Key factors that influence US home prices

In economics, demand and supply play crucial roles in determining the prices of commodities.

In this context, examining the dynamics of demand and supply will help us understand the extent to which various factors influence U.S. home prices.

Target:

S&P Case-Schiller Home Price Index (CSUSHPISA) :

These indices represent the aggregate change in home prices across 10 and 20 major U.S. metropolitan areas.
Frequency: Monthly
Unit: Percentage change from a base value of 100 set in 2000.

Factors:

1. Federal Funds Effective Rate (DFF):

The federal funds rate is the central interest rate in the U.S. financial market. It influences other interest rates such as the prime rate, which is the rate banks charge their customers with higher credit ratings. Additionally, the federal funds rate indirectly influences longer-term interest rates such as mortgages, loans, and savings, all of which are very important to consumer wealth and confidence.
Frequency: Monthly
Unit: Percent

2. Gross Domestic Product (GDP):

The featured measure of U.S. output, is the market value of the goods and services produced by labor and property located in the United States.
Frequency: Quarterly
Unit: Billions of Dollars

3. Median Household Income in the United States (MEHOINUSA672N):

The "Median Household Income" in the United States represents the middle point of all household incomes, separating the higher and lower halves, and is a key indicator of the economic well-being of the average American household.
Freuqency: Annually
Unit: Dollars

4. Housing Inventory Estimate: Vacant Housing Units in the United States (EVACANTUSQ176N):

The Housing Inventory Estimate for vacant housing units refers to the estimated number of residential properties that are unoccupied.
Frequency: Quarterly
Unit: Thousands of Units

5. Housing Inventory Estimate: Total Housing Units in the United States (ETOTALUSQ176N):

The total housing units in the United States refer to the estimated number of places designed for people to live.The count includes both occupied and vacant units.
Frequency: Quarterly
Unit: Thousands of Units

6. Consumer Price Index for All Urban Consumers: All Items in U.S. City Average (CPIAUCSL):

The CPI can be used to recognize periods of inflation and deflation. Significant increases in the CPI within a short time frame might indicate a period of inflation, and significant decreases in CPI within a short time frame might indicate a period of deflation.
Frequency: Monthly
Unit: Percentage change from a base value of 100 set in 1982-1984

7. Monthly Supply of New Houses in the United States (MSACSR):

The months' supply is the ratio of new houses for sale to new houses sold. This statistic provides an indication of the size of the new for-sale inventory in relation to the number of new houses currently being sold. The months' supply indicates how long the current new for-sale inventory would last given the current sales rate if no additional new houses were built.
Frequency: Monthly
Units: Months' Supply

8. 15-Year Fixed Rate Mortgage Average in the United States (MORTGAGE15US):

A 15-year fixed-rate mortgage is a type of home loan where the interest rate remains constant for the entire 15-year term.
Frequency: Monthly
Unit: Percent

9. 30-Year Fixed Rate Mortgage Average in the United States (MORTGAGE30US):

A 30-year fixed-rate mortgage (FRM) is a type of home loan where the interest rate remains constant for the entire 30-year term.
Frequency: Monthly
Unit: Percent

10. New Privately-Owned Housing Units Authorized in Permit-Issuing Places: Total Units (PERMIT):

It refers to the total number of new housing units for which building permits have been authorized in permit-issuing places. This statistic is often used as an indicator of the level of new residential construction activity in a given area.
Frequency: Monthly
Units: Thousands of Units

11. Total Population: All Ages including Armed Forces Overseas (POP):

Population includes resident population plus armed forces overseas. The monthly estimate is the average of estimates for the first of the month and the first of the following month.

Frequency: Monthly
Unit: Thousands

12. Total Construction Spending: Residential in the United States (TLRESCONS):

Total Construction Spending: Residential refers to the total value of spending on residential construction projects in the United States. This includes the expenditures on the construction, renovation, and improvement of residential structures, such as single-family homes, multi-family buildings, and residential developments.
Frequency: Monthly
Unit: Millions of Dollars

13. New Privately-Owned Housing Units Started: Total Units (HOUST):

It is is a housing market indicator that measures the total number of new residential construction projects initiated, providing insights into the overall health and activity in the housing sector.
Frequency: Monthly
Unit: Thousands of Units

14. Unemployment Rate (UNRATE):

The unemployment rate represents the number of unemployed as a percentage of the labor force. Labor force data are restricted to people 16 years of age and older, who currently reside in 1 of the 50 states or the District of Columbia, who do not reside in institutions (e.g., penal and mental facilities, homes for the aged), and who are not on active duty in the Armed Forces.
Frequency: Monthly
Unit: Percent

15. University of Michigan: Consumer Sentiment (UMCSENT):

It is an economic indicator that measures the confidence and optimism of consumers regarding the overall state of the U.S. economy.
Frequency: Monthly
Unit: Percentage change from a base value of 100 set in 1966.

All data collected for this project is sourced exclusively from https://fred.stlouisfed.org.

Here are the features categorized by supply and demand:

Supply Indicators:

---PERMIT (Housing Units Authorized by Building Permits):-
the potential future supply of houses

---TLRESCONS (Total Construction residential Spendin):
the level of investment in constructoin, impacting housing supply

---HOUST (New Privately-Owned Housing Units Started):-
the initiation of new housing construction

---EVACANTUSQ176N (Homeowner Vacancy Rate):-
the availablility of vacant homes impacting supply

---ETOTALUSQ176N (Total Housing Inventory):-
Indicaets the total supply of housing available in the market.

Demand Indicators:

---DFF (Effective Federal Funds Rate):
Can influence mortgage interest rates therfore demand

---CPIAUCSL (Consumer Price Index for All Urban Consumers):
Reflects inflation which can impact purchasing power and housing affordability

---MSASCR (Homeownership Rate):
homes that are owner-occupied, reflecting demand for homeownership

---MORTGAGE15US and MORTGAGE30US (15-Year and 30-Year Fixed Mortgage Rates):
Influence the cost of borrowing affecting demand

---POPTHM (Ttoal Population):
Affects overall housnig demand

---UNRATE (Unemployment Rate):
Can impact household income henceforth housnig demand

---UMCSENT (University of Michigan Consumer Sentiment Index):
Reflects consumer confidence

---GDP (Gross Domestic Product):-
Reflects overall economic health which impacts housing demand

---MEHOINUSA672N (Median Household Income in the United States):-
Affects housing affordability and demand.

In [1]:
```
#importing necessary libraries
import pandas as pd
import numpy as np
import warnings
warnings.filterwarnings('ignore')
import matplotlib.pyplot as plt
import seaborn as sns
from plotnine import ggplot, aes, geom_line, labs, theme_minimal
import plotly.express as px

from sklearn.preprocessing import MinMaxScaler

#VIF from statsmodel
from statsmodels.stats.outliers_influence import variance_inflation_factor

#importing modules from scikit learn
from sklearn.model_selection import train_test_split
from sklearn.feature_selection import RFE
```

```
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import cross_val_score
from sklearn.linear_model import LinearRegression,Lasso,Ridge
```

In [2]:
```
#reading the different csv files and making them into dataframe
df1 = pd.read_csv('D:/data for notebook/CSUSHPISA.csv')
df2 = pd.read_csv('D:/data for notebook/Federal Funds Effective Rate.csv')
df3 = pd.read_csv('D:/data for notebook/GDP.csv')
df4 = pd.read_csv('D:/data for notebook/houshold income median.csv')
df5 = pd.read_csv('D:/data for notebook/Housing inventory estimates vacant (quarterly).csv')
df6 = pd.read_csv('D:/data for notebook/Housing total inventory estimates (quarterly).csv')
df7 = pd.read_csv('D:/data for notebook/Infaltion (cpi urban).csv')
df10= pd.read_csv('D:/data for notebook/Monthly Supply of New Houses.csv')
df11= pd.read_csv('D:/data for notebook/MORTGAGE15US.csv')
df12= pd.read_csv('D:/data for notebook/MORTGAGE30US.csv')
df13= pd.read_csv('D:/data for notebook/New Housing Units Authorized Permit.csv')
df14= pd.read_csv('D:/data for notebook/POP.csv')
df16= pd.read_csv('D:/data for notebook/Total Construction Spending Residential.csv')
df17= pd.read_csv('D:/data for notebook/Total New Housing Started.csv')
df18= pd.read_csv('D:/data for notebook/unemployement rate.csv')
df19= pd.read_csv('D:/data for notebook/University of Michigan Consumer Sentiment.csv')
```

In [3]:
```
# List of DataFrames representing monthly data
df_monthly=[df1,df2,df7,df10,df11,df12,df13,df14,df16,df17,df18,df19]

# List of DataFrames representing non-monthly data
df_not_monthly=[df3,df4,df5,df6]
```

In [4]:
```
# Looping each DataFrame in the df_monthly list and convert the 'DATE' column to datetime format
for df in df_monthly:
    df['DATE'] = pd.to_datetime(df['DATE'], format='%d-%m-%Y', errors='coerce')
```

In [5]:
```
#checking the Datatype after conversion
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 285 entries, 0 to 284
Data columns (total 2 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   DATE       285 non-null    datetime64[ns]
 1   CSUSHPISA  285 non-null    float64
dtypes: datetime64[ns](1), float64(1)
memory usage: 4.6 KB
```

In [6]:
```
#Merging all the DataFrames which contains mothnly data
merged_df = df1

# Loop through the rest of the DataFrames and merge
for df in df_monthly[1:]:
    merged_df = pd.merge(merged_df, df, on='DATE', how='outer')
```

In [7]:
```
#Examining the DataFrame
merged_df
```

Out[7]:

	DATE	CSUSHPISA	DFF	CPIAUCSL	MSACSR	MORTGAGE15US	MORTGAGE30US	PERMIT	POPTHM	TLRESCONS	HOUST	UNRATE	UMCSENT
0	2000-01-01	100.551	5.448387	169.300	4.3	7.8025	8.2100	1727	281083	NaN	1636	4.0	112.0
1	2000-02-01	101.339	5.734828	170.000	4.3	7.9325	8.3250	1692	281299	NaN	1737	4.1	111.3
2	2000-03-01	102.127	5.853548	171.000	4.3	7.8320	8.2400	1651	281531	NaN	1604	4.0	107.1
3	2000-04-01	102.922	6.019667	170.900	4.4	7.8000	8.1525	1597	281763	NaN	1626	3.8	109.2
4	2000-05-01	103.678	6.268065	171.200	4.4	8.1825	8.5150	1543	281996	NaN	1575	4.0	110.7
...	...	...	...	...	...	...	...	...	...	...	...	...	...
280	2023-05-01	302.566	5.055806	303.294	7.2	5.8075	6.4250	1496	335013	864027.0	1583	3.7	59.0
281	2023-06-01	304.593	5.076333	303.841	7.5	6.0880	6.7140	1441	335163	870655.0	1418	3.6	64.2
282	2023-07-01	306.767	5.120000	304.348	7.1	6.1775	6.8400	1443	335329	865747.0	1451	3.5	71.5
283	2023-08-01	309.155	5.330000	306.269	7.8	6.4300	7.0720	1541	335501	885776.0	1305	3.8	69.4
284	2023-09-01	311.175	5.330000	307.481	7.2	6.5725	7.2000	1471	335675	884184.0	1346	3.8	67.9

285 rows × 13 columns

In [8]:
```
# Looping each DataFrame in the df_not_monthly list and convert the 'DATE' column to datetime format
for df in df_not_monthly:
    df['DATE'] = pd.to_datetime(df['DATE'], format='%Y-%m-%d')
```

Now dataframes GDP, Median Household Income, Vacant housing and Total housing are quartely, in order to merge them to merge_df we are going to forward fill the missing values

In [9]:
```
#setting DATE column as index
df3.set_index('DATE', inplace=True)
df4.set_index('DATE', inplace=True)
df5.set_index('DATE', inplace=True)
df6.set_index('DATE', inplace=True)
# Resample quarterly data to monthly and forward-fill
df3_resampled = df3.resample('MS').ffill()
df4_resampled = df4.resample('MS').ffill()
df5_resampled = df5.resample('MS').ffill()
df6_resampled = df6.resample('MS').ffill()
```

In [10]:
```
#reseting the indices
df3_resampled.reset_index(inplace=True)
df4_resampled.reset_index(inplace=True)
df5_resampled.reset_index(inplace=True)
df6_resampled.reset_index(inplace=True)
```

In [11]:
```
#mergeing the dataframes after forward-fill
merged_df = pd.merge(merged_df, df3_resampled, on='DATE', how='outer')
merged_df = pd.merge(merged_df, df4_resampled, on='DATE', how='outer')
merged_df = pd.merge(merged_df, df5_resampled, on='DATE', how='outer')
merged_df = pd.merge(merged_df, df6_resampled, on='DATE', how='outer')
```

In [12]:
```
merged_df
```

Out[12]:

	DATE	CSUSHPISA	DFF	CPIAUCSL	MSACSR	MORTGAGE15US	MORTGAGE30US	PERMIT	POPTHM	TLRESCONS	HOUST	UNRATE	UMCSENT	GDP	MEHOINUSA672N	EVACANTUSQ
0	2000-01-01	100.551	5.448387	169.300	4.3	7.8025	8.2100	1727	281083	NaN	1636	4.0	112.0	10002.179	67470.0	
1	2000-02-01	101.339	5.734828	170.000	4.3	7.9325	8.3250	1692	281299	NaN	1737	4.1	111.3	10002.179	67470.0	
2	2000-03-01	102.127	5.853548	171.000	4.3	7.8320	8.2400	1651	281531	NaN	1604	4.0	107.1	10002.179	67470.0	
3	2000-04-01	102.922	6.019667	170.900	4.4	7.8000	8.1525	1597	281763	NaN	1626	3.8	109.2	10247.720	67470.0	13
4	2000-05-01	103.678	6.268065	171.200	4.4	8.1825	8.5150	1543	281996	NaN	1575	4.0	110.7	10247.720	67470.0	13
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
280	2023-05-01	302.566	5.055806	303.294	7.2	5.8075	6.4250	1496	335013	864027.0	1583	3.7	59.0	27063.012	NaN	15
281	2023-06-01	304.593	5.076333	303.841	7.5	6.0880	6.7140	1441	335163	870655.0	1418	3.6	64.2	27063.012	NaN	15
282	2023-07-01	306.767	5.120000	304.348	7.1	6.1775	6.8400	1443	335329	865747.0	1451	3.5	71.5	27644.463	NaN	15
283	2023-08-01	309.155	5.330000	306.269	7.8	6.4300	7.0720	1541	335501	885776.0	1305	3.8	69.4	NaN	NaN	
284	2023-09-01	311.175	5.330000	307.481	7.2	6.5725	7.2000	1471	335675	884184.0	1346	3.8	67.9	NaN	NaN	

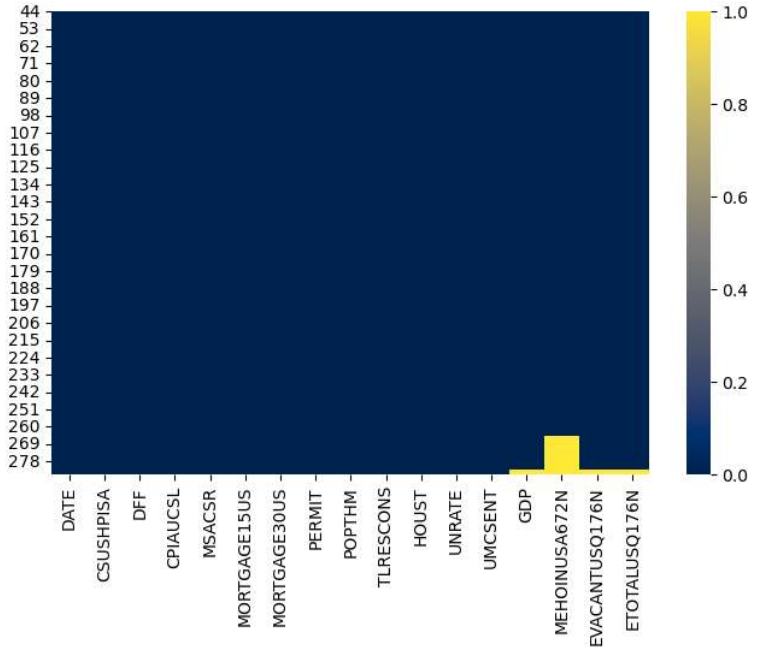
285 rows × 17 columns

Our analysis focuses solely on a 20-year span of house price index data, therefore we are excluding data predating September 1, 2003.

