0.1 Project #4:

- Student name: Milena Afeworki
- Student pace: full time
- Scheduled project review date/time: 08/05/2021 @ 10:15 PT
- Instructor name: Abhineet Kulkarni
- Blog post URL:

1 Time series Prediction on California Home sales

With 54 of the Fortune 500 companies headquartered in California, like Google, Apple, Disney, Oracle and Intel among others, California is positioned for continued job growth. High employment rates draw renters and buyers and, for investors, enhance the likelihood of consistent cash flow. Though not the lowest in the country, California has favorable property tax rates, which will help control investors' expenses and improve cash flow. The combination of job growth and a world-renowned lifestyle and culture supports home values. People buy where they want to live, and millions of people want to live in California. All of the demand mentioned above also leads to increasing home values.

1.1 Business Problem

If a real estate company is looking to flip homes, what are the top 5 zip codes to invest in? How will we scale our data? What defines best?

For this dataset, we will only include data from CA. Because of the housing market crash, any modelling that uses only recent years may be misleading. We will use every value from 1996 to 2018 so we can have the most accurate picture of home values in CA through the years.

1.2 Data understanding

Zillow provides their users the opportunity to use their platform to access specific datasets for research purposes. The dataset that we will be using contains the median home sales prices throughout all states sorted by their zip codes. With this dataset we can extract a lot of insight through out all states with the potential to understand markets and develop investment strategies. This platform allows the public to do independent research in any market in the US.

This dataset contains 14723 rows and 272 columns.

2 Load the Data/Filtering for Chosen Zipcodes

In [1]:

```
#Import libraries
import numpy as np
import folium
import pandas as pd
import matplotlib as mpl
from matplotlib import pyplot as plt
from scipy import stats
from random import gauss as gs
import math
import datetime
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
from math import sqrt
import statsmodels.api as sm
from statsmodels.tsa.arima_model import ARMA
from statsmodels.tsa.stattools import adfuller, acf, pacf
from statsmodels.graphics.tsaplots import plot_acf,plot_pacf
from pandas.plotting import autocorrelation_plot, lag_plot
from statsmodels.tsa.seasonal import seasonal_decompose
from statsmodels.tsa.statespace.sarimax import SARIMAX
from pandas.plotting import register_matplotlib_converters
register_matplotlib_converters()
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
import warnings
warnings.filterwarnings('ignore')
import pmdarima as pm
from pmdarima import auto_arima
from matplotlib.pylab import rcParams
%matplotlib inline
plt.style.use('ggplot')
```

executed in 2.51s, finished 18:57:36 2021-08-07

In [2]:

```
df = pd.read_csv('zillow_data.csv')
df.head()
```

executed in 382ms, finished 18:57:36 2021-08-07

Out[2]:

	RegionID	RegionName	City	State	Metro	CountyName	SizeRank	1996-04	1996-05	1996-06	...	2017-07	2017-08	2
0	84654	60657	Chicago	IL	Chicago	Cook	1	334200.0	335400.0	336500.0	...	1005500	1007500	1
1	90668	75070	McKinney	TX	Dallas-Fort Worth	Collin	2	235700.0	236900.0	236700.0	...	308000	310000	
2	91982	77494	Katy	TX	Houston	Harris	3	210400.0	212200.0	212200.0	...	321000	320600	
3	84616	60614	Chicago	IL	Chicago	Cook	4	498100.0	500900.0	503100.0	...	1289800	1287700	1.
4	93144	79936	El Paso	TX	El Paso	El Paso	5	77300.0	77300.0	77300.0	...	119100	119400	

5 rows × 272 columns

In [3]:

```
df.shape
```

executed in 10ms, finished 18:57:36 2021-08-07

Out[3]:

```
(14723, 272)
```

3 Data Preprocessing

Let's replace the column 'RegionName' by 'ZipCode' and then select only the data for California and drop the rest.

In [4]:

```
#Rename RegionName
df.rename({'RegionName': 'ZipCode'}, axis='columns', inplace=True)
```

executed in 14ms, finished 18:57:36 2021-08-07

In [5]:

```
#Delete all but CA zipcodes
df_ca = df.loc[df['State']== 'CA'].reset_index()
df_ca.drop(['index', 'RegionID', 'SizeRank'], axis=1, inplace=True)
print('Total Zipcodes in DataFrame:', len(df_ca))
```

executed in 30ms, finished 18:57:36 2021-08-07

```
Total Zipcodes in DataFrame: 1224
```

In [6]:

```
#Check for zeros
df_ca.describe()
```

executed in 352ms, finished 18:57:37 2021-08-07

Out[6]:

	ZipCode	1996-04	1996-05	1996-06	1996-07	1996-08	1996-09	1996-10	1996
count	1224.000000	1.188000e+03	1.188000e+03	1.188000e+03	1.188000e+03	1.188000e+03	1.188000e+03	1.188000e+03	1.188000e-
mean	93308.559641	2.002210e+05	2.001248e+05	2.000359e+05	1.999656e+05	1.999588e+05	2.000376e+05	2.002770e+05	2.006430e-
std	1800.792540	1.260446e+05	1.265337e+05	1.270587e+05	1.276189e+05	1.282251e+05	1.289055e+05	1.297123e+05	1.306263e-
min	90001.000000	4.440000e+04	4.390000e+04	4.350000e+04	4.290000e+04	4.240000e+04	4.180000e+04	4.120000e+04	4.070000e-
25%	92013.250000	1.294750e+05	1.286750e+05	1.283750e+05	1.280000e+05	1.277000e+05	1.275750e+05	1.272000e+05	1.268750e-
50%	93302.500000	1.635000e+05	1.628500e+05	1.625000e+05	1.622000e+05	1.623500e+05	1.622000e+05	1.625000e+05	1.624000e-
75%	95035.500000	2.322500e+05	2.322250e+05	2.325000e+05	2.326000e+05	2.330250e+05	2.336250e+05	2.339750e+05	2.350500e-
max	96161.000000	1.179200e+06	1.184300e+06	1.189700e+06	1.195400e+06	1.201200e+06	1.207300e+06	1.214100e+06	1.221200e-

8 rows × 266 columns

4 Reshape from Wide to Long Format

Our next step would be to change the format of the data frame from wide format to long format and index by the'Date' column

In [7]:
```
def melt_data(df):
    melted = pd.melt(df, id_vars=['ZipCode', 'City', 'State', 'Metro', 'CountyName'],
                     var_name='Date')
    melted['Date'] = pd.to_datetime(melted['Date'], infer_datetime_format=True)
    melted = melted.dropna(subset=['value'])
    return melted
```
executed in 11ms, finished 18:57:37 2021-08-07

In [8]:
```
melted_df = melt_data(df_ca)
```
executed in 116ms, finished 18:57:37 2021-08-07

We will also want to make sure we change zip code into a string so it is not confused for an integer.

In [9]:
```
#Change Zipcode dtype to 'str'
melted_df['ZipCode'] = melted_df['ZipCode'].astype(str)

# Make sure the data type of the 'Date' column is datetime
melted_df['Date'] = pd.to_datetime(melted_df['Date'], format='%m/%y')

# Set the 'Date' column as index
melted_df.set_index('Date', inplace=True)
```
executed in 220ms, finished 18:57:37 2021-08-07

In [10]:
```
melted_df.index
```
executed in 22ms, finished 18:57:37 2021-08-07

Out[10]:
```
DatetimeIndex(['1996-04-01', '1996-04-01', '1996-04-01', '1996-04-01',
               '1996-04-01', '1996-04-01', '1996-04-01', '1996-04-01',
               '1996-04-01', '1996-04-01',
               ...
               '2018-04-01', '2018-04-01', '2018-04-01', '2018-04-01',
               '2018-04-01', '2018-04-01', '2018-04-01', '2018-04-01',
               '2018-04-01', '2018-04-01'],
              dtype='datetime64[ns]', name='Date', length=318744, freq=None)
```

In [11]:
```
melted_df.head()
```
executed in 20ms, finished 18:57:37 2021-08-07

Out[11]:

	ZipCode	City	State	Metro	CountyName	value
Date						
1996-04-01	94109	San Francisco	CA	San Francisco	San Francisco	766000.0
1996-04-01	90250	Hawthorne	CA	Los Angeles-Long Beach-Anaheim	Los Angeles	152500.0
1996-04-01	94565	Pittsburg	CA	San Francisco	Contra Costa	139200.0
1996-04-01	90046	Los Angeles	CA	Los Angeles-Long Beach-Anaheim	Los Angeles	340600.0
1996-04-01	94501	Alameda	CA	San Francisco	Alameda	222400.0

In [12]:
```
melted_df.tail()
```
executed in 21ms, finished 18:57:37 2021-08-07

Out[12]:

	ZipCode	City	State	Metro	CountyName	value
Date						
2018-04-01	93517	Bridgeport	CA	NaN	Mono	272500.0
2018-04-01	95728	Truckee	CA	Truckee	Nevada	496300.0
2018-04-01	95497	Annapolis	CA	Santa Rosa	Sonoma	848700.0
2018-04-01	92322	Crestline	CA	Riverside	San Bernardino	200100.0
2018-04-01	92341	Green Valley Lake	CA	Riverside	San Bernardino	183600.0

In [13]:
```
melted_df.shape
```
executed in 12ms, finished 18:57:37 2021-08-07

Out[13]: (318744, 6)

5 EDA and Visualization

On this step we will be creating visualizations to get a better idea of what we are working with and also to understand the trends of the values in our data.

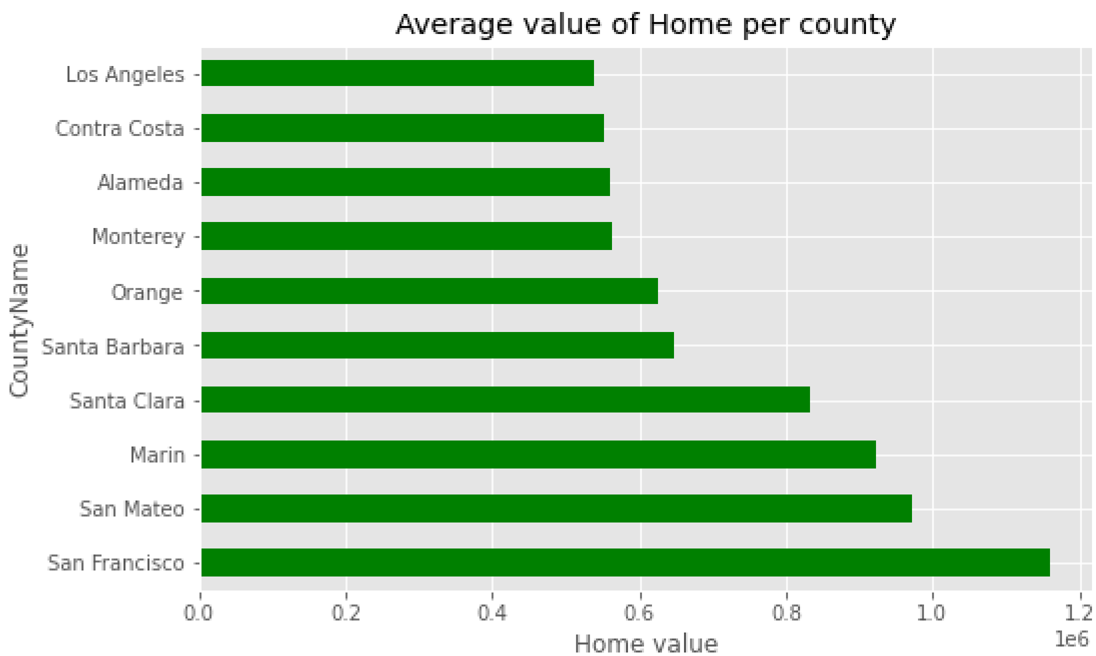
In [14]:
```
#check for nulls
melted_df.isna().sum()
```
executed in 77ms, finished 18:57:37 2021-08-07

Out[14]:
```
ZipCode          0
City             0
State            0
Metro        10602
CountyName       0
value            0
dtype: int64
```

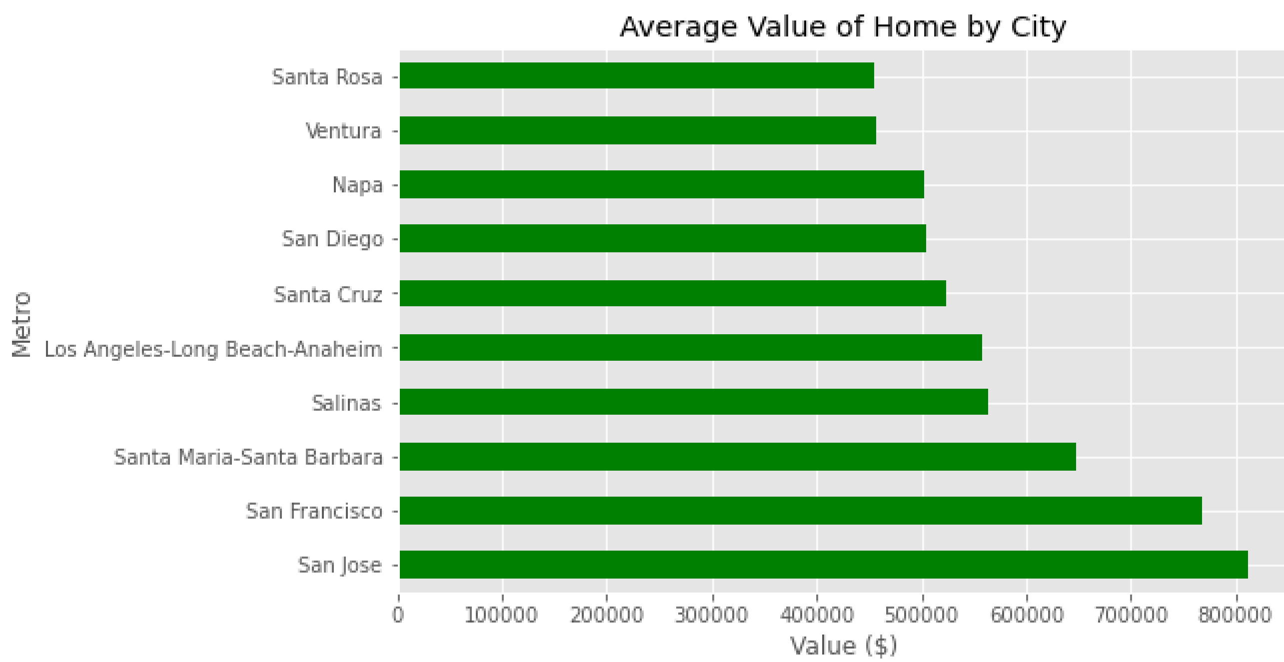
In [15]:
```
metro = melted_df.groupby('Metro')
metro = metro.value.mean()
metro = metro.sort_values(ascending=False).head(10)
```
executed in 32ms, finished 18:57:37 2021-08-07

In [16]:
```
county = melted_df.groupby('CountyName')
county = county.value.mean()
county = county.sort_values(ascending=False).head(10)
```
executed in 32ms, finished 18:57:37 2021-08-07

In [17]:
```
fig =plt.figure(figsize=(8,5))
county.plot.barh(color='green')
plt.title('Average value of Home per county')
plt.xlabel('Home value')
plt.show()
```
executed in 270ms, finished 18:57:37 2021-08-07



In [18]:
```
fig = plt.figure(figsize=(8,5))
metro.plot.barh(color='green')
plt.title('Average Value of Home by City')
plt.xlabel('Value ($)')
plt.show()
```
executed in 234ms, finished 18:57:38 2021-08-07



Here we get a good idea of the average home value per County name.

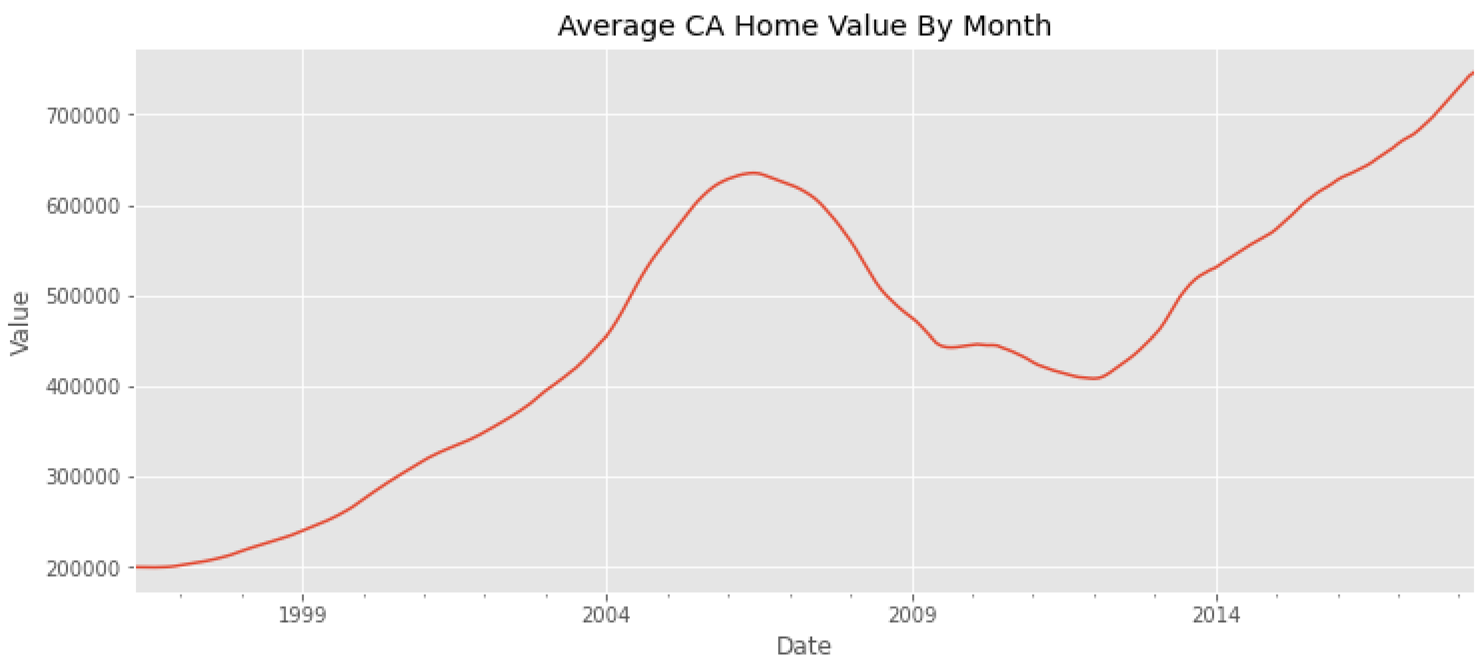
In [19]:
```
print('Average CA home value' ,round(melted_df['value'].mean()))
```
executed in 14ms, finished 18:57:38 2021-08-07

```
Average CA home value 457429
```

In [20]:

```
#data resampled by month
monthly_data = melted_df['value'].resample('MS').mean()
monthly_data = monthly_data.fillna(monthly_data.bfill())
monthly_data.plot(figsize=(12,5))
plt.title('Average CA Home Value By Month')
plt.ylabel('Value')
plt.show()
print(monthly_data.head())
```

executed in 250ms, finished 18:57:38 2021-08-07



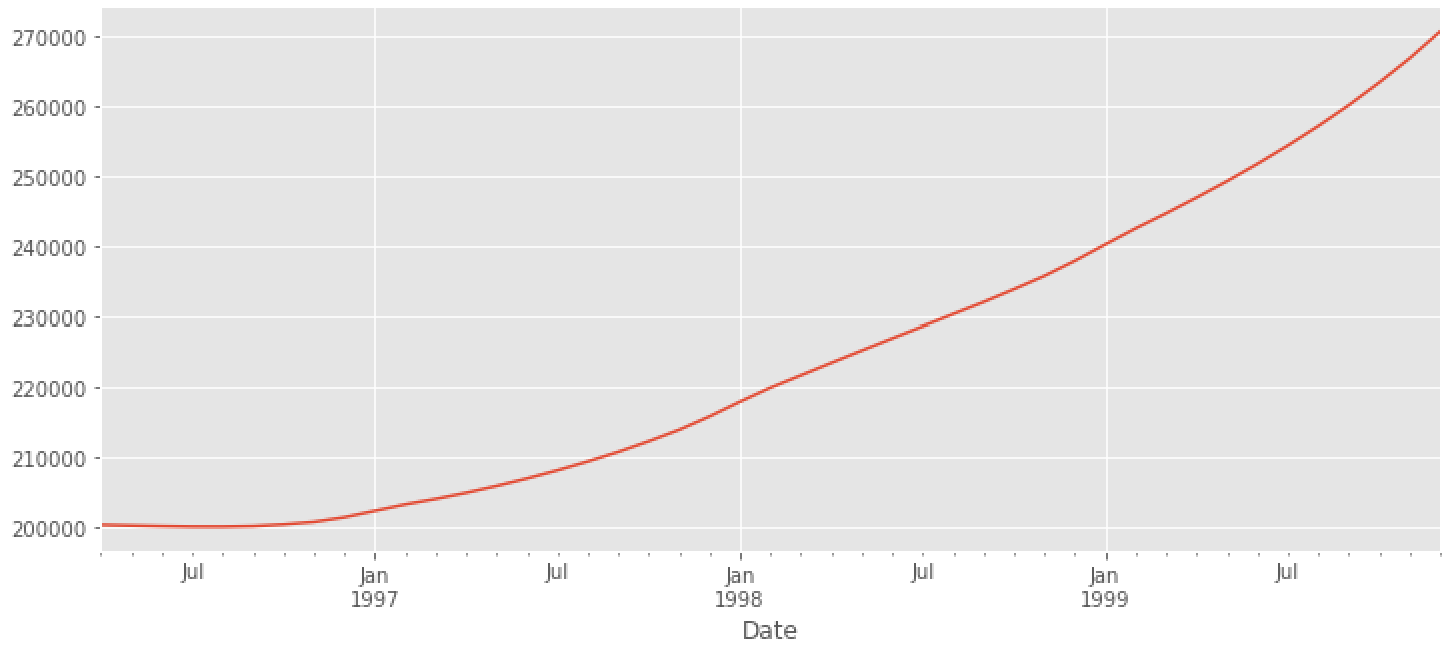

```
Date
1996-04-01    200220.959596
1996-05-01    200124.831650
1996-06-01    200035.942761
1996-07-01    199965.572391
1996-08-01    199958.754209
Freq: MS, Name: value, dtype: float64
```

In [21]:

```
monthly_data['1996':'1999'].plot(figsize=(12,5))
```

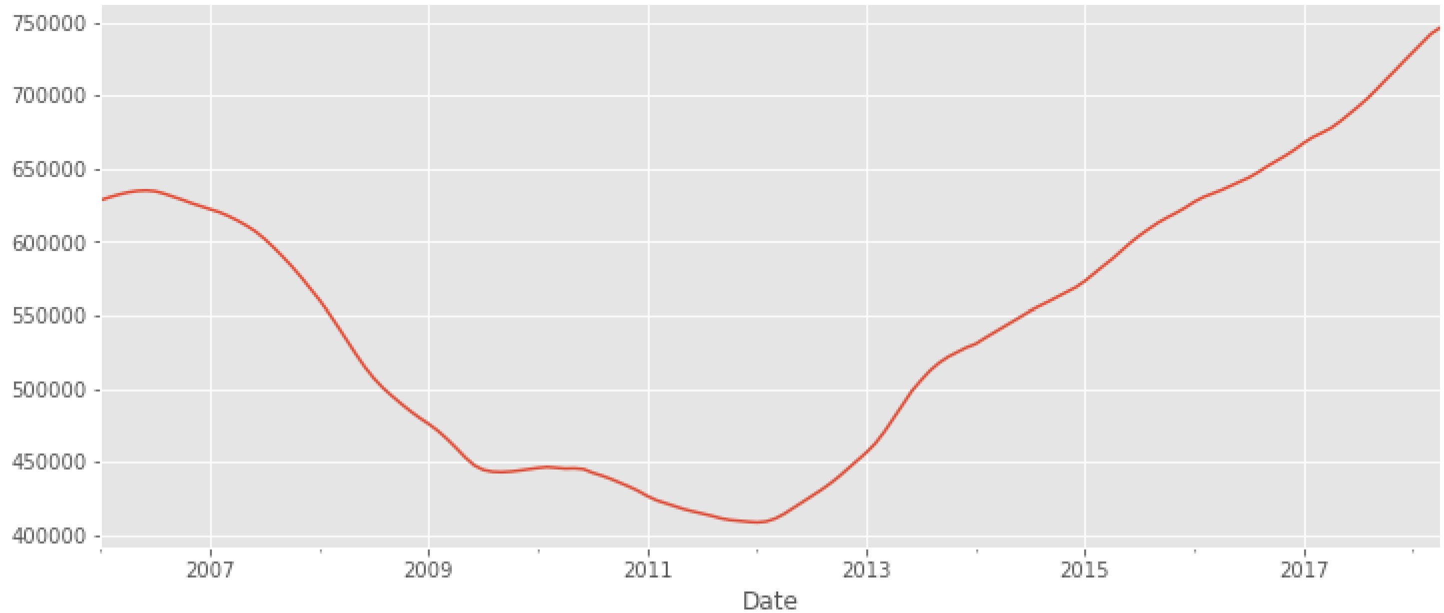
executed in 315ms, finished 18:57:38 2021-08-07

Out[21]: <AxesSubplot:xlabel='Date'>



In [22]:
```
monthly_data['2006':].plot(figsize=(12,5))
```
executed in 180ms, finished 18:57:38 2021-08-07

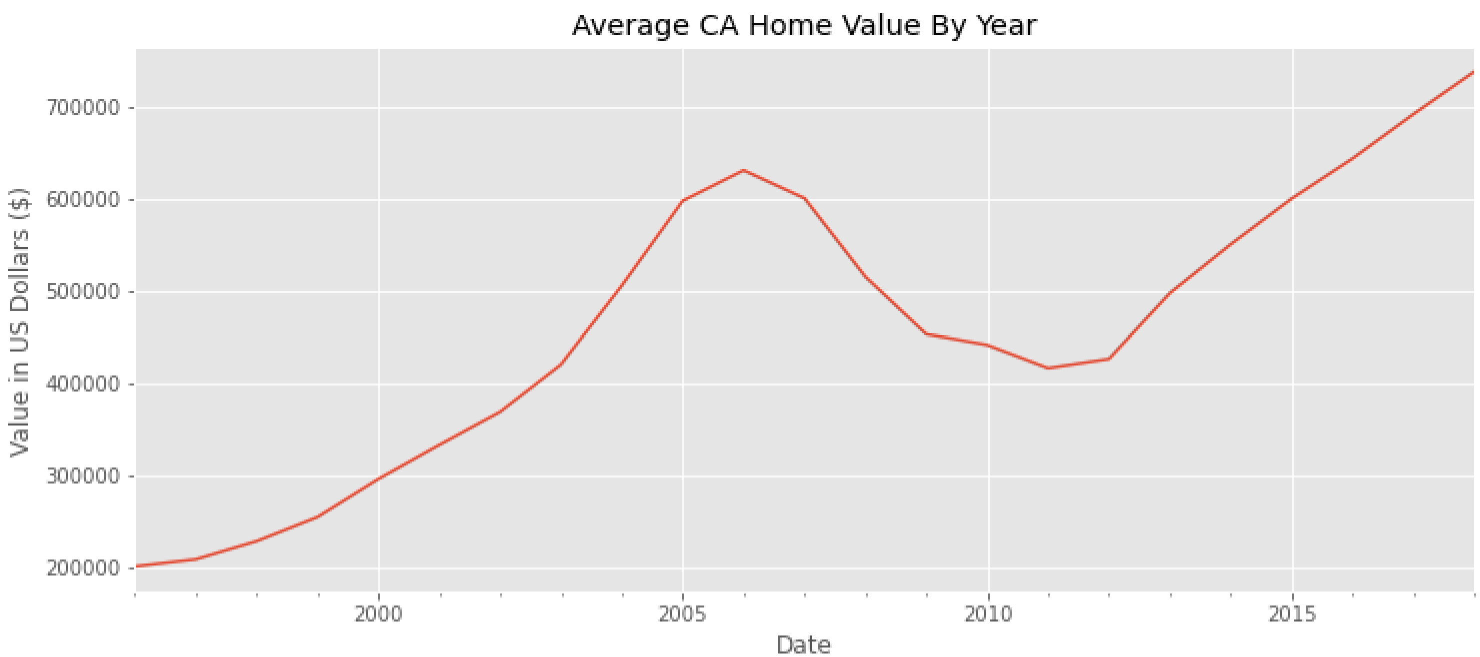
Out[22]: <AxesSubplot:xlabel='Date'>



In [23]:

```
yearly_data = melted_df['value'].resample('A').mean()
yearly_data.plot.line(figsize=(12,5))
plt.title('Average CA Home Value By Year')
plt.ylabel('Value in US Dollars ($)')
plt.show()
print(yearly_data.head())
```

executed in 205ms, finished 18:57:39 2021-08-07



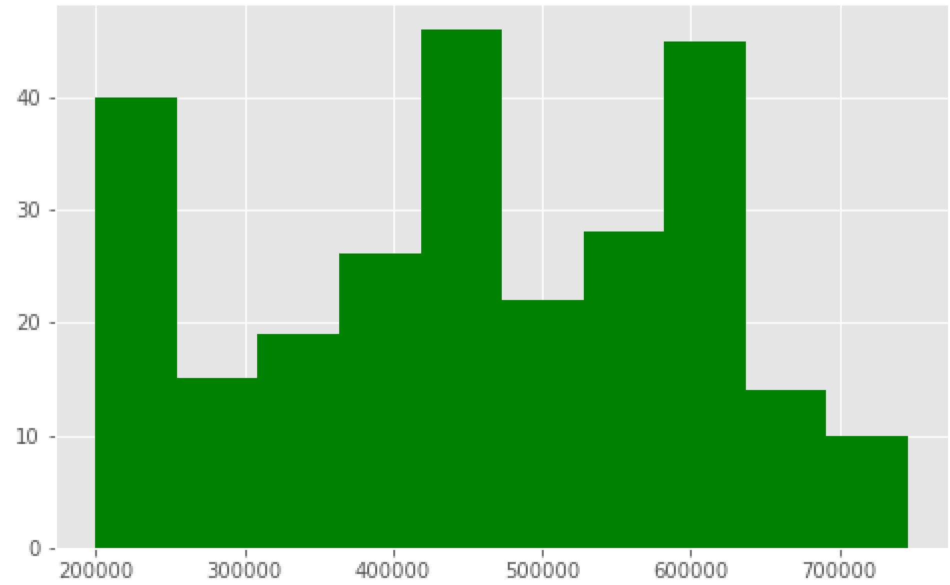

```
Date
1996-12-31    200285.933408
1997-12-31    208054.924242
1998-12-31    227790.333895
1999-12-31    254072.804433
2000-12-31    295432.870370
Freq: A-DEC, Name: value, dtype: float64
```

In [24]:

```
fig = plt.figure( figsize=(8,5))
monthly_data.hist(color='green')
```

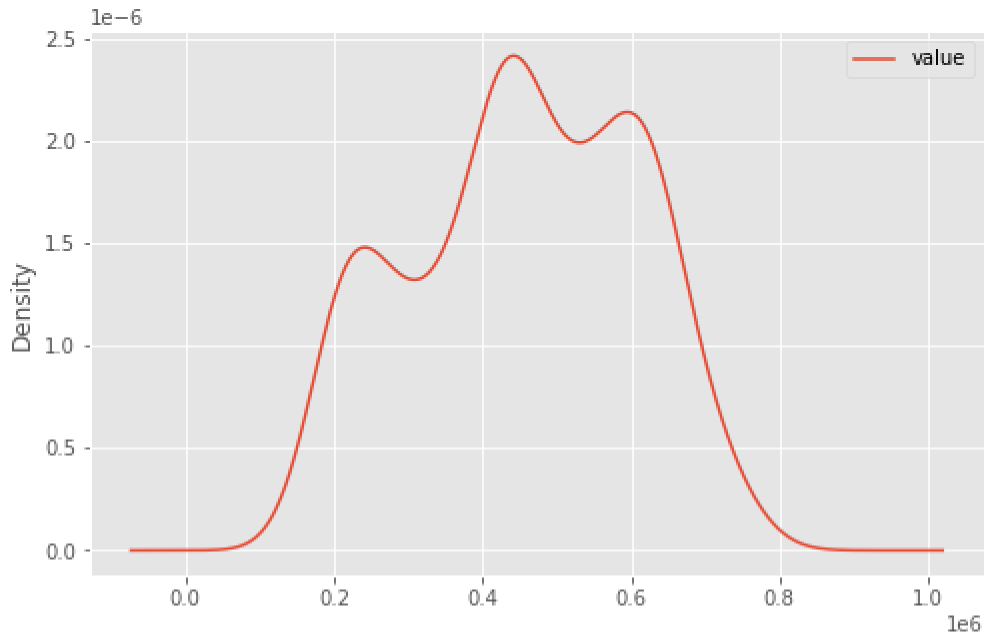
executed in 140ms, finished 18:57:39 2021-08-07

Out[24]: <AxesSubplot:>



In [25]:
```
fig = plt.figure( figsize=(8,5))
monthly_data.plot(kind='kde')
plt.legend()
```
executed in 239ms, finished 18:57:39 2021-08-07

Out[25]: <matplotlib.legend.Legend at 0x293aa09eca0>



The monthly and yearly home values seem to be similar. There doesn't seem to be any seasonality now but we will look into data decomposition and explore it further, but we do see a generaly upward trend with a dip downward between the years 2007-2012. Next, we will cut down on variation to ensure we get the true most valuable zip codes.

