In [600]:
```
import pandas as pd
!pip install openpyxl
```

```
Requirement already satisfied: openpyxl in /opt/miniconda3/lib/python3.9/
site-packages (3.0.10)
Requirement already satisfied: et-xmlfile in /opt/miniconda3/lib/python3.
9/site-packages (from openpyxl) (1.1.0)
```

In [601]:
```
df=pd.read_excel('file:///Users/dilipkumarallu/Desktop/PROJECT_DATASET.xlsx
```

In [602]:
```
df=pd.DataFrame(df)
```

In [603]:
```
df=df.dropna(axis='columns')
```

In [604]:
```
df.to_csv('inf.csv')
```

In [605]:
```
df = pd.read_csv('inf.csv')
```

In [606]:
```
df.head()
```

Out[606]:

	Unnamed: 0	YEAR	World	Purchasing Power	Inflation rate, end of period consumer prices (Annual percent change)	Population(in millions)	Current account balance U.S. dollars (Billions of U.S. dollars)	GDP	Percapita GDP
0	0	1980	2.1	13400.209	17.3	4012.401	-57.235	11238.265	2862.333
1	1	1981	2.0	14930.382	15.1	4086.032	-83.002	11498.276	2876.220
2	2	1982	0.7	15909.301	14.2	4161.918	-92.745	11286.618	2772.215
3	3	1983	2.7	16934.331	13.4	4236.276	-76.953	11589.045	2797.085
4	4	1984	4.6	18321.225	14.0	4311.430	-69.007	12001.655	2846.673

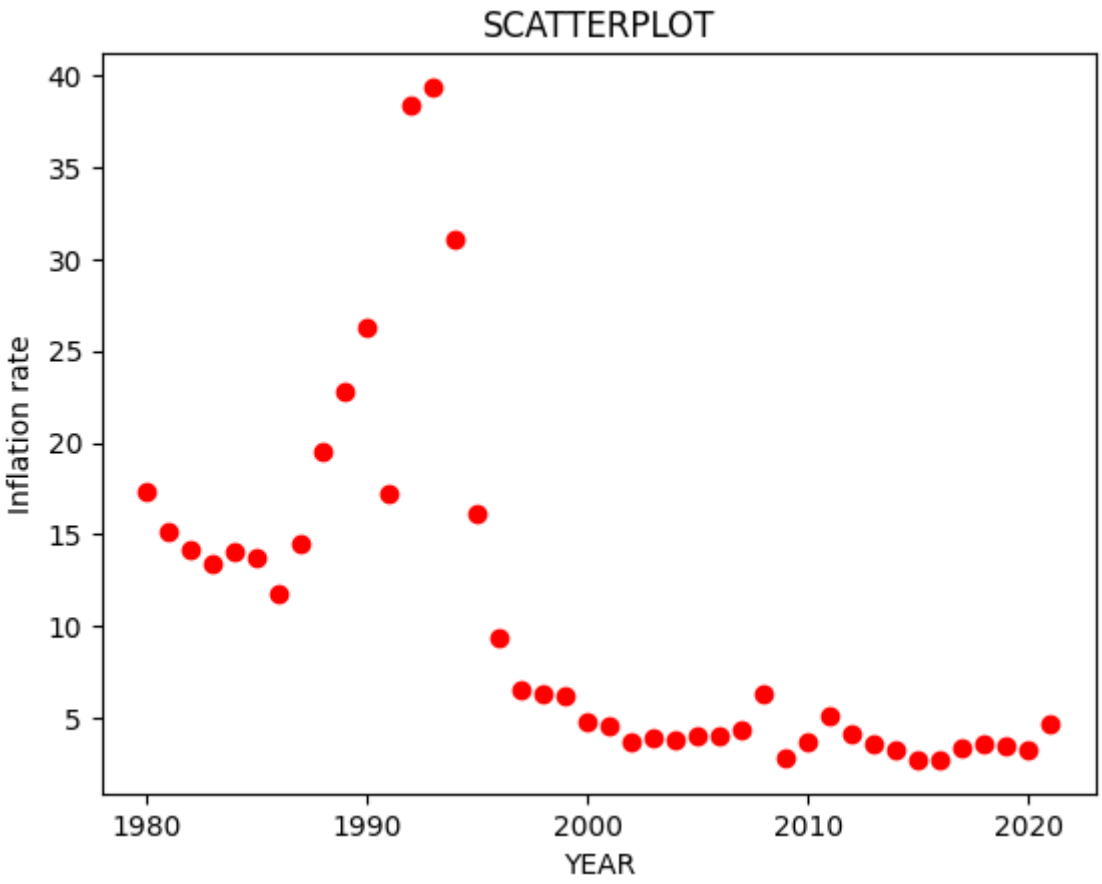
In [607]:
```
df=df.dropna(axis='columns')
```