In [13]:
```
start_date = pd.to_datetime('2003-09-01')

# Keep rows starting from the specified date
merged_df = merged_df[merged_df['DATE'] >= start_date]
```

In [14]:
```
#Checkign for any missing values
plt.figure(figsize=(8,5))
sns.heatmap(merged_df.isna(),cmap='cividis')
merged_df.isna().sum()
```

Out[14]:
```
DATE               0
CSUSHPISA          0
DFF                0
CPIAUCSL           0
MSACSR             0
MORTGAGE15US       0
MORTGAGE30US       0
PERMIT             0
POPTHM             0
TLRESCONS          0
HOUST              0
UNRATE             0
UMCSENT            0
GDP                2
MEHOINUSA672N     20
EVACANTUSQ176N     2
ETOTALUSQ176N      2
dtype: int64
```



Even after forward fill the quartely data to monthly there are still missing values.

This missing values can be filled by using forward fill (because this data was converted from quartely/annually to monthly).

In [15]:
```
# Forward-fill missing values
merged_df['MEHOINUSA672N'] = merged_df['MEHOINUSA672N'].fillna(method='ffill')

# Forward-fill missing values
```

```
merged_df['GDP'] = merged_df['GDP'].fillna(method='ffill')

# Forward-fill missing values
merged_df['EVACANTUSQ176N'] = merged_df['EVACANTUSQ176N'].fillna(method='ffill')

# Forward-fill missing values
merged_df['ETOTALUSQ176N'] = merged_df['ETOTALUSQ176N'].fillna(method='ffill')
```

In [16]:
```
merged_df
```

Out[16]:

	DATE	CSUSHPISA	DFF	CPIAUCSL	MSACSR	MORTGAGE15US	MORTGAGE30US	PERMIT	POPTHM	TLRESCONS	HOUST	UNRATE	UMCSENT	GDP	MEHOINUSA672N	EVACANTUSQ
44	2003-09-01	136.294	1.010000	185.100	3.8	5.4575	6.1475	1961	291222	463954.0	1939	6.1	87.7	11566.669	65860.0	15
45	2003-10-01	137.531	1.010000	184.900	3.8	5.2740	5.9520	2012	291463	475234.0	1967	6.0	89.6	11772.234	65860.0	15
46	2003-11-01	138.794	0.996000	185.000	4.1	5.2725	5.9325	1918	291677	490441.0	2083	5.8	93.7	11772.234	65860.0	15
47	2003-12-01	140.179	0.984194	185.500	4.0	5.2040	5.8760	1987	291868	508637.0	2057	5.7	92.6	11772.234	65860.0	15
48	2004-01-01	141.646	0.997097	186.300	3.8	5.0150	5.7125	1952	292046	503659.0	1911	5.7	103.8	11923.447	65760.0	15
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
280	2023-05-01	302.566	5.055806	303.294	7.2	5.8075	6.4250	1496	335013	864027.0	1583	3.7	59.0	27063.012	74580.0	15
281	2023-06-01	304.593	5.076333	303.841	7.5	6.0880	6.7140	1441	335163	870655.0	1418	3.6	64.2	27063.012	74580.0	15
282	2023-07-01	306.767	5.120000	304.348	7.1	6.1775	6.8400	1443	335329	865747.0	1451	3.5	71.5	27644.463	74580.0	15
283	2023-08-01	309.155	5.330000	306.269	7.8	6.4300	7.0720	1541	335501	885776.0	1305	3.8	69.4	27644.463	74580.0	15
284	2023-09-01	311.175	5.330000	307.481	7.2	6.5725	7.2000	1471	335675	884184.0	1346	3.8	67.9	27644.463	74580.0	15

241 rows × 17 columns

In [17]:
```
#Rechecking missing values
merged_df.isna().sum()
```

Out[17]:
```
DATE             0
CSUSHPISA        0
DFF              0
CPIAUCSL         0
MSACSR           0
MORTGAGE15US     0
MORTGAGE30US     0
PERMIT           0
POPTHM           0
TLRESCONS        0
HOUST            0
UNRATE           0
UMCSENT          0
GDP              0
MEHOINUSA672N    0
EVACANTUSQ176N   0
ETOTALUSQ176N    0
dtype: int64
```

No missing data found.

In [18]:
```
#remaing merged_df to df
df=merged_df
df
```

Out[18]:

	DATE	CSUSHPISA	DFF	CPIAUCSL	MSACSR	MORTGAGE15US	MORTGAGE30US	PERMIT	POPTHM	TLRESCONS	HOUST	UNRATE	UMCSENT	GDP	MEHOINUSA672N	EVACANTUSQ
44	2003-09-01	136.294	1.010000	185.100	3.8	5.4575	6.1475	1961	291222	463954.0	1939	6.1	87.7	11566.669	65860.0	15
45	2003-10-01	137.531	1.010000	184.900	3.8	5.2740	5.9520	2012	291463	475234.0	1967	6.0	89.6	11772.234	65860.0	15
46	2003-11-01	138.794	0.996000	185.000	4.1	5.2725	5.9325	1918	291677	490441.0	2083	5.8	93.7	11772.234	65860.0	15
47	2003-12-01	140.179	0.984194	185.500	4.0	5.2040	5.8760	1987	291868	508637.0	2057	5.7	92.6	11772.234	65860.0	15
48	2004-01-01	141.646	0.997097	186.300	3.8	5.0150	5.7125	1952	292046	503659.0	1911	5.7	103.8	11923.447	65760.0	15
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
280	2023-05-01	302.566	5.055806	303.294	7.2	5.8075	6.4250	1496	335013	864027.0	1583	3.7	59.0	27063.012	74580.0	15
281	2023-06-01	304.593	5.076333	303.841	7.5	6.0880	6.7140	1441	335163	870655.0	1418	3.6	64.2	27063.012	74580.0	15
282	2023-07-01	306.767	5.120000	304.348	7.1	6.1775	6.8400	1443	335329	865747.0	1451	3.5	71.5	27644.463	74580.0	15
283	2023-08-01	309.155	5.330000	306.269	7.8	6.4300	7.0720	1541	335501	885776.0	1305	3.8	69.4	27644.463	74580.0	15
284	2023-09-01	311.175	5.330000	307.481	7.2	6.5725	7.2000	1471	335675	884184.0	1346	3.8	67.9	27644.463	74580.0	15

241 rows × 17 columns

In [19]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 241 entries, 44 to 284
Data columns (total 17 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   DATE            241 non-null    datetime64[ns]
 1   CSUSHPISA       241 non-null    float64
 2   DFF             241 non-null    float64
 3   CPIAUCSL        241 non-null    float64
 4   MSACSR          241 non-null    float64
 5   MORTGAGE15US    241 non-null    float64
 6   MORTGAGE30US    241 non-null    float64
 7   PERMIT          241 non-null    int64
 8   POPTHM          241 non-null    int64
 9   TLRESCONS       241 non-null    float64
 10  HOUST           241 non-null    int64
 11  UNRATE          241 non-null    float64
 12  UMCSENT         241 non-null    float64
 13  GDP             241 non-null    float64
 14  MEHOINUSA672N   241 non-null    float64
 15  EVACANTUSQ176N  241 non-null    float64
 16  ETOTALUSQ176N   241 non-null    float64
dtypes: datetime64[ns](1), float64(13), int64(3)
memory usage: 33.9 KB
```

The data types of the DataFrame align with our requiremnets.

According to the data source, recessions occurred during the following periods:

March 2001 to November 2001,
December 2007 to June 2009, and
February 2020 to April 2020.

Hence, we are introducing a new column named 'recession_indicator' into the DataFrame to facilitate a more detailed analysis.

In [20]:
```
# Defining recession periods
recession_periods = [
    (pd.to_datetime('2001-03-01'), pd.to_datetime('2001-11-01')),
    (pd.to_datetime('2007-12-01'), pd.to_datetime('2009-06-01')),
    (pd.to_datetime('2020-02-01'), pd.to_datetime('2020-04-01'))
]

# Function to check if a date is within a recession period
def check_recession(date):
    for start_date, end_date in recession_periods:
        if start_date <= date <= end_date:
            return 1  # 1 indicates a recession period
    return 0  # 0 indicates a non-recession period

# Applyign the function to create the 'recession_indicator' column
df['recession_indicator'] = df['DATE'].apply(check_recession)

print(df[['DATE', 'recession_indicator']])
```

```
          DATE  recession_indicator
44  2003-09-01                   0
45  2003-10-01                   0
46  2003-11-01                   0
47  2003-12-01                   0
48  2004-01-01                   0
..         ...                 ...
280 2023-05-01                   0
281 2023-06-01                   0
282 2023-07-01                   0
283 2023-08-01                   0
284 2023-09-01                   0

[241 rows x 2 columns]
```

In [21]:
```
#Checkign for its values
df['recession_indicator'].value_counts()
```

Out[21]:
```
0    219
1     22
Name: recession_indicator, dtype: int64
```

In [22]:
```
df
```

Out[22]:

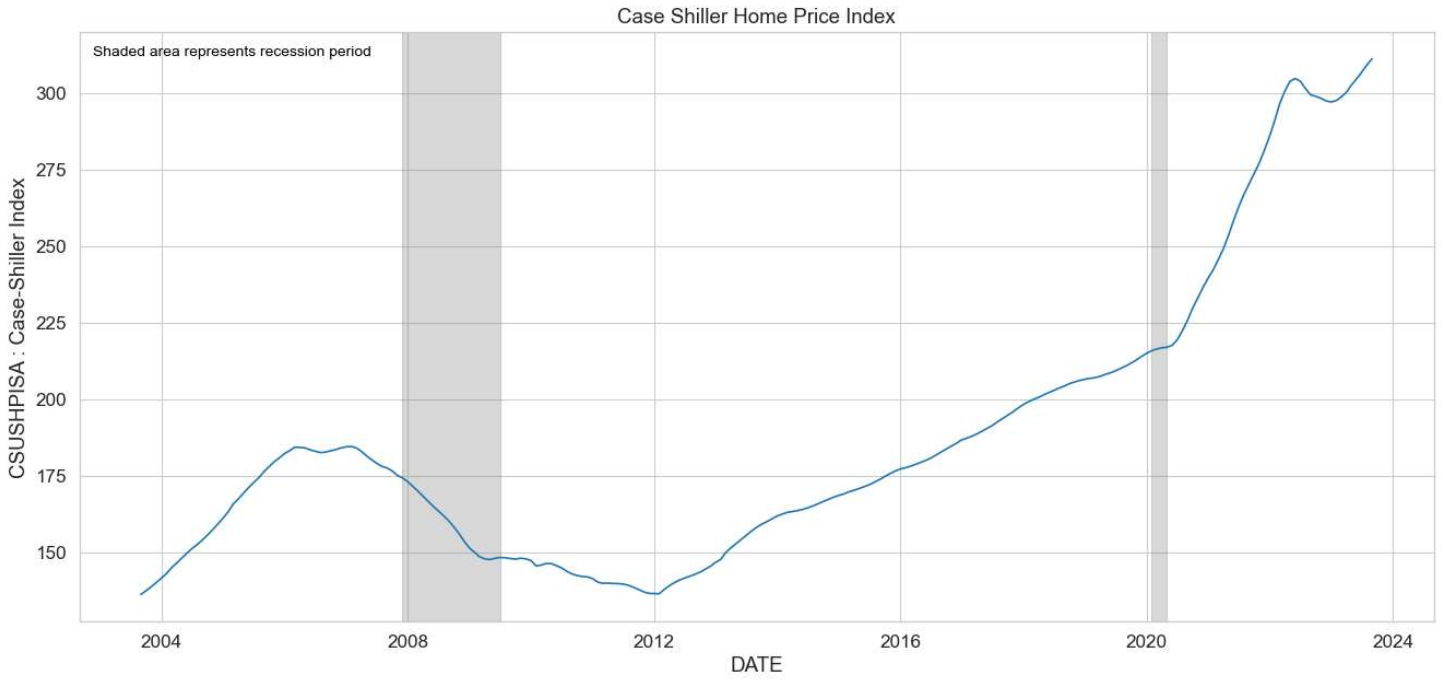
	DATE	CSUSHPISA	DFF	CPIAUCSL	MSACSR	MORTGAGE15US	MORTGAGE30US	PERMIT	POPTHM	TLRESCONS	HOUST	UNRATE	UMCSENT	GDP	MEHOINUSA672N	EVACANTUSQ
44	2003-09-01	136.294	1.010000	185.100	3.8	5.4575	6.1475	1961	291222	463954.0	1939	6.1	87.7	11566.669	65860.0	15
45	2003-10-01	137.531	1.010000	184.900	3.8	5.2740	5.9520	2012	291463	475234.0	1967	6.0	89.6	11772.234	65860.0	15
46	2003-11-01	138.794	0.996000	185.000	4.1	5.2725	5.9325	1918	291677	490441.0	2083	5.8	93.7	11772.234	65860.0	15
47	2003-12-01	140.179	0.984194	185.500	4.0	5.2040	5.8760	1987	291868	508637.0	2057	5.7	92.6	11772.234	65860.0	15
48	2004-01-01	141.646	0.997097	186.300	3.8	5.0150	5.7125	1952	292046	503659.0	1911	5.7	103.8	11923.447	65760.0	15
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
280	2023-05-01	302.566	5.055806	303.294	7.2	5.8075	6.4250	1496	335013	864027.0	1583	3.7	59.0	27063.012	74580.0	15
281	2023-06-01	304.593	5.076333	303.841	7.5	6.0880	6.7140	1441	335163	870655.0	1418	3.6	64.2	27063.012	74580.0	15
282	2023-07-01	306.767	5.120000	304.348	7.1	6.1775	6.8400	1443	335329	865747.0	1451	3.5	71.5	27644.463	74580.0	15
283	2023-08-01	309.155	5.330000	306.269	7.8	6.4300	7.0720	1541	335501	885776.0	1305	3.8	69.4	27644.463	74580.0	15
284	2023-09-01	311.175	5.330000	307.481	7.2	6.5725	7.2000	1471	335675	884184.0	1346	3.8	67.9	27644.463	74580.0	15

241 rows × 18 columns

Exploratory Data Analysis

- Case Shiller Home Price Index

In [23]:
```
plt.figure(figsize=(16, 7))
sns.set_style('whitegrid')
sns.set_context('paper', font_scale=1.4)

sns.lineplot(data=df, x='DATE', y='CSUSHPISA')

