1 Texas Housing Time Analysis

By: Brian Lee

1.1 Business problem:

As multiple markets like the streaming and tech companies are interested in Texas due to relatively low house prices and no income tax. We have been hired by a housing agency that is looking into investing into Houston, Texas early. This will allow for the company to better expect price jumps early, allowing for earlier buys at lower prices and selling at higher prices.

1.2 Data Preparation

In [1]:

```
# Import necessary packages
import pandas as pd
import numpy as np
import matplotlib as mpl
from matplotlib import pyplot as plt
%matplotlib inline
plt.style.use('seaborn-whitegrid')

import warnings
warnings.filterwarnings('ignore')
import itertools
import statsmodels.api as sm

from statsmodels.tsa.seasonal import seasonal_decompose
from statsmodels.graphics.tsaplots import plot_acf,plot_pacf
from pandas.plotting import autocorrelation_plot,lag_plot

from statsmodels.tsa.statespace.sarimax import SARIMAX
```

executed in 1.53s, finished 22:25:54 2021-11-14

In [2]:
```
df = pd.read_csv('time-series/zillow_data.csv')
df.head()
```
executed in 410ms, finished 22:25:54 2021-11-14

Out[2]:

	RegionID	RegionName	City	State	Metro	CountyName	SizeRank	1996-04	1996
0	84654	60657	Chicago	IL	Chicago	Cook	1	334200.0	33540
1	90668	75070	McKinney	TX	Dallas-Fort Worth	Collin	2	235700.0	23690
2	91982	77494	Katy	TX	Houston	Harris	3	210400.0	21220
3	84616	60614	Chicago	IL	Chicago	Cook	4	498100.0	50090
4	93144	79936	El Paso	TX	El Paso	El Paso	5	77300.0	7730

5 rows × 272 columns

In [3]:
```
# Extract only Texas data values
TX_df = df[df['State'] == 'TX']
TX_df.head()
```
executed in 31ms, finished 22:25:54 2021-11-14

Out[3]:

	RegionID	RegionName	City	State	Metro	CountyName	SizeRank	1996-04	1996
1	90668	75070	McKinney	TX	Dallas-Fort Worth	Collin	2	235700.0	23690
2	91982	77494	Katy	TX	Houston	Harris	3	210400.0	21220
4	93144	79936	El Paso	TX	El Paso	El Paso	5	77300.0	7730
5	91733	77084	Houston	TX	Houston	Harris	6	95000.0	9520
8	91940	77449	Katy	TX	Houston	Harris	9	95400.0	9560

5 rows × 272 columns

In [4]:
```
# Houston dataset
houston_df = TX_df[TX_df['Metro'] == 'Houston']
```
executed in 14ms, finished 22:25:54 2021-11-14

We will use **Return on Investment (ROI)** in order to determine whether the home value in the area is best for the model.

Calculated (Final Value - Initial Value) / Cost of Investment

In [5]:

```
# 5 year ROI
houston_df['ROI_5_years'] = round((houston_df['2018-04'] - houston_df['2013-0

# 3 year ROI
houston_df['ROI_10_years'] = round((houston_df['2018-04'] - houston_df['2008-

houston_df.head()
```

executed in 30ms, finished 22:25:54 2021-11-14

Out[5]:

	RegionID	RegionName	City	State	Metro	CountyName	SizeRank	1996-04	1996
2	91982	77494	Katy	TX	Houston	Harris	3	210400.0	21220
5	91733	77084	Houston	TX	Houston	Harris	6	95000.0	9520
8	91940	77449	Katy	TX	Houston	Harris	9	95400.0	9560
22	92036	77573	League City	TX	Houston	Galveston	23	141400.0	14100
23	92045	77584	Pearland	TX	Houston	Brazoria	24	138500.0	13870

5 rows × 274 columns

In [6]:

```
# Reshape from Wide to Long Format
def melt_data(df):
    melted = pd.melt(df, id_vars=['RegionName', 'City', 'CountyName', 'ROI_5_
    melted['Date'] = pd.to_datetime(melted['Date'], infer_datetime_format=Tru
    melted = melted.dropna(subset=['value'])
    return melted
```

executed in 15ms, finished 22:25:54 2021-11-14

In [7]:

```
# Remove unnecessary columns
houston_df.drop(columns=['RegionID','State','Metro','SizeRank'], inplace=True
```

executed in 15ms, finished 22:25:54 2021-11-14

In [8]:
```
houston_df.head()
```
executed in 30ms, finished 22:25:54 2021-11-14

Out[8]:

	RegionName	City	CountyName	1996-04	1996-05	1996-06	1996-07	1996-08	'
2	77494	Katy	Harris	210400.0	212200.0	212200.0	210700.0	208300.0	2(
5	77084	Houston	Harris	95000.0	95200.0	95400.0	95700.0	95900.0	(
8	77449	Katy	Harris	95400.0	95600.0	95800.0	96100.0	96400.0	(
22	77573	League City	Galveston	141400.0	141000.0	140600.0	140500.0	140400.0	1‹
23	77584	Pearland	Brazoria	138500.0	138700.0	139200.0	139900.0	140700.0	1‹

5 rows × 270 columns

In [9]:
```
houston_df.isna().sum().sort_values(ascending=False)
```
executed in 14ms, finished 22:25:54 2021-11-14

Out[9]:
```
ROI_10_years    0
2003-01         0
2004-03         0
2004-02         0
2004-01         0
               ..
2010-07         0
2010-06         0
2010-05         0
2010-04         0
RegionName      0
Length: 270, dtype: int64
```

In [10]:
```
melt_df = melt_data(houston_df)
melt_df.head()
```
executed in 46ms, finished 22:25:54 2021-11-14

Out[10]:

	RegionName	City	CountyName	ROI_5_years	ROI_10_years	Date	value
0	77494	Katy	Harris	0.2842	0.2983	1996-04-01	210400.0
1	77084	Houston	Harris	0.4617	0.2494	1996-04-01	95000.0
2	77449	Katy	Harris	0.5021	0.2908	1996-04-01	95400.0
3	77573	League City	Galveston	0.4130	0.3189	1996-04-01	141400.0
4	77584	Pearland	Brazoria	0.3783	0.2749	1996-04-01	138500.0

In [11]:

```
melt_df.rename(columns = {"RegionName": "Zipcode", "CountyName": "County", "t
melt_df.head()
```

executed in 15ms, finished 22:25:54 2021-11-14

Out[11]:

	Zipcode	City	County	ROI_5_years	ROI_10_years	Date	Price
0	77494	Katy	Harris	0.2842	0.2983	1996-04-01	210400.0
1	77084	Houston	Harris	0.4617	0.2494	1996-04-01	95000.0
2	77449	Katy	Harris	0.5021	0.2908	1996-04-01	95400.0
3	77573	League City	Galveston	0.4130	0.3189	1996-04-01	141400.0
4	77584	Pearland	Brazoria	0.3783	0.2749	1996-04-01	138500.0

In [12]:

```
# Zipcode type to string
melt_df['Zipcode'] = melt_df['Zipcode'].astype(str)

# 'Date' column to datetime format
melt_df['Date'] = pd.to_datetime(melt_df['Date'], format='%m/%y')

# 'Date' column as dataframe index
melt_df.set_index('Date', inplace=True)
```

executed in 63ms, finished 22:25:54 2021-11-14

In [13]:

```
melt_df
```

executed in 15ms, finished 22:25:54 2021-11-14

Out[13]:

