Recession

The effects of recession on the real estate market

T

EDA

December 17, 2022

[713]:

```
import numpy as np
import pandas as pd
import math
import base64
import seaborn as sns
import matplotlib.pyplot as plt
```

Recession and it's effect on the Real Estate Market

To start the analysis, we will first define what is a recession and at it's indicators. A recession is a prolonged downturn in economy activity Our analysis will be focus on the 2001 and 2008 recession: to do so we will need to take a look at the year and after the those recession periods to notice any dips or peaks in the data sets. This article from https://corporatefinanceinstitute.com/resources/economics/recession/ states that recession indicators are:

1- Gross Domestic Product (GDP): total value generated by an economy (through goods and services produced) in a given time frame, adjusted for inflation.

2- Real income: calculated by measuring personal income, adjusting it for inflation, and discounting social security measures such as welfare payments

3- Manufacturing: health manufacturing sector

4- Employment: The number of people employed

5- Whole Retail Sale:market performance of goods

For the purpose of this analysis, we are going to focus on the first 4

1 Importing All Four Datasets Into Python

[714]:

```
gdp = pd.read_csv('GDP.csv')
real_personal_income = pd.read_csv('Real Personal Income.csv')
employment_rate = pd.read_csv('Employment rate.csv')
industrial_productions = pd.read_csv('Industrial Production.csv')
```

2 Looking at Each Data Set

Identifying the columns needed and deleting the ones are not relevant to the analysis

2.1 GDP

```
[715]: gdp.shape  #returns the number of rows by the number of columns
```

```
[715]: (303, 2)
```

```
[716]: gdp.head() #returns first 5 rows
```

```
[716]:         DATE      GDP
       0   1/1/1947  243.164
       1   4/1/1947  245.968
       2   7/1/1947  249.585
       3  10/1/1947  259.745
       4   1/1/1948  265.742
```

```
[717]: gdp.columns # list the number of columns
```

```
[717]: Index(['DATE', 'GDP'], dtype='object')
```

```
[718]: gdp.nunique(axis=0) #list amount of unique values
```

```
[718]: DATE    303
       GDP     303
       dtype: int64
```

```
[719]: gdp.info() #identifying what each columns are
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 303 entries, 0 to 302
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   DATE    303 non-null    object
 1   GDP     303 non-null    float64
dtypes: float64(1), object(1)
memory usage: 4.9+ KB
```

```
[720]: #converting date column and only keeping the months and year
       gdp['DATE'] = pd.to_datetime(gdp['DATE'])
       gdp['DATE'] = gdp['DATE'].dt.to_period('M')
       gdp
```

```
[720]:         DATE        GDP
       0     1947-01    243.164
       1     1947-04    245.968
       2     1947-07    249.585
       3     1947-10    259.745
       4     1948-01    265.742
```

```
..         …          …
298   2021-07   23550.420
299   2021-10   24349.121
300   2022-01   24740.480
301   2022-04   25248.476
302   2022-07   25698.960

[303 rows x 2 columns]
```

```
[721]: gdp = gdp[~(gdp['DATE'] < '1999-01')] #removing the years before 1999
       gdp.head(30)
```

```
[721]:        DATE         GDP
208   1999-01    9411.682
209   1999-04    9526.210
210   1999-07    9686.626
211   1999-10    9900.169
212   2000-01   10002.179
213   2000-04   10247.720
214   2000-07   10318.165
215   2000-10   10435.744
216   2001-01   10470.231
217   2001-04   10599.000
218   2001-07   10598.020
219   2001-10   10660.465
220   2002-01   10783.500
221   2002-04   10887.460
222   2002-07   10984.040
223   2002-10   11061.433
224   2003-01   11174.129
225   2003-04   11312.766
226   2003-07   11566.669
227   2003-10   11772.234
228   2004-01   11923.447
229   2004-04   12112.815
230   2004-07   12305.307
231   2004-10   12527.214
232   2005-01   12767.286
233   2005-04   12922.656
234   2005-07   13142.642
235   2005-10   13324.204
236   2006-01   13599.160
237   2006-04   13753.424
```

```
[722]: gdp.describe()
```

[722]:

```
                 GDP
count      95.000000
mean    15938.119179
std      4158.308728
min      9411.682000
25%     12647.250000
50%     15309.471000
75%     18871.750000
max     25698.960000
```

[723]:

```python
# selecting data set
beginning = '1998-10'
ending = '2010-10'
mask = (gdp['DATE'] > beginning) & (gdp['DATE'] <= ending)
gdp_recession = gdp.loc[mask]
gdp_recession = gdp_recession.set_index('DATE', drop=True)
gdp_recession
```

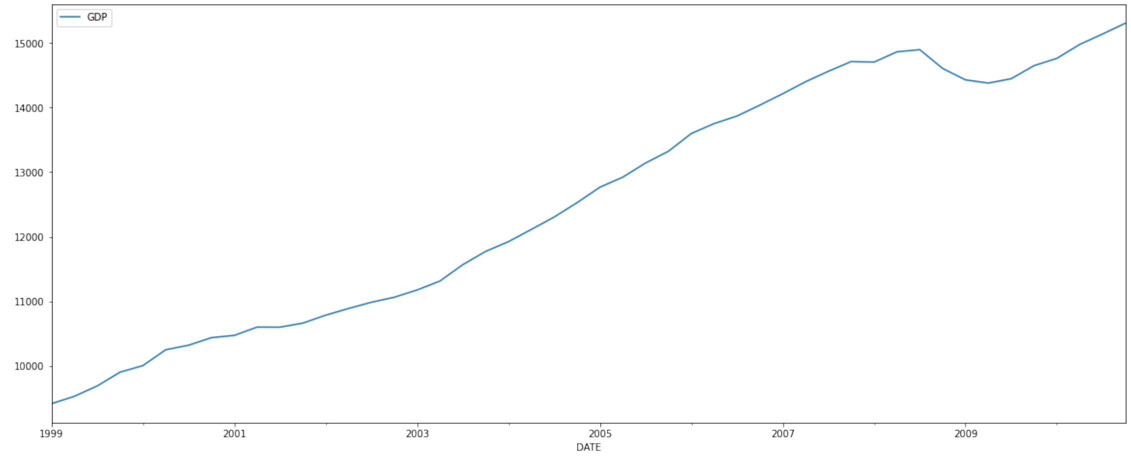
[723]:

```
               GDP
DATE
1999-01   9411.682
1999-04   9526.210
1999-07   9686.626
1999-10   9900.169
2000-01  10002.179
2000-04  10247.720
2000-07  10318.165
2000-10  10435.744
2001-01  10470.231
2001-04  10599.000
2001-07  10598.020
2001-10  10660.465
2002-01  10783.500
2002-04  10887.460
2002-07  10984.040
2002-10  11061.433
2003-01  11174.129
2003-04  11312.766
2003-07  11566.669
2003-10  11772.234
2004-01  11923.447
2004-04  12112.815
2004-07  12305.307
2004-10  12527.214
2005-01  12767.286
2005-04  12922.656
2005-07  13142.642
```

```
2005-10  13324.204
2006-01  13599.160
2006-04  13753.424
2006-07  13870.188
2006-10  14039.560
2007-01  14215.651
2007-04  14402.082
2007-07  14564.117
2007-10  14715.058
2008-01  14706.538
2008-04  14865.701
2008-07  14898.999
2008-10  14608.208
2009-01  14430.901
2009-04  14381.236
2009-07  14448.882
2009-10  14651.248
2010-01  14764.611
2010-04  14980.193
2010-07  15141.605
2010-10  15309.471
```

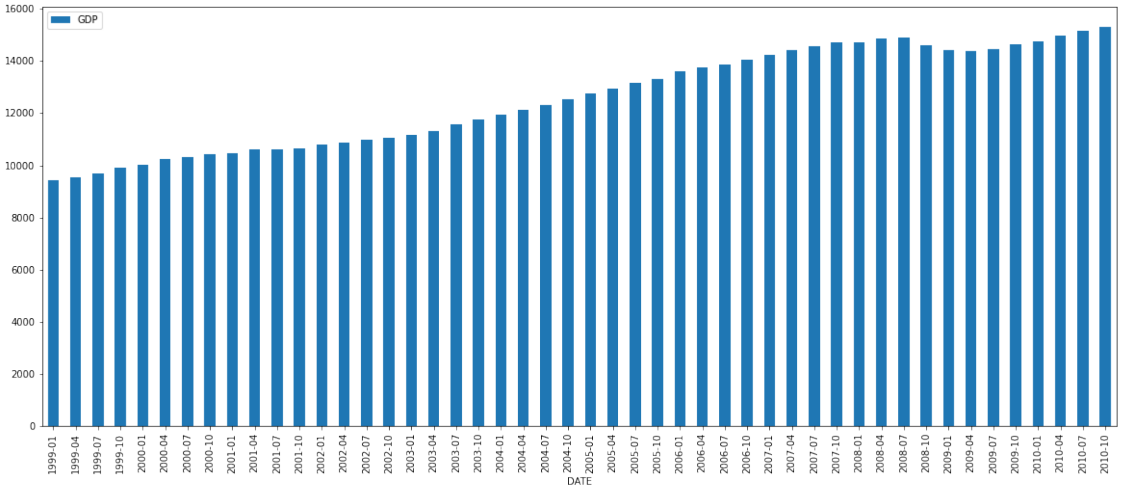
```
[724]: gdp_recession.plot.line(figsize=(20,8))
```

```
[724]: <AxesSubplot: xlabel='DATE'>
```



```
[725]: gdp_recession.plot.bar(figsize=(20,8))
```

```
[725]: <AxesSubplot: xlabel='DATE'>
```



2.2 Real Personal Income

[726]:
```
real_personal_income.shape
```

[726]:
```
(766, 2)
```

[727]:
```
real_personal_income.head()
```

[727]:
```
       DATE       RPI
0  1/1/1959  2442.158
1  2/1/1959  2451.778
2  3/1/1959  2467.594
3  4/1/1959  2483.671
4  5/1/1959  2498.026
```

[728]:
```
real_personal_income.columns
```

[728]:
```
Index(['DATE', 'RPI'], dtype='object')
```

[729]:
```
real_personal_income.nunique(axis=0)
```

[729]:
```
DATE    766
RPI     766
dtype: int64
```

[730]:
```
real_personal_income.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 766 entries, 0 to 765
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
```

```
---  ------  --------------  -----
 0   DATE    766 non-null    object
 1   RPI     766 non-null    float64
dtypes: float64(1), object(1)
memory usage: 12.1+ KB
```

[731]:

```
#converting date column only keeping the months and year
real_personal_income ['DATE']= pd.to_datetime(real_personal_income['DATE'])
real_personal_income ['DATE']= real_personal_income['DATE'].dt.to_period('M')
real_personal_income
```

[731]:

```
        DATE        RPI
0    1959-01   2442.158
1    1959-02   2451.778
2    1959-03   2467.594
3    1959-04   2483.671
4    1959-05   2498.026
..       …          …
761  2022-06  17558.655
762  2022-07  17660.561
763  2022-08  17674.301
764  2022-09  17685.155
765  2022-10  17750.811

[766 rows x 2 columns]
```

[732]:

```
real_personal_income =␣
 ↪real_personal_income[~(real_personal_income['DATE']<'1999-01')] #removing␣
 ↪the years before 1999
real_personal_income.head(30)
```

[732]:

```
        DATE        RPI
480  1999-01  10373.294
481  1999-02  10417.250
482  1999-03  10430.593
483  1999-04  10402.170
484  1999-05  10427.595
485  1999-06  10466.837
486  1999-07  10494.430
487  1999-08  10544.201
488  1999-09  10539.991
489  1999-10  10603.750
490  1999-11  10677.288
491  1999-12  10764.920
492  2000-01  10861.073
493  2000-02  10906.015
494  2000-03  10936.525
```

```
495  2000-04  10996.136
496  2000-05  11035.828
497  2000-06  11066.822
498  2000-07  11124.906
499  2000-08  11184.673
500  2000-09  11193.727
501  2000-10  11220.903
502  2000-11  11226.227
503  2000-12  11250.439
504  2001-01  11323.538
505  2001-02  11350.432
506  2001-03  11387.620
507  2001-04  11358.418
508  2001-05  11324.681
509  2001-06  11307.367
```

[733]:

```
#setting date column as index and pulling 1,4,7,10 months to match GDP data␣
 ↪format

real_personal_income = real_personal_income.set_index('DATE', drop=True)
real_personal_income = real_personal_income[real_personal_income.index.month.
 ↪isin([1,4,7,10])]
real_personal_income
```

[733]:

```
               RPI
DATE
1999-01  10373.294
1999-04  10402.170
1999-07  10494.430
1999-10  10603.750
2000-01  10861.073
…              …
2021-10  17937.674
2022-01  17749.994
2022-04  17664.987
2022-07  17660.561
2022-10  17750.811

[96 rows x 1 columns]
```

[734]:

```
real_personal_income.describe() #return basic stat info
```

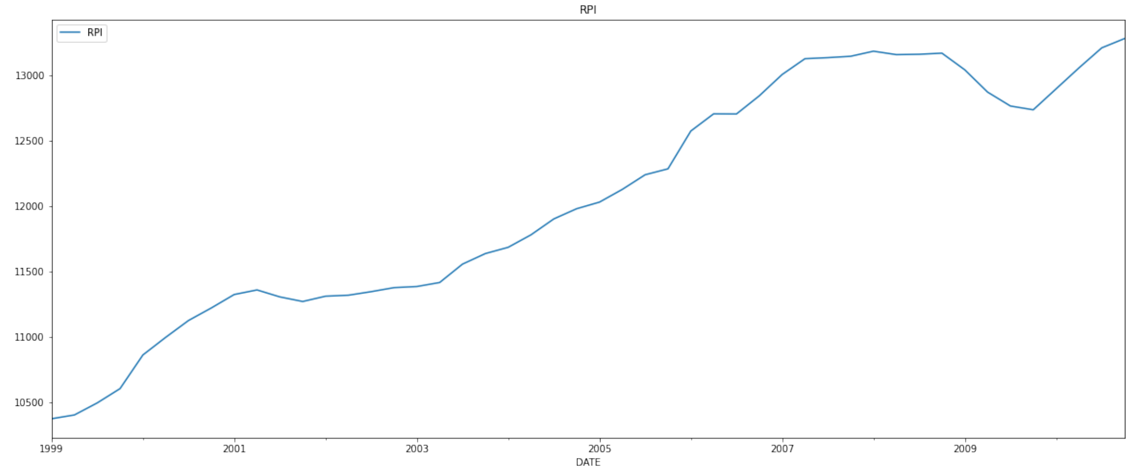
[734]:

```
                RPI
count     96.000000
mean   13936.528677
std     2314.591073
min    10373.294000
```

```
25%     12017.684500
50%     13405.872000
75%     15552.040000
max     19297.189000
```

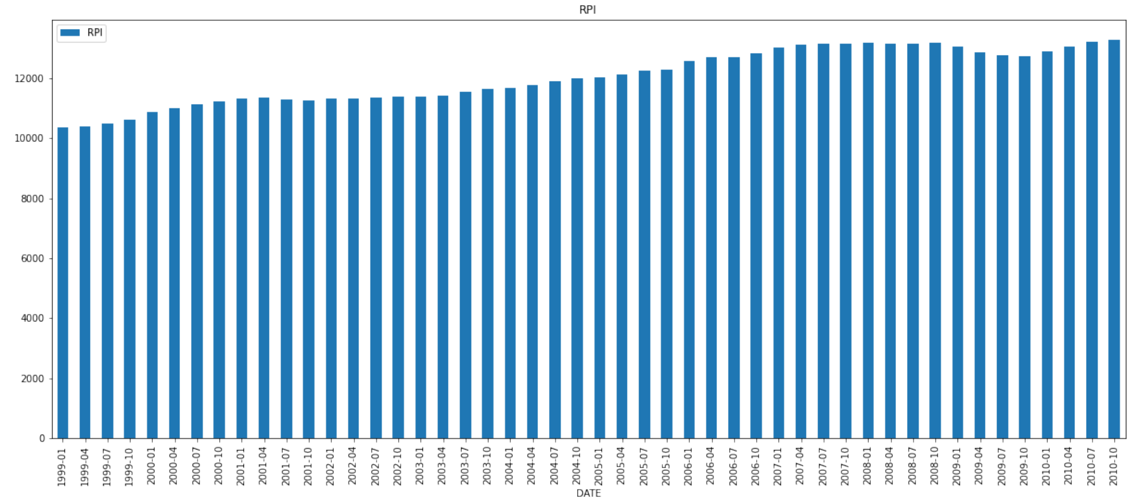
```
[735]: real_personal_income.loc['1999-01':'2010-10'].plot(title = 'RPI',␣
        ↪figsize=(20,8)) #line plot
```

```
[735]: <AxesSubplot: title={'center': 'RPI'}, xlabel='DATE'>
```



```
[736]: real_personal_income.loc['1999-01':'2010-10'].plot.bar(title = 'RPI',␣
        ↪figsize=(20,8)) #bar graph
```

```
[736]: <AxesSubplot: title={'center': 'RPI'}, xlabel='DATE'>
```



2.3 Employement Rate

[737]:
```
employment_rate.shape
```

[737]:
```
(2810, 8)
```

[738]:
```
employment_rate.head()
```

[738]:
```
  LOCATION INDICATOR SUBJECT    MEASURE FREQUENCY     TIME     Value  \
0      AUS       EMP     TOT  PC_WKGPOP         M  1981-07  65.34498
1      AUS       EMP     TOT  PC_WKGPOP         M  1981-08  65.38754
2      AUS       EMP     TOT  PC_WKGPOP         M  1981-09  65.50307
3      AUS       EMP     TOT  PC_WKGPOP         M  1981-10  65.19621
4      AUS       EMP     TOT  PC_WKGPOP         M  1981-11  65.00010

  Flag Codes
0        NaN
1        NaN
2        NaN
3        NaN
4        NaN
```