# Darken the area when recession_indicator is 1
in_recession = False
start_date = None

for i, row in df.iterrows():
    if row['recession_indicator'] == 1:
        if not in_recession:
            in_recession = True
            start_date = row['DATE']
    elif in_recession:
        in_recession = False
        plt.axvspan(start_date, row['DATE'], color='gray', alpha=0.3)


plt.text(0.01, 0.96, 'Shaded area represents recession period', transform=plt.gca().transAxes, fontsize=10, color='black')
plt.xlabel('DATE')
plt.ylabel('CSUSHPISA : Case-Shiller Index')
plt.title('Case Shiller Home Price Index')
plt.show()
```



---From the start of our analysis data late 2003, he Case Shiller Home Price Index exhibited a notable upward trend, reflecting a robust rise in housing values.However, this upward trajectory encountered a significant setback during the economic meltdown, resulting in a marked decline in the index.

---At the start of 2012 the index value nearly mirrored that of September 2003 (lowest).

---However from the early 2010s until 2020, the index demonstrated a consistent and gradual ascent.This shows renewed confindenec in real estate market.

---Following the pandemic in 2020,the index experienced remarkable surge till Jun 2022, later it fell down till Jan 2023 followed by the resurgence showing the dynamic housing market in US.

- Case Shiller Home Price Index reveals distinct patterns during recession periods

---In the recession from December 2007 to June 2009, a notable decline in the index is observed.

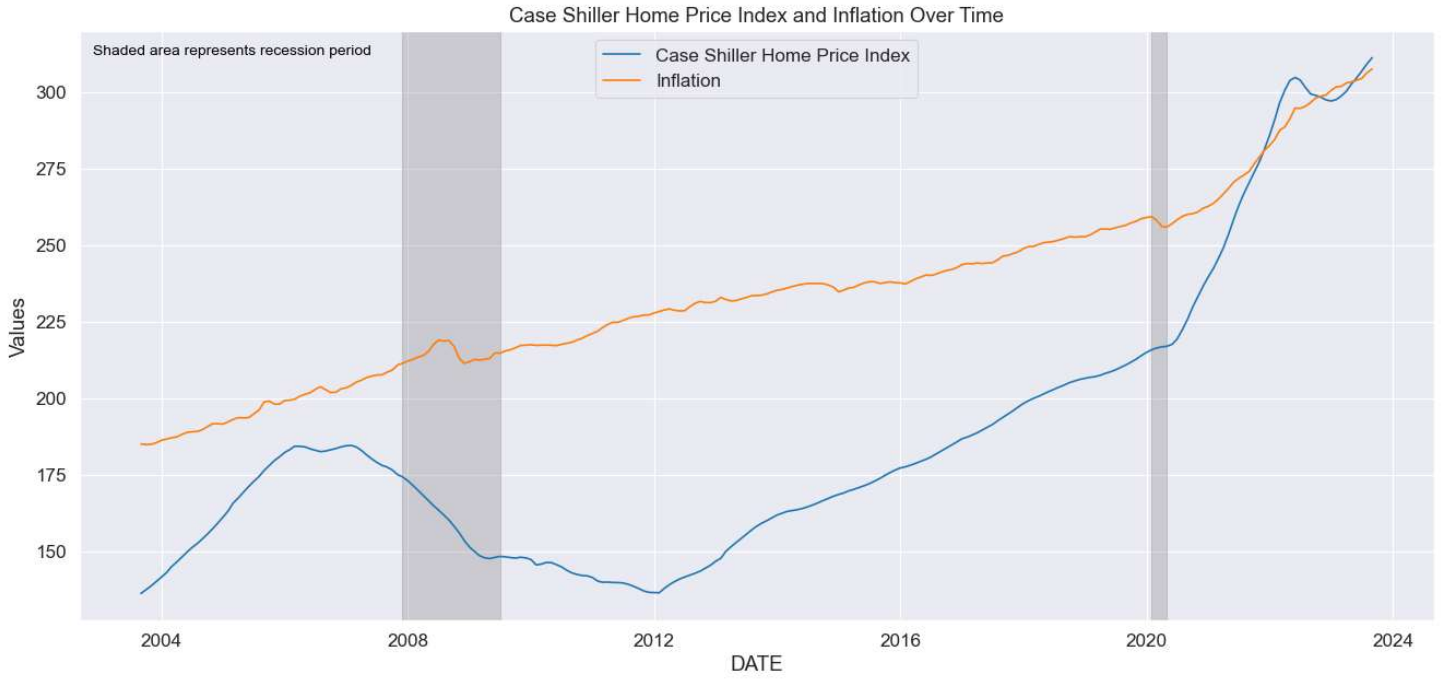
---Also prior to recession, there is a gradual descent indicating the incoming economic recession.

---However,during the more recent recession due to the COVID-19 pandemic (February 2020 to April 2020) CSUSHPISA demonstrates a different behavior.The index exhibits constant or slight upward tragectory.

- Case Shiller Home Price Index and Inflation

In [24]:
```
plt.figure(figsize=(16, 7))
sns.set_style('darkgrid')
sns.set_context('paper', font_scale=1.4)

# Plot the first line
ax = sns.lineplot(data=df, x='DATE', y='CSUSHPISA', label='Case Shiller Home Price Index')

# Plot the second line on the same axes
sns.lineplot(data=df, x='DATE', y='CPIAUCSL', ax=ax, label='Inflation')

in_recession = False
start_date = None

for i, row in df.iterrows():
    if row['recession_indicator'] == 1:
        if not in_recession:
```

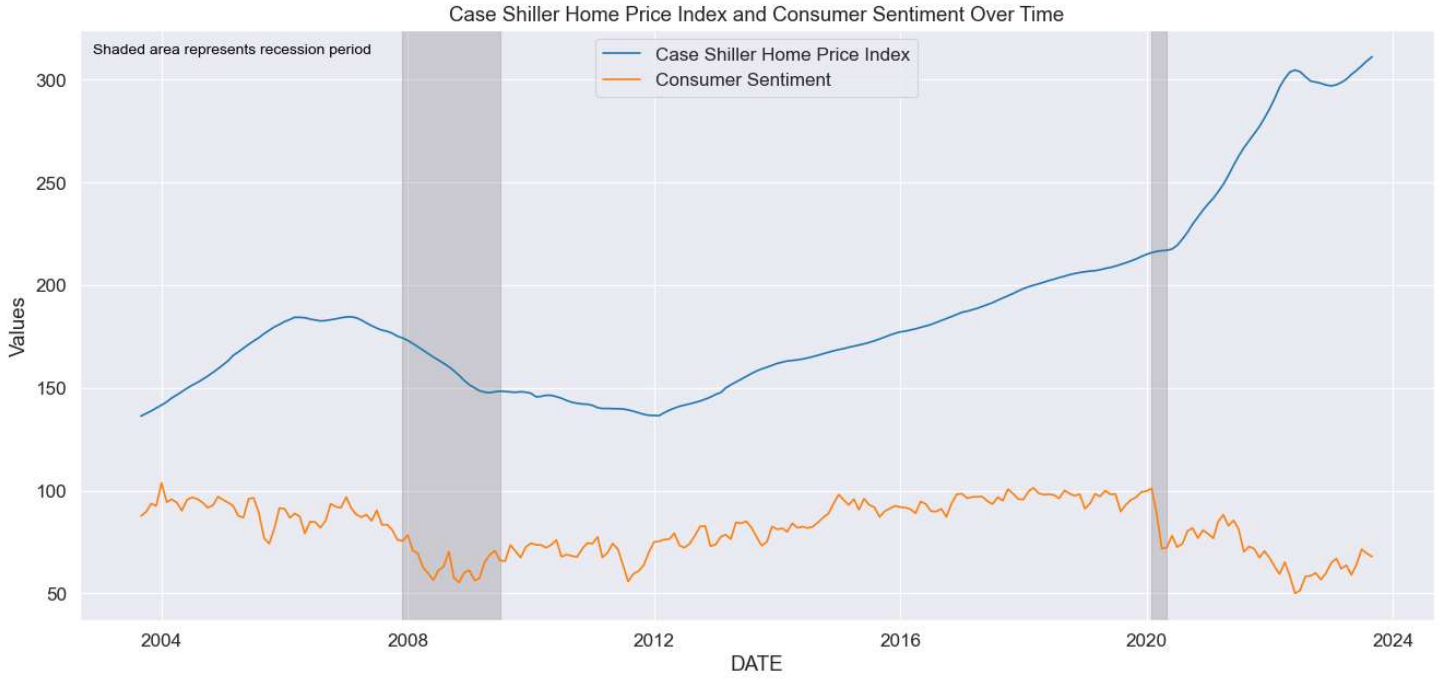
```
            in_recession = True
            start_date = row['DATE']
    elif in_recession:
        in_recession = False
        plt.axvspan(start_date, row['DATE'], color='gray', alpha=0.3)


plt.text(0.01, 0.96, 'Shaded area represents recession period', transform=plt.gca().transAxes, fontsize=10, color='black')

plt.xlabel('DATE')
plt.ylabel('Values')
plt.title('Case Shiller Home Price Index and Inflation Over Time')
plt.legend()
plt.show()
```



---Inflation has exhibited a consistent upward trend since September 2003 with slight decrease observed during recession periods.
---Similar to Case Shiller Home Price Index, inflation rate has increased substantially post May 2020.
---After analysis of the above graph, we can conclude that CS House price index and inflation share a high positive linear relationship.
---Inflation often leads to an increase in the cost of construction materials and labor. Developers may pass these higher costs onto buyers, which contrivutes to higher prices.
---As real state is often considered an appreciating asset. As the value of currency decreases due to inflation, investors tend to invest more real estate therefore House prices increases.

- Case Shiller Home Price Index and Consumer Sentiment

In [25]:
```
plt.figure(figsize=(16, 7))
sns.set_style('darkgrid')
sns.set_context('paper', font_scale=1.4)

# Plot the first line
ax = sns.lineplot(data=df, x='DATE', y='CSUSHPISA', label='Case Shiller Home Price Index')

# Plot the second line on the same axes
sns.lineplot(data=df, x='DATE', y='UMCSENT', ax=ax, label='Consumer Sentiment')

in_recession = False
start_date = None

for i, row in df.iterrows():
    if row['recession_indicator'] == 1:
        if not in_recession:
            in_recession = True
            start_date = row['DATE']
    elif in_recession:
        in_recession = False
        plt.axvspan(start_date, row['DATE'], color='gray', alpha=0.3)


plt.text(0.01, 0.96, 'Shaded area represents recession period', transform=plt.gca().transAxes, fontsize=10, color='black')

plt.xlabel('DATE')
plt.ylabel('Values')
plt.title('Case Shiller Home Price Index and Consumer Sentiment Over Time')
plt.legend()
plt.show()
```

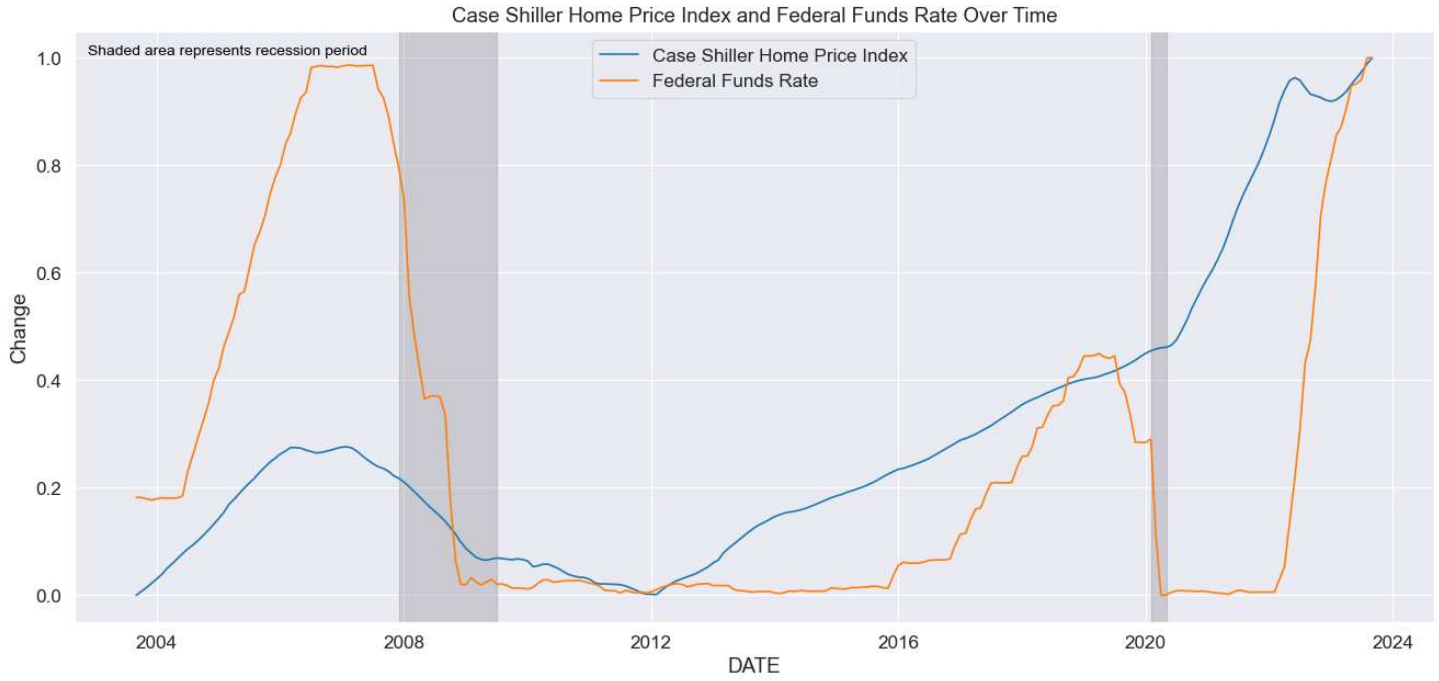


---As consumer sentiment is influenced by many factors such as employment, change in government policies, economic health, etc.
---In regards to Case Shiller Home Price Index, recent years reveal a slight negative linear relationship between them.
---As expected,the increase in House Prices inversely affects the Consumer sentiment.

- Case Shiller Home Price Index and Federal Fund Rate

In [26]:
```
#Normalizing both Case Shiller Home price index and Federal Funds Rate
scaler = MinMaxScaler()
df_normalized = df[['CSUSHPISA', 'DFF']].copy()
df_normalized[['CSUSHPISA', 'DFF']] = scaler.fit_transform(df_normalized)
df_normalized['DATE']=df['DATE']
```

In [27]:
```
plt.figure(figsize=(16, 7))
sns.set_style('darkgrid')
sns.set_context('paper', font_scale=1.4)

