as px
9 import seaborn as sns
10 import matplotlib.patches as mpatches
11 from matplotlib.pylab import rcParams
12 import time
13
14 # Modeling
15 import statsmodels.api as sm
16 from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
17 from statsmodels.tsa.statespace.sarimax import SARIMAX
18 from sklearn.metrics import mean_squared_error
19 from statsmodels.tsa.seasonal import seasonal_decompose
20 from statsmodels.tsa.arima.model import ARIMA
21 from statsmodels.tsa.stattools import acf, pacf, adfuller
22 from sklearn.linear_model import LassoLarsCV
23
24 # Warnings
25 import warnings
26 from statsmodels.tools.sm_exceptions import ConvergenceWarning
27 warnings.simplefilter('ignore', ConvergenceWarning)
28 warnings.filterwarnings('ignore')
```

```
In [2]: ▾ 1 # load in zillow home price data as df
  2 df = pd.read_csv('data/zillow_2022.csv')
  3 df = df.drop(columns = ['RegionType', 'StateName'], axis=1)
  4 df.head()
```

Out[2]:

	RegionID	SizeRank	RegionName	State	City	Metro	CountyName	2000-01-31	2000-02-29
0	61639	0	10025	NY	New York	New York-Newark-Jersey City	New York County	NaN	NaN
1	84654	1	60657	IL	Chicago	Chicago-Naperville-Elgin	Cook County	378936.0	380056.0
2	61637	2	10023	NY	New York	New York-Newark-Jersey City	New York County	NaN	NaN
3	91982	3	77494	TX	Katy	Houston-The Woodlands-Sugar Land	Harris County	225068.0	225385.0
4	84616	4	60614	IL	Chicago	Chicago-Naperville-Elgin	Cook County	497125.0	498166.0

5 rows × 277 columns

```
In [3]: ▾ 1 # pd.set_option('display.max_rows', 1000)
  2 df['Metro'].value_counts()
```

```
Out[3]: New York-Newark-Jersey City      894
        Chicago-Naperville-Elgin       377
        Los Angeles-Long Beach-Anaheim    357
        Philadelphia-Camden-Wilmington    339
        Washington-Arlington-Alexandria    309
                                         ...
Hereford                               1
Connersville                            1
Los Alamos                               1
Vernon                                    1
Ketchikan                                1
Name: Metro, Length: 857, dtype: int64
```

```
In [4]: 1 #Dataframe containing only zips from San Antonio metro
2 sa_df = df[df['Metro']=='San Antonio-New Braunfels']
3 sa_df.head()
```

Out[4]:

	RegionID	SizeRank	RegionName	State	City	Metro	CountyName	2000-01-31	2000-02-28
21	92271	23	78130	TX	New Braunfels	San Antonio-New Braunfels	Comal County	NaN	NaN
44	92341	47	78245	TX	San Antonio	San Antonio-New Braunfels	Bexar County	116692.0	116770.0
150	92336	153	78240	TX	San Antonio	San Antonio-New Braunfels	Bexar County	114140.0	114360.0
378	92345	382	78249	TX	San Antonio	San Antonio-New Braunfels	Bexar County	130419.0	130549.0
386	92350	390	78254	TX	San Antonio	San Antonio-New Braunfels	Bexar County	140116.0	140318.0

5 rows × 277 columns

```
In [5]: 1 sa_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 105 entries, 21 to 26415
Columns: 277 entries, RegionID to 2022-06-30
dtypes: float64(270), int64(3), object(4)
memory usage: 228.0+ KB
```

In [6]: 1 sa_df.describe()

Out[6]:

	RegionID	SizeRank	RegionName	2000-01-31	2000-02-29	2000-03-31	
count	105.000000	105.000000	105.000000	69.000000	69.000000	69.000000	
mean	92308.304762	8461.390476	78193.685714	130346.579710	130606.971014	130841.608696	131100.000000
std	93.859564	7199.818934	156.834143	68273.344025	68513.528089	68645.817295	68770.000000
min	92192.000000	23.000000	78002.000000	52991.000000	53036.000000	53262.000000	53488.000000
25%	92255.000000	2025.000000	78108.000000	85073.000000	85130.000000	85117.000000	85117.000000
50%	92307.000000	7824.000000	78211.000000	106068.000000	105880.000000	106357.000000	106357.000000
75%	92336.000000	12186.000000	78240.000000	160421.000000	160555.000000	160815.000000	160815.000000
max	92727.000000	29964.000000	78886.000000	427022.000000	429011.000000	430687.000000	430687.000000

8 rows × 273 columns

In [7]: 1 sa_df.isna().sum()

Out[7]:

RegionID	0
SizeRank	0
RegionName	0
State	0
City	0
..	
2022-02-28	0
2022-03-31	0
2022-04-30	0
2022-05-31	0
2022-06-30	0

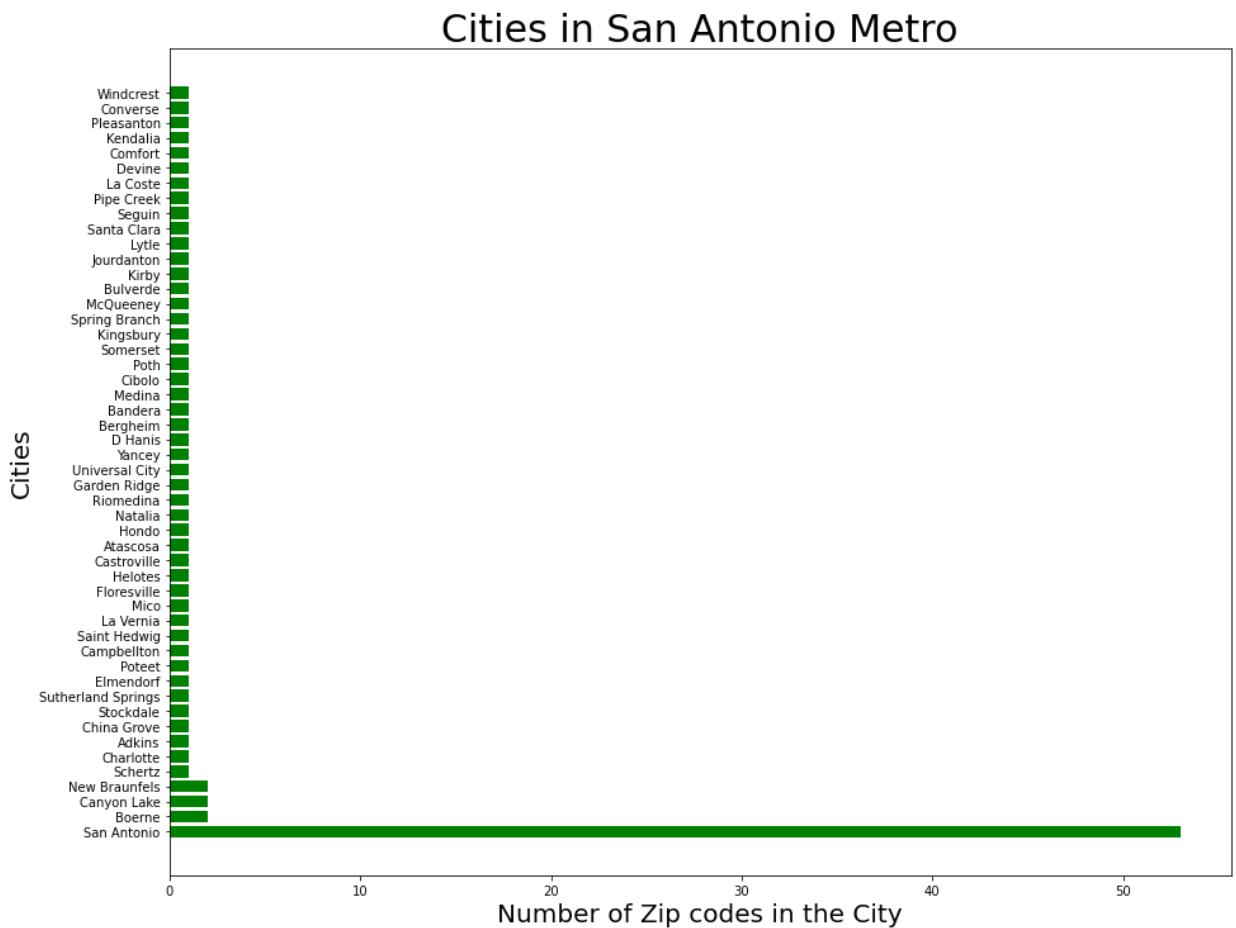
Length: 277, dtype: int64

Our filtered dataset contains a wide range of median home values, from a starting range of 53k to 427k for 105 zip codes in the SA metro area with several null values. The null values are mostly contained in the early years of the data, but due to the massive growth of the area, there is data in the later years of the dataset.

▼ 4 EDA

▼ 4.1 Number of Zip Codes per City in SA Metro

```
In [8]: 1 #Visualizing the value counts of zipcodes in each city in the San Antonio Metro area
2
3 fig, ax = plt.subplots(figsize=(15,12))
4
5 y = [a for a in sa_df['City'].value_counts()]
6
7 x = [a for a in sa_df['City'].value_counts().keys()]
8
9 ax.barh(x,y,color='green')
10 ax.set_title('Cities in San Antonio Metro',fontsize=30)
11 ax.set_ylabel('Cities',fontsize=20)
12 ax.set_xlabel('Number of Zip codes in the City',fontsize=20)
13 fig.savefig('./images/zip_per_city.jpeg');
```



There are several small cities in the SA metro area that only have 1 zip code, while San Antonio itself contains 55 zip codes.

4.2 Reshape from Wide to Long Format

The data needs to be melted from wide to long format in order to proceed with visualization and modeling. Let's go ahead and do that now!

```
In [9]: ▾ 1 def melt_data(df):
2     """
3         Takes the zillow_data dataset in wide form or a subset of the zil-
4         Returns a long-form datetime dataframe
5         with the datetime column names as the index and the values as the
6
7         If more than one row is passes in the wide-form dataset, the value
8         will be the mean of the values from the datetime columns in all o-
9     """
10
11     melted = pd.melt(df, id_vars=['RegionName', 'RegionID', 'SizeRank']
12     melted['time'] = pd.to_datetime(melted['time'], format = '%Y-%m')
13     melted = melted.dropna(subset=['value'])
14     return melted.groupby('time').aggregate({'value':'mean'})
```

```
In [10]: ▾ 1 # Get average of San Antonio metro zipcodes
2 satx_df_melted = melt_data(sa_df)
3 satx_df_melted.tail()
```

Out[10]:

	value
time	
2022-02-28	326126.866667
2022-03-31	331758.419048
2022-04-30	337743.838095
2022-05-31	343256.361905
2022-06-30	347860.333333

```
In [11]: ▾ 1 # Get average of all US zipcodes
2 df_melted = melt_data(df)
3 df_melted.head()
```

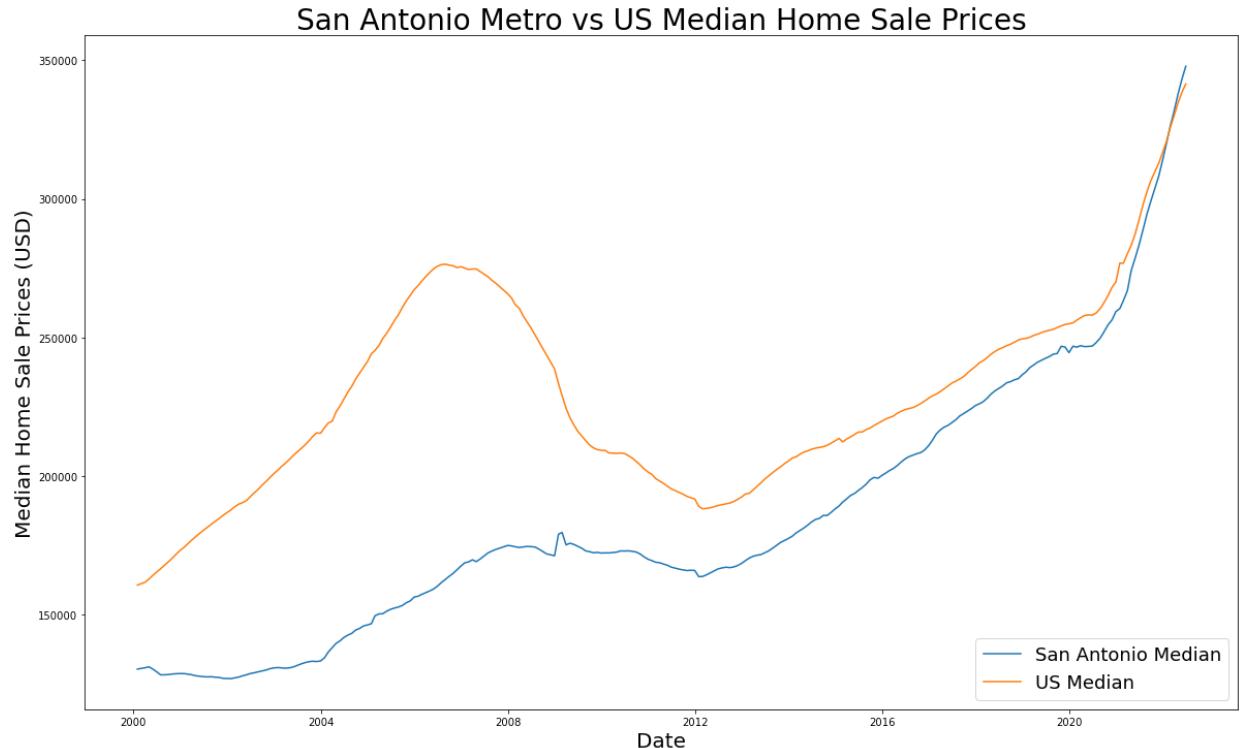
Out[11]:

	value
time	
2000-01-31	160670.656547
2000-02-29	161164.847325
2000-03-31	161690.402760
2000-04-30	162842.504295
2000-05-31	164204.550578

▼ 4.3 San Antonio vs US Median Home Prices

In order to compare how San Antonio home prices stack up the the US Median home price, I created a time series for the average median prices of the San Antonio area and a time series for the average median price of the US. Plotted together, we can see the differences.

```
In [12]: # Plot home value time series for all zip codes averaged
# Define figure
ax = plt.figure(figsize=(20,12))
# Plot
plt.plot(satx_df_melted, label = 'San Antonio Median')
plt.plot(df_melted, label = 'US Median')
# Title, x & y labels
plt.title('San Antonio Metro vs US Median Home Sale Prices', fontsize=24)
plt.ylabel('Median Home Sale Prices (USD)', fontsize=20)
plt.xlabel('Date', fontsize=20)
plt.legend(loc='lower right', fontsize=18)
plt.savefig('./images/sa_vs_us.jpeg')
plt.show()
```



The median home value in San Antonio metro used to be much lower than the national average, but has actually surpassed it this year! This is due to the massive growth in the area and the low inventory of homes that are available for sale. We can also see that the housing market in San Antonio was not as affected by the crash in 2008. I will leave all of the data in the model to provide my client with a more conservative model to forecast housing prices.

Next, I will create another time series that will contain all the location info in long format.

```
In [13]: 1 #Create dataframe for new melted data
          2 def melt_df(df):
          3     merged = []
          4     for zipcode in df.RegionName:
          5         melted = melt_data(df.loc[df['RegionName'] == zipcode])
          6         row = df.loc[df['RegionName'] == zipcode].iloc[:, :6]
          7         rows = pd.concat([row]*len(melted), ignore_index=True)
          8         merge = pd.concat([rows, melted.reset_index()], axis=1)
          9         merged.append(merge)
         10    melted_df = pd.concat(merged)
         11    return melted_df
```

```
In [14]: 1 satx_df = melt_df(sa_df)
          2 satx_df.head()
```

Out[14]:

	RegionID	SizeRank	RegionName	State	City	Metro	time	value
0	92271	23	78130	TX	New Braunfels	San Antonio-New Braunfels	2007-01-31	167985.0
1	92271	23	78130	TX	New Braunfels	San Antonio-New Braunfels	2007-02-28	168585.0
2	92271	23	78130	TX	New Braunfels	San Antonio-New Braunfels	2007-03-31	169390.0
3	92271	23	78130	TX	New Braunfels	San Antonio-New Braunfels	2007-04-30	170118.0
4	92271	23	78130	TX	New Braunfels	San Antonio-New Braunfels	2007-05-31	170191.0

```
In [15]: 1 # Drop columns no longer needed
          2 model_df = satx_df.drop(['RegionID', 'City', 'State', 'Metro', 'SizeRank']
          3 model_df.head()
```

Out[15]:

	RegionName	value
time		
2007-01-31	78130	167985.0
2007-02-28	78130	168585.0
2007-03-31	78130	169390.0
2007-04-30	78130	170118.0
2007-05-31	78130	170191.0

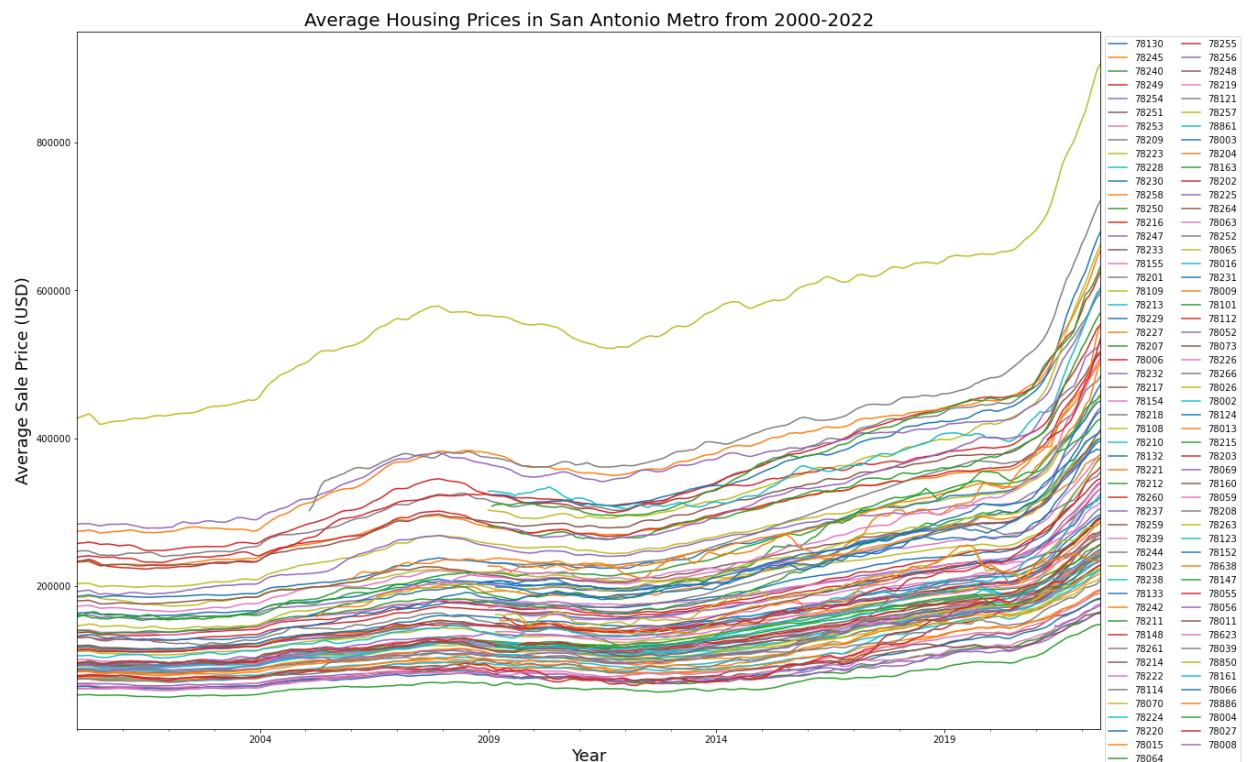
In [16]: 1 model_df.info()

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 24245 entries, 2007-01-31 to 2022-06-30
Data columns (total 2 columns):
 #   Column      Non-Null Count  Dtype  
---  --          -----          Dtype  
 0   RegionName  24245 non-null   int64  
 1   value        24245 non-null   float64 
dtypes: float64(1), int64(1)
memory usage: 568.2 KB
```

4.4 Average Housing Prices In San Antonio Metro Over Time

Let's take a look at the time series plotted for all 105 zip codes. We will be able to see the overall trend, but will be very dense on the plot.

In [17]: 1 *#plot the values grouped by zipcode*
2 plt.figure(figsize=(20,14))
3 for zipcode in model_df.RegionName.unique():
4 model_df.loc[model_df['RegionName'] == zipcode].value.plot(label=zipcode)
5 plt.legend(bbox_to_anchor=(1,1), loc='upper left', ncol=2)
6
7 plt.xlabel("Year", fontsize = 18)
8 plt.ylabel("Average Sale Price (USD)", fontsize=18)
9 plt.title('Average Housing Prices in San Antonio Metro from 2000-2022')
10 plt.savefig('./images/all_sa_prices.jpeg')
11 plt.show();
12



```
In [18]: ▾ 1 # Which zipcode is the one with the highest median value?  
2 model_df.loc[model_df['value'] >= 800000]
```

Out[18]:

	RegionName	value
time		
2021-11-30	78257	802164.0
2021-12-31	78257	815748.0
2022-01-31	78257	829571.0
2022-02-28	78257	842365.0
2022-03-31	78257	860608.0
2022-04-30	78257	878970.0
2022-05-31	78257	897059.0
2022-06-30	78257	907023.0

Much to my surprise, 78257 is outside of the main city loop. These must be a development of custom luxury homes in Leon Springs.

▼ 4.5 Zipcodes with Highest 5 Year ROI

In order to narrow down our focus before modeling, we will only keep 10 zip codes with highest ROI based on our last 5 years of historical data.

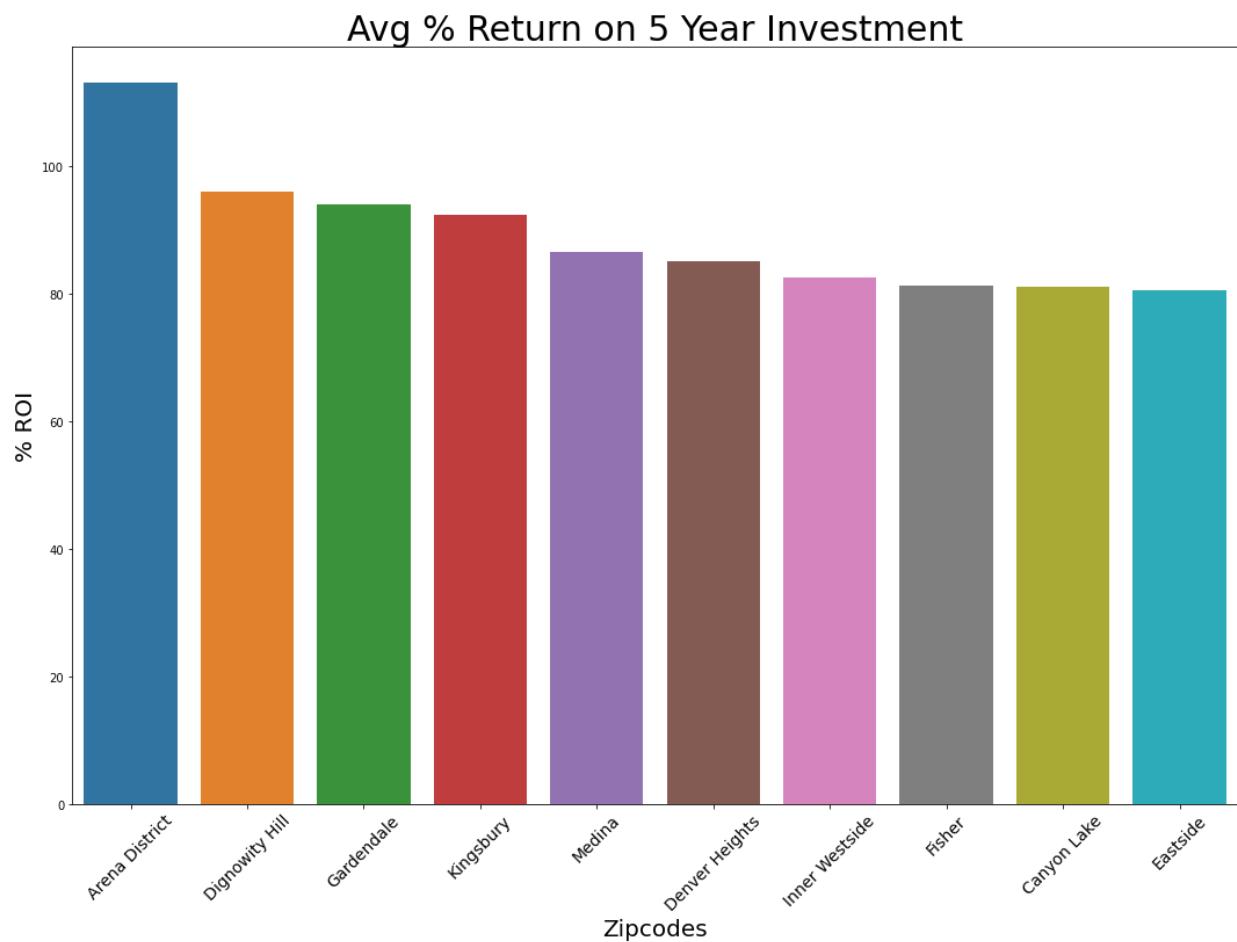
```
In [19]: ▾ 1 # Create copy of df to calculate ROIs
  2 roi_df = sa_df.copy()
  3 roi_df.head()
```

Out[19]:

	RegionID	SizeRank	RegionName	State	City	Metro	CountyName	2000-01-31	2000-02-28
21	92271	23	78130	TX	New Braunfels	San Antonio-New Braunfels	Comal County	NaN	NaN
44	92341	47	78245	TX	San Antonio	San Antonio-New Braunfels	Bexar County	116692.0	116770.0
150	92336	153	78240	TX	San Antonio	San Antonio-New Braunfels	Bexar County	114140.0	114360.0
378	92345	382	78249	TX	San Antonio	San Antonio-New Braunfels	Bexar County	130419.0	130549.0
386	92350	390	78254	TX	San Antonio	San Antonio-New Braunfels	Bexar County	140116.0	140318.0