[739]:
```
employment_rate.columns
```

[739]:
```
Index(['LOCATION', 'INDICATOR', 'SUBJECT', 'MEASURE', 'FREQUENCY', 'TIME',
       'Value', 'Flag Codes'],
      dtype='object')
```

[740]:
```
employment_rate.nunique(axis=0)
```

[740]:
```
LOCATION         8
INDICATOR        1
SUBJECT          1
MEASURE          1
FREQUENCY        1
TIME           495
Value         2428
Flag Codes       1
dtype: int64
```

[741]:
```
employment_rate.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2810 entries, 0 to 2809
Data columns (total 8 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   LOCATION    2810 non-null   object
 1   INDICATOR   2810 non-null   object
 2   SUBJECT     2810 non-null   object
 3   MEASURE     2810 non-null   object
 4   FREQUENCY   2810 non-null   object
 5   TIME        2810 non-null   object
 6   Value       2810 non-null   float64
 7   Flag Codes  2 non-null      object
dtypes: float64(1), object(7)
memory usage: 175.8+ KB
```

[742]:
```
employment_rate['LOCATION'].unique() #identifying different locations in the␣
 ↪data set
```

[742]:
```
array(['AUS', 'CAN', 'JPN', 'KOR', 'USA', 'CHL', 'COL', 'RUS'],
      dtype=object)
```

[743]:
```
employment_rate['TIME'].unique()
```

[743]:
```
array(['1981-07', '1981-08', '1981-09', '1981-10', '1981-11', '1981-12',
       '1982-01', '1982-02', '1982-03', '1982-04', '1982-05', '1982-06',
       '1982-07', '1982-08', '1982-09', '1982-10', '1982-11', '1982-12',
       '1983-01', '1983-02', '1983-03', '1983-04', '1983-05', '1983-06',
       '1983-07', '1983-08', '1983-09', '1983-10', '1983-11', '1983-12',
       '1984-01', '1984-02', '1984-03', '1984-04', '1984-05', '1984-06',
       '1984-07', '1984-08', '1984-09', '1984-10', '1984-11', '1984-12',
       '1985-01', '1985-02', '1985-03', '1985-04', '1985-05', '1985-06',
       '1985-07', '1985-08', '1985-09', '1985-10', '1985-11', '1985-12',
       '1986-01', '1986-02', '1986-03', '1986-04', '1986-05', '1986-06',
       '1986-07', '1986-08', '1986-09', '1986-10', '1986-11', '1986-12',
       '1987-01', '1987-02', '1987-03', '1987-04', '1987-05', '1987-06',
       '1987-07', '1987-08', '1987-09', '1987-10', '1987-11', '1987-12',
       '1988-01', '1988-02', '1988-03', '1988-04', '1988-05', '1988-06',
       '1988-07', '1988-08', '1988-09', '1988-10', '1988-11', '1988-12',
       '1989-01', '1989-02', '1989-03', '1989-04', '1989-05', '1989-06',
       '1989-07', '1989-08', '1989-09', '1989-10', '1989-11', '1989-12',
       '1990-01', '1990-02', '1990-03', '1990-04', '1990-05', '1990-06',
       '1990-07', '1990-08', '1990-09', '1990-10', '1990-11', '1990-12',
       '1991-01', '1991-02', '1991-03', '1991-04', '1991-05', '1991-06',
       '1991-07', '1991-08', '1991-09', '1991-10', '1991-11', '1991-12',
       '1992-01', '1992-02', '1992-03', '1992-04', '1992-05', '1992-06',
       '1992-07', '1992-08', '1992-09', '1992-10', '1992-11', '1992-12',
       '1993-01', '1993-02', '1993-03', '1993-04', '1993-05', '1993-06',
       '1993-07', '1993-08', '1993-09', '1993-10', '1993-11', '1993-12',
       '1994-01', '1994-02', '1994-03', '1994-04', '1994-05', '1994-06',
       '1994-07', '1994-08', '1994-09', '1994-10', '1994-11', '1994-12',
       '1995-01', '1995-02', '1995-03', '1995-04', '1995-05', '1995-06',
       '1995-07', '1995-08', '1995-09', '1995-10', '1995-11', '1995-12',
       '1996-01', '1996-02', '1996-03', '1996-04', '1996-05', '1996-06',
       '1996-07', '1996-08', '1996-09', '1996-10', '1996-11', '1996-12',
       '1997-01', '1997-02', '1997-03', '1997-04', '1997-05', '1997-06',
       '1997-07', '1997-08', '1997-09', '1997-10', '1997-11', '1997-12',
       '1998-01', '1998-02', '1998-03', '1998-04', '1998-05', '1998-06',
       '1998-07', '1998-08', '1998-09', '1998-10', '1998-11', '1998-12',
       '1999-01', '1999-02', '1999-03', '1999-04', '1999-05', '1999-06',
       '1999-07', '1999-08', '1999-09', '1999-10', '1999-11', '1999-12',
       '2000-01', '2000-02', '2000-03', '2000-04', '2000-05', '2000-06',
       '2000-07', '2000-08', '2000-09', '2000-10', '2000-11', '2000-12',
       '2001-01', '2001-02', '2001-03', '2001-04', '2001-05', '2001-06',
       '2001-07', '2001-08', '2001-09', '2001-10', '2001-11', '2001-12',
       '2002-01', '2002-02', '2002-03', '2002-04', '2002-05', '2002-06',
       '2002-07', '2002-08', '2002-09', '2002-10', '2002-11', '2002-12',
       '2003-01', '2003-02', '2003-03', '2003-04', '2003-05', '2003-06',
       '2003-07', '2003-08', '2003-09', '2003-10', '2003-11', '2003-12',
       '2004-01', '2004-02', '2004-03', '2004-04', '2004-05', '2004-06',
       '2004-07', '2004-08', '2004-09', '2004-10', '2004-11', '2004-12',
       '2005-01', '2005-02', '2005-03', '2005-04', '2005-05', '2005-06',
       '2005-07', '2005-08', '2005-09', '2005-10', '2005-11', '2005-12',
       '2006-01', '2006-02', '2006-03', '2006-04', '2006-05', '2006-06',
       '2006-07', '2006-08', '2006-09', '2006-10', '2006-11', '2006-12',
       '2007-01', '2007-02', '2007-03', '2007-04', '2007-05', '2007-06',
       '2007-07', '2007-08', '2007-09', '2007-10', '2007-11', '2007-12',
       '2008-01', '2008-02', '2008-03', '2008-04', '2008-05', '2008-06',
       '2008-07', '2008-08', '2008-09', '2008-10', '2008-11', '2008-12',
       '2009-01', '2009-02', '2009-03', '2009-04', '2009-05', '2009-06',
       '2009-07', '2009-08', '2009-09', '2009-10', '2009-11', '2009-12',
       '2010-01', '2010-02', '2010-03', '2010-04', '2010-05', '2010-06',
       '2010-07', '2010-08', '2010-09', '2010-10', '2010-11', '2010-12',
       '2011-01', '2011-02', '2011-03', '2011-04', '2011-05', '2011-06',
       '2011-07', '2011-08', '2011-09', '2011-10', '2011-11', '2011-12',
       '2012-01', '2012-02', '2012-03', '2012-04', '2012-05', '2012-06',
       '2012-07', '2012-08', '2012-09', '2012-10', '2012-11', '2012-12',
       '2013-01', '2013-02', '2013-03', '2013-04', '2013-05', '2013-06',
       '2013-07', '2013-08', '2013-09', '2013-10', '2013-11', '2013-12',
       '2014-01', '2014-02', '2014-03', '2014-04', '2014-05', '2014-06',
       '2014-07', '2014-08', '2014-09', '2014-10', '2014-11', '2014-12',
       '2015-01', '2015-02', '2015-03', '2015-04', '2015-05', '2015-06',
       '2015-07', '2015-08', '2015-09', '2015-10', '2015-11', '2015-12',
       '2016-01', '2016-02', '2016-03', '2016-04', '2016-05', '2016-06',
       '2016-07', '2016-08', '2016-09', '2016-10', '2016-11', '2016-12',
       '2017-01', '2017-02', '2017-03', '2017-04', '2017-05', '2017-06',
       '2017-07', '2017-08', '2017-09', '2017-10', '2017-11', '2017-12',
       '2018-01', '2018-02', '2018-03', '2018-04', '2018-05', '2018-06',
       '2018-07', '2018-08', '2018-09', '2018-10', '2018-11', '2018-12',
       '2019-01', '2019-02', '2019-03', '2019-04', '2019-05', '2019-06',
       '2019-07', '2019-08', '2019-09', '2019-10', '2019-11', '2019-12',
       '2020-01', '2020-02', '2020-03', '2020-04', '2020-05', '2020-06',
       '2020-07', '2020-08', '2020-09', '2020-10', '2020-11', '2020-12',
       '2021-01', '2021-02', '2021-03', '2021-04', '2021-05', '2021-06',
       '2021-07', '2021-08', '2021-09', '2021-10', '2021-11', '2021-12',
       '2022-01', '2022-02', '2022-03', '2022-04', '2022-05', '2022-06',
       '2022-07', '2022-08', '2022-09'], dtype=object)
```

[744]:

```
#cleaning data set: we are only interested in the USA location from date 1999 -␣
 ↪2010 (1,4,7,10 months to match the time frame of our other data sets)
#Converting Time column to pd.to_datetime
#making a new dataframe with only columns needed date, employment, location

employment_rate2 = pd.DataFrame().assign(DATE = employment_rate['TIME'],␣
 ↪EMPLOYMENT = employment_rate['Value'], LOCATION =␣
 ↪employment_rate['LOCATION'])

employment_rate2 ['DATE']= pd.to_datetime(employment_rate2['DATE'])

employment_rate2 ['DATE']= employment_rate2['DATE'].dt.to_period('M')

employment_rate2 = employment_rate2[~(employment_rate2['DATE']<'1999-01')]

employment_rate2 = employment_rate2.query("LOCATION == 'USA' ")

employment_rate2 = employment_rate2.set_index('DATE', drop=True)

employment_rate2 = employment_rate2[employment_rate2.index.month.
 ↪isin([1,4,7,10])]

employment_rate2
```

[744]:

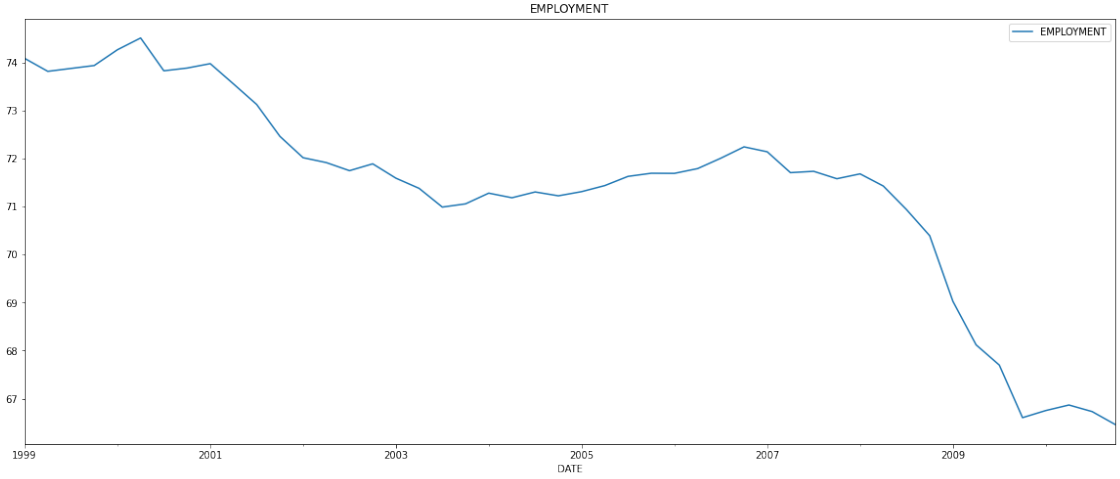
```
         EMPLOYMENT LOCATION
DATE
1999-01    74.08511      USA
1999-04    73.81467      USA
1999-07    73.87531      USA
1999-10    73.93589      USA
2000-01    74.26408      USA
…                …        …
2021-07    69.65714      USA
2021-10    70.16923      USA
2022-01    70.76403      USA
2022-04    71.26014      USA
2022-07    71.29388      USA

[95 rows x 2 columns]
```

```
[745]: employment_rate2.describe()
```

```
[745]:        EMPLOYMENT
       count   95.000000
       mean    70.082036
       std      2.496251
       min     60.253580
       25%     68.144985
       50%     70.860770
       75%     71.699315
       max     74.509090
```

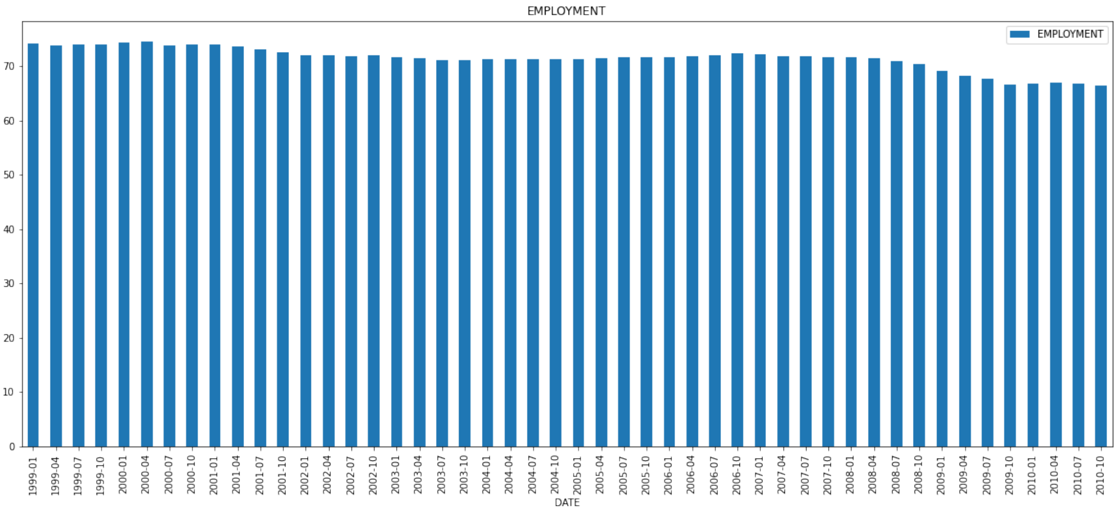
```
[746]: employment_rate2.loc['1999-01':'2010-10'].plot(title = 'EMPLOYMENT',␣
        ↪figsize=(20,8))
```

```
[746]: <AxesSubplot: title={'center': 'EMPLOYMENT'}, xlabel='DATE'>
```



```
[747]: employment_rate2.loc['1999-01':'2010-10'].plot.bar(title = 'EMPLOYMENT',␣
        ↪figsize=(20,8))
```

```
[747]: <AxesSubplot: title={'center': 'EMPLOYMENT'}, xlabel='DATE'>
```



2.4 Industrial Production

[748]:
```
industrial_productions.shape
```

[748]:
```
(12174, 8)
```

[749]:
```
industrial_productions.head()
```

[749]:
```
  LOCATION INDICATOR SUBJECT  MEASURE FREQUENCY     TIME     Value Flag Codes
0      AUT   INDPROD     MFG  IDX2015         M  1998-09  63.04453        NaN
1      AUT   INDPROD     MFG  IDX2015         M  1998-10  62.38003        NaN
2      AUT   INDPROD     MFG  IDX2015         M  1998-11  61.54940        NaN
3      AUT   INDPROD     MFG  IDX2015         M  1998-12  60.13734        NaN
4      AUT   INDPROD     MFG  IDX2015         M  1999-01  61.71553        NaN
```

[750]:
```
industrial_productions.columns
```

[750]:
```
Index(['LOCATION', 'INDICATOR', 'SUBJECT', 'MEASURE', 'FREQUENCY', 'TIME',
       'Value', 'Flag Codes'],
      dtype='object')
```

[751]:
```
industrial_productions.nunique(axis=0)
```

[751]:
```
LOCATION        43
INDICATOR        1
SUBJECT          1
MEASURE          1
FREQUENCY        1
TIME           290
Value         8079
```

```
Flag Codes       3
dtype: int64
```

[752]:

```
industrial_productions.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 12174 entries, 0 to 12173
Data columns (total 8 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   LOCATION    12174 non-null  object
 1   INDICATOR   12174 non-null  object
 2   SUBJECT     12174 non-null  object
 3   MEASURE     12174 non-null  object
 4   FREQUENCY   12174 non-null  object
 5   TIME        12174 non-null  object
 6   Value       12174 non-null  float64
 7   Flag Codes  514 non-null    object
dtypes: float64(1), object(7)
memory usage: 761.0+ KB
```

[753]:

```
industrial_productions['LOCATION'].unique()
```

[753]:

```
array(['AUT', 'BEL', 'CAN', 'CZE', 'DNK', 'FIN', 'FRA', 'DEU', 'GRC',
       'HUN', 'ISL', 'IRL', 'ITA', 'JPN', 'KOR', 'LUX', 'MEX', 'NLD',
       'NOR', 'POL', 'PRT', 'SVK', 'ESP', 'SWE', 'TUR', 'GBR', 'USA',
       'BRA', 'CHL', 'COL', 'EST', 'IND', 'IDN', 'ISR', 'LVA', 'LTU',
       'RUS', 'SVN', 'ZAF', 'EA19', 'CRI', 'EU27_2020', 'CHE'],
      dtype=object)
```

