PHASE 4 PROJECT

Group 3 collaborators:

Students Names:

- Catherine Gakii
- Bryson Shitsukane
- Brenda Kinya
- Nazra Nyangwara
- Robin Mutai
- Tobias Ng'ong'a
- Jonathan Okwaro - Team Lead

ZILLOW TIME SERIES ANALYSIS PROJECT

1.0 Business Understanding

1.1 Problem statement

- Boma Yangu real estate investment firm is seeking to make strategic real estate investments in specific geographic areas in the US. Their aim is to maximize returns on their investments by identifying regions with high growth potential, favorable market conditions, and strong demand for real estate properties. They have contracted our services to analyze relevant data and provide a recommendation on the top 5 best zip codes to invest in.

1.2 Objectives

- To provide a recommendation for the top 5 best zip codes to invest in
- To develop a model that can accurately forecast expected returns from the property market over time

1.3 Metrics of Success

What defines our top best performing zipcodes?

- Return On Investment: We will use Return On Investment ratio to measure 'best'.
- The metrics we will be optimizing against/minimizing are the AIC and RMSE. The lower the AIC and RMSE, the better our model will be at forecasting.

2.0 Data Understanding

- This dataset was obtained from Zillow Research. It contains real estate property values from April 1996 to April 2018 for various Zipcodes. The columns include:
  - RegionID: An identifier for the region.
  - RegionName: The zip code.
  - City: The city where the zip code is located.
  - State: The state where the zip code is located.
  - Metro: The metropolitan area related to the zip code.
  - CountyName: The county where the zip code is located.
  - SizeRank: A ranking based on the size or importance of the zip code.
  - Monthly Data (e.g., 1996-04, 1996-05, ...): These columns represent the median real estate prices for a given zip code for each month from April 1996 to April 2018.

In [1]:
```
# Import the relevant libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import matplotlib
from matplotlib import rcParams
import seaborn as sns
from math import sqrt
from pandas.plotting import autocorrelation_plot
import statsmodels.api as sm
from statsmodels.tsa.stattools import adfuller
from statsmodels.tsa.seasonal import seasonal_decompose
from statsmodels.tsa.arima.model import ARIMA
from statsmodels.tsa.statespace.sarimax import SARIMAX
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
from statsmodels.graphics.tsaplots import acf, pacf
from sklearn.metrics import mean_squared_error
from pmdarima import auto_arima
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
# Load the dataset
data = pd.read_csv('zillow_data.csv')

# Preview the first 5 rows
data.head(5)
```

Out[2]:

	RegionID	RegionName	City	State	Metro	CountyName	SizeRank	1996-04	1996-0
0	84654	60657	Chicago	IL	Chicago	Cook	1	334200.0	335400.
1	90668	75070	McKinney	TX	Dallas-Fort Worth	Collin	2	235700.0	236900.
2	91982	77494	Katy	TX	Houston	Harris	3	210400.0	212200.
3	84616	60614	Chicago	IL	Chicago	Cook	4	498100.0	500900.
4	93144	79936	El Paso	TX	El Paso	El Paso	5	77300.0	77300.

5 rows × 272 columns

In [3]:
```
# View the last 5 rows
data.tail(5)
```

Out[3]:

	RegionID	RegionName	City	State	Metro	CountyName	SizeRank	1996-04
14718	58333	1338	Ashfield	MA	Greenfield Town	Franklin	14719	94600.0
14719	59107	3293	Woodstock	NH	Claremont	Grafton	14720	92700.0
14720	75672	40404	Berea	KY	Richmond	Madison	14721	57100.0
14721	93733	81225	Mount Crested Butte	CO	NaN	Gunnison	14722	191100.0
14722	95851	89155	Mesquite	NV	Las Vegas	Clark	14723	176400.0

5 rows × 272 columns

In [4]:
```
# Check the shape
data.shape
```

Out[4]:
```
(14723, 272)
```

In [5]:
```
# Check the information of the dataset
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14723 entries, 0 to 14722
Columns: 272 entries, RegionID to 2018-04
dtypes: float64(219), int64(49), object(4)
memory usage: 30.6+ MB
```

In [6]:
```
# Check descriptive statistics
data.describe()
```

Out[6]:

	RegionID	RegionName	SizeRank	1996-04	1996-05	1996-06	1
count	14723.000000	14723.000000	14723.000000	1.368400e+04	1.368400e+04	1.368400e+04	1
mean	81075.010052	48222.348706	7362.000000	1.182991e+05	1.184190e+05	1.185374e+05	1
std	31934.118525	29359.325439	4250.308342	8.600251e+04	8.615567e+04	8.630923e+04	8
min	58196.000000	1001.000000	1.000000	1.130000e+04	1.150000e+04	1.160000e+04	1
25%	67174.500000	22101.500000	3681.500000	6.880000e+04	6.890000e+04	6.910000e+04	6
50%	78007.000000	46106.000000	7362.000000	9.950000e+04	9.950000e+04	9.970000e+04	9
75%	90920.500000	75205.500000	11042.500000	1.432000e+05	1.433000e+05	1.432250e+05	1
max	753844.000000	99901.000000	14723.000000	3.676700e+06	3.704200e+06	3.729600e+06	3

8 rows × 268 columns

In [7]:
```
# Check for missing values
data.isna().sum()
```

Out[7]:
```
RegionID         0
RegionName       0
City             0
State            0
Metro         1043
              ...
2017-12          0
2018-01          0
2018-02          0
2018-03          0
2018-04          0
Length: 272, dtype: int64
```

In [8]:
```
# Checking how many unique values we have on each column, besides the time da
data.iloc[:,0:7].nunique()
```

Out[8]:
```
RegionID      14723
RegionName    14723
City           7554
State            51
Metro           701
CountyName     1212
SizeRank      14723
dtype: int64
```

- Observations:
  - The dataset contains 14722 rows and 272 colums
  - Some columns in the dataset contain missing values

- Data has both continuous and categorical features comprising of the following data types; objects, integers, floats
- The dataset is in wide format with the time periods appearing as columns. We will need to convert it into long format

3.0 Data preparation

3.1 Data cleaning

In [9]:
```
# Drop unnecessary columns
data = data.drop(columns = ['Metro','CountyName'], axis=1)
data
```

Out[9]:

	RegionID	RegionName	City	State	SizeRank	1996-04	1996-05	1996-06	1996
0	84654	60657	Chicago	IL	1	334200.0	335400.0	336500.0	3376(
1	90668	75070	McKinney	TX	2	235700.0	236900.0	236700.0	2354(
2	91982	77494	Katy	TX	3	210400.0	212200.0	212200.0	2107(
3	84616	60614	Chicago	IL	4	498100.0	500900.0	503100.0	5046(
4	93144	79936	El Paso	TX	5	77300.0	77300.0	77300.0	773(
...	...	...	...	...	...	...	...	...	
14718	58333	1338	Ashfield	MA	14719	94600.0	94300.0	94000.0	937(
14719	59107	3293	Woodstock	NH	14720	92700.0	92500.0	92400.0	922(
14720	75672	40404	Berea	KY	14721	57100.0	57300.0	57500.0	577(
14721	93733	81225	Mount Crested Butte	CO	14722	191100.0	192400.0	193700.0	1950(
14722	95851	89155	Mesquite	NV	14723	176400.0	176300.0	176100.0	1760(

14723 rows × 270 columns

3.2 Data Preprocessing

- To make informed investment decisions, we will enrich our dataset by calculating a few key metrics that provide insights into the historical performance and variability of housing prices for each zip code. Here is the breakdown of the metrics we will use:

    1. Historical Return on Investment (ROI):
        - This metric provides a measure of how much the value of a property in a given zip code has appreciated (or depreciated) over the entire span of our dataset. A

higher ROI indicates that properties in this zip code have historically appreciated in value at a faster rate.

2. Standard Deviation of Monthly Values (std):
   - This metric measures the volatility or variability in monthly housing prices for each zip code over the time span of the dataset. A higher standard deviation indicates greater volatility and potential risk, but also potential reward.
3. Historical Mean Value (mean):
   - This metric provides the average monthly housing price for each zip code over the duration of our dataset. It gives us a general sense of the typical housing price in a given zip code.
4. Coefficient of Variance (CV):
   - The CV is a standardized measure of dispersion of a probability or frequency distribution. For our dataset, it provides a relative measure of the volatility in housing prices for each zip code. A higher CV indicates more volatility when

In [10]:
```
# Calculate historical return on investment
data['ROI']= (data['2018-04']/data['1996-04'])-1

# Calculate standard deviation of monthly values
data['std']=data.loc[:,'1996-04':'2018-04'].std(skipna=True, axis=1)

# Calculate historical mean value
data['mean']=data.loc[:,'1996-04':'2018-04'].mean(skipna=True, axis=1)

# Calculate coefficient of variance
data['CV']=data['std']/data['mean']

# Show calculated values
data[['RegionName','std','mean','ROI','CV']].head(100)
```

Out[10]:

	RegionName	std	mean	ROI	CV
0	60657	190821.103965	743978.867925	2.083782	0.256487
1	75070	33537.101427	219655.849057	0.365295	0.152680
2	77494	37730.794353	262110.566038	0.567966	0.143950
3	60614	231225.944628	974139.245283	1.623971	0.237364
4	79936	18167.079218	101875.471698	0.571798	0.178326
...	...	...	...	...	...
95	11234	123160.303830	374560.000000	2.443413	0.328813
96	92683	148048.331872	420310.188679	2.814750	0.352236
97	85710	33439.316277	143930.188679	0.909091	0.232330
98	78745	41733.948394	172890.566038	1.182963	0.241389
99	11355	192468.455520	476348.301887	3.869803	0.404050

100 rows × 5 columns

- We are renaming the column 'RegionName' to "Zipcode" to make the dataset more intuitive, especially when referencing or filtering by zip codes in subsequent analysis.

In [11]:
```
# Renaming column that refers to zipcodes
data.rename({'RegionName': 'Zipcode'}, axis='columns', inplace=True)
data.head(5)
```

Out[11]:

	RegionID	Zipcode	City	State	SizeRank	1996-04	1996-05	1996-06	1996-07	199(
0	84654	60657	Chicago	IL	1	334200.0	335400.0	336500.0	337600.0	3385(
1	90668	75070	McKinney	TX	2	235700.0	236900.0	236700.0	235400.0	2333(
2	91982	77494	Katy	TX	3	210400.0	212200.0	212200.0	210700.0	2083(
3	84616	60614	Chicago	IL	4	498100.0	500900.0	503100.0	504600.0	5055(
4	93144	79936	El Paso	TX	5	77300.0	77300.0	77300.0	77300.0	774(

5 rows × 274 columns

Identifying Top Investment Opportunities

- Examine the CV for each zip code to understand the volatility of housing prices.
- Set an upper limit for CV using the 60th percentile. This filters out zip codes with higher-than-acceptable risk.
- Identify zip codes that offer the best historical ROI and also fit within the defined risk profile.

In [12]:
```
# Descriptive statistics of coefficients of variance.
print(data.CV.describe())

# Define upper limit of CV according to risk profile.
upper_cv = data.CV.quantile(.6)
print(f'\nCV upper limit: {upper_cv}')

# Get the first 10 zipcodes with highest ROIs within the firms risk profile.
top_10 = data[data['CV']<upper_cv].sort_values('ROI',axis=0,ascending=False)[

print('\n Best 10 Zipcodes:')
top_10[['Zipcode','ROI','CV']]
```

```
count    14723.000000
mean         0.219913
std          0.084749
min          0.019330
25%          0.156041
50%          0.216008
75%          0.276864
max          0.697541
Name: CV, dtype: float64

CV upper limit: 0.2398825103642624

 Best 10 Zipcodes:
```

Out[12]:

	Zipcode	ROI	CV
13314	49309	3.353982	0.231683
13356	40107	3.018970	0.231908
12963	48822	2.963519	0.213312
10690	49265	2.931034	0.232729
12238	49425	2.903614	0.230111
8353	29645	2.883333	0.232148
8412	66206	2.677966	0.233424
13357	48835	2.590674	0.214027
13753	48894	2.561947	0.190111
13282	15486	2.531250	0.235270

Observation

- The average CV across all zip codes is approximately 0.220
- Based on our risk tolerance, the upper CV limit was set at 0.240
- The top 10 zip codes showcase high ROIs while also aligning with the firm's risk profile.

Getting the location of the top 10 Zipcodes based on Cv score and ROI

In [13]:
```
# Get the 10 zipcodes with highest ROIs within the firm's risk profile
top_10 = data[data['CV'] < upper_cv].sort_values('ROI', axis=0, ascending=Fal

# Get Location Names for the best 10 zipcodes
best10_zipcodes = list(top_10.Zipcode.values)
for i in best10_zipcodes:
    city = data[data['Zipcode'] == i].City.values[0]
    state = data[data['Zipcode'] == i].State.values[0]
    print(f'Zipcode: {i} \nLocation: {city}, {state}\n')
```

```
Zipcode: 49309
Location: Bitely, MI

Zipcode: 40107
Location: Boston, KY

Zipcode: 48822
Location: Eagle, MI

Zipcode: 49265
Location: Onsted, MI

Zipcode: 49425
Location: Holton, MI

Zipcode: 29645
Location: Gray Court, SC

Zipcode: 66206
Location: Leawood, KS

Zipcode: 48835
Location: Fowler, MI

Zipcode: 48894
Location: Westphalia, MI

Zipcode: 15486
Location: Franklin, PA
```

- By narrowing down to the top 10 zip codes, this highlighted the best-performing areas which align with Boma Yangu risk profile and offer the highest potential ROI.

Reshaping the dataset from wide to long format

In [14]:
```
# Reshape the dataset to long format
df_long = pd.melt(top_10, id_vars=['RegionID', 'Zipcode', 'City', 'State','Si
                  var_name='time', value_name='value')

df_long = df_long.dropna(subset=['value'])

df_long['time'] = pd.to_datetime(df_long.time)

# Display the reshaped dataset
df_long
```

Out[14]:

	RegionID	Zipcode	City	State	SizeRank	ROI	CV	std	
0	79678	49309	Bitely	MI	13315	3.353982	0.231683	8755.699818	3779
1	75522	40107	Boston	KY	13357	3.018970	0.231908	21638.431224	93306
2	79397	48822	Eagle	MI	12964	2.963519	0.213312	30813.609976	144453
3	79648	49265	Onsted	MI	10691	2.931034	0.232729	28923.199467	124278
4	79742	49425	Holton	MI	12239	2.903614	0.230111	15550.058490	67576
...	...	...	...	...	...	...	...	...	
2645	70626	29645	Gray Court	SC	8354	2.883333	0.232148	19289.795188	83092
2646	87122	66206	Leawood	KS	8413	2.677966	0.233424	65332.460851	279887
2647	79409	48835	Fowler	MI	13358	2.590674	0.214027	23191.800720	108359
2648	79465	48894	Westphalia	MI	13754	2.561947	0.190111	24428.957688	128498
2649	64135	15486	Franklin	PA	13283	2.531250	0.235270	20749.032919	88192

2650 rows × 11 columns

In [15]:
```
# Check the datatypes
df_long.dtypes
```

Out[15]:
```
RegionID             int64
Zipcode              int64
City                object
State               object
SizeRank             int64
ROI                float64
CV                 float64
std                float64
mean               float64
time        datetime64[ns]
value              float64
dtype: object
```

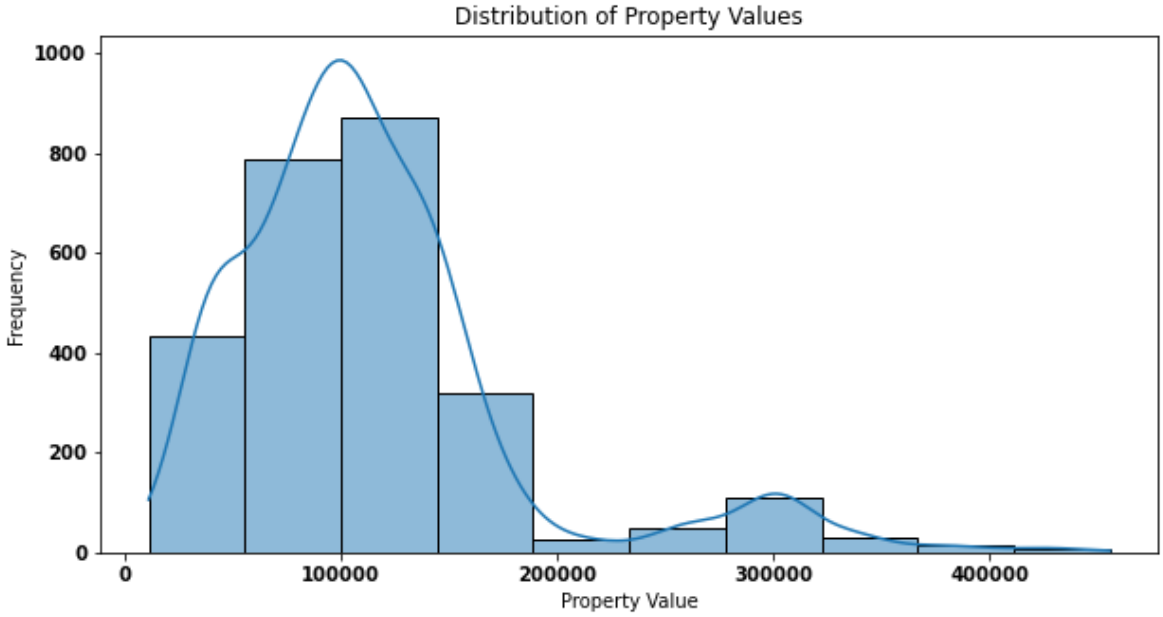
In [16]:
```
font = {'family' : 'sans-serif',
        'weight' : 'bold',
        'size'   : 10}

plt.rc('font', **font)

# NOTE: if you visualizations are too cluttered to read, try calling 'plt.gcf
```

4.0 EDA and Visualization

We will now proceed to visualizing our reshaped dataset.

a) Distribution of Property Values for top 10 Zip Codes

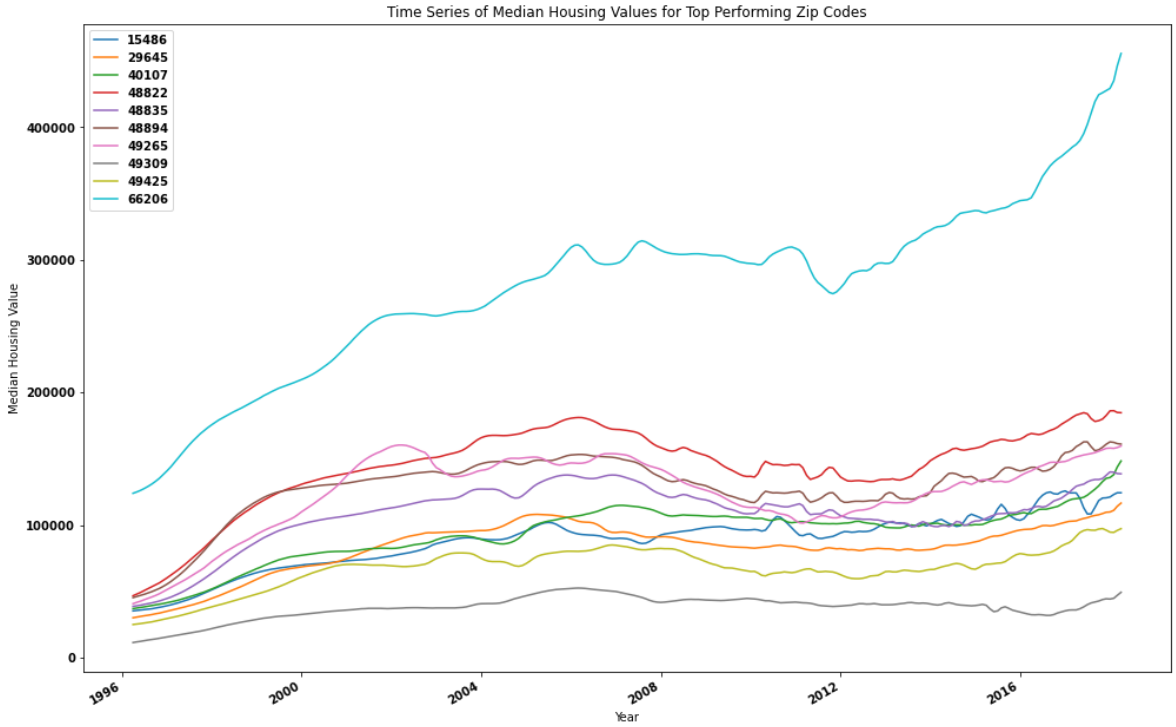
In [17]:
```
# Visualize the distribution of real estate prices
plt.figure(figsize=(10, 5))
sns.histplot(df_long['value'], bins=10, kde=True)
plt.xlabel('Property Value')
plt.ylabel('Frequency')
plt.title('Distribution of Property Values')
plt.show()
```



- From the plot, we can infer the common price ranges within these top-performing zip codes and identify any patterns or anomalies. This knowledge aids in understanding the typical property values in these areas and can inform investment decisions.
- The distribution of real estate values is right-skewed, with a long tail on the right side, indicating that the majority of the property values are concentrated in the lower price range.
- The shape of the distribution suggests that the real estate market consists of a mix of moderately priced properties and a smaller number of high-priced properties which form the tail of the distribution.

b) Time Series Plot for the median property values of the top 10 zip codes

In [18]:
```
# Pivot the data to get median values for each zip code
pivot_df = df_long.pivot_table(index='Zipcode', columns='time', values='value

pivot_df
# Sort zip codes by ROI in descending order
top_performers = df_long['ROI'].sort_values(ascending=False).head(10)
top_performers


# Assuming 'pivot_df' contains the pivot table of median housing sales values
plt.figure(figsize=(14, 10))
for col in pivot_df.index[:10]:  # Plot top 10 zip codes
    plt.plot(pivot_df.columns, pivot_df.loc[col], label=col)
plt.xlabel('Year')
plt.ylabel('Median Housing Value')
plt.title('Time Series of Median Housing Values for Top Performing Zip Codes'
plt.legend()
plt.tight_layout()
plt.gcf().autofmt_xdate()
plt.show()
```



- From this plot, we observe the historical trends in property values for each zip code. Some zip codes show steady appreciation over time, while others might have more volatility or distinct growth phases.
- There were noticeable dips around 2008-2010, which correspond with the global financial crisis. After this period, there's a clear recovery and steady growth for most Zipcodes.

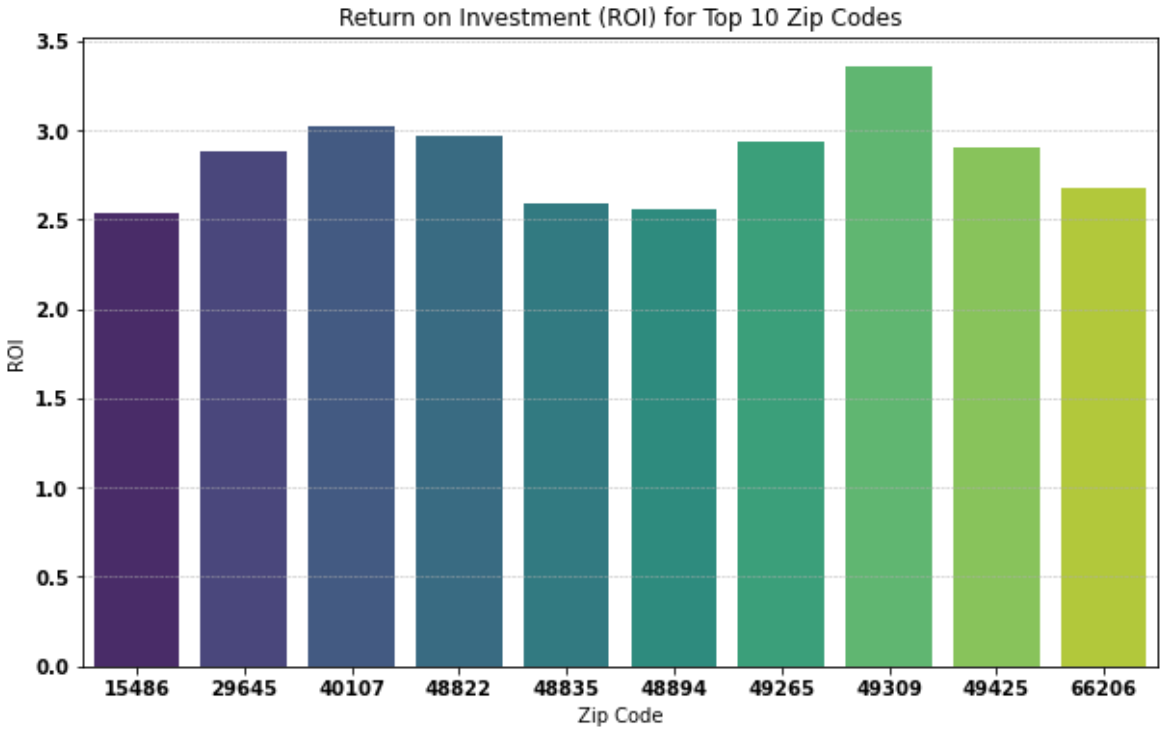
c) Distribution of Property Values for top 10 Zip Codes

In [19]:
```
# Box Plot for distribution of property values for top 10 zip codes

# Filter out rows with non-numeric values in the 'value' column
df_long_filtered = df_long[pd.to_numeric(df_long['value'], errors='coerce').n
df_long_filtered['value'] = df_long_filtered['value'].astype(float)

plt.figure(figsize=(12, 6))
sns.boxplot(data=df_long_filtered, x='Zipcode', y='value')
plt.title('Distribution of Property Values for top 10 Zip Codes')
plt.xlabel('Zip Code')
plt.ylabel('Property Value')
plt.grid(True, which='both', axis='y', linestyle='--', linewidth=0.5)
plt.show()
```

- From the plot, you can infer the median, variability, and potential outliers in property values for each zip code.

d) Return on Investment (ROI) for top 10 Zip Codes

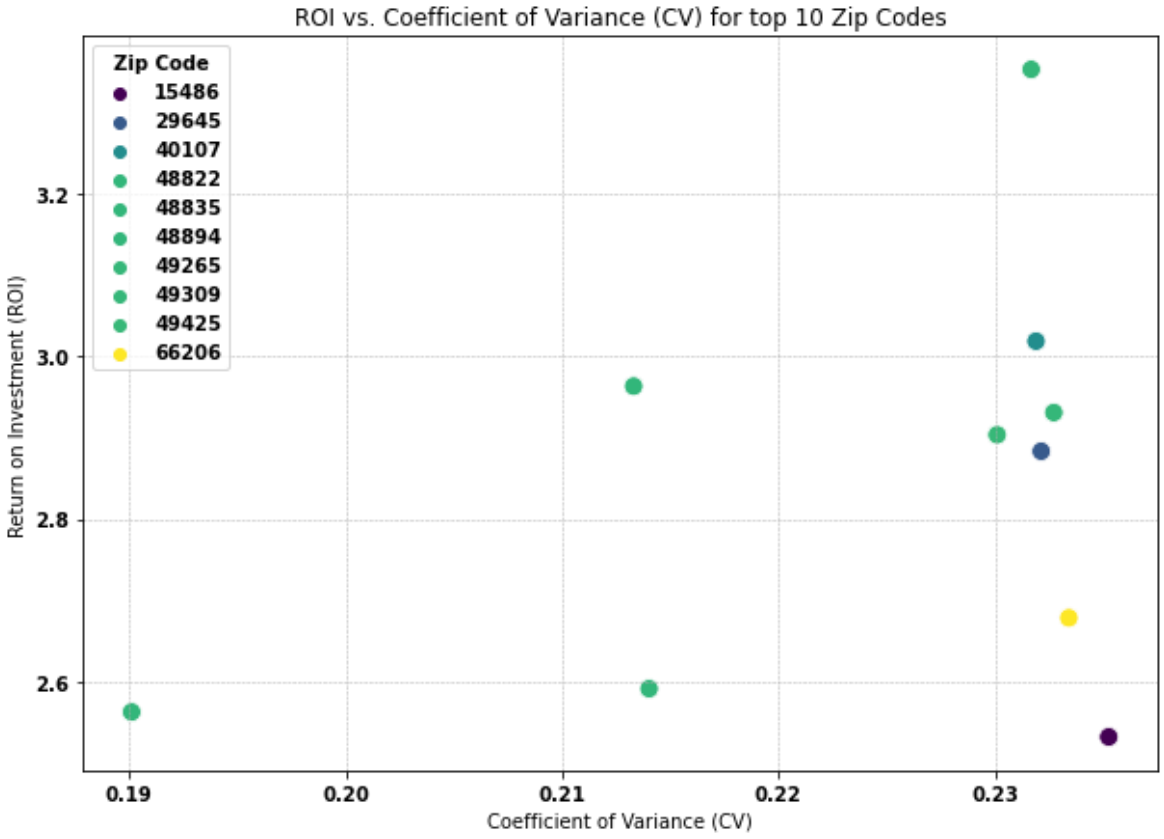
In [20]:
```
# Bar Plot to compare ROI for top 10 zip codes

plt.figure(figsize=(10, 6))
sns.barplot(data=df_long, x='Zipcode', y='ROI', palette='viridis')
plt.title('Return on Investment (ROI) for Top 10 Zip Codes')
plt.xlabel('Zip Code')
plt.ylabel('ROI')
plt.grid(True, which='both', axis='y', linestyle='--', linewidth=0.5)
plt.show()
```



- From this visualization, we can infer that Zipcode 49309 has yielded the highest ROI over the years, followed by 40107.

e) Scatter Plot showing ROI vs. Coefficient of Variance (CV) for top 10 Zip Codes

In [21]:
```
# Scatter Plot to show relationship between ROI and CV for top 10 zip codes

plt.figure(figsize=(10, 7))
sns.scatterplot(data=df_long, x='CV', y='ROI', hue='Zipcode', s=100, palette=
plt.title('ROI vs. Coefficient of Variance (CV) for top 10 Zip Codes')
plt.xlabel('Coefficient of Variance (CV)')
plt.ylabel('Return on Investment (ROI)')
plt.grid(True, which='both', linestyle='--', linewidth=0.5)
plt.legend(title='Zip Code')
plt.show()
```



- The scatter plot provides insights into the trade-off between risk (volatility in property values) and return (historical appreciation). Zip codes closer to the top-left corner offer higher ROI with lower risk, making them more desirable for investment.

f) Time series Analysis

In [22]:
```
TS_top10 = df_long

# Set the time column as the index
top_10_index = TS_top10.set_index('time')

print('Time series data for the 10 zip codes:\n', TS_top10.head())

# Create individualized time series for each zipcode
dfs_ts = []

for zc in TS_top10['Zipcode'].unique():
    # Create separate dataframes for each zipcode with a monthly frequency
    df = top_10_index[top_10_index['Zipcode'] == zc].asfreq('MS')
    dfs_ts.append(df)

# Print the time series data for the first zipcode in the list
print(f'\nZipcode {TS_top10["Zipcode"].unique()[0]} time series:')
dfs_ts[0].head()
```

```
Time series data for the 10 zip codes:
    RegionID  Zipcode    City State  SizeRank       ROI        CV  \
0     79678    49309  Bitely    MI     13315  3.353982  0.231683
1     75522    40107  Boston    KY     13357  3.018970  0.231908
2     79397    48822   Eagle    MI     12964  2.963519  0.213312
3     79648    49265  Onsted    MI     10691  2.931034  0.232729
4     79742    49425  Holton    MI     12239  2.903614  0.230111

            std           mean       time    value
0   8755.699818   37791.698113 1996-04-01  11300.0
1  21638.431224   93306.037736 1996-04-01  36900.0
2  30813.609976  144453.207547 1996-04-01  46600.0
3  28923.199467  124278.490566 1996-04-01  40600.0
4  15550.058490   67576.226415 1996-04-01  24900.0

Zipcode 49309 time series:
```

Out[22]:

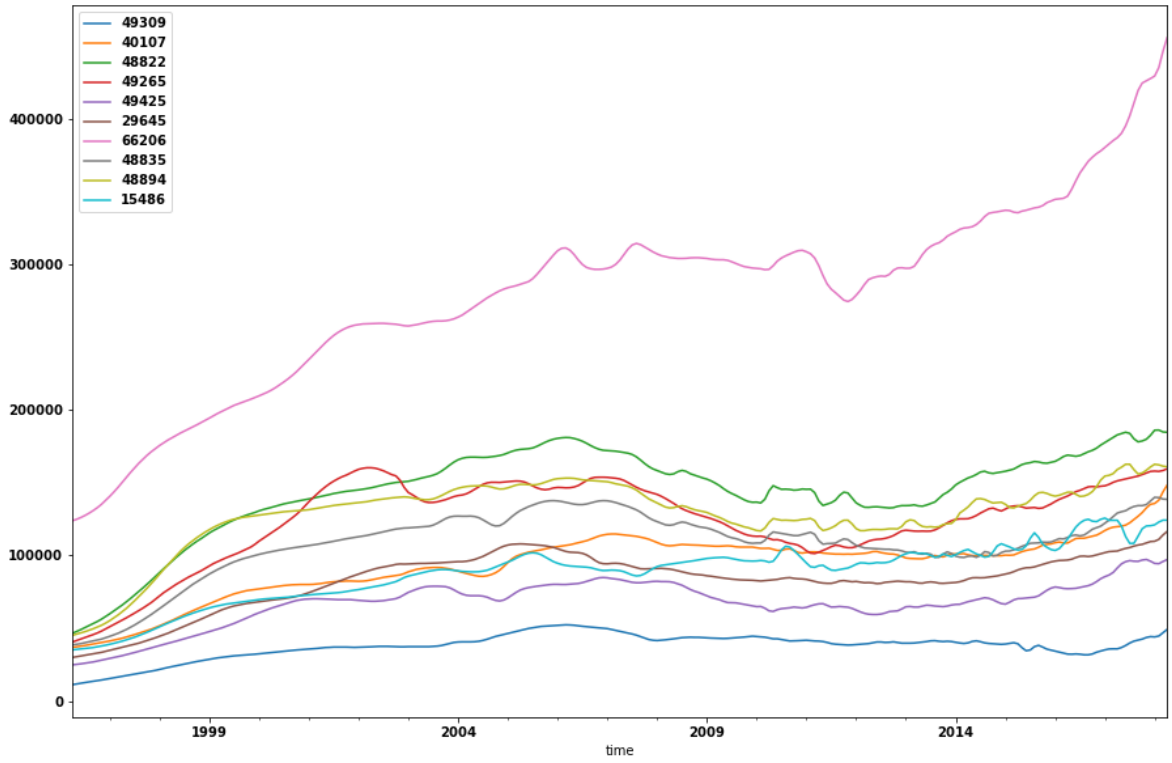
time	RegionID	Zipcode	City	State	SizeRank	ROI	CV	std	mea
1996-04-01	79678	49309	Bitely	MI	13315	3.353982	0.231683	8755.699818	37791.69811
1996-05-01	79678	49309	Bitely	MI	13315	3.353982	0.231683	8755.699818	37791.69811
1996-06-01	79678	49309	Bitely	MI	13315	3.353982	0.231683	8755.699818	37791.69811
1996-07-01	79678	49309	Bitely	MI	13315	3.353982	0.231683	8755.699818	37791.69811
1996-08-01	79678	49309	Bitely	MI	13315	3.353982	0.231683	8755.699818	37791.69811

In [23]:
```
# Descriptive statistics for the chosen zipcodes
for i in range(len(dfs_ts)):
    print(f'Value descriptive statistics for zipcode {dfs_ts[i].Zipcode[0]}:'
    print(f'{dfs_ts[i].value.describe()}\n')
```

```
Value descriptive statistics for zipcode 49309:
count      265.000000
mean     37791.698113
std       8755.699818
min      11300.000000
25%      34900.000000
50%      39600.000000
75%      43000.000000
max      52400.000000
Name: value, dtype: float64

Value descriptive statistics for zipcode 40107:
count       265.000000
mean      93306.037736
std       21638.431224
min       36900.000000
25%       82300.000000
50%      100500.000000
75%      106700.000000
max      148300.000000
Name: value, dtype: float64

Value descriptive statistics for zipcode 48822:
count       265.000000
mean     144453.207547
std       30813.609976
min       46600.000000
25%      134700.000000
50%      148500.000000
75%      167400.000000
max      186200.000000
Name: value, dtype: float64

Value descriptive statistics for zipcode 49265:
count       265.000000
mean     124278.490566
std       28923.199467
min       40600.000000
25%      109900.000000
50%      132100.000000
75%      147100.000000
max      160300.000000
Name: value, dtype: float64

Value descriptive statistics for zipcode 49425:
count      265.000000
mean     67576.226415
std      15550.058490
min      24900.000000
25%      63800.000000
50%      69500.000000
75%      78100.000000
max      97300.000000
Name: value, dtype: float64

Value descriptive statistics for zipcode 29645:
count       265.000000
```

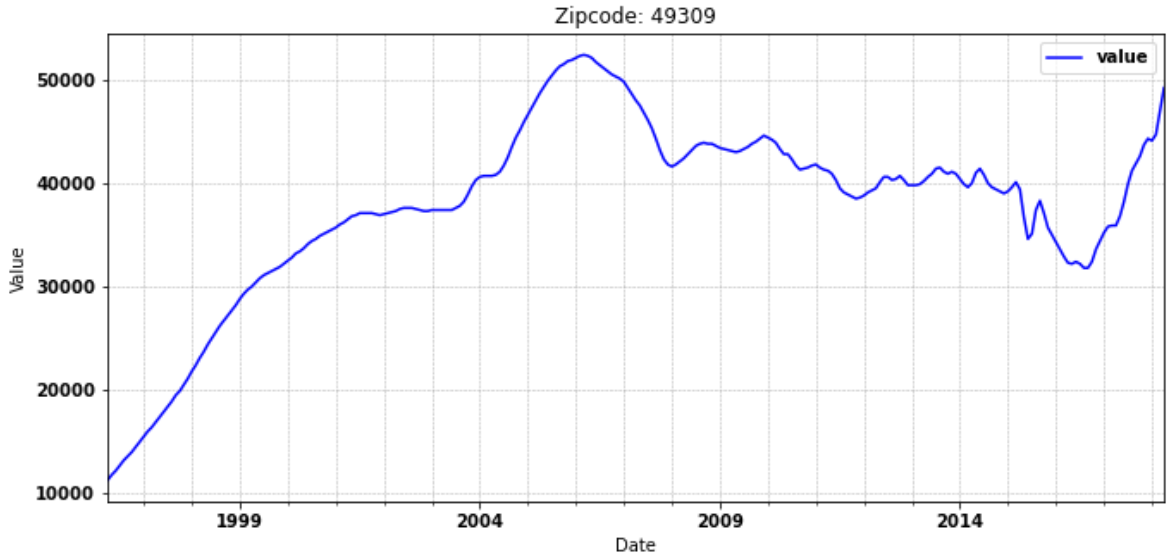
```
mean      83092.830189
std       19289.795188
min       30000.000000
25%       80900.000000
50%       85200.000000
75%       95100.000000
max      116500.000000
Name: value, dtype: float64

Value descriptive statistics for zipcode 66206:
count       265.000000
mean     279887.169811
std       65332.460851
min      123900.000000
25%      256300.000000
50%      296400.000000
75%      310200.000000
max      455700.000000
Name: value, dtype: float64

Value descriptive statistics for zipcode 48835:
count       265.000000
mean     108359.245283
std       23191.800720
min       38600.000000
25%      102100.000000
50%      111500.000000
75%      122300.000000
max      140000.000000
Name: value, dtype: float64

Value descriptive statistics for zipcode 48894:
count       265.000000
mean     128498.113208
std       24428.957688
min       45200.000000
25%      121600.000000
50%      133300.000000
75%      143500.000000
max      162700.000000
Name: value, dtype: float64

Value descriptive statistics for zipcode 15486:
count       265.000000
mean      88192.452830
std       20749.032919
min       35200.000000
25%       75300.000000
50%       92600.000000
75%      100200.000000
max      125700.000000
Name: value, dtype: float64
```

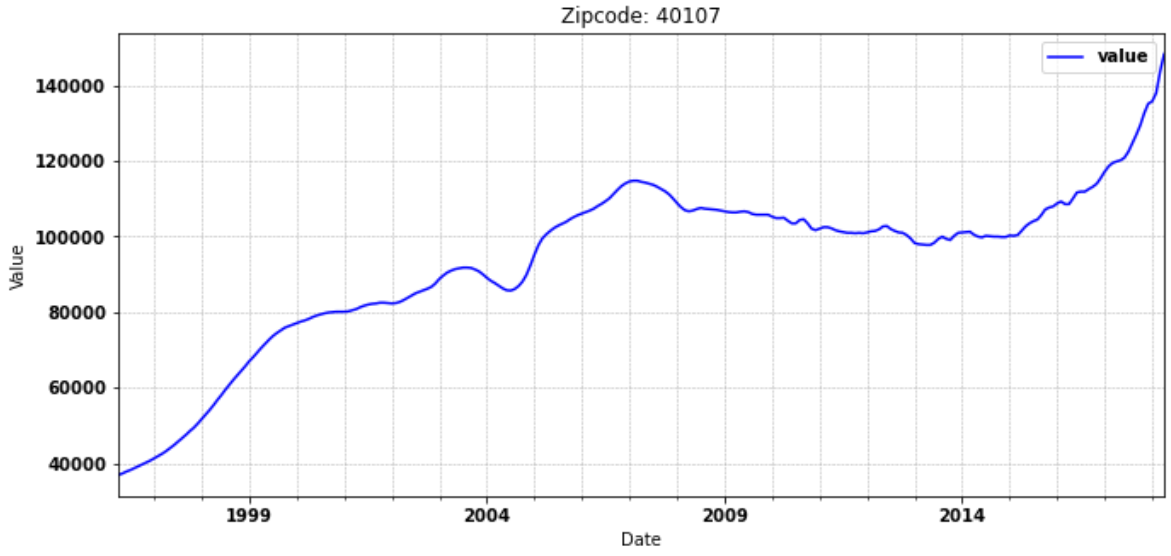
In [24]:
```
# Time series plots for the chosen zipcodes
for i in range(10):
    dfs_ts[i].value.plot(label=dfs_ts[i].Zipcode[0],figsize=(15,10))
    plt.legend()
```

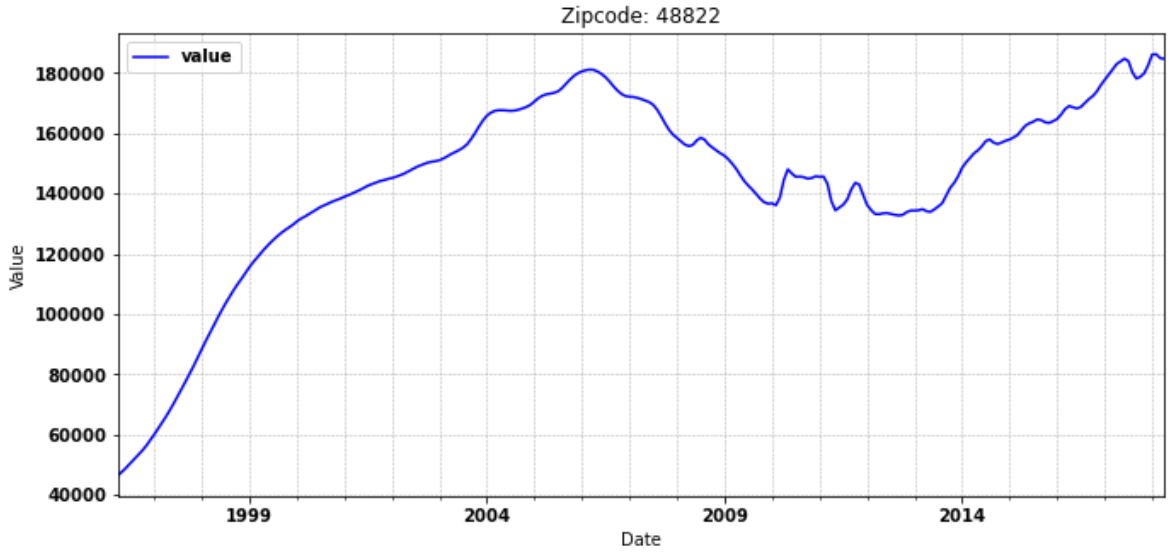


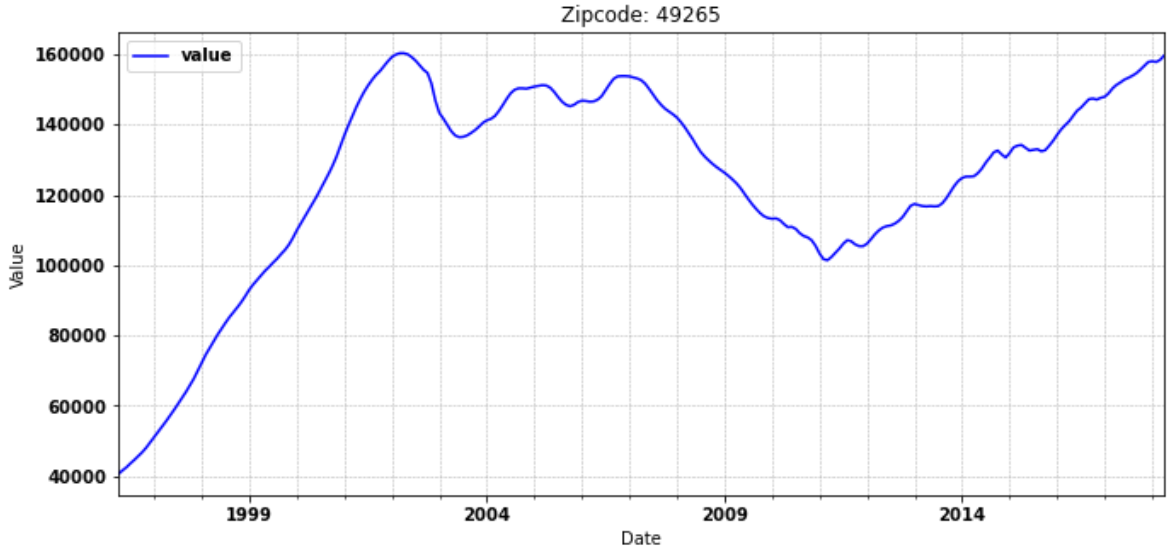
- From this plot, you can observe the historical trends and patterns in property values for each zip code. Some zip codes show steady appreciation over time, while others might have periods of stagnation or more pronounced fluctuations.