5 rows × 277 columns

```
In [20]: 1 # Calculate 5 year ROI and keep top 10
2 roi_df['roi_5_year'] = ((roi_df['2022-06-30'] - roi_df['2017-06-30']).sum()
3 sa_5_year_roi = roi_df.sort_values('roi_5_year', ascending=False).head(10)
4 sa_5_year_roi['RegionName'] = sa_5_year_roi['RegionName'].astype(str)
5
6 #Plot
7 fig,ax=plt.subplots(figsize=(18,12))
8 sns.barplot('RegionName', 'roi_5_year', data = sa_5_year_roi)
9
10 zip_name_list = ['Arena District', 'Dignowity Hill', 'Gardendale', 'Kingsbury',
11                   'Inner Westside', 'Fisher', 'Canyon Lake', 'Eastside']
12
13 #rotate the x-axis text and change to month name
14 ax.set_xticklabels(zip_name_list, rotation = 45, size = 14)
15
16 # Axis labels and title
17 ax.set_ylabel('% ROI', fontsize='20')
18 ax.set_xlabel('Zipcodes', fontsize='20')
19 ax.set_title('Avg % Return on 5 Year Investment', fontsize='30')
20
21 #Save fig
22 plt.savefig('./images/roi_5_year.jpeg');
```

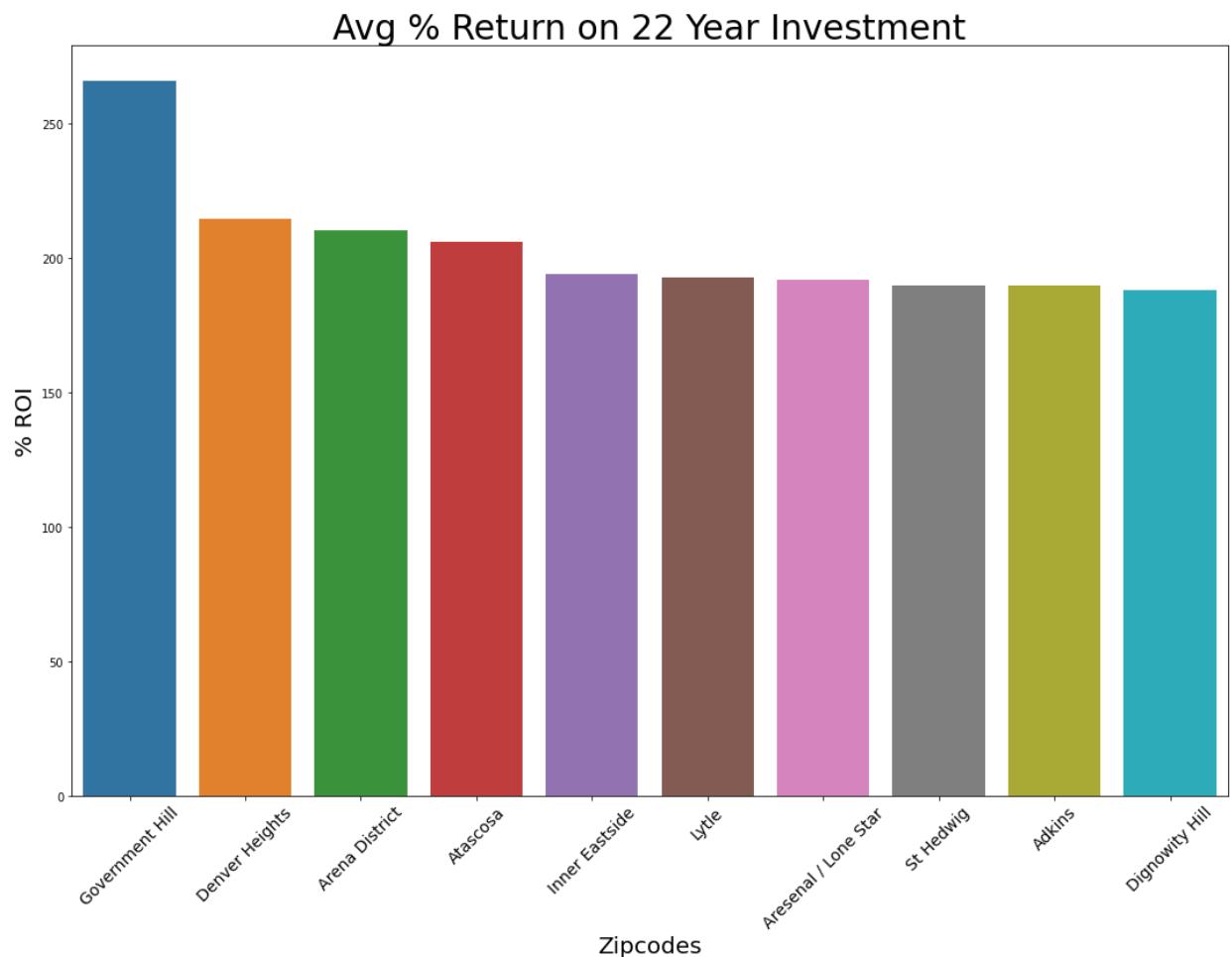


78203 contains the Arena District, which is the area surrounding the AT&T Center. The other zip codes with highest ROI are spread all around the city in every direction. While most are around the downtown area, there are several suburbs like Canyon Lake and Medina that have taken the top ROI spots!

▼ **4.6 Zip Codes with Highest 22 Year ROI**

Out of curiosity, are the zip codes with the highest ROI over 22 years the same as the top 10 zip codes over the last 5 years?

```
In [21]: # Calculate 22 year ROI and keep top 10
1 roi_df['roi_22year'] = ((roi_df['2022-06-30'] - roi_df['2000-01-31']).sum(1))
2 sa_22_year_roi = roi_df.sort_values('roi_22year', ascending=False).head(10)
3 sa_22_year_roi['RegionName'] = sa_22_year_roi['RegionName'].astype(str)
4
5
6 #Plot
7 fig,ax=plt.subplots(figsize=(18,12))
8 sns.barplot('RegionName', 'roi_22year', data = sa_22_year_roi)
9
10 zip_name_list_22 = ['Government Hill', 'Denver Heights', 'Arena District', 'Lytle', 'Aresenal / Lone Star', 'St Hedwig', 'Adkins', 'Dignowity Hill', 'Inner Eastside', 'Atascosa']
11
12 #rotate the x-axis text and change to month name
13 ax.set_xticklabels(zip_name_list_22, rotation = 45, size = 14)
14
15 # Title and Axis Labels
16 ax.set_ylabel('% ROI', fontsize='20')
17 ax.set_xlabel('Zipcodes', fontsize='20')
18 ax.set_title('Avg % Return on 22 Year Investment', fontsize='30');
```



Only 3 zip codes that have the highest ROI on a 5 year investment are in the highest ROI over a 22 year investment. This is once again due to new developments in the suburban areas, as well as gentrification of areas that were once undesirable.

▼ 4.7 Seasonality EDA

According to the National Association of Realtors, the number of home sales increases significantly in the spring, with home sales increasing by 34% in February and March. Sales continue upward with the busiest home selling months being May through August, accounting for 40% of United States annual home sales volume. The slowest months are November through February. Prices of homes slightly increase during surge months when the demand in the market is higher. Therefore, selling homes during these peak times could prove to be advantageous for maximizing profits.^[4] (<https://www.nar.realtor/blogs/economists-outlook/seasonality-in-the-housing-market>)

In order to study the seasonality in our data further and determine the best time of year to buy and sell houses in our top zip codes for maximizing profits, I am going to calculate the average 6 month ROI for winter and summer sales (i.e. a winter sale is a house that was purchased the previous summer and sold in the winter, while a summer sale is a house that was purchased during the winter and sold the next summer). I am going to calculate this over the last 5 years and take the average ROI gains for each time period.

In [22]:

```

1 # Calculate summer sale ROI
2 sa_5_year_roi['roi_summer_sale_2018'] = ((sa_5_year_roi['2018-07-31']
3                                         (sa_5_year_roi['2018-01-31'])
4 sa_5_year_roi['roi_summer_sale_2019'] = ((sa_5_year_roi['2019-07-31']
5                                         (sa_5_year_roi['2019-01-31'])
6 sa_5_year_roi['roi_summer_sale_2020'] = ((sa_5_year_roi['2020-07-31']
7                                         (sa_5_year_roi['2020-01-31'])
8 sa_5_year_roi['roi_summer_sale_2021'] = ((sa_5_year_roi['2021-07-31']
9                                         (sa_5_year_roi['2021-01-31'])
10 sa_5_year_roi['roi_summer_sale_2022'] = ((sa_5_year_roi['2022-06-30']
11                                         (sa_5_year_roi['2022-01-31'])
12 sa_5_year_roi['average_summer_sale_roi'] = sa_5_year_roi[['roi_summer'
13                                         'roi_summer'
14                                         'roi_summer'
15 #Calculate winter sale ROI
16 sa_5_year_roi['roi_winter_sale_2018'] = ((sa_5_year_roi['2018-01-31']
17                                         (sa_5_year_roi['2017-07-31'])
18 sa_5_year_roi['roi_winter_sale_2019'] = ((sa_5_year_roi['2019-01-31']
19                                         (sa_5_year_roi['2018-07-31'])
20 sa_5_year_roi['roi_winter_sale_2020'] = ((sa_5_year_roi['2020-01-31']
21                                         (sa_5_year_roi['2019-07-31'])
22 sa_5_year_roi['roi_winter_sale_2021'] = ((sa_5_year_roi['2021-01-31']
23                                         (sa_5_year_roi['2020-07-31'])
24 sa_5_year_roi['roi_winter_sale_2022'] = ((sa_5_year_roi['2022-01-31']
25                                         (sa_5_year_roi['2021-07-31'])
26 sa_5_year_roi['average_winter_sale_roi'] = sa_5_year_roi[['roi_winter'
27                                         'roi_winter'
28                                         'roi_winter'
29
30 # Sort rows by highest summer sale ROI
31 sa_summer_roi = sa_5_year_roi.sort_values('average_summer_sale_roi', as
32
33 # Make column for neighborhood name
34 # Top 10 zip code names
35 top_zip_names = ['Gardendale', 'Kingsbury', 'Dignowity Hill', 'Inner
36                                         'Arena District', 'Canyon Lake', 'Medina', 'Denver He
37 sa_summer_roi['Neighborhood'] = top_zip_names
38
39 # Keep columns we want
40 sa_seasonal_roi = sa_summer_roi.reindex(columns=['RegionName', 'Neigh
41                                         'average_winter_sale_'
42 sa_seasonal_roi

```

Out[22]:

	RegionName	Neighborhood	average_summer_sale_roi	average_winter_sale_roi
1643	78207	Gardendale	7.61	6.16
17365	78638	Kingsbury	7.28	6.23
9383	78202	Dignowity Hill	7.14	6.62
3806	78237	Inner Westside	7.13	5.29
18678	78623	Fisher	6.65	5.80
13033	78203	Arena District	6.34	8.64

	RegionName	Neighborhood	average_summer_sale	roi	average_winter_sale	roi
4637	78133	Canyon Lake	6.27	6.21		
17996	78055	Medina	5.95		6.89	
2710	78210	Denver Heights	5.88		6.48	
6754	78220	Eastside	5.70		6.18	

For most of our top zip codes, we can see about a 1% average ROI gain if we were to buy in the winter and sell in the summer. Interestingly enough, 4 of our zip codes show higher ROI gains if we were to buy in the summer and sell in the winter. The Arena District has over 2% higher ROI for a winter sale as opposed to a summer sale. A winter sale in Medina will increase ROI by 1%, while Denver Heights and the Eastside are about 0.5% increase.

▼ 4.8 Discussion on EDA

There is a similar trend amongst all of our neighborhoods in the SA metro area. There is also a very wide range of median housing values, where the more affluent neighborhoods have a much higher value than others.

We can also clearly see the housing bubble and 2008 crash in the data. Values increased rapidly starting in 2004 and took a dip through 2009, plateaued, and begin rising again in 2012. All zip codes in the San Antonio metro were mildly affected by this crash in comparison to the rest of the US. A number of reasons contributed to this crash outside of the typical housing market highs and lows, and while they are rare, they do occur and are unpredictable. I am going to leave this data in for our model to create a more conservative forecast for my client.

As discussed above, we know that there is going to be seasonality in our data and I will use SARIMAX for my time series model.

▼ 5 SARIMAX Modeling

▼ 5.1 Create Time Series for Each Zip Code in Our Top 10

```
In [23]: ▾ 1 # Define function to create individual time series for each zipcode
  ▾ 2 def zipcode_ts(df, zipcode):
    ...
  ▾ 4   Input:
    5     df : dataframe of Zillow housing data.
    6     zipcode : list of zip codes to iterate through.
    ...
  8     zipcode_df = df[df['RegionName'] == zipcode]
  9     zipcode_df = zipcode_df.drop(columns = 'RegionName', axis=1)
 10    return zipcode_df
```

```
In [24]: 1 # Empty list
          2 zip_df = []
          3
          4 #Zipcodes with highest 5 year ROI
          5 zip_list = [78203,78202,78207,78638,78055,78210,78237,78623,78133,782]
          6
          7 # Return list of time series for each zipcode in our zip_list
          8 for z in model_df['RegionName'].unique():
          9     for z in zip_list:
          10         training_data = zipcode_ts(model_df, z)
          11         zip_df.append(training_data)
          12     else:
          13         continue
```

```
In [25]: 1 # Sanity check
          2 zip_df[0]
```

Out[25]:

value

time

	time	value
2000-01-31		77799.0
2000-02-29		77923.0
2000-03-31		78250.0
2000-04-30		78620.0
2000-05-31		78765.0
...	...	
2022-02-28		227521.0
2022-03-31		231757.0
2022-04-30		235924.0
2022-05-31		239372.0
2022-06-30		241568.0

270 rows × 1 columns

Our zip_df now contains a list of 10 time series for each of our 10 zipcodes of interest.

5.2 Seasonal Decomposition

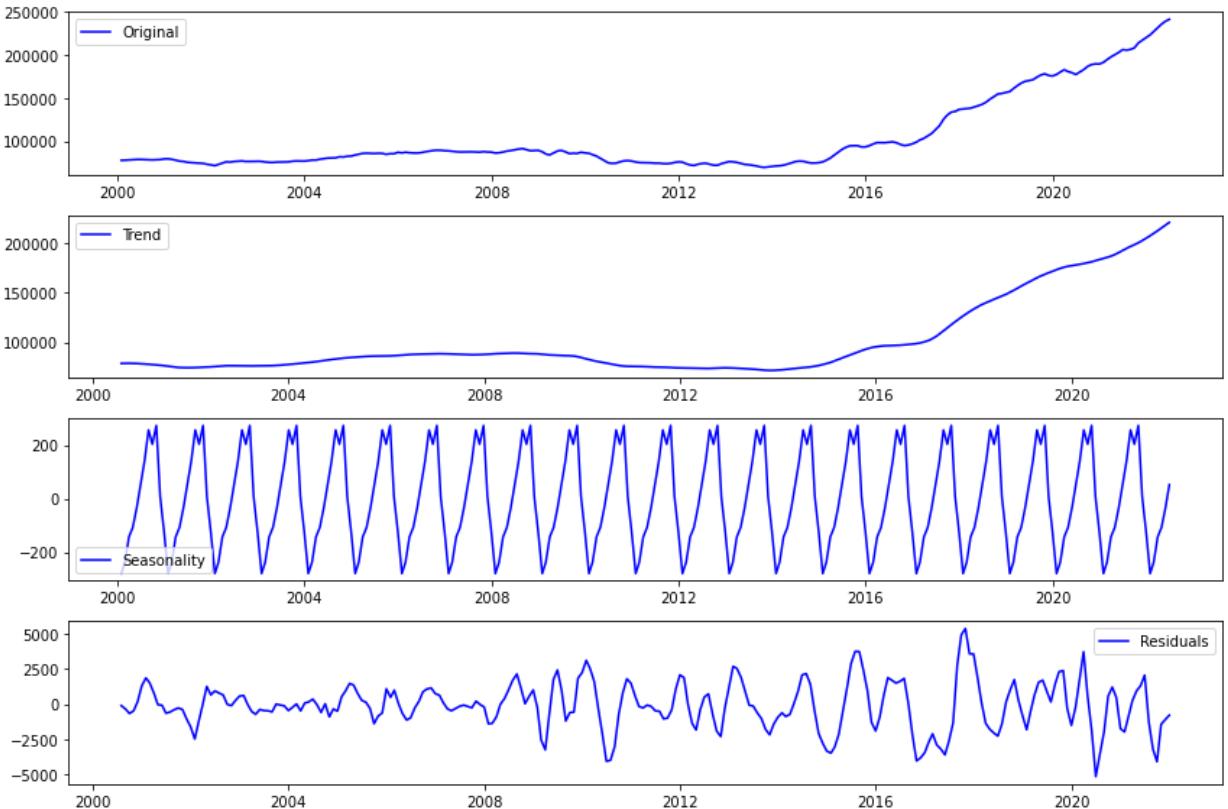
We can also decompose our time series to look for trend and seasonality.

Let's start by looking at our top ROI zip code - 78203 (Arena District), and 78133 (Canyon Lake), which has had huge ROI gains in the last 2 years compared to all data in the dataset.

```
In [26]: 1 def decomposition_plot(ts):
2     decomposition = sm.tsa.seasonal_decompose(ts, model='additive')
3
4     #Gather the trend, seasonality, and residuals
5     trend = decomposition.trend
6     seasonal = decomposition.seasonal
7     residual = decomposition.resid
8
9     # Plot gathered statistics
10    plt.figure(figsize=(12,8))
11    plt.subplot(411)
12    plt.plot(ts, label='Original', color='blue')
13    plt.legend(loc='best')
14    plt.subplot(412)
15    plt.plot(trend, label='Trend', color='blue')
16    plt.legend(loc='best')
17    plt.subplot(413)
18    plt.plot(seasonal,label='Seasonality', color='blue')
19    plt.legend(loc='best')
20    plt.subplot(414)
21    plt.plot(residual, label='Residuals', color='blue')
22    plt.legend(loc='best')
23    plt.tight_layout();
24    return decomposition_plot
```

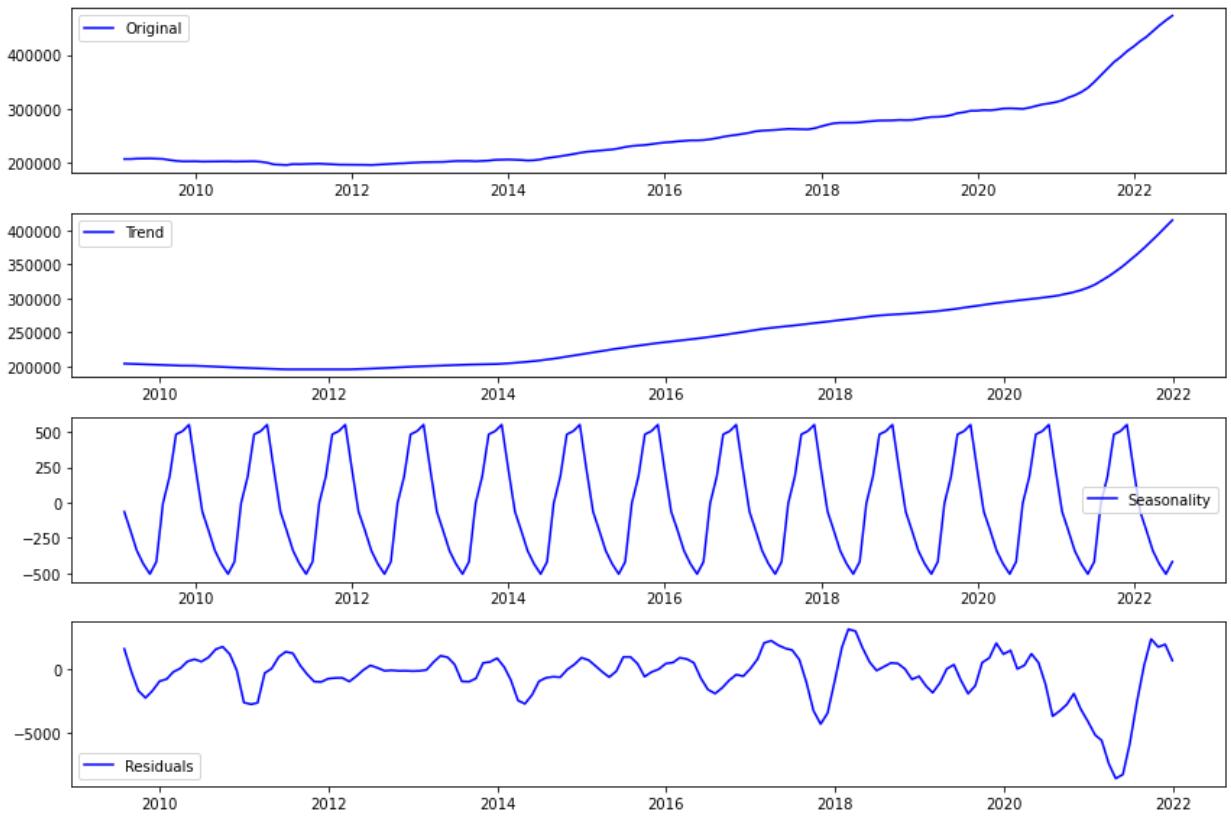
```
In [27]: 1 #Decomposition plot of the Arena District
2 decomposition_plot(zip_df[0])
```

Out[27]: <function __main__.decomposition_plot(ts)>



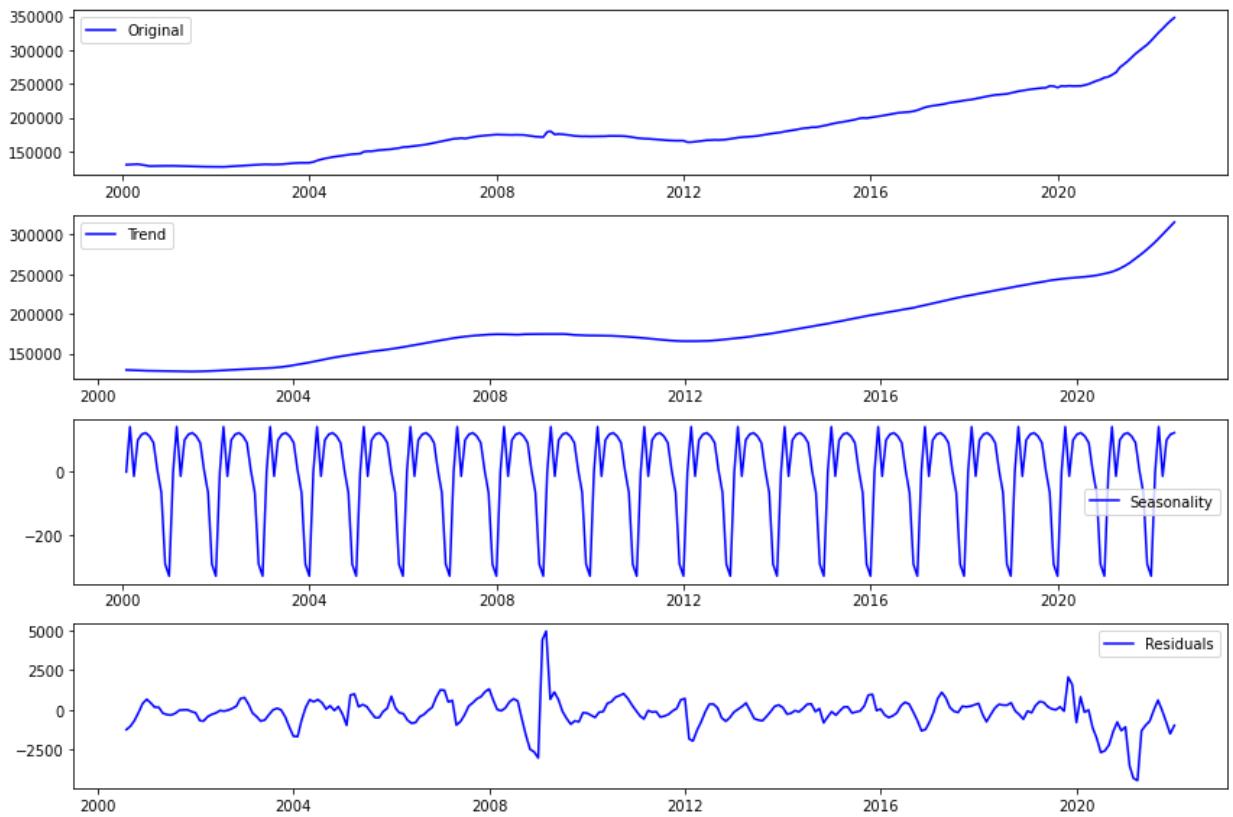
```
In [28]: 1 # #Decomposition plot of Canyon Lake
2 decomposition_plot(zip_df[8])
```

Out[28]: <function __main__.decomposition_plot(ts)>



```
In [29]: 1 # Decomposition plot of the SA Metro area averaged
          2 decomposition_plot(satx_df_melted)
```

Out[29]: <function __main__.decomposition_plot(ts)>



We can clearly see the seasonality in the individual zip codes and the SA Median average. There is also an overall upwards trend after the small dip after the housing bubble popped, with a steep increase over the last 2 years.