In [608]:
```
!pip install seaborn
import seaborn as sns
```

```
Requirement already satisfied: seaborn in /opt/miniconda3/lib/python3.9/s
ite-packages (0.12.1)
Requirement already satisfied: pandas>=0.25 in /opt/miniconda3/lib/python
3.9/site-packages (from seaborn) (1.5.1)
Requirement already satisfied: numpy>=1.17 in /opt/miniconda3/lib/python
3.9/site-packages (from seaborn) (1.23.4)
Requirement already satisfied: matplotlib!=3.6.1,>=3.1 in /opt/miniconda
3/lib/python3.9/site-packages (from seaborn) (3.6.2)
Requirement already satisfied: kiwisolver>=1.0.1 in /opt/miniconda3/lib/p
ython3.9/site-packages (from matplotlib!=3.6.1,>=3.1->seaborn) (1.4.4)
Requirement already satisfied: packaging>=20.0 in /opt/miniconda3/lib/pyt
hon3.9/site-packages (from matplotlib!=3.6.1,>=3.1->seaborn) (21.3)
Requirement already satisfied: pyparsing>=2.2.1 in /opt/miniconda3/lib/py
thon3.9/site-packages (from matplotlib!=3.6.1,>=3.1->seaborn) (3.0.9)
Requirement already satisfied: fonttools>=4.22.0 in /opt/miniconda3/lib/p
ython3.9/site-packages (from matplotlib!=3.6.1,>=3.1->seaborn) (4.38.0)
Requirement already satisfied: cycler>=0.10 in /opt/miniconda3/lib/python
3.9/site-packages (from matplotlib!=3.6.1,>=3.1->seaborn) (0.11.0)
Requirement already satisfied: pillow>=6.2.0 in /opt/miniconda3/lib/pytho
n3.9/site-packages (from matplotlib!=3.6.1,>=3.1->seaborn) (9.3.0)
Requirement already satisfied: contourpy>=1.0.1 in /opt/miniconda3/lib/py
thon3.9/site-packages (from matplotlib!=3.6.1,>=3.1->seaborn) (1.0.6)
Requirement already satisfied: python-dateutil>=2.7 in /opt/miniconda3/li
b/python3.9/site-packages (from matplotlib!=3.6.1,>=3.1->seaborn) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /opt/miniconda3/lib/python
3.9/site-packages (from pandas>=0.25->seaborn) (2022.5)
Requirement already satisfied: six>=1.5 in /opt/miniconda3/lib/python3.9/
site-packages (from python-dateutil>=2.7->matplotlib!=3.6.1,>=3.1->seabor
n) (1.16.0)
```