In [26]:

```
#taking into account the last 5years
df_ca.iloc[:,-60:].head()
```

executed in 26ms, finished 18:57:39 2021-08-07

Out[26]:

	2013-05	2013-06	2013-07	2013-08	2013-09	2013-10	2013-11	2013-12	2014-01	2014-02	...	2017-07	2017-
0	3024300.0	3084000.0	3128400.0	3149900.0	3168700.0	3181800.0	3177400.0	3171800.0	3181200.0	3197700.0	...	3767700	37639
1	394600.0	401100.0	406500.0	411200.0	414400.0	415900.0	416600.0	417300.0	419000.0	421900.0	...	579300	5857
2	206100.0	210000.0	215500.0	222300.0	228100.0	233100.0	239400.0	246200.0	251800.0	255800.0	...	394900	3984
3	1185600.0	1200800.0	1214100.0	1228900.0	1242500.0	1253600.0	1260200.0	1263100.0	1265900.0	1273000.0	...	1839800	18611
4	669900.0	688000.0	698800.0	705000.0	709500.0	713500.0	713600.0	713000.0	715500.0	719700.0	...	965100	9750

5 rows × 60 columns

In [27]:

```
df_ca['yr_avg']=df_ca.iloc[:,-60:].mean(skipna=True, axis=1)

#Get zipcodes with an average value 2 decile above the median and 2 deciles below.
print(df_ca['yr_avg'].describe(),'\n')

#Calculate the 70% cutoff value (2 decile above).
q_70 = df_ca['yr_avg'].quantile(q=0.70)
print(f'Average Value 70% cutoff value: {round(q_70,2)}')

#Calculate the 30% cutoff value (2 deciles below).
q_30 = df_ca['yr_avg'].quantile(q=0.30)
print(f'Average Value 30% cutoff value: {round(q_30,2)}')

#Get data frame with selected zipcodes.
df_avg = df_ca[(df_ca['yr_avg']<q_70) & (df_ca['yr_avg']>q_30)]
print(f'Amount of zipcodes: {len(df_avg)}')
```

executed in 31ms, finished 18:57:39 2021-08-07

```
count    1.224000e+03
mean     6.146968e+05
std      5.553401e+05
min      5.477833e+04
25%      2.826767e+05
50%      4.664775e+05
75%      7.432292e+05
max      5.319428e+06
Name: yr_avg, dtype: float64

Average Value 70% cutoff value: 654150.17
Average Value 30% cutoff value: 313131.83
Amount of zipcodes: 490
```

In [28]:
```
df_avg.head()
```
executed in 29ms, finished 18:57:39 2021-08-07

Out[28]:

	ZipCode	City	State	Metro	CountyName	1996-04	1996-05	1996-06	1996-07	1996-08	...	2017-08	2017-09	20
1	90250	Hawthorne	CA	Los Angeles-Long Beach-Anaheim	Los Angeles	152500.0	152600.0	152600.0	152600.0	152600.0	...	585700	590900	5947
2	94565	Pittsburg	CA	San Francisco	Contra Costa	139200.0	138300.0	137500.0	136600.0	135600.0	...	398400	401600	4054
7	90044	Los Angeles	CA	Los Angeles-Long Beach-Anaheim	Los Angeles	119500.0	119500.0	119400.0	119300.0	119200.0	...	386900	390600	3943
8	90805	Long Beach	CA	Los Angeles-Long Beach-Anaheim	Los Angeles	128300.0	128100.0	127800.0	127500.0	127100.0	...	430600	434800	4393
9	95630	Folsom	CA	Sacramento	Sacramento	190000.0	189300.0	188500.0	187800.0	187300.0	...	529700	529500	5310

5 rows × 271 columns

In finance, the coefficient of variation allows investors to determine how much volatility, or risk, is assumed in comparison to the amount of return expected from investments. Ideally, if the coefficient of variation formula should result in a lower ratio of the standard deviation to mean return, then the better the risk-return trade-off. Therefore, in these next steps we are going to filter the data some more by calculating the CV value and only selecting values with in the company's risk factor (assume 60 percentile).

In [29]:

```
#Calculate historical return on investment
df_avg['ROI'] = (df_avg['yr_avg']/df_avg['1996-04'])-1

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

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

#Calculate coefficient of variation
df_avg['CV'] = df_avg['std']/df_avg['mean']

#Show calculated values
df_avg[['ZipCode','std','mean','ROI','CV','CountyName']].head()
```

executed in 30ms, finished 18:57:39 2021-08-07

Out[29]:

	ZipCode	std	mean	ROI	CV	CountyName
1	90250	138601.533036	365464.150943	2.262240	0.379248	Los Angeles
2	94565	104006.852657	263874.339623	1.334291	0.394153	Contra Costa
7	90044	97293.744047	243829.811321	1.686262	0.399023	Los Angeles
8	90805	108971.123756	281955.471698	1.918213	0.386483	Los Angeles
9	95630	106331.603220	373722.264151	1.513737	0.284520	Sacramento

In [30]:
```
#find out the top 10 couties with highest ROI
grp_county = df_avg.groupby('CountyName', group_keys=False).sum()['ROI']
grp_county.sort_values(ascending=False)[:10]

# sorted(round(grouped_county,2), reverse=True)[:10]
```
executed in 13ms, finished 18:57:39 2021-08-07

Out[30]:
```
CountyName
Los Angeles       313.935844
San Diego         113.669953
Orange             70.569764
Riverside          41.097313
Ventura            32.937638
Sacramento         32.498056
San Bernardino     31.662873
Alameda            29.279048
Placer             27.415028
Sonoma             26.812512
Name: ROI, dtype: float64
```

In [31]:
```
#top 10 counties with highest ROI before considering risk factor CV
grp_county.sort_values(ascending=False)[:10].keys()
```
executed in 13ms, finished 18:57:39 2021-08-07

Out[31]:
```
Index(['Los Angeles', 'San Diego', 'Orange', 'Riverside', 'Ventura',
       'Sacramento', 'San Bernardino', 'Alameda', 'Placer', 'Sonoma'],
      dtype='object', name='CountyName')
```

In [32]:
```
#Descriptive statistics of coefficients of variance.
print(df_avg.CV.describe())

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

#Get the 10 counties with highest ROIs within the firms risk profile.
df_top10 = df_avg[df_avg['CV']<upper_cv].sort_values('ROI', axis=0, ascending=False)

#find out the top 10 couties with highest ROI
grp_county = df_top10.groupby('CountyName').sum()['ROI']
grp_county.sort_values(ascending=False)[:10]
```

executed in 21ms, finished 18:57:39 2021-08-07

```
count    490.000000
mean       0.342695
std        0.045802
min        0.062004
25%        0.316948
50%        0.343743
75%        0.367369
max        0.496292
Name: CV, dtype: float64

CV upper limit: 0.352771200684699
```

Out[32]:
```
CountyName
San Diego          81.000270
Los Angeles        69.132190
Orange             29.551015
Riverside          28.391854
Placer             27.415028
Sonoma             26.812512
San Luis Obispo    24.860144
Sacramento         24.044008
Ventura            22.830799
Alameda            14.629467
Name: ROI, dtype: float64
```

Now for each county lets look into the zipcode with the highest ROI value and move onto the time series analysis

In [33]:

```
top10_county = list(grp_county.sort_values(ascending=False)[:10].index)
top10_county
```

executed in 13ms, finished 18:57:39 2021-08-07

Out[33]:

```
['San Diego',
 'Los Angeles',
 'Orange',
 'Riverside',
 'Placer',
 'Sonoma',
 'San Luis Obispo',
 'Sacramento',
 'Ventura',
 'Alameda']
```

In [34]:

```
df_top10.shape
```

executed in 12ms, finished 18:57:39 2021-08-07

Out[34]:

```
(294, 275)
```

In [35]:

```
df_top10 = df_top10.loc[df_top10['CountyName'].isin(top10_county)]
df_top10.shape
```

executed in 15ms, finished 18:57:39 2021-08-07

Out[35]:

```
(204, 275)
```

In [36]:

```
df_top10['CountyName'].value_counts()
```

executed in 12ms, finished 18:57:39 2021-08-07

Out[36]:

```
San Diego          44
Los Angeles        40
Riverside          21
Placer             18
Sacramento         17
Sonoma             16
Orange             14
Ventura            13
San Luis Obispo    13
Alameda             8
Name: CountyName, dtype: int64
```

In [37]:
```
df_top10.groupby('CountyName').max()['ROI']
```
executed in 25ms, finished 18:57:39 2021-08-07

Out[37]:
```
CountyName
Alameda            2.231758
Los Angeles        2.216251
Orange             2.530311
Placer             2.508959
Riverside          1.791631
Sacramento         2.292465
San Diego          2.675068
San Luis Obispo    2.605580
Sonoma             1.911364
Ventura            2.002716
Name: ROI, dtype: float64
```

In [38]:
```
df_top10.isna().sum()
```
executed in 13ms, finished 18:57:39 2021-08-07

Out[38]:
```
ZipCode       0
City          0
State         0
Metro         0
CountyName    0
             ..
yr_avg        0
ROI           3
std           0
mean          0
CV            0
Length: 275, dtype: int64
```

In [39]:

```
#Get city and state names for each zip code
ziplist = []
top_ROI = {}

for i in top10_county:
    City = df_top10[df_top10['CountyName']==i].City.values[0]
    Metro = df_top10[df_top10['CountyName']==i].Metro.values[0]
    Zipcode = df_top10[df_top10['CountyName']==i].ZipCode.values[0]
    roi = (df_top10[df_top10['CountyName']==i].max()['ROI'])*100

    ziplist.append(Zipcode)
    top_ROI[i] = roi
    print(f'County: {i} \nCity: {City}, Zipcode: {Zipcode}, Metro: {Metro}\n')
```

executed in 207ms, finished 18:57:40 2021-08-07

```
County: San Diego
City: San Diego, Zipcode: 92101, Metro: San Diego

County: Los Angeles
City: Monterey Park, Zipcode: 91754, Metro: Los Angeles-Long Beach-Anaheim

County: Orange
City: Orange, Zipcode: 92866, Metro: Los Angeles-Long Beach-Anaheim

County: Riverside
City: Norco, Zipcode: 92860, Metro: Riverside

County: Placer
City: Homewood, Zipcode: 96141, Metro: Sacramento

County: Sonoma
City: Geyserville, Zipcode: 95441, Metro: Santa Rosa

County: San Luis Obispo
City: San Luis Obispo, Zipcode: 93405, Metro: San Luis Obispo

County: Sacramento
City: Sacramento, Zipcode: 95818, Metro: Sacramento

County: Ventura
```

```
City: Ventura, Zipcode: 93003, Metro: Ventura

County: Alameda
City: Castro Valley, Zipcode: 94546, Metro: San Francisco
```

6 Time Series Analysis

In [40]:
```
ziplist
```
executed in 14ms, finished 18:57:40 2021-08-07

Out[40]:
```
[92101, 91754, 92866, 92860, 96141, 95441, 93405, 95818, 93003, 94546]
```

In [41]:
```
ziplist = ['92101', '91754', '92866', '92860', '96141', '95441', '93405',
           '95818', '93003', '94546']
```
executed in 11ms, finished 18:57:40 2021-08-07

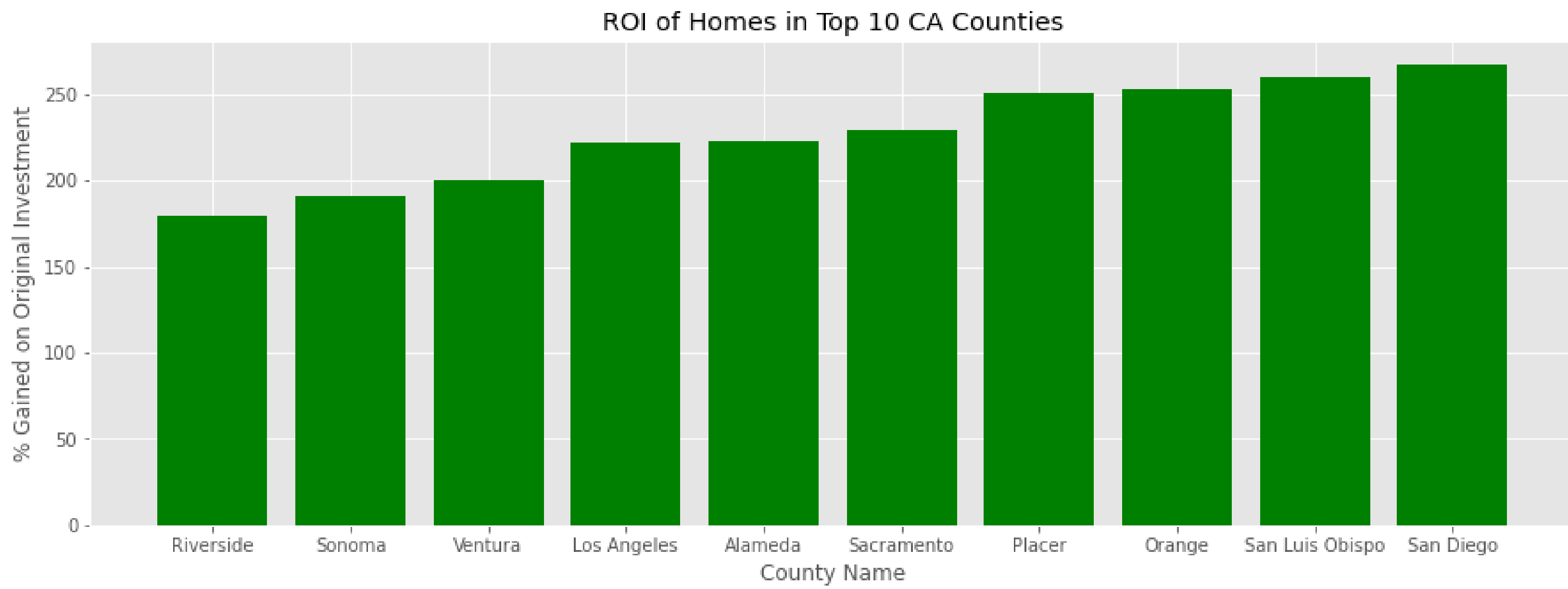
In [42]:
```
x = dict(sorted(top_ROI.items(), key=lambda item: item[1])).keys()
```
executed in 12ms, finished 18:57:40 2021-08-07

In [43]:
```
y = dict(sorted(top_ROI.items(), key=lambda item: item[1])).values()
```
executed in 14ms, finished 18:57:40 2021-08-07

In [44]:

```
plt.figure(figsize=(15,5))
plt.bar(x, y, color='green', )
plt.title('ROI of Homes in Top 10 CA Counties')
plt.xlabel('County Name')
plt.ylabel('% Gained on Original Investment')
plt.show()
```

executed in 218ms, finished 18:57:40 2021-08-07



The home sale values have turned up to show that our 10 counties have had an ROI gain of at least 150% of their original value from 1996. With Placer, Orange county, San Luis Obispo and San Diego making it to above 250%.

In [45]:

```
#create a dictionary for each zipcode
ts = {}
for zc in ziplist:
    temp_df = melted_df.groupby('ZipCode').get_group(zc).sort_index()['value']
    ts[zc] = temp_df
```

executed in 283ms, finished 18:57:40 2021-08-07

In [46]:

```
ts
```

executed in 29ms, finished 18:57:40 2021-08-07

Out[46]:

```
{'92101': Date
 1996-04-01    147000.0
 1996-05-01    147400.0
 1996-06-01    147700.0
 1996-07-01    148100.0
 1996-08-01    148500.0
                 ...
 2017-12-01    624900.0
 2018-01-01    625200.0
 2018-02-01    631800.0
 2018-03-01    644200.0
 2018-04-01    652600.0
 Name: value, Length: 265, dtype: float64,
 '91754': Date
 1996-04-01    188600.0
 1996-05-01    188000.0
 1996-06-01    187400.0
 1996-07-01    186900.0
 1996-08-01    186500.0
```

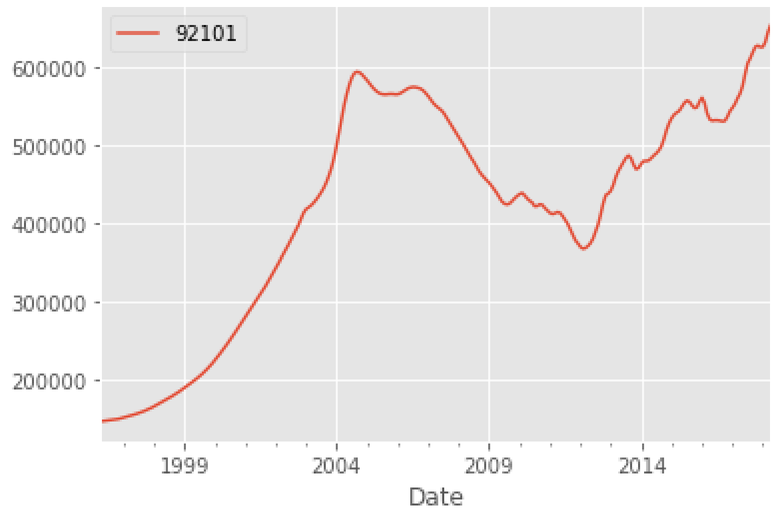
In [47]:
```
ts_df = pd.DataFrame(ts)
ts_df.head()
```
executed in 29ms, finished 18:57:40 2021-08-07

Out[47]:

Date	92101	91754	92866	92860	96141	95441	93405	95818	93003	94546
1996-04-01	147000.0	188600.0	173700.0	162900.0	170600.0	223800.0	181000.0	144000.0	182900.0	202100.0
1996-05-01	147400.0	188000.0	173600.0	162200.0	171800.0	222900.0	181700.0	144300.0	182700.0	201600.0
1996-06-01	147700.0	187400.0	173500.0	161500.0	172900.0	221900.0	182500.0	144500.0	182400.0	201100.0
1996-07-01	148100.0	186900.0	173500.0	160800.0	174000.0	220900.0	183300.0	144500.0	182200.0	200600.0
1996-08-01	148500.0	186500.0	173600.0	160200.0	175100.0	220000.0	184300.0	144600.0	182100.0	200200.0

In [48]:
```
zip_1 = ziplist[0]
```
executed in 13ms, finished 18:57:40 2021-08-07

In [49]:
```
ts_zip1 = ts_df[zip_1].copy()
ax = ts_zip1.plot()
ax.legend()
plt.show()
```
executed in 264ms, finished 18:57:40 2021-08-07



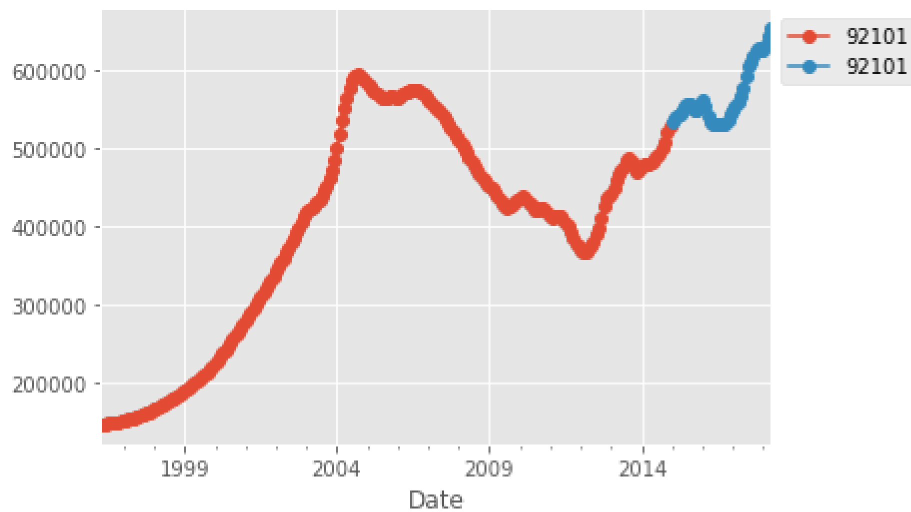
6.1 Model 1

6.1.1 Baseline Model

In [50]:
```
# selected params
d = 1
p = 1
q = 1
```
executed in 13ms, finished 18:57:40 2021-08-07

In [51]:
```
train_size = 0.85 #leaving approximately 3year for test size.
split_idx = round(len(ts_zip1)* train_size)
split_idx

## Split
train = ts_zip1.iloc[:split_idx]
test = ts_zip1.iloc[split_idx:]

## Visualize split
fig,ax= plt.subplots()
kws = dict(ax=ax,marker='o')
train.plot(**kws)
test.plot(**kws)
ax.legend(bbox_to_anchor=[1,1])
plt.show()
```

executed in 296ms, finished 18:57:41 2021-08-07



In [52]:
```
from statsmodels.tsa.statespace.sarimax import SARIMAX

## Baseline model from eye-balled params
model = SARIMAX(train,order=(p,d,q),).fit()
display(model.summary())
model.plot_diagnostics(figsize=(10,8));
plt.show()
```

executed in 1.06s, finished 18:57:42 2021-08-07

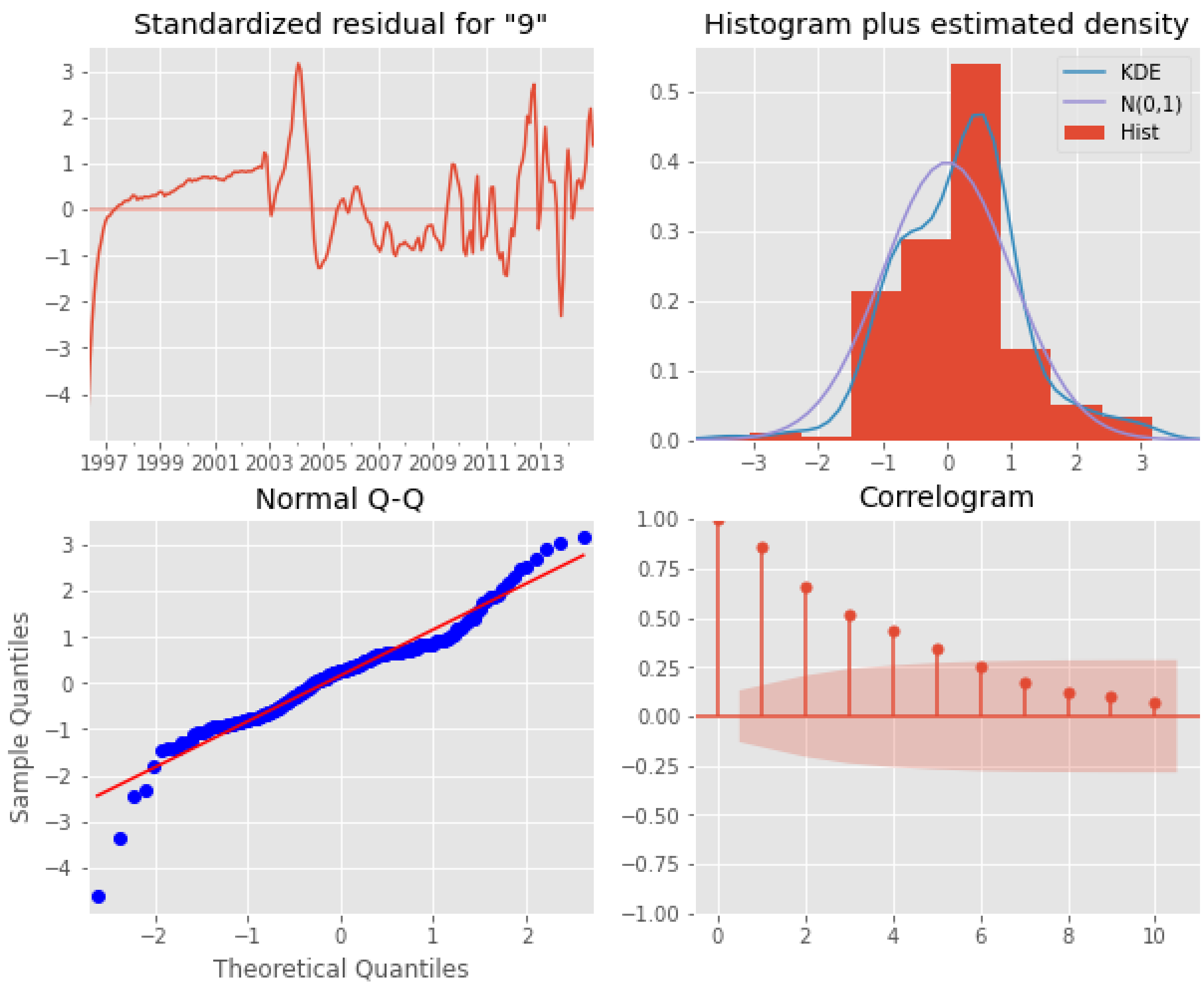
SARIMAX Results

Dep. Variable:	92101	**No. Observations:**	225
Model:	SARIMAX(1, 1, 1)	**Log Likelihood**	-2186.172
Date:	Sat, 07 Aug 2021	**AIC**	4378.344
Time:	18:57:41	**BIC**	4388.579
Sample:	04-01-1996	**HQIC**	4382.475
	- 12-01-2014		
Covariance Type:	opg		

	coef	**std err**	**z**	**P>|z|**	**[0.025**	**0.975]**
ar.L1	0.8545	0.028	30.691	0.000	0.800	0.909
ma.L1	-0.7447	0.034	-22.066	0.000	-0.811	-0.679
sigma2	1.717e+07	1.81e-10	9.51e+16	0.000	1.72e+07	1.72e+07

Ljung-Box (L1) (Q):	166.96	**Jarque-Bera (JB):**	78.23
Prob(Q):	0.00	**Prob(JB):**	0.00
Heteroskedasticity (H):	1.23	**Skew:**	-0.33
Prob(H) (two-sided):	0.36	**Kurtosis:**	5.82

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
[2] Covariance matrix is singular or near-singular, with condition number 8.48e+32. Standard errors may be unstable.



In [53]:

```
## obtaining forecast
from sklearn import metrics
forecast = model.get_forecast(steps=len(test))
```

executed in 15ms, finished 18:57:42 2021-08-07

In [54]:

```
def forecast_to_df(forecast,zipcode):
    test_pred = forecast.conf_int()
    test_pred[zipcode] = forecast.predicted_mean
    test_pred.columns = ['lower','upper','prediction']
    return test_pred

pred_df = forecast_to_df(forecast,zip_1)
pred_df.head()
```

executed in 21ms, finished 18:57:42 2021-08-07

Out[54]:

	lower	upper	prediction
2015-01-01	522605.595442	538850.383548	530727.989495
2015-02-01	521181.403406	545449.226187	533315.314796
2015-03-01	519943.687139	551108.524646	535526.105892
2015-04-01	518665.508443	556164.811388	537415.159916
2015-05-01	517299.090773	560759.507940	539029.299356

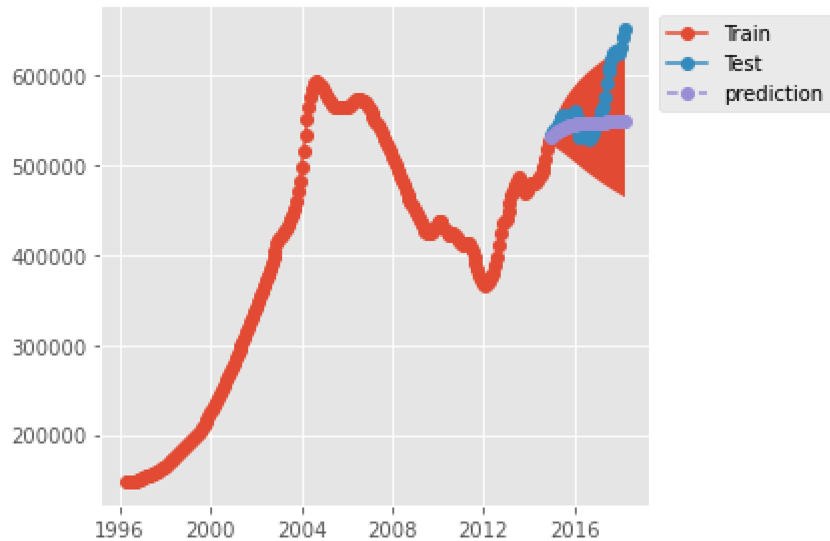
In [55]:

```
def plot_train_test_pred(train,test,pred_df):
    fig,ax = plt.subplots()
    kws = dict(marker='o')

    ax.plot(train,label='Train',**kws)
    ax.plot(test,label='Test',**kws)
    ax.plot(pred_df['prediction'],label='prediction',ls='--',**kws)

    ax.fill_between(x=pred_df.index,y1=pred_df['lower'],y2=pred_df['upper'])
    ax.legend(bbox_to_anchor=[1,1])
    fig.tight_layout()
    return fig,ax
```

executed in 16ms, finished 18:57:42 2021-08-07

In [56]:
```
plot_train_test_pred(train,test,pred_df)

plt.show()
```

executed in 224ms, finished 18:57:42 2021-08-07



Our first model doesn't seem to predict our test set very well. Let's use auto arima to generate a grid search for the optimum p, q values and see how well our model would perform then.