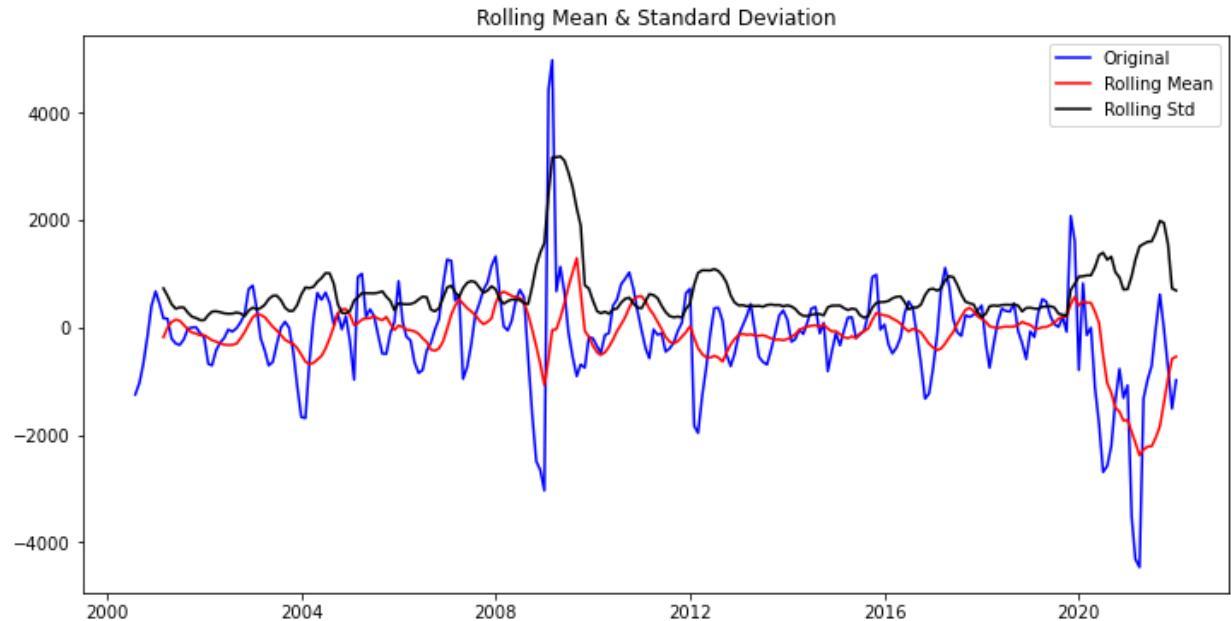
5.3 Stationarity Check

```
In [30]: 1 def stationarity_check(TS):
2
3     # Calculate rolling statistics
4     roll_mean = TS.rolling(window=8, center=False).mean()
5     roll_std = TS.rolling(window=8, center=False).std()
6
7     # Perform the Dickey Fuller test
8     dfoutput = adfuller(TS)
9
10    # Plot rolling statistics:
11    fig = plt.figure(figsize=(12,6))
12    orig = plt.plot(TS, color='blue', label='Original')
13    mean = plt.plot(roll_mean, color='red', label='Rolling Mean')
14    std = plt.plot(roll_std, color='black', label = 'Rolling Std')
15    plt.legend(loc='best')
16    plt.title('Rolling Mean & Standard Deviation')
17    plt.show(block=False)
18
19    # Print Dickey-Fuller test results
20    print('Results of Dickey-Fuller Test: \n')
21
22    dfoutput = pd.Series(dfoutput[0:4], index=['Test Statistic', 'p-value',
23                           '#Lags Used', 'Number of Observations'])
24
25    for key, value in dfoutput[4].items():
26        dfoutput['Critical Value (%s)' %key] = value
27
28    print(dfoutput)

29
30    return None
```

```
In [31]: 1 # Get decomposition for stationarity check
2 decomposition = sm.tsa.seasonal_decompose(satx_df_melted, model='additive')
3 residual = decomposition.resid
4 ts_log_decompose = residual
5 ts_log_decompose.dropna(inplace=True)
```

```
In [32]: 1 stationarity_check(ts_log_decompose)
```



Results of Dickey-Fuller Test:

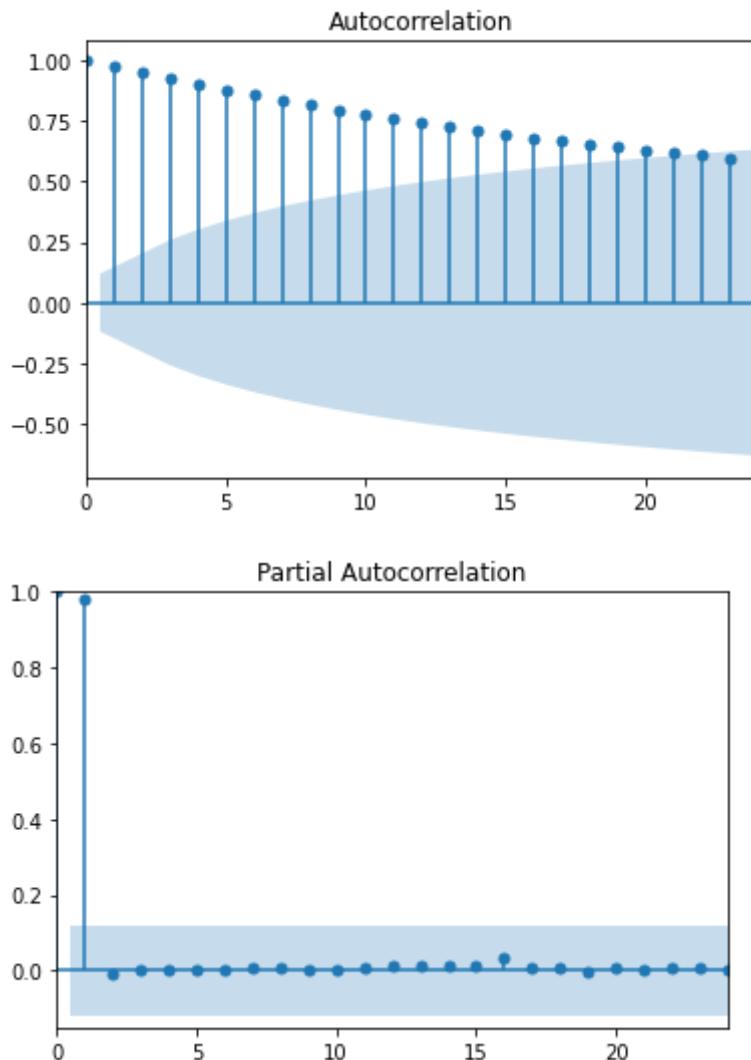
```
Test Statistic           -4.393976
p-value                 0.000304
#Lags Used              6.000000
Number of Observations Used 251.000000
Critical Value (1%)      -3.456674
Critical Value (5%)       -2.873125
Critical Value (10%)      -2.572944
dtype: float64
```

The p-value is <0.05 , which means we accept the null hypothesis that the residuals are stationary. We can move on to check our autocorrelation and partial autocorrelation plots.

▼ 5.4 Autocorrelation and Partial Autocorrelation

In [33]:

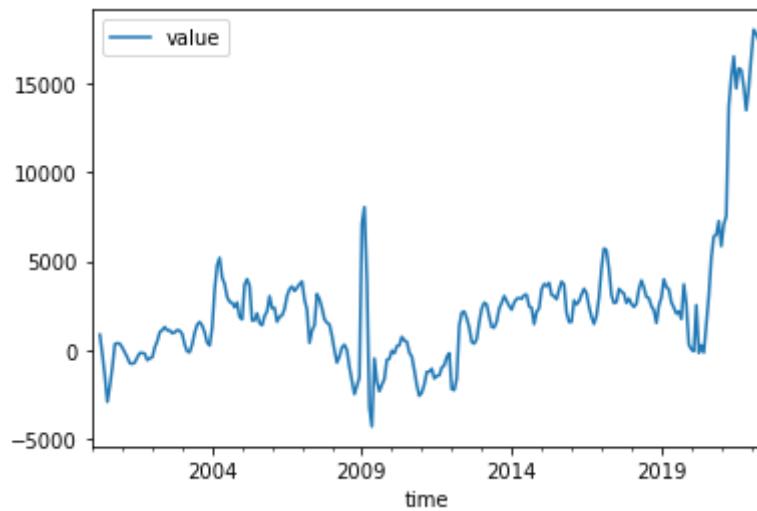
```
1 plot_acf(satx_df_melted); plt.xlim(0,24); plt.show()
2 plot_pacf(satx_df_melted); plt.xlim(0,24); plt.ylim(-0.15,1);plt.show
```



There is a slow decline in our autocorrelation plot, while the partial autocorrelation drops off right away. This means we should use an Autoregressive model for our data, but we are going to run gridsearch anyway to find best parameters. We can try to plot again with a lag of 3, which would make sense for different seasons of the year.

```
In [34]: 1 satx_df_melted.diff(periods=3).plot()
```

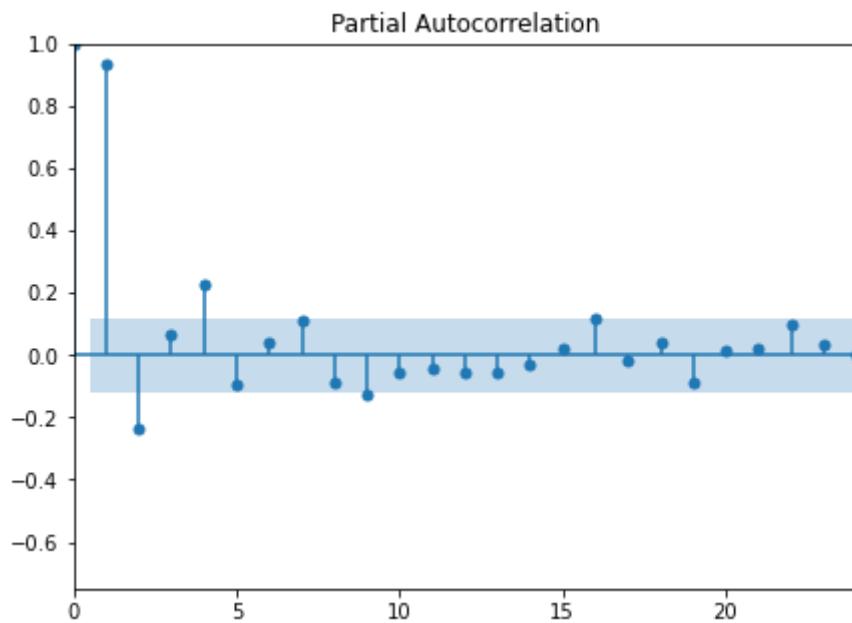
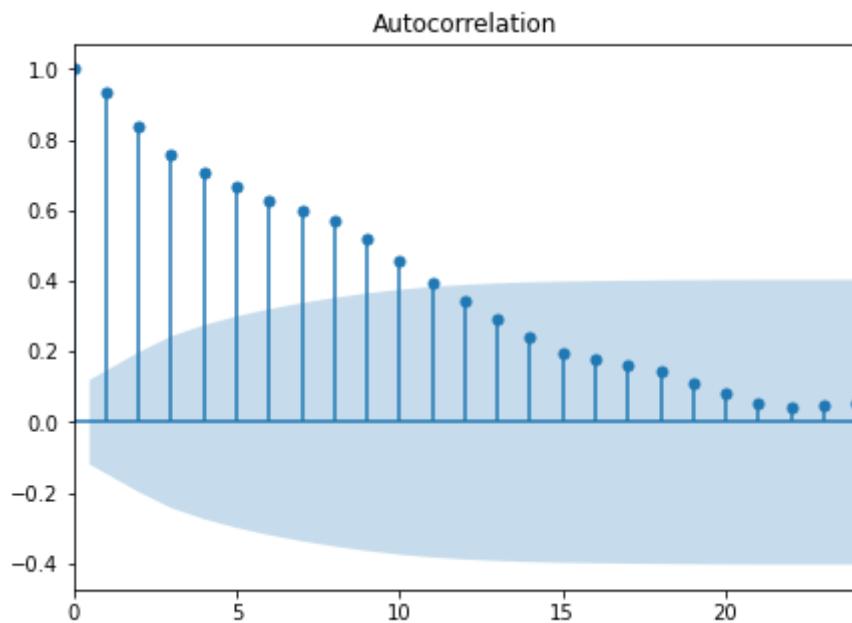
```
Out[34]: <AxesSubplot:xlabel='time'>
```



We can see some autocorrelation at a lag of 3.

In [35]:

```
1 rcParams['figure.figsize']=7,5
2 plot_acf(satx_df_melted.diff(periods=3).bfill()); plt.xlim(0,24); plt
3 plot_pacf(satx_df_melted.diff(periods=3).bfill()); plt.xlim(0,24); plt
```



In [36]: 1 model_df

Out[36]:

	RegionName	value
time		
2007-01-31	78130	167985.0
2007-02-28	78130	168585.0
2007-03-31	78130	169390.0
2007-04-30	78130	170118.0
2007-05-31	78130	170191.0
...
2022-02-28	78008	165845.0
2022-03-31	78008	167301.0
2022-04-30	78008	169326.0
2022-05-31	78008	171558.0
2022-06-30	78008	174902.0

24245 rows × 2 columns

▼ 5.5 Gridsearch for SARIMAX Model

Next, I am going to conduct gridsearch to find optimal values for (p,d,q) and (P,D,Q,s) ranked by lowest AIC. This will provide optimal paramets for each of my 10 zip codes of interest.

In [37]:

```

1 # Define the p, d and q parameters to take any value between 0 and 2
2 p = d = q = range(0, 2)
3
4 # Generate all different combinations of p, q and q triplets
5 pdq = list(itertools.product(p, d, q))
6
7 # Generate all different combinations of seasonal p, q and q triplets
8 pdqs = [(x[0], x[1], x[2], 12) for x in list(itertools.product(p, d, q))]

```

```
In [38]:  
1 # Run a grid with pdq and seasonal pdq parameters calculated above and  
2 ans = []  
3 for df, name in zip(zip_df, zip_list):  
4     for comb in pdq:  
5         for combs in pdqs:  
6             try:  
7                 mod = sm.tsa.statespace.SARIMAX(df,  
8                                              order=comb,  
9                                              seasonal_order=combs,  
10                                             enforce_stationarity=True,  
11                                             enforce_invertibility=True)  
12  
13                 output = mod.fit()  
14                 ans.append([name, comb, combs, output.aic])  
15                 print('Result for {}' .format(name) + ' ARIMA {} x {}12'.format(*combs))  
16             except:  
17                 continue  
18
```

```
Result for 78203 ARIMA (0, 0, 0) x (0, 0, 0, 12)12 : AIC Calculated = 701  
9.4166327271905  
Result for 78203 ARIMA (0, 0, 0) x (0, 0, 1, 12)12 : AIC Calculated = 840  
09.30465452891  
Result for 78203 ARIMA (0, 0, 0) x (0, 1, 0, 12)12 : AIC Calculated = 562  
2.870386243331  
Result for 78203 ARIMA (0, 0, 0) x (0, 1, 1, 12)12 : AIC Calculated = 537  
1.74710460903  
Result for 78203 ARIMA (0, 0, 0) x (1, 0, 0, 12)12 : AIC Calculated = 550  
0.263984338111  
Result for 78203 ARIMA (0, 0, 0) x (1, 0, 1, 12)12 : AIC Calculated = 548  
1.179697261492  
Result for 78203 ARIMA (0, 0, 0) x (1, 1, 0, 12)12 : AIC Calculated = 523  
8.3061736622185  
Result for 78203 ARIMA (0, 0, 0) x (1, 1, 1, 12)12 : AIC Calculated = 646  
1.6577264304215  
Result for 78203 ARIMA (0, 0, 1) x (0, 0, 0, 12)12 : AIC Calculated = 679  
9.994138687387  
Result for 78203 ARIMA (0, 0, 1) x (0, 0, 1, 12)12 : AIC Calculated = 777  
22.65070101767
```

In [39]:

```
1 # Find the parameters with minimal AIC value
2 ans_df = pd.DataFrame(ans, columns=['name', 'pdq', 'pdqs', 'aic'])
3 ans_df
```

Out[39]:

	name	pdq	pdqs	aic
0	78203	(0, 0, 0)	(0, 0, 0, 12)	7019.416633
1	78203	(0, 0, 0)	(0, 0, 1, 12)	84009.304655
2	78203	(0, 0, 0)	(0, 1, 0, 12)	5622.870386
3	78203	(0, 0, 0)	(0, 1, 1, 12)	5371.747105
4	78203	(0, 0, 0)	(1, 0, 0, 12)	5500.263984
...
635	78220	(1, 1, 1)	(0, 1, 1, 12)	3807.534751
636	78220	(1, 1, 1)	(1, 0, 0, 12)	3942.412763
637	78220	(1, 1, 1)	(1, 0, 1, 12)	3929.565653
638	78220	(1, 1, 1)	(1, 1, 0, 12)	3830.610297
639	78220	(1, 1, 1)	(1, 1, 1, 12)	3812.442178

640 rows × 4 columns

In [40]:

```
1 #Return the best set of parameters based on AIC
2 best_para = ans_df.loc[ans_df.groupby("name")["aic"].idxmin()]
3 best_para
```

Out[40]:

	name	pdq	pdqs	aic
319	78055	(1, 1, 1)	(1, 1, 1, 12)	2318.584674
575	78133	(1, 1, 1)	(1, 1, 1, 12)	2308.113622
123	78202	(1, 1, 1)	(0, 1, 1, 12)	4068.244419
63	78203	(1, 1, 1)	(1, 1, 1, 12)	4144.531317
187	78207	(1, 1, 1)	(0, 1, 1, 12)	3833.302485
383	78210	(1, 1, 1)	(1, 1, 1, 12)	3868.587565
635	78220	(1, 1, 1)	(0, 1, 1, 12)	3807.534751
443	78237	(1, 1, 1)	(0, 1, 1, 12)	3843.766288
511	78623	(1, 1, 1)	(1, 1, 1, 12)	2765.827273
255	78638	(1, 1, 1)	(1, 1, 1, 12)	2641.330470

5.6 Best Parameters SARIMAX Model

Now that we have our optimal parameters for each zip code from our gridsearch, we will use the to

build a sarimax model for each zip code. Following are 3 functions we will use to build, evaluate and test each model, and forecast predictions for each of our top 10 zip codes.

```
In [41]: 1 # Define function to build SARIMAX model, print summary and output di...
  2 def sarimax(ts, order=(1, 1, 1), seasonal_order=(1, 1, 1, 12)):
  3     """
  4     Input:
  5     ts : Time series data.
  6     order : (p, d, q) values for ARIMA. Default = (1, 1, 1)
  7     seasonal_order : Seasonal (P, D, Q, s) values for SARIMA.
  8             Default = (1, 1, 1, 12).
  9
 10    Output:
 11    Prints model output summary.
 12    Plots model diagnostics.
 13
 14    Returns: Model output.
 15    """
 16
 17    # Sarimax model
 18    SARIMAX = sm.tsa.statespace.SARIMAX(ts,
 19                                         order=order,
 20                                         seasonal_order=seasonal_order,
 21                                         enforce_stationarity=False,
 22                                         enforce_invertibility=False)
 23
 24    # Fit the model and print results
 25    output = SARIMAX.fit()
 26
 27    # Print output summary
 28    print(output.summary().tables[1])
 29    output.plot_diagnostics(figsize=(15,18));
 30
 31    return output
```

```
In [42]: 1 def ose_forecast(ts, output, neighborhood = ''):  
2     '''  
3     Plot forecast with real and predicted data.  
4  
5     Input:  
6         ts : Time series.  
7         output : SARIMAX model output.  
8         neighborhood : Name of city for plot title.  
9  
10    Output:  
11        Prints RMSE.  
12        Plots real vs. model predicted plot.  
13        '''  
14  
15    # Get predictions starting from 2019-06-30 and calculate confidence intervals.  
16  
17    pred = output.get_prediction(start=pd.to_datetime('2019-06-30')),  
18  
19    # Get the real and predicted values  
20    ts_forecasted = pred.predicted_mean  
21    ts_truth = ts['2019-06-30':].value  
22  
23    # Calc RMSE  
24    mse = mean_squared_error(ts_truth, ts_forecasted)  
25    rmse = np.sqrt(mse)  
26  
27    # Print RMSE  
28    print('The RMSE of our forecasts is {}'.format(round(rmse, 2)))  
29  
30    # Confidence Intervals  
31    pred_conf = pred.conf_int()  
32  
33    ### Plot real vs predicted w/ confidence intervals ###  
34  
35    rcParams['figure.figsize'] = 15, 6  
36  
37    # Plot observed values  
38    ax = ts['2017-06-30':].plot(label='observed')  
39  
40    # Plot predicted values  
41  
42    ts_forecasted.plot(ax=ax, label='one-step ahead forecast', alpha=.5)  
43  
44    # Plot the range for confidence intervals  
45    ax.fill_between(pred_conf.index,  
46                      pred_conf.iloc[:, 0],  
47                      pred_conf.iloc[:, 1], color='g', alpha=.3)  
48  
49    # Title  
50    ax.set_title(neighborhood)  
51  
52    # Set axes labels  
53    ax.set_xlabel('Date')  
54    ax.set_ylabel('Median Home Value (USD)')  
55    plt.legend()  
56
```

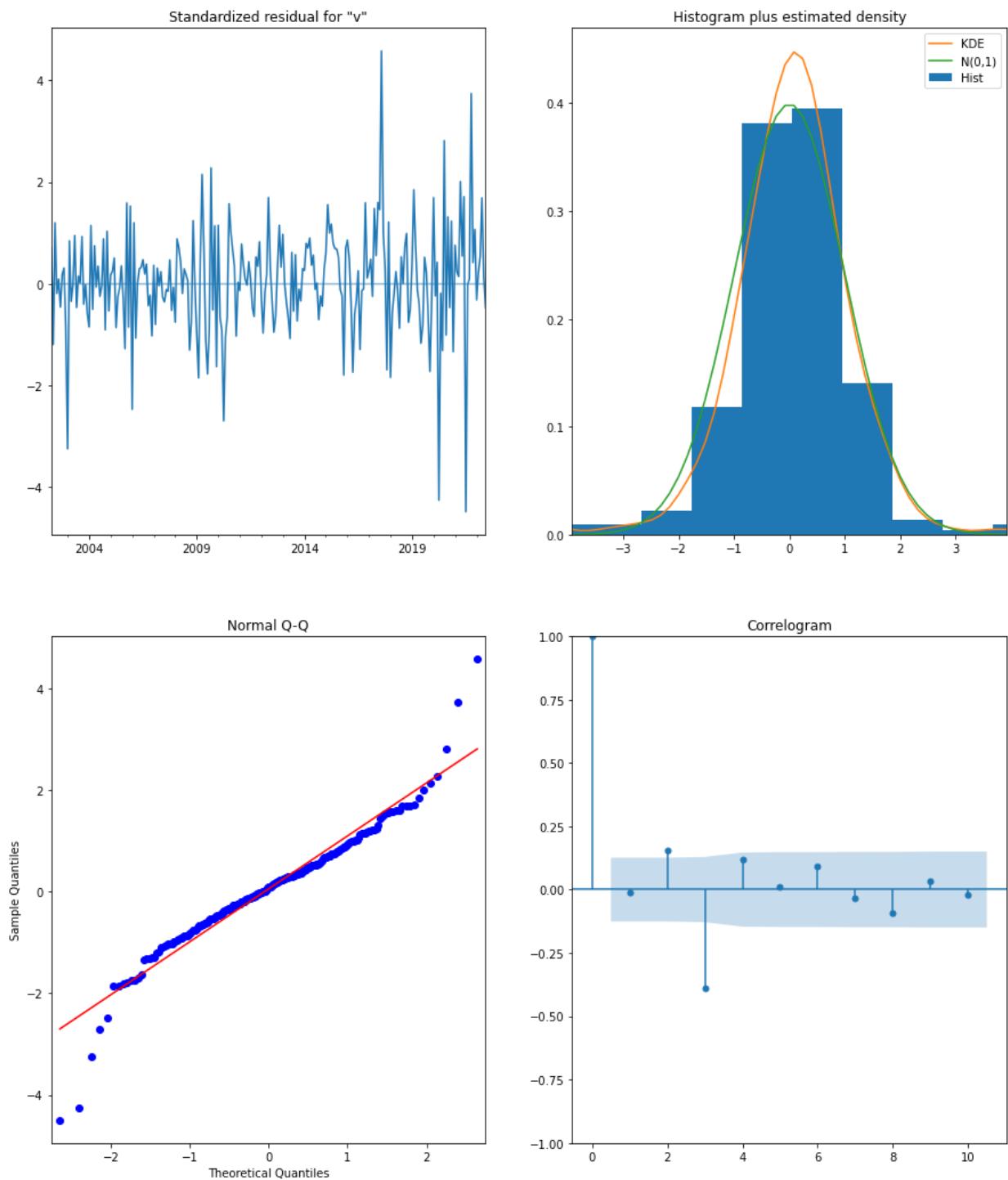
```
57 |     return plt.show()
```

```
In [43]: 1 # Forecast 5 year predictions - Home Sale Values #
2
3 def dynamic_forecast(ts, model_output, years=5, neighborhood='', save_
4   '''
5     Plots dynamic forecast for specified time into the future.
6
7   Inputs:
8     ts : Time series data.
9     model_output : Output results from our model.
10    years : n-years to forecast into the future.
11    neighborhood: Name of neighborhood for plots
12
13   Outputs:
14     Prints : Time series plot with one step ahead forecast.
15     Returns : Dictionary of predictions.
16
17   ...
18
19   # Calcualte steps
20   steps = years*12
21
22   # Get forecast and confidence interval for steps ahead in future
23   future = model_output.get_forecast(steps=steps, dynamic=True, full_
24   future_conf = future.conf_int()
25
26   ### Plot forecast ####
27
28   # Observed
29   ax = ts.plot(label='Observed', figsize=(12, 6))
30   # Predicted
31   future.predicted_mean.plot(ax=ax, label='dynamic forecast', alpha=.
32
33   # Confidence Intervals
34   ax.fill_between(future_conf.index,
35                   future_conf.iloc[:, 0],
36                   future_conf.iloc[:, 1], color='k', alpha=.25)
37
38   # Title
39   ax.set_title(f"5-Year Forecast for {neighborhood}")
40
41   # x & y-labels
42   ax.set_xlabel('Date')
43   ax.set_ylabel('Median Home Sale Value (USD)')
44
45   # Legend
46   ax.legend()
47
48   if save_fig:
49     plt.savefig(f'./images/dynamic_forecast_{neighborhood}.jpeg')
50
51   # Print plot
52   plt.show()
53
54   # Forecast prediction for n-years into the future
55   forecast = future.predicted_mean[-1]
56   maximum = future_conf.iloc[-1, 1]
```

```
57     minimum = future_conf.iloc[-1,0]
58
59     # Create dictionary of predictions
60     predictions = {}
61     predictions['forecast'] = forecast.round()
62     predictions['minimum'] = minimum.round()
63     predictions['maximum'] = maximum.round()
64
65     return predictions
```