[754]:

```
industrial_productions['TIME'].unique()
```

[754]:

```
array(['1998-09', '1998-10', '1998-11', '1998-12', '1999-01', '1999-02',
       '1999-03', '1999-04', '1999-05', '1999-06', '1999-07', '1999-08',
       '1999-09', '1999-10', '1999-11', '1999-12', '2000-01', '2000-02',
       '2000-03', '2000-04', '2000-05', '2000-06', '2000-07', '2000-08',
       '2000-09', '2000-10', '2000-11', '2000-12', '2001-01', '2001-02',
       '2001-03', '2001-04', '2001-05', '2001-06', '2001-07', '2001-08',
       '2001-09', '2001-10', '2001-11', '2001-12', '2002-01', '2002-02',
       '2002-03', '2002-04', '2002-05', '2002-06', '2002-07', '2002-08',
       '2002-09', '2002-10', '2002-11', '2002-12', '2003-01', '2003-02',
       '2003-03', '2003-04', '2003-05', '2003-06', '2003-07', '2003-08',
       '2003-09', '2003-10', '2003-11', '2003-12', '2004-01', '2004-02',
       '2004-03', '2004-04', '2004-05', '2004-06', '2004-07', '2004-08',
       '2004-09', '2004-10', '2004-11', '2004-12', '2005-01', '2005-02',
       '2005-03', '2005-04', '2005-05', '2005-06', '2005-07', '2005-08',
       '2005-09', '2005-10', '2005-11', '2005-12', '2006-01', '2006-02',
       '2006-03', '2006-04', '2006-05', '2006-06', '2006-07', '2006-08',
       '2006-09', '2006-10', '2006-11', '2006-12', '2007-01', '2007-02',
       '2007-03', '2007-04', '2007-05', '2007-06', '2007-07', '2007-08',
       '2007-09', '2007-10', '2007-11', '2007-12', '2008-01', '2008-02',
       '2008-03', '2008-04', '2008-05', '2008-06', '2008-07', '2008-08',
       '2008-09', '2008-10', '2008-11', '2008-12', '2009-01', '2009-02',
       '2009-03', '2009-04', '2009-05', '2009-06', '2009-07', '2009-08',
       '2009-09', '2009-10', '2009-11', '2009-12', '2010-01', '2010-02',
       '2010-03', '2010-04', '2010-05', '2010-06', '2010-07', '2010-08',
       '2010-09', '2010-10', '2010-11', '2010-12', '2011-01', '2011-02',
       '2011-03', '2011-04', '2011-05', '2011-06', '2011-07', '2011-08',
       '2011-09', '2011-10', '2011-11', '2011-12', '2012-01', '2012-02',
       '2012-03', '2012-04', '2012-05', '2012-06', '2012-07', '2012-08',
       '2012-09', '2012-10', '2012-11', '2012-12', '2013-01', '2013-02',
       '2013-03', '2013-04', '2013-05', '2013-06', '2013-07', '2013-08',
       '2013-09', '2013-10', '2013-11', '2013-12', '2014-01', '2014-02',
       '2014-03', '2014-04', '2014-05', '2014-06', '2014-07', '2014-08',
       '2014-09', '2014-10', '2014-11', '2014-12', '2015-01', '2015-02',
       '2015-03', '2015-04', '2015-05', '2015-06', '2015-07', '2015-08',
       '2015-09', '2015-10', '2015-11', '2015-12', '2016-01', '2016-02',
       '2016-03', '2016-04', '2016-05', '2016-06', '2016-07', '2016-08',
       '2016-09', '2016-10', '2016-11', '2016-12', '2017-01', '2017-02',
       '2017-03', '2017-04', '2017-05', '2017-06', '2017-07', '2017-08',
       '2017-09', '2017-10', '2017-11', '2017-12', '2018-01', '2018-02',
       '2018-03', '2018-04', '2018-05', '2018-06', '2018-07', '2018-08',
       '2018-09', '2018-10', '2018-11', '2018-12', '2019-01', '2019-02',
       '2019-03', '2019-04', '2019-05', '2019-06', '2019-07', '2019-08',
       '2019-09', '2019-10', '2019-11', '2019-12', '2020-01', '2020-02',
       '2020-03', '2020-04', '2020-05', '2020-06', '2020-07', '2020-08',
       '2020-09', '2020-10', '2020-11', '2020-12', '2021-01', '2021-02',
       '2021-03', '2021-04', '2021-05', '2021-06', '2021-07', '2021-08',
       '2021-09', '2021-10', '2021-11', '2021-12', '2022-01', '2022-02',
       '2022-03', '2022-04', '2022-05', '2022-06', '2022-07', '2022-08',
       '2022-09', '2022-10'], dtype=object)
```

[755]:
```
industrial_productions['FREQUENCY'].unique()
```

[755]:
```
array(['M'], dtype=object)
```

[756]:
```
#cleaning data set: we are only interested in the USA location from date 1999 -␣
 ↪2010 (1,4,7,10 months to match the time frame of our other data sets)
#Converting Time column to pd.to_datetime
#making a new dataframe with only columns needed date, employment, location
```

```
industrial_productions2 = pd.DataFrame().assign(DATE =␣
 ↪industrial_productions['TIME'], INDUSTRIAL =␣
 ↪industrial_productions['Value'], LOCATION =␣
 ↪industrial_productions['LOCATION'])

industrial_productions2 ['DATE']= pd.
 ↪to_datetime(industrial_productions2['DATE'])

industrial_productions2 ['DATE']= industrial_productions2['DATE'].dt.
 ↪to_period('M')

industrial_productions2 =␣
 ↪industrial_productions2[~(industrial_productions2['DATE']<'1999-01')]

industrial_productions2 = industrial_productions2.query("LOCATION == 'USA' ")

industrial_productions2 = industrial_productions2.set_index('DATE', drop=True)

industrial_productions2 = industrial_productions2[industrial_productions2.index.
 ↪month.isin([1,4,7,10])]

industrial_productions2
```

[756]:

```
         INDUSTRIAL LOCATION
DATE
1999-01    90.90402      USA
1999-04    92.00492      USA
1999-07    92.89662      USA
1999-10    94.47752      USA
2000-01    95.80798      USA
…               …       …
2021-10    99.63799      USA
2022-01    99.79150      USA
2022-04   102.16460      USA
2022-07   101.57790      USA
2022-10   102.06880      USA

[96 rows x 2 columns]
```

[757]:

```
industrial_productions2.describe()
```

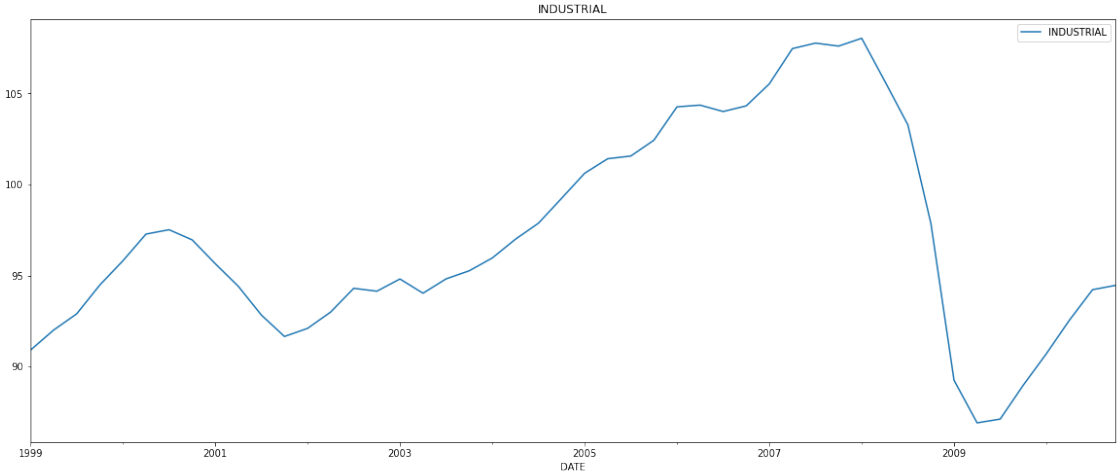
[757]:

```
       INDUSTRIAL
count   96.000000
mean    97.970442
std      4.750328
min     80.065860
25%     95.002418
```

```
50%       98.784810
75%      100.575950
max      108.025200
```

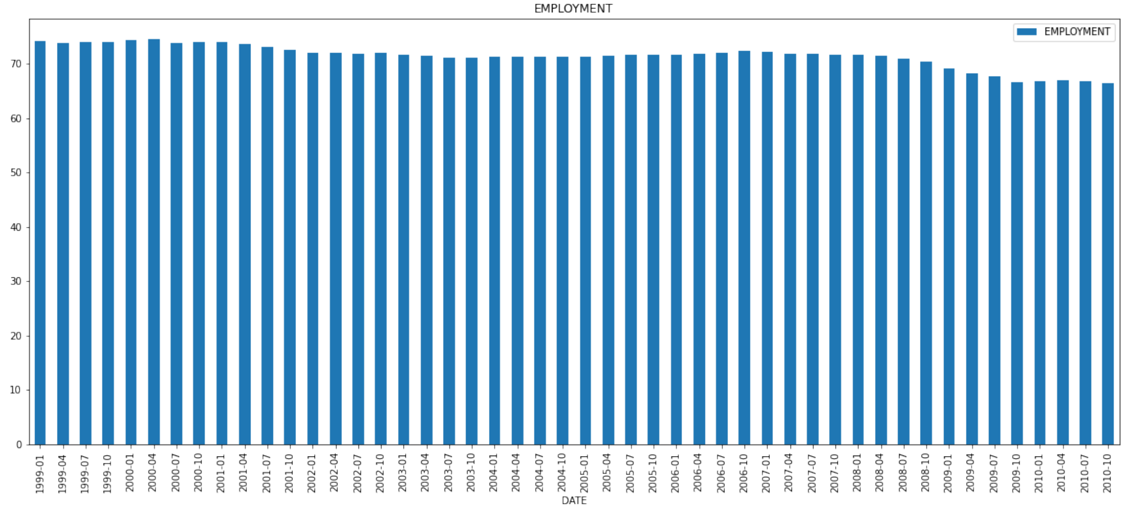
[758]:
```
industrial_productions2.loc['1999-01':'2010-10'].plot(title = 'INDUSTRIAL',␣
 ↪figsize=(20,8))
```

[758]:
```
<AxesSubplot: title={'center': 'INDUSTRIAL'}, xlabel='DATE'>
```



[759]:
```
employment_rate2.loc['1999-01':'2010-10'].plot.bar(title = 'EMPLOYMENT',␣
 ↪figsize=(20,8))
```

[759]:
```
<AxesSubplot: title={'center': 'EMPLOYMENT'}, xlabel='DATE'>
```



3 Observation

After graphing all of the recession indicators, I noticed that the GDP and Real Personal Income did not perform as expected during the 2001 recession in comparison to 2009. What prompted the economists to call a recession ? Observing industrial production and employement rate, we do see a significant dip in the 2001 and 2009 recession period. Could two indicators be enough to call a recession? A more in depth research is needed to fully understand what happended in 2001 that put the economy in a declining state; even more so understanding the factors of 2009 where we see the biggest percentage change in all of the data sets.

4 CPI: Consumer Price Index

The CPI tells us how much a certain cost or good has changed over time For the purpose of this analysis we will be looking at:

1- Housing Cost Index

2- Inflation Rate

3- Rent of Primary Residence

4- Purchasing Power of the Consumer Dollar

[760]:

```
# reading and pulling the datasets as dataframes

inflation_rate = pd.read_csv('CPI INFLATION RATE.xlsx - BLS Data Series.csv')
purchasing_power = pd.read_csv('CPI Purchasing power of the consumer dollar in␣
 ↪US.xlsx - BLS Data Series.csv')
housing_cost_index = pd.read_csv('CPI Housing Cost.xlsx - BLS Data Series.csv')
rent_primary_index = pd.read_csv('CPI OF RENT OF PRIMARY RESIDENCE.xlsx - BLS␣
 ↪Data Series.csv')
```

4.1 Inflation Rate

[761]:

```
inflation_rate.shape
```

[761]:

```
(23, 15)
```

[762]:

```
inflation_rate.head()
```

[762]:

```
   Year  Jan  Feb  Mar  Apr  May  Jun  Jul  Aug  Sep  Oct  Nov  Dec  HALF1  \
0  2000  0.3  0.4  0.6 -0.1  0.2  0.6  0.3  0.0  0.5  0.2  0.2  0.2    NaN
1  2001  0.6  0.2  0.1  0.2  0.5  0.2 -0.2  0.0  0.4 -0.3 -0.1 -0.1    NaN
2  2002  0.2  0.2  0.3  0.4  0.1  0.1  0.2  0.3  0.2  0.2  0.2  0.2    NaN
3  2003  0.4  0.5  0.2 -0.4 -0.2  0.1  0.3  0.4  0.3 -0.1  0.1  0.3    NaN
4  2004  0.4  0.2  0.2  0.2  0.4  0.4  0.1  0.1  0.3  0.5  0.5  0.0    NaN

   HALF2
0    NaN
```

```
1     NaN
2     NaN
3     NaN
4     NaN
```

```
[763]: inflation_rate.columns
```

```
[763]: Index(['Year', 'Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep',
              'Oct', 'Nov', 'Dec', 'HALF1', 'HALF2'],
             dtype='object')
```

```
[764]: inflation_rate.nunique(axis=0)
```

```
[764]: Year     23
       Jan       8
       Feb       7
       Mar      10
       Apr      11
       May      11
       Jun      12
       Jul       9
       Aug       7
       Sep       9
       Oct      11
       Nov      11
       Dec      10
       HALF1     0
       HALF2     0
       dtype: int64
```

```
[765]: inflation_rate.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 23 entries, 0 to 22
Data columns (total 15 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Year    23 non-null     int64
 1   Jan     23 non-null     float64
 2   Feb     23 non-null     float64
 3   Mar     23 non-null     float64
 4   Apr     23 non-null     float64
 5   May     23 non-null     float64
 6   Jun     23 non-null     float64
 7   Jul     23 non-null     float64
 8   Aug     23 non-null     float64
 9   Sep     23 non-null     float64
 10  Oct     23 non-null     float64
 11  Nov     22 non-null     float64
 12  Dec     22 non-null     float64
 13  HALF1   0 non-null      float64
 14  HALF2   0 non-null      float64
dtypes: float64(14), int64(1)
memory usage: 2.8 KB
```

[766]:

```
inflation_rate = inflation_rate.drop(["HALF1", "HALF2"], axis=1)
inflation_rate
```

[766]:

```
    Year  Jan  Feb  Mar  Apr  May  Jun  Jul  Aug  Sep  Oct  Nov  Dec
0   2000  0.3  0.4  0.6 -0.1  0.2  0.6  0.3  0.0  0.5  0.2  0.2  0.2
1   2001  0.6  0.2  0.1  0.2  0.5  0.2 -0.2  0.0  0.4 -0.3 -0.1 -0.1
2   2002  0.2  0.2  0.3  0.4  0.1  0.1  0.2  0.3  0.2  0.2  0.2  0.2
3   2003  0.4  0.5  0.2 -0.4 -0.2  0.1  0.3  0.4  0.3 -0.1  0.1  0.3
4   2004  0.4  0.2  0.2  0.2  0.4  0.4  0.1  0.1  0.3  0.5  0.5  0.0
5   2005 -0.1  0.4  0.4  0.3 -0.1  0.1  0.6  0.6  1.4  0.2 -0.5  0.0
6   2006  0.6  0.1  0.2  0.5  0.3  0.2  0.5  0.4 -0.5 -0.4  0.0  0.5
7   2007  0.2  0.4  0.5  0.3  0.4  0.2  0.2  0.0  0.4  0.3  0.8  0.3
8   2008  0.3  0.2  0.4  0.2  0.6  1.0  0.7 -0.1  0.1 -0.9 -1.8 -0.8
9   2009  0.3  0.4 -0.1  0.1  0.1  0.8  0.0  0.3  0.2  0.3  0.3  0.1
10  2010  0.1 -0.1  0.0  0.0 -0.1  0.0  0.2  0.1  0.2  0.3  0.3  0.4
11  2011  0.3  0.3  0.5  0.5  0.3  0.0  0.3  0.3  0.2  0.1  0.2  0.0
12  2012  0.3  0.2  0.2  0.2 -0.2 -0.1  0.0  0.6  0.5  0.3 -0.2  0.0
13  2013  0.2  0.5 -0.3 -0.2  0.0  0.2  0.2  0.2  0.0  0.1  0.2  0.3
14  2014  0.2  0.1  0.2  0.2  0.2  0.1  0.1  0.0  0.0  0.0 -0.2 -0.3
15  2015 -0.6  0.3  0.3  0.1  0.3  0.3  0.2  0.0 -0.2  0.1  0.1 -0.1
16  2016  0.0 -0.1  0.3  0.4  0.2  0.3 -0.1  0.2  0.3  0.2  0.1  0.3
17  2017  0.4  0.2  0.0  0.1 -0.1  0.1  0.0  0.4  0.5  0.1  0.3  0.2
18  2018  0.4  0.3  0.1  0.2  0.3  0.1  0.1  0.2  0.2  0.2 -0.1  0.0
19  2019  0.0  0.3  0.4  0.4  0.1  0.0  0.2  0.1  0.2  0.3  0.2  0.2
20  2020  0.2  0.1 -0.3 -0.8 -0.1  0.5  0.5  0.4  0.2  0.1  0.1  0.3
21  2021  0.2  0.4  0.6  0.6  0.7  0.9  0.5  0.3  0.4  0.9  0.7  0.6
22  2022  0.6  0.8  1.2  0.3  1.0  1.3  0.0  0.1  0.4  0.4  NaN  NaN
```

[767]:

```
inflation_rate = inflation_rate.melt(id_vars=["Year"], var_name="Month")
inflation_rate
```

[767]:

```
     Year Month  value
0    2000   Jan    0.3
1    2001   Jan    0.6
2    2002   Jan    0.2
3    2003   Jan    0.4
4    2004   Jan    0.4
..    …     …     …
271  2018   Dec    0.0
```

```
272  2019   Dec    0.2
273  2020   Dec    0.3
274  2021   Dec    0.6
275  2022   Dec    NaN

[276 rows x 3 columns]
```