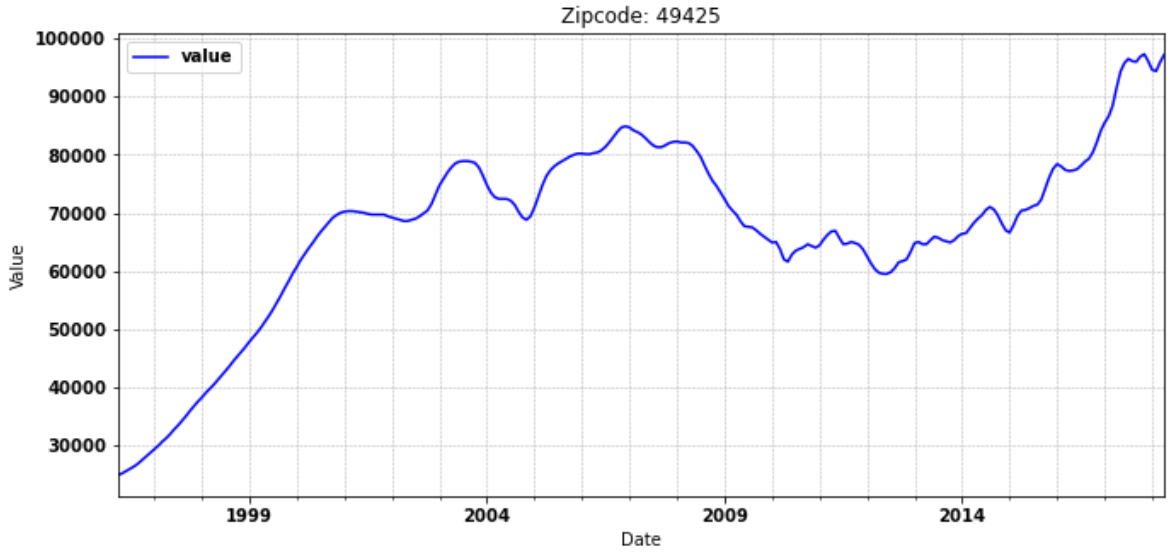
In [25]:
```
# Visualizing the housing prices per zipcode
for zc in range(len(dfs_ts)):
    dfs_ts[zc]['ret'] = dfs_ts[zc]['value']

# Plotting the monthly returns for each of the top 10 zip codes
for i in range(len(dfs_ts)):
    dfs_ts[i]['value'].plot(figsize=(11,5), color = 'b')
    plt.title(f'Zipcode: {dfs_ts[i].Zipcode.iloc[0]}')
    plt.xlabel('Date')
    plt.ylabel('Value')
    plt.legend(loc='best')
    plt.grid(True, which='both', linestyle='--', linewidth=0.5)
    plt.show()
```

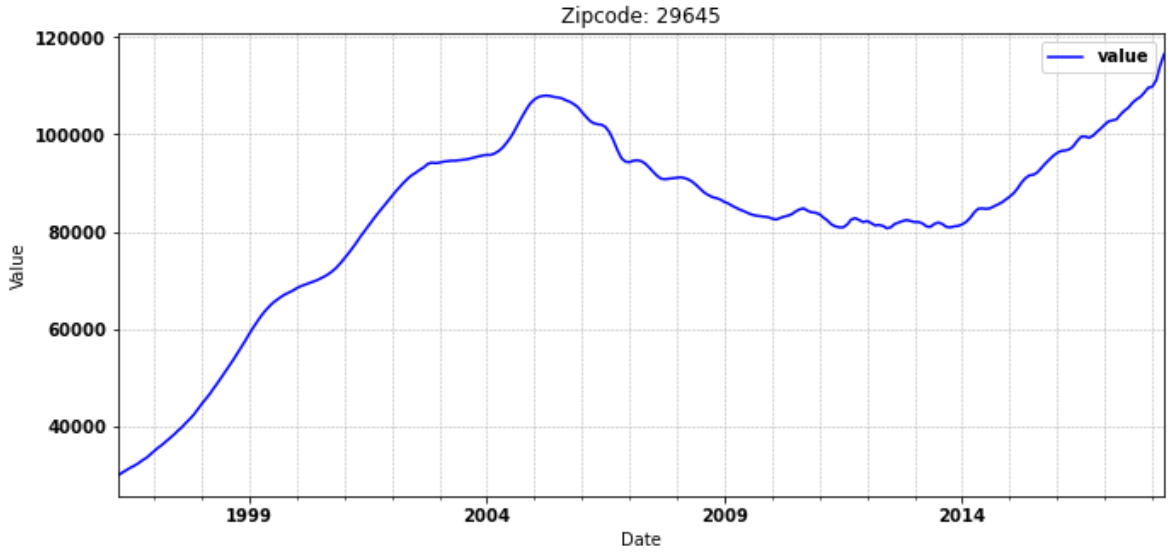


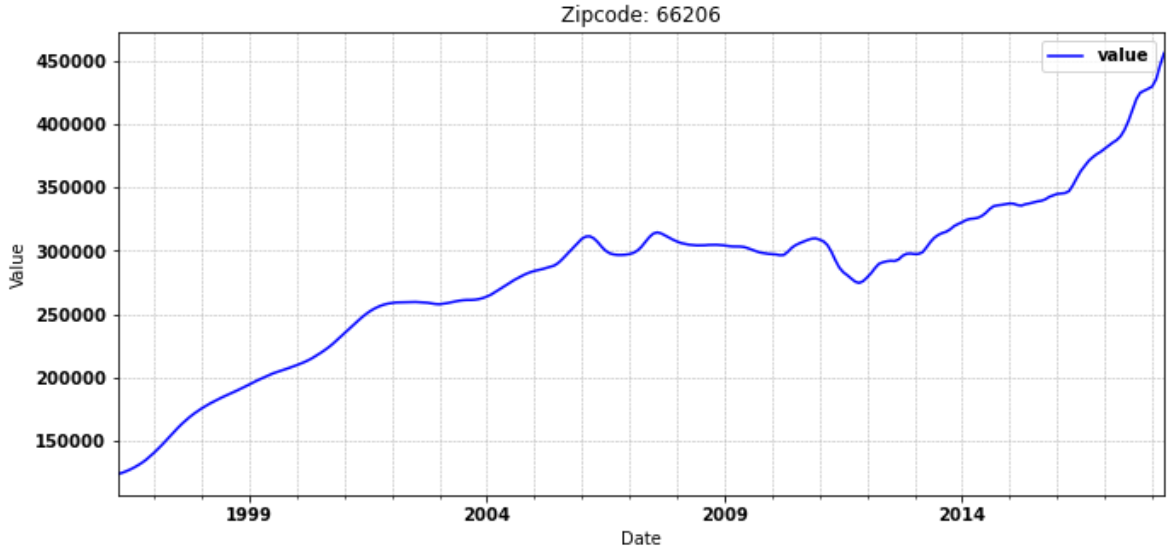


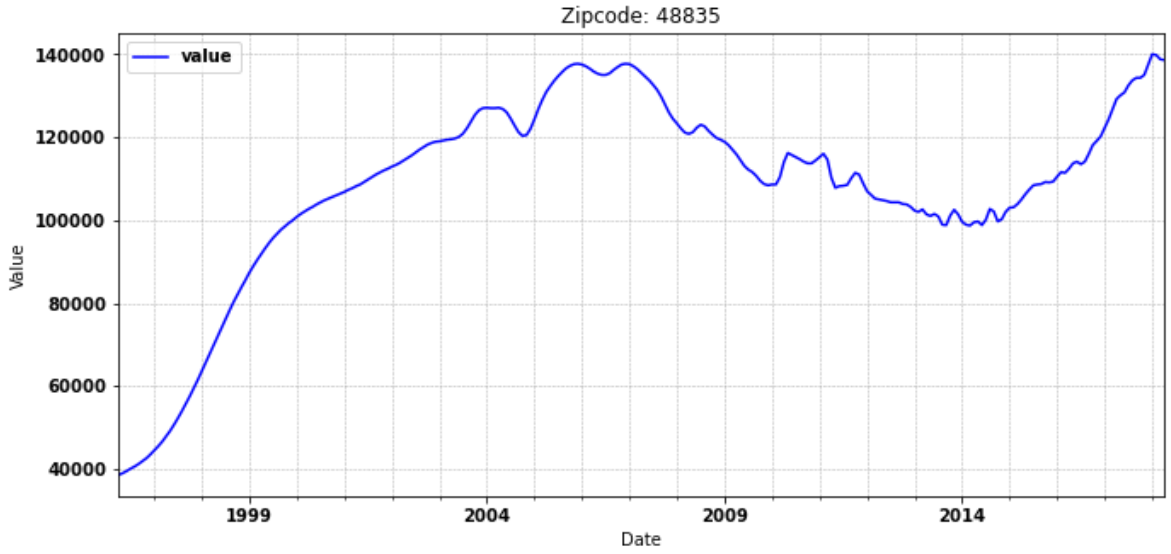

Zipcode: 48822
value
180000
160000
140000
120000
100000
80000
60000
40000
Value
1999
2004
2009
2014
Date

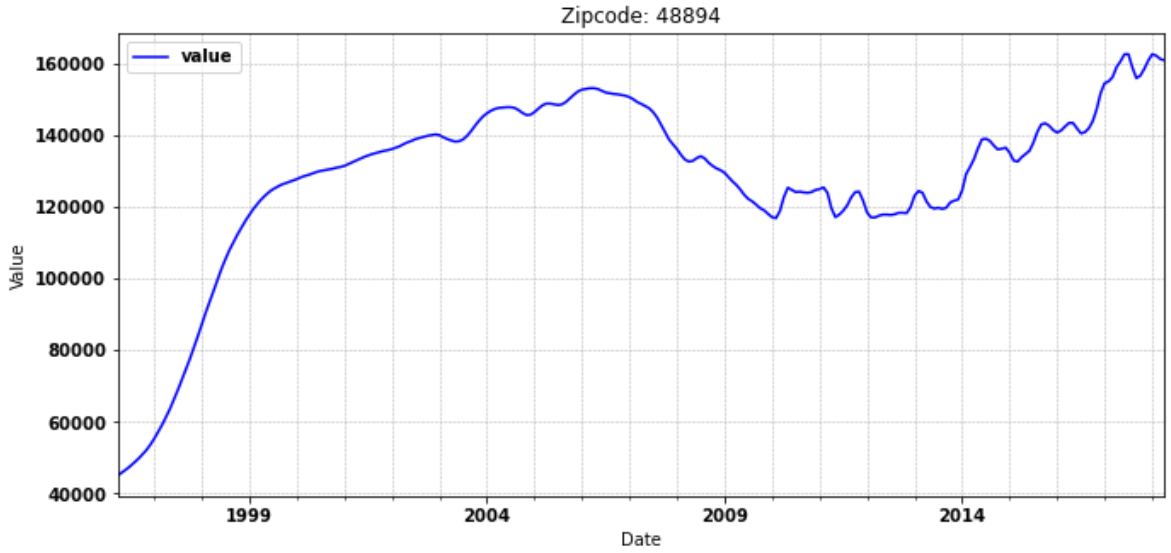

Zipcode: 49265
value
160000
140000
120000
100000
80000
60000
40000
Value
1999
2004
2009
2014
Date

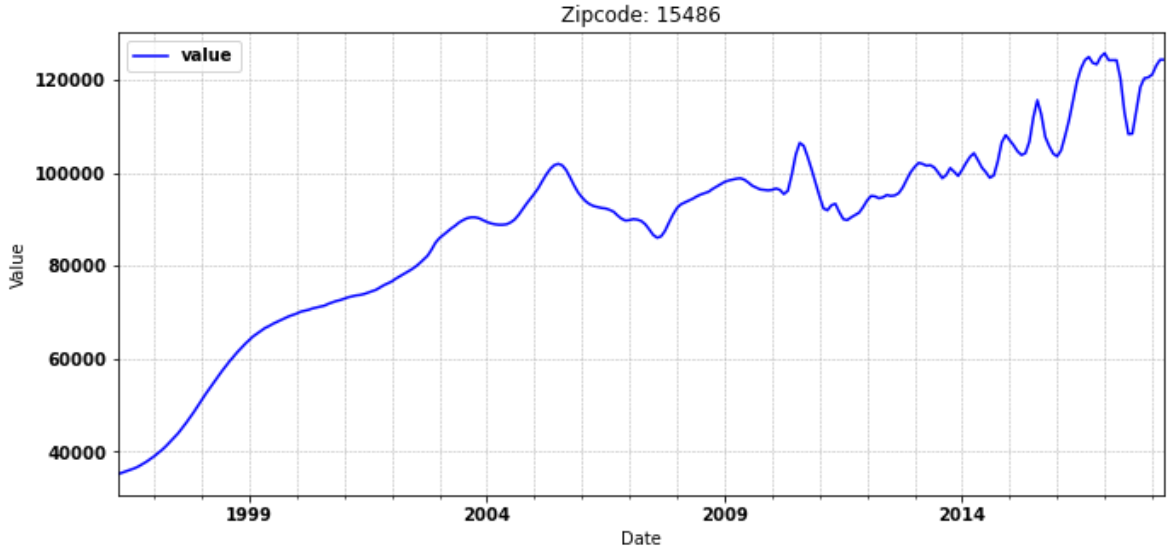

Zipcode: 49425
value
100000
90000
80000
70000
60000
50000
40000
30000
Value
1999
2004
2009
2014
Date


Zipcode: 29645
value
120000
100000
80000
60000
40000
Value
1999
2004
2009
2014
Date


Zipcode: 66206
value
450000
400000
350000
300000
250000
200000
150000
Value
1999
2004
2009
2014
Date

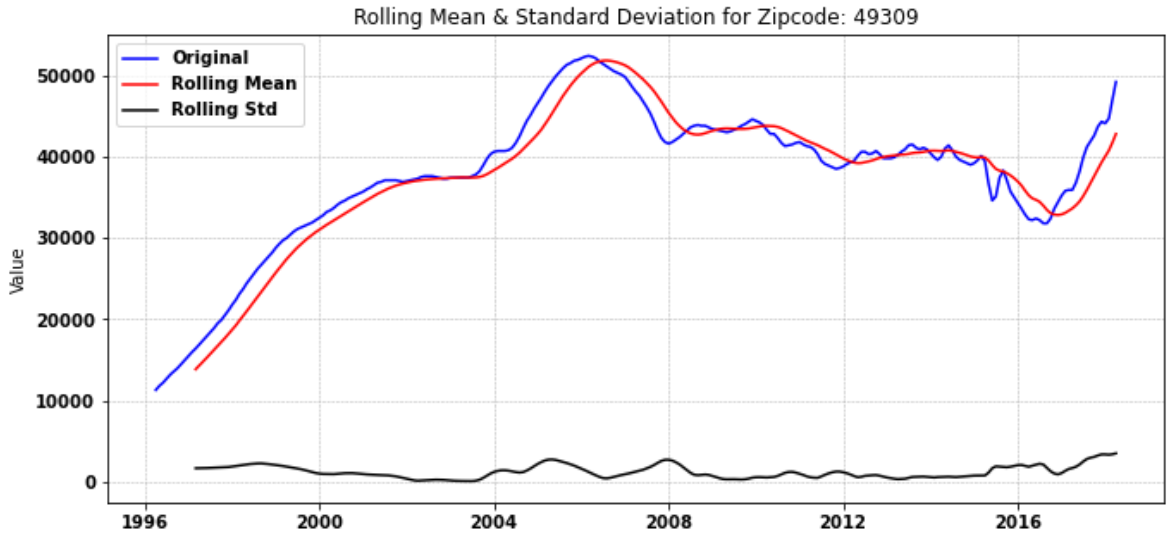

Zipcode: 48835
value
140000
120000
100000
80000
60000
40000
Value
1999
2004
2009
2014
Date

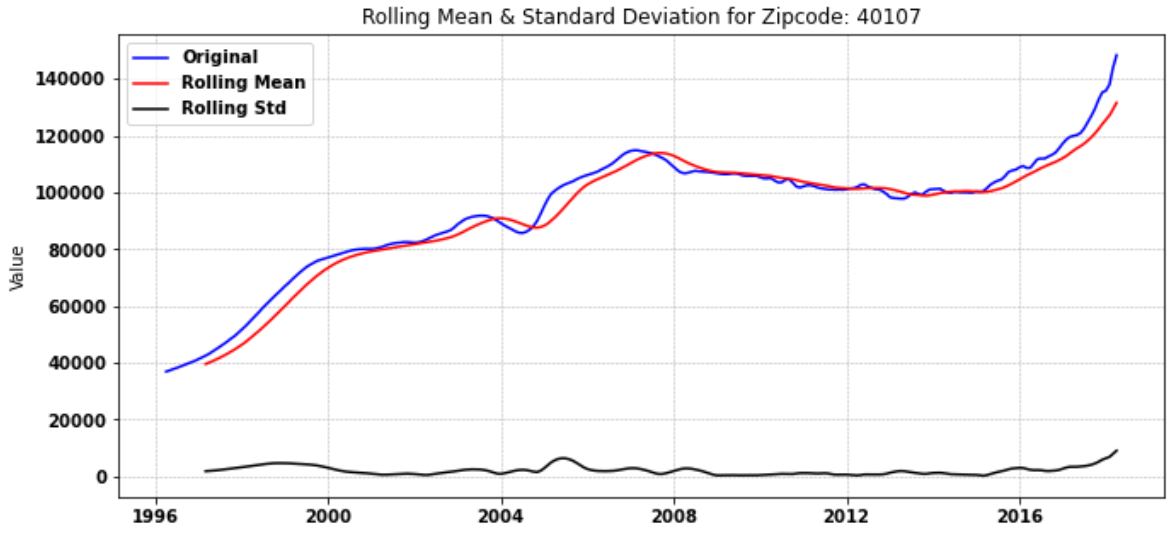


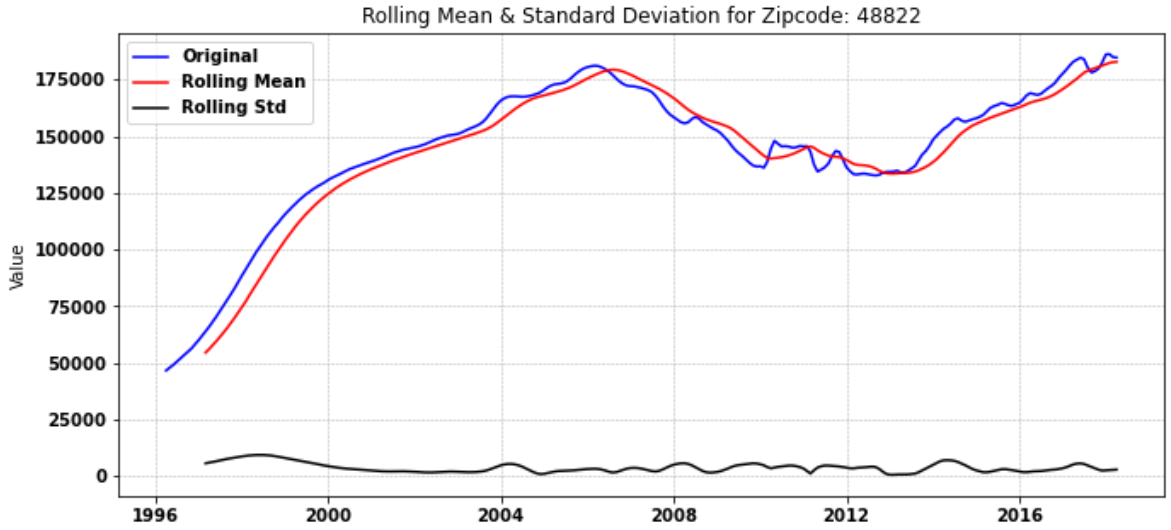


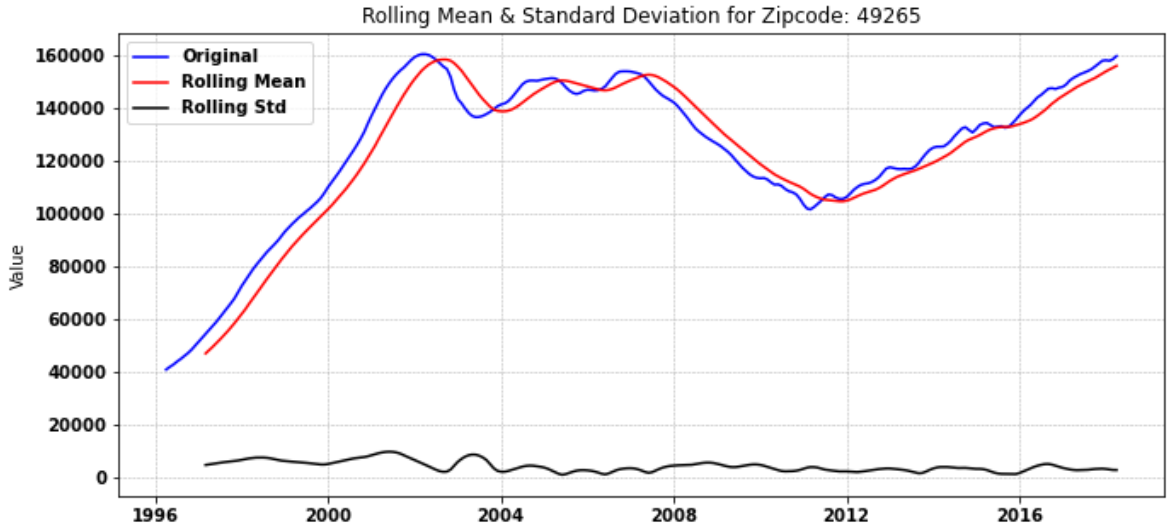
- These graphs shows the fluctuations in returns over time for each zip code. The monthly returns highlight the volatility, trends, and potential patterns in the property values.

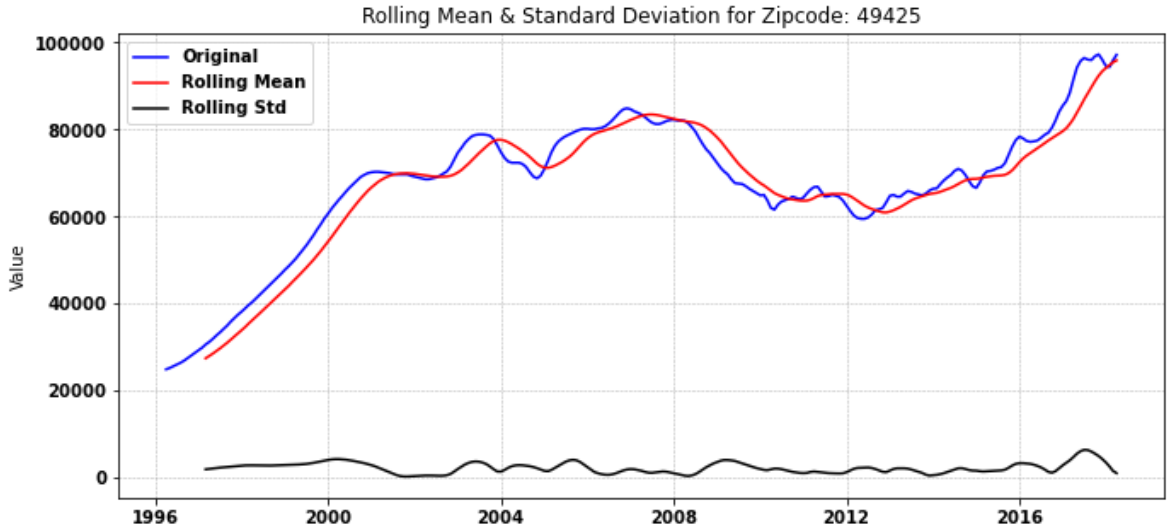
In [26]:
```
# Plotting monthly returns with their respective rolling mean and rolling std
for i in range(len(dfs_ts)):
    rolmean = dfs_ts[i]['value'].rolling(window=12, center=False).mean()
    rolstd = dfs_ts[i]['value'].rolling(window=12, center=False).std()

    # Plotting the metrics
    fig = plt.figure(figsize=(11,5))
    orig = plt.plot(dfs_ts[i]['value'], color='blue', label='Original')
    mean = plt.plot(rolmean, color='red', label='Rolling Mean')
    std = plt.plot(rolstd, color='black', label='Rolling Std')

    plt.legend(loc='best')
    plt.title(f'Rolling Mean & Standard Deviation for Zipcode: {dfs_ts[i].Zip
    plt.ylabel('Value')
    plt.grid(True, which='both', linestyle='--', linewidth=0.5)
    plt.show()
```

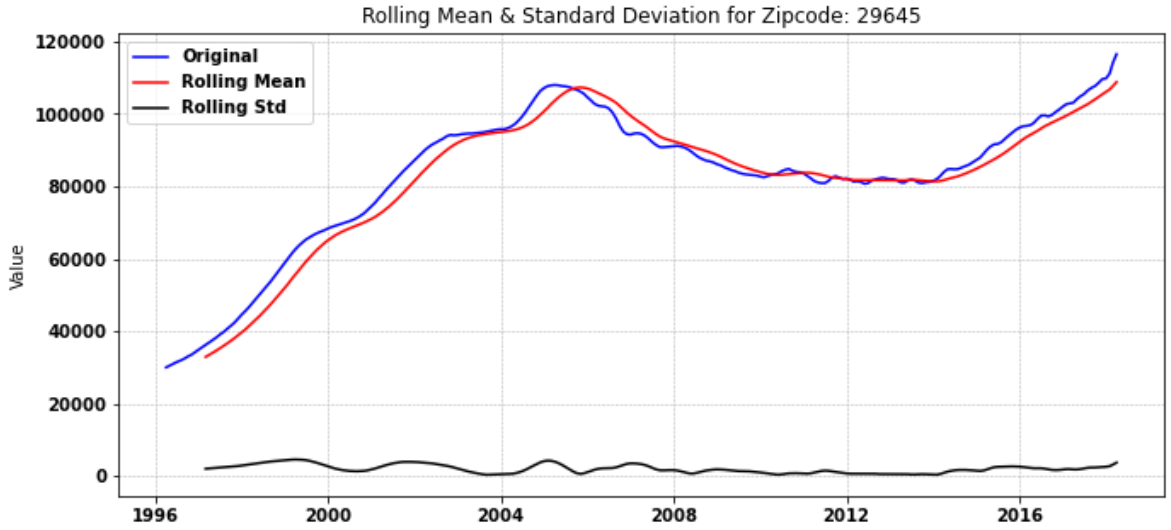


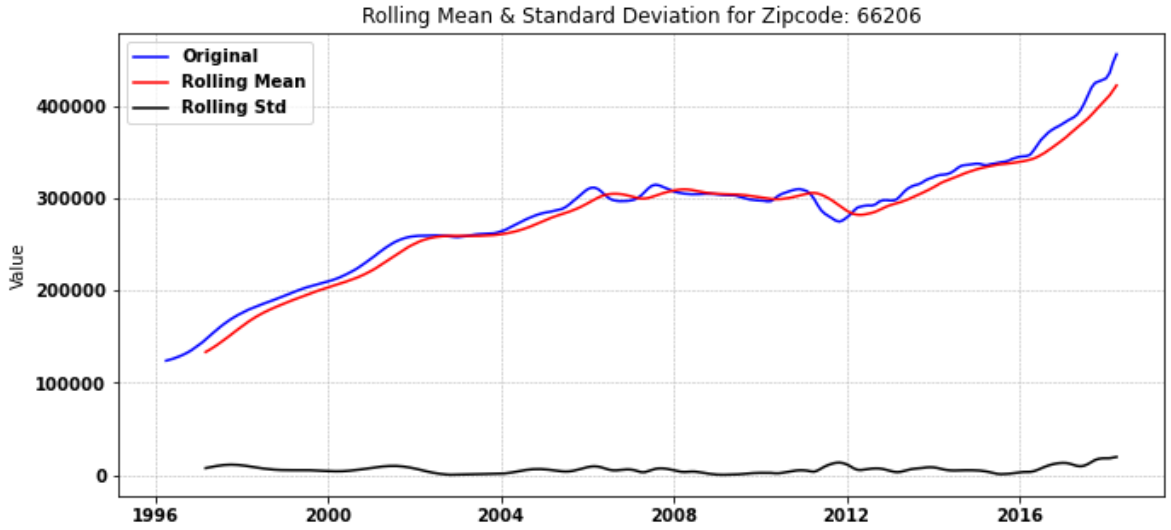


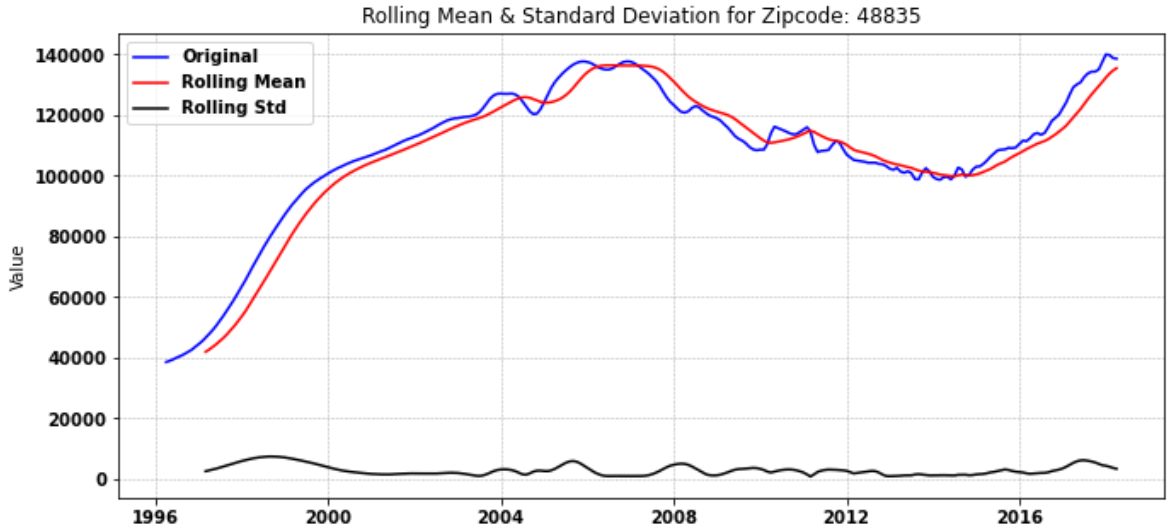

Rolling Mean & Standard Deviation for Zipcode: 48822
Original
Rolling Mean
Rolling Std
Value
175000
150000
125000
100000
75000
50000
25000
0
1996
2000
2004
2008
2012
2016

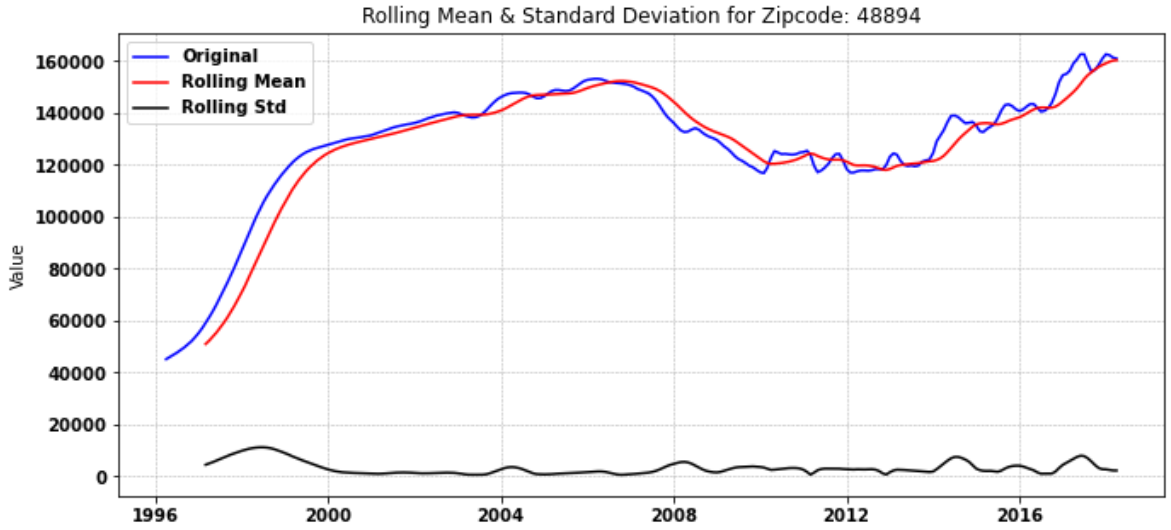

Rolling Mean & Standard Deviation for Zipcode: 49265
Original
Rolling Mean
Rolling Std
Value
160000
140000
120000
100000
80000
60000
40000
20000
0
1996
2000
2004
2008
2012
2016

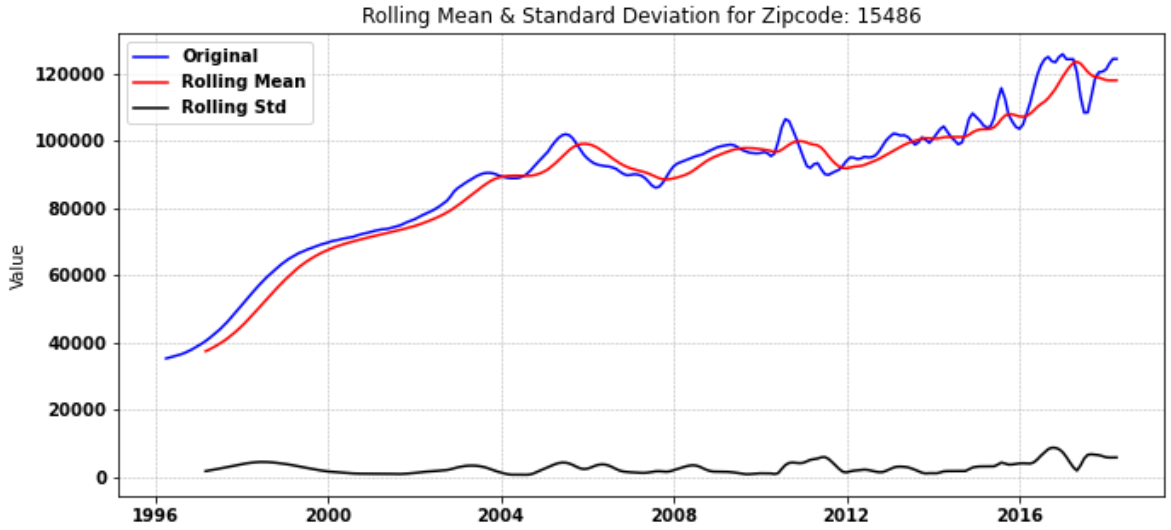

Rolling Mean & Standard Deviation for Zipcode: 49425
Original
Rolling Mean
Rolling Std
Value
100000
80000
60000
40000
20000
0
1996
2000
2004
2008
2012
2016


Rolling Mean & Standard Deviation for Zipcode: 29645
Original
Rolling Mean
Rolling Std
Value
120000
100000
80000
60000
40000
20000
0
1996
2000
2004
2008
2012
2016


Rolling Mean & Standard Deviation for Zipcode: 66206
Original
Rolling Mean
Rolling Std
Value
400000
300000
200000
100000
0
1996
2000
2004
2008
2012
2016


Rolling Mean & Standard Deviation for Zipcode: 48835
Original
Rolling Mean
Rolling Std
Value
140000
120000
100000
80000
60000
40000
20000
0
1996
2000
2004
2008
2012
2016





- At a first glance, the data appears to exhibit a stable pattern. To ascertain the extent of this stability, we will proceed to conduct the Augmented Dickey-Fuller test. This test will provide a deeper insight into the stationarity of the data by subjecting it to a statistical challenge. In essence, the test posits the null hypothesis that an order of integration exists within the data, implying it's non-stationary. Conversely, the alternative hypothesis suggests the absence of such integration, indicating data stationarity.
- Employing a confidence level of 95%, my criterion for rejecting the null hypothesis rests on a p-value lower than 0.05. This choice allows for a robust determination of whether the data is truly stationary or not.