▼ **5.6.1 Arena District**

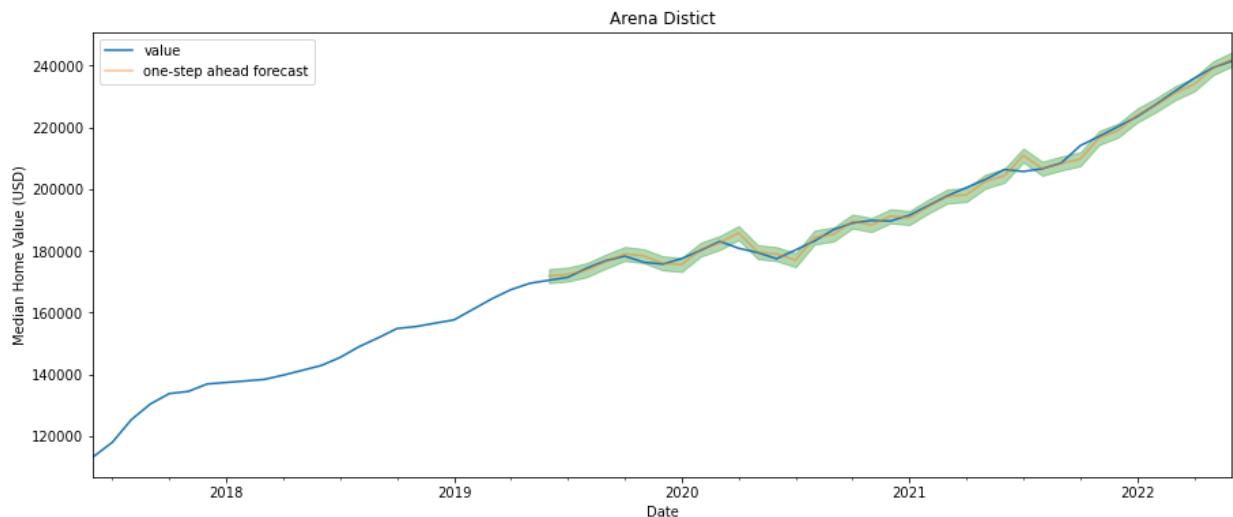
In [44]:



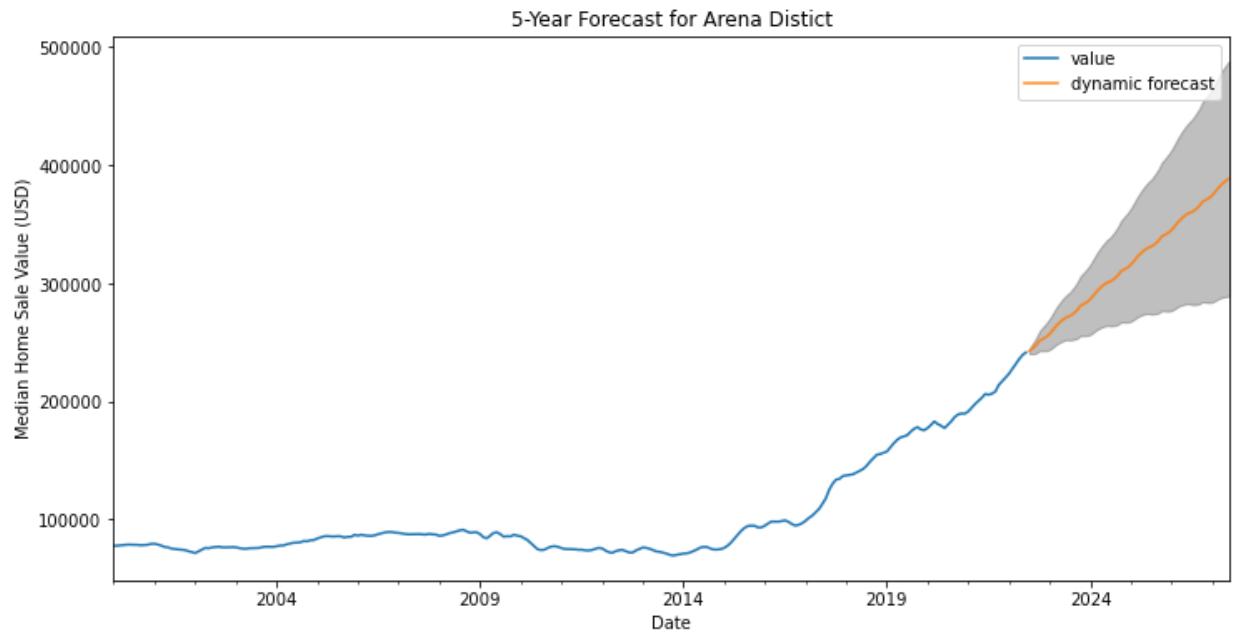
The residuals look normally distributed, even though there are some heavy outliers presence in the tails of the Q-Q plot. We are likely going to see this in several of our following models. This is due to the model being "confused" over the dip after the housing bubble, and the huge increases that have taken place in this area over the last few years. We can see negative correlation in the correlogram at a month 3. This makes sense knowing that we have seasonality in our data. The `ma.L1` has a p-value that is >0.05 , and we will likely see this in all of our following zipcodes. I am going to leave the model as is since I ran gridsearch to find optimal parameters.

```
In [45]: 1 close_forecast(arena_district, output_arena_district, neighborhood='Arena District')
```

The RMSE of our forecasts is 1816.45



```
In [46]: 1 dynamic_forecast(arena_district, output_arena_district, neighborhood='Arena District')
```



```
Out[46]: {'forecast': 388893.0, 'minimum': 289050.0, 'maximum': 488735.0}
```

The RMSE of our one step ahead forecast for the Arena District look great at \$1,816! The confidence interval is small and the ROI of our minimum bounds are higher than the current value, which guarantees a positive ROI.

▼ 5.6.2 Dignowity Hill

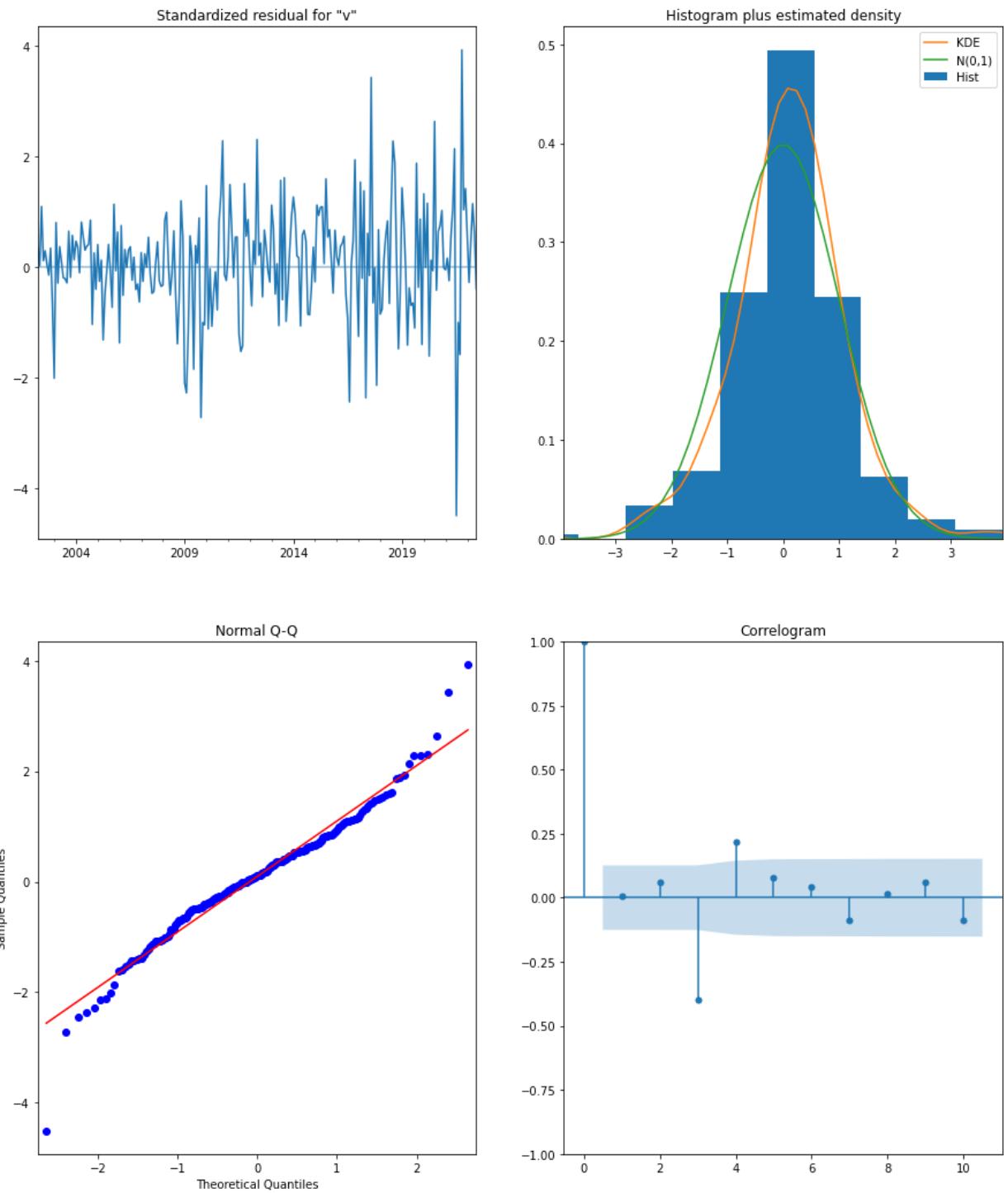
In [47]:

```
1 dig_hill = zip_df[1]
2 output_dig_hill = sarimax(dig_hill, order=(1, 1, 1), seasonal_order=(0, 0, 0, 0))
=====
=====
```

	coef	std err	z	P> z	[0.025
0.975]					

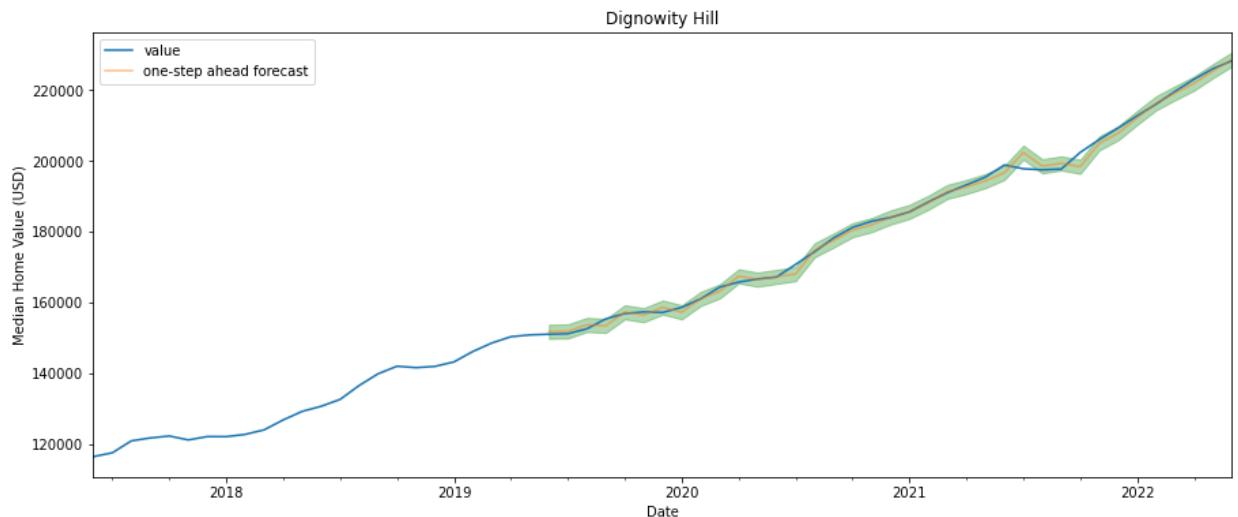
ar.L1	0.4494	0.109	4.136	0.000	0.236
0.662					
ma.L1	0.1275	0.130	0.981	0.326	-0.127
0.382					
ar.S.L12	-0.3287	0.071	-4.651	0.000	-0.467
0.190					
ma.S.L12	-0.1885	0.049	-3.876	0.000	-0.284
0.093					
sigma2	1.045e+06	7.6e+04	13.747	0.000	8.96e+05
9e+06					1.1

=====					



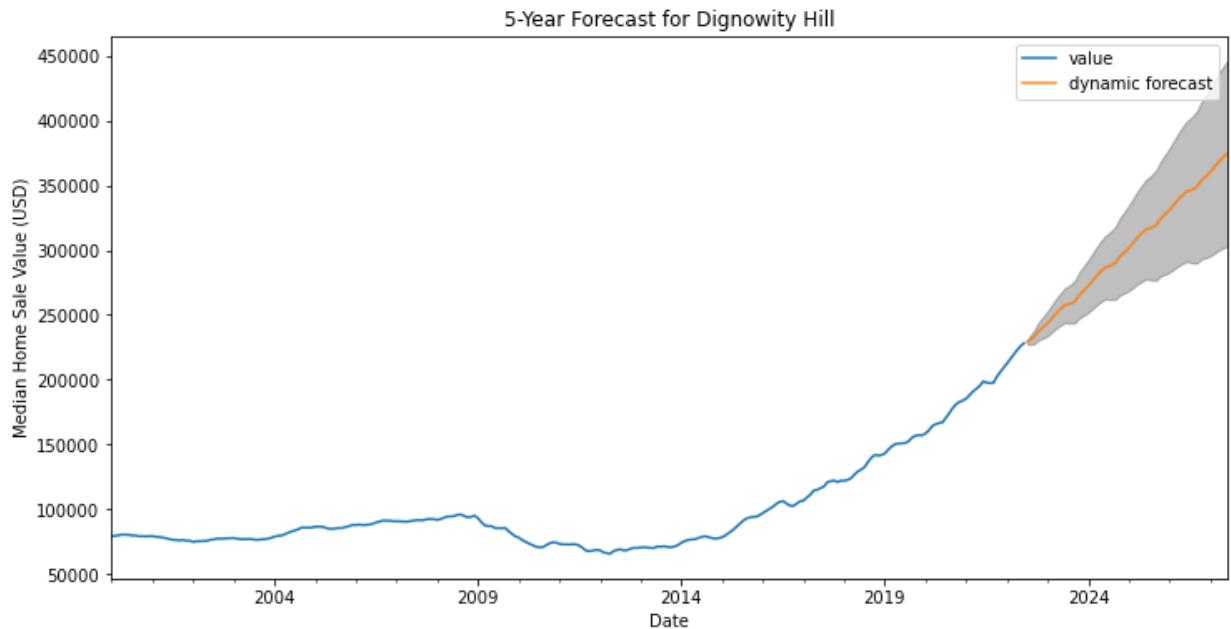
```
In [48]: 1 ose_forecast(dig_hill, output_dig_hill, neighborhood='Dignowity Hill')
```

The RMSE of our forecasts is 1446.54



In [49]:

```
1 dynamic_forecast(dig_hill, output_dig_hill, neighborhood='Dignowity Hill')
```



Out[49]: `{'forecast': 374878.0, 'minimum': 303162.0, 'maximum': 446593.0}`

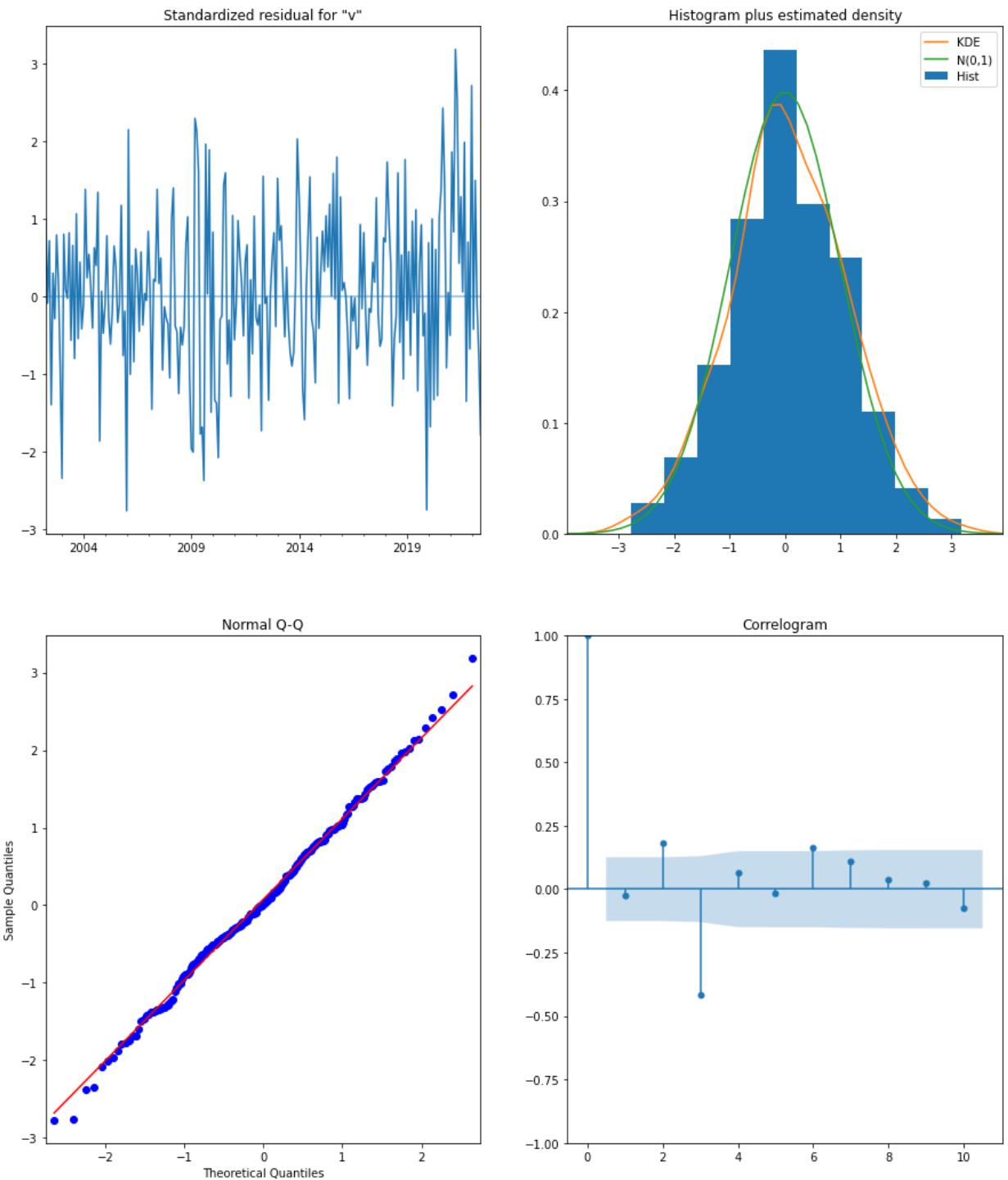
Once again, we see negative correlation at month 3, this should be expected with all of the rest of our zipcode plots. The residuals do have a few tail outliers, but are distributed normally. The confidence interval of our forecasts is smaller than what we saw in the Arena District, and the RMSE is lower at \$1446.

▼ **5.6.3 Gardendale / Brady Gardens**

In [50]:

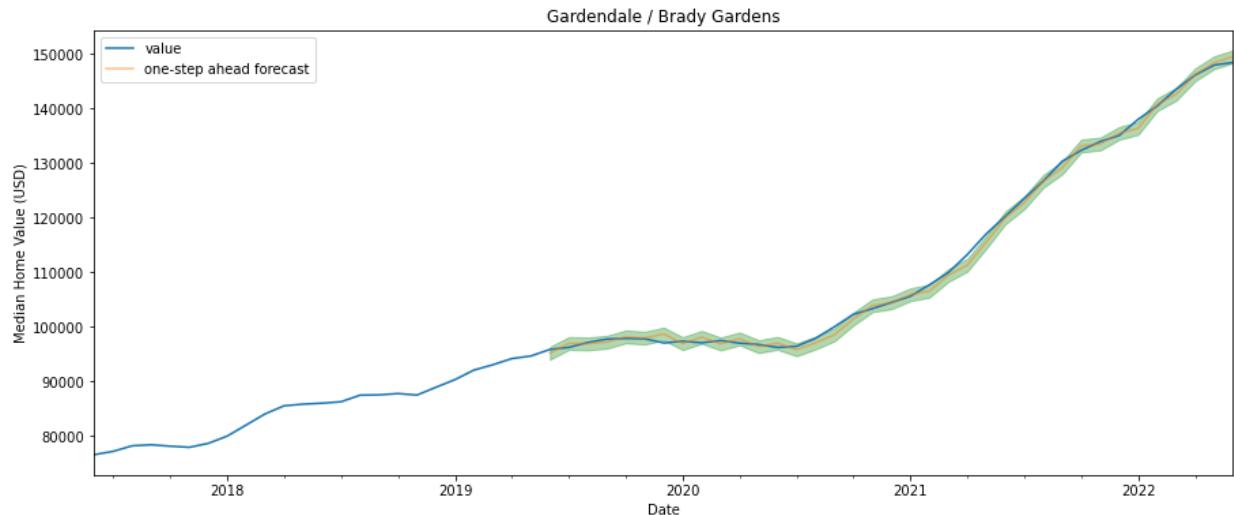
```
1 b_gardens = zip_df[2]
2 output_b_gardens = sarimax(b_gardens, order=(1, 1, 1), seasonal_order=[0, 0, 0, 0])
=====
=====
```

	coef	std err	z	P> z	[0.025
0.975]					
-----	-----	-----	-----	-----	-----
ar.L1	0.7256	0.049	14.850	0.000	0.630
0.821					
ma.L1	0.0109	0.070	0.156	0.876	-0.126
0.147					
ma.S.L12	-0.6518	0.043	-15.195	0.000	-0.736
0.568					
sigma2	3.649e+05	3.2e+04	11.394	0.000	3.02e+05
8e+05					4.2
-----	-----	-----	-----	-----	-----
=====	=====	=====	=====	=====	=====

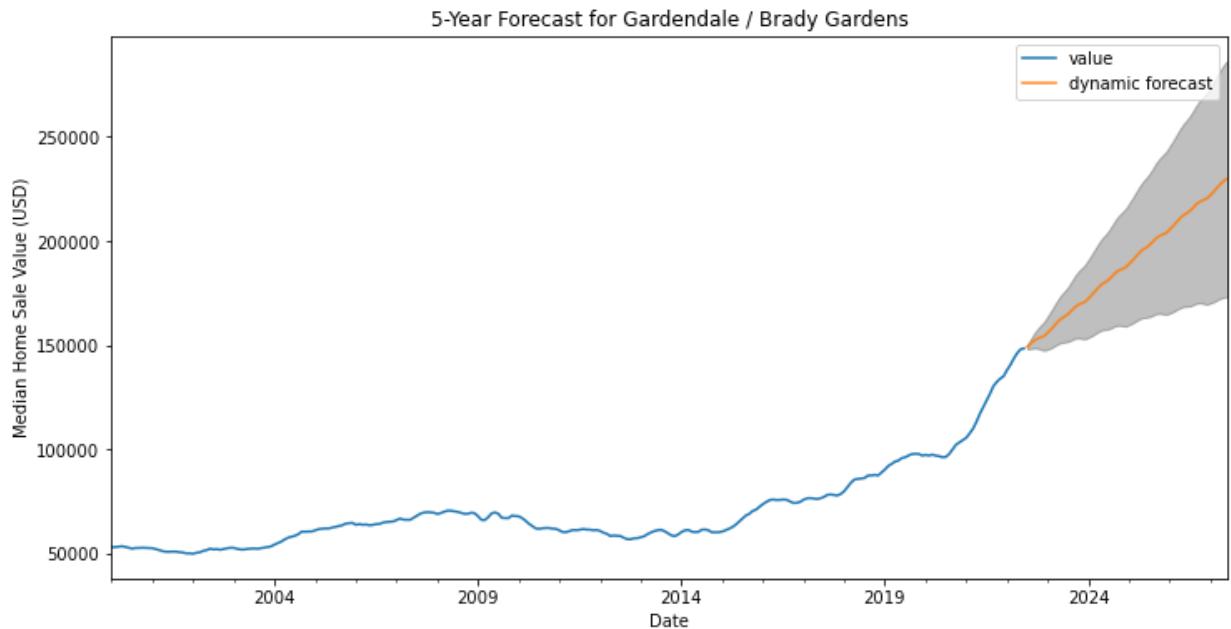


```
In [51]: 1 ose_forecast(b_gardens, output_b_gardens, neighborhood='Gardendale / I
```

The RMSE of our forecasts is 856.1



```
In [52]: 1 dynamic_forecast(b_gardens, output_b_gardens, neighborhood='Gardendale')
```



```
Out[52]: {'forecast': 229884.0, 'minimum': 173058.0, 'maximum': 286710.0}
```

Once again, we achieved a lower RMSE for Gardendale area at \$856. The residuals are normally distributed without any significant outliers.