	Zipcode	City	County	ROI_5_years	ROI_10_years	Price
Date						
1996-04-01	77494	Katy	Harris	0.2842	0.2983	210400.0
1996-04-01	77084	Houston	Harris	0.4617	0.2494	95000.0
1996-04-01	77449	Katy	Harris	0.5021	0.2908	95400.0
1996-04-01	77573	League City	Galveston	0.4130	0.3189	141400.0
1996-04-01	77584	Pearland	Brazoria	0.3783	0.2749	138500.0
...	...	...	...	...	...	...
2018-04-01	77514	Anahuac	Chambers	0.3933	0.2795	136400.0
2018-04-01	77050	Houston	Harris	0.5695	0.2468	115200.0
2018-04-01	77650	Port Bolivar	Galveston	0.2758	0.1968	247500.0
2018-04-01	77534	Danbury	Brazoria	0.3290	0.2795	164800.0
2018-04-01	77577	Liverpool	Brazoria	0.4772	0.2110	149200.0

49555 rows × 6 columns

In [14]:
```
# Double Check for missing values
melt_df.isna().sum()
```
executed in 29ms, finished 22:25:54 2021-11-14

Out[14]:
```
Zipcode         0
City            0
County          0
ROI_5_years     0
ROI_10_years    0
Price           0
dtype: int64
```

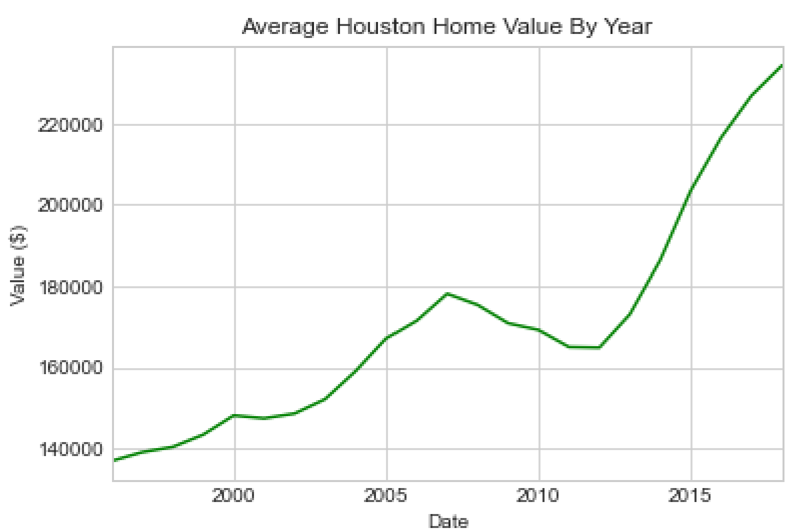
1.3 EDA and Visualization

Questions we can ask:

1. How have Houston housing prices generally trended over time?
2. Which counties have higher home prices?
3. Which counties have the highest ROIs?
4. Which Zipcodes in Houston have the highest home prices?

1.3.1 How have Houston housing prices generally trended over time?

In [15]:
```
yearly_df = melt_df['Price'].resample(rule='A').mean()
yearly_df.plot.line(color='green')
plt.title('Average Houston Home Value By Year')
plt.ylabel('Value ($)')
plt.show()
```
executed in 220ms, finished 22:25:55 2021-11-14

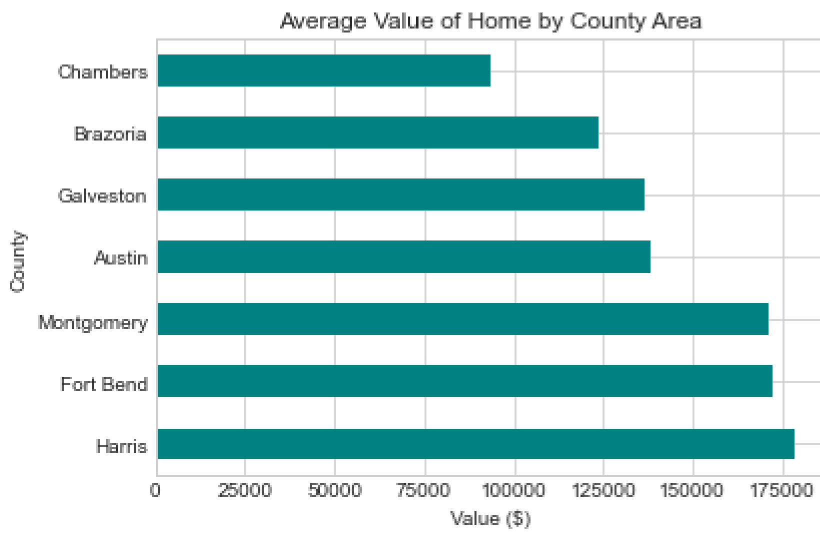


- Outside of the dip in housing prices between 2007 and 2011, the average price of a Texas home has **steadily increased**.

1.3.2 Which counties have higher home prices?

In [16]:
```
county_df = melt_df.copy()
county_df = county_df.groupby('County').Price.mean().sort_values(ascending=Fa
```
executed in 28ms, finished 22:25:55 2021-11-14

In [17]:
```
county_df.plot.barh(color='teal')
plt.title('Average Value of Home by County Area')
plt.xlabel('Value ($)')
plt.show()
```
executed in 181ms, finished 22:25:55 2021-11-14



- **Harris, Fort Bend, and Montgomery** have clearly the highest average home values in the past 20 years.

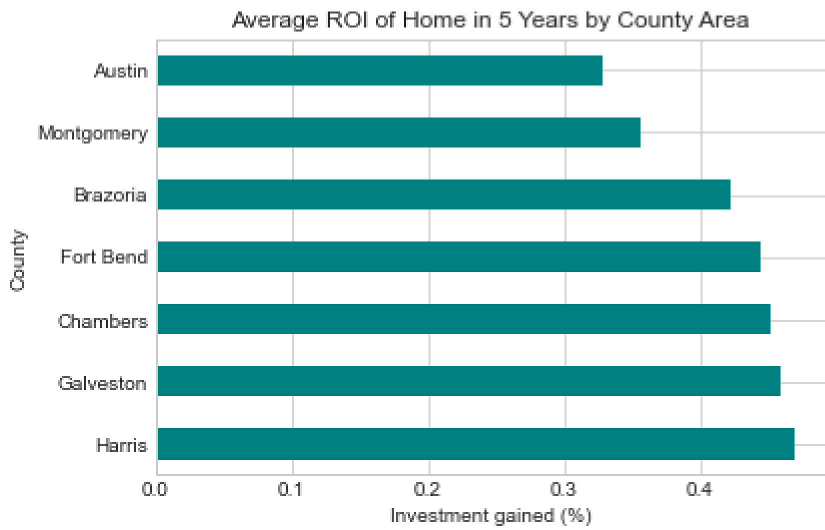
1.3.3 Which counties have the highest ROIs?