# Plot the first line
ax = sns.lineplot(data=df_normalized, x='DATE', y='CSUSHPISA', label='Case Shiller Home Price Index')

# Plot the second line on the same axes
sns.lineplot(data=df_normalized, x='DATE', y='DFF', ax=ax, label='Federal Funds Rate')

in_recession = False
start_date = None

for i, row in df.iterrows():
    if row['recession_indicator'] == 1:
        if not in_recession:
            in_recession = True
            start_date = row['DATE']
    elif in_recession:
        in_recession = False
        plt.axvspan(start_date, row['DATE'], color='gray', alpha=0.3)


plt.text(0.01, 0.96, 'Shaded area represents recession period', transform=plt.gca().transAxes, fontsize=10, color='black')

plt.xlabel('DATE')
plt.ylabel('Change')
plt.title('Case Shiller Home Price Index and Federal Funds Rate Over Time')
plt.legend()
plt.show()
```

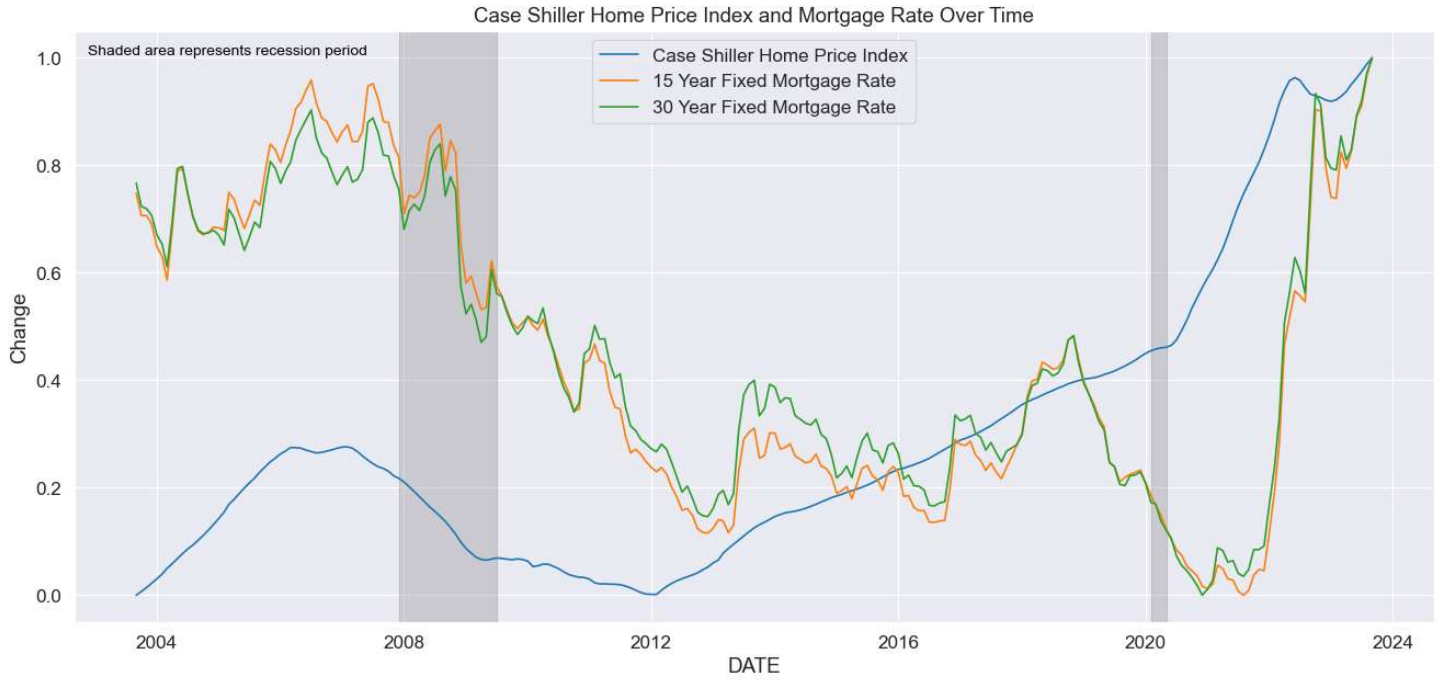


---As expected, Federal Funds Rate shows downward trend during recessionary periods.
---After the first recession, the DFF didnot increase till late 2015 and till 2022 after covid recession.
---Upon analyzing the trends of both Case Shiller Home Price Index and the Federal Funds Rate, it becomes clear that there is a slight positive linear relationship.
---While DFF appears to influence housing market dynamics, the observed correlation is moderate which indicates impact of other factors between them.

- Case Shiller Home Price Index and Mortgage Rate

In [28]:
```
#Normalizing both Case Shiller Home price index and Mortgage Rates
df_normalized = df[['CSUSHPISA', 'MORTGAGE15US','MORTGAGE30US']].copy()
df_normalized[['CSUSHPISA', 'MORTGAGE15US','MORTGAGE30US']] = scaler.fit_transform(df_normalized)
df_normalized['DATE']=df['DATE']
```

In [29]:
```
plt.figure(figsize=(16, 7))
sns.set_style('darkgrid')
sns.set_context('paper', font_scale=1.4)

# Plot the first line
ax = sns.lineplot(data=df_normalized, x='DATE', y='CSUSHPISA', label='Case Shiller Home Price Index')

# Plot the second line on the same axes
sns.lineplot(data=df_normalized, x='DATE', y='MORTGAGE15US', ax=ax, label='15 Year Fixed Mortgage Rate')
sns.lineplot(data=df_normalized, x='DATE', y='MORTGAGE30US', ax=ax, label='30 Year Fixed Mortgage Rate')

in_recession = False
start_date = None

for i, row in df.iterrows():
    if row['recession_indicator'] == 1:
        if not in_recession:
            in_recession = True
            start_date = row['DATE']
    elif in_recession:
        in_recession = False
        plt.axvspan(start_date, row['DATE'], color='gray', alpha=0.3)


plt.text(0.01, 0.96, 'Shaded area represents recession period', transform=plt.gca().transAxes, fontsize=10, color='black')

plt.xlabel('DATE')
plt.ylabel('Change')
plt.title('Case Shiller Home Price Index and Mortgage Rate Over Time')
plt.legend()
plt.show()
```

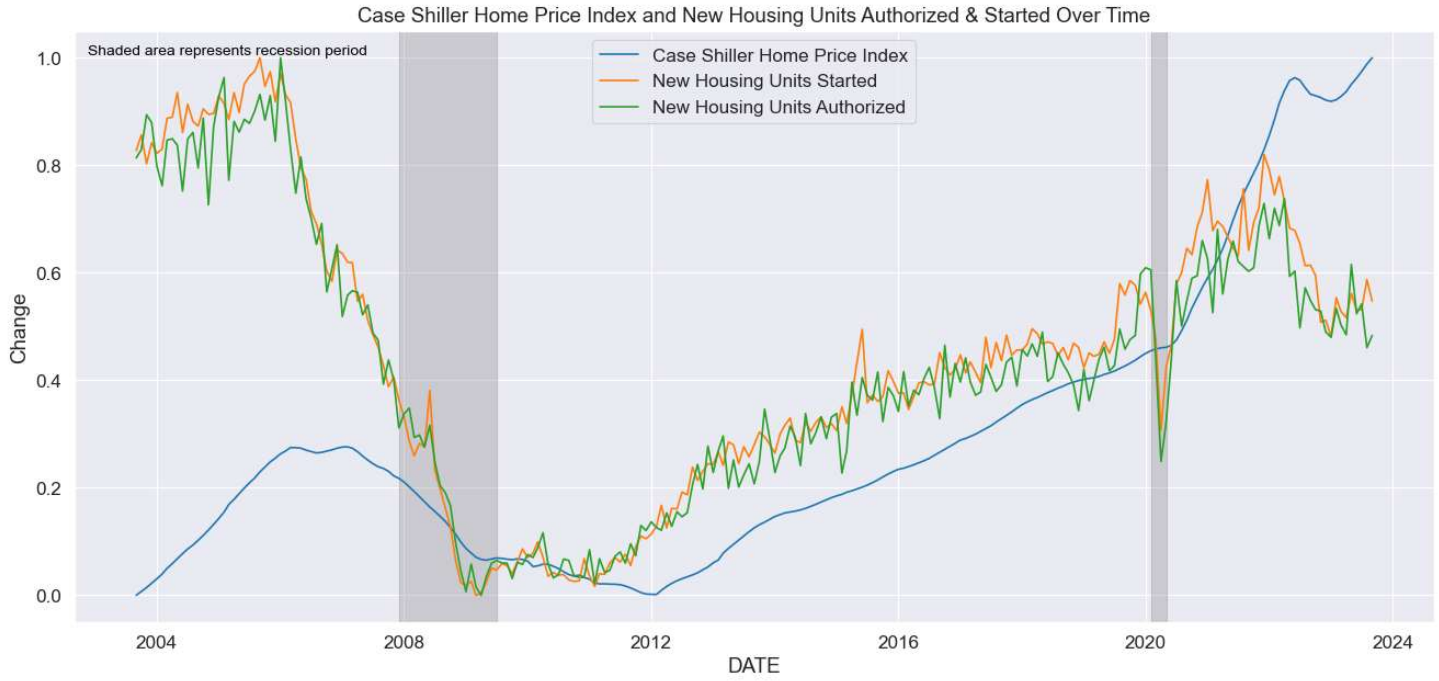


---Both 15 Year Fixed Mortgage Rate & 30 Year Fixed Mortgage Rate share similar trends.
---As expected, both Mortgage rates decrease during recessionnay periods like other economic factors.
---Contrary to common belief, House Price Index and Mortgage Rates show very minute to no correlation.
---These variables are largely independent of each other, at least in terms of a linear relationship.
---This observation suggests that the movements in these variables are largely independent of each other which highlights the complexity of factors influencing Housing market beyond just the influence of Mortgage rates.

- Case Shiller Home Price Index and New Housing Units Authorized & Started

In [30]:
```
#Normalizing both Case Shiller Home price index and Authorized Started Housing units
df_normalized = df[['CSUSHPISA', 'PERMIT','HOUST']].copy()
df_normalized[['CSUSHPISA', 'PERMIT','HOUST']] = scaler.fit_transform(df_normalized)
df_normalized['DATE']=df['DATE']
```

In [31]:
```
plt.figure(figsize=(16, 7))
sns.set_style('darkgrid')
sns.set_context('paper', font_scale=1.4)

# Plot the first line
ax = sns.lineplot(data=df_normalized, x='DATE', y='CSUSHPISA', label='Case Shiller Home Price Index')

# Plot the second line on the same axes
sns.lineplot(data=df_normalized, x='DATE', y='PERMIT', ax=ax, label='New Housing Units Started')
sns.lineplot(data=df_normalized, x='DATE', y='HOUST', ax=ax, label='New Housing Units Authorized')

in_recession = False
start_date = None

for i, row in df.iterrows():
    if row['recession_indicator'] == 1:
        if not in_recession:
            in_recession = True
            start_date = row['DATE']
    elif in_recession:
        in_recession = False
        plt.axvspan(start_date, row['DATE'], color='gray', alpha=0.3)


plt.text(0.01, 0.96, 'Shaded area represents recession period', transform=plt.gca().transAxes, fontsize=10, color='black')

plt.xlabel('DATE')
plt.ylabel('Change')
plt.title('Case Shiller Home Price Index and New Housing Units Authorized & Started Over Time')
plt.legend()
plt.show()
```

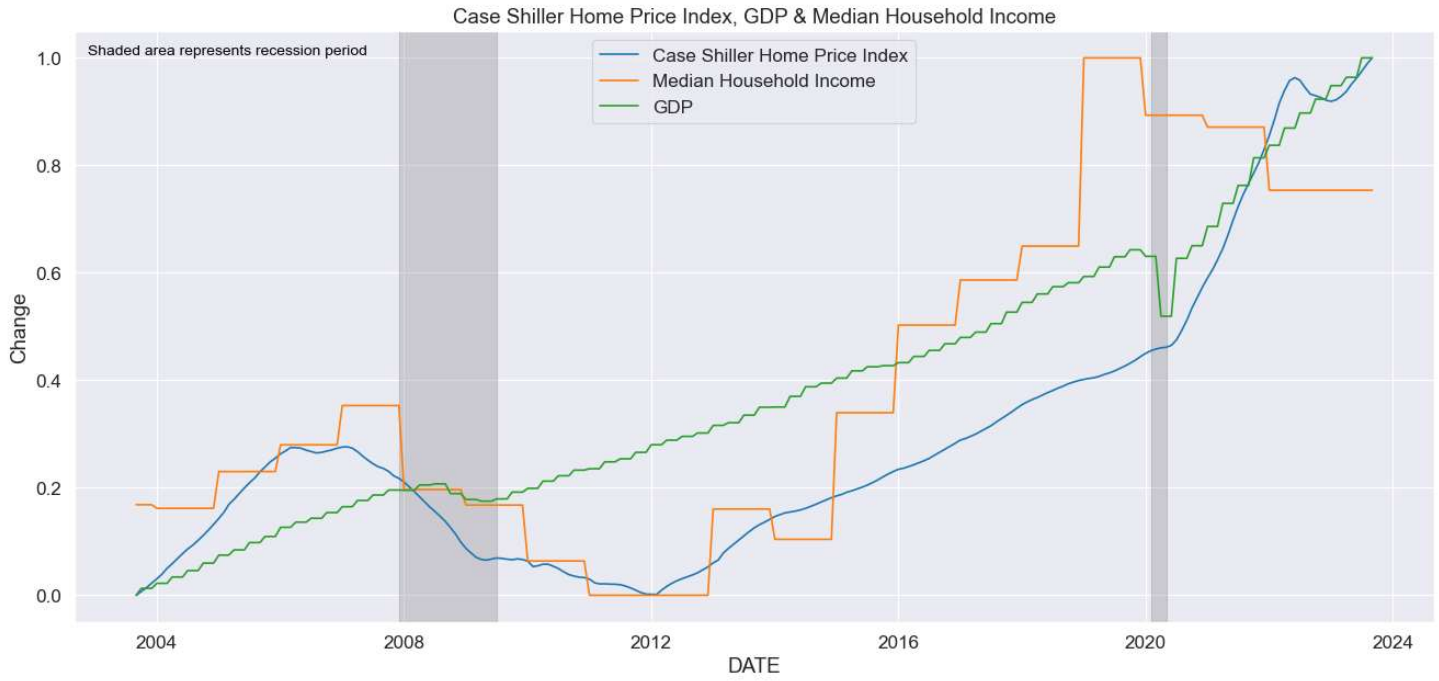


---The number of New Housing Units Authorized and New Housing Units Started share a near perfect linear relationship.
---And both New Housing variables show moderate positive correlation with Case Shiller Home Price Index.
---This reveals how housing unit authorization & commencement dynamics moderately correlate with changes in home prices.

- Case Shiller Home Price Index, GDP & Median Household Income

In [32]:
```
#Normalizing both Case Shiller Home price index, GDP and Median Household Income
df_normalized = df[['CSUSHPISA', 'MEHOINUSA672N','GDP']].copy()
df_normalized[['CSUSHPISA', 'MEHOINUSA672N','GDP']] = scaler.fit_transform(df_normalized)
df_normalized['DATE']=df['DATE']
```

In [33]:
```
plt.figure(figsize=(16, 7))
sns.set_style('darkgrid')
sns.set_context('paper', font_scale=1.4)