6.2 Model 2

In [57]:
```
auto_model = auto_arima(train,start_p=0,start_q=0)
display(auto_model.summary())
auto_model.plot_diagnostics(figsize=(10,8));
```
executed in 1.02s, finished 18:57:43 2021-08-07

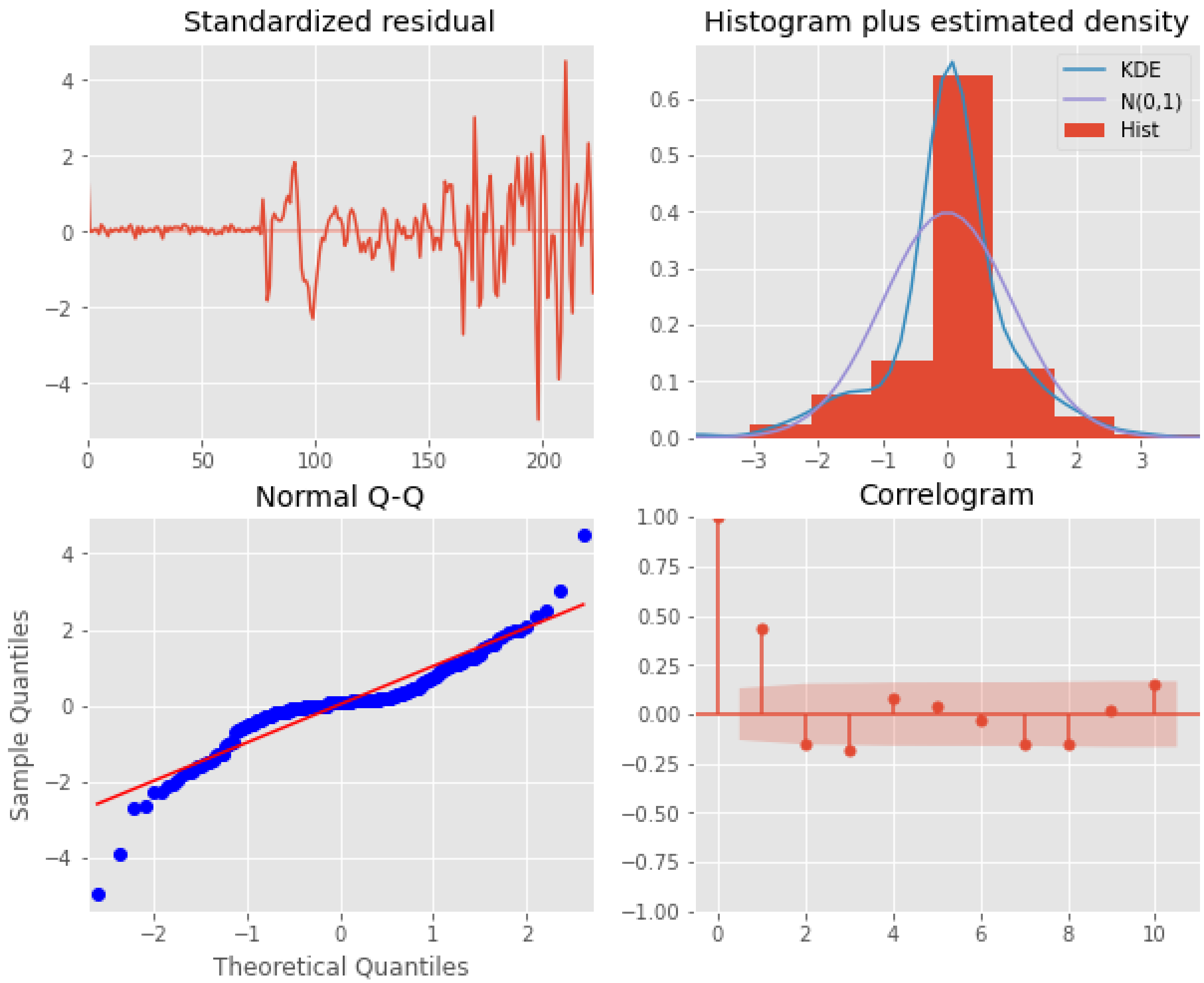
SARIMAX Results

Dep. Variable:	y	**No. Observations:**	225
Model:	SARIMAX(0, 2, 1)	**Log Likelihood**	-1969.676
Date:	Sat, 07 Aug 2021	**AIC**	3943.351
Time:	18:57:43	**BIC**	3950.165
Sample:	0	**HQIC**	3946.102
	- 225		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ma.L1	0.0335	0.010	3.394	0.001	0.014	0.053
sigma2	2.722e+06	1.38e+05	19.727	0.000	2.45e+06	2.99e+06

Ljung-Box (L1) (Q):	42.82	**Jarque-Bera (JB):**	252.12
Prob(Q):	0.00	**Prob(JB):**	0.00
Heteroskedasticity (H):	46.81	**Skew:**	-0.51
Prob(H) (two-sided):	0.00	**Kurtosis:**	8.11

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


Standardized residual
Histogram plus estimated density
KDE
N(0,1)
Hist
Normal Q-Q
Sample Quantiles
Theoretical Quantiles
Correlogram

In [58]:
```
model3 = SARIMAX(ts_zip1,order=auto_model.order,
                     seasonal_order=auto_model.seasonal_order).fit()
display(model3.summary())
model3.plot_diagnostics(figsize=(10,8));
```

executed in 719ms, finished 18:57:44 2021-08-07

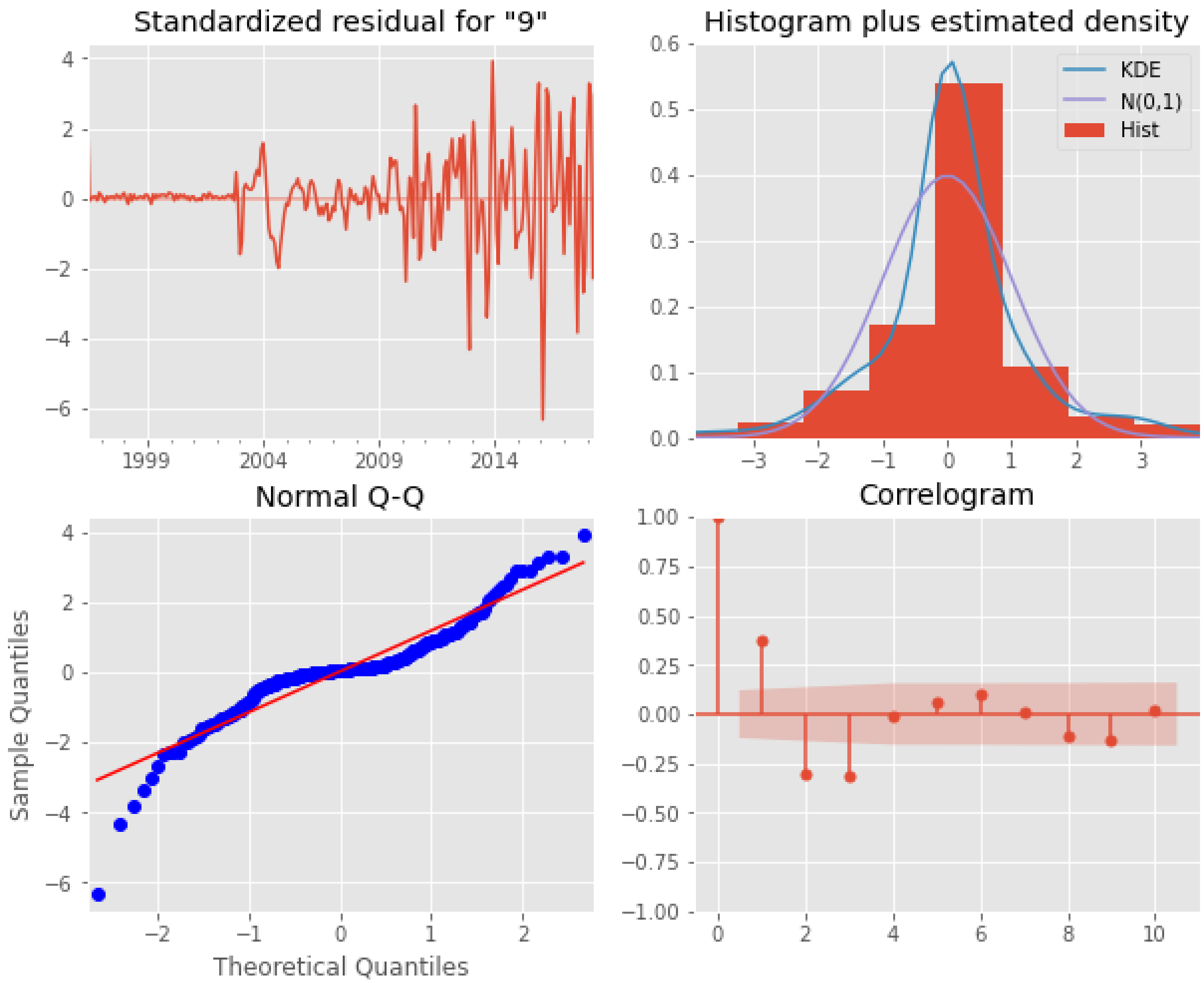
SARIMAX Results

Dep. Variable:	92101	**No. Observations:**	265
Model:	SARIMAX(0, 2, 1)	**Log Likelihood**	-2402.061
Date:	Sat, 07 Aug 2021	**AIC**	4808.121
Time:	18:57:43	**BIC**	4815.266
Sample:	04-01-1996	**HQIC**	4810.993
	- 04-01-2018		
Covariance Type:	opg		

	coef	**std err**	**z**	**P>|z|**	**[0.025**	**0.975]**
ma.L1	0.0484	0.007	6.748	0.000	0.034	0.062
sigma2	3.546e+06	1.25e+05	28.369	0.000	3.3e+06	3.79e+06

Ljung-Box (L1) (Q):	36.26	**Jarque-Bera (JB):**	280.13
Prob(Q):	0.00	**Prob(JB):**	0.00
Heteroskedasticity (H):	23.11	**Skew:**	-0.62
Prob(H) (two-sided):	0.00	**Kurtosis:**	7.90

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

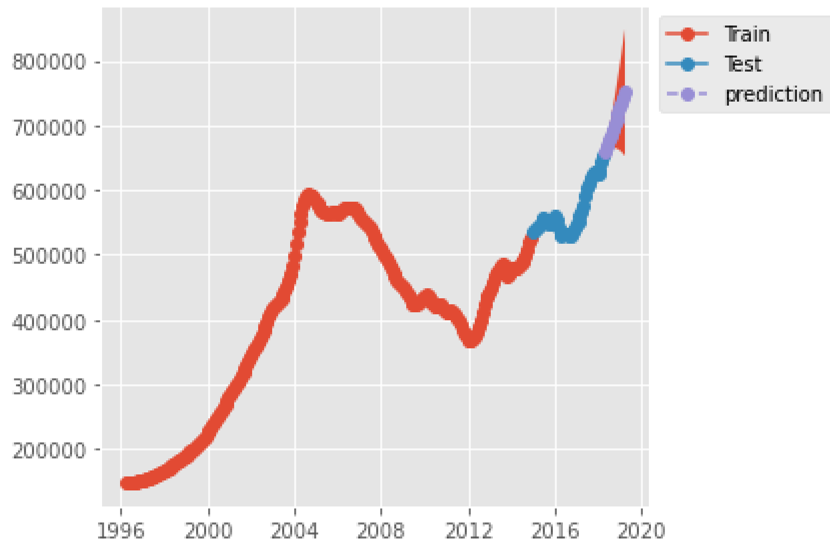

Standardized residual for "9"
Histogram plus estimated density
KDE
N(0,1)
Hist
Normal Q-Q
Sample Quantiles
Theoretical Quantiles
Correlogram

In [59]:

```
pred = model3.get_forecast(steps=12)#start=test.index[0],end=test.index[-1])
pred_df = forecast_to_df(pred,zip_1)
display(plot_train_test_pred(train,test,pred_df));
plt.show()
```

executed in 227ms, finished 18:57:44 2021-08-07

```
(<Figure size 432x288 with 1 Axes>, <AxesSubplot:>)
```



This looks much better and we will apply this same algorithm to the rest of the other zipcodes to get their forcast.

In [60]:

```
RESULTS = {}

for zc in ziplist:
    print(zc)

    ## Make empty dict for district data
    zipcode_d = {}

    ## Copy Time Series
    ts_final = ts_df[zc].copy()

    ## Train Test Split Index
    train_size = 0.85
    split_idx = round(len(ts_df)* train_size)

    ## Split
    train = ts_final.iloc[:split_idx]
    test = ts_final.iloc[split_idx:]


    ## Get best params using auto_arima
    gridsearch_model = auto_arima(ts_final,start_p=0,start_q=0)
    model3 = SARIMAX(ts_final,order=gridsearch_model.order,
                     seasonal_order=gridsearch_model.seasonal_order).fit()

    ## Get predictions
    pred  = model3.get_forecast(steps=36)#start=test.index[0],end=test.index[-36])
    pred_df = forecast_to_df(pred,zc)

    # Get the real and predicted values
    output = model3.get_prediction(start='2015-01',end='2018-04', dynamic=True)
    value_forcasted = output.predicted_mean
    print('Predicted mean budget: ', round(value_forcasted.max(), 1))
    value_truth = test[:]

    train_pred = model3.get_prediction(start='1996-04',end='2014-12')
    train_forcast = train_pred.predicted_mean
    train_true = train[:]

     # Compute the root mean square error for train set
#     mse_train = ((train_forcast - train_true) ** 2).mean()
#     rmse_train = math.sqrt(mse_train)
```

```
    rmse_train = np.sqrt(metrics.mean_squared_error(train_true, train_forcast))

    print('SARIMA model RMSE on train data: {}'.format(round(rmse_train, 1)))

    # Compute the root mean square error for test set
    mse = ((value_forcasted - value_truth) ** 2).mean()
    rmse = sqrt(mse)
    print('SARIMA model RMSE on test data: {}'.format(round(rmse, 1)))

    ## Save info to dict
    zipcode_d['pred_df'] = pred_df
    zipcode_d['model'] = model3
    zipcode_d['train'] = train
    zipcode_d['test'] = test

    ## Display Results
    display(model3.summary())
    plot_train_test_pred(train,test,pred_df)
    plt.xlabel('Year')
    plt.ylabel('Value in US Dollars ($)')
    plt.show()


    ## Save district dict in RESULTS
    RESULTS[zc] = zipcode_d
    print('---'*20,end='\n\n')
```

executed in 33.2s, finished 18:58:17 2021-08-07

```
92101
Predicted mean budget:  551370.7
SARIMA model RMSE on train data: 10714.2
SARIMA model RMSE on test data: 40096.9
```

SARIMAX Results

Dep. Variable:	92101	**No. Observations:**	265

In [61]:

```
#save data on each of the 10 zip codes
zip_96141 = melted_df[melted_df.ZipCode == '96141']
zip_93405 = melted_df[melted_df.ZipCode == '93405']
zip_92866 = melted_df[melted_df.ZipCode == '92866']
zip_92101 = melted_df[melted_df.ZipCode == '92101']
zip_95441 = melted_df[melted_df.ZipCode == '95441']
zip_94546 = melted_df[melted_df.ZipCode == '94546']
zip_91754 = melted_df[melted_df.ZipCode == '91754']
zip_92860 = melted_df[melted_df.ZipCode == '92860']
zip_95818 = melted_df[melted_df.ZipCode == '95818']
zip_93003 = melted_df[melted_df.ZipCode == '93003']
```

executed in 206ms, finished 18:58:17 2021-08-07

In [62]:

```
#Create a dataframe for the top10 zipcodes
zip_df = pd.DataFrame()
zip_df = zip_df.append(zip_96141)
zip_df = zip_df.append(zip_93405)
zip_df = zip_df.append(zip_92866)
zip_df = zip_df.append(zip_92101)
zip_df = zip_df.append(zip_95441)
zip_df = zip_df.append(zip_94546)
zip_df = zip_df.append(zip_91754)
zip_df = zip_df.append(zip_92860)
zip_df = zip_df.append(zip_95818)
zip_df = zip_df.append(zip_93003)
zip_df.head()
```

executed in 30ms, finished 18:58:17 2021-08-07

Out[62]:

	ZipCode	City	State	Metro	CountyName	value
Date						
1996-04-01	96141	Homewood	CA	Sacramento	Placer	170600.0
1996-05-01	96141	Homewood	CA	Sacramento	Placer	171800.0
1996-06-01	96141	Homewood	CA	Sacramento	Placer	172900.0
1996-07-01	96141	Homewood	CA	Sacramento	Placer	174000.0
1996-08-01	96141	Homewood	CA	Sacramento	Placer	175100.0

In [63]:
```
zip_ts = []
for zc in zip_df.ZipCode.unique():
    #Create separate dataframes for each zipcode with a monthly frequency.
    top5_df = zip_df[zip_df['ZipCode']==zc].asfreq('MS')
    zip_ts.append(top5_df)
zip_ts[0].head()
```

executed in 75ms, finished 18:58:18 2021-08-07

Out[63]:

	ZipCode	City	State	Metro	CountyName	value
Date						
1996-04-01	96141	Homewood	CA	Sacramento	Placer	170600.0
1996-05-01	96141	Homewood	CA	Sacramento	Placer	171800.0
1996-06-01	96141	Homewood	CA	Sacramento	Placer	172900.0
1996-07-01	96141	Homewood	CA	Sacramento	Placer	174000.0
1996-08-01	96141	Homewood	CA	Sacramento	Placer	175100.0

In [64]:

```
#checking how much each zipcode was impacted during the recession
for i in range(len(zip_ts)):
    print(zip_ts[i].ZipCode[0])
    ROI_crash = (zip_ts[i]['2011-01'].value[0]/zip_ts[i]['2006-01'].value[0])-1
    print('Price crash %', ROI_crash )
    print('-------------------------')
```

executed in 45ms, finished 18:58:18 2021-08-07

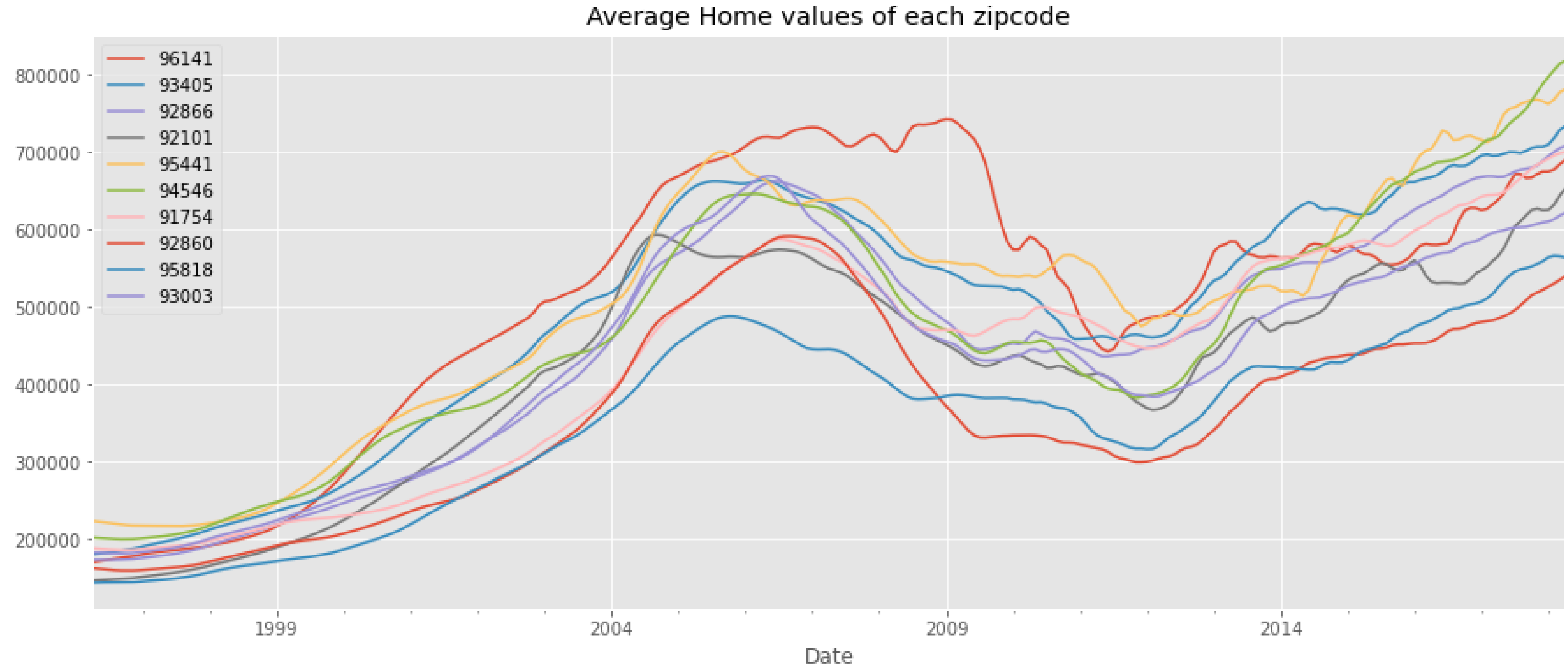
```
96141
Price crash % -0.31246466930469197
-------------------------
93405
Price crash % -0.30432145564821833
-------------------------
92866
Price crash % -0.30068836045056324
-------------------------
92101
Price crash % -0.26899238533734726
-------------------------
95441
Price crash % -0.16940133037694016
-------------------------
94546
Price crash % -0.3575964070001548
-------------------------
91754
Price crash % -0.1371367824238129
-------------------------
92860
Price crash % -0.4283687943262411
-------------------------
95818
Price crash % -0.26165910028889805
-------------------------
93003
Price crash % -0.336237200061134
-------------------------
```

In [65]:
```
for i in range(len(zip_ts)):
    print(f'Value descriptive statistics for zipcode {zip_ts[i].ZipCode[0]}:')
    print(f'{zip_ts[i].value.describe()}\n')
```

executed in 62ms, finished 18:58:18 2021-08-07

```
Value descriptive statistics for zipcode 96141:
count       265.000000
mean     515559.245283
std      174156.566619
min      170600.000000
25%      436900.000000
50%      564100.000000
75%      665600.000000
max      742600.000000
Name: value, dtype: float64

Value descriptive statistics for zipcode 93405:
count       265.000000
mean     492692.452830
std      161513.696118
min      181000.000000
25%      381300.000000
50%      526800.000000
75%      629700.000000
```

In [66]:
```
for i in range(10):
    zip_ts[i].value.plot(label=zip_ts[i].ZipCode[0],figsize=(15,6))
    plt.title('Average Home values of each zipcode')
    plt.legend()
```
executed in 491ms, finished 18:58:18 2021-08-07



6.3 Decomposition

Just to have a visual for the seasonality of each zip code we will take a look at the decomposition of one sample Zip Code.

In [67]:

```
# Import and apply seasonal_decompose()
def decompose(i):
    print('Zip code:', zip_ts[i]['ZipCode'][1])
    decomposition = seasonal_decompose(zip_ts[i]['value'])

    # Gather the trend, seasonality, and residuals
    trend = decomposition.trend
    seasonal = decomposition.seasonal
    residual = decomposition.resid

    # Plot gathered statistics
    plt.figure(figsize=(12,8))
    plt.subplot(411)
    plt.plot(zip_ts[i]['value'], label='Original', color='blue')
    plt.legend(loc='best')
    plt.subplot(412)
    plt.plot(trend, label='Trend', color='blue')
    plt.legend(loc='best')
    plt.subplot(413)
    plt.plot(seasonal,label='Seasonality', color='blue')
    plt.legend(loc='best')
    plt.subplot(414)
    plt.plot(residual, label='Residuals', color='blue')
    plt.legend(loc='best')
    plt.tight_layout()
```

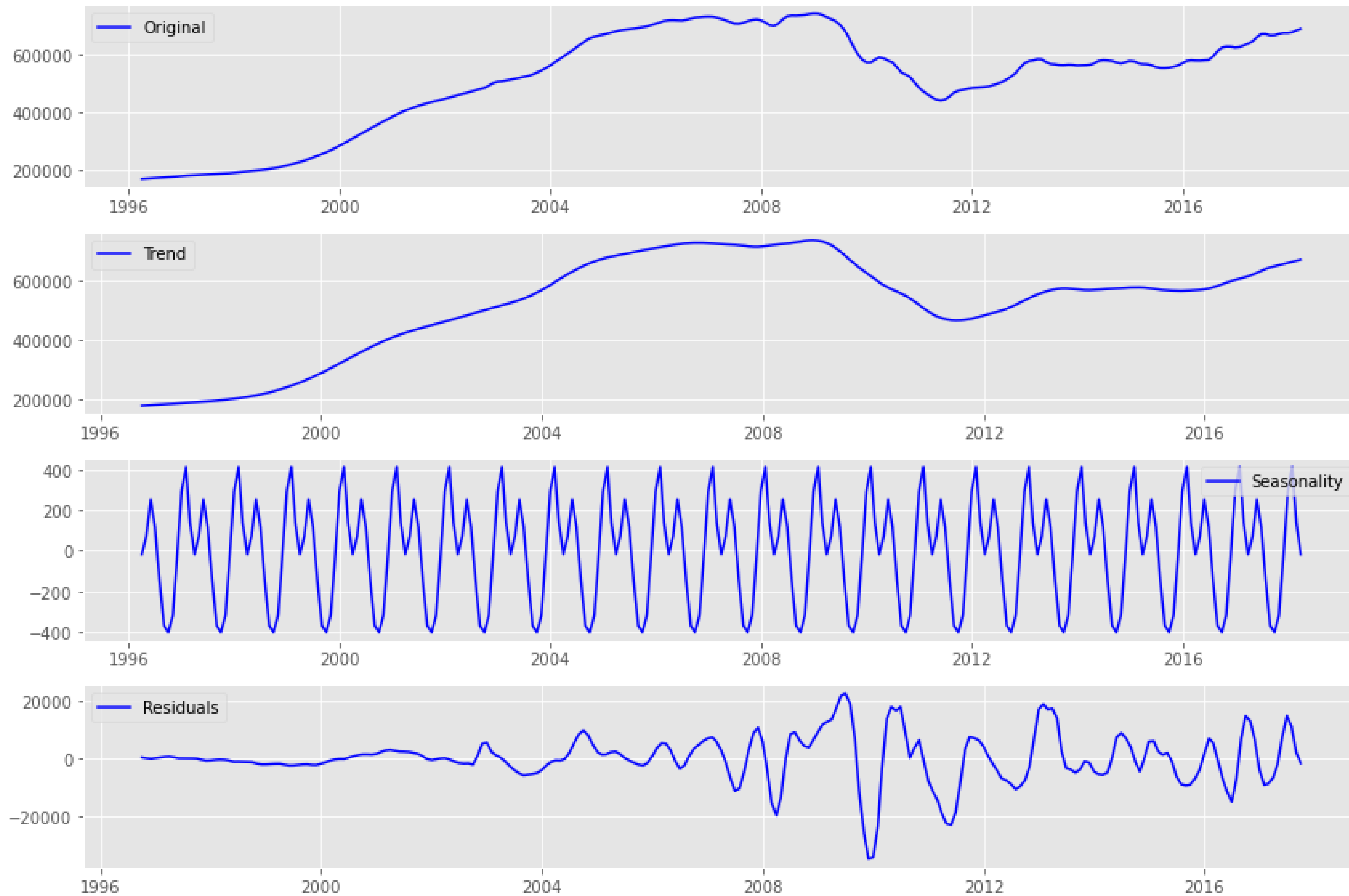
executed in 7ms, finished 18:58:18 2021-08-07

In [68]:

```
decompose(0)
```

executed in 675ms, finished 18:58:19 2021-08-07

```
Zip code: 96141
```



6.4 Checking Stationarity

In [69]:
```
#Calculate monthly returns in new column 'ret' for each zipcode.
for zc in range(len(zip_ts)):
    zip_ts[zc]['ret']=np.nan*len(zip_ts[zc])
    for i in range(len(zip_ts[zc])-1):
        zip_ts[zc]['ret'][i+1]= (zip_ts[zc].value.iloc[i+1] / zip_ts[zc].value.iloc[i]) - 1
```

executed in 283ms, finished 18:58:19 2021-08-07

In [70]:

```
for i in range(10):
    results = adfuller(zip_ts[i].ret.dropna())
    print(f'ADFuller test p-value for zipcode: {zip_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')
```

executed in 124ms, finished 18:58:19 2021-08-07

```
ADFuller test p-value for zipcode: 96141
p-value: 0.243487362447878
Fail to reject the null hypothesis. Data is not stationary.

ADFuller test p-value for zipcode: 93405
p-value: 0.16677983660591333
Fail to reject the null hypothesis. Data is not stationary.

ADFuller test p-value for zipcode: 92866
p-value: 0.3421318160724609
Fail to reject the null hypothesis. Data is not stationary.

ADFuller test p-value for zipcode: 92101
p-value: 0.4027524899782298
Fail to reject the null hypothesis. Data is not stationary.

ADFuller test p-value for zipcode: 95441
p-value: 0.16014965341653276
Fail to reject the null hypothesis. Data is not stationary.

ADFuller test p-value for zipcode: 94546
p-value: 0.1548027613969144
Fail to reject the null hypothesis. Data is not stationary.

ADFuller test p-value for zipcode: 91754
p-value: 0.07484124248592604
Fail to reject the null hypothesis. Data is not stationary.

ADFuller test p-value for zipcode: 92860
p-value: 0.17266830963718194
Fail to reject the null hypothesis. Data is not stationary.
```

```
ADFuller test p-value for zipcode: 95818
p-value: 0.014372364546267133
Reject the null hypothesis. Data is stationary.

ADFuller test p-value for zipcode: 93003
p-value: 0.17104926074824278
Fail to reject the null hypothesis. Data is not stationary.
```

In [71]:

```
results
```

executed in 14ms, finished 18:58:19 2021-08-07

Out[71]:

```
(-2.302853661235002,
 0.17104926074824278,
 9,
 254,
 {'1%': -3.456360306409983,
  '5%': -2.8729872043802356,
  '10%': -2.572870232500465},
 -2388.4581644761092)
```

Except for the one zipcode 95818 we fail to reject the null hypothesis that the data is not stationary.