▼ 5.6.4 Kingsbury

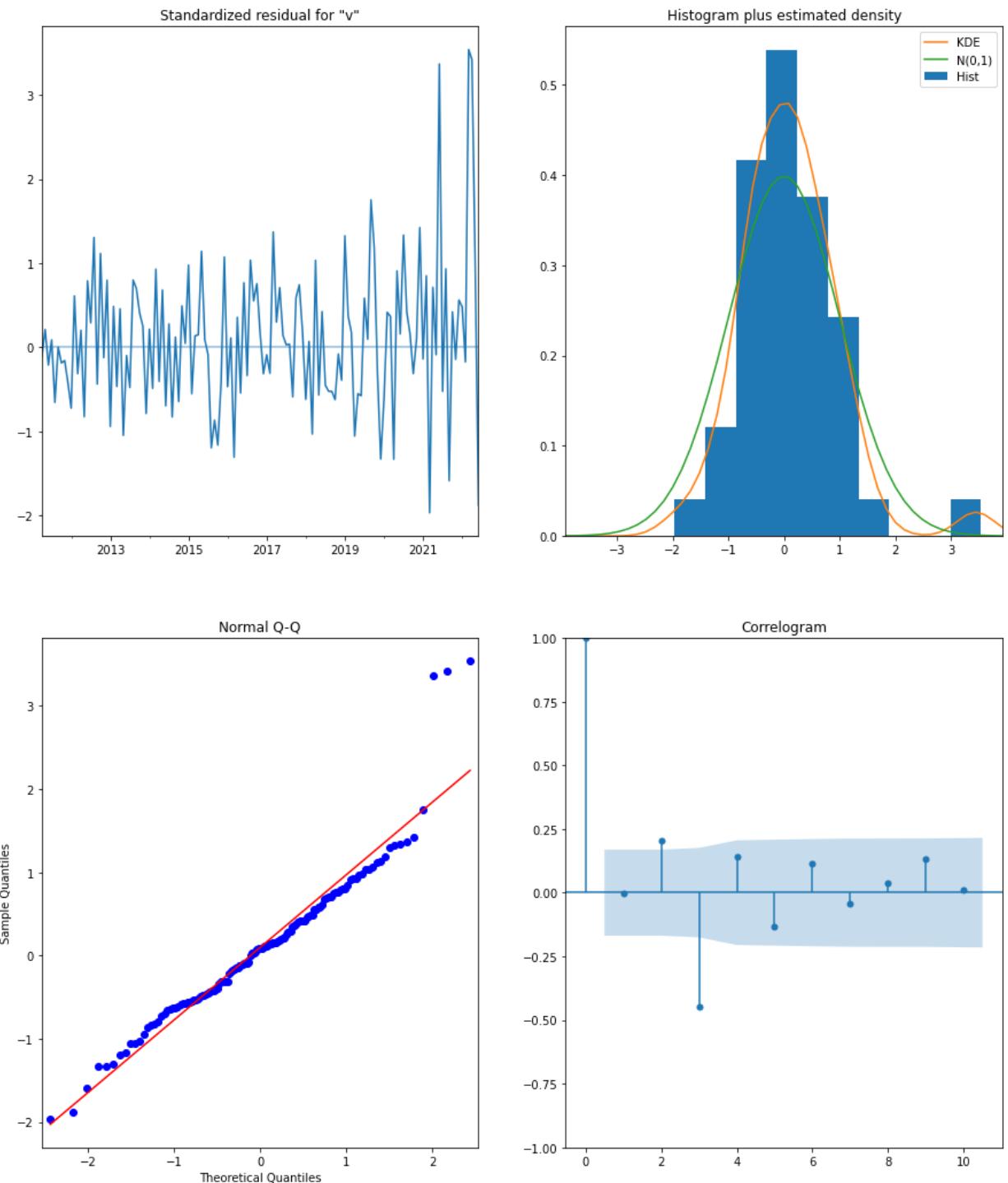
In [53]:

```
1 kingsbury = zip_df[3]
2 output_kingsbury = sarimax(kingsbury, order=(1, 1, 1), seasonal_order=[0, 0, 0, 0])
=====
=====
```

	coef	std err	z	P> z	[0.025
0.975]					

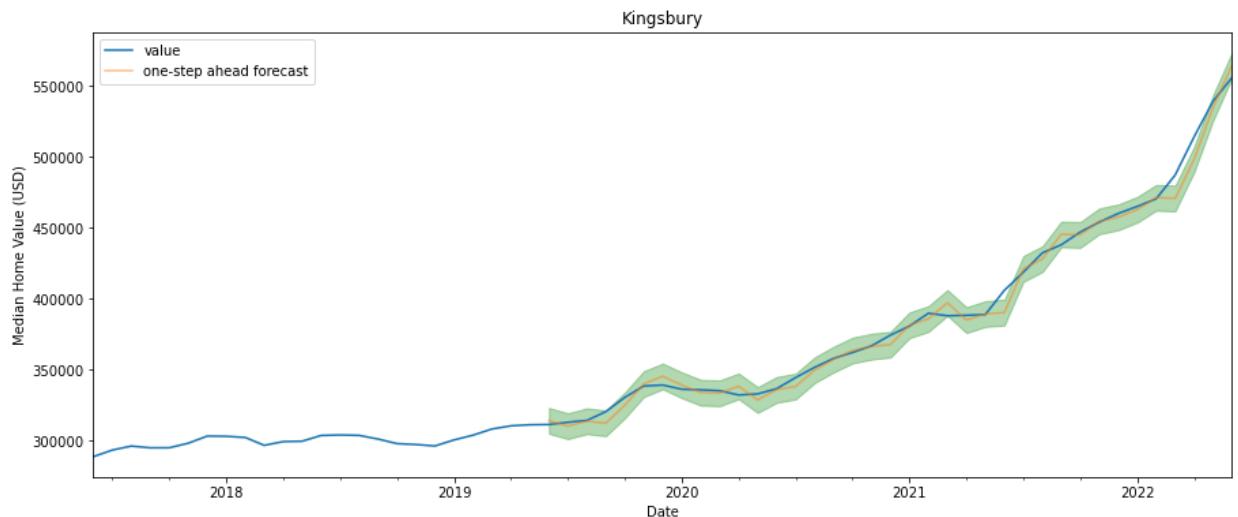
ar.L1	0.6708	0.102	6.553	0.000	0.470
0.871					
ma.L1	0.0476	0.137	0.348	0.728	-0.221
0.316					
ar.S.L12	-0.5986	0.112	-5.325	0.000	-0.819
0.378					
ma.S.L12	-0.0105	0.061	-0.171	0.864	-0.131
0.110					
sigma2	2.151e+07	2.25e-09	9.56e+15	0.000	2.15e+07
5e+07					2.1

=====					

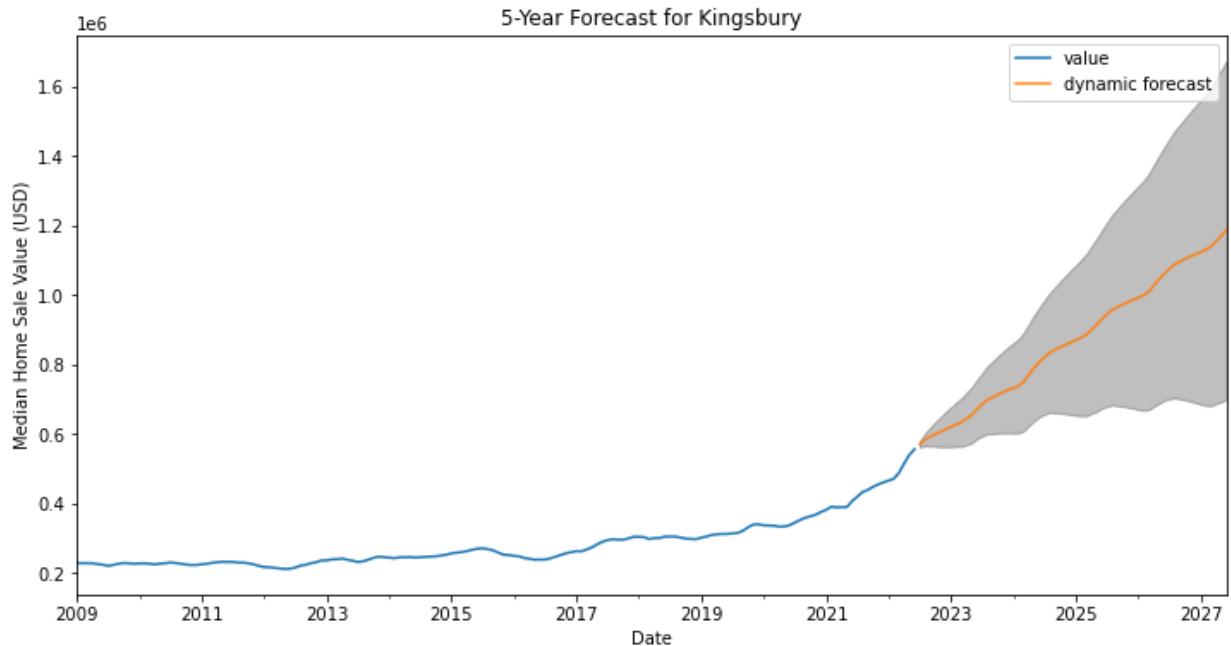


```
In [54]: 1 ose_forecast(kingsbury, output_kingsbury, neighborhood='Kingsbury' )
```

The RMSE of our forecasts is 6104.66



```
In [55]: 1 dynamic_forecast(kingsbury, output_kingsbury, neighborhood='Kingsbury')
```



```
Out[55]: {'forecast': 1187524.0, 'minimum': 698989.0, 'maximum': 1676059.0}
```

The RMSE of our predictions is \$6,104 which is higher than our other zipcodes but the price of the homes are much higher in Kingsbury than the other areas. However, Kingsbury has some pretty significant outliers on the high end which is skewing the distribution of the residuals.

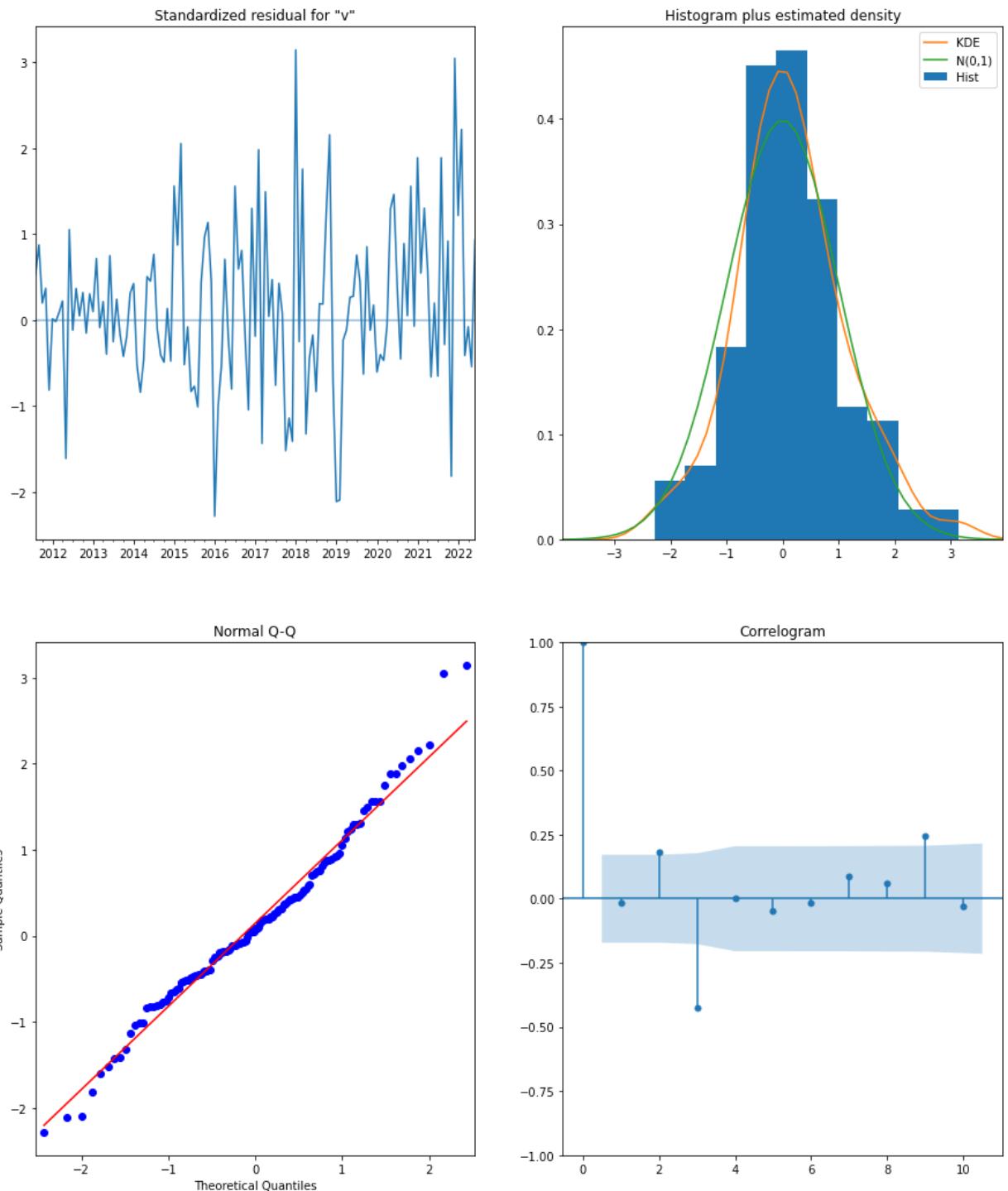
▼ **5.6.5 Medina**

In [56]:

```
1 medina = zip_df[4]
2 output_medina = sarimax(medina, order=(1, 1, 1), seasonal_order=(0, 1
=====
=====
```

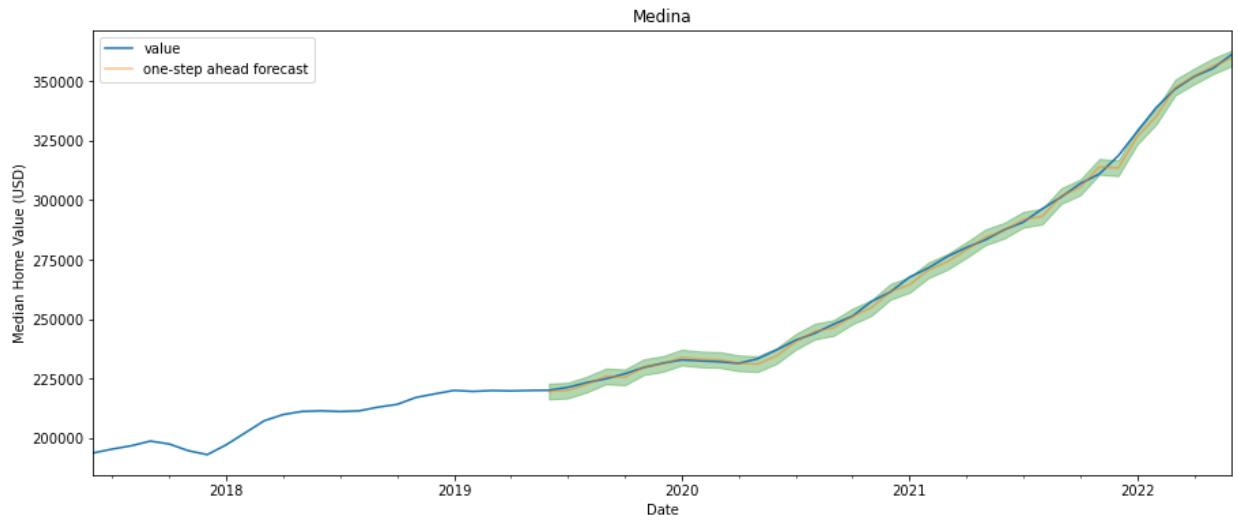
	coef	std err	z	P> z	[0.025
0.975]					
ar.L1	0.6341	0.093	6.837	0.000	0.452
0.816					
ma.L1	0.0627	0.119	0.527	0.599	-0.171
0.296					
ma.S.L12	-0.2064	0.039	-5.249	0.000	-0.284
0.129					
sigma2	2.866e+06	3.32e+05	8.626	0.000	2.21e+06
2e+06					3.5

```
=====
=====
```



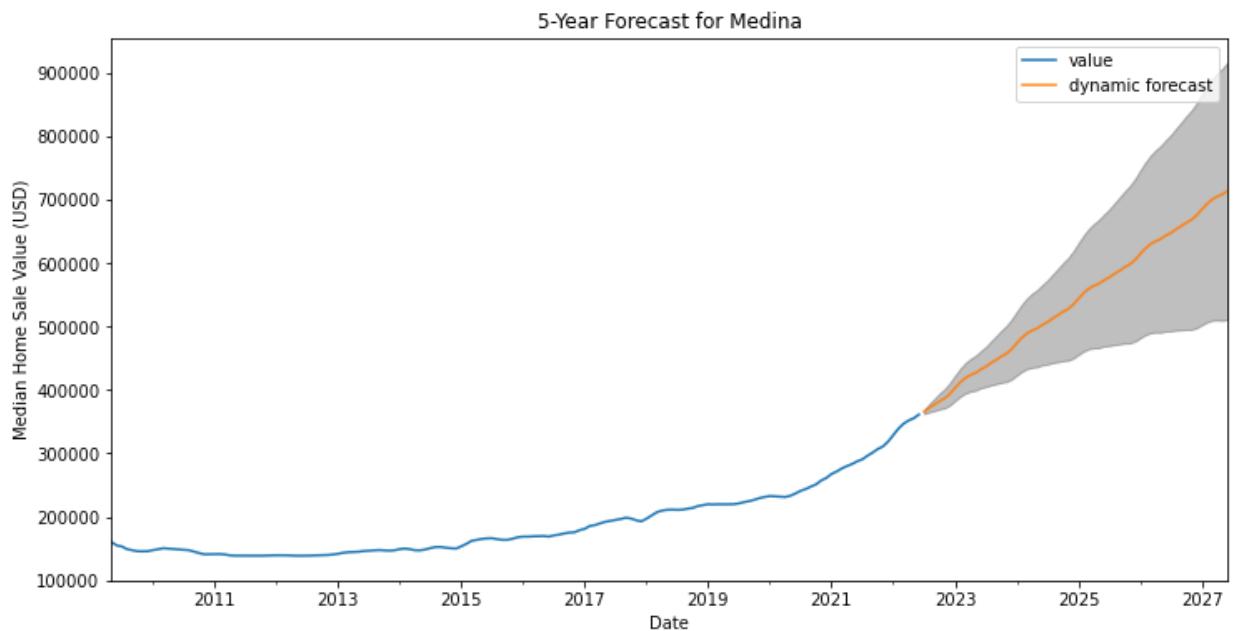
```
In [57]: 1 ose_forecast(medina, output_medina, neighborhood='Medina')
```

The RMSE of our forecasts is 1798.26



In [58]:

1 dynamic_forecast(medina, output_medina, neighborhood='Medina', years=1)



Out[58]: {'forecast': 713491.0, 'minimum': 510850.0, 'maximum': 916132.0}

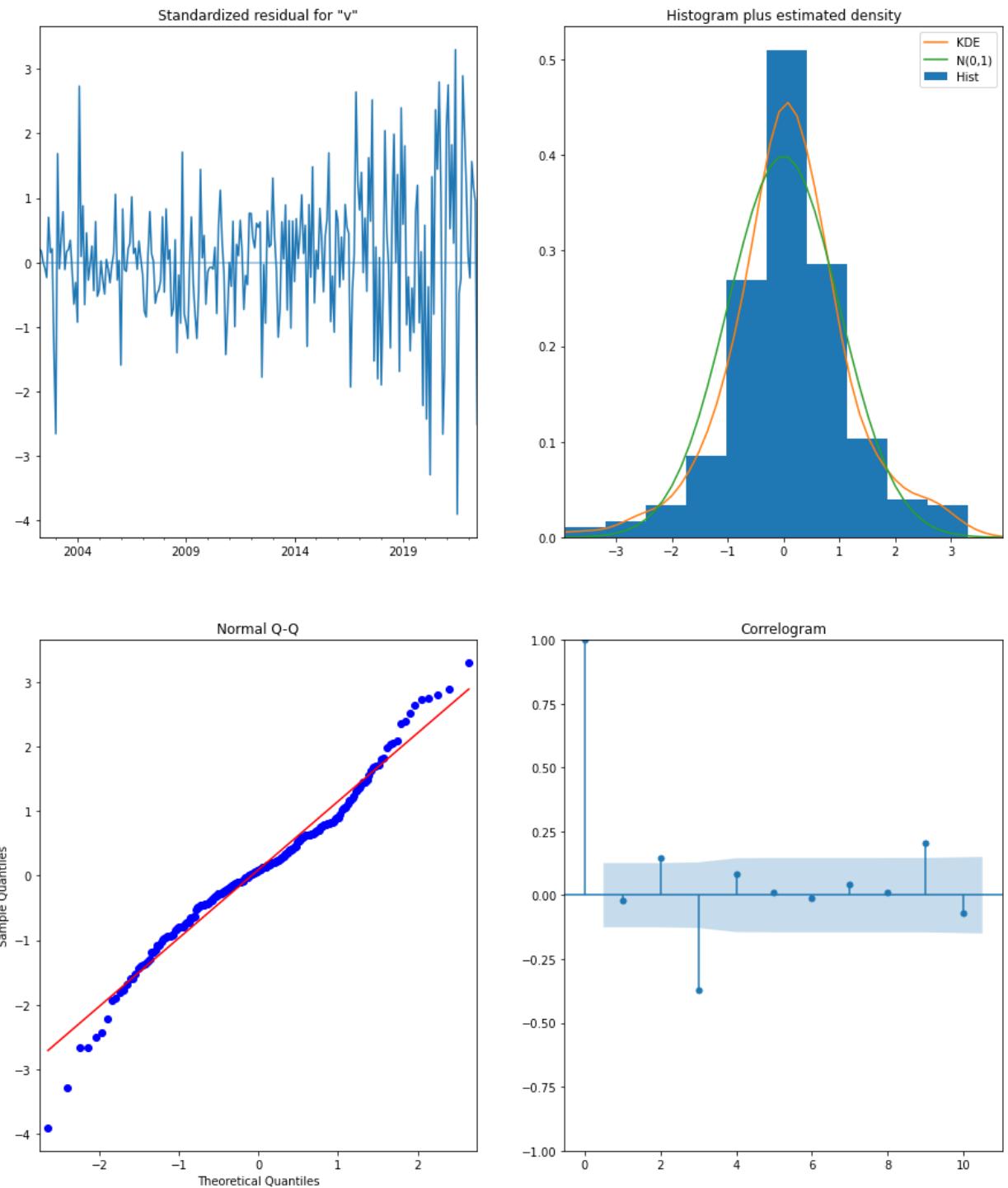
The RMSE of our predictions for Medina is \$1,798, which is similar to some of our other zipcodes we have already seen, and we can see the presence of a few outliers in the Q-Q plot.

▼ **5.6.6 Denver Heights / Highland Park**

In [59]:

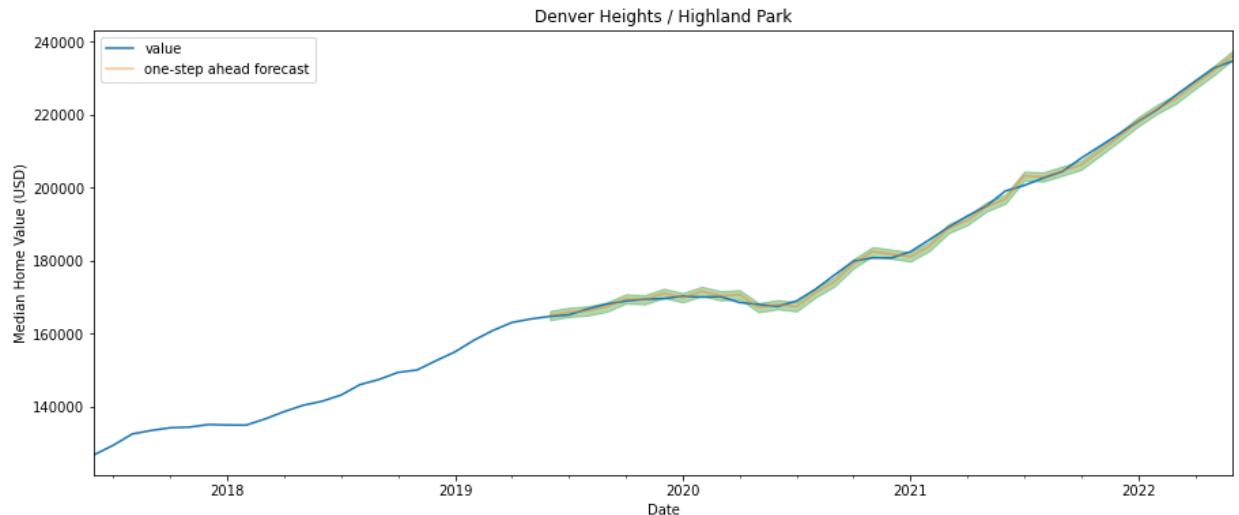
```
1 highland_park = zip_df[5]
2 output_highland_park = sarimax(highland_park, order=(1, 1, 1), seasonal=4)
=====
=====
```

	coef	std err	z	P> z	[0.025
0.975]					
-----	-----	-----	-----	-----	-----
ar.L1	0.6916	0.040	17.311	0.000	0.613
0.770					
ma.L1	0.1066	0.067	1.597	0.110	-0.024
0.237					
ma.S.L12	-0.5716	0.034	-16.696	0.000	-0.639
0.504					
sigma2	4.35e+05	3.03e+04	14.333	0.000	3.76e+05
4e+05					4.9
-----	-----	-----	-----	-----	-----
=====	=====	=====	=====	=====	=====



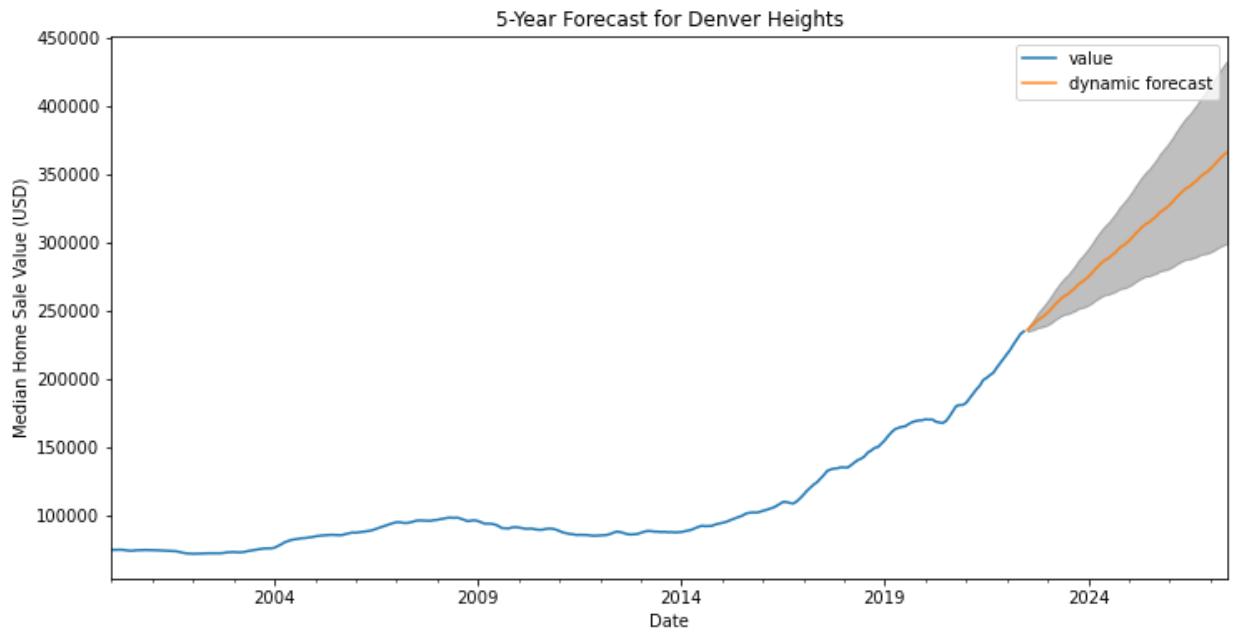
```
In [60]: 1 ose_forecast(highland_park, output_highland_park, neighborhood='Denver')
```