# Plot the first line
ax = sns.lineplot(data=df_normalized, x='DATE', y='CSUSHPISA', label='Case Shiller Home Price Index')

# Plot the second line on the same axes
sns.lineplot(data=df_normalized, x='DATE', y='MEHOINUSA672N', ax=ax, label='Median Household Income')
sns.lineplot(data=df_normalized, x='DATE', y='GDP', ax=ax, label='GDP')

in_recession = False
start_date = None

for i, row in df.iterrows():
    if row['recession_indicator'] == 1:
        if not in_recession:
            in_recession = True
            start_date = row['DATE']
    elif in_recession:
        in_recession = False
        plt.axvspan(start_date, row['DATE'], color='gray', alpha=0.3)


plt.text(0.01, 0.96, 'Shaded area represents recession period', transform=plt.gca().transAxes, fontsize=10, color='black')

plt.xlabel('DATE')
plt.ylabel('Change')
plt.title('Case Shiller Home Price Index, GDP & Median Household Income')
plt.legend()
plt.show()
```

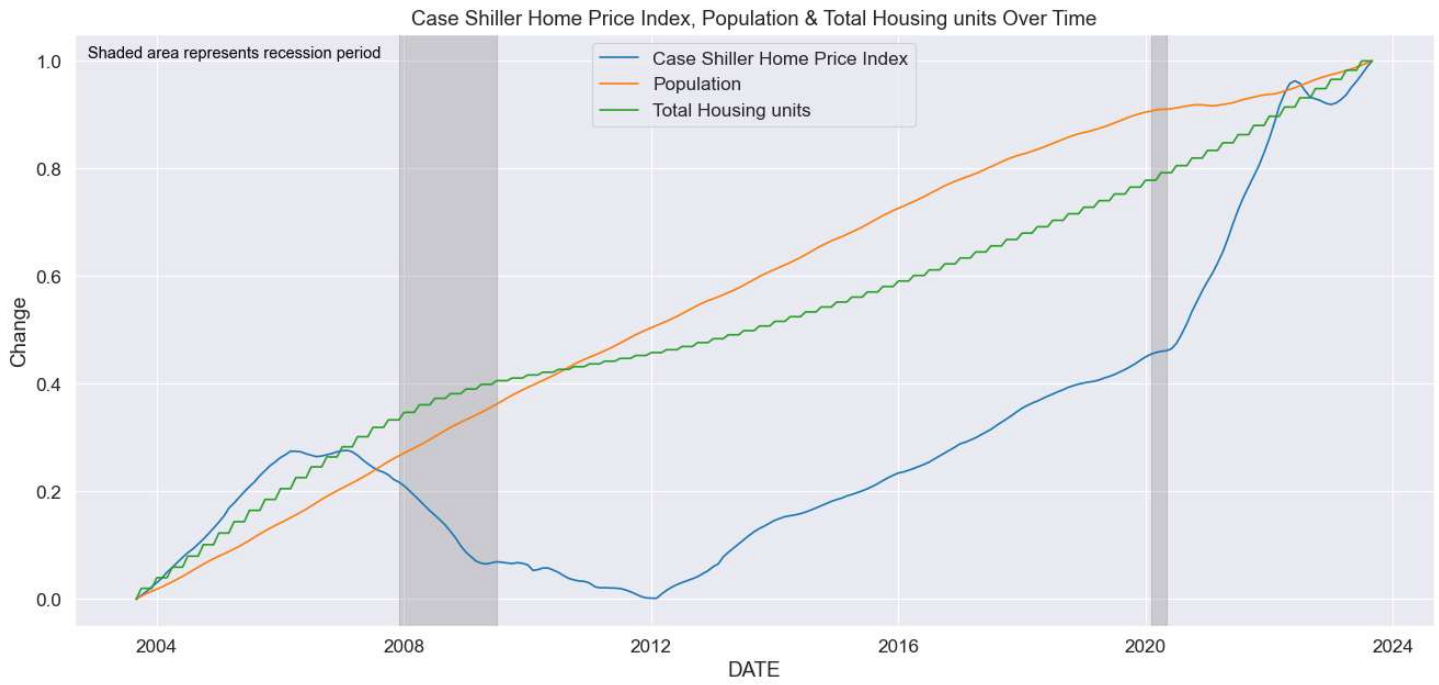


---Case Shiller Home Price Index, GDP & Median Household Income all three shows strong positive relationship among them.
---Emphasizing that increase in home prices align with overall economic growth and improvements in income.

- Case Shiller Home Price Index, Population & Total Housing units

In [34]:
```
#Normalizing both Case Shiller Home price index, Population & Total Housing units
df_normalized = df[['CSUSHPISA', 'POPTHM','ETOTALUSQ176N']].copy()
df_normalized[['CSUSHPISA', 'POPTHM','ETOTALUSQ176N']] = scaler.fit_transform(df_normalized)
df_normalized['DATE']=df['DATE']
```

In [35]:
```
plt.figure(figsize=(16, 7))
sns.set_style('darkgrid')
sns.set_context('paper', font_scale=1.4)

# Plot the first line
ax = sns.lineplot(data=df_normalized, x='DATE', y='CSUSHPISA', label='Case Shiller Home Price Index')

# Plot the second line on the same axes
sns.lineplot(data=df_normalized, x='DATE', y='POPTHM', ax=ax, label='Population')
sns.lineplot(data=df_normalized, x='DATE', y='ETOTALUSQ176N', ax=ax, label='Total Housing units')

in_recession = False
start_date = None

for i, row in df.iterrows():
    if row['recession_indicator'] == 1:
        if not in_recession:
            in_recession = True
            start_date = row['DATE']
    elif in_recession:
        in_recession = False
        plt.axvspan(start_date, row['DATE'], color='gray', alpha=0.3)


plt.text(0.01, 0.96, 'Shaded area represents recession period', transform=plt.gca().transAxes, fontsize=10.5, color='black')
plt.xlabel('DATE')
plt.ylabel('Change')
plt.title('Case Shiller Home Price Index, Population & Total Housing units Over Time')
plt.legend()
plt.show()
```

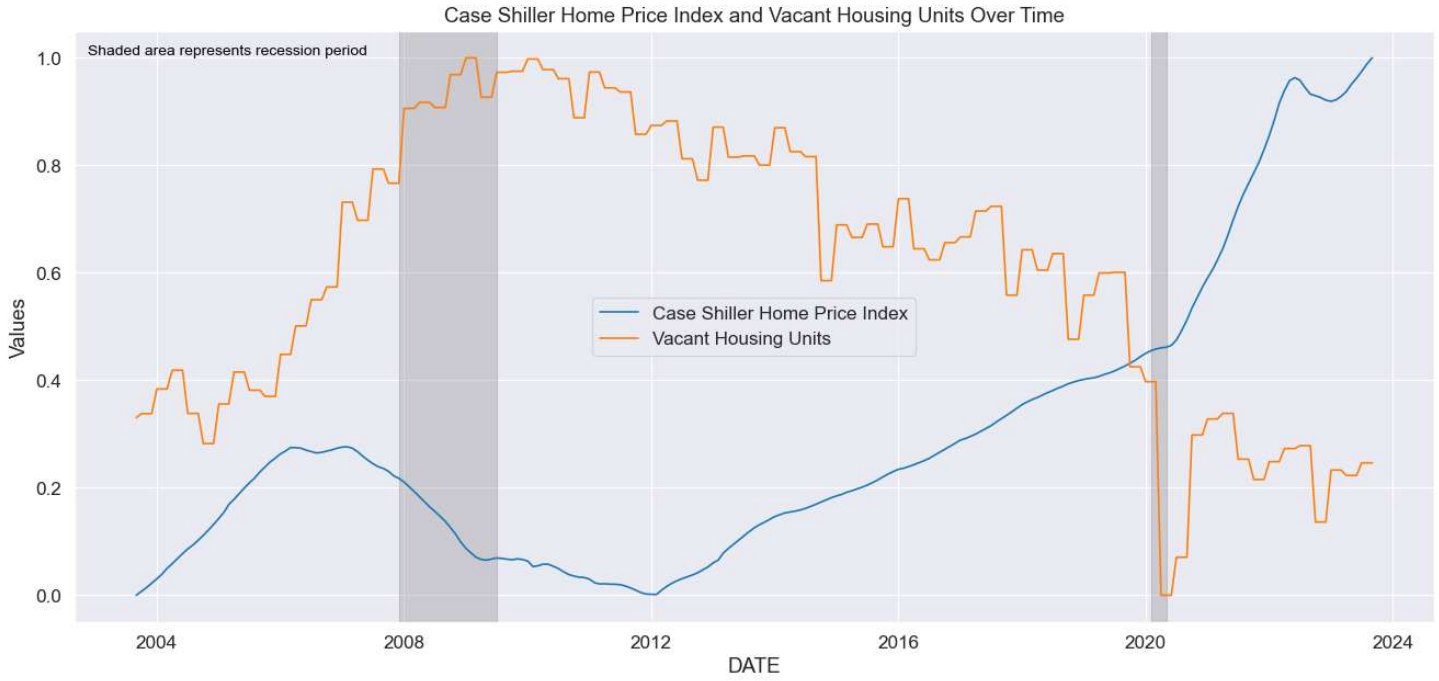


---In line with common knowledge, as populations grow, the demand for housing increases, leading to an increase in the total housing units.
---Case Shiller Home Price Index also has robust positive relationship to both.
---This indicates that on average home prices tend to rise with an expanding population and a greater availability of homes.
---However,it's noteworthy that this positive relationship doesn't follow uniformly during economic recessions. Because during recessions other economic factors come into play, which mitigate the usual positive correlation.
---Above graph illustrates intricate interplay of factors affecting housing prices beyong demographic trends.

- Case Shiller Home Price Index and Vacant Housing Units

In [36]:
```
#Normalizing both Case Shiller Home price index and Vacant Housing Units
df_normalized = df[['CSUSHPISA', 'EVACANTUSQ176N']].copy()
df_normalized[['CSUSHPISA', 'EVACANTUSQ176N']] = scaler.fit_transform(df_normalized)
df_normalized['DATE']=df['DATE']
```

In [37]:
```
plt.figure(figsize=(16, 7))
sns.set_style('darkgrid')
sns.set_context('paper', font_scale=1.4)

# Plot the first line
ax = sns.lineplot(data=df_normalized, x='DATE', y='CSUSHPISA', label='Case Shiller Home Price Index')

# Plot the second line on the same axes
sns.lineplot(data=df_normalized, x='DATE', y='EVACANTUSQ176N', ax=ax, label='Vacant Housing Units')

in_recession = False
start_date = None

for i, row in df.iterrows():
    if row['recession_indicator'] == 1:
        if not in_recession:
            in_recession = True
            start_date = row['DATE']
    elif in_recession:
        in_recession = False
        plt.axvspan(start_date, row['DATE'], color='gray', alpha=0.3)


plt.text(0.01, 0.96, 'Shaded area represents recession period', transform=plt.gca().transAxes, fontsize=10, color='black')

plt.xlabel('DATE')
plt.ylabel('Values')
plt.title('Case Shiller Home Price Index and Vacant Housing Units Over Time')
plt.legend()
plt.show()
```

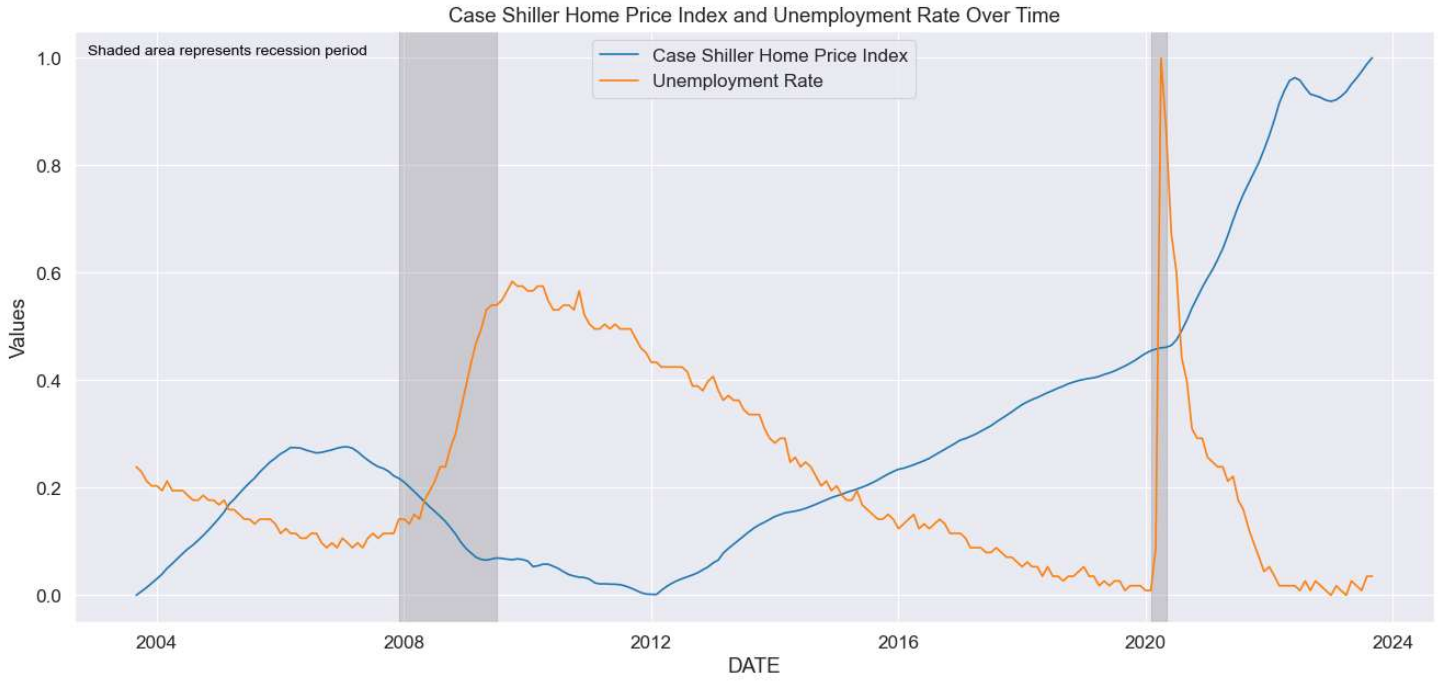


---There is a strong negative correlation between Case Shiller Home Price Index and the Number of Vacant housing units.
---An abundance of vacant units may indicate lower demand and potentially lower home prices.

- Case Shiller Home Price Index and Unemployment Rate

In [38]:
```
#Normalizing both Case Shiller Home price index and Unemployment rate
df_normalized = df[['CSUSHPISA', 'UNRATE']].copy()
df_normalized[['CSUSHPISA', 'UNRATE']] = scaler.fit_transform(df_normalized)
df_normalized['DATE']=df['DATE']
```

In [39]:
```
plt.figure(figsize=(16, 7))
sns.set_style('darkgrid')
sns.set_context('paper', font_scale=1.4)