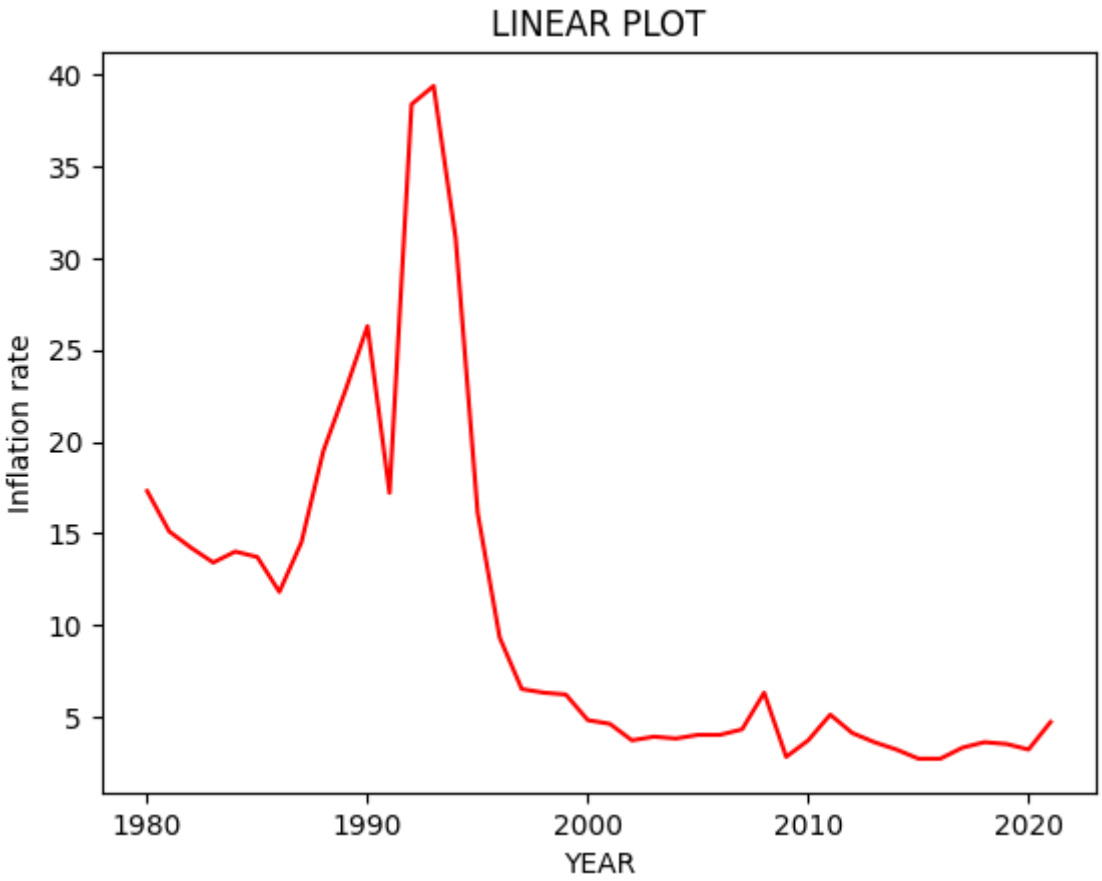
In [609]:
```
from sklearn import linear_model
import matplotlib.pyplot as plt
x = df['YEAR']
y = df['Inflation rate, end of period consumer prices (Annual percent chang
plt.scatter(x,y,c='r')
plt.xlabel('YEAR')
plt.ylabel("Inflation rate")
plt.title('SCATTERPLOT')
```

Out[609]:
```
Text(0.5, 1.0, 'SCATTERPLOT')
```



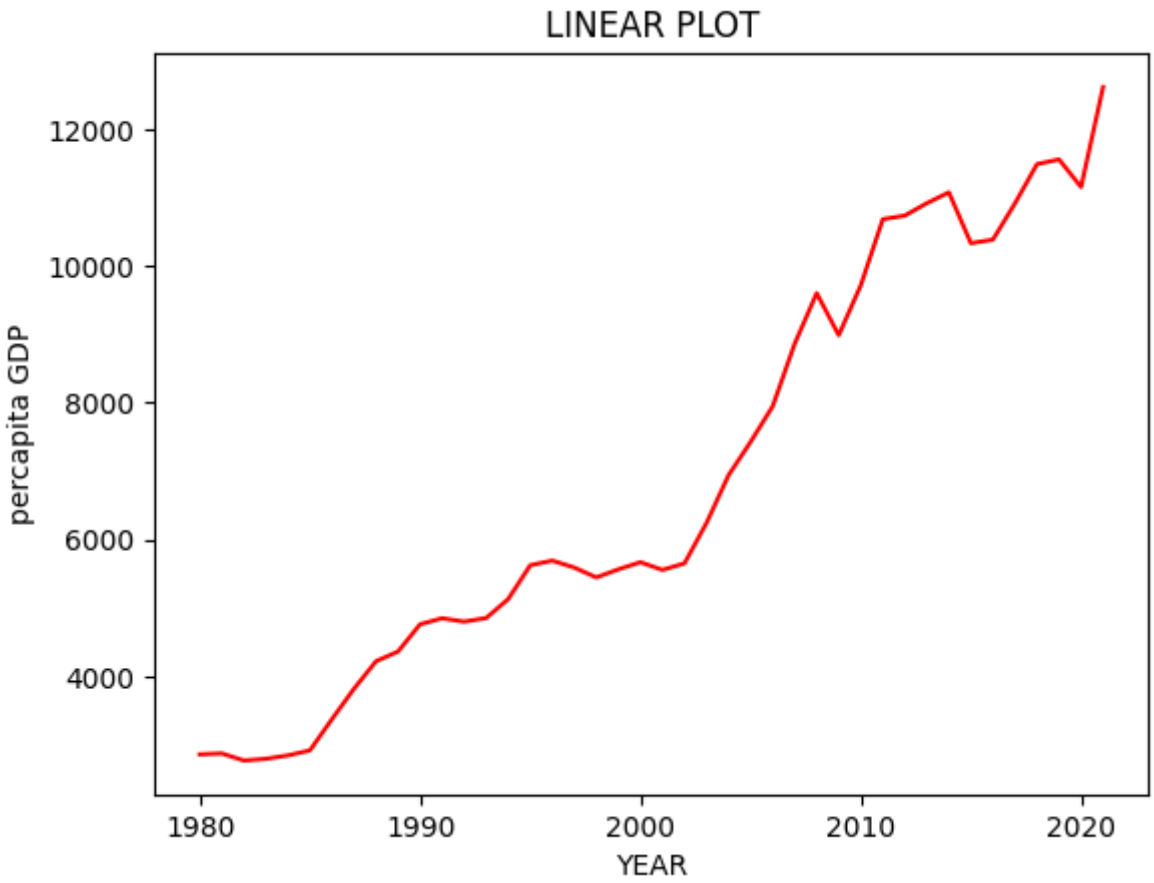
In [610]:
```
x = df['YEAR']
y = df['Inflation rate, end of period consumer prices (Annual percent chang
plt.plot(x,y,c='r')
plt.xlabel('YEAR')
plt.ylabel("Inflation rate")
plt.title('LINEAR PLOT')
```