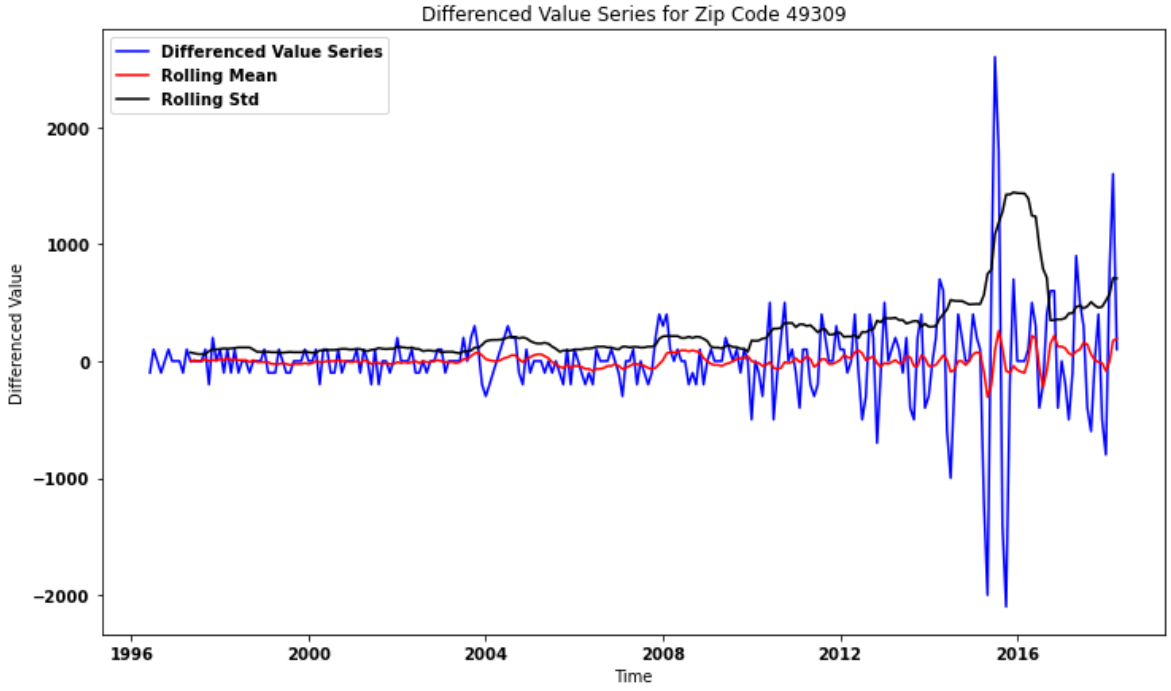
In [27]:
```
# Check for stationarity
for i in range(10):
    results = adfuller(dfs_ts[i].ret.dropna())
    print(f'ADFuller test p-value for zipcode: {dfs_ts[i].Zipcode[0]}')
    print('p-value:',results[1])
    if results[1]>0.05:
        print('Fail to reject the null hypothesis. Data is not stationary.\n'
    else:
        print('Reject the null hypothesis. Data is stationary.\n')
```

```
ADFuller test p-value for zipcode: 49309
p-value: 0.14237302731657203
Fail to reject the null hypothesis. Data is not stationary.

ADFuller test p-value for zipcode: 40107
p-value: 0.9550929842851106
Fail to reject the null hypothesis. Data is not stationary.

ADFuller test p-value for zipcode: 48822
p-value: 0.06934714490501559
Fail to reject the null hypothesis. Data is not stationary.

ADFuller test p-value for zipcode: 49265
p-value: 0.12368703580975493
Fail to reject the null hypothesis. Data is not stationary.

ADFuller test p-value for zipcode: 49425
p-value: 0.34712161138449865
Fail to reject the null hypothesis. Data is not stationary.

ADFuller test p-value for zipcode: 29645
p-value: 0.6064933101422972
Fail to reject the null hypothesis. Data is not stationary.

ADFuller test p-value for zipcode: 66206
p-value: 0.9938595718454561
Fail to reject the null hypothesis. Data is not stationary.

ADFuller test p-value for zipcode: 48835
p-value: 0.05193835593732367
Fail to reject the null hypothesis. Data is not stationary.

ADFuller test p-value for zipcode: 48894
p-value: 0.01662469463284938
Reject the null hypothesis. Data is stationary.

ADFuller test p-value for zipcode: 15486
p-value: 0.21532844657975897
Fail to reject the null hypothesis. Data is not stationary.
```

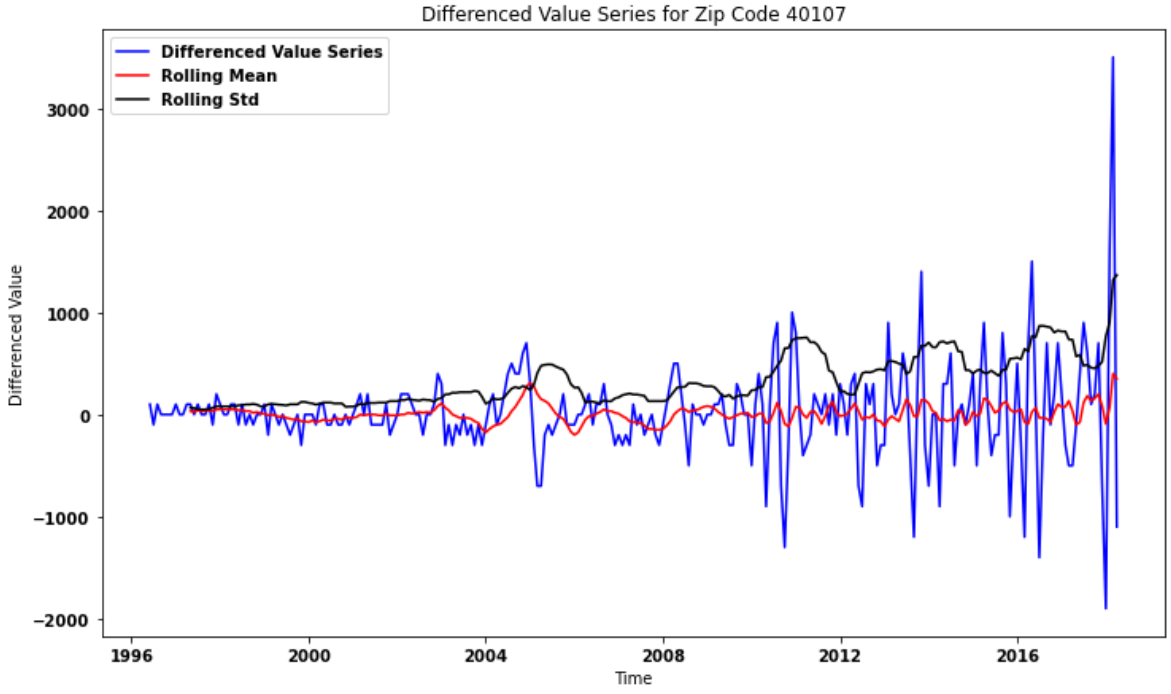
i) Differencing the non-stationary time series and visualizing them

In [28]:
```
# Differencing the non-stationary time series and testing for stationarity ag
differenced_adf_results = []

for i in range(10):
    # Differencing the time series
    dfs_ts[i]['differenced_ret'] = dfs_ts[i]['ret'].diff().dropna()
    dfs_ts[i]['differenced_ret'] = dfs_ts[i]['differenced_ret'].diff().dropna

    # Get the rolling mean and std
    rolmean = dfs_ts[i]['differenced_ret'].rolling(window=12, center=False).m
    rolstd = dfs_ts[i]['differenced_ret'].rolling(window=12, center=False).st

    # Conducting the ADF test on the differenced series
    results = adfuller(dfs_ts[i]['differenced_ret'].dropna())
    zipcode = dfs_ts[i]['Zipcode'].iloc[0]
    p_value = results[1]
    differenced_adf_results.append((zipcode, p_value))

    # Displaying the results
    print(f'ADFuller test p-value for differenced series of zipcode: {zipcode
    print('p-value:', p_value)
    if p_value > 0.05:
        print('Fail to reject the null hypothesis. Differenced data is not st
    else:
        print('Reject the null hypothesis. Differenced data is stationary.\n'

    # Plotting the differenced time series
    plt.figure(figsize=(12, 7))
    plt.plot(dfs_ts[i].index, dfs_ts[i]['differenced_ret'], color='blue', lab
    plt.plot(rolmean, label='Rolling Mean', color='red')
    plt.plot(rolstd, label='Rolling Std', color='black')
    plt.title(f'Differenced Value Series for Zip Code {zipcode}')
    plt.xlabel('Time')
    plt.ylabel('Differenced Value')
    plt.legend()
    plt.show()

differenced_adf_results
```

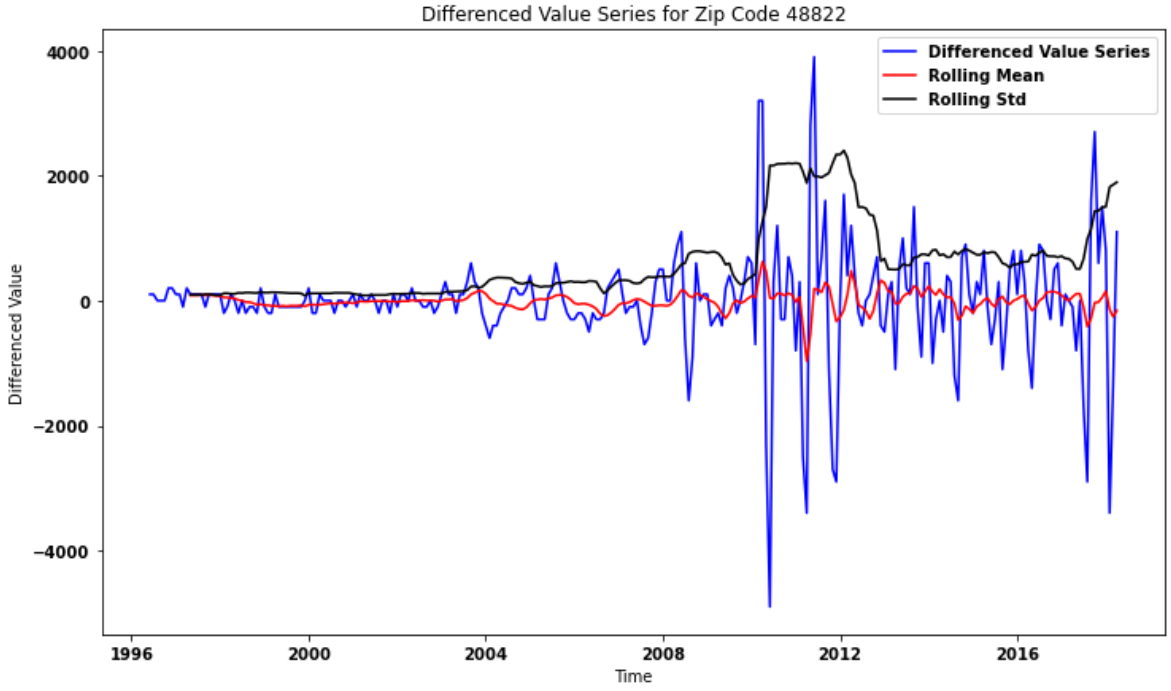
```
ADFuller test p-value for differenced series of zipcode: 49309
p-value: 0.00023355632659848946
Reject the null hypothesis. Differenced data is stationary.
```



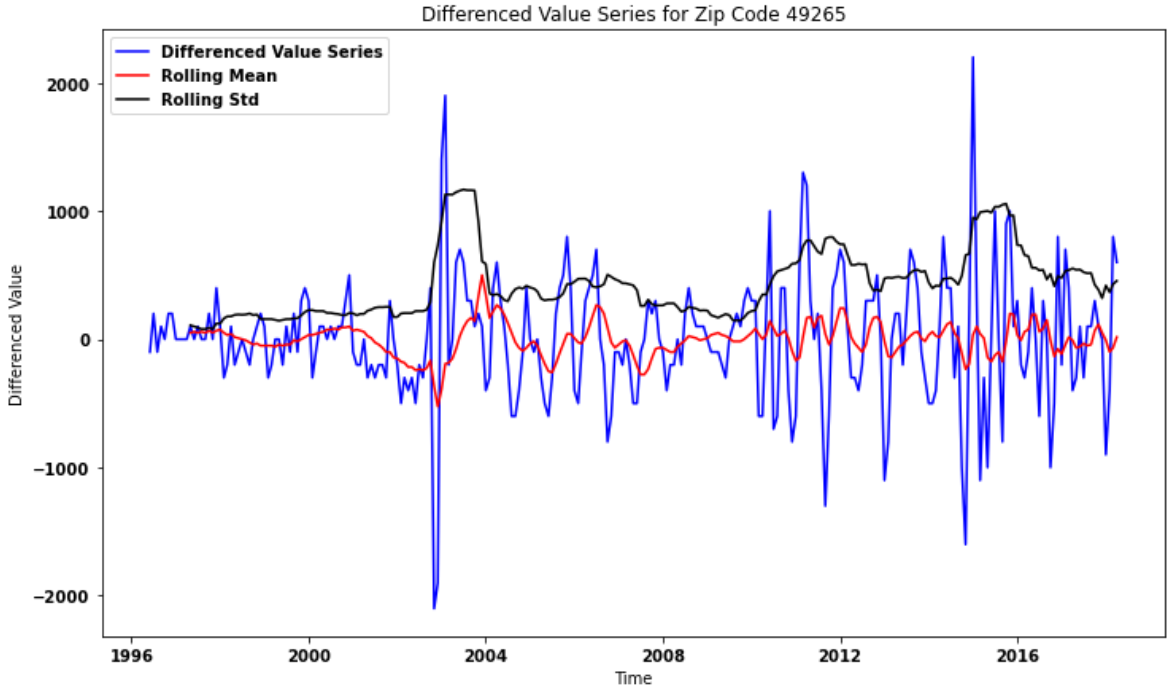

```
ADFuller test p-value for differenced series of zipcode: 40107
p-value: 1.6566222819649275e-09
Reject the null hypothesis. Differenced data is stationary.
```



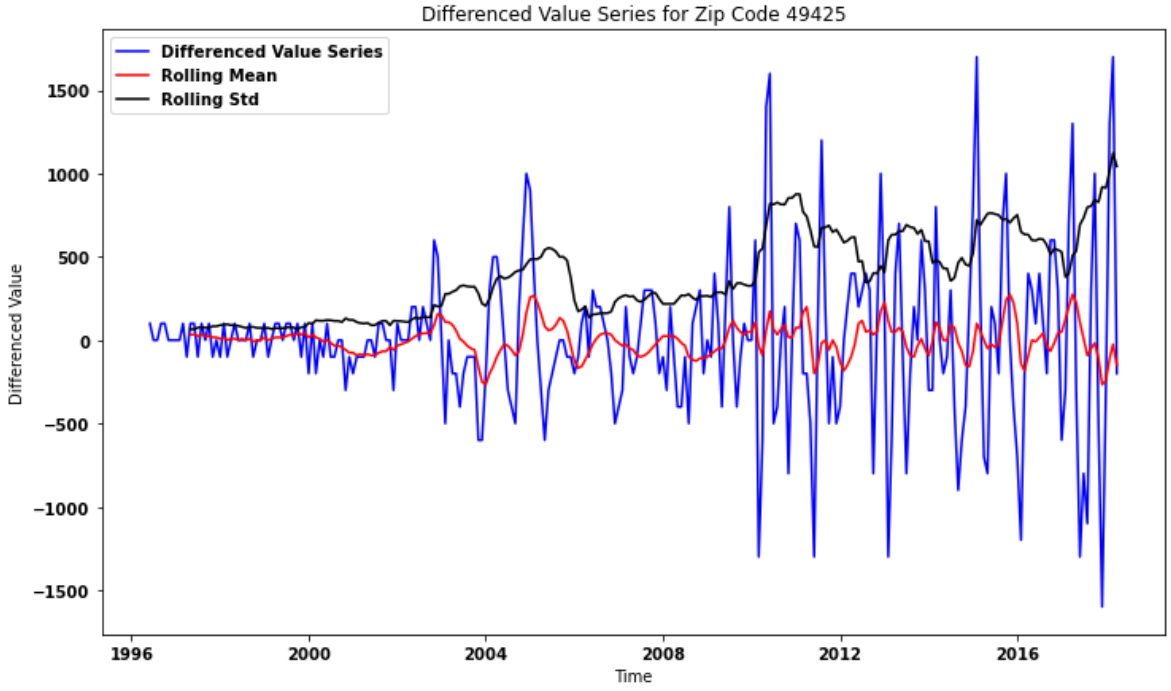

```
ADFuller test p-value for differenced series of zipcode: 48822
p-value: 1.6875246856621119e-12
Reject the null hypothesis. Differenced data is stationary.
```



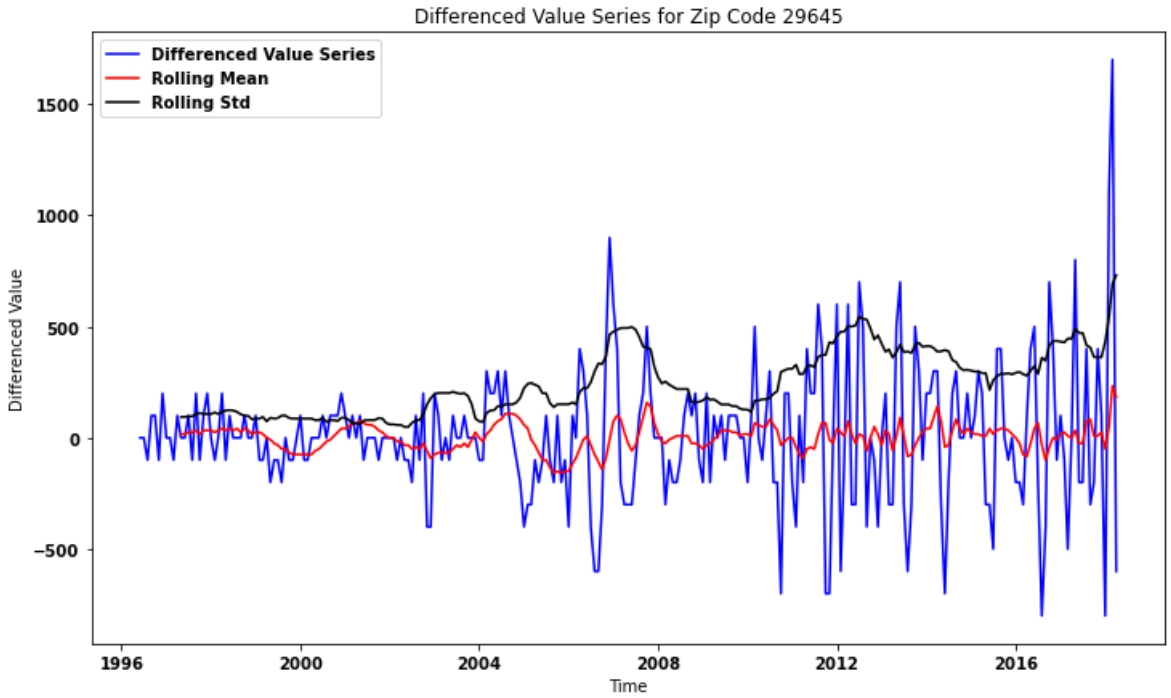

```
ADFuller test p-value for differenced series of zipcode: 49265
p-value: 1.7895122065251984e-05
Reject the null hypothesis. Differenced data is stationary.
```



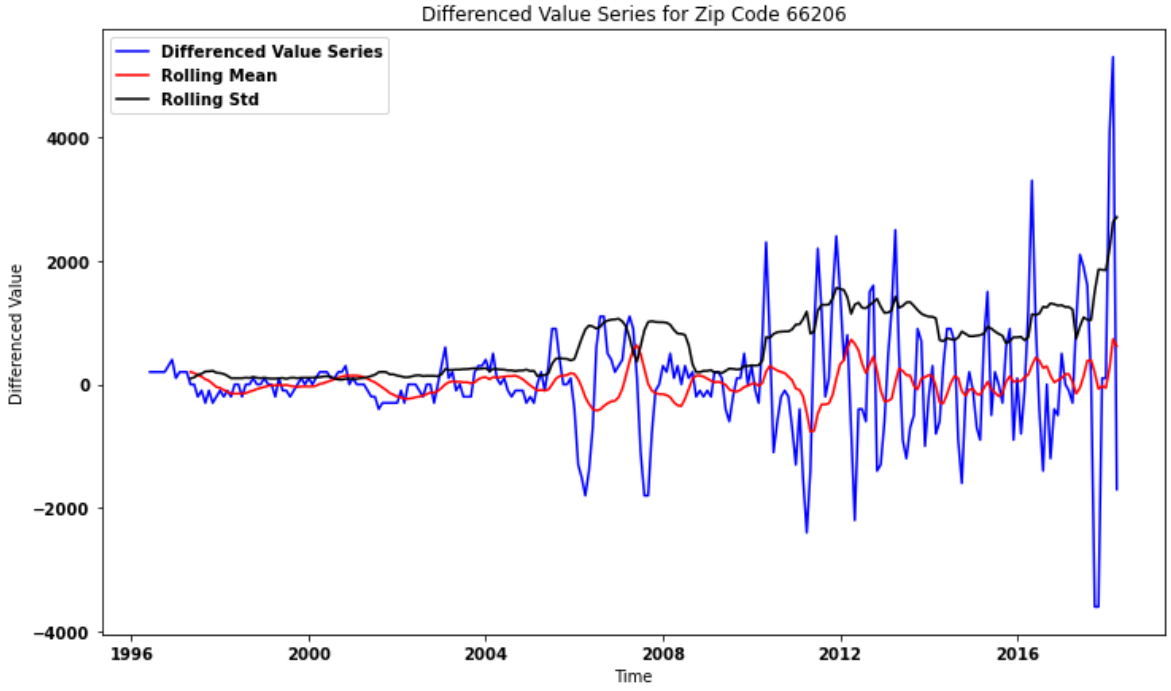

```
ADFuller test p-value for differenced series of zipcode: 49425
p-value: 1.7214650193725787e-11
Reject the null hypothesis. Differenced data is stationary.
```



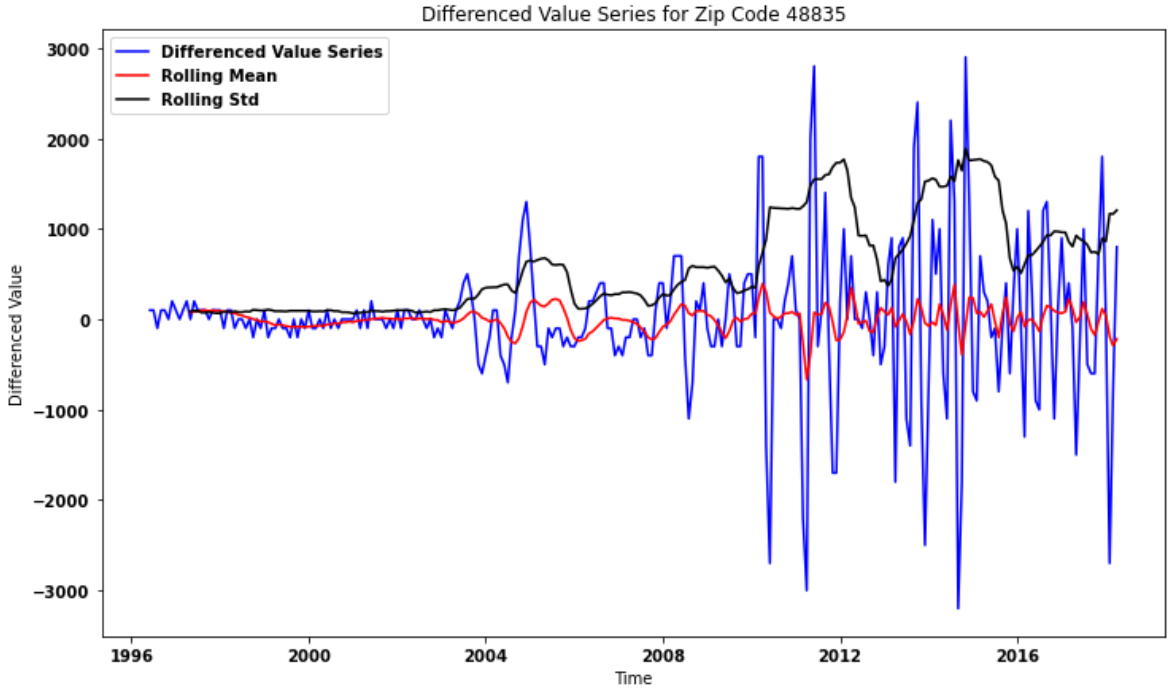

```
ADFuller test p-value for differenced series of zipcode: 29645
p-value: 3.089904083016291e-13
Reject the null hypothesis. Differenced data is stationary.
```



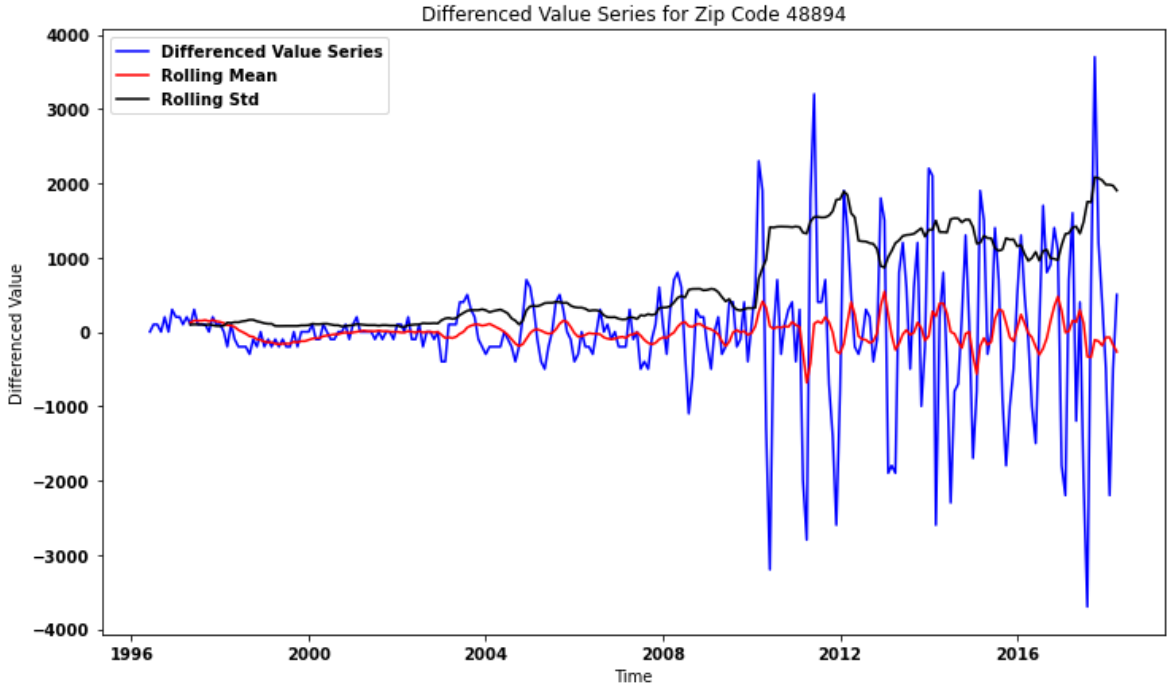

```
ADFuller test p-value for differenced series of zipcode: 66206
p-value: 2.2419500316558596e-07
Reject the null hypothesis. Differenced data is stationary.
```



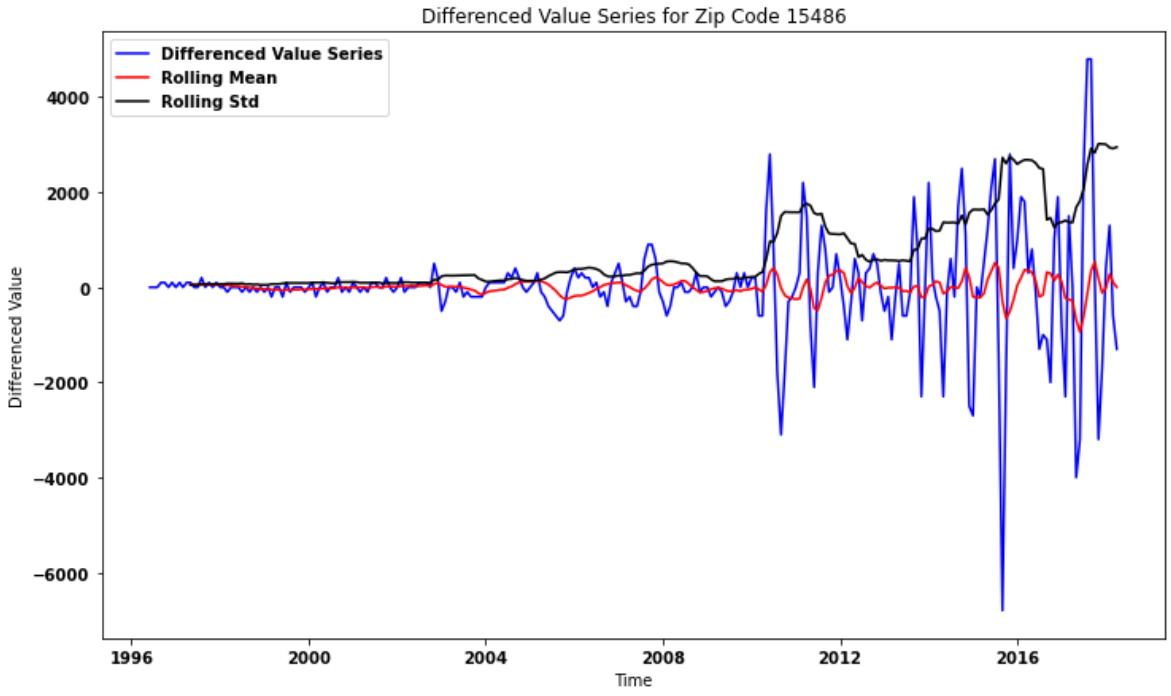

```
ADFuller test p-value for differenced series of zipcode: 48835
p-value: 1.4601641153387778e-10
Reject the null hypothesis. Differenced data is stationary.
```




```
ADFuller test p-value for differenced series of zipcode: 48894
p-value: 2.7423702137801032e-14
Reject the null hypothesis. Differenced data is stationary.
```




```
ADFuller test p-value for differenced series of zipcode: 15486
p-value: 5.333483230652911e-14
Reject the null hypothesis. Differenced data is stationary.
```



Out[28]:
```
[(49309, 0.00023355632659848946),
 (40107, 1.6566222819649275e-09),
 (48822, 1.6875246856621119e-12),
 (49265, 1.7895122065251984e-05),
 (49425, 1.7214650193725787e-11),
 (29645, 3.089904083016291e-13),
 (66206, 2.2419500316558596e-07),
 (48835, 1.4601641153387778e-10),
 (48894, 2.7423702137801032e-14),
 (15486, 5.333483230652911e-14)]
```

- By applying differencing, we successfully transformed the non-stationary time series data into stationary data for all of our top 10 zip codes.
- We will go ahead and plot their ACF and PACF plots, which will guide us on how to pass in our `p,d,q` parameters when fitting our time series models.

In [29]:
```
# The dataframes are contained in a list
dfs_ts
```

Out[29]:
```
[            RegionID  Zipcode    City State  SizeRank       ROI        CV
\
time
1996-04-01     79678    49309  Bitely    MI     13315  3.353982  0.231683
1996-05-01     79678    49309  Bitely    MI     13315  3.353982  0.231683
1996-06-01     79678    49309  Bitely    MI     13315  3.353982  0.231683
1996-07-01     79678    49309  Bitely    MI     13315  3.353982  0.231683
1996-08-01     79678    49309  Bitely    MI     13315  3.353982  0.231683
...              ...      ...     ...   ...       ...       ...       ...
2017-12-01     79678    49309  Bitely    MI     13315  3.353982  0.231683
2018-01-01     79678    49309  Bitely    MI     13315  3.353982  0.231683
2018-02-01     79678    49309  Bitely    MI     13315  3.353982  0.231683
2018-03-01     79678    49309  Bitely    MI     13315  3.353982  0.231683
2018-04-01     79678    49309  Bitely    MI     13315  3.353982  0.231683

                    std          mean    value      ret  differenced_ret
time
1996-04-01  8755.699818  37791.698113  11300.0  11300.0             NaN
1996-05-01  8755.699818  37791.698113  11800.0  11800.0             NaN
1996-06-01  8755.699818  37791.698113  12200.0  12200.0          -100.0
1996-07-01  8755.699818  37791.698113  12700.0  12700.0           100.0
1996-08-01  8755.699818  37791.698113  13200.0  13200.0             0.0
...                 ...           ...      ...      ...             ...
2017-12-01  8755.699818  37791.698113  44300.0  44300.0          -500.0
2018-01-01  8755.699818  37791.698113  44100.0  44100.0          -800.0
2018-02-01  8755.699818  37791.698113  44700.0  44700.0           800.0
2018-03-01  8755.699818  37791.698113  46900.0  46900.0          1600.0
2018-04-01  8755.699818  37791.698113  49200.0  49200.0           100.0

[265 rows x 12 columns],
            RegionID  Zipcode    City State  SizeRank      ROI        CV  \
time
1996-04-01     75522    40107  Boston    KY     13357  3.01897  0.231908
1996-05-01     75522    40107  Boston    KY     13357  3.01897  0.231908
1996-06-01     75522    40107  Boston    KY     13357  3.01897  0.231908
1996-07-01     75522    40107  Boston    KY     13357  3.01897  0.231908
1996-08-01     75522    40107  Boston    KY     13357  3.01897  0.231908
...              ...      ...     ...   ...       ...      ...       ...
2017-12-01     75522    40107  Boston    KY     13357  3.01897  0.231908
2018-01-01     75522    40107  Boston    KY     13357  3.01897  0.231908
2018-02-01     75522    40107  Boston    KY     13357  3.01897  0.231908
2018-03-01     75522    40107  Boston    KY     13357  3.01897  0.231908
2018-04-01     75522    40107  Boston    KY     13357  3.01897  0.231908

                    std          mean     value       ret  differenced_ret
time
1996-04-01  21638.431224  93306.037736   36900.0   36900.0             NaN
1996-05-01  21638.431224  93306.037736   37300.0   37300.0             NaN
1996-06-01  21638.431224  93306.037736   37800.0   37800.0           100.0
1996-07-01  21638.431224  93306.037736   38200.0   38200.0          -100.0
1996-08-01  21638.431224  93306.037736   38700.0   38700.0           100.0
...                 ...           ...       ...       ...             ...
2017-12-01  21638.431224  93306.037736  135200.0  135200.0          -700.0
2018-01-01  21638.431224  93306.037736  135800.0  135800.0         -1900.0
2018-02-01  21638.431224  93306.037736  138000.0  138000.0          1600.0
2018-03-01  21638.431224  93306.037736  143700.0  143700.0          3500.0
2018-04-01  21638.431224  93306.037736  148300.0  148300.0         -1100.0
```

```
 [265 rows x 12 columns],
             RegionID  Zipcode   City State  SizeRank       ROI        CV  \
 time
 1996-04-01     79397    48822  Eagle    MI     12964  2.963519  0.213312
 1996-05-01     79397    48822  Eagle    MI     12964  2.963519  0.213312
 1996-06-01     79397    48822  Eagle    MI     12964  2.963519  0.213312
 1996-07-01     79397    48822  Eagle    MI     12964  2.963519  0.213312
 1996-08-01     79397    48822  Eagle    MI     12964  2.963519  0.213312
 ...              ...      ...    ...   ...       ...       ...       ...
 2017-12-01     79397    48822  Eagle    MI     12964  2.963519  0.213312
 2018-01-01     79397    48822  Eagle    MI     12964  2.963519  0.213312
 2018-02-01     79397    48822  Eagle    MI     12964  2.963519  0.213312
 2018-03-01     79397    48822  Eagle    MI     12964  2.963519  0.213312
 2018-04-01     79397    48822  Eagle    MI     12964  2.963519  0.213312

                     std           mean     value       ret  differenced_re
t
 time
 1996-04-01  30813.609976  144453.207547   46600.0   46600.0              Na
N
 1996-05-01  30813.609976  144453.207547   47800.0   47800.0              Na
N
 1996-06-01  30813.609976  144453.207547   49100.0   49100.0            100.
0
 1996-07-01  30813.609976  144453.207547   50500.0   50500.0            100.
0
 1996-08-01  30813.609976  144453.207547   51900.0   51900.0              0.
0
 ...                 ...           ...       ...       ...
...
 2017-12-01  30813.609976  144453.207547  182600.0  182600.0           1500.
0
 2018-01-01  30813.609976  144453.207547  186100.0  186100.0            800.
0
 2018-02-01  30813.609976  144453.207547  186200.0  186200.0          -3400.
0
 2018-03-01  30813.609976  144453.207547  184900.0  184900.0          -1400.
0
 2018-04-01  30813.609976  144453.207547  184700.0  184700.0           1100.
0

 [265 rows x 12 columns],
             RegionID  Zipcode    City State  SizeRank       ROI        CV
\
 time
 1996-04-01     79648    49265  Onsted    MI     10691  2.931034  0.232729
 1996-05-01     79648    49265  Onsted    MI     10691  2.931034  0.232729
 1996-06-01     79648    49265  Onsted    MI     10691  2.931034  0.232729
 1996-07-01     79648    49265  Onsted    MI     10691  2.931034  0.232729
 1996-08-01     79648    49265  Onsted    MI     10691  2.931034  0.232729
 ...              ...      ...     ...   ...       ...       ...       ...
 2017-12-01     79648    49265  Onsted    MI     10691  2.931034  0.232729
 2018-01-01     79648    49265  Onsted    MI     10691  2.931034  0.232729
 2018-02-01     79648    49265  Onsted    MI     10691  2.931034  0.232729
 2018-03-01     79648    49265  Onsted    MI     10691  2.931034  0.232729
 2018-04-01     79648    49265  Onsted    MI     10691  2.931034  0.232729
```

```
                     std           mean     value       ret  differenced_re
t
 time
 1996-04-01  28923.199467  124278.490566   40600.0   40600.0              Na
N
 1996-05-01  28923.199467  124278.490566   41600.0   41600.0              Na
N
 1996-06-01  28923.199467  124278.490566   42500.0   42500.0           -100.
0
 1996-07-01  28923.199467  124278.490566   43600.0   43600.0            200.
0
 1996-08-01  28923.199467  124278.490566   44600.0   44600.0           -100.
0
 ...                  ...           ...       ...       ...
...
 2017-12-01  28923.199467  124278.490566  157800.0  157800.0              0.
0
 2018-01-01  28923.199467  124278.490566  158000.0  158000.0           -900.
0
 2018-02-01  28923.199467  124278.490566  157800.0  157800.0           -400.
0
 2018-03-01  28923.199467  124278.490566  158400.0  158400.0            800.
0
 2018-04-01  28923.199467  124278.490566  159600.0  159600.0            600.
0

 [265 rows x 12 columns],
             RegionID  Zipcode    City State  SizeRank       ROI        CV
\
 time
 1996-04-01     79742    49425  Holton    MI     12239  2.903614  0.230111
 1996-05-01     79742    49425  Holton    MI     12239  2.903614  0.230111
 1996-06-01     79742    49425  Holton    MI     12239  2.903614  0.230111
 1996-07-01     79742    49425  Holton    MI     12239  2.903614  0.230111
 1996-08-01     79742    49425  Holton    MI     12239  2.903614  0.230111
 ...              ...      ...     ...   ...       ...       ...       ...
 2017-12-01     79742    49425  Holton    MI     12239  2.903614  0.230111
 2018-01-01     79742    49425  Holton    MI     12239  2.903614  0.230111
 2018-02-01     79742    49425  Holton    MI     12239  2.903614  0.230111
 2018-03-01     79742    49425  Holton    MI     12239  2.903614  0.230111
 2018-04-01     79742    49425  Holton    MI     12239  2.903614  0.230111

                    std          mean    value      ret  differenced_ret
 time
 1996-04-01  15550.05849  67576.226415  24900.0  24900.0              NaN
 1996-05-01  15550.05849  67576.226415  25200.0  25200.0              NaN
 1996-06-01  15550.05849  67576.226415  25600.0  25600.0            100.0
 1996-07-01  15550.05849  67576.226415  26000.0  26000.0              0.0
 1996-08-01  15550.05849  67576.226415  26400.0  26400.0              0.0
 ...                 ...           ...      ...      ...              ...
 2017-12-01  15550.05849  67576.226415  96100.0  96100.0          -1600.0
 2018-01-01  15550.05849  67576.226415  94600.0  94600.0           -300.0
 2018-02-01  15550.05849  67576.226415  94400.0  94400.0           1300.0
 2018-03-01  15550.05849  67576.226415  95900.0  95900.0           1700.0
 2018-04-01  15550.05849  67576.226415  97200.0  97200.0           -200.0
```

```
 [265 rows x 12 columns],
            RegionID  Zipcode        City State  SizeRank       ROI
CV  \
 time
 1996-04-01     70626    29645  Gray Court    SC      8354  2.883333  0.2321
48
 1996-05-01     70626    29645  Gray Court    SC      8354  2.883333  0.2321
48
 1996-06-01     70626    29645  Gray Court    SC      8354  2.883333  0.2321
48
 1996-07-01     70626    29645  Gray Court    SC      8354  2.883333  0.2321
48
 1996-08-01     70626    29645  Gray Court    SC      8354  2.883333  0.2321
48
 ...              ...      ...         ...   ...       ...       ...
...
 2017-12-01     70626    29645  Gray Court    SC      8354  2.883333  0.2321
48
 2018-01-01     70626    29645  Gray Court    SC      8354  2.883333  0.2321
48
 2018-02-01     70626    29645  Gray Court    SC      8354  2.883333  0.2321
48
 2018-03-01     70626    29645  Gray Court    SC      8354  2.883333  0.2321
48
 2018-04-01     70626    29645  Gray Court    SC      8354  2.883333  0.2321
48

                     std          mean     value       ret  differenced_ret
 time
 1996-04-01  19289.795188  83092.830189   30000.0   30000.0             NaN
 1996-05-01  19289.795188  83092.830189   30500.0   30500.0             NaN
 1996-06-01  19289.795188  83092.830189   31000.0   31000.0             0.0
 1996-07-01  19289.795188  83092.830189   31500.0   31500.0             0.0
 1996-08-01  19289.795188  83092.830189   31900.0   31900.0          -100.0
 ...                  ...           ...       ...       ...             ...
 2017-12-01  19289.795188  83092.830189  109600.0  109600.0           100.0
 2018-01-01  19289.795188  83092.830189  109800.0  109800.0          -800.0
 2018-02-01  19289.795188  83092.830189  111100.0  111100.0          1100.0
 2018-03-01  19289.795188  83092.830189  114100.0  114100.0          1700.0
 2018-04-01  19289.795188  83092.830189  116500.0  116500.0          -600.0

 [265 rows x 12 columns],
            RegionID  Zipcode     City State  SizeRank       ROI        CV
\
 time
 1996-04-01     87122    66206  Leawood    KS      8413  2.677966  0.233424
 1996-05-01     87122    66206  Leawood    KS      8413  2.677966  0.233424
 1996-06-01     87122    66206  Leawood    KS      8413  2.677966  0.233424
 1996-07-01     87122    66206  Leawood    KS      8413  2.677966  0.233424
 1996-08-01     87122    66206  Leawood    KS      8413  2.677966  0.233424
 ...              ...      ...      ...   ...       ...       ...       ...
 2017-12-01     87122    66206  Leawood    KS      8413  2.677966  0.233424
 2018-01-01     87122    66206  Leawood    KS      8413  2.677966  0.233424
 2018-02-01     87122    66206  Leawood    KS      8413  2.677966  0.233424
 2018-03-01     87122    66206  Leawood    KS      8413  2.677966  0.233424
 2018-04-01     87122    66206  Leawood    KS      8413  2.677966  0.233424
```

```
                      std           mean     value       ret  differenced_re
t
 time
 1996-04-01  65332.460851  279887.169811  123900.0  123900.0             Na
N
 1996-05-01  65332.460851  279887.169811  124900.0  124900.0             Na
N
 1996-06-01  65332.460851  279887.169811  126100.0  126100.0           200.
0
 1996-07-01  65332.460851  279887.169811  127500.0  127500.0           200.
0
 1996-08-01  65332.460851  279887.169811  129100.0  129100.0           200.
0
 ...                  ...            ...       ...       ...
...
 2017-12-01  65332.460851  279887.169811  427700.0  427700.0           100.
0
 2018-01-01  65332.460851  279887.169811  429400.0  429400.0           100.
0
 2018-02-01  65332.460851  279887.169811  435200.0  435200.0          4100.
0
 2018-03-01  65332.460851  279887.169811  446300.0  446300.0          5300.
0
 2018-04-01  65332.460851  279887.169811  455700.0  455700.0         -1700.
0

 [265 rows x 12 columns],
             RegionID  Zipcode    City State  SizeRank       ROI        CV
\
 time
 1996-04-01     79409    48835  Fowler    MI     13358  2.590674  0.214027
 1996-05-01     79409    48835  Fowler    MI     13358  2.590674  0.214027
 1996-06-01     79409    48835  Fowler    MI     13358  2.590674  0.214027
 1996-07-01     79409    48835  Fowler    MI     13358  2.590674  0.214027
 1996-08-01     79409    48835  Fowler    MI     13358  2.590674  0.214027
 ...              ...      ...     ...   ...       ...       ...       ...
 2017-12-01     79409    48835  Fowler    MI     13358  2.590674  0.214027
 2018-01-01     79409    48835  Fowler    MI     13358  2.590674  0.214027
 2018-02-01     79409    48835  Fowler    MI     13358  2.590674  0.214027
 2018-03-01     79409    48835  Fowler    MI     13358  2.590674  0.214027
 2018-04-01     79409    48835  Fowler    MI     13358  2.590674  0.214027

                     std           mean     value       ret  differenced_ret
 time
 1996-04-01  23191.80072  108359.245283   38600.0   38600.0             NaN
 1996-05-01  23191.80072  108359.245283   39000.0   39000.0             NaN
 1996-06-01  23191.80072  108359.245283   39500.0   39500.0           100.0
 1996-07-01  23191.80072  108359.245283   40100.0   40100.0           100.0
 1996-08-01  23191.80072  108359.245283   40600.0   40600.0          -100.0
 ...                 ...            ...       ...       ...             ...
 2017-12-01  23191.80072  108359.245283  137500.0  137500.0          1800.0
 2018-01-01  23191.80072  108359.245283  140000.0  140000.0             0.0
 2018-02-01  23191.80072  108359.245283  139800.0  139800.0         -2700.0
 2018-03-01  23191.80072  108359.245283  138800.0  138800.0          -800.0
 2018-04-01  23191.80072  108359.245283  138600.0  138600.0           800.0

 [265 rows x 12 columns],
```

```
             RegionID  Zipcode        City State  SizeRank       ROI
CV  \
 time
 1996-04-01     79465    48894  Westphalia    MI     13754  2.561947  0.1901
11
 1996-05-01     79465    48894  Westphalia    MI     13754  2.561947  0.1901
11
 1996-06-01     79465    48894  Westphalia    MI     13754  2.561947  0.1901
11
 1996-07-01     79465    48894  Westphalia    MI     13754  2.561947  0.1901
11
 1996-08-01     79465    48894  Westphalia    MI     13754  2.561947  0.1901
11
 ...              ...      ...         ...   ...       ...       ...
...
 2017-12-01     79465    48894  Westphalia    MI     13754  2.561947  0.1901
11
 2018-01-01     79465    48894  Westphalia    MI     13754  2.561947  0.1901
11
 2018-02-01     79465    48894  Westphalia    MI     13754  2.561947  0.1901
11
 2018-03-01     79465    48894  Westphalia    MI     13754  2.561947  0.1901
11
 2018-04-01     79465    48894  Westphalia    MI     13754  2.561947  0.1901
11

                     std           mean     value       ret  differenced_re
t
 time
 1996-04-01  24428.957688  128498.113208   45200.0   45200.0             Na
N
 1996-05-01  24428.957688  128498.113208   46000.0   46000.0             Na
N
 1996-06-01  24428.957688  128498.113208   46800.0   46800.0              0.
0
 1996-07-01  24428.957688  128498.113208   47700.0   47700.0            100.
0
 1996-08-01  24428.957688  128498.113208   48700.0   48700.0            100.
0
 ...                  ...            ...       ...       ...
...
 2017-12-01  24428.957688  128498.113208  160900.0  160900.0            400.
0
 2018-01-01  24428.957688  128498.113208  162700.0  162700.0           -500.
0
 2018-02-01  24428.957688  128498.113208  162300.0  162300.0          -2200.
0
 2018-03-01  24428.957688  128498.113208  161400.0  161400.0           -500.
0
 2018-04-01  24428.957688  128498.113208  161000.0  161000.0            500.
0

 [265 rows x 12 columns],
             RegionID  Zipcode      City State  SizeRank      ROI       CV
\
 time
 1996-04-01     64135    15486  Franklin    PA     13283  2.53125  0.23527
```

```
1996-05-01     64135    15486  Franklin    PA     13283  2.53125  0.23527
1996-06-01     64135    15486  Franklin    PA     13283  2.53125  0.23527
1996-07-01     64135    15486  Franklin    PA     13283  2.53125  0.23527
1996-08-01     64135    15486  Franklin    PA     13283  2.53125  0.23527
...              ...      ...       ...   ...       ...      ...      ...
2017-12-01     64135    15486  Franklin    PA     13283  2.53125  0.23527
2018-01-01     64135    15486  Franklin    PA     13283  2.53125  0.23527
2018-02-01     64135    15486  Franklin    PA     13283  2.53125  0.23527
2018-03-01     64135    15486  Franklin    PA     13283  2.53125  0.23527
2018-04-01     64135    15486  Franklin    PA     13283  2.53125  0.23527

                    std         mean     value       ret  differenced_ret
time
1996-04-01  20749.032919  88192.45283   35200.0   35200.0              NaN
1996-05-01  20749.032919  88192.45283   35500.0   35500.0              NaN
1996-06-01  20749.032919  88192.45283   35800.0   35800.0              0.0
1996-07-01  20749.032919  88192.45283   36100.0   36100.0              0.0
1996-08-01  20749.032919  88192.45283   36400.0   36400.0              0.0
...                  ...          ...       ...       ...              ...
2017-12-01  20749.032919  88192.45283  120500.0  120500.0          -1700.0
2018-01-01  20749.032919  88192.45283  121100.0  121100.0            400.0
2018-02-01  20749.032919  88192.45283  123000.0  123000.0           1300.0
2018-03-01  20749.032919  88192.45283  124300.0  124300.0           -600.0
2018-04-01  20749.032919  88192.45283  124300.0  124300.0          -1300.0

[265 rows x 12 columns]]
```

In [30]:
```
# Concatenate the list of DataFrames into a single DataFrame
df_combined = pd.concat(dfs_ts)

# Print the resulting DataFrame
df_combined
```