# Plot the first line
ax = sns.lineplot(data=df_normalized, x='DATE', y='CSUSHPISA', label='Case Shiller Home Price Index')

# Plot the second line on the same axes
sns.lineplot(data=df_normalized, x='DATE', y='UNRATE', ax=ax, label='Unemployment Rate')

in_recession = False
start_date = None

for i, row in df.iterrows():
    if row['recession_indicator'] == 1:
        if not in_recession:
            in_recession = True
            start_date = row['DATE']
    elif in_recession:
        in_recession = False
        plt.axvspan(start_date, row['DATE'], color='gray', alpha=0.3)


plt.text(0.01, 0.96, 'Shaded area represents recession period', transform=plt.gca().transAxes, fontsize=10, color='black')

plt.xlabel('DATE')
plt.ylabel('Values')
plt.title('Case Shiller Home Price Index and Unemployment Rate Over Time')
plt.legend()
plt.show()
```

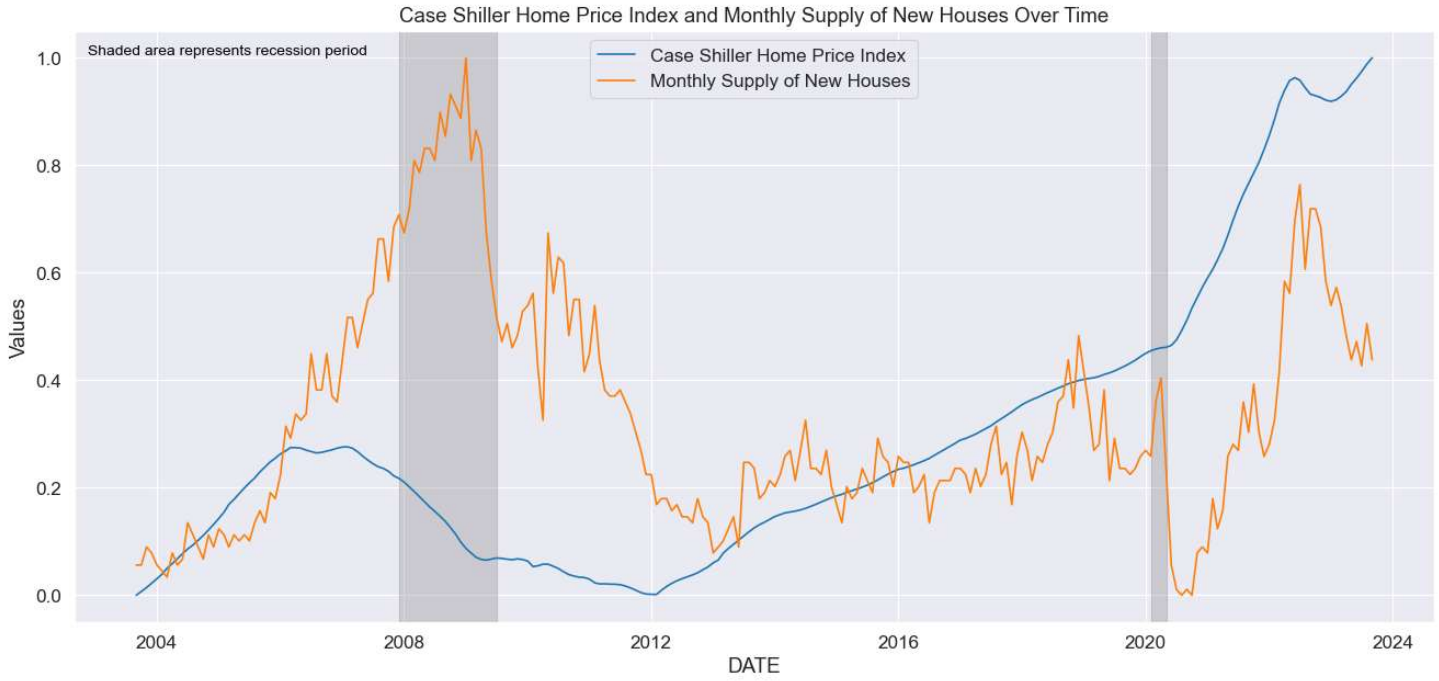


---There is moderate negative linear relationship between Case Shiller Home Price Index and Unemployment Rate.
---Lower Unemployment indicates healthier economy, home prices tend to rise.
---As shown in illustration, during recession periods, the unemplyment rate is high.Therefore housing market may experienced reduce demand, impacting prices.

- Case Shiller Home Price Index and Monthly Supply of New Houses

In [40]:
```
#Normalizing both Case Shiller Home price index and Monthly Supply of New Houses
scaler = MinMaxScaler()
df_normalized = df[['CSUSHPISA', 'MSACSR']].copy()
df_normalized[['CSUSHPISA', 'MSACSR']] = scaler.fit_transform(df_normalized)
df_normalized['DATE']=df['DATE']
```

In [41]:
```
plt.figure(figsize=(16, 7))
sns.set_style('darkgrid')
sns.set_context('paper', font_scale=1.4)

# Plot the first line
ax = sns.lineplot(data=df_normalized, x='DATE', y='CSUSHPISA', label='Case Shiller Home Price Index')

# Plot the second line on the same axes
sns.lineplot(data=df_normalized, x='DATE', y='MSACSR', ax=ax, label='Monthly Supply of New Houses')

in_recession = False
start_date = None

for i, row in df.iterrows():
    if row['recession_indicator'] == 1:
        if not in_recession:
            in_recession = True
            start_date = row['DATE']
    elif in_recession:
        in_recession = False
        plt.axvspan(start_date, row['DATE'], color='gray', alpha=0.3)


plt.text(0.01, 0.96, 'Shaded area represents recession period', transform=plt.gca().transAxes, fontsize=10, color='black')

plt.xlabel('DATE')
plt.ylabel('Values')
plt.title('Case Shiller Home Price Index and Monthly Supply of New Houses Over Time')
plt.legend()
plt.show()
```

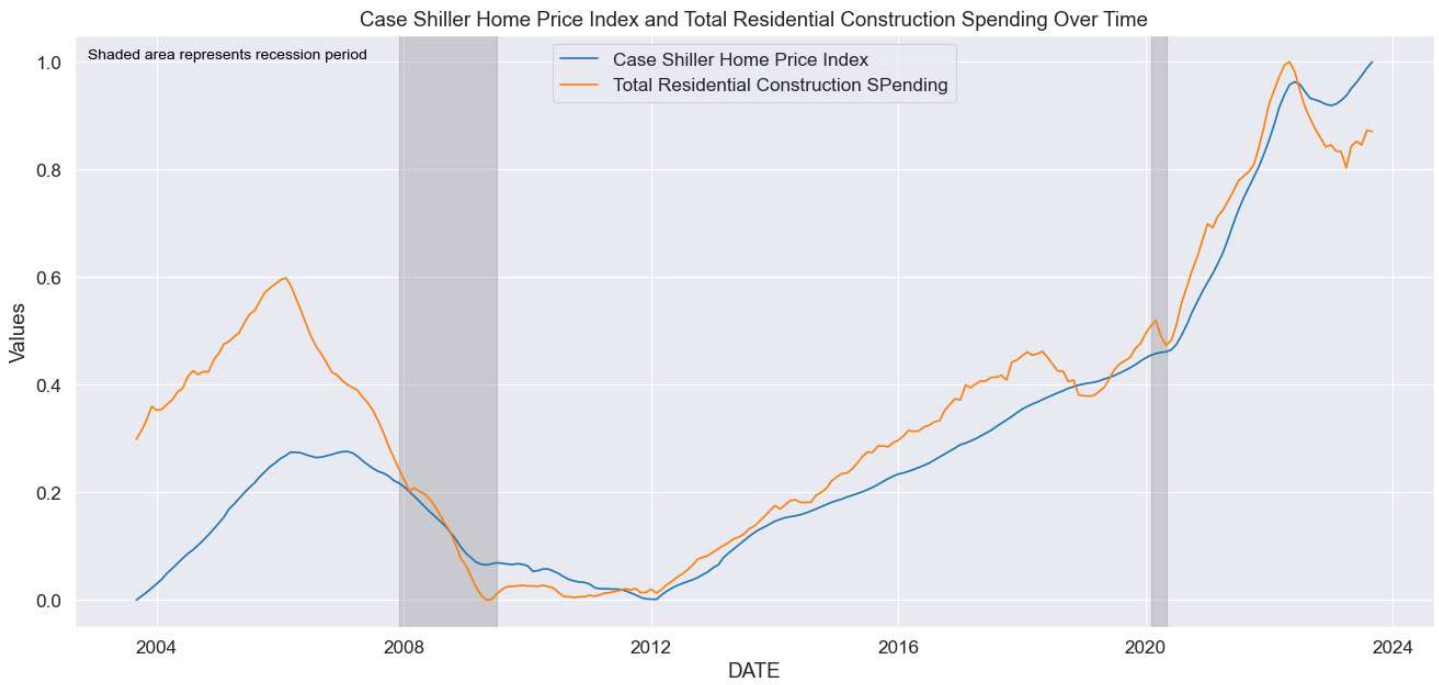


---There is a slight linear relationship between Case Shiller Home Price Index and Monthly Supply of New Houses.
---When the supply of new houses increases, it can contribute to an upward trend in home prices.
---Although, it may be expected that an increase in supply would lead to lower prices, the dynamics are more intricate.
---This graph underscores a crucial point: Evaluating housing supply alone doesn't offer a definitive forecast for the direction of house prices.
---The intricate balance with demand plays a pivotal role, often having even greater influence in the dynamics of pricing trends.

- Case Shiller Home Price Index and Total Residential Construction Spending

In [42]:
```
#Normalizing both Case Shiller Home price index and Total Residential Construction Spending
scaler = MinMaxScaler()
df_normalized = df[['CSUSHPISA', 'TLRESCONS']].copy()
df_normalized[['CSUSHPISA', 'TLRESCONS']] = scaler.fit_transform(df_normalized)
df_normalized['DATE']=df['DATE']
```

In [43]:
```
plt.figure(figsize=(16, 7))
sns.set_style('darkgrid')
sns.set_context('paper', font_scale=1.4)

# Plot the first line
ax = sns.lineplot(data=df_normalized, x='DATE', y='CSUSHPISA', label='Case Shiller Home Price Index')

# Plot the second line on the same axes
sns.lineplot(data=df_normalized, x='DATE', y='TLRESCONS', ax=ax, label='Total Residential Construction SPending')

in_recession = False
start_date = None

for i, row in df.iterrows():
    if row['recession_indicator'] == 1:
        if not in_recession:
            in_recession = True
            start_date = row['DATE']
    elif in_recession:
        in_recession = False
        plt.axvspan(start_date, row['DATE'], color='gray', alpha=0.3)


plt.text(0.01, 0.96, 'Shaded area represents recession period', transform=plt.gca().transAxes, fontsize=10, color='black')