In [72]:

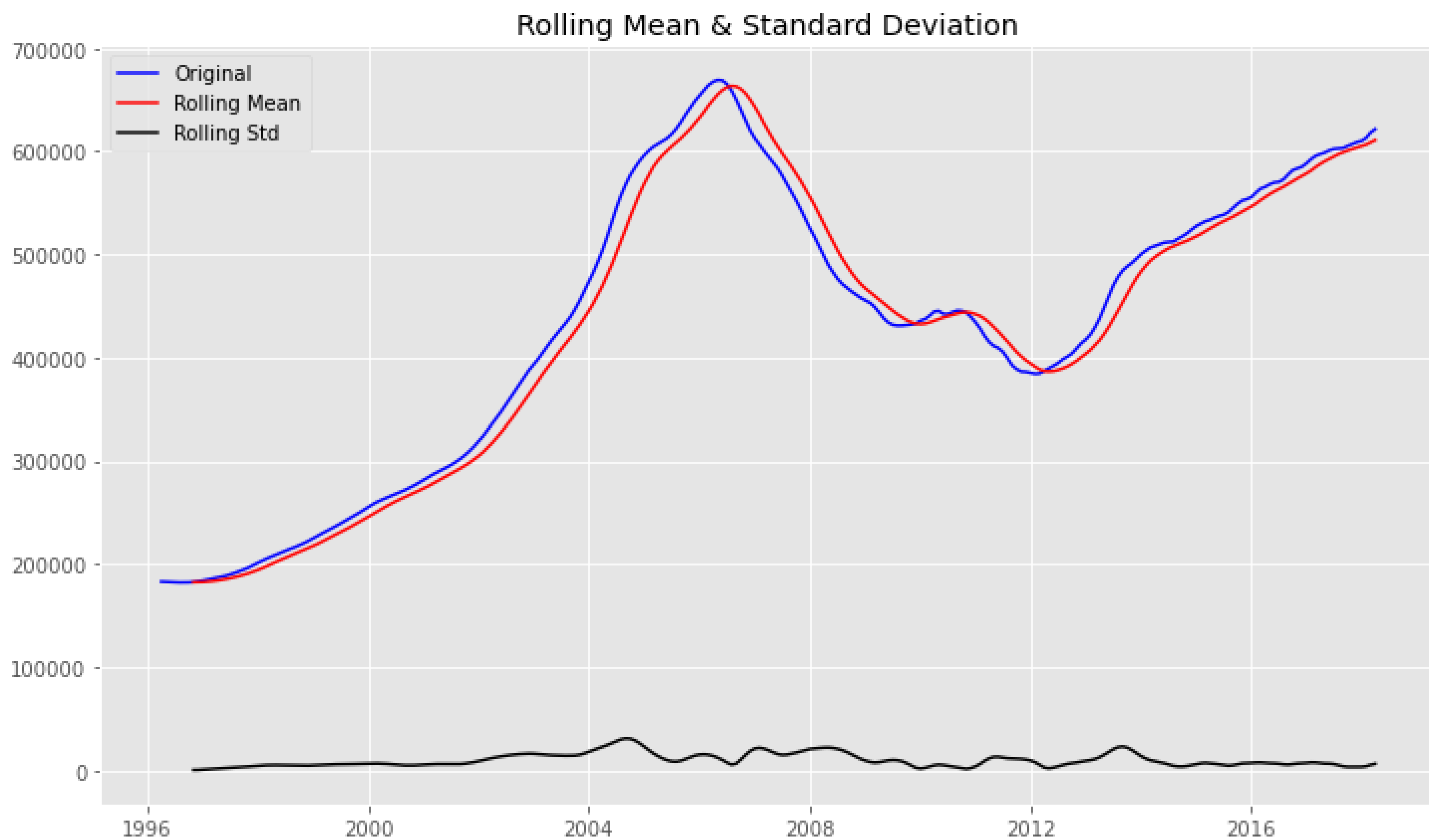
```
roll_mean = top5_df['value'].rolling(window=8, center=False).mean()
roll_std = top5_df['value'].rolling(window=8, center=False).std()
```

executed in 14ms, finished 18:58:19 2021-08-07

In [73]:

```
fig = plt.figure(figsize=(12,7))
plt.plot(top5_df['value'], color='blue', label='Original')
plt.plot(roll_mean, color='red', label='Rolling Mean')
plt.plot(roll_std, color='black', label = 'Rolling Std')
plt.legend(loc='best')
plt.title('Rolling Mean & Standard Deviation')
plt.show(block=False)
```

executed in 265ms, finished 18:58:20 2021-08-07



In [74]:
```
for i in range(10):
    #Perform adfuller test and drop NaN values created when calculating monthly returns.
    results = adfuller(zip_ts[i].ret.diff().dropna()) #differencing by 12 month for stationarity
    print(f'ADFuller test p-value for zipcode: {zip_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')
```

executed in 138ms, finished 18:58:20 2021-08-07

```
ADFuller test p-value for zipcode: 96141
p-value: 6.372408708384718e-06
Reject the null hypothesis. Data is stationary.

ADFuller test p-value for zipcode: 93405
p-value: 7.132708870112685e-09
Reject the null hypothesis. Data is stationary.

ADFuller test p-value for zipcode: 92866
p-value: 1.0822992598653813e-05
Reject the null hypothesis. Data is stationary.

ADFuller test p-value for zipcode: 92101
p-value: 6.688588032968088e-10
Reject the null hypothesis. Data is stationary.

ADFuller test p-value for zipcode: 95441
p-value: 5.945230489199505e-09
Reject the null hypothesis. Data is stationary.

ADFuller test p-value for zipcode: 94546
p-value: 1.261712636851411e-05
Reject the null hypothesis. Data is stationary.

ADFuller test p-value for zipcode: 91754
p-value: 4.067941774457979e-06
Reject the null hypothesis. Data is stationary.

ADFuller test p-value for zipcode: 92860
p-value: 0.0021966709002367406
Reject the null hypothesis. Data is stationary.
```

```
ADFuller test p-value for zipcode: 95818
p-value: 1.5766521837150803e-05
Reject the null hypothesis. Data is stationary.

ADFuller test p-value for zipcode: 93003
p-value: 1.4440409858025173e-06
Reject the null hypothesis. Data is stationary.
```

In [75]:

```
results
```

executed in 13ms, finished 18:58:20 2021-08-07

Out[75]:

```
(-5.574267864700025,
 1.4440409858025173e-06,
 8,
 254,
 {'1%': -3.456360306409983,
  '5%': -2.8729872043802356,
  '10%': -2.572870232500465},
 -2374.7864994036972)
```

In [76]:
```
zip_ts
```

executed in 77ms, finished 18:58:20 2021-08-07

Out[76]:
```
[            ZipCode      City State       Metro CountyName     value       ret
 Date
 1996-04-01    96141  Homewood    CA  Sacramento     Placer  170600.0       NaN
 1996-05-01    96141  Homewood    CA  Sacramento     Placer  171800.0  0.007034
 1996-06-01    96141  Homewood    CA  Sacramento     Placer  172900.0  0.006403
 1996-07-01    96141  Homewood    CA  Sacramento     Placer  174000.0  0.006362
 1996-08-01    96141  Homewood    CA  Sacramento     Placer  175100.0  0.006322
 ...             ...       ...   ...         ...        ...       ...       ...
 2017-12-01    96141  Homewood    CA  Sacramento     Placer  675000.0  0.004763
 2018-01-01    96141  Homewood    CA  Sacramento     Placer  675000.0  0.000000
 2018-02-01    96141  Homewood    CA  Sacramento     Placer  677500.0  0.003704
 2018-03-01    96141  Homewood    CA  Sacramento     Placer  684400.0  0.010185
 2018-04-01    96141  Homewood    CA  Sacramento     Placer  689700.0  0.007744

 [265 rows x 7 columns],
             ZipCode             City State            Metro       CountyName  \
 Date
 1996-04-01    93405  San Luis Obispo    CA  San Luis Obispo  San Luis Obispo
 1996-05-01    93405  San Luis Obispo    CA  San Luis Obispo  San Luis Obispo
```

In [77]:

```
#Create individual time series for each of the positive zipcodes
TS_96141 = zip_ts[0].ret.dropna()
TS_96141d = zip_ts[0].ret.diff().dropna()

TS_93405 = zip_ts[1].ret.dropna()
TS_93405d = zip_ts[1].ret.diff().dropna()

TS_92866 = zip_ts[2].ret.dropna()
TS_92866d = zip_ts[2].ret.diff().dropna()

TS_92101 = zip_ts[3].ret.dropna()
TS_92101d = zip_ts[3].ret.diff().dropna()

TS_95441 = zip_ts[4].ret.dropna()
TS_95441d = zip_ts[4].ret.diff().dropna()

TS_94546 = zip_ts[5].ret.dropna()
TS_94546d = zip_ts[5].ret.diff().dropna()

TS_91754 = zip_ts[6].ret.dropna()
TS_91754d = zip_ts[6].ret.diff().dropna()

TS_92860 = zip_ts[7].ret.dropna()
TS_92860d = zip_ts[7].ret.diff().dropna()

TS_95818 = zip_ts[8].ret.dropna()
# TS_95818d = zip_ts[8].ret.diff().dropna()

TS_93003 = zip_ts[9].ret.dropna()
TS_93003d = zip_ts[9].ret.diff().dropna()
```

executed in 14ms, finished 18:58:20 2021-08-07

In [78]:

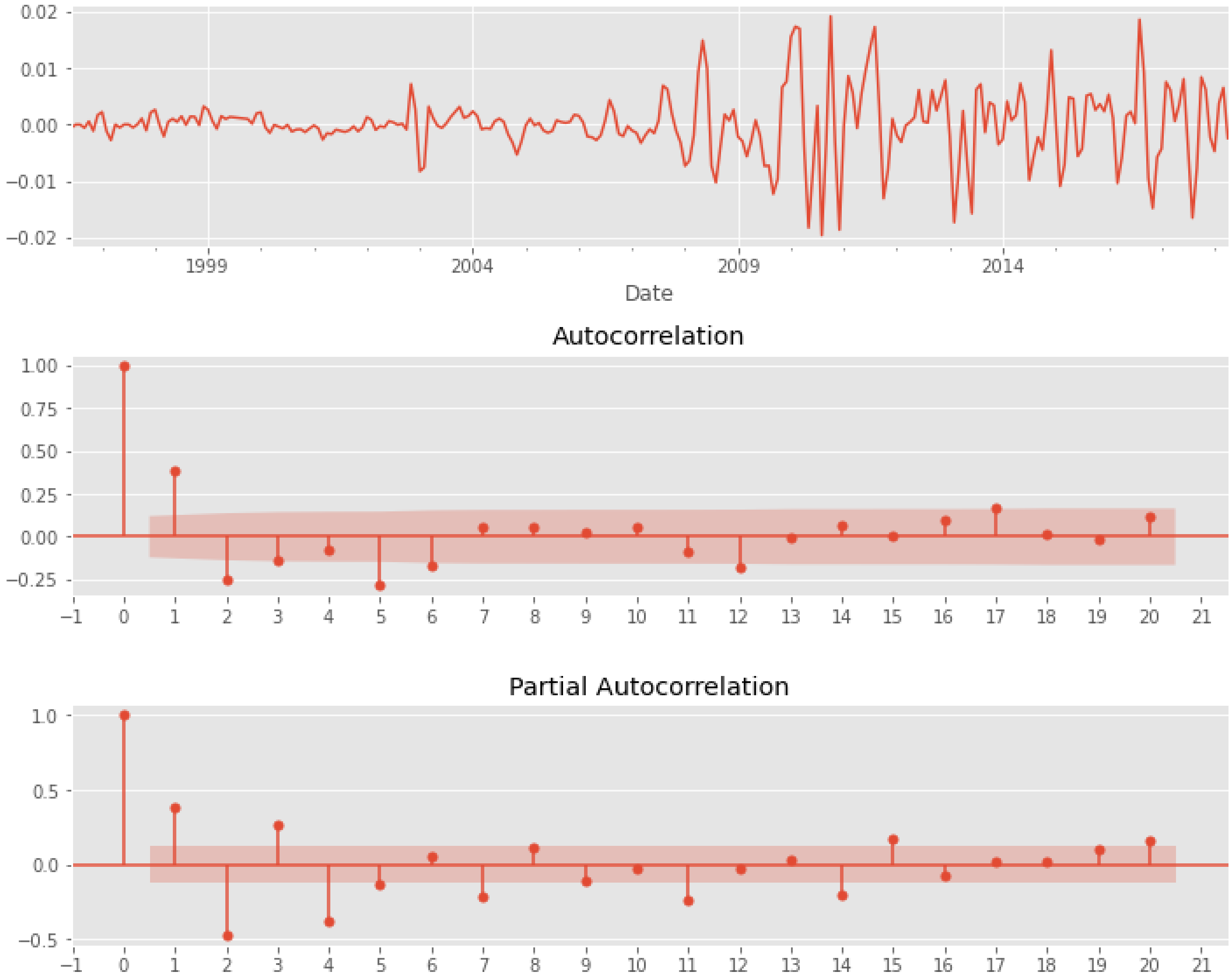
```
def plot_acf_pacf(ts, figsize=(10,8),lags=24):

    fig,ax = plt.subplots(nrows=3, figsize=figsize)

    ts.plot(ax=ax[0])

    plot_acf(ts,ax=ax[1],lags=lags)
    plot_pacf(ts, ax=ax[2],lags=lags)
    fig.tight_layout()


    for a in ax[1:]:
        a.xaxis.set_major_locator(mpl.ticker.MaxNLocator(min_n_ticks=lags, integer=True))
        a.xaxis.grid()
    return fig,ax

def seasonal_plots(df,N=4,lags=[12,24,36,48,60,72]):
    #Differencing the rolling mean to find seasonality in the resulting acf plot.
    fig,(ax1,ax2) = plt.subplots(2,1,figsize=(13,8))
    rolling = df - df.rolling(N).mean()
    plot_acf(rolling.dropna(),lags=lags,ax=ax1)
    plot_pacf(rolling.dropna(),lags=lags,ax=ax2)
    plt.show();

def model_fit(df,pdq=(1,0,1),pdqs=(0,0,0,1)):
    train, test = train_test(df)
    model = SARIMAX(train,order=pdq,seasonal_order=pdqs)
    results = model.fit()
    results.summary
    residuals = results.resid
    print(results.summary())
    results.plot_diagnostics(figsize=(11,8))
    plt.show();
    return train, test, results

def forecast_model(df,pdq=(1,0,1),pdqs=(0,0,0,12), display=True,zc='input zipcode'):
    model = SARIMAX(df, order=pdq,seasonal_order=pdqs)
    model_fit = model.fit()
    output = model_fit.get_prediction(start='2018-04',end='2028-04', dynamic=True)

    forecast_ci = output.conf_int()
    if display:
```

```
        fig, ax = plt.subplots(figsize=(13,6))
        output.predicted_mean.plot(label='Forecast')
        ax.fill_between(forecast_ci.index,forecast_ci.iloc[:, 0],forecast_ci.iloc[:, 1],
                        color='k', alpha=.25,label='Conf Interval')
        plt.title('Forecast of Monthly Returns')
        plt.xlabel('Time')
        plt.legend(loc='best')
        plt.show()
#     year_1= (1+output.predicted_mean[:12]).prod()-1
    year_1= (1+output.predicted_mean[:12]).prod()-1
    year_3=(1+output.predicted_mean[:36]).prod()-1
    year_5= (1+output.predicted_mean[:60]).prod()-1
    year_10=(1+output.predicted_mean).prod()-1
    print(f'Total expected return in 1 year: {round(year_1*100,2)}%')
    print(f'Total expected return in 3 years: {round(year_3*100,2)}%')
    print(f'Total expected return in 5 year: {round(year_5*100,2)}%')
    print(f'Total expected return in 10 years: {round(year_10*100,2)}%')
    tot_ret = [zc,year_1,year_3,year_5,year_10]
    return tot_ret
```

executed in 29ms, finished 18:58:20 2021-08-07

7 Zipcode 96141: Placer county

In [79]:
```
plot_acf_pacf(TS_96141d,lags=20);
```
executed in 629ms, finished 18:58:20 2021-08-07



Even though the data lines after differencing do seem to be fluctuating, the movements seem to be completely random, and the same conclusion holds for the original time series.

In [80]:
```
results = pm.auto_arima(TS_96141d,information_criterion='aic',m=12,
                        start_p=0,start_q=0, max_p=3, max_q=3,
                        stepwise=True,trace=True,error_action='ignore',suppress_warnings=True)
results
```

executed in 29.9s, finished 18:58:50 2021-08-07

```
Performing stepwise search to minimize aic
 ARIMA(0,0,0)(1,0,1)[12] intercept   : AIC=-1973.757, Time=0.39 sec
 ARIMA(0,0,0)(0,0,0)[12] intercept   : AIC=-1967.345, Time=0.05 sec
 ARIMA(1,0,0)(1,0,0)[12] intercept   : AIC=-2016.851, Time=0.41 sec
 ARIMA(0,0,1)(0,0,1)[12] intercept   : AIC=-2085.945, Time=0.38 sec
 ARIMA(0,0,0)(0,0,0)[12]             : AIC=-1969.345, Time=0.03 sec
 ARIMA(0,0,1)(0,0,0)[12] intercept   : AIC=-2075.458, Time=0.11 sec
 ARIMA(0,0,1)(1,0,1)[12] intercept   : AIC=-2080.524, Time=0.15 sec
 ARIMA(0,0,1)(0,0,2)[12] intercept   : AIC=-2112.062, Time=0.36 sec
 ARIMA(0,0,1)(1,0,2)[12] intercept   : AIC=-2102.526, Time=0.69 sec
 ARIMA(0,0,0)(0,0,2)[12] intercept   : AIC=-2023.178, Time=0.28 sec
 ARIMA(1,0,1)(0,0,2)[12] intercept   : AIC=inf, Time=0.96 sec
 ARIMA(0,0,2)(0,0,2)[12] intercept   : AIC=-2109.295, Time=1.03 sec
 ARIMA(1,0,0)(0,0,2)[12] intercept   : AIC=-2052.711, Time=0.63 sec
 ARIMA(1,0,2)(0,0,2)[12] intercept   : AIC=-2115.132, Time=0.75 sec
 ARIMA(1,0,2)(0,0,1)[12] intercept   : AIC=-2086.189, Time=0.58 sec
 ARIMA(1,0,2)(1,0,2)[12] intercept   : AIC=inf, Time=1.16 sec
 ARIMA(1,0,2)(1,0,1)[12] intercept   : AIC=-2082.136, Time=0.83 sec
 ARIMA(2,0,2)(0,0,2)[12] intercept   : AIC=-2131.931, Time=0.81 sec
 ARIMA(2,0,2)(0,0,1)[12] intercept   : AIC=-2104.012, Time=0.63 sec
 ARIMA(2,0,2)(1,0,2)[12] intercept   : AIC=-2116.655, Time=0.74 sec
 ARIMA(2,0,2)(1,0,1)[12] intercept   : AIC=-2101.921, Time=0.78 sec
 ARIMA(2,0,1)(0,0,2)[12] intercept   : AIC=-2134.237, Time=0.63 sec
 ARIMA(2,0,1)(0,0,1)[12] intercept   : AIC=-2110.821, Time=0.58 sec
 ARIMA(2,0,1)(1,0,2)[12] intercept   : AIC=-2125.313, Time=0.90 sec
 ARIMA(2,0,1)(1,0,1)[12] intercept   : AIC=-2100.346, Time=0.21 sec
 ARIMA(2,0,0)(0,0,2)[12] intercept   : AIC=-2118.687, Time=1.21 sec
 ARIMA(3,0,1)(0,0,2)[12] intercept   : AIC=-2136.928, Time=0.55 sec
 ARIMA(3,0,1)(0,0,1)[12] intercept   : AIC=-2112.538, Time=0.58 sec
 ARIMA(3,0,1)(1,0,2)[12] intercept   : AIC=-2128.287, Time=0.68 sec
 ARIMA(3,0,1)(1,0,1)[12] intercept   : AIC=-2109.969, Time=0.87 sec
 ARIMA(3,0,0)(0,0,2)[12] intercept   : AIC=-2133.646, Time=1.05 sec
 ARIMA(3,0,2)(0,0,2)[12] intercept   : AIC=-2137.208, Time=1.47 sec
 ARIMA(3,0,2)(0,0,1)[12] intercept   : AIC=-2103.105, Time=0.32 sec
 ARIMA(3,0,2)(1,0,2)[12] intercept   : AIC=-2121.555, Time=0.59 sec
 ARIMA(3,0,2)(1,0,1)[12] intercept   : AIC=-2097.252, Time=0.18 sec
```

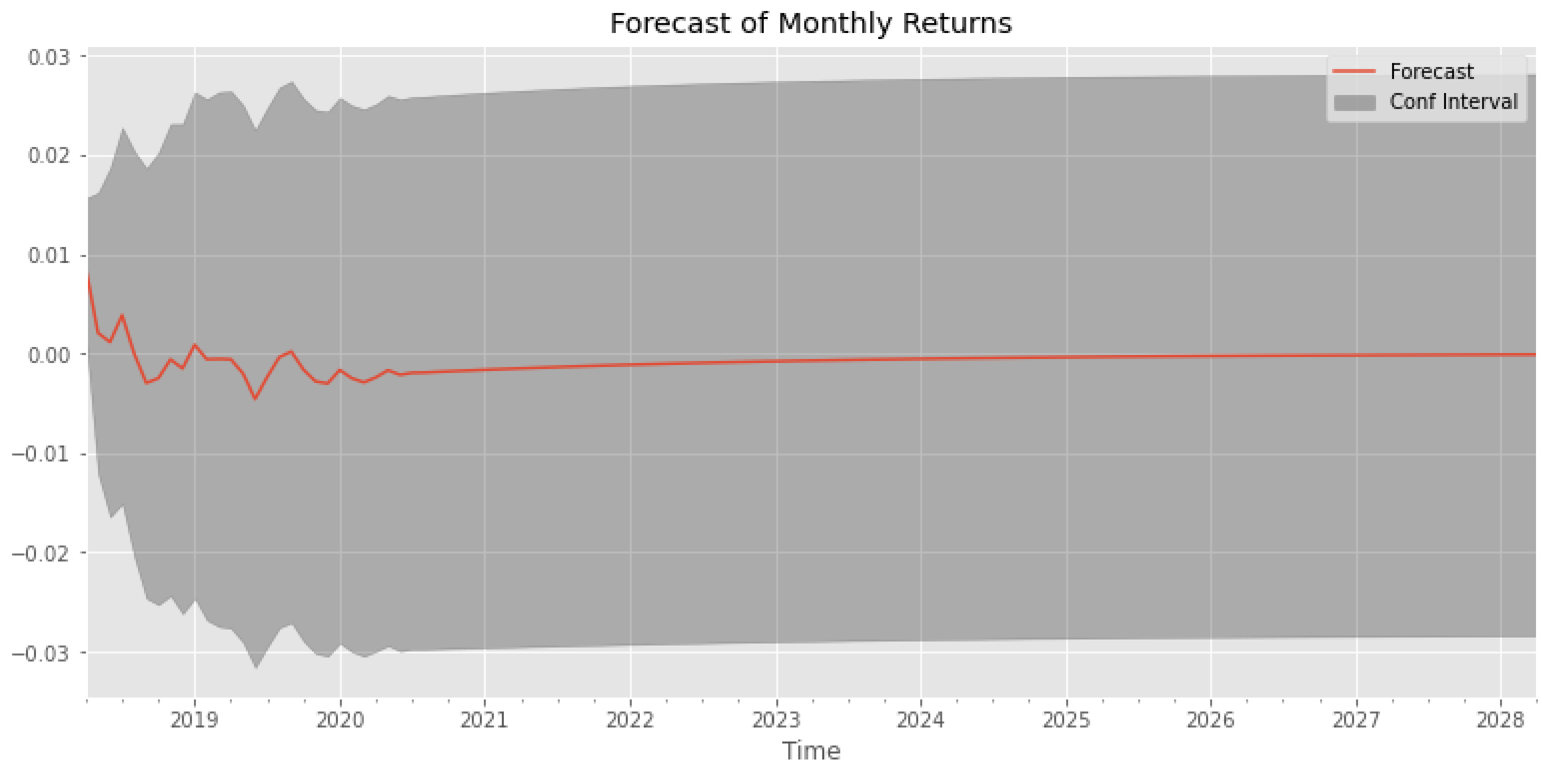
```
 ARIMA(3,0,3)(0,0,2)[12] intercept   : AIC=-2166.499, Time=1.22 sec
 ARIMA(3,0,3)(0,0,1)[12] intercept   : AIC=-2142.380, Time=0.33 sec
 ARIMA(3,0,3)(1,0,2)[12] intercept   : AIC=-2157.848, Time=0.70 sec
 ARIMA(3,0,3)(1,0,1)[12] intercept   : AIC=-2136.598, Time=0.63 sec
 ARIMA(2,0,3)(0,0,2)[12] intercept   : AIC=-2171.395, Time=1.43 sec
 ARIMA(2,0,3)(0,0,1)[12] intercept   : AIC=-2162.669, Time=0.67 sec
 ARIMA(2,0,3)(1,0,2)[12] intercept   : AIC=-2146.341, Time=0.85 sec
 ARIMA(2,0,3)(1,0,1)[12] intercept   : AIC=-2125.027, Time=0.43 sec
 ARIMA(1,0,3)(0,0,2)[12] intercept   : AIC=-2169.993, Time=0.99 sec
 ARIMA(2,0,3)(0,0,2)[12]             : AIC=-2157.290, Time=1.05 sec

Best model:  ARIMA(2,0,3)(0,0,2)[12] intercept
Total fit time: 29.859 seconds
```

Out[80]:

```
ARIMA(maxiter=50, method='lbfgs', order=(2, 0, 3), out_of_sample_size=0,
      scoring='mse', scoring_args={}, seasonal_order=(0, 0, 2, 12),
      start_params=None, suppress_warnings=True, trend=None,
      with_intercept=True)
```

In [81]:
```
pdq = (2, 0, 3)
pdqs = (0, 0, 2, 12)
ret_96141 = forecast_model(TS_96141,pdq=pdq,pdqs=pdqs,zc=96141)
```

executed in 1.37s, finished 18:58:52 2021-08-07



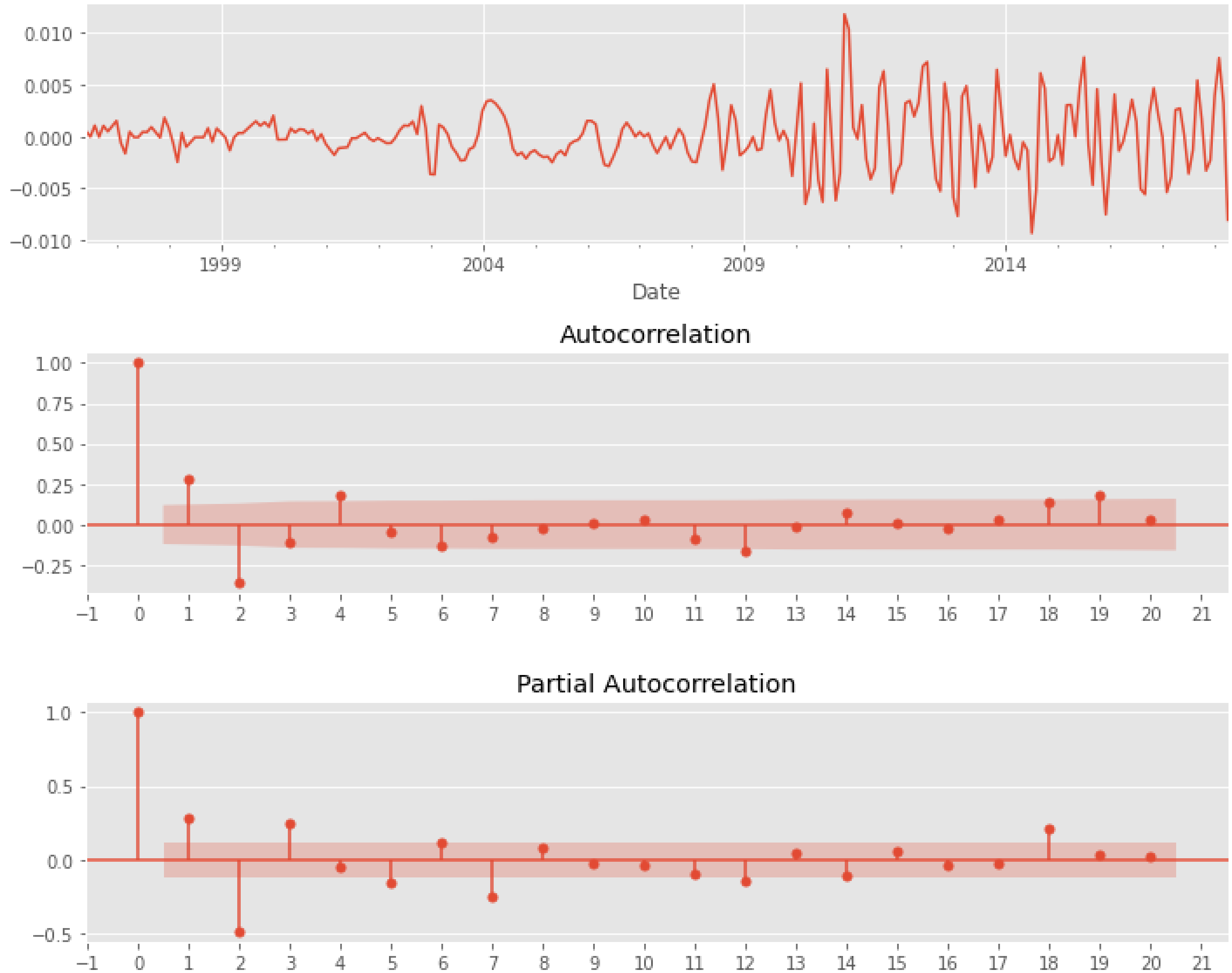

```
Total expected return in 1 year: 0.77%
Total expected return in 3 years: -3.8%
Total expected return in 5 year: -6.27%
Total expected return in 10 years: -8.15%
```

In [ ]:
```
```

8 Zipcode 93405: San Luis Obispo

In [82]:
```
plot_acf_pacf(TS_93405d,lags=20)
```
executed in 539ms, finished 18:58:52 2021-08-07

Out[82]:
```
(<Figure size 720x576 with 3 Axes>,
 array([<AxesSubplot:xlabel='Date'>,
        <AxesSubplot:title={'center':'Autocorrelation'}>,
        <AxesSubplot:title={'center':'Partial Autocorrelation'}>],
       dtype=object))
```



The ACF and PACF have just one very strong correlation, right at 1 month.