Out[30]:

time	RegionID	Zipcode	City	State	SizeRank	ROI	CV	std	r
1996-04-01	79678	49309	Bitely	MI	13315	3.353982	0.231683	8755.699818	37791.69
1996-05-01	79678	49309	Bitely	MI	13315	3.353982	0.231683	8755.699818	37791.69
1996-06-01	79678	49309	Bitely	MI	13315	3.353982	0.231683	8755.699818	37791.69
1996-07-01	79678	49309	Bitely	MI	13315	3.353982	0.231683	8755.699818	37791.69
1996-08-01	79678	49309	Bitely	MI	13315	3.353982	0.231683	8755.699818	37791.69
...	...	...	...	...	...	...	...	...	
2017-12-01	64135	15486	Franklin	PA	13283	2.531250	0.235270	20749.032919	88192.45
2018-01-01	64135	15486	Franklin	PA	13283	2.531250	0.235270	20749.032919	88192.45
2018-02-01	64135	15486	Franklin	PA	13283	2.531250	0.235270	20749.032919	88192.45
2018-03-01	64135	15486	Franklin	PA	13283	2.531250	0.235270	20749.032919	88192.45
2018-04-01	64135	15486	Franklin	PA	13283	2.531250	0.235270	20749.032919	88192.45

2650 rows × 12 columns

In [31]:
```
# Create a dictionary to store dataframes for each zipcode
zipcode_dataframes = {}

# Extract unique zipcodes from the dataframes
unique_zipcodes = df_combined['Zipcode'].unique()

# Iterate over unique zipcodes and create dataframes for each
for zipcode in unique_zipcodes:
    # Filter dataframe for the current zipcode
    zipcode_df = df_combined[df_combined['Zipcode'] == zipcode]

    # Drop rows with null values
    zipcode_df = zipcode_df.dropna()

    # Store the dataframe in the dictionary
    zipcode_dataframes[zipcode] = zipcode_df

# Access dataframe for a specific zipcode
# [49309, 40107, 48822, 49265, 49425, 29645, 66206, 48835, 48894, 15486]
zipcode_49309_df = zipcode_dataframes[49309]
zipcode_40107_df = zipcode_dataframes[40107]
zipcode_48822_df = zipcode_dataframes[48822]
zipcode_49265_df = zipcode_dataframes[49265]
zipcode_49425_df = zipcode_dataframes[49425]
zipcode_29645_df = zipcode_dataframes[29645]
zipcode_66206_df = zipcode_dataframes[66206]
zipcode_48835_df = zipcode_dataframes[48835]
zipcode_48894_df = zipcode_dataframes[48894]
zipcode_15486_df = zipcode_dataframes[15486]


# Confirm if it has worked
zipcode_15486_df
```

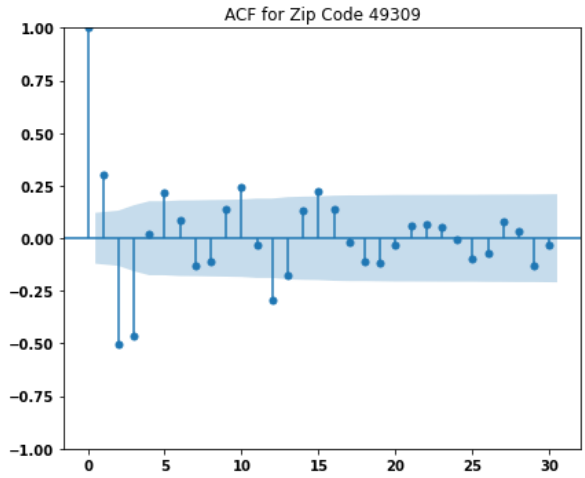
Out[31]:

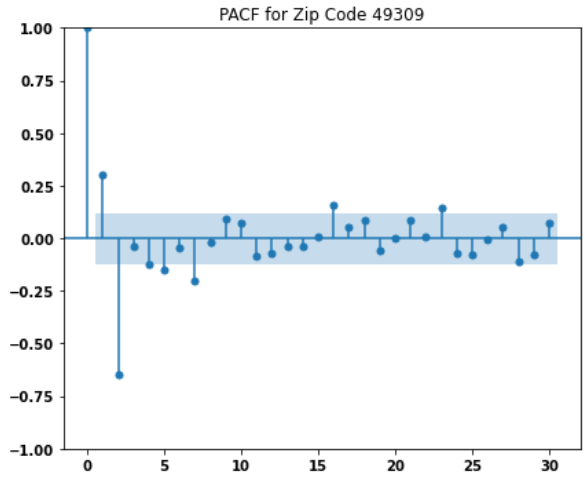
time	RegionID	Zipcode	City	State	SizeRank	ROI	CV	std	mea
1996-06-01	64135	15486	Franklin	PA	13283	2.53125	0.23527	20749.032919	88192.4528
1996-07-01	64135	15486	Franklin	PA	13283	2.53125	0.23527	20749.032919	88192.4528
1996-08-01	64135	15486	Franklin	PA	13283	2.53125	0.23527	20749.032919	88192.4528
1996-09-01	64135	15486	Franklin	PA	13283	2.53125	0.23527	20749.032919	88192.4528
1996-10-01	64135	15486	Franklin	PA	13283	2.53125	0.23527	20749.032919	88192.4528
...	...	...	...	...	...	...	...	...	.
2017-12-01	64135	15486	Franklin	PA	13283	2.53125	0.23527	20749.032919	88192.4528
2018-01-01	64135	15486	Franklin	PA	13283	2.53125	0.23527	20749.032919	88192.4528
2018-02-01	64135	15486	Franklin	PA	13283	2.53125	0.23527	20749.032919	88192.4528
2018-03-01	64135	15486	Franklin	PA	13283	2.53125	0.23527	20749.032919	88192.4528
2018-04-01	64135	15486	Franklin	PA	13283	2.53125	0.23527	20749.032919	88192.4528

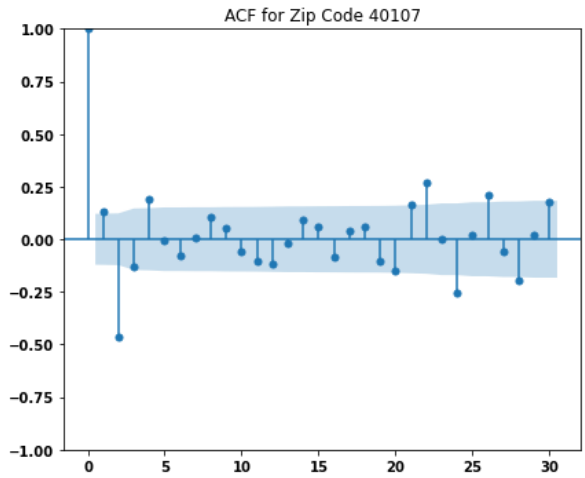
263 rows × 12 columns

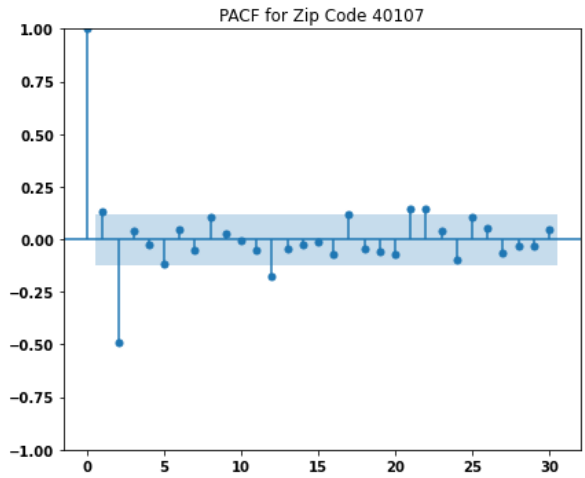
ii) Plotting ACF and PACF

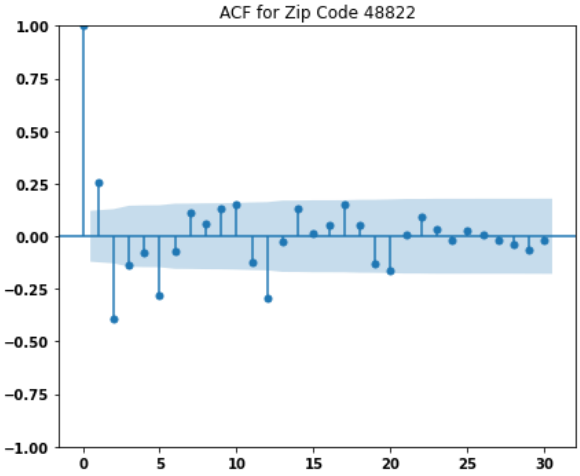
In [32]:
```
# Iterate over the dictionary of zipcode dataframes
for zipcode, zipcode_df in zipcode_dataframes.items():
    # Drop nulls
    zipcode_df.dropna(inplace=True)

    plt.figure(figsize=(12, 5))

    plt.subplot(121)
    plot_acf(zipcode_df['differenced_ret'], lags=30, ax=plt.gca())
    plt.title(f'ACF for Zip Code {zipcode}')

    plt.subplot(122)
    plot_pacf(zipcode_df['differenced_ret'], lags=30, ax=plt.gca())
    plt.title(f'PACF for Zip Code {zipcode}')

    plt.tight_layout()
    plt.show()
```

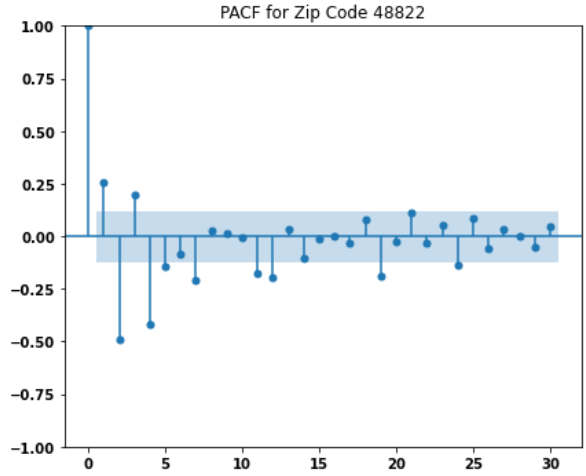


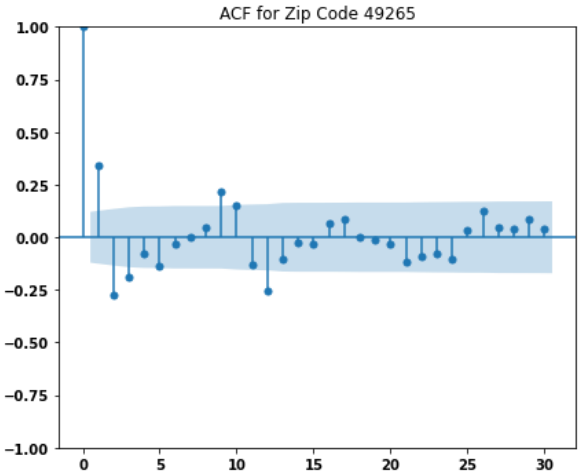


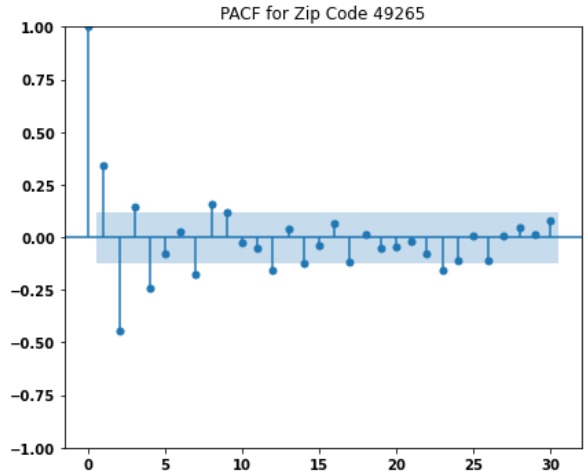


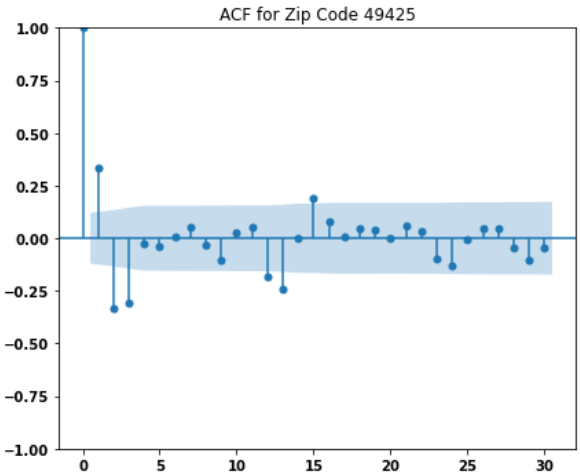


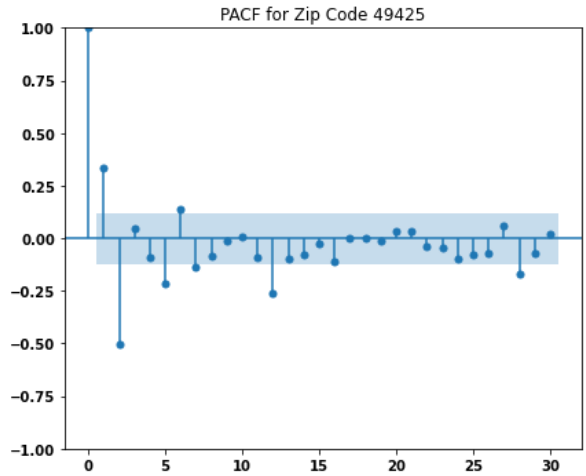

ACF for Zip Code 48822
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
5
10
15
20
25
30

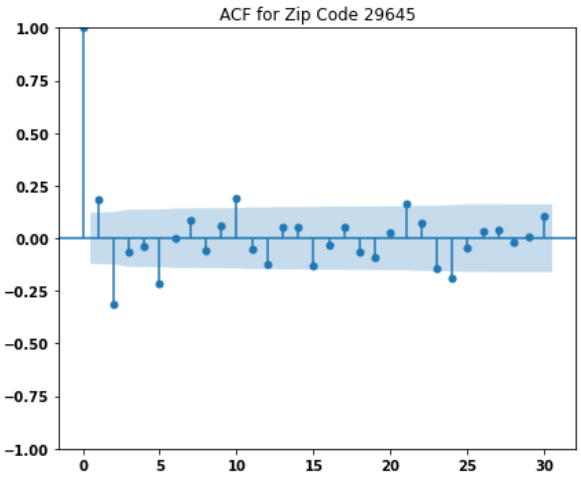

PACF for Zip Code 48822
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
5
10
15
20
25
30

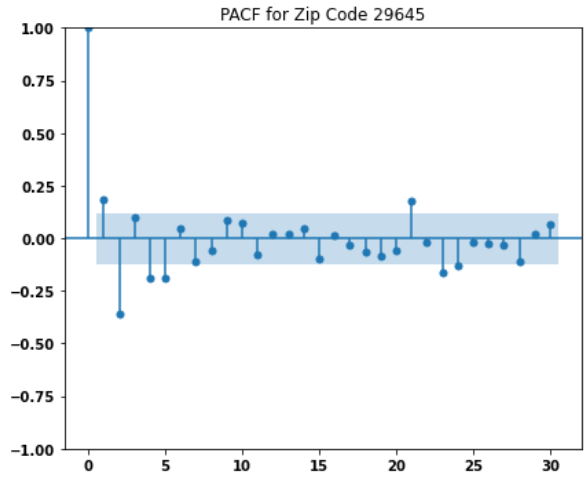

ACF for Zip Code 49265
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
5
10
15
20
25
30

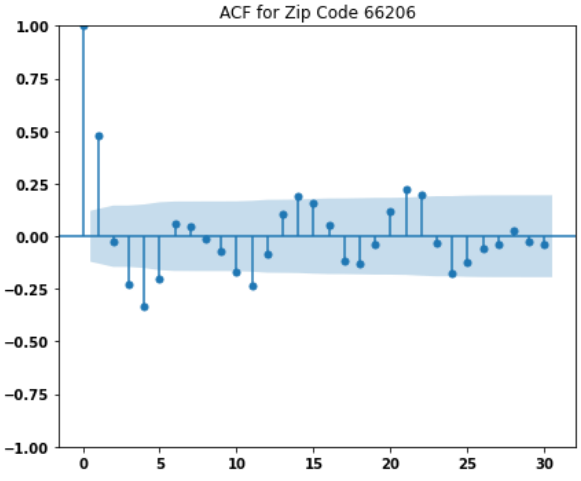

PACF for Zip Code 49265
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
5
10
15
20
25
30

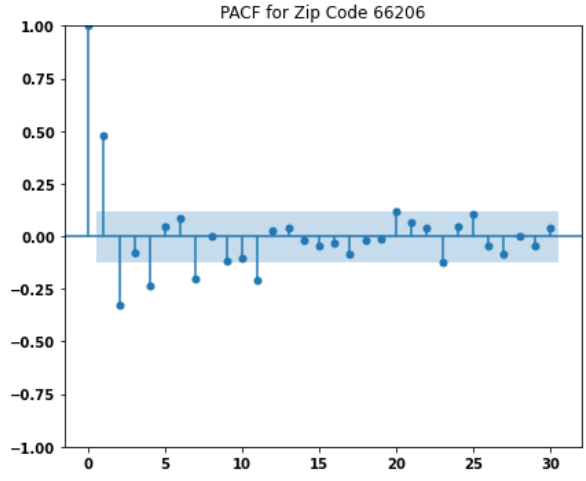

ACF for Zip Code 49425
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
5
10
15
20
25
30

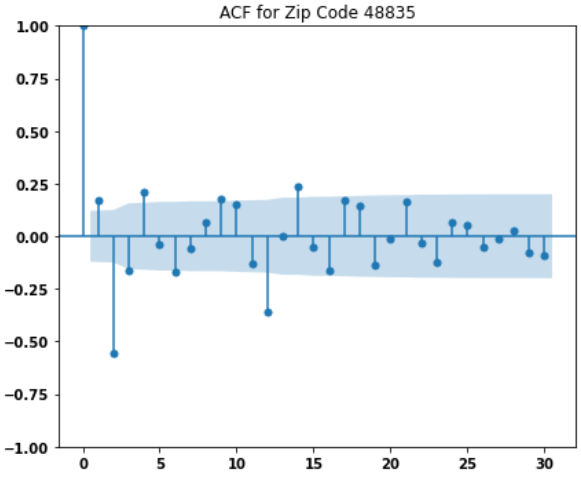

PACF for Zip Code 49425
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
5
10
15
20
25
30

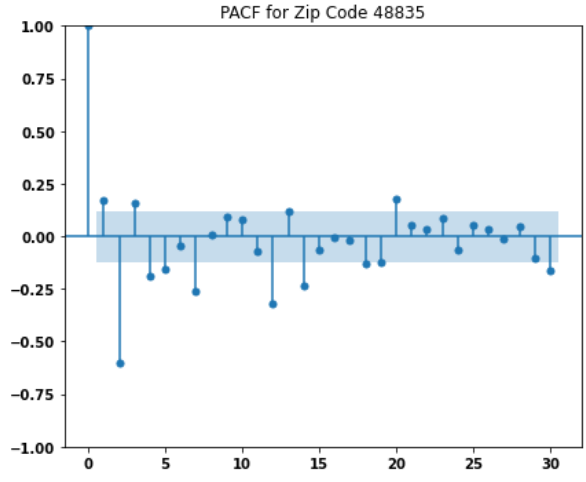

ACF for Zip Code 29645

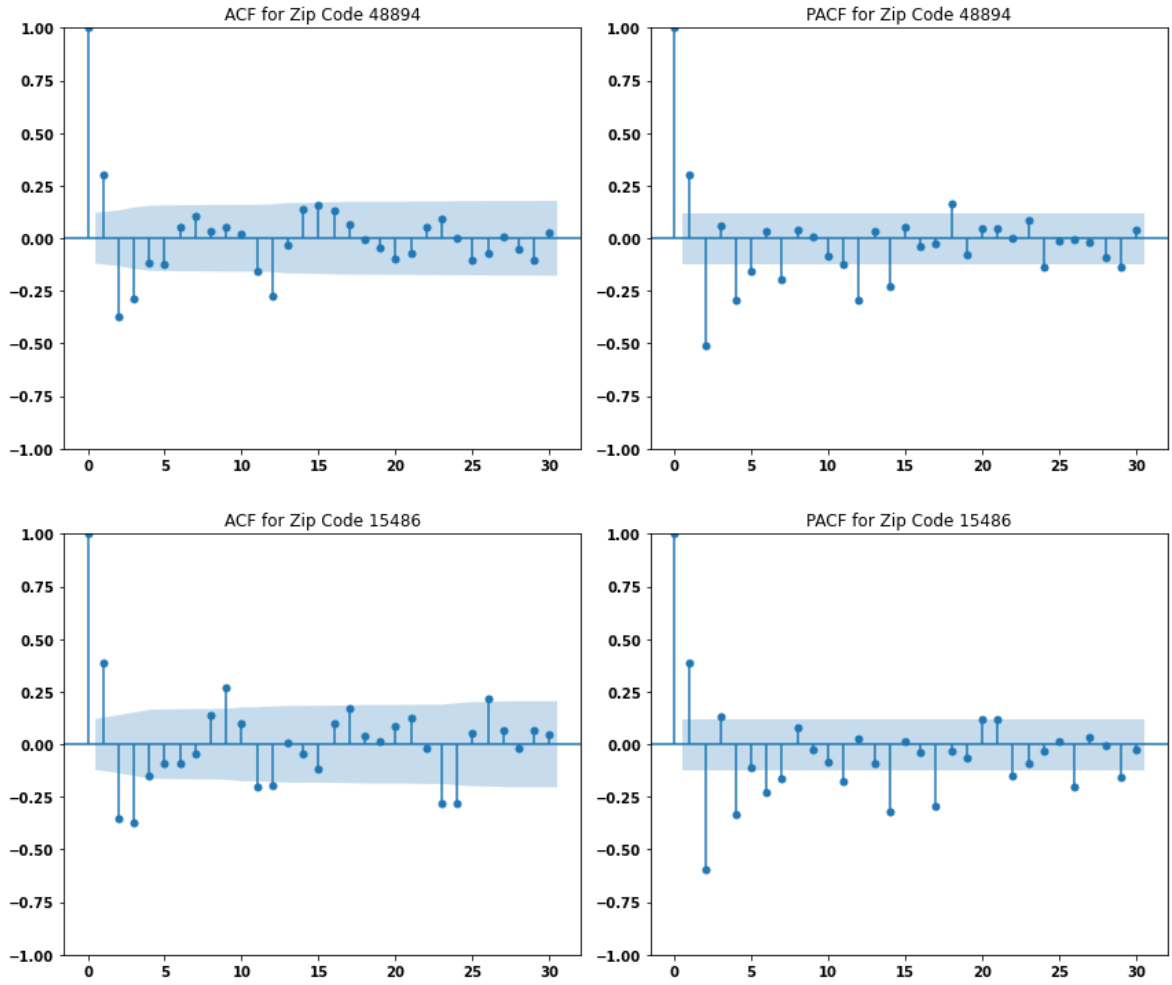

PACF for Zip Code 29645


ACF for Zip Code 66206


PACF for Zip Code 66206


ACF for Zip Code 48835


PACF for Zip Code 48835



- Now we can begin the modeling process with the observations from our ACF and PACF graphs.

5.0 Modeling

5.1 ARIMA modeling

- The following will be the steps for our modelling:

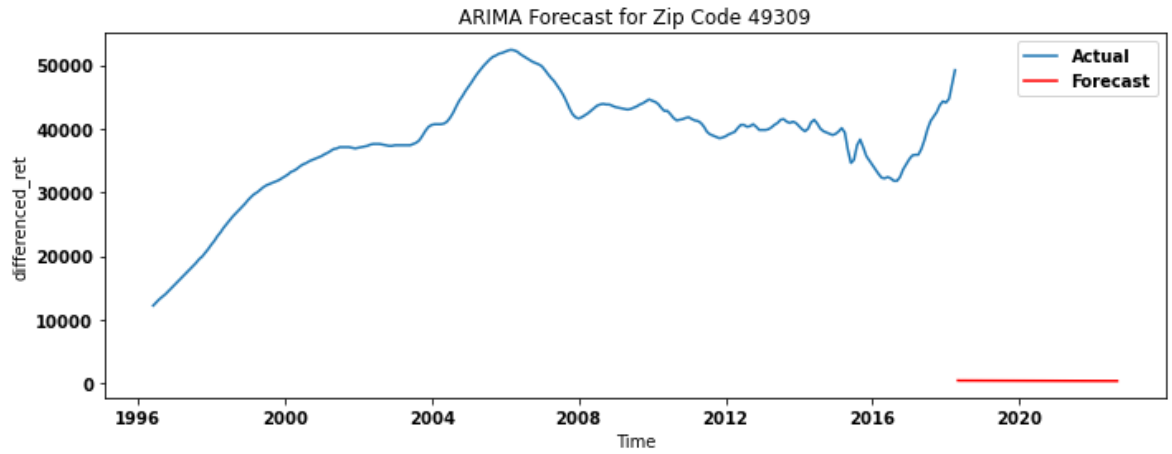
1. Determine the ARIMA parameters (p,d,q) for each zipcode. This involves:
    - p: Lag order (lags of the autoregressive term).
    - d: Degree of differencing.
    - q: Order of moving average.
2. Fit the ARIMA model to the time series data for each zipcode.
3. Forecast future values and assess model performance.

In [33]:
```
# Define a function for ARIMA modeling

def build_arima_model(zipcode_df, column, value, order):
    # Split the data into train and test sets
    train_size = int(0.8 * len(zipcode_df))
    train, test = zipcode_df[:train_size], zipcode_df[train_size:]

    # Fit the ARIMA model
    model = ARIMA(train[column], order=order)
    model_fit = model.fit()

    # Forecast
    forecast_steps = len(test)
    forecast_diff = model_fit.forecast(steps=forecast_steps)  # Forecast on t

    # Calculate the accumulated forecasted values
    #forecast_accumulated = np.cumsum(forecast_diff)

    # Calculate the last observed value in the original time series
    #last_observed_value = zipcode_df['value'][-1]

    # Back-transform the forecasted values to the original scale
    #forecast_original = forecast_accumulated + last_observed_value



    # Back-transform the forecasted differenced series to obtain 'value' serie
    forecast_original = forecast_diff.cumsum() + train[column].iloc[-1]

    # Create a time index for the forecasted values
    forecast_index = pd.date_range(start=test.index[-1], periods=forecast_ste

    # Calculate AIC value
    aic = model_fit.aic

    # Calculate RMSE
    #rmse = np.sqrt(mean_squared_error(test[column], forecast_original))

    # Calculate RMSE using the back-transformed forecasted 'value' series
    rmse = np.sqrt(mean_squared_error(test[column], forecast_original))

    print(f'ARIMA Parameters: p={order[0]}, d={order[1]}, q={order[2]}')
    print(f'RMSE: {rmse}')
    print(f'AIC: {aic}')

    # Plot the original data and the forecasted values in the original scale
    plt.figure(figsize=(10, 4))
    plt.plot(zipcode_df.index, zipcode_df['value'], label='Actual')
    plt.plot(forecast_index, forecast_original, label='Forecast', color='red'
    plt.legend()
    plt.title(f'ARIMA Forecast for Zip Code {zipcode_df["Zipcode"].iloc[0]}')
    plt.xlabel('Time')
    plt.ylabel(column)
    plt.tight_layout()
    plt.show()
```

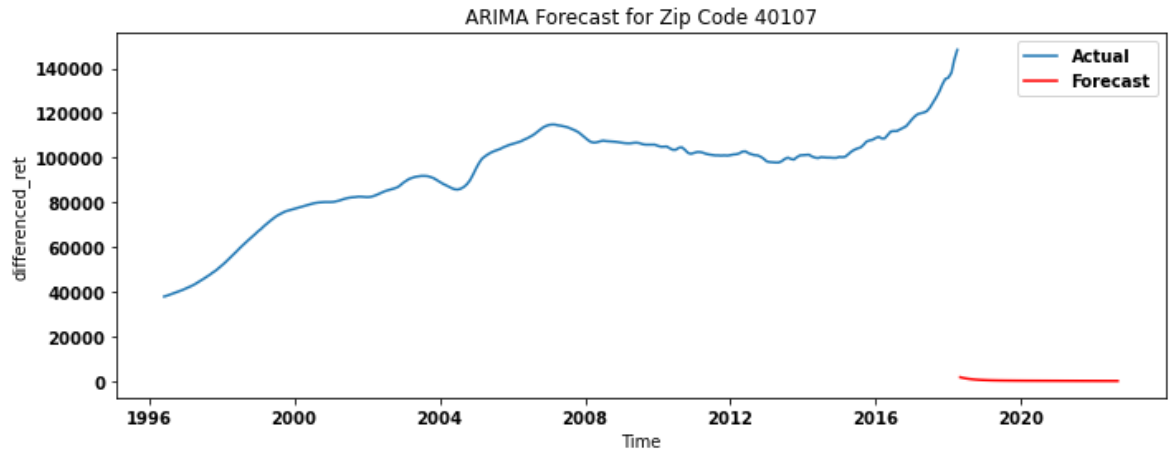
```
    return aic
```

In [34]:
```
# Specify the ARIMA order for each zipcode
order_by_zipcode = {
    49309: (1, 0, 1),
    40107: (1, 0, 2),
    48822: (2, 0, 1),
    49265: (2, 0, 1),
    49425: (1, 0, 1),
    29645: (1, 0, 1),
    66206: (1, 0, 1),
    48835: (2, 0, 2),
    48894: (1, 0, 1),
    15486: (2, 0, 1)
}

# Iterate over the dictionary of zipcode dataframes and call the function
for zipcode, zipcode_df in zipcode_dataframes.items():
    if zipcode in order_by_zipcode:
        order = order_by_zipcode[zipcode]
        aic = build_arima_model(zipcode_df, 'differenced_ret', 'value', order
```

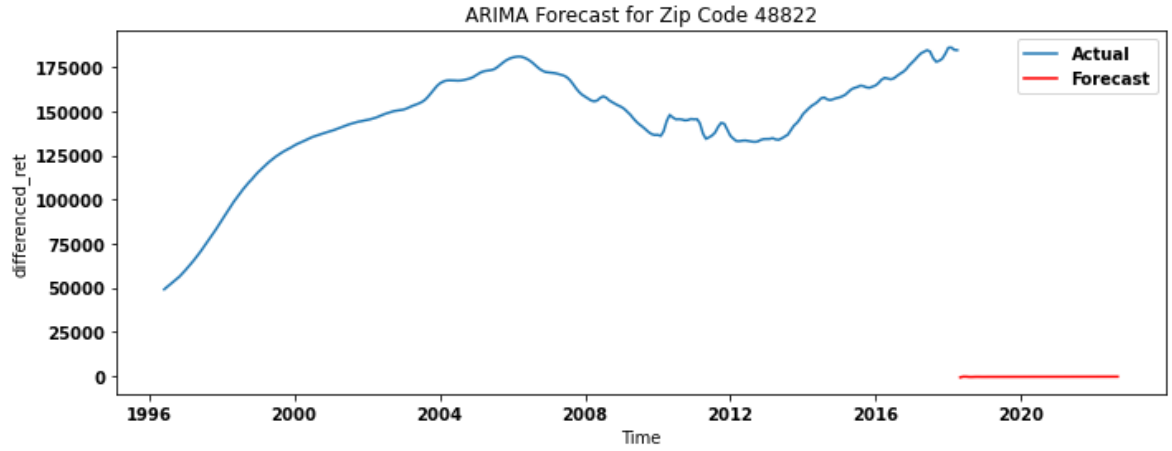
```
ARIMA Parameters: p=1, d=0, q=1
RMSE: 871.9353445974897
AIC: 2766.212031057967
```



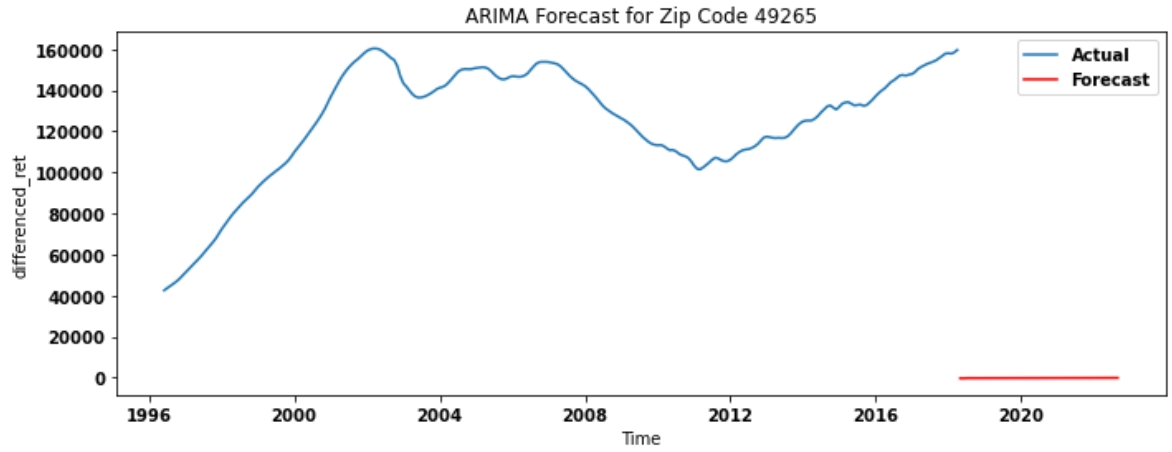

```
ARIMA Parameters: p=1, d=0, q=2
RMSE: 945.6208739836765
AIC: 2965.068010098814
```



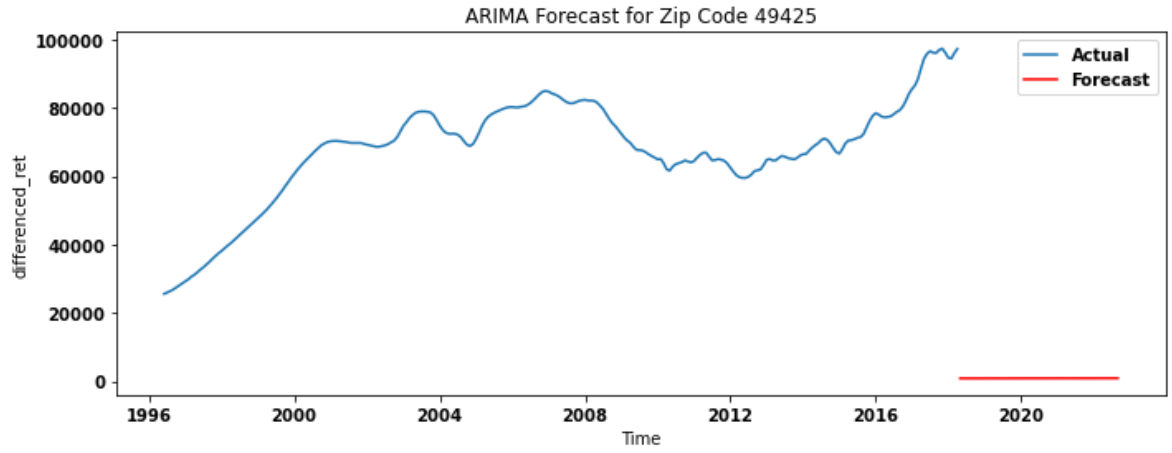
```
ARIMA Parameters: p=2, d=0, q=1
RMSE: 1130.9810085151266
AIC: 3322.6227268982075
```



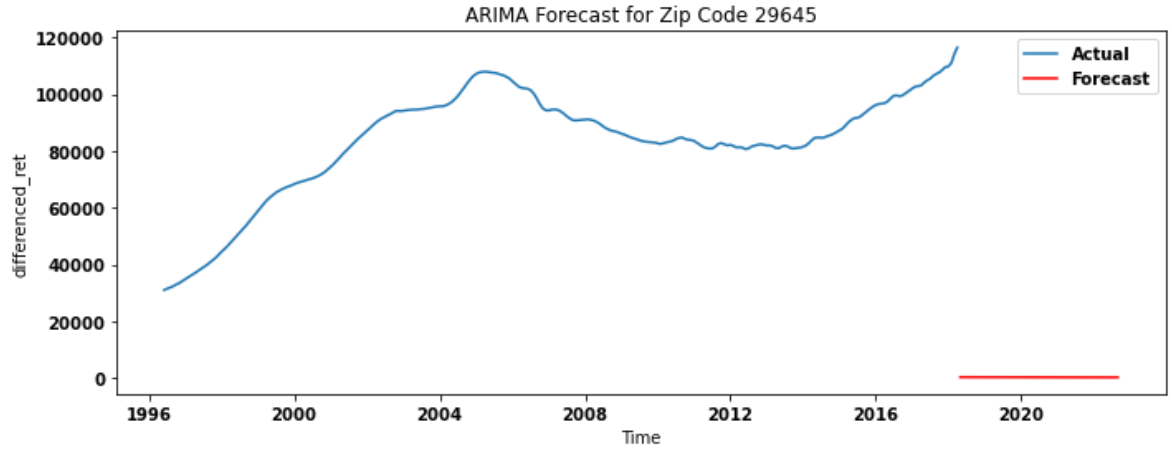

```
ARIMA Parameters: p=2, d=0, q=1
RMSE: 674.2780049670749
AIC: 3068.28683055296
```



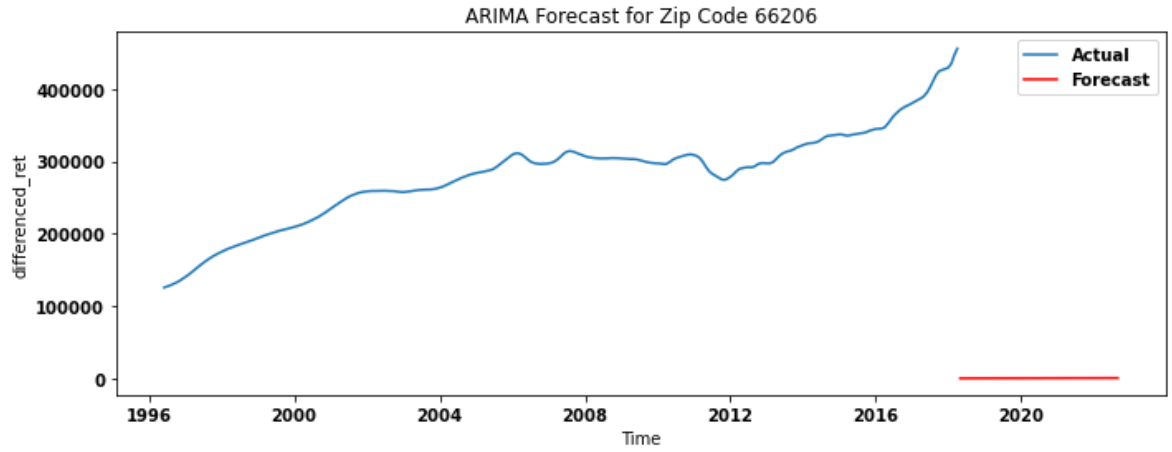

```
ARIMA Parameters: p=1, d=0, q=1
RMSE: 1154.5556709684224
AIC: 3048.455669954312
```



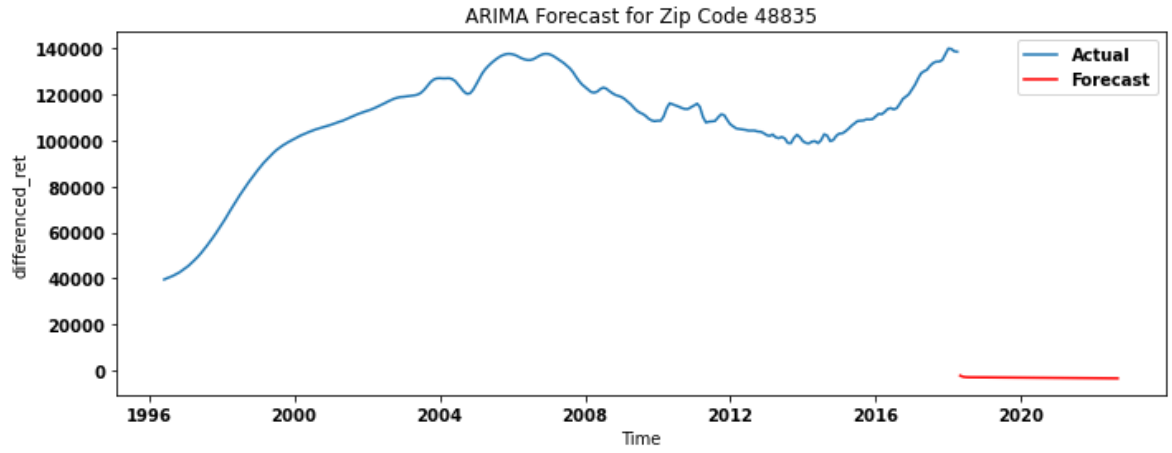
```
ARIMA Parameters: p=1, d=0, q=1
RMSE: 502.56917584502094
AIC: 2899.795834463568
```



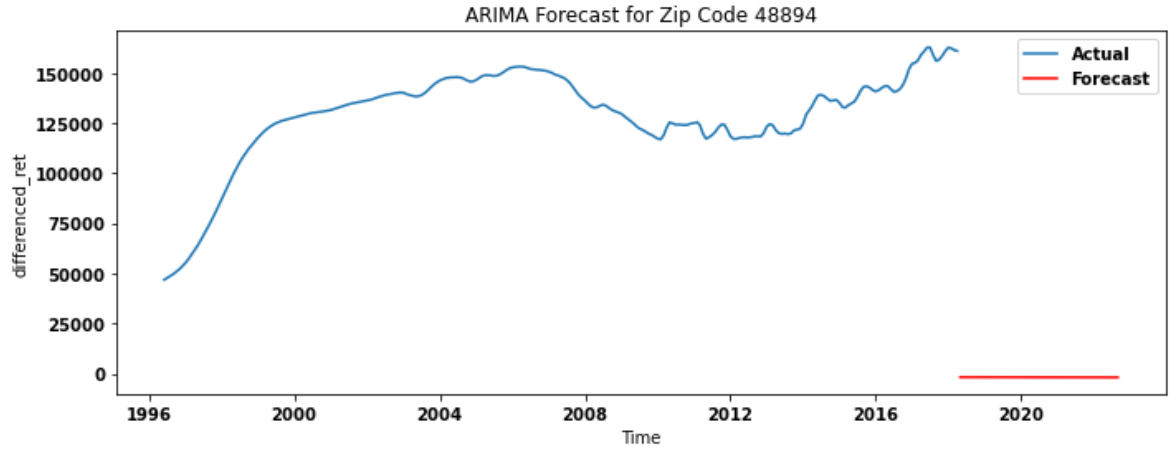

```
ARIMA Parameters: p=1, d=0, q=1
RMSE: 1743.0441673087516
AIC: 3210.2442972144013
```



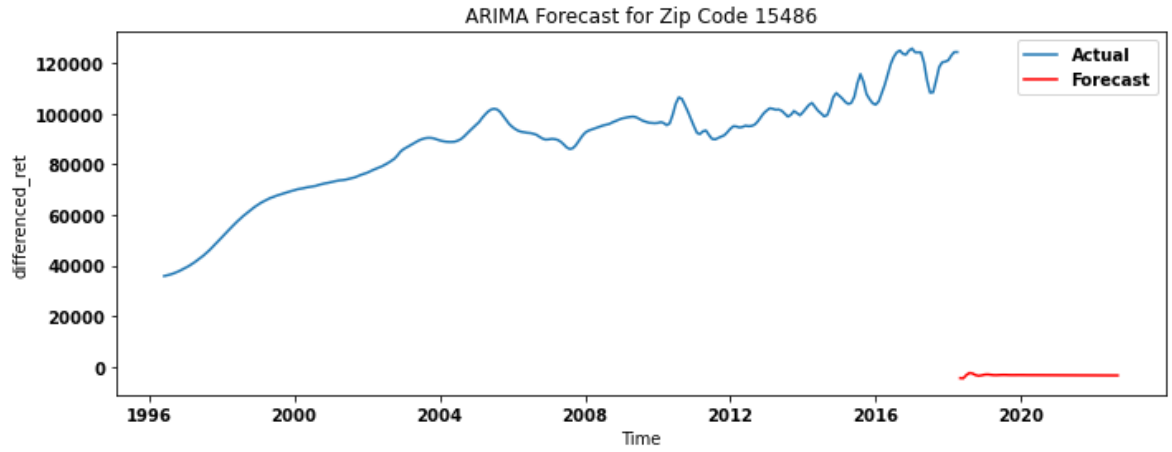

```
ARIMA Parameters: p=2, d=0, q=2
RMSE: 3429.8284023492397
AIC: 3229.38012908581
```



```
ARIMA Parameters: p=1, d=0, q=1
RMSE: 2298.3393356366605
AIC: 3270.4608594273586
```




```
ARIMA Parameters: p=2, d=0, q=1
RMSE: 4028.660414504807
AIC: 3129.623379612433
```



- Since the data has seasonality, ARIMA model does not do well in making forecasts/predictions.