plt.xlabel('DATE')
plt.ylabel('Values')
plt.title('Case Shiller Home Price Index and Total Residential Construction Spending Over Time')
plt.legend()
plt.show()
```

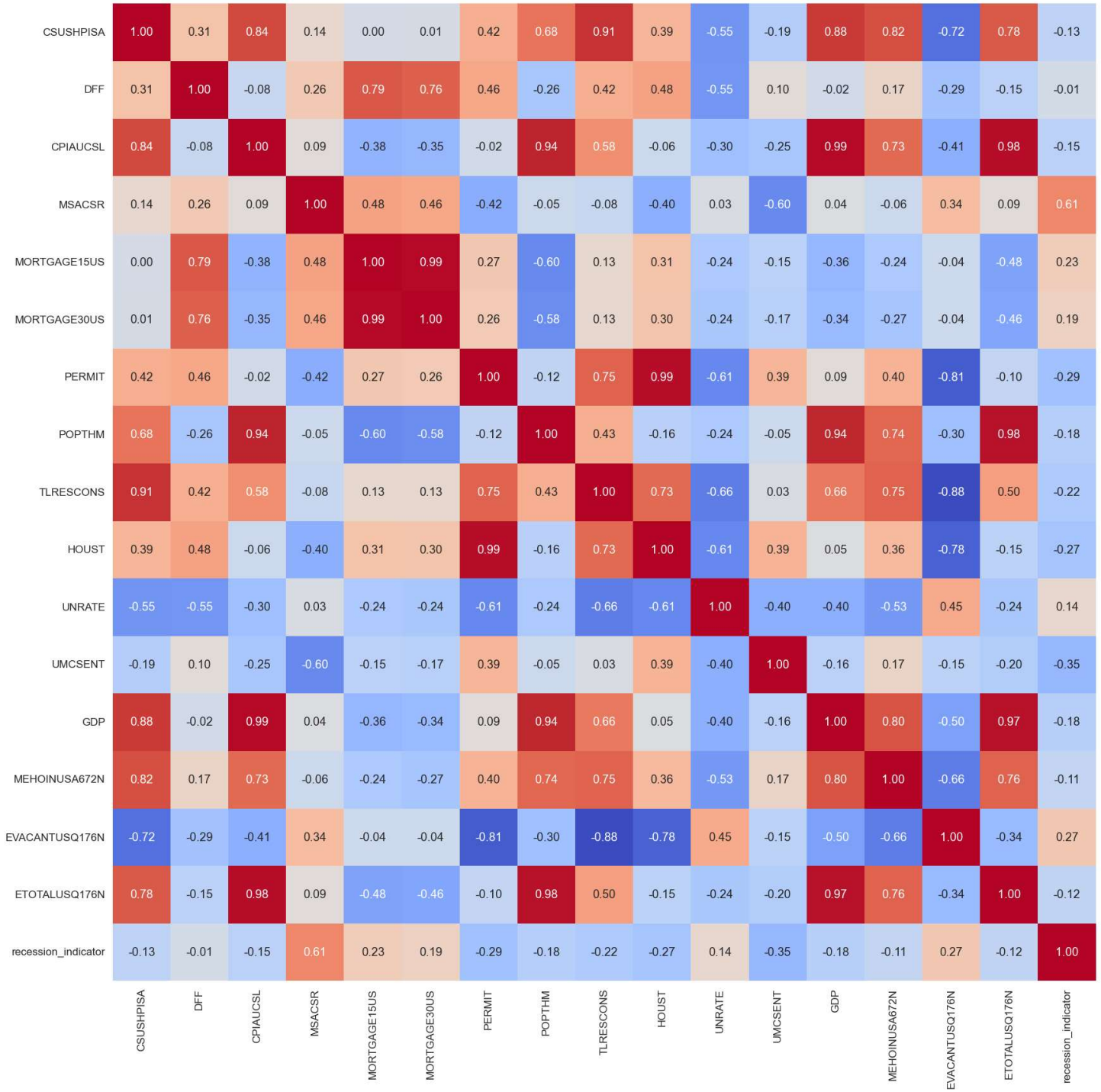


---Very strong positive linear relationship between Case Shiller Home Price Index and Total Residential Construction Spending.
---As spending on residential construction increases, home prices also increases.
---This shows the significant influence of construction spending on Case Shiller Home Price Index.

Relationship between features:

In [44]:
```
plt.figure(figsize=(20,20))
sns.heatmap(df.corr(),annot=True,cmap='coolwarm', fmt=".2f", cbar=False)
```

Out[44]:
```
<Axes: >
```



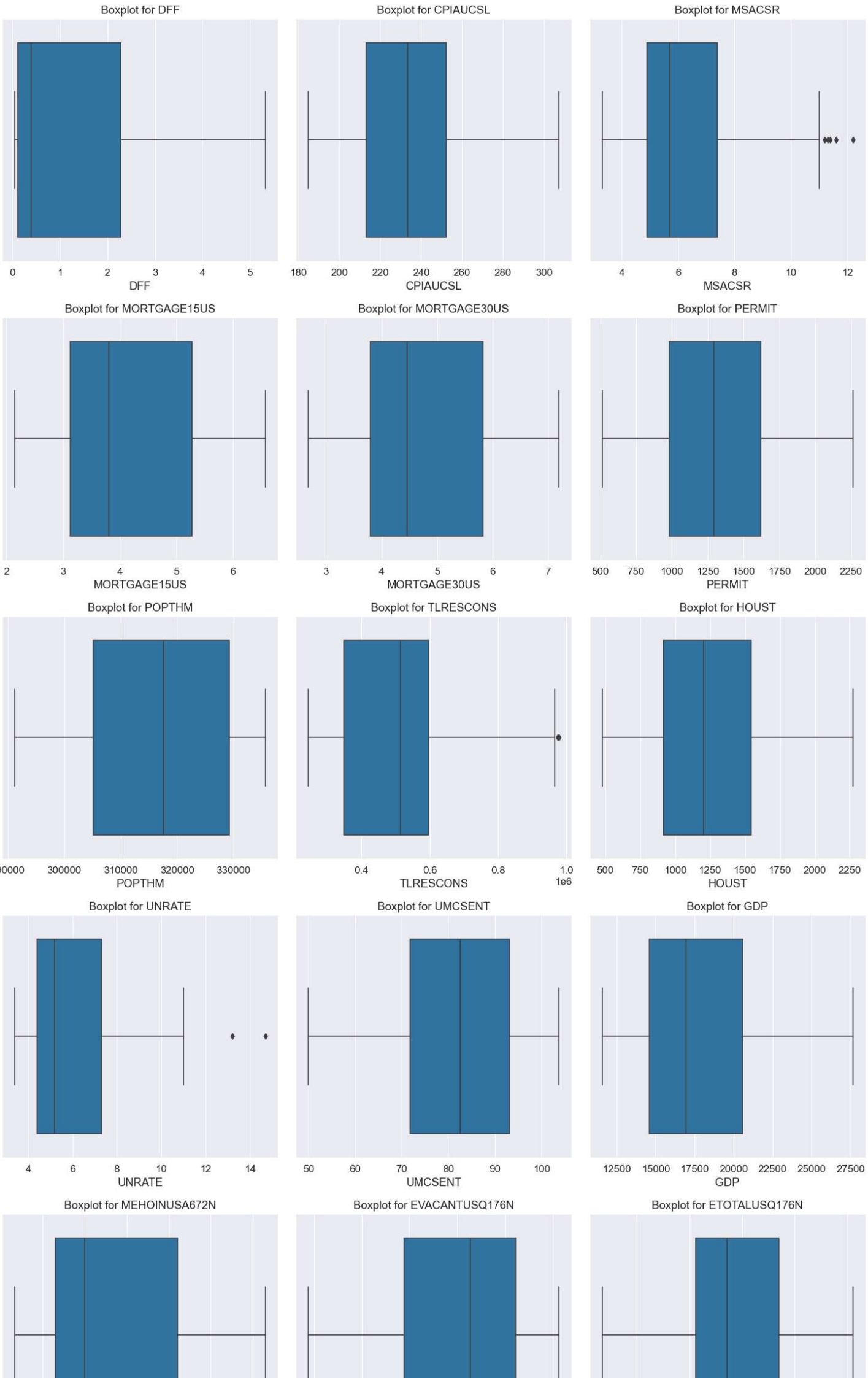
Correlation with Target variable:

```
---CSUSHPISA and CPIAUCSL (0.84) : Strong positive correlation
---CSUSHPISA and TLRESCONS (0.91) : Very high positive correlation
---CSUSHPISA and GDP (0.88) : Strong positive correlatio
---CSUSHPISA and UNRATE (-0.55) : Negative correlation
---CSUSHPISA and EVACANTUSQ176N (-0.72) : Strong negative correlation
---CSUSHPISA and ETOTALUSQ176N (0.78) : Strong positive correlation
```

Checking for outliers

In [45]:
```
plt.figure(figsize=(10, 10))
columns_to_plot = df.columns[2:-1]
num_columns = 3
num_rows = (len(columns_to_plot) + num_columns - 1) // num_columns
fig, axes = plt.subplots(num_rows, num_columns, figsize=(15, 5 * num_rows))
axes = axes.flatten()
for i, column in enumerate(columns_to_plot):
    sns.boxplot(x=df[column], ax=axes[i])
    axes[i].set_title(f'Boxplot for {column}')

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

```
<Figure size 1000x1000 with 0 Axes>
```


Boxplot for DFF
DFF
Boxplot for CPIAUCSL
CPIAUCSL
Boxplot for MSACSR
MSACSR
Boxplot for MORTGAGE15US
MORTGAGE15US
Boxplot for MORTGAGE30US
MORTGAGE30US
Boxplot for PERMIT
PERMIT
Boxplot for POPTHM
POPTHM
Boxplot for TLRESCONS
TLRESCONS
1e6
Boxplot for HOUST
HOUST
Boxplot for UNRATE
UNRATE
Boxplot for UMCSENT
UMCSENT
Boxplot for GDP
GDP
Boxplot for MEHOINUSA672N
Boxplot for EVACANTUSQ176N
Boxplot for ETOTALUSQ176N

---MSACSR (Monthly Supply of New Houses) has few outliers present.
---TLRESCONS (Total Residential Construction Spending) also has few outliers.
---Finally UNRATE (Unemployement rate) has couple of outliers (mainly due to Covid-19 recession).

As our dataset is very small 241 rows and 18 columns and these outliers indicates the recession period,we have chosen not to eliminate outliers

In [46]:
```
X=df.drop(['CSUSHPISA','DATE','recession_indicator'],axis=1) #Creating Features datafram and droppping extra columns
y=df['CSUSHPISA'] #Target dataframe
```

In [47]:
```
X
```

Out[47]:

	DFF	CPIAUCSL	MSACSR	MORTGAGE15US	MORTGAGE30US	PERMIT	POPTHM	TLRESCONS	HOUST	UNRATE	UMCSENT	GDP	MEHOINUSA672N	EVACANTUSQ176N	ETOTALUSQ17
44	1.010000	185.100	3.8	5.4575	6.1475	1961	291222	463954.0	1939	6.1	87.7	11566.669	65860.0	15614.0	1216
45	1.010000	184.900	3.8	5.2740	5.9520	2012	291463	475234.0	1967	6.0	89.6	11772.234	65860.0	15654.0	1221
46	0.996000	185.000	4.1	5.2725	5.9325	1918	291677	490441.0	2083	5.8	93.7	11772.234	65860.0	15654.0	1221
47	0.984194	185.500	4.0	5.2040	5.8760	1987	291868	508637.0	2057	5.7	92.6	11772.234	65860.0	15654.0	1221
48	0.997097	186.300	3.8	5.0150	5.7125	1952	292046	503659.0	1911	5.7	103.8	11923.447	65760.0	15895.0	1226
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
280	5.055806	303.294	7.2	5.8075	6.4250	1496	335013	864027.0	1583	3.7	59.0	27063.012	74580.0	15049.0	1451
281	5.076333	303.841	7.5	6.0880	6.7140	1441	335163	870655.0	1418	3.6	64.2	27063.012	74580.0	15049.0	1451
282	5.120000	304.348	7.1	6.1775	6.8400	1443	335329	865747.0	1451	3.5	71.5	27644.463	74580.0	15172.0	1455
283	5.330000	306.269	7.8	6.4300	7.0720	1541	335501	885776.0	1305	3.8	69.4	27644.463	74580.0	15172.0	1455
284	5.330000	307.481	7.2	6.5725	7.2000	1471	335675	884184.0	1346	3.8	67.9	27644.463	74580.0	15172.0	1455

241 rows × 15 columns

In [48]:
```
from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
x_s=scaler.fit_transform(X)
X=pd.DataFrame(x_s,columns=X.columns)
X
```

Out[48]:

	DFF	CPIAUCSL	MSACSR	MORTGAGE15US	MORTGAGE30US	PERMIT	POPTHM	TLRESCONS	HOUST	UNRATE	UMCSENT	GDP	MEHOINUSA672N	EVACANTUSQ176N	ETOTALU
0	-0.250669	-1.677959	-1.286011	1.094095	1.238152	1.418068	-1.879933	-0.227620	1.562716	0.086301	0.470283	-1.537876	-0.727718	-1.093944	-2
1	-0.250669	-1.684778	-1.286011	0.946084	1.068477	1.526963	-1.861933	-0.168868	1.625040	0.038156	0.617214	-1.487377	-0.727718	-1.064740	-2
2	-0.258828	-1.681368	-1.129771	0.944874	1.051553	1.326255	-1.845950	-0.089661	1.883239	-0.058134	0.934277	-1.487377	-0.727718	-1.064740	-2
3	-0.265708	-1.664322	-1.181851	0.889622	1.002517	1.473583	-1.831684	0.005113	1.825367	-0.106279	0.849211	-1.487377	-0.727718	-1.064740	-2
4	-0.258188	-1.637047	-1.286011	0.737175	0.860615	1.398851	-1.818389	-0.020815	1.500392	-0.106279	1.715333	-1.450231	-0.749265	-0.888782	-1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
236	2.107220	2.351709	0.484712	1.376405	1.478993	0.425204	1.390768	1.856178	0.770312	-1.069178	-1.749154	2.268916	1.151139	-1.506459	1
237	2.119183	2.370358	0.640953	1.602656	1.729816	0.307769	1.401971	1.890701	0.403046	-1.117323	-1.347026	2.268916	1.151139	-1.506459	1
238	2.144632	2.387643	0.432632	1.674846	1.839172	0.312039	1.414369	1.865137	0.476499	-1.165468	-0.782500	2.411753	1.151139	-1.416655	1
239	2.267020	2.453137	0.797193	1.878513	2.040524	0.521288	1.427216	1.969459	0.151524	-1.021033	-0.944898	2.411753	1.151139	-1.416655	1
240	2.267020	2.494459	0.484712	1.993453	2.151615	0.371824	1.440212	1.961167	0.242784	-1.021033	-1.060896	2.411753	1.151139	-1.416655	1

241 rows × 15 columns

In [49]:
```
y
```

Out[49]:
```
44     136.294
45     137.531
46     138.794
47     140.179
48     141.646
        ...
280    302.566
281    304.593
282    306.767
283    309.155
284    311.175
Name: CSUSHPISA, Length: 241, dtype: float64
```

We are devising two approaches for our building prediction model:

Approach 1: Identifying the most significant features through feature selection techniques

Approach 2: Pruning features based on multicollinearity

We will use the following metrics in order to select the most effective model and approach:

1. R-sqaured score (R2) - The R-squared (R2) score is a statistical measure that represents how well the model is explaining the variations of target variable. It ranges from 0 to 1 with 0 being the worst fit and 1 denoting the best fit model.
2.Mean Squared Error (NSE) - This a value is measured by calculating the squared differences between actual and predicted values. A lower MSE inndicates models prediction are on average close to actual values on average.
3.Cross Validation Score - Cross Validation score helps analyze the generalization ability of model. In other words, the performance of model on unseen data. This process aids in checking for potential overfitting and ensures the model's reliability beyond the training dataset.

Approach -1

Identifying the most significant features through feature selection techniques

---We are going to use Recursive Feature Elimination (RFE) for model training.

Recursive Feature Elimination (RFE)

---RFE a feature selection technique used to identify and retain the most relevant feattures for model training.
---In this process, it focusing on the most informative features and discarding the least important ones.
---In each iteration, Model assesses the importance of each feature and systematically discards the least relevant ones until the optimal subset of features is reached.