In [83]:

```
results = pm.auto_arima(TS_93405d,information_criterion='aic',m=12,d=0,
                        start_p=1,start_q=1, max_p=3, max_q=3,
                        stepwise=True,trace=True,error_action='ignore',suppress_warnings=True)
results
```

executed in 24.6s, finished 18:59:17 2021-08-07

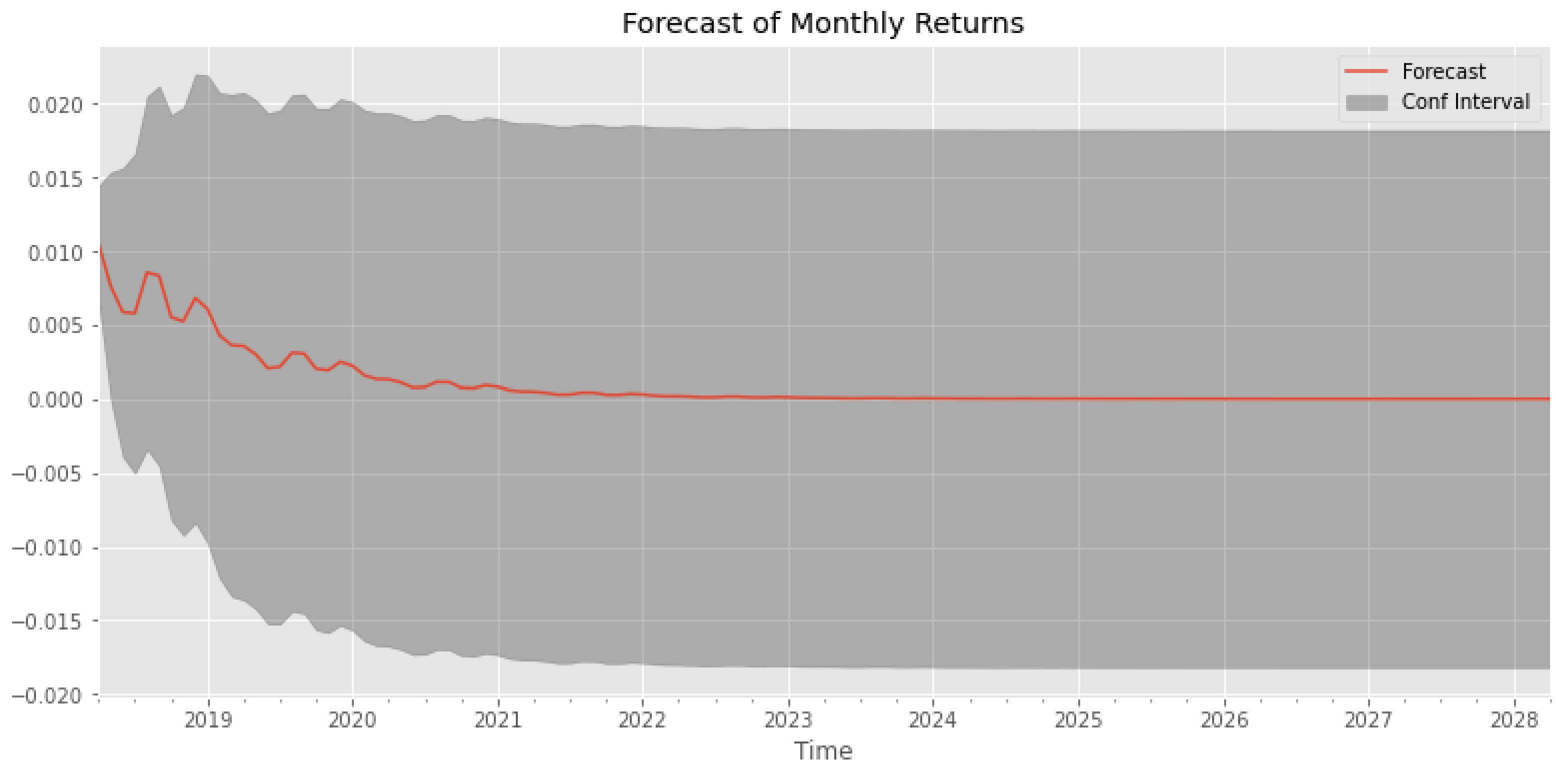
```
Performing stepwise search to minimize aic
 ARIMA(1,0,1)(1,0,1)[12] intercept   : AIC=-2407.891, Time=0.62 sec
 ARIMA(0,0,0)(0,0,0)[12] intercept   : AIC=-2317.198, Time=0.03 sec
 ARIMA(1,0,0)(1,0,0)[12] intercept   : AIC=-2343.228, Time=0.11 sec
 ARIMA(0,0,1)(0,0,1)[12] intercept   : AIC=-2395.193, Time=0.32 sec
 ARIMA(0,0,0)(0,0,0)[12]             : AIC=-2319.194, Time=0.03 sec
 ARIMA(1,0,1)(0,0,1)[12] intercept   : AIC=-2358.138, Time=0.36 sec
 ARIMA(1,0,1)(1,0,0)[12] intercept   : AIC=-2356.349, Time=0.68 sec
 ARIMA(1,0,1)(2,0,1)[12] intercept   : AIC=-2407.837, Time=1.18 sec
 ARIMA(1,0,1)(1,0,2)[12] intercept   : AIC=-2340.974, Time=0.85 sec
 ARIMA(1,0,1)(0,0,0)[12] intercept   : AIC=-2352.938, Time=0.17 sec
 ARIMA(1,0,1)(0,0,2)[12] intercept   : AIC=-2348.885, Time=0.41 sec
 ARIMA(1,0,1)(2,0,0)[12] intercept   : AIC=-2369.916, Time=0.99 sec
 ARIMA(1,0,1)(2,0,2)[12] intercept   : AIC=-2381.941, Time=1.26 sec
 ARIMA(0,0,1)(1,0,1)[12] intercept   : AIC=-2389.867, Time=0.35 sec
 ARIMA(1,0,0)(1,0,1)[12] intercept   : AIC=-2343.619, Time=0.22 sec
 ARIMA(2,0,1)(1,0,1)[12] intercept   : AIC=-2414.390, Time=0.26 sec
 ARIMA(2,0,1)(0,0,1)[12] intercept   : AIC=-2420.668, Time=0.21 sec
 ARIMA(2,0,1)(0,0,0)[12] intercept   : AIC=-2413.421, Time=0.14 sec
 ARIMA(2,0,1)(0,0,2)[12] intercept   : AIC=-2460.003, Time=1.07 sec
 ARIMA(2,0,1)(1,0,2)[12] intercept   : AIC=-2438.956, Time=0.61 sec
 ARIMA(2,0,0)(0,0,2)[12] intercept   : AIC=-2438.971, Time=0.44 sec
 ARIMA(3,0,1)(0,0,2)[12] intercept   : AIC=-2450.951, Time=0.50 sec
 ARIMA(2,0,2)(0,0,2)[12] intercept   : AIC=-2465.467, Time=0.67 sec
 ARIMA(2,0,2)(0,0,1)[12] intercept   : AIC=-2444.857, Time=0.34 sec
 ARIMA(2,0,2)(1,0,2)[12] intercept   : AIC=-2460.028, Time=1.13 sec
 ARIMA(2,0,2)(1,0,1)[12] intercept   : AIC=-2439.151, Time=0.24 sec
 ARIMA(1,0,2)(0,0,2)[12] intercept   : AIC=-2419.366, Time=0.73 sec
 ARIMA(3,0,2)(0,0,2)[12] intercept   : AIC=-2336.672, Time=0.68 sec
 ARIMA(2,0,3)(0,0,2)[12] intercept   : AIC=-2478.180, Time=0.60 sec
 ARIMA(2,0,3)(0,0,1)[12] intercept   : AIC=-2465.347, Time=0.50 sec
 ARIMA(2,0,3)(1,0,2)[12] intercept   : AIC=-2472.671, Time=1.31 sec
 ARIMA(2,0,3)(1,0,1)[12] intercept   : AIC=-2484.115, Time=0.88 sec
 ARIMA(2,0,3)(1,0,0)[12] intercept   : AIC=-2449.766, Time=0.34 sec
 ARIMA(2,0,3)(2,0,1)[12] intercept   : AIC=-2464.572, Time=0.60 sec
 ARIMA(2,0,3)(0,0,0)[12] intercept   : AIC=-2452.813, Time=0.36 sec
```

```
 ARIMA(2,0,3)(2,0,0)[12] intercept   : AIC=-2465.499, Time=0.65 sec
 ARIMA(2,0,3)(2,0,2)[12] intercept   : AIC=-2461.646, Time=0.82 sec
 ARIMA(1,0,3)(1,0,1)[12] intercept   : AIC=-2478.075, Time=0.84 sec
 ARIMA(3,0,3)(1,0,1)[12] intercept   : AIC=-2448.276, Time=1.18 sec
 ARIMA(1,0,2)(1,0,1)[12] intercept   : AIC=-2391.704, Time=0.70 sec
 ARIMA(3,0,2)(1,0,1)[12] intercept   : AIC=-2418.895, Time=0.82 sec
 ARIMA(2,0,3)(1,0,1)[12]             : AIC=-2466.412, Time=0.36 sec

Best model:  ARIMA(2,0,3)(1,0,1)[12] intercept
Total fit time: 24.583 seconds
```

Out[83]:
```
ARIMA(maxiter=50, method='lbfgs', order=(2, 0, 3), out_of_sample_size=0,
      scoring='mse', scoring_args={}, seasonal_order=(1, 0, 1, 12),
      start_params=None, suppress_warnings=True, trend=None,
      with_intercept=True)
```

In [84]:
```
pdq = (2, 0, 3)
pdqs = (1, 0, 1, 12)
ret_93405 = forecast_model(TS_93405,pdq=pdq,pdqs=pdqs,zc=90504)
```

executed in 954ms, finished 18:59:18 2021-08-07

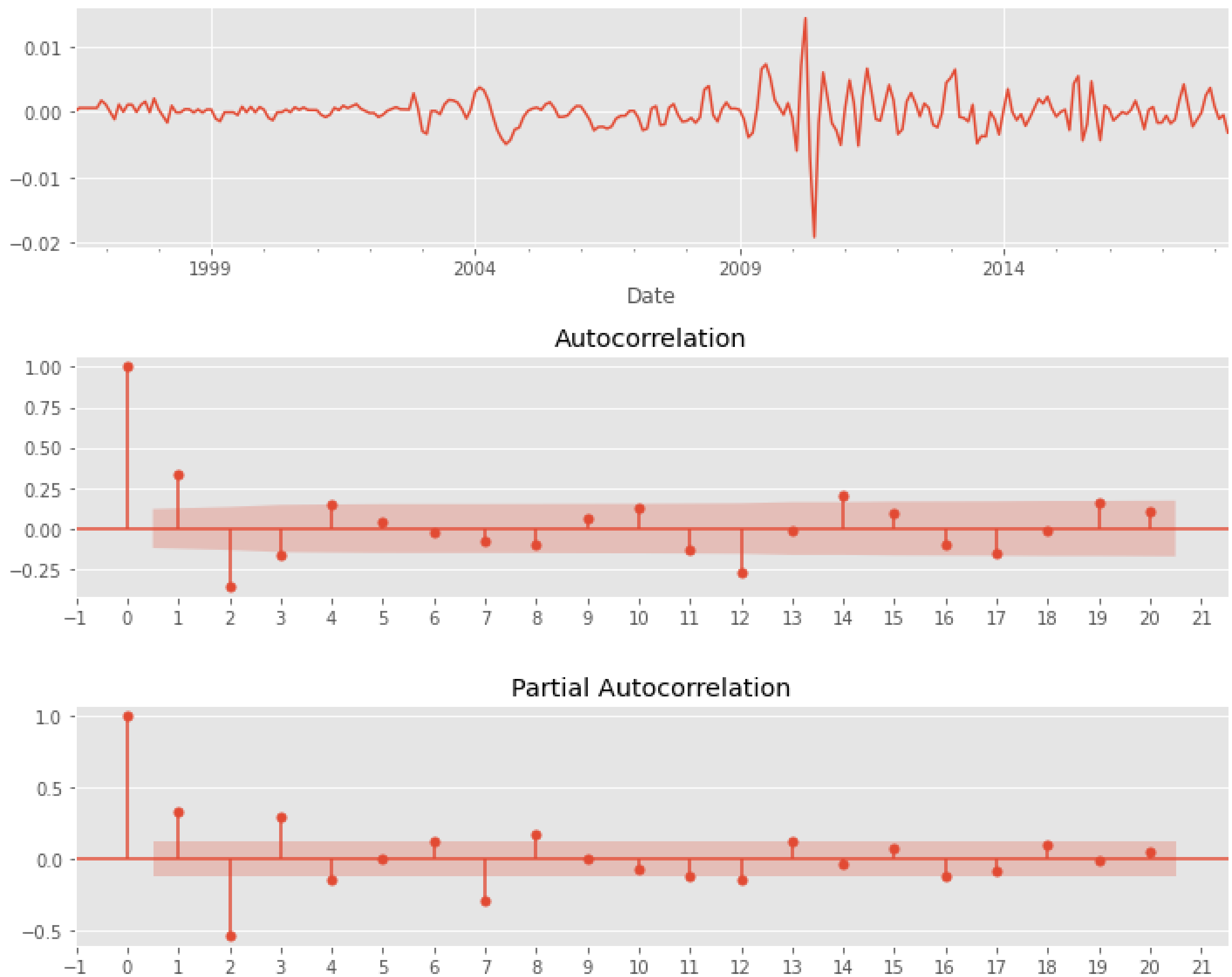



```
Total expected return in 1 year: 8.1%
Total expected return in 3 years: 12.39%
Total expected return in 5 year: 13.0%
Total expected return in 10 years: 13.13%
```

9 Zipcode 92866: Los Angeles-Long Beach-Anaheim

In [85]:
```
plot_acf_pacf(TS_92866d,lags=20)
```
executed in 542ms, finished 18:59:18 2021-08-07

Out[85]:
```
(<Figure size 720x576 with 3 Axes>,
 array([<AxesSubplot:xlabel='Date'>,
        <AxesSubplot:title={'center':'Autocorrelation'}>,
        <AxesSubplot:title={'center':'Partial Autocorrelation'}>],
       dtype=object))
```



In [86]:
```
results = pm.auto_arima(TS_92866d,information_criterion='aic',m=12,d=0,
                        start_p=1,start_q=1, max_p=3, max_q=3,
                        stepwise=True,trace=True,error_action='ignore',suppress_warnings=True)
results
```

executed in 5.95s, finished 18:59:24 2021-08-07

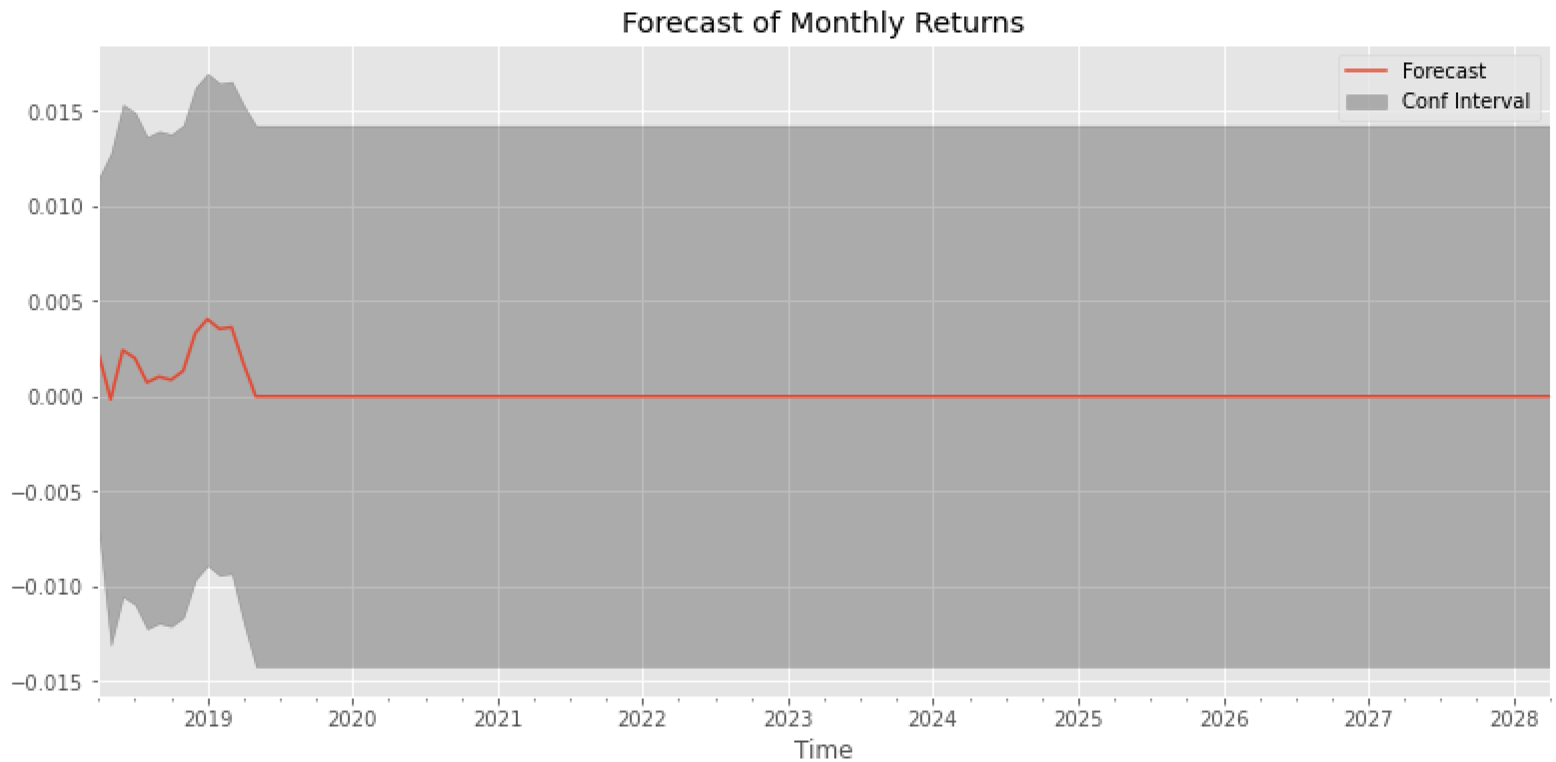
```
Performing stepwise search to minimize aic
 ARIMA(1,0,1)(1,0,1)[12] intercept   : AIC=-2364.192, Time=0.25 sec
 ARIMA(0,0,0)(0,0,0)[12] intercept   : AIC=-2374.893, Time=0.05 sec
 ARIMA(1,0,0)(1,0,0)[12] intercept   : AIC=-2424.396, Time=0.23 sec
 ARIMA(0,0,1)(0,0,1)[12] intercept   : AIC=-2488.801, Time=0.22 sec
 ARIMA(0,0,0)(0,0,0)[12]             : AIC=-2376.881, Time=0.02 sec
 ARIMA(0,0,1)(0,0,0)[12] intercept   : AIC=-2471.526, Time=0.15 sec
 ARIMA(0,0,1)(1,0,1)[12] intercept   : AIC=-2483.753, Time=0.36 sec
 ARIMA(0,0,1)(0,0,2)[12] intercept   : AIC=-2488.244, Time=0.41 sec
 ARIMA(0,0,1)(1,0,0)[12] intercept   : AIC=-2485.630, Time=0.20 sec
 ARIMA(0,0,1)(1,0,2)[12] intercept   : AIC=-2484.540, Time=0.60 sec
 ARIMA(0,0,0)(0,0,1)[12] intercept   : AIC=-2396.960, Time=0.28 sec
 ARIMA(1,0,1)(0,0,1)[12] intercept   : AIC=-2370.483, Time=0.24 sec
 ARIMA(0,0,2)(0,0,1)[12] intercept   : AIC=-2317.288, Time=0.56 sec
 ARIMA(1,0,0)(0,0,1)[12] intercept   : AIC=-2427.547, Time=0.18 sec
 ARIMA(1,0,2)(0,0,1)[12] intercept   : AIC=-2328.877, Time=0.33 sec
 ARIMA(0,0,1)(0,0,1)[12]             : AIC=-2490.804, Time=0.09 sec
 ARIMA(0,0,1)(0,0,0)[12]             : AIC=-2473.500, Time=0.04 sec
 ARIMA(0,0,1)(1,0,1)[12]             : AIC=-2484.268, Time=0.13 sec
 ARIMA(0,0,1)(0,0,2)[12]             : AIC=-2490.247, Time=0.20 sec
 ARIMA(0,0,1)(1,0,0)[12]             : AIC=-2486.901, Time=0.10 sec
 ARIMA(0,0,1)(1,0,2)[12]             : AIC=-2486.544, Time=0.21 sec
 ARIMA(0,0,0)(0,0,1)[12]             : AIC=-2398.959, Time=0.07 sec
 ARIMA(1,0,1)(0,0,1)[12]             : AIC=-2372.809, Time=0.25 sec
 ARIMA(0,0,2)(0,0,1)[12]             : AIC=-2320.488, Time=0.44 sec
 ARIMA(1,0,0)(0,0,1)[12]             : AIC=-2429.552, Time=0.09 sec
 ARIMA(1,0,2)(0,0,1)[12]             : AIC=-2331.129, Time=0.18 sec

Best model:  ARIMA(0,0,1)(0,0,1)[12]
Total fit time: 5.906 seconds
```

Out[86]:
```
ARIMA(maxiter=50, method='lbfgs', order=(0, 0, 1), out_of_sample_size=0,
      scoring='mse', scoring_args={}, seasonal_order=(0, 0, 1, 12),
      start_params=None, suppress_warnings=True, trend=None,
      with_intercept=False)
```

In [87]:

```
pdq = (0, 0, 1)
pdqs = (0, 0, 1, 12)
ret_92866 = forecast_model(TS_92866,pdq=pdq,pdqs=pdqs,zc=92866)
```

executed in 530ms, finished 18:59:25 2021-08-07

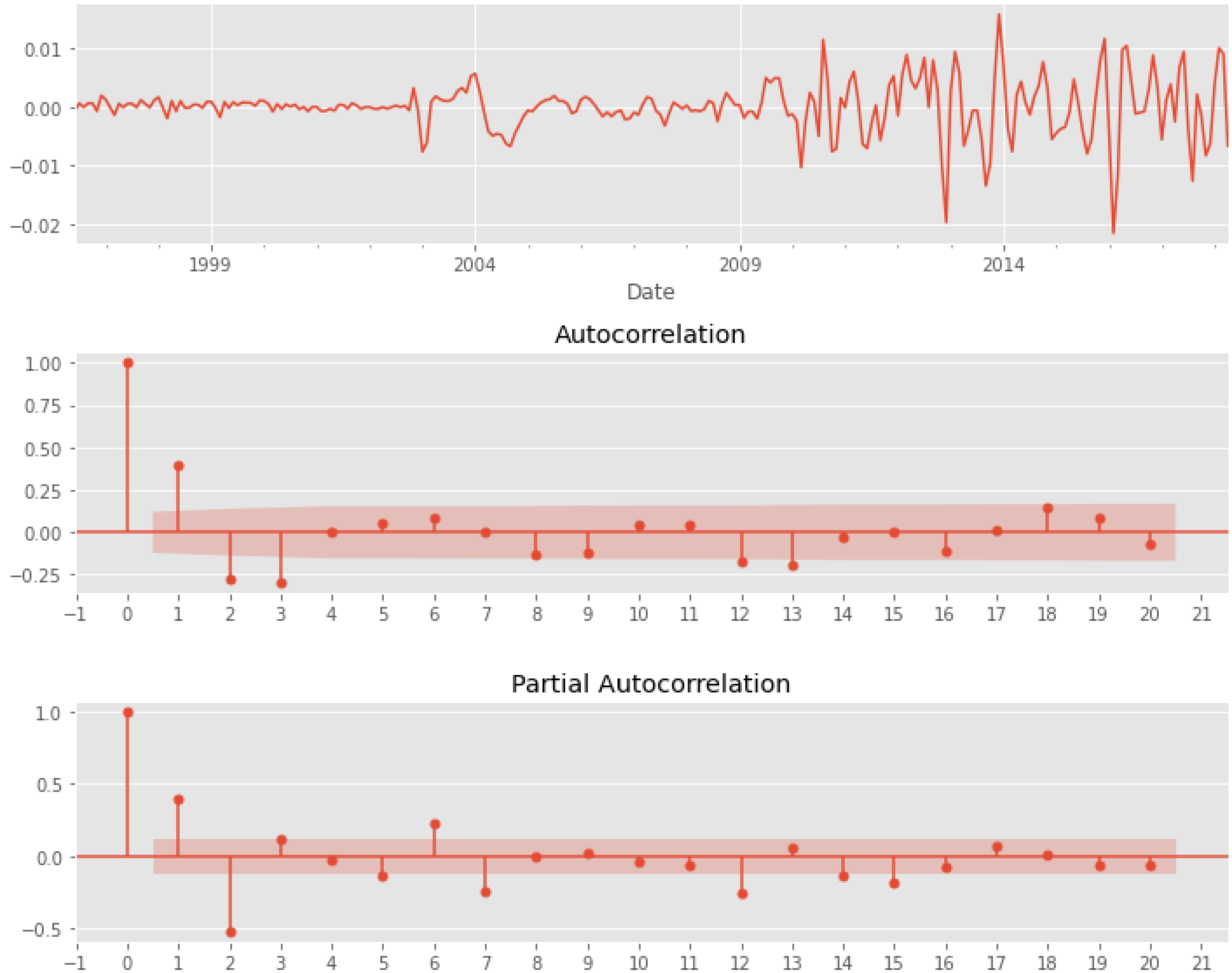



```
Total expected return in 1 year: 2.5%
Total expected return in 3 years: 2.67%
Total expected return in 5 year: 2.67%
Total expected return in 10 years: 2.67%
```

10 Zipcode 92101: San Diego

In [88]:
```
plot_acf_pacf(TS_92101d,lags=20)
```
executed in 473ms, finished 18:59:25 2021-08-07

Out[88]:
```
(<Figure size 720x576 with 3 Axes>,
 array([<AxesSubplot:xlabel='Date'>,
        <AxesSubplot:title={'center':'Autocorrelation'}>,
        <AxesSubplot:title={'center':'Partial Autocorrelation'}>],
       dtype=object))
```



The ACF and PACF have just one very strong correlation, right at 2 month.