The RMSE of our forecasts is 1198.29



In [61]:

1 dynamic_forecast(highland_park, output_highland_park, neighborhood='De

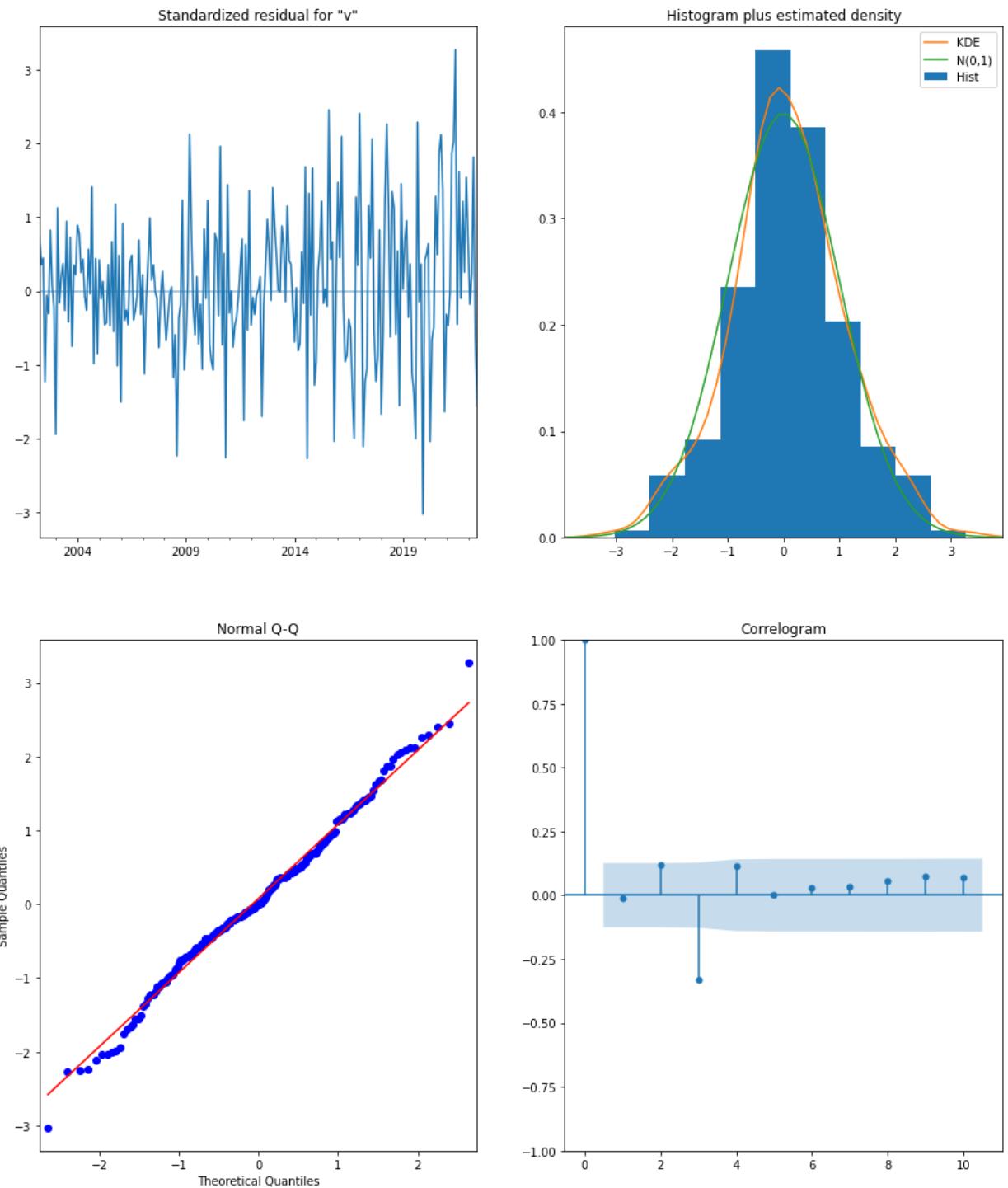


Out[61]: {'forecast': 366017.0, 'minimum': 298896.0, 'maximum': 433138.0}

Once again, residuals are mostly normal with outliers on the tails. The RMSE of our predictions is \$1198. The confidence interval appears to be smaller than several others we have seen.

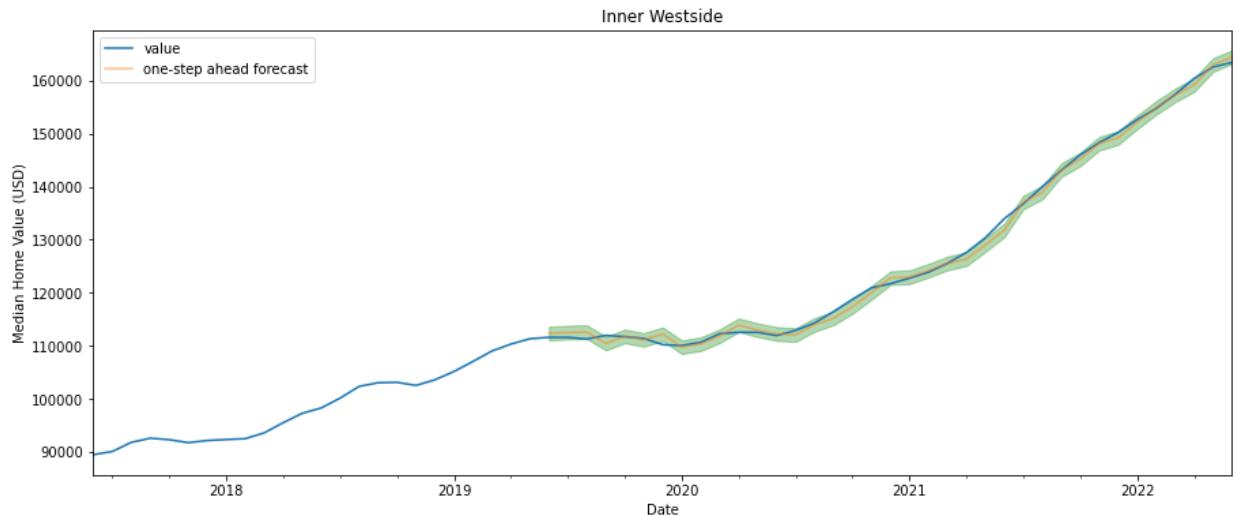
▼ **5.6.7 Inner Westside**

In [62]:

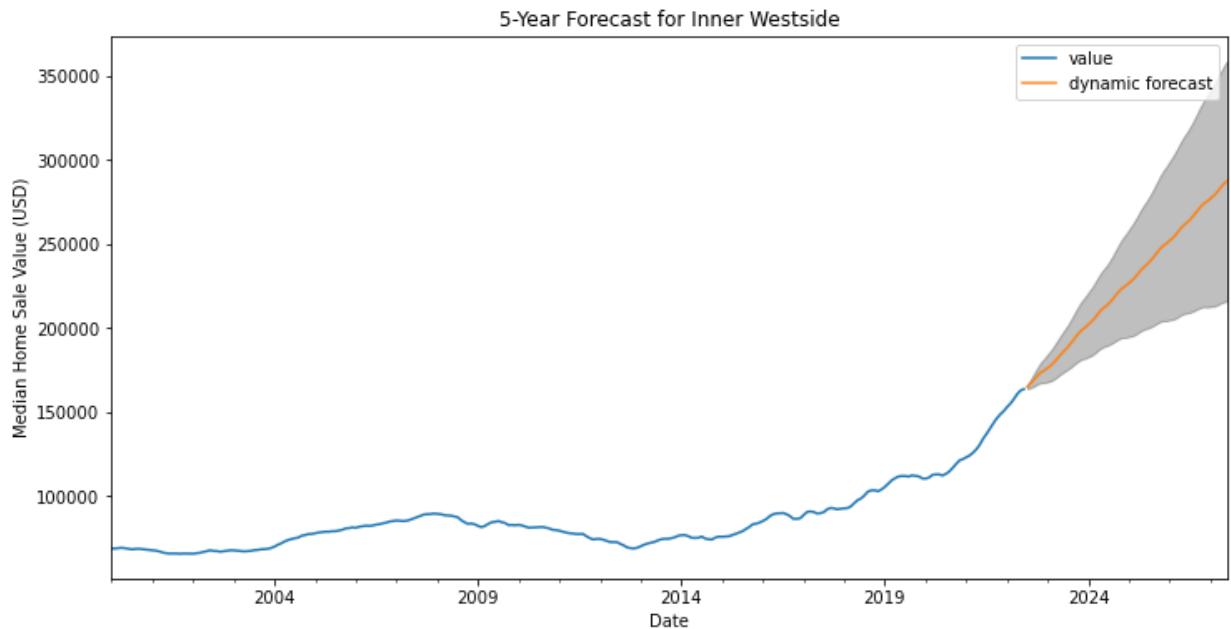


```
In [63]: 1 ose_forecast(inner_westside, output_inner_westside, neighborhood='Inne
```

The RMSE of our forecasts is 933.67



```
In [64]: 1 dynamic_forecast(inner_westside, output_inner_westside, neighborhood=
```



```
Out[64]: {'forecast': 287589.0, 'minimum': 215798.0, 'maximum': 359380.0}
```

The residuals are normal for the inner westside zipcode, and we have an RMSE of \$933. The price point for this neighborhood is around 150k right now, with the forecast expected to increase to 287k in 5 years.

▼ **5.6.8 Fisher**

In [65]:

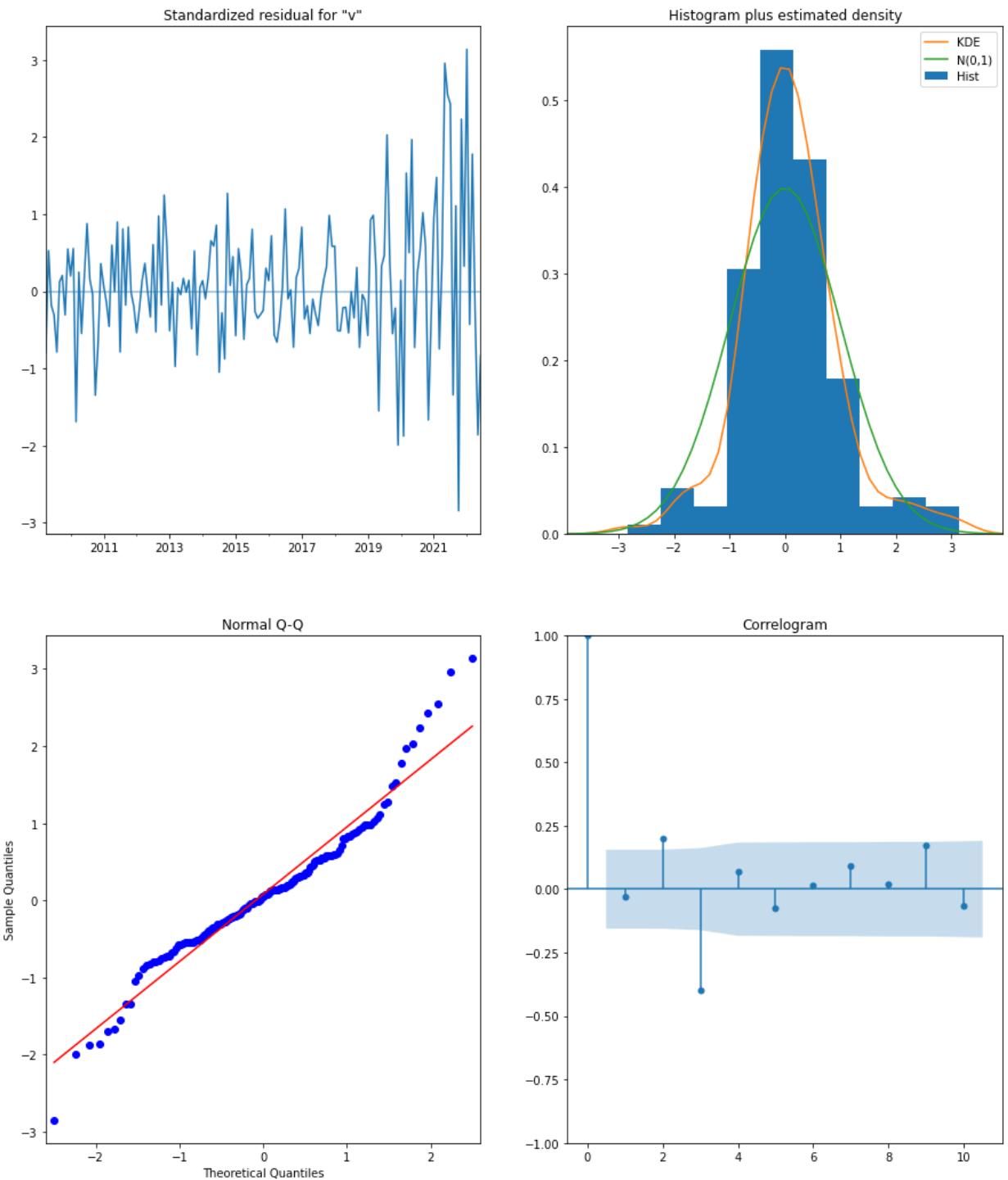
```
1 fisher = zip_df[7]
2 output_fisher = sarimax(fisher, order=(1, 1, 1), seasonal_order=(1, 1, 1, 1))

=====
```

	coef	std err	z	P> z	[0.025
0.975]					
ar.L1	0.8604	0.033	25.747	0.000	0.795
0.926					
ma.L1	0.1090	0.075	1.456	0.145	-0.038
0.256					
ar.S.L12	-0.6351	0.085	-7.473	0.000	-0.802
0.469					
ma.S.L12	0.0176	0.046	0.385	0.700	-0.072
0.108					
sigma2	2.488e+06	2.55e+05	9.751	0.000	1.99e+06
9e+06					2.9

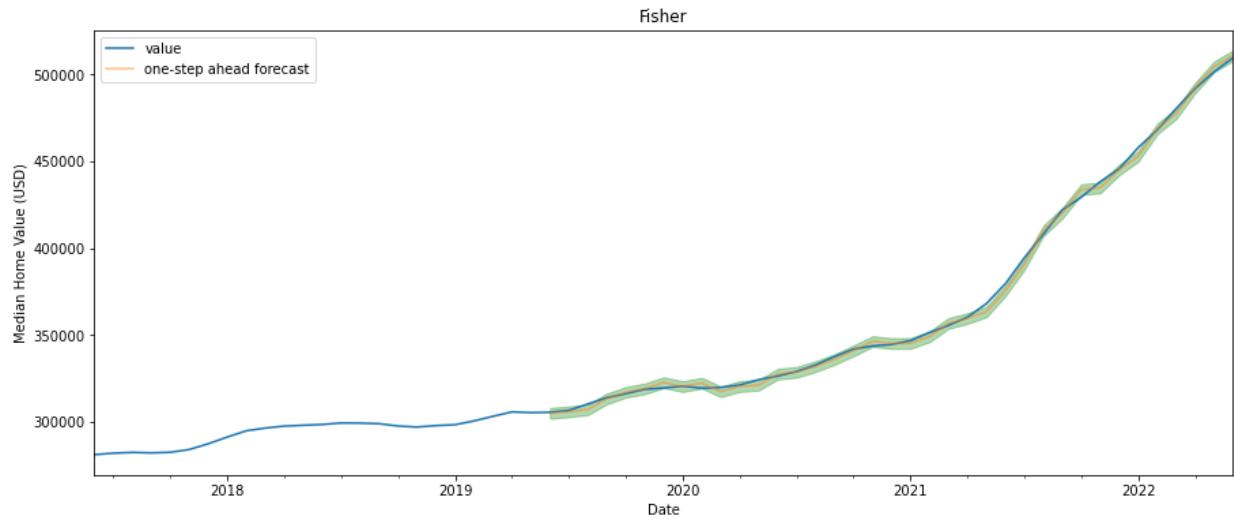
```
=====
```

=====



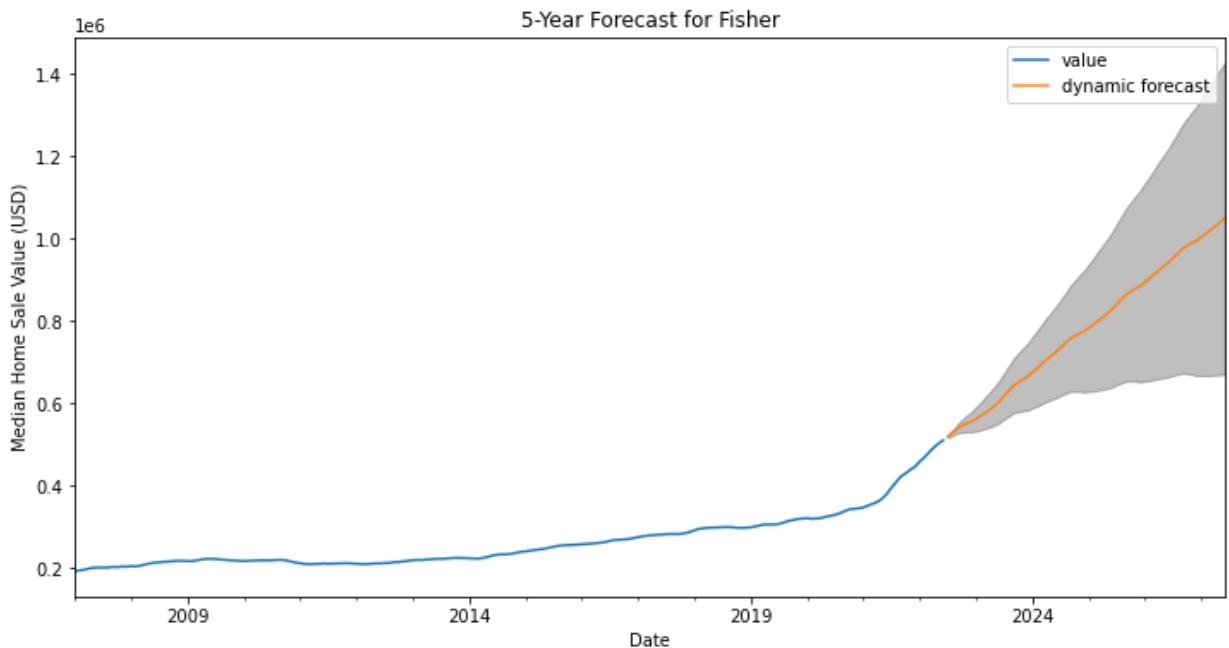
```
In [66]: 1 ose_forecast(fisher, output_fisher, neighborhood='Fisher')
```

The RMSE of our forecasts is 2358.43



In [67]:

1 dynamic_forecast(fisher, output_fisher, neighborhood='Fisher', years=5)



Out[67]: {'forecast': 1048416.0, 'minimum': 669827.0, 'maximum': 1427004.0}

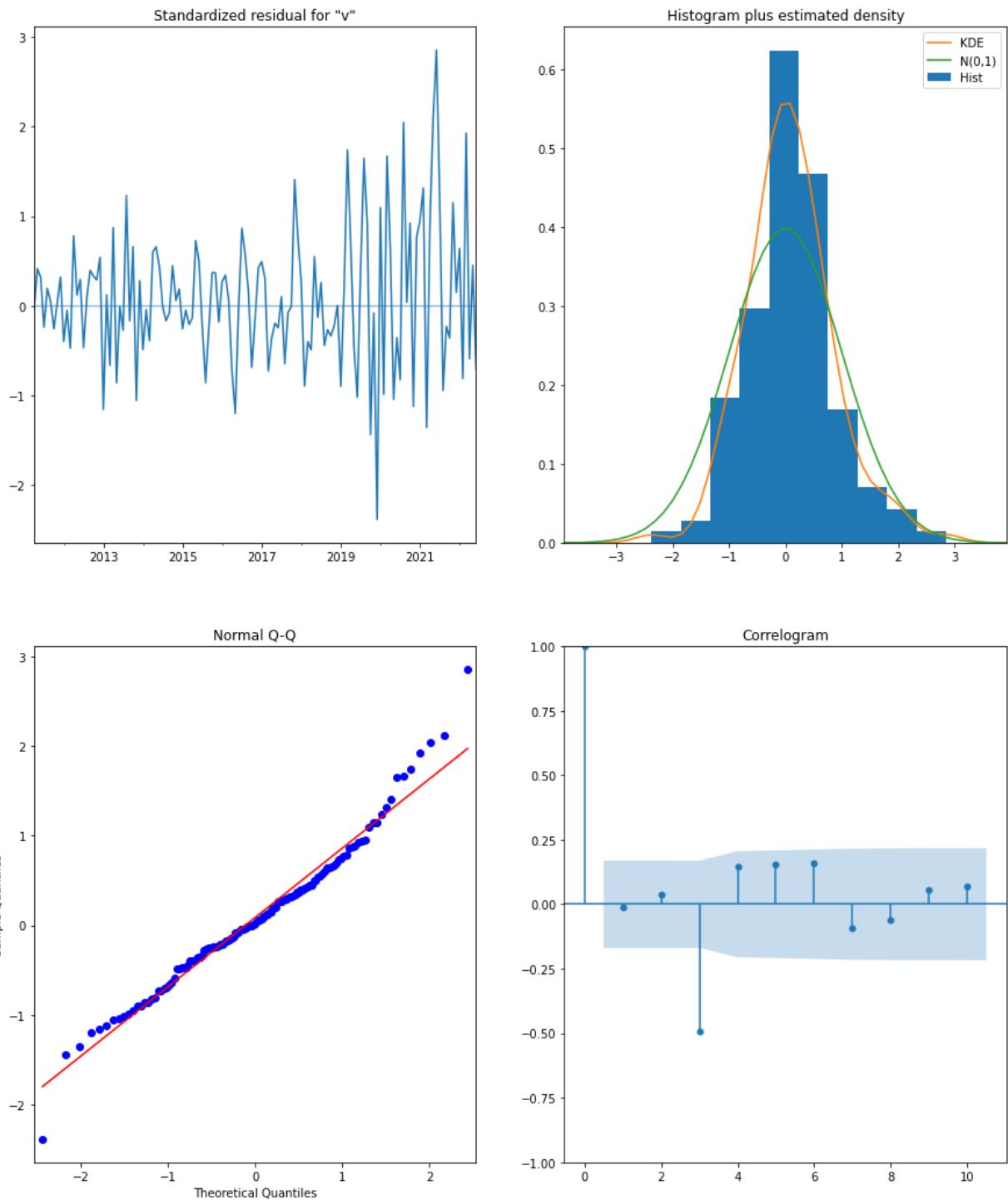
We can see the presence of heavy outliers. This is an area that is relatively new near Canyon Lake. There are many custom luxury homes in the area due to its location near the lake. The RMSE of our predictions is \$2358. The confidence interval for our forecasts is very large, which means there is a large amount of uncertainty. The median home value has shot up 150k in the past 2 years due to it becoming a highly desirable area for retirees. While the gains could be huge in this area, it is not likely to be the safest investment based on the very steep increase over the past 2 years which could taper off.