[768]:
```
inflation_rate['Year'] = inflation_rate['Year'].astype(str)
inflation_rate['Month'] = inflation_rate['Month'].astype(str)
inflation_rate
```

[768]:
```
     Year Month  value
0    2000   Jan    0.3
1    2001   Jan    0.6
2    2002   Jan    0.2
3    2003   Jan    0.4
4    2004   Jan    0.4
..    …    …     …
271  2018   Dec    0.0
272  2019   Dec    0.2
273  2020   Dec    0.3
274  2021   Dec    0.6
275  2022   Dec    NaN

[276 rows x 3 columns]
```

[769]:
```
month_dict = {
    "Jan": "January",
    "Feb": "February",
    "Mar": "March",
    "Apr":"April",
    "May":"May",
    "Jun":"June",
    "Jul":"July",
    "Aug":"August",
    "Sep":"September",
    "Oct":"October",
    "Nov":"November",
    "Dec":"December",
}
inflation_rate['Month'] = inflation_rate['Month'].replace(month_dict)
inflation_rate
```

[769]:
```
     Year     Month  value
0    2000   January    0.3
1    2001   January    0.6
2    2002   January    0.2
```

```
3    2003   January    0.4
4    2004   January    0.4
..    …        …    …
271  2018  December    0.0
272  2019  December    0.2
273  2020  December    0.3
274  2021  December    0.6
275  2022  December    NaN

[276 rows x 3 columns]
```

[770]:
```
inflation_rate['Date'] = inflation_rate ['Year'] + " "+ inflation_rate['Month']
inflation_rate
```

[770]:
```
     Year     Month  value           Date
0    2000   January    0.3   2000 January
1    2001   January    0.6   2001 January
2    2002   January    0.2   2002 January
3    2003   January    0.4   2003 January
4    2004   January    0.4   2004 January
..    …        …    …             …
271  2018  December    0.0  2018 December
272  2019  December    0.2  2019 December
273  2020  December    0.3  2020 December
274  2021  December    0.6  2021 December
275  2022  December    NaN  2022 December

[276 rows x 4 columns]
```

[771]:
```
inflation_rate['Date'] = pd.to_datetime(inflation_rate['Date'])
inflation_rate['Date'] = inflation_rate['Date'].dt.to_period('M')
inflation_rate
```

[771]:
```
     Year     Month  value     Date
0    2000   January    0.3  2000-01
1    2001   January    0.6  2001-01
2    2002   January    0.2  2002-01
3    2003   January    0.4  2003-01
4    2004   January    0.4  2004-01
..    …        …    …       …
271  2018  December    0.0  2018-12
272  2019  December    0.2  2019-12
273  2020  December    0.3  2020-12
274  2021  December    0.6  2021-12
275  2022  December    NaN  2022-12

[276 rows x 4 columns]
```

[772]:

```
#inflation_rate = inflation_rate.set_index('Date', append = True)
#inflation_rate
inflation_rate.columns
```

[772]:

```
Index(['Year', 'Month', 'value', 'Date'], dtype='object')
```

[773]:

```
inflation_rate = inflation_rate.drop(["Year", "Month"], axis=1)
inflation_rate = inflation_rate.set_index('Date')
inflation_rate = inflation_rate.rename(columns ={'value': 'Inflation'})
inflation_rate = inflation_rate.sort_index()
```

[774]:

```
inflation_rate
```

[774]:

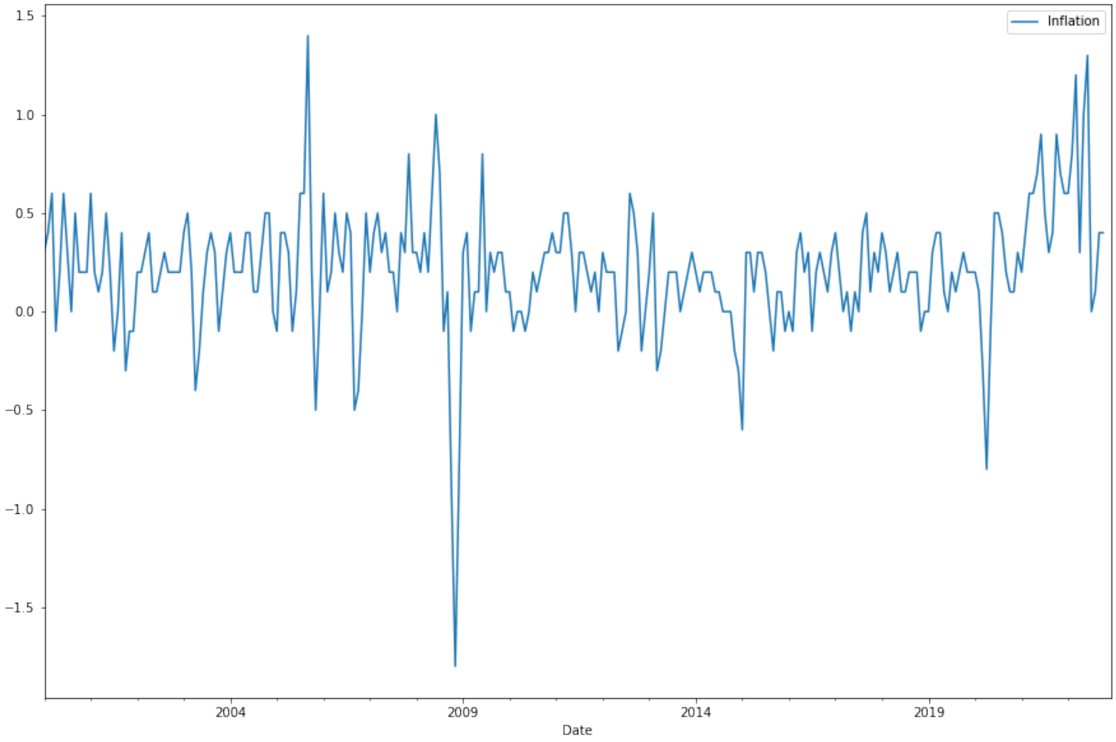
```
         Inflation
Date
2000-01        0.3
2000-02        0.4
2000-03        0.6
2000-04       -0.1
2000-05        0.2
…             …
2022-08        0.1
2022-09        0.4
2022-10        0.4
2022-11        NaN
2022-12        NaN

[276 rows x 1 columns]
```

[775]:

```
inflation_rate.plot.line(figsize=(15, 10))
```

[775]:

```
<AxesSubplot: xlabel='Date'>
```



4.2 Purchasing Power Of The Consumer

[776]:
```
purchasing_power.shape
```

[776]:
```
(23, 15)
```

[777]:
```
purchasing_power.head()
```

[777]:
```
   Year   Jan   Feb   Mar   Apr   May   Jun   Jul   Aug   Sep   Oct   Nov  \
0  2000  59.2  58.9  58.4  58.4  58.3  58.0  57.9  57.9  57.6  57.5  57.4
1  2001  57.1  56.9  56.7  56.5  56.3  56.2  56.3  56.3  56.1  56.3  56.4
2  2002  56.5  56.2  55.9  55.6  55.6  55.6  55.5  55.4  55.3  55.2  55.2
3  2003  55.0  54.6  54.3  54.4  54.5  54.4  54.4  54.2  54.0  54.0  54.2
4  2004  54.0  53.7  53.4  53.2  52.9  52.7  52.8  52.8  52.7  52.4  52.4

    Dec  HALF1  HALF2
0  57.5    NaN    NaN
1  56.6    NaN    NaN
2  55.3    NaN    NaN
3  54.3    NaN    NaN
4  52.5    NaN    NaN
```

[778]:
```
purchasing_power.columns
```

```
[778]: Index(['Year', 'Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep',
              'Oct', 'Nov', 'Dec', 'HALF1', 'HALF2'],
             dtype='object')
```

```
[779]: purchasing_power.nunique(axis=0)
```

```
[779]: Year     23
       Jan      21
       Feb      22
       Mar      23
       Apr      23
       May      22
       Jun      23
       Jul      23
       Aug      22
       Sep      22
       Oct      23
       Nov      22
       Dec      21
       HALF1     0
       HALF2     0
       dtype: int64
```

```
[780]: purchasing_power.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 23 entries, 0 to 22
Data columns (total 15 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Year    23 non-null     int64
 1   Jan     23 non-null     float64
 2   Feb     23 non-null     float64
 3   Mar     23 non-null     float64
 4   Apr     23 non-null     float64
 5   May     23 non-null     float64
 6   Jun     23 non-null     float64
 7   Jul     23 non-null     float64
 8   Aug     23 non-null     float64
 9   Sep     23 non-null     float64
 10  Oct     23 non-null     float64
 11  Nov     22 non-null     float64
 12  Dec     22 non-null     float64
 13  HALF1   0 non-null      float64
 14  HALF2   0 non-null      float64
dtypes: float64(14), int64(1)
memory usage: 2.8 KB
```

```
[781]: purchasing_power = purchasing_power.drop(["HALF1", "HALF2"], axis=1)
       purchasing_power
```

```
[781]:     Year   Jan   Feb   Mar   Apr   May   Jun   Jul   Aug   Sep   Oct   Nov  \
       0   2000  59.2  58.9  58.4  58.4  58.3  58.0  57.9  57.9  57.6  57.5  57.4
       1   2001  57.1  56.9  56.7  56.5  56.3  56.2  56.3  56.3  56.1  56.3  56.4
       2   2002  56.5  56.2  55.9  55.6  55.6  55.6  55.5  55.4  55.3  55.2  55.2
       3   2003  55.0  54.6  54.3  54.4  54.5  54.4  54.4  54.2  54.0  54.0  54.2
       4   2004  54.0  53.7  53.4  53.2  52.9  52.7  52.8  52.8  52.7  52.4  52.4
       5   2005  52.4  52.1  51.7  51.4  51.4  51.4  51.2  50.9  50.3  50.2  50.6
       6   2006  50.4  50.3  50.0  49.6  49.4  49.3  49.1  49.0  49.3  49.5  49.6
       7   2007  49.4  49.1  48.7  48.4  48.1  48.0  48.0  48.1  48.0  47.9  47.6
       8   2008  47.4  47.2  46.8  46.5  46.2  45.7  45.5  45.6  45.7  46.2  47.1
       9   2009  47.4  47.1  47.0  46.9  46.8  46.4  46.4  46.3  46.3  46.3  46.2
       10  2010  46.1  46.1  45.9  45.9  45.8  45.9  45.9  45.8  45.8  45.7  45.7
       11  2011  45.4  45.2  44.7  44.5  44.3  44.3  44.3  44.1  44.1  44.2  44.2
       12  2012  44.1  43.9  43.6  43.5  43.5  43.6  43.6  43.4  43.2  43.2  43.4
       13  2013  43.4  43.1  43.0  43.0  42.9  42.8  42.8  42.8  42.7  42.8  42.9
       14  2014  42.8  42.6  42.3  42.2  42.0  42.0  42.0  42.0  42.0  42.1  42.3
       15  2015  42.8  42.6  42.4  42.3  42.1  41.9  41.9  42.0  42.0  42.0  42.1
       16  2016  42.2  42.2  42.0  41.8  41.6  41.5  41.6  41.5  41.4  41.4  41.4
       17  2017  41.2  41.1  41.0  40.9  40.9  40.8  40.9  40.7  40.5  40.5  40.5
       18  2018  40.3  40.2  40.1  39.9  39.7  39.7  39.7  39.7  39.6  39.5  39.7
       19  2019  39.7  39.6  39.3  39.1  39.0  39.0  39.0  39.0  38.9  38.9  38.9
       20  2020  38.8  38.7  38.7  39.0  39.0  38.8  38.6  38.5  38.4  38.4  38.4
       21  2021  38.2  38.0  37.8  37.4  37.1  36.8  36.6  36.6  36.5  36.2  36.0
       22  2022  35.6  35.2  34.8  34.6  34.2  33.7  33.8  33.8  33.7  33.6   NaN

            Dec
       0   57.5
       1   56.6
       2   55.3
       3   54.3
       4   52.5
       5   50.8
       6   49.6
       7   47.6
       8   47.6
       9   46.3
       10  45.6
       11  44.3
       12  43.6
       13  42.9
       14  42.6
       15  42.3
       16  41.4
       17  40.6
```

```
18   39.8
19   38.9
20   38.4
21   35.9
22    NaN
```

[782]:
```
purchasing_power = purchasing_power.melt(id_vars=["Year"], var_name="Month")
purchasing_power['Year'] = purchasing_power['Year'].astype(str)
purchasing_power['Month'] = purchasing_power['Month'].astype(str)
purchasing_power
```

[782]:
```
     Year Month  value
0    2000   Jan   59.2
1    2001   Jan   57.1
2    2002   Jan   56.5
3    2003   Jan   55.0
4    2004   Jan   54.0
..    …    …     …
271  2018   Dec   39.8
272  2019   Dec   38.9
273  2020   Dec   38.4
274  2021   Dec   35.9
275  2022   Dec    NaN

[276 rows x 3 columns]
```

[783]:
```
month_dict1 = {
    "Jan": "January",
    "Feb": "February",
    "Mar": "March",
    "Apr":"April",
    "May":"May",
    "Jun":"June",
    "Jul":"July",
    "Aug":"August",
    "Sep":"September",
    "Oct":"October",
    "Nov":"November",
    "Dec":"December",
}
purchasing_power['Month'] = purchasing_power['Month'].replace(month_dict1)
purchasing_power['Date'] = purchasing_power ['Year'] + " "+␣
 ↪purchasing_power['Month']
purchasing_power
```

[783]:
```
     Year     Month  value          Date
0    2000   January   59.2  2000 January
```

```
1    2001    January   57.1   2001  January
2    2002    January   56.5   2002  January
3    2003    January   55.0   2003  January
4    2004    January   54.0   2004  January
..    …          …     …           …
271  2018   December   39.8  2018  December
272  2019   December   38.9  2019  December
273  2020   December   38.4  2020  December
274  2021   December   35.9  2021  December
275  2022   December    NaN  2022  December

[276 rows x 4 columns]
```

[784]:

```
purchasing_power['Date'] = pd.to_datetime(purchasing_power['Date'])
purchasing_power['Date'] = purchasing_power['Date'].dt.to_period('M')
purchasing_power
```

[784]:

```
     Year      Month  value     Date
0    2000    January   59.2  2000-01
1    2001    January   57.1  2001-01
2    2002    January   56.5  2002-01
3    2003    January   55.0  2003-01
4    2004    January   54.0  2004-01
..    …          …     …       …
271  2018   December   39.8  2018-12
272  2019   December   38.9  2019-12
273  2020   December   38.4  2020-12
274  2021   December   35.9  2021-12
275  2022   December    NaN  2022-12

[276 rows x 4 columns]
```

[785]:

```
purchasing_power = purchasing_power.drop(["Year", "Month"], axis=1)
purchasing_power = purchasing_power.set_index('Date')
purchasing_power = purchasing_power.rename(columns ={'value': 'Purchasing␣
 ↪Power'})
purchasing_power = purchasing_power.sort_index()
purchasing_power
```

[785]:

```
         Purchasing Power
Date
2000-01              59.2
2000-02              58.9
2000-03              58.4
2000-04              58.4
2000-05              58.3
…                    …
```

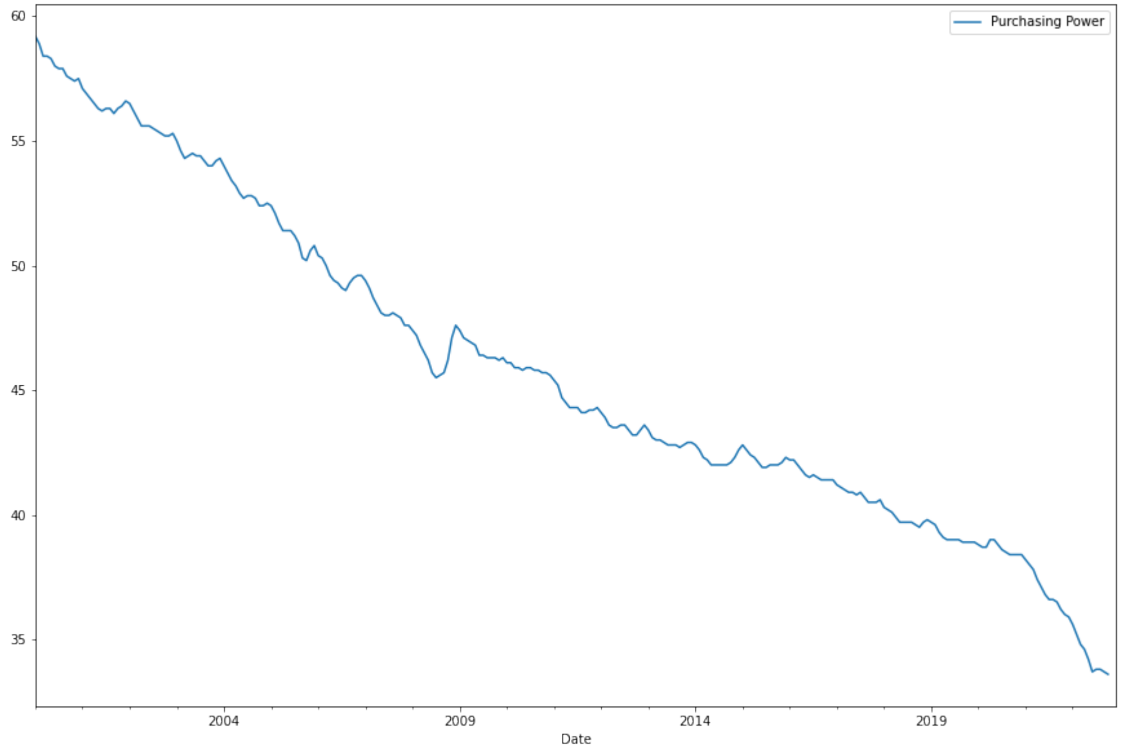
```
2022-08               33.8
2022-09               33.7
2022-10               33.6
2022-11                NaN
2022-12                NaN

[276 rows x 1 columns]
```