- Therefore, we will use `auto_arima` to perform hyperparameter tuning; this automates the process of selecting the best combination of hyperparameters (p, d, q) for an ARIMA model by searching over a range of values and selecting the configuration that minimizes a chosen metric, in our case, AIC.

Hyperparameter tuning with `auto_arima`

In [35]:
```
# Create a dictionary to store best ARIMA parameters for each zipcode
best_params_by_zipcode = {}

# Iterate over the dictionary of zipcode dataframes
for zipcode, zipcode_df in zipcode_dataframes.items():
    # Extract the time series column
    column = zipcode_df['ret']

    # Find the best ARIMA parameters using auto_arima
    stepwise_model = auto_arima(column, seasonal=True, m=12, trace=True)
    best_params = stepwise_model.get_params()['order'], stepwise_model.get_pa

    # Store the best parameters in the dictionary
    best_params_by_zipcode[zipcode] = best_params

    # Print the AIC value for the best model
    print(f"AIC for Zip Code {zipcode}: {stepwise_model.aic()}")
```

```
Performing stepwise search to minimize aic
 ARIMA(2,2,2)(1,0,1)[12]             : AIC=3664.736, Time=1.27 sec
 ARIMA(0,2,0)(0,0,0)[12]             : AIC=3858.601, Time=0.04 sec
 ARIMA(1,2,0)(1,0,0)[12]             : AIC=3814.487, Time=0.21 sec
 ARIMA(0,2,1)(0,0,1)[12]             : AIC=3759.459, Time=0.23 sec
 ARIMA(2,2,2)(0,0,1)[12]             : AIC=3671.516, Time=0.52 sec
 ARIMA(2,2,2)(1,0,0)[12]             : AIC=3673.471, Time=0.60 sec
 ARIMA(2,2,2)(2,0,1)[12]             : AIC=3666.681, Time=2.36 sec
 ARIMA(2,2,2)(1,0,2)[12]             : AIC=3666.700, Time=2.40 sec
 ARIMA(2,2,2)(0,0,0)[12]             : AIC=3679.579, Time=0.18 sec
 ARIMA(2,2,2)(0,0,2)[12]             : AIC=3669.408, Time=1.48 sec
 ARIMA(2,2,2)(2,0,0)[12]             : AIC=3674.145, Time=0.93 sec
 ARIMA(2,2,2)(2,0,2)[12]             : AIC=3666.148, Time=2.98 sec
 ARIMA(1,2,2)(1,0,1)[12]             : AIC=3706.648, Time=0.89 sec
 ARIMA(2,2,1)(1,0,1)[12]             : AIC=3673.024, Time=0.97 sec
 ARIMA(3,2,2)(1,0,1)[12]             : AIC=3664.480, Time=1.27 sec
 ARIMA(3,2,2)(0,0,1)[12]             : AIC=3670.638, Time=0.81 sec
 ARIMA(3,2,2)(1,0,0)[12]             : AIC=3672.407, Time=0.80 sec
 ARIMA(3,2,2)(2,0,1)[12]             : AIC=3666.591, Time=3.07 sec
 ARIMA(3,2,2)(1,0,2)[12]             : AIC=3666.745, Time=3.43 sec
 ARIMA(3,2,2)(0,0,0)[12]             : AIC=3678.617, Time=0.32 sec
 ARIMA(3,2,2)(0,0,2)[12]             : AIC=3669.119, Time=2.50 sec
 ARIMA(3,2,2)(2,0,0)[12]             : AIC=3673.332, Time=1.70 sec
 ARIMA(3,2,2)(2,0,2)[12]             : AIC=inf, Time=3.61 sec
 ARIMA(3,2,1)(1,0,1)[12]             : AIC=3663.679, Time=1.20 sec
 ARIMA(3,2,1)(0,0,1)[12]             : AIC=3669.848, Time=0.74 sec
 ARIMA(3,2,1)(1,0,0)[12]             : AIC=3671.492, Time=0.74 sec
 ARIMA(3,2,1)(2,0,1)[12]             : AIC=3673.830, Time=3.10 sec
 ARIMA(3,2,1)(1,0,2)[12]             : AIC=inf, Time=2.64 sec
 ARIMA(3,2,1)(0,0,0)[12]             : AIC=3678.880, Time=0.32 sec
 ARIMA(3,2,1)(0,0,2)[12]             : AIC=3668.967, Time=2.17 sec
 ARIMA(3,2,1)(2,0,0)[12]             : AIC=3672.839, Time=1.57 sec
 ARIMA(3,2,1)(2,0,2)[12]             : AIC=inf, Time=3.82 sec
 ARIMA(3,2,0)(1,0,1)[12]             : AIC=3675.236, Time=1.00 sec
 ARIMA(4,2,1)(1,0,1)[12]             : AIC=3663.534, Time=1.55 sec
 ARIMA(4,2,1)(0,0,1)[12]             : AIC=3669.984, Time=0.84 sec
 ARIMA(4,2,1)(1,0,0)[12]             : AIC=3671.820, Time=0.88 sec
 ARIMA(4,2,1)(2,0,1)[12]             : AIC=3665.544, Time=3.97 sec
 ARIMA(4,2,1)(1,0,2)[12]             : AIC=3668.192, Time=3.17 sec
 ARIMA(4,2,1)(0,0,0)[12]             : AIC=3677.358, Time=0.38 sec
 ARIMA(4,2,1)(0,0,2)[12]             : AIC=3667.805, Time=1.95 sec
 ARIMA(4,2,1)(2,0,0)[12]             : AIC=3672.390, Time=1.59 sec
 ARIMA(4,2,1)(2,0,2)[12]             : AIC=inf, Time=3.80 sec
 ARIMA(4,2,0)(1,0,1)[12]             : AIC=3674.127, Time=1.35 sec
 ARIMA(5,2,1)(1,0,1)[12]             : AIC=inf, Time=2.12 sec
 ARIMA(4,2,2)(1,0,1)[12]             : AIC=inf, Time=1.71 sec
 ARIMA(5,2,0)(1,0,1)[12]             : AIC=3663.930, Time=1.35 sec
 ARIMA(5,2,2)(1,0,1)[12]             : AIC=inf, Time=2.16 sec
 ARIMA(4,2,1)(1,0,1)[12] intercept   : AIC=3668.457, Time=1.84 sec

Best model:  ARIMA(4,2,1)(1,0,1)[12]
Total fit time: 78.583 seconds
AIC for Zip Code 49309: 3663.533947022121
Performing stepwise search to minimize aic
 ARIMA(2,1,2)(1,0,1)[12] intercept   : AIC=4088.165, Time=1.21 sec
 ARIMA(0,1,0)(0,0,0)[12] intercept   : AIC=4281.904, Time=0.03 sec
 ARIMA(1,1,0)(1,0,0)[12] intercept   : AIC=4129.258, Time=0.39 sec
```

```
 ARIMA(0,1,1)(0,0,1)[12] intercept   : AIC=4163.468, Time=0.65 sec
 ARIMA(0,1,0)(0,0,0)[12]             : AIC=4337.596, Time=0.03 sec
 ARIMA(2,1,2)(0,0,1)[12] intercept   : AIC=4082.328, Time=1.09 sec
 ARIMA(2,1,2)(0,0,0)[12] intercept   : AIC=4091.315, Time=0.40 sec
 ARIMA(2,1,2)(0,0,2)[12] intercept   : AIC=4079.168, Time=2.41 sec
 ARIMA(2,1,2)(1,0,2)[12] intercept   : AIC=4081.989, Time=3.21 sec
 ARIMA(1,1,2)(0,0,2)[12] intercept   : AIC=4077.819, Time=1.98 sec
 ARIMA(1,1,2)(0,0,1)[12] intercept   : AIC=4081.002, Time=1.08 sec
 ARIMA(1,1,2)(1,0,2)[12] intercept   : AIC=inf, Time=2.56 sec
 ARIMA(1,1,2)(1,0,1)[12] intercept   : AIC=4072.681, Time=1.59 sec
 ARIMA(1,1,2)(1,0,0)[12] intercept   : AIC=4084.088, Time=0.78 sec
 ARIMA(1,1,2)(2,0,1)[12] intercept   : AIC=inf, Time=2.70 sec
 ARIMA(1,1,2)(0,0,0)[12] intercept   : AIC=4089.775, Time=0.29 sec
 ARIMA(1,1,2)(2,0,0)[12] intercept   : AIC=4080.835, Time=2.22 sec
 ARIMA(1,1,2)(2,0,2)[12] intercept   : AIC=inf, Time=3.16 sec
 ARIMA(0,1,2)(1,0,1)[12] intercept   : AIC=4132.869, Time=1.26 sec
 ARIMA(1,1,1)(1,0,1)[12] intercept   : AIC=4125.943, Time=0.78 sec
 ARIMA(1,1,3)(1,0,1)[12] intercept   : AIC=4073.425, Time=1.61 sec
 ARIMA(0,1,1)(1,0,1)[12] intercept   : AIC=4165.095, Time=0.84 sec
 ARIMA(0,1,3)(1,0,1)[12] intercept   : AIC=4134.582, Time=1.26 sec
 ARIMA(2,1,1)(1,0,1)[12] intercept   : AIC=inf, Time=1.13 sec
 ARIMA(2,1,3)(1,0,1)[12] intercept   : AIC=4063.294, Time=1.74 sec
 ARIMA(2,1,3)(0,0,1)[12] intercept   : AIC=4072.461, Time=1.45 sec
 ARIMA(2,1,3)(1,0,0)[12] intercept   : AIC=4074.181, Time=1.59 sec
 ARIMA(2,1,3)(2,0,1)[12] intercept   : AIC=4069.714, Time=3.67 sec
 ARIMA(2,1,3)(1,0,2)[12] intercept   : AIC=4068.777, Time=3.98 sec
 ARIMA(2,1,3)(0,0,0)[12] intercept   : AIC=4078.688, Time=0.63 sec
 ARIMA(2,1,3)(0,0,2)[12] intercept   : AIC=4069.122, Time=3.49 sec
 ARIMA(2,1,3)(2,0,0)[12] intercept   : AIC=4070.716, Time=2.88 sec
 ARIMA(2,1,3)(2,0,2)[12] intercept   : AIC=inf, Time=4.10 sec
 ARIMA(3,1,3)(1,0,1)[12] intercept   : AIC=4055.638, Time=1.75 sec
 ARIMA(3,1,3)(0,0,1)[12] intercept   : AIC=4062.807, Time=1.54 sec
 ARIMA(3,1,3)(1,0,0)[12] intercept   : AIC=4065.071, Time=1.17 sec
 ARIMA(3,1,3)(2,0,1)[12] intercept   : AIC=4063.764, Time=3.64 sec
 ARIMA(3,1,3)(1,0,2)[12] intercept   : AIC=4062.665, Time=4.21 sec
 ARIMA(3,1,3)(0,0,0)[12] intercept   : AIC=4070.240, Time=0.68 sec
 ARIMA(3,1,3)(0,0,2)[12] intercept   : AIC=4060.734, Time=3.56 sec
 ARIMA(3,1,3)(2,0,0)[12] intercept   : AIC=4063.576, Time=3.31 sec
 ARIMA(3,1,3)(2,0,2)[12] intercept   : AIC=inf, Time=4.37 sec
 ARIMA(3,1,2)(1,0,1)[12] intercept   : AIC=inf, Time=1.64 sec
 ARIMA(4,1,3)(1,0,1)[12] intercept   : AIC=4066.495, Time=2.03 sec
 ARIMA(3,1,4)(1,0,1)[12] intercept   : AIC=4065.274, Time=2.19 sec
 ARIMA(2,1,4)(1,0,1)[12] intercept   : AIC=4065.507, Time=1.95 sec
 ARIMA(4,1,2)(1,0,1)[12] intercept   : AIC=inf, Time=1.77 sec
 ARIMA(4,1,4)(1,0,1)[12] intercept   : AIC=4068.244, Time=2.10 sec
 ARIMA(3,1,3)(1,0,1)[12]             : AIC=4066.911, Time=1.78 sec

Best model:  ARIMA(3,1,3)(1,0,1)[12] intercept
Total fit time: 93.931 seconds
AIC for Zip Code 40107: 4055.6375774999988
Performing stepwise search to minimize aic
 ARIMA(2,2,2)(1,0,1)[12]             : AIC=4221.572, Time=0.60 sec
 ARIMA(0,2,0)(0,0,0)[12]             : AIC=4276.097, Time=0.03 sec
 ARIMA(1,2,0)(1,0,0)[12]             : AIC=4261.870, Time=0.09 sec
 ARIMA(0,2,1)(0,0,1)[12]             : AIC=4255.529, Time=0.13 sec
 ARIMA(2,2,2)(0,0,1)[12]             : AIC=4222.010, Time=0.48 sec
 ARIMA(2,2,2)(1,0,0)[12]             : AIC=4222.781, Time=0.32 sec
```

```
ARIMA(2,2,2)(2,0,1)[12]             : AIC=4223.206, Time=1.65 sec
ARIMA(2,2,2)(1,0,2)[12]             : AIC=4223.244, Time=1.71 sec
ARIMA(2,2,2)(0,0,0)[12]             : AIC=4229.910, Time=0.19 sec
ARIMA(2,2,2)(0,0,2)[12]             : AIC=4223.131, Time=1.29 sec
ARIMA(2,2,2)(2,0,0)[12]             : AIC=4223.804, Time=0.73 sec
ARIMA(2,2,2)(2,0,2)[12]             : AIC=4225.068, Time=3.02 sec
ARIMA(1,2,2)(1,0,1)[12]             : AIC=4220.139, Time=0.57 sec
ARIMA(1,2,2)(0,0,1)[12]             : AIC=4220.356, Time=0.27 sec
ARIMA(1,2,2)(1,0,0)[12]             : AIC=4221.058, Time=0.27 sec
ARIMA(1,2,2)(2,0,1)[12]             : AIC=4220.992, Time=2.41 sec
ARIMA(1,2,2)(1,0,2)[12]             : AIC=4221.030, Time=2.01 sec
ARIMA(1,2,2)(0,0,0)[12]             : AIC=4227.768, Time=0.20 sec
ARIMA(1,2,2)(0,0,2)[12]             : AIC=4221.575, Time=0.82 sec
ARIMA(1,2,2)(2,0,0)[12]             : AIC=4222.172, Time=0.62 sec
ARIMA(1,2,2)(2,0,2)[12]             : AIC=4223.472, Time=2.98 sec
```

```
 ARIMA(0,2,2)(1,0,1)[12]             : AIC=4243.885, Time=0.75 sec
 ARIMA(1,2,1)(1,0,1)[12]             : AIC=4249.595, Time=1.00 sec
 ARIMA(1,2,3)(1,0,1)[12]             : AIC=4221.311, Time=0.67 sec
 ARIMA(0,2,1)(1,0,1)[12]             : AIC=4253.506, Time=0.44 sec
 ARIMA(0,2,3)(1,0,1)[12]             : AIC=4232.908, Time=0.49 sec
 ARIMA(2,2,1)(1,0,1)[12]             : AIC=4222.355, Time=0.49 sec
 ARIMA(2,2,3)(1,0,1)[12]             : AIC=4221.076, Time=1.20 sec
 ARIMA(1,2,2)(1,0,1)[12] intercept   : AIC=4220.160, Time=1.26 sec

Best model:  ARIMA(1,2,2)(1,0,1)[12]
Total fit time: 26.720 seconds
AIC for Zip Code 48822: 4220.138516037111
Performing stepwise search to minimize aic
 ARIMA(2,2,2)(1,0,1)[12]             : AIC=inf, Time=1.85 sec
 ARIMA(0,2,0)(0,0,0)[12]             : AIC=3982.113, Time=0.03 sec
 ARIMA(1,2,0)(1,0,0)[12]             : AIC=3954.026, Time=0.20 sec
 ARIMA(0,2,1)(0,0,1)[12]             : AIC=3929.316, Time=0.25 sec
 ARIMA(0,2,1)(0,0,0)[12]             : AIC=3941.120, Time=0.11 sec
 ARIMA(0,2,1)(1,0,1)[12]             : AIC=inf, Time=0.46 sec
 ARIMA(0,2,1)(0,0,2)[12]             : AIC=3915.192, Time=0.91 sec
 ARIMA(0,2,1)(1,0,2)[12]             : AIC=inf, Time=2.05 sec
 ARIMA(0,2,0)(0,0,2)[12]             : AIC=3962.882, Time=0.66 sec
 ARIMA(1,2,1)(0,0,2)[12]             : AIC=3913.196, Time=1.06 sec
 ARIMA(1,2,1)(0,0,1)[12]             : AIC=3926.617, Time=0.31 sec
 ARIMA(1,2,1)(1,0,2)[12]             : AIC=inf, Time=2.66 sec
 ARIMA(1,2,1)(1,0,1)[12]             : AIC=inf, Time=0.58 sec
 ARIMA(1,2,0)(0,0,2)[12]             : AIC=3939.687, Time=0.94 sec
 ARIMA(2,2,1)(0,0,2)[12]             : AIC=3904.250, Time=1.40 sec
 ARIMA(2,2,1)(0,0,1)[12]             : AIC=3918.256, Time=0.39 sec
 ARIMA(2,2,1)(1,0,2)[12]             : AIC=inf, Time=2.51 sec
 ARIMA(2,2,1)(1,0,1)[12]             : AIC=inf, Time=0.87 sec
 ARIMA(2,2,0)(0,0,2)[12]             : AIC=3905.486, Time=1.12 sec
 ARIMA(3,2,1)(0,0,2)[12]             : AIC=3906.175, Time=1.63 sec
 ARIMA(2,2,2)(0,0,2)[12]             : AIC=3901.268, Time=2.04 sec
 ARIMA(2,2,2)(0,0,1)[12]             : AIC=3914.853, Time=0.61 sec
 ARIMA(2,2,2)(1,0,2)[12]             : AIC=inf, Time=3.02 sec
 ARIMA(1,2,2)(0,0,2)[12]             : AIC=3900.122, Time=1.50 sec
 ARIMA(1,2,2)(0,0,1)[12]             : AIC=3913.113, Time=0.60 sec
 ARIMA(1,2,2)(1,0,2)[12]             : AIC=inf, Time=2.47 sec
 ARIMA(1,2,2)(1,0,1)[12]             : AIC=inf, Time=1.00 sec
 ARIMA(0,2,2)(0,0,2)[12]             : AIC=3909.845, Time=1.12 sec
 ARIMA(1,2,3)(0,0,2)[12]             : AIC=3895.467, Time=2.85 sec
 ARIMA(1,2,3)(0,0,1)[12]             : AIC=3908.819, Time=0.80 sec
 ARIMA(1,2,3)(1,0,2)[12]             : AIC=inf, Time=3.95 sec
 ARIMA(1,2,3)(1,0,1)[12]             : AIC=inf, Time=1.48 sec
 ARIMA(0,2,3)(0,0,2)[12]             : AIC=3897.181, Time=1.34 sec
 ARIMA(2,2,3)(0,0,2)[12]             : AIC=3896.633, Time=2.57 sec
 ARIMA(1,2,4)(0,0,2)[12]             : AIC=inf, Time=3.67 sec
 ARIMA(0,2,4)(0,0,2)[12]             : AIC=3898.291, Time=2.11 sec
 ARIMA(2,2,4)(0,0,2)[12]             : AIC=3896.736, Time=3.70 sec
 ARIMA(1,2,3)(0,0,2)[12] intercept   : AIC=3897.223, Time=3.39 sec

Best model:  ARIMA(1,2,3)(0,0,2)[12]
Total fit time: 58.259 seconds
AIC for Zip Code 49265: 3895.46708087815
Performing stepwise search to minimize aic
 ARIMA(2,2,2)(1,0,1)[12]             : AIC=inf, Time=1.59 sec
```

```
 ARIMA(0,2,0)(0,0,0)[12]             : AIC=3950.193, Time=0.04 sec
 ARIMA(1,2,0)(1,0,0)[12]             : AIC=3922.842, Time=0.09 sec
 ARIMA(0,2,1)(0,0,1)[12]             : AIC=3887.746, Time=0.19 sec
 ARIMA(0,2,1)(0,0,0)[12]             : AIC=3893.293, Time=0.09 sec
 ARIMA(0,2,1)(1,0,1)[12]             : AIC=inf, Time=0.56 sec
 ARIMA(0,2,1)(0,0,2)[12]             : AIC=3877.676, Time=0.79 sec
 ARIMA(0,2,1)(1,0,2)[12]             : AIC=inf, Time=2.39 sec
 ARIMA(0,2,0)(0,0,2)[12]             : AIC=3934.381, Time=0.81 sec
 ARIMA(1,2,1)(0,0,2)[12]             : AIC=3878.423, Time=0.87 sec
 ARIMA(0,2,2)(0,0,2)[12]             : AIC=3874.795, Time=1.19 sec
 ARIMA(0,2,2)(0,0,1)[12]             : AIC=3885.797, Time=0.36 sec
 ARIMA(0,2,2)(1,0,2)[12]             : AIC=inf, Time=2.01 sec
 ARIMA(0,2,2)(1,0,1)[12]             : AIC=inf, Time=0.68 sec
 ARIMA(1,2,2)(0,0,2)[12]             : AIC=3846.509, Time=1.33 sec
 ARIMA(1,2,2)(0,0,1)[12]             : AIC=3854.556, Time=0.58 sec
 ARIMA(1,2,2)(1,0,2)[12]             : AIC=inf, Time=2.58 sec
 ARIMA(1,2,2)(1,0,1)[12]             : AIC=inf, Time=1.25 sec
 ARIMA(2,2,2)(0,0,2)[12]             : AIC=3841.656, Time=2.57 sec
 ARIMA(2,2,2)(0,0,1)[12]             : AIC=3850.317, Time=0.65 sec
 ARIMA(2,2,2)(1,0,2)[12]             : AIC=inf, Time=3.41 sec
 ARIMA(2,2,1)(0,0,2)[12]             : AIC=3846.556, Time=1.28 sec
 ARIMA(3,2,2)(0,0,2)[12]             : AIC=3837.403, Time=3.22 sec
 ARIMA(3,2,2)(0,0,1)[12]             : AIC=3844.610, Time=1.26 sec
 ARIMA(3,2,2)(1,0,2)[12]             : AIC=inf, Time=3.13 sec
 ARIMA(3,2,2)(1,0,1)[12]             : AIC=3838.822, Time=1.48 sec
 ARIMA(3,2,1)(0,0,2)[12]             : AIC=3848.406, Time=1.81 sec
 ARIMA(4,2,2)(0,0,2)[12]             : AIC=3842.326, Time=3.26 sec
 ARIMA(3,2,3)(0,0,2)[12]             : AIC=3827.469, Time=2.66 sec
 ARIMA(3,2,3)(0,0,1)[12]             : AIC=3832.376, Time=1.07 sec
 ARIMA(3,2,3)(1,0,2)[12]             : AIC=inf, Time=3.87 sec
 ARIMA(3,2,3)(1,0,1)[12]             : AIC=inf, Time=1.58 sec
 ARIMA(2,2,3)(0,0,2)[12]             : AIC=3827.851, Time=2.26 sec
 ARIMA(4,2,3)(0,0,2)[12]             : AIC=3829.457, Time=3.15 sec
 ARIMA(3,2,4)(0,0,2)[12]             : AIC=3835.567, Time=4.54 sec
 ARIMA(2,2,4)(0,0,2)[12]             : AIC=3828.088, Time=3.65 sec
 ARIMA(4,2,4)(0,0,2)[12]             : AIC=3834.461, Time=5.22 sec
 ARIMA(3,2,3)(0,0,2)[12] intercept   : AIC=3829.387, Time=3.55 sec

Best model:  ARIMA(3,2,3)(0,0,2)[12]
Total fit time: 71.072 seconds
AIC for Zip Code 49425: 3827.4688221139136
Performing stepwise search to minimize aic
 ARIMA(2,2,2)(1,0,1)[12]             : AIC=3662.955, Time=0.96 sec
 ARIMA(0,2,0)(0,0,0)[12]             : AIC=3725.340, Time=0.03 sec
 ARIMA(1,2,0)(1,0,0)[12]             : AIC=3715.253, Time=0.18 sec
 ARIMA(0,2,1)(0,0,1)[12]             : AIC=3698.630, Time=0.22 sec
 ARIMA(2,2,2)(0,0,1)[12]             : AIC=3671.852, Time=0.62 sec
 ARIMA(2,2,2)(1,0,0)[12]             : AIC=3674.135, Time=0.45 sec
 ARIMA(2,2,2)(2,0,1)[12]             : AIC=3664.350, Time=2.44 sec
 ARIMA(2,2,2)(1,0,2)[12]             : AIC=3664.171, Time=2.44 sec
 ARIMA(2,2,2)(0,0,0)[12]             : AIC=3676.759, Time=0.27 sec
 ARIMA(2,2,2)(0,0,2)[12]             : AIC=3664.596, Time=1.66 sec
 ARIMA(2,2,2)(2,0,0)[12]             : AIC=3667.788, Time=1.13 sec
 ARIMA(2,2,2)(2,0,2)[12]             : AIC=3665.362, Time=3.74 sec
 ARIMA(1,2,2)(1,0,1)[12]             : AIC=3662.013, Time=1.04 sec
 ARIMA(1,2,2)(0,0,1)[12]             : AIC=3670.465, Time=0.43 sec
 ARIMA(1,2,2)(1,0,0)[12]             : AIC=3672.544, Time=0.47 sec
```

```
ARIMA(1,2,2)(2,0,1)[12]             : AIC=3668.227, Time=1.68 sec
ARIMA(1,2,2)(1,0,2)[12]             : AIC=3663.219, Time=2.26 sec
ARIMA(1,2,2)(0,0,0)[12]             : AIC=3674.845, Time=0.19 sec
ARIMA(1,2,2)(0,0,2)[12]             : AIC=3663.294, Time=1.32 sec
ARIMA(1,2,2)(2,0,0)[12]             : AIC=3666.231, Time=0.91 sec
ARIMA(1,2,2)(2,0,2)[12]             : AIC=3664.303, Time=2.88 sec
ARIMA(0,2,2)(1,0,1)[12]             : AIC=3672.911, Time=0.62 sec
ARIMA(1,2,1)(1,0,1)[12]             : AIC=3683.334, Time=0.77 sec
ARIMA(1,2,3)(1,0,1)[12]             : AIC=3649.579, Time=1.15 sec
ARIMA(1,2,3)(0,0,1)[12]             : AIC=3661.803, Time=0.50 sec
ARIMA(1,2,3)(1,0,0)[12]             : AIC=3663.568, Time=0.46 sec
ARIMA(1,2,3)(2,0,1)[12]             : AIC=3650.717, Time=2.52 sec
ARIMA(1,2,3)(1,0,2)[12]             : AIC=3650.633, Time=2.47 sec
ARIMA(1,2,3)(0,0,0)[12]             : AIC=3665.146, Time=0.27 sec
ARIMA(1,2,3)(0,0,2)[12]             : AIC=3653.779, Time=1.50 sec
ARIMA(1,2,3)(2,0,0)[12]             : AIC=3658.411, Time=0.91 sec
ARIMA(1,2,3)(2,0,2)[12]             : AIC=inf, Time=3.63 sec
ARIMA(0,2,3)(1,0,1)[12]             : AIC=3652.050, Time=1.03 sec
```

```
 ARIMA(2,2,3)(1,0,1)[12]             : AIC=inf, Time=1.46 sec
 ARIMA(1,2,4)(1,0,1)[12]             : AIC=3651.579, Time=2.07 sec
 ARIMA(0,2,4)(1,0,1)[12]             : AIC=3650.607, Time=1.46 sec
 ARIMA(2,2,4)(1,0,1)[12]             : AIC=inf, Time=1.95 sec
 ARIMA(1,2,3)(1,0,1)[12] intercept   : AIC=3651.585, Time=1.63 sec

Best model:  ARIMA(1,2,3)(1,0,1)[12]
Total fit time: 49.791 seconds
AIC for Zip Code 29645: 3649.5792231147316
Performing stepwise search to minimize aic
 ARIMA(2,1,2)(1,0,1)[12] intercept   : AIC=inf, Time=1.40 sec
 ARIMA(0,1,0)(0,0,0)[12] intercept   : AIC=4750.129, Time=0.04 sec
 ARIMA(1,1,0)(1,0,0)[12] intercept   : AIC=4753.964, Time=0.90 sec
 ARIMA(0,1,1)(0,0,1)[12] intercept   : AIC=4745.862, Time=0.21 sec
 ARIMA(0,1,0)(0,0,0)[12]             : AIC=4830.027, Time=0.05 sec
 ARIMA(0,1,1)(0,0,0)[12] intercept   : AIC=4746.628, Time=0.09 sec
 ARIMA(0,1,1)(1,0,1)[12] intercept   : AIC=4744.103, Time=0.36 sec
 ARIMA(0,1,1)(1,0,0)[12] intercept   : AIC=4744.894, Time=0.19 sec
 ARIMA(0,1,1)(2,0,1)[12] intercept   : AIC=4746.074, Time=0.56 sec
 ARIMA(0,1,1)(1,0,2)[12] intercept   : AIC=4746.065, Time=0.91 sec
 ARIMA(0,1,1)(0,0,2)[12] intercept   : AIC=4747.862, Time=0.36 sec
 ARIMA(0,1,1)(2,0,0)[12] intercept   : AIC=4746.837, Time=0.30 sec
 ARIMA(0,1,1)(2,0,2)[12] intercept   : AIC=inf, Time=2.75 sec
 ARIMA(0,1,0)(1,0,1)[12] intercept   : AIC=4752.220, Time=0.23 sec
 ARIMA(1,1,1)(1,0,1)[12] intercept   : AIC=inf, Time=1.30 sec
 ARIMA(0,1,2)(1,0,1)[12] intercept   : AIC=4727.349, Time=1.26 sec
 ARIMA(0,1,2)(0,0,1)[12] intercept   : AIC=4727.100, Time=0.58 sec
 ARIMA(0,1,2)(0,0,0)[12] intercept   : AIC=4726.846, Time=0.27 sec
 ARIMA(0,1,2)(1,0,0)[12] intercept   : AIC=4726.757, Time=0.47 sec
 ARIMA(0,1,2)(2,0,0)[12] intercept   : AIC=4728.766, Time=1.33 sec
 ARIMA(0,1,2)(2,0,1)[12] intercept   : AIC=4729.320, Time=1.61 sec
 ARIMA(1,1,2)(1,0,0)[12] intercept   : AIC=4708.955, Time=0.96 sec
 ARIMA(1,1,2)(0,0,0)[12] intercept   : AIC=4706.497, Time=0.44 sec
 ARIMA(1,1,2)(0,0,1)[12] intercept   : AIC=4708.975, Time=1.06 sec
 ARIMA(1,1,2)(1,0,1)[12] intercept   : AIC=5221.458, Time=1.52 sec
 ARIMA(1,1,1)(0,0,0)[12] intercept   : AIC=4712.698, Time=0.39 sec
 ARIMA(2,1,2)(0,0,0)[12] intercept   : AIC=inf, Time=0.57 sec
 ARIMA(1,1,3)(0,0,0)[12] intercept   : AIC=4706.953, Time=0.56 sec
 ARIMA(0,1,3)(0,0,0)[12] intercept   : AIC=4725.443, Time=0.32 sec
 ARIMA(2,1,1)(0,0,0)[12] intercept   : AIC=4703.826, Time=0.38 sec
 ARIMA(2,1,1)(1,0,0)[12] intercept   : AIC=4705.863, Time=1.09 sec
 ARIMA(2,1,1)(0,0,1)[12] intercept   : AIC=4705.939, Time=1.06 sec
 ARIMA(2,1,1)(1,0,1)[12] intercept   : AIC=5056.492, Time=1.38 sec
 ARIMA(2,1,0)(0,0,0)[12] intercept   : AIC=4742.955, Time=0.28 sec
 ARIMA(3,1,1)(0,0,0)[12] intercept   : AIC=4704.599, Time=0.61 sec
 ARIMA(1,1,0)(0,0,0)[12] intercept   : AIC=4729.845, Time=0.26 sec
 ARIMA(3,1,0)(0,0,0)[12] intercept   : AIC=4742.875, Time=0.32 sec
 ARIMA(3,1,2)(0,0,0)[12] intercept   : AIC=5423.321, Time=0.74 sec
 ARIMA(2,1,1)(0,0,0)[12]             : AIC=inf, Time=0.34 sec

Best model:  ARIMA(2,1,1)(0,0,0)[12] intercept
Total fit time: 27.497 seconds
AIC for Zip Code 66206: 4703.825674288546
Performing stepwise search to minimize aic
 ARIMA(2,2,2)(1,0,1)[12]             : AIC=4127.421, Time=1.37 sec
 ARIMA(0,2,0)(0,0,0)[12]             : AIC=4218.627, Time=0.03 sec
 ARIMA(1,2,0)(1,0,0)[12]             : AIC=4198.224, Time=0.20 sec
```

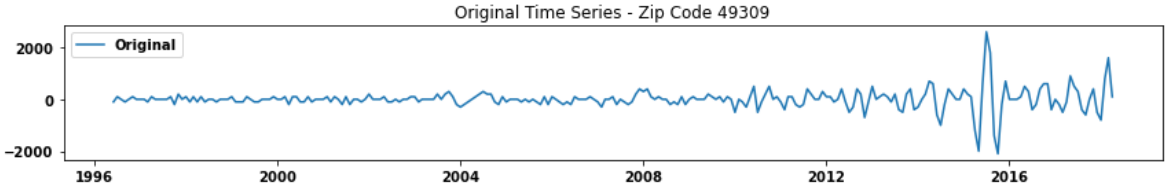
```
 ARIMA(0,2,1)(0,0,1)[12]             : AIC=4187.653, Time=0.25 sec
 ARIMA(2,2,2)(0,0,1)[12]             : AIC=4127.370, Time=0.86 sec
 ARIMA(2,2,2)(0,0,0)[12]             : AIC=4148.444, Time=0.21 sec
 ARIMA(2,2,2)(0,0,2)[12]             : AIC=4129.232, Time=2.33 sec
 ARIMA(2,2,2)(1,0,0)[12]             : AIC=4128.076, Time=0.81 sec
 ARIMA(2,2,2)(1,0,2)[12]             : AIC=4129.914, Time=2.82 sec
 ARIMA(1,2,2)(0,0,1)[12]             : AIC=4126.457, Time=0.64 sec
 ARIMA(1,2,2)(0,0,0)[12]             : AIC=4146.990, Time=0.31 sec
 ARIMA(1,2,2)(1,0,1)[12]             : AIC=4126.640, Time=1.16 sec
 ARIMA(1,2,2)(0,0,2)[12]             : AIC=4128.314, Time=1.27 sec
 ARIMA(1,2,2)(1,0,0)[12]             : AIC=4127.172, Time=0.50 sec
 ARIMA(1,2,2)(1,0,2)[12]             : AIC=inf, Time=2.59 sec
 ARIMA(0,2,2)(0,0,1)[12]             : AIC=4146.583, Time=0.44 sec
 ARIMA(1,2,1)(0,0,1)[12]             : AIC=4176.575, Time=0.34 sec
 ARIMA(1,2,3)(0,0,1)[12]             : AIC=4121.239, Time=0.77 sec
 ARIMA(1,2,3)(0,0,0)[12]             : AIC=4144.472, Time=0.35 sec
 ARIMA(1,2,3)(1,0,1)[12]             : AIC=4121.083, Time=1.40 sec
 ARIMA(1,2,3)(1,0,0)[12]             : AIC=4122.039, Time=0.74 sec
 ARIMA(1,2,3)(2,0,1)[12]             : AIC=4126.964, Time=2.60 sec
 ARIMA(1,2,3)(1,0,2)[12]             : AIC=inf, Time=2.72 sec
 ARIMA(1,2,3)(0,0,2)[12]             : AIC=4123.083, Time=1.81 sec
 ARIMA(1,2,3)(2,0,0)[12]             : AIC=4123.883, Time=1.35 sec
 ARIMA(1,2,3)(2,0,2)[12]             : AIC=4124.435, Time=3.78 sec
 ARIMA(0,2,3)(1,0,1)[12]             : AIC=4120.515, Time=1.02 sec
 ARIMA(0,2,3)(0,0,1)[12]             : AIC=4120.432, Time=0.48 sec
 ARIMA(0,2,3)(0,0,0)[12]             : AIC=4142.683, Time=0.20 sec
 ARIMA(0,2,3)(0,0,2)[12]             : AIC=4122.333, Time=1.39 sec
 ARIMA(0,2,3)(1,0,0)[12]             : AIC=4121.007, Time=0.36 sec
 ARIMA(0,2,3)(1,0,2)[12]             : AIC=4120.020, Time=2.31 sec
 ARIMA(0,2,3)(2,0,2)[12]             : AIC=4123.403, Time=3.08 sec
 ARIMA(0,2,3)(2,0,1)[12]             : AIC=inf, Time=2.35 sec
 ARIMA(0,2,2)(1,0,2)[12]             : AIC=inf, Time=2.28 sec
 ARIMA(0,2,4)(1,0,2)[12]             : AIC=4125.916, Time=3.09 sec
 ARIMA(1,2,4)(1,0,2)[12]             : AIC=inf, Time=3.52 sec
 ARIMA(0,2,3)(1,0,2)[12] intercept   : AIC=inf, Time=3.05 sec

Best model:  ARIMA(0,2,3)(1,0,2)[12]
Total fit time: 54.857 seconds
AIC for Zip Code 48835: 4120.019801409065
Performing stepwise search to minimize aic
 ARIMA(2,2,2)(1,0,1)[12]             : AIC=4223.740, Time=0.83 sec
 ARIMA(0,2,0)(0,0,0)[12]             : AIC=4284.806, Time=0.04 sec
 ARIMA(1,2,0)(1,0,0)[12]             : AIC=4268.858, Time=0.11 sec
 ARIMA(0,2,1)(0,0,1)[12]             : AIC=4262.505, Time=0.13 sec
 ARIMA(2,2,2)(0,0,1)[12]             : AIC=4222.838, Time=0.40 sec
 ARIMA(2,2,2)(0,0,0)[12]             : AIC=4229.201, Time=0.31 sec
 ARIMA(2,2,2)(0,0,2)[12]             : AIC=4224.547, Time=1.03 sec
 ARIMA(2,2,2)(1,0,0)[12]             : AIC=4223.241, Time=0.31 sec
 ARIMA(2,2,2)(1,0,2)[12]             : AIC=4225.141, Time=1.91 sec
 ARIMA(1,2,2)(0,0,1)[12]             : AIC=4224.222, Time=0.30 sec
 ARIMA(2,2,1)(0,0,1)[12]             : AIC=4222.059, Time=0.30 sec
 ARIMA(2,2,1)(0,0,0)[12]             : AIC=4228.093, Time=0.17 sec
 ARIMA(2,2,1)(1,0,1)[12]             : AIC=4222.925, Time=0.57 sec
 ARIMA(2,2,1)(0,0,2)[12]             : AIC=4223.740, Time=0.88 sec
 ARIMA(2,2,1)(1,0,0)[12]             : AIC=4222.447, Time=0.30 sec
 ARIMA(2,2,1)(1,0,2)[12]             : AIC=4224.215, Time=2.19 sec
 ARIMA(1,2,1)(0,0,1)[12]             : AIC=4261.100, Time=0.37 sec
```

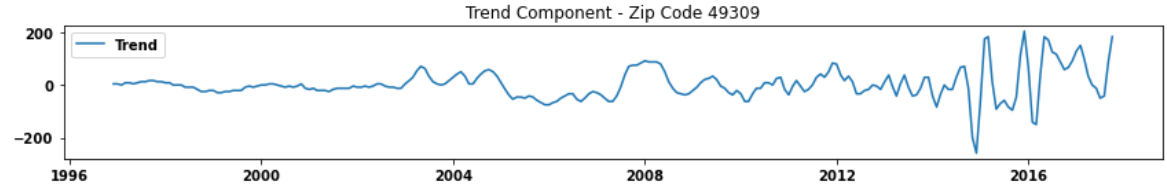
```
 ARIMA(2,2,0)(0,0,1)[12]             : AIC=4247.841, Time=0.13 sec
 ARIMA(3,2,1)(0,0,1)[12]             : AIC=4223.352, Time=0.50 sec
 ARIMA(1,2,0)(0,0,1)[12]             : AIC=4268.530, Time=0.10 sec
 ARIMA(3,2,0)(0,0,1)[12]             : AIC=4245.159, Time=0.15 sec
 ARIMA(3,2,2)(0,0,1)[12]             : AIC=4224.787, Time=0.58 sec
 ARIMA(2,2,1)(0,0,1)[12] intercept   : AIC=4223.042, Time=0.62 sec

Best model:  ARIMA(2,2,1)(0,0,1)[12]
Total fit time: 12.274 seconds
AIC for Zip Code 48894: 4222.059335934166
Performing stepwise search to minimize aic
 ARIMA(2,1,2)(1,0,1)[12] intercept   : AIC=4445.448, Time=1.21 sec
 ARIMA(0,1,0)(0,0,0)[12] intercept   : AIC=4548.910, Time=0.03 sec
 ARIMA(1,1,0)(1,0,0)[12] intercept   : AIC=4463.758, Time=0.38 sec
 ARIMA(0,1,1)(0,0,1)[12] intercept   : AIC=4449.620, Time=0.61 sec
 ARIMA(0,1,0)(0,0,0)[12]             : AIC=4561.442, Time=0.03 sec
 ARIMA(2,1,2)(0,0,1)[12] intercept   : AIC=4448.084, Time=0.96 sec
 ARIMA(2,1,2)(1,0,0)[12] intercept   : AIC=4449.181, Time=1.07 sec
 ARIMA(2,1,2)(2,0,1)[12] intercept   : AIC=4444.926, Time=2.91 sec
 ARIMA(2,1,2)(2,0,0)[12] intercept   : AIC=4444.597, Time=2.65 sec

 ARIMA(1,1,2)(2,0,0)[12] intercept   : AIC=4449.010, Time=2.23 sec
 ARIMA(2,1,1)(2,0,0)[12] intercept   : AIC=4443.526, Time=1.76 sec
 ARIMA(2,1,1)(1,0,0)[12] intercept   : AIC=4448.573, Time=0.83 sec
 ARIMA(2,1,1)(2,0,1)[12] intercept   : AIC=4443.608, Time=2.61 sec
 ARIMA(2,1,1)(1,0,1)[12] intercept   : AIC=4444.306, Time=1.10 sec
 ARIMA(1,1,1)(2,0,0)[12] intercept   : AIC=4446.168, Time=1.94 sec
 ARIMA(2,1,0)(2,0,0)[12] intercept   : AIC=4441.630, Time=1.36 sec
 ARIMA(2,1,0)(1,0,0)[12] intercept   : AIC=4446.642, Time=0.58 sec
 ARIMA(2,1,0)(2,0,1)[12] intercept   : AIC=4441.741, Time=1.78 sec
 ARIMA(2,1,0)(1,0,1)[12] intercept   : AIC=4442.481, Time=0.83 sec
 ARIMA(1,1,0)(2,0,0)[12] intercept   : AIC=4462.842, Time=0.37 sec
 ARIMA(3,1,0)(2,0,0)[12] intercept   : AIC=4443.481, Time=1.36 sec
 ARIMA(3,1,1)(2,0,0)[12] intercept   : AIC=4445.421, Time=2.53 sec
 ARIMA(2,1,0)(2,0,0)[12]             : AIC=4455.007, Time=0.68 sec

Best model:  ARIMA(2,1,0)(2,0,0)[12] intercept
Total fit time: 29.846 seconds
AIC for Zip Code 15486: 4441.630013772288
```

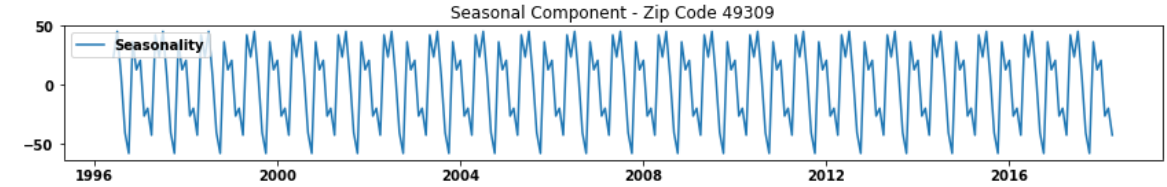
- Next we will perform seasonal decomposition to visually inspect the various components of our time series data such as the trend, seasonality and residuals.