In [89]:

```
results = pm.auto_arima(TS_92101d,information_criterion='aic',m=12,d=0,
                        start_p=1,start_q=1, max_p=3, max_q=3,
                        stepwise=True,trace=True,error_action='ignore',suppress_warnings=True)
results
```

executed in 15.6s, finished 18:59:41 2021-08-07

```
Performing stepwise search to minimize aic
 ARIMA(1,0,1)(1,0,1)[12] intercept   : AIC=-2187.409, Time=0.23 sec
 ARIMA(0,0,0)(0,0,0)[12] intercept   : AIC=-2101.257, Time=0.05 sec
 ARIMA(1,0,0)(1,0,0)[12] intercept   : AIC=-2151.089, Time=0.13 sec
 ARIMA(0,0,1)(0,0,1)[12] intercept   : AIC=-2198.253, Time=0.10 sec
 ARIMA(0,0,0)(0,0,0)[12]             : AIC=-2103.236, Time=0.03 sec
 ARIMA(0,0,1)(0,0,0)[12] intercept   : AIC=-2190.008, Time=0.08 sec
 ARIMA(0,0,1)(1,0,1)[12] intercept   : AIC=-2193.963, Time=0.26 sec
 ARIMA(0,0,1)(0,0,2)[12] intercept   : AIC=-2205.221, Time=0.22 sec
 ARIMA(0,0,1)(1,0,2)[12] intercept   : AIC=-2200.315, Time=0.45 sec
 ARIMA(0,0,0)(0,0,2)[12] intercept   : AIC=-2110.345, Time=0.30 sec
 ARIMA(1,0,1)(0,0,2)[12] intercept   : AIC=-2217.491, Time=1.13 sec
 ARIMA(1,0,1)(0,0,1)[12] intercept   : AIC=-2191.779, Time=0.37 sec
 ARIMA(1,0,1)(1,0,2)[12] intercept   : AIC=-2214.227, Time=1.34 sec
 ARIMA(1,0,0)(0,0,2)[12] intercept   : AIC=-2153.584, Time=0.35 sec
 ARIMA(2,0,1)(0,0,2)[12] intercept   : AIC=-2261.205, Time=0.45 sec
 ARIMA(2,0,1)(0,0,1)[12] intercept   : AIC=-2246.145, Time=0.14 sec
 ARIMA(2,0,1)(1,0,2)[12] intercept   : AIC=-2268.264, Time=1.24 sec
 ARIMA(2,0,1)(1,0,1)[12] intercept   : AIC=-2241.346, Time=0.36 sec
 ARIMA(2,0,1)(2,0,2)[12] intercept   : AIC=-2263.521, Time=1.23 sec
 ARIMA(2,0,1)(2,0,1)[12] intercept   : AIC=-2238.322, Time=0.74 sec
 ARIMA(2,0,0)(1,0,2)[12] intercept   : AIC=-2263.065, Time=0.90 sec
 ARIMA(3,0,1)(1,0,2)[12] intercept   : AIC=-2252.585, Time=0.43 sec
 ARIMA(2,0,2)(1,0,2)[12] intercept   : AIC=-2257.868, Time=0.98 sec
 ARIMA(1,0,0)(1,0,2)[12] intercept   : AIC=-2154.269, Time=0.57 sec
 ARIMA(1,0,2)(1,0,2)[12] intercept   : AIC=-2204.390, Time=0.95 sec
 ARIMA(3,0,0)(1,0,2)[12] intercept   : AIC=-2256.545, Time=0.55 sec
 ARIMA(3,0,2)(1,0,2)[12] intercept   : AIC=-2255.203, Time=1.52 sec
 ARIMA(2,0,1)(1,0,2)[12]             : AIC=-2262.228, Time=0.49 sec

Best model:  ARIMA(2,0,1)(1,0,2)[12] intercept
Total fit time: 15.608 seconds
```

Out[89]:

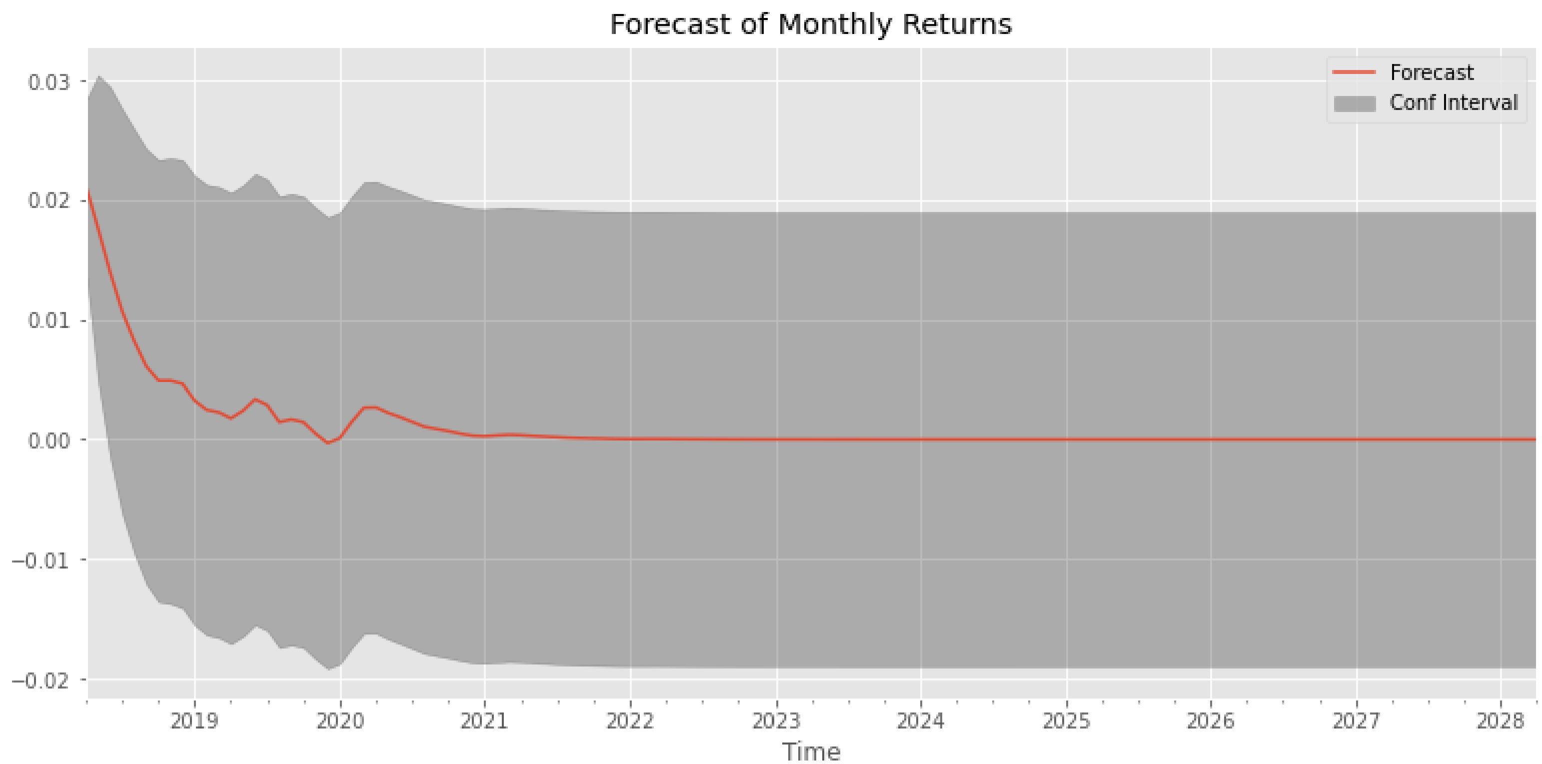
```
ARIMA(maxiter=50, method='lbfgs', order=(2, 0, 1), out_of_sample_size=0,
      scoring='mse', scoring_args={}, seasonal_order=(1, 0, 2, 12),
```

```
        start_params=None, suppress_warnings=True, trend=None,
        with_intercept=True)
```

In [90]:

```
pdq = (2, 0, 1)
pdqs = (1, 0, 2, 12)
ret_92101 = forecast_model(TS_92101,pdq=pdq,pdqs=pdqs,zc=92101)
```

executed in 919ms, finished 18:59:42 2021-08-07

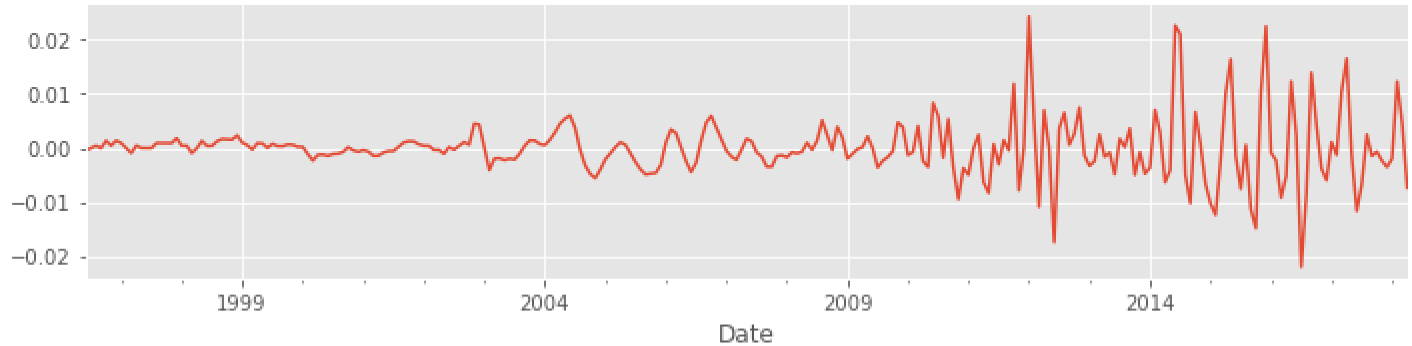


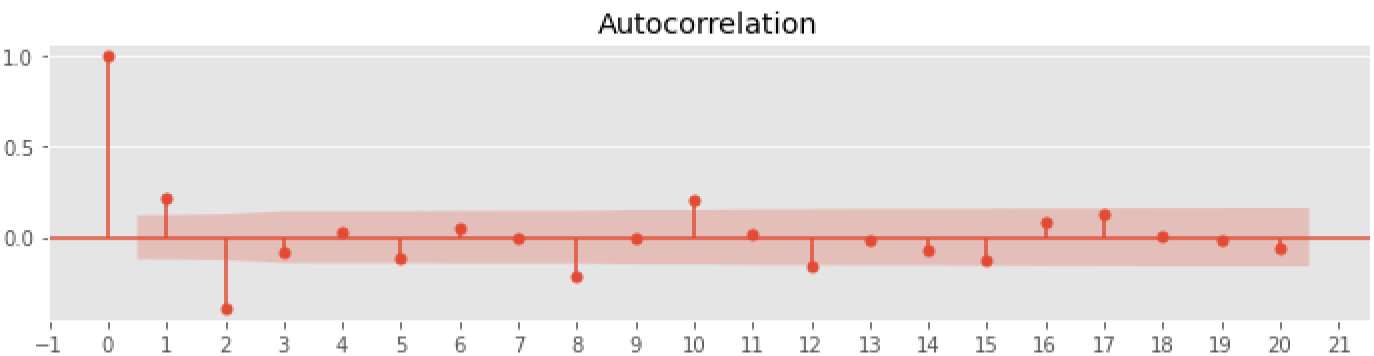

```
Total expected return in 1 year: 10.47%
Total expected return in 3 years: 14.06%
Total expected return in 5 year: 14.27%
Total expected return in 10 years: 14.27%
```

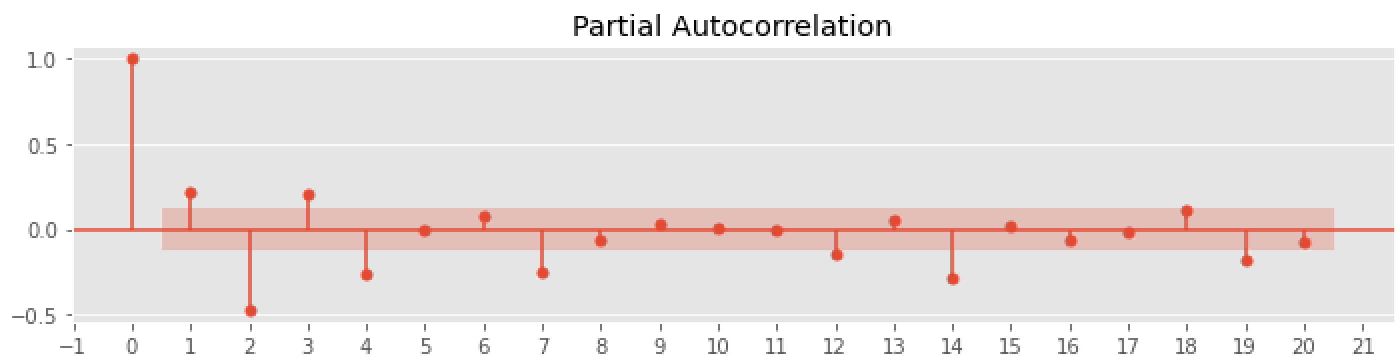
11 Zipcode 95441: Sonoma county

In [91]:
```
plot_acf_pacf(TS_95441d, lags=20)
```
executed in 455ms, finished 18:59:42 2021-08-07

Out[91]:
```
(<Figure size 720x576 with 3 Axes>,
 array([<AxesSubplot:xlabel='Date'>,
        <AxesSubplot:title={'center':'Autocorrelation'}>,
        <AxesSubplot:title={'center':'Partial Autocorrelation'}>],
       dtype=object))
```







The ACF and PACF have just one very strong correlation, right at 2 month.

In [92]:

```
results = pm.auto_arima(TS_95441d,information_criterion='aic',m=12,d=0,
                        start_p=1,start_q=1, max_p=3, max_q=3,
                        stepwise=True,trace=True,error_action='ignore',suppress_warnings=True)
results
```

executed in 31.8s, finished 19:00:14 2021-08-07

```
Performing stepwise search to minimize aic
 ARIMA(1,0,1)(1,0,1)[12] intercept   : AIC=inf, Time=0.60 sec
 ARIMA(0,0,0)(0,0,0)[12] intercept   : AIC=-1998.357, Time=0.03 sec
 ARIMA(1,0,0)(1,0,0)[12] intercept   : AIC=-2016.057, Time=0.19 sec
 ARIMA(0,0,1)(0,0,1)[12] intercept   : AIC=inf, Time=0.38 sec
 ARIMA(0,0,0)(0,0,0)[12]             : AIC=-2000.346, Time=0.02 sec
 ARIMA(1,0,0)(0,0,0)[12] intercept   : AIC=-2009.582, Time=0.06 sec
 ARIMA(1,0,0)(2,0,0)[12] intercept   : AIC=-2031.396, Time=0.61 sec
 ARIMA(1,0,0)(2,0,1)[12] intercept   : AIC=inf, Time=1.06 sec
 ARIMA(1,0,0)(1,0,1)[12] intercept   : AIC=inf, Time=0.48 sec
 ARIMA(0,0,0)(2,0,0)[12] intercept   : AIC=-2016.292, Time=0.61 sec
 ARIMA(2,0,0)(2,0,0)[12] intercept   : AIC=-2093.136, Time=1.07 sec
 ARIMA(2,0,0)(1,0,0)[12] intercept   : AIC=-2080.740, Time=0.17 sec
 ARIMA(2,0,0)(2,0,1)[12] intercept   : AIC=-2099.042, Time=0.49 sec
 ARIMA(2,0,0)(1,0,1)[12] intercept   : AIC=-2096.725, Time=0.31 sec
 ARIMA(2,0,0)(2,0,2)[12] intercept   : AIC=-2127.348, Time=1.22 sec
 ARIMA(2,0,0)(1,0,2)[12] intercept   : AIC=inf, Time=1.24 sec
 ARIMA(1,0,0)(2,0,2)[12] intercept   : AIC=-2070.294, Time=1.04 sec
 ARIMA(3,0,0)(2,0,2)[12] intercept   : AIC=-2131.429, Time=1.77 sec
 ARIMA(3,0,0)(1,0,2)[12] intercept   : AIC=inf, Time=0.84 sec
 ARIMA(3,0,0)(2,0,1)[12] intercept   : AIC=-2127.555, Time=1.39 sec
 ARIMA(3,0,0)(1,0,1)[12] intercept   : AIC=-2102.450, Time=0.33 sec
 ARIMA(3,0,1)(2,0,2)[12] intercept   : AIC=-2133.370, Time=1.09 sec
 ARIMA(3,0,1)(1,0,2)[12] intercept   : AIC=inf, Time=0.60 sec
 ARIMA(3,0,1)(2,0,1)[12] intercept   : AIC=-2109.813, Time=0.72 sec
 ARIMA(3,0,1)(1,0,1)[12] intercept   : AIC=-2105.916, Time=0.32 sec
 ARIMA(2,0,1)(2,0,2)[12] intercept   : AIC=-2122.426, Time=0.54 sec
 ARIMA(3,0,2)(2,0,2)[12] intercept   : AIC=-2122.903, Time=1.45 sec
 ARIMA(2,0,2)(2,0,2)[12] intercept   : AIC=-2140.077, Time=1.45 sec
 ARIMA(2,0,2)(1,0,2)[12] intercept   : AIC=inf, Time=0.98 sec
 ARIMA(2,0,2)(2,0,1)[12] intercept   : AIC=-2114.903, Time=0.65 sec
 ARIMA(2,0,2)(1,0,1)[12] intercept   : AIC=-2109.497, Time=0.28 sec
 ARIMA(1,0,2)(2,0,2)[12] intercept   : AIC=-2126.662, Time=1.61 sec
 ARIMA(2,0,3)(2,0,2)[12] intercept   : AIC=-2003.898, Time=1.44 sec
 ARIMA(1,0,1)(2,0,2)[12] intercept   : AIC=inf, Time=1.31 sec
 ARIMA(1,0,3)(2,0,2)[12] intercept   : AIC=-2108.822, Time=1.49 sec
```

```
 ARIMA(3,0,3)(2,0,2)[12] intercept   : AIC=inf, Time=2.18 sec
 ARIMA(2,0,2)(2,0,2)[12]             : AIC=inf, Time=1.62 sec

Best model:  ARIMA(2,0,2)(2,0,2)[12] intercept
Total fit time: 31.699 seconds
```

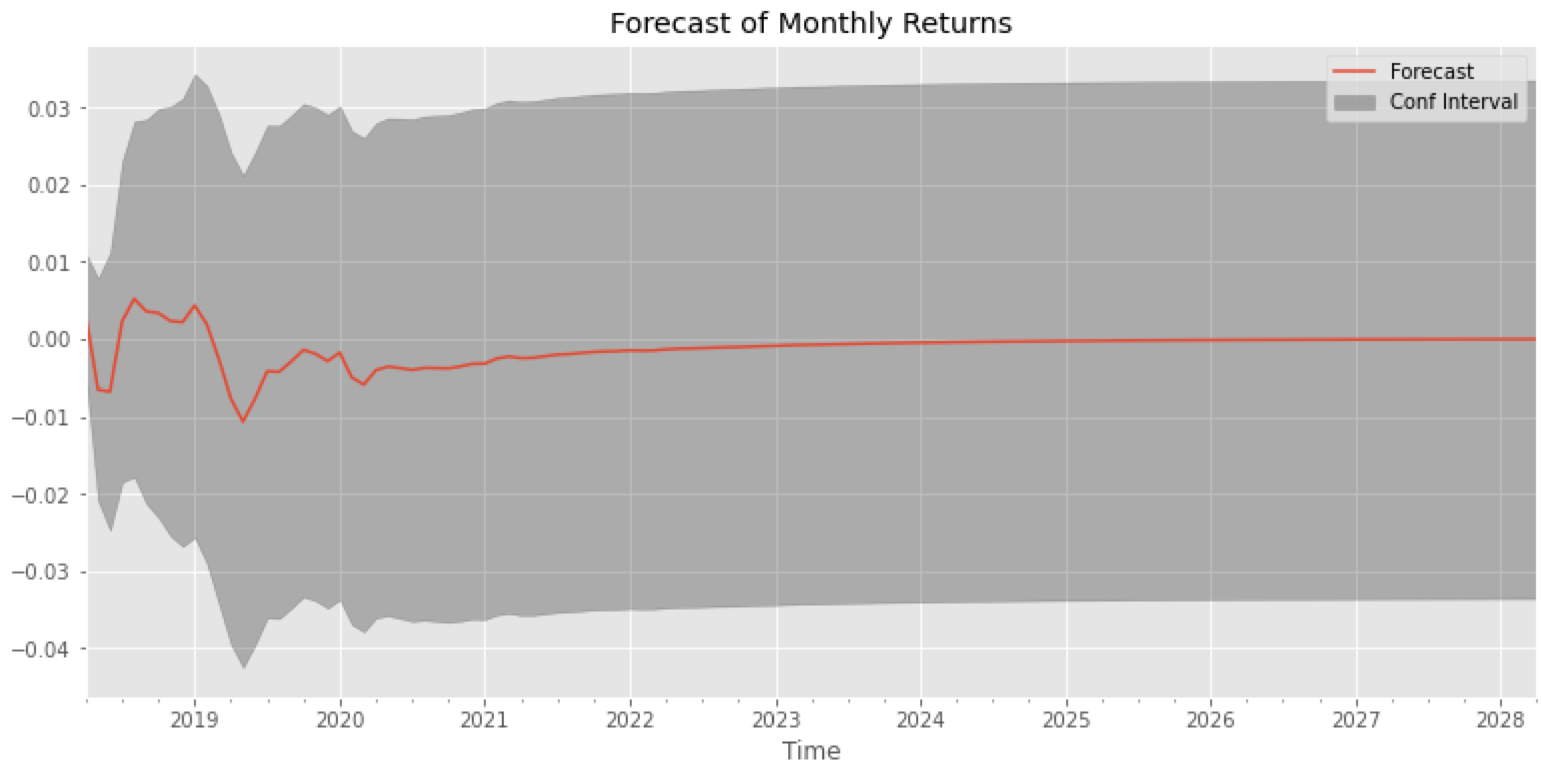
Out[92]:

```
ARIMA(maxiter=50, method='lbfgs', order=(2, 0, 2), out_of_sample_size=0,
      scoring='mse', scoring_args={}, seasonal_order=(2, 0, 2, 12),
      start_params=None, suppress_warnings=True, trend=None,
      with_intercept=True)
```

In [93]:

```
pdq = (2, 0, 2)
pdqs = (2, 0, 2, 12)
ret_93405 = forecast_model(TS_95441,pdq=pdq,pdqs=pdqs,zc=93405)
```

executed in 1.65s, finished 19:00:16 2021-08-07

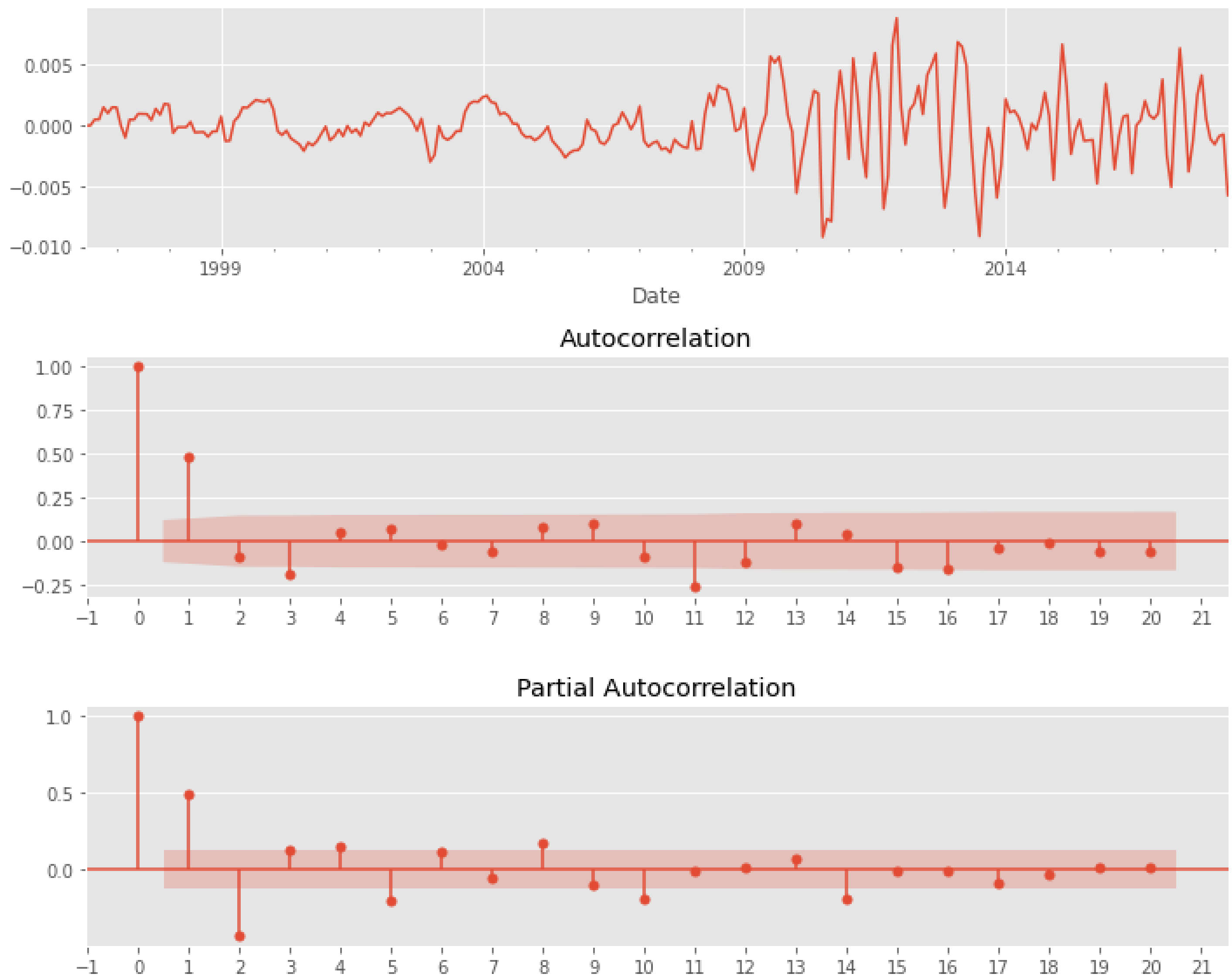



```
Total expected return in 1 year: 1.22%
Total expected return in 3 years: -8.26%
Total expected return in 5 year: -11.45%
Total expected return in 10 years: -12.82%
```

12 Zipcode 94546: San Francisco

In [94]:
```
plot_acf_pacf(TS_94546d,lags=20)
```
executed in 469ms, finished 19:00:16 2021-08-07

Out[94]:
```
(<Figure size 720x576 with 3 Axes>,
 array([<AxesSubplot:xlabel='Date'>,
        <AxesSubplot:title={'center':'Autocorrelation'}>,
        <AxesSubplot:title={'center':'Partial Autocorrelation'}>],
       dtype=object))
```



In [95]:

```
results = pm.auto_arima(TS_94546d,information_criterion='aic',m=12,d=0,
                        start_p=1,start_q=1, max_p=3, max_q=3,
                        stepwise=True,trace=True,error_action='ignore',suppress_warnings=True)
results
```

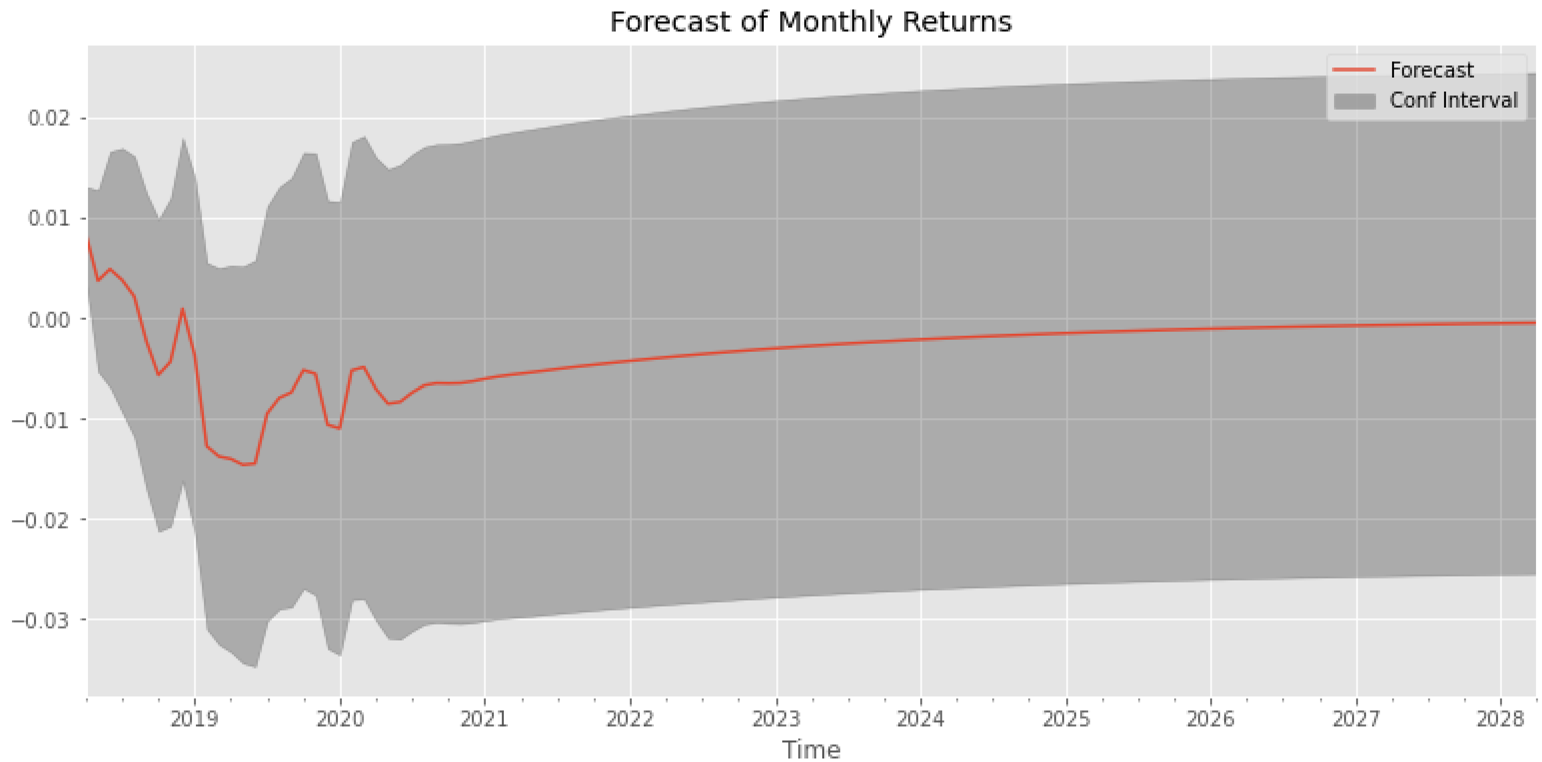
executed in 19.3s, finished 19:00:36 2021-08-07

```
Performing stepwise search to minimize aic
 ARIMA(1,0,1)(1,0,1)[12] intercept   : AIC=-2482.597, Time=0.58 sec
 ARIMA(0,0,0)(0,0,0)[12] intercept   : AIC=-2376.105, Time=0.04 sec
 ARIMA(1,0,0)(1,0,0)[12] intercept   : AIC=-2447.279, Time=0.37 sec
 ARIMA(0,0,1)(0,0,1)[12] intercept   : AIC=-2409.437, Time=0.22 sec
 ARIMA(0,0,0)(0,0,0)[12]             : AIC=-2378.078, Time=0.02 sec
 ARIMA(1,0,1)(0,0,1)[12] intercept   : AIC=-2482.016, Time=0.51 sec
 ARIMA(1,0,1)(1,0,0)[12] intercept   : AIC=-2482.146, Time=0.56 sec
 ARIMA(1,0,1)(2,0,1)[12] intercept   : AIC=-2475.892, Time=1.42 sec
 ARIMA(1,0,1)(1,0,2)[12] intercept   : AIC=-2441.143, Time=0.85 sec
 ARIMA(1,0,1)(0,0,0)[12] intercept   : AIC=-2446.143, Time=0.05 sec
 ARIMA(1,0,1)(0,0,2)[12] intercept   : AIC=-2444.345, Time=0.74 sec
 ARIMA(1,0,1)(2,0,0)[12] intercept   : AIC=-2443.174, Time=0.51 sec
 ARIMA(1,0,1)(2,0,2)[12] intercept   : AIC=-2487.206, Time=1.18 sec
 ARIMA(0,0,1)(2,0,2)[12] intercept   : AIC=-2398.444, Time=0.28 sec
 ARIMA(1,0,0)(2,0,2)[12] intercept   : AIC=-2446.544, Time=0.31 sec
 ARIMA(2,0,1)(2,0,2)[12] intercept   : AIC=-2485.189, Time=0.61 sec
 ARIMA(1,0,2)(2,0,2)[12] intercept   : AIC=-2473.136, Time=0.71 sec
 ARIMA(0,0,0)(2,0,2)[12] intercept   : AIC=-2376.378, Time=0.71 sec
 ARIMA(0,0,2)(2,0,2)[12] intercept   : AIC=-2463.147, Time=0.75 sec
 ARIMA(2,0,0)(2,0,2)[12] intercept   : AIC=-2491.792, Time=0.71 sec
 ARIMA(2,0,0)(1,0,2)[12] intercept   : AIC=-2502.490, Time=0.50 sec
 ARIMA(2,0,0)(0,0,2)[12] intercept   : AIC=-2505.312, Time=0.30 sec
 ARIMA(2,0,0)(0,0,1)[12] intercept   : AIC=-2499.512, Time=0.25 sec
 ARIMA(2,0,0)(1,0,1)[12] intercept   : AIC=-2497.163, Time=0.33 sec
 ARIMA(1,0,0)(0,0,2)[12] intercept   : AIC=-2459.869, Time=0.76 sec
 ARIMA(3,0,0)(0,0,2)[12] intercept   : AIC=-2505.553, Time=0.51 sec
 ARIMA(3,0,0)(0,0,1)[12] intercept   : AIC=-2500.325, Time=0.49 sec
 ARIMA(3,0,0)(1,0,2)[12] intercept   : AIC=-2502.683, Time=0.95 sec
 ARIMA(3,0,0)(1,0,1)[12] intercept   : AIC=-2497.987, Time=0.50 sec
 ARIMA(3,0,1)(0,0,2)[12] intercept   : AIC=-2492.931, Time=0.45 sec
 ARIMA(2,0,1)(0,0,2)[12] intercept   : AIC=-2499.540, Time=0.66 sec
 ARIMA(3,0,0)(0,0,2)[12]             : AIC=-2507.565, Time=0.35 sec
 ARIMA(3,0,0)(0,0,1)[12]             : AIC=-2502.320, Time=0.27 sec
 ARIMA(3,0,0)(1,0,2)[12]             : AIC=-2504.694, Time=0.51 sec
 ARIMA(3,0,0)(1,0,1)[12]             : AIC=-2499.986, Time=0.28 sec
```

```
 ARIMA(2,0,0)(0,0,2)[12]             : AIC=-2507.316, Time=0.26 sec
 ARIMA(3,0,1)(0,0,2)[12]             : AIC=-2494.935, Time=0.43 sec
 ARIMA(2,0,1)(0,0,2)[12]             : AIC=-2501.544, Time=0.30 sec

Best model:  ARIMA(3,0,0)(0,0,2)[12]
Total fit time: 19.266 seconds
```