[786]:

```
purchasing_power.plot.line(figsize=(15, 10))
```

[786]:

```
<AxesSubplot: xlabel='Date'>
```



4.3 Housing Cost Index

[787]:

```
housing_cost_index.shape
```

[787]:

```
(23, 15)
```

[788]:

```
housing_cost_index.head()
```

[788]:

```
   Year    Jan    Feb    Mar    Apr    May    Jun    Jul    Aug    Sep    Oct  \
0  2000  166.0  167.1  167.8  167.9  168.1  169.6  170.6  170.9  171.4  171.7
1  2001  174.1  174.7  175.4  175.4  175.9  177.3  177.6  178.0  177.4  176.7
```

```
2  2002  177.6  178.5  179.1  179.5  179.7  180.7  181.2  181.7  181.5  181.4
3  2003  182.3  183.2  184.3  184.1  184.5  185.3  185.9  186.1  185.8  185.7
4  2004  186.3  187.0  187.9  188.4  188.9  190.3  190.9  191.2  191.0  191.0

     Nov    Dec  HALF1  HALF2
0  171.6  171.9  167.8  171.4
1  176.9  176.9  175.5  177.3
2  181.2  181.1  179.2  181.4
3  185.1  185.1  184.0  185.6
4  190.8  190.7  188.1  190.9
```

[789]:

```
housing_cost_index.columns
```

[789]:

```
Index(['Year', 'Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep',
       'Oct', 'Nov', 'Dec', 'HALF1', 'HALF2'],
      dtype='object')
```

[790]:

```
housing_cost_index.nunique(axis=0)
```

[790]:

```
Year     23
Jan      23
Feb      23
Mar      23
Apr      23
May      23
Jun      23
Jul      23
Aug      23
Sep      23
Oct      23
Nov      22
Dec      22
HALF1    23
HALF2    22
dtype: int64
```

[791]:

```
housing_cost_index.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 23 entries, 0 to 22
Data columns (total 15 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Year    23 non-null     int64
 1   Jan     23 non-null     float64
 2   Feb     23 non-null     float64
 3   Mar     23 non-null     float64
 4   Apr     23 non-null     float64
 5   May     23 non-null     float64
 6   Jun     23 non-null     float64
 7   Jul     23 non-null     float64
 8   Aug     23 non-null     float64
 9   Sep     23 non-null     float64
 10  Oct     23 non-null     float64
 11  Nov     22 non-null     float64
 12  Dec     22 non-null     float64
 13  HALF1   23 non-null     float64
 14  HALF2   22 non-null     float64
dtypes: float64(14), int64(1)
memory usage: 2.8 KB
```

[792]:

```
housing_cost_index =housing_cost_index.drop(["HALF1", "HALF2"], axis=1)
housing_cost_index
```

[792]:

```
    Year      Jan      Feb      Mar      Apr      May      Jun      Jul  \
0   2000  166.000  167.100  167.800  167.900  168.100  169.600  170.600
1   2001  174.100  174.700  175.400  175.400  175.900  177.300  177.600
2   2002  177.600  178.500  179.100  179.500  179.700  180.700  181.200
3   2003  182.300  183.200  184.300  184.100  184.500  185.300  185.900
4   2004  186.300  187.000  187.900  188.400  188.900  190.300  190.900
5   2005  191.800  192.700  194.100  194.400  194.500  195.500  196.600
6   2006  200.000  200.500  201.300  201.700  202.200  203.700  204.700
7   2007  206.057  207.177  208.080  208.541  208.902  210.649  211.286
8   2008  212.244  213.026  214.389  214.890  215.809  217.941  219.610
9   2009  216.928  217.180  217.374  217.126  216.971  218.071  218.085
10  2010  215.925  215.841  216.023  215.798  215.981  216.778  217.076
11  2011  216.739  217.259  217.707  217.901  218.484  219.553  220.230
12  2012  220.805  221.117  221.487  221.682  221.971  223.051  223.316
13  2013  224.790  225.382  225.643  225.986  226.896  228.068  228.374
14  2014  230.256  230.905  231.968  231.689  232.744  233.894  234.475
15  2015  235.485  236.016  236.435  236.777  237.175  238.568  239.085
16  2016  240.424  241.015  241.485  241.790  242.811  244.280  244.936
17  2017  247.942  248.693  248.978  249.514  250.376  251.629  251.870
18  2018  254.857  255.713  256.388  256.969  257.907  258.710  259.268
19  2019  262.284  263.057  263.886  264.452  265.137  266.461  267.101
20  2020  269.468  270.281  270.273  270.184  270.823  271.831  272.445
21  2021  274.336  275.137  276.028  277.258  278.648  280.366  281.604
22  2022  289.889  291.504  293.577  295.259  297.868  300.927  302.327

        Aug      Sep      Oct      Nov      Dec
0   170.900  171.400  171.700  171.600  171.900
1   178.000  177.400  176.700  176.900  176.900
2   181.700  181.500  181.400  181.200  181.100
3   186.100  185.800  185.700  185.100  185.100
```

```
4   191.200  191.000  191.000  190.800  190.700
5   196.900  197.000  198.400  198.500  198.300
6   205.100  205.000  204.400  204.500  204.800
7   211.098  210.865  210.701  210.745  210.933
8   219.148  218.184  217.383  216.467  216.073
9   217.827  217.178  216.612  215.808  215.523
10  216.976  216.602  216.100  215.830  216.142
11  220.506  220.540  220.138  219.969  220.193
12  223.699  223.901  223.708  223.814  224.032
13  228.564  228.808  228.362  228.449  228.892
14  234.571  234.675  234.434  234.315  234.658
15  239.298  239.651  239.395  239.325  239.514
16  245.472  246.127  246.264  246.271  246.795
17  252.615  252.984  253.125  253.177  253.845
18  259.884  259.941  260.268  260.473  261.360
19  267.263  267.822  267.794  267.925  268.236
20  272.866  273.116  273.014  273.290  273.684
21  282.391  283.744  285.310  286.308  287.511
22  304.506  306.521  307.816      NaN      NaN
```

[793]:

```
housing_cost_index = housing_cost_index.melt(id_vars=["Year"], var_name="Month")
housing_cost_index['Year'] = housing_cost_index['Year'].astype(str)
housing_cost_index['Month'] = housing_cost_index['Month'].astype(str)
housing_cost_index
```

[793]:

```
     Year Month    value
0    2000   Jan  166.000
1    2001   Jan  174.100
2    2002   Jan  177.600
3    2003   Jan  182.300
4    2004   Jan  186.300
..    …    …       …
271  2018   Dec  261.360
272  2019   Dec  268.236
273  2020   Dec  273.684
274  2021   Dec  287.511
275  2022   Dec      NaN

[276 rows x 3 columns]
```

[794]:

```
month_dict2 = {
    "Jan": "January",
    "Feb": "February",
    "Mar": "March",
    "Apr":"April",
    "May":"May",
    "Jun":"June",
    "Jul":"July",
    "Aug":"August",
    "Sep":"September",
    "Oct":"October",
    "Nov":"November",
    "Dec":"December",
}
housing_cost_index['Month'] = housing_cost_index['Month'].replace(month_dict2)
housing_cost_index['Date'] = housing_cost_index ['Year'] + " "+␣
 ↪housing_cost_index['Month']
housing_cost_index
```

[794]:

```
     Year     Month    value           Date
0    2000   January  166.000   2000 January
1    2001   January  174.100   2001 January
2    2002   January  177.600   2002 January
3    2003   January  182.300   2003 January
4    2004   January  186.300   2004 January
..    …        …        …           …
271  2018  December  261.360  2018 December
272  2019  December  268.236  2019 December
273  2020  December  273.684  2020 December
274  2021  December  287.511  2021 December
275  2022  December      NaN  2022 December

[276 rows x 4 columns]
```

[795]:

```
housing_cost_index['Date'] = pd.to_datetime(housing_cost_index['Date'])
housing_cost_index['Date'] = housing_cost_index['Date'].dt.to_period('M')
housing_cost_index
```

[795]:

```
     Year     Month    value     Date
0    2000   January  166.000  2000-01
1    2001   January  174.100  2001-01
2    2002   January  177.600  2002-01
3    2003   January  182.300  2003-01
4    2004   January  186.300  2004-01
..    …        …        …       …
271  2018  December  261.360  2018-12
272  2019  December  268.236  2019-12
273  2020  December  273.684  2020-12
274  2021  December  287.511  2021-12
275  2022  December      NaN  2022-12

[276 rows x 4 columns]
```

[796]:

```
housing_cost_index = housing_cost_index.drop(["Year", "Month"], axis=1)
housing_cost_index = housing_cost_index.set_index('Date')
housing_cost_index = housing_cost_index.rename(columns ={'value': 'Housing Cost␣
 ↪Index'})
housing_cost_index= housing_cost_index.sort_index()
housing_cost_index
```

[796]:

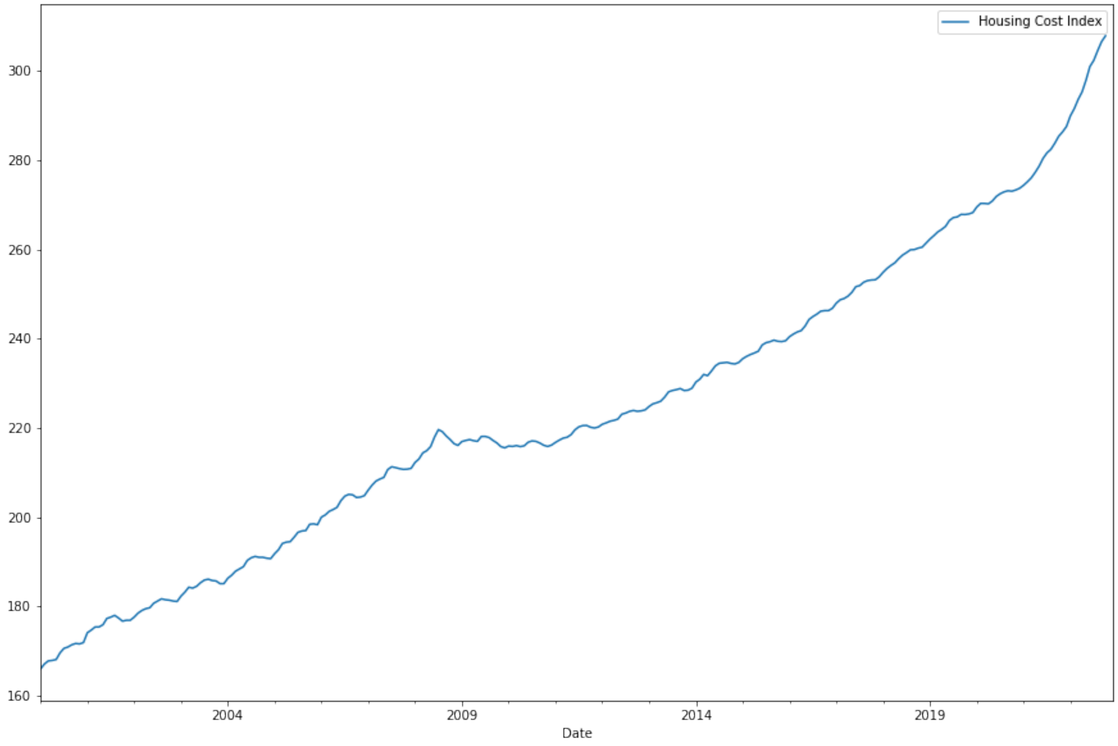
```
         Housing Cost Index
Date
2000-01             166.000
2000-02             167.100
2000-03             167.800
2000-04             167.900
2000-05             168.100
…                      …
2022-08             304.506
2022-09             306.521
2022-10             307.816
2022-11                 NaN
2022-12                 NaN

[276 rows x 1 columns]
```

[797]:

```
housing_cost_index.plot.line(figsize=(15, 10))
```

[797]:

```
<AxesSubplot: xlabel='Date'>
```



4.4 Rent of Primary Residence

```
[798]: rent_primary_index.shape
```

```
[798]: (23, 15)
```

```
[799]: rent_primary_index.head()
```

```
[799]:    Year    Jan    Feb    Mar    Apr    May    Jun    Jul    Aug    Sep    Oct  \
       0  2000  181.1  181.5  182.0  182.3  182.7  183.2  183.9  184.6  185.3  186.1
       1  2001  188.2  188.9  189.6  190.2  191.0  191.6  192.3  193.1  193.9  194.7
       2  2002  197.0  197.7  198.2  198.5  198.8  199.3  199.8  200.2  200.7  201.3
       3  2003  203.3  203.7  204.1  204.5  204.9  205.1  205.6  206.1  206.6  206.9
       4  2004  208.3  208.8  209.2  209.7  210.2  210.7  211.2  211.9  212.4  212.8

            Nov    Dec  HALF1  HALF2
       0  186.8  187.6  182.1  185.7
       1  195.5  196.4  189.9  194.3
       2  202.0  202.5  198.3  201.1
       3  207.5  207.9  204.3  206.8
       4  213.2  213.9  209.5  212.6
```

```
[800]: rent_primary_index.columns
```

```
[800]: Index(['Year', 'Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep',
              'Oct', 'Nov', 'Dec', 'HALF1', 'HALF2'],
             dtype='object')
```

```
[801]: rent_primary_index.nunique(axis=0)
```

```
[801]: Year     23
       Jan      23
       Feb      23
       Mar      23
       Apr      23
       May      23
       Jun      23
       Jul      23
       Aug      23
       Sep      23
       Oct      23
       Nov      22
       Dec      22
       HALF1    23
       HALF2    22
       dtype: int64
```

```
[802]: rent_primary_index.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 23 entries, 0 to 22
Data columns (total 15 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Year    23 non-null     int64
 1   Jan     23 non-null     float64
 2   Feb     23 non-null     float64
 3   Mar     23 non-null     float64
 4   Apr     23 non-null     float64
 5   May     23 non-null     float64
 6   Jun     23 non-null     float64
 7   Jul     23 non-null     float64
 8   Aug     23 non-null     float64
 9   Sep     23 non-null     float64
 10  Oct     23 non-null     float64
 11  Nov     22 non-null     float64
 12  Dec     22 non-null     float64
 13  HALF1   23 non-null     float64
 14  HALF2   22 non-null     float64
dtypes: float64(14), int64(1)
memory usage: 2.8 KB
```

```
[803]: rent_primary_index = rent_primary_index.drop(["HALF1", "HALF2"], axis=1)
       rent_primary_index
```

```
[803]:     Year      Jan      Feb      Mar      Apr      May      Jun      Jul  \
       0   2000  181.100  181.500  182.000  182.300  182.700  183.200  183.900
       1   2001  188.200  188.900  189.600  190.200  191.000  191.600  192.300
       2   2002  197.000  197.700  198.200  198.500  198.800  199.300  199.800
       3   2003  203.300  203.700  204.100  204.500  204.900  205.100  205.600
       4   2004  208.300  208.800  209.200  209.700  210.200  210.700  211.200
       5   2005  214.500  215.000  215.500  216.000  216.400  216.800  217.500
       6   2006  220.900  221.600  222.300  222.900  223.600  224.400  225.200
       7   2007  230.806  231.739  232.495  232.980  233.549  234.071  234.732
       8   2008  239.850  240.325  240.874  241.474  241.803  242.640  243.367
       9   2009  247.974  248.305  248.639  248.899  249.069  249.092  248.994
       10  2010  249.144  249.017  249.089  249.012  248.925  248.999  249.126
       11  2011  251.555  251.829  252.145  252.221  252.393  252.592  253.085
       12  2012  257.714  258.184  258.569  258.922  259.231  259.407  260.107
       13  2013  264.700  265.256  265.821  265.984  266.559  266.905  267.482
       14  2014  272.317  272.733  273.486  274.100  274.710  275.321  276.248
       15  2015  281.572  282.389  283.130  283.598  284.245  285.031  286.090
       16  2016  292.004  292.777  293.489  294.175  295.036  295.902  296.862
       17  2017  303.467  304.211  304.868  305.477  306.379  307.314  308.173
       18  2018  314.788  315.277  315.883  316.763  317.490  318.318  319.351
       19  2019  325.597  326.351  327.513  328.678  329.333  330.648  331.605
       20  2020  337.825  338.616  339.519  340.135  340.811  341.294  341.950
       21  2021  344.758  345.242  345.717  346.267  347.016  347.833  348.469
       22  2022  357.737  359.627  361.083  362.951  365.116  367.927  370.448

               Aug      Sep      Oct      Nov      Dec
       0   184.600  185.300  186.100  186.800  187.600
       1   193.100  193.900  194.700  195.500  196.400
       2   200.200  200.700  201.300  202.000  202.500
       3   206.100  206.600  206.900  207.500  207.900
       4   211.900  212.400  212.800  213.200  213.900
       5   218.000  218.600  219.300  220.000  220.500
       6   226.200  227.100  228.000  228.900  230.000
       7   235.311  236.058  237.135  238.169  239.102
       8   244.181  244.926  245.855  246.681  247.278
       9   249.029  248.965  248.888  248.886  248.999
       10  249.024  249.368  249.618  250.317  250.986
       11  254.003  254.628  255.651  256.367  257.189
       12  260.677  261.421  262.707  263.365  264.098
       13  268.505  269.137  269.960  270.698  271.688
       14  277.048  277.998  278.985  280.123  280.874
       15  287.068  288.306  289.428  290.322  291.204
       16  297.916  298.962  300.400  301.587  302.735
       17  309.479  310.268  311.501  312.670  313.904
```

```
18  320.651  321.533  322.628  323.968  324.815
19  332.638  333.834  334.680  335.819  336.789
20  342.444  342.910  343.615  344.039  344.455
21  349.710  351.255  352.892  354.526  355.931
22  373.283  376.569  379.436      NaN      NaN
```