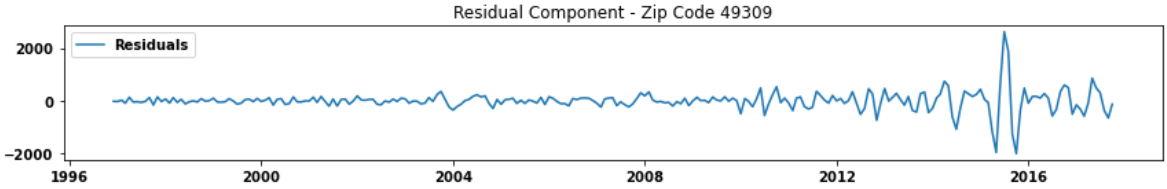
Seasonal Decomposition

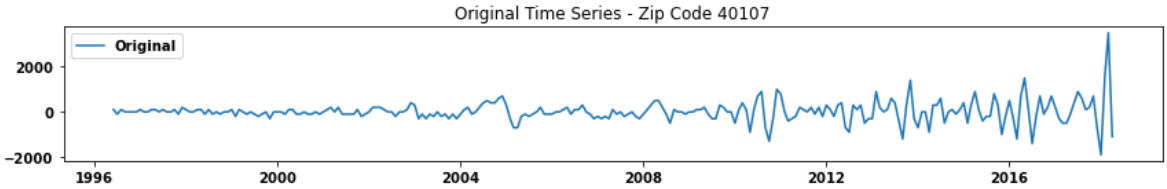
In [36]:
```
# Specify the frequency of seasonality(monthly data)
seasonality_frequency = 12

# Iterate over the dictionary of zipcode dataframes
for zipcode, zipcode_df in zipcode_dataframes.items():
    time_series = zipcode_df['differenced_ret']

    # Perform seasonal decomposition
    decomposition = seasonal_decompose(time_series, period=seasonality_freque

    # Plot the decomposed components
    trend = decomposition.trend
    seasonal = decomposition.seasonal
    residual = decomposition.resid

    plt.figure(figsize=(12, 8))

    plt.subplot(411)
    plt.plot(time_series, label='Original')
    plt.legend(loc='upper left')
    plt.title(f'Original Time Series - Zip Code {zipcode}')

    plt.subplot(412)
    plt.plot(trend, label='Trend')
    plt.legend(loc='upper left')
    plt.title(f'Trend Component - Zip Code {zipcode}')

    plt.subplot(413)
    plt.plot(seasonal,label='Seasonality')
    plt.legend(loc='upper left')
    plt.title(f'Seasonal Component - Zip Code {zipcode}')

    plt.subplot(414)
    plt.plot(residual, label='Residuals')
    plt.legend(loc='upper left')
    plt.title(f'Residual Component - Zip Code {zipcode}')

    plt.tight_layout()
    plt.show()
```

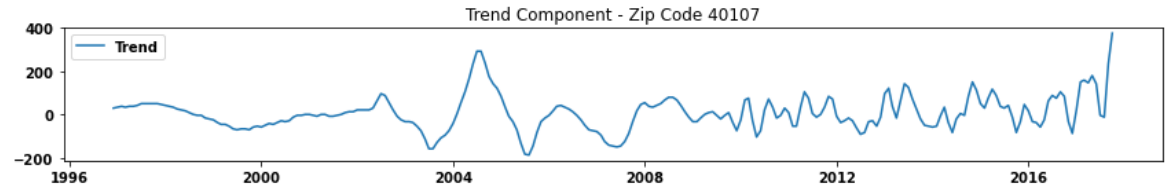

Original Time Series - Zip Code 49309
Original
2000
0
−2000
1996
2000
2004
2008
2012
2016

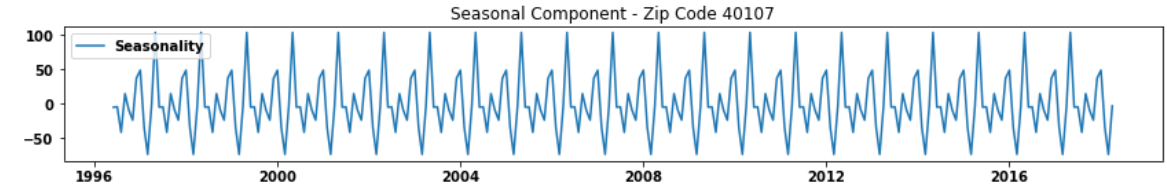

Trend Component - Zip Code 49309
Trend
200
0
−200
1996
2000
2004
2008
2012
2016

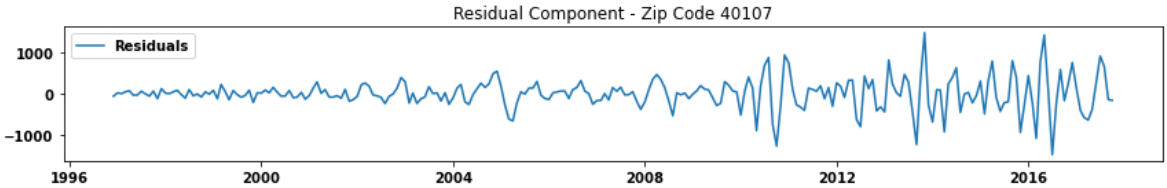

Seasonal Component - Zip Code 49309
Seasonality
50
0
−50
1996
2000
2004
2008
2012
2016

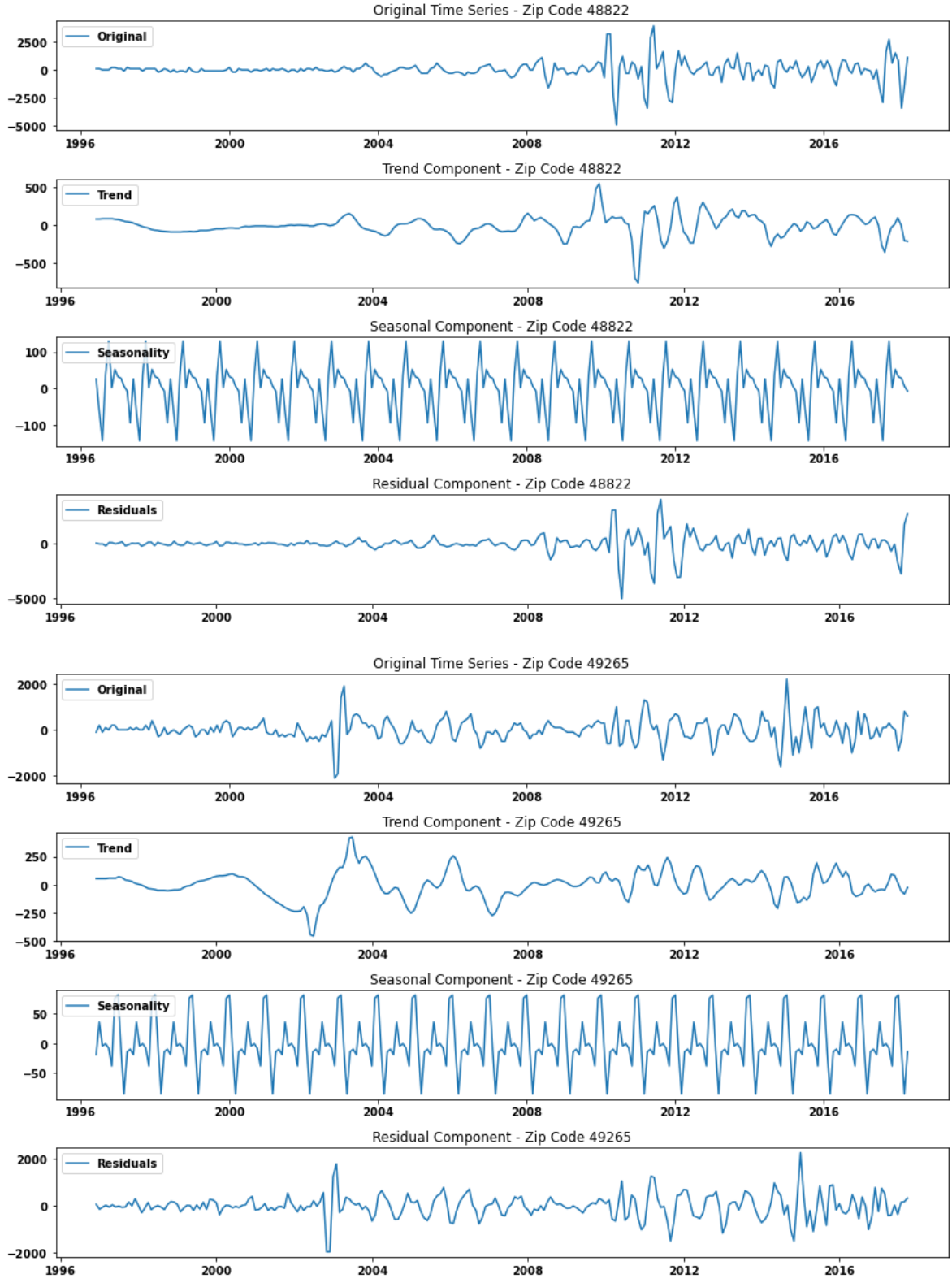

Residual Component - Zip Code 49309
Residuals
2000
0
−2000
1996
2000
2004
2008
2012
2016

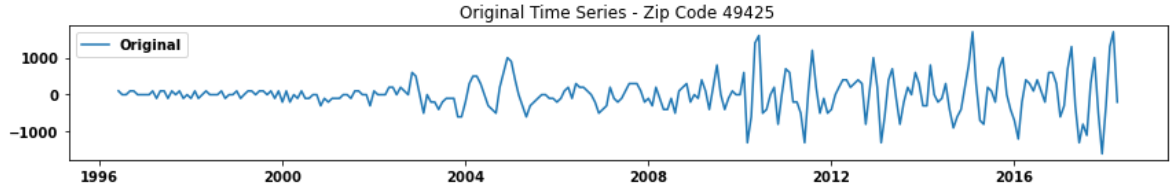

Original Time Series - Zip Code 40107
Original
2000
0
−2000
1996
2000
2004
2008
2012
2016

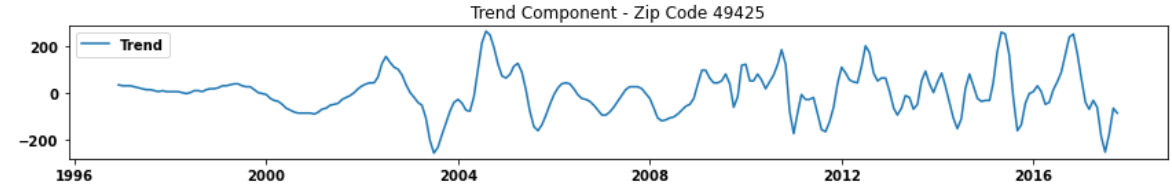

Trend Component - Zip Code 40107
Trend
400
200
0
−200
1996
2000
2004
2008
2012
2016

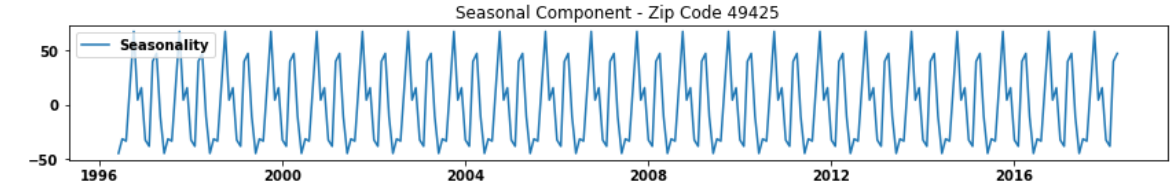

Seasonal Component - Zip Code 40107
Seasonality
100
50
0
−50
1996
2000
2004
2008
2012
2016

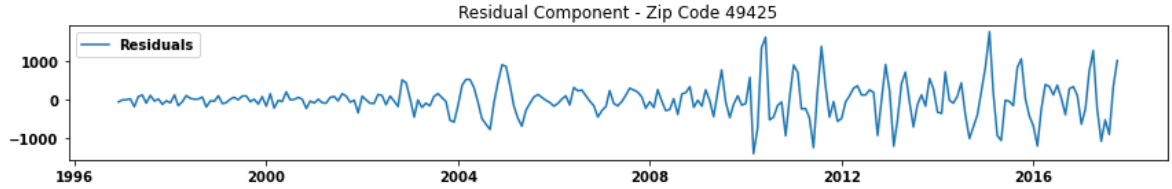

Residual Component - Zip Code 40107
Residuals
1000
0
−1000
1996
2000
2004
2008
2012
2016


Original Time Series - Zip Code 48822
Original
Trend Component - Zip Code 48822
Trend
Seasonal Component - Zip Code 48822
Seasonality
Residual Component - Zip Code 48822
Residuals
Original Time Series - Zip Code 49265
Original
Trend Component - Zip Code 49265
Trend
Seasonal Component - Zip Code 49265
Seasonality
Residual Component - Zip Code 49265
Residuals

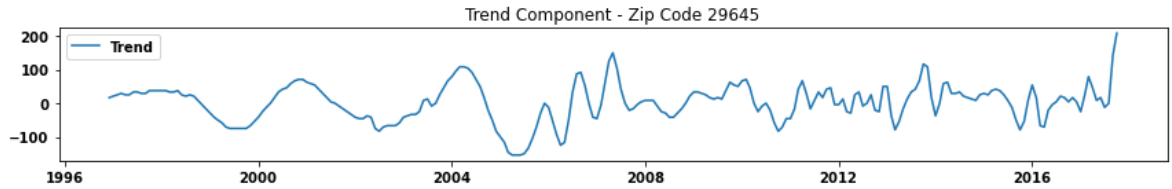

Original Time Series - Zip Code 49425
Original
1000
0
−1000
1996
2000
2004
2008
2012
2016

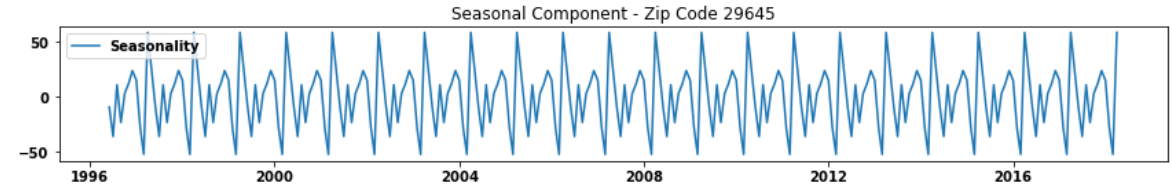

Trend Component - Zip Code 49425
Trend
200
0
−200
1996
2000
2004
2008
2012
2016

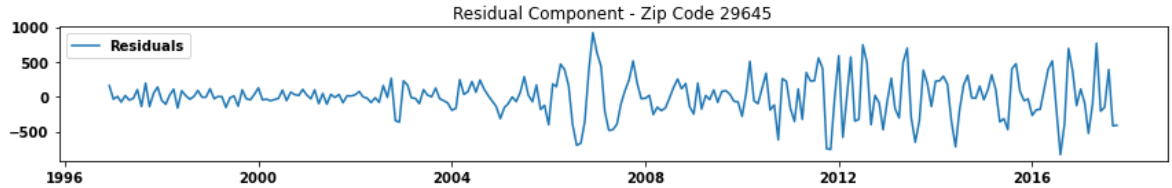

Seasonal Component - Zip Code 49425
Seasonality
50
0
−50
1996
2000
2004
2008
2012
2016

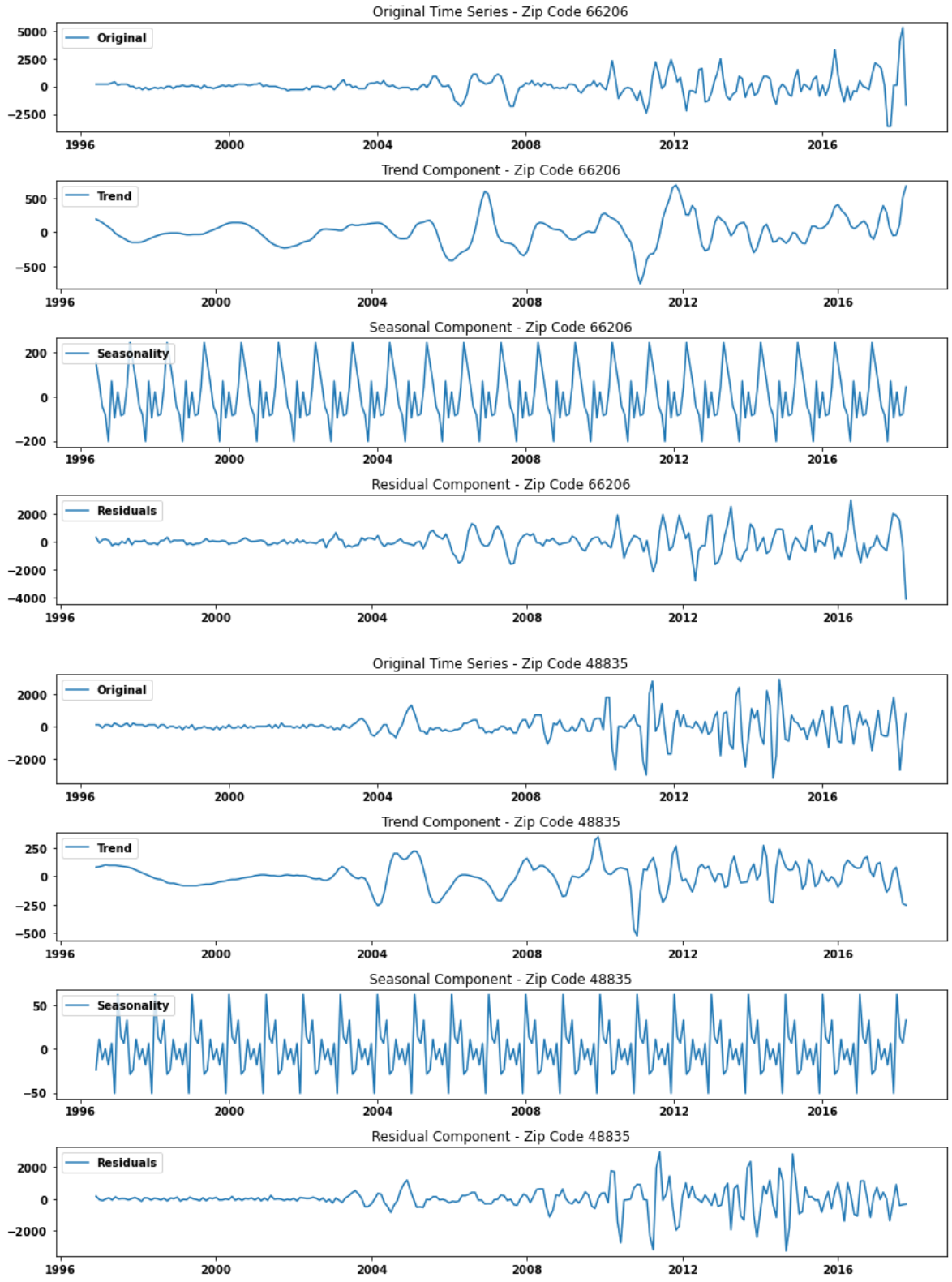

Residual Component - Zip Code 49425
Residuals
1000
0
−1000
1996
2000
2004
2008
2012
2016

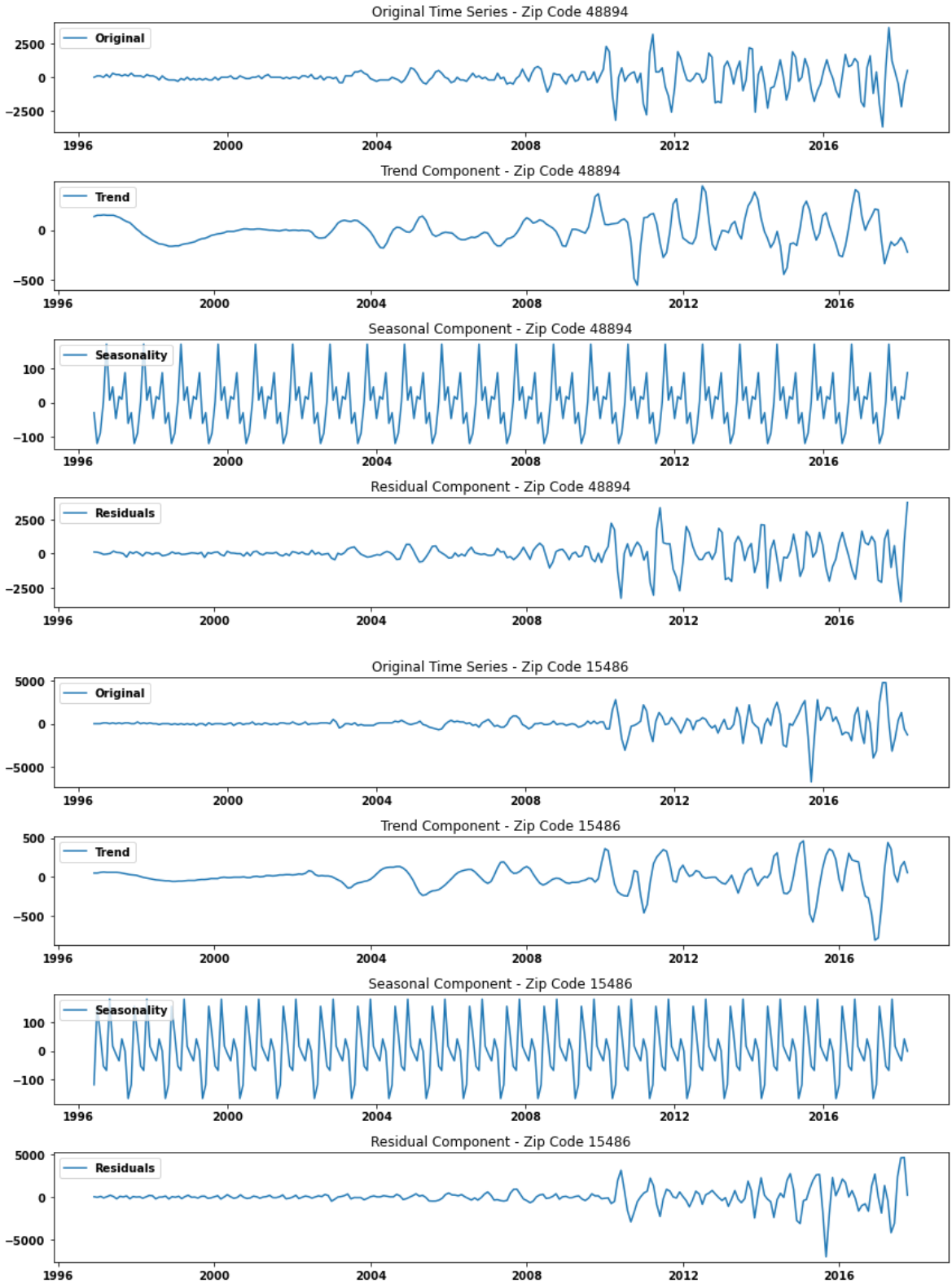

Original Time Series - Zip Code 29645
Original
1000
0
1996
2000
2004
2008
2012
2016


Trend Component - Zip Code 29645
Trend
200
100
0
−100
1996
2000
2004
2008
2012
2016


Seasonal Component - Zip Code 29645
Seasonality
50
0
−50
1996
2000
2004
2008
2012
2016


Residual Component - Zip Code 29645
Residuals
1000
500
0
−500
1996
2000
2004
2008
2012
2016


Original Time Series - Zip Code 66206
Original
Trend Component - Zip Code 66206
Trend
Seasonal Component - Zip Code 66206
Seasonality
Residual Component - Zip Code 66206
Residuals
Original Time Series - Zip Code 48835
Original
Trend Component - Zip Code 48835
Trend
Seasonal Component - Zip Code 48835
Seasonality
Residual Component - Zip Code 48835
Residuals
1996
2000
2004
2008
2012
2016



- From our seasonal decomposition above, we can denote seasonality in all our zipcodes by viewing the `seasonal component` plots.
- Next up, using the best model parameters from our `auto_arima` , we will model our time series data using SARIMA which is more complex and advanced than ARIMA.
- SARIMA extends the capabilities of ARIMA by incorporating seasonal components of the time series data into the model.

In [37]:
```
# Visualizing the split data
train_size = int(len(zipcode_df) * 0.8)
train, test = zipcode_df[:train_size], zipcode_df[train_size:]

# Plotting
plt.figure(figsize=(10, 6))
plt.plot(zipcode_df.index, zipcode_df['differenced_ret'], label='Original Dat
plt.plot(train.index, train['differenced_ret'], label='Train Set', color='gre
plt.plot(test.index, test['differenced_ret'], label='Test Set', color='red')
plt.xlabel('Date')
plt.ylabel('Value')
plt.title('Train-Test Split of Time Series Data')
plt.legend()
plt.show()
```

5.2 SARIMA Modeling

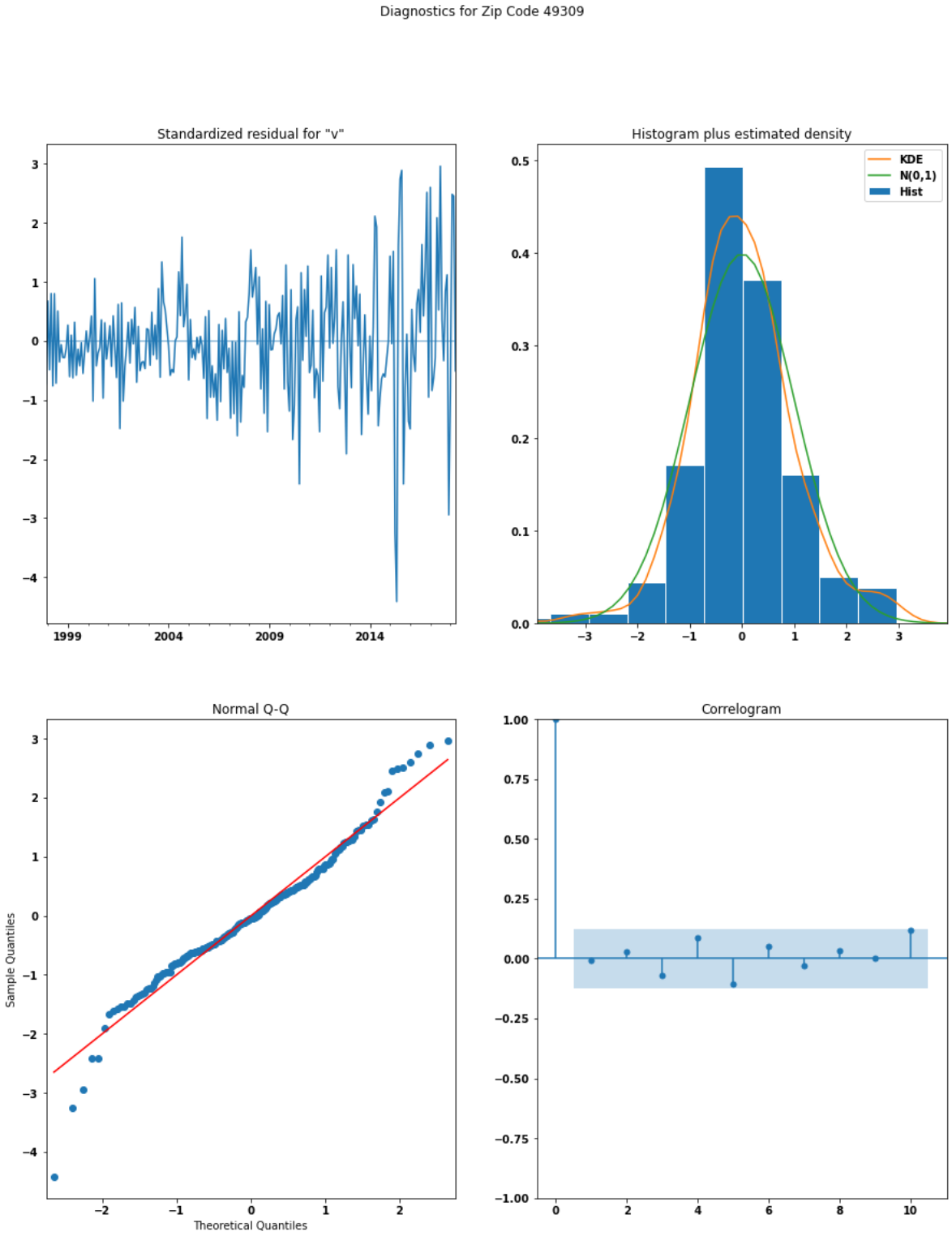
In [38]:
```
# Create a dictionary of the best parameters from auto_arima
best_params_by_zipcode = {
    49309: ((4, 2, 1), (1, 0, 1, 12)),
    40107: ((3, 1 ,3), (1, 0, 1, 12)),
    48822: ((1, 2, 2), (1, 0, 1, 12)),
    49265: ((1, 2, 3), (0, 0, 2, 12)),
    49425: ((3, 2, 3), (0, 0, 2, 12)),
    29645: ((1, 2, 3), (1, 0, 1, 12)),
    66206: ((2, 1, 1), (0, 0, 0, 12)),
    48835: ((0, 2, 3), (2, 0, 1, 12)),
    48894: ((2, 2, 1), (0, 0, 1, 12)),
    15486: ((2, 1, 0), (2, 0, 0, 12))
}
```

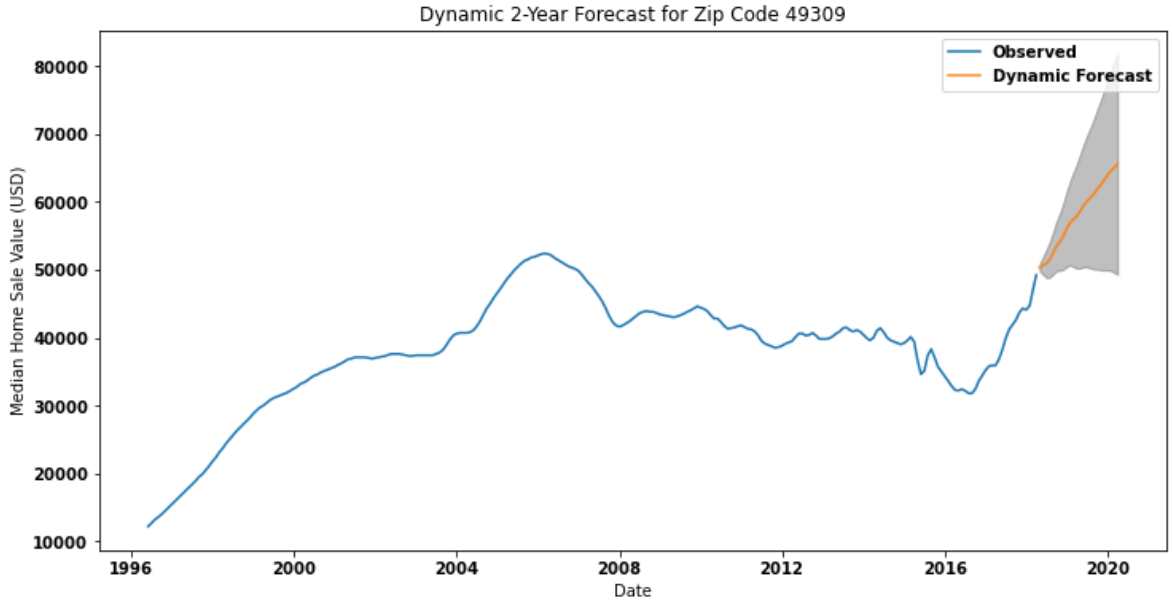
In [39]:
```
# Define a function for SARIMA modeling
def build_sarima_model(ts, order=(1, 1, 1), seasonal_order=(1, 1, 1, 12), nei
    # Use the best parameters
    if ts['Zipcode'][0] in best_params_by_zipcode:
        order, seasonal_order = best_params_by_zipcode[ts['Zipcode'][0]]

    # SARIMA model
    model = SARIMAX(ts['value'],
                    order=order,
                    seasonal_order=seasonal_order,
                    enforce_stationarity=False,
                    enforce_invertibility=False)

    # Fit the model and print results
    output = model.fit(disp=False)

    # Print output summary
    print(f"Summary for {neighborhood}:")
    print(output.summary())

    # Plot diagnostics
    output.plot_diagnostics(figsize=(15, 18))
    plt.suptitle(f"Diagnostics for {neighborhood}")
    plt.show()

    return output


# Define a function for one step forecasting
def ose_forecast_adjusted(ts, output, neighborhood=''):
    # Get predictions starting from 2017-04-01 and calculate confidence inter
    pred = output.get_prediction(start=pd.to_datetime('2017-04-01'), dynamic=

    # Get the real and predicted values
    ts_forecasted = pred.predicted_mean
    ts_truth = ts['2017-04-01':]

    # Calculate RMSE
    mse = mean_squared_error(ts_truth, ts_forecasted)
    rmse = np.sqrt(mse)
    print(f'The RMSE of forecasts for {neighborhood} is {round(rmse, 2)}')

    # Confidence Intervals
    pred_conf = pred.conf_int()

    # Plot real vs predicted with confidence intervals
    plt.figure(figsize=(15, 6))
    plt.plot(ts['2016-04-01':], label='observed')
    plt.plot(ts_forecasted, label='one-step ahead forecast', alpha=0.5)
    plt.fill_between(pred_conf.index, pred_conf.iloc[:, 0], pred_conf.iloc[:,
    plt.title(f"{neighborhood} Forecast vs. Observed")
    plt.xlabel('Date')
    plt.ylabel('Median Home Value (USD)')
    plt.legend()
    plt.show()

    return {'rmse': rmse}
```

```
# Define a function for dynamic forecasting
def dynamic_forecast(ts, output, years=2, neighborhood=''):
    # Calculate steps
    steps = years * 12

    # Get forecast and confidence interval for steps ahead in the future
    future = output.get_forecast(steps=steps, dynamic=True, full_results=True
    future_conf = future.conf_int()

    # Plot forecast
    plt.figure(figsize=(12, 6))
    plt.plot(ts, label='Observed')
    plt.plot(future.predicted_mean, label='Dynamic Forecast', alpha=0.9)
    plt.fill_between(future_conf.index, future_conf.iloc[:, 0], future_conf.i
    plt.title(f"Dynamic {years}-Year Forecast for {neighborhood}")
    plt.xlabel('Date')
    plt.ylabel('Median Home Sale Value (USD)')
    plt.legend()
    plt.show()

    # Forecast prediction for n-years into the future
    forecast = future.predicted_mean[-1]
    maximum = future_conf.iloc[-1, 1]
    minimum = future_conf.iloc[-1, 0]

    # Create dictionary of predictions
    predictions = {
        'forecast': forecast.round(),
        'minimum': minimum.round(),
        'maximum': maximum.round()
    }

    return predictions
```

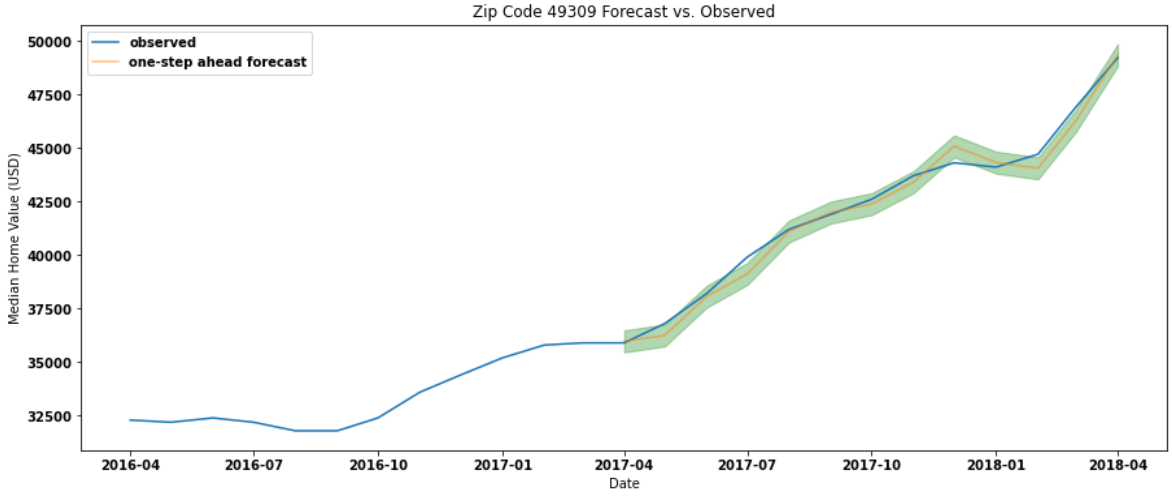
In [40]:
```
# Iterate over the dictionary of zip code dataframes
for zipcode, zipcode_df in zipcode_dataframes.items():
    # Build SARIMA model
    order = (1, 1, 1)
    seasonal_order = (1, 1, 1, 12)
    output = build_sarima_model(zipcode_df, order, seasonal_order, neighborho

    # Perform dynamic forecast
    dynamic_forecast(zipcode_df['value'], output, years=2, neighborhood=f'Zip

    # Perform OSE forecast adjusted
    ose_forecast_adjusted(zipcode_df['value'], output, neighborhood=f'Zip Cod
```

```
Summary for Zip Code 49309:
                                     SARIMAX Results
==========================================================================
==============
Dep. Variable:                               value   No. Observations:
263
Model:             SARIMAX(4, 2, 1)x(1, 0, 1, 12)   Log Likelihood
-1716.373
Date:                            Wed, 30 Aug 2023   AIC
3448.747
Time:                                    21:22:55   BIC
3476.757
Sample:                                06-01-1996   HQIC
3460.027
                                     - 04-01-2018
Covariance Type:                               opg
==========================================================================
==
                 coef    std err          z      P>|z|      [0.025      0.97
5]
--------------------------------------------------------------------------
--
ar.L1          1.1066      0.116      9.578      0.000       0.880       1.3
33
ar.L2         -1.0372      0.074    -13.989      0.000      -1.183      -0.8
92
ar.L3          0.4355      0.096      4.525      0.000       0.247       0.6
24
ar.L4         -0.1279      0.062     -2.049      0.040      -0.250      -0.0
06
ma.L1         -0.7358      0.109     -6.755      0.000      -0.949      -0.5
22
ar.S.L12       0.4311      0.157      2.740      0.006       0.123       0.7
40
ma.S.L12      -0.7255      0.137     -5.287      0.000      -0.994      -0.4
57
sigma2        6.9e+04   5324.440     12.958      0.000    5.86e+04    7.94e+
04
==========================================================================
=======
Ljung-Box (L1) (Q):                   0.01   Jarque-Bera (JB):
45.38
Prob(Q):                              0.91   Prob(JB):
0.00
Heteroskedasticity (H):               6.09   Skew:
-0.09
Prob(H) (two-sided):                  0.00   Kurtosis:
5.10
==========================================================================
=======

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

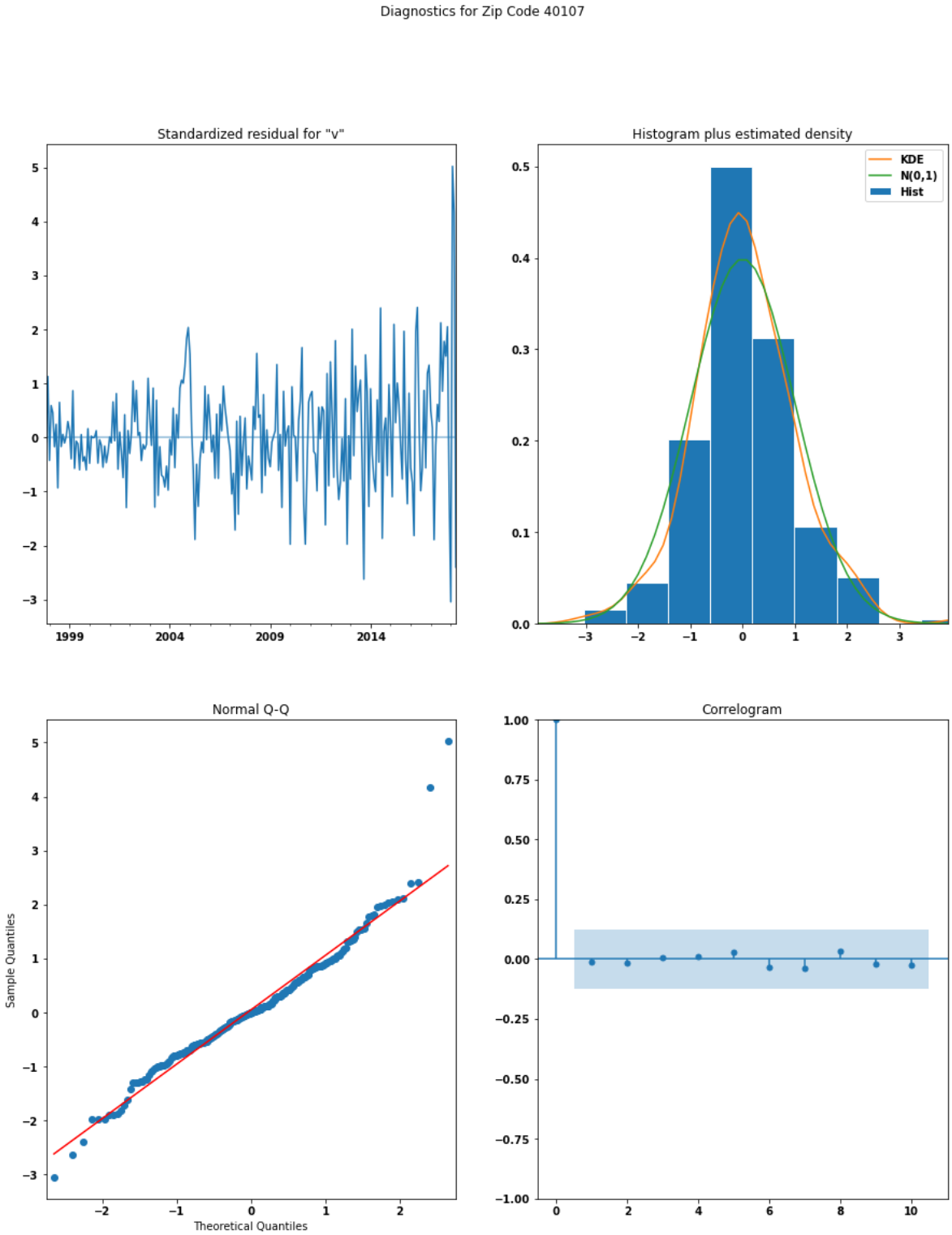

Diagnostics for Zip Code 49309
Standardized residual for "v"
Histogram plus estimated density
KDE
N(0,1)
Hist
Normal Q-Q
Sample Quantiles
Theoretical Quantiles
Correlogram

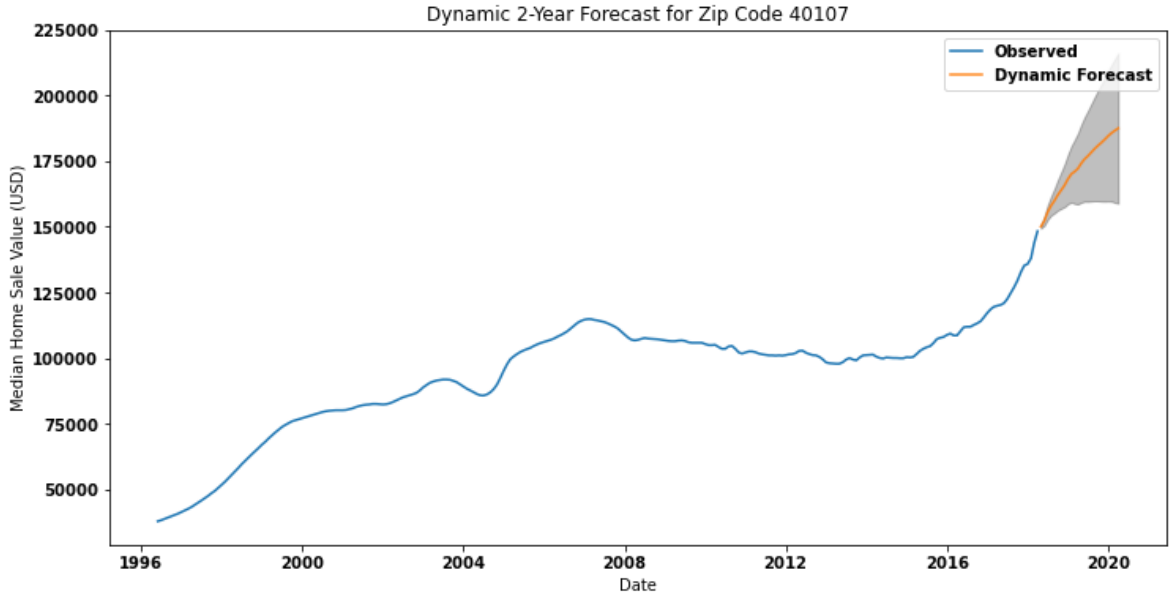


The RMSE of forecasts for Zip Code 49309 is 446.12

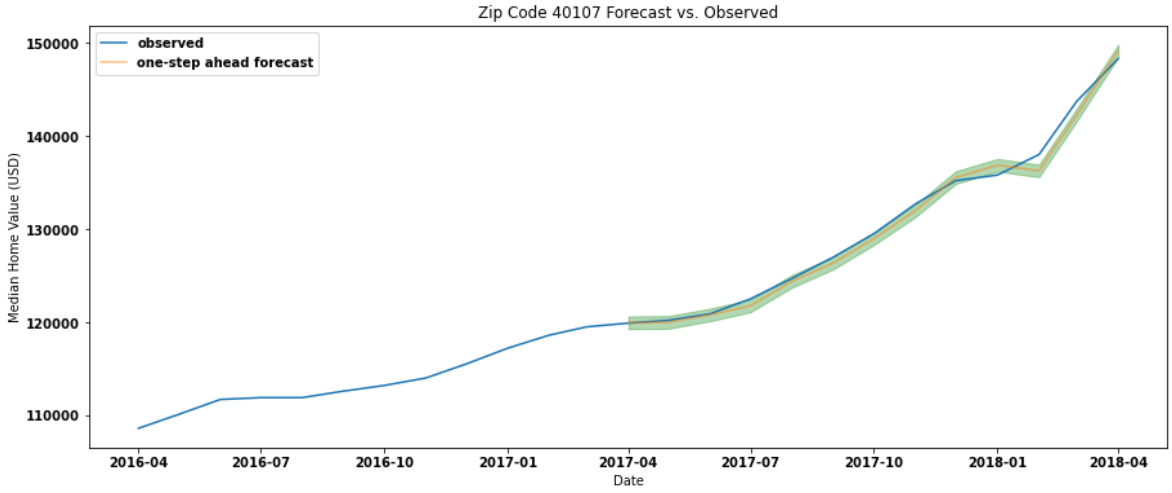