In [18]:
```
# 5 year ROI
ROI_5_df = melt_df.copy()
ROI_5_df = ROI_5_df.groupby('County').ROI_5_years.mean().sort_values(ascendir
```
executed in 14ms, finished 22:25:55 2021-11-14

In [19]:

```
ROI_5_df.plot.barh(color='teal')
plt.title('Average ROI of Home in 5 Years by County Area')
plt.xlabel('Investment gained (%)')
plt.show()
```

executed in 155ms, finished 22:25:55 2021-11-14



In [20]:

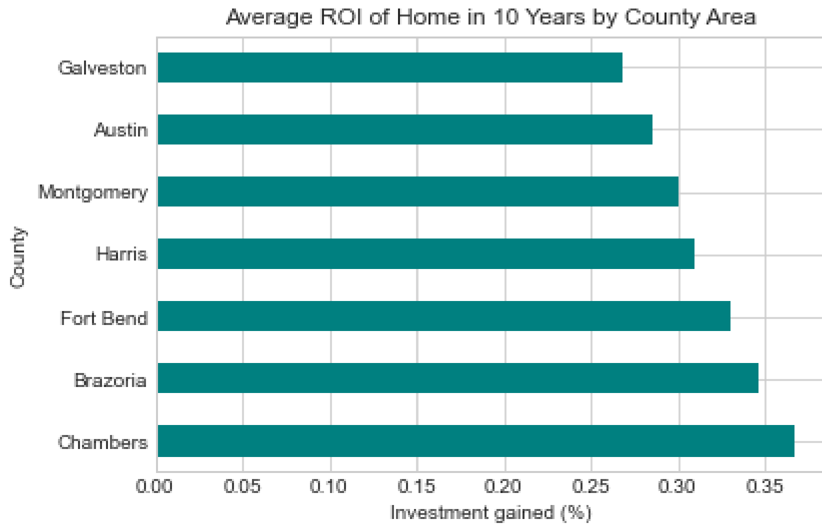
```
# 10 year ROI
ROI_10_df = melt_df.copy()
ROI_10_df = ROI_10_df.groupby('County').ROI_10_years.mean().sort_values(ascer
```

executed in 15ms, finished 22:25:55 2021-11-14

In [21]:

```
ROI_10_df.plot.barh(color='teal')
plt.title('Average ROI of Home in 10 Years by County Area')
plt.xlabel('Investment gained (%)')
plt.show()
```

executed in 205ms, finished 22:25:55 2021-11-14



1.3.4 Which Zipcodes in Houston have the highest home prices?

In [22]:

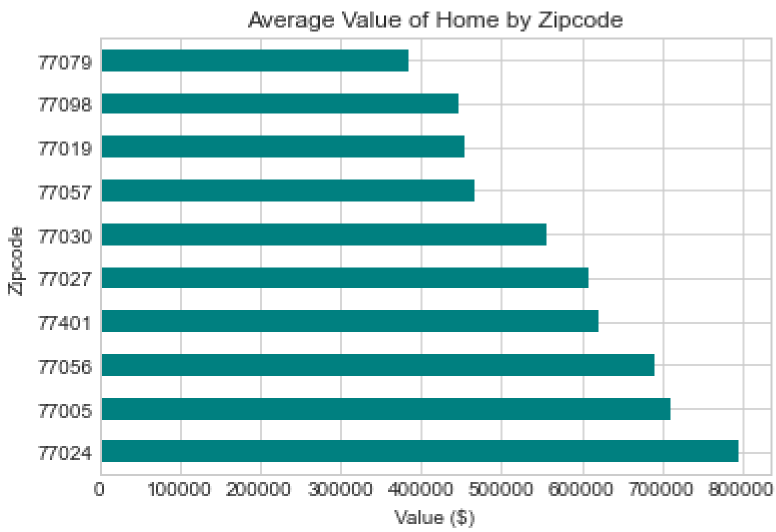
```
zip_df = melt_df.copy()
zip_df = zip_df.groupby('Zipcode').Price.mean().sort_values(ascending=False).
```

executed in 15ms, finished 22:25:55 2021-11-14

In [23]:

```
zip_df.plot.barh(color='teal')
plt.title('Average Value of Home by Zipcode')
plt.xlabel('Value ($)')
plt.show()
```

executed in 185ms, finished 22:25:55 2021-11-14



- **77024** appears to be the zipcode with the Highest average home value in the past 20 years.

In [24]:

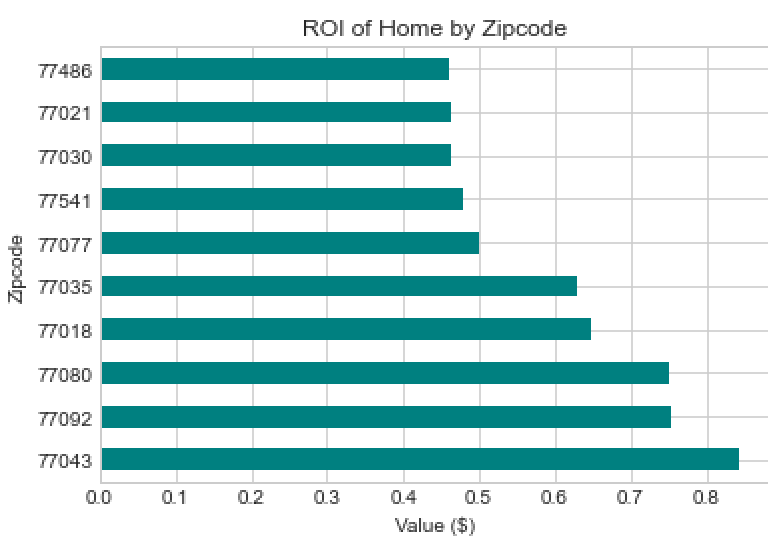
```
zip_df = melt_df.copy()
zip_df = zip_df.groupby('Zipcode').ROI_10_years.mean().sort_values(ascending=
```

executed in 15ms, finished 22:25:55 2021-11-14

In [25]:

```
zip_df.plot.barh(color='teal')
plt.title('ROI of Home by Zipcode')
plt.xlabel('Value ($)')
plt.show()
```

executed in 189ms, finished 22:25:56 2021-11-14



1.4 Time Series Modeling

Type *Markdown* and LaTeX: α^2

In [26]:
```
zip_list = ['77486', '77021', '77030', '77541', '77077', '77035', '77018', '7
```
executed in 14ms, finished 22:25:56 2021-11-14

In [27]:
```
ts = {}
for code in zip_list:
    temp_df = melt_df.groupby('Zipcode').get_group(code).sort_index()['Price'
    ts[code] = temp_df
```
executed in 60ms, finished 22:25:56 2021-11-14

In [28]:
```
ts_df = pd.DataFrame(ts)
ts_df.head()
```
executed in 29ms, finished 22:25:56 2021-11-14

Out[28]:

	77486	77021	77030	77541	77077	77035	77018	77080	77092
Date									
1996-04-01	67000.0	45500.0	559900.0	45600.0	177100.0	118800.0	182500.0	107400.0	95600.0
1996-05-01	66700.0	45200.0	563500.0	45500.0	180000.0	119700.0	185900.0	106100.0	96900.0
1996-06-01	66500.0	44900.0	567200.0	45400.0	182700.0	120900.0	189100.0	105100.0	98400.0
1996-07-01	66200.0	44800.0	570900.0	45300.0	185100.0	122300.0	191700.0	104200.0	99900.0
1996-08-01	66100.0	44600.0	574500.0	45200.0	187100.0	124100.0	193400.0	103500.0	101500.0