[804]:

```
rent_primary_index = rent_primary_index.melt(id_vars=["Year"], var_name="Month")
rent_primary_index['Year'] = rent_primary_index['Year'].astype(str)
rent_primary_index['Month'] = rent_primary_index['Month'].astype(str)
rent_primary_index
```

[804]:

```
     Year Month    value
0    2000   Jan  181.100
1    2001   Jan  188.200
2    2002   Jan  197.000
3    2003   Jan  203.300
4    2004   Jan  208.300
..    …    …       …
271  2018   Dec  324.815
272  2019   Dec  336.789
273  2020   Dec  344.455
274  2021   Dec  355.931
275  2022   Dec      NaN

[276 rows x 3 columns]
```

[805]:

```
month_dict3 = {
    "Jan": "January",
    "Feb": "February",
    "Mar": "March",
    "Apr":"April",
    "May":"May",
    "Jun":"June",
    "Jul":"July",
    "Aug":"August",
    "Sep":"September",
    "Oct":"October",
    "Nov":"November",
    "Dec":"December",
}
rent_primary_index['Month'] = rent_primary_index['Month'].replace(month_dict3)
rent_primary_index['Date'] =rent_primary_index ['Year'] + " "+␣
 ↪rent_primary_index['Month']
rent_primary_index
```

[805]:

```
     Year     Month    value          Date
0    2000   January  181.100  2000 January
```

```
1    2001   January  188.200   2001 January
2    2002   January  197.000   2002 January
3    2003   January  203.300   2003 January
4    2004   January  208.300   2004 January
..    …        …        …         …
271  2018  December  324.815  2018 December
272  2019  December  336.789  2019 December
273  2020  December  344.455  2020 December
274  2021  December  355.931  2021 December
275  2022  December      NaN  2022 December

[276 rows x 4 columns]
```

[806]:
```
rent_primary_index['Date'] = pd.to_datetime(rent_primary_index['Date'])
rent_primary_index['Date'] = rent_primary_index['Date'].dt.to_period('M')
rent_primary_index
```

[806]:
```
     Year     Month    value     Date
0    2000   January  181.100  2000-01
1    2001   January  188.200  2001-01
2    2002   January  197.000  2002-01
3    2003   January  203.300  2003-01
4    2004   January  208.300  2004-01
..    …        …        …       …
271  2018  December  324.815  2018-12
272  2019  December  336.789  2019-12
273  2020  December  344.455  2020-12
274  2021  December  355.931  2021-12
275  2022  December      NaN  2022-12

[276 rows x 4 columns]
```

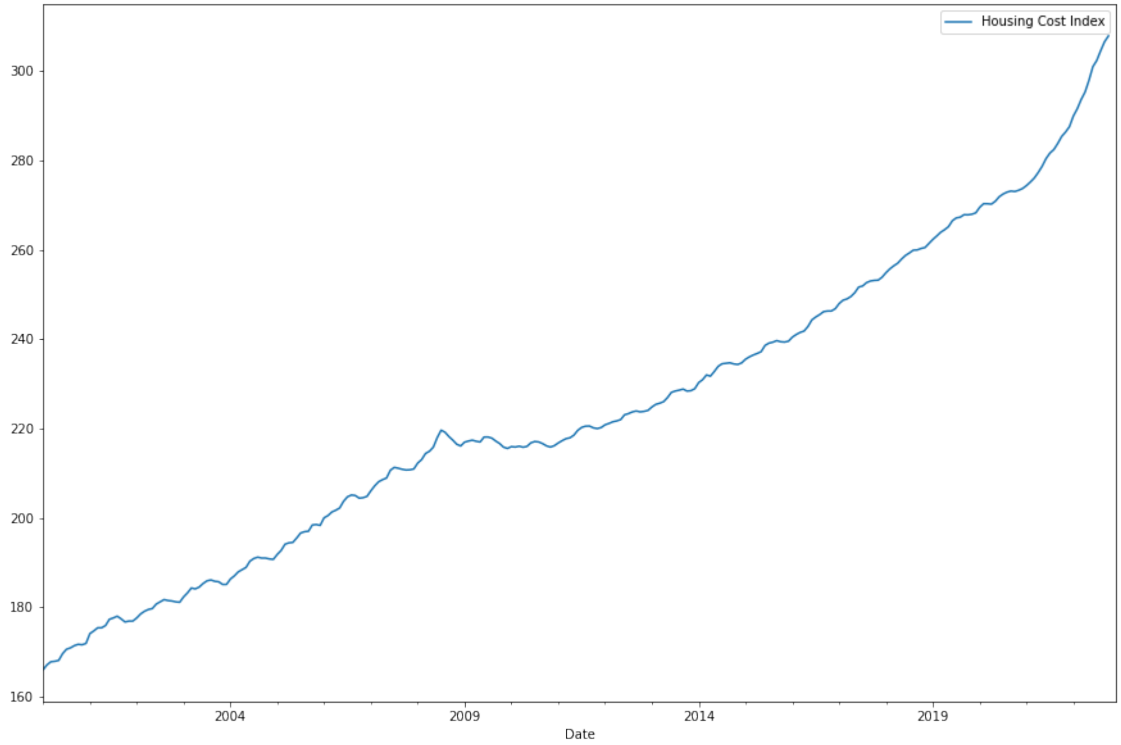
[807]:
```
rent_primary_index = rent_primary_index.drop(["Year", "Month"], axis=1)
rent_primary_index = rent_primary_index.set_index('Date')
rent_primary_index = rent_primary_index.rename(columns ={'value': 'Rent Primary␣
 ↪Index'})
rent_primary_index = rent_primary_index.sort_index()
rent_primary_index
```

[807]:
```
         Rent Primary Index
Date
2000-01             181.100
2000-02             181.500
2000-03             182.000
2000-04             182.300
2000-05             182.700
…                     …
```

```
2022-08              373.283
2022-09              376.569
2022-10              379.436
2022-11                  NaN
2022-12                  NaN

[276 rows x 1 columns]
```

[808]:
```
housing_cost_index.plot.line(figsize=(15, 10))
```

[808]:
```
<AxesSubplot: xlabel='Date'>
```



4.5 Combining All CPI

[809]:
```
recession_cpi = pd.merge(pd.merge(inflation_rate, purchasing_power,␣
 ↪on='Date'),rent_primary_index,on='Date')
recession_cpi
```

[809]:
```
         Inflation  Purchasing Power  Rent Primary Index
Date
2000-01        0.3              59.2             181.100
2000-02        0.4              58.9             181.500
2000-03        0.6              58.4             182.000
```

```
2000-04       -0.1              58.4             182.300
2000-05        0.2              58.3             182.700
…            …               …                 …
2022-08        0.1              33.8             373.283
2022-09        0.4              33.7             376.569
2022-10        0.4              33.6             379.436
2022-11        NaN               NaN                 NaN
2022-12        NaN               NaN                 NaN

[276 rows x 3 columns]
```

[810]:

```
recession_cpi = recession_cpi.merge(housing_cost_index,on='Date')
recession_cpi
```

[810]:

```
         Inflation  Purchasing Power  Rent Primary Index  Housing Cost Index
Date
2000-01        0.3              59.2             181.100             166.000
2000-02        0.4              58.9             181.500             167.100
2000-03        0.6              58.4             182.000             167.800
2000-04       -0.1              58.4             182.300             167.900
2000-05        0.2              58.3             182.700             168.100
…            …               …                 …                 …
2022-08        0.1              33.8             373.283             304.506
2022-09        0.4              33.7             376.569             306.521
2022-10        0.4              33.6             379.436             307.816
2022-11        NaN               NaN                 NaN                 NaN
2022-12        NaN               NaN                 NaN                 NaN

[276 rows x 4 columns]
```

[811]:

```
recession_cpi = recession_cpi.rename(columns ={'Rent Primary Index': 'Rent␣
 ↪Primary', 'Housing Cost Index_x':'Housing Cost', 'Housing Cost Index_y':
 ↪'Housing Cost'})
recession_cpi
```

[811]:

```
         Inflation  Purchasing Power  Rent Primary  Housing Cost Index
Date
2000-01        0.3              59.2       181.100             166.000
2000-02        0.4              58.9       181.500             167.100
2000-03        0.6              58.4       182.000             167.800
2000-04       -0.1              58.4       182.300             167.900
2000-05        0.2              58.3       182.700             168.100
…            …               …           …                 …
2022-08        0.1              33.8       373.283             304.506
2022-09        0.4              33.7       376.569             306.521
2022-10        0.4              33.6       379.436             307.816
2022-11        NaN               NaN           NaN                 NaN
```

```
2022-12         NaN                 NaN           NaN                   NaN

[276 rows x 4 columns]
```

[812]:

```
recession_cpi.dropna(inplace=True)
```

[813]:

```
recession_cpi
```

[813]:

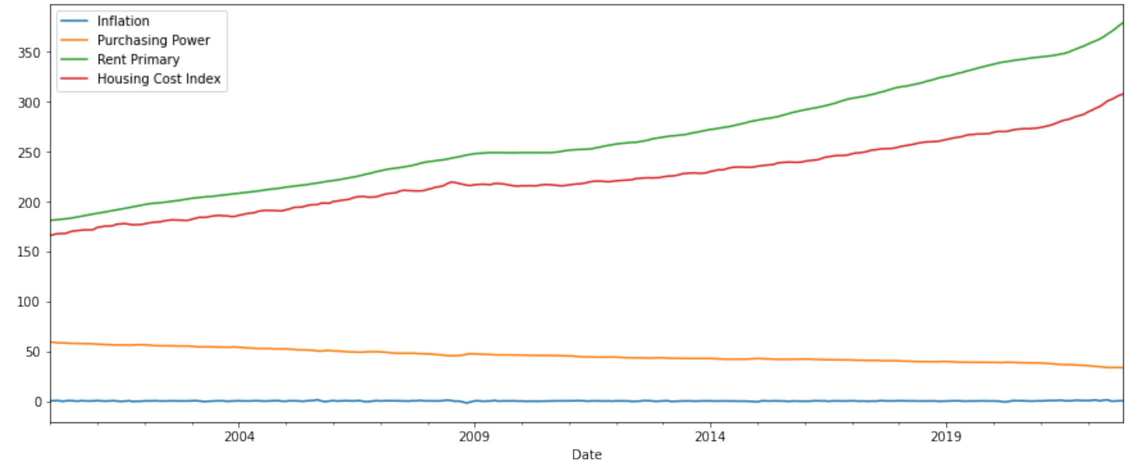
```
         Inflation  Purchasing Power  Rent Primary  Housing Cost Index
Date
2000-01        0.3              59.2       181.100             166.000
2000-02        0.4              58.9       181.500             167.100
2000-03        0.6              58.4       182.000             167.800
2000-04       -0.1              58.4       182.300             167.900
2000-05        0.2              58.3       182.700             168.100
…            …                …            …                  …
2022-06        1.3              33.7       367.927             300.927
2022-07        0.0              33.8       370.448             302.327
2022-08        0.1              33.8       373.283             304.506
2022-09        0.4              33.7       376.569             306.521
2022-10        0.4              33.6       379.436             307.816

[274 rows x 4 columns]
```

[819]:

```
recession_cpi.plot.line(figsize=(15,6))
```

[819]:

```
<AxesSubplot: xlabel='Date'>
```



5 Obsevation

To analyze the CPI, they were each graphed individually at first to see how they performed over time. Once completed, all of the data sets were combined to plot them against each other to identify if any particular trend existed. The Housing Cost Index and the Rent Primary Index has been on upward trend since the year 2000, it appears to have leveled out around the 2009 recession but quickly returned to exponentially increase. The inflation rate index varies greatly over the last 20 years. Similarly to the above observation, the 2001 and 2009 imppact on the inflation is vastly different. 2009 yet again has the biggest decrease than 2001. Which brings to mind the question, why did a recession occur in 2001.

[ ]:

Federal_Reserve_Interest_Rates

December 18, 2022

Federal Reserve Interest Rates

[1]:

```python
import pandas as pd
import numpy as np
import seaborn as sns
import plotly.express as px
import matplotlib.pyplot as plt
from numpy import nan as NA
```

[3]:

```python
#load the data
df = pd.read_csv("index.csv")
```

[4]:

```python
#Access the first 5 rows of the data
df.head()
```

[4]:

```
   Year  Month  Day  Federal Funds Target Rate  Federal Funds Upper Target  \
0  1954      7    1                        NaN                         NaN
1  1954      8    1                        NaN                         NaN
2  1954      9    1                        NaN                         NaN
3  1954     10    1                        NaN                         NaN
4  1954     11    1                        NaN                         NaN

   Federal Funds Lower Target  Effective Federal Funds Rate  \
0                         NaN                          0.80
1                         NaN                          1.22
2                         NaN                          1.06
3                         NaN                          0.85
4                         NaN                          0.83

   Real GDP (Percent Change)  Unemployment Rate  Inflation Rate
0                        4.6                5.8             NaN
1                        NaN                6.0             NaN
2                        NaN                6.1             NaN
3                        8.0                5.7             NaN
4                        NaN                5.3             NaN
```

[5]:

```python
#Access the last 5 rows of the data
df.tail()
```

[5]:
```
     Year  Month  Day  Federal Funds Target Rate  Federal Funds Upper Target  \
899  2016     12   14                        NaN                        0.75
900  2017      1    1                        NaN                        0.75
901  2017      2    1                        NaN                        0.75
902  2017      3    1                        NaN                        0.75
903  2017      3   16                        NaN                        1.00

     Federal Funds Lower Target  Effective Federal Funds Rate  \
899                        0.50                           NaN
900                        0.50                          0.65
901                        0.50                          0.66
902                        0.50                           NaN
903                        0.75                           NaN

     Real GDP (Percent Change)  Unemployment Rate  Inflation Rate
899                        NaN                NaN             NaN
900                        NaN                4.8             2.3
901                        NaN                4.7             2.2
902                        NaN                NaN             NaN
903                        NaN                NaN             NaN
```

[6]:
```
#Count the number of rows and columns in the data
df.shape
```

[6]:
```
(904, 10)
```

[7]:
```
#Count the number of null values in each column
df.isnull().sum()
```

[7]:
```
Year                              0
Month                             0
Day                               0
Federal Funds Target Rate       442
Federal Funds Upper Target      801
Federal Funds Lower Target      801
Effective Federal Funds Rate    152
Real GDP (Percent Change)       654
Unemployment Rate               152
Inflation Rate                  194
dtype: int64
```

[8]:
```
#Delete the columns we are not interested with and also the
#rows with empty values
data_new = df.iloc[:,:4]
df2 = data_new.dropna()
```

[9]:
```
df2
```

```
[9]:      Year  Month  Day  Federal Funds Target Rate
     339  1982      9   27                      10.25
     340  1982     10    1                      10.00
     341  1982     10    7                       9.50
     342  1982     11    1                       9.50
     343  1982     11   19                       9.00
     ..    …     …   …                         …
     796  2008     10    1                       2.00
     797  2008     10    8                       1.50
     798  2008     10   29                       1.00
     799  2008     11    1                       1.00
     800  2008     12    1                       1.00

     [462 rows x 4 columns]
```

```
[21]: #Create a new column called "date" by merging the Year, Month and
      #Day together
      df2['Date'] = df2[df2.columns[0:3]].apply(lambda x: '/'.join(x.astype(str)),␣
       ↪axis = 1 )

      #print the first 5 rows
      df2.head()
```

```
C:\Users\koseb\Anaconda3\lib\site-packages\ipykernel_launcher.py:3:
SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
```

```
[21]:      Year  Month  Day  Federal Funds Target Rate        Date
      339  1982      9   27                      10.25   1982/9/27
      340  1982     10    1                      10.00   1982/10/1
      341  1982     10    7                       9.50   1982/10/7
      342  1982     11    1                       9.50   1982/11/1
      343  1982     11   19                       9.00  1982/11/19
```

```
[22]: #Delete the first three columns and print the output
      df3 = pd.DataFrame(df2, columns = ["Federal Funds Target Rate", 'Date'])
      print(df3)
```

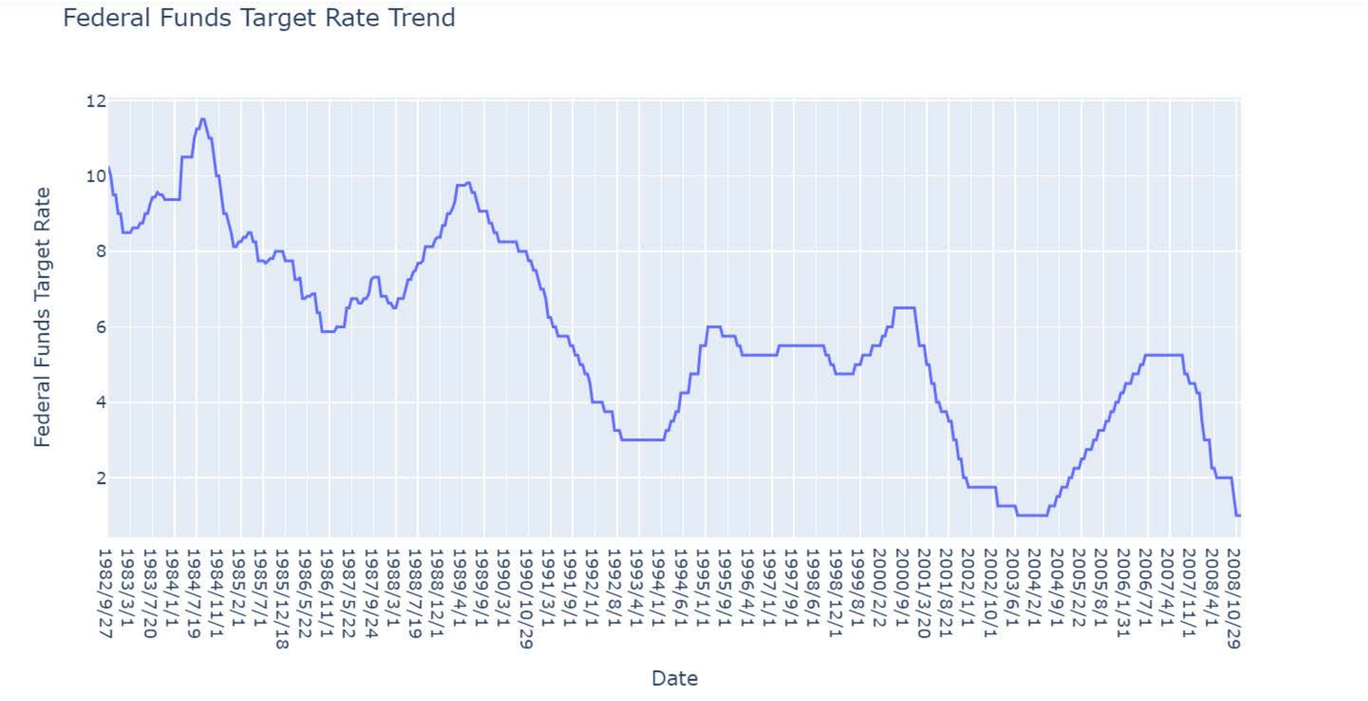
```
     Federal Funds Target Rate        Date
339                      10.25   1982/9/27
340                      10.00   1982/10/1
```

```
341                        9.50   1982/10/7
342                        9.50   1982/11/1
343                        9.00  1982/11/19
..                          …          …
796                        2.00   2008/10/1
797                        1.50   2008/10/8
798                        1.00  2008/10/29
799                        1.00   2008/11/1
800                        1.00   2008/12/1

[462 rows x 2 columns]
```

[23]:

```
#plot shows the federal funds target rate trend.
fig = px.line(df3, x="Date", y="Federal Funds Target Rate", title='Federal␣
 ↪Funds Target Rate Trend')
fig.show()
```

From the plot, rates have risen by the most in a single year since the 1980s. It then began drifting downward sharply, falling first to a target range of 5-6 percent on Nov. 1, 1986, then down to 3 percent on November 1, 1982. After several oscillations, interest rates haven't eclipsed 10 percent since November 1984. It can also be seen that the Federal Funds Target Rate in 2008 had the lowest of 1 percent.