▼ 5.6.9 Canyon Lake

In [68]:

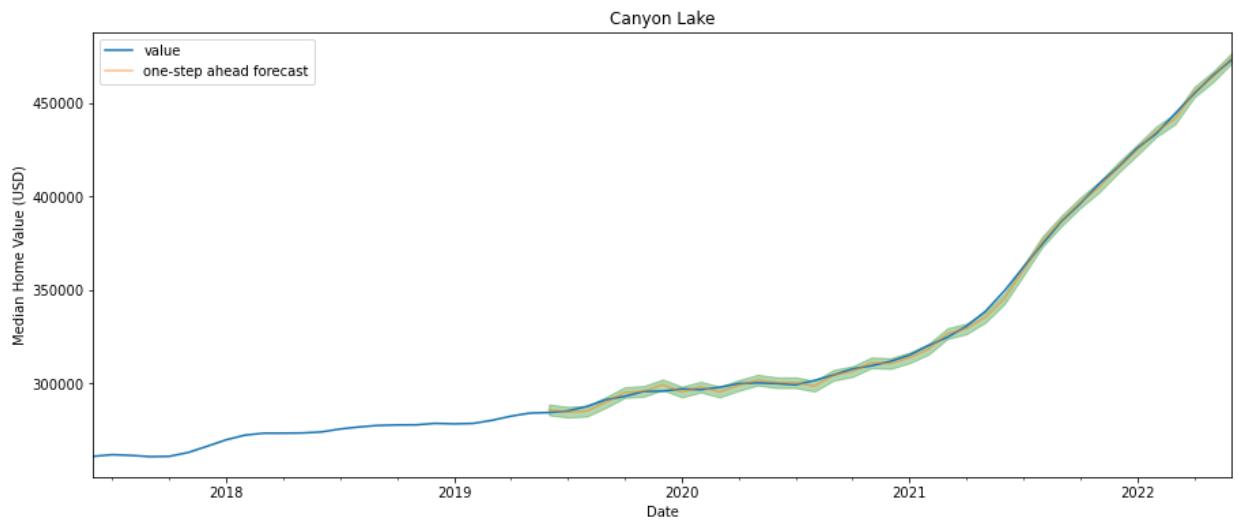
```
1 canyon_lake = zip_df[8]
2 output_canyon_lake = sarimax(canyon_lake, order=(1, 1, 1), seasonal_order=(0, 0, 0, 0))
=====
=====
```

	coef	std err	z	P> z	[0.025
0.975]					
-----	-----	-----	-----	-----	-----
ar.L1	0.9157	0.061	14.986	0.000	0.796
1.035					
ma.L1	0.1537	0.137	1.121	0.262	-0.115
0.422					
ar.S.L12	-0.7337	0.125	-5.855	0.000	-0.979
0.488					
ma.S.L12	-0.0034	0.067	-0.052	0.959	-0.134
0.127					
sigma2	2.152e+06	3.35e+05	6.429	0.000	1.5e+06
1e+06					2.8
=====	=====	=====	=====	=====	=====
=====	=====	=====	=====	=====	=====

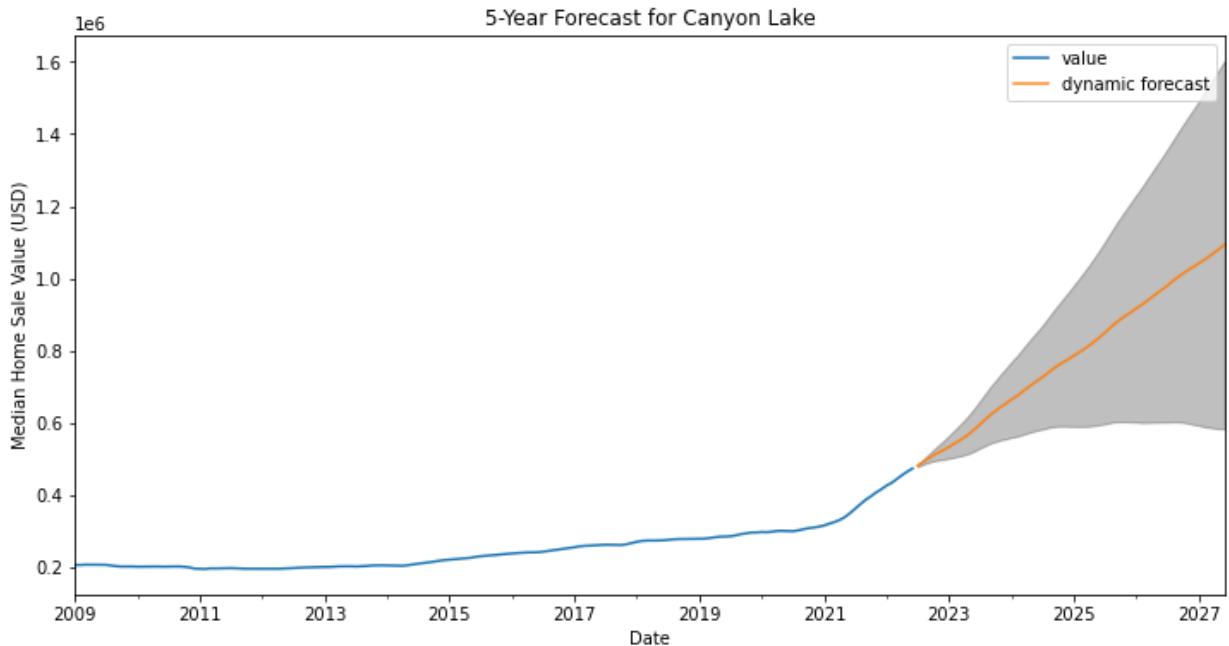


```
In [69]: 1 close_forecast(canyon_lake, output_canyon_lake, neighborhood='Canyon Lake')
```

The RMSE of our forecasts is 1773.47



```
In [70]: 1 dynamic_forecast(canyon_lake, output_canyon_lake, neighborhood='Canyon Lake')
```



```
Out[70]: {'forecast': 1093740.0, 'minimum': 582966.0, 'maximum': 1604515.0}
```

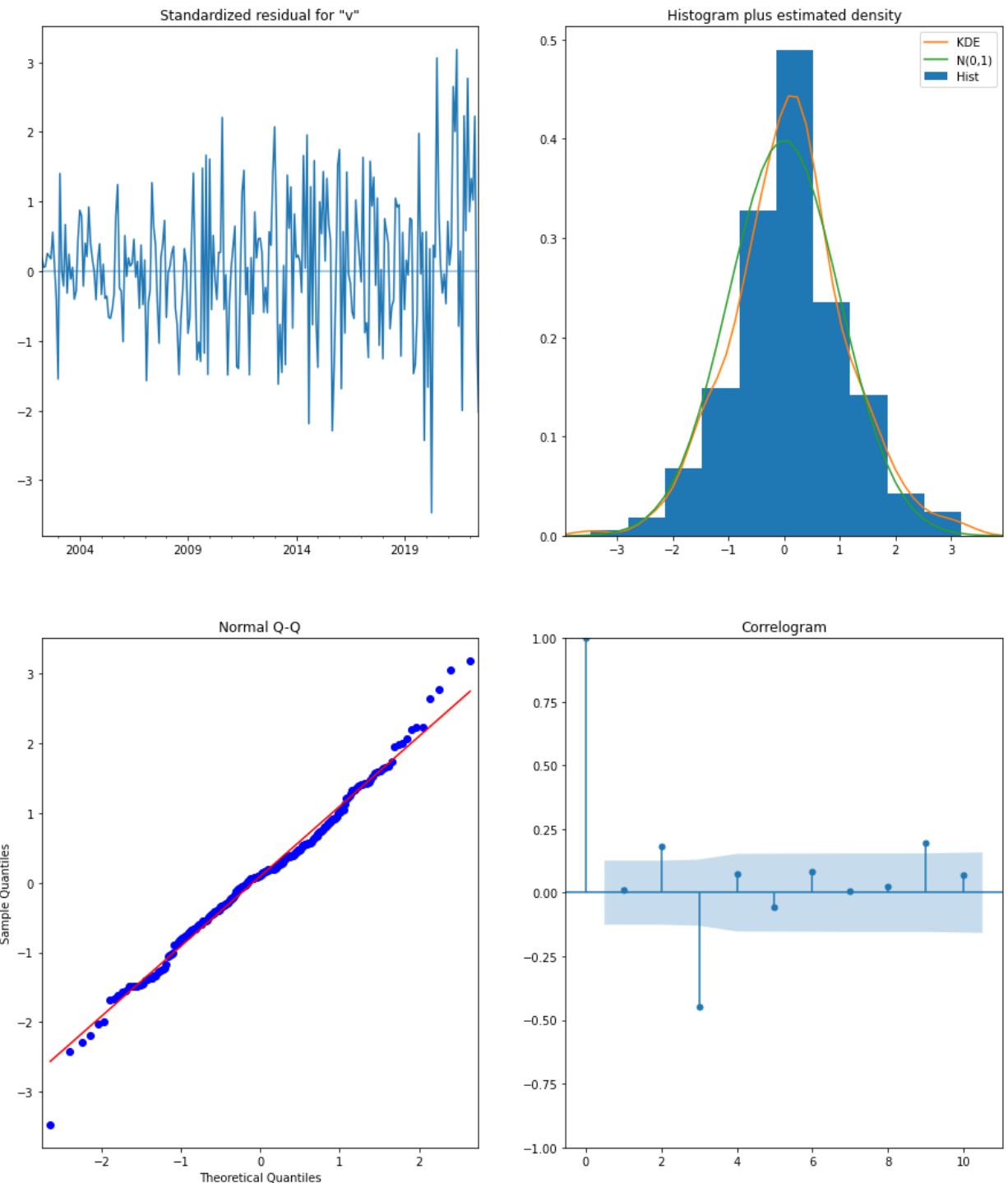
We can see the same thing happening in Canyon Lake, the median value of homes has shot up 150k in the past year, and Canyon Lake also has a high entry point into the market right now. Due to the high increases in the past year, the confidence interval is very large for Canyon Lake and this would be a riskier investment. There is a spread of 1 million dollars in our confidence interval!

▼ **5.6.10 Eastside**

In [71]:

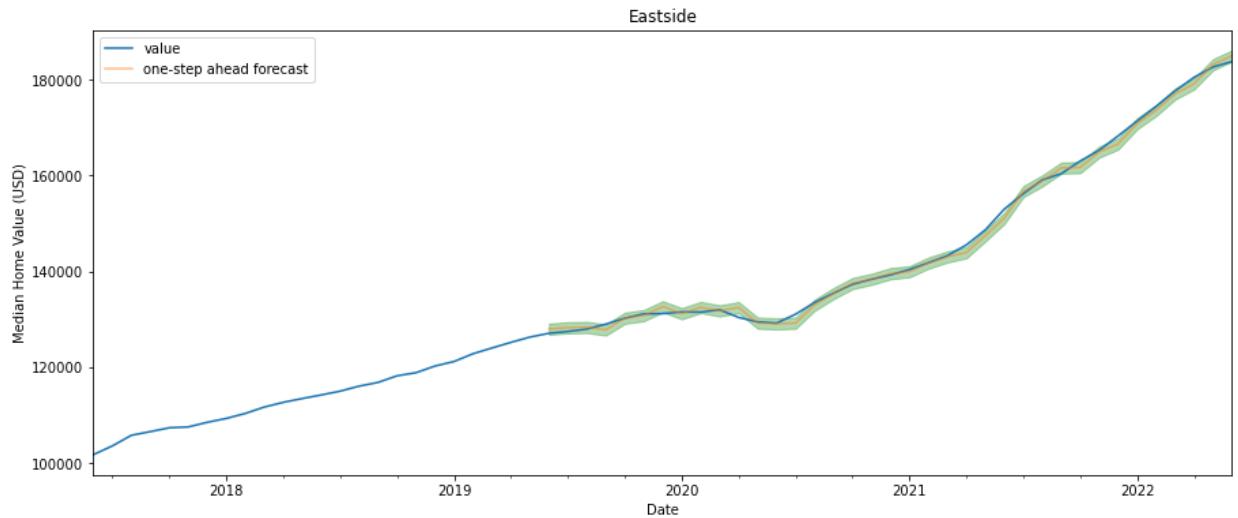
```
1 eastside = zip_df[9]
2 output_eastside = sarimax(eastside, order=(1, 1, 1), seasonal_order=(0, 0, 0, 0))
=====
=====
```

	coef	std err	z	P> z	[0.025
0.975]					
-----	-----	-----	-----	-----	-----
ar.L1	0.6822	0.051	13.399	0.000	0.582
0.782					
ma.L1	0.0998	0.078	1.284	0.199	-0.053
0.252					
ar.S.L12	-0.5868	0.054	-10.903	0.000	-0.692
0.481					
ma.S.L12	-0.1070	0.034	-3.151	0.002	-0.174
0.040					
sigma2	3.593e+05	2.94e+04	12.241	0.000	3.02e+05
7e+05					4.1
=====	=====	=====	=====	=====	=====
=====	=====	=====	=====	=====	=====

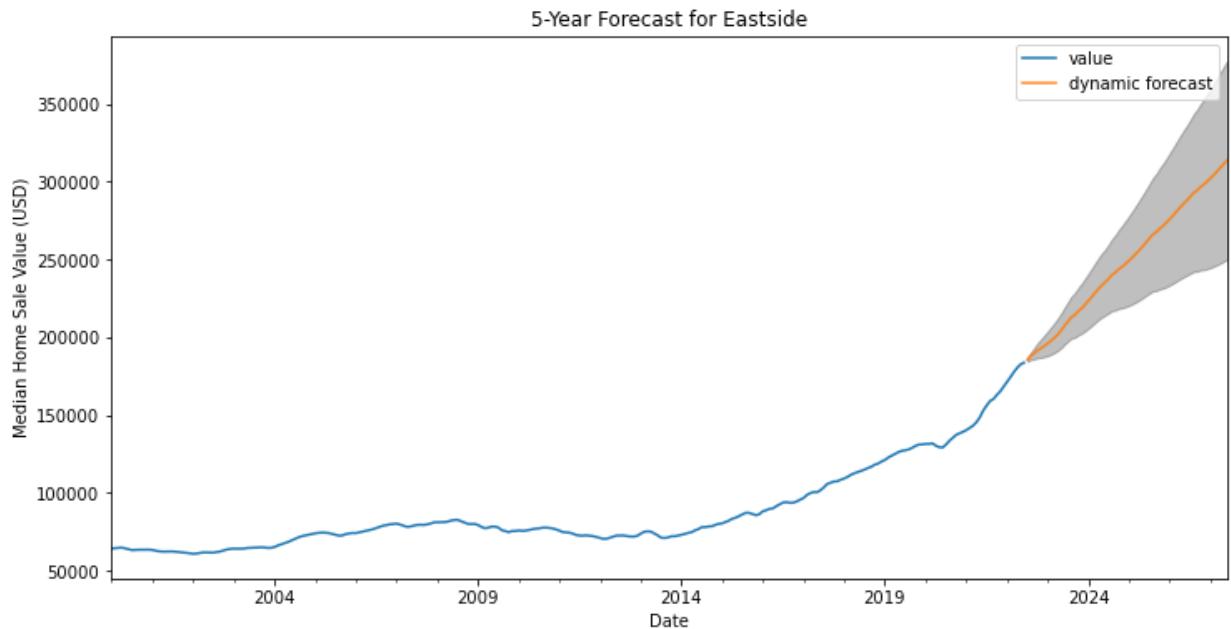


```
In [72]: 1 ose_forecast(eastside, output_eastside, neighborhood='Eastside')
```

The RMSE of our forecasts is 951.41



```
In [73]: 1 dynamic_forecast(eastside, output_eastside, neighborhood='Eastside', )
```



```
Out[73]: {'forecast': 313952.0, 'minimum': 249978.0, 'maximum': 377926.0}
```

The RMSE of our predictions for the Eastside is \$951, with a relatively small confidence interval for our forecasts. The entry point into the neighborhood is low at around 180k, with our 5 year forecast predicting to increase to 313k. The residuals are normal with some outliers on the tails.

▼ 6 Interpret Results

▼ 6.1 Projections for 3 and 5 year ROI

I am now going to make a dataframe that will contain our projected metrics based upon our sarimax model forecasting.

```
In [74]:  
  1 def projections(ts, zipcodes, years):  
  2     '''  
  3         Calculate projections for each zip code.  
  4             - Current Value  
  5             - ROI (return on investment)  
  6     '''  
  7     Input:  
  8         ts : time series data.  
  9         zipcodes : list of zip codes to iterate through.  
 10        years : n-number of years to forecast.  
 11  
 12     Output:  
 13         Returns dataframe with:  
 14             - zip code  
 15             - investment value (current value)  
 16             - predicted value and errors  
 17             - predicted ROI and errors  
 18  
 19     '''  
 20  
 21     # Define steps for forecasting  
 22     steps = years*12  
 23  
 24     # Create list to append projection metrics  
 25     projections = []  
 26  
 27     # Iterate through time series of each zip code  
 28     for zipcode in zipcodes:  
 29  
 30         # Time series of zip code  
 31         zip_ts = ts[ts['RegionName'] == zipcode]  
 32         to_drop = ['RegionName']  
 33         zip_ts = zip_ts.drop(to_drop, axis=1)  
 34  
 35         # Plug the optimal parameter values into a new SARIMAX model  
 36         # I used the values that were the most frequent for out top 10  
 37     SARIMAX = sm.tsa.statespace.SARIMAX(zip_ts,  
 38                                         order=(1, 1, 1),  
 39                                         seasonal_order=(1, 1, 1, 1),  
 40                                         enforce_stationarity=False,  
 41                                         enforce_invertibility=False)  
 42  
 43         # Fit the model and print results  
 44         output = SARIMAX.fit()  
 45  
 46         # Get forecast and confidence interval for steps ahead in future  
 47         forecast = output.get_forecast(steps=steps, dynamic=True)  
 48         conf_int = forecast.conf_int()  
 49  
 50         # Show prediction for end of step-period  
 51         pred_mean = forecast.predicted_mean[-1]  
 52         pred_mean_min = conf_int.iloc[-1, 0]  
 53         pred_mean_max = conf_int.iloc[-1, 1]  
 54  
 55         # Last value of the time series - this will be used to calculate  
 56         current_value = zip_ts['value'][-1]
```

```

57
58     # ROI (Return on Investment) as Percentage
59     ROI = ((pred_mean-current_value)/current_value *100)
60     ROI_min = ((pred_mean_min-current_value)/current_value *100)
61     ROI_max = ((pred_mean_max-current_value)/current_value *100)
62
63     # Create DataFrame
64
65     # Initiate empty dictionary
66     predicted = {}
67
68     # Columns for df
69     predicted['Zip Code'] = zipcode
70     predicted['Investment Value ($)'] = current_value.round(2)
71     predicted['Predicted Sale Price ($)'] = pred_mean.round(2)
72     predicted['Min Pred Sale Price ($)'] = pred_mean_min.round(2)
73     predicted['Max Pred Sale Price ($)'] = pred_mean_max.round(2)
74     predicted['Predicted ROI (%)'] = ROI.round()
75     predicted['Min Pred ROI (%)'] = ROI_min.round()
76     predicted['Max Pred ROI (%)'] = ROI_max.round()
77
78     # Append to list
79     projections.append(predicted)
80
81     # Return in DataFrame format
82     return pd.DataFrame.from_dict(projections)

```

In [75]: ▾

```

1 # Get 3 year projections for top 10 zip codes
2 three_year_projections = projections(model_df, zip_list, years=3)
3 three_year_projections

```

Out[75]:

	Zip Code	Investment Value (\$)	Predicted Sale Price (\$)	Min Pred Sale Price (\$)	Max Pred Sale Price (\$)	Predicted ROI (%)	Min Pred ROI (%)	Max Pred ROI (%)
0	78203	241568.0	339839.66	283808.21	395871.11	41.0	17.0	64.0
1	78202	228219.0	316210.20	277737.86	354682.55	39.0	22.0	55.0
2	78207	148461.0	217501.24	180366.22	254636.25	47.0	21.0	72.0
3	78638	555727.0	930184.77	673410.44	1186959.09	67.0	21.0	114.0
4	78055	361314.0	569344.07	465645.71	673042.44	58.0	29.0	86.0
5	78210	234712.0	337287.14	295798.77	378775.51	44.0	26.0	61.0
6	78237	163415.0	237227.36	199425.67	275029.05	45.0	22.0	68.0
7	78623	509544.0	827218.41	638932.68	1015504.15	62.0	25.0	99.0
8	78133	473175.0	837285.82	594395.25	1080176.39	77.0	26.0	128.0
9	78220	183720.0	260936.10	226854.29	295017.91	42.0	23.0	61.0

```
In [76]: 1 # Make column for neighborhood name
          2 # Top 10 zip code names
          3 top_10_names = ['Arena District', 'Dignowity Hill', 'Gardendale', 'Kingsbury',
          4                      'Denver Heights', 'Inner Westside', 'Fisher', 'Canyon Lake']
          5 three_year_projections['Neighborhood'] = top_10_names
          6 three_year_projections = three_year_projections.reindex(columns=[ 'Neighborhood', 'Predicted Sale Price ($)', 'Min Pred Sale Price ($)', 'Max Pred Sale Price ($)', 'Predicted ROI (%)', 'Min Pred ROI (%)', 'Max Pred ROI (%)'])
          7
          8
          9
          10 three_year_projections.set_index('Neighborhood')
```

Out[76]:

	Zip Code	Investment Value (\$)	Predicted Sale Price (\$)	Min Pred Sale Price (\$)	Max Pred Sale Price (\$)	Predicted ROI (%)	Min Pred ROI (%)	Max Pred ROI (%)
Neighborhood								
Arena District	78203	241568.0	339839.66	283808.21	395871.11	41.0	17.0	64.0
Dignowity Hill	78202	228219.0	316210.20	277737.86	354682.55	39.0	22.0	55.0
Gardendale	78207	148461.0	217501.24	180366.22	254636.25	47.0	21.0	72.0
Kingsbury	78638	555727.0	930184.77	673410.44	1186959.09	67.0	21.0	114.0
Medina	78055	361314.0	569344.07	465645.71	673042.44	58.0	29.0	86.0
Denver Heights	78210	234712.0	337287.14	295798.77	378775.51	44.0	26.0	61.0
Inner Westside	78237	163415.0	237227.36	199425.67	275029.05	45.0	22.0	68.0
Fisher	78623	509544.0	827218.41	638932.68	1015504.15	62.0	25.0	99.0
Canyon Lake	78133	473175.0	837285.82	594395.25	1080176.39	77.0	26.0	128.0
Eastside	78220	183720.0	260936.10	226854.29	295017.91	42.0	23.0	61.0

```
In [77]: ▾ 1 # Get 5 year projections for top 10 zip codes
  2 five_year_projections = projections(model_df, zip_list, years=5)
  3 five_year_projections
```

Out[77]:

	Zip Code	Investment Value (\$)	Predicted Sale Price (\$)	Min Pred Sale Price (\$)	Max Pred Sale Price (\$)	Predicted ROI (%)	Min Pred ROI (%)	Max Pred ROI (%)
0	78203	241568.0	405675.97	301026.60	510325.34	68.0	25.0	111.0
1	78202	228219.0	374877.90	303162.47	446593.33	64.0	33.0	96.0
2	78207	148461.0	264755.34	195127.43	334383.26	78.0	31.0	125.0
3	78638	555727.0	1187523.76	698988.57	1676058.95	114.0	26.0	202.0
4	78055	361314.0	708782.46	507903.17	909661.75	96.0	41.0	152.0
5	78210	234712.0	406120.38	327701.41	484539.36	73.0	40.0	106.0
6	78237	163415.0	287589.35	215798.43	359380.27	76.0	32.0	120.0
7	78623	509544.0	1048415.61	669826.72	1427004.49	106.0	31.0	180.0
8	78133	473175.0	1093740.35	582965.84	1604514.85	131.0	23.0	239.0
9	78220	183720.0	313951.99	249978.26	377925.71	71.0	36.0	106.0

```
In [78]: 
  1 # Set neighborhood names for five year projections df
  2 five_year_projections['Neighborhood'] = top_10_names
  3 five_year_projections = five_year_projections.reindex(columns=[ 'Neighbo
  4 'Predicted Sale Price (%)'
  5 'Max Predicted Sale Price ($)'
  6 'Min Predicted Sale Price ($)
  7 five_year_projections.set_index('Neighborhood')
```

Out[78]:

	Zip Code	Investment Value (\$)	Predicted Sale Price (\$)	Min Pred Sale Price (\$)	Max Pred Sale Price (\$)	Predicted ROI (%)	Min Pred ROI (%)	Max Pred ROI (%)	
Neighborhood									
Arena District	78203	241568.0	405675.97	301026.60	510325.34	68.0	25.0	111.0	
Dignowity Hill	78202	228219.0	374877.90	303162.47	446593.33	64.0	33.0	96.0	
Gardendale	78207	148461.0	264755.34	195127.43	334383.26	78.0	31.0	125.0	
Kingsbury	78638	555727.0	1187523.76	698988.57	1676058.95	114.0	26.0	202.0	
Medina	78055	361314.0	708782.46	507903.17	909661.75	96.0	41.0	152.0	
Denver Heights	78210	234712.0	406120.38	327701.41	484539.36	73.0	40.0	106.0	
Inner Westside	78237	163415.0	287589.35	215798.43	359380.27	76.0	32.0	120.0	
Fisher	78623	509544.0	1048415.61	669826.72	1427004.49	106.0	31.0	180.0	
Canyon Lake	78133	473175.0	1093740.35	582965.84	1604514.85	131.0	23.0	239.0	
Eastside	78220	183720.0	313951.99	249978.26	377925.71	71.0	36.0	106.0	

In [79]: ▾ 1 # Sort values based on ROI for 3 year projections
2 three_year_projections.sort_values('Predicted ROI (%)')

Out[79]:

	Neighborhood	Zip Code	Investment Value (\$)	Predicted Sale Price (\$)	Min Pred Sale Price (\$)	Max Pred Sale Price (\$)	Predicted ROI (%)	Min Pred ROI (%)	Max Pred ROI (%)
1	Dignowity Hill	78202	228219.0	316210.20	277737.86	354682.55	39.0	22.0	55.0
0	Arena District	78203	241568.0	339839.66	283808.21	395871.11	41.0	17.0	64.0
9	Eastside	78220	183720.0	260936.10	226854.29	295017.91	42.0	23.0	61.0
5	Denver Heights	78210	234712.0	337287.14	295798.77	378775.51	44.0	26.0	61.0
6	Inner Westside	78237	163415.0	237227.36	199425.67	275029.05	45.0	22.0	68.0
2	Gardendale	78207	148461.0	217501.24	180366.22	254636.25	47.0	21.0	72.0
4	Medina	78055	361314.0	569344.07	465645.71	673042.44	58.0	29.0	86.0
7	Fisher	78623	509544.0	827218.41	638932.68	1015504.15	62.0	25.0	99.0
3	Kingsbury	78638	555727.0	930184.77	673410.44	1186959.09	67.0	21.0	114.0
8	Canyon Lake	78133	473175.0	837285.82	594395.25	1080176.39	77.0	26.0	128.0

In [80]: ▾ 1 # Sort values based on ROI for 5 year projections
2 five_year_projections.sort_values('Predicted ROI (%)')

Out[80]:

	Neighborhood	Zip Code	Investment Value (\$)	Predicted Sale Price (\$)	Min Pred Sale Price (\$)	Max Pred Sale Price (\$)	Predicted ROI (%)	Min Pred ROI (%)	Max Pred ROI (%)
1	Dignowity Hill	78202	228219.0	374877.90	303162.47	446593.33	64.0	33.0	96.0
0	Arena District	78203	241568.0	405675.97	301026.60	510325.34	68.0	25.0	111.0
9	Eastside	78220	183720.0	313951.99	249978.26	377925.71	71.0	36.0	106.0
5	Denver Heights	78210	234712.0	406120.38	327701.41	484539.36	73.0	40.0	106.0
6	Inner Westside	78237	163415.0	287589.35	215798.43	359380.27	76.0	32.0	120.0
2	Gardendale	78207	148461.0	264755.34	195127.43	334383.26	78.0	31.0	125.0
4	Medina	78055	361314.0	708782.46	507903.17	909661.75	96.0	41.0	152.0
7	Fisher	78623	509544.0	1048415.61	669826.72	1427004.49	106.0	31.0	180.0
3	Kingsbury	78638	555727.0	1187523.76	698988.57	1676058.95	114.0	26.0	202.0
8	Canyon Lake	78133	473175.0	1093740.35	582965.84	1604514.85	131.0	23.0	239.0

Our projections for 3 years and 5 years have the same ranking order of each zip code which is great. In order to aid visualization, I am going to write a function to produce a bar plot with predicted ROI and errors associated with each zip code.