```
Summary for Zip Code 40107:
                                     SARIMAX Results
==========================================================================
================
Dep. Variable:                                value   No. Observations:
263
Model:             SARIMAX(3, 1, 3)x(1, 0, [1], 12)   Log Likelihood
-1793.502
Date:                              Wed, 30 Aug 2023   AIC
3605.003
Time:                                     21:22:58   BIC
3636.551
Sample:                                 06-01-1996   HQIC
3617.706
                                      - 04-01-2018
Covariance Type:                              opg
==========================================================================
==
                 coef    std err          z      P>|z|      [0.025      0.97
5]
--------------------------------------------------------------------------
--
ar.L1          0.6216      0.086      7.207      0.000       0.453       0.7
91
ar.L2         -0.0626      0.093     -0.672      0.502      -0.245       0.1
20
ar.L3          0.4189      0.076      5.495      0.000       0.269       0.5
68
ma.L1          0.7519      0.104      7.231      0.000       0.548       0.9
56
ma.L2          0.0046      0.135      0.034      0.973      -0.260       0.2
70
ma.L3         -0.4698      0.086     -5.470      0.000      -0.638      -0.3
01
ar.S.L12       0.2996      0.149      2.008      0.045       0.007       0.5
92
ma.S.L12      -0.6727      0.127     -5.311      0.000      -0.921      -0.4
24
sigma2      1.205e+05   9464.094     12.733      0.000    1.02e+05    1.39e+
05
==========================================================================
=======
Ljung-Box (L1) (Q):                   0.02   Jarque-Bera (JB):
108.77
Prob(Q):                              0.88   Prob(JB):
0.00
Heteroskedasticity (H):               6.65   Skew:
0.65
Prob(H) (two-sided):                  0.00   Kurtosis:
5.98
==========================================================================
=======

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

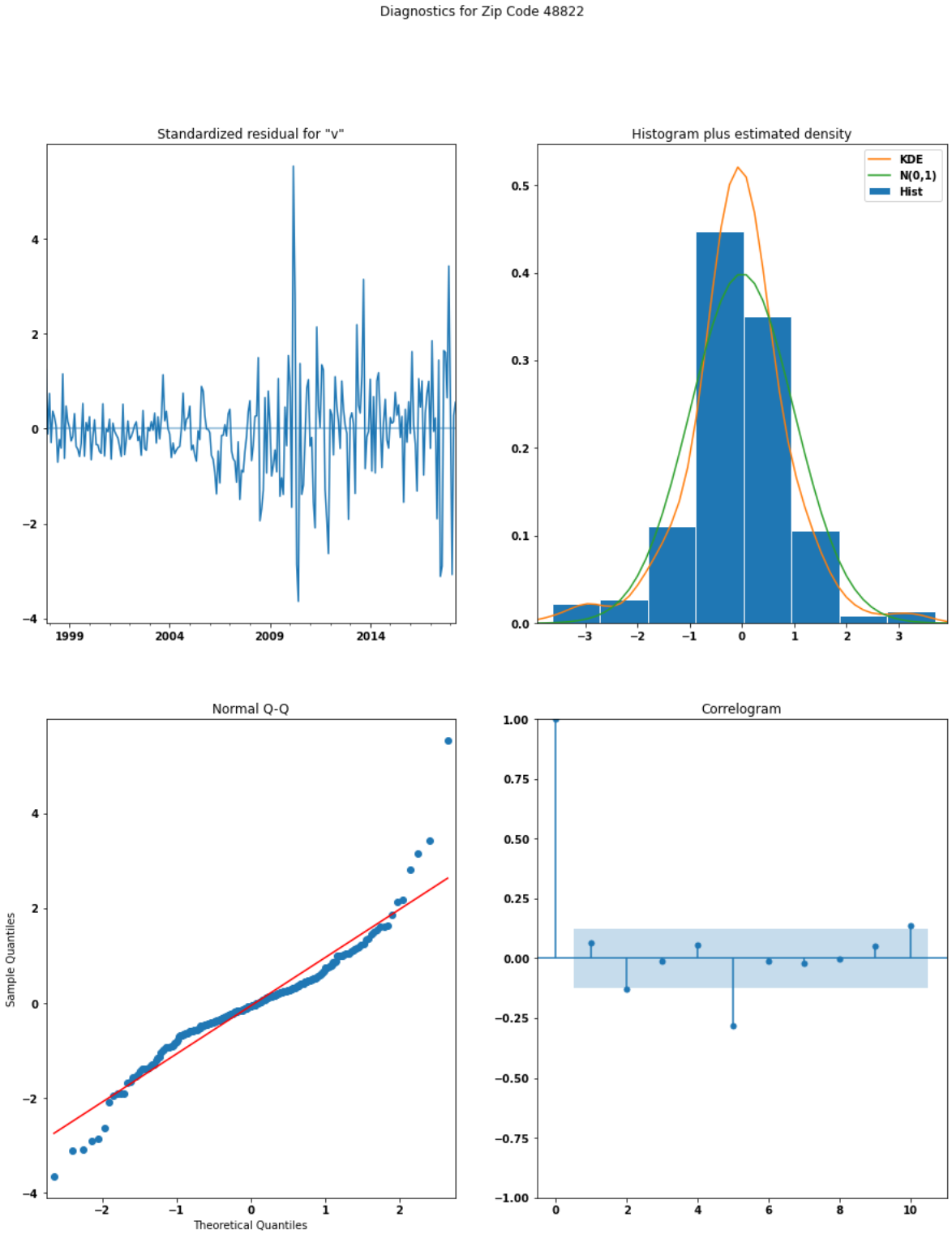

Diagnostics for Zip Code 40107
Standardized residual for "v"
Histogram plus estimated density
KDE
N(0,1)
Hist
Normal Q-Q
Sample Quantiles
Theoretical Quantiles
Correlogram

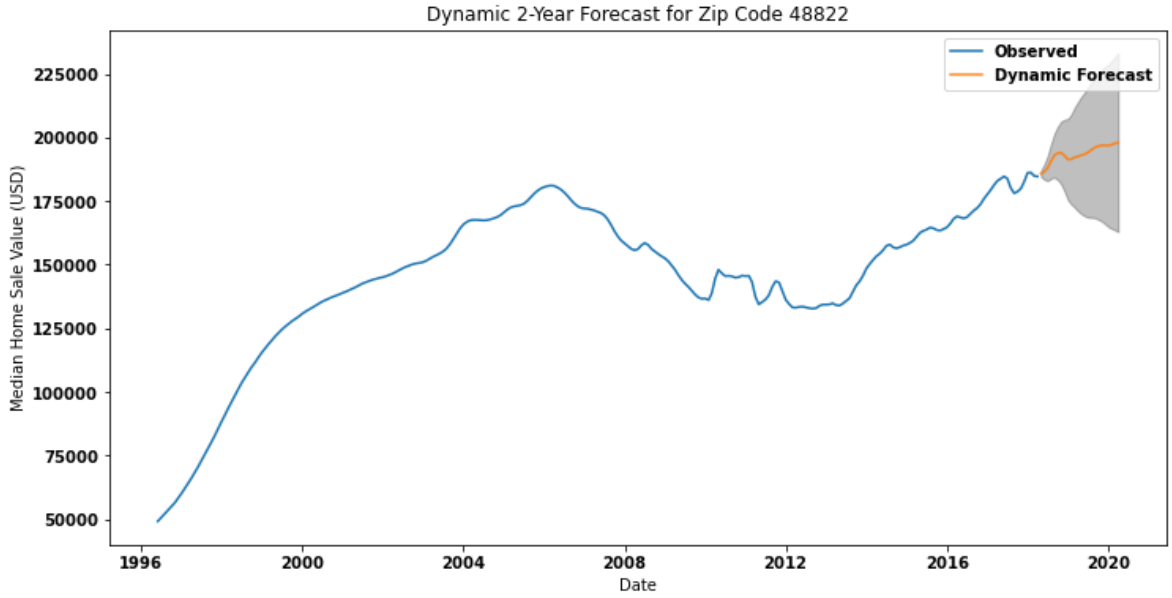


The RMSE of forecasts for Zip Code 40107 is 828.01

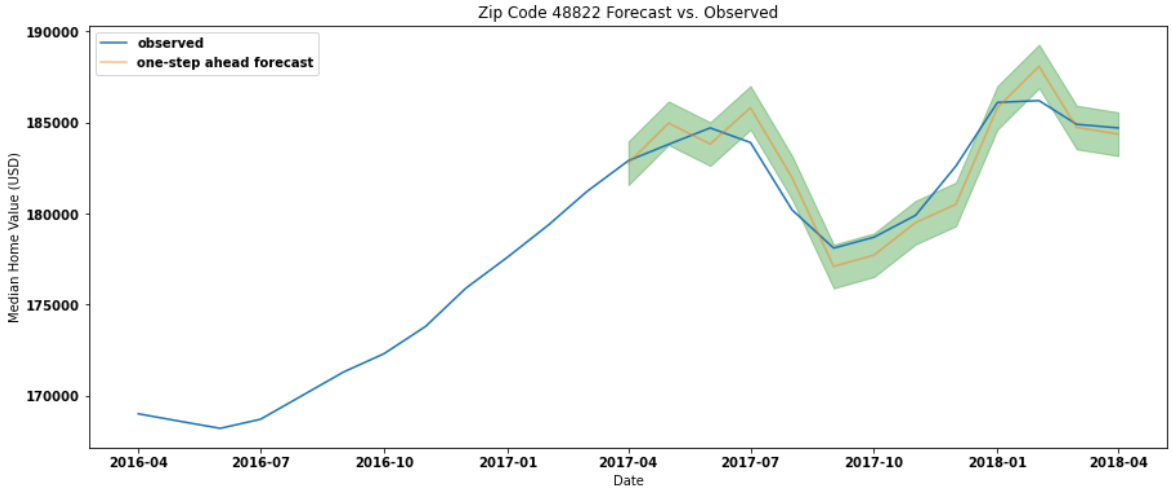


```
Summary for Zip Code 48822:
                                     SARIMAX Results
==========================================================================
================
Dep. Variable:                                value   No. Observations:
263
Model:             SARIMAX(1, 2, 2)x(1, 0, [1], 12)   Log Likelihood
-1938.654
Date:                              Wed, 30 Aug 2023   AIC
3889.308
Time:                                     21:23:00   BIC
3910.340
Sample:                                 06-01-1996   HQIC
3897.777
                                      - 04-01-2018
Covariance Type:                              opg
==========================================================================
==
                 coef    std err          z      P>|z|      [0.025      0.97
5]
--------------------------------------------------------------------------
--
ar.L1          0.5724      0.058      9.913      0.000       0.459       0.6
86
ma.L1         -0.2341      0.057     -4.076      0.000      -0.347      -0.1
22
ma.L2         -0.6121      0.043    -14.394      0.000      -0.695      -0.5
29
ar.S.L12       0.2682      0.071      3.794      0.000       0.130       0.4
07
ma.S.L12      -0.8695      0.062    -13.939      0.000      -0.992      -0.7
47
sigma2      3.744e+05   2.31e+04     16.234      0.000    3.29e+05     4.2e+
05
==========================================================================
=======
Ljung-Box (L1) (Q):                   1.01   Jarque-Bera (JB):
259.22
Prob(Q):                              0.32   Prob(JB):
0.00
Heteroskedasticity (H):               8.12   Skew:
0.45
Prob(H) (two-sided):                  0.00   Kurtosis:
7.95
==========================================================================
=======

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

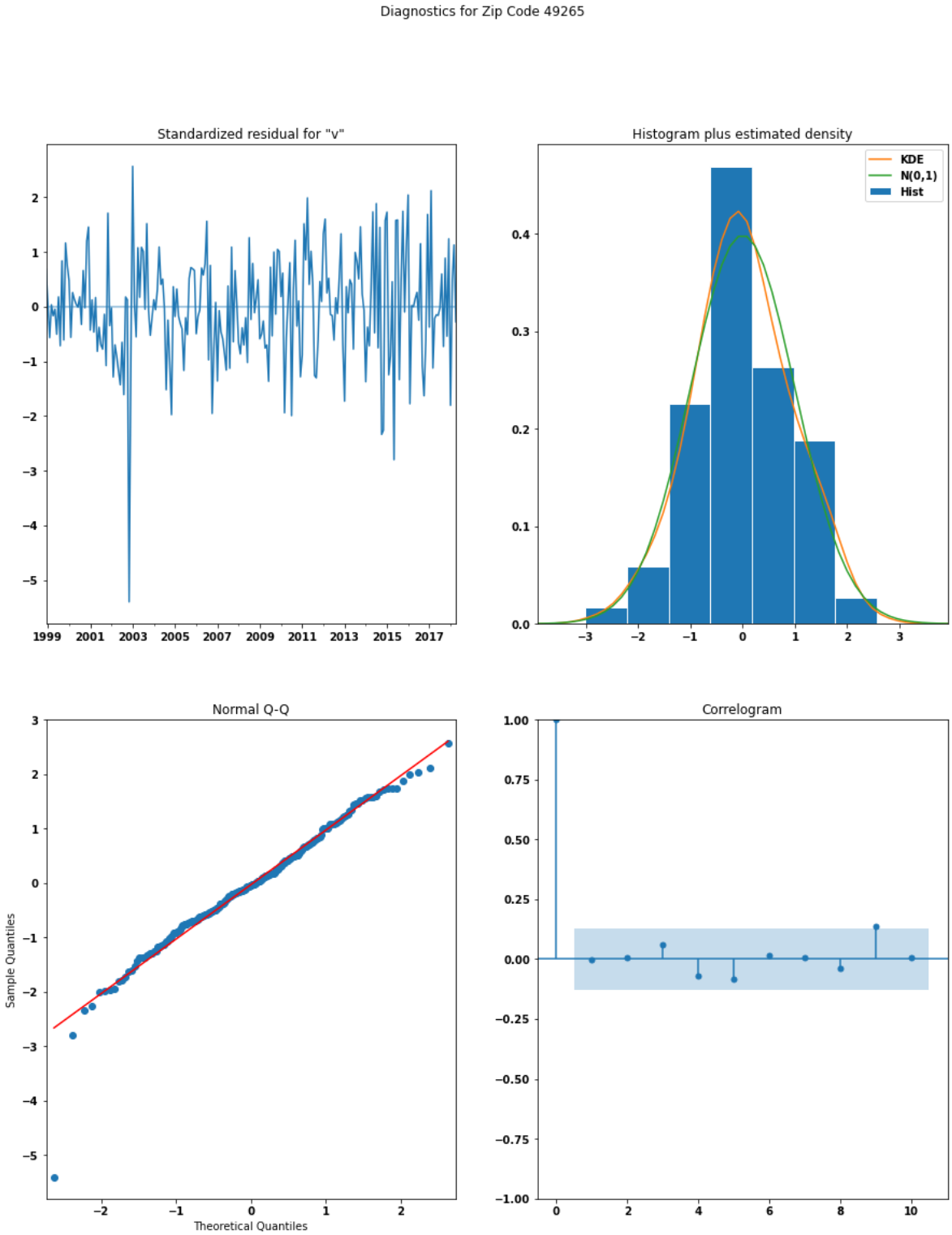

Diagnostics for Zip Code 48822
Standardized residual for "v"
Histogram plus estimated density
KDE
N(0,1)
Hist
Normal Q-Q
Sample Quantiles
Theoretical Quantiles
Correlogram

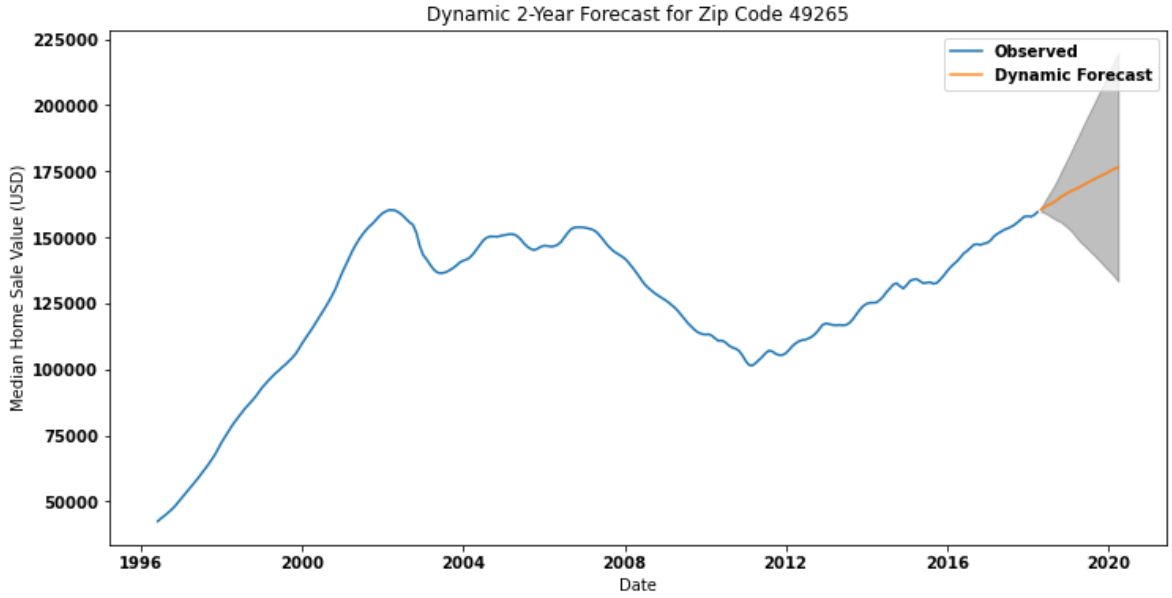


The RMSE of forecasts for Zip Code 48822 is 1216.52

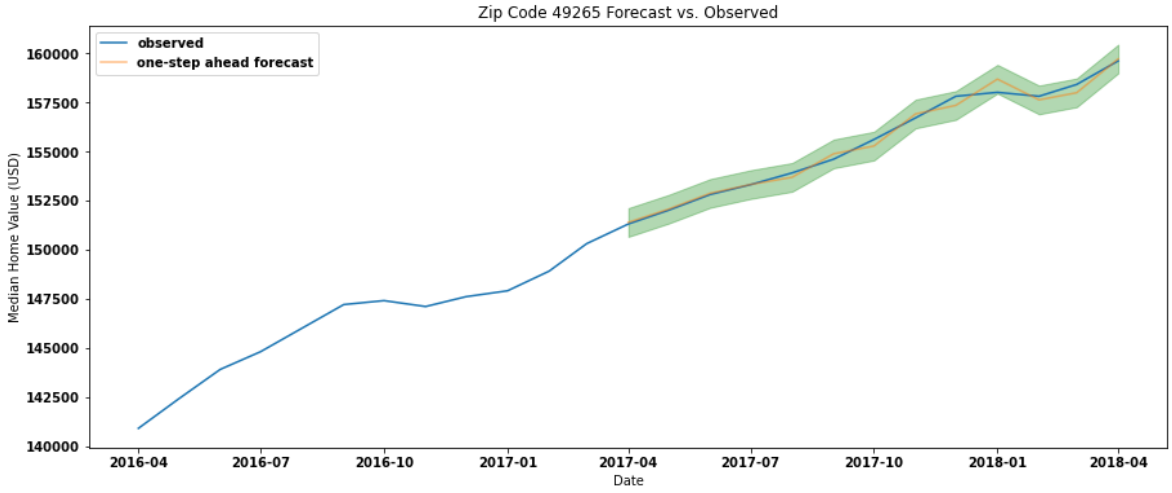


```
Summary for Zip Code 49265:
                                     SARIMAX Results
==========================================================================
==================
Dep. Variable:                                 value   No. Observations:
263
Model:             SARIMAX(1, 2, 3)x(0, 0, [1, 2], 12)   Log Likelihood
-1713.290
Date:                                 Wed, 30 Aug 2023   AIC
3440.580
Time:                                         21:23:04   BIC
3464.737
Sample:                                     06-01-1996   HQIC
3450.321
                                          - 04-01-2018
Covariance Type:                                  opg
==========================================================================
==
                 coef    std err          z      P>|z|      [0.025      0.97
5]
--------------------------------------------------------------------------
--
ar.L1         -0.1221      0.148     -0.824      0.410      -0.413       0.1
68
ma.L1          0.6992      0.133      5.266      0.000       0.439       0.9
59
ma.L2         -0.2287      0.093     -2.464      0.014      -0.411      -0.0
47
ma.L3         -0.4515      0.048     -9.464      0.000      -0.545      -0.3
58
ma.S.L12      -0.3994      0.072     -5.549      0.000      -0.540      -0.2
58
ma.S.L24      -0.2583      0.067     -3.852      0.000      -0.390      -0.1
27
sigma2      1.391e+05   1.12e+04     12.471      0.000    1.17e+05    1.61e+
05
==========================================================================
=======
Ljung-Box (L1) (Q):                   0.00   Jarque-Bera (JB):
92.29
Prob(Q):                              0.98   Prob(JB):
0.00
Heteroskedasticity (H):               1.23   Skew:
-0.61
Prob(H) (two-sided):                  0.36   Kurtosis:
5.83
==========================================================================
=======

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

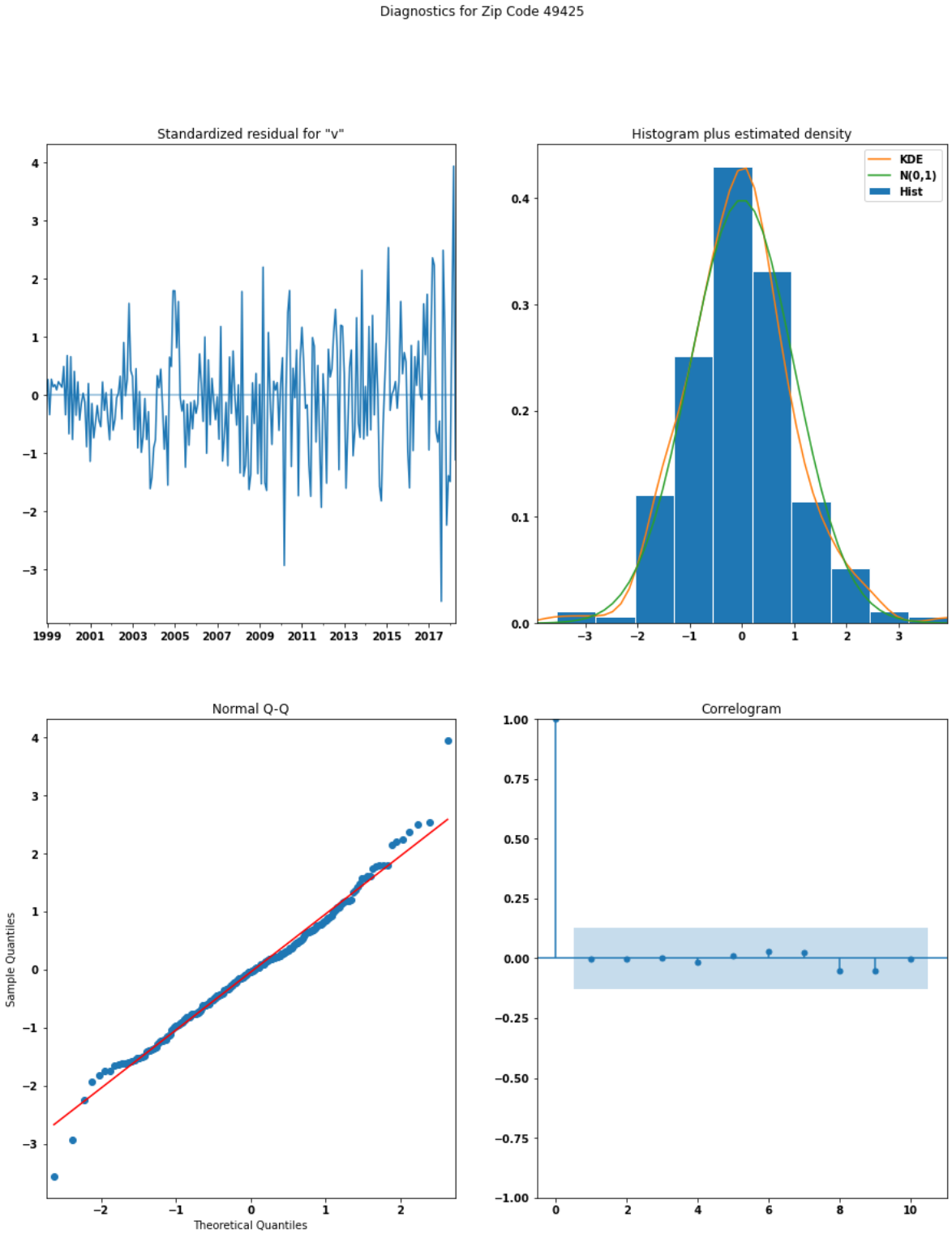

Diagnostics for Zip Code 49265
Standardized residual for "v"
Histogram plus estimated density
KDE
N(0,1)
Hist
Normal Q-Q
Sample Quantiles
Theoretical Quantiles
Correlogram

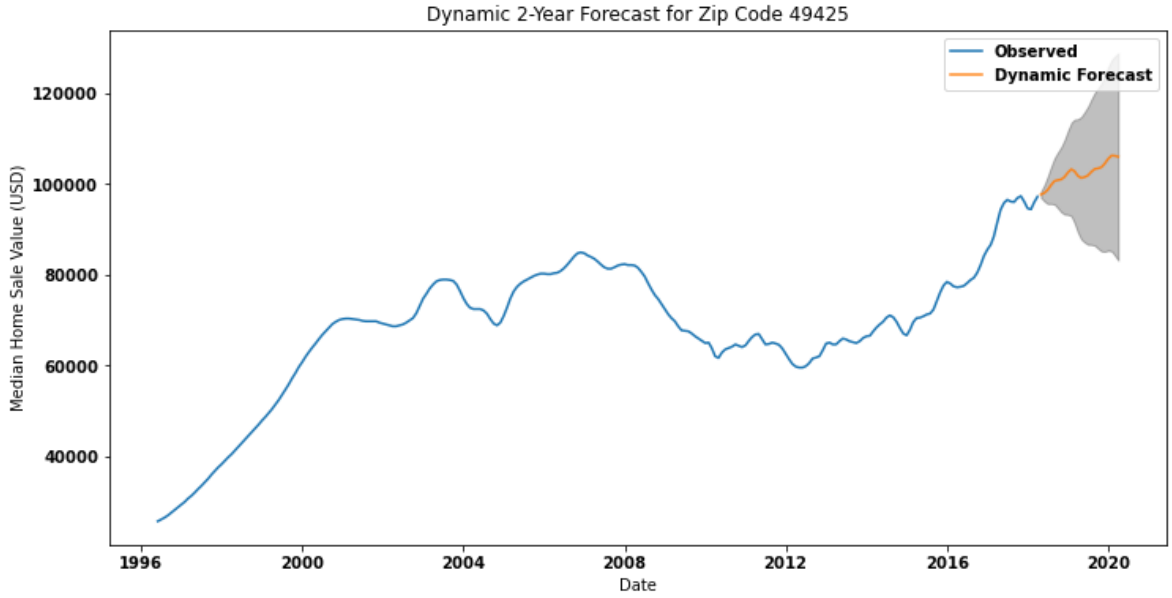


The RMSE of forecasts for Zip Code 49265 is 300.67

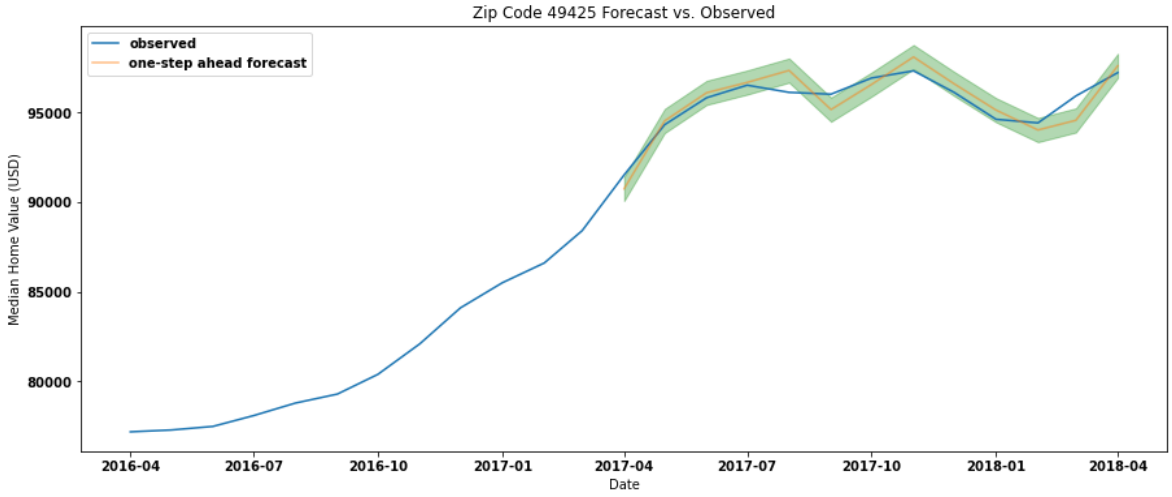


```
Summary for Zip Code 49425:
                                     SARIMAX Results
==========================================================================
==================
Dep. Variable:                                 value   No. Observations:
263
Model:             SARIMAX(3, 2, 3)x(0, 0, [1, 2], 12)   Log Likelihood
-1696.720
Date:                                 Wed, 30 Aug 2023   AIC
3411.440
Time:                                         21:23:08   BIC
3442.499
Sample:                                     06-01-1996   HQIC
3423.964
                                          - 04-01-2018
Covariance Type:                                  opg
==========================================================================
==
                 coef    std err          z      P>|z|      [0.025      0.97
5]
--------------------------------------------------------------------------
--
ar.L1          0.2581      0.196      1.315      0.189      -0.127       0.6
43
ar.L2         -0.1194      0.165     -0.723      0.470      -0.443       0.2
04
ar.L3          0.1189      0.107      1.111      0.267      -0.091       0.3
29
ma.L1          0.2284      0.189      1.208      0.227      -0.142       0.5
99
ma.L2         -0.4196      0.080     -5.268      0.000      -0.576      -0.2
63
ma.L3         -0.5033      0.091     -5.534      0.000      -0.682      -0.3
25
ma.S.L12      -0.4194      0.063     -6.607      0.000      -0.544      -0.2
95
ma.S.L24      -0.3873      0.082     -4.725      0.000      -0.548      -0.2
27
sigma2      1.181e+05   1.08e+04     10.956      0.000     9.7e+04    1.39e+
05
==========================================================================
=======
Ljung-Box (L1) (Q):                   0.00   Jarque-Bera (JB):
15.76
Prob(Q):                              0.98   Prob(JB):
0.00
Heteroskedasticity (H):               3.48   Skew:
0.22
Prob(H) (two-sided):                  0.00   Kurtosis:
4.19
==========================================================================
=======

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

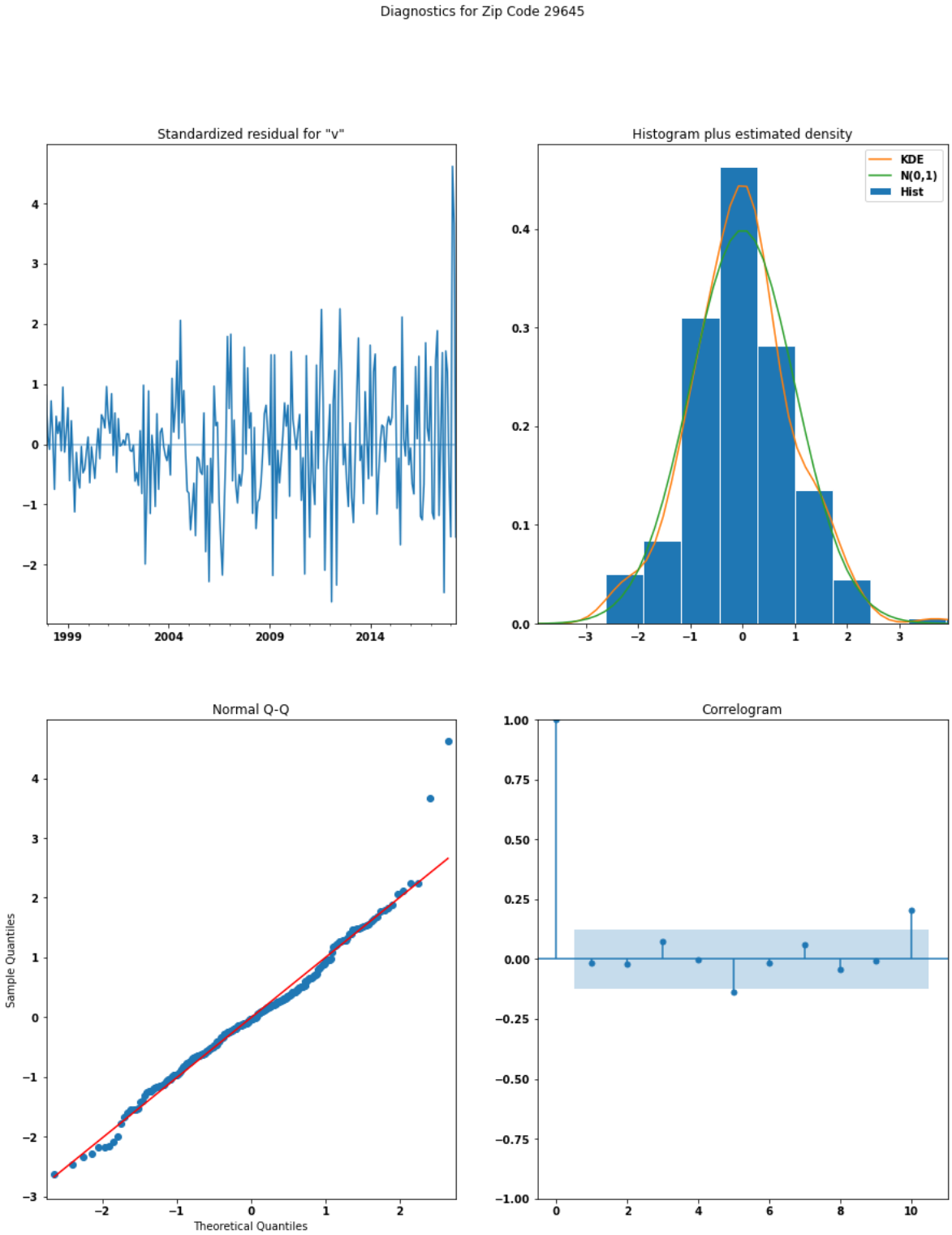

Diagnostics for Zip Code 49425
Standardized residual for "v"
Histogram plus estimated density
KDE
N(0,1)
Hist
Normal Q-Q
Sample Quantiles
Theoretical Quantiles
Correlogram

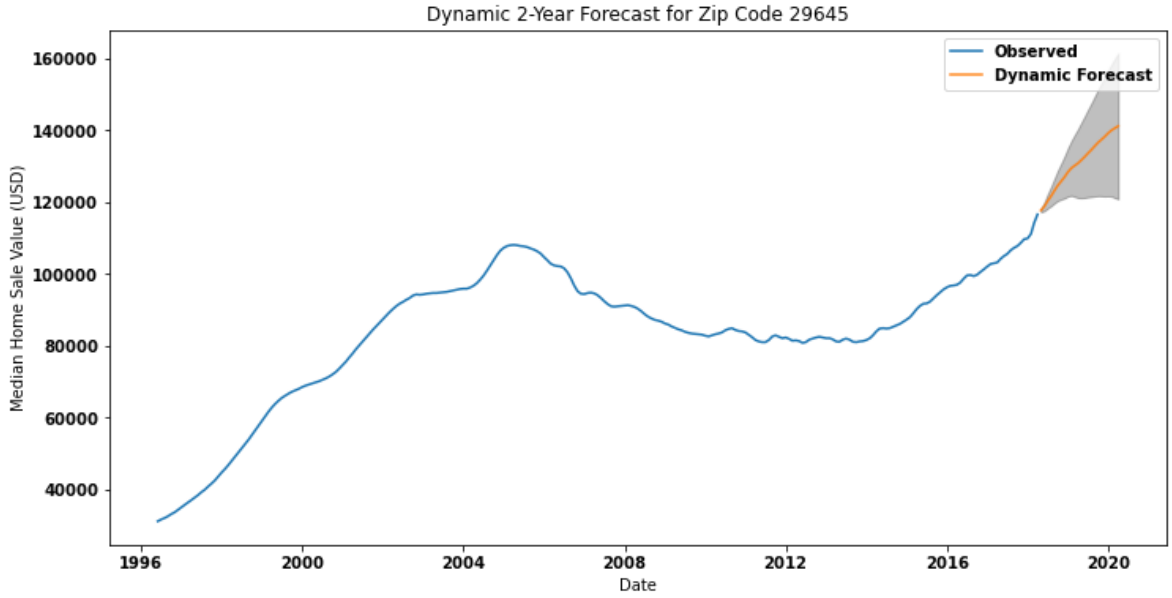


The RMSE of forecasts for Zip Code 49425 is 696.92

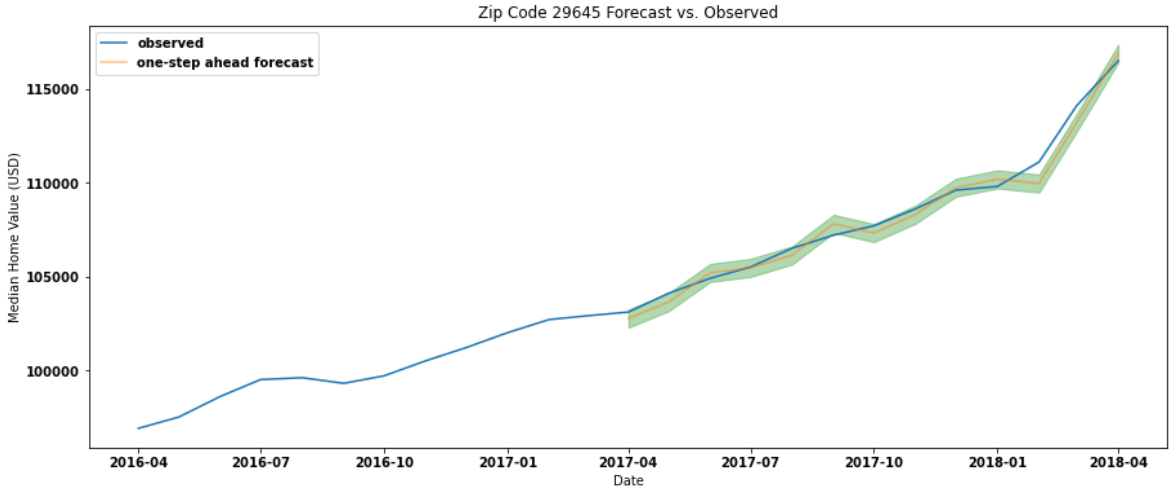


```
Summary for Zip Code 29645:
                                     SARIMAX Results
==========================================================================
================
Dep. Variable:                                value   No. Observations:
263
Model:             SARIMAX(1, 2, 3)x(1, 0, [1], 12)   Log Likelihood
-1708.597
Date:                              Wed, 30 Aug 2023   AIC
3431.195
Time:                                     21:23:10   BIC
3455.703
Sample:                                 06-01-1996   HQIC
3441.064
                                      - 04-01-2018
Covariance Type:                              opg
==========================================================================
==
                 coef    std err          z      P>|z|      [0.025      0.97
5]
--------------------------------------------------------------------------
--
ar.L1         -0.1938      0.113     -1.716      0.086      -0.415       0.0
27
ma.L1          0.5159      0.126      4.090      0.000       0.269       0.7
63
ma.L2         -0.3222      0.075     -4.312      0.000      -0.469      -0.1
76
ma.L3         -0.4707      0.061     -7.720      0.000      -0.590      -0.3
51
ar.S.L12       0.5297      0.102      5.181      0.000       0.329       0.7
30
ma.S.L12      -0.9090      0.093     -9.809      0.000      -1.091      -0.7
27
sigma2      6.084e+04   5242.997     11.604      0.000    5.06e+04    7.11e+
04
==========================================================================
=======
Ljung-Box (L1) (Q):                   0.06   Jarque-Bera (JB):
43.09
Prob(Q):                              0.81   Prob(JB):
0.00
Heteroskedasticity (H):               4.56   Skew:
0.46
Prob(H) (two-sided):                  0.00   Kurtosis:
4.84
==========================================================================
=======