Out[95]:
```
ARIMA(maxiter=50, method='lbfgs', order=(3, 0, 0), out_of_sample_size=0,
      scoring='mse', scoring_args={}, seasonal_order=(0, 0, 2, 12),
      start_params=None, suppress_warnings=True, trend=None,
      with_intercept=False)
```

In [96]:
```
pdq = (3, 0, 0)
pdqs = (0, 0, 2, 12)
ret_94546 = forecast_model(TS_94546,pdq=pdq,pdqs=pdqs,zc=94546)
```
executed in 1.37s, finished 19:00:37 2021-08-07




```
Total expected return in 1 year: -1.89%
Total expected return in 3 years: -19.09%
Total expected return in 5 year: -26.65%
Total expected return in 10 years: -32.41%
```

13 Zipcode 91754: Los Angeles

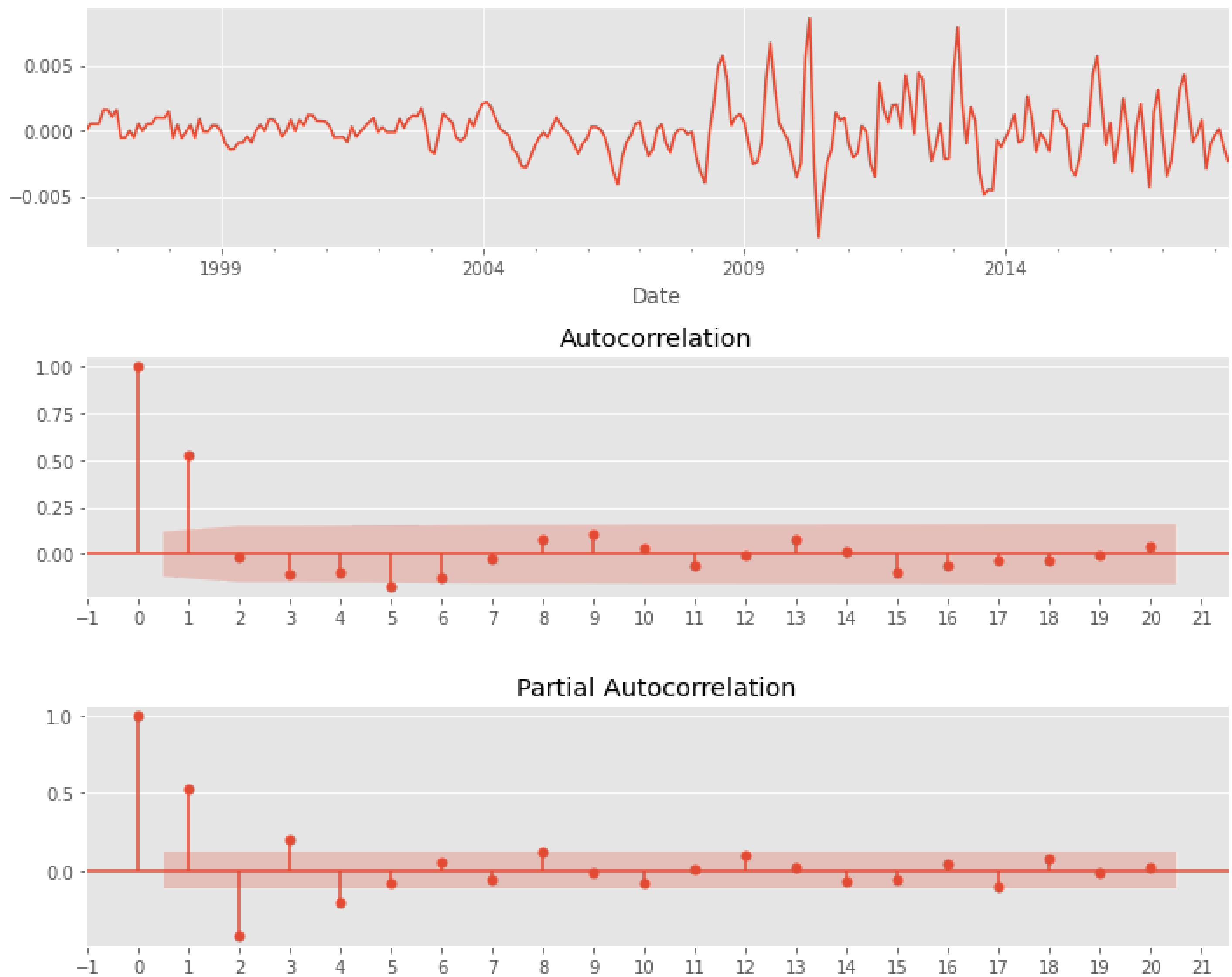
In [97]:

```
plot_acf_pacf(TS_91754d,lags=20)
```

executed in 533ms, finished 19:00:38 2021-08-07

Out[97]:

```
(<Figure size 720x576 with 3 Axes>,
 array([<AxesSubplot:xlabel='Date'>,
        <AxesSubplot:title={'center':'Autocorrelation'}>,
        <AxesSubplot:title={'center':'Partial Autocorrelation'}>],
       dtype=object))
```



The ACF and PACF have just one very strong correlation, right at 1 month.

In [98]:

```
results = pm.auto_arima(TS_91754d,information_criterion='aic',m=12,d=0,
                        start_p=1,start_q=1, max_p=3, max_q=3,
                        stepwise=True,trace=True,error_action='ignore',suppress_warnings=True)
results
```

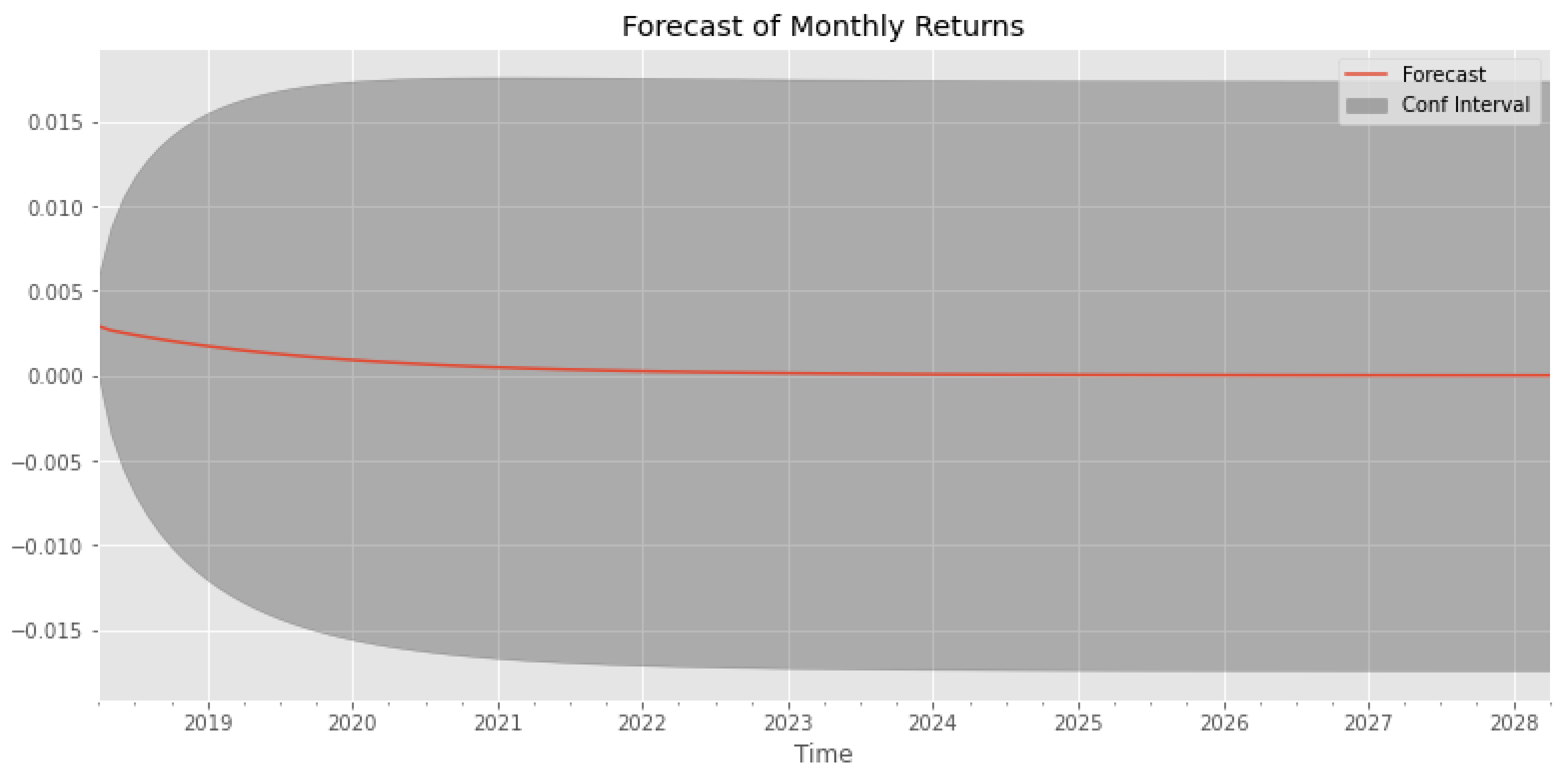
executed in 5.53s, finished 19:00:43 2021-08-07

```
Performing stepwise search to minimize aic
 ARIMA(1,0,1)(1,0,1)[12] intercept   : AIC=-2624.350, Time=0.23 sec
 ARIMA(0,0,0)(0,0,0)[12] intercept   : AIC=-2503.870, Time=0.07 sec
 ARIMA(1,0,0)(1,0,0)[12] intercept   : AIC=-2586.321, Time=0.34 sec
 ARIMA(0,0,1)(0,0,1)[12] intercept   : AIC=-2596.583, Time=0.35 sec
 ARIMA(0,0,0)(0,0,0)[12]             : AIC=-2505.844, Time=0.03 sec
 ARIMA(1,0,1)(0,0,1)[12] intercept   : AIC=-2624.743, Time=0.16 sec
 ARIMA(1,0,1)(0,0,0)[12] intercept   : AIC=-2630.902, Time=0.10 sec
 ARIMA(1,0,1)(1,0,0)[12] intercept   : AIC=-2628.893, Time=0.36 sec
 ARIMA(0,0,1)(0,0,0)[12] intercept   : AIC=-2606.347, Time=0.14 sec
 ARIMA(1,0,0)(0,0,0)[12] intercept   : AIC=-2588.315, Time=0.03 sec
 ARIMA(2,0,1)(0,0,0)[12] intercept   : AIC=-2647.300, Time=0.13 sec
 ARIMA(2,0,1)(1,0,0)[12] intercept   : AIC=-2645.299, Time=0.25 sec
 ARIMA(2,0,1)(0,0,1)[12] intercept   : AIC=-2643.344, Time=0.27 sec
 ARIMA(2,0,1)(1,0,1)[12] intercept   : AIC=-2642.984, Time=0.26 sec
 ARIMA(2,0,0)(0,0,0)[12] intercept   : AIC=-2635.182, Time=0.11 sec
 ARIMA(3,0,1)(0,0,0)[12] intercept   : AIC=-2645.188, Time=0.18 sec
 ARIMA(2,0,2)(0,0,0)[12] intercept   : AIC=-2644.184, Time=0.16 sec
 ARIMA(1,0,2)(0,0,0)[12] intercept   : AIC=-2640.794, Time=0.26 sec
 ARIMA(3,0,0)(0,0,0)[12] intercept   : AIC=-2643.638, Time=0.25 sec
 ARIMA(3,0,2)(0,0,0)[12] intercept   : AIC=-2643.300, Time=0.35 sec
 ARIMA(2,0,1)(0,0,0)[12]             : AIC=-2649.293, Time=0.07 sec
 ARIMA(2,0,1)(1,0,0)[12]             : AIC=-2647.292, Time=0.15 sec
 ARIMA(2,0,1)(0,0,1)[12]             : AIC=-2645.337, Time=0.17 sec
 ARIMA(2,0,1)(1,0,1)[12]             : AIC=-2644.980, Time=0.30 sec
 ARIMA(1,0,1)(0,0,0)[12]             : AIC=-2632.935, Time=0.06 sec
 ARIMA(2,0,0)(0,0,0)[12]             : AIC=-2637.169, Time=0.05 sec
 ARIMA(3,0,1)(0,0,0)[12]             : AIC=-2647.181, Time=0.16 sec
 ARIMA(2,0,2)(0,0,0)[12]             : AIC=-2646.182, Time=0.10 sec
 ARIMA(1,0,0)(0,0,0)[12]             : AIC=-2590.314, Time=0.07 sec
 ARIMA(1,0,2)(0,0,0)[12]             : AIC=-2631.090, Time=0.16 sec
 ARIMA(3,0,0)(0,0,0)[12]             : AIC=-2645.632, Time=0.05 sec
 ARIMA(3,0,2)(0,0,0)[12]             : AIC=-2645.299, Time=0.09 sec

Best model:  ARIMA(2,0,1)(0,0,0)[12]
Total fit time: 5.503 seconds
```

Out[98]:
```
ARIMA(maxiter=50, method='lbfgs', order=(2, 0, 1), out_of_sample_size=0,
      scoring='mse', scoring_args={}, seasonal_order=(0, 0, 0, 12),
      start_params=None, suppress_warnings=True, trend=None,
      with_intercept=False)
```

In [99]:
```
pdq = (2, 0, 1)
pdqs = (0, 0, 0, 12)
ret_91754 = forecast_model(TS_91754,pdq=pdq,pdqs=pdqs,zc=91754)
```

executed in 373ms, finished 19:00:43 2021-08-07




```
Total expected return in 1 year: 2.6%
Total expected return in 3 years: 4.72%
Total expected return in 5 year: 5.32%
Total expected return in 10 years: 5.54%
```

14 Zipcode 92860: Riverside

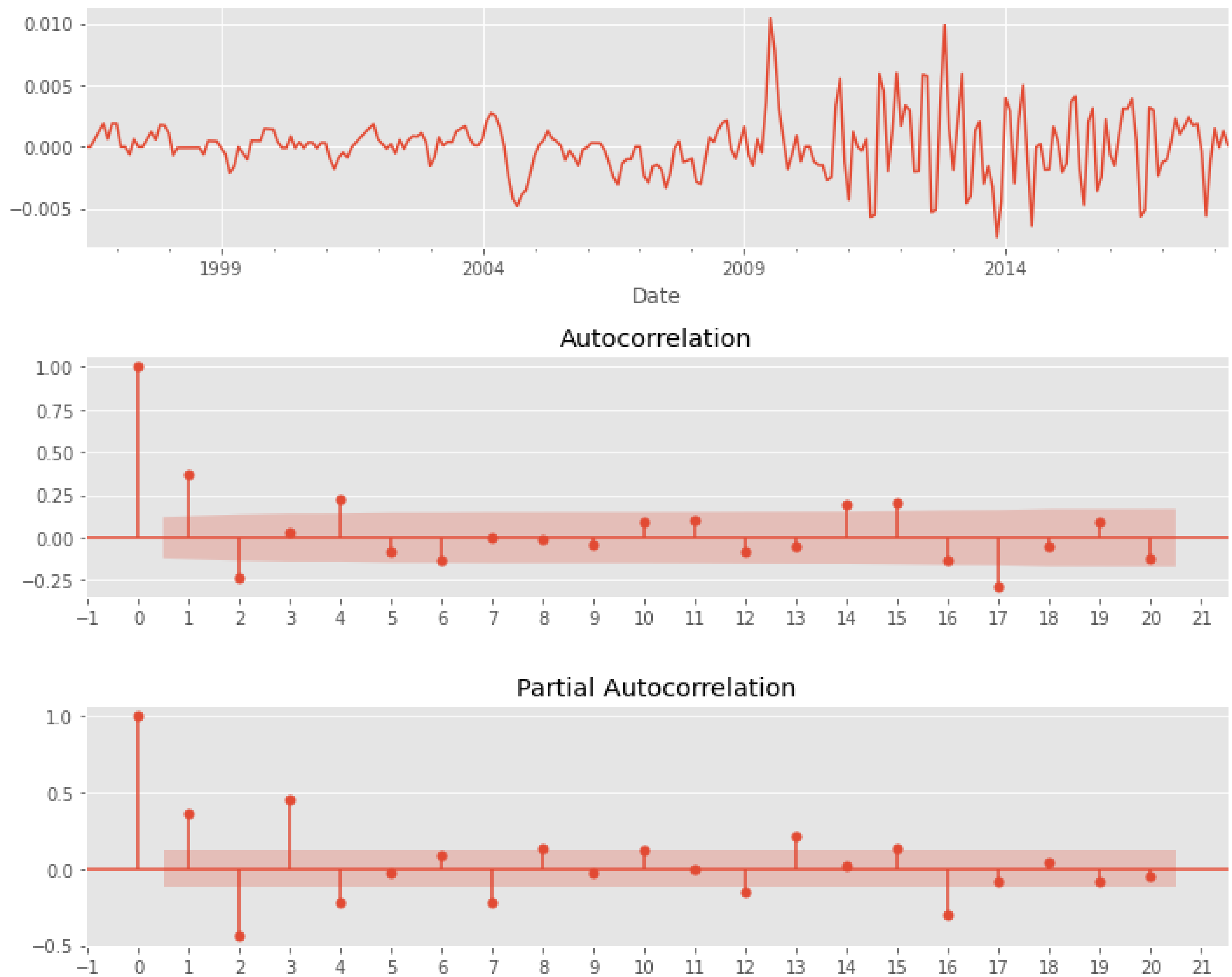
In [100]:

```
plot_acf_pacf(TS_92860d,lags=20)
```

executed in 475ms, finished 19:00:44 2021-08-07

Out[100]:

```
(<Figure size 720x576 with 3 Axes>,
 array([<AxesSubplot:xlabel='Date'>,
        <AxesSubplot:title={'center':'Autocorrelation'}>,
        <AxesSubplot:title={'center':'Partial Autocorrelation'}>],
       dtype=object))
```



In [101]:

```
results = pm.auto_arima(TS_92860d,information_criterion='aic',m=12,d=0,
                        start_p=1,start_q=1, max_p=3, max_q=3,
                        stepwise=True,trace=True,error_action='ignore',suppress_warnings=True)
results
```

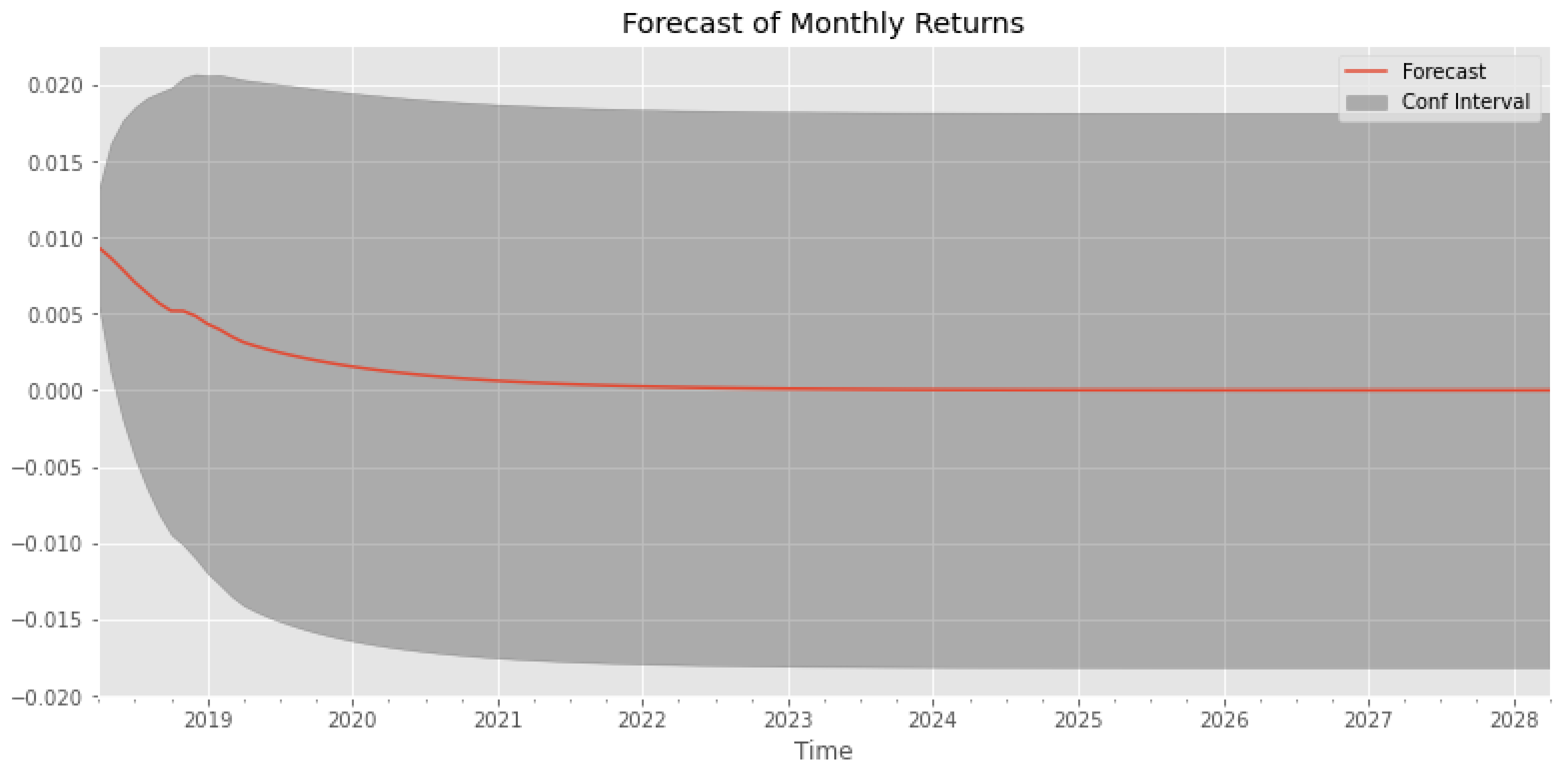
executed in 22.3s, finished 19:01:06 2021-08-07

```
Performing stepwise search to minimize aic
 ARIMA(1,0,1)(1,0,1)[12] intercept   : AIC=-2520.124, Time=0.62 sec
 ARIMA(0,0,0)(0,0,0)[12] intercept   : AIC=-2415.055, Time=0.05 sec
 ARIMA(1,0,0)(1,0,0)[12] intercept   : AIC=-2452.756, Time=0.19 sec
 ARIMA(0,0,1)(0,0,1)[12] intercept   : AIC=-2528.244, Time=0.26 sec
 ARIMA(0,0,0)(0,0,0)[12]             : AIC=-2416.942, Time=0.03 sec
 ARIMA(0,0,1)(0,0,0)[12] intercept   : AIC=-2522.581, Time=0.06 sec
 ARIMA(0,0,1)(1,0,1)[12] intercept   : AIC=-2521.375, Time=0.17 sec
 ARIMA(0,0,1)(0,0,2)[12] intercept   : AIC=inf, Time=1.19 sec
 ARIMA(0,0,1)(1,0,0)[12] intercept   : AIC=-2524.445, Time=0.15 sec
 ARIMA(0,0,1)(1,0,2)[12] intercept   : AIC=-2543.881, Time=0.87 sec
 ARIMA(0,0,1)(2,0,2)[12] intercept   : AIC=-2532.789, Time=0.82 sec
 ARIMA(0,0,1)(2,0,1)[12] intercept   : AIC=-2532.759, Time=0.87 sec
 ARIMA(0,0,0)(1,0,2)[12] intercept   : AIC=-2404.598, Time=0.27 sec
 ARIMA(1,0,1)(1,0,2)[12] intercept   : AIC=-2539.967, Time=1.18 sec
 ARIMA(0,0,2)(1,0,2)[12] intercept   : AIC=inf, Time=1.30 sec
 ARIMA(1,0,0)(1,0,2)[12] intercept   : AIC=-2449.157, Time=0.40 sec
 ARIMA(1,0,2)(1,0,2)[12] intercept   : AIC=-2556.047, Time=1.26 sec
 ARIMA(1,0,2)(0,0,2)[12] intercept   : AIC=-2506.804, Time=1.11 sec
 ARIMA(1,0,2)(1,0,1)[12] intercept   : AIC=-2472.873, Time=0.49 sec
 ARIMA(1,0,2)(2,0,2)[12] intercept   : AIC=-2483.960, Time=1.45 sec
 ARIMA(1,0,2)(0,0,1)[12] intercept   : AIC=-2475.245, Time=0.41 sec
 ARIMA(1,0,2)(2,0,1)[12] intercept   : AIC=-2489.860, Time=1.25 sec
 ARIMA(2,0,2)(1,0,2)[12] intercept   : AIC=-2579.605, Time=0.73 sec
 ARIMA(2,0,2)(0,0,2)[12] intercept   : AIC=-2586.768, Time=0.41 sec
 ARIMA(2,0,2)(0,0,1)[12] intercept   : AIC=-2592.059, Time=0.85 sec
 ARIMA(2,0,2)(0,0,0)[12] intercept   : AIC=-2582.152, Time=0.19 sec
 ARIMA(2,0,2)(1,0,1)[12] intercept   : AIC=-2586.835, Time=0.49 sec
 ARIMA(2,0,2)(1,0,0)[12] intercept   : AIC=-2590.851, Time=0.62 sec
 ARIMA(2,0,1)(0,0,1)[12] intercept   : AIC=-2545.172, Time=0.62 sec
 ARIMA(3,0,2)(0,0,1)[12] intercept   : AIC=-2576.965, Time=0.83 sec
 ARIMA(2,0,3)(0,0,1)[12] intercept   : AIC=-2589.571, Time=0.68 sec
 ARIMA(1,0,1)(0,0,1)[12] intercept   : AIC=-2526.598, Time=0.52 sec
 ARIMA(1,0,3)(0,0,1)[12] intercept   : AIC=-2584.451, Time=0.60 sec
 ARIMA(3,0,1)(0,0,1)[12] intercept   : AIC=-2563.541, Time=0.63 sec
 ARIMA(3,0,3)(0,0,1)[12] intercept   : AIC=-2578.219, Time=0.34 sec
```

```
 ARIMA(2,0,2)(0,0,1)[12]             : AIC=-2585.189, Time=0.32 sec

Best model:  ARIMA(2,0,2)(0,0,1)[12] intercept
Total fit time: 22.219 seconds
```

Out[101]:
```
ARIMA(maxiter=50, method='lbfgs', order=(2, 0, 2), out_of_sample_size=0,
      scoring='mse', scoring_args={}, seasonal_order=(0, 0, 1, 12),
      start_params=None, suppress_warnings=True, trend=None,
      with_intercept=True)
```

In [102]:
```
pdq = (2, 0, 2)
pdqs = (0, 0, 1, 12)
ret_92860 = forecast_model(TS_92860,pdq=pdq,pdqs=pdqs,zc=92860)
```
executed in 763ms, finished 19:01:07 2021-08-07




```
Total expected return in 1 year: 7.43%
Total expected return in 3 years: 11.23%
Total expected return in 5 year: 11.82%
Total expected return in 10 years: 11.93%
```