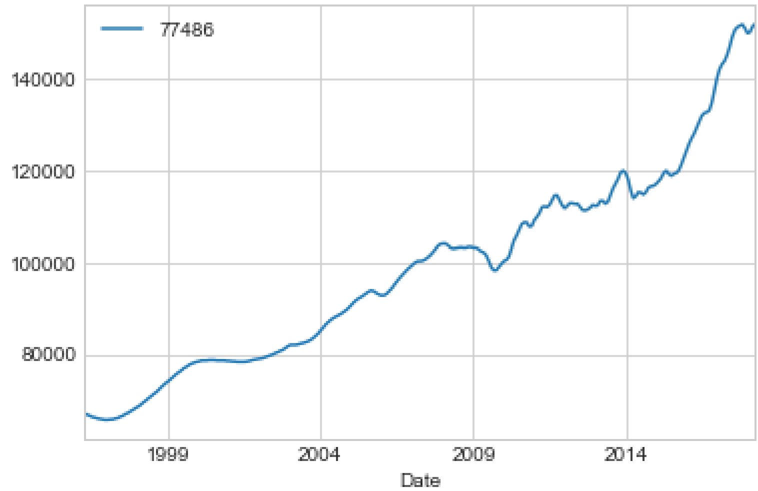
Let's start with our first Zipcode

In [29]:
```
zip_1 = zip_list[0]
```
executed in 15ms, finished 22:25:56 2021-11-14

In [30]:

```
ts = ts_df[zip_1].copy()
ax = ts.plot()
ax.legend()
plt.show()
```

executed in 172ms, finished 22:25:56 2021-11-14



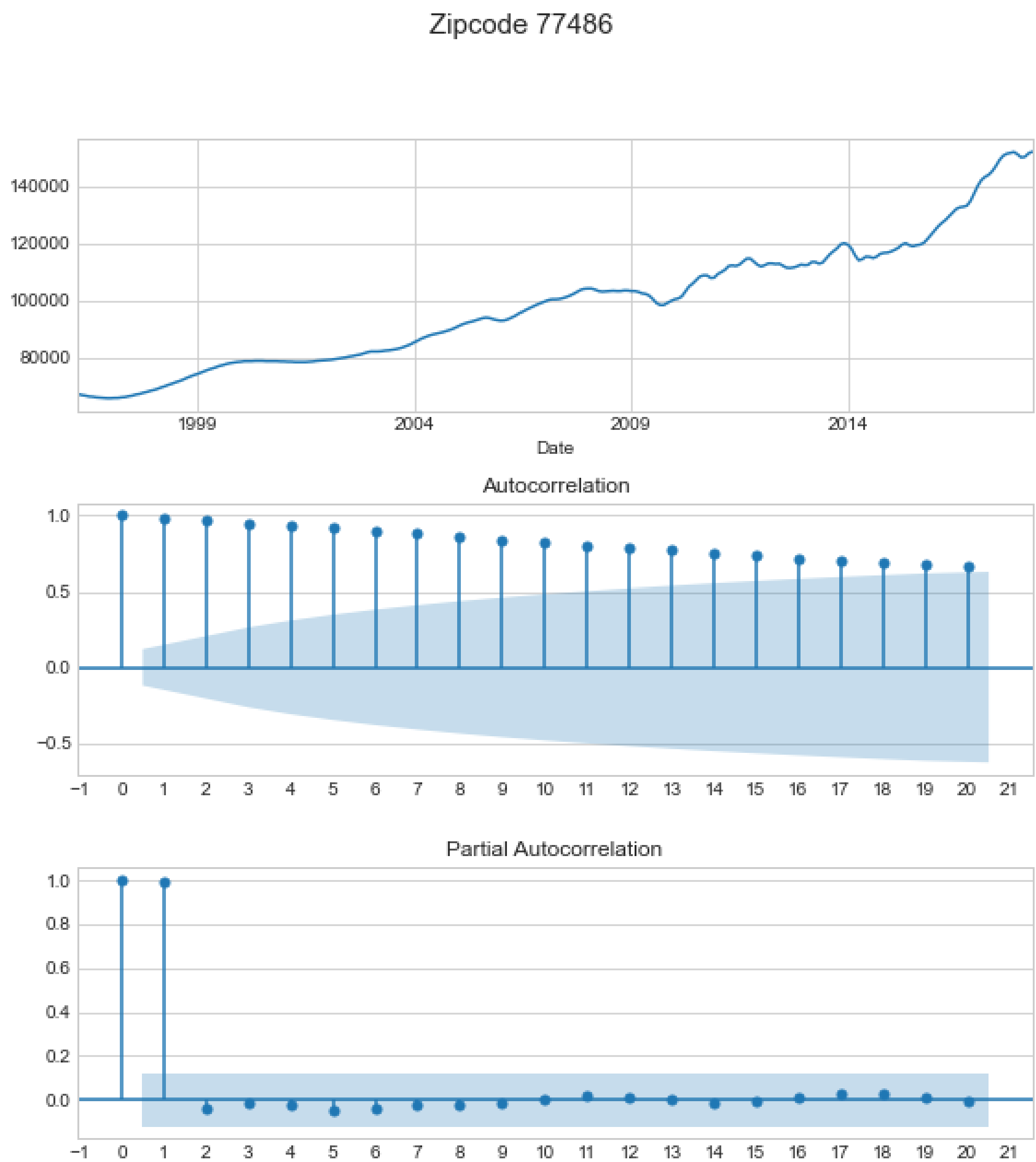
In [31]:

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

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

     ## Plot ts
    ts.plot(ax=ax[0])

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

    fig.suptitle(f"Zipcode {ts.name}",y=1.1,fontsize=15)

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

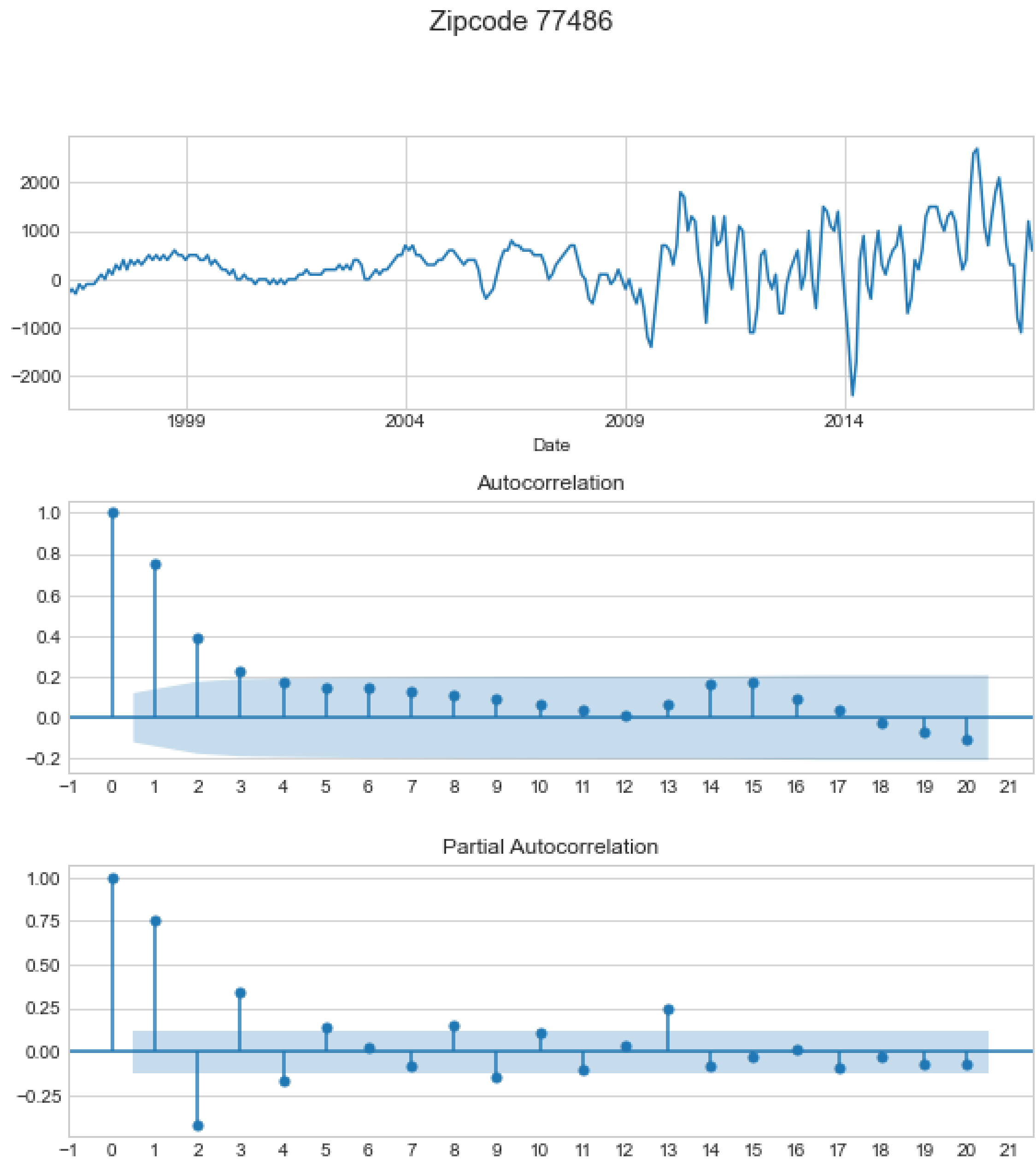
executed in 15ms, finished 22:25:56 2021-11-14

In [32]:
```
plot_autocorr(ts,lags=20);
```

executed in 530ms, finished 22:25:56 2021-11-14



In [33]:
```
# Let's do a difference to remove trends
d = 1
plot_autocorr(ts.diff(d).dropna(),lags=20);
```
executed in 579ms, finished 22:25:57 2021-11-14



1.4.1 Model 1

In [34]:

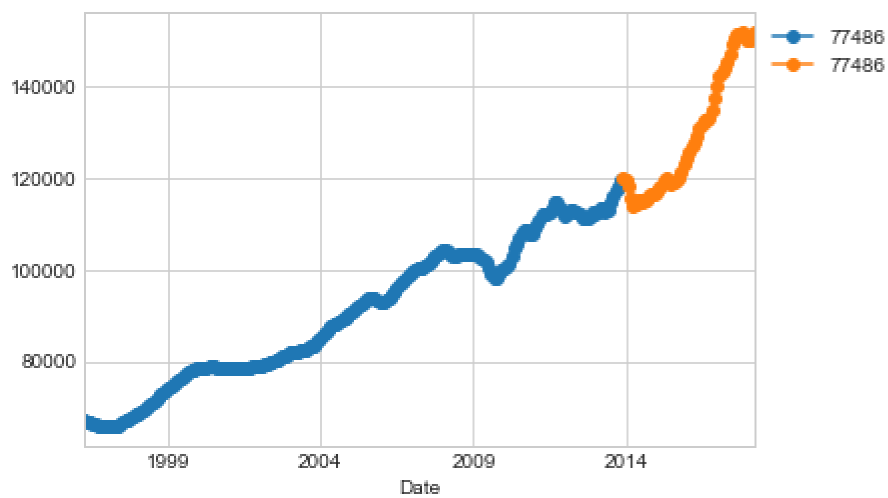
```
# selected params
d = 1
p = 1
q = 1
```

executed in 14ms, finished 22:25:57 2021-11-14

In [35]:

```
# Train Test Split
train_size = 0.8
split_idx = round(len(ts)* train_size)
split_idx

## Split
train = ts.iloc[:split_idx]
test = ts.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 216ms, finished 22:25:57 2021-11-14



In [36]:

```
# 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 796ms, finished 22:25:58 2021-11-14

```
C:\Users\leebr\anaconda3\envs\learn-env\lib\site-packages\statsmodels\tsa\b
ase\tsa_model.py:524: ValueWarning: No frequency information was provided,
so inferred frequency MS will be used.
  warnings.warn('No frequency information was'
C:\Users\leebr\anaconda3\envs\learn-env\lib\site-packages\statsmodels\tsa\b
ase\tsa_model.py:524: ValueWarning: No frequency information was provided,
so inferred frequency MS will be used.
  warnings.warn('No frequency information was'
```

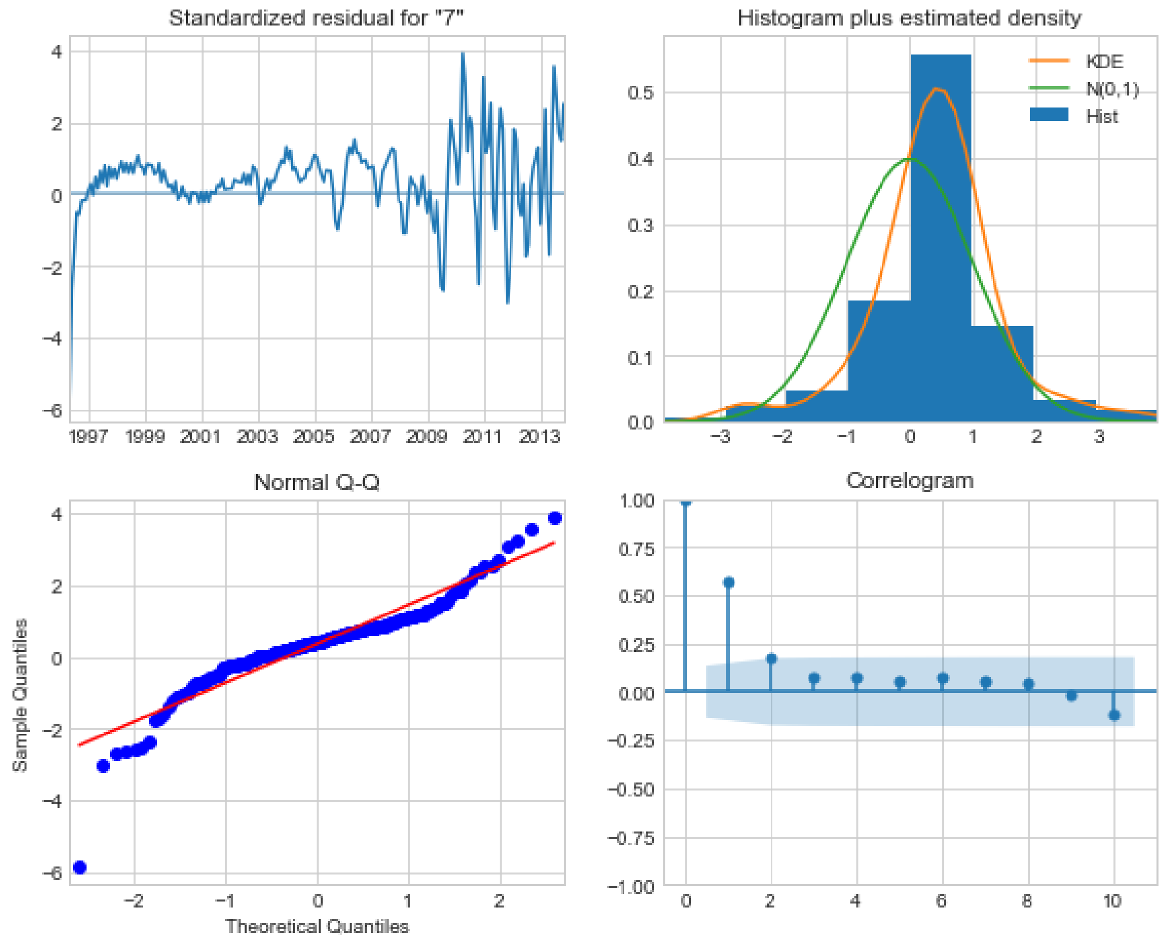
SARIMAX Results

Dep. Variable:	77486	**No. Observations:**	212
Model:	SARIMAX(1, 1, 1)	**Log Likelihood**	-1600.990
Date:	Sun, 14 Nov 2021	**AIC**	3207.980
Time:	22:25:57	**BIC**	3218.036
Sample:	04-01-1996	**HQIC**	3212.045
	- 11-01-2013		
Covariance Type:	opg		

	coef	**std err**	**z**	**P>|z|**	**[0.025**	**0.975]**
ar.L1	0.6493	0.052	12.499	0.000	0.548	0.751
ma.L1	-0.4618	0.058	-7.992	0.000	-0.575	-0.349
sigma2	1.672e+05	1.12e+04	14.905	0.000	1.45e+05	1.89e+05

Ljung-Box (L1) (Q):	69.61	**Jarque-Bera (JB):**	354.53
Prob(Q):	0.00	**Prob(JB):**	0.00
Heteroskedasticity (H):	2.95	**Skew:**	-0.95
Prob(H) (two-sided):	0.00	**Kurtosis:**	9.06

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



In [37]:

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

executed in 405ms, finished 22:25:58 2021-11-14

In [38]:

```
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
```

executed in 15ms, finished 22:25:58 2021-11-14

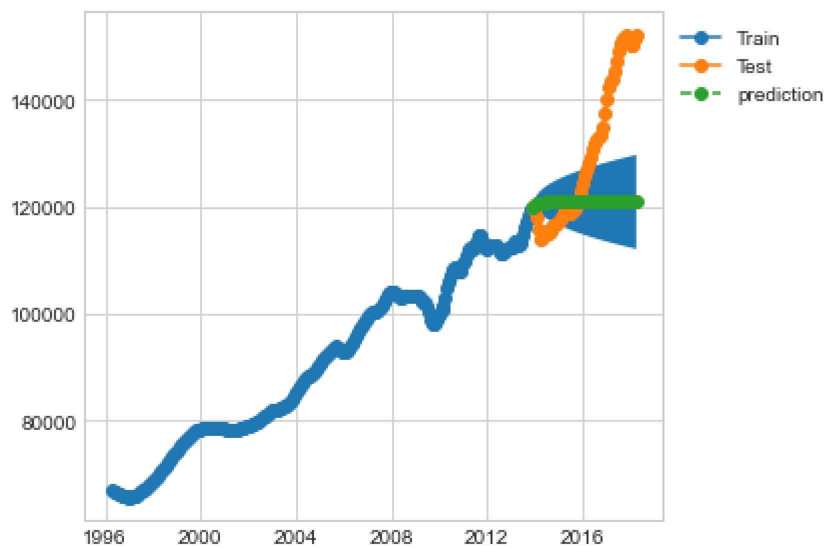
In [39]:

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

executed in 15ms, finished 22:25:58 2021-11-14

In [40]:
```
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 14ms, finished 22:25:58 2021-11-14

In [41]:
```
plot_train_test_pred(train,test,pred_df)
plt.show()
```
executed in 204ms, finished 22:25:59 2021-11-14



1.4.2 Model 2

In [42]:
```
!pip install --user pmdarima
import pmdarima as pm
from pmdarima import auto_arima
```
executed in 3.44s, finished 22:26:02 2021-11-14

```
Requirement already satisfied: pmdarima in c:\users\leebr\appdata\roaming\p
ython\python38\site-packages (1.8.2)
Requirement already satisfied: joblib>=0.11 in c:\users\leebr\anaconda3\env
s\learn-env\lib\site-packages (from pmdarima) (0.17.0)
Requirement already satisfied: scikit-learn>=0.22 in c:\users\leebr\anacond
a3\envs\learn-env\lib\site-packages (from pmdarima) (0.23.2)
Requirement already satisfied: pandas>=0.19 in c:\users\leebr\anaconda3\env
s\learn-env\lib\site-packages (from pmdarima) (1.1.3)
Requirement already satisfied: statsmodels!=0.12.0,>=0.11 in c:\users\leebr
\anaconda3\envs\learn-env\lib\site-packages (from pmdarima) (0.12.2)
Requirement already satisfied: setuptools!=50.0.0,>=38.6.0 in c:\users\leeb
r\anaconda3\envs\learn-env\lib\site-packages (from pmdarima) (50.3.0.post20
201103)
Requirement already satisfied: Cython!=0.29.18,>=0.29 in c:\users\leebr\ana
conda3\envs\learn-env\lib\site-packages (from pmdarima) (0.29.21)
Requirement already satisfied: numpy~=1.19.0 in c:\users\leebr\anaconda3\en
vs\learn-env\lib\site-packages (from pmdarima) (1.19.5)
Requirement already satisfied: urllib3 in c:\users\leebr\anaconda3\envs\lea
rn-env\lib\site-packages (from pmdarima) (1.25.10)
Requirement already satisfied: scipy>=1.3.2 in c:\users\leebr\anaconda3\env
s\learn-env\lib\site-packages (from pmdarima) (1.5.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\leebr\anaco
nda3\envs\learn-env\lib\site-packages (from scikit-learn>=0.22->pmdarima)
(2.1.0)
Requirement already satisfied: pytz>=2017.2 in c:\users\leebr\anaconda3\env
s\learn-env\lib\site-packages (from pandas>=0.19->pmdarima) (2020.1)
Requirement already satisfied: python-dateutil>=2.7.3 in c:\users\leebr\ana
conda3\envs\learn-env\lib\site-packages (from pandas>=0.19->pmdarima) (2.8.
1)
Requirement already satisfied: patsy>=0.5 in c:\users\leebr\anaconda3\envs
\learn-env\lib\site-packages (from statsmodels!=0.12.0,>=0.11->pmdarima)
(0.5.1)
Requirement already satisfied: six>=1.5 in c:\users\leebr\anaconda3\envs\le
arn-env\lib\site-packages (from python-dateutil>=2.7.3->pandas>=0.19->pmdar
ima) (1.15.0)
```

In [43]:

```
auto_model = auto_arima(train,start_p=0,start_q=0)
display(auto_model.summary())
auto_model.plot_diagnostics(figsize=(10,8));
```

executed in 1.37s, finished 22:26:03 2021-11-14

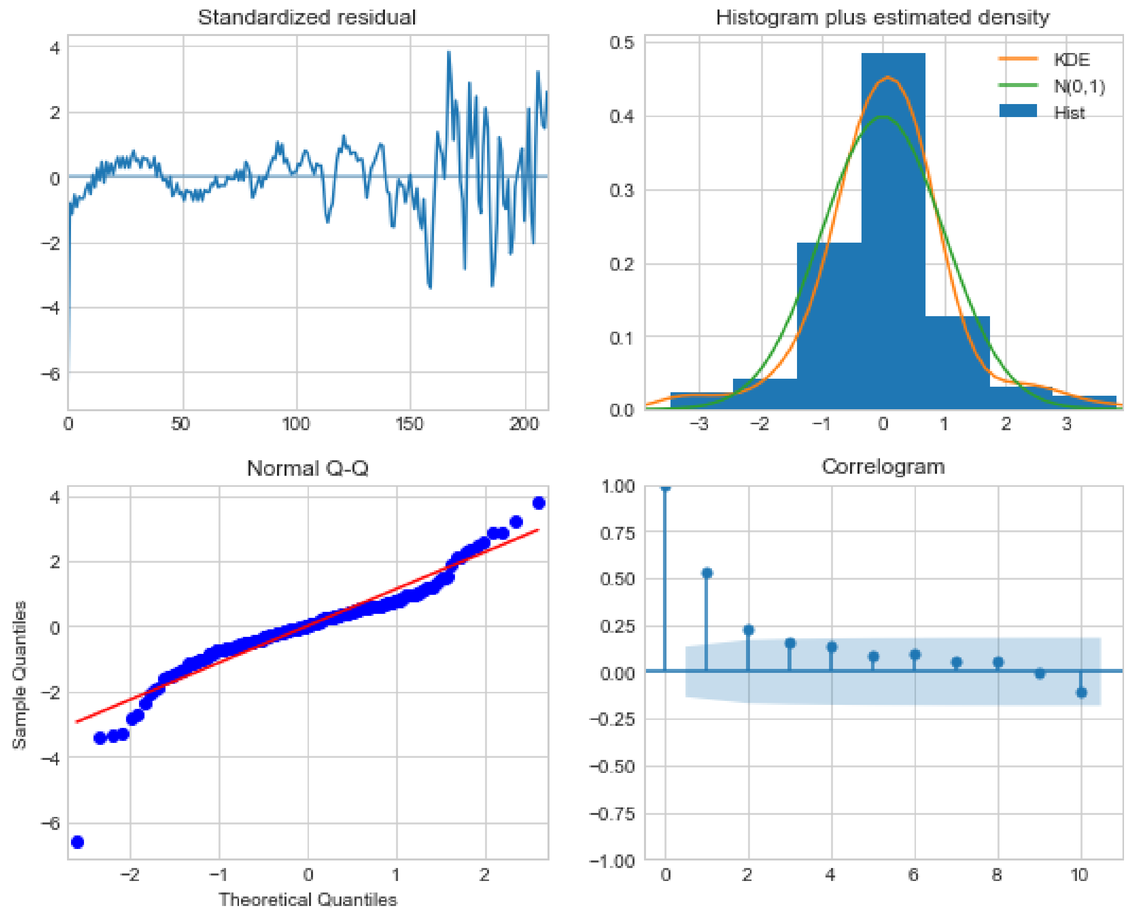
SARIMAX Results

Dep. Variable:	y	**No. Observations:**	212
Model:	SARIMAX(1, 1, 0)	**Log Likelihood**	-1583.510
Date:	Sun, 14 Nov 2021	**AIC**	3173.019
Time:	22:26:03	**BIC**	3183.075
Sample:	0	**HQIC**	3177.084
	- 212		
Covariance Type:	opg		

	coef	**std err**	**z**	**P>|z|**	**[0.025**	**0.975]**
intercept	180.8387	25.951	6.968	0.000	129.975	231.702
ar.L1	0.2309	0.008	27.555	0.000	0.214	0.247
sigma2	1.45e+05	9471.287	15.309	0.000	1.26e+05	1.64e+05

Ljung-Box (L1) (Q):	61.20	**Jarque-Bera (JB):**	377.38
Prob(Q):	0.00	**Prob(JB):**	0.00
Heteroskedasticity (H):	3.15	**Skew:**	-0.88
Prob(H) (two-sided):	0.00	**Kurtosis:**	9.31

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



1.4.3 Facebook Prophet Model

In [50]:
```
# Installing from a jupyter notebook
!pip install pystan
!pip install fbprophet
```
executed in 4.19s, finished 22:28:29 2021-11-14

...

In [58]:
```
from fbprophet import Prophet
```
executed in 2ms, finished 22:37:30 2021-11-14

In [59]:
```
# Set the uncertainty interval to 95% (the Prophet default is 80%)
Model = Prophet(interval_width=0.95)
```
executed in 941ms, finished 22:37:32 2021-11-14

1.4.4 Model 3

In [44]:

```
model3 = SARIMAX(ts,order=auto_model.order,
                 seasonal_order=auto_model.seasonal_order).fit()
display(model3.summary())
model3.plot_diagnostics(figsize=(10,8));
```

executed in 671ms, finished 22:26:04 2021-11-14