SingleFamilyResidence by City

December 17, 2022

[2]:

```
pip install matplotlib
```

```
Requirement already satisfied: matplotlib in
/home/jovyan/.local/lib/python3.8/site-packages (3.6.2)
Requirement already satisfied: fonttools>=4.22.0 in
/home/jovyan/.local/lib/python3.8/site-packages (from matplotlib) (4.38.0)
Requirement already satisfied: packaging>=20.0 in
/home/jovyan/.local/lib/python3.8/site-packages (from matplotlib) (22.0)
Requirement already satisfied: python-dateutil>=2.7 in
/opt/conda/lib/python3.8/site-packages (from matplotlib) (2.8.1)
Requirement already satisfied: cycler>=0.10 in
/home/jovyan/.local/lib/python3.8/site-packages (from matplotlib) (0.11.0)
Collecting pillow>=6.2.0
  Using cached Pillow-9.3.0-cp38-cp38-manylinux_2_28_x86_64.whl (3.3 MB)
Requirement already satisfied: kiwisolver>=1.0.1 in
/home/jovyan/.local/lib/python3.8/site-packages (from matplotlib) (1.4.4)
Requirement already satisfied: numpy>=1.19 in /opt/conda/lib/python3.8/site-
packages (from matplotlib) (1.20.3)
Requirement already satisfied: contourpy>=1.0.1 in
/home/jovyan/.local/lib/python3.8/site-packages (from matplotlib) (1.0.6)
Requirement already satisfied: pyparsing>=2.2.1 in
/opt/conda/lib/python3.8/site-packages (from matplotlib) (2.4.7)
Requirement already satisfied: six>=1.5 in /opt/conda/lib/python3.8/site-
packages (from python-dateutil>=2.7->matplotlib) (1.15.0)
Installing collected packages: pillow
Successfully installed pillow-9.3.0
Note: you may need to restart the kernel to use updated packages.
```

[1]:

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
```

[2]:

```python
#read the data
df = pd.read_csv('data/City_Zhvi_SingleFamilyResidence - Copy.csv')
```

[3]:

```python
df.head()
```

```
[3]:    Unnamed: 0  RegionID  SizeRank   RegionName RegionType StateName State  \
     0           0      6181         0     New York       City        NY    NY
     1           1     12447         1  Los Angeles       City        CA    CA
     2           2     39051         2      Houston       City        TX    TX
     3           3     17426         3      Chicago       City        IL    IL
     4           4      6915         4  San Antonio       City        TX    TX

                                   Metro         CountyName  1/31/1996  …  \
     0       New York-Newark-Jersey City      Queens County   208545.0  …
     1    Los Angeles-Long Beach-Anaheim  Los Angeles County   192855.0  …
     2  Houston-The Woodlands-Sugar Land      Harris County    95018.0  …
     3           Chicago-Naperville-Elgin        Cook County   126867.0  …
     4          San Antonio-New Braunfels       Bexar County    94406.0  …

        6/30/2019  7/31/2019  8/31/2019  9/30/2019  10/31/2019  11/30/2019  \
     0     672433     671924     671423     670719      669974      669118
     1     745290     746729     748924     751756      755716      759279
     2     189803     190437     191052     191483      192124      192620
     3     226322     226635     226796     226645      226505      226430
     4     183622     184246     184831     185752      186401      187159

        12/31/2019  1/31/2020  2/29/2020  3/31/2020
     0      668736     668740     668581     668030
     1      764877     770853     779717     788751
     2      193202     193427     193991     194986
     3      226454     226727     227077     227605
     4      187339     187886     188055     188650

     [5 rows x 300 columns]
```

```
[4]: #delete columns that are not necessary in the analysis
     df = df.drop(['SizeRank','RegionID','RegionType','StateName', 'Metro','State',␣
      ↪'CountyName'], axis = 1)
     df = df.drop(df.columns[0], axis = 1)
```

```
[5]: df.head()
```

```
[5]:     RegionName  1/31/1996  2/29/1996  3/31/1996  4/30/1996  5/31/1996  \
     0     New York   208545.0   207968.0   207669.0   207086.0   206852.0
     1  Los Angeles   192855.0   192899.0   192974.0   193133.0   193265.0
     2      Houston    95018.0    95117.0    95124.0    95286.0    95445.0
     3      Chicago   126867.0   126739.0   126485.0   126450.0   126115.0
     4  San Antonio    94406.0    94372.0    94339.0    94323.0    94261.0

        6/30/1996  7/31/1996  8/31/1996  9/30/1996  …  6/30/2019  7/31/2019  \
     0   206672.0   206607.0   206508.0   206465.0  …     672433     671924
     1   193453.0   193710.0   193742.0   193617.0  …     745290     746729
```

```
2    95552.0    95601.0    95689.0    95917.0  …     189803     190437
3   126217.0   126089.0   126459.0   126959.0  …     226322     226635
4    94248.0    94251.0    94355.0    94475.0  …     183622     184246

   8/31/2019  9/30/2019  10/31/2019  11/30/2019  12/31/2019  1/31/2020  \
0     671423     670719      669974      669118      668736     668740
1     748924     751756      755716      759279      764877     770853
2     191052     191483      192124      192620      193202     193427
3     226796     226645      226505      226430      226454     226727
4     184831     185752      186401      187159      187339     187886

   2/29/2020  3/31/2020
0     668581     668030
1     779717     788751
2     193991     194986
3     227077     227605
4     188055     188650

[5 rows x 292 columns]
```

[6]:

```
#filter the 3 cities New York, San Francisco and Dallas
cities_df = df[df["RegionName"].isin(['New York','San Francisco', 'Dallas'])]
```

[7]:

```
cities_df
```

[7]:

```
       RegionName  1/31/1996  2/29/1996  3/31/1996  4/30/1996  5/31/1996  \
0        New York   208545.0   207968.0   207669.0   207086.0   206852.0
9          Dallas    98466.0    98511.0    98716.0    99079.0    99450.0
14  San Francisco   299060.0   298066.0   297387.0   296175.0   295267.0

    6/30/1996  7/31/1996  8/31/1996  9/30/1996  …  6/30/2019  7/31/2019  \
0    206672.0   206607.0   206508.0   206465.0  …     672433     671924
9     99701.0    99742.0    99887.0    99968.0  …     232420     232925
14   294811.0   294422.0   294795.0   295419.0  …    1463199    1464488

    8/31/2019  9/30/2019  10/31/2019  11/30/2019  12/31/2019  1/31/2020  \
0      671423     670719      669974      669118      668736     668740
9      233400     234679      235001      235436      234761     235030
14    1462393    1467249     1473085     1482111     1495964    1504169

    2/29/2020  3/31/2020
0      668581     668030
9      235163     235553
14    1512624    1515959

[3 rows x 292 columns]
```

[8]:

```
#Transpose the data
cities_3_df = cities_df.T
```

[9]:

```
cities_3_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 292 entries, RegionName to 3/31/2020
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   0       292 non-null    object
 1   9       292 non-null    object
 2   14      292 non-null    object
dtypes: object(3)
memory usage: 17.2+ KB
```

[10]:

```
#reset the column headers
cities_3_df.columns = cities_3_df.iloc[0]
cities_3_df = cities_3_df[1:]
```

[11]:

```
cities_3_df.head()
```

[11]:

```
RegionName  New York   Dallas San Francisco
1/31/1996   208545.0  98466.0      299060.0
2/29/1996   207968.0  98511.0      298066.0
3/31/1996   207669.0  98716.0      297387.0
4/30/1996   207086.0  99079.0      296175.0
5/31/1996   206852.0  99450.0      295267.0
```

[12]:

```
#change the month format to datetime format
cities_3_df.index = pd.to_datetime(cities_3_df.index)
```

[13]:

```
cities_3_df.head()
```

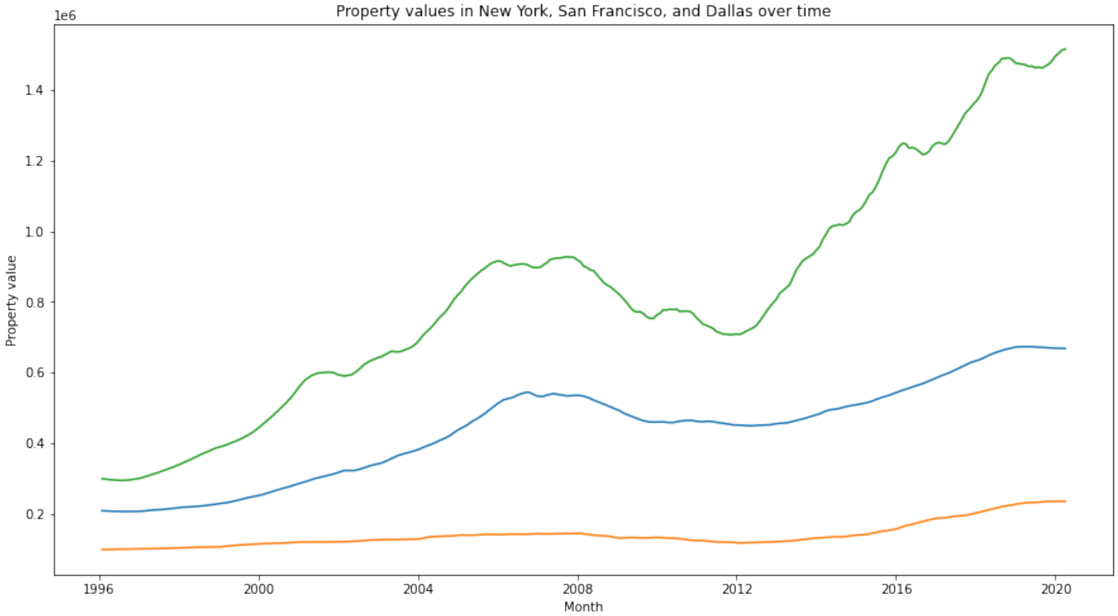
[13]:

```
RegionName  New York   Dallas San Francisco
1996-01-31  208545.0  98466.0      299060.0
1996-02-29  207968.0  98511.0      298066.0
1996-03-31  207669.0  98716.0      297387.0
1996-04-30  207086.0  99079.0      296175.0
1996-05-31  206852.0  99450.0      295267.0
```

Create the Plot

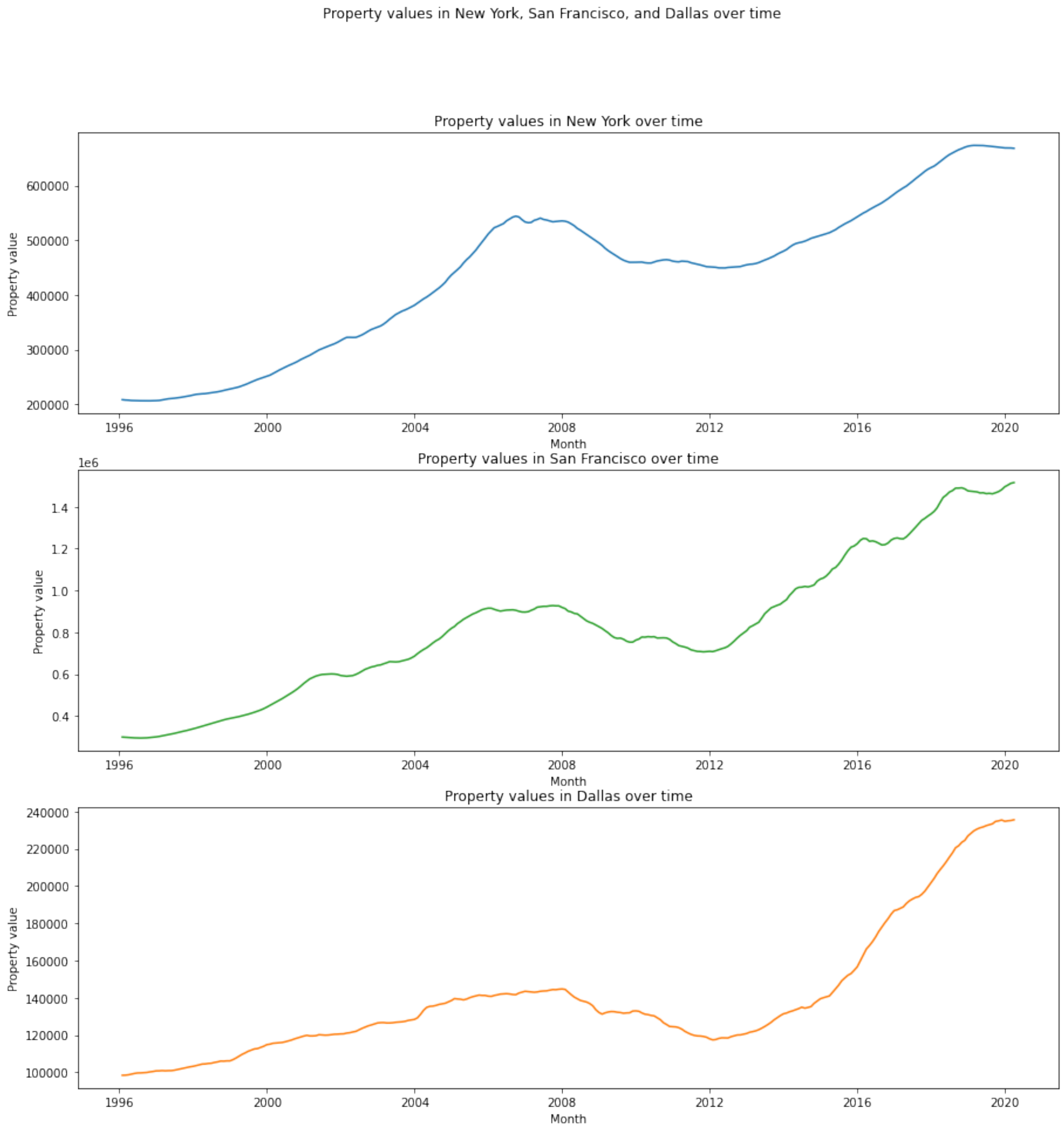
[38]:

```
fig, ax = plt.subplots(figsize=(15,8))
ax.plot(cities_3_df)
plt.title('Property values in New York, San Francisco, and Dallas over time')
plt.xlabel('Month')
plt.ylabel('Property value')
plt.show()
```



[42]:
```
fig, (ax1, ax2, ax3) = plt.subplots(3, figsize=(15,15))
fig.suptitle('Property values in New York, San Francisco, and Dallas over time')
ax1.plot(cities_3_df.index, cities_3_df['New York'], 'tab:blue')
ax1.set_title('Property values in New York over time')
ax1.set_xlabel('Month')
ax1.set_ylabel('Property value')
ax2.plot(cities_3_df.index, cities_3_df['San Francisco'], 'tab:green')
ax2.set_title('Property values in San Francisco over time')
ax2.set_xlabel('Month')
ax2.set_ylabel('Property value')
ax3.plot(cities_3_df.index, cities_3_df.Dallas, 'tab:orange')
ax3.set_title('Property values in Dallas over time')
ax3.set_xlabel('Month')
ax3.set_ylabel('Property value')
```

[42]:
```
Text(0, 0.5, 'Property value')
```



Methods ued to clean the

Observation

The individual plots for New York, San Francisco and Dallas clearly shows a major fall starting from 2008. The plots also show that the trend starts going up from 2012.

Home Price Index

December 18, 2022

[1]:
```
%pip install matplotlib
%pip install seaborn
%pip install numpy
```