6.2 Barplots of ROI Projections and Errors

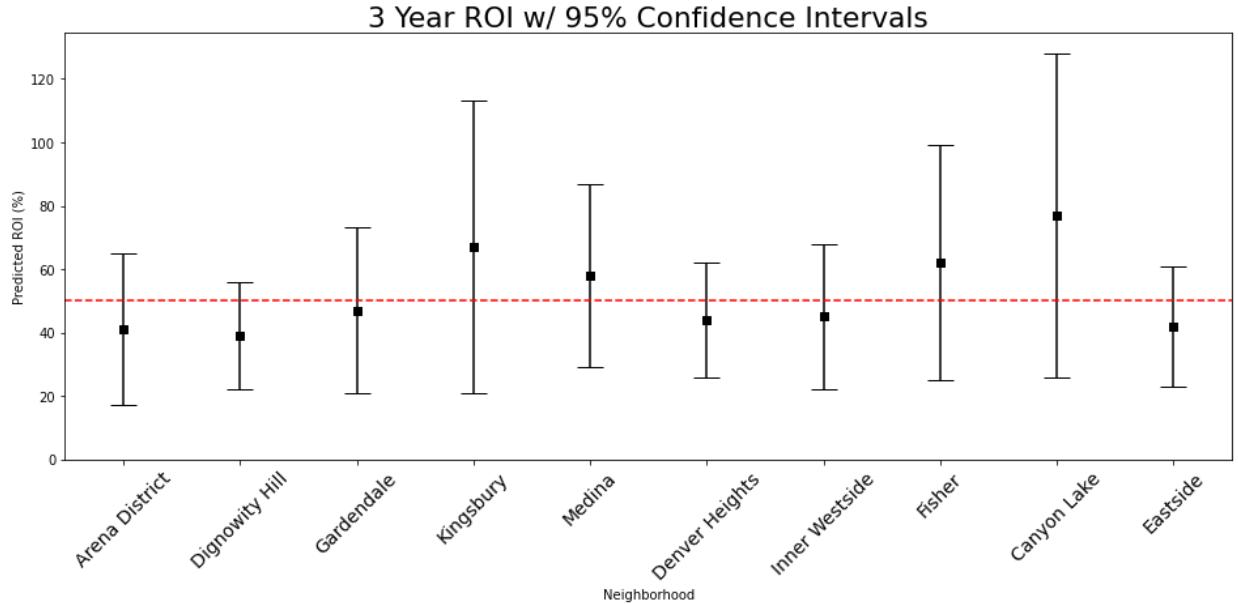
```
In [81]:  
  1 # Barplot for x years projection with 95% confidence intervals  
  2 def projection_barplot(df, years=3, save_fig=False):  
  3     # Plot Predicted Mean Home Sale Values #  
  4  
  5     # Get errors; (pred mean - lower bound of conf interval)  
  6     errors = df['Predicted ROI (%)'] - df['Min Pred ROI (%)']  
  7     df['errors'] = errors  
  8  
  9     # Define figure, axes, and plot  
10    fig, ax = plt.subplots(figsize=(16, 6))  
11  
12    # Plot barplot of ROI with errors  
13    plt.rcParams.update({'font.size': 18})  
14    df.plot(x='Neighborhood',  
15              y='Predicted ROI (%)',  
16              kind='bar',  
17              ax=ax,  
18              color='none',  
19              ecolor='black',  
20              capszie=10,  
21              yerr='errors',  
22              legend=False)  
23    #Labels  
24    ax.set_xlabel('Neighborhood')  
25    ax.set_ylabel('Predicted ROI (%)')  
26    plt.title(f'{years} Year ROI w/ 95% Confidence Intervals')  
27  
28    # Predicted ROI  
29    ax.scatter(x=pd.np.arange(df.shape[0]),  
30                 marker='s',  
31                 s=40,  
32                 y=df['Predicted ROI (%)'],  
33                 color='black')  
34  
35    # Line to help visualize on the y-axis  
36    ax.axhline(y=50, linestyle='--', color='red', linewidth=1.5)  
37  
38    if save_fig:  
39        fig.savefig(f'./images/barplot_{years}.jpeg')  
40  
41    # Rotate zip code labels 45 degrees  
42    plt.xticks(rotation = 45, size = 14)  
43    plt.show();
```

In [82]:

```

1 # 3 year ROI projections
2 projection_barplot(three_year_projections, years=3)

```

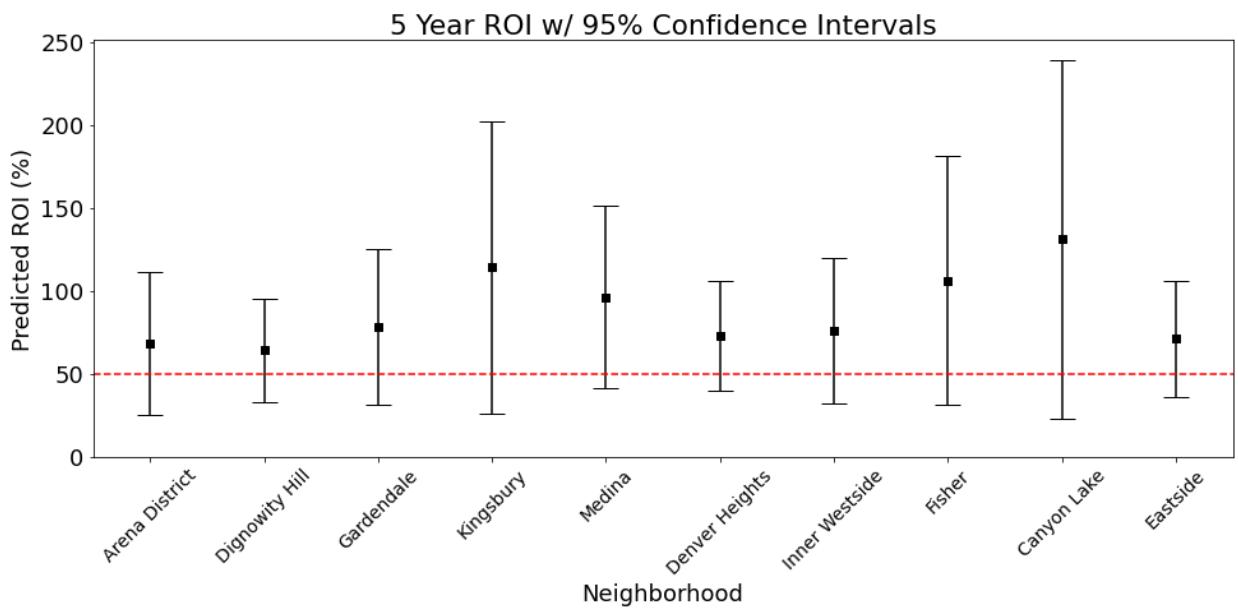


In [83]:

```

1 # 5 year ROI projections
2 projection_barplot(five_year_projections, years=5)

```



Since my client very concerned with low risk investments, I am going to select the top 5 zip codes from this list with the smallest confidence intervals. This will remove those zip codes that have large

variability and more fluctuation in the market values. These zip codes also happen to be the ones with the lowest ROI, but it appears it is hard to go wrong with any zip code in this metro area right now with its massive growth with no end in sight.

▼ 6.3 Top 5 Zip Codes

```
In [84]: ▾ 1 # Top 5 zip codes
  ▾ 2 top_zip_projections = three_year_projections.sort_values('Predicted ROI', ascending=False).head(5)
  ▾ 3
  ▾ 4 top_zip_projections
```

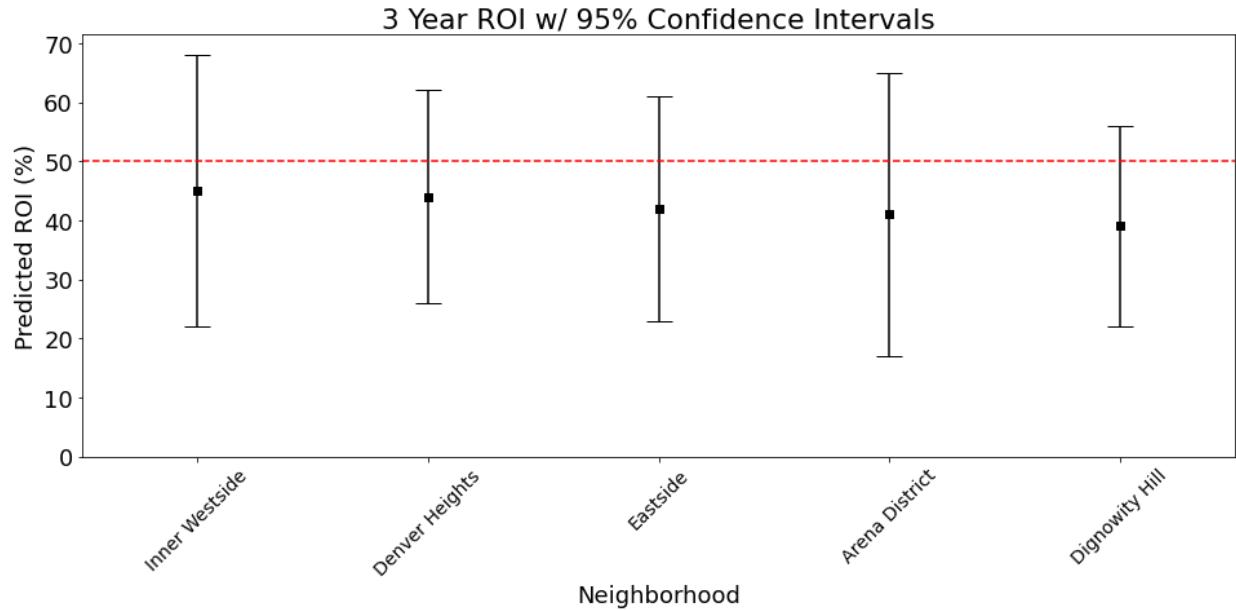
Out[84]:

	Neighborhood	Zip Code	Investment Value (\$)	Predicted Sale Price (\$)	Min Pred Sale Price (\$)	Max Pred Sale Price (\$)	Predicted ROI (%)	Min Pred ROI (%)	Max Pred ROI (%)	error
6	Inner Westside	78237	163415.0	237227.36	199425.67	275029.05	45.0	22.0	68.0	23.
5	Denver Heights	78210	234712.0	337287.14	295798.77	378775.51	44.0	26.0	61.0	18.
9	Eastside	78220	183720.0	260936.10	226854.29	295017.91	42.0	23.0	61.0	19.
0	Arena District	78203	241568.0	339839.66	283808.21	395871.11	41.0	17.0	64.0	24.
1	Dignowity Hill	78202	228219.0	316210.20	277737.86	354682.55	39.0	22.0	55.0	17.

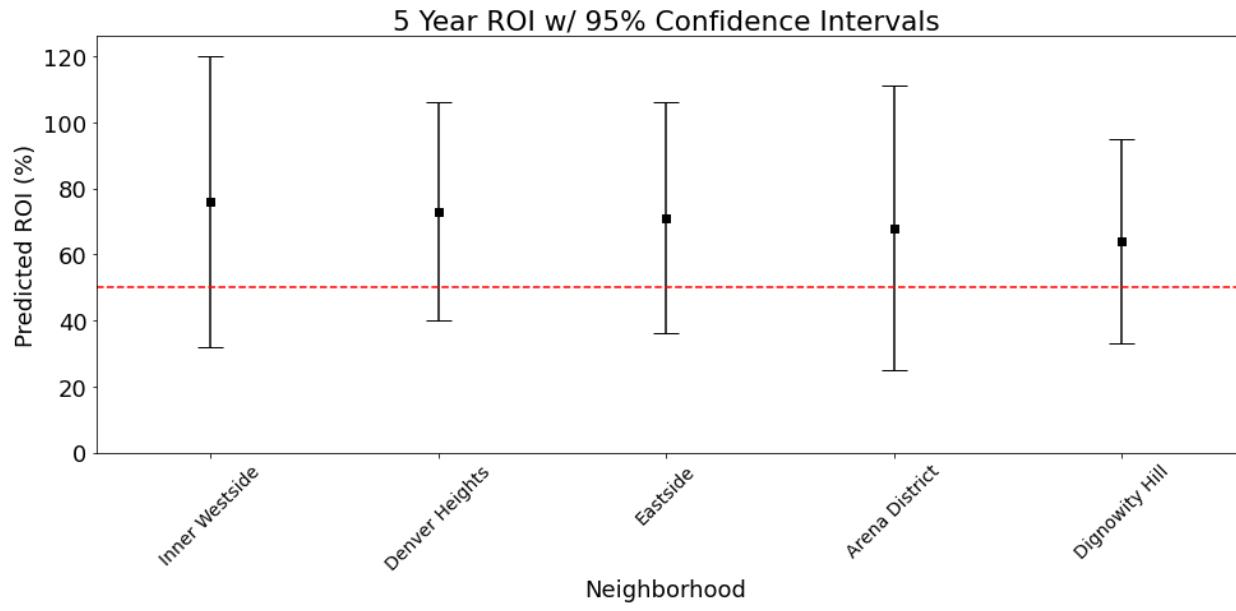
Our top 5 zip codes are 78237 - Inner Westside, 78210 - Denver Heights, 78220 - Eastside, 78203 - Arena District, and 78202 - Dignowity Hill. All 5 of these zip codes boast projected returns of 39% to 45%, and have great entry point prices ranging from 163k to 241k, far lower than the median value of the city.

78237 - Inner Westside, has the highest projected 3 year ROI and the lowest entry point at 163k, while 78202 - Dignowity Hill has the smallest variation and is therefore the safest investment, but also has a lower projected ROI.

```
In [85]: ▾ 1 # Barplot of our 3 year projected ROI for top zip codes
  2 projection_barplot(top_zip_projections, years=3, save_fig=True)
```



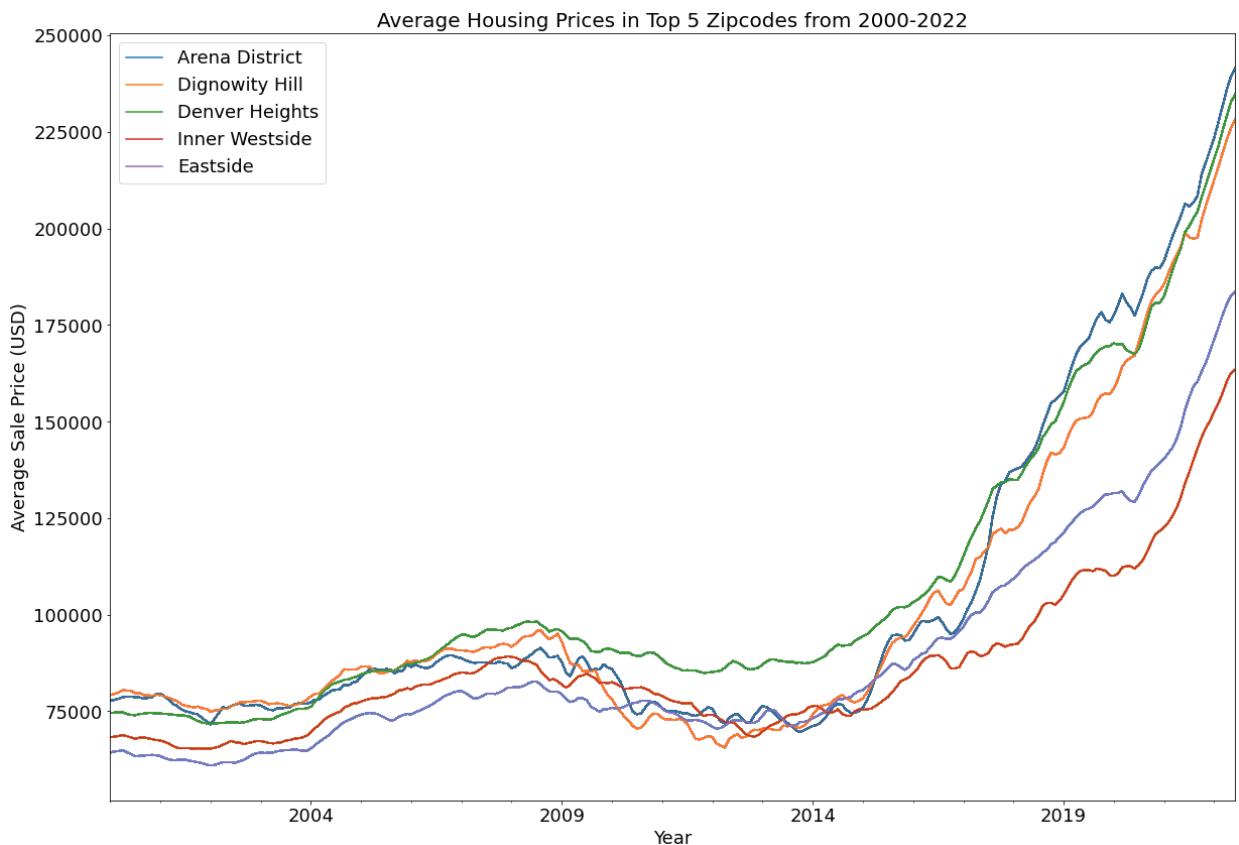
```
In [86]: ▾ 1 # Barplot of our 5 year projected ROI for our top zip codes
  2 top_5year_projections = five_year_projections.sort_values('Predicted I
  3
  4 # Plot 5 year projections barplot
  5 projection_barplot(top_5year_projections, years=5, save_fig=True)
```



Our top zip codes have projected 5 year ROI of 64% to 76%! Now lets visualize the average sale prices or our top 5 zip codes over 22 years.

▼ 6.4 Average Housing Prices of our Top 5 Zip Codes from 2000 - 2022

```
In [87]: 1 # Top 5 zip codes
2 top_5_zipcodes = [78203, 78202, 78210, 78237, 78220]
3
4 #plot the values grouped by zip code
5 plt.figure(figsize=(20,14))
6 for zipcode in model_df.RegionName.unique():
7     for zipcode in top_5_zipcodes:
8         model_df.loc[model_df['RegionName'] == zipcode].value.plot(la
9
10 plt.legend(labels=('Arena District', 'Dignowity Hill', 'Denver Heights'
11             'Inner Westside', 'Eastside'), loc='upper left')
12
13 plt.xlabel("Year", fontsize = 18)
14 plt.ylabel("Average Sale Price (USD)", fontsize=18)
15 plt.title('Average Housing Prices in Top 5 Zipcodes from 2000-2022',
16 # Save fig
17 plt.savefig('./images/top_5_over_time.jpeg')
18 plt.show();
```



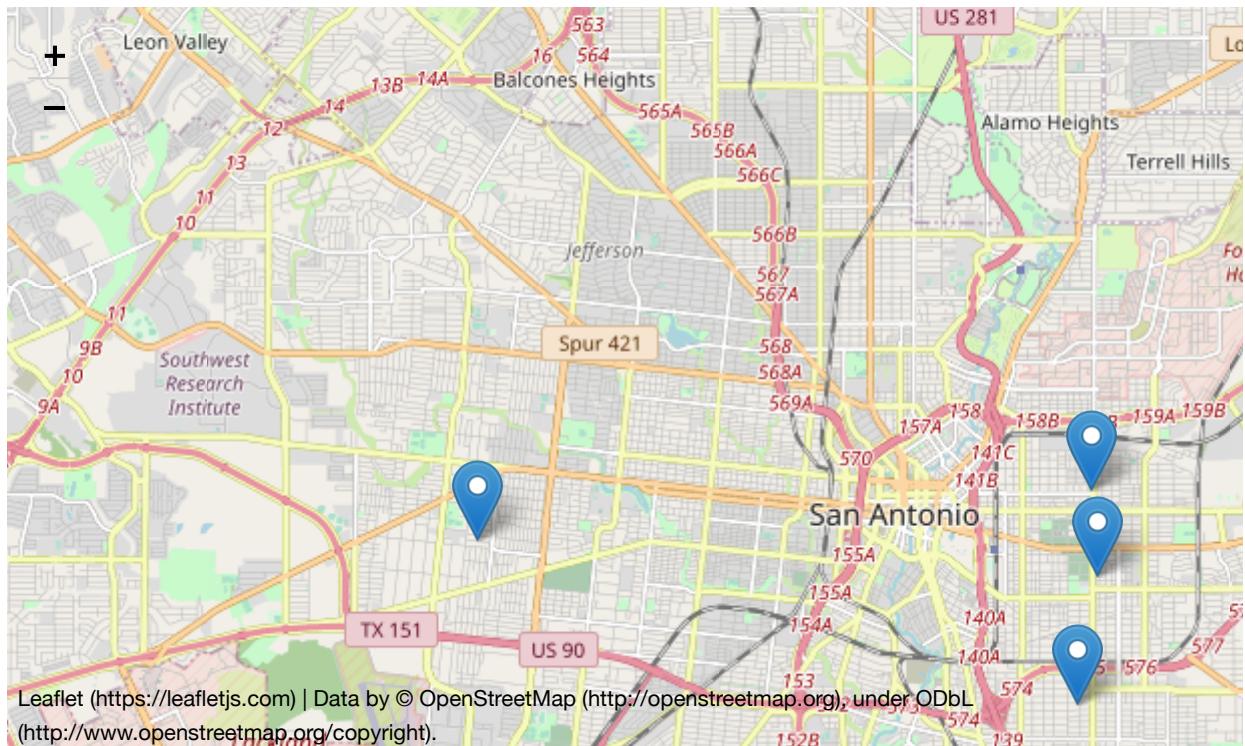
We can see a similar trend among all of our top 5 zip codes, although a bit more pronounced in the Arena District, Dignowity Hill, and Denver Heights. This is likely due to the gentrification that has been taking place in these neighborhoods over the past several years and becoming desirable hotspots. Unfortunately, the crime rate is high in all of these neighborhoods and the schools all have below average ratings. It will likely take the schools several years to improve with the transition.

6.5 Map Visualization of Top 5 Zip Codes

In [88]:

```
1 #Import
2 import folium
3
4 #Create map
5 this_map = folium.Map(location = [29.5, -98.5])
6
7 # Add markers for zip codes
8 folium.Marker(location=[29.4151, -98.4605], tooltip = 'Arena District')
9 folium.Marker(location=[29.4278, -98.4615], tooltip = 'Dignowity Hill')
10 folium.Marker(location=[29.3958, -98.4638], tooltip = 'Denver Heights')
11 folium.Marker(location=[29.4204, -98.5671], tooltip = 'Inner Westside')
12 folium.Marker(location=[29.4139, -98.3889], tooltip = 'Eastside').add_
13
14 #Set the zoom to fit our bounds
15 this_map.fit_bounds(this_map.get_bounds())
16
17 this_map
```

Out[88]:



In [89]: 1 top_zip_projections

Out[89]:

	Neighborhood	Zip Code	Investment Value (\$)	Predicted Sale Price (\$)	Min Pred Sale Price (\$)	Max Pred Sale Price (\$)	Predicted ROI (%)	Min Pred ROI (%)	Max Pred ROI (%)	error
6	Inner Westside	78237	163415.0	237227.36	199425.67	275029.05	45.0	22.0	68.0	23.
5	Denver Heights	78210	234712.0	337287.14	295798.77	378775.51	44.0	26.0	61.0	18.
9	Eastside	78220	183720.0	260936.10	226854.29	295017.91	42.0	23.0	61.0	19.
0	Arena District	78203	241568.0	339839.66	283808.21	395871.11	41.0	17.0	64.0	24.
1	Dignowity Hill	78202	228219.0	316210.20	277737.86	354682.55	39.0	22.0	55.0	17.

▼ 7 Recommendations

The top 5 zip codes ranked by highest return on investment (ROI) with the least amount of unpredictability and variation are:

- 78237: Inner Westside
- 78210: Denver Heights
- 78220: Eastside
- 78203: Arena District
- 78202: Dignowity Hill

These 5 zip codes boast projected returns on a 3 year investment ranging from 38% to 45%, and 5 year returns from 64% to 75%! The entry price into these neighborhoods range from 163k to 241k, much lower than the average median price of the whole SA metro area at 348k. Following are details and my recommendations for purchasing in each of these areas.

7.1 Inner Westside

The Inner Westside has the lowest initial investment value at 163k and the highest projected 3 and 5 year ROI at 45% and 76%, respectively. Homes should be bought in the winter and sold in the summer for an additional 2% ROI.

7.2 Denver Heights

Denver Heights has an initial investment value of 235k, which is the 2nd highest in our recommended zip codes. It has a projected 3 year ROI of 44% and 5 year ROI at 73%. It has become a millennial hot spot with a median age of 27 due to its close proximity to downtown and is already in the process of becoming an up and coming neighborhood. It has low variation in the median sale price month over month which makes one of our safest investments.

7.3 Eastside

The Eastside has an initial investment value of 184k, making it the 2nd lowest on our list. It has a 3 year projected ROI of 42%, while its 5 year projected ROI is 71%. It also has low variation in the data making it a safe investment. This area offers opportunities to purchase larger plots of land, as this area was mostly used for agricultural purposes up until recently, as the demand has increased from people wanting more land and privacy.

7.4 Arena District

The Arena District surrounds the AT&T Center where the Spurs play, and is loaded with bars, coffee shops, restaurants, and parks. This has been another rising millennial hot spot over the last several years. It is projected to have a 3 and 5 year ROI of 41% and 68%, respectively. The initial investment will cost 242k, which is the highest of our recommended zip codes. It also has largest errors in its predictions, making it our riskiest investment, relatively speaking. Interestingly enough, homes should be purchased in the winter and sold in the summer which does not follow the normal trends that we typically see. This strategy will add an additional 2% ROI when buying and selling in the Arena District.

7.5 Dignowity Hill

Dignowity Hill was the first residential suburb in San Antonio and is now a historic zone. The area now offers parks, restaurants, historic buildings, and renovated high end estates making it a place for everyone. It has an initial investment value of 228k, and projected 3 and 5 year ROI of 39% and 64%, respectively. Homes should be purchased in the winter and sold in the summer for an additional gain of 1% ROI. There is the smallest error in our predictions for this zip code, making this our safest investment.

As stated before, all of the home prices in these areas are significantly lower than the median home price of the San Antonio area. It should be noted that there is a caveat to these low prices, the schools are below average in every one of these zip codes and the rate of crime is high. This is typical in big cities, and the schools typically improve as time goes on in neighborhoods with increasing home value. If any of these factors are a concern, I recommend taking a look at the other zip codes that were in our top 10 ROI. These other zip codes are in more desirable suburban areas that have good schools, but also come at a much higher entry point and more risk associated with the purchase.

8 Future Work

Our model only looked at factors endogenous to the time series data. While this is useful for understanding how real estate markets change over time, it does nothing to explain all the other factors that are driving changes in housing prices. Some next steps in this project include added exogenous factors to our model:

- Interest rates
- Crime data
- Median income over time

I would also like to explore other areas in the dataset and modeling packages such as Facebook Prophet.