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

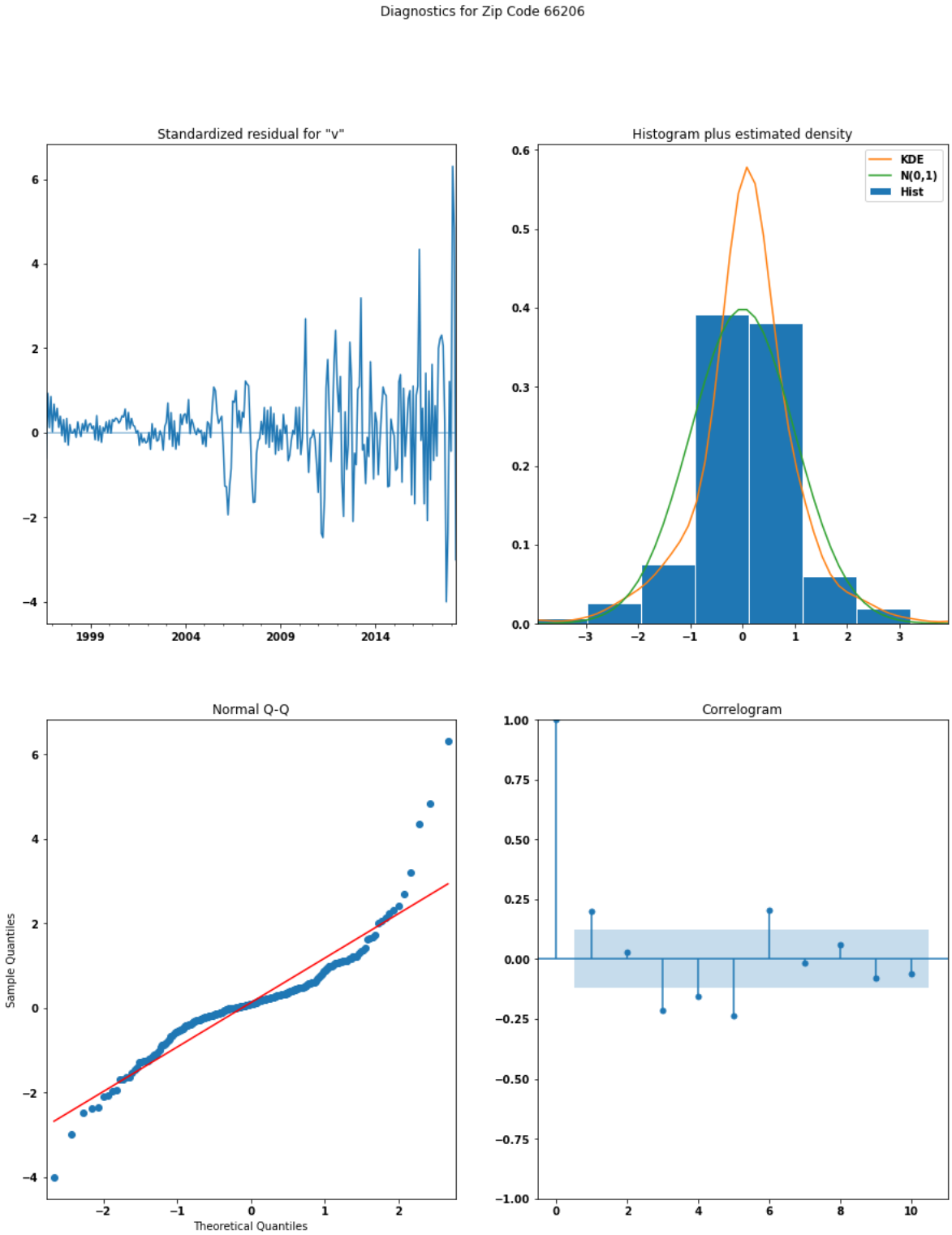

Diagnostics for Zip Code 29645
Standardized residual for "v"
Histogram plus estimated density
KDE
N(0,1)
Hist
Normal Q-Q
Sample Quantiles
Theoretical Quantiles
Correlogram

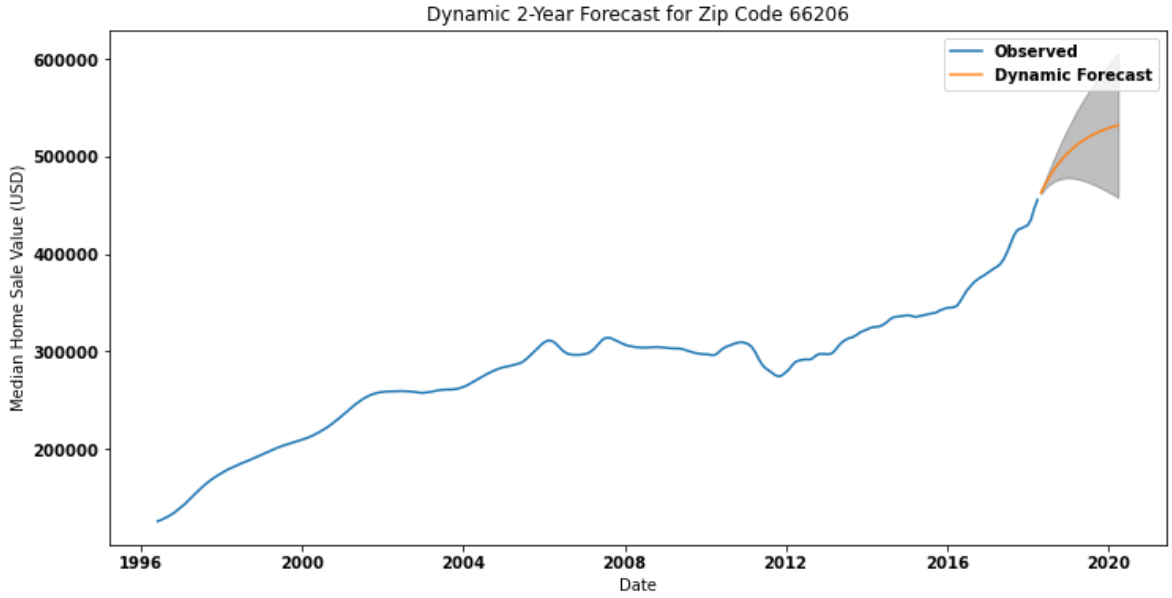


The RMSE of forecasts for Zip Code 29645 is 526.42

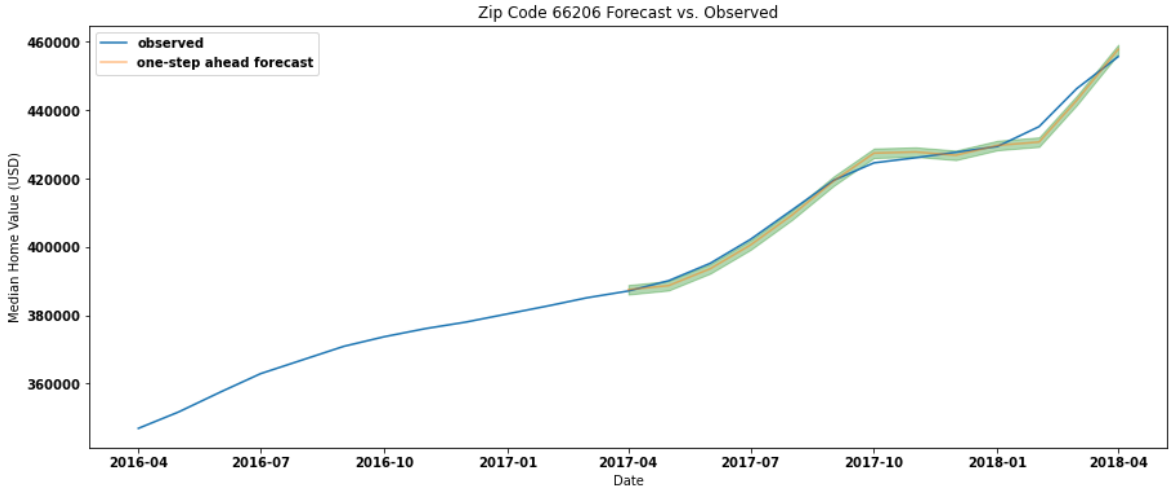


```
Summary for Zip Code 66206:
                               SARIMAX Results
==========================================================================
==
Dep. Variable:                   value   No. Observations:                   2
63
Model:               SARIMAX(2, 1, 1)   Log Likelihood               -2093.1
81
Date:                Wed, 30 Aug 2023   AIC                          4194.3
63
Time:                        21:23:12   BIC                          4208.6
06
Sample:                    06-01-1996   HQIC                         4200.0
89
                         - 04-01-2018
Covariance Type:                 opg
==========================================================================
==
                 coef    std err          z      P>|z|      [0.025      0.97
5]
--------------------------------------------------------------------------
--
ar.L1          0.7538      0.032     23.631      0.000       0.691       0.8
16
ar.L2          0.1458      0.030      4.899      0.000       0.087       0.2
04
ma.L1          0.6760      0.036     18.955      0.000       0.606       0.7
46
sigma2      5.082e+05   1.91e+04     26.578      0.000    4.71e+05    5.46e+
05
==========================================================================
=======
Ljung-Box (L1) (Q):                  10.53   Jarque-Bera (JB):
602.92
Prob(Q):                              0.00   Prob(JB):
0.00
Heteroskedasticity (H):              29.95   Skew:
0.97
Prob(H) (two-sided):                  0.00   Kurtosis:
10.20
==========================================================================
=======

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

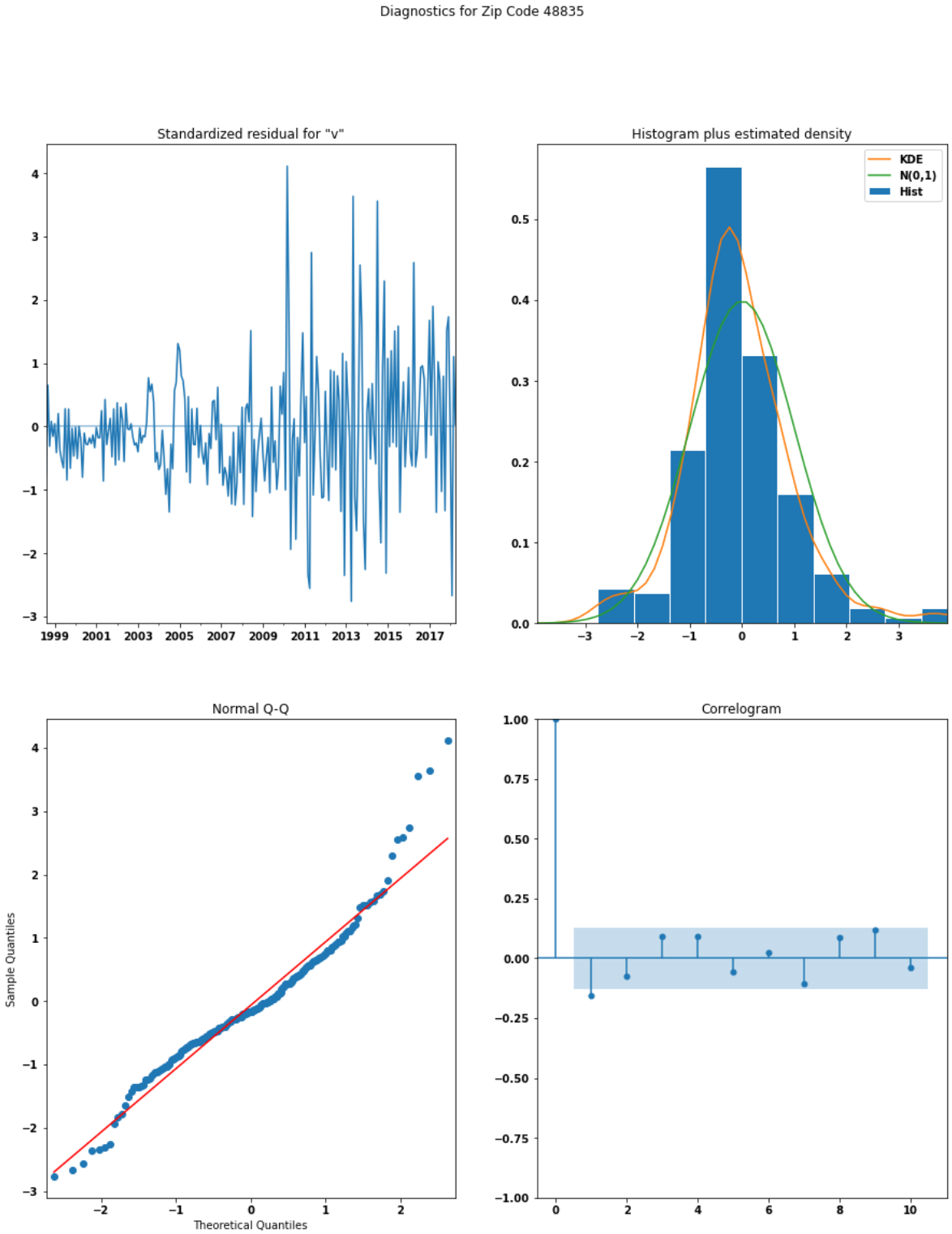

Diagnostics for Zip Code 66206
Standardized residual for "v"
Histogram plus estimated density
KDE
N(0,1)
Hist
Normal Q-Q
Sample Quantiles
Theoretical Quantiles
Correlogram

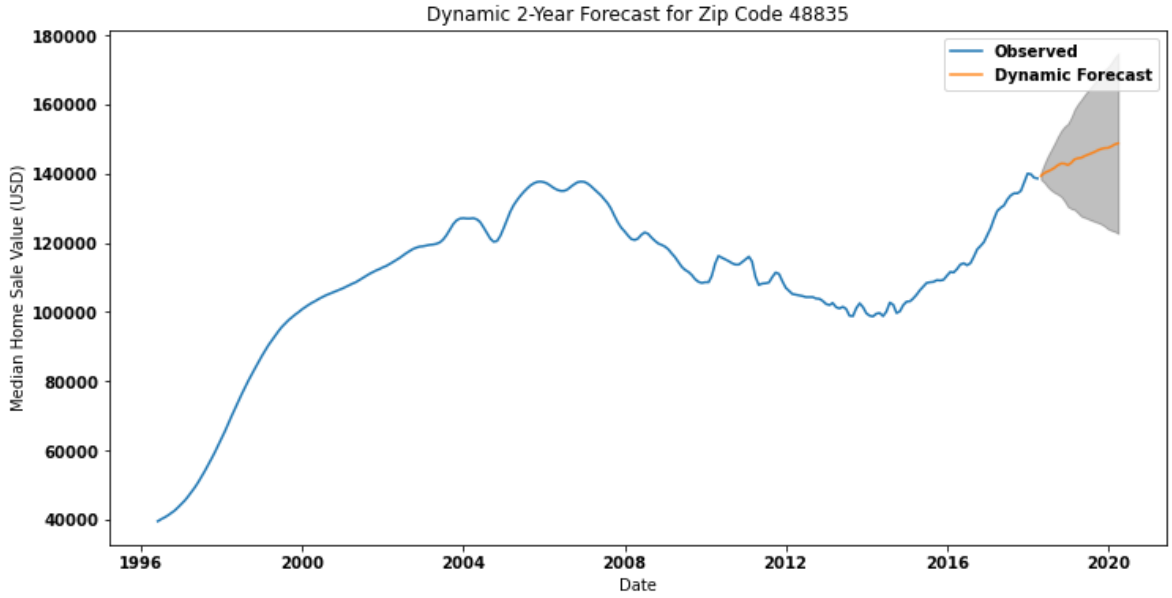


The RMSE of forecasts for Zip Code 66206 is 2112.8

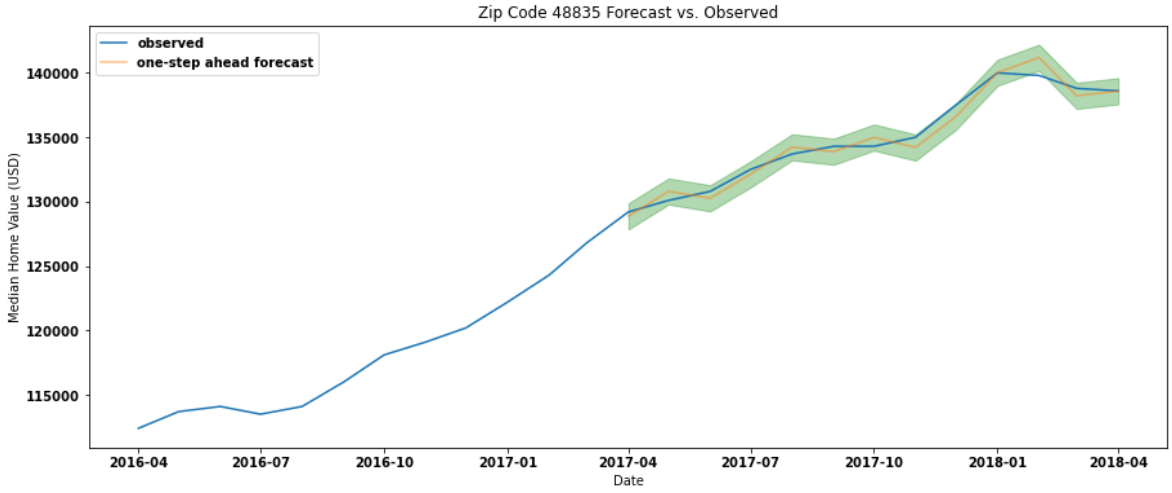


```
Summary for Zip Code 48835:
                                     SARIMAX Results
==========================================================================
================
Dep. Variable:                                value   No. Observations:
263
Model:             SARIMAX(0, 2, 3)x(2, 0, [1], 12)   Log Likelihood
-1826.506
Date:                              Wed, 30 Aug 2023   AIC
3667.012
Time:                                     21:23:15   BIC
3691.289
Sample:                                 06-01-1996   HQIC
3676.797
                                      - 04-01-2018
Covariance Type:                              opg
==========================================================================
==
                 coef    std err          z      P>|z|      [0.025      0.97
5]
--------------------------------------------------------------------------
--
ma.L1          0.4971      0.044     11.199      0.000       0.410       0.5
84
ma.L2         -0.5588      0.047    -11.835      0.000      -0.651      -0.4
66
ma.L3         -0.6451      0.045    -14.186      0.000      -0.734      -0.5
56
ar.S.L12       0.3906      0.083      4.705      0.000       0.228       0.5
53
ar.S.L24       0.0140      0.033      0.424      0.672      -0.051       0.0
79
ma.S.L12      -0.8882      0.075    -11.900      0.000      -1.034      -0.7
42
sigma2      2.689e+05   2.31e+04     11.661      0.000    2.24e+05    3.14e+
05
==========================================================================
=======
Ljung-Box (L1) (Q):                   5.93   Jarque-Bera (JB):
78.89
Prob(Q):                              0.01   Prob(JB):
0.00
Heteroskedasticity (H):               6.94   Skew:
0.68
Prob(H) (two-sided):                  0.00   Kurtosis:
5.48
==========================================================================
=======

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

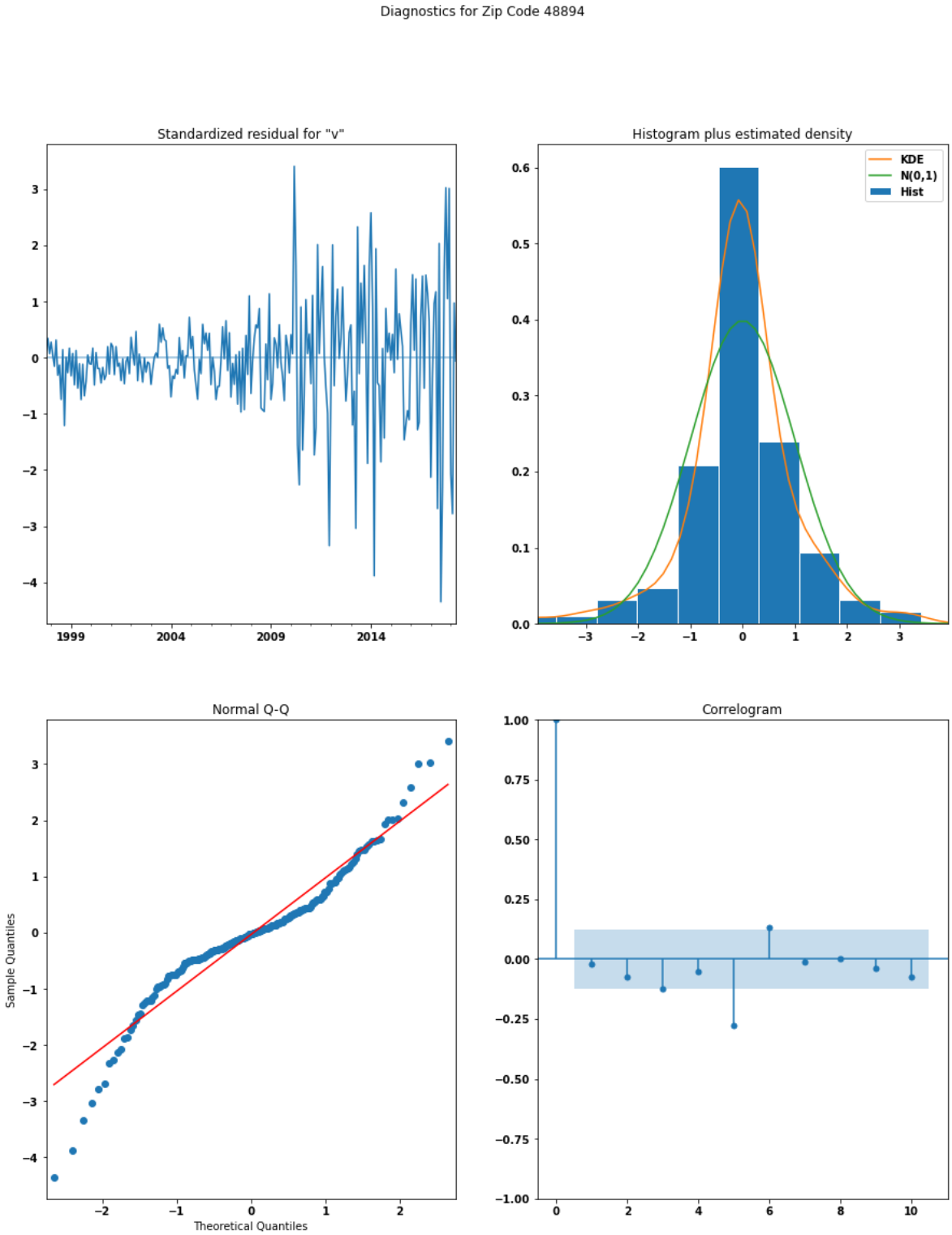

Diagnostics for Zip Code 48835
Standardized residual for "v"
Histogram plus estimated density
KDE
N(0,1)
Hist
Normal Q-Q
Sample Quantiles
Theoretical Quantiles
Correlogram

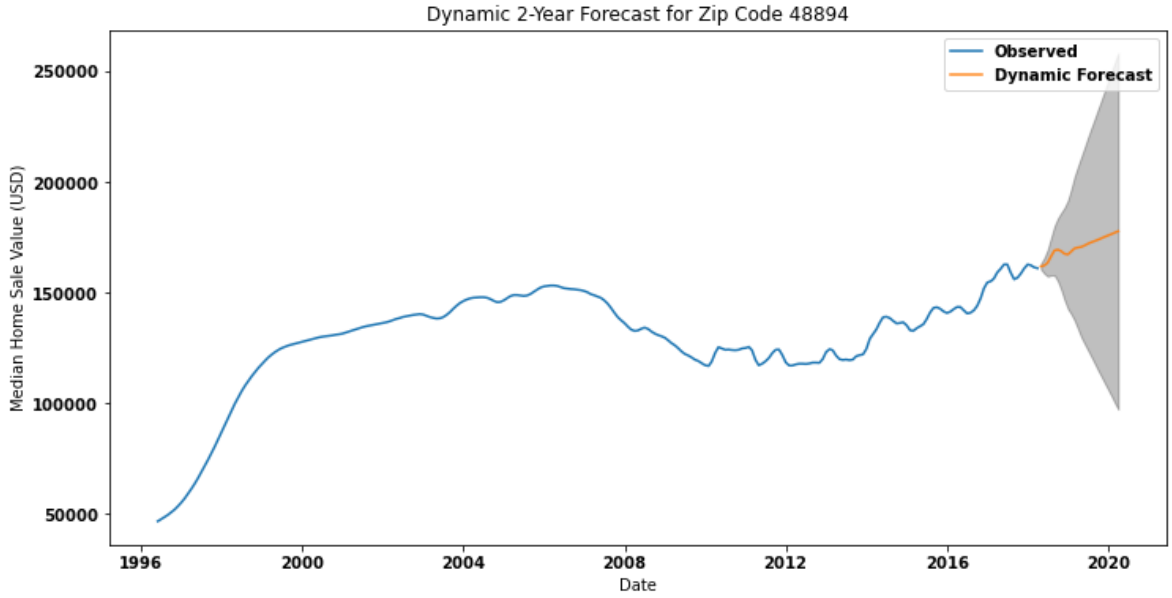


The RMSE of forecasts for Zip Code 48835 is 658.92

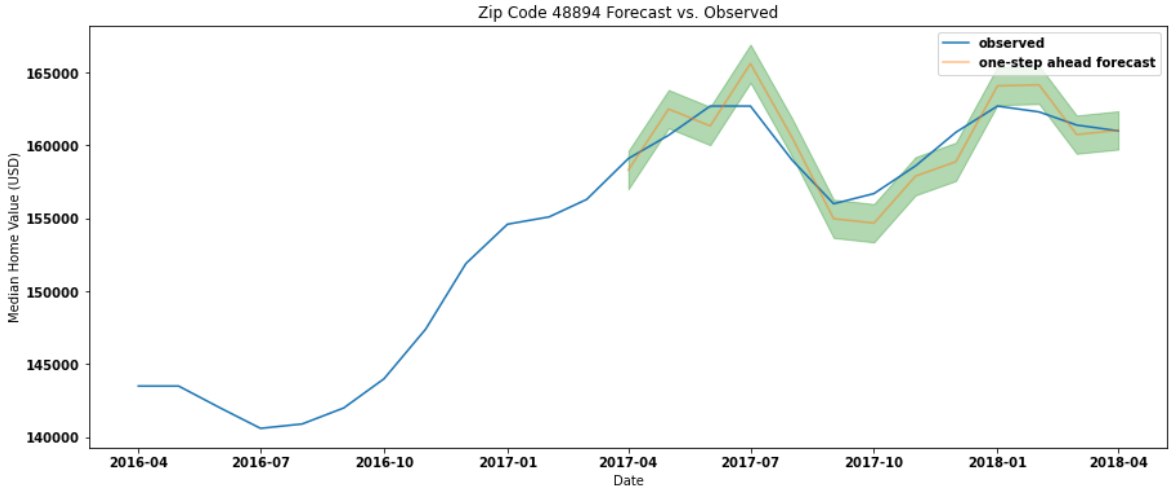


```
Summary for Zip Code 48894:
                                     SARIMAX Results
==========================================================================
==============
Dep. Variable:                               value   No. Observations:
263
Model:             SARIMAX(2, 2, 1)x(0, 0, 1, 12)   Log Likelihood
-1961.297
Date:                            Wed, 30 Aug 2023   AIC
3932.593
Time:                                    21:23:17   BIC
3950.140
Sample:                                06-01-1996   HQIC
3939.658
                                     - 04-01-2018
Covariance Type:                               opg
==========================================================================
==
                 coef    std err          z      P>|z|      [0.025      0.97
5]
--------------------------------------------------------------------------
--
ar.L1          0.2221      0.073      3.057      0.002       0.080       0.3
64
ar.L2         -0.4624      0.039    -11.857      0.000      -0.539      -0.3
86
ma.L1          0.3025      0.072      4.188      0.000       0.161       0.4
44
ma.S.L12      -0.5735      0.059     -9.761      0.000      -0.689      -0.4
58
sigma2      4.483e+05   2.59e+04     17.312      0.000    3.98e+05    4.99e+
05
==========================================================================
=======
Ljung-Box (L1) (Q):                   0.13   Jarque-Bera (JB):
114.42
Prob(Q):                              0.72   Prob(JB):
0.00
Heteroskedasticity (H):              20.13   Skew:
-0.41
Prob(H) (two-sided):                  0.00   Kurtosis:
6.23
==========================================================================
=======

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

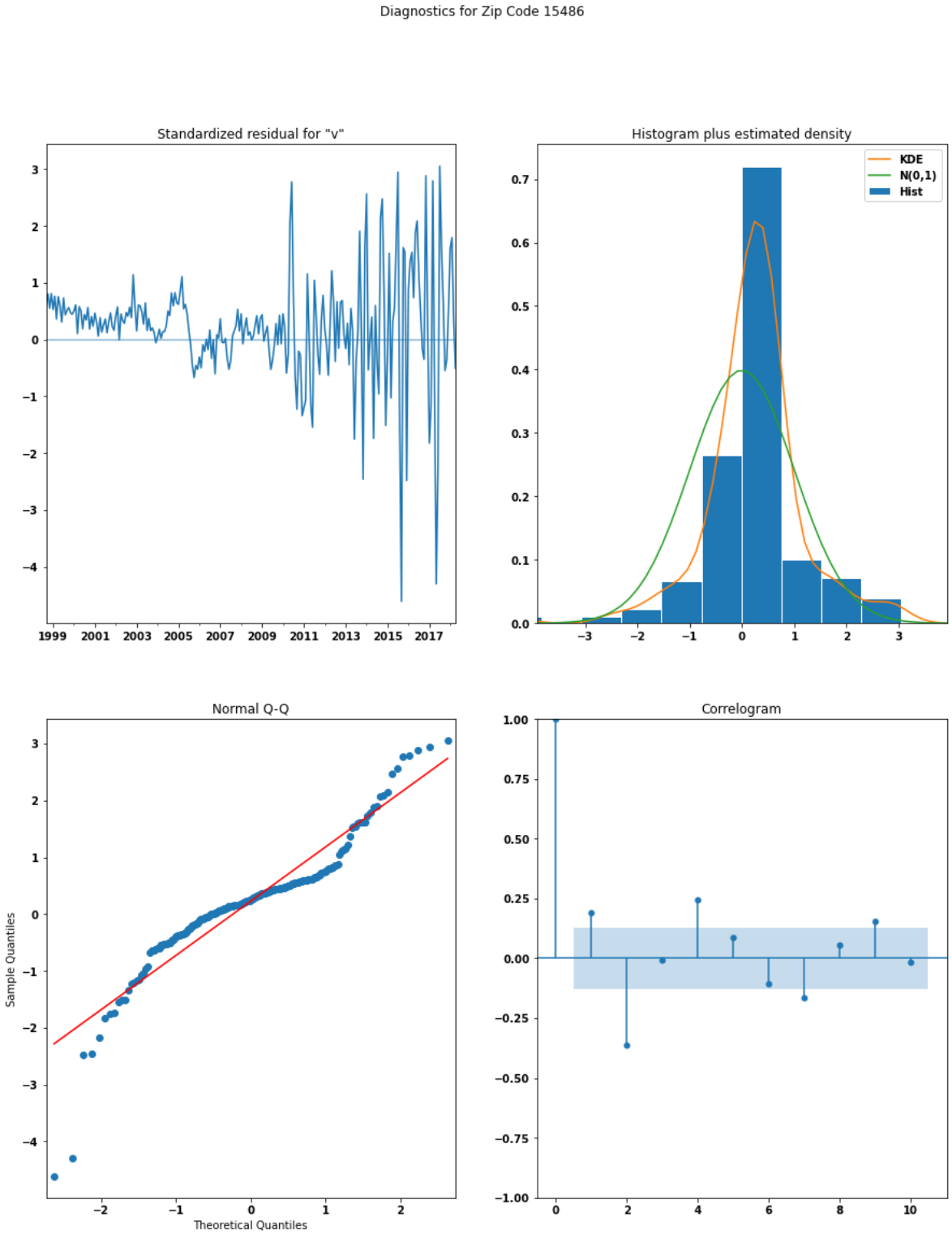

Diagnostics for Zip Code 48894
Standardized residual for "v"
Histogram plus estimated density
KDE
N(0,1)
Hist
Normal Q-Q
Sample Quantiles
Theoretical Quantiles
Correlogram

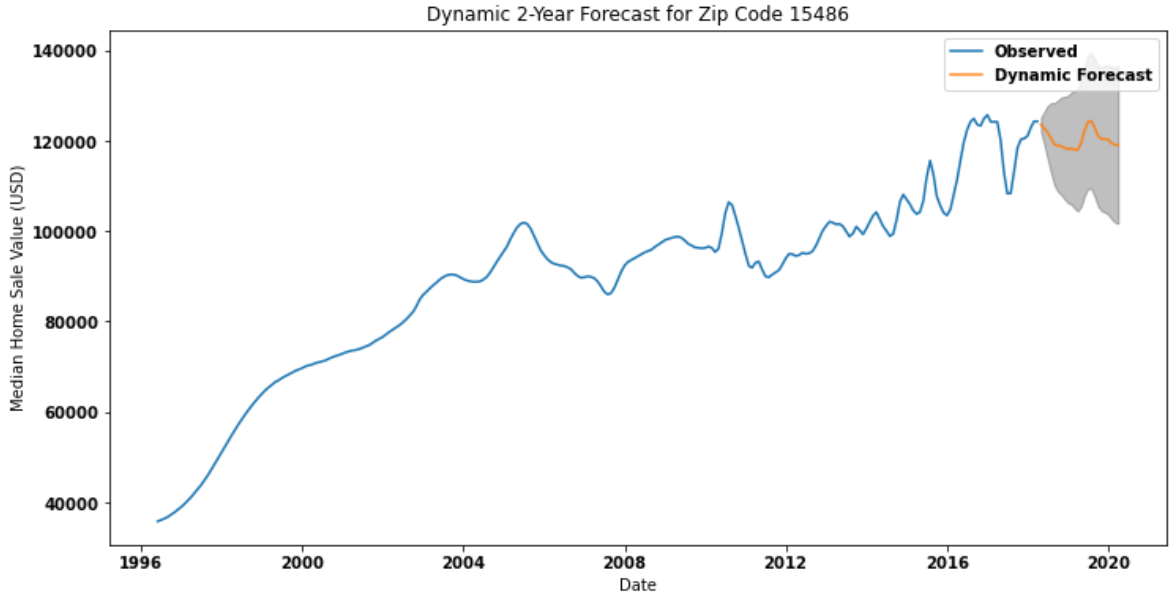


The RMSE of forecasts for Zip Code 48894 is 1571.15

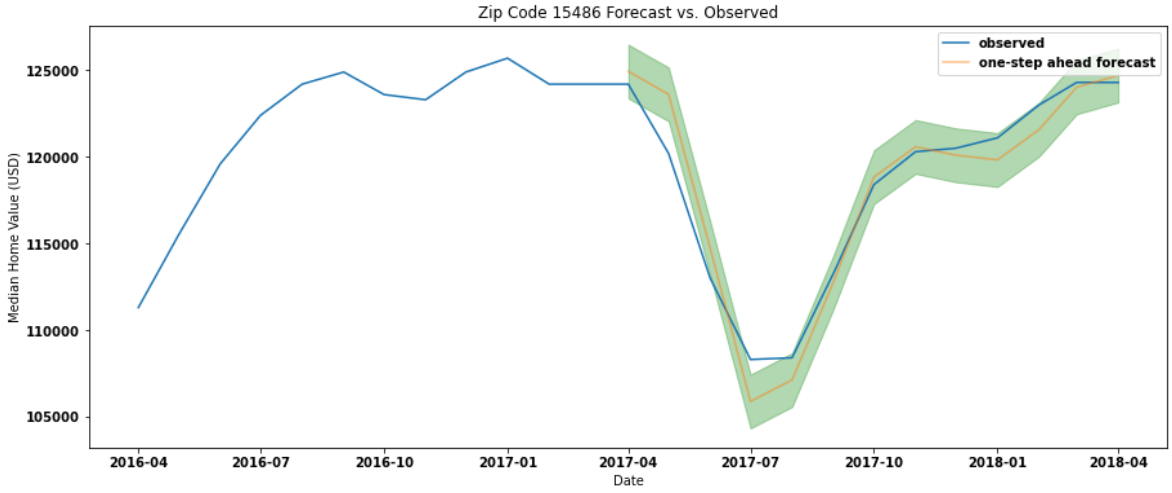


```
Summary for Zip Code 15486:
                                     SARIMAX Results
==========================================================================
==============
Dep. Variable:                               value   No. Observations:
263
Model:             SARIMAX(2, 1, 0)x(2, 0, 0, 12)   Log Likelihood
-1905.836
Date:                            Wed, 30 Aug 2023   AIC
3821.672
Time:                                    21:23:19   BIC
3838.991
Sample:                                06-01-1996   HQIC
3828.653
                                     - 04-01-2018
Covariance Type:                               opg
==========================================================================
==
                 coef    std err          z      P>|z|      [0.025      0.97
5]
--------------------------------------------------------------------------
--
ar.L1          1.1722      0.041     28.883      0.000       1.093       1.2
52
ar.L2         -0.5762      0.042    -13.639      0.000      -0.659      -0.4
93
ar.S.L12      -0.1862      0.056     -3.340      0.001      -0.295      -0.0
77
ar.S.L24      -0.3636      0.061     -5.970      0.000      -0.483      -0.2
44
sigma2      6.292e+05   4.01e+04     15.673      0.000    5.51e+05    7.08e+
05
==========================================================================
=======
Ljung-Box (L1) (Q):                   8.59   Jarque-Bera (JB):
301.29
Prob(Q):                              0.00   Prob(JB):
0.00
Heteroskedasticity (H):               9.25   Skew:
-0.74
Prob(H) (two-sided):                  0.00   Kurtosis:
8.34
==========================================================================
=======

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


Diagnostics for Zip Code 15486
Standardized residual for "v"
Histogram plus estimated density
KDE
N(0,1)
Hist
Normal Q-Q
Sample Quantiles
Theoretical Quantiles
Correlogram




```
The RMSE of forecasts for Zip Code 15486 is 1447.46
```



These graphs show the autoregressive terms, moving average terms, coefficients for the AR and MA terms and their statistical significance given their z-scores and p-values relative to 0.05. They also highlight the Ljung-Box test (Q statistic), the Jarque-Bera test, and the Heteroskedasticity test

Getting the top 5 best Zipcodes

- Here we calculate the projections for each zip code.
  - We will treat the last data point of the time series as our Current Value.
  - Then the Projected Value will be the summation of the current value and the forecasted value.
  - Using the Current Value and the Projected Value, we will calculate the ROI for each zipcode.
  - We then sort the zipcodes by ROIs, and finally determine the top 5 best zipcodes.

In [41]:
```
# A dictionary of the best parameters
best_params_by_zipcode = {
    49309: ((4, 2, 1), (1, 0, 1, 12)),
    40107: ((3, 1 ,3), (1, 0, 1, 12)),
    48822: ((1, 2, 2), (1, 0, 1, 12)),
    49265: ((1, 2, 3), (0, 0, 2, 12)),
    49425: ((3, 2, 3), (0, 0, 2, 12)),
    29645: ((1, 2, 3), (1, 0, 1, 12)),
    66206: ((2, 1, 1), (0, 0, 0, 12)),
    48835: ((0, 2, 3), (2, 0, 1, 12)),
    48894: ((2, 2, 1), (0, 0, 1, 12)),
    15486: ((2, 1, 0), (2, 0, 0, 12))
}
```

a) Forecasting

In [42]:
```
# Define a function for calculating projections

def calculate_projections(zipcode_df, zipcodes, years):

    # Define steps for forecasting
    steps = years * 12

    # Create list to append projection metrics
    projections = []

    # Iterate through each zip code
    for zipcode in zipcodes:

        # Get the specific dataframe for the current zipcode
        zip_ts = zipcode_df[zipcode_df['Zipcode'] == zipcode]

        if not zip_ts.empty:
            # Get the last value of the time series
            current_value = zip_ts['value'].iloc[-1]

            # Plug the optimal parameter values into a SARIMAX model
            order, seasonal_order = best_params_by_zipcode.get(zipcode, ((1, (

            SARIMAX = sm.tsa.statespace.SARIMAX(zip_ts['differenced_ret'],
                                                order=order,
                                                seasonal_order=seasonal_order
                                                enforce_stationarity=False,
                                                enforce_invertibility=False)

            # Fit the model
            output = SARIMAX.fit()

            # Get forecast and confidence interval for steps ahead in future
            forecast_steps = steps
            forecast = output.get_forecast(steps=forecast_steps)
            forecast_mean = forecast.predicted_mean
            conf_int = forecast.conf_int()

            # Calculate ROI based on the forecasted mean
            projected_value = current_value + forecast_mean.sum()
            roi = ((projected_value / current_value)- 1) * 100

            # Create dictionary to store projections
            projected_data = {
                'Zip Code': zipcode,
                'Investment Value ($)': current_value.round(2),
                'Predicted Sale Price ($)': projected_value.round(2),
                'Predicted ROI (%)': roi.round(2)
            }

            # Append to projections list
            projections.append(projected_data)

    # Return projections as a DataFrame
```

```
    return pd.DataFrame(projections)
```

2-year forecast

In [43]:
```
# Call the function
zipcodes_to_project = [49309, 40107, 48822, 49265, 49425, 29645, 66206, 48835
years_to_forecast = 2

# List to store the projected data for all zipcodes
all_projected_data = []

# Iterate over the dictionary of zipcode dataframes and call the function
for zipcode, zipcode_df in zipcode_dataframes.items():
    if zipcode in order_by_zipcode:
        order = order_by_zipcode[zipcode]
        projected_data = calculate_projections(zipcode_df, zipcodes_to_projec
        all_projected_data.append(projected_data)

# Concatenate all projected data into a single DataFrame
final_projected_data = pd.concat(all_projected_data, ignore_index=True)

final_projected_data.sort_values(by='Predicted ROI (%)', ascending=False)
```

Out[43]:

	Zip Code	Investment Value ($)	Predicted Sale Price ($)	Predicted ROI (%)
2	48822	184700.0	197993.38	7.20
0	49309	49200.0	51780.13	5.24
5	29645	116500.0	118174.60	1.44
7	48835	138600.0	138619.15	0.01
3	49265	159600.0	158825.19	-0.49
8	48894	161000.0	160023.85	-0.61
4	49425	97200.0	96589.34	-0.63
6	66206	455700.0	452083.46	-0.79
1	40107	148300.0	146367.82	-1.30
9	15486	124300.0	119227.76	-4.08

In [44]:
```
final_projected_data.sort_values(by='Predicted ROI (%)', ascending=False).hea
```

Out[44]:

	Zip Code	Investment Value ($)	Predicted Sale Price ($)	Predicted ROI (%)
2	48822	184700.0	197993.38	7.20
0	49309	49200.0	51780.13	5.24
5	29645	116500.0	118174.60	1.44
7	48835	138600.0	138619.15	0.01
3	49265	159600.0	158825.19	-0.49

Conclusion / Interpreting Results

From our analysis, the best Zipcodes to invest in based on a combination of predicted sale price and positive predicted ROI are:

- **Zip Code 48822:**
  - Investment Value: $184,700
  - Predicted Sale Price after 2 years: $197,993.26
  - Predicted ROI: 7.20%
- This Zipcode stands out with a promising ROI of 7.20%, indicating a potential return on investment. With an initial investment of $184,700, the projected sale price after 2 years suggests favorable growth potential. This could be attributed to positive market conditions and demand factors.
- **Zip Code 49309:**
  - Investment Value: $49,200
  - Predicted Sale Price after 2 years: $51,777.49
  - Predicted ROI: 5.24%
- Zip code 49309 offers a solid investment option with a predicted ROI of 5.24%. Despite its lower initial investment value, the projected sale price after 2 years indicates notable growth potential. This suggests a favorable investment opportunity considering the moderate risk
- **Zip Code 29645:**
  - Investment Value: $116,500
  - Predicted Sale Price after 2 years: $118,177.38
  - Predicted ROI: 1.44%
- Zip code 29645 presents a stable investment prospect with a predicted ROI of 1.44%. While the ROI is relatively modest, the expected increase in sale price after 2 years could provide steady returns, suitable for conservative investors seeking stable growth.
- **Zip Code 48835:**
  - Investment Value: $138,600
  - Predicted Sale Price after 2 years: $138,618.56
  - Predicted ROI: 0.01%
- Zip code 48835 offers a nearly negligible ROI of 0.01%. While the projected growth is minimal, the initial investment value of $138,600 remains relatively stable. This might attract risk-averse investors who prioritize stability over high returns.

- **Zip Code 49265:**
  - Investment Value: $159,600
  - Predicted Sale Price after 2 years: $158,830.09
  - Predicted ROI: -0.48%
- In contrast to the top 4 zipcodes, zip code 49265 displays a negative ROI of -0.48%, indicating a potential loss based on the projected sale price after 2 years. Investors might want to exercise caution with this zip code, as it suggests a declining property value trend.

Recommendations

- Zipcodes 48822, 49309, 29645, and 48835 stand out as the top investment choices based on their projected ROIs and anticipated property value growth over the next 2 years. Considering these insights, we strongly advise Boma Yangu to consider these options for their investment prospects. These selections offer a compelling blend of attractive sale prices and favorable returns on investment, all while accounting for Boma Yangu's risk threshold.
- For zip code 49265, we strongly urge Boma Yangu to approach it with caution, as it signals the possibility of incurring a loss.

Next Steps

1. Collect more data with added exogenous variables like economic indicators, interest rates and demographic data that could potentially influence the housing price values, then model them in a SARIMAX model. This could lead to better predictions and better models overall.

2. Build more complex and advanced models such as RNNs, e.g. LSTMs which make use of neural networks. These algorithms are more powerful than traditional time series. They are able to capture any complex patterns in our time series, hence leading to better forecasts/predictions.