15 Zipcode 95818: Sacramento

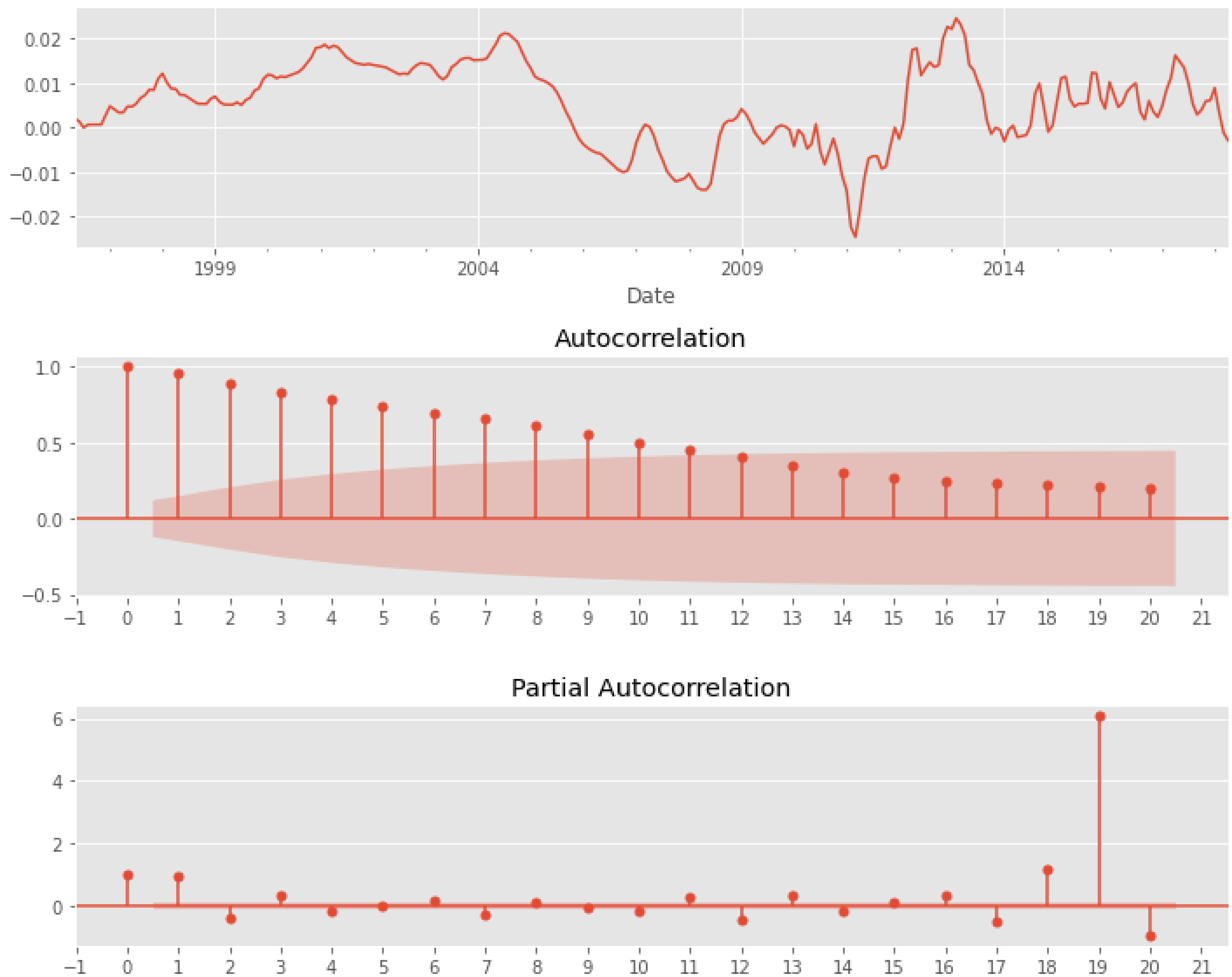
In [103]:

```
plot_acf_pacf(TS_95818,lags=20)
```

executed in 467ms, finished 19:01:07 2021-08-07

Out[103]:

```
(<Figure size 720x576 with 3 Axes>,
 array([<AxesSubplot:xlabel='Date'>,
        <AxesSubplot:title={'center':'Autocorrelation'}>,
        <AxesSubplot:title={'center':'Partial Autocorrelation'}>],
       dtype=object))
```



In [104]:

```
results = pm.auto_arima(TS_95818,information_criterion='aic',m=12,d=0,
                        start_p=1,start_q=1, max_p=3, max_q=3,
                        stepwise=True,trace=True,error_action='ignore',suppress_warnings=True)
results
```

executed in 17.9s, finished 19:01:25 2021-08-07

```
Performing stepwise search to minimize aic
 ARIMA(1,0,1)(1,0,1)[12] intercept   : AIC=-2464.527, Time=0.52 sec
 ARIMA(0,0,0)(0,0,0)[12] intercept   : AIC=-1726.900, Time=0.11 sec
 ARIMA(1,0,0)(1,0,0)[12] intercept   : AIC=-2337.912, Time=0.19 sec
 ARIMA(0,0,1)(0,0,1)[12] intercept   : AIC=-2077.241, Time=0.37 sec
 ARIMA(0,0,0)(0,0,0)[12]             : AIC=-1653.966, Time=0.02 sec
 ARIMA(1,0,1)(0,0,1)[12] intercept   : AIC=-2470.576, Time=0.17 sec
 ARIMA(1,0,1)(0,0,0)[12] intercept   : AIC=-2472.576, Time=0.09 sec
 ARIMA(1,0,1)(1,0,0)[12] intercept   : AIC=-2397.204, Time=0.21 sec
 ARIMA(0,0,1)(0,0,0)[12] intercept   : AIC=-2037.676, Time=0.11 sec
 ARIMA(1,0,0)(0,0,0)[12] intercept   : AIC=-2409.108, Time=0.06 sec
 ARIMA(2,0,1)(0,0,0)[12] intercept   : AIC=-2470.579, Time=0.25 sec
 ARIMA(1,0,2)(0,0,0)[12] intercept   : AIC=-2467.663, Time=0.23 sec
 ARIMA(0,0,2)(0,0,0)[12] intercept   : AIC=-2280.463, Time=0.29 sec
 ARIMA(2,0,0)(0,0,0)[12] intercept   : AIC=-2452.945, Time=0.04 sec
 ARIMA(2,0,2)(0,0,0)[12] intercept   : AIC=-2474.195, Time=0.25 sec
 ARIMA(2,0,2)(1,0,0)[12] intercept   : AIC=-2474.717, Time=0.59 sec
 ARIMA(2,0,2)(2,0,0)[12] intercept   : AIC=-2477.959, Time=1.71 sec
 ARIMA(2,0,2)(2,0,1)[12] intercept   : AIC=-2477.458, Time=1.51 sec
 ARIMA(2,0,2)(1,0,1)[12] intercept   : AIC=-2448.633, Time=0.83 sec
 ARIMA(1,0,2)(2,0,0)[12] intercept   : AIC=-2390.838, Time=0.38 sec
 ARIMA(2,0,1)(2,0,0)[12] intercept   : AIC=-2465.074, Time=1.20 sec
 ARIMA(3,0,2)(2,0,0)[12] intercept   : AIC=-2496.718, Time=1.41 sec
 ARIMA(3,0,2)(1,0,0)[12] intercept   : AIC=-2411.542, Time=0.37 sec
 ARIMA(3,0,2)(2,0,1)[12] intercept   : AIC=-2481.733, Time=1.58 sec
 ARIMA(3,0,2)(1,0,1)[12] intercept   : AIC=-2491.712, Time=0.79 sec
 ARIMA(3,0,1)(2,0,0)[12] intercept   : AIC=-2468.280, Time=1.28 sec
 ARIMA(3,0,3)(2,0,0)[12] intercept   : AIC=-2478.847, Time=1.47 sec
 ARIMA(2,0,3)(2,0,0)[12] intercept   : AIC=-2491.720, Time=1.54 sec
 ARIMA(3,0,2)(2,0,0)[12]             : AIC=-2415.327, Time=0.30 sec

Best model:  ARIMA(3,0,2)(2,0,0)[12] intercept
Total fit time: 17.882 seconds
```

Out[104]:

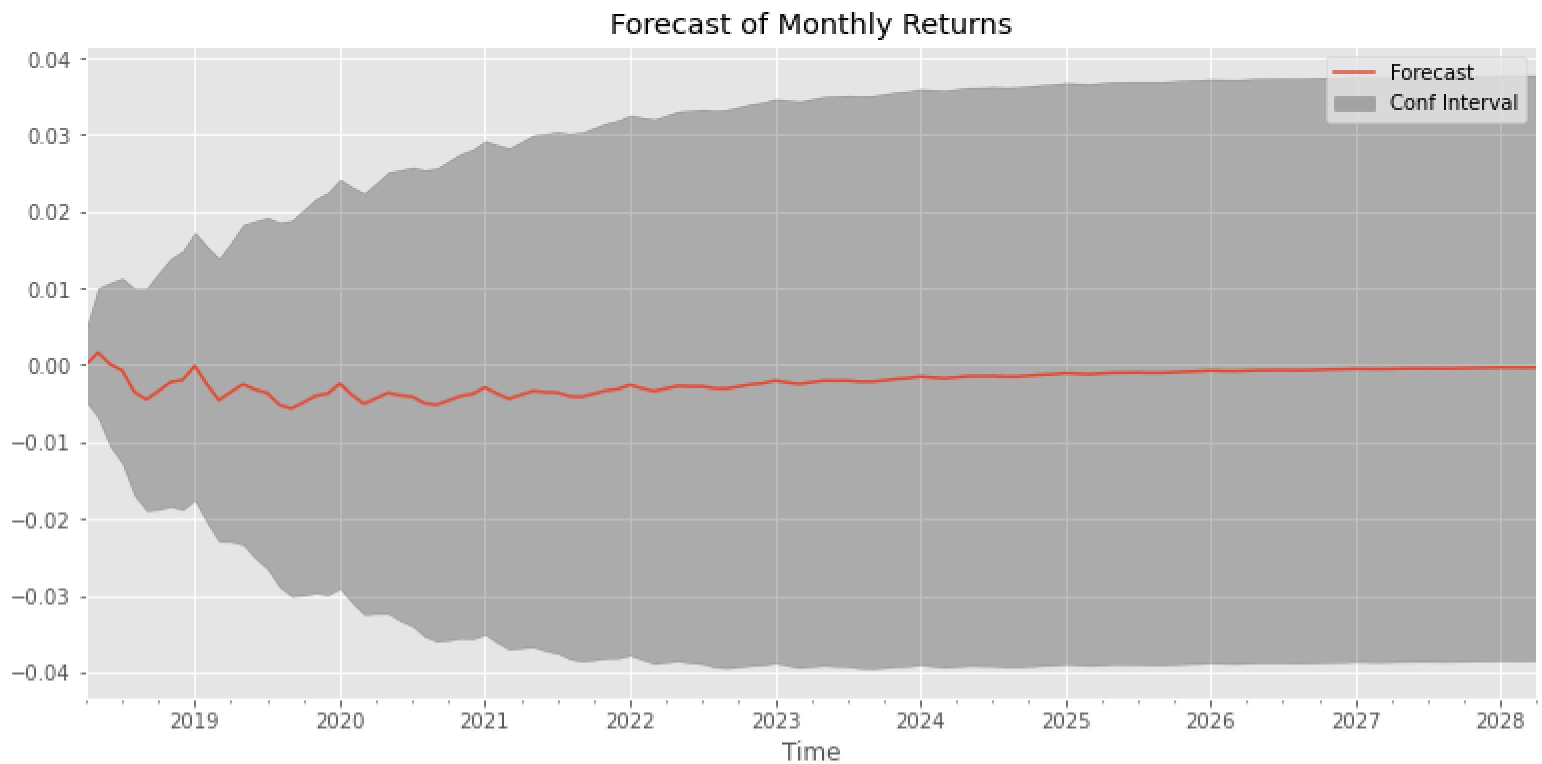
```
ARIMA(maxiter=50, method='lbfgs', order=(3, 0, 2), out_of_sample_size=0,
      scoring='mse', scoring_args={}, seasonal_order=(2, 0, 0, 12),
```

```
        start_params=None, suppress_warnings=True, trend=None,
        with_intercept=True)
```

In [105]:

```
pdq = (3, 0, 2)
pdqs = (2, 0, 0, 12)
ret_95818 = forecast_model(TS_95818,pdq=pdq,pdqs=pdqs,zc=95818)
```

executed in 652ms, finished 19:01:26 2021-08-07




```
Total expected return in 1 year: -2.13%
Total expected return in 3 years: -11.14%
Total expected return in 5 year: -17.44%
Total expected return in 10 years: -22.51%
```

16 Zipcode 93003: Ventura

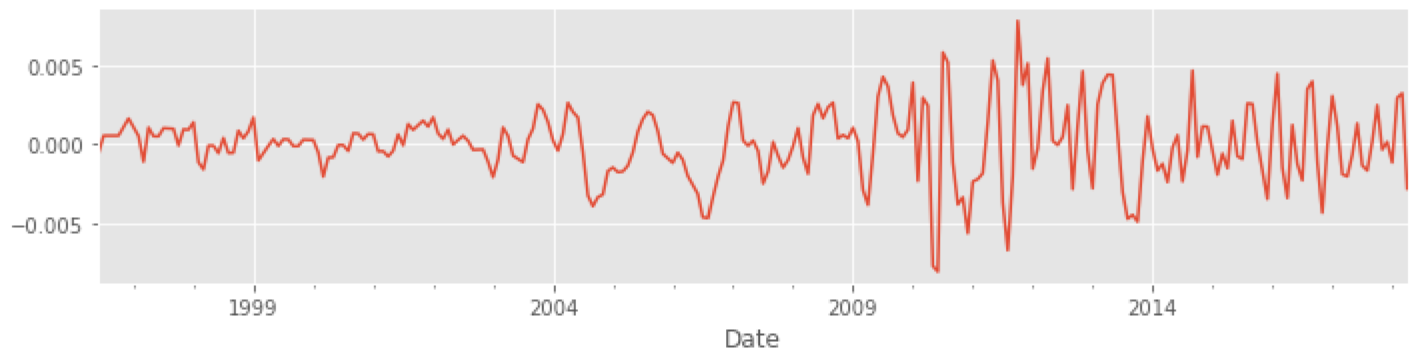
In [106]:

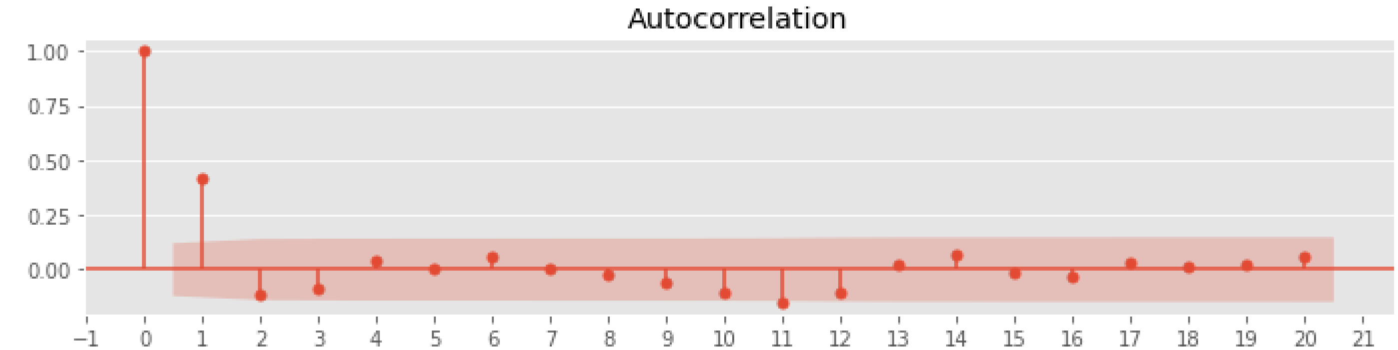
```
plot_acf_pacf(TS_93003d,lags=20)
```

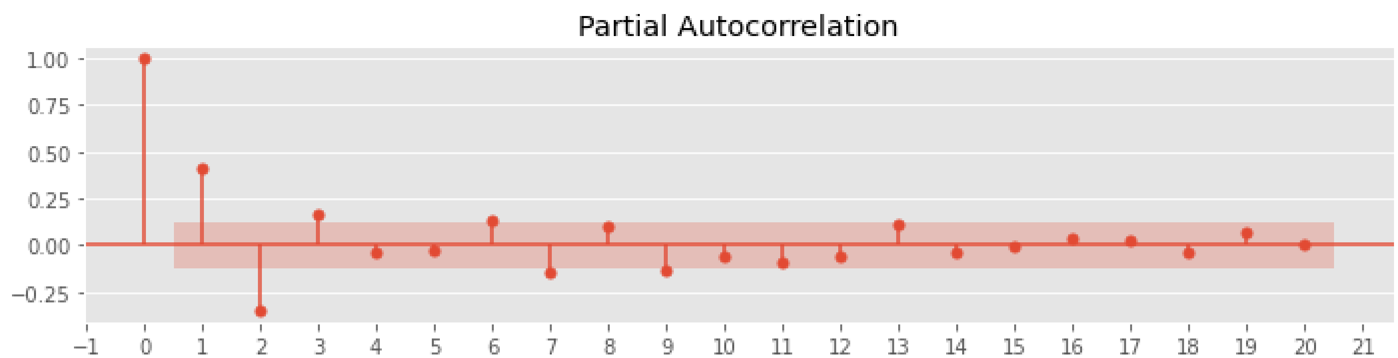
executed in 474ms, finished 19:01:26 2021-08-07

Out[106]:

```
(<Figure size 720x576 with 3 Axes>,
 array([<AxesSubplot:xlabel='Date'>,
        <AxesSubplot:title={'center':'Autocorrelation'}>,
        <AxesSubplot:title={'center':'Partial Autocorrelation'}>],
       dtype=object))
```


0.005
0.000
−0.005
1999
2004
2009
2014
Date


Autocorrelation
1.00
0.75
0.50
0.25
0.00
−1 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21


Partial Autocorrelation
1.00
0.75
0.50
0.25
0.00
−0.25
−1 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21

In [107]:

```
results = pm.auto_arima(TS_93003d,information_criterion='aic',m=12,d=0,
                        start_p=1,start_q=1, max_p=3, max_q=3,
                        stepwise=True,trace=True,error_action='ignore',suppress_warnings=True)
results
```

executed in 2.68s, finished 19:01:29 2021-08-07

```
Performing stepwise search to minimize aic
 ARIMA(1,0,1)(1,0,1)[12] intercept   : AIC=-2504.307, Time=0.48 sec
 ARIMA(0,0,0)(0,0,0)[12] intercept   : AIC=-2453.717, Time=0.04 sec
 ARIMA(1,0,0)(1,0,0)[12] intercept   : AIC=-2500.996, Time=0.17 sec
 ARIMA(0,0,1)(0,0,1)[12] intercept   : AIC=-2528.199, Time=0.14 sec
 ARIMA(0,0,0)(0,0,0)[12]             : AIC=-2455.686, Time=0.03 sec
 ARIMA(0,0,1)(0,0,0)[12] intercept   : AIC=-2530.655, Time=0.07 sec
 ARIMA(0,0,1)(1,0,0)[12] intercept   : AIC=-2528.722, Time=0.12 sec
 ARIMA(0,0,1)(1,0,1)[12] intercept   : AIC=-2525.603, Time=0.40 sec
 ARIMA(1,0,1)(0,0,0)[12] intercept   : AIC=-2510.799, Time=0.13 sec
 ARIMA(0,0,2)(0,0,0)[12] intercept   : AIC=-2518.694, Time=0.11 sec
 ARIMA(1,0,0)(0,0,0)[12] intercept   : AIC=-2501.420, Time=0.07 sec
 ARIMA(1,0,2)(0,0,0)[12] intercept   : AIC=-2517.399, Time=0.24 sec
 ARIMA(0,0,1)(0,0,0)[12]             : AIC=-2532.654, Time=0.06 sec
 ARIMA(0,0,1)(1,0,0)[12]             : AIC=-2530.718, Time=0.06 sec
 ARIMA(0,0,1)(0,0,1)[12]             : AIC=-2530.196, Time=0.07 sec
 ARIMA(0,0,1)(1,0,1)[12]             : AIC=-2527.600, Time=0.16 sec
 ARIMA(1,0,1)(0,0,0)[12]             : AIC=-2512.816, Time=0.07 sec
 ARIMA(0,0,2)(0,0,0)[12]             : AIC=-2520.709, Time=0.09 sec
 ARIMA(1,0,0)(0,0,0)[12]             : AIC=-2503.415, Time=0.04 sec
 ARIMA(1,0,2)(0,0,0)[12]             : AIC=-2519.411, Time=0.06 sec

Best model:  ARIMA(0,0,1)(0,0,0)[12]
Total fit time: 2.648 seconds
```

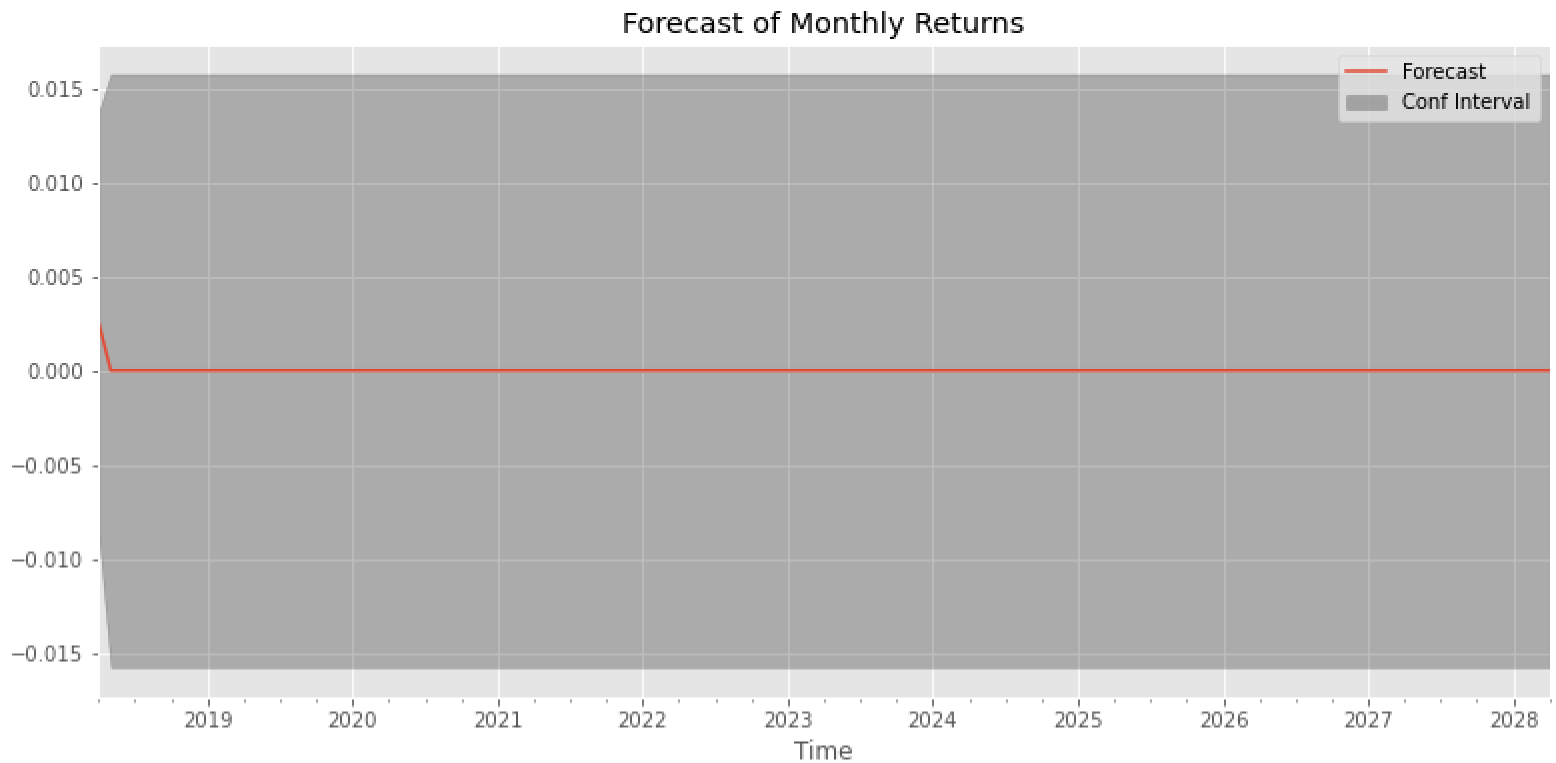
Out[107]:

```
ARIMA(maxiter=50, method='lbfgs', order=(0, 0, 1), out_of_sample_size=0,
      scoring='mse', scoring_args={}, seasonal_order=(0, 0, 0, 12),
      start_params=None, suppress_warnings=True, trend=None,
      with_intercept=False)
```

In [108]:

```
pdq = (0, 0, 1)
pdqs = (0, 0, 0, 12)
ret_93003 = forecast_model(TS_93003,pdq=pdq,pdqs=pdqs,zc=93003)
```

executed in 420ms, finished 19:01:30 2021-08-07




```
Total expected return in 1 year: 0.26%
Total expected return in 3 years: 0.26%
Total expected return in 5 year: 0.26%
Total expected return in 10 years: 0.26%
```

16.1 Plotting best 5 Zipcodes on the map

In [109]:

```
# Import the pandas library
import pandas as pd

# Make an empty map
m = folium.Map(location=[36,-120], tiles="OpenStreetMap", zoom_start=6)

# Show the map
m

# Make a data frame with dots to show on the map
data = pd.DataFrame({
   'lon':[-120.6423, -120.6636, -117.0262, -116.0358, -118.2371],
   'lat':[39.1308, 35.2715, 32.5655, 33.7326, 34.1381],
   'name':['Placer county', 'San Luis Obispo', 'San Diego', ' Riverside', 'Los Angeles'],
   'value':[583638, 641957, 551370, 439238, 587200]
}, dtype=str)

data
```

executed in 29ms, finished 19:01:30 2021-08-07

Out[109]:

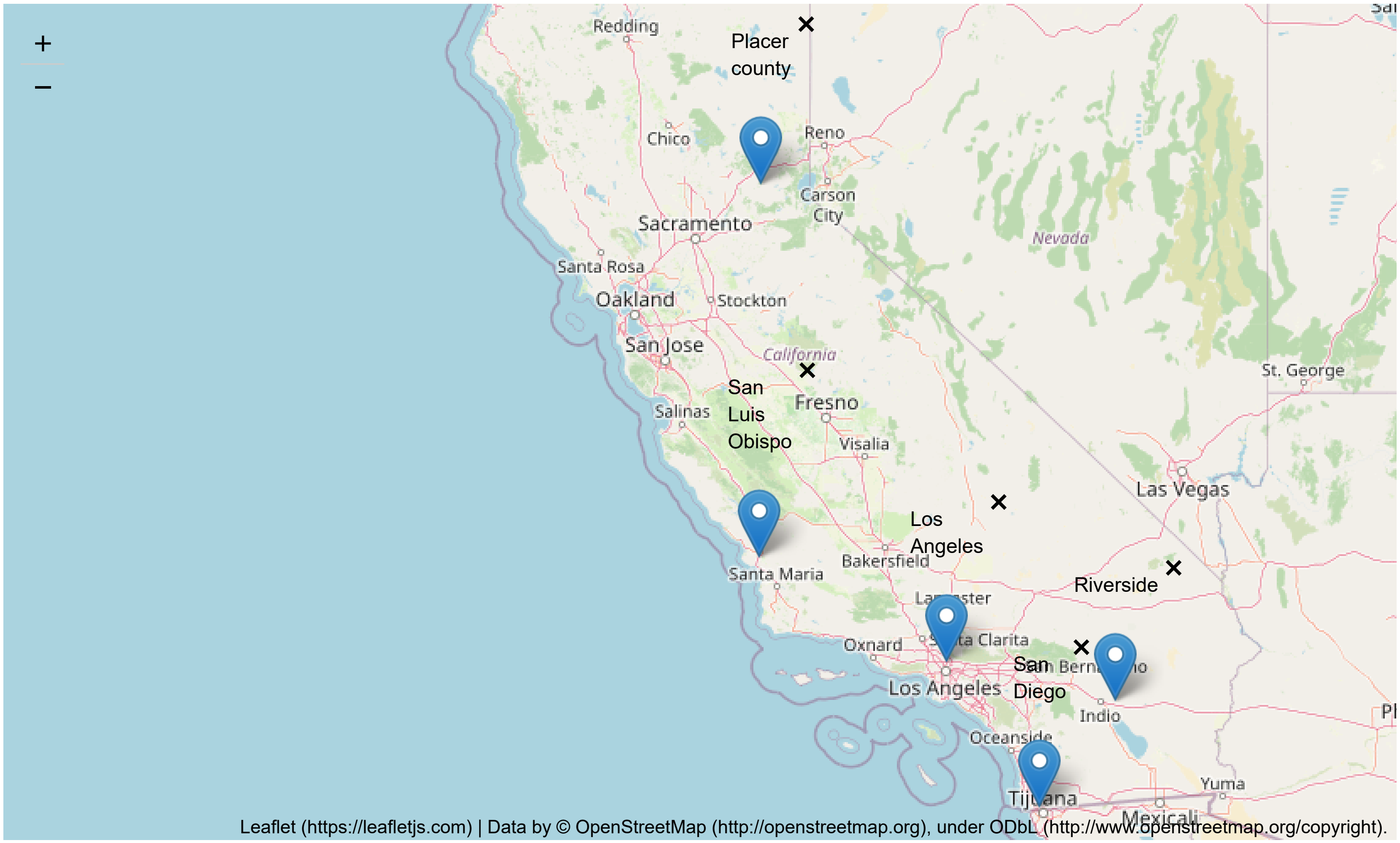
	lon	lat	name	value
0	-120.6423	39.1308	Placer county	583638
1	-120.6636	35.2715	San Luis Obispo	641957
2	-117.0262	32.5655	San Diego	551370
3	-116.0358	33.7326	Riverside	439238
4	-118.2371	34.1381	Los Angeles	587200

In [110]:

```
for i in range(0,len(data)):
    folium.Marker(location=[data.iloc[i]['lat'], data.iloc[i]['lon']],
     popup=folium.Popup(data.iloc[i]['name'], show=True),
   ).add_to(m)


m
```

executed in 20ms, finished 19:01:30 2021-08-07

Out[110]:



16.2 Conclusion and Recommendation

For the Real estate looking to immediately invest in the following zipcodes, here are the recommendations on the budget worth of a home and whether it is advisable to buy ,flip and sell the house, or buy and hold.

Zip code 92866 (LA- Long Beach county): Buy, Flip and sell homes within a year. (Budget of $584,000)

```
Total expected return in 1 year: 2.5%
Total expected return in 3 years: 2.67%
Total expected return in 5 year: 2.67%
Total expected return in 10 years: 2.67%
```

Zip code 93405 (San Luis Obispo): Buy and hold for the next 5-10 years. (Budget of $642,000)

```
Total expected return in 1 year: 8.1%
Total expected return in 3 years: 12.39%
Total expected return in 5 year: 13.0%
Total expected return in 10 years: 13.13%
```

Zip code 92101 (San Diego county): Buy and hold for the next 3-5 years. (Budget of $552,000)

```
Total expected return in 1 year: 10.47%
Total expected return in 3 years: 14.06%
Total expected return in 5 year: 14.27%
Total expected return in 10 years: 14.27%
```

Zip code 92860 (Riverside County): Buy, flip and sell within a year. (Budget of $439,000)

```
Total expected return in 1 year: 7.43%
Total expected return in 3 years: 11.23%
Total expected return in 5 year: 11.82%
Total expected return in 10 years: 11.93%
```

Zip code 91754 (Los Angeles): Buy and hold for atleast 10years. (Budget of $587,000)

```
Total expected return in 1 year: 2.6%
Total expected return in 3 years: 4.72%
Total expected return in 5 year: 5.32%
Total expected return in 10 years: 5.54%
```

16.3 Further study

- The model is unable to correctly adjust to unique events such as exogenous data. Interest rates, rent values and GDP would be important factors to explore the relationship they would have with the home values. Rent income should exceed the costs of maintenance, mortgage, insurance, taxes and other expenses. Any gains that may be realized from selling the property later should also be factored into the calculation.
- Model would be more effective with more recent years data and considering the impact of recent events on Real Estate investment.

In [ ]: