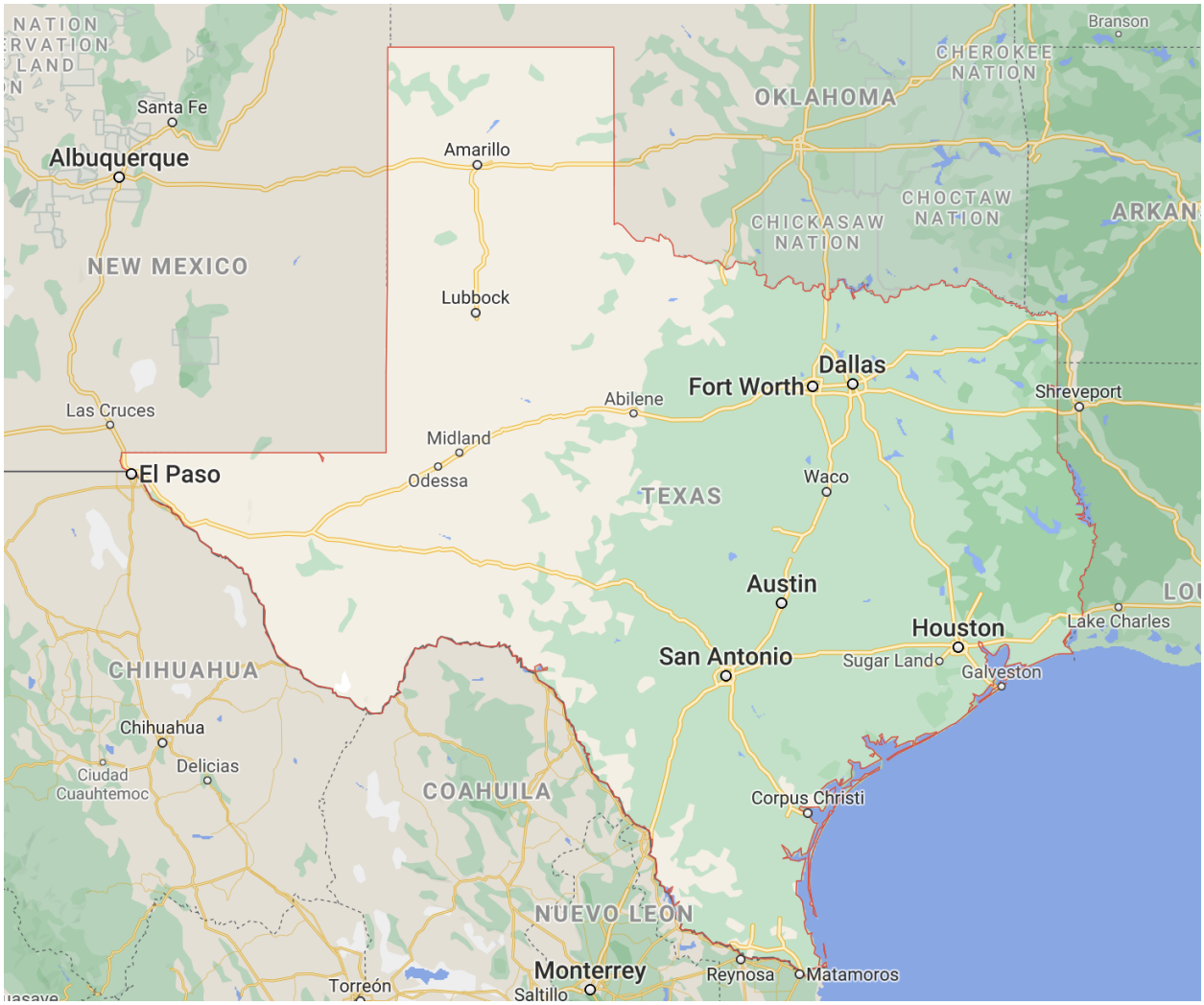
1 Business Case



An investment group would like to invest in single family homes in 5 unique zip codes. They have a desired hold period of 1 to 3 years. With low interest rates, they would like to lock in purchases in the immediate to near future.

I have the breadth of the entire country but have decided to hone in on Texas due to its favorable tax code and robust in place and emerging industries. Texas boasts the largest oil and gas production in the US and head quarters for 24 Fortune 500 companies. Austin, and other major metros, are becoming popular landing spots for tech entrepreneurs as opposed to the Bay Area.

Over the past 10 years, the Texas population has grown 16%, the 3rd largest of any state. The population rate continues to grow strongly with new migration from surrounding states. Austin has over 400 people moving to the city each day. With historically low housing supply after the 2008 housing crisis and an environment with tight labor supply and high commodity costs, transactions are happening at record speeds. Houses no longer sit on the market for months on end, it's a sellers market.

Rather than view Texas without borders, I decided to hone in the top 10 metros by population. I am looking for quality locations that have shown exemplary short term/medium term/long term growth. I also want locations that are resiliant to macroeconomic factors like the 2008 housing bust.

The remainder of this noteboook will show my thought process for how I honed in on the top 5 zip codes in Texas.

In [1]:
```
import pandas as pd
import numpy as np
```
executed in 428ms, finished 09:13:16 2021-06-20

In [2]:
```
df = pd.read_csv('zillow_data.csv')
```
executed in 306ms, finished 09:13:16 2021-06-20

In [3]:
```
#Make sure data was imported correctly

df.head()
```
executed in 16ms, finished 09:13:16 2021-06-20

Out[3]:

	RegionID	RegionName	City	State	Metro	CountyName	SizeRank	1996-04	1996-05
0	84654	60657	Chicago	IL	Chicago	Cook	1	334200.0	335400.0
1	90668	75070	McKinney	TX	Dallas-Fort Worth	Collin	2	235700.0	236900.0
2	91982	77494	Katy	TX	Houston	Harris	3	210400.0	212200.0
3	84616	60614	Chicago	IL	Chicago	Cook	4	498100.0	500900.0
4	93144	79936	El Paso	TX	El Paso	El Paso	5	77300.0	77300.0

5 rows × 272 columns

2 Step 2: Data Preprocessing

In [4]:
```
#Helpter function to pivot from wide to long format

def melt_data(df):
    melted = pd.melt(df, id_vars=['RegionName', 'City', 'State', 'Metro
    melted['time'] = pd.to_datetime(melted['time'], infer_datetime_form
    melted = melted.dropna(subset=['value'])
    return melted#melted.groupby('time').aggregate({'value':'mean'})
```
executed in 2ms, finished 09:13:16 2021-06-20

In [5]:
```
# Drop columns that aren't included in the helper function

df.drop(columns=['RegionID','SizeRank'], axis=1, inplace=True)
```
executed in 10ms, finished 09:13:16 2021-06-20

In [6]:
```
# Ensure data is melted

df_melted = melt_data(df)
df_melted.head(5)
```
executed in 1.55s, finished 09:13:17 2021-06-20

Out[6]:

	RegionName	City	State	Metro	CountyName	time	value
0	60657	Chicago	IL	Chicago	Cook	1996-04-01	334200.0
1	75070	McKinney	TX	Dallas-Fort Worth	Collin	1996-04-01	235700.0
2	77494	Katy	TX	Houston	Harris	1996-04-01	210400.0
3	60614	Chicago	IL	Chicago	Cook	1996-04-01	498100.0
4	79936	El Paso	TX	El Paso	El Paso	1996-04-01	77300.0

In [7]:
```
# Set DataFrame to only show Texas values

df_texas = df_melted[df_melted['State']=='TX']
```
executed in 231ms, finished 09:13:18 2021-06-20

In [8]:
```
# Set index

df_texas.set_index('time', inplace=True)
```
executed in 2ms, finished 09:13:18 2021-06-20

In [9]:
```
# Set index as datetime object

df_texas.index = pd.to_datetime(df_texas.index)
```
executed in 13ms, finished 09:13:18 2021-06-20

3 Business Exploration

I want to find counties that are the most resistant to marcoeconomic downturns. Ideally, the county has a strong economy and businesses that continue to grow in the face of a negative backdrop.

I chose to analyze how the top 10 metros in Texas responded to the 2008 housing bust. I analyzed how long each counties' respective zip codes took to reach price levels before the housing crisis. The longer a zip code took to recover, the less resiliant I considered it to be to negative macroeconomic factors.

I judged recovery time based on the mean, median, and standard deviation for zip codes recovery time. It is is important to have quick recovery but is also important that the spread is tight so the investor does not got caught holding the home in the zip code that is usually quick to recover but can also be the slowest.

3.1 2008 Housing Bubble Recovery Time

- Defining functions for analyzing reovery times

In [10]:
```
def df_maker(df, county):
    """
    Takes a DataFrame and returns sliced version only including specifi
    ***Parameters:***
    DataFrame, country
    ***Returns:***
    DataFrame sliced to only include specified county
    """
    df = df_texas[df_texas['CountyName']==county].copy()
    return df
```

executed in 2ms, finished 09:13:18 2021-06-20

In [11]:
```
def county_dict_maker(df):
    """
    Takes a DataFrame and returns a dictionary where keys are zip codes
    are DataFrames with specified zip codes and median home values
    ***Parameters:***
    DataFrame
    ***Returns:***
    Dictionary with zip code keys and DataFrame values
    """
    df.drop(columns=['City', 'State', 'Metro', 'CountyName'], axis=1, i
    zips=list(df['RegionName'].unique())
    dict_full = {}
    for zip_ in zips:
        dict_full[zip_] = df[df['RegionName']==zip_]
    return dict_full
```

executed in 3ms, finished 09:13:18 2021-06-20

In [12]:

```
# Try except because certain zip codes do not full into time constra
def housing_bust_finder(df, dict_full):
    """
    Takes a DataFrame and dictionary and returns a DataFrame where each
    the number of days it took to reach the max price between 2005 and
    showing the frequency of number of days to reach previous maximum v
    ***Parameters:***
    DataFrame and dictionary
    ***Returns:***
    DataFrame where each zipcode specifies the number of days until rec
    keys are frequency and values are days until recovery
    """
    # Create list of unique zip codes in the region
    zips=list(df['RegionName'].unique())

    # Create dictionary where keys is zip code and value is days until
    # Use try/except because certain zip codes will not correspond with
    recover_housing_dict = {}
    for zip_ in zips:
        for zip_, vals in dict_full.items():
            try:
                idx_max_pre = dict_full[zip_]['2005-01-01':'2008-12-31'
                val_max_pre = dict_full[zip_]['2005-01-01':'2008-12-31'
                idx_max_post=(dict_full[zip_]['2008-12-30':]['value']>=
                td = idx_max_post - idx_max_pre
                days = td.days
                recover_housing_dict[zip_]=days
            except ValueError:
                pass

    # Create dictionary where key is number of days until recovery and
    recover_housing_counts={}
    for k, v in recover_housing_dict.items():
        if v in recover_housing_counts.keys():
            recover_housing_counts[v] +=1
        else:
            recover_housing_counts[v] = 1

    # Swap keys and values and return the above dictionary in the form
    housing_recover=pd.Series(recover_housing_counts)
    housing_recover=pd.Series(housing_recover.index.values, index=housi
    df_housing_recover=pd.DataFrame(housing_recover)
    df_housing_recover.rename(columns={0:'Days_Until_Recovery'}, inplac
    df_housing_recover.sort_values(by='Days_Until_Recovery', ascending=
    return df_housing_recover, recover_housing_dict
```

executed in 5ms, finished 09:13:18 2021-06-20

In [13]:
```
df_texas['CountyName'].value_counts(1).nlargest(10)
```
executed in 26ms, finished 09:13:18 2021-06-20

Out[13]:
```
Harris        0.125698
Dallas        0.071883
Tarrant       0.061291
Bexar         0.057135
Travis        0.046747
Denton        0.024932
Collin        0.024261
El Paso       0.021004
Montgomery    0.020776
McLennan      0.015857
Name: CountyName, dtype: float64
```

- Create data frames of the top 10 major metros
- Create dictionaries of metros where each key holds a dataframe which is a time series of the respective zip code

In [14]:
```
# Create Harris county DataFrame
df_Harris = df_maker(df_texas, 'Harris')

# Create Harris county dictionary
Harris_dict_full = county_dict_maker(df_Harris)

# Create Harris county recovery dataframe and dictionary
df_Harris_days, dict_Harris_zips = housing_bust_finder(df_Harris,Harris
```
executed in 24.8s, finished 09:13:42 2021-06-20

Create Dallas county data

In [15]:
```
# Create Dallas county DataFrame
df_Dallas = df_maker(df_texas, 'Dallas')

# Create Dallas county dictionary
Dallas_dict_full = county_dict_maker(df_Dallas)

# Create Dallas county recovery dataframe and dictionary
df_Dallas_days, dict_Dallas_zips = housing_bust_finder(df_Dallas,Dallas
```
executed in 8.15s, finished 09:13:51 2021-06-20

In [16]:
```
# Create Travis county DataFrame
df_Travis = df_maker(df_texas, 'Travis')

# Create Travis county dictionary
travis_dict_full = county_dict_maker(df_Travis)

# Create Travis county recovery dataframe and dictionary
df_Travis_days, dict_Travis_zips = housing_bust_finder(df_Travis,travis
```
executed in 3.43s, finished 09:13:54 2021-06-20

Create Tarrant county data

In [17]:
```
# Create Tarrant county DataFrame
df_Tarrant = df_maker(df_texas, 'Tarrant')

# Create Tarrant county dictionary
Tarrant_dict_full = county_dict_maker(df_Tarrant)

# Create Tarrant county recovery dataframe and dictionary
df_Tarrant_days, dict_Tarrant_zips = housing_bust_finder(df_Tarrant,Tar
```

executed in 5.86s, finished 09:14:00 2021-06-20

Create Bexar county data

In [18]:
```
# Create Bexar county DataFrame
df_Bexar = df_maker(df_texas, 'Bexar')

# Create Bexar county dictionary
Bexar_dict_full = county_dict_maker(df_Bexar)

# Create Bexar county recovery dataframe and dictionary
df_Bexar_days, dict_Bexar_zips = housing_bust_finder(df_Bexar,Bexar_dic
```

executed in 5.11s, finished 09:14:05 2021-06-20

Create Denton county data

In [19]:
```
# Create Denton county DataFrame
df_Denton = df_maker(df_texas, 'Denton')

# Create Denton county dictionary
Denton_dict_full = county_dict_maker(df_Denton)

# Create Denton county recovery dataframe and dictionary
df_Denton_days, dict_Denton_zips = housing_bust_finder(df_Denton,Denton
```

executed in 1.01s, finished 09:14:06 2021-06-20

Create Collin county data

In [20]:
```
# Create Collin county DataFrame
df_Collin = df_maker(df_texas, 'Collin')

# Create Collin county dictionary
Collin_dict_full = county_dict_maker(df_Collin)

# Create Collin county recovery dataframe and dictionary
df_Collin_days, dict_Collin_zips = housing_bust_finder(df_Collin,Collin
```

executed in 1.02s, finished 09:14:07 2021-06-20

Create El Paso county data

In [21]:
```
# Create El Paso county DataFrame
df_El_Paso = df_maker(df_texas, 'El Paso')

# Create El Paso county dictionary
El_Paso_dict_full = county_dict_maker(df_El_Paso)

# Create El Paso county recovery dataframe and dictionary
df_El_Paso_days, dict_El_Paso_zips = housing_bust_finder(df_El_Paso,El_
```

executed in 769ms, finished 09:14:08 2021-06-20

Create Montgomery county data

In [22]:
```
# Create Montgomery county DataFrame
df_Montgomery = df_maker(df_texas, 'Montgomery')

# Create Montgomery county dictionary
Montgomery_dict_full = county_dict_maker(df_Montgomery)

# Create Montgomery county recovery dataframe and dictionary
df_Montgomery_days, dict_Montgomery_zips = housing_bust_finder(df_Montg
```

executed in 738ms, finished 09:14:09 2021-06-20

Create McLennan county data

In [23]:
```
# Create McLennan county DataFrame
df_McLennan = df_maker(df_texas, 'McLennan')

# Create McLennan county dictionary
McLennan_dict_full = county_dict_maker(df_McLennan)

# Create McLennan county recovery dataframe and dictionary
df_McLennan_days, dict_McLennan_zips = housing_bust_finder(df_McLennan,
```

executed in 429ms, finished 09:14:09 2021-06-20

I am subdiving the metros into primary and secondary markets. Primary markets tend to have higher valuations and more name recognition. Secondary markets have more favorable prices but still have high quality assets.

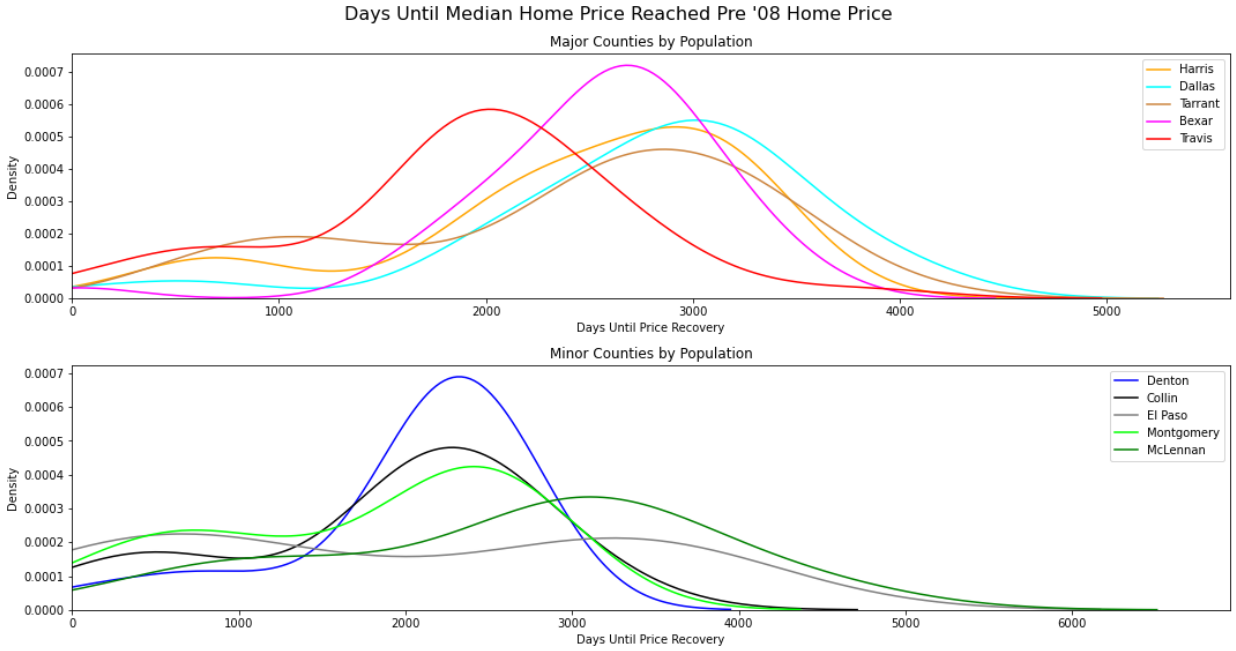
I would like to invest in a combination of primary and secondary markets so I am more diversified. The primary markets have high lower cap rates so there is more interest rate risk exposure. The secondard markets have higher cap rates so there is more flexibility if interest rates rise.

Primary and secondary markets have strong demand for tenants. Exposure to both minimizes risk. Out of the 5 counties per subdivision, I will select the top 3 from each based on recovery time.

In [24]:

```
import seaborn as sns
import matplotlib.pyplot as plt


fig, axes = plt.subplots(figsize=(15,8), nrows=2)
fig.suptitle('Days Until Median Home Price Reached Pre \'08 Home Price'
sns.kdeplot(data=df_Harris_days, ax=axes[0],label='Harris', palette=['o
sns.kdeplot(data=df_Dallas_days, ax=axes[0],label='Dallas', palette=['c
sns.kdeplot(data=df_Tarrant_days, ax=axes[0],label='Tarrant', palette=[
sns.kdeplot(data=df_Bexar_days, ax=axes[0],label='Bexar', palette=['mag
sns.kdeplot(data=df_Travis_days, ax=axes[0],label='Travis', palette=['r
axes[0].set_xlabel('Days Until Price Recovery')
axes[0].set_title('Major Counties by Population')
sns.kdeplot(data=df_Denton_days, ax=axes[1],label='Denton', palette=['b
sns.kdeplot(data=df_Collin_days, ax=axes[1],label='Collin', palette=['b
sns.kdeplot(data=df_El_Paso_days, ax=axes[1],label='El Paso', palette=[
sns.kdeplot(data=df_Montgomery_days, ax=axes[1],label='Montgomery', pal
sns.kdeplot(data=df_McLennan_days, ax=axes[1],label='McLennan', palette
axes[1].set_xlabel('Days Until Price Recovery')
axes[1].set_title('Minor Counties by Population')
axes[0].set_xlim(left=0)
axes[1].set_xlim(left=0)
axes[0].legend()
axes[1].legend()
fig.tight_layout();
```

executed in 1.04s, finished 09:14:10 2021-06-20



In [25]:
```
big_counties_names = ['Harris','Dallas','Tarrant','Bexar','Travis']
big_counties_df = [df_Harris_days, df_Dallas_days, df_Tarrant_days, df_
big_recovery_agg = {}
for name, df in zip(big_counties_names, big_counties_df):
    big_recovery_agg[name] = df['Days_Until_Recovery'].agg(['mean', 'me
big_county_recovery = pd.DataFrame(big_recovery_agg).T
big_county_recovery.sort_values(by=['median', 'mean', 'std', 'max'])
```

executed in 12ms, finished 09:14:10 2021-06-20

Out[25]:

	mean	median	std	max
Travis	1879.512821	1947.0	790.199304	3835.0
Tarrant	2405.063830	2619.0	926.935681	3986.0
Bexar	2537.119048	2646.5	611.639189	3592.0
Harris	2432.412500	2662.5	846.344899	3622.0
Dallas	2767.155556	2953.0	835.392748	4079.0

- **Travis:** Selected due to lowest mean/median
- Tarrant: Not selected due to high standard deviation
- **Bexar:** Selected due to moderate mean/median and lowest standard deviation
- **Harris:** Selected due to superior metrics to Dallas
- Dallas: Not selected

In [26]:
```
small_counties_names = ['Denton','Collin','El_Paso','Montgomery','McLen
small_counties_df = [df_Denton_days, df_Collin_days, df_El_Paso_days, d
small_recovery_agg = {}
for name, df in zip(small_counties_names, small_counties_df):
    small_recovery_agg[name] = df['Days_Until_Recovery'].agg(['mean', '
small_county_recovery = pd.DataFrame(small_recovery_agg).T
small_county_recovery.sort_values(by=['median', 'mean', 'std', 'max'])
```

executed in 12ms, finished 09:14:10 2021-06-20

Out[26]:

	mean	median	std	max
El_Paso	1922.615385	1704.0	1416.997032	3623.0
Collin	1867.875000	2115.5	914.517058	3134.0
Montgomery	1749.888889	2176.5	931.835563	2800.0
Denton	2031.809524	2253.0	723.378782	2769.0
McLennan	2643.866667	3011.0	1165.708238	4473.0

- **El Paso:** Selected due to lowest mean/median, however high standard deviation
- **Collin:** Selected due to low median and low standard deviation
- Montgomery: Not selected due to high standard deviation
- **Denton:** Selected due to similar median/mean as Montgomery but lower standard deviation
- McLennan: Not selected

3.2 Top Growth Prospects

- Based on recovery time further inspecting: Travis, Bexar, Harris, El Paso, Collin, and Denton county
- Looking for zip codes that have shown higher than average growth over the past 2 years, 5 years, and 10 years
- Due to a 1 to 3 year hold period, I am placing a greater emphasis on short term than long term growth prospects
- I have created a 'cumulative' score which models growth such that: 40% of score based on 2-year return, 35% of score based on 5-year return, and 25% of score based on 10-year return
    - Using this metric, I have an agnostic way to select the top 3 zip codes per county to further analyze

In [27]:

```
def test_county_zip_perc(df_county, county_dict_full):
    """
    Takes a DataFrame and dictionary and returns a DataFrame where each
    the 2-year, 5-year, 10-year/Std/Cumulative Score
    Parameters:
    DataFrame and dictionary
    Returns:
    DataFrame where each row is a zip code and values correspond to cum
    and standard deviation
    """
    county_dict_annual = {}
    county_zips = list(county_dict_full.keys())
    for zip_ in county_zips:
        county_dict_annual[zip_] = df_county[df_county['RegionName']==z
        county_dict_annual[zip_]['2_Year_Pct_Change'] = county_dict_ann
        county_dict_annual[zip_]['5_Year_Pct_Change'] = county_dict_ann
        county_dict_annual[zip_]['10_Year_Pct_Change'] = county_dict_an
        county_dict_annual[zip_]['Std'] = county_dict_annual[zip_]['val

    county_comparison = {}
    county_zips = list(county_dict_annual.keys())
    for zip_ in county_zips:
        for zip_, vals in county_dict_annual.items():
            county_comparison[zip_] = county_dict_annual[zip_].iloc[-1]
    df = pd.DataFrame(county_comparison).T
    df.drop(columns='RegionName', inplace=True)
    df['Cumulative_Score'] = df['2_Year_Pct_Change']*(0.4)+df['5_Year_P
    df.sort_values(by='Cumulative_Score', inplace=True, ascending=False
    return df
```

executed in 5ms, finished 09:14:10 2021-06-20

In [28]:
```
# Gather the names of all of the counties and their respective dictiona
# dataframes to plug into a for loop

county_name_by_recovery = ['Travis', 'Bexar', 'Harris', 'El_Paso', 'Col
county_df_by_recovery = [df_Travis, df_Bexar, df_Harris, df_El_Paso, df
county_dict_by_recovery = [travis_dict_full, Bexar_dict_full, Harris_di
                           El_Paso_dict_full, Collin_dict_full, Denton_

all_perc_change_dict = {}
for name, df_, dict_ in zip(county_name_by_recovery, county_df_by_recov
    all_perc_change_dict[name] = test_county_zip_perc(df_, dict_)
```

executed in 2.47s, finished 09:14:13 2021-06-20

In [29]:
```
all_perc_change_dict.keys()
```

executed in 3ms, finished 09:14:13 2021-06-20

Out[29]:
```
dict_keys(['Travis', 'Bexar', 'Harris', 'El_Paso', 'Collin', 'Denton'])
```

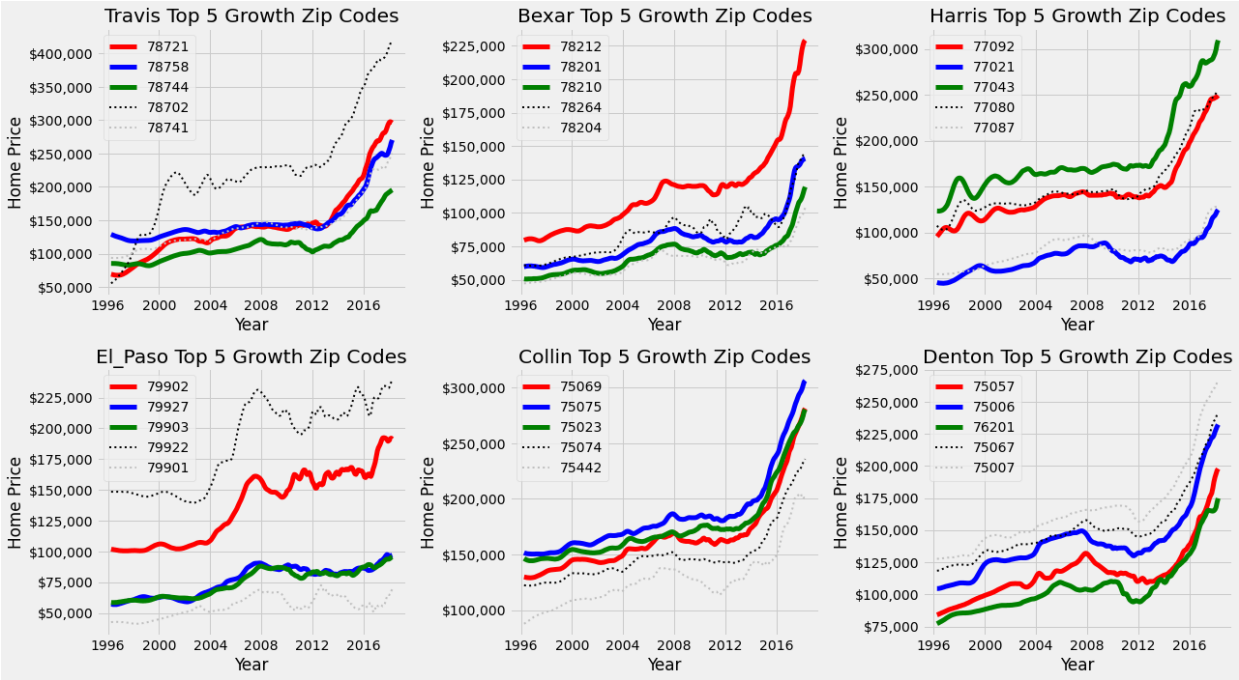
In [ ]:
```

```

executed in 1ms, finished 09:22:08 2021-06-17

In [30]:

```
import matplotlib.ticker as mtick
county_name = ['Travis', 'Bexar', 'Harris', 'El_Paso', 'Collin', 'Dento
county_dict_full = [travis_dict_full, Bexar_dict_full, Harris_dict_full
                            El_Paso_dict_full, Collin_dict_full, Denton_

with plt.style.context('fivethirtyeight'):
    fig, axes = plt.subplots(nrows=2, ncols=3, figsize=(18,10))
    for ax, name, dict_ in zip(axes.flatten(), county_name, county_dict
        top_5=list(all_perc_change_dict[name].index)[:5]
        fmt = '${x:,.0f}'
        tick = mtick.StrMethodFormatter(fmt)
        ax.yaxis.set_major_formatter(tick)
        sns.lineplot(data=dict_[top_5[0]]['value'], color='red', label=
        sns.lineplot(data=dict_[top_5[1]]['value'], color='blue', label
        sns.lineplot(data=dict_[top_5[2]]['value'], color='green', labe
        sns.lineplot(data=dict_[top_5[3]]['value'], color='black', labe
        sns.lineplot(data=dict_[top_5[4]]['value'], color='silver', lab
        ax.set_title(f"{name} Top 5 Growth Zip Codes")
        ax.set_ylabel('Home Price')
        ax.set_xlabel('Year')
        ax.legend()
        fig.tight_layout()
```

executed in 1.54s, finished 09:14:14 2021-06-20



- Visualization showing the top 5 growth zip codes per county
- The colored lines are the 3 with the highest cumulative score (ordered: red, blue, green)
- The grey and black lines represent the 4th and 5th highest scores which will not be analyzed
- As displayed, 2 year growth is the most highly desired characteristic for the zip code

In [31]:
```
# Create a dictionary where each county is the key and the top 3 zipcod

county_name = ['Travis', 'Bexar', 'Harris', 'El_Paso', 'Collin', 'Dento
top_zip_per_county = {}
for county in county_name:
    top_zip_per_county[county]=list(all_perc_change_dict[county].index)

top_zip_per_county
```

executed in 4ms, finished 09:14:14 2021-06-20

Out[31]:
```
{'Travis': [78721, 78758, 78744],
 'Bexar': [78212, 78201, 78210],
 'Harris': [77092, 77021, 77043],
 'El_Paso': [79902, 79927, 79903],
 'Collin': [75069, 75075, 75023],
 'Denton': [75057, 75006, 76201]}
```

In [32]:
```
# Make a list of all of the top zip codes (18 total)

total_top_zips = []
for k, v in top_zip_per_county.items():
    total_top_zips.extend(v)

total_top_zips[:5]
```

executed in 3ms, finished 09:14:14 2021-06-20

Out[32]:
```
[78721, 78758, 78744, 78212, 78201]
```

4 Step 5: SARIMA Modeling

- Showing all helper modeling functions

In [33]:
```
from statsmodels.tsa.statespace.sarimax import SARIMAX
from pmdarima import auto_arima

def find_auto_order(df_train):
    """
    Takes a DataFrame and returns ARMA and Seasonal order based on auto
    ***Parameters:***
    Train data
    ***Returns:***
    Summary of auto_arima findings and ARMA/Seasonal order
    """
    model = auto_arima(df_train,start_p=1, start_q=1, max_p=4, max_q=4,
    start_Q=1, max_d=3, max_D=3, max_Q=4, max_P=4,seasonal=True, m=12,
    summary = model.summary()
    ARMA_order = model.order
    seasonal_order = model.seasonal_order
    return summary, ARMA_order, seasonal_order
```

executed in 728ms, finished 09:14:15 2021-06-20

In [34]:

```
# def SARIMAX_man_results(train_data, arma_order, seasonal_order):
#     """
#     Takes a training data and ARMA/Seasonal order and fits a SARIMAX
#     different parameters and returns the best SARIMAX model based on
#     Parameters:
#     Train data, seasonal/arma order
#     Returns:
#     Dictionary with various iterations of parameters and their AIC sc
#     parameters
#     """
#     model = SARIMAX(train_data,order=arma_order, seasonal_order=seasc
#                 enforce_stationarity=False, enforce_invertibility=Fal
#                 freq='MS', k_trend=2)
#     results = model.fit()
#     score_ = results.aic



#     diagnostics = results.plot_diagnostics(figsize=(8,8))
#     results_dict['diag_summary'] = diagnostics

#     best_params['arma_order'] = arma_order
#     best_params['seasonal_order'] = seasonal_order



#     return results_dict, best_params
```

executed in 2ms, finished 09:14:15 2021-06-20

In [35]:

```
def fit_final_model(df_train, arma_order, seasonal_order):
    """
    Fit final model according to results from SARIMAX grid search
    ***Parameters:***
    Train data, ARMA/Seasonal order, MLE_regression, and concentrate sc
    ***Returns:***
    Fit model on traning data
    """
    model = SARIMAX(df_train,order=arma_order,seasonal_order=seasonal_o
                    enforce_stationarity=False, freq='MS')
    best_model = model.fit(maxiter=150)
    diagnostics = best_model.plot_diagnostics(figsize=(8,8))
    return best_model
```

executed in 2ms, finished 09:14:15 2021-06-20

In [36]:
```
def get_forecast(best_model, test_data):
    """
    Takes a fitted model and returns dataframe where index is a time an
    and predicted mean
    ***Parameters:***
    Fitted model and test data
    ***Returns:***
    DataFrame showing predictions to compare to test data
    """
    forecast = best_model.get_forecast(steps=len(test_data))

    ## save forecasted mean and upper/lower ci as df
    forecast_df = forecast.conf_int()
    forecast_df.columns = ['Lower CI','Upper CI']
    forecast_df['Forecast'] = forecast.predicted_mean
    forecast_df
    return forecast_df
```

executed in 2ms, finished 09:14:15 2021-06-20

In [37]:
```
def plot_forecast(train_data, test_data, prediction_table, code, county
    """
    Plot forecasted predictions as opposed to test data
    ***Parameters:***
    Training data, test data, predictions, and zip code
    ***Returns:***
    Plot comparing predictions based off fitted model to test data
    """
    with plt.style.context('fivethirtyeight'):
        fig,ax = plt.subplots(figsize=(10,4))

        # Plotting Training and test data
        fmt = '${x:,.0f}'
        tick = mtick.StrMethodFormatter(fmt)
        ax.yaxis.set_major_formatter(tick)
        train_data['2014-01-01':].plot(label='Training Data')
        test_data['2014-01-01':].plot(label='Test Data')

        ## Plotting Forefcast and CI
        prediction_table['Forecast'].plot(ax=ax,label='Forecast')
        ax.fill_between(prediction_table.index,
                        prediction_table['Lower CI'],
                        prediction_table['Upper CI'],color='g',alpha=0.

        ax.set(ylabel=f'{code} Home Value')
        ax.legend()
        ax.set_title(f"{county} County - {code} Current Predictions")

        return fig;
```

executed in 4ms, finished 09:14:15 2021-06-20

In [38]:

```
def fitall_final_model(df_all, arma_order, seasonal_order):
    """
    Fits final model on all data using specified parameters
    ***Parameters:***
    Zip code dataframe, final model best parameters
    ***Returns:***
    Best fit model on all data
    """
    model = SARIMAX(df_all,order=arma_order,seasonal_order=seasonal_ord
                    enforce_stationarity=False, freq='MS')
    bestall_model = model.fit()
    return bestall_model
```

executed in 2ms, finished 09:14:15 2021-06-20

In [39]:

```
# Rerun on entire dataset to get further forecasts
import matplotlib.ticker as mtick

def plot_future_forecast(final_model, test_data, train_data, code, coun
    """
    Plot showing traing data, test data, and future predictions out to
    ***Parameters:***
    Final model, test data, training data, zip code, and county
    ***Returns:***
    Plot showing training, test, and future values. Dictionary of final
    confidicene interval. Dataframe showing all future predictions
    """
    with plt.style.context('fivethirtyeight'):

        ## Get forecast
        forecast = final_model.get_forecast(steps=len(test_data))

        ## save forecasted mean and upper/lower ci as df
        forecast_dict={}
        forecast_df = forecast.conf_int()
        forecast_df.columns = ['Lower CI','Upper CI']
        forecast_df['Forecast'] = forecast.predicted_mean
        forecast_dict['low'] = forecast_df.iloc[-1]['Lower CI']
        forecast_dict['high'] = forecast_df.iloc[-1]['Upper CI']
        forecast_dict['mean'] = forecast_df.iloc[-1]['Forecast']

        ## Plot
        last_n_lags=52

        fig,ax = plt.subplots(figsize=(10,4))


        # Plotting Training and test data
        fmt = '${x:,.0f}'
        tick = mtick.StrMethodFormatter(fmt)
        ax.yaxis.set_major_formatter(tick)
        ax.plot(train_data['2012-01-01':], label='Training Data')
        ax.plot(test_data['2012-01-01':], label='Test Data')
        ax.axvline(train_data.index[-1],ls=':')
        ax.set_xlabel('Year')
        ax.set_ylabel('Home Value')
        ax.set_title(f"{county} County - {code} Future Predictions")


        ## Plotting Forefcast and CI
        ax.plot(forecast_df['Forecast'], label='Forecast')
        ax.fill_between(forecast_df.index,
                        forecast_df['Lower CI'],
                        forecast_df['Upper CI'],color='g',alpha=0.3)
        ax.legend(loc=2)
        return fig, forecast_dict, forecast_df
```

executed in 6ms, finished 09:14:15 2021-06-20

In [40]:
```
def roi_dict(forecast_df):
    """
    Dictionary that shows percent change from end of dataset to forecas
    initial $100,000 investment and the dollar difference between princ
    ***Parameters:***
    DataFrame of forecasted predictions
    ***Returns:***
    Dictionary with returns on the upper/lower/and mean range
    """
    forecast_dict = {}
    cols = ['Lower CI', 'Upper CI', 'Forecast']
    for col in cols:
        perc_change=(forecast_df[col][-1]-forecast_df[col][0])/forecast
        dollar_val=(perc_change+1)*100000
        dollar_dif = dollar_val-100000
        forecast_dict[col]=[]
        forecast_dict[col].append(perc_change)
        forecast_dict[col].append(dollar_val)
        forecast_dict[col].append(dollar_dif)
    return forecast_dict
```

executed in 3ms, finished 09:14:15 2021-06-20

In [41]:
```
# Bring in ticker formats
import matplotlib.ticker as mtick

def model_predictions(train_data, test_data, all_data, code, county):
    """
    Uses all helper functions to create diagnostic summary, forecast on
    Parameters:
    Training data, test data, all data, zip code, and county
    Returns:
    Plot showing diagnostics and forecasts. Dictinary with predictions
    """
    summary, arma_order, seasonal_order = find_auto_order(train_data)
#     best_params = SARIMAX_man_results(train_data, arma_order, seasona
    best_model = fit_final_model(train_data, arma_order, seasonal_order
    forecast_df=get_forecast(best_model, test_data)
    test_forecast_fig=plot_forecast(train_data, test_data, forecast_df,
    bestall_model=fitall_final_model(all_data, arma_order, seasonal_ord

    fig, forecast_dict, forecast_df=plot_future_forecast(bestall_model,
    roi_helper = roi_dict(forecast_df)
    return fig, forecast_dict, forecast_df, roi_helper
```

executed in 3ms, finished 09:14:15 2021-06-20

4.1 Travis County Modeling

- Capitol of Texas
- Located in Austin Metro
- Population: 1,273,954
- Median home value of $270,400 as of April 2018

4.1.1 78758: EDA and SARIMAX

- North Austin
- 4 miles from downtown Austin

In [42]:
```
def create_zip_data (dict_full, zip_):
    """
    Takes a county dictionary and returns dataframe with values for spe
    Parameters:
    Dictionary and zip code
    Returns:
    DataFrame which shows value and frequency is corrected to monthly d
    """
    df = dict_full[zip_].copy()
    df.drop(columns='RegionName', axis=1, inplace=True)
    df=df.resample('MS').asfreq()
    return df
```

executed in 2ms, finished 09:14:15 2021-06-20

In [43]:
```
df_78758=create_zip_data(travis_dict_full, 78758)
```

executed in 6ms, finished 09:14:15 2021-06-20

In [44]:

```
def zip_eda(df_full, code, county):
    """
    Takes a zip code and returns historic prices, price distributions,
    box represents a year of data to better assess spread of prices
    Parameters:
    DataFrame, zip code, and county
    Returns:
    Summary figure and descriptive statistics per zip code
    """

    from matplotlib.gridspec import GridSpec
    with plt.style.context('bmh'):

        fig = plt.figure(figsize=(19, 7), constrained_layout=False)
        gs = GridSpec(nrows=2, ncols=2)
        # First axes

        ax0 = fig.add_subplot(gs[0, 0])
        fmt = '${x:,.0f}'
        tick = mtick.StrMethodFormatter(fmt)
        ax0.yaxis.set_major_formatter(tick)
        ax0.plot(df_full['value'], color='r')
        ax0.set_title(f'{code} Price History')
        ax0.set_xlabel('Year')
        ax0.set_ylabel('Home Price')
        ax0.axvline(x='2000-01-01', label='Dotcom Boom', ls=':')
        ax0.axvline(x='2008-01-01', label='Housing Boom', ls=':', color
        ax0.legend()
        # Second axes
        ax1 = fig.add_subplot(gs[1, 0])
        ax1.xaxis.set_major_formatter(tick)
        sns.kdeplot(df_full['value'], ax=ax1, color='r')
        ax1.set_title(f'{code} Price Distribution')
        ax1.set_xlabel('Home Price')

        mean=df_full['value'].mean()
        median=df_full['value'].median()
        ax1.axvline(x=mean, label=f"Mean: ${round(mean,2)}", ls=':', co
        ax1.axvline(x=median, label=f"Median: ${round(median,2)}", ls='
        ax1.legend()
        # Third axes

        groups = df_full['1997-01-01':'2017-12-12'].groupby(pd.Grouper(
        df_annual=pd.DataFrame()
        for name, group in groups:
            df_annual[name.year]=group.values.ravel()
        ax2 = fig.add_subplot(gs[:, 1])
        ax2.yaxis.set_major_formatter(tick)
        sns.boxplot(data=df_annual, ax=ax2)
        xticklabels = (range(1997,2018))
        ax2.set_xlabel('Year')
        ax2.set_xticklabels(xticklabels, rotation = 45, ha="right")
        ax2.set_title('Yearly Distribution of Median Home Prices')

        fig.suptitle(f'{county} County - {code}', fontsize=20)
        fig.tight_layout();
```

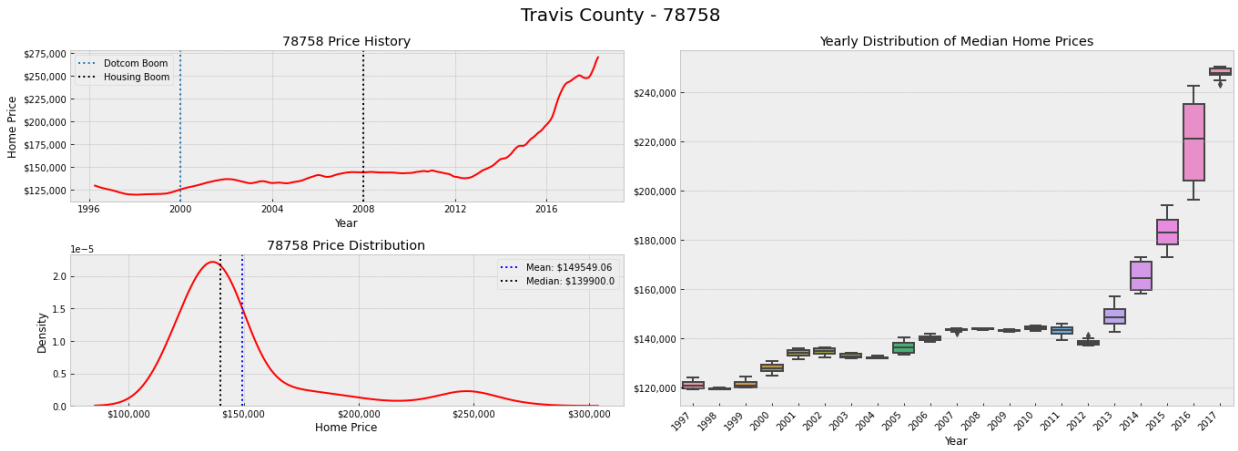
```
        res = df_full.describe()
        return fig, res;
```

executed in 7ms, finished 09:14:15 2021-06-20

In [45]:
```
zip_eda(df_78758, 78758, 'Travis')
```

executed in 597ms, finished 09:14:15 2021-06-20

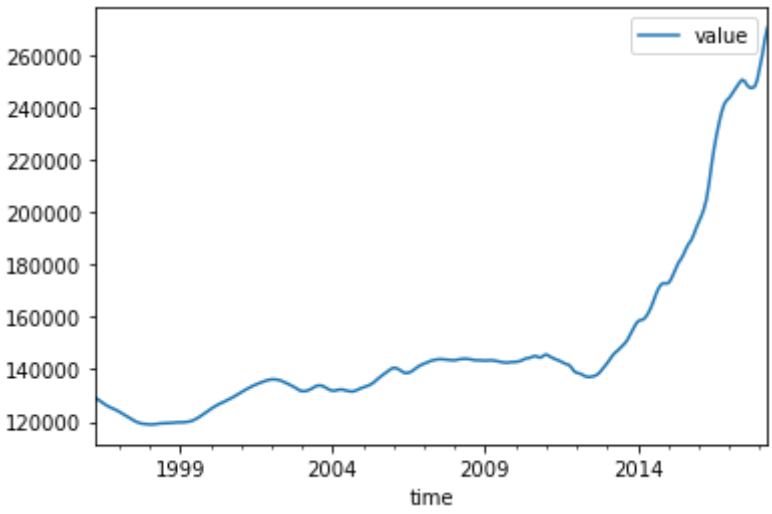
Out[45]:
```
(<Figure size 1368x504 with 3 Axes>,
                value
count      265.000000
mean    149549.056604
std      34194.863952
min     119000.000000
25%     131900.000000
50%     139900.000000
75%     145000.000000
max     270400.000000)
```



- Minimal impact on prices during dotcom crash and housing bust
- Data is right skewed based on mean > median
- From 1997 to 2012 homes have minimal spread but begin to increase after that with significantly larger spread in 2016

In [46]:
```
# Plot data to determine accurate split point

df_78758.plot()
```
executed in 108ms, finished 09:14:16 2021-06-20

Out[46]: <AxesSubplot:xlabel='time'>

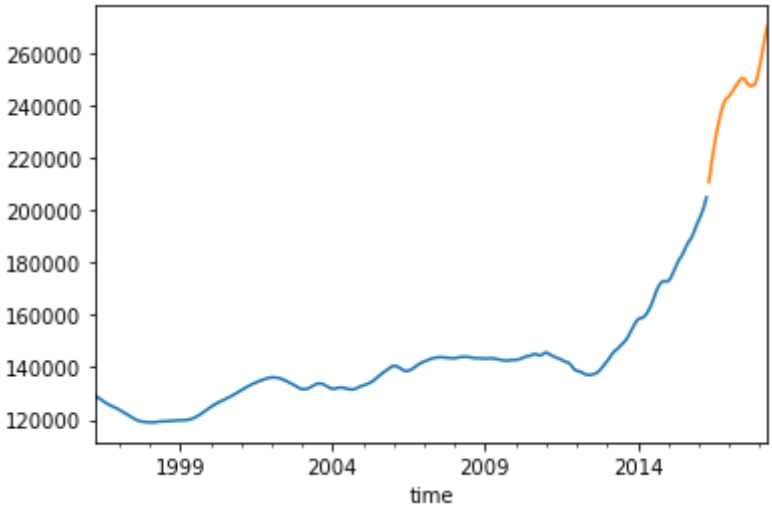


In [47]:
```
# int acts like floor division

def create_train_test_split(df_full, split):
    """
    Takes a DataFrame and split value and divides data into train/test
    to training
    Parameters:
    DataFrame and split value
    Returns:
    Train and test data with size specified by split value
    """
    total_rows=len(df_full)
    train = df_full.iloc[:int(total_rows*split)]['value']
    test = df_full.iloc[int(total_rows*split):]['value']
    return train, test
```
executed in 2ms, finished 09:14:16 2021-06-20

In [48]:
```
train_78758, test_78758 = create_train_test_split(df_78758, 0.91)
train_78758.plot()
test_78758.plot()
```

executed in 98ms, finished 09:14:16 2021-06-20

Out[48]: <AxesSubplot:xlabel='time'>



- Baseline split of 0.90
- Adjusted to 0.91 to account for trend

In [49]:

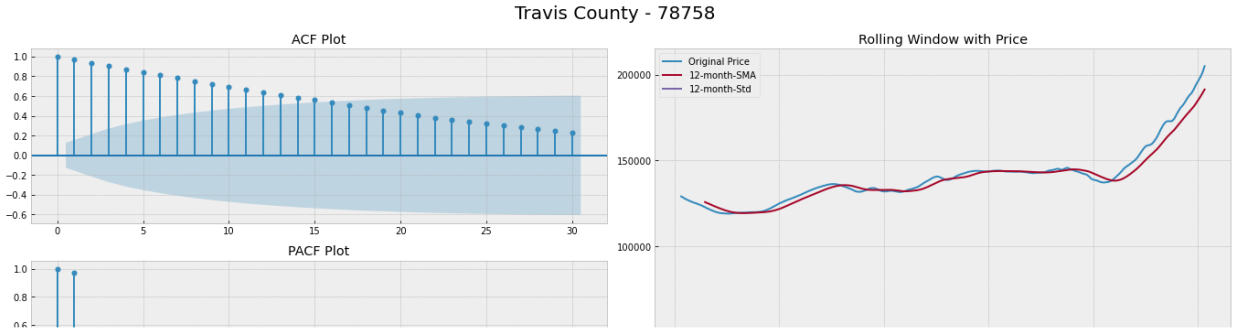
```
# Create diagnostic helper function
# Add .diff() to PACF/ACF dropna()

from statsmodels.graphics.tsaplots import plot_pacf, plot_acf
from matplotlib.gridspec import GridSpec
from statsmodels.tsa.stattools import adfuller

def pacf_acf_rolling(train, county, zip_, lags=30):
    with plt.style.context('bmh'):

        train=train.to_frame()
        fig = plt.figure(figsize=(19, 7), constrained_layout=False)
        gs = GridSpec(nrows=2, ncols=2)
        lags=30

        # First axes
        ax0 = fig.add_subplot(gs[0, 0])
        plot_acf(train['value'],lags=lags, ax=ax0);
        ax0.set_title('ACF Plot')

        # Second axes
        ax1 = fig.add_subplot(gs[1, 0])
        plot_pacf(train['value'],lags=lags, ax=ax1)
        ax1.set_title('PACF Plot')


        # Third axes
        ax2 = fig.add_subplot(gs[:, 1])

        train['12-month-SMA'] = train['value'].rolling(window=12).mean(
        train['12-month-Std'] = train['value'].rolling(window=12).std()
        ax2.plot(train['value'], label='Original Price')
        ax2.plot(train['12-month-SMA'], label='12-month-SMA')
        ax2.plot(train['12-month-Std'], label='12-month-Std')
        ax2.set_title('Rolling Window with Price')
        ax2.legend()

        fig.suptitle(f'{county} County - {zip_}', fontsize=20)
        fig.tight_layout();

        # Dickey Fuller
        print('Augmented Dickey-Fuller Test on 78722')
        dftest = adfuller(train['value'])
        dfout = pd.Series(dftest[0:4],index=['ADF test statistic','p-va
        for key,val in dftest[4].items():
            dfout[f'critical value ({key})']=val
        print(dfout)
```

executed in 5ms, finished 09:14:16 2021-06-20

In [50]:
```
1 pacf_acf_rolling(train_78758, 'Travis', 78758)
```

executed in 308ms, finished 09:14:16 2021-06-20

```
Augmented Dickey-Fuller Test on 78722
ADF test statistic        2.205379
p-value                   0.998887
# lags used              14.000000
# observations          226.000000
critical value (1%)      -3.459620
critical value (5%)      -2.874415
critical value (10%)     -2.573632
dtype: float64
```



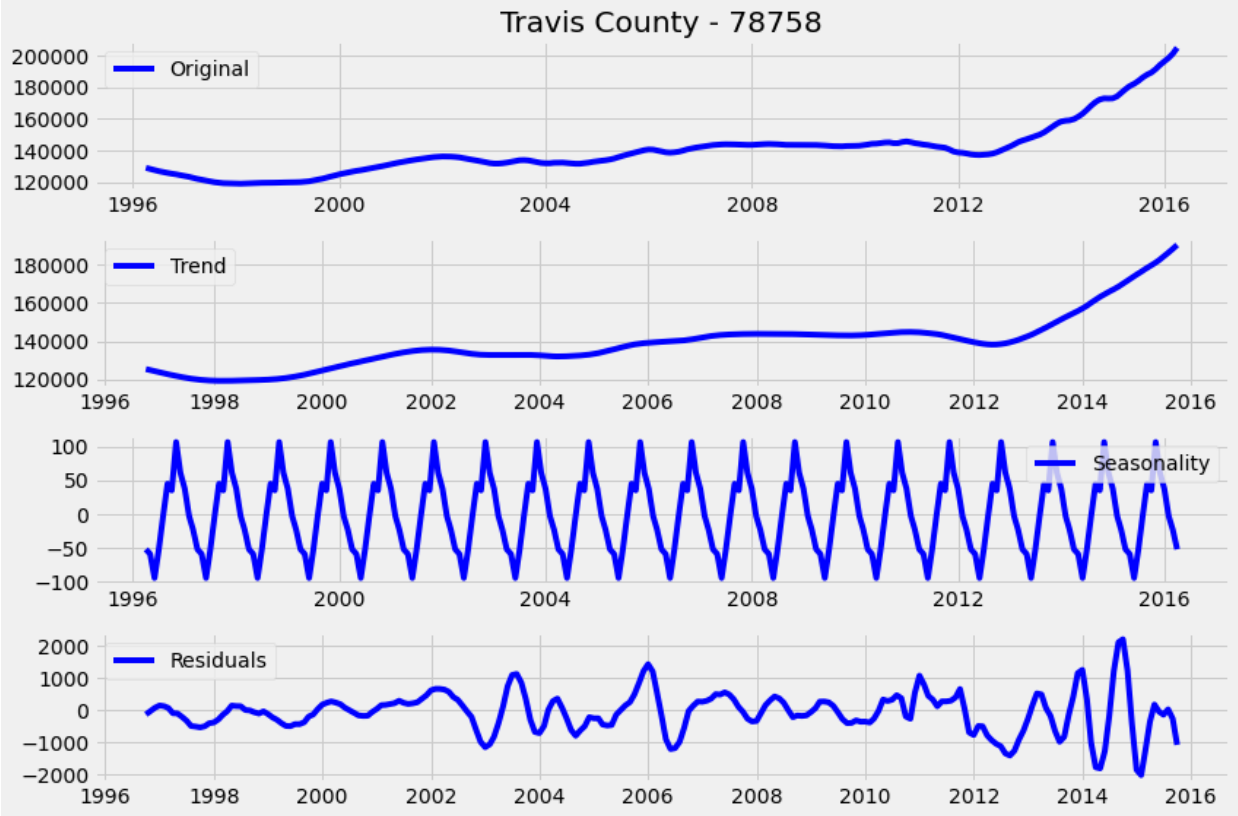
- ACF plot suggests data is not stationary because it has a downward gradual slope
  - Significant correlation up to 15 lags
- PACF plot drops off steeply after first lag suggesting little relationship to time
- Standard deviation picks up around 2013 and there is a clear upward trend at this time

In [51]:

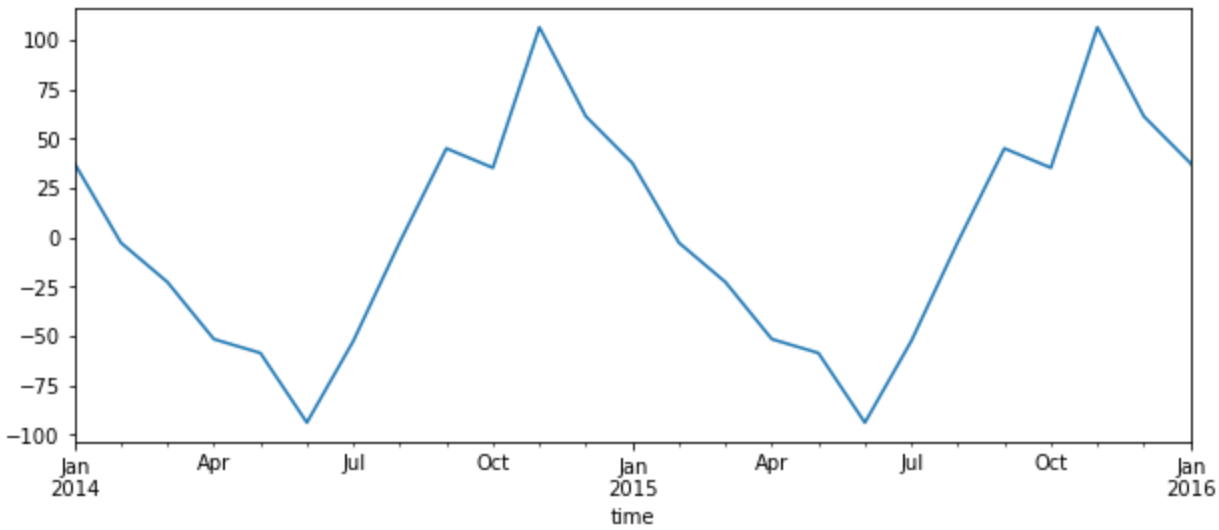
```
# Import and apply seasonal_decompose()
from statsmodels.tsa.seasonal import seasonal_decompose

def seasonal_decomposition(train, county, zip_, model_type='Additive'):
    # Gather the trend, seasonality, and residuals
    train=train.to_frame()
    decomposition = seasonal_decompose(train['value'],model=model_type)
    trend = decomposition.trend
    seasonal = decomposition.seasonal
    residual = decomposition.resid

    # Plot gathered statistics
    with plt.style.context('fivethirtyeight'):

        plt.figure(figsize=(12,8))
        plt.subplot(411, title=f'{county} County - {zip_}')
        plt.plot(train['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 4ms, finished 09:14:16 2021-06-20

In [52]:
```
1 seasonal_decomposition(train_78758, 'Travis', 78758)
```
executed in 342ms, finished 09:14:16 2021-06-20



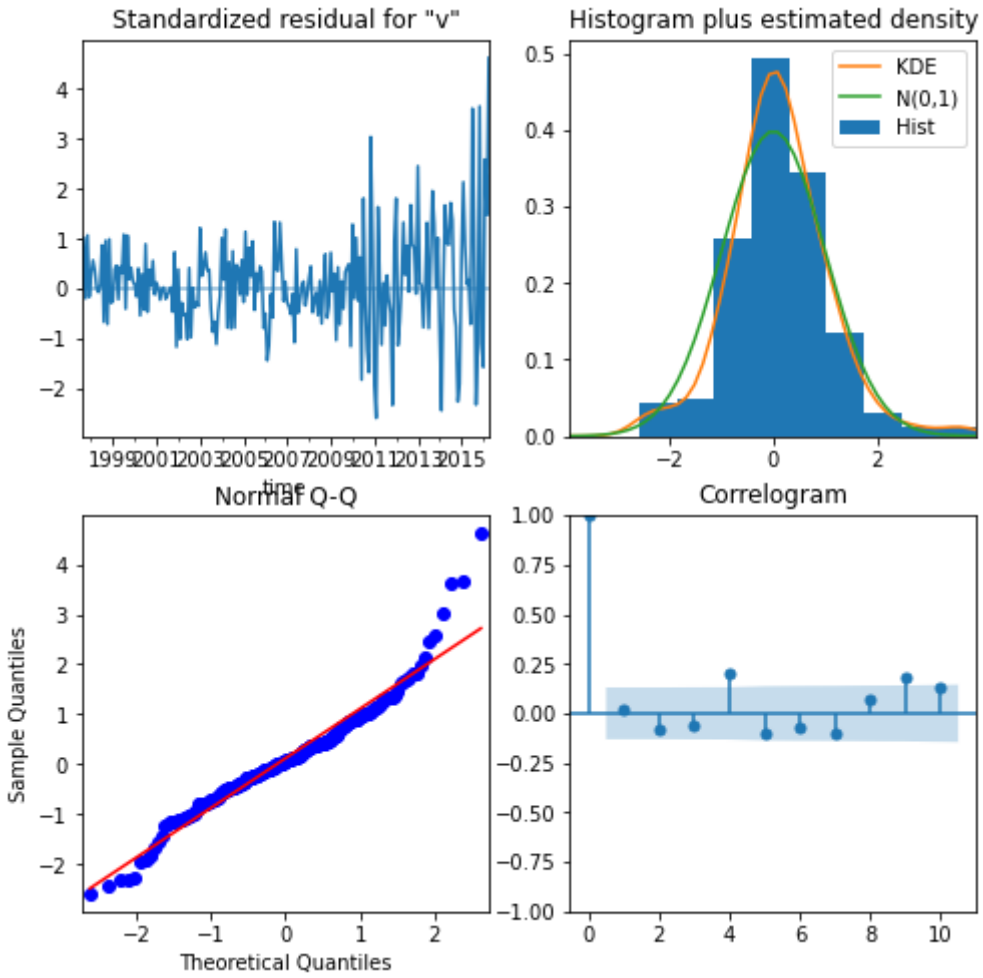
In [53]:
```
1 decomposition = seasonal_decompose(train_78758,model='Additive')
2 seasonal = decomposition.seasonal
3 seasonal.plot(figsize=(10,4), xlim=('2014-01-01', '2016-01-01'));
```
executed in 155ms, finished 09:14:16 2021-06-20

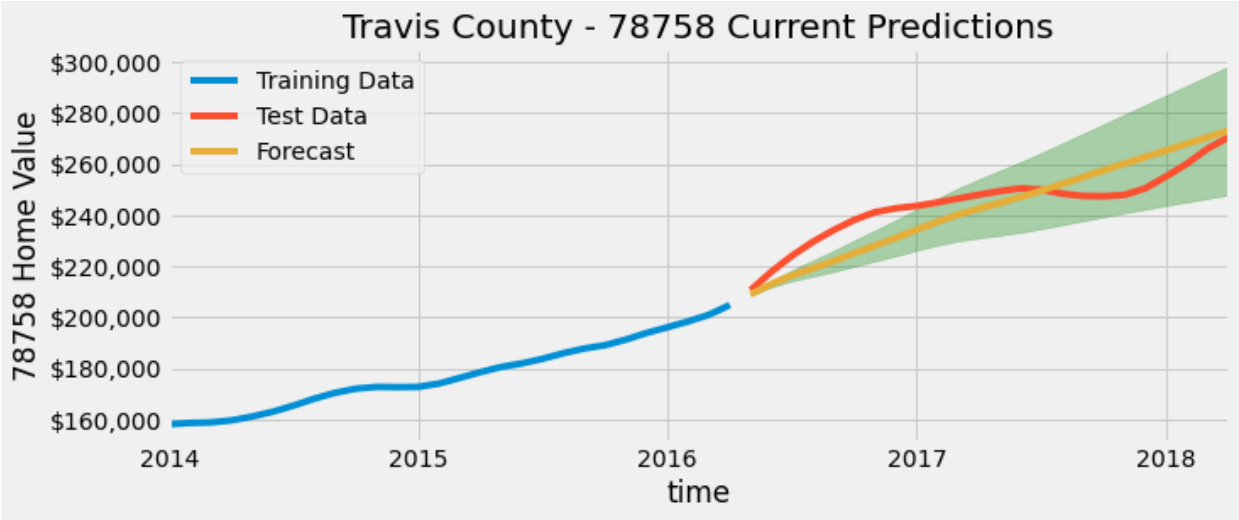


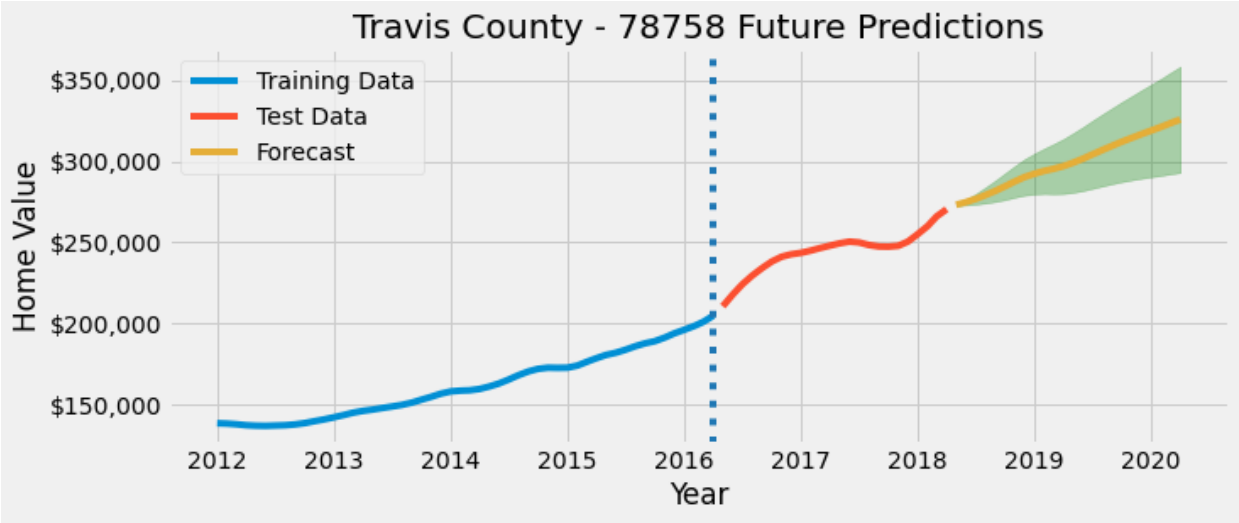
- There is an upward trend beginning in 2013
- Data is seasonal wiht peaks in the summer months (June-Sept) and dips in the winter months
- Seasonality appears constant
- Residuals suggest more variance after the upward trend in 2013

In [54]:
```
1 fig_78758, future_78758, forecast_df_78758, roi_78758 = model_predictio
```

executed in 16.4s, finished 09:14:33 2021-06-20







In [55]:
```
1 roi_78758
```
executed in 2ms, finished 09:14:33 2021-06-20

Out[55]:
```
{'Lower CI': [0.07520575524266447, 107520.57552426643, 7520.57552426643
1],
 'Upper CI': [0.30851281020908994, 130851.281020909, 30851.281020908995],
 'Forecast': [0.19213891296389998, 119213.89129639, 19213.891296389993]}
```

- Current predictions
  - Model captures general trend compared to test predictions
  - Confidence interal does a better job at capturing values after July 2017
- Future predictions
  - Model continues to follow general trend upward
  - Has a descent in mid-2019 and then continues upwards
- An investment of $100,000 today (05/01/2018) by 04/01/2020 would yield (ROI):
  - Conservative estimate: 7.5% ($7,520)
  - Mean estimate: 19.2% ($19,213)
  - Upper estimate: 30.9% ($30,851)

4.1.2 78721: EDA AND SARIMAX

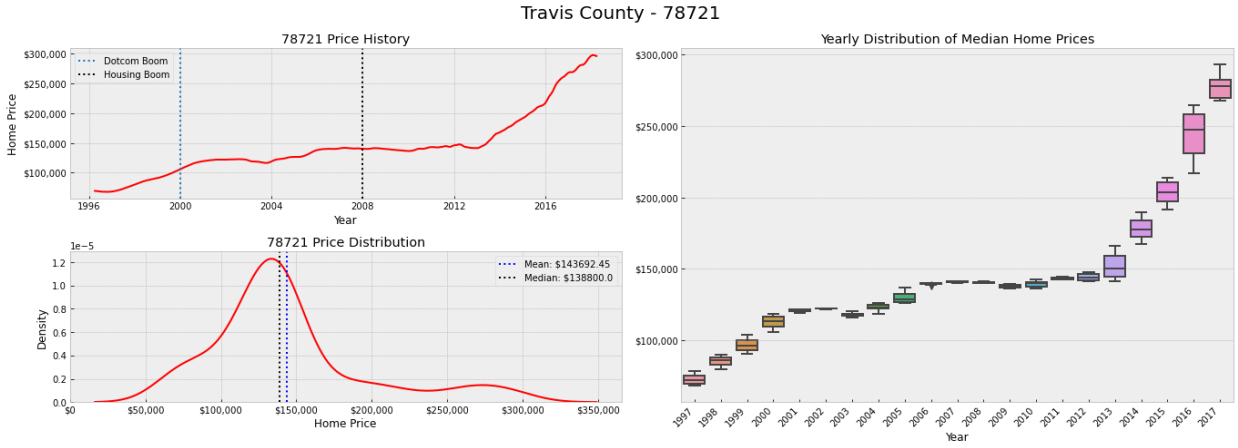
- East Austin
- 5 miles from downtown Austin

In [56]:
```
1 # Create 78721 dataframe
2
3 df_78721=create_zip_data(travis_dict_full, 78721)
```
executed in 5ms, finished 09:14:33 2021-06-20

In [57]:
```
1 zip_eda(df_78721, 78721, 'Travis')
```

executed in 614ms, finished 09:14:33 2021-06-20

Out[57]:
```
(<Figure size 1368x504 with 3 Axes>,
                value
count      265.000000
mean    143692.452830
std      51937.036652
min      67600.000000
25%     118700.000000
50%     138800.000000
75%     144600.000000
max     297800.000000)
```

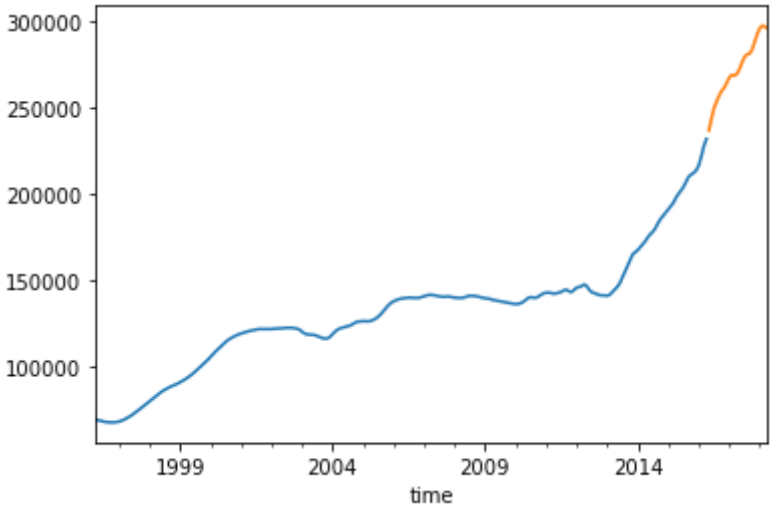


- Minimal impact on prices during dotcom crash and housing bust
- No large downturn but prices were stagnant between 2008 and 2012 before rising again
- Data is right skewed based on mean > median
- Historically prices are close to North Austin (78758)
- From 1997 to 2012 homes have minimal spread but begin to increase after that with significantly larger spread in 2013 and 2016

In [58]:
```
train_78721, test_78721 = create_train_test_split(df_78721, 0.91)
train_78721.plot()
test_78721.plot()
```

executed in 95ms, finished 09:14:34 2021-06-20

Out[58]: <AxesSubplot:xlabel='time'>

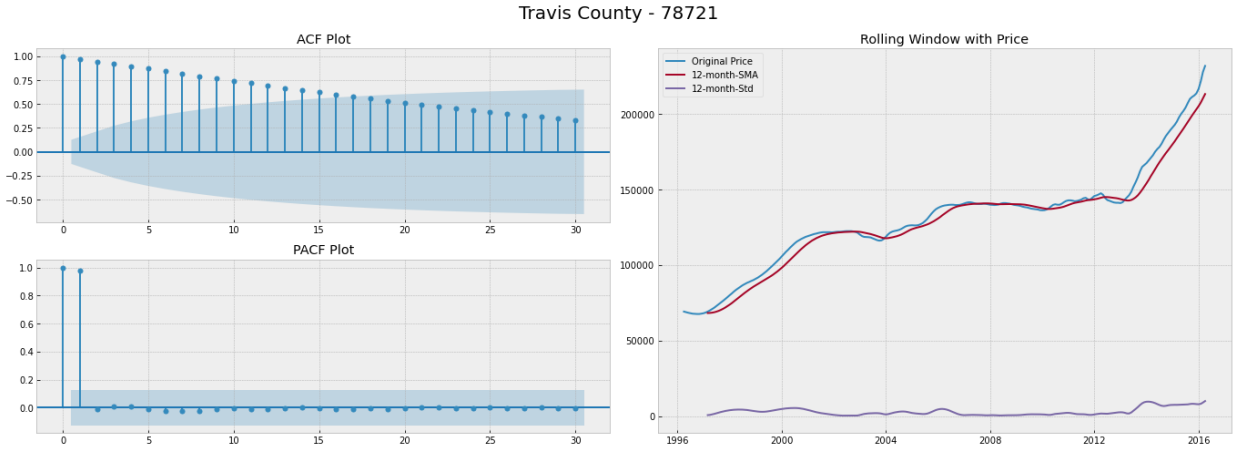


Split point looks like it accurately captures upward trend after 2013

In [59]:
```
1 pacf_acf_rolling(train_78721, 'Travis', 78721)
```

executed in 296ms, finished 09:14:34 2021-06-20

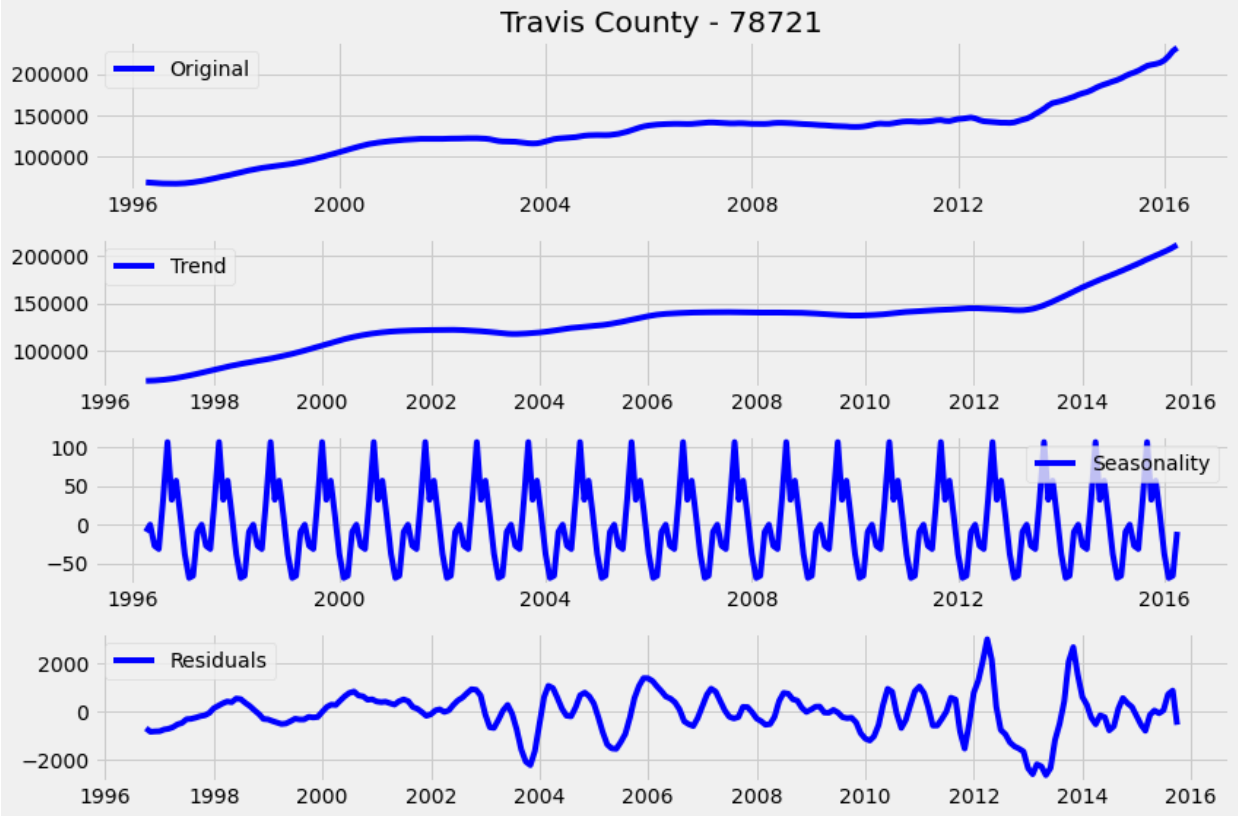
```
Augmented Dickey-Fuller Test on 78722
ADF test statistic        0.753836
p-value                   0.990853
# lags used               9.000000
# observations          231.000000
critical value (1%)      -3.458980
critical value (5%)      -2.874135
critical value (10%)     -2.573482
dtype: float64
```



- Based on Dickey Fuller test result data is not stationary, 0.99 > 0.05
- ACF plot suggests data is not stationary because it has a downward gradual slope
  - Significant correlation up to 16 lags
- PACF plot drops off steeply after first lag suggesting little relationship to time
- Standard deviation picks up around 2013 and there is a clear upward trend at this time
- 12 month rolling average follows original data closely

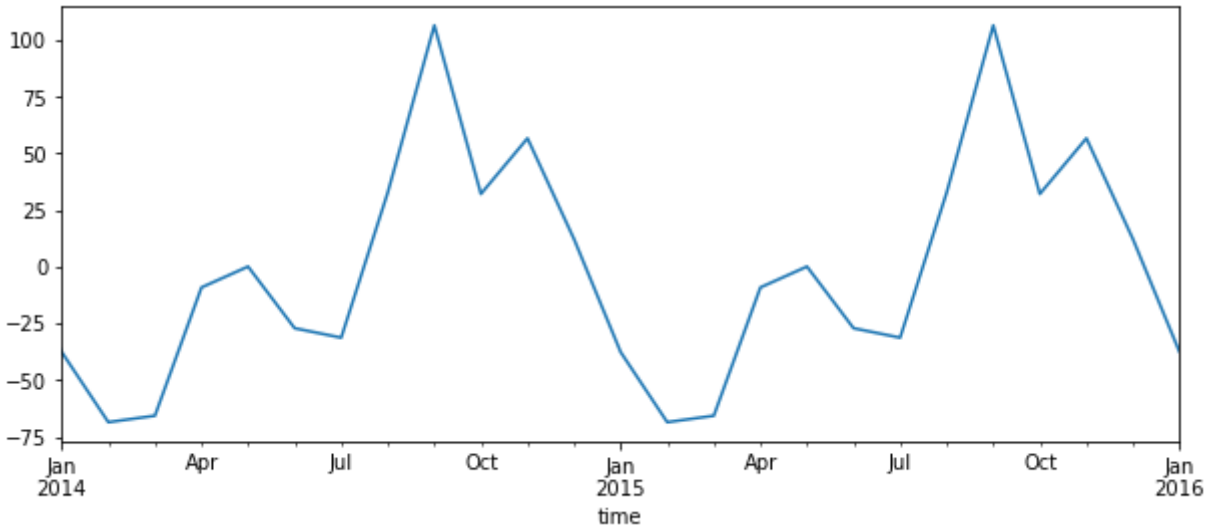
In [60]:
```
1 seasonal_decomposition(train_78721, 'Travis', 78721)
```

executed in 359ms, finished 09:14:34 2021-06-20



In [61]:
```
decomposition = seasonal_decompose(train_78721,model='Additive')
seasonal = decomposition.seasonal
seasonal.plot(figsize=(10,4), xlim=('2014-01-01', '2016-01-01'));
```

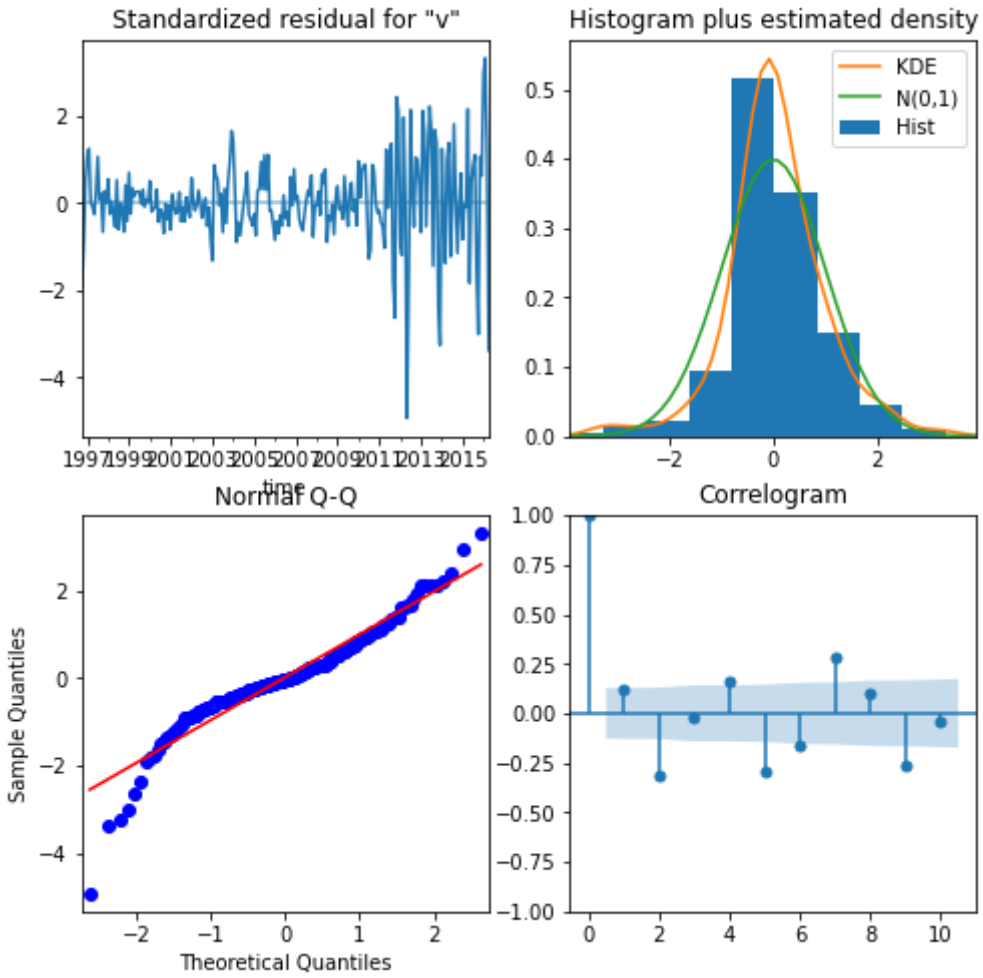
executed in 108ms, finished 09:14:34 2021-06-20

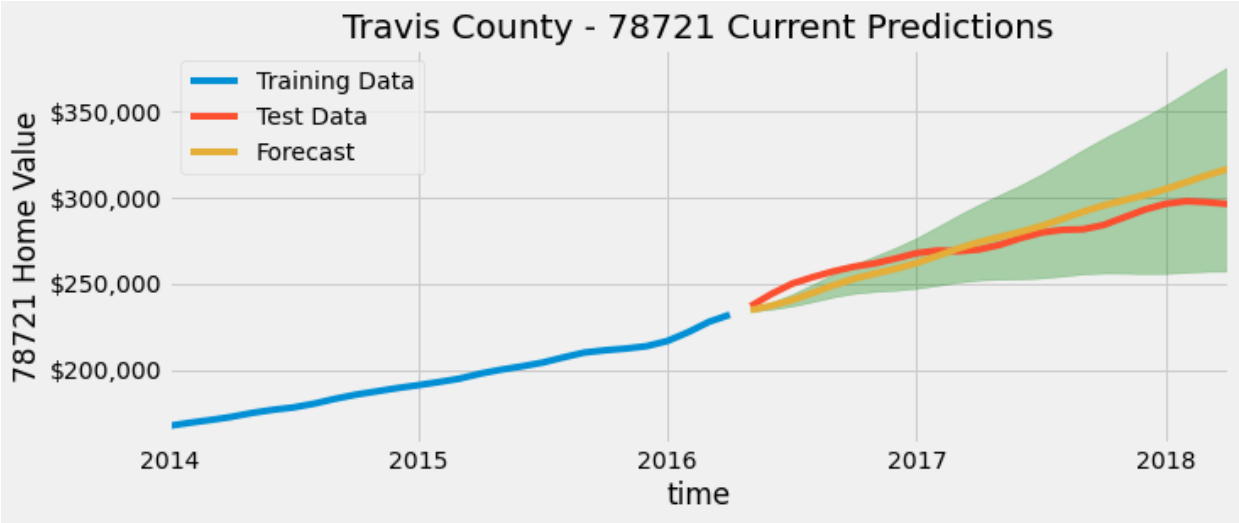


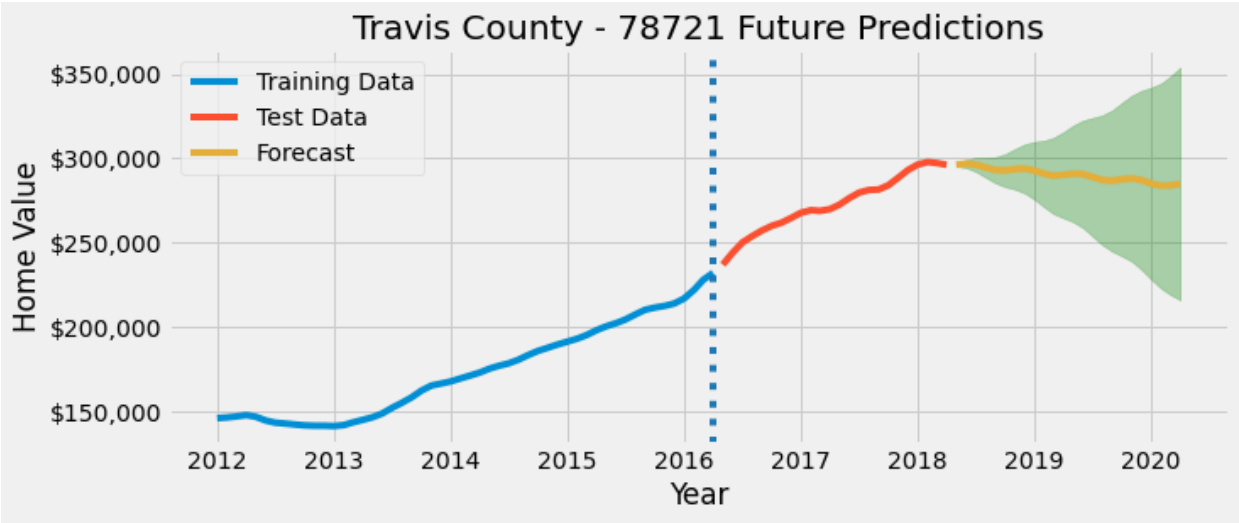
- There is an upward trend beginning in 2013
- Data is annually seasonal with peaks in months (Aug-Nov) and dips in (Feb-July)
- Seasonality appears constant
- Residuals suggest more variance after the upward trend in 2013, however they appear to borderline homoskedastic
- Looks like data needs 1 degree of differencing on seasonality

In [62]:
```
1 fig_78721, future_78721, forecast_df_78721, roi_78721 = model_predictio
```

executed in 25.9s, finished 09:15:00 2021-06-20







In [63]:
```
1 roi_78721
```

executed in 3ms, finished 09:15:00 2021-06-20

Out[63]:
```
{'Lower CI': [-0.2688608776766025, 73113.91223233975, -26886.087767660254],
 'Upper CI': [0.1911430110027367, 119114.30110027367, 19114.301100273675],
 'Forecast': [-0.03811420402484699, 96188.5795975153, -3811.4204024846986]}
```

- Current predictions
  - Model semi-captures general trend compared to test predictions
  - It tends to undershoot the test data
  - Confidence interal does a better job at capturing values after July 2017
- Future predictions
  - Model becomes more stagnant after 2018 and has a wide confidence interval
  - Slopes upward a bit in 2019
- An investment of $100,000 today (05/01/2018) by 04/01/2020 would yield (ROI):
  - Conservative estimate: -26.9% (-$26,886)
  - Mean estimate: -3.8% (-$3,811)
  - Upper estimate: 19.1% (+$19,114)

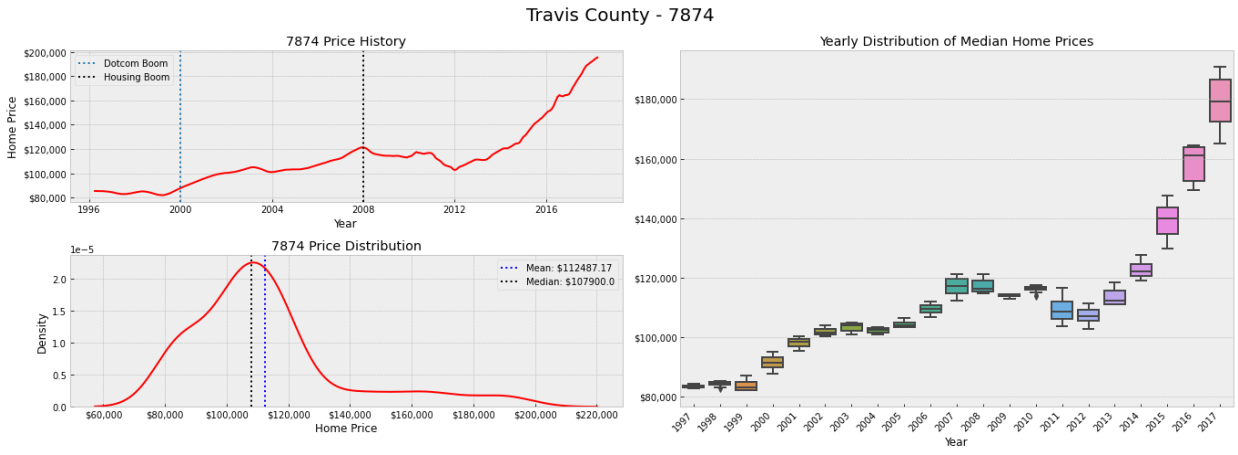
4.1.3 78744: EDA and SARIMAX

- South East Austin
- 6.5 miles from downtown Austin

In [64]:
```
# Create 78744 dataframe

df_78744=create_zip_data(travis_dict_full, 78744)
```
executed in 5ms, finished 09:15:00 2021-06-20

In [65]:
```
zip_eda(df_78744, 7874, 'Travis')
```
executed in 606ms, finished 09:15:01 2021-06-20

Out[65]:
```
(<Figure size 1368x504 with 3 Axes>,
                value
count      265.000000
mean    112487.169811
std      25161.294558
min      82000.000000
25%      99700.000000
50%     107900.000000
75%     116900.000000
max     195500.000000)
```



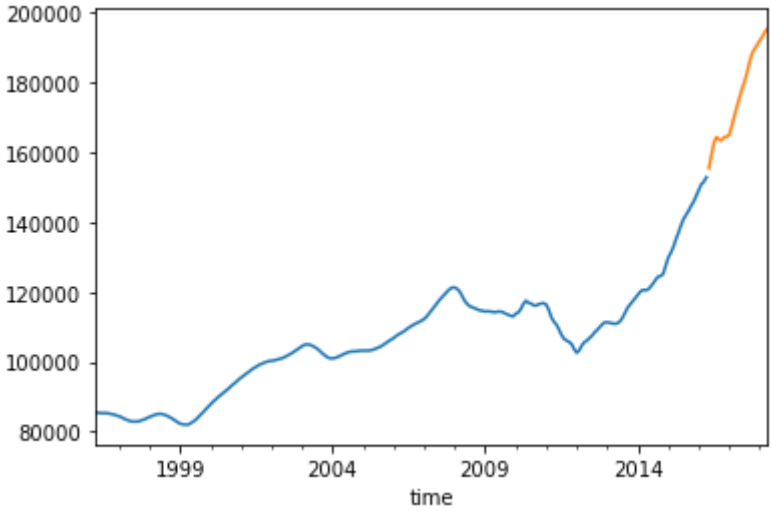
- Minimal impact on prices during dotcom crash and but large downturn after 2008 with another

dip in 2012
    - Looks like the median house price recovered in 2013
- Data is right skewed based on mean > median
- Lowest standard deviation in Travis County of those compared agasint
- From 1997 to 2006 homes had minimal spread but 2007 and 2008 had more variability, spread increased from 2015 onwards
    - Suggests transactions are taking place at a wider range of values

In [66]:
```
1 train_78744, test_78744 = create_train_test_split(df_78744, 0.91)
2 train_78744.plot()
3 test_78744.plot()
```

executed in 102ms, finished 09:15:01 2021-06-20

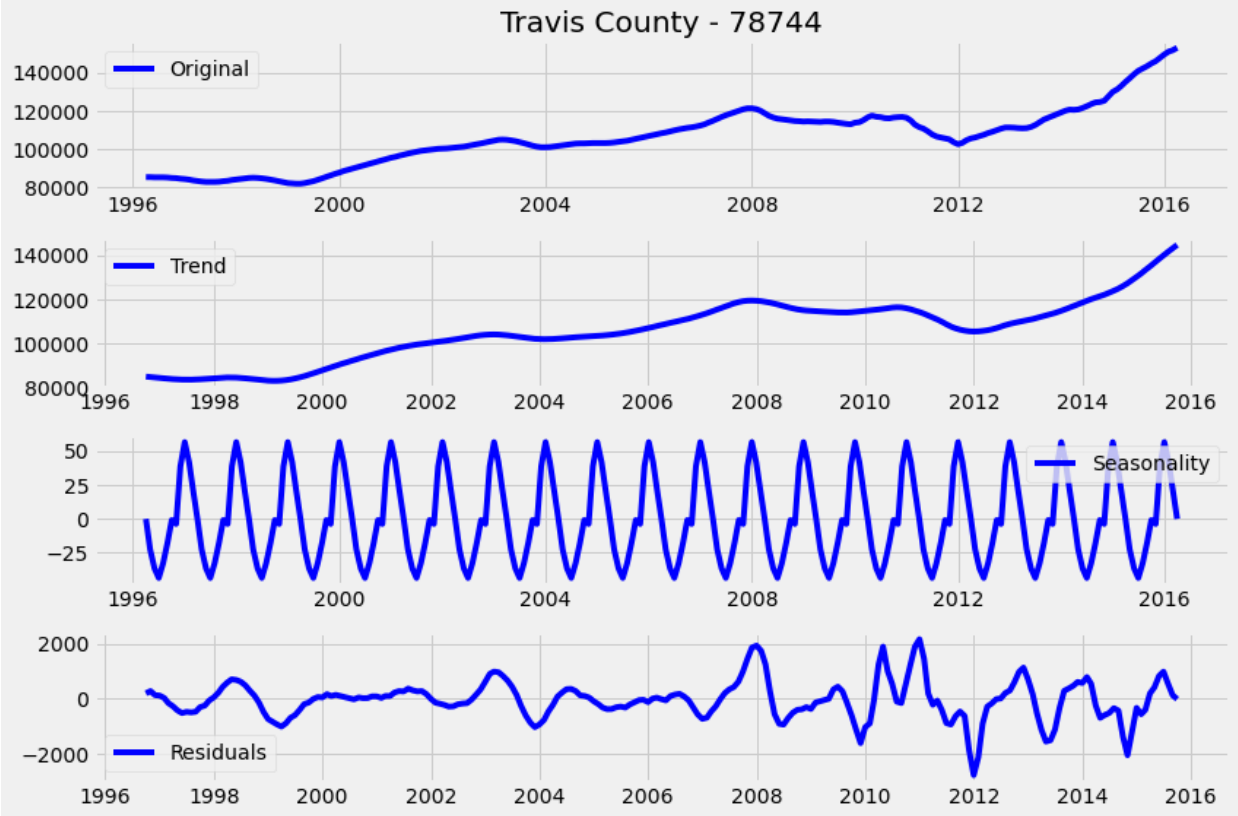
Out[66]: <AxesSubplot:xlabel='time'>



Split captures upwards trend and precedes slight dip in 2016

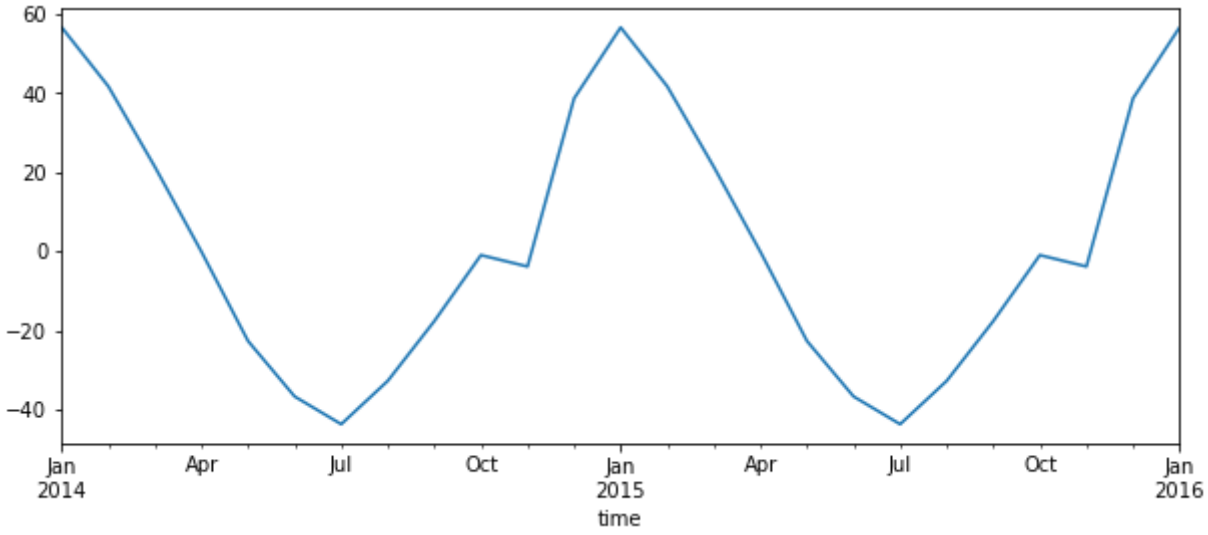
In [67]:
```
1 seasonal_decomposition(train_78744, 'Travis', 78744)
```

executed in 334ms, finished 09:15:01 2021-06-20



In [68]:
```
decomposition = seasonal_decompose(train_78744,model='Additive')
seasonal = decomposition.seasonal
seasonal.plot(figsize=(10,4), xlim=('2014-01-01', '2016-01-01'));
```

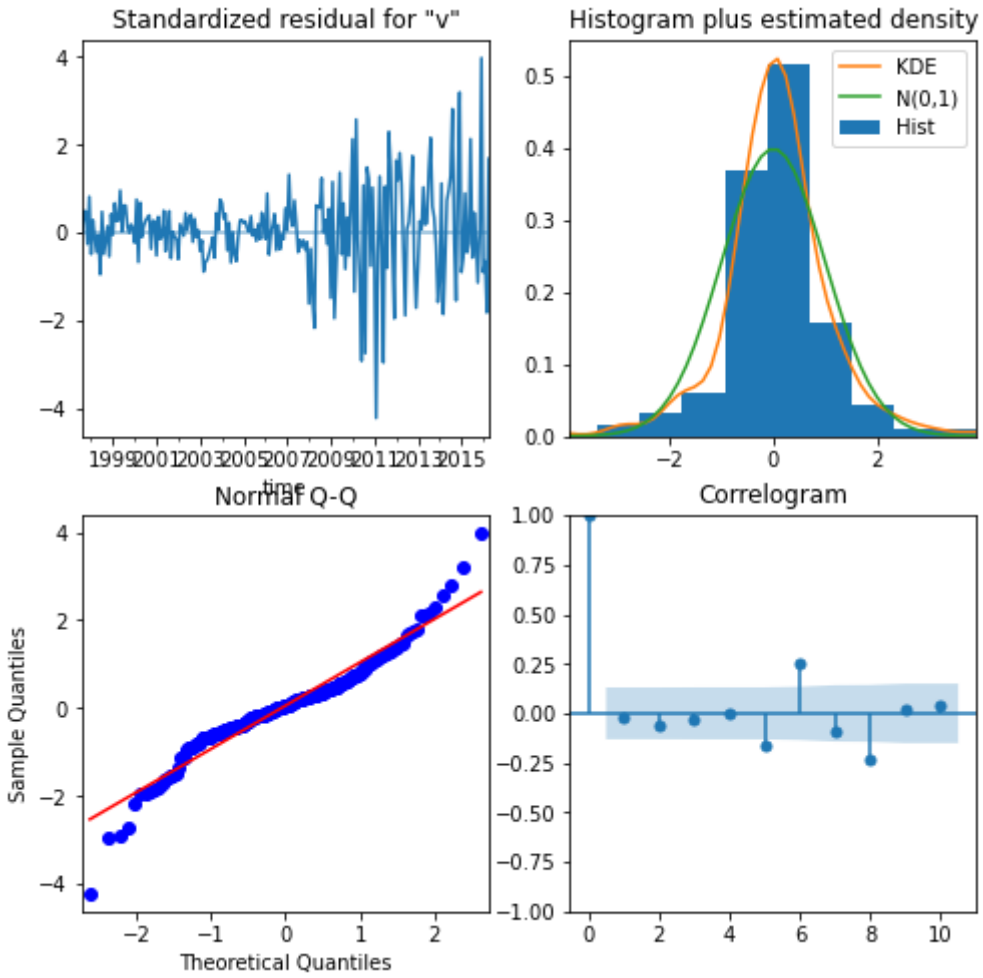
executed in 102ms, finished 09:15:01 2021-06-20

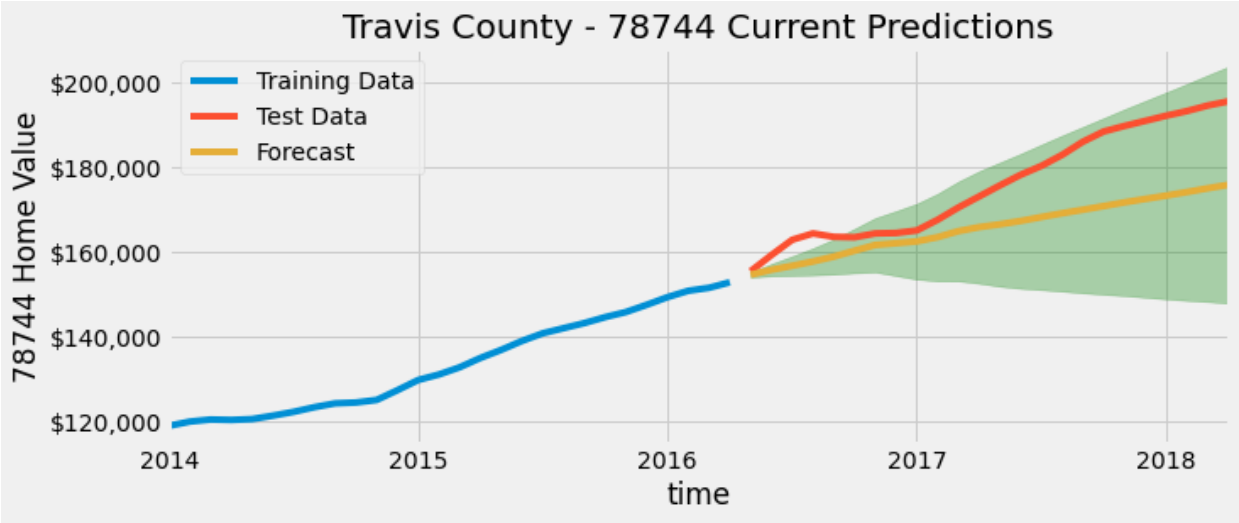


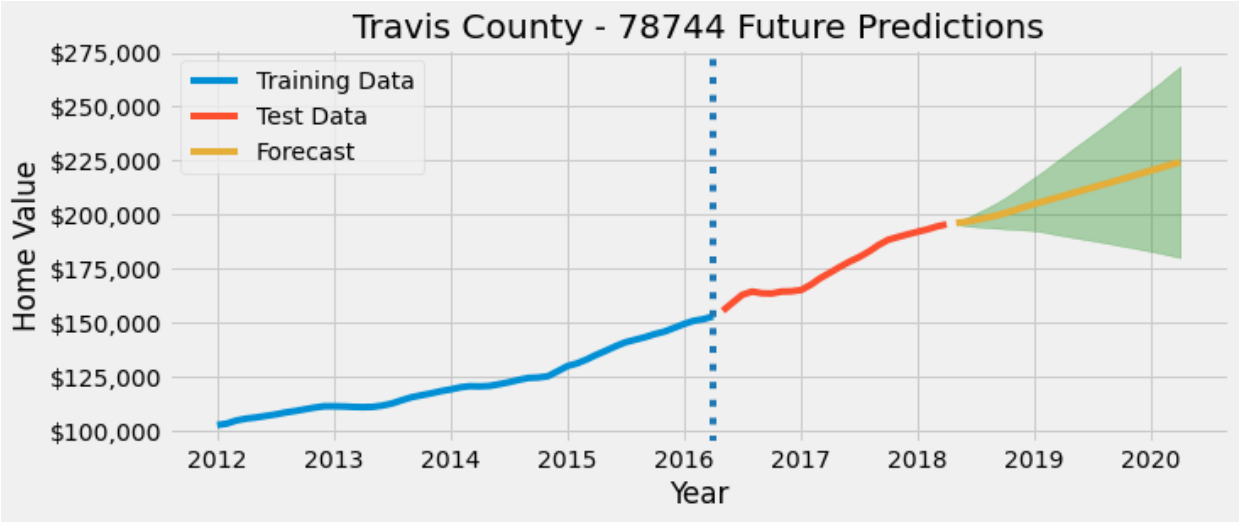
- There is an upward trend beginning in 2012
- Data is annually seasonal with peaks in months (Nov-Feb) and dips in summer (June-Sept)
- Seasonality appears constant
- Residuals suggest more variance in 2008 and a similar amount following that point in time
- Looks like data needs 1 degree of differencing on seasonality

In [69]:
```
1 fig_78744, future_78744, forecast_df_78744, roi_78744 = model_predictio
```

executed in 26.7s, finished 09:15:28 2021-06-20







In [70]:
```
1 roi_78744
```

executed in 2ms, finished 09:15:28 2021-06-20

Out[70]:
```
{'Lower CI': [-0.0782107601600331, 92178.92398399669, -7821.07601600331],
 'Upper CI': [0.36463656856079163, 136463.65685607918, 36463.6568560791
8],
 'Forecast': [0.14404750671074446, 114404.75067107445, 14404.7506710744
5]}
```

- Current predictions
  - Model captures general trend compared to test predictions, all test values fall within CI
  - It tends to undershoot the test data, especially after 2017
  - As test data tapers off the model does capture that
- Future predictions
  - Model follows trend of test data
  - Slopes upward a bit in mid - 2019
- An investment of $100,000 today (05/01/2018) by 04/01/2020 would yield (ROI):
  - Conservative estimate: -7.8% (-$7,821)
  - Mean estimate: 14.4% (+$14,404)

- Upper estimate: 36.4% (+$36,463)

4.2 Travis County Conclusions

In [71]:
```
def corr_check(df1, df2, df3, zip_1, zip_2, zip_3):
    df = pd.concat([df1, df2, df3], axis=1)
    df.columns = [zip_1, zip_2, zip_3]
    return df.corr()
```

executed in 3ms, finished 09:15:28 2021-06-20

In [72]:
```
corr_check(df_78758, df_78721, df_78744, 78758, 78721, 78744)
```

executed in 6ms, finished 09:15:28 2021-06-20

Out[72]:

	78758	78721	78744
78758	1.000000	0.967931	0.970744
78721	0.967931	1.000000	0.980363
78744	0.970744	0.980363	1.000000

- 78758 moves more closely with 78744
- 78721 moves more closely with 78744
- 78744 moves more closely with 78721
- This can be useful for picking up on trends using other nearby zip codes. They all move very closely together

In [73]:

```
# mess around with colors

def county_forecast_comparison(df1_all, zip_1, forecast1, df2_all, zip_
    with plt.style.context('fivethirtyeight'):
        fig, ax = plt.subplots(figsize=(12,5))

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

        # Ax1
        ax.plot(df1_all['2006-01-01':], label=zip_1)
        ax.plot(forecast1['Forecast'], color='Blue',label=f'{zip_1} For
        ax.fill_between(forecast1.index,forecast1['Lower CI'], forecast
                        color='Blue', ls=':', facecolor='blue', lw=3)
        ax.axvline(df1_all.index[-1], ls =':', lw=2)

        # Ax2
        ax.plot(df2_all['2006-01-01':], label=zip_2)
        ax.plot(forecast2['Forecast'], label=f'{zip_2} Forecast', color
        ax.fill_between(forecast2.index,forecast2['Lower CI'], forecast
                        color='red', ls=':', facecolor='red', lw=3)

        #7 Ax3
        ax.plot(df3_all['2006-01-01':], label=zip_3)
        ax.plot(forecast3['Forecast'], label=f'{zip_3} Forecast', color
        ax.fill_between(forecast3.index,forecast3['Lower CI'], forecast
                        color='green', ls=':', facecolor='green', lw=3)

        ax.legend(loc=2)
```

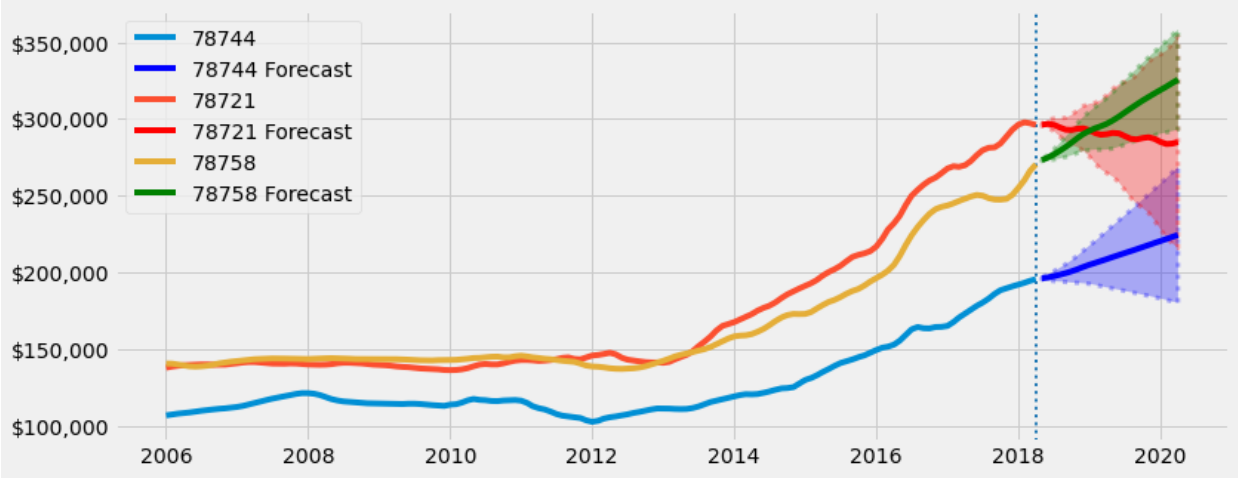
executed in 5ms, finished 09:15:28 2021-06-20

In [74]:

```
def county_forecast_perc_comparison(roi1, zip_1, roi2, zip_2, roi3, zip
    test_1 = pd.DataFrame(roi1, index=[f'{zip_1} Perc Change', 'Val', '
    test_2 = pd.DataFrame(roi2, index=[f'{zip_2} Perc Change', 'Val', '
    test_3 = pd.DataFrame(roi3, index=[f'{zip_3} Perc Change', 'Val', '
    df = pd.concat([test_1, test_2, test_3], axis=0)
    return df
```

executed in 3ms, finished 09:15:28 2021-06-20

In [75]:
```
1 county_forecast_comparison(df_78744, 78744, forecast_df_78744, df_78721
```
executed in 139ms, finished 09:15:28 2021-06-20



- 78758 has the highest ending point with the tightest confidence interval
- 78721 has almost as high of upside potential as 78758 but it has a much greater standard devation and tends to curve downwards
- 78744 slopes upward and has a standard deviation between the two other zip codes

In [76]:
```
1  # Zip codes are their own row
2  # Column would be percent change val/$ diff
3  # original that works
4
5  def county_forecast_perc_comparison(roi1, zip_1, roi2, zip_2, roi3, zip
6      test_1 = pd.DataFrame(roi1, index=[f'{zip_1} Perc Change', 'Val', '
7      test_2 = pd.DataFrame(roi2, index=[f'{zip_2} Perc Change', 'Val', '
8      test_3 = pd.DataFrame(roi3, index=[f'{zip_3} Perc Change', 'Val', '
9      df = pd.concat([test_1, test_2, test_3], axis=0)
10     return df
```
executed in 2ms, finished 09:15:28 2021-06-20

In [77]:
```
1  # def county_forecast_perc_comparison(roi1, zip_1, roi2, zip_2, roi3, z
2  #     test_1 = pd.DataFrame(roi1, columns=['Perc Change', 'Val', '$ Dif
3  #     test_2 = pd.DataFrame(roi2, columns=['Perc Change', 'Val', '$ Dif
4  #     test_3 = pd.DataFrame(roi3, columns=['Perc Change', 'Val', '$ Dif
5  #     df = pd.concat([test_1, test_2, test_3], axis=0)
6  #     return df
```
executed in 2ms, finished 09:15:28 2021-06-20

In [78]:
```
1 travis_perc_comparison = county_forecast_perc_comparison(roi_78758, 787
```
executed in 3ms, finished 09:15:28 2021-06-20

In [79]:
```
1 pd.DataFrame(roi_78721, index=['Perc Change', 'Val', '$ Diff']).T
```
executed in 5ms, finished 09:15:28 2021-06-20

Out[79]:

	Perc Change	Val	$ Diff
Lower CI	-0.268861	73113.912232	-26886.087768
Upper CI	0.191143	119114.301100	19114.301100
Forecast	-0.038114	96188.579598	-3811.420402

In [80]:
```
1 # # Loop thru columns to
2
3 # pd.DataFrame({78758:pd.Series(roi_78758)[0], 78721:pd.Series(roi_7872
```
executed in 2ms, finished 09:15:28 2021-06-20

In [81]:
```
1 travis_perc_comparison
```
executed in 5ms, finished 09:15:28 2021-06-20

Out[81]:

	Lower CI	Upper CI	Forecast
78758 Perc Change	0.075206	0.308513	0.192139
Val	107520.575524	130851.281021	119213.891296
$ Diff	7520.575524	30851.281021	19213.891296
78721 Perc Change	-0.268861	0.191143	-0.038114
Val	73113.912232	119114.301100	96188.579598
$ Diff	-26886.087768	19114.301100	-3811.420402
78744 Perc Change	-0.078211	0.364637	0.144048
Val	92178.923984	136463.656856	114404.750671
$ Diff	-7821.076016	36463.656856	14404.750671

- Based on the downside risk, upside return, and mean predicted value, 78758 seems like the superior zip code
- While it's upside is not as high as 78744, it has a much more predictable return profile and has the highest predicted forecast
- 78721 has the worst metrics across the board
- **In Travis county, 78758 has the best prospects for near term growth**

4.3 Bexar County Modeling

- Located in San Antonio metro
- Population: 2,003,554

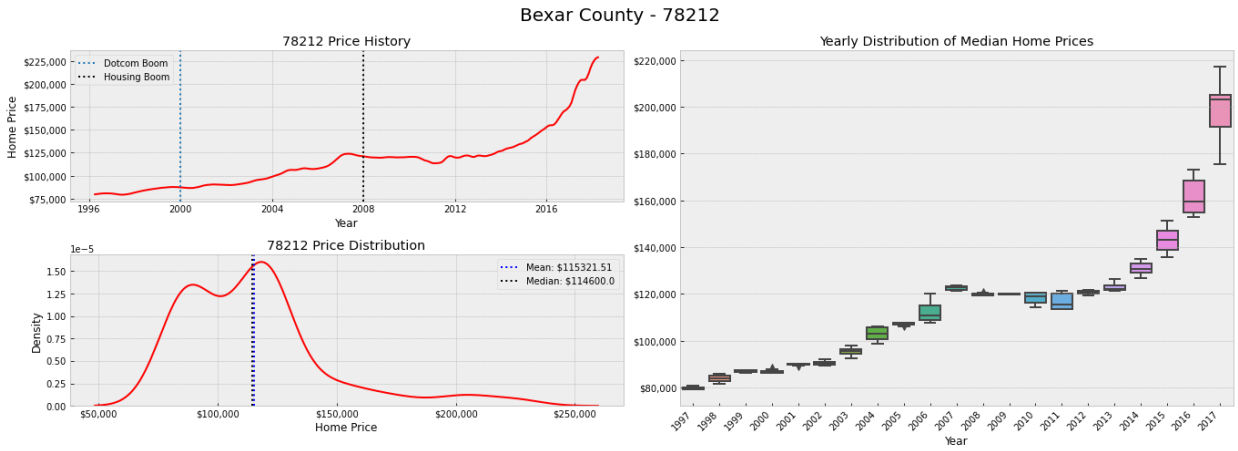
4.3.1 78212: EDA and SARIMAX

- Midtown San Antonio
- 4 miles from downtown San Antonio

In [82]:
```
# Create 78212 dataframe

df_78212=create_zip_data(Bexar_dict_full, 78212)
```
executed in 5ms, finished 09:15:28 2021-06-20

In [83]:
```
zip_eda(df_78212, 78212, 'Bexar')
```
executed in 595ms, finished 09:15:29 2021-06-20

Out[83]:
```
(<Figure size 1368x504 with 3 Axes>,
                value
count      265.000000
mean    115321.509434
std      31134.681006
min      79100.000000
25%      89900.000000
50%     114600.000000
75%     121900.000000
max     229100.000000)
```

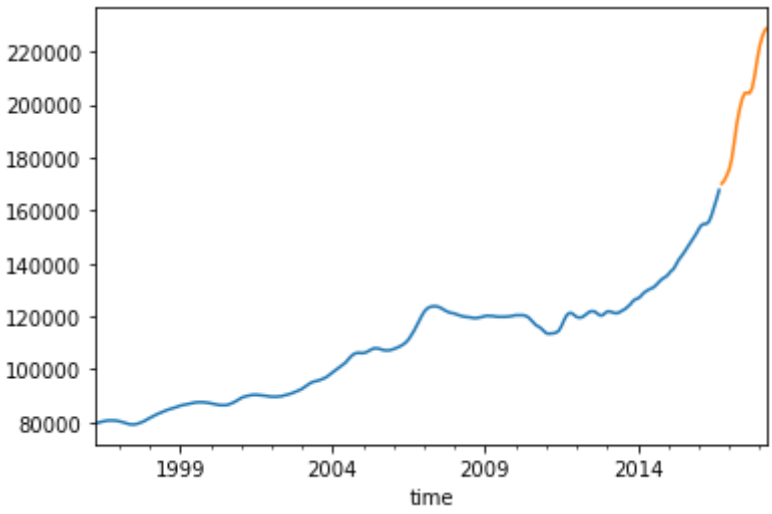


- Minimal impact on prices during dotcom crash and minimala downturn after 2008 with another dip in 2012 followed by strong recovery shortly after
  - Looks like the median house price recovered in 2013
- Data is slightly right skewed based on mean > median
- Data appears sort of bimodal suggesting there were swings when prices were much lower and now much higher, was not a continuous shift
- From 1997 to 2005 homes had minimal spread but 2006 had more variability, spread increased from 2015 onwards
  - Suggests transactions are taking place at a wider range of values throughout the year

In [84]:
```
train_78212, test_78212 = create_train_test_split(df_78212, 0.93)
train_78212.plot()
test_78212.plot()
```

executed in 97ms, finished 09:15:29 2021-06-20

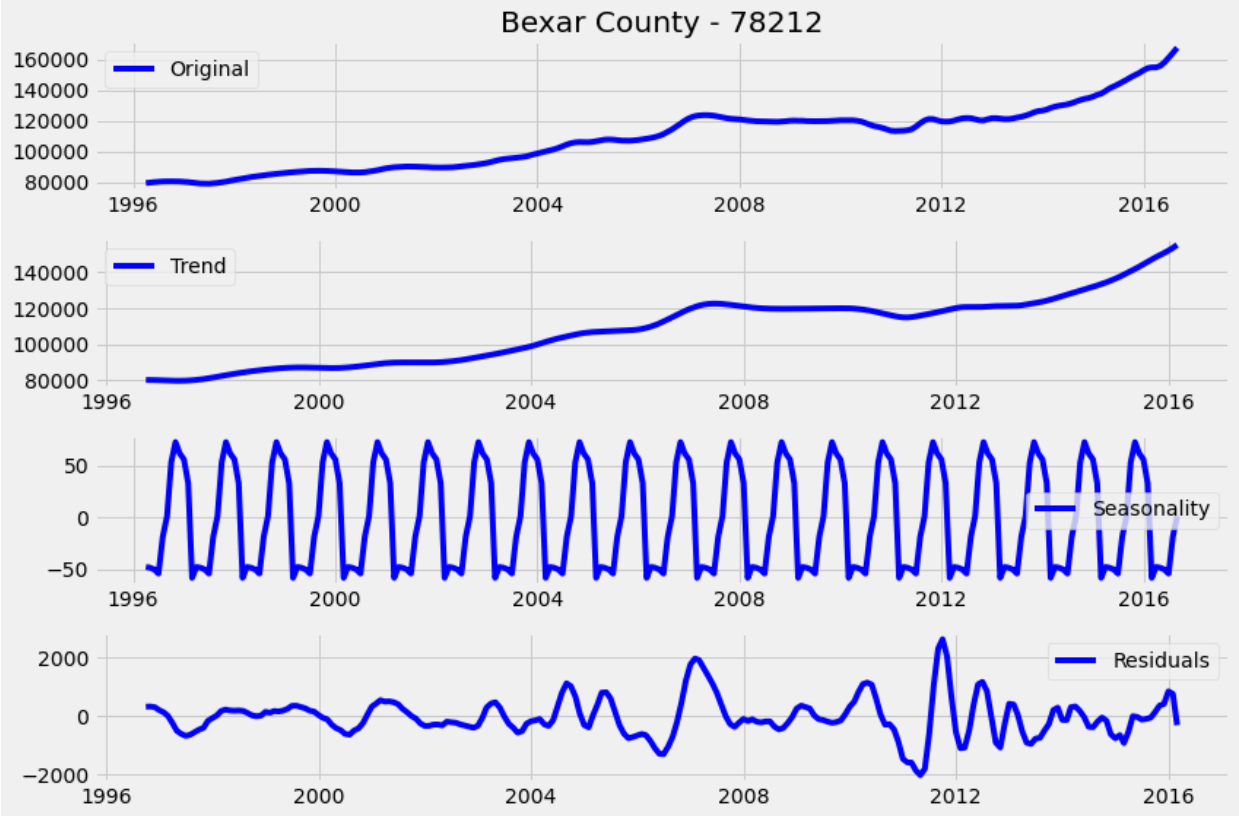
Out[84]: <AxesSubplot:xlabel='time'>



- Split point detects recent upward (possibly exponential) trend
- Initially used 0.90 but it was not capturing the trend well so used 0.93 of the training data

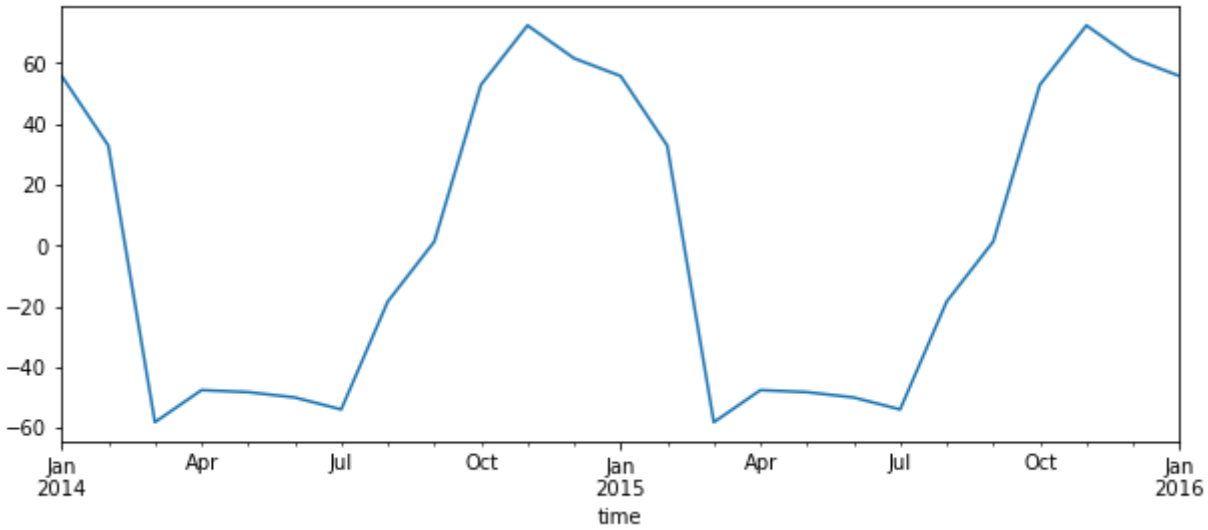
In [85]:
```
1 seasonal_decomposition(train_78212, 'Bexar', 78212)
```

executed in 311ms, finished 09:15:29 2021-06-20



In [86]:
```
decomposition = seasonal_decompose(train_78212,model='Additive')
seasonal = decomposition.seasonal
seasonal.plot(figsize=(10,4), xlim=('2014-01-01', '2016-01-01'));
```

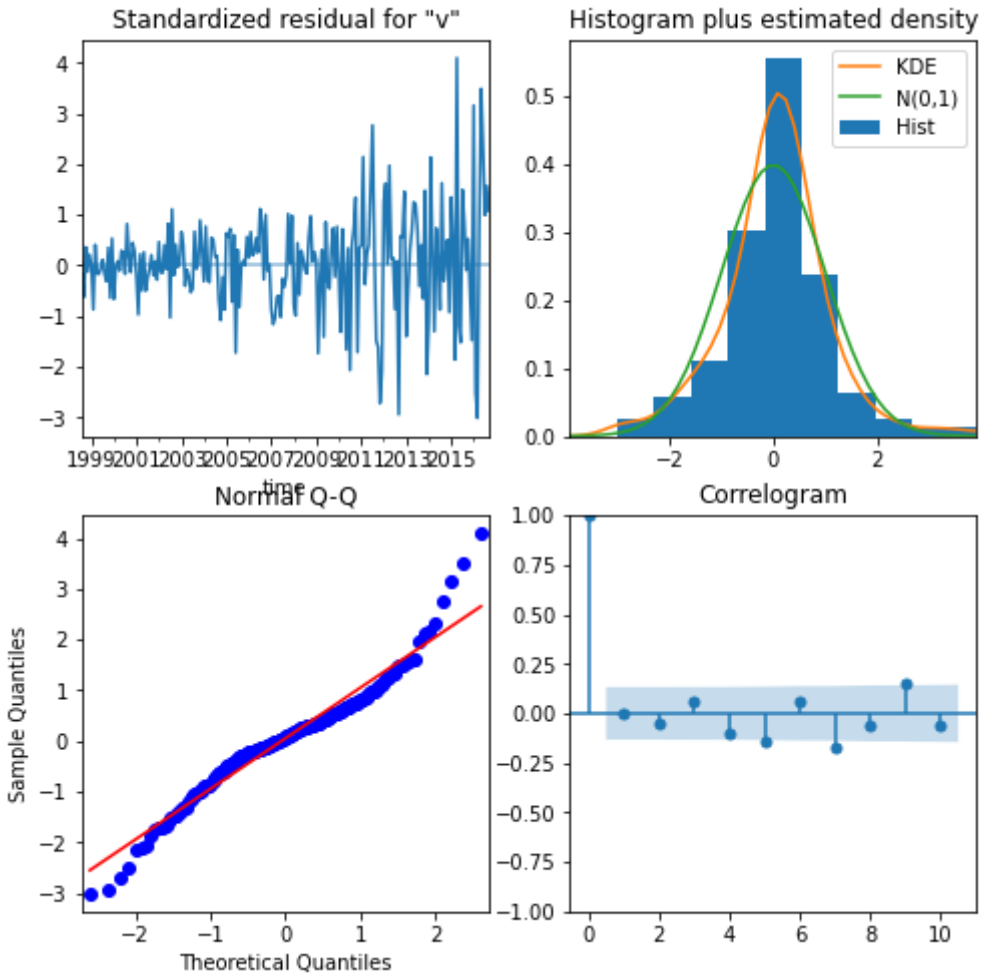
executed in 104ms, finished 09:15:29 2021-06-20

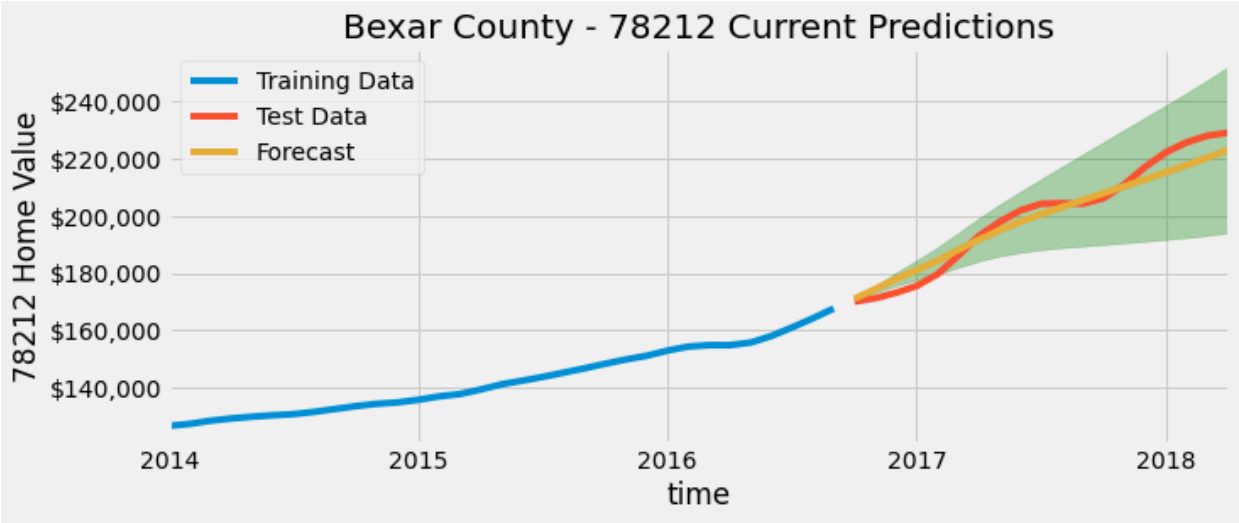


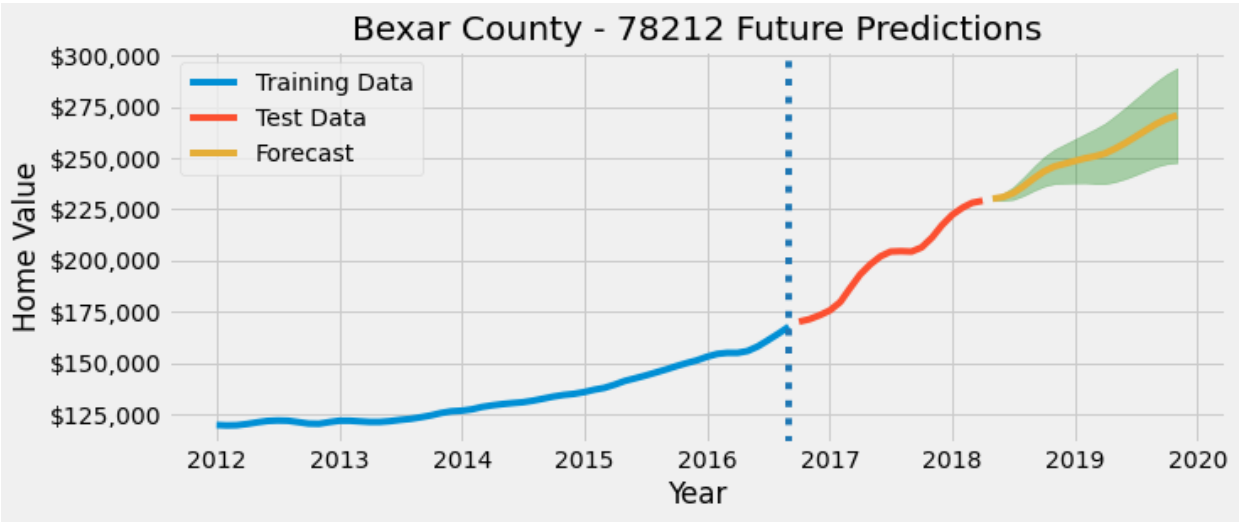
- There is an upward trend beginning in 2011
- Data is annually seasonal with peaks in months (Oct-Feb) and dips in summer (June-Sept)
- Seasonality appears constant
- Residuals suggest more variance in 2008 and a similar amount following that point in time
  - Lower variance beginning in 2014
- Looks like data needs 1 degree of differencing on seasonality

In [87]:
```
1 fig_78212, future_78212, forecast_df_78212, roi_78212 = model_predictio
```

executed in 2m 24s, finished 09:17:54 2021-06-20







In [88]:
```
1 roi_78212
```

executed in 2ms, finished 09:17:54 2021-06-20

Out[88]:
```
{'Lower CI': [0.07944511492223134, 107944.51149222313, 7944.51149222313],
 'Upper CI': [0.27361853144748954, 127361.85314474895, 27361.8531447489
5],
 'Forecast': [0.17680210000466143, 117680.21000046613, 17680.21000046613
3]}
```

- Current predictions
  - Model captures general trend compared to test predictions, all test values fall within CI
  - The predictions look like they follow a rolling average of the model
- Future predictions
  - Model follows trend of test data
  - Slopes upward a bit in mid - 2019
- An investment of $100,000 today (05/01/2018) by 11/01/2019 would yield (ROI):
  - Conservative estimate: 7.9% ($7,994)
  - Mean estimate: 17.7% (+$17,680)
  - Upper estimate: 27.4% (+$27,361)

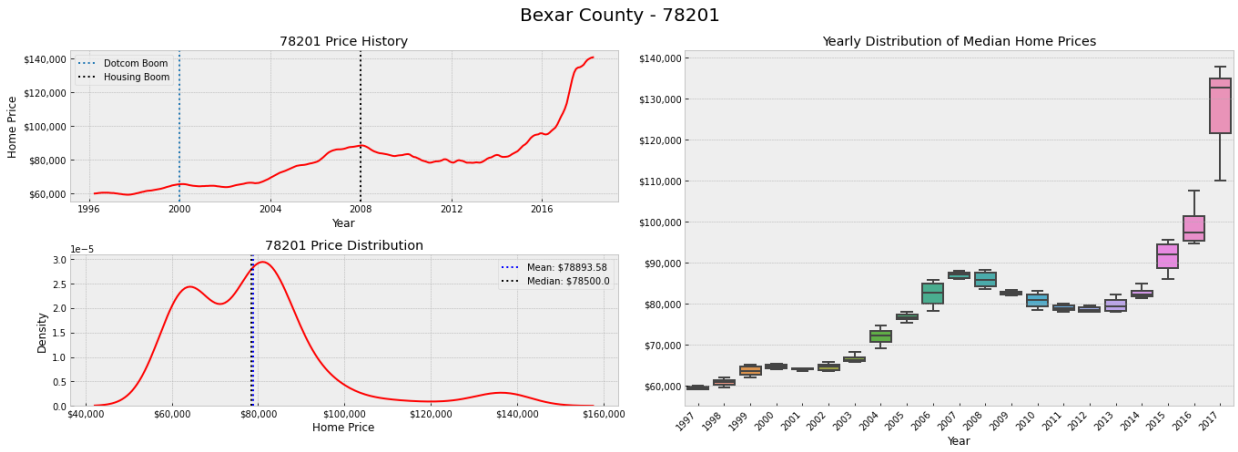
4.3.2 78201: EDA and SARIMAX

- Monticello Park, San Antonio
- 5.4 miles from downtown San Antonio

In [89]:
```
# Create 78201 dataframe

df_78201=create_zip_data(Bexar_dict_full, 78201)
```
executed in 4ms, finished 09:17:54 2021-06-20

In [90]:
```
zip_eda(df_78201, 78201, 'Bexar')
```
executed in 615ms, finished 09:17:54 2021-06-20

Out[90]:
```
(<Figure size 1368x504 with 3 Axes>,
                value
count      265.000000
mean     78893.584906
std      17210.957034
min      59100.000000
25%      64800.000000
50%      78500.000000
75%      84300.000000
max     140600.000000)
```

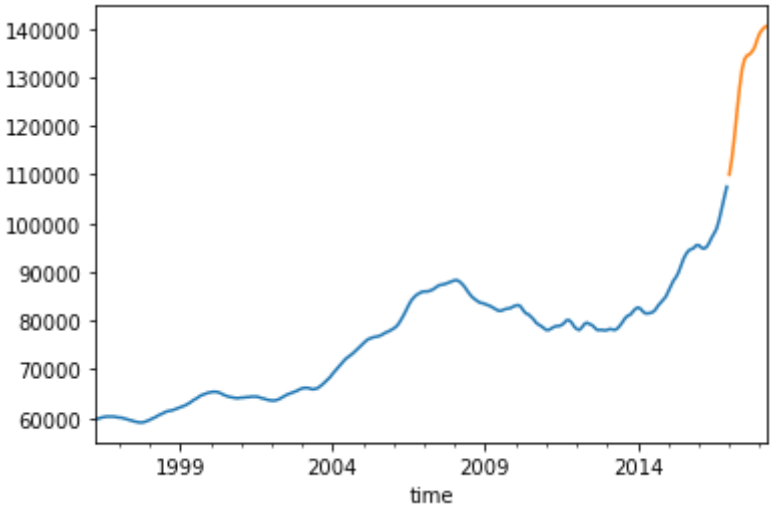


- Small downward trend after dotcom boom with a rise up to 2008 and then a big fall
  - median house price recovered in 2015
- Data is slightly right skewed
- Data appears sort of bimodal suggesting there were swings when prices were much lower and now much higher, was not a continuous shift
- Not too much spread between home prices except for in 2006 and a very large spread in 2017

In [91]:
```
train_78201, test_78201 = create_train_test_split(df_78201, 0.94)
train_78201.plot()
test_78201.plot()
```

executed in 103ms, finished 09:17:54 2021-06-20

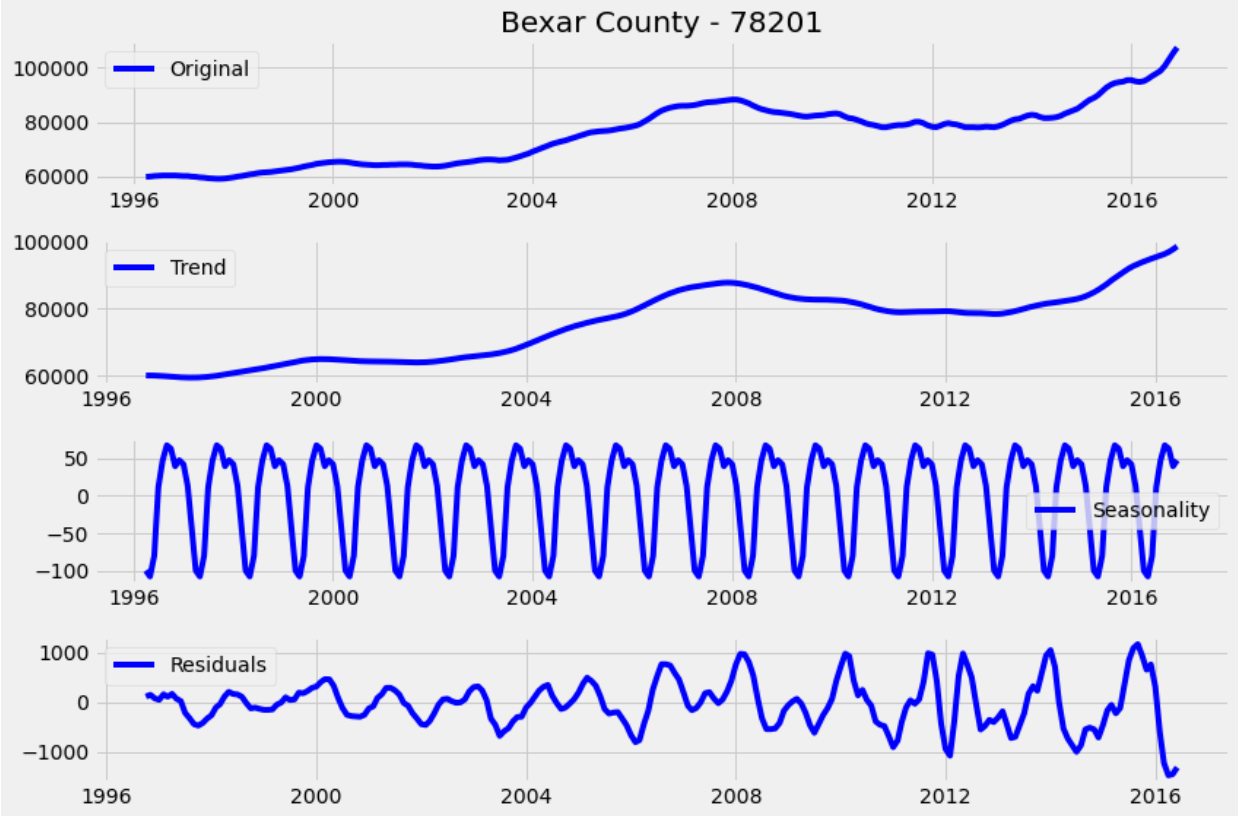
Out[91]: <AxesSubplot:xlabel='time'>



- Had to adjust split point from 0.90 to 0.94 because model was not able to capture new upward trend

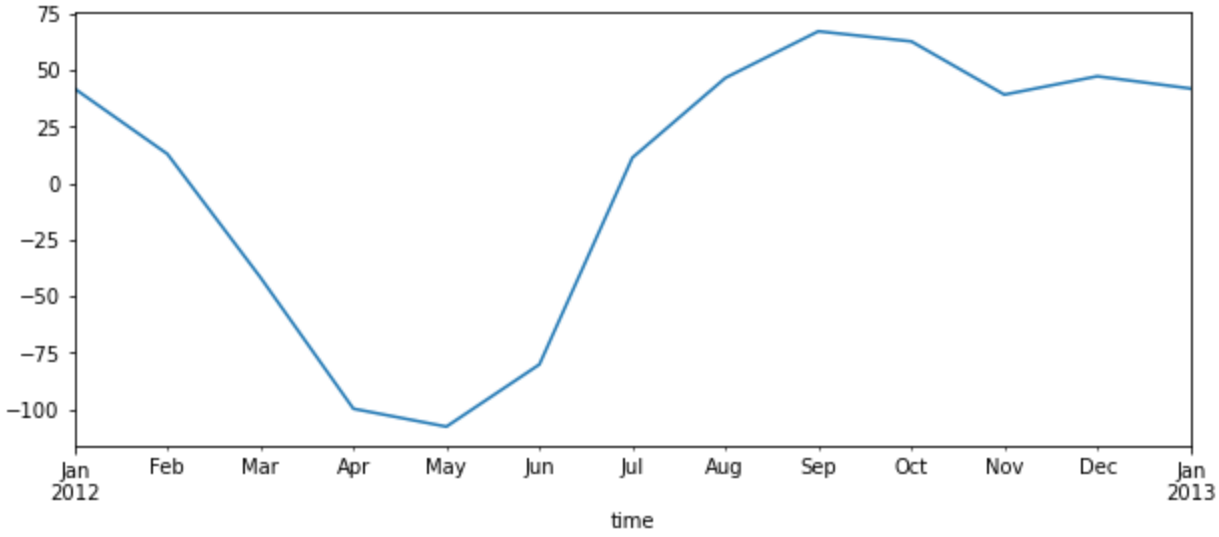
In [92]:
```
1 seasonal_decomposition(train_78201, 'Bexar', 78201)
```

executed in 314ms, finished 09:17:55 2021-06-20



In [93]:
```
decomposition = seasonal_decompose(train_78201,model='Additive')
seasonal = decomposition.seasonal
seasonal.plot(figsize=(10,4), xlim=('2012-01-01', '2013-01-01'));
```

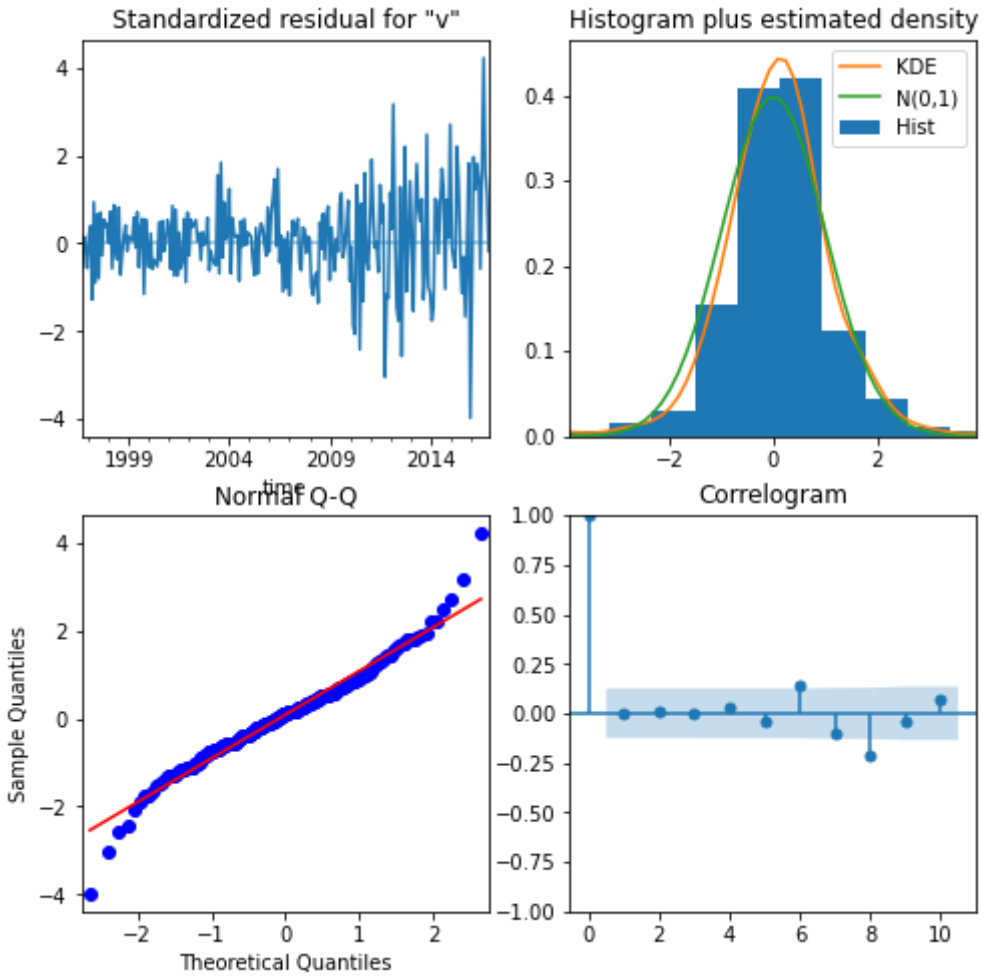
executed in 99ms, finished 09:17:55 2021-06-20

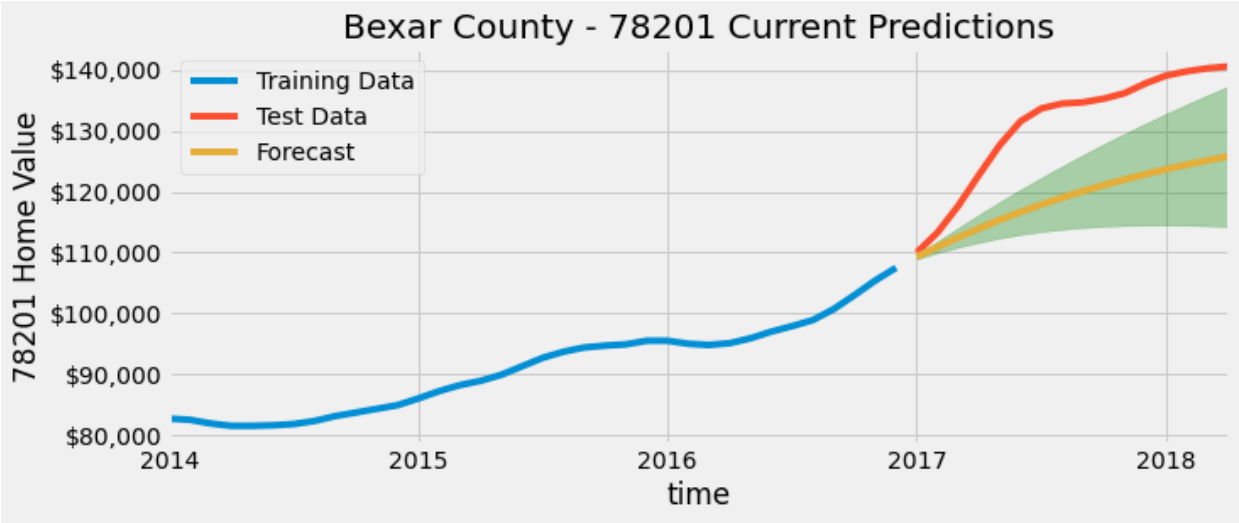


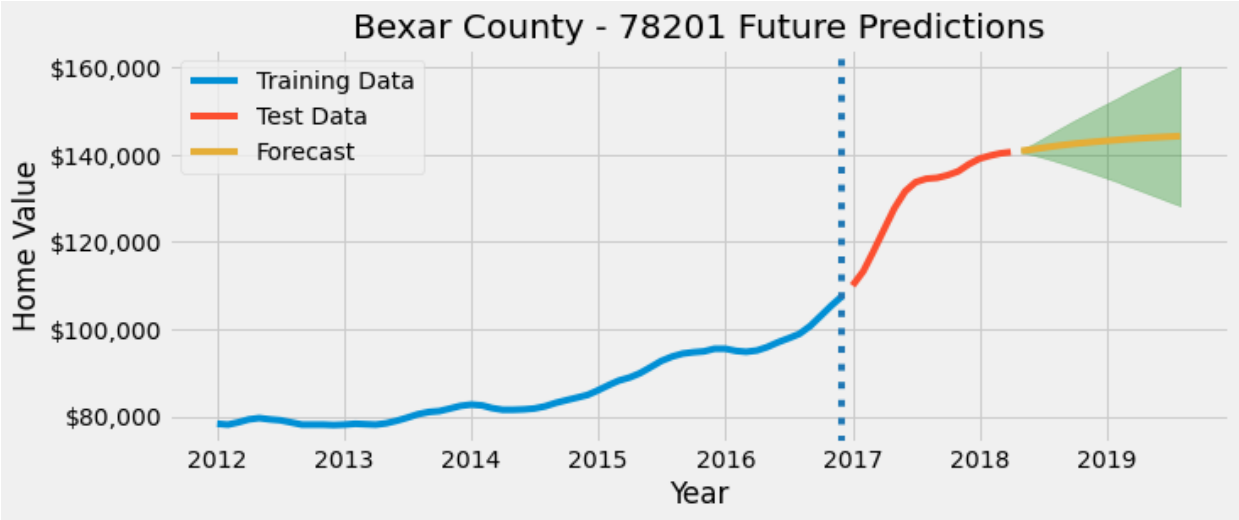
- There is an upward trend beginning in 2013
- Affected more by 2008 crash than other zip codes
- Data is annually seasonal with peaks in months (July-Oct) and dips in spring (Mar-May)
- Seasonality appears constant
- Residuals suggest more variance in 2008 and a similar amount following that point in time
  - Continuously increases
- Looks like data needs 1 degree of differencing on seasonality

In [94]:
```
1 fig_78201, future_78201, forecast_df_78201, roi_78201 = model_predictio
```

executed in 8.33s, finished 09:18:03 2021-06-20







In [95]:
```
1 roi_78201
```

executed in 2ms, finished 09:18:03 2021-06-20

Out[95]:
```
{'Lower CI': [-0.08668127170465845, 91331.87282953416, -8668.127170465843],
 'Upper CI': [0.13400088311669692, 113400.0883116697, 13400.088311669693],
 'Forecast': [0.024010324701326687, 102401.03247013266, 2401.032470132661]}
```

- Current predictions
  - Model captures general trend compared to test predictions, however it undershoots the test data
  - Same shape but shifted below by a constant value
- Future predictions
  - Stays stagnant after test data
- An investment of $100,000 today (05/01/2018) by 08/01/2019 would yield (ROI):
  - Conservative estimate: -8.7% (-$8,668)
  - Mean estimate: 2.3% (+$2,401)
  - Upper estimate: 13.4% (+$13,400)

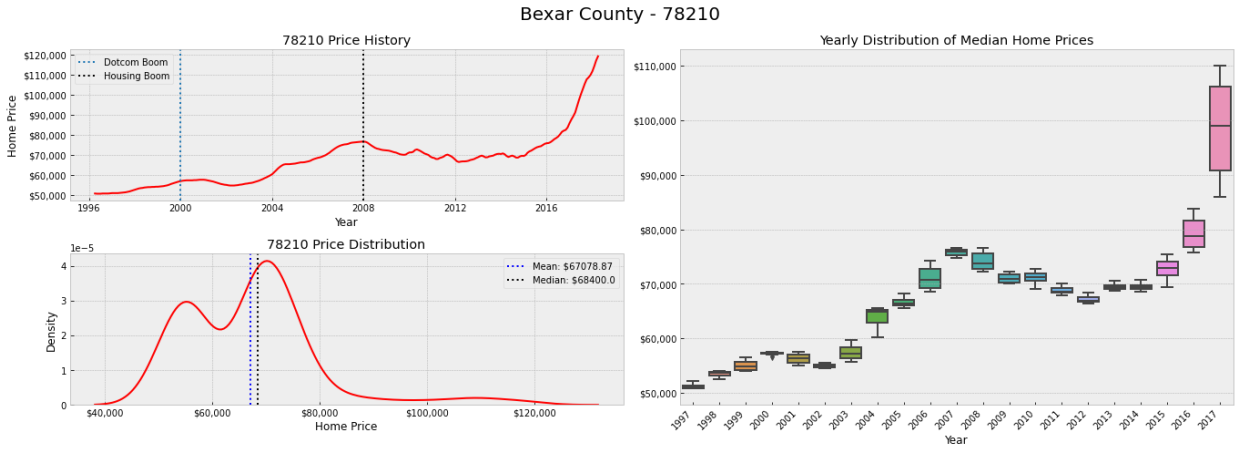
4.3.3 78210: EDA and SARIMAX

- Highland Park, San Antonio
- 3 miles from Downtown San Antonio

In [96]:
```
# Create 78210 dataframe

df_78210=create_zip_data(Bexar_dict_full, 78210)
```
executed in 5ms, finished 09:18:03 2021-06-20

In [97]:
```
zip_eda(df_78210, 78210, 'Bexar')
```
executed in 644ms, finished 09:18:04 2021-06-20

Out[97]:
```
(<Figure size 1368x504 with 3 Axes>,
                value
count      265.000000
mean     67078.867925
std      12593.568233
min      50500.000000
25%      56600.000000
50%      68400.000000
75%      72200.000000
max     119400.000000)
```

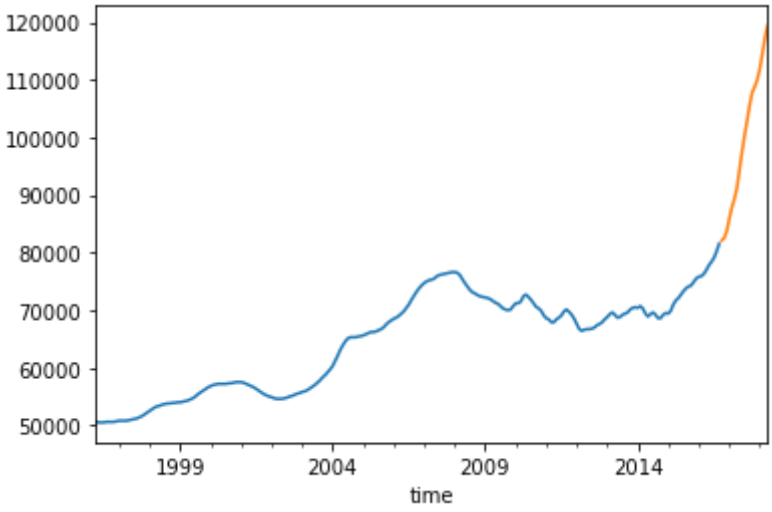


- Small downward trend after dotcom boom with a rise up to 2008 and then a big fall
  - Median house price recovered in 2015
- Data is slightly right skewed
- Data appears sort of bimodal suggesting there were swings when prices were much lower and now much higher, was not a continuous shift
  - There was a peak price around 2006 and then a new peak price reached in 2017
- Not too much spread between home prices except for in 2006 and a very large spread in 2017

In [98]:
```
train_78210, test_78210 = create_train_test_split(df_78210, 0.93)
train_78210.plot()
test_78210.plot()
```

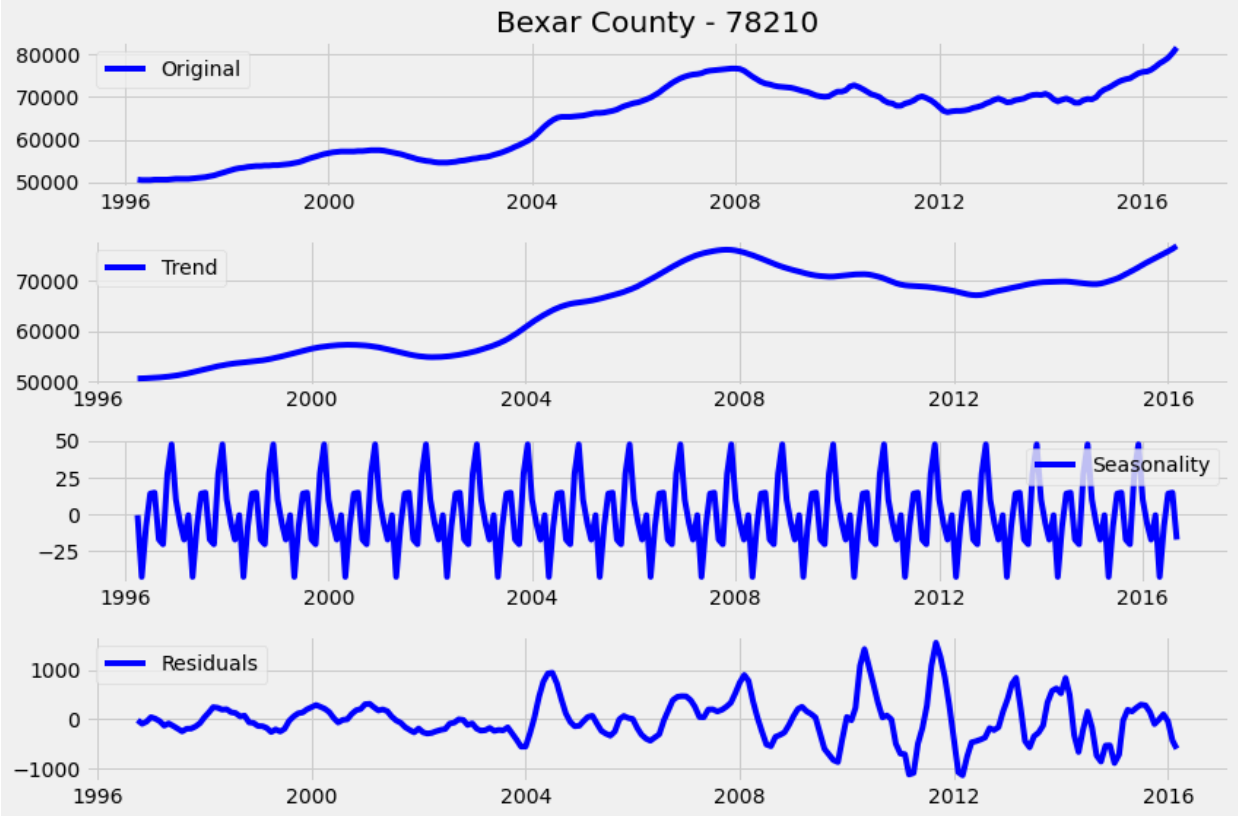
executed in 101ms, finished 09:18:04 2021-06-20

Out[98]: <AxesSubplot:xlabel='time'>

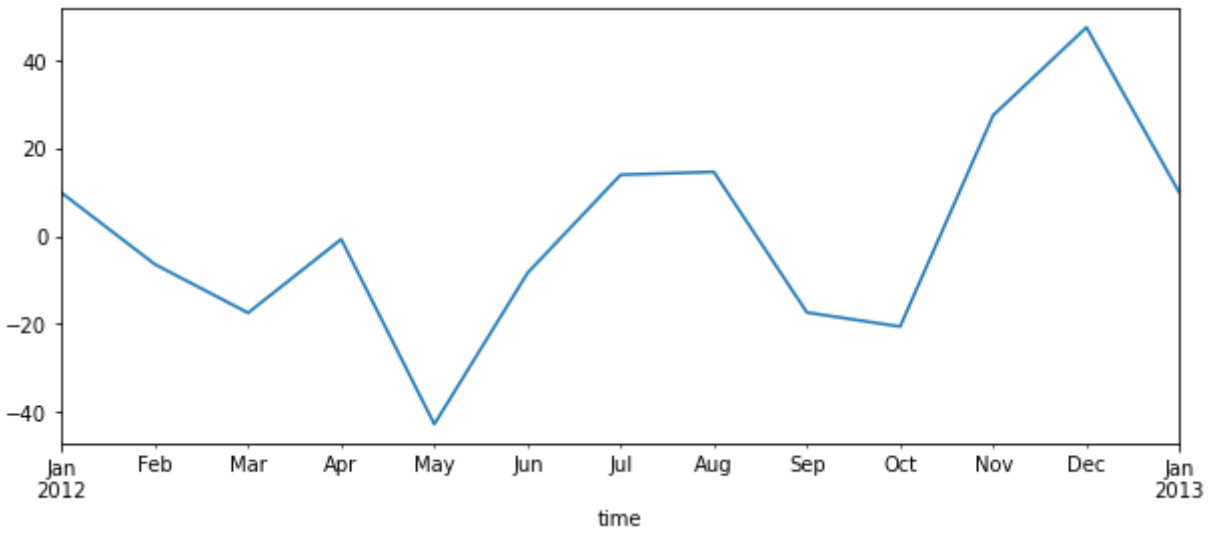


- Original split at 0.90 but deviated to 0.93 to better capture trend

In [99]:
```
1 seasonal_decomposition(train_78210, 'Bexar', 78210)
```
executed in 321ms, finished 09:18:04 2021-06-20



In [100]:
```
1 decomposition = seasonal_decompose(train_78210,model='Additive')
2 seasonal = decomposition.seasonal
3 seasonal.plot(figsize=(10,4), xlim=('2012-01-01', '2013-01-01'));
```
executed in 96ms, finished 09:18:04 2021-06-20

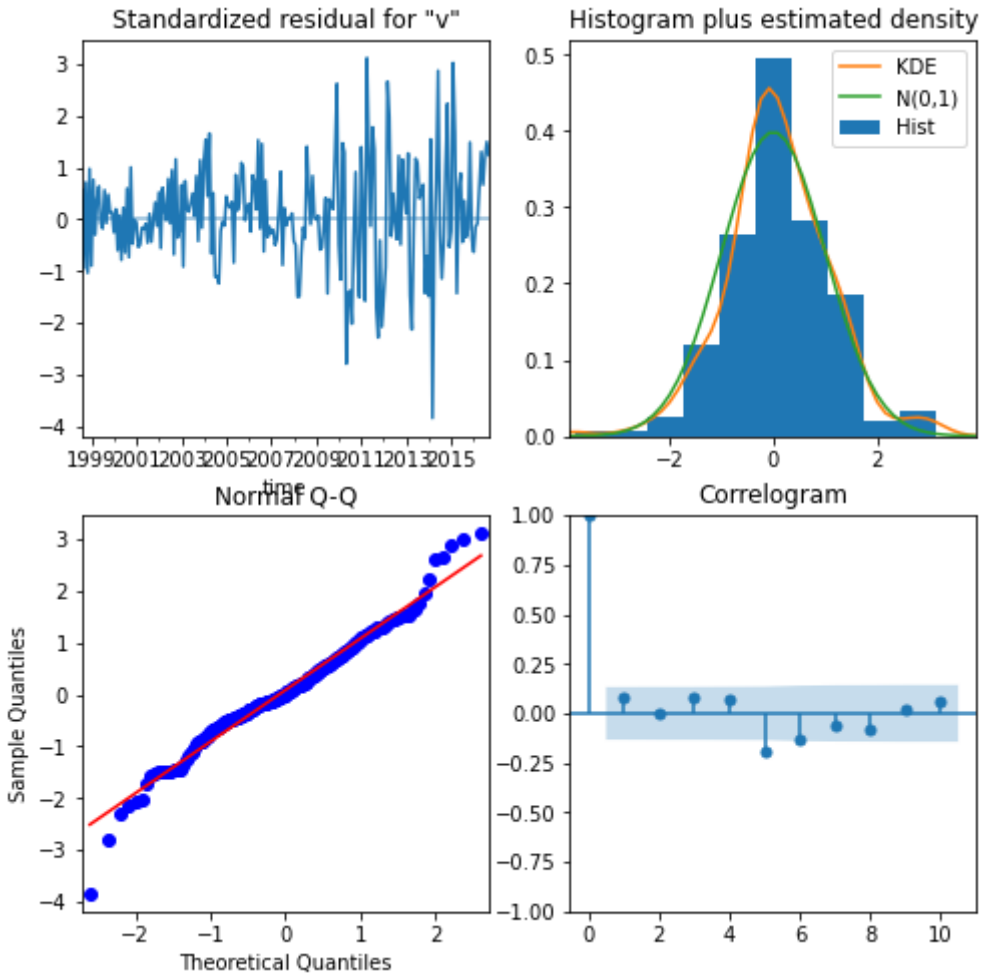


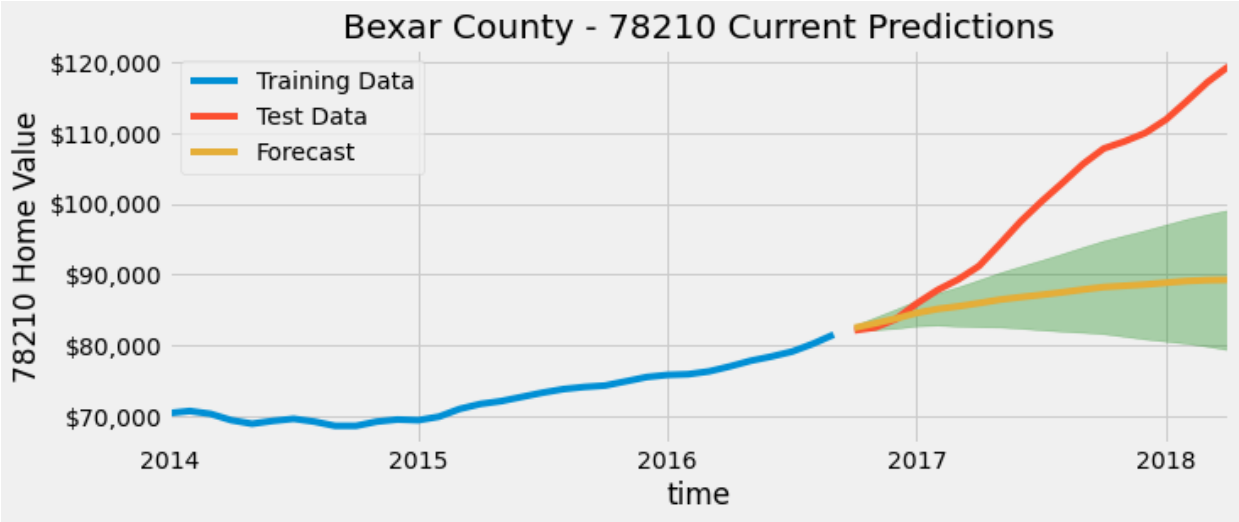
- There is an upward trend beginning in 2013
- Big downward swing in trend in 2008 and 2002
- Data is annually seasonal with peaks in months (Oct-Jan) and dips in spring (Apr-June)
- Seasonality appears constant
- Residuals have the most variance in 2012

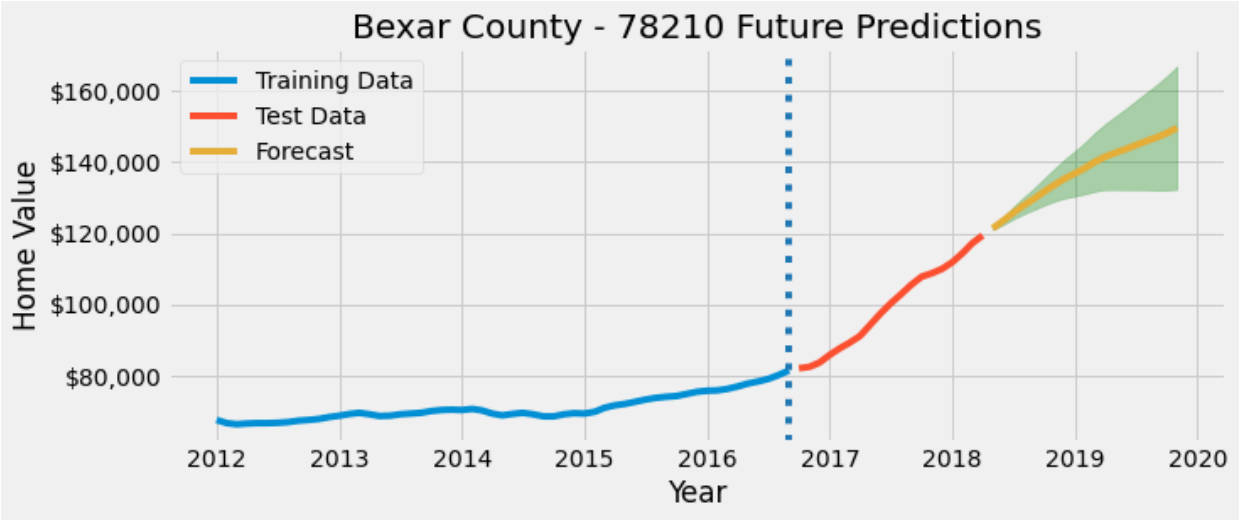
- Looks like data needs 1 degree of differencing on seasonality

In [101]:
```
1 fig_78210, future_78210, forecast_df_78210, roi_78210 = model_predictio
```

executed in 1m 38.9s, finished 09:19:43 2021-06-20







In [102]:
```
1 roi_78210
```
executed in 3ms, finished 09:19:43 2021-06-20

Out[102]:
```
{'Lower CI': [0.09419980807344343, 109419.98080734434, 9419.98080734434
2],
 'Upper CI': [0.3700735265708664, 137007.35265708662, 37007.35265708662],
 'Forecast': [0.23263775992395297, 123263.77599239531, 23263.77599239531
3]}
```

- Current predictions
  - Model captures general trend poorly especially as the test values continue into 2018
  - The predictions shift downward which is the opposite of what truly occurs
- Future predictions
  - Continues upward with a slight downward bend in mid-2019
- An investment of $100,000 today (05/01/2018) by 11/01/2019 would yield (ROI):
  - Conservative estimate: 9.4% ($9,419)
  - Mean estimate: 23.3% (+$23,263)
  - Upper estimate: 37.0% (+$37,007)

4.4 Bexar County Conclusion

In [103]:
```
1 corr_check(df_78212, df_78201, df_78210, 78212, 78201, 78210)
```
executed in 5ms, finished 09:19:43 2021-06-20

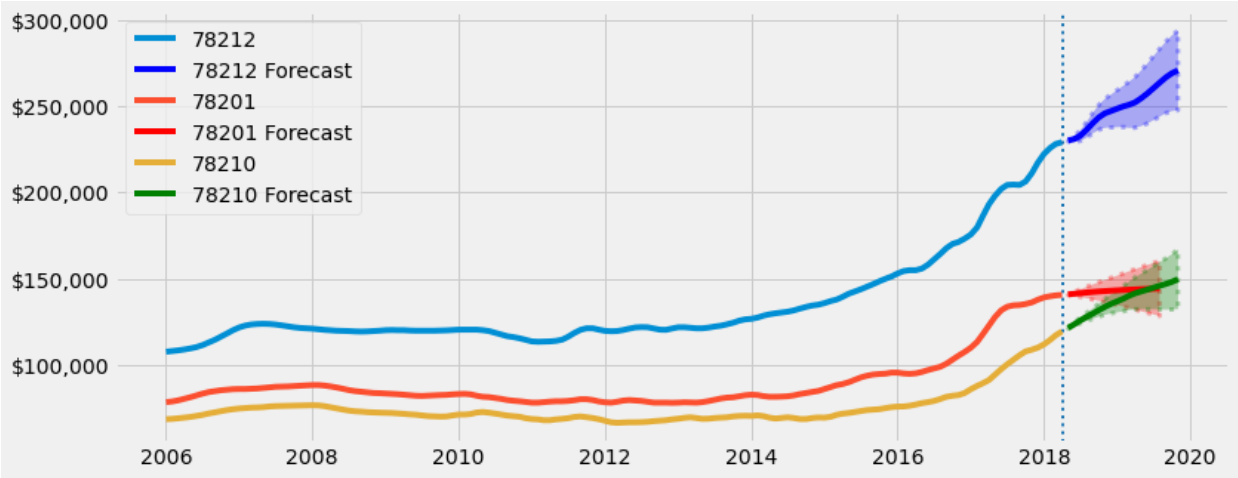
Out[103]:

	78212	78201	78210
78212	1.000000	0.985177	0.962089
78201	0.985177	1.000000	0.984328
78210	0.962089	0.984328	1.000000

- 78212 moves more closely with 78201
- 78201 moves equally closely with 78212 and 78210
- 78210 moves more closely with 78701

- This can be useful for picking up on trends using other nearby zip codes. They all move very closely together

In [104]:
```
1 county_forecast_comparison(df_78212, 78212, forecast_df_78212, df_78201
```
executed in 133ms, finished 09:19:43 2021-06-20



- 78212 has the highest ending point with the tightest confidence interval
- 78201 has almost as high of upside potential as 78210 and they are both relatively close
  - Curves downward at the end slightly
- 78210 follows a similar trajectory as 78201 but has a more upward slope towards the end of the predictions

In [105]:
```
1 bexar_perc_comparison = county_forecast_perc_comparison(roi_78212, 7821
2 bexar_perc_comparison
```
executed in 8ms, finished 09:19:43 2021-06-20

Out[105]:

	Lower CI	Upper CI	Forecast
78212 Perc Change	0.079445	0.273619	0.176802
Val	107944.511492	127361.853145	117680.210000
$ Diff	7944.511492	27361.853145	17680.210000
78201 Perc Change	-0.086681	0.134001	0.024010
Val	91331.872830	113400.088312	102401.032470
$ Diff	-8668.127170	13400.088312	2401.032470
78210 Perc Change	0.094200	0.370074	0.232638
Val	109419.980807	137007.352657	123263.775992
$ Diff	9419.980807	37007.352657	23263.775992

- Based on the downside risk, upside return, and mean predicted value, 78210 seems like the superior zip code
- It has the highest upside, the highest predictions

- 78212 has promising predictions but it does not have as much upside, however it does have a more predictable range of values
- **In Bexar county, 78210 has the best prospects for near term growth**

4.5 Harris County Modeling

- Located in Houston metro
- Population: 4,698,619

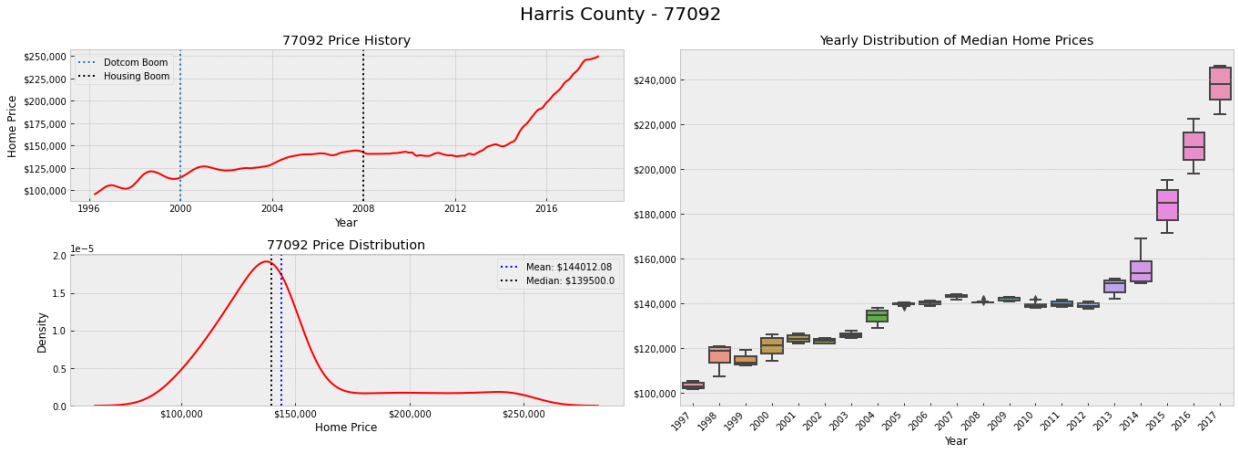
4.5.1 77092: EDA and SARIMAX

- Fairbanks Houston
- 11.5 miles from downtown Houston

In [106]:
```
# Create 78210 dataframe

df_77092=create_zip_data(Harris_dict_full, 77092)
```
executed in 6ms, finished 09:19:43 2021-06-20

In [107]:
```
zip_eda(df_77092, 77092, 'Harris')
```
executed in 638ms, finished 09:19:44 2021-06-20

Out[107]:
```
(<Figure size 1368x504 with 3 Axes>,
                value
count      265.000000
mean    144012.075472
std      33847.416163
min      95600.000000
25%     124000.000000
50%     139500.000000
75%     143700.000000
max     249300.000000)
```



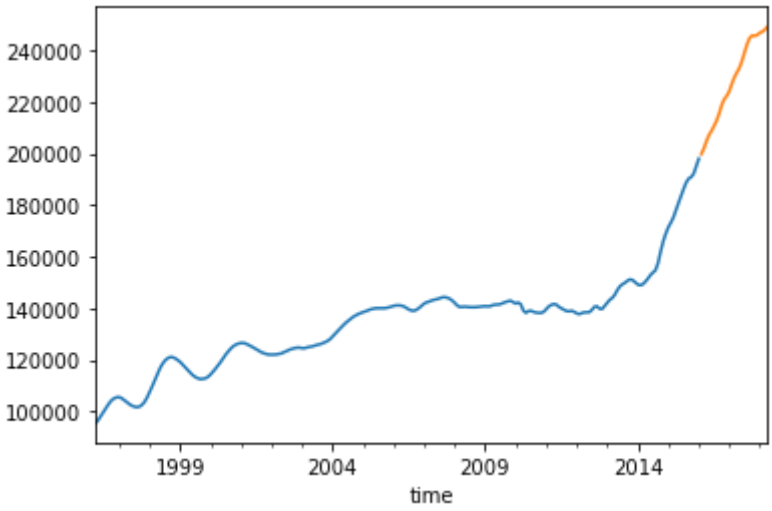
- Small downward trend after dotcom boom with a rise up to 2008 and then a gradual leveling until an increase in 2014
  - Median house price recovered in 2014 but was not significant affected

- Data is slightly right skewed
- Based on the shape there are more homes that are more expensive than the average than less which reflects the recent upward trend the zip code has been experiencing
  - Prices have steadily risen
- Minimal spread between years unitl 2014 and then spread has gradually increased

In [108]:
```
train_77092, test_77092 = create_train_test_split(df_77092, 0.90)
train_77092.plot()
test_77092.plot()
```

executed in 124ms, finished 09:19:44 2021-06-20

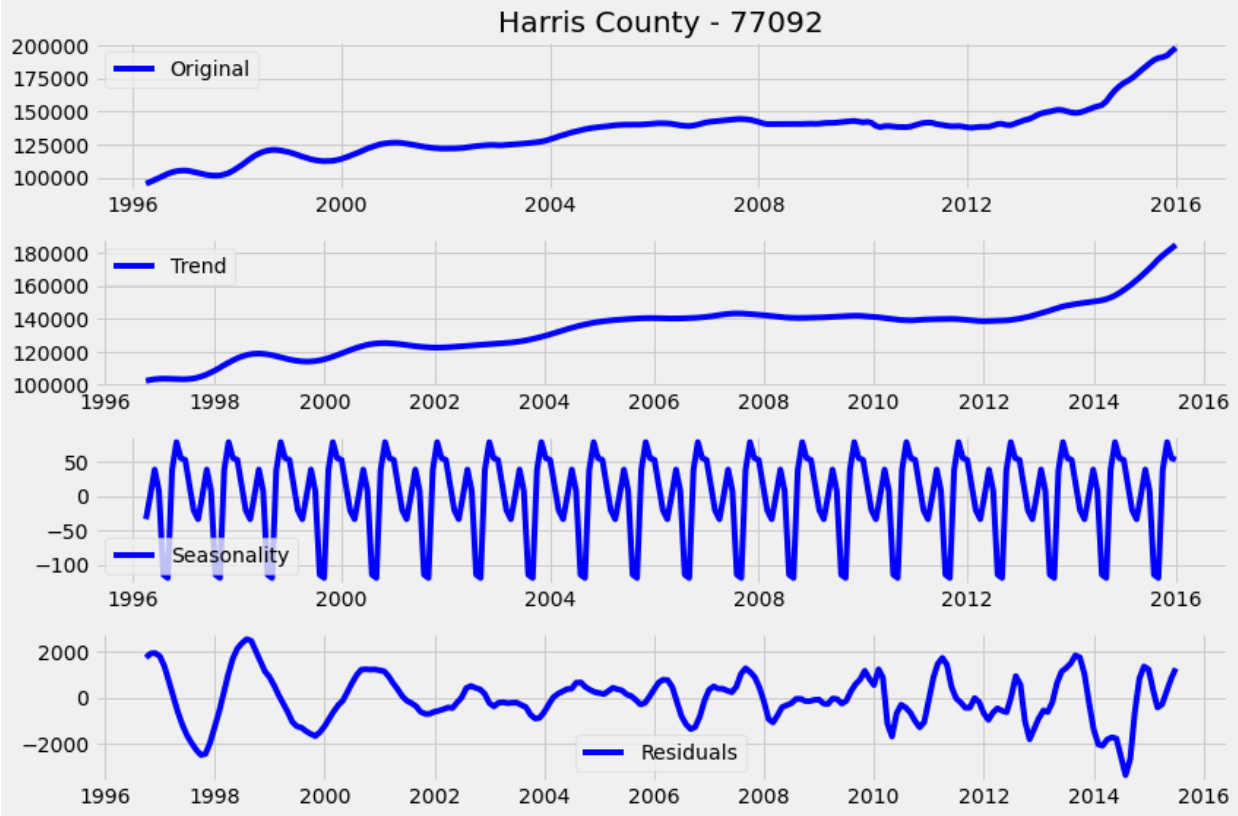
Out[108]: <AxesSubplot:xlabel='time'>



Baseline split at 0.90

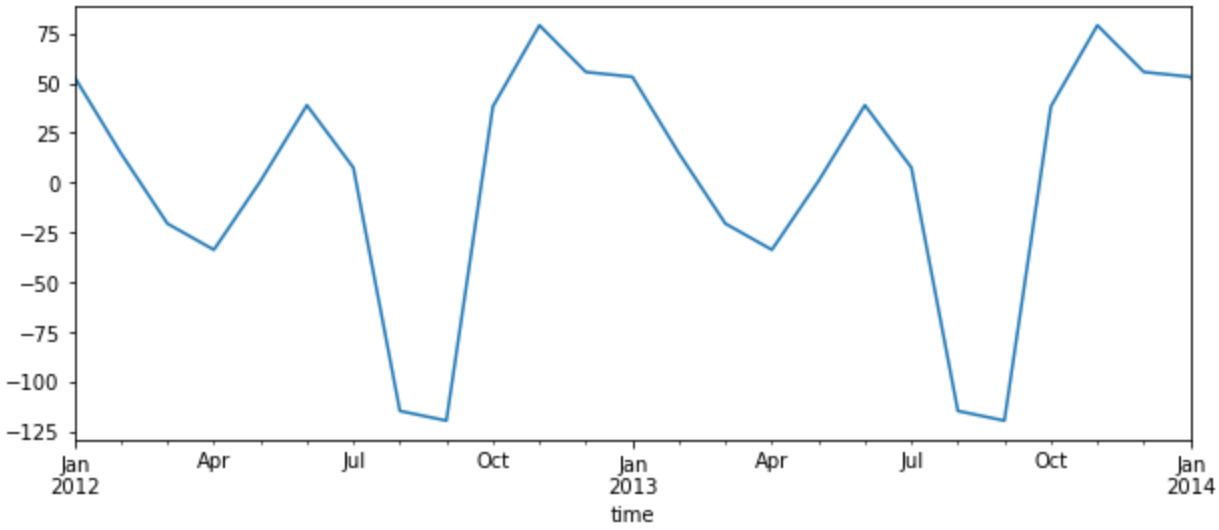
In [109]:
```
1 seasonal_decomposition(train_77092, 'Harris', 77092)
```

executed in 429ms, finished 09:19:45 2021-06-20



In [110]:
```
decomposition = seasonal_decompose(train_77092,model='Additive')
seasonal = decomposition.seasonal
seasonal.plot(figsize=(10,4), xlim=('2012-01-01', '2014-01-01'));
```

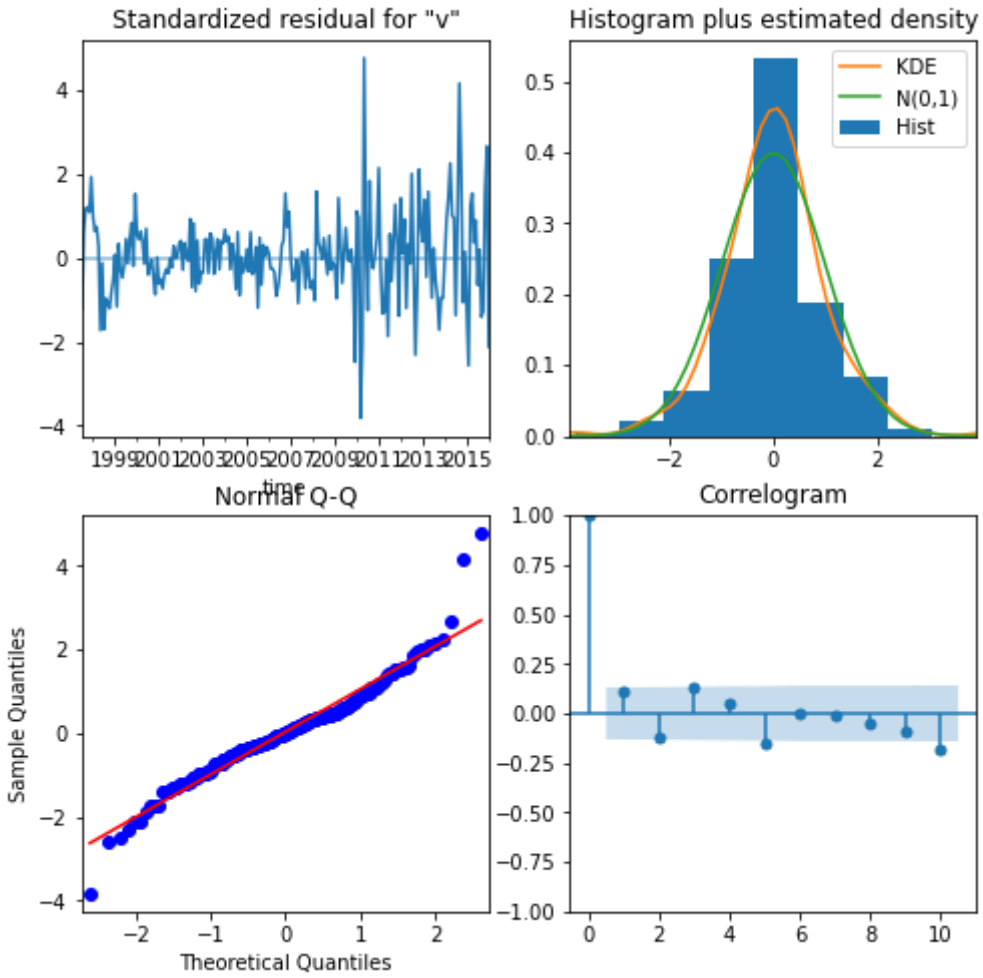
executed in 120ms, finished 09:19:45 2021-06-20

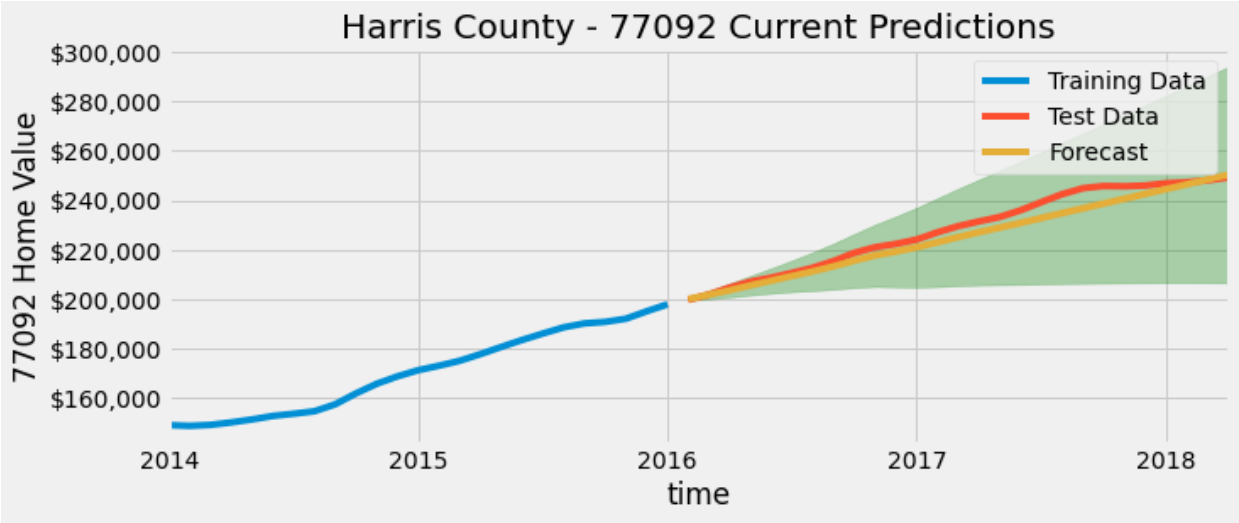


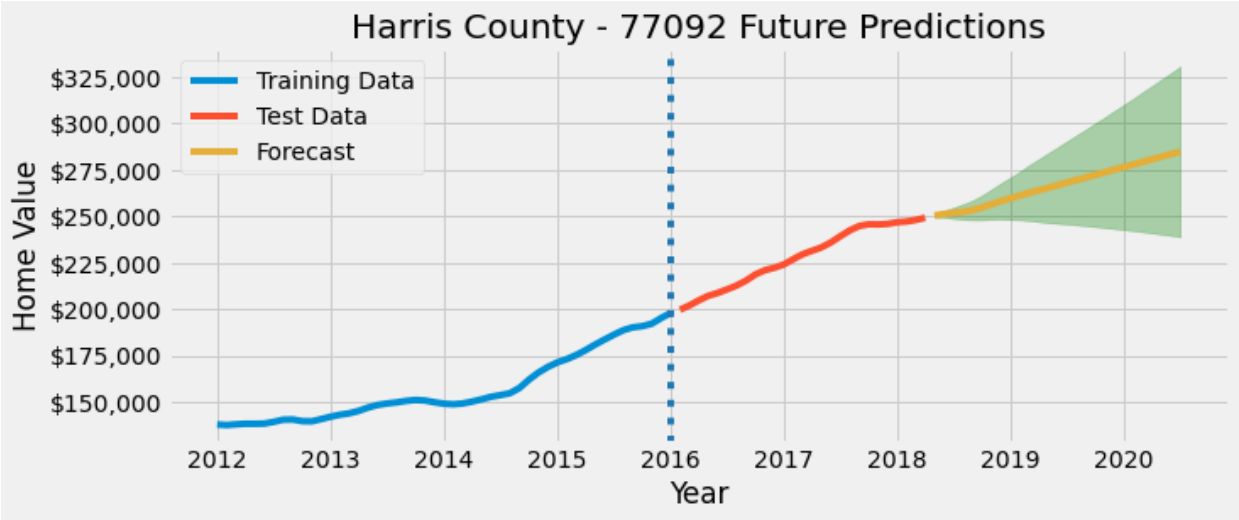
- There is an upward trend beginning in 2014
  - Prior, trend was stagnant
- Data is annually seasonal with peaks in months (Oct-Jan) and dips in spring (June-Aug)
- Seasonality appears constant
- Residuals have the most variance up to 2000 and past 2012
- Looks like data needs 1 degree of differencing on seasonality

In [111]:
```
1 fig_77092, future_77092, forecast_df_77092, roi_77092 = model_predictio
```

executed in 20.9s, finished 09:20:06 2021-06-20







In [112]:
```
1 roi_77092
```

executed in 2ms, finished 09:20:06 2021-06-20

Out[112]:
```
{'Lower CI': [-0.04321599776274212, 95678.4002237258, -4321.599776274204
5],
 'Upper CI': [0.31696593080350516, 131696.59308035052, 31696.59308035051
8],
 'Forecast': [0.13736790632659834, 113736.79063265985, 13736.79063265984
6]}
```

- Current predictions
  - Model captures general trend very well, stays tight with the real values
  - Slight noise in true values that the prediction does not pick up on
- Future predictions
  - Increases upward slope with a wide confidence interval
- An investment of $100,000 today (05/01/2018) by 07/01/2020 would yield (ROI):
  - Conservative estimate: -4.3% (-$4,321)
  - Mean estimate: 13.7% (+$13,736)

- Upper estimate: 31.7% (+$31,696)

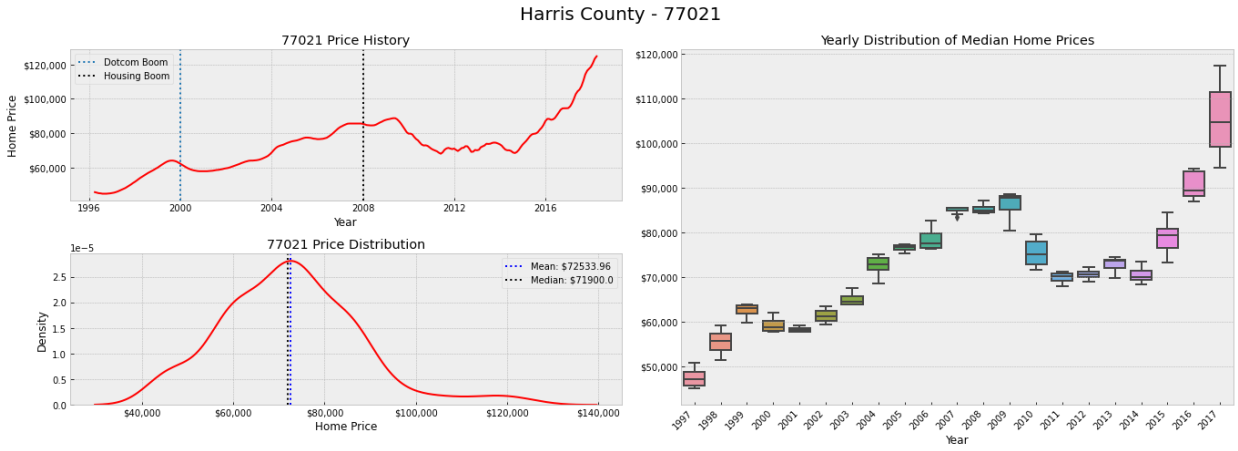
4.5.2 77021: EDA and SARIMAX

- South Houston
- 6.4 miles from downtown Houston

In [113]:
```
# Create 782021 dataframe

df_77021=create_zip_data(Harris_dict_full, 77021)
```
executed in 5ms, finished 09:20:06 2021-06-20

In [114]:
```
zip_eda(df_77021, 77021, 'Harris')
```
executed in 728ms, finished 09:20:06 2021-06-20

Out[114]:
```
(<Figure size 1368x504 with 3 Axes>,
                value
count      265.000000
mean     72533.962264
std      15263.205204
min      44600.000000
25%      61900.000000
50%      71900.000000
75%      81800.000000
max     124700.000000)
```

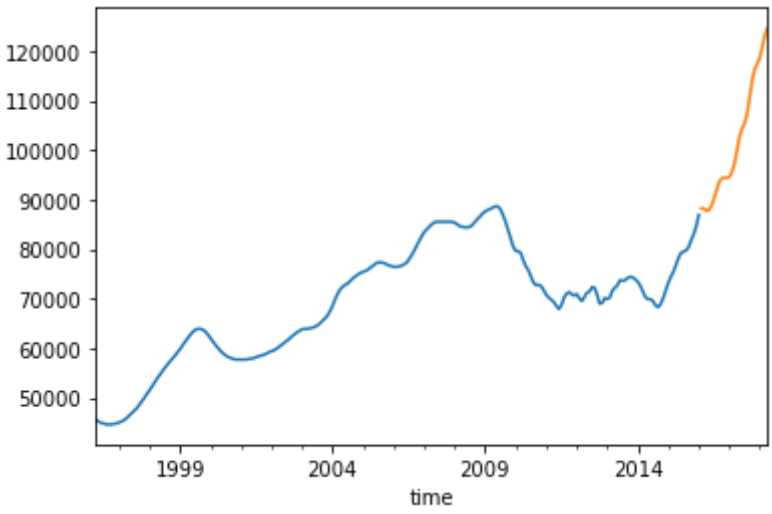


- Small downward trend after dotcom boom with a rise up to 2008 and then a crash until 2015
  - Median house price recovered in 2016, pretty strongly affected
- Data is slightly right skewed
- Based on the shape there are more homes that are more expensive than the average than less which reflects the recent upward trend the zip code has been experiencing
  - Prices have steadily risen since 2016
- Minimal spread between years unitl 2008 and then spread has been increasing again since 2016

In [115]:
```
train_77021, test_77021 = create_train_test_split(df_77021, 0.90)
train_77021.plot()
test_77021.plot()
```

executed in 112ms, finished 09:20:07 2021-06-20

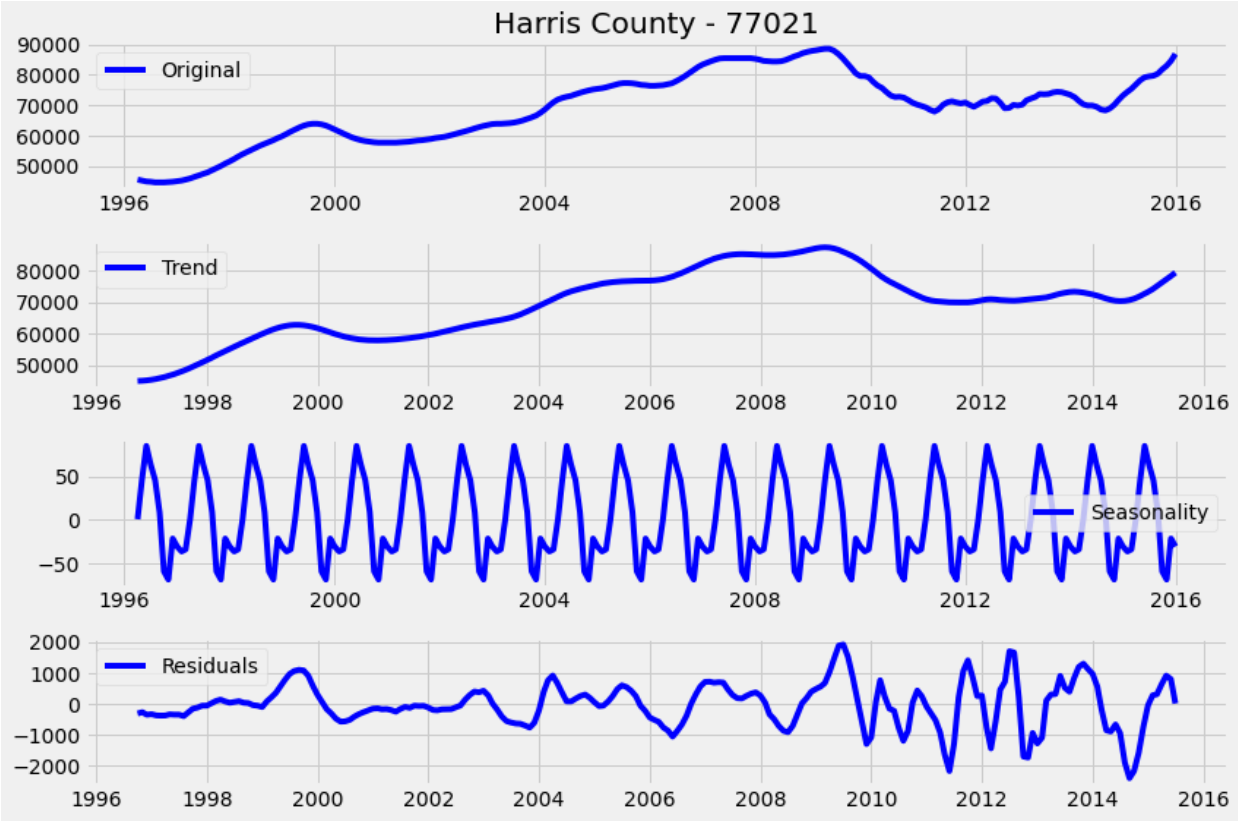
Out[115]: <AxesSubplot:xlabel='time'>



Baseline split point 0.90

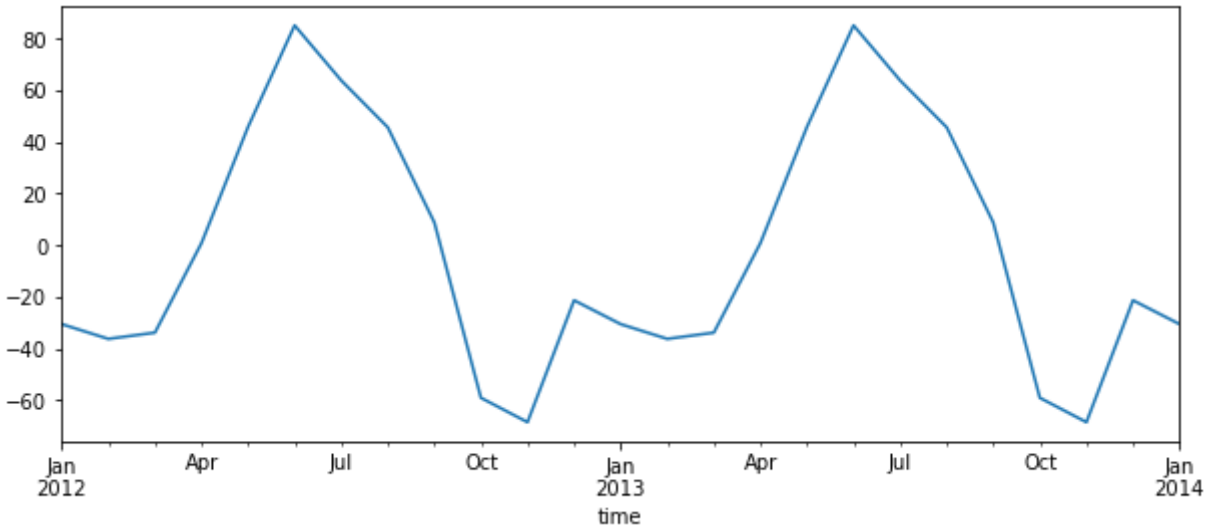
In [116]:
```
seasonal_decomposition(train_77021, 'Harris', 77021)
```

executed in 368ms, finished 09:20:07 2021-06-20



In [117]:
```
decomposition = seasonal_decompose(train_77021,model='Additive')
seasonal = decomposition.seasonal
seasonal.plot(figsize=(10,4), xlim=('2012-01-01', '2014-01-01'));
```

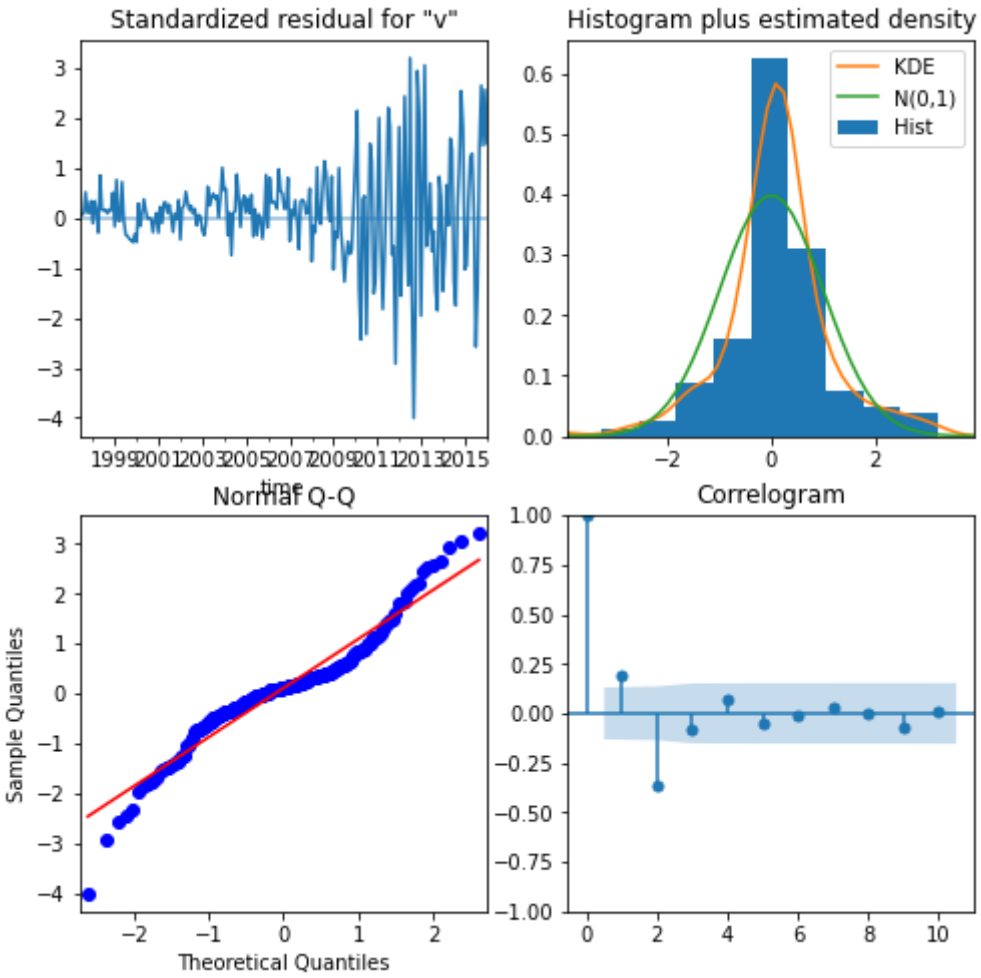
executed in 117ms, finished 09:20:07 2021-06-20

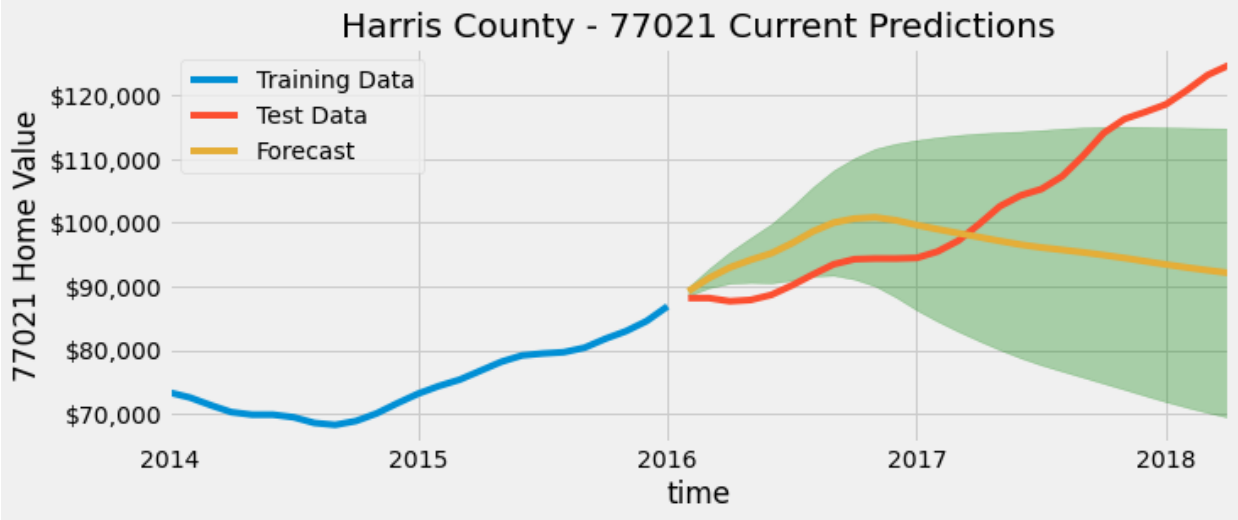


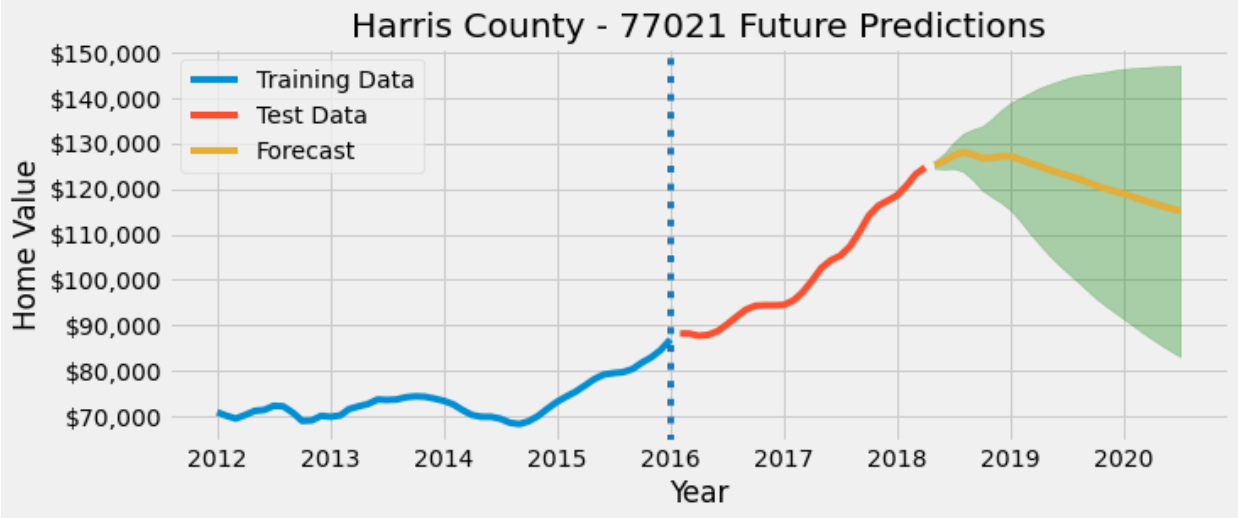
- There is an upward trend from 2002 to 2009 and then it dips and begins to recover in 2013
- Data is annually seasonal with peaks in months (April-August) and dips in spring (Sept-Jan)
- Seasonality appears constant
- Residuals have the most variance between 2010 and 2016
- Looks like data needs 1 degree of differencing on seasonality

In [118]:
```
1 fig_77021, future_77021, forecast_df_77021, roi_77021 = model_predictio
```

executed in 16.6s, finished 09:20:24 2021-06-20







In [119]:
```
1 roi_77021
```

executed in 2ms, finished 09:20:24 2021-06-20

Out[119]:
```
{'Lower CI': [-0.3322637747665876 4, 66773.62252334124, -33226.3774766587 6],
 'Upper CI': [0.16710466083824432, 116710.46608382443, 16710.4660838244 3],
 'Forecast': [-0.08097462904054722, 91902.53709594528, -8097.46290405471 5]}
```

- Current predictions
  - Model captures general trend very well in terms of confidence interval but skews down as test data goes up

- Future predictions
  - Deviates significant from the trend and tilts downward as the past data goes up
- An investment of $100,000 today (05/01/2018) by 07/01/2020 would yield (ROI):
  - Conservative estimate: -33.2% (-$33,226)
  - Mean estimate: -8.0% (-$8,097)
  - Upper estimate: 16.7% (+$16,710)

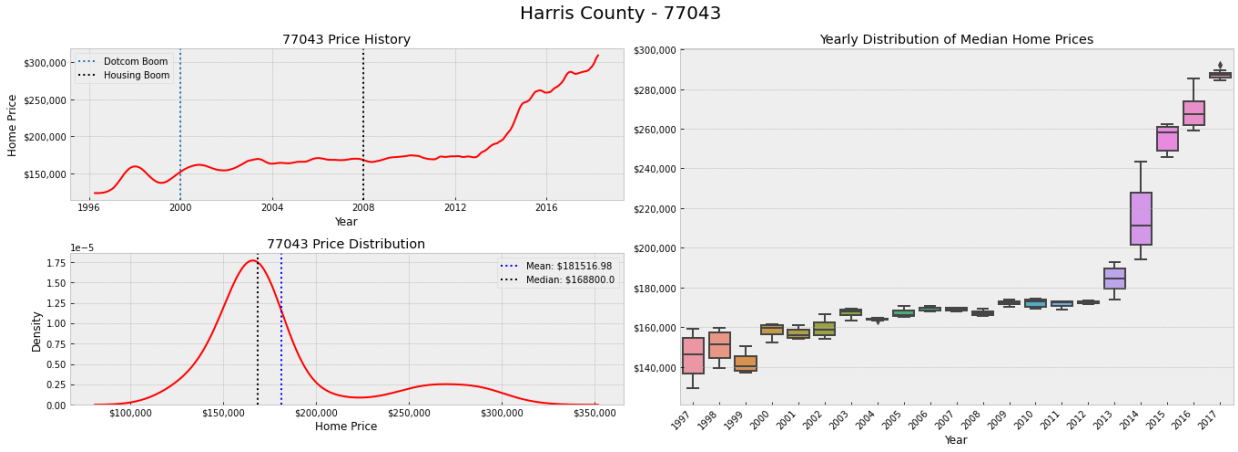
4.5.3 77043: EDA and SARIMAX

- Spring Branch Houston
- 15.6 miles from downtown Houston

In [120]:
```
# Create 77043 dataframe

df_77043=create_zip_data(Harris_dict_full, 77043)
```
executed in 4ms, finished 09:20:24 2021-06-20

In [121]:
```
zip_eda(df_77043, 77043, 'Harris')
```
executed in 657ms, finished 09:20:24 2021-06-20

Out[121]:
```
(<Figure size 1368x504 with 3 Axes>,
                value
count      265.000000
mean    181516.981132
std      42886.499853
min     123300.000000
25%     159400.000000
50%     168800.000000
75%     174000.000000
max     309500.000000)
```



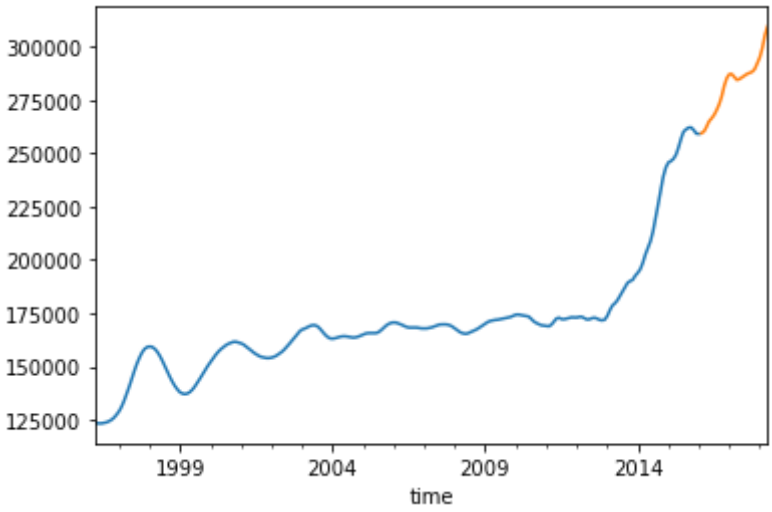
- Prices were relatively stable after 2008, stayed stagnant, and then moved up in 2013
  - Median house price was not affected by 2008 crash
- Data is slightly right skewed
- Based on the shape there are more homes that are more expensive than the average than less which reflects the recent upward trend the zip code has been experiencing
  - Prices have steadily risen since 2013

- Minimal spread between years unitl 2013 and then spread began to decrease in 2015

In [122]:
```
train_77043, test_77043 = create_train_test_split(df_77043, 0.90)
train_77043.plot()
test_77043.plot()
```

executed in 117ms, finished 09:20:24 2021-06-20

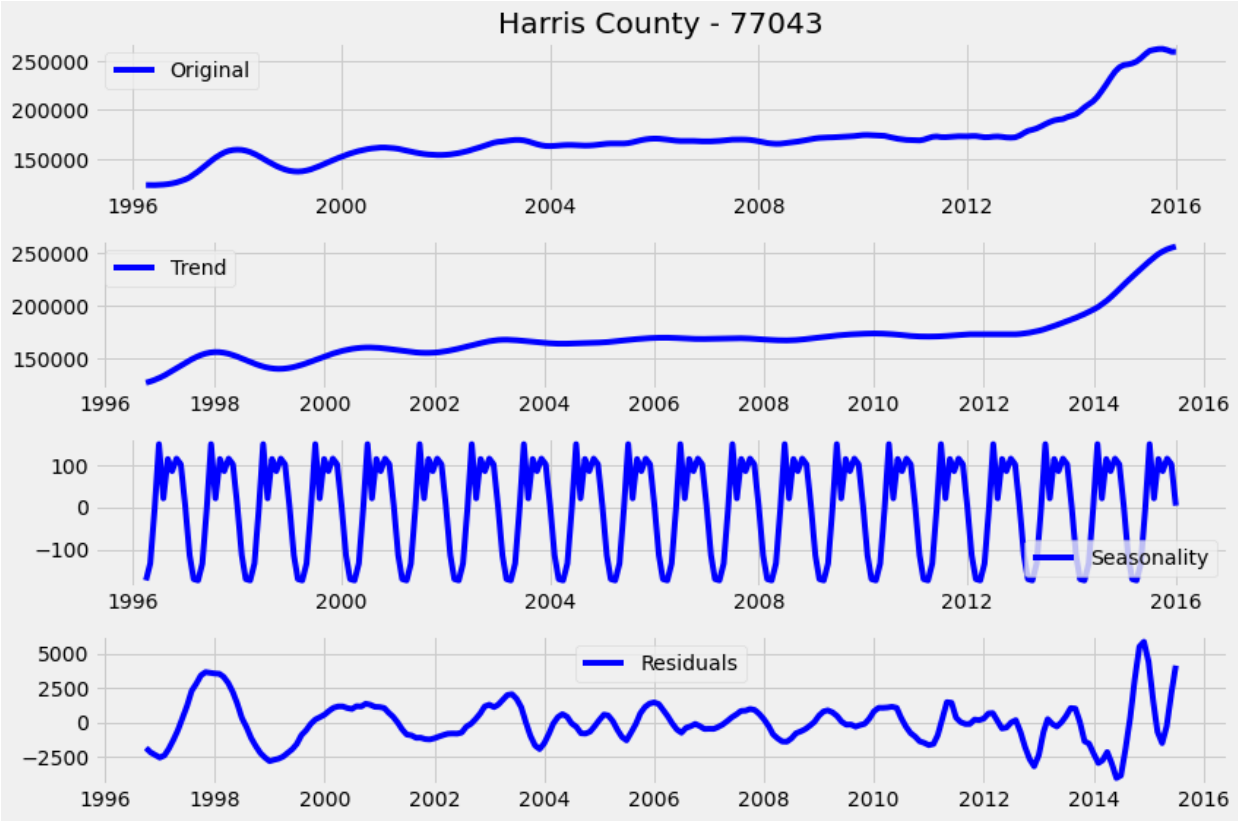
Out[122]: <AxesSubplot:xlabel='time'>



Baseline split point of 0.90

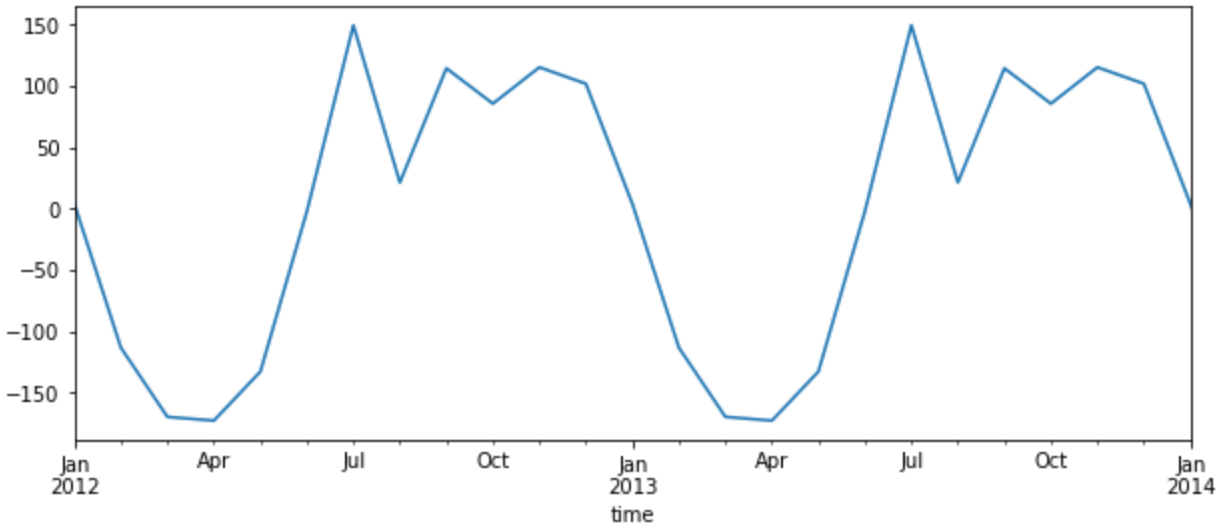
In [123]:
```
seasonal_decomposition(train_77043, 'Harris', 77043)
```

executed in 450ms, finished 09:20:25 2021-06-20



In [124]:
```
decomposition = seasonal_decompose(train_77043,model='Additive')
seasonal = decomposition.seasonal
seasonal.plot(figsize=(10,4), xlim=('2012-01-01', '2014-01-01'));
```

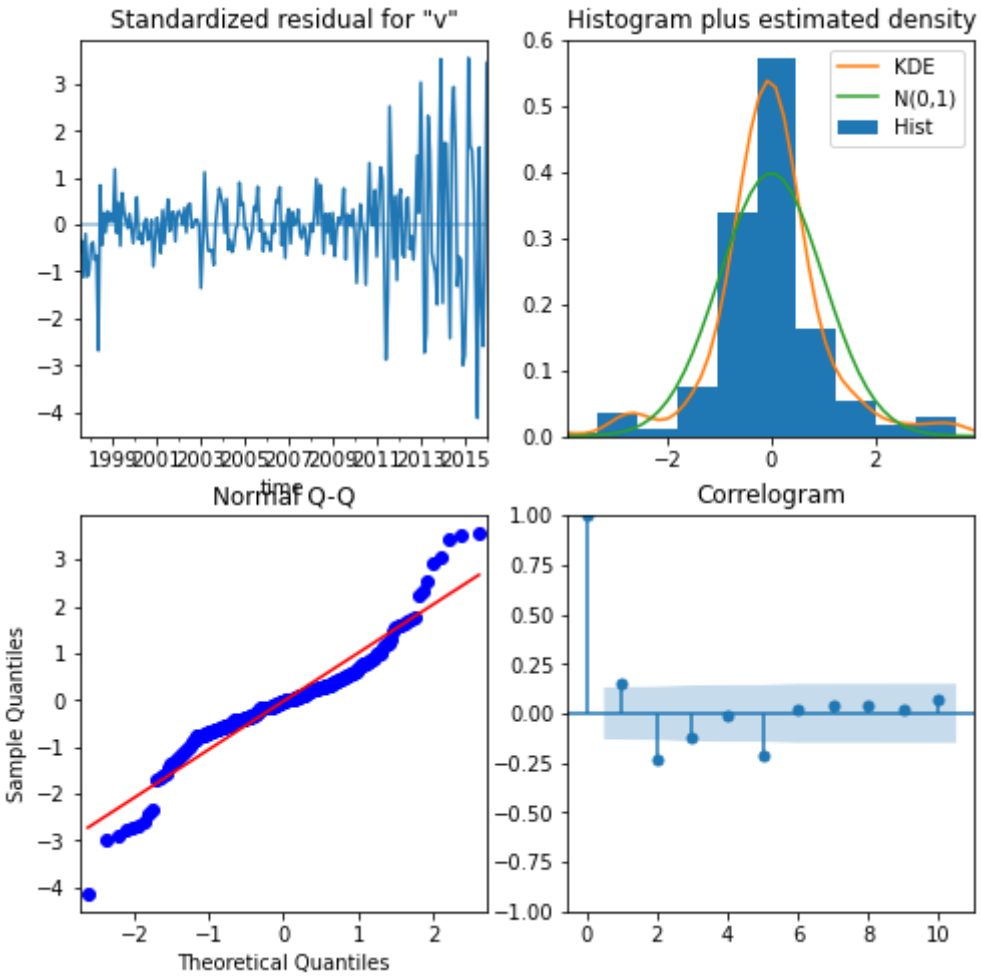
executed in 114ms, finished 09:20:25 2021-06-20

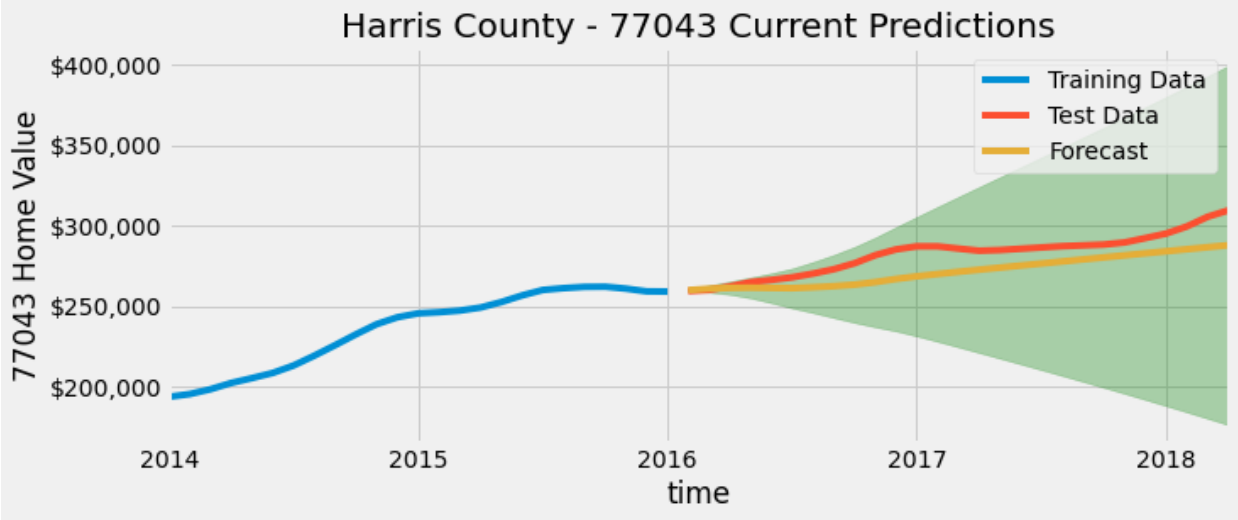


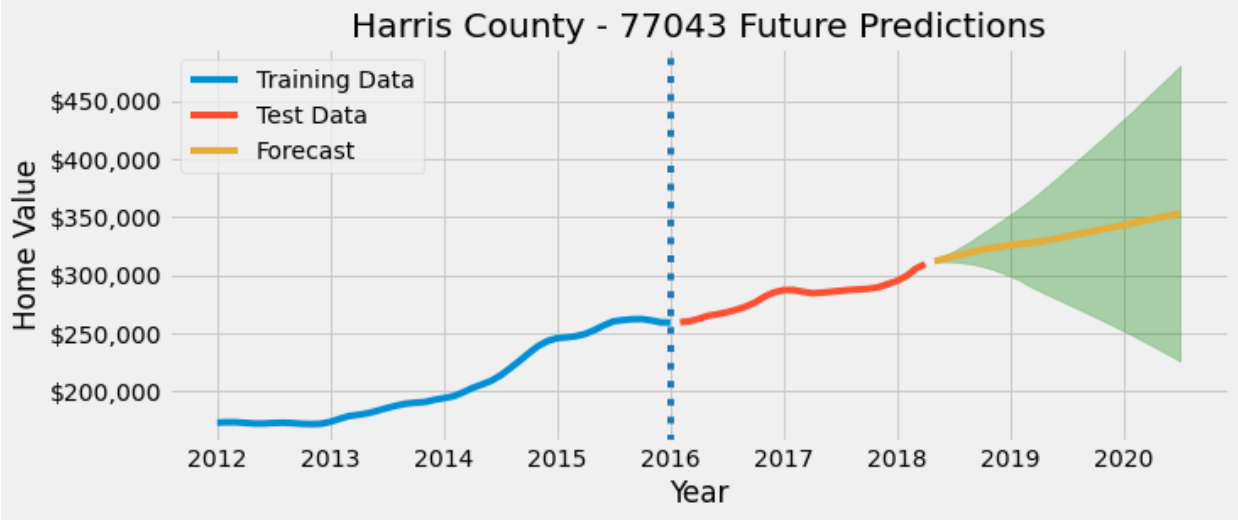
- Data is stangant until 2013 when an upward trend begins
- Data is annually seasonal with peaks in months (July-Oct) and dips in spring (Jan-April)
- Seasonality appears constant
- Residuals have the most variance between 2014 and 2016
- Looks like data needs 1 degree of differencing on seasonality

In [125]:
```
1 fig_77043, future_77043, forecast_df_77043, roi_77043 = model_predictio
```

executed in 11.3s, finished 09:20:36 2021-06-20







In [126]:
```
1 roi_77043
```

executed in 2ms, finished 09:20:36 2021-06-20

Out[126]:
```
{'Lower CI': [-0.2743308781323286, 72566.91218676713, -27433.08781323286
5],
 'Upper CI': [0.5360134350550573, 153601.34350550573, 53601.34350550573
4],
 'Forecast': [0.13213589068135906, 113213.5890681359, 13213.58906813590
1]}
```

- Current predictions
  - Model captures general trend very well in terms of confidence interval

    - Makes sense that predictions are relatively flat given there is not much trend to be captured
- Future predictions
    - Titls slightly upward
    - Has an extremely large confidence interval
- An investment of $100,000 today (05/01/2018) by 07/01/2020 would yield (ROI):
    - Conservative estimate: -27.4% (-$27,433)
    - Mean estimate: 13.2% (-$13,213)
    - Upper estimate: 53.6% (+$53,601)

4.6 Harris County Conclusion

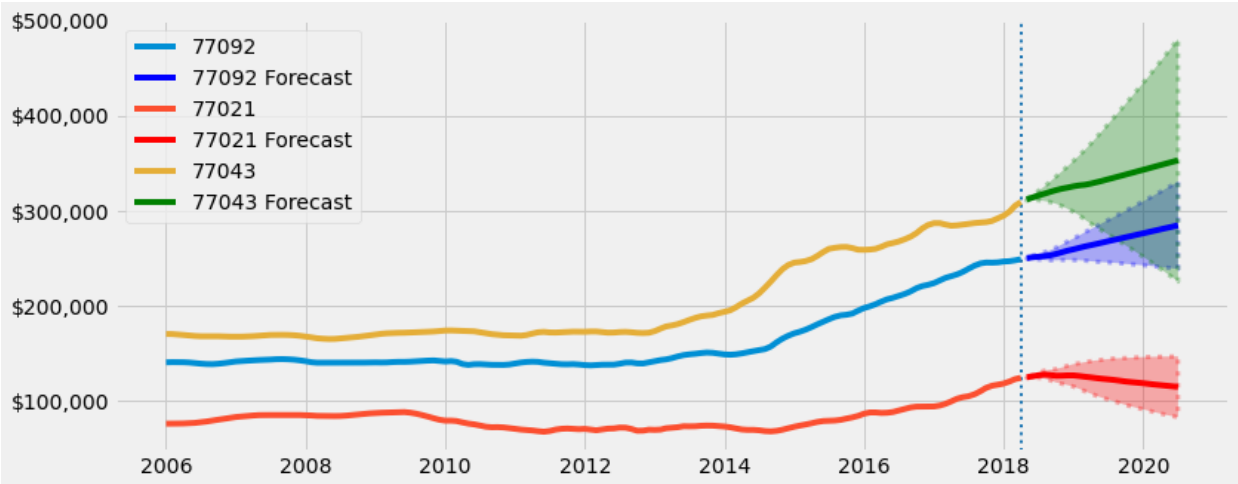
In [127]:
```
1 corr_check(df_77092, df_77021, df_77043, 77092, 77021, 77043)
```
executed in 5ms, finished 09:20:36 2021-06-20

Out[127]:

	77092	77021	77043
77092	1.000000	0.863189	0.968172
77021	0.863189	1.000000	0.759268
77043	0.968172	0.759268	1.000000

- 77092 moves more closely with 77043, faily large margin
- 77021 moves more closely with 77092
- 77043 moves more closely with 77092
- This can be useful for picking up on trends using other nearby zip codes. Especially useful because there are large differences in correlation

In [128]:
```
1 county_forecast_comparison(df_77092, 77092, forecast_df_77092, df_77021
```
executed in 139ms, finished 09:20:36 2021-06-20



- 77043 has the highest ending point but also has a very large confidence interval
- 77092 moves steadily upward and has an upwards slope but a much tigher range
    - Curves upward at the end slightly

- 77021 trends lower in many years and trends down towards the end in a tight confidence interval

In [129]:
```
harris_perc_comparison = county_forecast_perc_comparison(roi_77092, 770
harris_perc_comparison
```
executed in 6ms, finished 09:20:36 2021-06-20

Out[129]:

	Lower CI	Upper CI	Forecast
77092 Perc Change	-0.043216	0.316966	0.137368
Val	95678.400224	131696.593080	113736.790633
$ Diff	-4321.599776	31696.593080	13736.790633
77021 Perc Change	-0.332264	0.167105	-0.080975
Val	66773.622523	116710.466084	91902.537096
$ Diff	-33226.377477	16710.466084	-8097.462904
77043 Perc Change	-0.274331	0.536013	0.132136
Val	72566.912187	153601.343506	113213.589068
$ Diff	-27433.087813	53601.343506	13213.589068

- Based on the downside risk, upside return, and mean predicted value, 77092 seems like the superior zip code
- It has the second highest upside, the highest predicted mean
- 77021 has a lower predicted mean forecast and a tight spread meaning minimal potential upside
- 77043 has too large a standard deviation, more risk than 77092 with a lower predicted return
- **In Harris county, 77092 has the best prospects for near term growth**

4.7 El Paso County Modeling

- Located in El Paso metro
- Population: 681,000

4.7.1 79902: EDA and SARIMAX

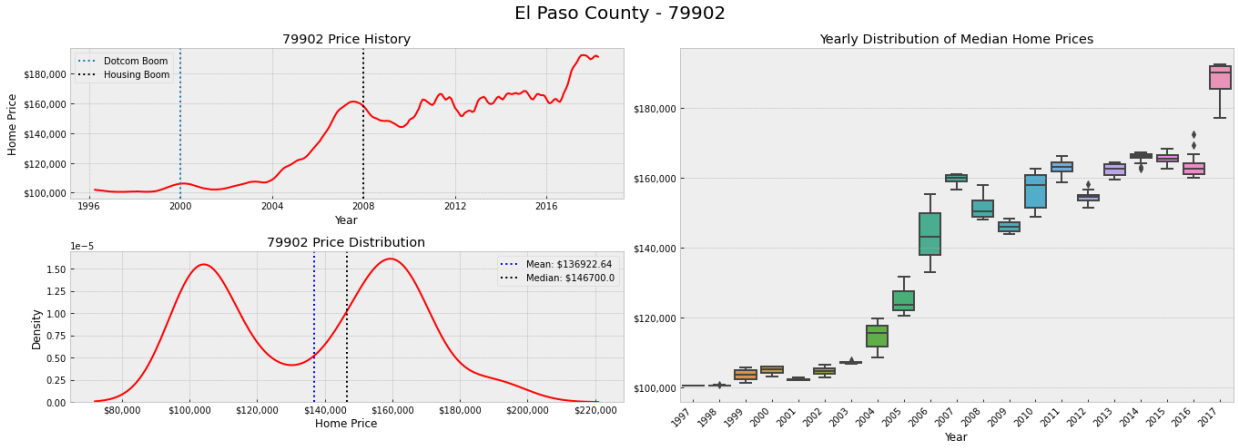
- Mission Hills South, El Paso
- 3.5 miles from downtown El Paso

In [130]:
```
# Create 79902 dataframe

df_79902=create_zip_data(El_Paso_dict_full, 79902)
```
executed in 5ms, finished 09:20:36 2021-06-20

In [131]:
```
1 zip_eda(df_79902, 79902, 'El Paso')
```

executed in 658ms, finished 09:20:37 2021-06-20

Out[131]:
```
(<Figure size 1368x504 with 3 Axes>,
                value
count      265.000000
mean    136922.641509
std      28948.753320
min     100500.000000
25%     105200.000000
50%     146700.000000
75%     161500.000000
max     192400.000000)
```

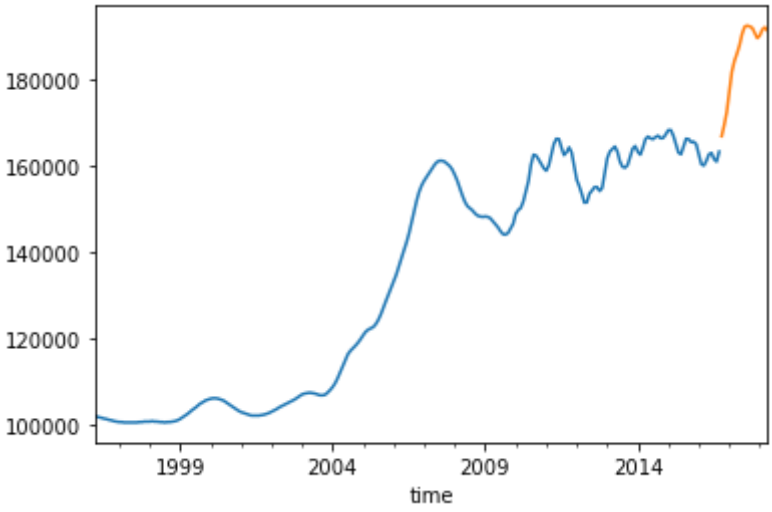


- Upward trend unitl 2008 and then a hard crash unitl recovery in 2010, trending upwards post 2016 but has slowed down
  - Median house price recovered in 2012, signficant dip between those years
- Data is bimodal
- Based on the shape, prices were strong at one point in time, retreated lower, and were continuously higher at another point
  - High prices in 2008 and a strong post crisis recovery in prices
- High spread in 2006, 2010, and 2017
  - Scatter distributions across years

In [132]:
```
train_79902, test_79902 = create_train_test_split(df_79902, 0.93)
train_79902.plot()
test_79902.plot()
```

executed in 95ms, finished 09:20:37 2021-06-20

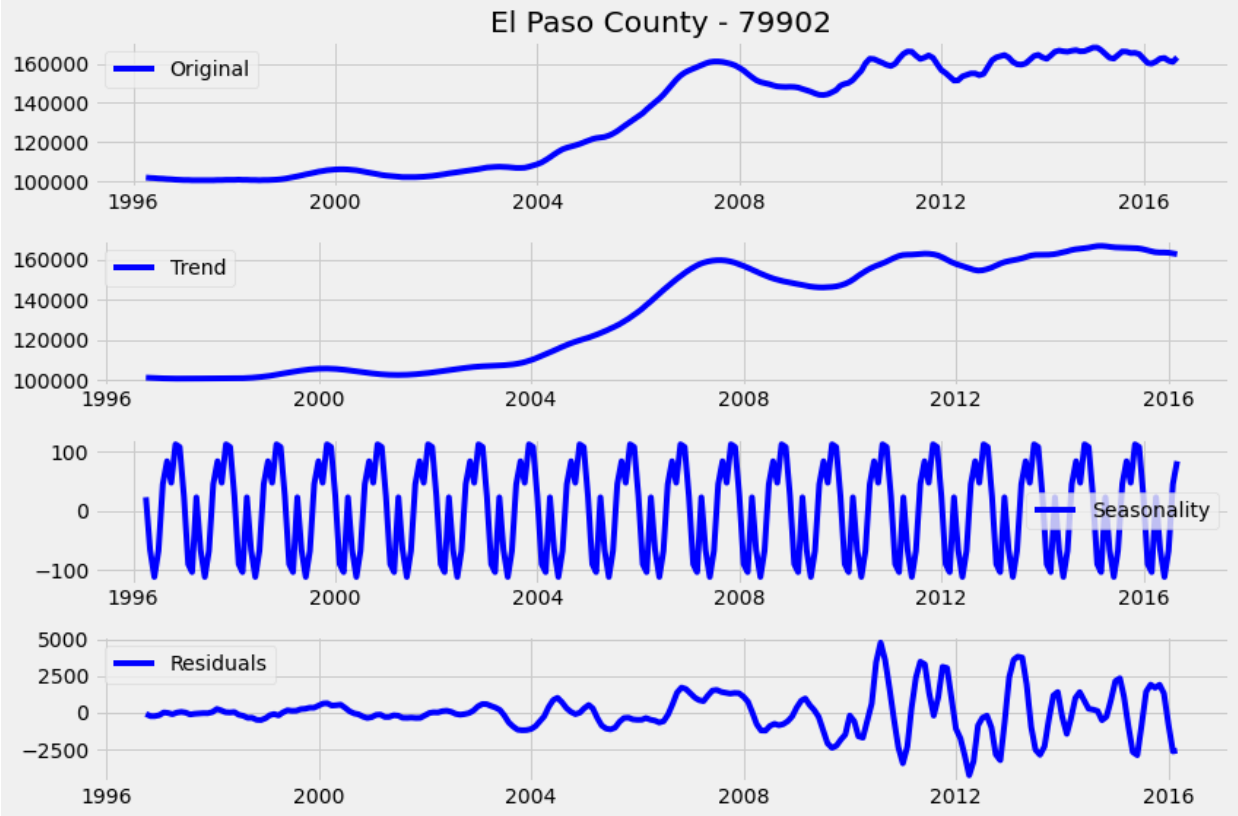
Out[132]: <AxesSubplot:xlabel='time'>



- Split data at 0.93
- Attempted to split at 0.90 but was not achieving accurate predictions

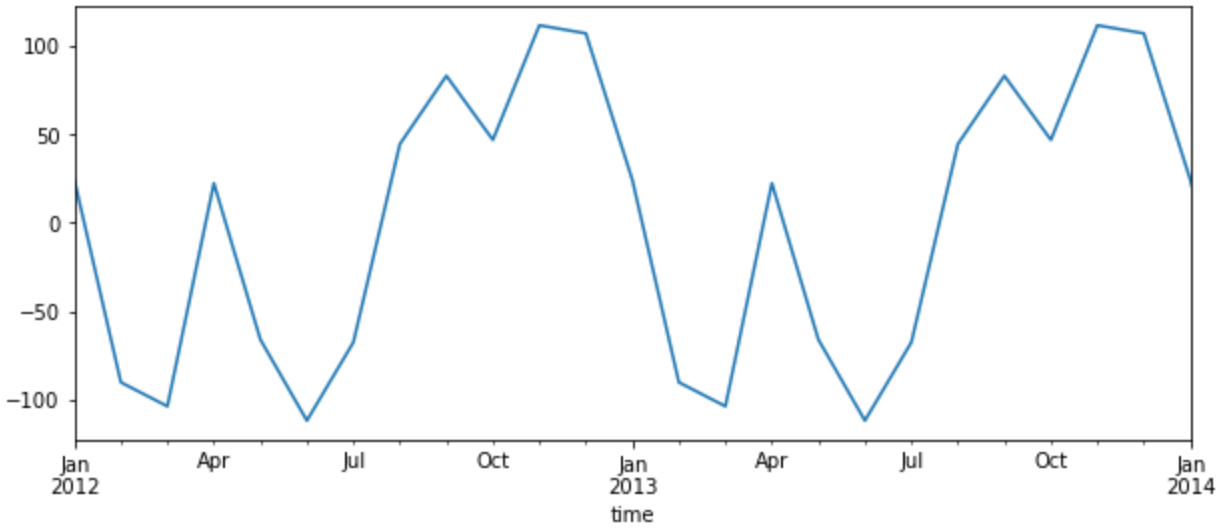
In [133]:
```
1 seasonal_decomposition(train_79902, 'El Paso', 79902)
```

executed in 342ms, finished 09:20:38 2021-06-20



In [134]:
```
decomposition = seasonal_decompose(train_79902,model='Additive')
seasonal = decomposition.seasonal
seasonal.plot(figsize=(10,4), xlim=('2012-01-01', '2014-01-01'));
```

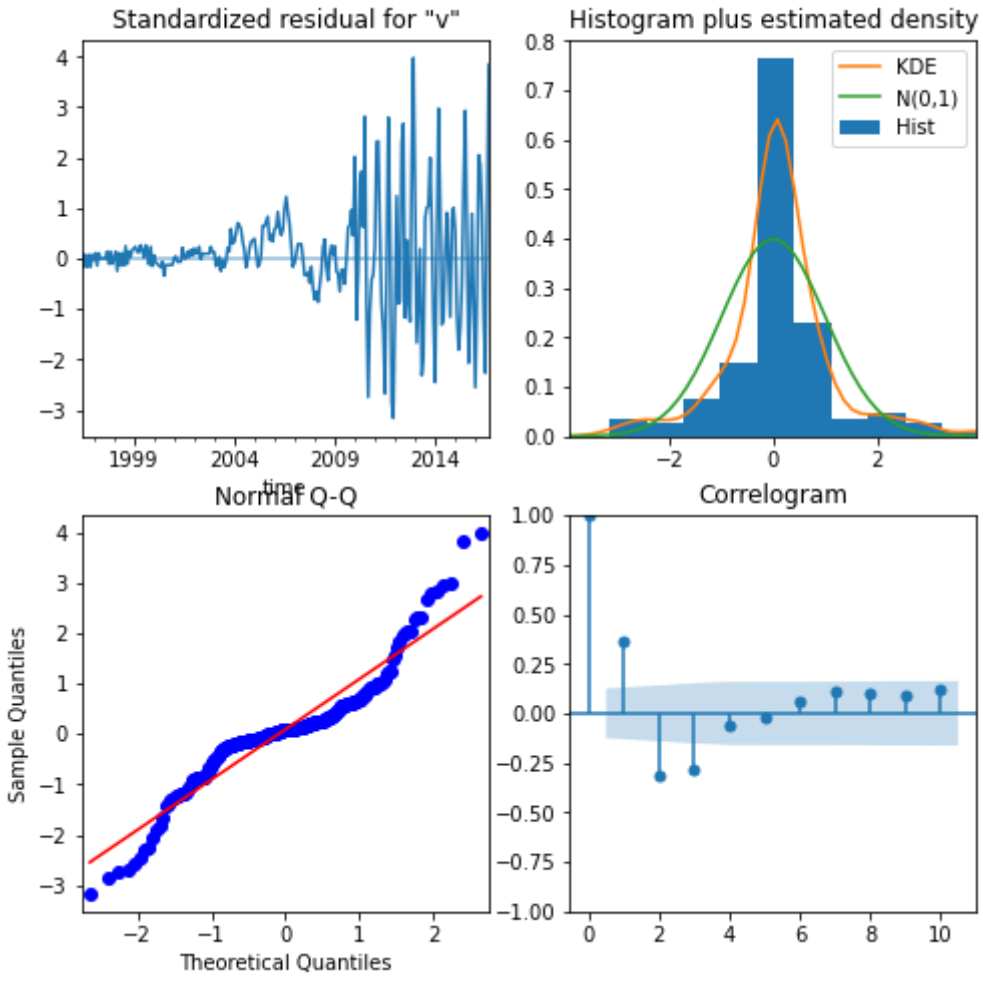
executed in 109ms, finished 09:20:38 2021-06-20

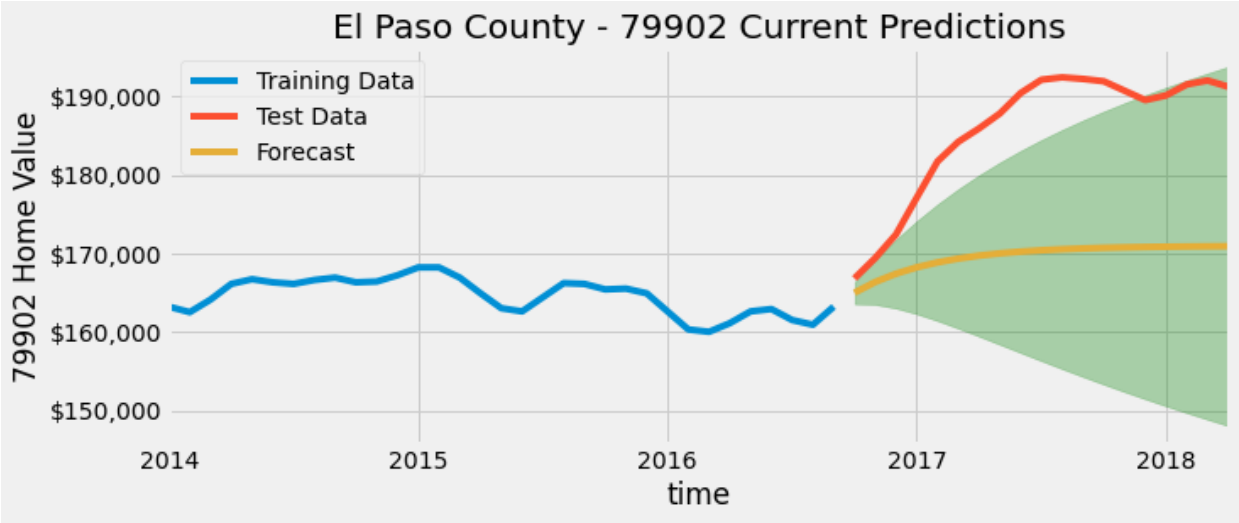


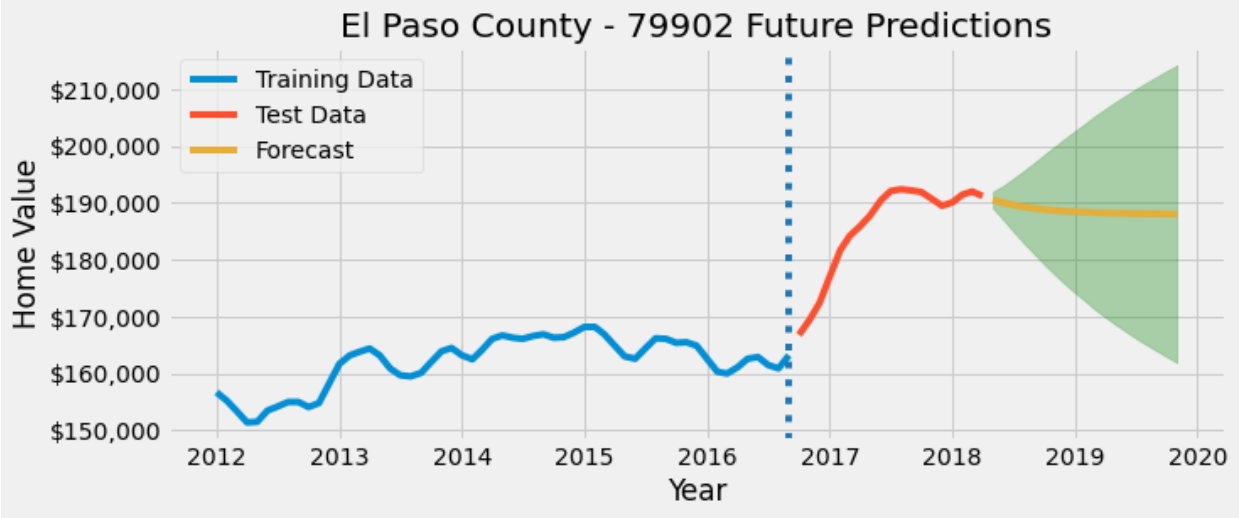
- Prices peak in 2007 and begin recovery in 2010
- Data is annually seasonal with peaks in months (July-Nov) and dips in spring (Feb-June)
- Seasonality appears constant
- Residuals have the most variance between 2010 and 2016
- Looks like data needs 1 degree of differencing on seasonality

In [135]:
```
1 fig_79902, future_79902, forecast_df_79902, roi_79902 = model_predictio
```

executed in 4.13s, finished 09:20:42 2021-06-20




El Paso County - 79902 Current Predictions
Training Data
Test Data
Forecast
79902 Home Value
$190,000
$180,000
$170,000
$160,000
$150,000
2014
2015
2016
2017
2018
time


El Paso County - 79902 Future Predictions
Training Data
Test Data
Forecast
Home Value
$210,000
$200,000
$190,000
$180,000
$170,000
$160,000
$150,000
2012
2013
2014
2015
2016
2017
2018
2019
2020
Year

In [136]:
```
1 roi_79902
```
executed in 2ms, finished 09:20:42 2021-06-20

Out[136]:
```
{'Lower CI': [-0.14380023881060847, 85619.97611893916, -14380.023881060843],
 'Upper CI': [0.11568589846377143, 111568.58984637714, 11568.589846377145],
 'Forecast': [-0.013059566605697518, 98694.04333943025, -1305.9566605697473]}
```

- Current predictions
  - Model undershoots predictions
  - Slopes upward and then becomes stagnant
- Future predictions
  - Titls slightly downward
  - Has a large confidence interval
- An investment of $100,000 today (05/01/2018) by 11/01/2019 would yield (ROI):
  - Conservative estimate: -14.4% (-$14,380)
  - Mean estimate: -1.3% (-$1,305)
  - Upper estimate: 11.6% (+$11,568)

4.7.2 79927: EDA and SARIMAX

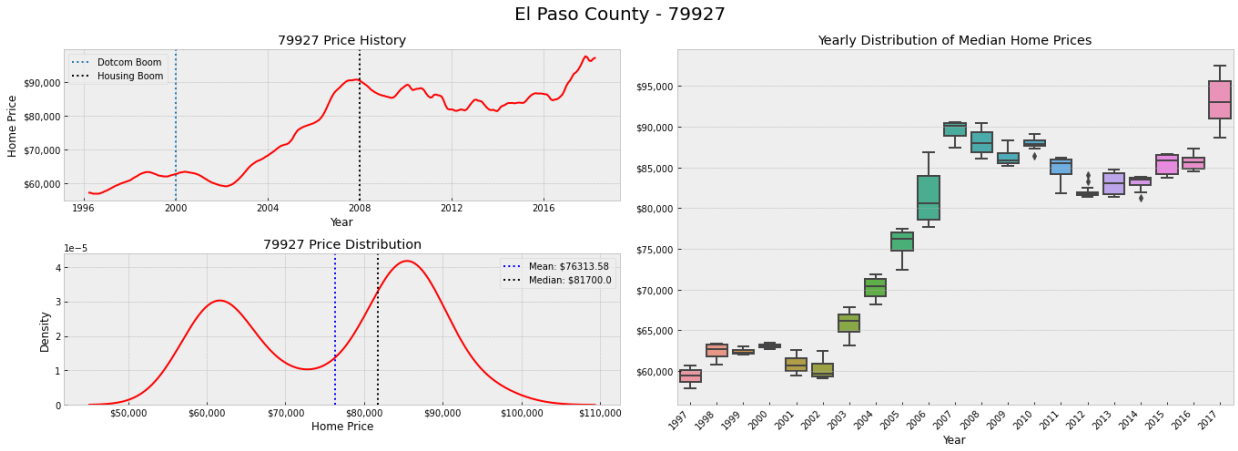
- Aldama States El Paso
- 18 miles from downtown El Paso

In [137]:
```
1 # Create 79927 dataframe
2
3 df_79927=create_zip_data(El_Paso_dict_full, 79927)
```
executed in 4ms, finished 09:20:42 2021-06-20

In [138]:
```
1 zip_eda(df_79927, 79927, 'El Paso')
```

executed in 728ms, finished 09:20:43 2021-06-20

Out[138]:
```
(<Figure size 1368x504 with 3 Axes>,
                value
count      265.000000
mean     76313.584906
std      12031.412656
min      56900.000000
25%      62900.000000
50%      81700.000000
75%      86200.000000
max      97500.000000)
```

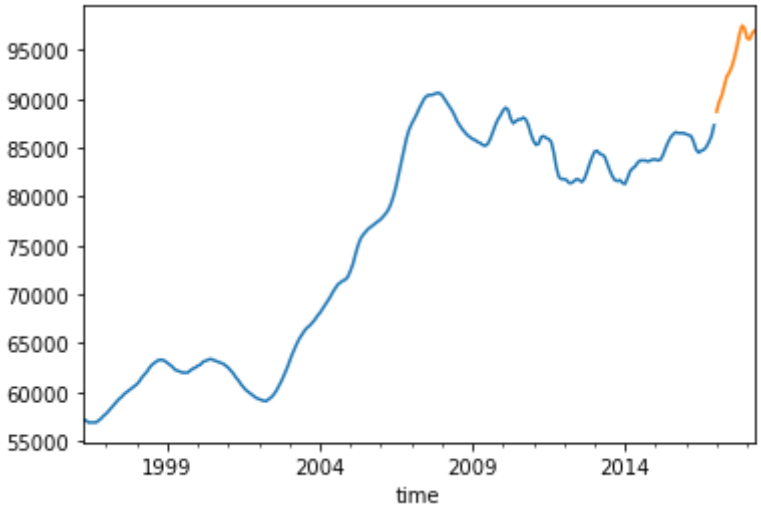


- Upward trend unitl 2008 and then a hard crash unitl recovery in 2010, and then there was a downward trend until 2016
  - Median house price recovered in 2010, signficant dip between those years, and then further dip
- Data is bimodal
- Based on the shape, prices were strong at one point in time, retreated lower, and were continuously higher at another point
  - High prices in 2008 and a strong post crisis recovery in prices in 2016
- High spread in 2006 and 2017
  - Scattered distributions across years

In [139]:
```
train_79927, test_79927 = create_train_test_split(df_79927, 0.94)
train_79927.plot()
test_79927.plot()
```

executed in 100ms, finished 09:20:43 2021-06-20

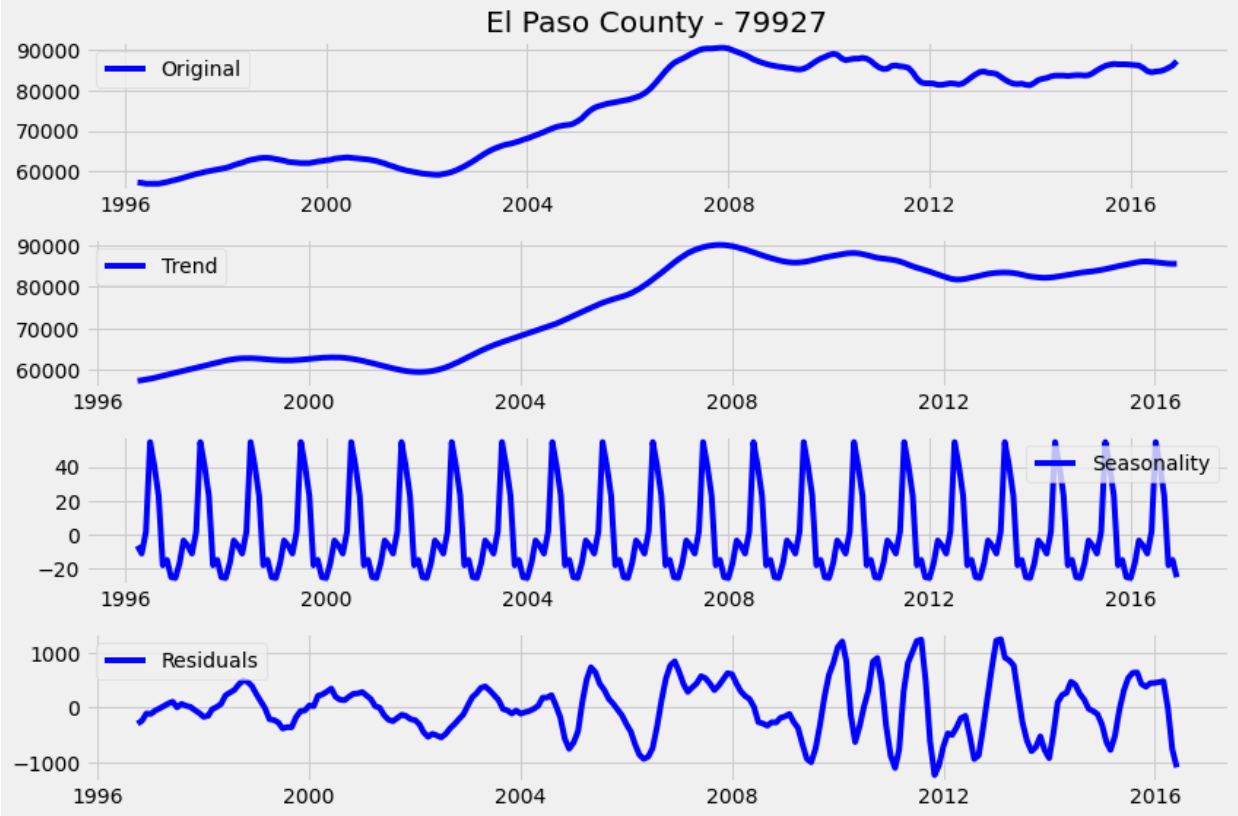
Out[139]: <AxesSubplot:xlabel='time'>



- Baseline score of 0.90
- Had to adjust to 0.94

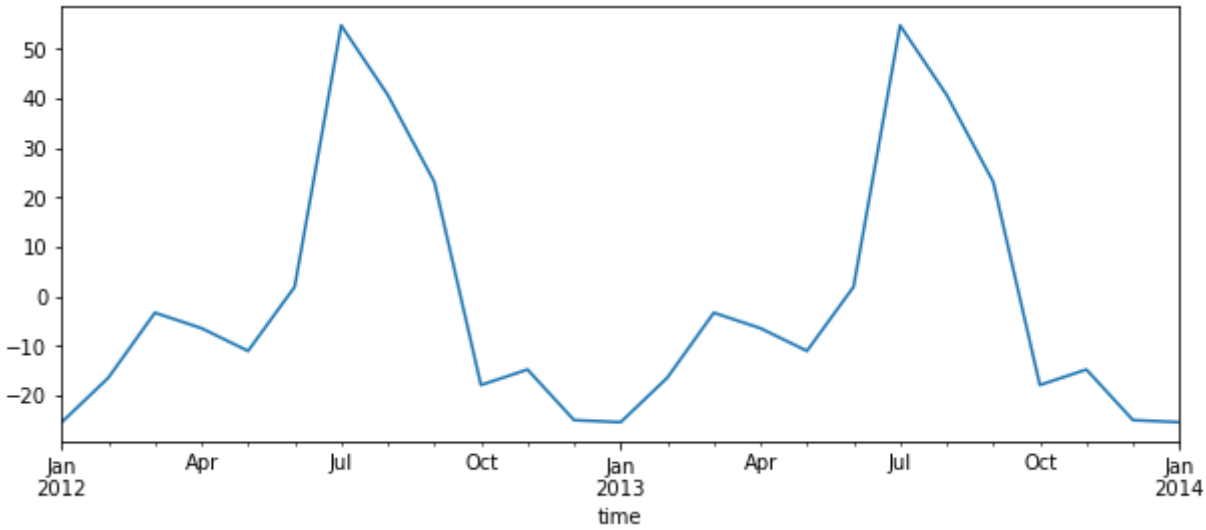
In [140]:
```
1 seasonal_decomposition(train_79927, 'El Paso', 79927)
```

executed in 325ms, finished 09:20:43 2021-06-20



In [141]:
```
decomposition = seasonal_decompose(train_79927,model='Additive')
seasonal = decomposition.seasonal
seasonal.plot(figsize=(10,4), xlim=('2012-01-01', '2014-01-01'));
```

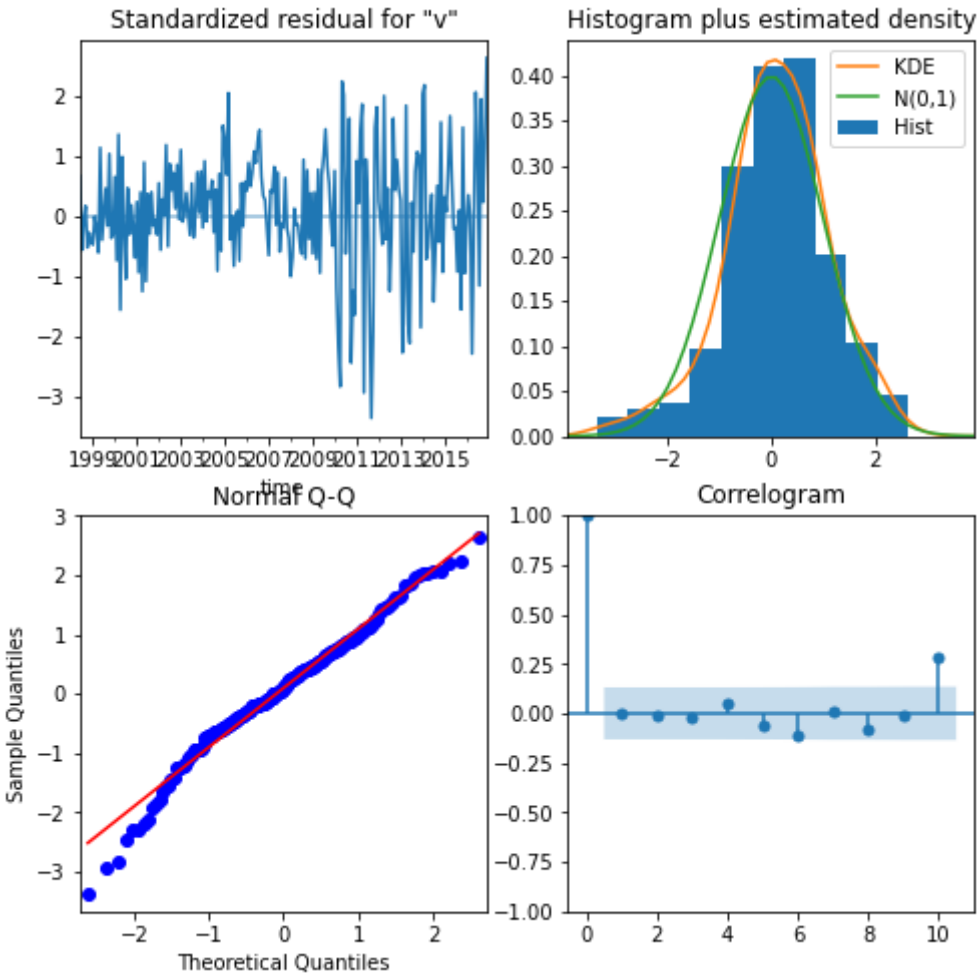
executed in 137ms, finished 09:20:43 2021-06-20

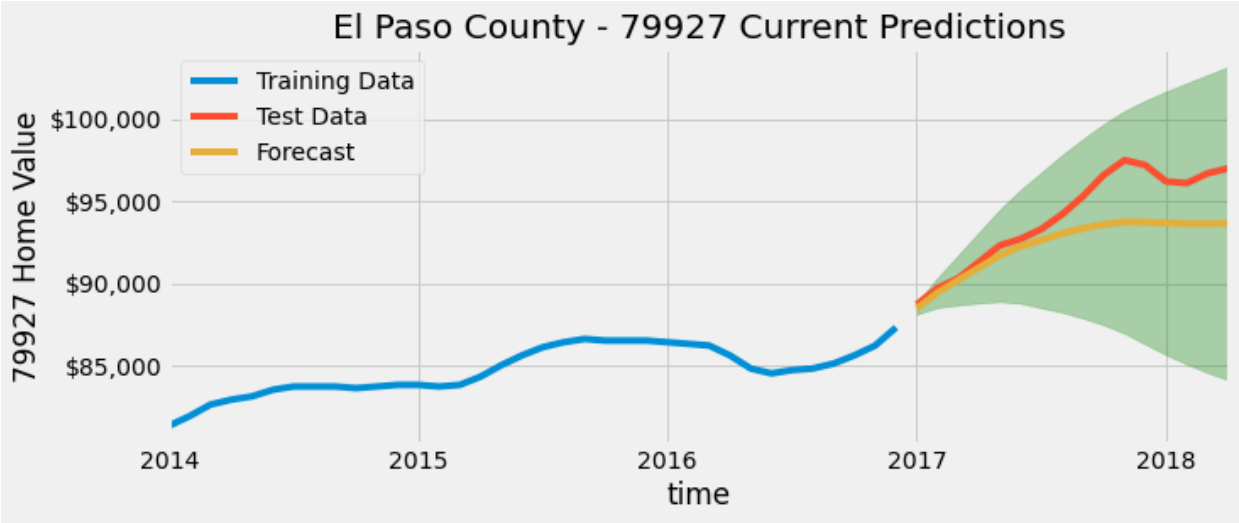


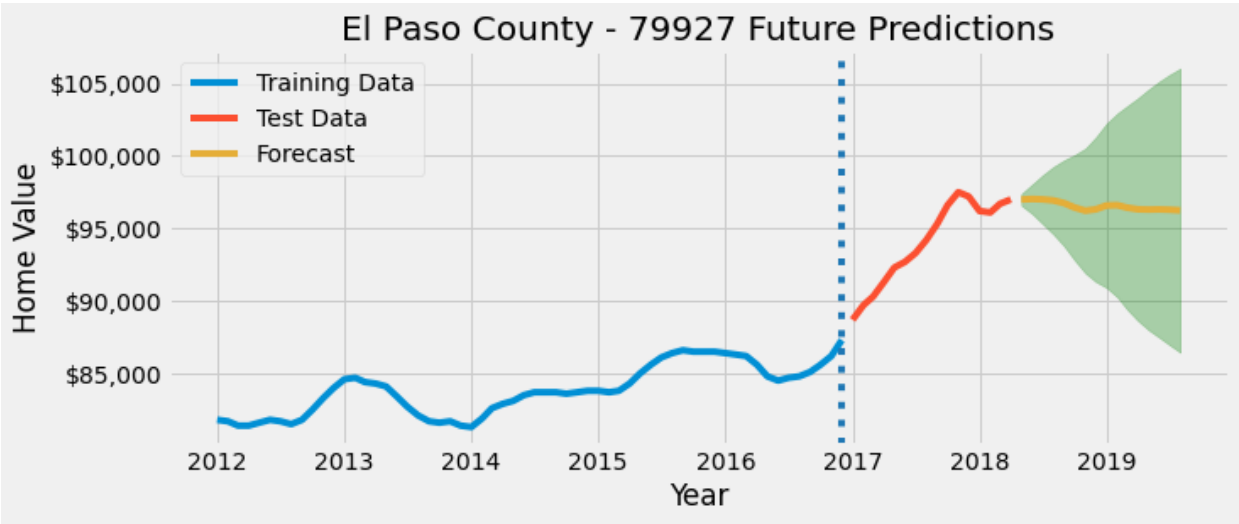
- There is an upward trend from 2002 to 2009 and then it dips and begins to recover in 2015
- Data is annually seasonal with peaks in months (May-July) and dips in spring (Oct-March)
- Seasonality appears constant
- Residuals have the most variance between 2012 and 2016
- Looks like data needs 1 degree of differencing on seasonality

In [142]:
```
1 fig_79927, future_79927, forecast_df_79927, roi_79927 = model_predictio
```

executed in 31.7s, finished 09:21:15 2021-06-20







In [143]:
```
1 roi_79927
```

executed in 3ms, finished 09:21:15 2021-06-20

Out[143]:
```
{'Lower CI': [-0.10494525347771554, 89505.47465222844, -10494.525347771556],
 'Upper CI': [0.08875662528383005, 108875.662528383, 8875.662528383007],
 'Forecast': [-0.007689366924543338, 99231.06330754566, -768.9366924543428]}
```

- Current predictions
  - Model undershoots predictions
  - Slopes upward and then becomes stagnant
- Future predictions
  - Titls slightly downward
  - Has a large confidence interval
- An investment of $100,000 today (05/01/2018) by 08/01/2019 would yield (ROI):
  - Conservative estimate: -10.5% (-$10,494)
  - Mean estimate: -0.1% (-$768)
  - Upper estimate: 8.9% (+$8,875)

4.7.3 79903: EDA and SARIMAX

- Timberwolf El Paso
- 4.7 miles from downtown El Paso

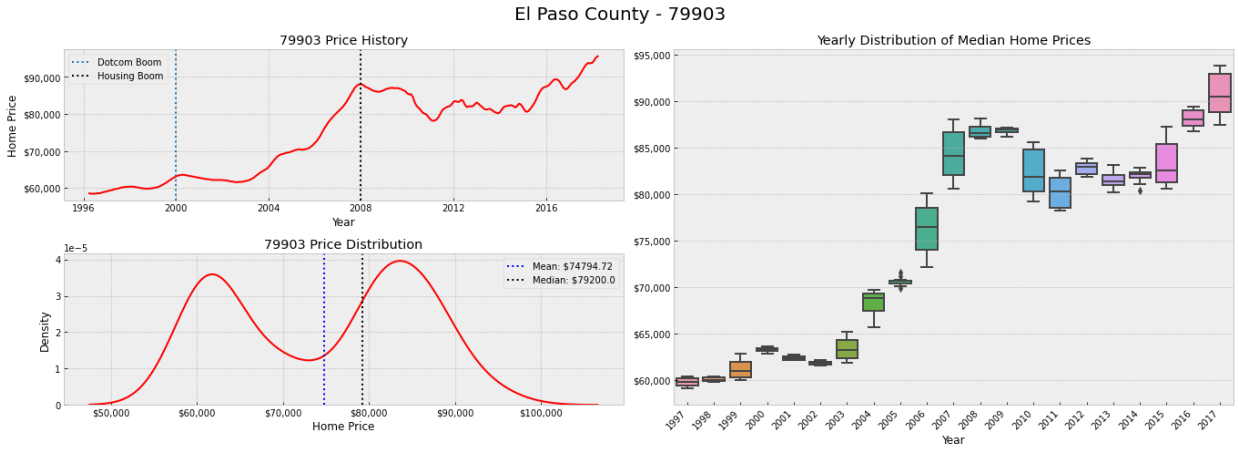
In [144]:
```
# Create 79903 dataframe

df_79903=create_zip_data(El_Paso_dict_full, 79903)
```

executed in 4ms, finished 09:21:15 2021-06-20

In [145]:
```
zip_eda(df_79903, 79903, 'El Paso')
```

executed in 629ms, finished 09:21:15 2021-06-20

Out[145]:
```
(<Figure size 1368x504 with 3 Axes>,
                value
count      265.000000
mean     74794.716981
std      11213.721405
min      58500.000000
25%      62400.000000
50%      79200.000000
75%      83900.000000
max      95600.000000)
```

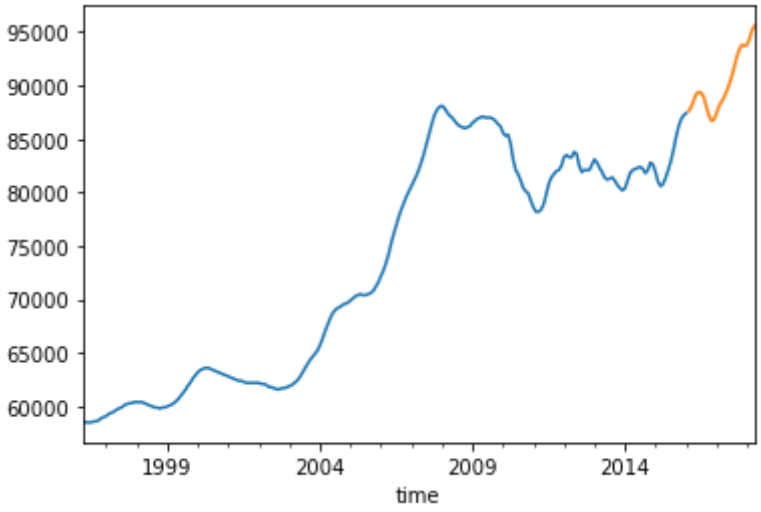


- Upward trend unitl 2008 and then a hard crash unitl recovery in 2016, and then there was a downward trend until 2016
  - Median house price recovered in 2010, signficant dip between those years
- Data is bimodal
- Based on the shape, prices were strong at one point in time, retreated lower, and were continuously higher at another point
  - High prices in 2008 and a strong post crisis recovery in prices in 2016
- High spread in 2006 and 2017
  - Scattered distributions across years
  - Lower distributions in 2011 through 2014

In [146]:
```
train_79903, test_79903 = create_train_test_split(df_79903, 0.90)
train_79903.plot()
test_79903.plot()
```

executed in 110ms, finished 09:21:16 2021-06-20

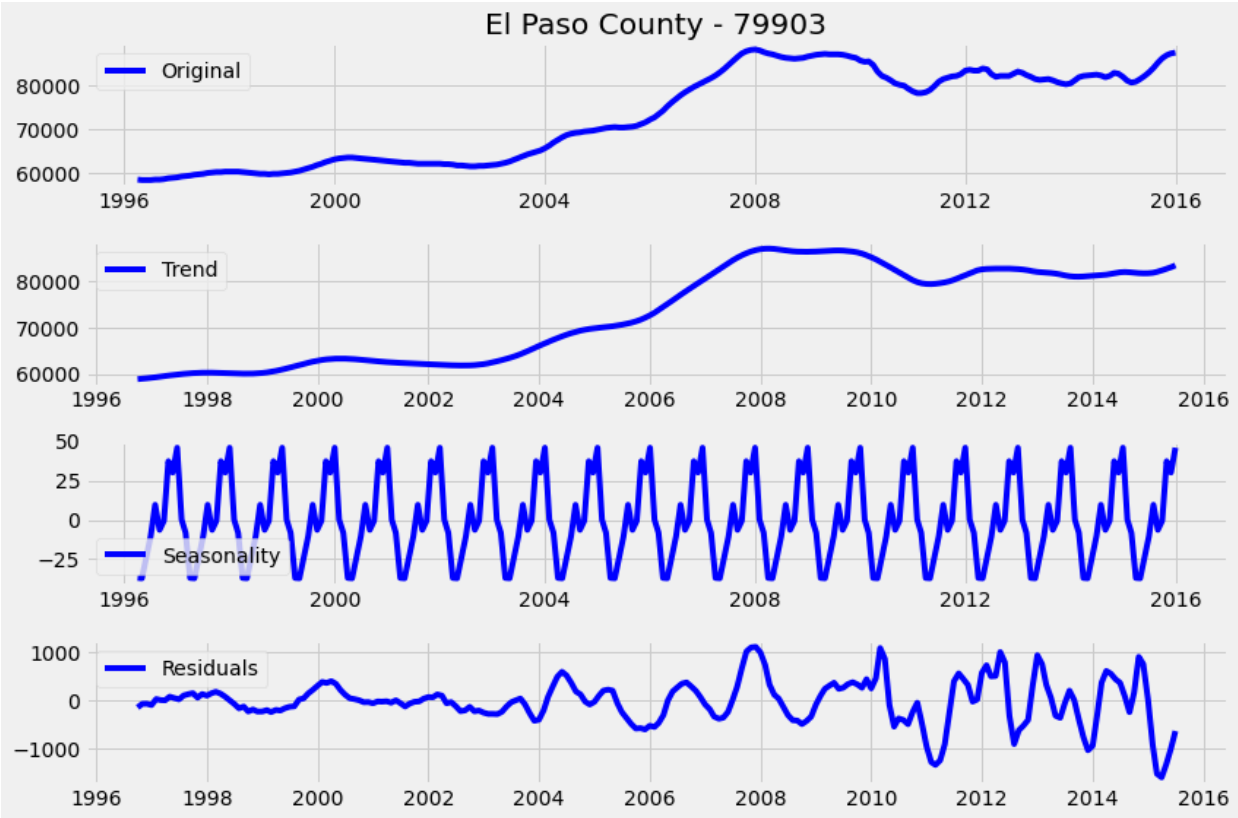
Out[146]: <AxesSubplot:xlabel='time'>



- Baseline split at 0.90

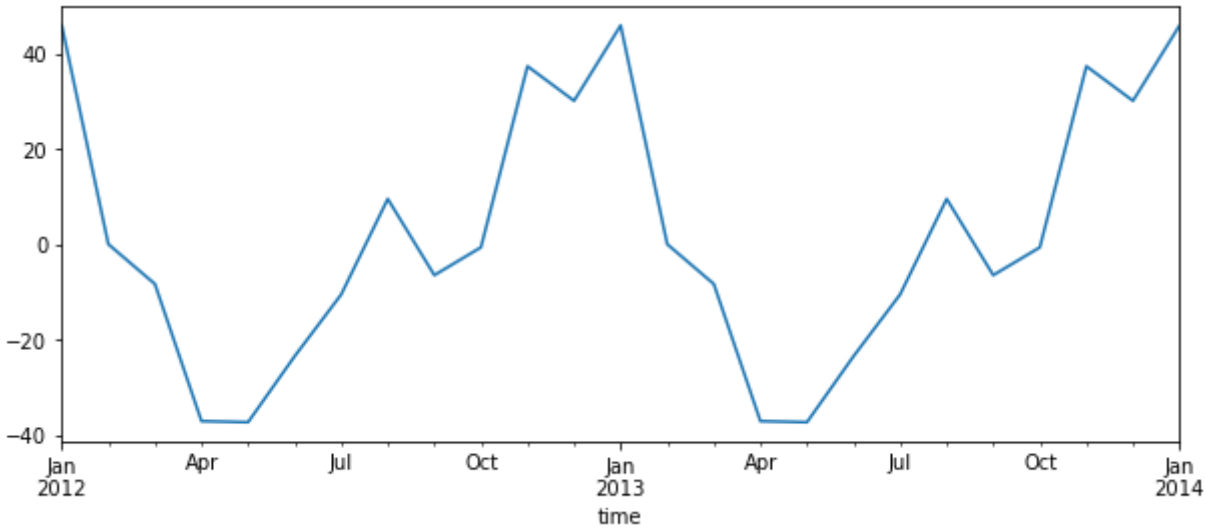
In [147]:
```
seasonal_decomposition(train_79903, 'El Paso', 79903)
```

executed in 364ms, finished 09:21:16 2021-06-20



In [148]:
```
decomposition = seasonal_decompose(train_79903,model='Additive')
seasonal = decomposition.seasonal
seasonal.plot(figsize=(10,4), xlim=('2012-01-01', '2014-01-01'));
```

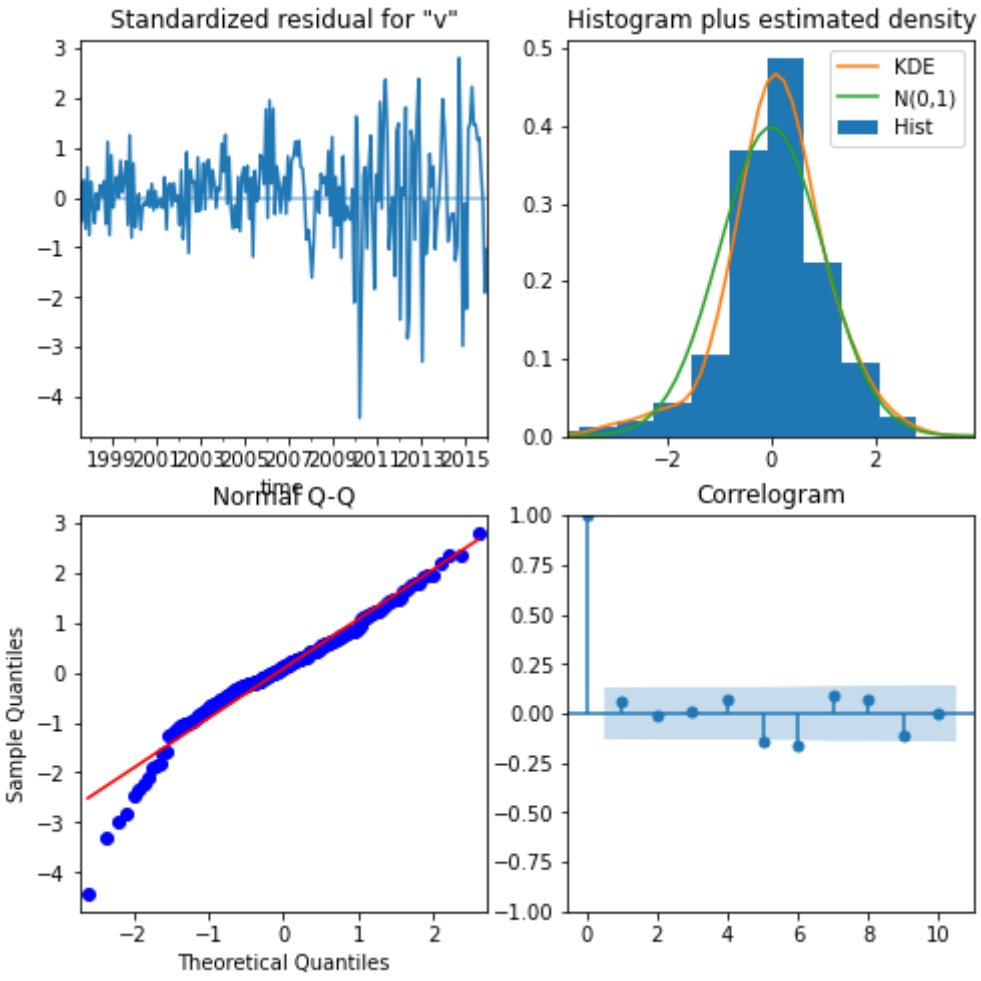
executed in 102ms, finished 09:21:16 2021-06-20

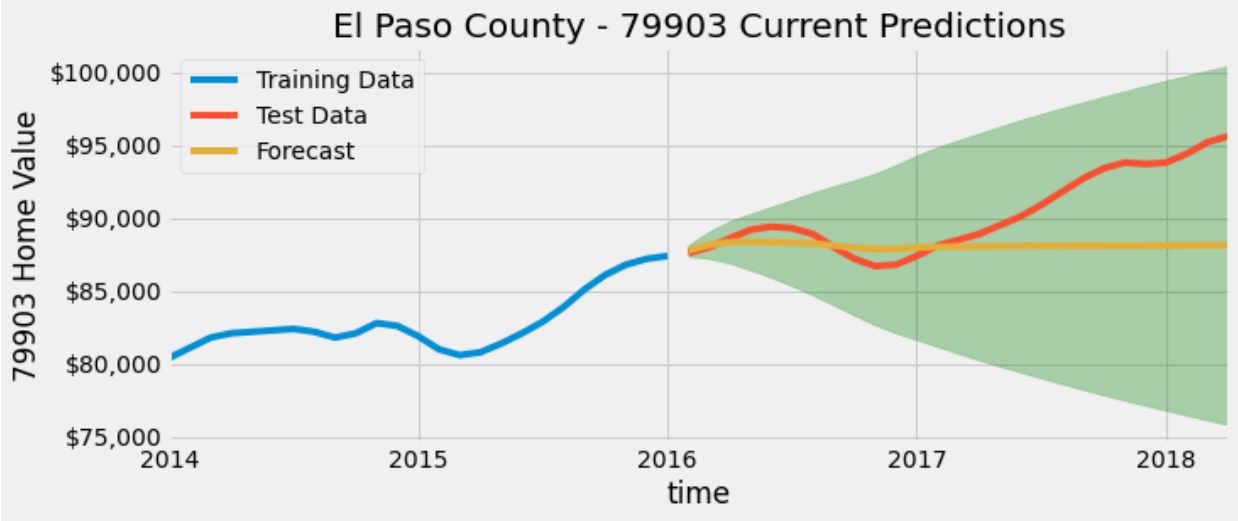


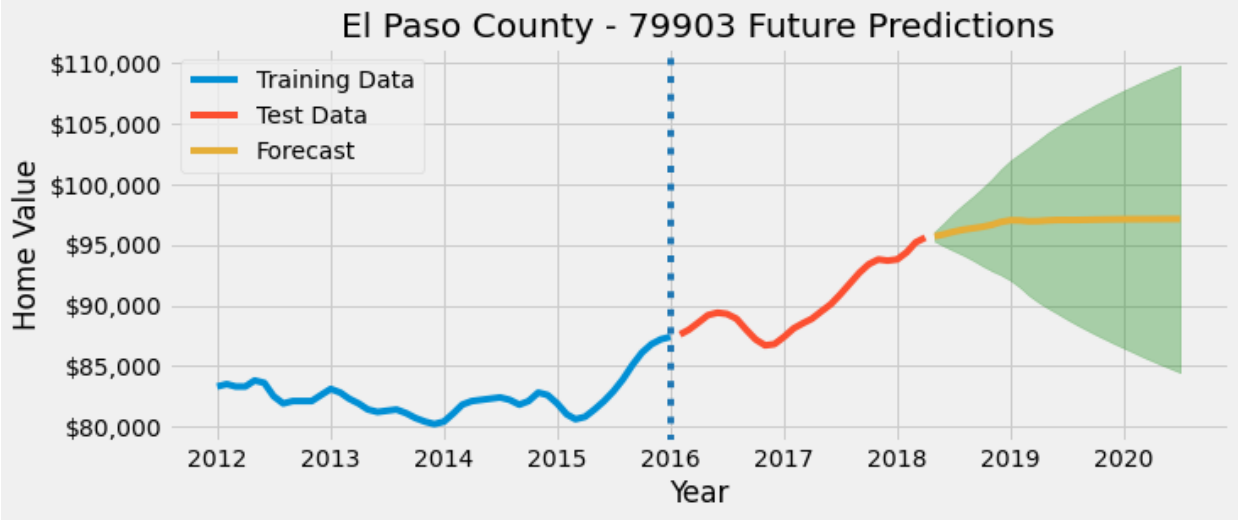
- There is an upward trend from 2002 to 2009 and then it dips and begins to recover in 2015
- Data is annually seasonal with peaks in months (Oct-Jan) and dips in spring (Feb-May)
- Seasonality appears constant
- Residuals have the most variance between 2012 and 2016
- Looks like data needs 1 degree of differencing on seasonality

In [149]:
```
1 fig_79903, future_79903, forecast_df_79903, roi_79903 = model_predictio
```

executed in 51.0s, finished 09:22:07 2021-06-20







In [150]:
```
1 roi_79903
```

executed in 2ms, finished 09:22:07 2021-06-20

Out[150]:
```
{'Lower CI': [-0.11384695147451204, 88615.3048525488, -11384.695147451203],
 'Upper CI': [0.14274538755690414, 114274.5387556904, 14274.538755690402],
 'Forecast': [0.01499434571049657, 101499.43457104966, 1499.4345710496564]}
```

- Current predictions
  - Model undershoots predictions
  - Stays stagnant and has a large confidence interval

- Future predictions
  - Titls slightly upward and then levels off
  - Doesn't follow trend of test data
  - Has a large confidence interval
- An investment of $100,000 today (05/01/2018) by 07/01/2020 would yield (ROI):
  - Conservative estimate: -11.4% (-$11,384)
  - Mean estimate: 1.5% ($1,499)
  - Upper estimate: 14.3% (+$14,274)

4.8 El Paso Conclusion

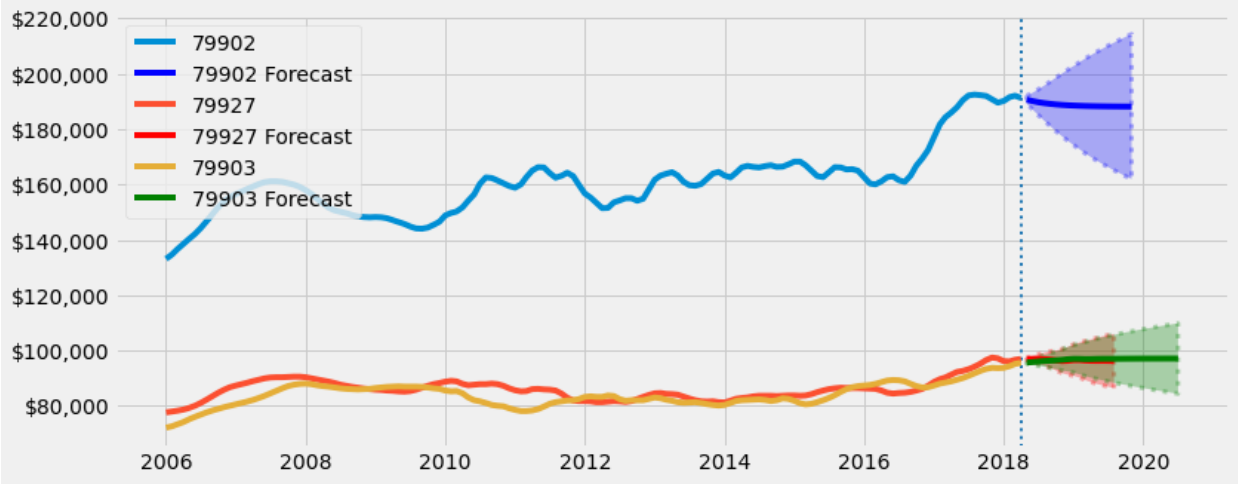
In [151]:
```
1 corr_check(df_79902, df_79927, df_79903, 79902, 79927, 79903)
```
executed in 6ms, finished 09:22:07 2021-06-20

Out[151]:

	79902	79927	79903
79902	1.000000	0.957589	0.954831
79927	0.957589	1.000000	0.975394
79903	0.954831	0.975394	1.000000

- All of the correlations fall into a fairly tight range (~0.02)
- 79902 moves more closely with 79927, very thin margin
- 79927 moves more closely with 77092, very thin margin
- 79903 moves more closely with 79927

In [152]:
```
1 county_forecast_comparison(df_79902, 79902, forecast_df_79902, df_79927
```
executed in 147ms, finished 09:22:07 2021-06-20



- 79902 has the highest ending point but also has a very large confidence interval
  - Slopes slightly downward
- 79927 & 79903 move very tightly together, especially for predictions
  - 79903 extends further because it has more training data
  - Both have very similar predictions

In [153]:
```
el_paso_perc_comparison = county_forecast_perc_comparison(roi_79902, 79
el_paso_perc_comparison
```
executed in 7ms, finished 09:22:07 2021-06-20

Out[153]:

	Lower CI	Upper CI	Forecast
79902 Perc Change	-0.143800	0.115686	-0.013060
Val	85619.976119	111568.589846	98694.043339
$ Diff	-14380.023881	11568.589846	-1305.956661
79927 Perc Change	-0.104945	0.088757	-0.007689
Val	89505.474652	108875.662528	99231.063308
$ Diff	-10494.525348	8875.662528	-768.936692
79903 Perc Change	-0.113847	0.142745	0.014994
Val	88615.304853	114274.538756	101499.434571
$ Diff	-11384.695147	14274.538756	1499.434571

- Based on the downside risk, upside return, and mean predicted value, 79903 seems like the superior zip code
- It has the second highest upside, highest predicted mean, and second lowest downside risk
- 79927 has a very similar risk return profile
- 79902 has more downside risk
- **In El Paso county, 79903 has the best prospects for near term growth**

4.9 Collin County Modeling

- Located in Dallas-Fort Worth metro
- Population: 782,341

4.9.1 75069: EDA and SARIMAX

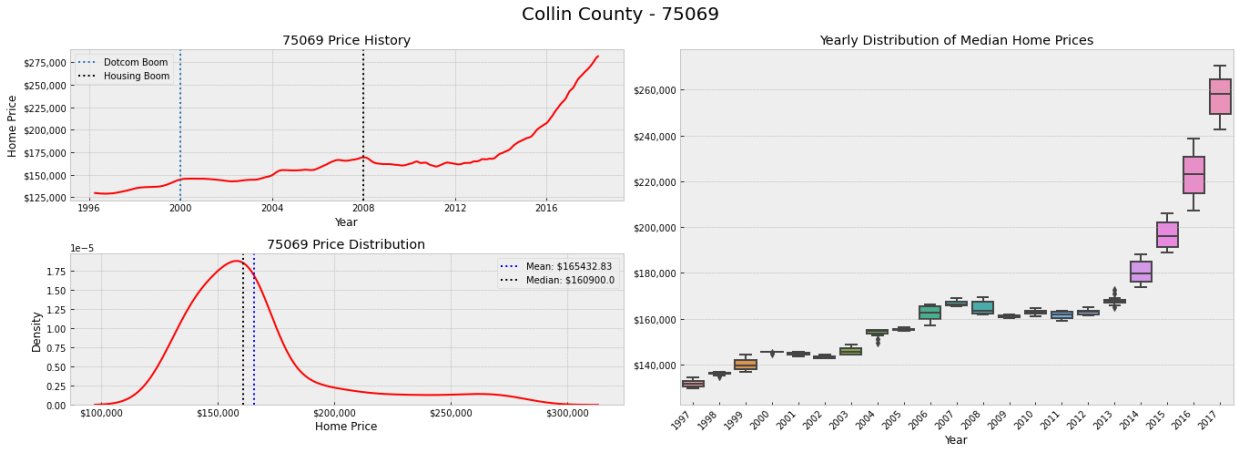
- Fairview, Texas
- 33 miles from downtown Dallas

In [154]:
```
# Create 75069 dataframe

df_75069 = create_zip_data(Collin_dict_full, 75069)
```
executed in 4ms, finished 09:22:07 2021-06-20

In [155]:
```
1 zip_eda(df_75069, 75069, 'Collin')
```

executed in 696ms, finished 09:22:08 2021-06-20

Out[155]:
```
(<Figure size 1368x504 with 3 Axes>,
                value
count      265.000000
mean    165432.830189
std      32335.945168
min     129000.000000
25%     144700.000000
50%     160900.000000
75%     167200.000000
max     281400.000000)
```

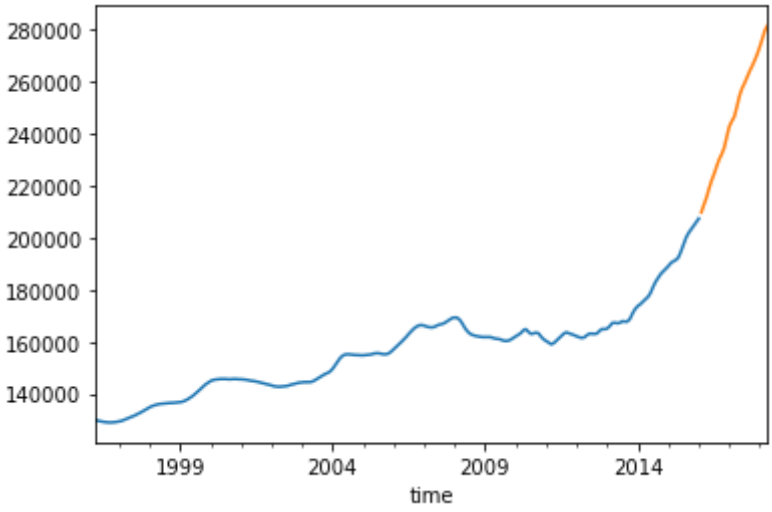


- Upward trend unitl 2008 and then stagnant decline unitl 2013
  - Median house price recovered in 2013, minimal dip between those years
- Data is right skewed as mean is greater than the median
- Based on the shape, prices have continuously trended upwards because of the right skew
- High spread in 2016 and 2017
  - Prior, narrow distribution

In [156]:
```
train_75069, test_75069 = create_train_test_split(df_75069, 0.90)
train_75069.plot()
test_75069.plot()
```

executed in 111ms, finished 09:22:08 2021-06-20

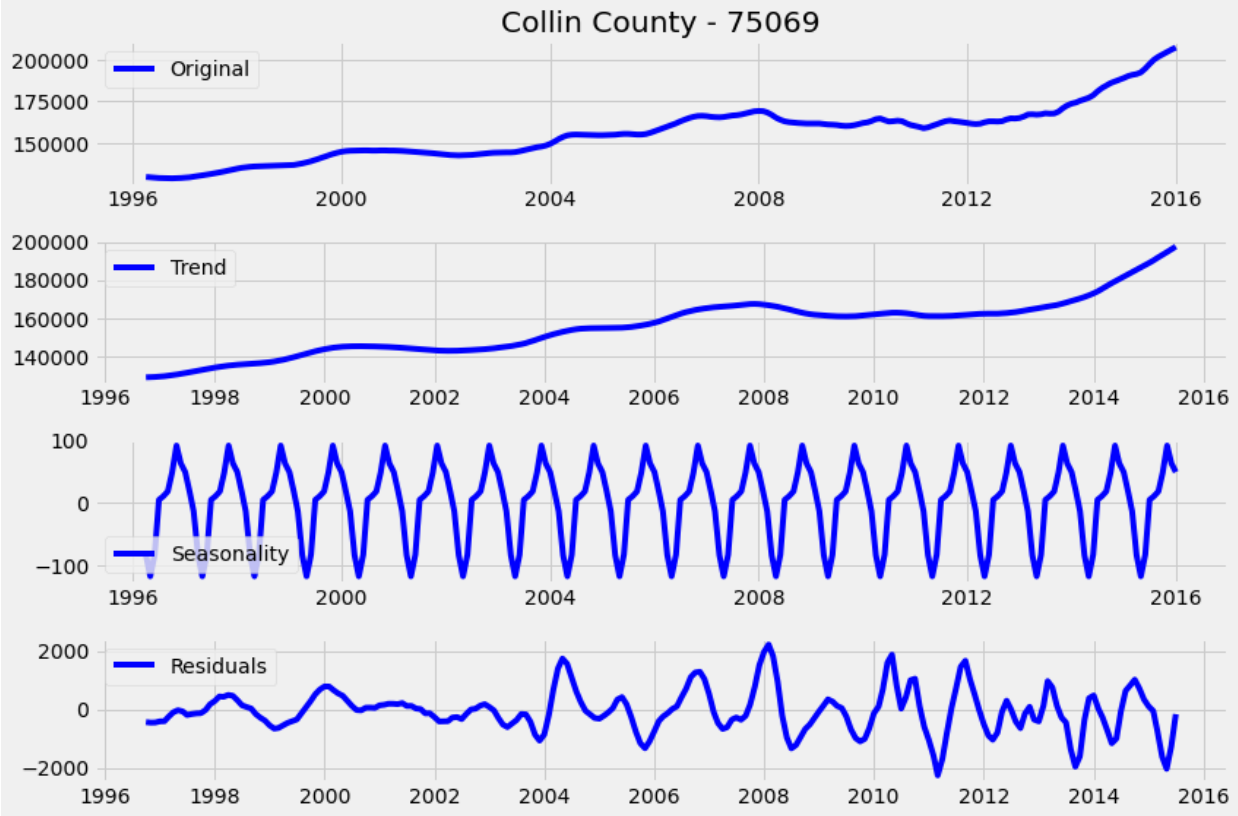
Out[156]: <AxesSubplot:xlabel='time'>



- Baseline split of 0.90

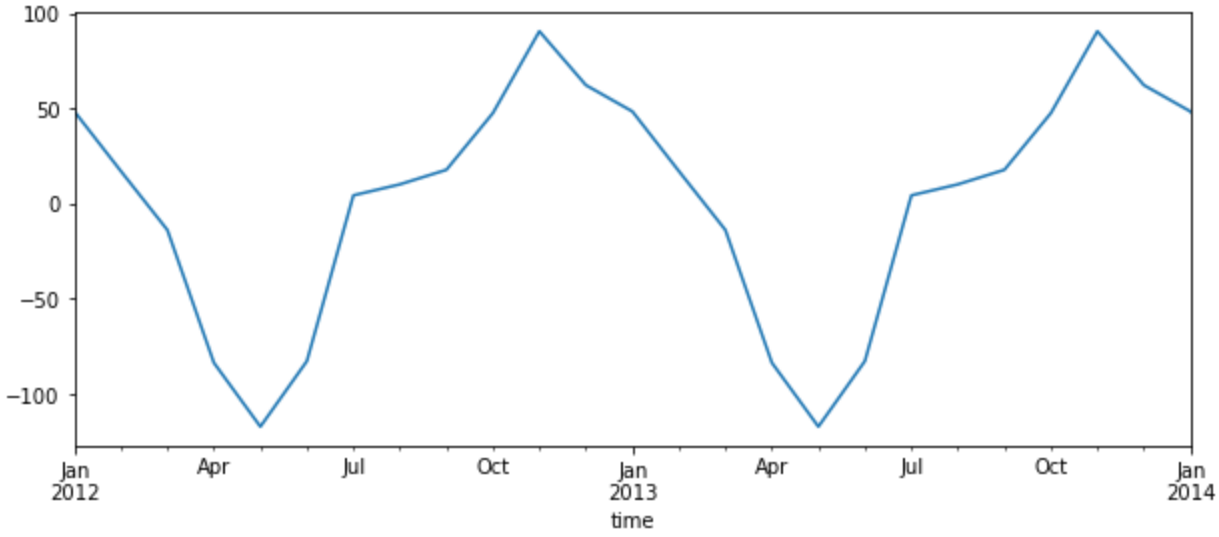
In [157]:
```
1 seasonal_decomposition(train_75069, 'Collin', 75069)
```

executed in 358ms, finished 09:22:08 2021-06-20



In [158]:
```
decomposition = seasonal_decompose(train_75069,model='Additive')
seasonal = decomposition.seasonal
seasonal.plot(figsize=(10,4), xlim=('2012-01-01', '2014-01-01'));
```

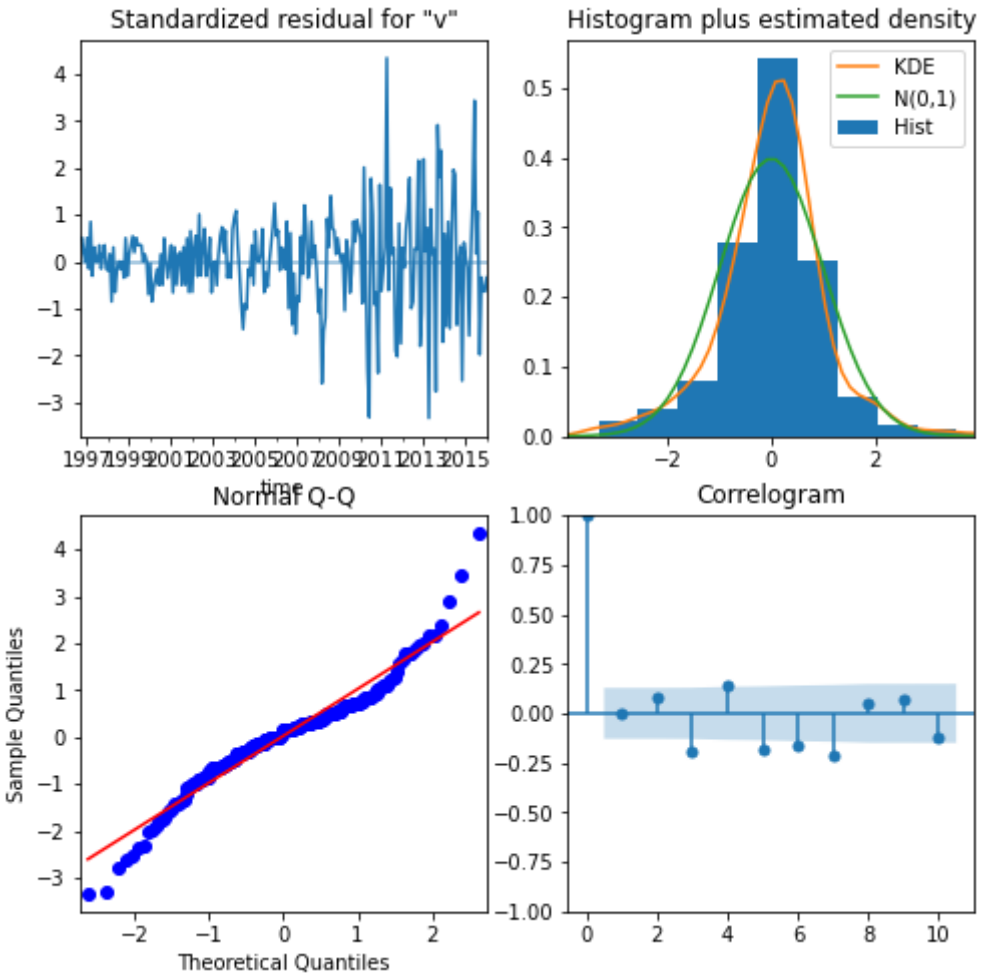
executed in 108ms, finished 09:22:08 2021-06-20

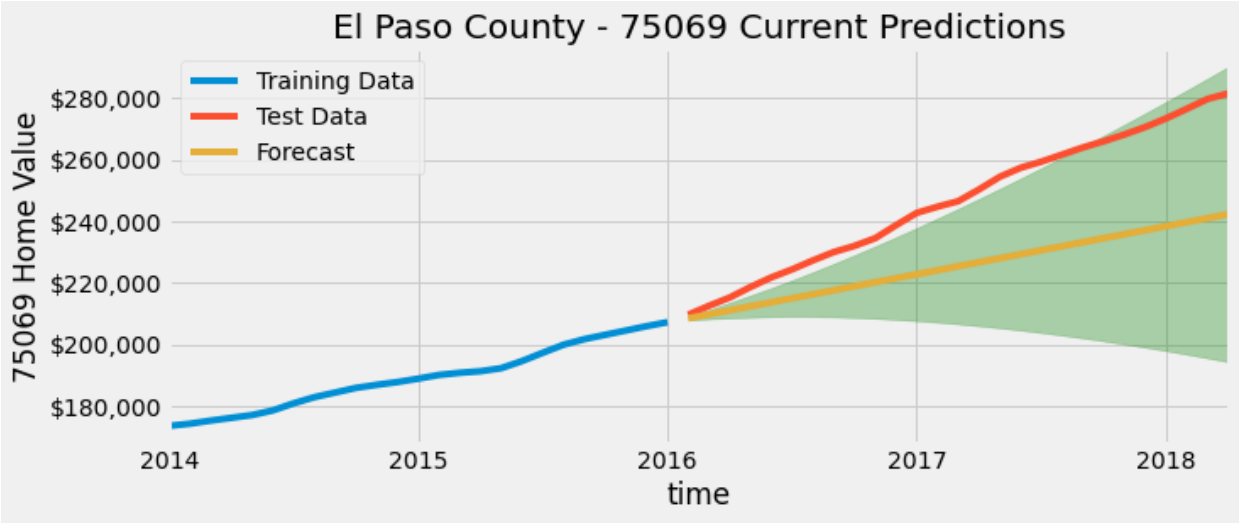


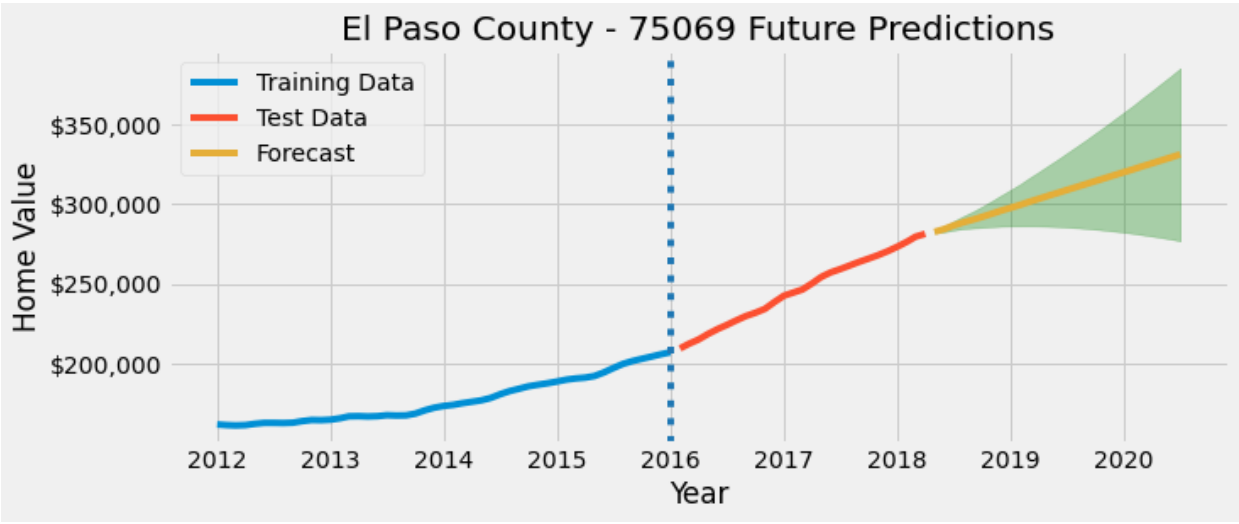
- There is an upward trend from 2002 to 2008 and then it dips slightly and begins to recover in 2012
- Data is annually seasonal with peaks in months (Oct-Jan) and dips in spring (Feb-May)
- Seasonality appears constant
- Residuals are relatively stagnant but increase in 2011 and then level out
- Looks like data needs 1 degree of differencing on seasonality

In [159]:
```
1 fig_75069, future_75069, forecast_df_75069, roi_75069 = model_predictio
```

executed in 11.7s, finished 09:22:20 2021-06-20







In [160]:
```
1 roi_75069
```

executed in 3ms, finished 09:22:20 2021-06-20

Out[160]:
```
{'Lower CI': [-0.017313637298197678, 98268.63627018023, -1731.3637298197718],
 'Upper CI': [0.35963065896632984, 135963.065896633, 35963.065896632994],
 'Forecast': [0.17160388414793543, 117160.38841479355, 17160.388414793546]}
```

- Current predictions
  - Model undershoots predictions

    - Confidence interval captures some of the test data towards mid - 2017
- Future predictions
    - Titls upwards
    - Follows trend of test data
    - Has a narrow confidence interval
- An investment of $100,000 today (05/01/2018) by 07/01/2020 would yield (ROI):
    - Conservative estimate: -1.7% (-$1,731)
    - Mean estimate: 17.2% ($17,160)
    - Upper estimate: 36.0% (+$35,963)

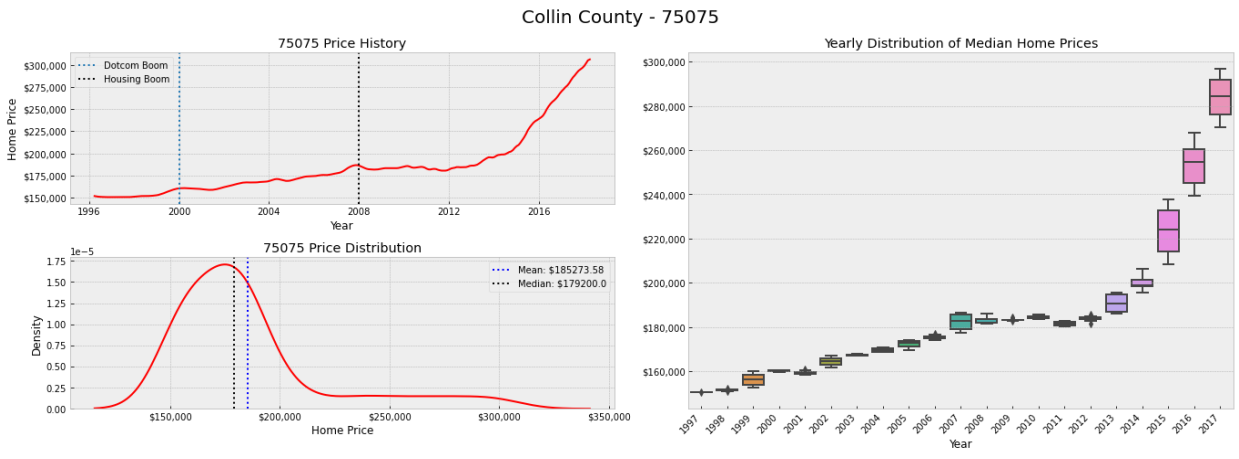
4.9.2 75075: EDA and SARIMAX

- Plano, Texas
- 20 miles from downtown Dallas

In [161]:
```
# Create 75075 dataframe

df_75075 = create_zip_data(Collin_dict_full, 75075)
```
executed in 4ms, finished 09:22:20 2021-06-20

In [162]:
```
zip_eda(df_75075, 75075, 'Collin')
```
executed in 631ms, finished 09:22:21 2021-06-20

Out[162]:
```
(<Figure size 1368x504 with 3 Axes>,
                value
count      265.000000
mean    185273.584906
std      35514.162513
min     150400.000000
25%     160600.000000
50%     179200.000000
75%     186100.000000
max     306400.000000)
```



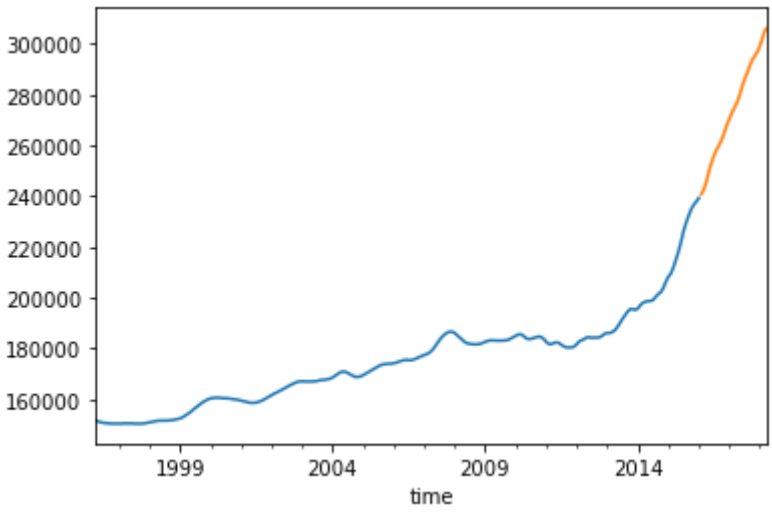
- Upward trend unitl 2008 and then stagnant decline unitl 2013
    - Median house price recovered in 2013, minimal dip between those years
- Data is right skewed as mean is greater than the median

- Based on the shape, prices have continuously trended upwards because of the right skew
- High spread in 2015 through 2017
  - Prior, narrow distribution

In [163]:
```
1 train_75075, test_75075 = create_train_test_split(df_75075, 0.90)
2 train_75075.plot()
3 test_75075.plot()
```

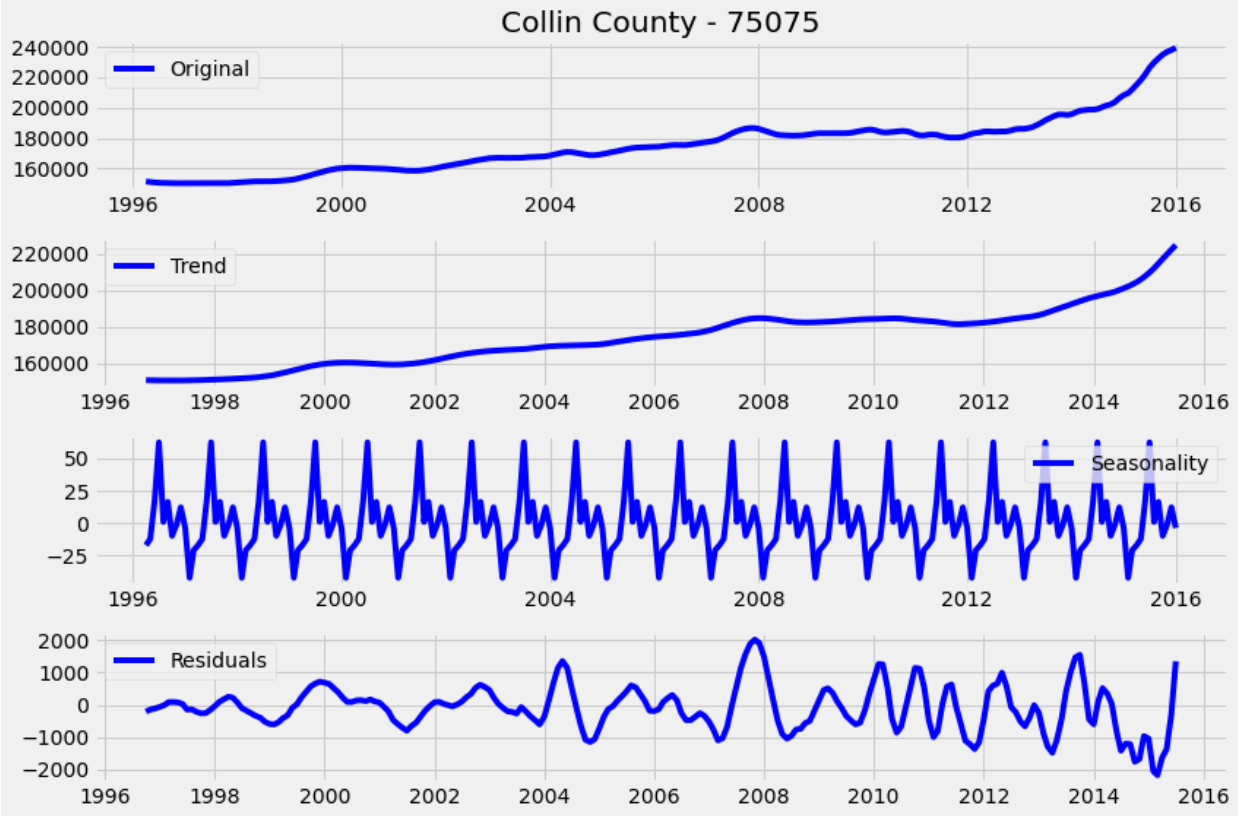
executed in 112ms, finished 09:22:21 2021-06-20

Out[163]: <AxesSubplot:xlabel='time'>



- Baseline 0.90 split

In [164]:
```
1 seasonal_decomposition(train_75075, 'Collin', 75075)
```
executed in 509ms, finished 09:22:21 2021-06-20



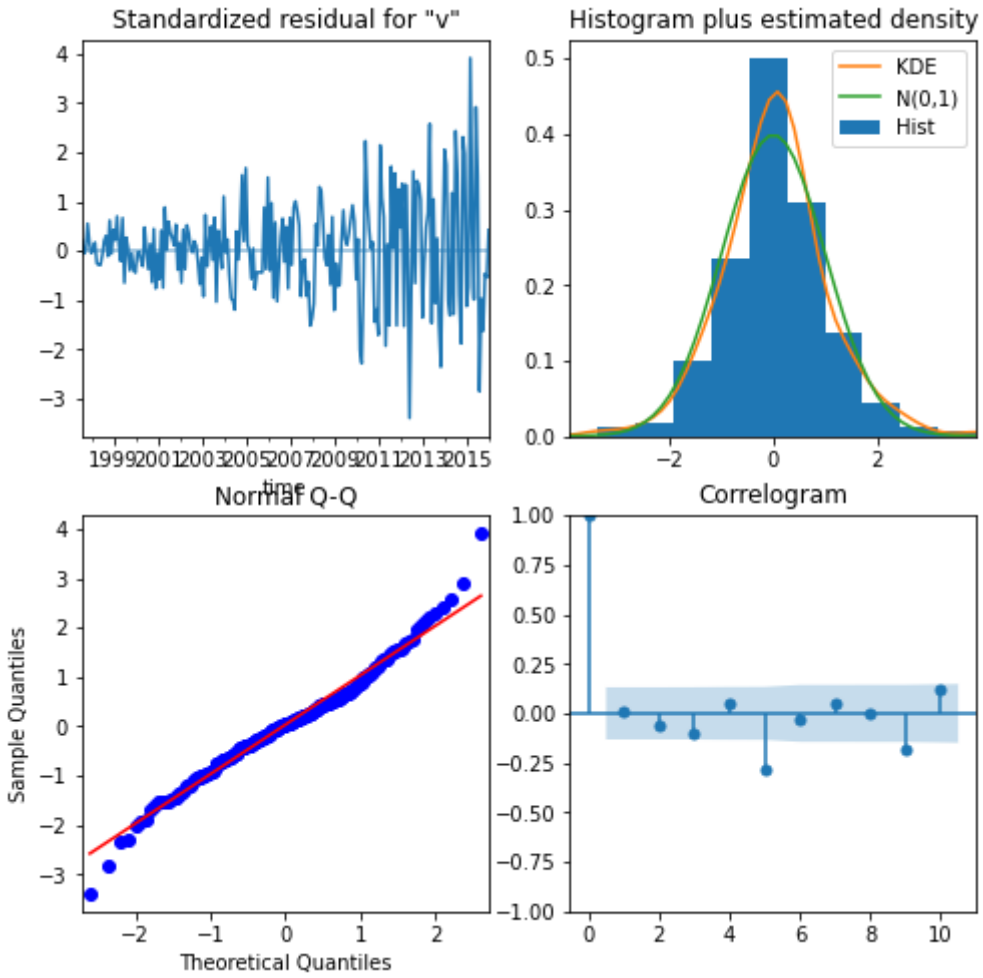
In [165]:
```
1 decomposition = seasonal_decompose(train_75075,model='Additive')
2 seasonal = decomposition.seasonal
3 seasonal.plot(figsize=(10,4), xlim=('2012-01-01', '2014-01-01'));
```
executed in 115ms, finished 09:22:22 2021-06-20

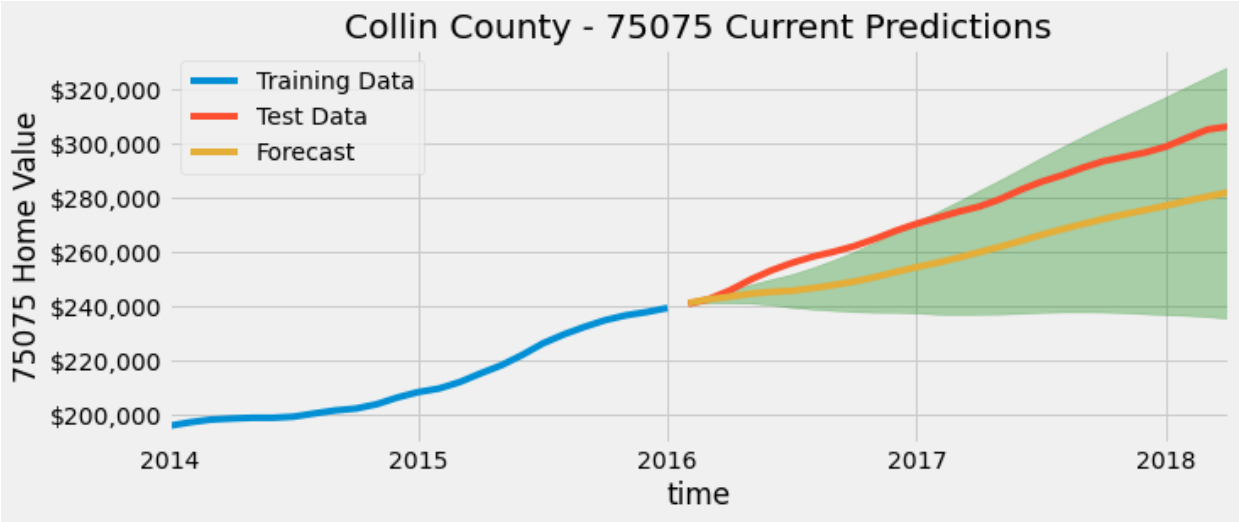
- There is an upward trend from 2002 to 2008 and then it dips slightly and begins to recover in 2012
- Data is annually seasonal with peaks in months (May-Aug) and dips in spring (Dec-Apr)
- Seasonality appears constant
- Residuals begin increasing in 2008

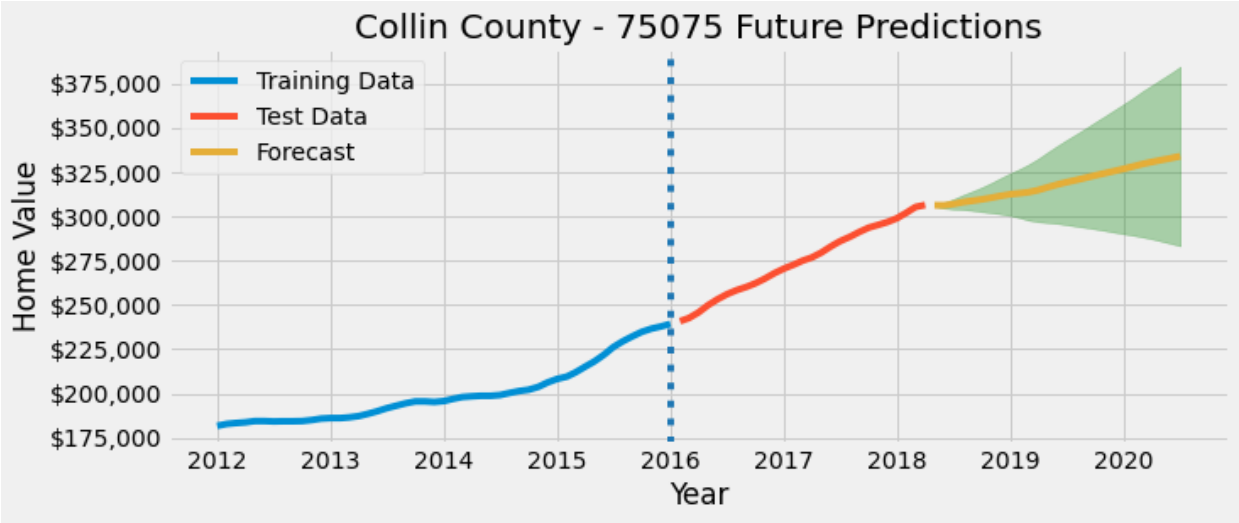
- Looks like data needs 1 degree of differencing on seasonality

In [166]:
```
1 fig_75075, future_75075, forecast_df_75075, roi_75075 = model_predictio
```

executed in 11.9s, finished 09:22:34 2021-06-20







In [167]:
```
1 roi_75075
```
executed in 3ms, finished 09:22:34 2021-06-20

Out[167]:
```
{'Lower CI': [-0.07295582287471396, 92704.41771252861, -7295.582287471392],
 'Upper CI': [0.2533666074028954, 125336.66074028953, 25336.66074028953],
 'Forecast': [0.09053616622890745, 109053.61662289075, 9053.616622890753]}
```

- Current predictions
  - Model undershoots predictions
  - Confidence interval captures all of the predicted values
- Future predictions
  - Titls upwards
  - Has a negative slope compared to test data
- An investment of $100,000 today (05/01/2018) by 07/01/2020 would yield (ROI):
  - Conservative estimate: -7.3% (-$7,295)
  - Mean estimate: 9.0% ($9,053)
  - Upper estimate: 25.3% (+$25,336)

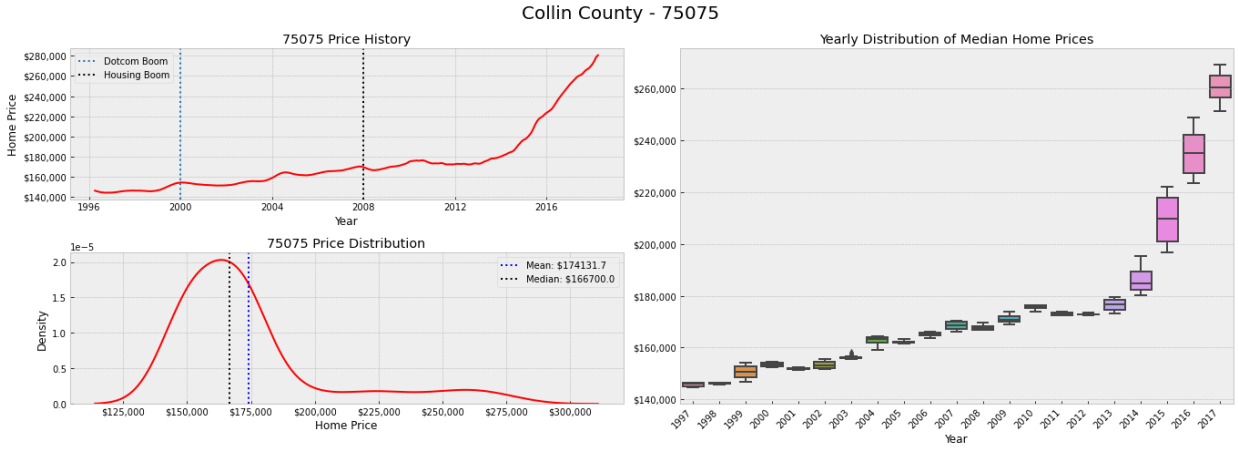
4.9.3 75023: EDA and SARIMAX

- Plano, Texas
- 24 miles from downtown Dallas

In [168]:
```
1 # Create 75075 dataframe
2
3 df_75023 = create_zip_data(Collin_dict_full, 75023)
```
executed in 5ms, finished 09:22:34 2021-06-20

In [169]:
```
1 zip_eda(df_75023, 75075, 'Collin')
```
executed in 634ms, finished 09:22:34 2021-06-20

Out[169]:
```
(<Figure size 1368x504 with 3 Axes>,
                value
count      265.000000
mean    174131.698113
std      31058.579411
min     144400.000000
25%     153400.000000
50%     166700.000000
75%     175700.000000
max     280500.000000)
```

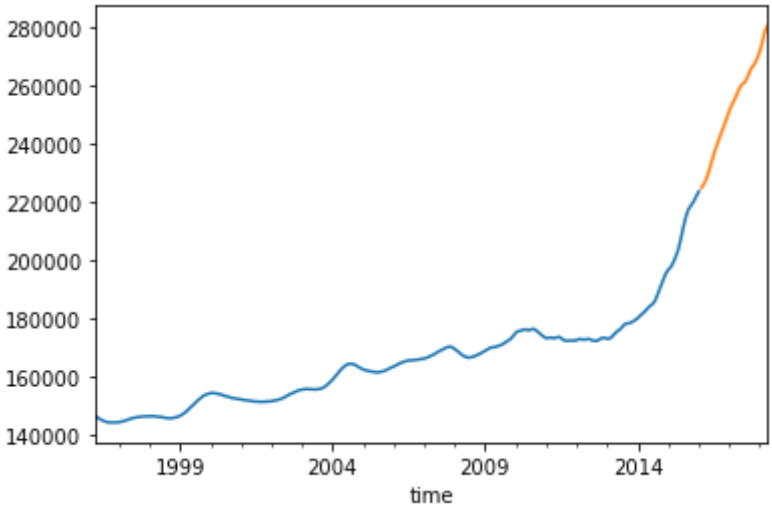


- Upward trend unitl 2008 and then stagnant decline unitl 2013
  - Median house price recovered in 2013, minimal dip between those years
- Data is right skewed as mean is greater than the median
- Based on the shape, prices have continuously trended upwards because of the right skew
- High spread in 2015 through 2017
  - Prior, narrow distribution

In [170]:
```
train_75023, test_75023 = create_train_test_split(df_75023, 0.90)
train_75023.plot()
test_75023.plot()
```

executed in 107ms, finished 09:22:34 2021-06-20

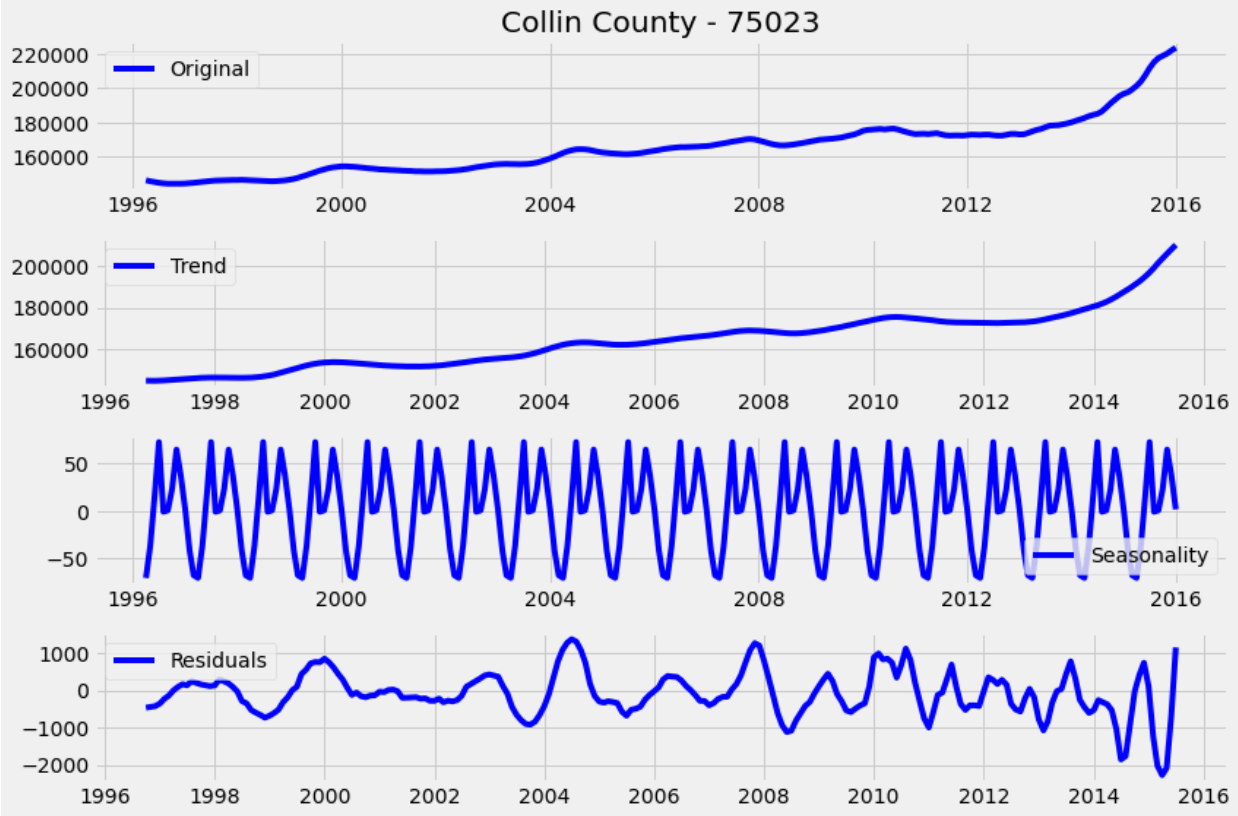
Out[170]: <AxesSubplot:xlabel='time'>



Baseline split of 0.90

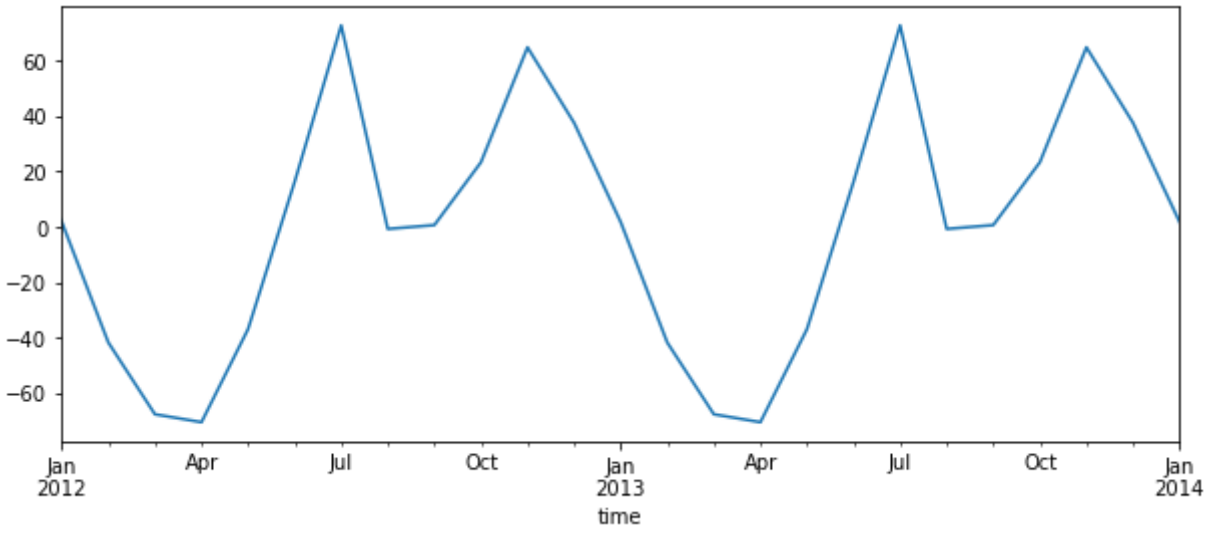
In [171]:
```
1 seasonal_decomposition(train_75023, 'Collin', 75023)
```

executed in 373ms, finished 09:22:35 2021-06-20



In [172]:
```
decomposition = seasonal_decompose(train_75023,model='Additive')
seasonal = decomposition.seasonal
seasonal.plot(figsize=(10,4), xlim=('2012-01-01', '2014-01-01'));
```

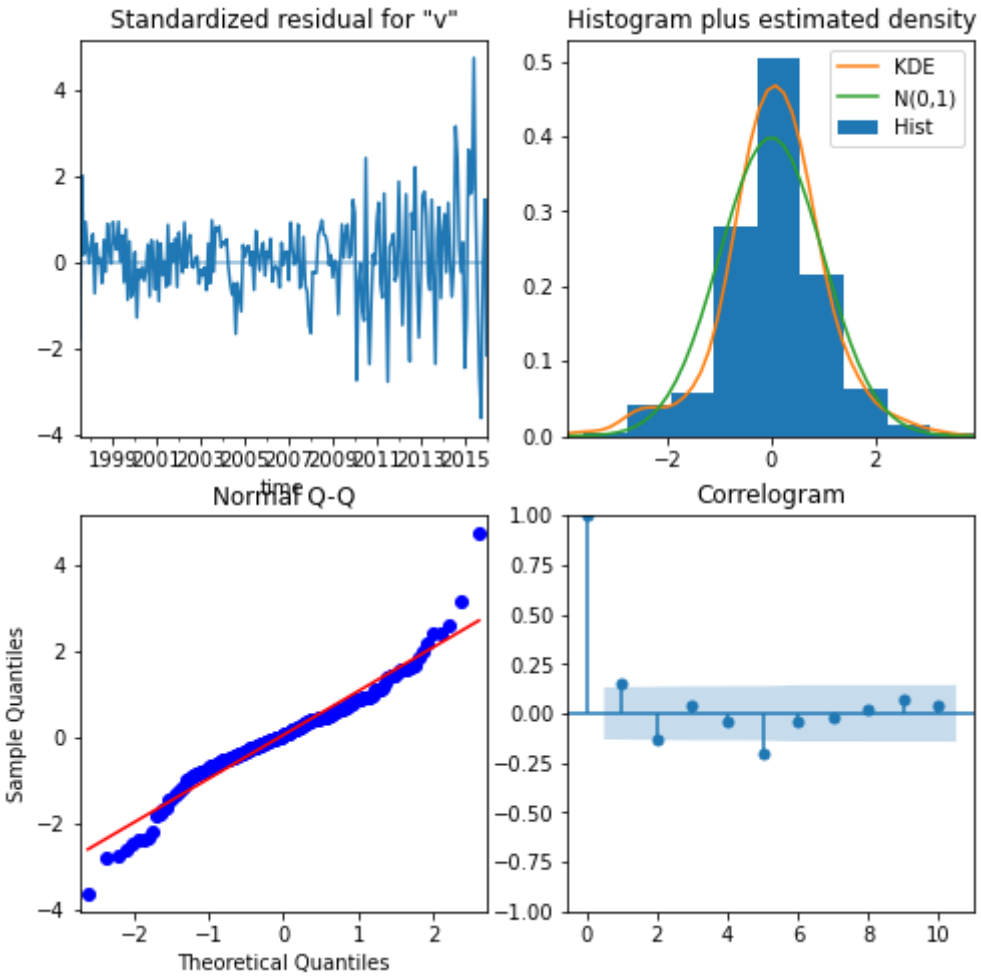
executed in 109ms, finished 09:22:35 2021-06-20

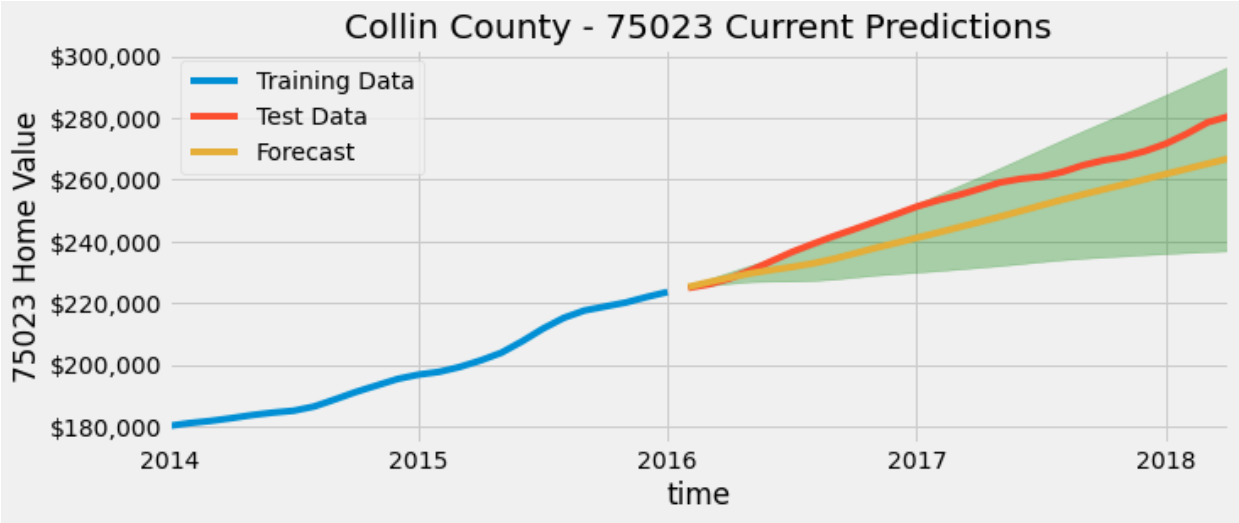


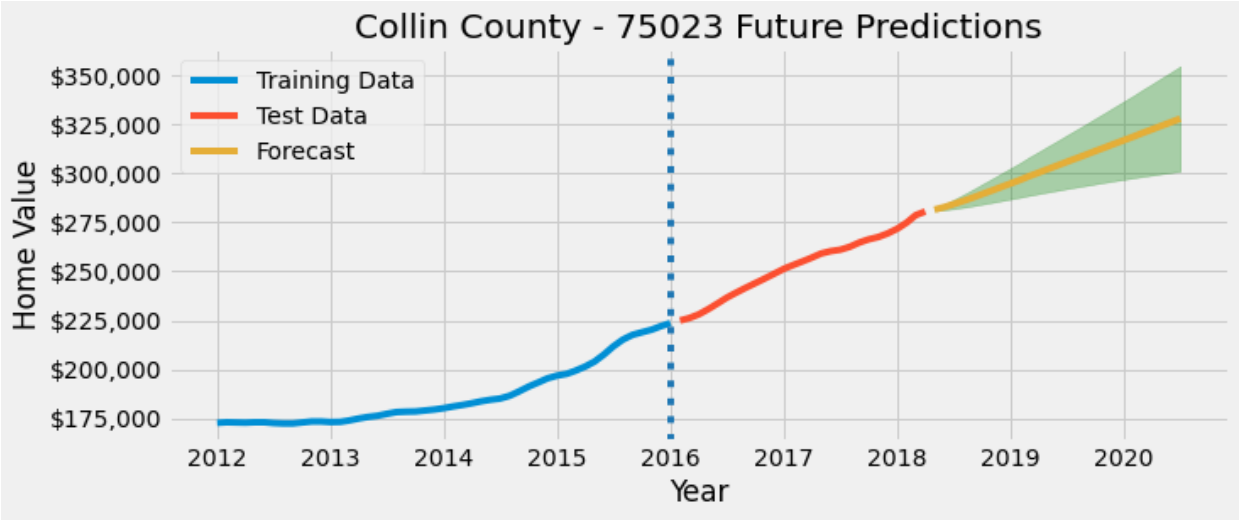
- There is an upward trend from 2002 to 2008 and then it dips slightly and begins to recover in 2010
- Data is annually seasonal with peaks in months (Jul-Nov) and dips in spring (Jan-Apr)
- Seasonality appears constant
- Residuals begin increasing in 2013
- Looks like data needs 1 degree of differencing on seasonality

In [173]:
```
1 fig_75023, future_75023, forecast_df_75023, roi_75023 = model_predictio
```

executed in 18.4s, finished 09:22:53 2021-06-20




Collin County - 75023 Current Predictions
Training Data
Test Data
Forecast
75023 Home Value
$300,000
$280,000
$260,000
$240,000
$220,000
$200,000
$180,000
2014
2015
2016
2017
2018
time


Collin County - 75023 Future Predictions
Training Data
Test Data
Forecast
Home Value
$350,000
$325,000
$300,000
$275,000
$250,000
$225,000
$200,000
$175,000
2012
2013
2014
2015
2016
2017
2018
2019
2020
Year

In [174]:
```
1 roi_75023
```
executed in 2ms, finished 09:22:53 2021-06-20

Out[174]:
```
{'Lower CI': [0.07191036322942652, 107191.03632294264, 7191.036322942644],
 'Upper CI': [0.2577965410207714, 125779.65410207714, 25779.65410207714],
 'Forecast': [0.1650364125346199, 116503.641253462, 16503.641253462003]}
```

- Current predictions
  - Model undershoots predictions
  - Confidence interval captures all of the predicted values
- Future predictions
  - Titls upwards
  - Has roughly the same slope as test data
- An investment of $100,000 today (05/01/2018) by 07/01/2020 would yield (ROI):
  - Conservative estimate: 7.2% ($7,191)
  - Mean estimate: 16.5% ($16,503)
  - Upper estimate: 25.8% (+$25,779)

4.10 Collin County Conclusion

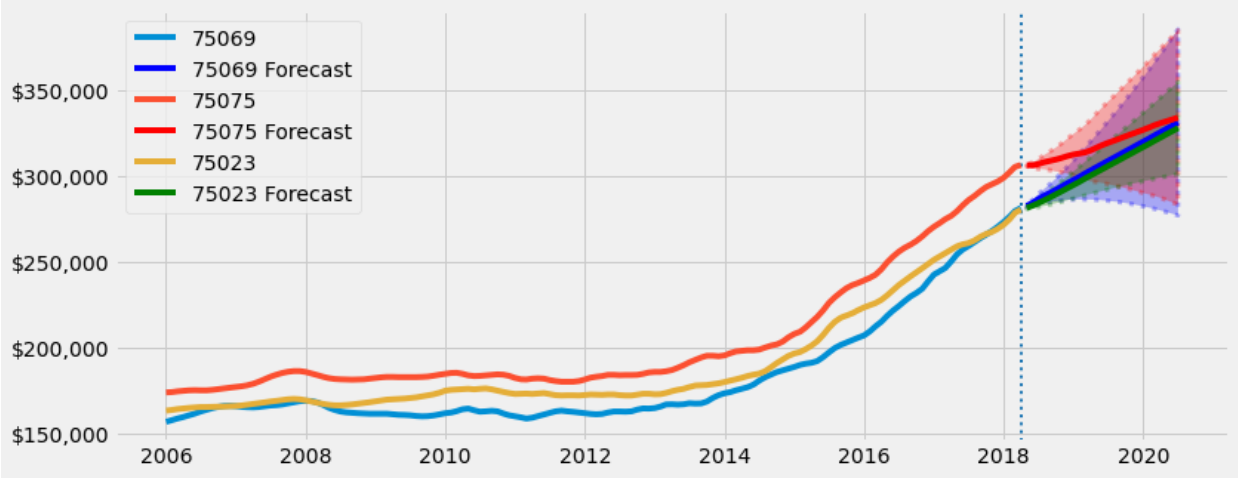
In [175]:
```
1 corr_check(df_75069, df_75075, df_75023, 75069, 75075, 75023)
```
executed in 5ms, finished 09:22:53 2021-06-20

Out[175]:

	75069	75075	75023
75069	1.000000	0.994549	0.992293
75075	0.994549	1.000000	0.997449
75023	0.992293	0.997449	1.000000

- All of the correlations are so similar that there is not much information to be gained in trying to compare the similarity of their movements

In [176]:
```
1 county_forecast_comparison(df_75069, 75069, forecast_df_75069, df_75075
```
executed in 140ms, finished 09:22:53 2021-06-20



- 75069 has the highest ending point but also has a very large confidence interval
  - Has a lower splope than test data
- 75075 & 75023 move very tightly together, especially for predictions
  - Both have very similar predictions

In [177]:
```
1 collin_perc_comparison = county_forecast_perc_comparison(roi_75069, 750
2 collin_perc_comparison
```
executed in 8ms, finished 09:22:53 2021-06-20

Out[177]:

	Lower CI	Upper CI	Forecast
75069 Perc Change	-0.017314	0.359631	0.171604
Val	98268.636270	135963.065897	117160.388415
$ Diff	-1731.363730	35963.065897	17160.388415
75075 Perc Change	-0.072956	0.253367	0.090536
Val	92704.417713	125336.660740	109053.616623
$ Diff	-7295.582287	25336.660740	9053.616623
75023 Perc Change	0.071910	0.257797	0.165036
Val	107191.036323	125779.654102	116503.641253
$ Diff	7191.036323	25779.654102	16503.641253

- Based on the downside risk, upside return, and mean predicted value, 75023 seems like the superior zip code
- It has the second highest upside, second highest predicted mean, and lowest downside risk
- 75069 has a similar risk profile but more downside risk
- 75075 is the least desirable
- **In Collin county, 75023 has the best prospects for near term growth**

4.11 Denton County

- Located in Dallas-Fort Worth metro
- Population: 662,614

4.11.1 75057: EDA and SARIMAX

- Lewisville, Texas
- 27 miles from downtown Dallas

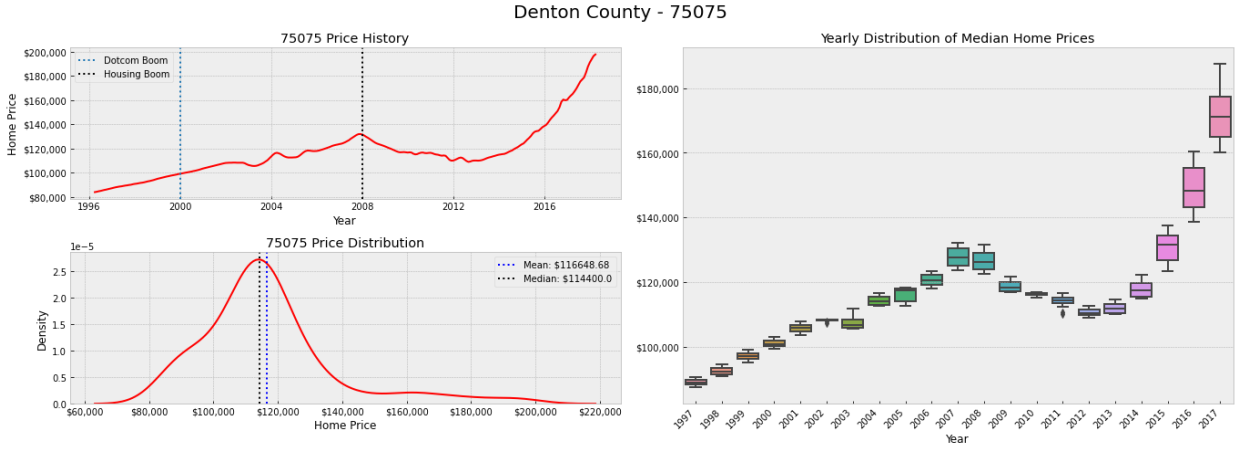
In [178]:
```
# Create 75057 dataframe

df_75057 = create_zip_data(Denton_dict_full, 75057)
```

executed in 6ms, finished 09:22:53 2021-06-20

In [179]:
```
zip_eda(df_75057, 75075, 'Denton')
```

executed in 645ms, finished 09:22:54 2021-06-20

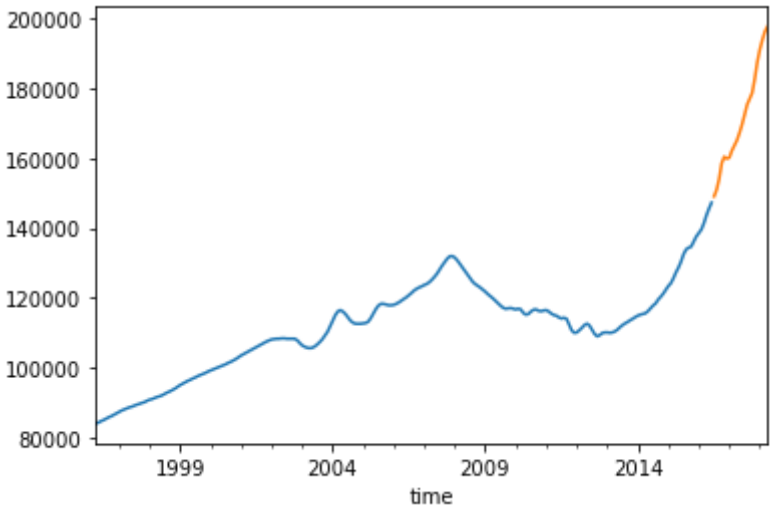
Out[179]:
```
(<Figure size 1368x504 with 3 Axes>,
                value
count      265.000000
mean    116648.679245
std      21164.212042
min      84000.000000
25%     105900.000000
50%     114400.000000
75%     122400.000000
max     197800.000000)
```



- Upward trend unitl 2008 and then large decline unitl 2013
  - Median house price recovered in 2015, significant dip between those years
- Data is right skewed as mean is greater than the median
- Based on the shape, prices have continuously trended upwards because of the right skew
- Spreads have been increasing from 2015 onwards, relatively large in 2007 and 2008
  - Prior, narrow distribution

In [180]:
```
train_75057, test_75057 = create_train_test_split(df_75057, 0.92)
train_75057.plot()
test_75057.plot();
```

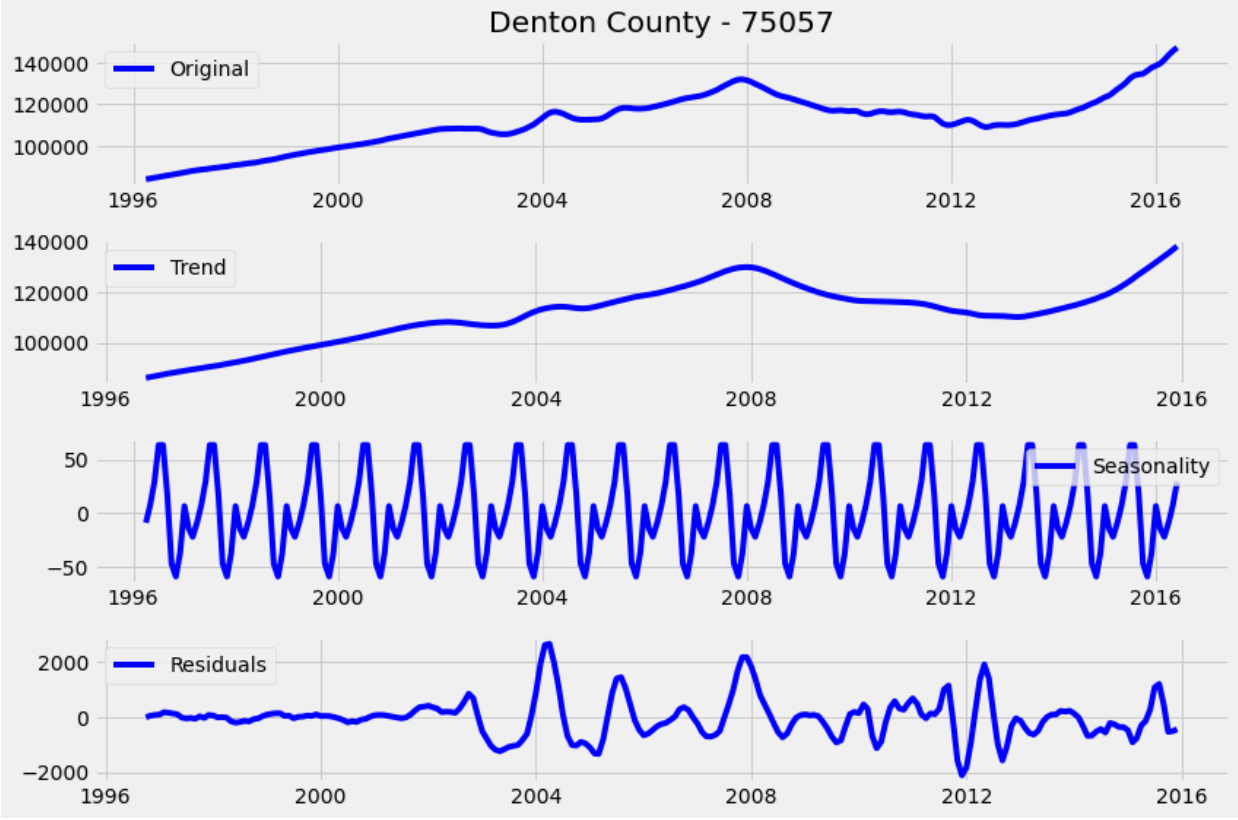
executed in 98ms, finished 09:22:54 2021-06-20



- Baseline split of 0.90
- Readjusted to 0.92 to capture more trend

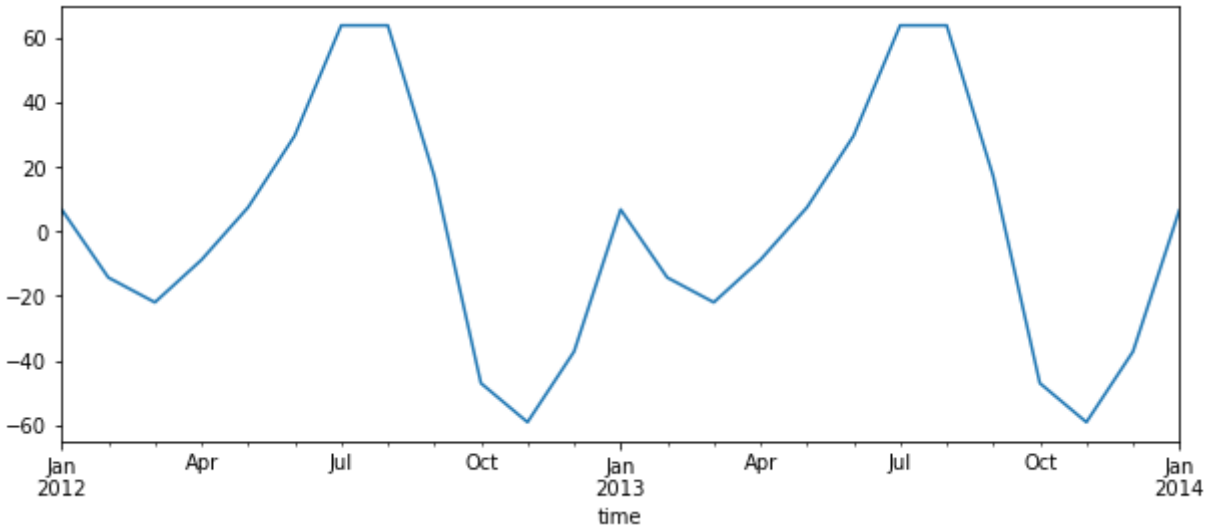
In [181]:
```
seasonal_decomposition(train_75057, 'Denton', 75057)
```

executed in 466ms, finished 09:22:55 2021-06-20



In [182]:
```
decomposition = seasonal_decompose(train_75057,model='Additive')
seasonal = decomposition.seasonal
seasonal.plot(figsize=(10,4), xlim=('2012-01-01', '2014-01-01'));
```

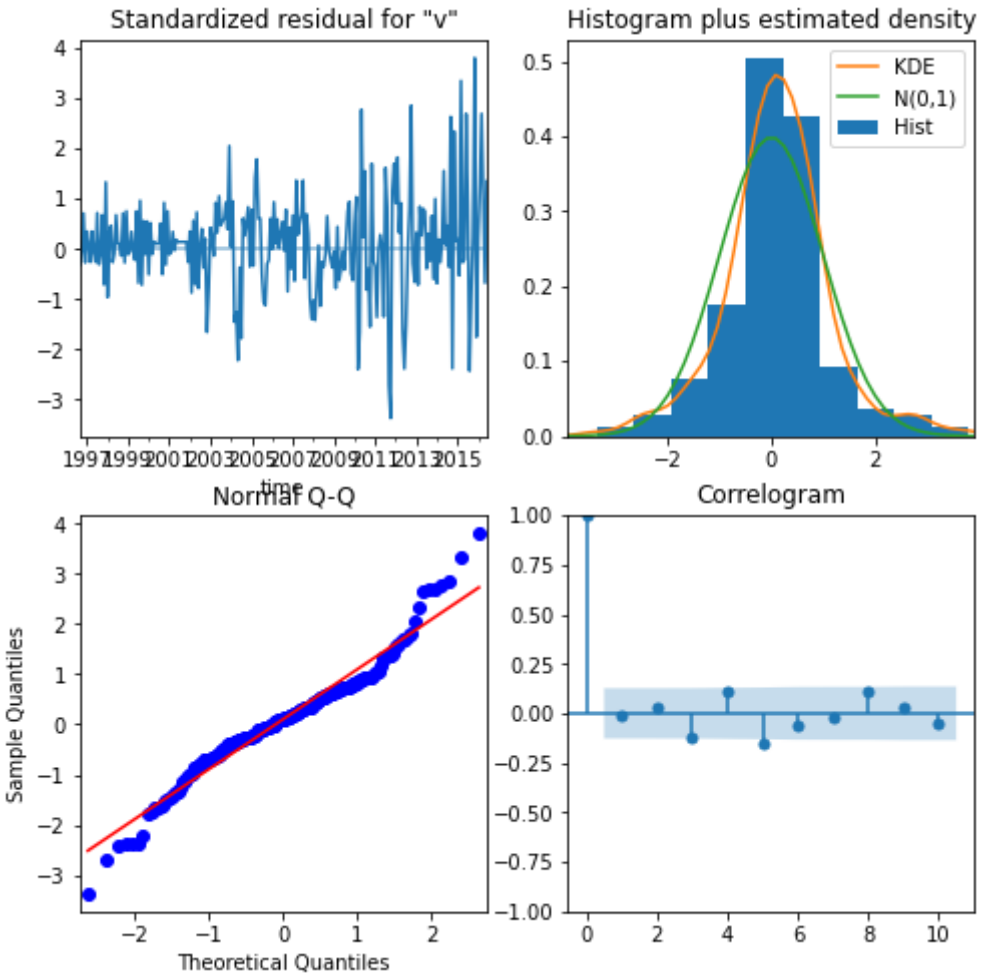
executed in 109ms, finished 09:22:55 2021-06-20

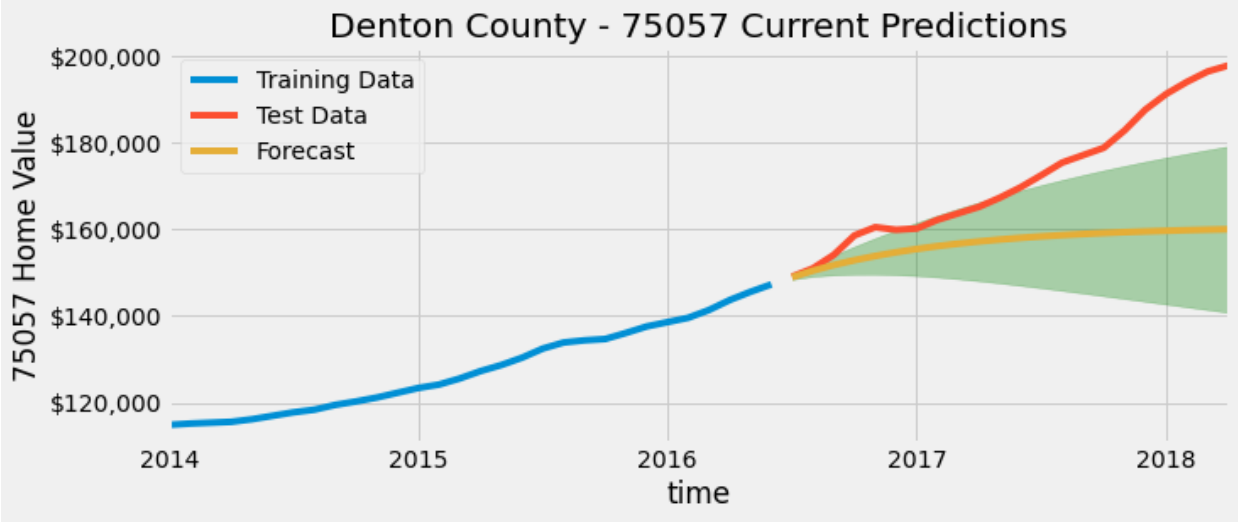


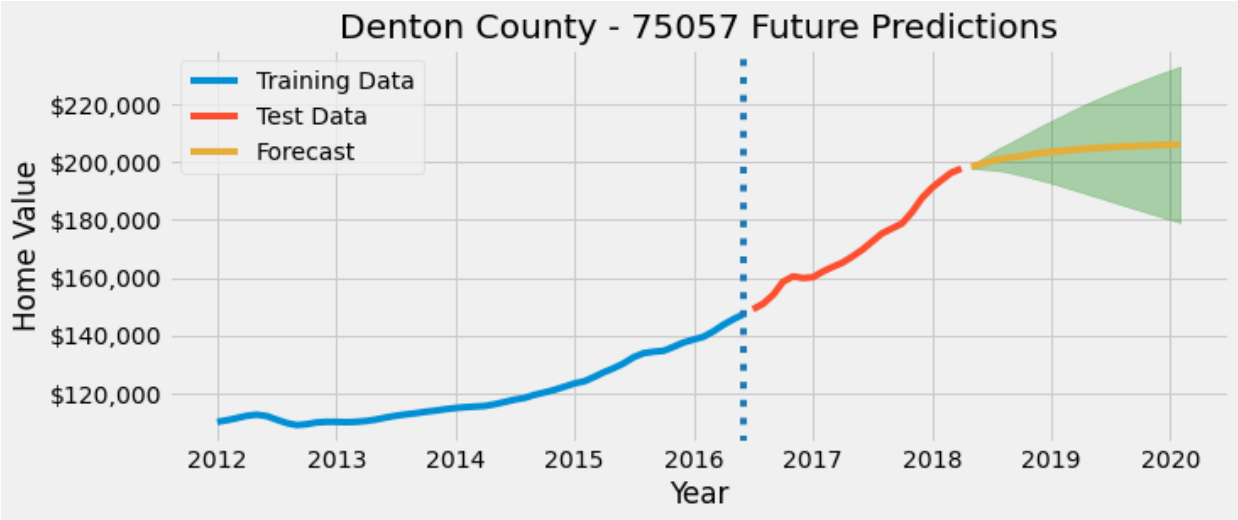
- There is an upward trend from until 2008 when there is a dip and recovery beginning in 2014
- Data is annually seasonal with peaks in months (May-Aug) and dips in spring (Sept-Dec)
- Seasonality appears constant
- Residuals begin increasing in 2012 but taper back to low levels
- Looks like data needs 1 degree of differencing on seasonality

In [183]:
```
1 fig_75057, future_75057, forecast_df_75057, roi_75057 = model_predictio
```

executed in 7.36s, finished 09:23:02 2021-06-20







In [184]:
```
1 roi_75057
```

executed in 2ms, finished 09:23:02 2021-06-20

Out[184]:
```
{'Lower CI': [-0.09476981692371161, 90523.01830762884, -9476.981692371162],
 'Upper CI': [0.17048600838397718, 117048.60083839772, 17048.600838397717],
 'Forecast': [0.03831553119596971, 103831.55311959698, 3831.5531195969816]}
```

- Current predictions
  - Model undershoots predictions
  - Confidence interval captures values unitl mid 2017 when test data slopes up and predictions slope downwards

- Future predictions
  - Titls slighlty upward
  - Has a lower slope than the test data
- An investment of $100,000 today (05/01/2018) by 02/01/2020 would yield (ROI):
  - Conservative estimate: -9.5% (-$9,476)
  - Mean estimate: 3.8% ($3,831)
  - Upper estimate: 17.0% (+$17,048)

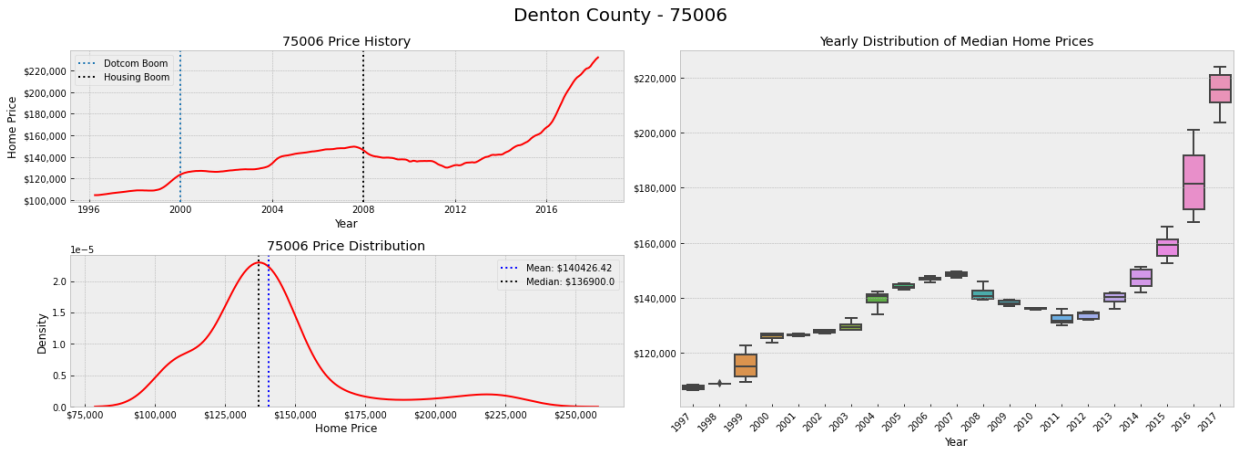
4.11.2 75006: EDA and SARIMAX

- Carroltown, Texas
- 18 miles from downtown Dallas

In [185]:
```
# Create 75006 dataframe

df_75006 = create_zip_data(Denton_dict_full, 75006)
```
executed in 4ms, finished 09:23:02 2021-06-20

In [186]:
```
zip_eda(df_75006, 75006, 'Denton')
```
executed in 665ms, finished 09:23:03 2021-06-20

Out[186]:
```
(<Figure size 1368x504 with 3 Axes>,
                value
count      265.000000
mean    140426.415094
std      26290.420186
min     104500.000000
25%     126900.000000
50%     136900.000000
75%     146600.000000
max     232200.000000)
```

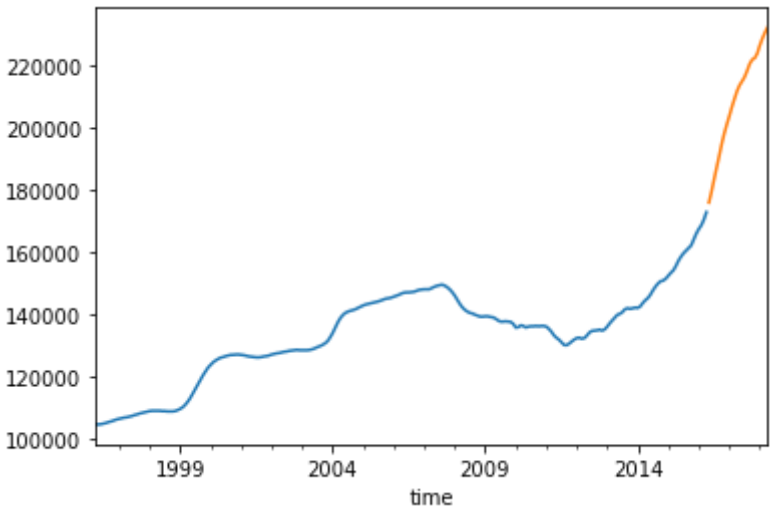


- Upward trend unitl 2008 and then large decline unitl 2012
  - Median house price recovered in 2015, significant dip between those years
- Data is right skewed as mean is greater than the median
- Based on the shape, prices have continuously trended upwards because of the right skew
- Spread was largest in 2016

- Prior, narrow distribution

In [187]:
```
train_75006, test_75006 = create_train_test_split(df_75006, 0.91)
train_75006.plot()
test_75006.plot();
```

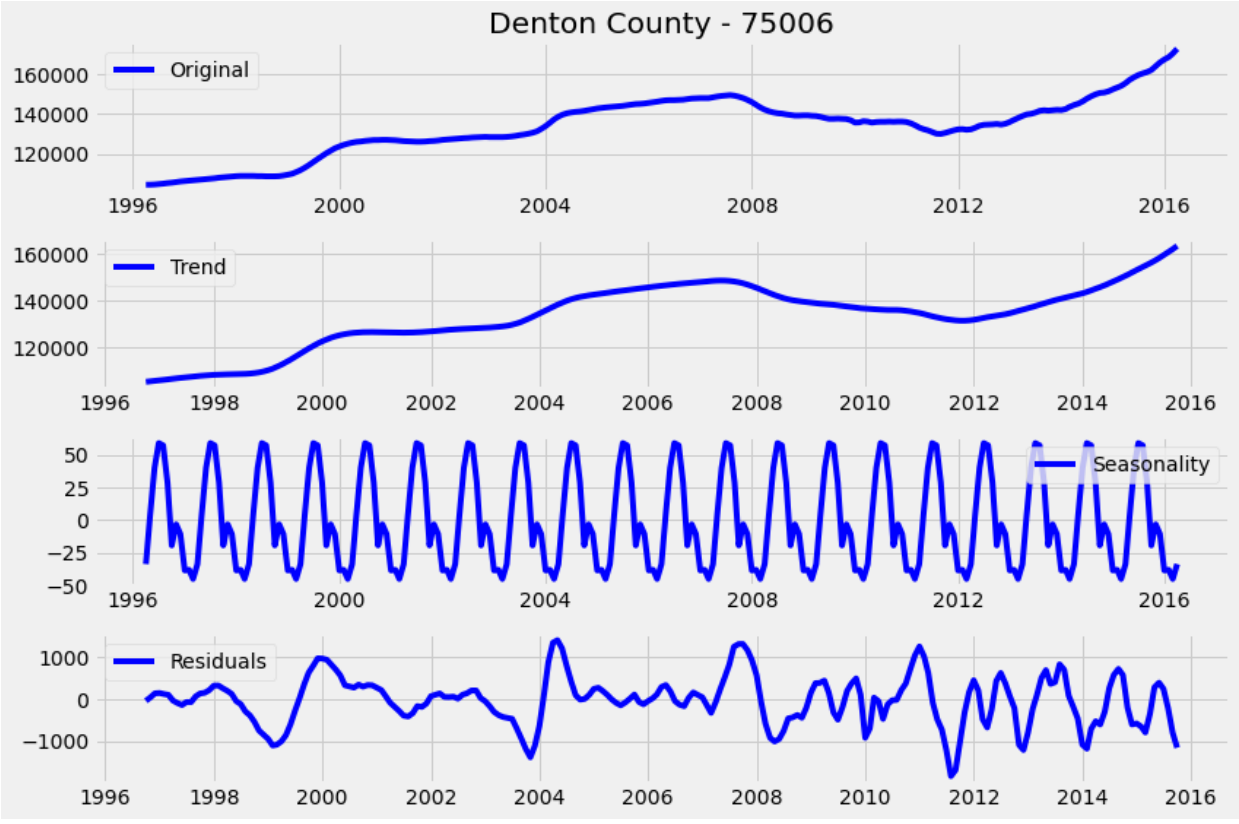
executed in 99ms, finished 09:23:03 2021-06-20



- Baseline split of 0.91
- Adjusted to 0.91 to capture more trend

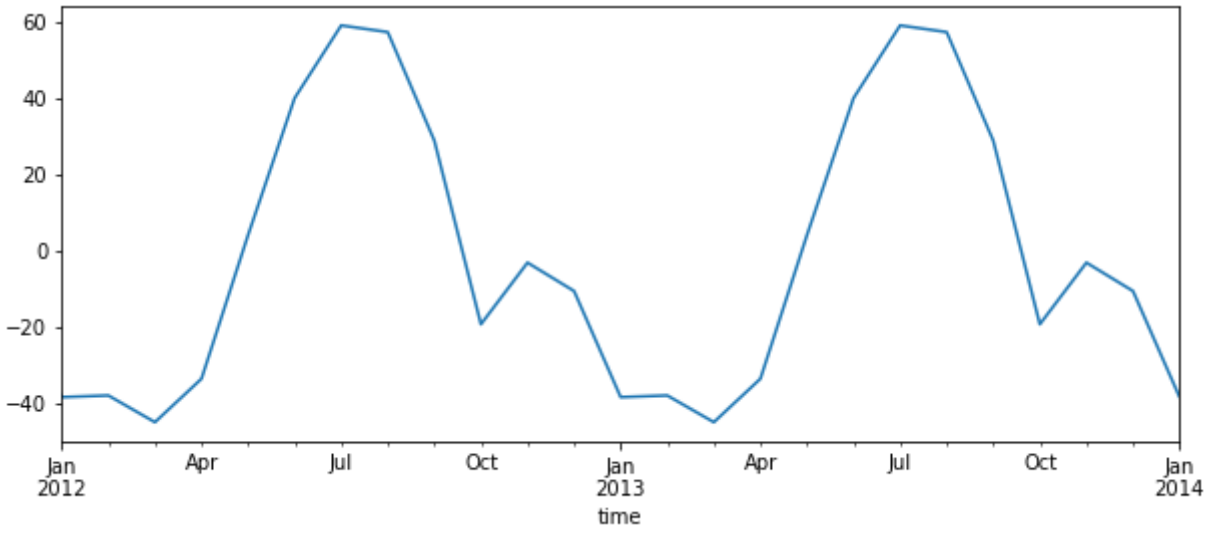
In [188]:
```
seasonal_decomposition(train_75006, 'Denton', 75006)
```

executed in 375ms, finished 09:23:03 2021-06-20



In [189]:
```
decomposition = seasonal_decompose(train_75006, model='Additive')
seasonal = decomposition.seasonal
seasonal.plot(figsize=(10,4), xlim=('2012-01-01', '2014-01-01'));
```

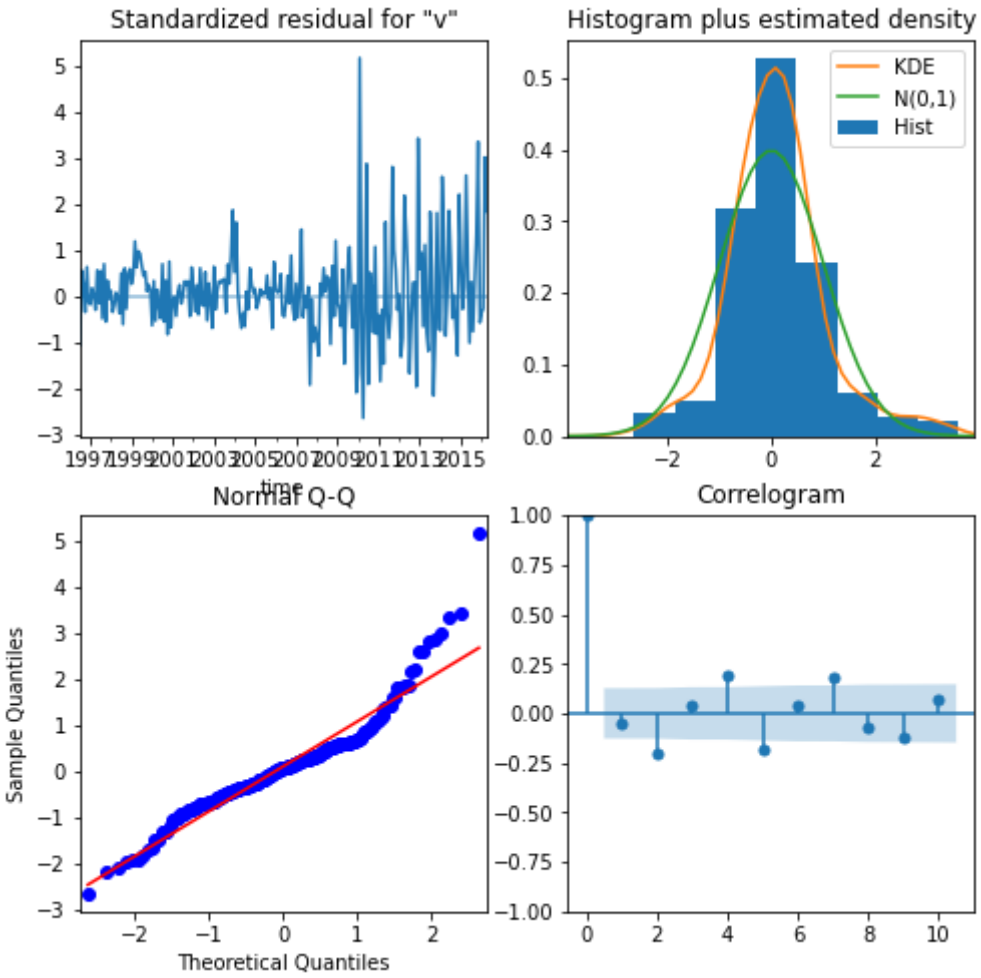
executed in 107ms, finished 09:23:03 2021-06-20

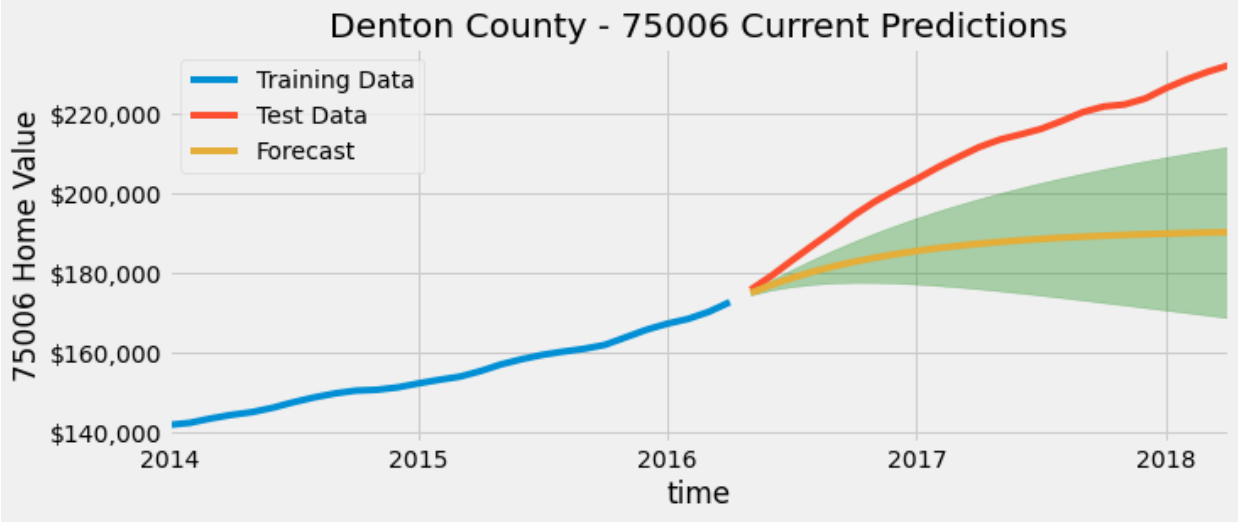


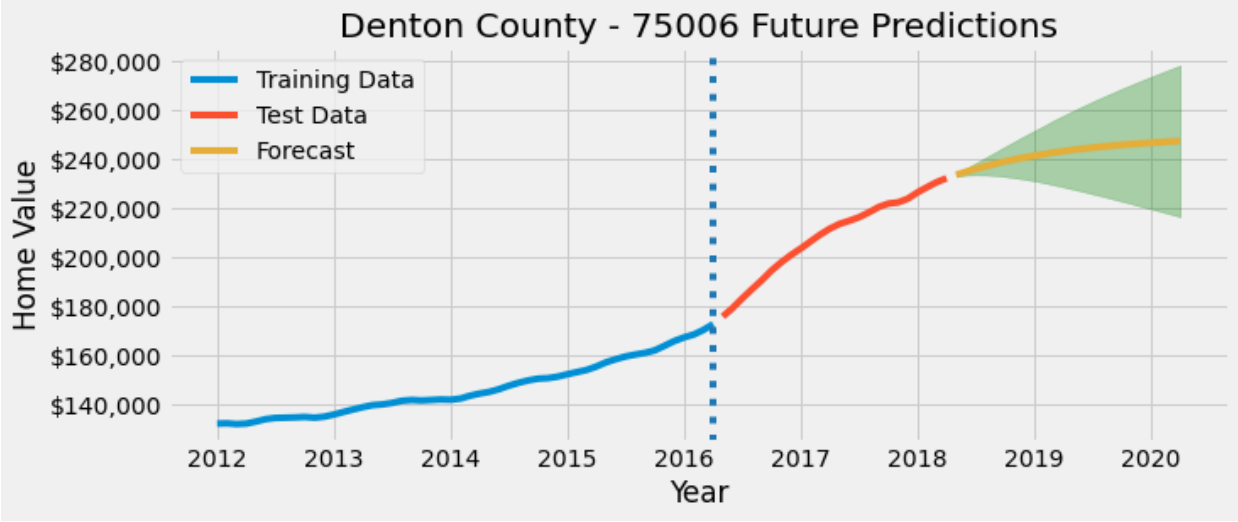
- There is an upward trend from until 2008 when there is a dip and recovery beginning in 2012
- Data is annually seasonal with peaks in months (May-Aug) and dips in spring (Nov-Feb)
- Seasonality appears constant
- Residuals begin increasing in 2012 but taper back to low levels
- Looks like data needs 1 degree of differencing on seasonality

In [190]:
```
1 fig_75006, future_75006, forecast_df_75006, roi_75006 = model_predictio
```

executed in 3.26s, finished 09:23:07 2021-06-20







In [191]:
```
1 roi_75006
```

executed in 2ms, finished 09:23:07 2021-06-20

Out[191]:
```
{'Lower CI': [-0.07082745962191238, 92917.25403780876, -7082.74596219124
2],
 'Upper CI': [0.18818875744123817, 118818.87574412383, 18818.87574412382
7],
 'Forecast': [0.05898410931867103, 105898.4109318671, 5898.410931867096]}
```

- Current predictions
  - Model undershoots predictions, and tilts upwards and then downwards
  - Confidence interval follows trend but but does not capture test data
- Future predictions

    - Titls slighlty upward, then a less positive slope
    - Has a lower slope than the test data, more flat
- An investment of $100,000 today (05/01/2018) by 04/01/2020 would yield (ROI):
    - Conservative estimate: -7.0% (-$7,082)
    - Mean estimate: 5.9% ($5,898)
    - Upper estimate: 18.8% (+$18,818)

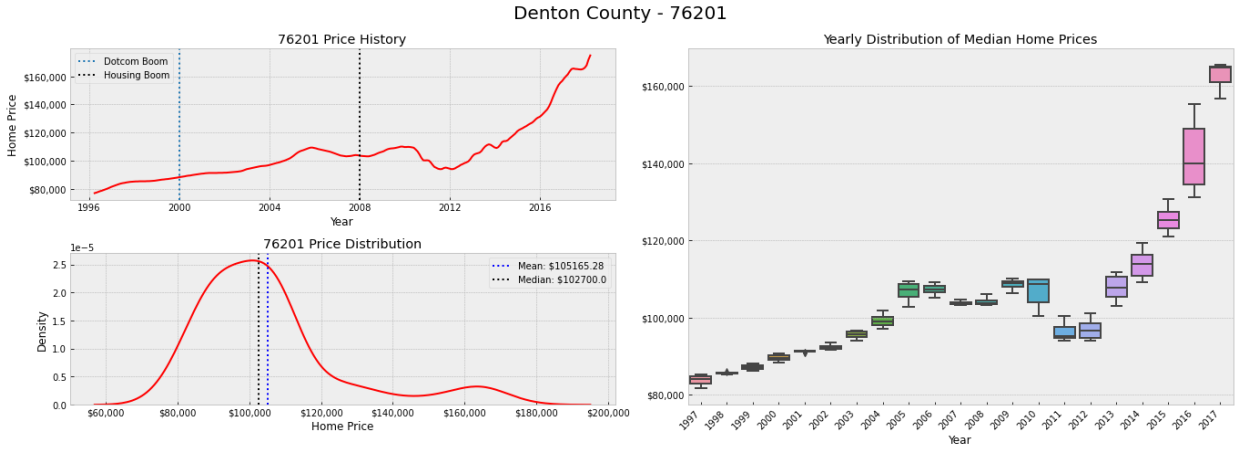
4.11.3 76201: EDA and SARIMAX

- Denton, Texas
- 40 miles from downtown Dallas

In [192]:
```
# Create 76201 dataframe