In [50]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
# Creating a linear regression model
model = LinearRegression()

# Choosing the number of features to select (n_features_to_select)
n_features = 9

# Appling RFE
selector = RFE(estimator=model, n_features_to_select=n_features)
X_train_selected = selector.fit_transform(X_train, y_train)
X_test_selected = selector.transform(X_test)

# Getting the selected feature names
selected_feature_names = X.columns[selector.support_]
selected_feature_names
```

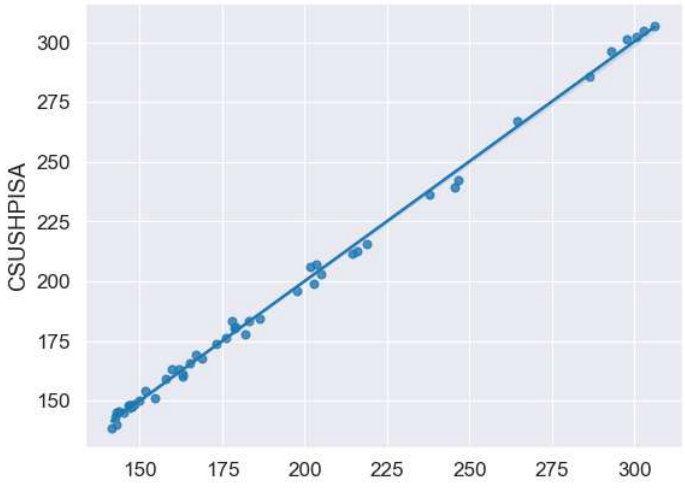
Out[50]:
```
Index(['MSACSR', 'MORTGAGE15US', 'MORTGAGE30US', 'POPTHM', 'TLRESCONS',
       'HOUST', 'GDP', 'MEHOINUSA672N', 'ETOTALUSQ176N'],
      dtype='object')
```

---These the most influential features.---

In [51]:
```
# Creating a linear regression model
model = LinearRegression()

# Training the model using only the selected features
model.fit(X_train_selected, y_train)

# Making predictions on the test set
y_pred = model.predict(X_test_selected)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f"Mean Squared Error: {mse}")
print(f"R-squared Score: {r2}")

# Performing K-foldcross-validation
cross_val_scores = cross_val_score(model, X, y, cv=5, scoring='r2')
print('Mean of Cross_validation_scorec: ',cross_val_scores.mean())
print("Cross-Validation Scores:", cross_val_scores)

sns.regplot(x=y_pred,y=y_test) #Graphing representation of comparison of test values & predicited values
```

```
Mean Squared Error: 6.42030981460218
R-squared Score: 0.9974426650265414
Mean of Cross_validation_scorec:  0.6364400237081351
Cross-Validation Scores: [ 0.49879764  0.92735073  0.91352967  0.9358636  -0.09334152]
```

Out[51]:
```
<Axes: ylabel='CSUSHPISA'>
```



--- Evaluation metrics of show that the model is fitting the data well.
--- Low MSE score and high R2 score suggests that the regression modle is is performing very well.
--- Mean cross validation score reflects the average perfomance of model accross different fold is respectable.
--- However, the negative value in the last fold (-0.22637502) could indicate potential issues such as overfitting or sensitive to specific trend of data.

This model is overfitting in certain cases and therefore it may not generalize well to unseen data. Hence, we will discard this approach from our final assessment of the factors affecting S&P Case-Schiller Home Price Index (CSUSHPISA).

Approach -2

Manually Pruning features based on multicollinearity

---From our data analysis and given the characteristics of the factors, we can conclude that numerous variables are highly correlated with each other.
---When the independent variables are interrelated, it leads to problems such as instability in judging the influence of each variable.
---While this doesn't affect the overall predictive ability of the model but constrains the identification of individual impacts of each

```
independent variable.
---This statistical phenomenon is known as Multicollinearity.
```

In order to tackle the multicollinearity, we will use the Variance Inflation Factor (VIF) and selectively prune features based on its analysis.

In [52]:
```
#Checking multicollinearity using the Variance Inflation Factor (VIF)
feature = X.columns.tolist()

vif_data = pd.DataFrame()
vif_data["Features"] = feature
vif_data["VIF"] = [variance_inflation_factor(X[feature].values, i) for i in range(len(feature))]
vif_data
```

Out[52]:

	Features	VIF
0	DFF	9.313408
1	CPIAUCSL	527.862177
2	MSACSR	10.763484
3	MORTGAGE15US	356.804101
4	MORTGAGE30US	313.345041
5	PERMIT	63.800716
6	POPTHM	286.543549
7	TLRESCONS	57.154082
8	HOUST	44.495920
9	UNRATE	15.431778
10	UMCSENT	8.599345
11	GDP	569.328587
12	MEHOINUSA672N	16.677318
13	EVACANTUSQ176N	17.074274
14	ETOTALUSQ176N	800.075570

VIF values under 10 are considered acceptable by statiscal standards.

- Observations:

```
---CPIAUCSL (VIF: 564.32): Extremely high VIF indicates significant multicollinearity
---MORTGAGE15US (VIF: 372.77): Strong multicollinearity
---POPTHM (VIF: 292.10): Also high multicollinearity
---MORTGAGE30US (VIF: 327.77): Similar to MORTGAGE15US, high VIF indicates multicollinearity
---GDP (VIF: 638.59): Exceptionally high VIF means very high degree of multicollinearity
---PERMIT, TLRESCONS and HOUST have moderate multicollinearity
---ETOTALUSQ176N (VIF: 813.44): Exceptionally High multicollinearity
---DFF, MSACSR, UNRATE, UMCSENT, MEHOINUSA672N, EVACANTUSQ176N have low VIF values which indicates lower levels of multicollinearity.
```

- Combating approach to multicollineaorty:

```
---MORTGAGE15US & MORTGAGE30US are expected to have high collinerity, hence we are going to drop MORTGAGE15US.
---CPIAUCSL,POPTHM and GDP are all correlated to each other hence droppping both POPTHM and GDP to migitate multicollinearity.
---Also dropping ETOTALUSQ176N from modeling phase due to an exceptionally high VIF value.
---PERMIT and HOUST are also hightly correlated features, hence dropping PERMIT.
```

In [53]:
```
X=X.drop(['MORTGAGE15US','POPTHM','GDP','ETOTALUSQ176N','PERMIT'],axis=1)
```

In [54]:
```
#Rechecking multicollinearity using the Variance Inflation Factor (VIF)
feature = X.columns.tolist()

vif_data = pd.DataFrame()
vif_data["Features"] = feature
vif_data["VIF"] = [variance_inflation_factor(X[feature].values, i) for i in range(len(feature))]
vif_data
```

Out[54]:

	Features	VIF
0	DFF	4.539019
1	CPIAUCSL	18.674783
2	MSACSR	7.248138
3	MORTGAGE30US	7.623744
4	TLRESCONS	30.979878
5	HOUST	33.603145
6	UNRATE	6.891479
7	UMCSENT	5.005099
8	MEHOINUSA672N	8.149024
9	EVACANTUSQ176N	12.263382

```
---CPIAUCSL (VIF: 19.11)  and EVACANTUSQ176N (VIF: 13.00) still shows low level of multicollinearity
---TLRESCONS (VIF: 32.93) and HOUST (VIF: 33.70) also have moderate level of multicollinearity
```

Given the strong demonstrated relationship with the target variable during Exploratory Data Analysis (EDA), CPIAUCSL and TLRESCONS will be retained in the model.

Therefore in order to address the persisting multicollineairty issue, dropping both EVACANTUSQ176N and HOUST.

In [55]:
```
X=X.drop(['HOUST','EVACANTUSQ176N'],axis=1)
```

In [56]:
```
#Rechecking multicollinearity using the Variance Inflation Factor (VIF)
feature = X.columns.tolist()

vif_data = pd.DataFrame()
vif_data["Features"] = feature
```

```
vif_data["VIF"] = [variance_inflation_factor(X[feature].values, i) for i in range(len(feature))]
vif_data
```

Out[56]:

	Features	VIF
0	DFF	4.331663
1	CPIAUCSL	4.265933
2	MSACSR	3.623114
3	MORTGAGE30US	6.591097
4	TLRESCONS	6.818836
5	UNRATE	3.841748
6	UMCSENT	4.789624
7	MEHOINUSA672N	7.119524

The VIF values now fall below the established statistical threshold.

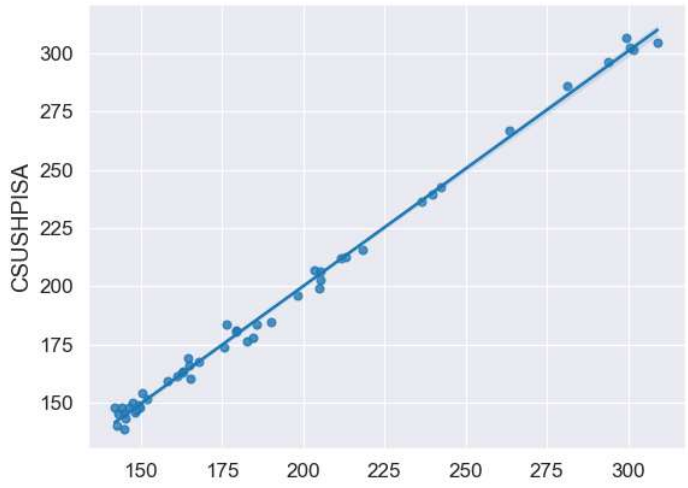
We will utilize this refined set of features for modeling phase.

In [57]:
```
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.20,random_state=42)
# Creating a linear regression model
model = LinearRegression()

# Training the model using only the selected features
model.fit(X_train, y_train)

# Making predictions on the test set
y_pred = model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f"Mean Squared Error: {mse}")
print(f"R-squared Score: {r2}")

# Performing K-foldcross-validation
cross_val_scores = cross_val_score(model, X, y, cv=5, scoring='r2')
print('Mean of Cross_validation_scorec: ',cross_val_scores.mean())
print("Cross-Validation Scores:", cross_val_scores)

sns.regplot(x=y_pred,y=y_test) #Graphing representation of comparison of test values & predicited values
```

```
Mean Squared Error: 11.314018125951144
R-squared Score: 0.9954934052904998
Mean of Cross_validation_scorec:  0.7272191884403603
Cross-Validation Scores: [0.69154609 0.68636049 0.94338801 0.78945973 0.52534162]
```

Out[57]:

```
<Axes: ylabel='CSUSHPISA'>
```



--- Evaluation metrics of show that the model is fitting the data well.
--- Low MSE score and high R2 score suggests that the regression model performance is good.
--- Mean cross validation score reflects the average perfomance of model accross different fold is respectable.
--- Cross validation scores in different folds are fairly consistent showing promising signs even though it may be overfitting slightly.

The Approach - 2 model demonstrates notable generalization capability with fairly less overfitting. Hence, we will adopt this approach for our final assessment of the factors affecting S&P Case-Schiller Home Price Index (CSUSHPISA).

Extracing the feature weight from the model

In [58]:
```
coefficients = model.coef_
feature_names = X.columns
intercept = model.intercept_
print("Intercept:", intercept)



# Print coefficients with corresponding feature names
for feature, coef in zip(feature_names, coefficients):
    print(f"{feature}: {coef}")
```

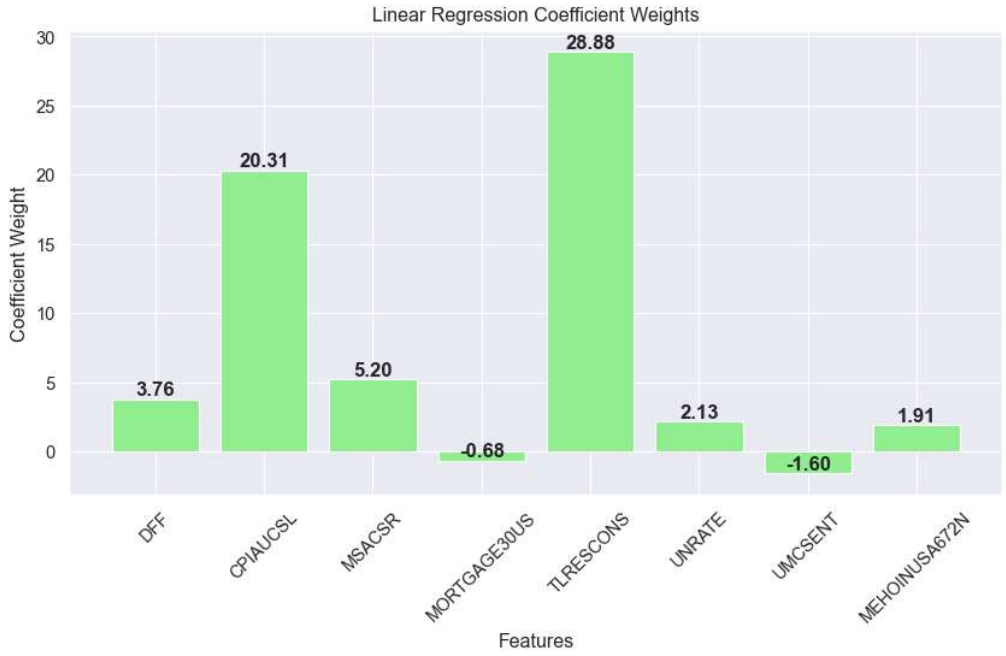
```
Intercept: 186.86837307361242
DFF: 3.7556651849348053
CPIAUCSL: 20.31356143759359
MSACSR: 5.195811397105125
MORTGAGE30US: -0.6849366627588651
TLRESCONS: 28.88043447298862
UNRATE: 2.1288625360406233
UMCSENT: -1.5965018830612117
MEHOINUSA672N: 1.9059839454061671
```

These coeeficients provides the impact of each feature on the overall model predicitions of S&P Case-Schiller Home Price Index (CSUSHPISA).

Positive coefficients suggest postive influence on CSUSHPISA while negative coefficients presents a negative influence.

The magnitude of the coefficient displays the strength of influence if other FACTORS are constant.

In [59]:
```
plt.figure(figsize=(10,5))
sns.set_style('darkgrid')
sns.set_context('paper', font_scale=1.2)
plt.bar(feature_names, coefficients, color='lightgreen')
plt.xlabel('Features')
plt.ylabel('Coefficient Weight')
plt.title('Linear Regression Coefficient Weights')
for feature, coef in zip(feature_names, coefficients):
    plt.text(feature, coef, f'{coef:.2f}', ha='center', va='bottom',fontsize=12,weight='bold')
plt.xticks(rotation=45)  # Rotate x-axis labels for better readability
plt.show()
```



Based on the findings, the individual factors affecting the S&P Case-Schiller Home Price Index (CSUSHPISA) in the last 20 years are:

Positive Influencers:

1.Total Construction Spending: Residential (TLRESCONS) - highest positive impact.

2.Consumer Price Index for All Urban Consumers (CPIAUCSL)- substantial positive impact.

3.Federal Funds Rate (DFF) - moderate positive influence

4.Monthly Supply of New Homes (MSACSR) - moderate positive influence

5.Median Household Income (MEHOINUSA672N) - low positive impact

6.Unemployment Rate (UNRATE) - surprisingly positive influence even if it is low.

Negative Influencers:

1. Mortgage Rates (15 Year & 30 Year) - Very low negative imapct.

2. University of Michigan Consumer Sentiment Index (UMCSENT) - Low negative influence.

In [60]:
```
#Saving our model
import joblib
joblib.dump(model,'USHomePricesIndex.pkl')
```

Out[60]:
```
['USHomePricesIndex.pkl']
```

Thank you for exploring till the end with me.

Best Regards,

Rahul Jadeja