```
Requirement already satisfied: matplotlib in /opt/conda/lib/python3.8/site-
packages (3.6.2)
Requirement already satisfied: pyparsing>=2.2.1 in
/opt/conda/lib/python3.8/site-packages (from matplotlib) (2.4.7)
Requirement already satisfied: cycler>=0.10 in /opt/conda/lib/python3.8/site-
packages (from matplotlib) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in
/opt/conda/lib/python3.8/site-packages (from matplotlib) (4.38.0)
Requirement already satisfied: python-dateutil>=2.7 in
/opt/conda/lib/python3.8/site-packages (from matplotlib) (2.8.1)
Requirement already satisfied: pillow>=6.2.0 in /opt/conda/lib/python3.8/site-
packages (from matplotlib) (9.3.0)
Requirement already satisfied: numpy>=1.19 in /opt/conda/lib/python3.8/site-
packages (from matplotlib) (1.20.3)
Requirement already satisfied: packaging>=20.0 in /opt/conda/lib/python3.8/site-
packages (from matplotlib) (20.9)
Requirement already satisfied: kiwisolver>=1.0.1 in
/opt/conda/lib/python3.8/site-packages (from matplotlib) (1.4.4)
Requirement already satisfied: contourpy>=1.0.1 in
/opt/conda/lib/python3.8/site-packages (from matplotlib) (1.0.6)
Requirement already satisfied: six>=1.5 in /opt/conda/lib/python3.8/site-
packages (from python-dateutil>=2.7->matplotlib) (1.15.0)
Note: you may need to restart the kernel to use updated packages.
Requirement already satisfied: seaborn in /opt/conda/lib/python3.8/site-packages
(0.12.1)
Requirement already satisfied: pandas>=0.25 in /opt/conda/lib/python3.8/site-
packages (from seaborn) (1.2.3)
Requirement already satisfied: matplotlib!=3.6.1,>=3.1 in
/opt/conda/lib/python3.8/site-packages (from seaborn) (3.6.2)
Requirement already satisfied: numpy>=1.17 in /opt/conda/lib/python3.8/site-
packages (from seaborn) (1.20.3)
Requirement already satisfied: python-dateutil>=2.7 in
/opt/conda/lib/python3.8/site-packages (from matplotlib!=3.6.1,>=3.1->seaborn)
(2.8.1)
```

```
Requirement already satisfied: fonttools>=4.22.0 in
/opt/conda/lib/python3.8/site-packages (from matplotlib!=3.6.1,>=3.1->seaborn)
(4.38.0)
Requirement already satisfied: pillow>=6.2.0 in /opt/conda/lib/python3.8/site-
packages (from matplotlib!=3.6.1,>=3.1->seaborn) (9.3.0)
Requirement already satisfied: cycler>=0.10 in /opt/conda/lib/python3.8/site-
packages (from matplotlib!=3.6.1,>=3.1->seaborn) (0.11.0)
Requirement already satisfied: pyparsing>=2.2.1 in
/opt/conda/lib/python3.8/site-packages (from matplotlib!=3.6.1,>=3.1->seaborn)
(2.4.7)
Requirement already satisfied: packaging>=20.0 in /opt/conda/lib/python3.8/site-
packages (from matplotlib!=3.6.1,>=3.1->seaborn) (20.9)
Requirement already satisfied: kiwisolver>=1.0.1 in
/opt/conda/lib/python3.8/site-packages (from matplotlib!=3.6.1,>=3.1->seaborn)
(1.4.4)
Requirement already satisfied: contourpy>=1.0.1 in
/opt/conda/lib/python3.8/site-packages (from matplotlib!=3.6.1,>=3.1->seaborn)
(1.0.6)
Requirement already satisfied: pytz>=2017.3 in /opt/conda/lib/python3.8/site-
packages (from pandas>=0.25->seaborn) (2021.1)
Requirement already satisfied: six>=1.5 in /opt/conda/lib/python3.8/site-
packages (from python-dateutil>=2.7->matplotlib!=3.6.1,>=3.1->seaborn) (1.15.0)
Note: you may need to restart the kernel to use updated packages.
Requirement already satisfied: numpy in /opt/conda/lib/python3.8/site-packages
(1.20.3)
Note: you may need to restart the kernel to use updated packages.
```

```
[ ]: import pandas as pd
     import numpy as np
     import matplotlib.pyplot as plt
     import seaborn as sns
     %matplotlib inline
```

```
[3]: #reading in the data
     df1 = pd.read_csv("20_city_composite.csv")
     df1.head()
```

```
[3]:    Unnamed: 0        DATE    ATXRSA_20221025  ATXRSA_20221129  \
     0           0  1996-10-01     83.04001512207        83.040015
     1           1  1996-11-01    83.324704973617        83.324705
     2           2  1996-12-01  83.42444219970899        83.424442
     3           3  1997-01-01    83.701866383281        83.701866
     4           4  1997-02-01    83.870200842087        83.870201

       BOXRSA_20221025  BOXRSA_20221129    CRXRSA_20221025  CRXRSA_20221129  \
     0  73.324347935338        73.324348  88.46497277465299        88.464973
     1  73.475798496643        73.475798    88.920429010441        88.920429
```

```
2     74.19628004694        74.196280  89.27445222752199        89.274452
3  74.60848785465801        74.608488    89.726637543049        89.726638
4    74.905668243927        74.905668    89.886168163619        89.886168

     CHXRSA_20221025  CHXRSA_20221129  …  SDXRSA_20221025  SDXRSA_20221129  \
0    85.812225671459        85.812226  …  71.833716793942        71.833717
1    86.119846279576        86.119846  …  72.147249192587        72.147249
2  86.45211961624099        86.452120  …  72.215761822679        72.215762
3    86.519564927134        86.519565  …  72.560199997531        72.560200
4    86.529694070888        86.529694  …  72.751005939748        72.751006

     SFXRSA_20221025  SFXRSA_20221129    SEXRSA_20221025  SEXRSA_20221129  \
0  68.26022797488001        68.260228    74.680576664407        74.680577
1    68.935062949034        68.935063    74.724470264252        74.724470
2  69.20716988556501        69.207170    75.150635123954        75.150635
3    69.638624273212        69.638624  75.72619340986999        75.726193
4    69.835337534175        69.835338    76.333770987516        76.333771

     TPXRSA_20221025  TPXRSA_20221129    WDXRSA_20221025  WDXRSA_20221129
0    88.212664894093        88.212665  88.81802295700199        88.818023
1    88.294845001021        88.294845  88.92154547951401        88.921545
2    88.724686962012        88.724687  88.67360543147801        88.673605
3    88.648499337757        88.648499  88.77522390968299        88.775224
4  88.64733475149099        88.647335  88.97133458937701        88.971335

[5 rows x 42 columns]
```

[4]:

```
# dropping the columns ending in '1025' as they represent the most previous␣
↪version of hpi data recorded as at
# October 25th, 2022 and thed unnamed column.
df2 = df1.drop(["Unnamed:␣
↪0","BOXRSA_20221025","CHXRSA_20221025","DNXRSA_20221025","LVXRSA_20221025","LXXRSA_20221025
              ␣
↪"MIXRSA_20221025","NYXRSA_20221025","SDXRSA_20221025","SFXRSA_20221025","WDXRSA_20221025",
              "SEXRSA_20221025", "TPXRSA_20221025", "CRXRSA_20221025",␣
↪"PHXRSA_20221025","POXRSA_20221025",
              ␣
↪"DAXRSA_20221025","DEXRSA_20221025","MNXRSA_20221025","CEXRSA_20221025","ATXRSA_20221025"],
↪= 1)
df2.head()
```

[4]:

```
         DATE  ATXRSA_20221129  BOXRSA_20221129  CRXRSA_20221129  \
0  1996-10-01        83.040015        73.324348        88.464973
1  1996-11-01        83.324705        73.475798        88.920429
2  1996-12-01        83.424442        74.196280        89.274452
3  1997-01-01        83.701866        74.608488        89.726638
4  1997-02-01        83.870201        74.905668        89.886168
```

```
   CHXRSA_20221129  CEXRSA_20221129  DAXRSA_20221129  DNXRSA_20221129  \
0        85.812226        87.617266              NaN        74.255800
1        86.119846        87.724310              NaN        74.740080
2        86.452120        87.960116              NaN        75.187379
3        86.519565        87.679939              NaN        75.389774
4        86.529694        88.476397              NaN        75.758766

   DEXRSA_20221129  LVXRSA_20221129  …  MIXRSA_20221129  MNXRSA_20221129  \
0        78.584696        89.663980  …        87.943394        78.846920
1        78.956569        89.480246  …        88.111182        79.304578
2        79.170554        90.142155  …        87.998531        79.585395
3        79.347642        90.781719  …        88.056231        79.975962
4        79.236397        91.630628  …        88.067212        79.971319

   NYXRSA_20221129  PHXRSA_20221129  POXRSA_20221129  SDXRSA_20221129  \
0        79.891283        82.067233        89.004448        71.833717
1        79.916579        82.299406        90.189059        72.147249
2        80.197635        82.475998        90.745698        72.215762
3        80.435989        82.803127        91.244230        72.560200
4        80.594881        83.179630        92.219688        72.751006

   SFXRSA_20221129  SEXRSA_20221129  TPXRSA_20221129  WDXRSA_20221129
0        68.260228        74.680577        88.212665        88.818023
1        68.935063        74.724470        88.294845        88.921545
2        69.207170        75.150635        88.724687        88.673605
3        69.638624        75.726193        88.648499        88.775224
4        69.835338        76.333771        88.647335        88.971335

[5 rows x 21 columns]
```

[5]:

```python
#Transpose dataframe to identify hpi data more easily by city and then month
df3 = df2.T
df3.head()
```

[5]:

```
                          0           1           2           3           4  \
DATE             1996-10-01  1996-11-01  1996-12-01  1997-01-01  1997-02-01
ATXRSA_20221129   83.040015   83.324705   83.424442   83.701866   83.870201
BOXRSA_20221129   73.324348   73.475798    74.19628   74.608488   74.905668
CRXRSA_20221129   88.464973   88.920429   89.274452   89.726638   89.886168
CHXRSA_20221129   85.812226   86.119846    86.45212   86.519565   86.529694

                          5           6           7           8           9  \
DATE             1997-03-01  1997-04-01  1997-05-01  1997-06-01  1997-07-01
ATXRSA_20221129   84.092308   84.515631   84.960143   85.131203   85.354779
BOXRSA_20221129   75.353969   75.725513   76.132558   76.494663   76.753209
CRXRSA_20221129   90.424284   90.374351   90.551782   90.656649   91.141161
```

```
CHXRSA_20221129   86.829768   86.924975   86.888873   86.889531   87.222319

                 …         302         303         304         305  \
DATE             …  2021-12-01  2022-01-01  2022-02-01  2022-03-01
ATXRSA_20221129  …   206.26432  210.246839  214.767384  220.368851
BOXRSA_20221129  …  286.030654  289.738561  296.750068  300.832537
CRXRSA_20221129  …  229.315463  233.674693  238.751329   244.26538
CHXRSA_20221129  …  174.032393  175.800818  178.308976  179.437101

                        306         307         308         309         310  \
DATE             2022-04-01  2022-05-01  2022-06-01  2022-07-01  2022-08-01
ATXRSA_20221129  224.821518  229.004624  231.289945  232.707145  232.376155
BOXRSA_20221129  305.951258  311.279562  313.125375  312.342453   309.54589
CRXRSA_20221129  249.469027  254.203744  258.030465  259.765199   259.50014
CHXRSA_20221129  181.383076  183.589038   185.58657  186.536383   185.94012

                        311
DATE             2022-09-01
ATXRSA_20221129  230.923044
BOXRSA_20221129  304.953081
CRXRSA_20221129  256.954596
CHXRSA_20221129  185.178035

[5 rows x 312 columns]
```

[6]:

```
#extracting data for selected cities of interest: dallas, new york, and san␣
 ↪francisco respectively
InterestCities = df2.loc[:, ("DATE","DAXRSA_20221129",␣
 ↪"NYXRSA_20221129","SFXRSA_20221129")]
InterestCities.head()
```

[6]:

```
         DATE  DAXRSA_20221129  NYXRSA_20221129  SFXRSA_20221129
0  1996-10-01              NaN        79.891283        68.260228
1  1996-11-01              NaN        79.916579        68.935063
2  1996-12-01              NaN        80.197635        69.207170
3  1997-01-01              NaN        80.435989        69.638624
4  1997-02-01              NaN        80.594881        69.835338
```

[7]:

```
#dropping the NaN values from the dataframe
InterestCities.dropna(inplace = True)
InterestCities.head()
```

[7]:

```
          DATE  DAXRSA_20221129  NYXRSA_20221129  SFXRSA_20221129
39  2000-01-01       100.713363       100.339230       101.449954
40  2000-02-01       103.037317       101.240657       104.170604
41  2000-03-01       102.709025       102.060631       107.337223
42  2000-04-01       103.276750       103.132504       110.632995
```

```
43  2000-05-01       103.715387       104.915869       113.800513
```

[8]:

```
InterestCities.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 273 entries, 39 to 311
Data columns (total 4 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   DATE             273 non-null    object
 1   DAXRSA_20221129  273 non-null    float64
 2   NYXRSA_20221129  273 non-null    float64
 3   SFXRSA_20221129  273 non-null    float64
dtypes: float64(3), object(1)
memory usage: 10.7+ KB
```

[9]:

```
InterestCities = InterestCities.set_index("DATE")
```

[10]:

```
#Converting the "DATE" column to a datetime64 data type for easy manipulation␣
↪and plotting
InterestCities.index = pd.to_datetime(InterestCities.index)
```

[11]:

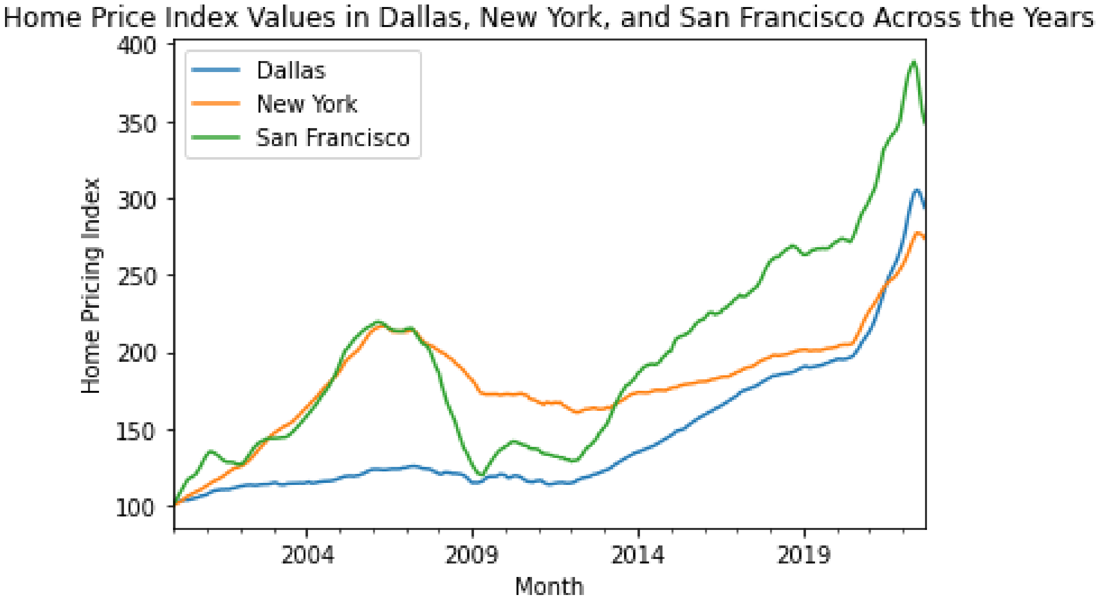
```
#Renaming columns for readability and setting the datetime period as index
InterestCities.rename(columns = {"DAXRSA_20221129":"Dallas","NYXRSA_20221129":
↪"New York","SFXRSA_20221129":"San Francisco"}, inplace = True)
InterestCities.head()
```

[11]:

```
                Dallas    New York  San Francisco
DATE
2000-01-01  100.713363  100.339230     101.449954
2000-02-01  103.037317  101.240657     104.170604
2000-03-01  102.709025  102.060631     107.337223
2000-04-01  103.276750  103.132504     110.632995
2000-05-01  103.715387  104.915869     113.800513
```

[12]:

```
#Creating plots
InterestCities.plot()
plt.title("Home Price Index Values in Dallas, New York, and San Francisco␣
↪Across the Years")
plt.xlabel("Month")
plt.ylabel("Home Pricing Index")
#plt.yscale('log')
plt.show()
```



Observations

Generally, home price index values across the 3 regions have risen over the years, currently being at least 2.5 times as high as index values in the early 2000s. Unlike New York and San Francisco which display an erratic rise and fall in home price index values between the years of 2000 and 2012, Dallas displays a relatively stable increase in home price index values across these years.

From the plot above, it is observed that home price index values in both New York and San Francisco took a major decline between 2007 and 2008, with San Francisco being the most affected. Dallas on the other hand experienced a much milder decrease in home price index values. Though it was hit the hardest, from 2012, home price index values in San Francisco rise very steeply by almost 4 times to surpass their highest price index value of around 225 before the major dip.

Similarly to San Francisco, after 2013, home price index values in Dallas begin to pick up more sharply, going from about 120 to 280 in 9 years unlike New York whose index values oinly reached a peak of around 260.

Overall, San Francisco displays the most fluctuating home price index values over time, closely followed by New York and then Dallas. Based on the plot, in 2022, the city of San Francisco posseses the highest home price index values, followed by New York and then Dallas. However, in the latter end of 2022, it is observed that home price index values in all 3 cities are dropping, with New York being the least affected.