```
C:\Users\leebr\anaconda3\envs\learn-env\lib\site-packages\statsmodels\tsa\b
ase\tsa_model.py:524: ValueWarning: No frequency information was provided,
so inferred frequency MS will be used.
  warnings.warn('No frequency information was'
C:\Users\leebr\anaconda3\envs\learn-env\lib\site-packages\statsmodels\tsa\b
ase\tsa_model.py:524: ValueWarning: No frequency information was provided,
so inferred frequency MS will be used.
  warnings.warn('No frequency information was'
```

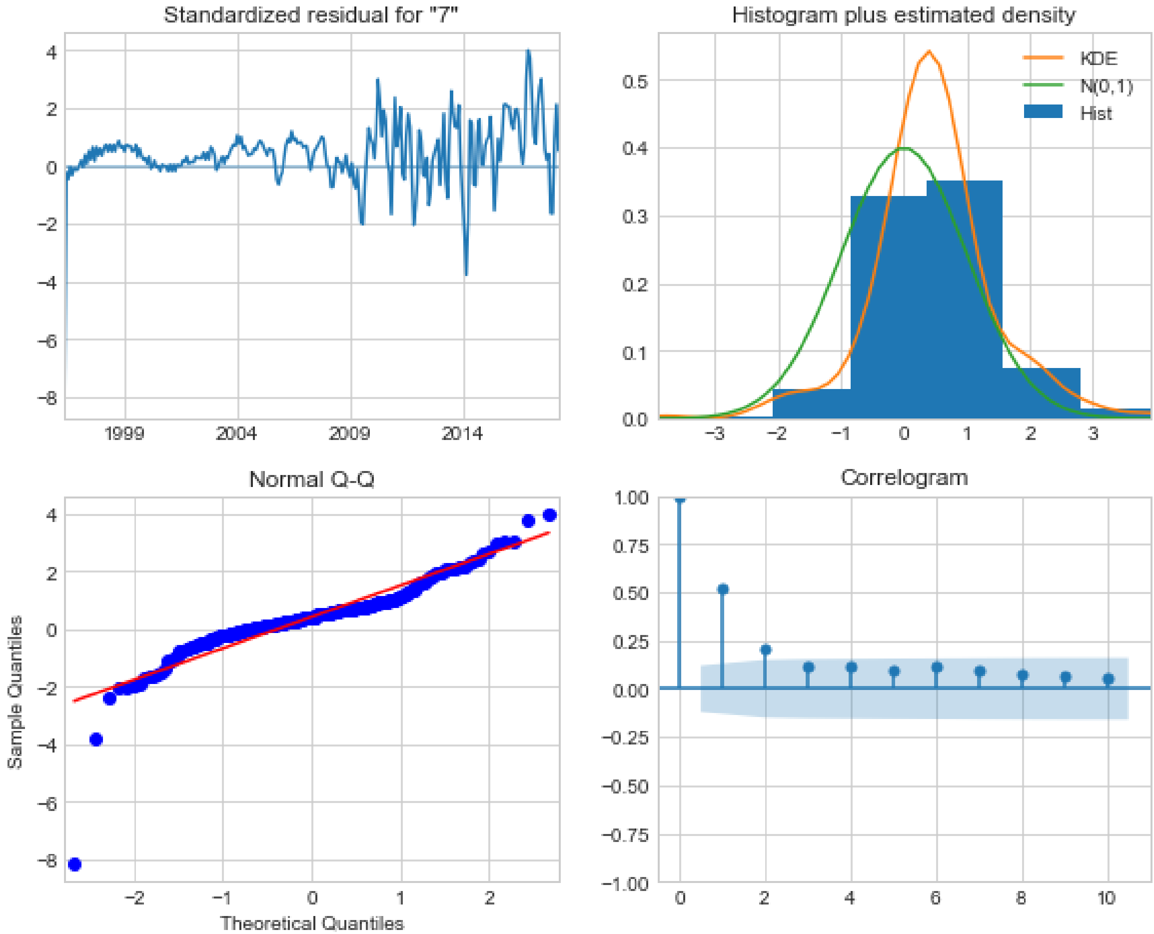
SARIMAX Results

Dep. Variable:	77486	**No. Observations:**	265
Model:	SARIMAX(1, 1, 0)	**Log Likelihood**	-2081.875
Date:	Sun, 14 Nov 2021	**AIC**	4167.751
Time:	22:26:04	**BIC**	4174.903
Sample:	04-01-1996	**HQIC**	4170.625
	- 04-01-2018		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	0.2651	0.006	45.176	0.000	0.254	0.277
sigma2	2.882e+05	1.82e+04	15.807	0.000	2.52e+05	3.24e+05

Ljung-Box (L1) (Q):	71.94	**Jarque-Bera (JB):**	2542.44
Prob(Q):	0.00	**Prob(JB):**	0.00
Heteroskedasticity (H):	2.69	**Skew:**	-1.76
Prob(H) (two-sided):	0.00	**Kurtosis:**	17.79

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

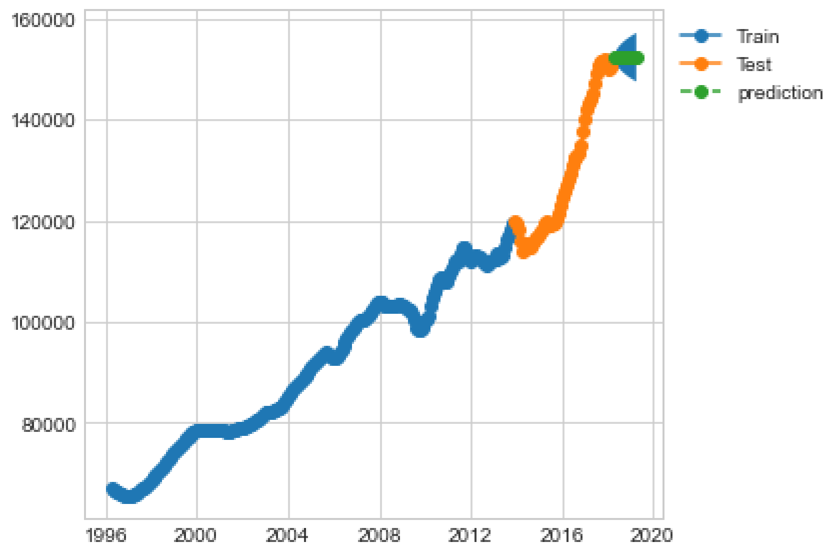

Standardized residual for "7"
Histogram plus estimated density
KDE
N(0,1)
Hist
Normal Q-Q
Sample Quantiles
Theoretical Quantiles
Correlogram

In [45]:

```
pred = model3.get_forecast(steps=12)
pred_df = forecast_to_df(pred,zip_1)
display(plot_train_test_pred(train,test,pred_df));
plt.show()
```

executed in 217ms, finished 22:26:04 2021-11-14

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



In [46]:

```
RESULTS = {}

for zc in zip_list:
    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 = 0.8
    split_idx = round(len(ts)* 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)
    pred_df = forecast_to_df(pred,zip_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 21.7s, finished 22:26:26 2021-11-14

```
77486
```

```
C:\Users\leebr\anaconda3\envs\learn-env\lib\site-packages\statsmodels\tsa
\base\tsa_model.py:524: ValueWarning: No frequency information was provid
ed, so inferred frequency MS will be used.
  warnings.warn('No frequency information was'
C:\Users\leebr\anaconda3\envs\learn-env\lib\site-packages\statsmodels\tsa
\base\tsa_model.py:524: ValueWarning: No frequency information was provid
ed, so inferred frequency MS will be used.
  warnings.warn('No frequency information was'
```

Best 3 Zipcodes appear to be:

- 77043
- 77018
- 77541

Looking at the models, theses Zipcodes have less a chance of decline with still great possibilities of the home price increasing.

1.5 Results

In [47]:
```
houston_df[houston_df['RegionName'] == 77043]
```
executed in 31ms, finished 22:26:26 2021-11-14

Out[47]:

	RegionName	City	CountyName	1996-04	1996-05	1996-06	1996-07	1996-08
4654	77043	Houston	Harris	123400.0	123300.0	123300.0	123500.0	123800.0

1 rows × 270 columns

In [48]:
```
houston_df[houston_df['RegionName'] == 77018]
```
executed in 31ms, finished 22:26:26 2021-11-14

Out[48]:

	RegionName	City	CountyName	1996-04	1996-05	1996-06	1996-07	1996-08
3380	77018	Houston	Harris	182500.0	185900.0	189100.0	191700.0	193400.0

1 rows × 270 columns

In [49]:
```
houston_df[houston_df['RegionName'] == 77541]
```
executed in 31ms, finished 22:26:26 2021-11-14

Out[49]:

	RegionName	City	CountyName	1996-04	1996-05	1996-06	1996-07	1996-08	199(
6457	77541	Freeport	Brazoria	45600.0	45500.0	45400.0	45300.0	45200.0	45300

1 rows × 270 columns

1.6 Conclusion

Invest in 3 zipcode areas early before the price increase further:

- 77043
- 77018
- 77541

1.7 Next Steps

- Test time series with model using Facebook prophet
- Get another dataset looking the years in which historical oil prices are listed. Houston is known as an "oil industry" city. Would be interesting to see impact on home prices