df_76201 = create_zip_data(Denton_dict_full, 76201)
```
executed in 4ms, finished 09:23:07 2021-06-20

In [193]:
```
zip_eda(df_76201, 76201, 'Denton')
```
executed in 645ms, finished 09:23:07 2021-06-20

Out[193]:
```
(<Figure size 1368x504 with 3 Axes>,
                value
count      265.000000
mean    105165.283019
std      20572.239736
min      77100.000000
25%      91500.000000
50%     102700.000000
75%     109400.000000
max     174800.000000)
```



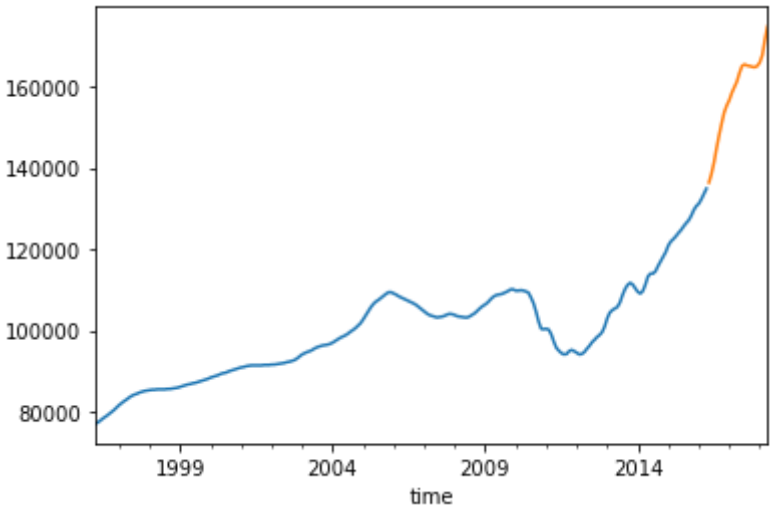
- Upward trend unitl 2006 and then large decline unitl 2008, took another dip and recoverd in 2013
    - Median house price recovered in 2014, significant dip between those years
- Data is right skewed as mean is greater than the median
- Based on the shape, prices have continuously trended upwards because of the right skew
- Spread was largest in 2016

- Prior, narrow distribution

In [194]:
```
train_76201, test_76201 = create_train_test_split(df_76201, 0.91)
train_76201.plot()
test_76201.plot()
```

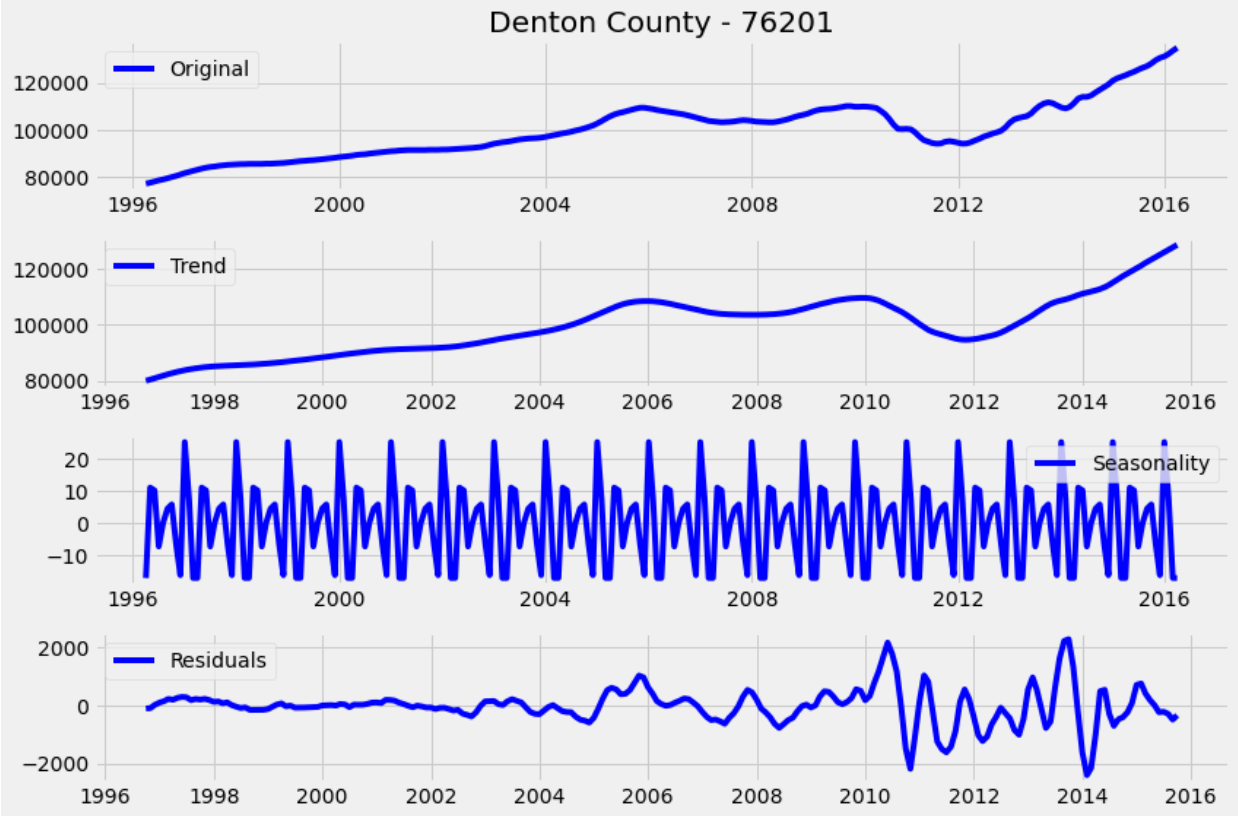
executed in 92ms, finished 09:23:07 2021-06-20

Out[194]: <AxesSubplot:xlabel='time'>

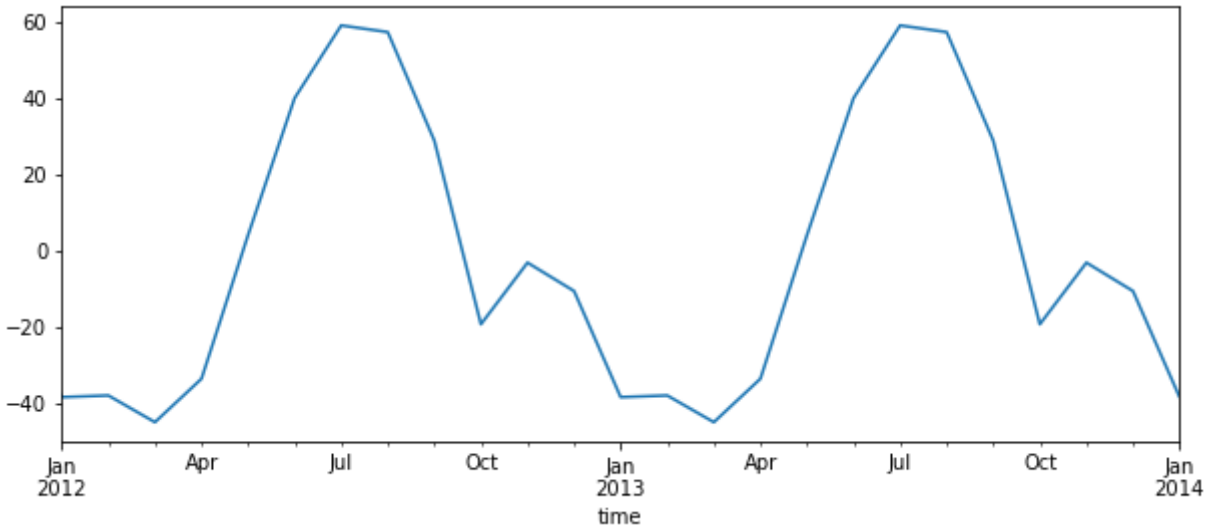


- Baseline split 0.90
- Adjusted to 0.91 to try and capture trend

In [195]:
```
1 seasonal_decomposition(train_76201, 'Denton', 76201)
```
executed in 368ms, finished 09:23:08 2021-06-20



In [196]:
```
1 decomposition = seasonal_decompose(train_75006, model='Additive')
2 seasonal = decomposition.seasonal
3 seasonal.plot(figsize=(10,4), xlim=('2012-01-01', '2014-01-01'));
```
executed in 112ms, finished 09:23:08 2021-06-20

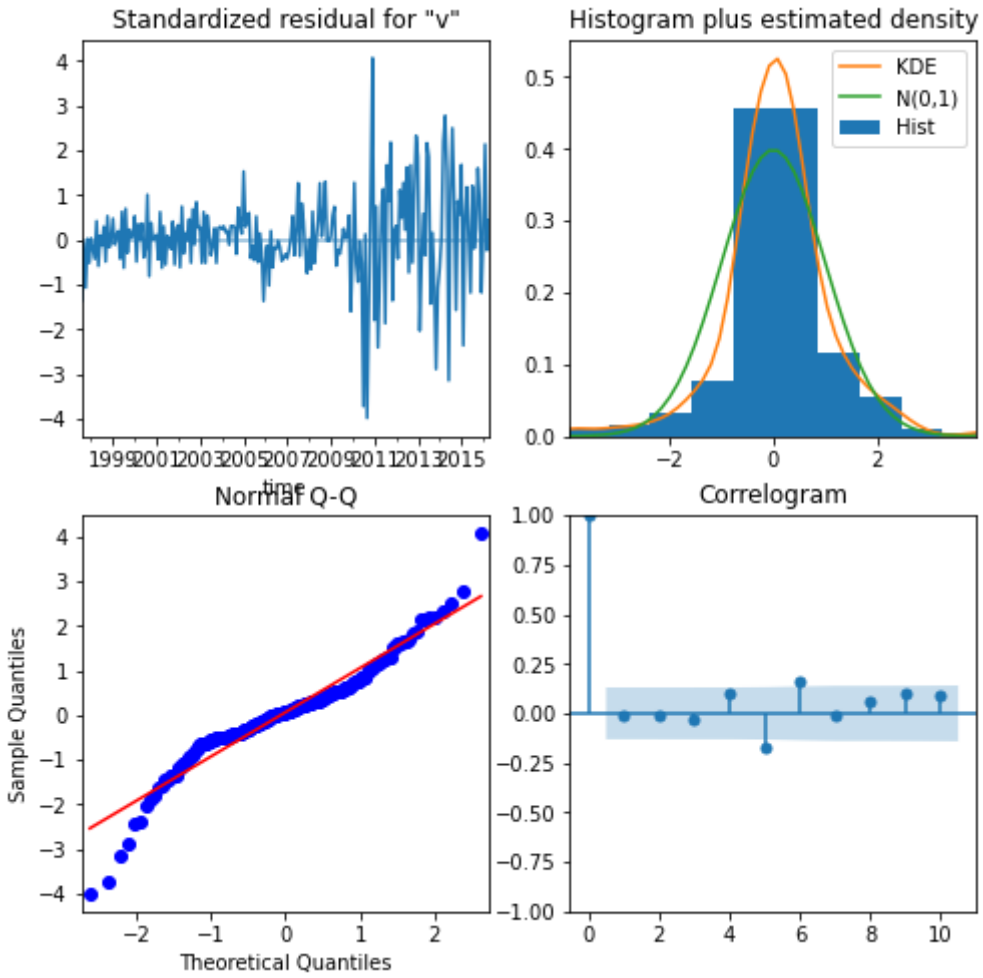


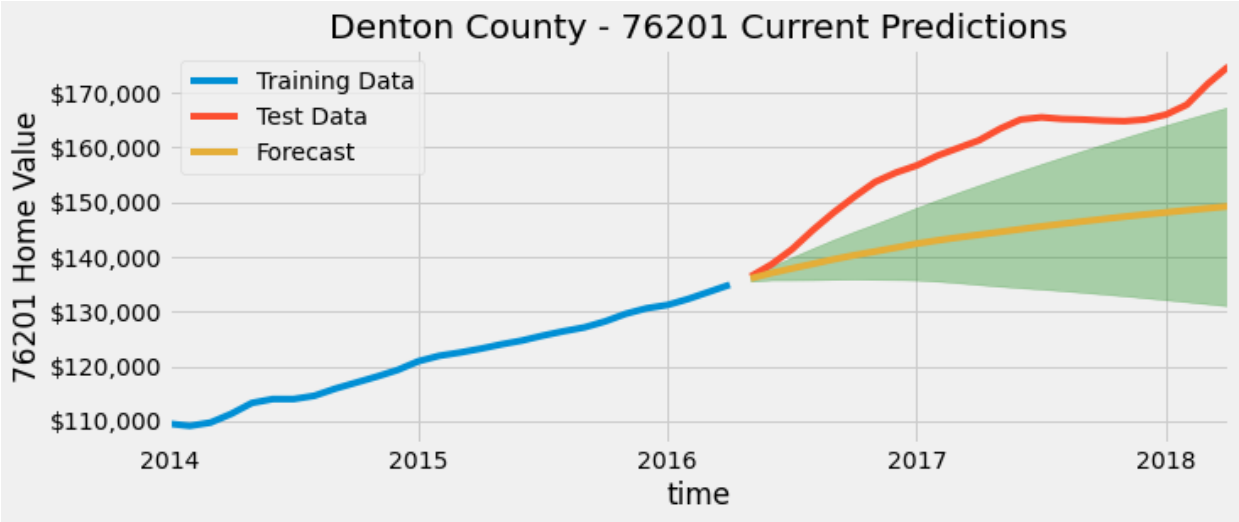
- There is an upward trend from until 2006 when there is a dip in 2008 and recovery beginning in 2012
- Data is annually seasonal with peaks in months (May-Aug) and dips in spring (Nov-Feb)
- Seasonality appears constant
- Residuals are largest in 2010 and 2014

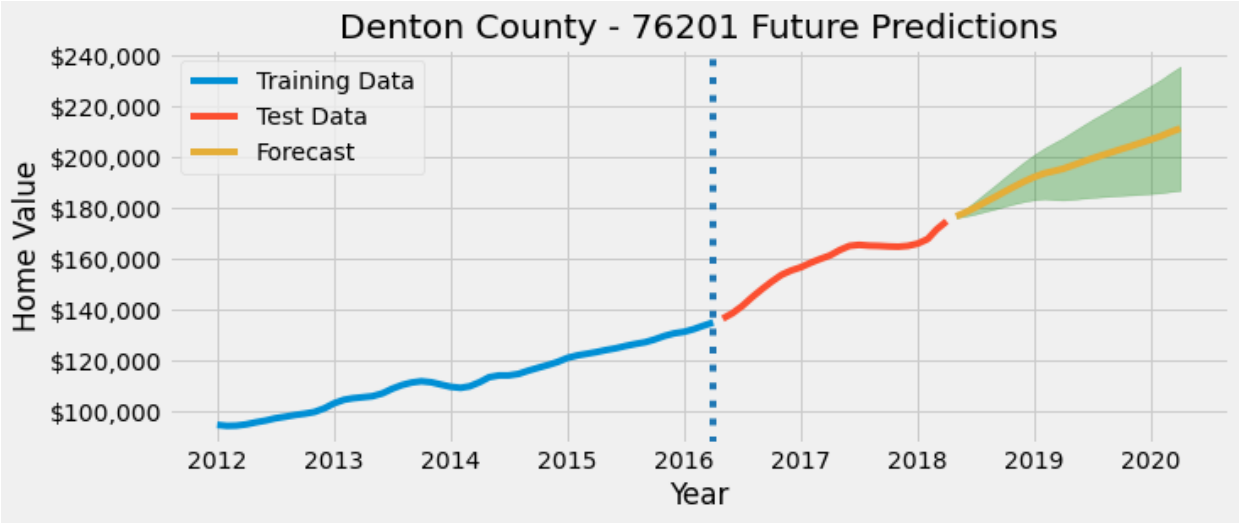
- Looks like data needs 1 degree of differencing on seasonality

In [197]:
```
1 fig_76201, future_76201, forecast_df_76201, roi_76201 = model_predictio
```

executed in 38.6s, finished 09:23:46 2021-06-20







In [198]:
```
1 roi_76201
```
executed in 2ms, finished 09:23:46 2021-06-20

Out[198]:
```
{'Lower CI': [0.0614400367648268, 106144.0036764827, 6144.003676482695],
 'Upper CI': [0.32978963819650703, 132978.9638196507, 32978.9638196507],
 'Forecast': [0.19608102824695128, 119608.10282469515, 19608.102824695146]}
```

- Current predictions
  - Model undershoots predictions, and tilts upwards and then downwards
  - Confidence interval follows trend but but does not capture test data
- Future predictions
  - Titls slighlty upward, then a less positive slope
  - Follows the same trend as the test data
- An investment of $100,000 today (05/01/2018) by 04/01/2020 would yield (ROI):
  - Conservative estimate: 6.1% (-$6,144)
  - Mean estimate: 33.0% ($32,978)
  - Upper estimate: 19.6% (+$19,608)

4.12 Denton County Conclusion

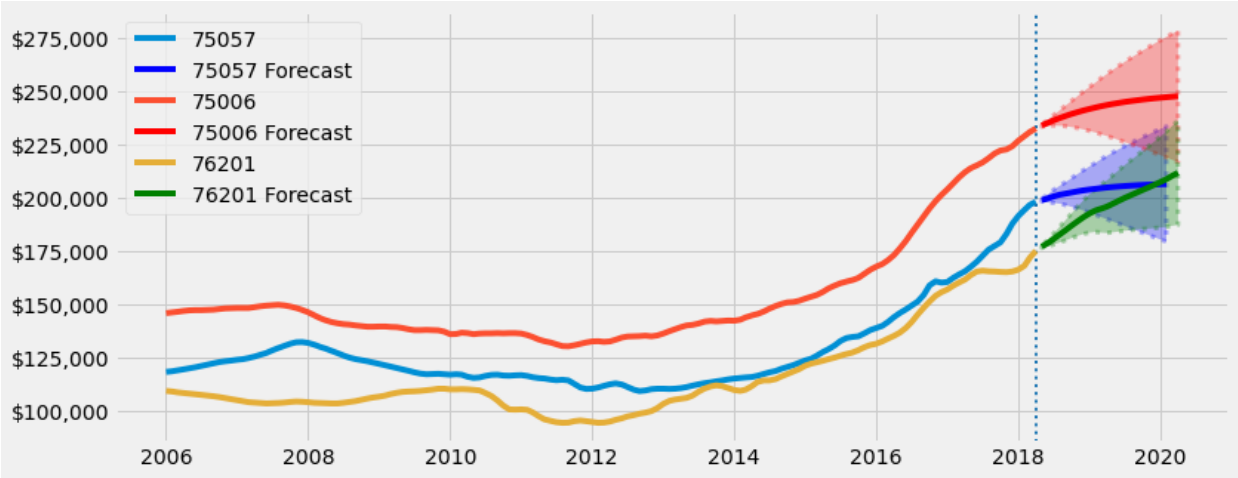
In [199]:
```
1 corr_check(df_75057, df_75006, df_76201, 75057, 75006, 76201)
```
executed in 5ms, finished 09:23:46 2021-06-20

Out[199]:

	75057	75006	76201
75057	1.000000	0.983245	0.957606
75006	0.983245	1.000000	0.979073
76201	0.957606	0.979073	1.000000

- 75057 moves more closely with 76201
- 75006 moves more closely with 75057, very thin margin
- 76201 moves more closely with 75006

In [200]:
```
1 county_forecast_comparison(df_75057, 75057, forecast_df_75057, df_75006
```
executed in 150ms, finished 09:23:47 2021-06-20



- 75006 has the highest ending point but also has a very large confidence interval
- 75057 & 76201 move very tightly together but have differing predictions
  - 76201 has a much more positive slope than 75057

In [201]:
```
1 denton_perc_comparison = county_forecast_perc_comparison(roi_75057, 750
2 denton_perc_comparison
```
executed in 7ms, finished 09:23:47 2021-06-20

Out[201]:

	Lower CI	Upper CI	Forecast
75057 Perc Change	-0.094770	0.170486	0.038316
Val	90523.018308	117048.600838	103831.553120
$ Diff	-9476.981692	17048.600838	3831.553120
75006 Perc Change	-0.070827	0.188189	0.058984
Val	92917.254038	118818.875744	105898.410932
$ Diff	-7082.745962	18818.875744	5898.410932
76201 Perc Change	0.061440	0.329790	0.196081
Val	106144.003676	132978.963820	119608.102825
$ Diff	6144.003676	32978.963820	19608.102825

- Based on the downside risk, upside return, and mean predicted value, 76201 seems like the superior zip code
- It has the highest upside, highest predicted mean, and lowest downside risk
- 75057 and 75006 are very similar
- **In Denton county, 76201 has the best prospects for near term growth**

5 Intepretation

- Observe current prices
- Analyze correlation between different zip codes subdivided by market type
- Use Annualized Rate of Return to determine which Zip Code will be eliminated

In [202]:
```
# test = pd.concat([df_78758, df_78210, df_77092, df_79903, df_75023, d
# test = test[[78758, 78210, 77092, 79903, 75023, 76201]]
# test
```

executed in 1ms, finished 09:23:47 2021-06-20

5.1 Where Prices Stand Today

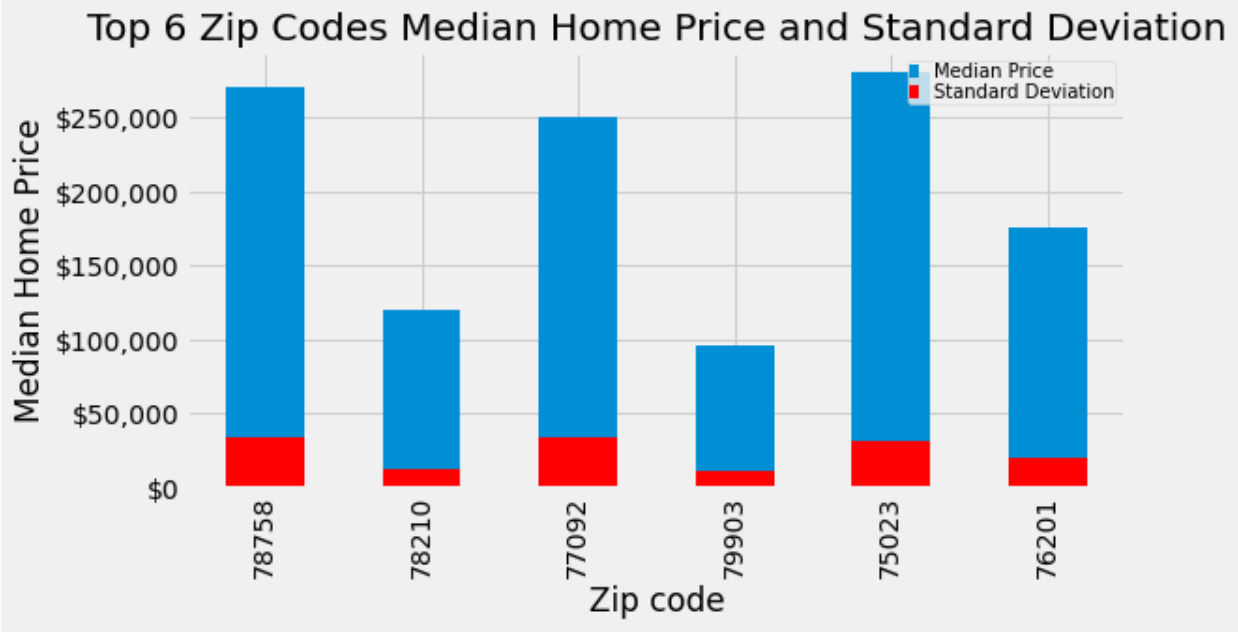
- **Primary Market**
  - 78758 (Travis - Austin)
  - 78210 (Bexar - San Antonio)
  - 77092 (Harris - Houston)
- **Secondary Market**
  - 79903 (El Paso - El Paso)
  - 75023 (Collin - DFW)
  - 76201 (Denton - DFW)

In [203]:
```
# Combine all 6 zip codes into one DataFrame

df_all = pd.concat([df_78758, df_78210, df_77092, df_79903, df_75023, d
df_all.columns = [78758, 78210, 77092, 79903, 75023, 76201]
```

executed in 3ms, finished 09:23:47 2021-06-20

In [204]:
```
with plt.style.context('fivethirtyeight'):
    fig, ax = plt.subplots(figsize=(8,4))
    fmt = '${x:,.0f}'
    tick = mtick.StrMethodFormatter(fmt)
    ax.yaxis.set_major_formatter(tick)

    df_all.iloc[-1].plot(kind='bar', ax=ax, label='Median Price',rot=18
    df_all.std().plot(kind='bar', ax=ax, color='red', label='Standard D
    ax.legend(handlelength=0.5, borderpad=0.1, labelspacing=.1, prop={'
    ax.set_xlabel('Zip code')
    ax.set_ylabel('Median Home Price')
    ax.set_title('Top 6 Zip Codes Median Home Price and Standard Deviat
```

executed in 110ms, finished 09:23:47 2021-06-20



- Entry point for 78758, 75023, and 77092 are all fairly close
  - Facotring in standard deviation, a home in each zip code could be acquired for the same asking price
- 76201 is in the middle
- 78210 and 79903 are faily close
  - Facotring in standard deviation, a home in each zip code could be acquired for the same asking price

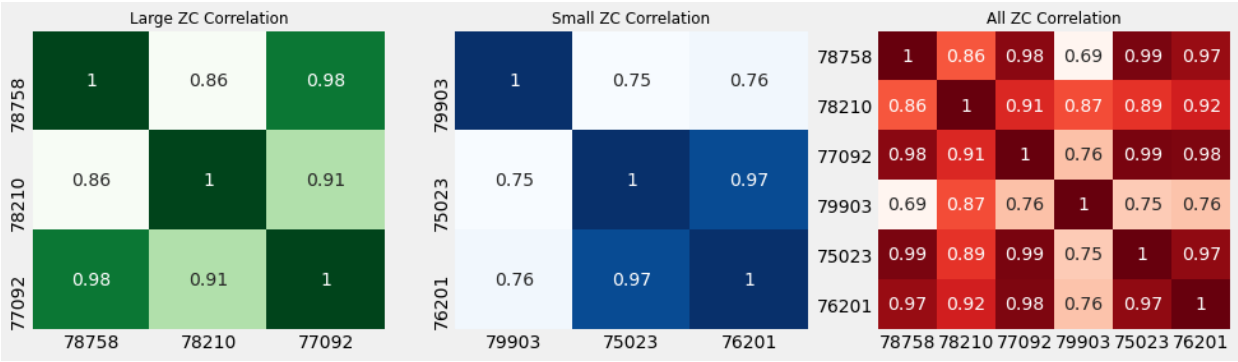
5.2 Correlation Amongst Various Zip Codes

- **Primary Market**
  - 78758 (Travis - Austin)
  - 78210 (Bexar - San Antonio)
  - 77092 (Harris - Houston)
- **Secondary Market**
  - 79903 (El Paso - El Paso)
  - 75023 (Collin - DFW)
  - 76201 (Denton - DFW)

In [205]:
```
test = pd.concat([df_78758, df_78210, df_77092, df_79903, df_75023, df_
test.columns =['78758', '78210', '77092', '79903', '75023', '76201']
# test = test[['78758', '78210', '77092', '79903', '75023', '76201']]
# test
```
executed in 3ms, finished 09:23:47 2021-06-20

In [206]:
```
# Divide DataFrame into primary and secondary markets

df_large_county = pd.concat([df_78758, df_78210, df_77092], axis=1)
df_large_county.columns = [78758, 78210, 77092]

df_small_county = pd.concat([df_79903, df_75023, df_76201], axis=1)
df_small_county.columns = [79903, 75023, 76201]
```
executed in 8ms, finished 09:23:47 2021-06-20

In [207]:
```
with plt.style.context('fivethirtyeight'):
    fig, axes = plt.subplots(figsize=(15,4), ncols=3)

    sns.heatmap(df_large_county.corr(), cmap='Greens', annot=True,ax=ax
    axes[0].set_title('Large ZC Correlation', fontsize=12);

    sns.heatmap(df_small_county.corr(), cmap='Blues', annot=True,ax=axe
    axes[1].set_title('Small ZC Correlation', fontsize=12);

    sns.heatmap(test.corr(), cmap='Reds', annot=True,ax=axes[2], cbar=F
    axes[2].set_title('All ZC Correlation', fontsize=12)
```
executed in 403ms, finished 09:23:47 2021-06-20



- Collin and Denton move very closely together which isn't surprising given they are both located in DFW
- Harris (Houston) and Travis (Austin) move very closely together
- Since one zip code needs to be eliminated, it makes the most sense to get rid of one from either of those pairs
- Going to tilt towards a primary or secondary market heavy portfolio
- Desirable to have uncorrelated zip codes because the portfolio will have more diversification
- Overall, there isn't a signficant correlation difference between primary and secondary markets

5.3 Compare Annualized Rate of Return

In [208]:
```
# Use zip code lengths to determine Annualized Rate of Returns

zip_lengths = []
top_zips = [forecast_df_78210, forecast_df_78758, forecast_df_77092,
            forecast_df_79903, forecast_df_75023, forecast_df_76201]
for zip_ in top_zips:
    zip_lengths.append(len(zip_))
zip_lengths = pd.Series(zip_lengths)

```
executed in 2ms, finished 09:23:47 2021-06-20

In [209]:
```
# Create DataFrame of top zip codes including predictions and confidenc

perc_val = pd.concat([bexar_perc_comparison[-3:], travis_perc_compariso
           el_paso_perc_comparison[-3:], collin_perc_comparison[-3:], d
```
executed in 4ms, finished 09:23:47 2021-06-20

In [210]:
```
# Prepare Annualized RoR by only separating out percent change rows

perc_top_zips = perc_val.iloc[range(0,16,3)]
perc_change = perc_top_zips.index.to_list()
perc_change = [x[0:6] for x in perc_change]
perc_top_zips.index = perc_change
perc_top_zips.reset_index(inplace=True)
apy = pd.concat([perc_top_zips,zip_lengths], axis=1)
apy.rename(columns={'index':'Zip'}, inplace=True)
apy.set_index('Zip', inplace=True)
apy.rename(columns={0:'Months'},inplace=True)

```
executed in 4ms, finished 09:23:47 2021-06-20

- **Primary Market**
  - 78758 (Travis - Austin)
  - 78210 (Bexar - San Antonio)
  - 77092 (Harris - Houston)
- **Secondary Market**
  - 79903 (El Paso - El Paso)
  - 75023 (Collin - DFW)
  - 76201 (Denton - DFW)

In [211]:
```
# https://smartasset.com/retirement/roi-return-on-investment
# Annualized RoR formula

apy['Annualized_RoR'] = ((apy['Forecast']+1)**(1/(apy['Months']/12))-1)
apy.sort_values(by='Annualized_RoR', ascending=False)
```

executed in 8ms, finished 09:23:47 2021-06-20

Out[211]:

Zip	Lower CI	Upper CI	Forecast	Months	Annualized_RoR
78210	0.094200	0.370074	0.232638	19	14.122104
76201	0.061440	0.329790	0.196081	24	9.365489
78758	0.075206	0.308513	0.192139	24	9.185114
75023	0.071910	0.257797	0.165036	27	7.024750
77092	-0.043216	0.316966	0.137368	27	5.887544
79903	-0.113847	0.142745	0.014994	27	0.663661

- Primay Market Order: 78210, 78758, 77092
- Secondary Market Order: 76201, 75023, 79903

Due to disparity between Annualized Rate of Return and preferred tilt towards quality market, will be selecting **77092 over 79903**

6 Implementation

- Compare predictions of all 5 zip codes
- Determine optimal month to buy and sell homes for each zip code
- Analyze prejected return on $500,000 investment beginning in May 2018

6.1 Top 5 Zip Code Forecasts

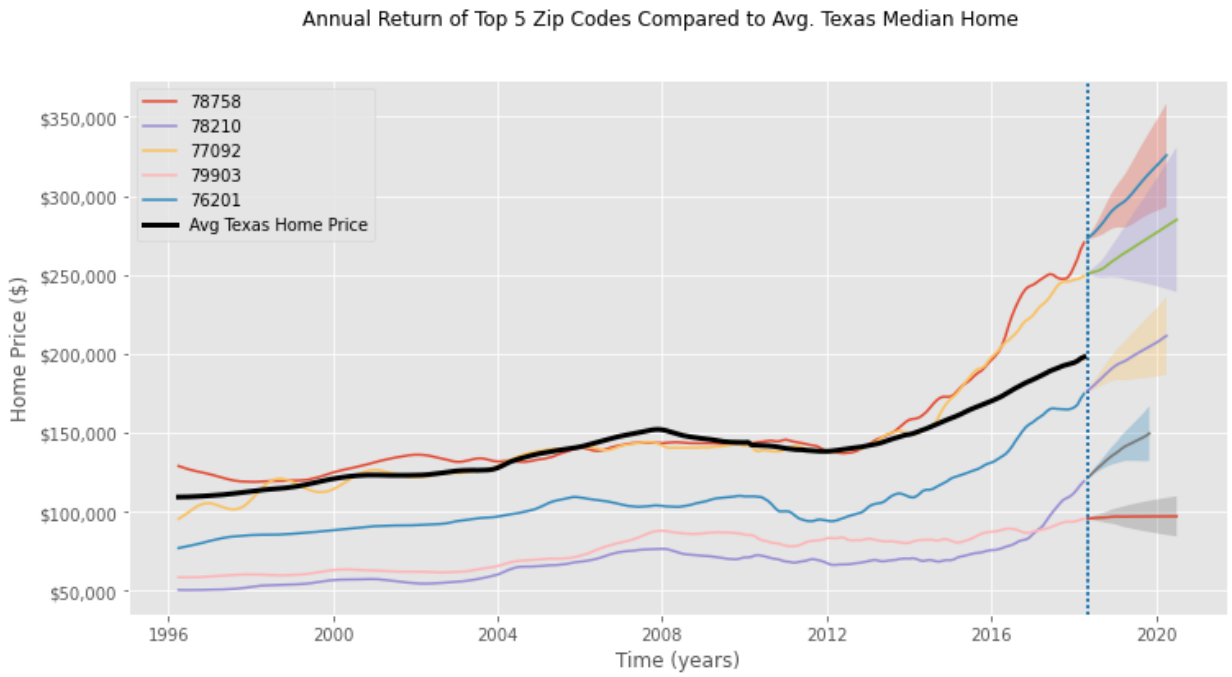
In [212]:
```
df_all_5 = df_all.iloc[:,[0,1,2,3,5]]
# df_all_5.plot()
```

executed in 2ms, finished 09:23:47 2021-06-20

In [213]:

```
with plt.style.context('ggplot'):

    fig, ax = plt.subplots(figsize=(12,6))

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

    texas_avg = df_texas.groupby('time').mean()
    texas_avg

    cols = df_all_5.columns.to_list()
    forecasts = [forecast_df_78758, forecast_df_78210, forecast_df_7709

    for col, forecast in zip(cols, forecasts):
        ax.plot(df_all_5[col], label=col)
        ax.plot(forecast['Forecast'])
        ax.fill_between(forecast.index, forecast['Lower CI'], forecast[
        ax.axvline(forecast.index[0], ls=':')

    ax.plot(texas_avg['value'], color='k', label='Avg Texas Home Price'
    ax.set_xlabel('Time (years)')
    ax.set_ylabel('Home Price ($)')
    fig.suptitle('Annual Return of Top 5 Zip Codes Compared to Avg. Tex
    ax.legend();
```

executed in 183ms, finished 09:23:47 2021-06-20

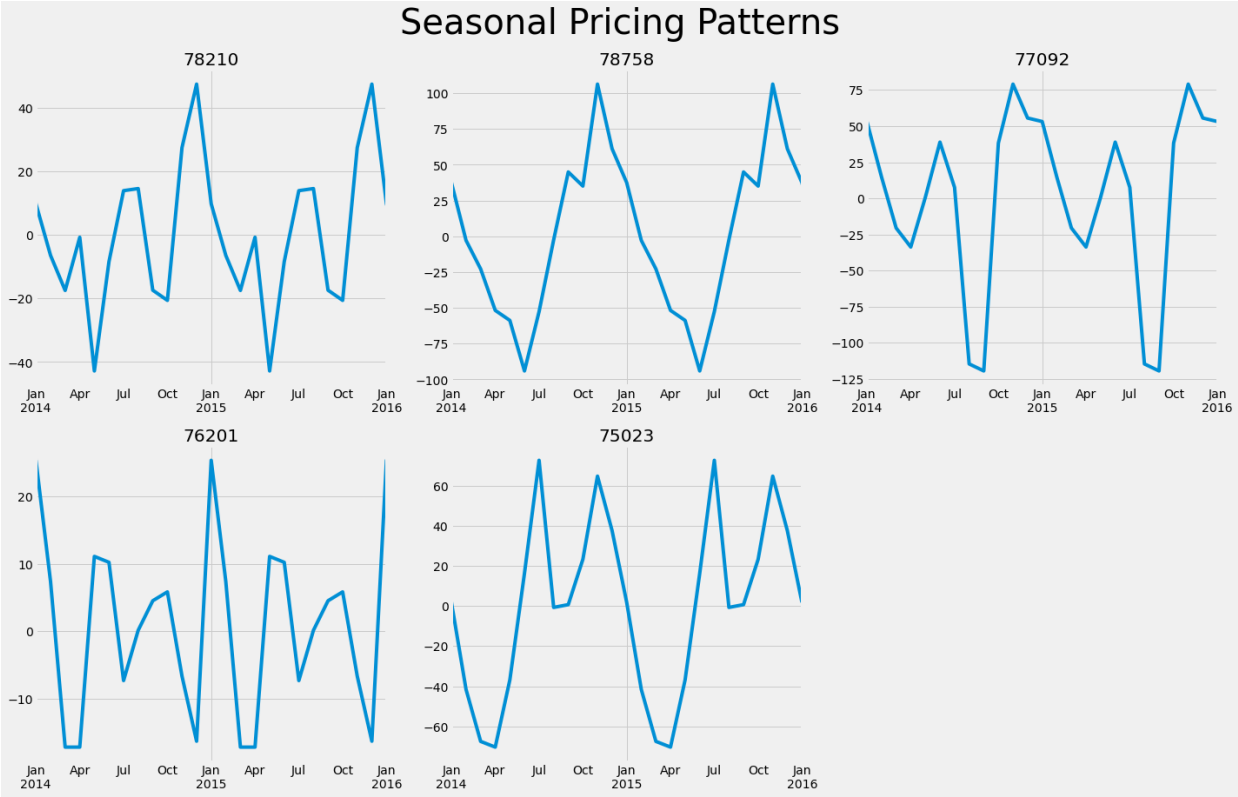


- Most of the top 5 zip codes have a greater upward trend than the average Texas home
- 79903 has lower returns but is included to diversify accross metros

6.2 Determing Optimal Time to Buy Based on Seasonality

In [214]:
```
with plt.style.context('fivethirtyeight'):
    fig, axes = plt.subplots(figsize=(20,13), nrows=2, ncols=3)
    training_models = [train_78210, train_78758, train_77092, train_762
    zips_ = [78210, 78758, 77092, 76201, 75023]
    fig.suptitle('Seasonal Pricing Patterns', fontsize=40)

    for ax, train, zips_ in zip(axes.flatten(), training_models, zips_)
        decomposition = seasonal_decompose(train,model='Additive')
        seasonal = decomposition.seasonal
        seasonal.plot(xlim=('2014-01-01', '2016-01-01'), ax=ax)
        ax.set_title(zips_)
        ax.set_xlabel('')
        fig.tight_layout()
    fig.delaxes(axes[1,2]);
```

executed in 784ms, finished 09:23:48 2021-06-20




```
- **78210:** Buy in April, sell in December
- **78758:** Buy in July, sell in November
- **77092:** Buy in July, sell in October
- **76201:** Buy in April, sell in January
- **75023:** Buy in April, sell in July
```

6.3 Projected Return

In [217]:
```
# Top zip codes

top_zips_forecast = [forecast_df_78210['Forecast'], forecast_df_78758['
            forecast_df_75023['Forecast'], forecast_df_76201['Forecast'
```
executed in 2ms, finished 09:23:48 2021-06-20

In [218]:
```
test = pd.concat(top_zips_forecast, axis=1)

test.columns = [1,2,3,4,5]
test['mean']= test.mean(axis=1)
test.head(2)
```
executed in 5ms, finished 09:23:48 2021-06-20

In [220]:
```
top_zips_upper = [forecast_df_78210['Upper CI'], forecast_df_78758['Upp
            forecast_df_75023['Upper CI'], forecast_df_76201['Upper CI'
upper = pd.concat(top_zips_upper, axis=1)
upper['mean']= upper.mean(axis=1)
upper.head(2)
```
executed in 9ms, finished 09:23:48 2021-06-20

Out[220]:

	Upper CI	Upper CI	Upper CI	Upper CI	Upper CI	mean
2018-05-01	121820.137341	273811.575689	251208.309081	281996.709042	177382.576451	221243.861521
2018-06-01	124615.731587	276605.302216	253151.449730	284156.993357	179986.443124	223703.184003

In [223]:
```
top_zips_lower = [forecast_df_78210['Lower CI'], forecast_df_78758['Low
            forecast_df_75023['Lower CI'], forecast_df_76201['Lower CI'
lower = pd.concat(top_zips_lower, axis=1)
lower['mean']= lower.mean(axis=1)
lower.head(2)
```
executed in 257ms, finished 11:39:35 2021-06-20

Out[223]:

	Lower CI	Lower CI	Lower CI	Lower CI	Lower CI	mean
2018-05-01	120938.252443	272502.007084	249836.862652	280888.657455	176154.211770	220063.998281
2018-06-01	122419.562629	273019.965192	249545.841021	281303.470003	176852.082006	220628.184170

In [224]:
```
top_5_ci = pd.concat([test['mean'], upper['mean'], lower['mean']], axis
top_5_ci.columns = ['mean', 'upper', 'lower']
top_5_ci['mean_pct'] = top_5_ci['mean'].pct_change(1)
500000/220653
len(top_5_ci)
```
executed in 32ms, finished 11:39:36 2021-06-20

Out[224]: 27

In [236]:

```
mean_mult = 500000/220653
upper_mult = 500000/221243
lower_mult = 500000/220063
top_5_ci['500_mean'] = top_5_ci['mean']*2.266001368664827
top_5_ci['500_lower'] = top_5_ci['lower']*lower_mult
top_5_ci['500_upper'] = top_5_ci['upper']*upper_mult
top_5_ci.head(2)
```

executed in 26ms, finished 12:02:52 2021-06-20

Out[236]:

	mean	upper	lower	mean_pct	500_mean	500_lower	
2018-05-01	220653.929901	221243.861521	220063.998281	NaN	500002.107156	500002.268170	500(
2018-06-01	222165.684086	223703.184003	220628.184170	0.006851	503427.744210	501284.141746	505

In [235]:

```
# Add points to graph

scatter = top_5_ci.iloc[range(0,27, 6)]
scatter
```

executed in 37ms, finished 12:02:47 2021-06-20

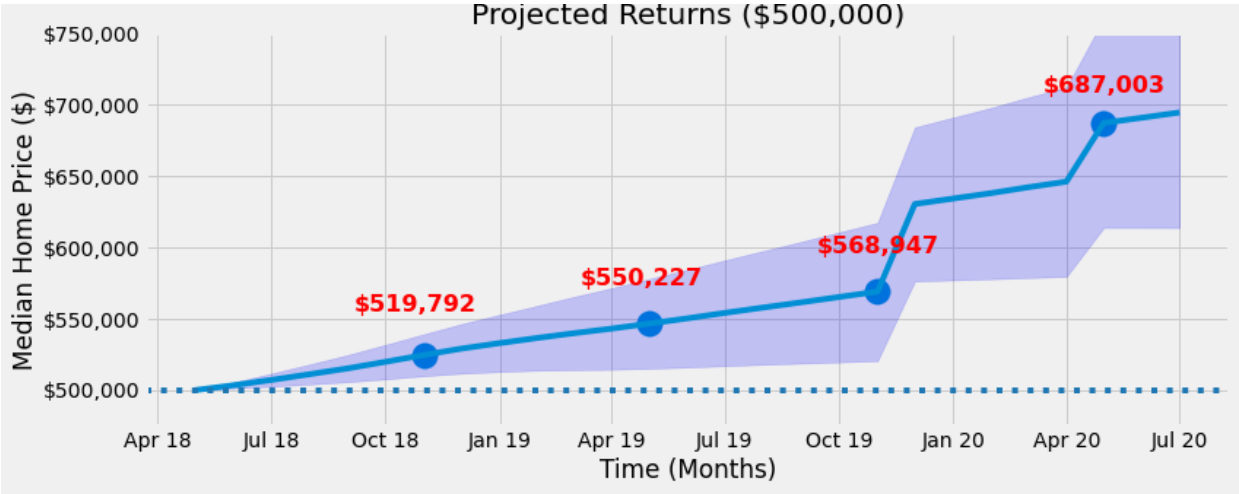
Out[235]:

	mean	upper	lower	mean_pct	500_mean	500_lower	
2018-05-01	220653.929901	221243.861521	220063.998281	NaN	500002.107156	500002.268170	500(
2018-11-01	231505.136121	238602.064856	224408.207386	0.009231	524590.955303	509872.644166	539
2019-05-01	241185.830017	255676.044773	226695.615262	0.006418	546527.420922	515069.810150	577
2019-11-01	251079.952709	273108.366433	229051.538985	0.006689	568947.516483	520422.649390	617
2020-05-01	303178.656533	336273.575878	270083.737188	0.063607	687003.250654	613650.948111	759

In [228]:

```
import matplotlib.dates as mdates
import datetime as dt




with plt.style.context('fivethirtyeight'):
    fig, ax = plt.subplots(figsize=(12,5))
    fmt = '${x:,.0f}'
    tick = mtick.StrMethodFormatter(fmt)
    ax.yaxis.set_major_formatter(tick)

    ax.plot(top_5_ci['500_mean'])
    ax.scatter(x=scatter.index[1:], y=scatter.iloc[1:]['500_mean'], s=3
    ax.fill_between(top_5_ci.index, top_5_ci['500_lower'], top_5_ci['50


    formatter = mdates.DateFormatter('%b %y')
    ax.xaxis.set_major_formatter(formatter)

    x = [dt.datetime(2018, 10, 25), dt.datetime(2019, 4, 25),
         dt.datetime(2019, 11, 1), dt.datetime(2020, 5, 1)]
    y = [559792, 578227, 600257.611795, 713003]

#     ax.annotate('Mean Value: $222,165', (mdates.date2num(x[1]), y[1])
#             textcoords='offset points', arrowprops=dict(arrowstyle='-
#     ax.text(x=x[1], y=[1], s=)
    plt.text(x[0], y[0], '$519,792', size=16, ha='center', va='center',
    plt.text(x[1], y[1], '$550,227', size=16, ha='center', va='center',
    plt.text(x[2], y[2], '$568,947', size=16, ha='center', va='center',
    plt.text(x[3], y[3], '$687,003', size=16, ha='center', va='center',

    ax.axhline(500000, ls=':')

    ax.set_ylim(475000, 750000)
    ax.set_xlabel('Time (Months)')
    ax.set_ylabel('Median Home Price ($)')
    ax.set_title('Projected Returns ($500,000)')
    fig.tight_layout()
```

executed in 246ms, finished 11:39:41 2021-06-20



- Various returns based on different hold periods
- Holding until at least May makes a significant difference on overall return
  - Highest marginal return in Nov 2019 and Apr 2020
- $500,000 invested in May 2018 has a project value of $687,000 in May 2020
  - 1 year hold - 15% return (15% Annualized RoR)
  - 2 year hold - 37% return (18.5% Annualized RoR)

7 Testing Functions

In [ ]:

```
with plt.style.context('fivethirtyeight'):
    fig, ax = plt.subplots(figsize=(12,5))

    fmt = '${x:,.0f}'
    tick = mtick.StrMethodFormatter(fmt)
    ax.yaxis.set_major_formatter(tick)
    # 78758
    ax.plot(df_78758['2006-01-01':], label='78758')
    ax.plot(forecast_df_78758['Forecast'], label='78758 Forecast')
    ax.fill_between(forecast_df.index,forecast_df_78758['Lower CI'], fo
                    color='green')
    ax.axvline(df_78758.index[-1], ls =':')

    #78721
    ax.plot(df_78721['2006-01-01':], label='78721')
    ax.plot(forecast_df_78721['Forecast'], label='78721 Forecast')
    ax.fill_between(forecast_df.index,forecast_df_78721['Lower CI'], fo
                    color='red')

    #78744
    ax.plot(df_78744['2006-01-01':], label='78744')
    ax.plot(forecast_df_78744['Forecast'], label='78744 Forecast')
    ax.fill_between(forecast_df.index,forecast_df_78744['Lower CI'], fo
                    color='blue')

    ax.legend(loc=2)
```

executed in 10m 32s, finished 09:23:48 2021-06-20

In [ ]:
```
1
```

In [ ]:
```
1
```

In [ ]:
```
1 summary, ARMA_order, seasonal_order=find_auto_order(train_78210)
2 summary
```
executed in 10m 32s, finished 09:23:48 2021-06-20

In [ ]:
```
1 scatter['500_mean']
```
executed in 10m 32s, finished 09:23:48 2021-06-20

In [ ]:
```
1 seasonal_order
```
executed in 10m 32s, finished 09:23:48 2021-06-20

In [ ]:
```
1 model = SARIMAX(train_78210,order=(1, 2, 2), seasonal_order=(2, 2, 2, 1
2                         enforce_stationarity=False, enforce_invertibili
3                         freq='MS')
4
```
executed in 10m 32s, finished 09:23:48 2021-06-20

In [ ]:
```
1 res = model.fit(maxiter=200)
2 res.summary()
```
executed in 10m 32s, finished 09:23:48 2021-06-20

In [ ]:
```
1 fore78210 = res.get_forecast(steps=len(test_78210))
```
executed in 10m 32s, finished 09:23:48 2021-06-20

In [ ]:
```
1 # save forecasted mean and upper/lower ci as df
2 forecast_df = fore78210.conf_int()
3 forecast_df.columns = ['Lower CI','Upper CI']
4 forecast_df['Forecast'] = fore78210.predicted_mean
5 forecast_df
```
executed in 10m 32s, finished 09:23:48 2021-06-20

In [ ]:
```
1 train_78210.plot()
2 test_78210.plot()
3 forecast_df['Forecast'].plot()
```
executed in 10m 32s, finished 09:23:48 2021-06-20

Trying another gridsearch method

In [ ]:
```
import itertools
# Define the p, d and q parameters to take any value between 0 and 2
p = d = q = range(0,3)

# Generate all different combinations of p, q and q triplets
pdq = list(itertools.product(p,d,q))

# Generate all different combinations of seasonal p, q and q triplets (
pdqs = [(x[0], x[1], x[2], 12) for x in list(itertools.product(p, d, q)
```
executed in 10m 32s, finished 09:23:48 2021-06-20

In [ ]:
```
cols

```
executed in 10m 32s, finished 09:23:48 2021-06-20

In [ ]:
```
forecast_df_78758
```
executed in 10m 32s, finished 09:23:48 2021-06-20

In [ ]:
```
aic_scores = []
for comb in pdq:
    for combs in pdqs:
        try:
            mod = SARIMAX(train_78210, order=comb, seasonal_order=combs
            output = mod.fit(maxiter=100)
            aic_scores.append([comb, combs, output.aic])
        except:
            contiue
aic_scores
```
executed in 10m 32s, finished 09:23:48 2021-06-20

In [ ]:
```
pd.DataFrame(aic_scores).sort_values(2)
```
executed in 10m 32s, finished 09:23:48 2021-06-20

In [ ]:

```
def SARIMAX_man_results(train_data, arma_order, seasonal_order):
    """
    Takes a training data and ARMA/Seasonal order and fits a SARIMAX mo
    different parameters and returns the best SARIMAX model based on AI
    Parameters:
    Train data, seasonal/arma order
    Returns:
    Dictionary with various iterations of parameters and their AIC scor
    parameters
    """
    grid_results = []
    grid_tests = []
    mle_regression=[True,False]
    concentrate_scale = [True, False]
    results_dict = {}
    best_params = {}

    for mle in mle_regression:
        for scale in concentrate_scale:
            model = SARIMAX(train_data,order=arma_order, seasonal_order
                        enforce_stationarity=False, enforce_invertibili
                        mle_regression=mle, concentrate_scale=scale,fre
            results = model.fit()
            score_ = results.aic

            grid_tests.append([mle, scale])
            grid_results.append(score_)
    df = pd.DataFrame(grid_tests, grid_results).sort_index()
    df = df.rename(columns={0: 'MLE_Regression', 1: 'Concentrate_Scale'
    results_dict['grid_search'] = df
    diagnostics = results.plot_diagnostics(figsize=(8,8))
    results_dict['diag_summary'] = diagnostics

    best_params['arma_order'] = arma_order
    best_params['seasonal_order'] = seasonal_order
    best_params['MLE_Regression'] = seasonal_order
    best_params['Concentrate_Scale'] = seasonal_order


    return results_dict, best_params
```

executed in 10m 32s, finished 09:23:48 2021-06-20

In [ ]:

```
fig, ax = plt.subplots()
for col, forecast in zip(cols, forecasts):
    ax.plot(df_all_5[col], label='col')
```

executed in 10m 32s, finished 09:23:48 2021-06-20

In [ ]:

```
def SARIMAX_man_results(train_data, arma_order, seasonal_order):
    """
    Takes a training data and ARMA/Seasonal order and fits a SARIMAX mo
    different parameters and returns the best SARIMAX model based on AI
    Parameters:
    Train data, seasonal/arma order
    Returns:
    Dictionary with various iterations of parameters and their AIC scor
    parameters
    """
    grid_results = []
    grid_tests = []
    mle_regression=[True,False]
    concentrate_scale = [True, False]
    results_dict = {}
    best_params = {}

    for mle in mle_regression:
        for scale in concentrate_scale:
            model = SARIMAX(train_data,order=arma_order, seasonal_order
                        enforce_stationarity=False, enforce_invertibili
                        mle_regression=mle, concentrate_scale=scale,fre
            results = model.fit()
            score_ = results.aic

            grid_tests.append([mle, scale])
            grid_results.append(score_)
    df = pd.DataFrame(grid_tests, grid_results).sort_index()
    df = df.rename(columns={0: 'MLE_Regression', 1: 'Concentrate_Scale'
    results_dict['grid_search'] = df
    diagnostics = results.plot_diagnostics(figsize=(8,8))
    results_dict['diag_summary'] = diagnostics

    best_params['arma_order'] = arma_order
    best_params['seasonal_order'] = seasonal_order
    best_params['MLE_Regression'] = seasonal_order
    best_params['Concentrate_Scale'] = seasonal_order


    return results_dict, best_params
```

executed in 10m 32s, finished 09:23:48 2021-06-20

In [ ]:
```
import matplotlib.ticker as mtick

def model_predictions(train_data, test_data, all_data, code, county):
    """
    Uses all helper functions to create diagnostic summary, forecast on
    Parameters:
    Training data, test data, all data, zip code, and county
    Returns:
    Plot showing diagnostics and forecasts. Dictinary with predictions
    """
    summary, arma_order, seasonal_order = find_auto_order(train_data)
    best_params = SARIMAX_man_results(train_data, arma_order, seasonal_
    best_model = fit_final_model(train_data, arma_order, seasonal_order
                                 best_params['MLE_Regression'], best_pa
    forecast_df=get_forecast(best_model, test_data)
    test_forecast_fig=plot_forecast(train_data, test_data, forecast_df,
    bestall_model=fitall_final_model(all_data, arma_order, seasonal_ord
                         best_params['MLE_Regression'], best_params['Con

    fig, forecast_dict, forecast_df=plot_future_forecast(bestall_model,
    roi_helper = roi_dict(forecast_df)
    return fig, forecast_dict, forecast_df, roi_helper
```
executed in 10m 32s, finished 09:23:48 2021-06-20

In [ ]:
```
bexar_perc_comparison = county_forecast_perc_comparison(roi_77092, 7709
bexar_perc_comparison
```
executed in 10m 32s, finished 09:23:48 2021-06-20

In [ ]:
```
# def plot_top3_5(all_perc_change_dict, county_dict_full, region):

#     top_5=list(all_perc_change_dict[region].index)[:5]
#     with plt.style.context('fivethirtyeight'):
#         fig, ax = plt.subplots(figsize=(12,5))
#         ax.plot(county_dict_full[top_5[0]]['value'], color='red', lab
#         ax.plot(county_dict_full[top_5[1]]['value'], color='blue', la
#         ax.plot(county_dict_full[top_5[2]]['value'], color='green', l
#         ax.plot(county_dict_full[top_5[3]]['value'], color='black', l
#         ax.plot(county_dict_full[top_5[4]]['value'], color='silver',
#         ax.set_title(f"{region} Top 5 Growth Zip Codes")
#         ax.set_ylabel('Home Price')
#         ax.set_xlabel('Year')
#         ax.legend()
#         return fig
```
executed in 10m 32s, finished 09:23:48 2021-06-20

8 Appendix

- FB Prophet

In [ ]:
```
import seaborn as sns
import matplotlib.pyplot as plt

with plt.style.context('fivethirtyeight'):
    fig, axes = plt.subplots(figsize=(15,8), nrows=2)
    fig.suptitle('Days Until Median Home Price Reached Pre \'08 Home Pr
    sns.kdeplot(data=df_Harris_days, ax=axes[0],label='Harris', palette
    sns.kdeplot(data=df_Dallas_days, ax=axes[0],label='Dallas', palette
    sns.kdeplot(data=df_Tarrant_days, ax=axes[0],label='Tarrant', palet
    sns.kdeplot(data=df_Bexar_days, ax=axes[0],label='Bexar', palette=[
    sns.kdeplot(data=df_Travis_days, ax=axes[0],label='Travis', palette
    axes[0].set_xlabel('Days Until Price Recovery')
    axes[0].set_title('Major Counties by Population')
    sns.kdeplot(data=df_Denton_days, ax=axes[1],label='Denton', palette
    sns.kdeplot(data=df_Collin_days, ax=axes[1],label='Collin', palette
    sns.kdeplot(data=df_El_Paso_days, ax=axes[1],label='El Paso', palet
    sns.kdeplot(data=df_Montgomery_days, ax=axes[1],label='Montgomery',
    sns.kdeplot(data=df_McLennan_days, ax=axes[1],label='McLennan', pal
    axes[1].set_xlabel('Days Until Price Recovery')
    axes[1].set_title('Minor Counties by Population')
    axes[0].set_xlim(left=0)
    axes[1].set_xlim(left=0)
    axes[0].legend(prop={'size': 17})
    axes[1].legend(prop={'size': 17})
    fig.tight_layout();
```
executed in 10m 32s, finished 09:23:48 2021-06-20

In [ ]:
```
harris_perc_comparison[:3]
```
executed in 10m 32s, finished 09:23:48 2021-06-20

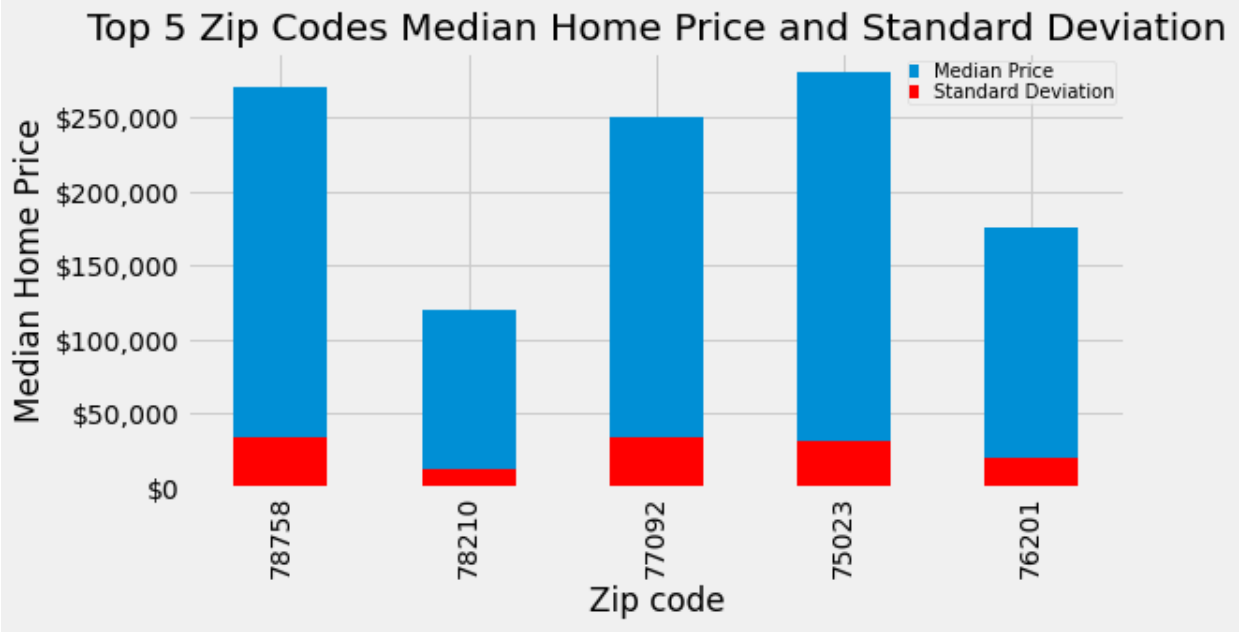
In [ ]:
```
with plt.style.context('fivethirtyeight'):

    fig, ax = plt.subplots(figsize=(8,4))
    fmt = '${x:,.0f}'
    tick = mtick.StrMethodFormatter(fmt)d
    ax.yaxis.set_major_formatter(tick)

    ax.bar(height=test.iloc[-1], x=test.columns, label='Median Price')
#     ax.bar(height=test.std(), x =test.columns, label='Standard Deviat
#     ax.legend(handlelength=0.5, borderpad=0.1, labelspacing=.1, prop=
```
executed in 10m 32s, finished 09:23:48 2021-06-20

In [232]:
```
df_all2 = pd.concat([df_78758, df_78210, df_77092, df_75023, df_76201],
df_all2.columns = [78758, 78210, 77092, 75023, 76201]
```
executed in 10ms, finished 11:42:39 2021-06-20

In [234]:

```
with plt.style.context('fivethirtyeight'):
    fig, ax = plt.subplots(figsize=(8,4))
    fmt = '${x:,.0f}'
    tick = mtick.StrMethodFormatter(fmt)
    ax.yaxis.set_major_formatter(tick)

    df_all2.iloc[-1].plot(kind='bar', ax=ax, label='Median Price',rot=1
    df_all2.std().plot(kind='bar', ax=ax, color='red', label='Standard
    ax.legend(handlelength=0.5, borderpad=0.1, labelspacing=.1, prop={'
    ax.set_xlabel('Zip code')
    ax.set_ylabel('Median Home Price')
    ax.set_title('Top 5 Zip Codes Median Home Price and Standard Deviat
```

executed in 133ms, finished 11:42:46 2021-06-20



In [ ]:

```
1
```