Out[610]:
```
Text(0.5, 1.0, 'LINEAR PLOT')
```



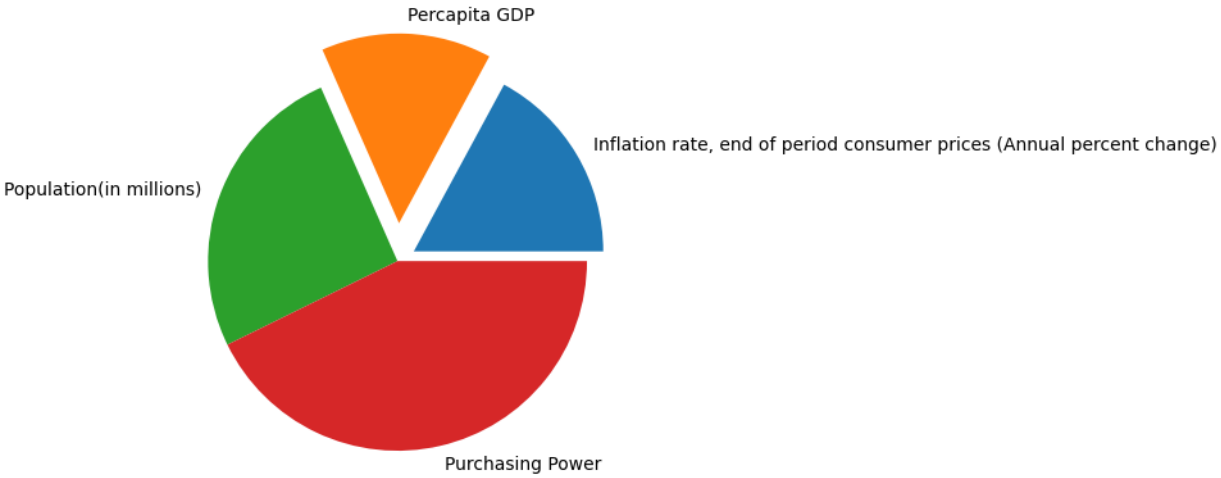
In [611]:
```
x = df['YEAR']
y = df['Percapita GDP']
plt.plot(x,y,c='r')
plt.xlabel('YEAR')
plt.ylabel("percapita GDP")
plt.title('LINEAR PLOT')
```

Out[611]: Text(0.5, 1.0, 'LINEAR PLOT')



In [612]:
```
labels = 'Inflation rate, end of period consumer prices (Annual percent cha
sizes = 200,170,300,500
explode=(0.1,0.2,0,0)

plt.pie(labels = labels,explode = explode,x=sizes ,shadow=False)
```

Out[612]:
```
([<matplotlib.patches.Wedge at 0x2e76d8670>,
  <matplotlib.patches.Wedge at 0x2e76d8b80>,
  <matplotlib.patches.Wedge at 0x2e76d8ca0>,
  <matplotlib.patches.Wedge at 0x2e76e6130>],
 [Text(1.0310816939863752, 0.6138978256405434, 'Inflation rate, end of pe
riod consumer prices (Annual percent change)'),
  Text(0.05234565821584228, 1.2989457001991846, 'Percapita GDP'),
  Text(-1.033661925916611, 0.3762220393735098, 'Population(in million
s)'),
  Text(0.24888508873373172, -1.0714738506403236, 'Purchasing Power')])
```



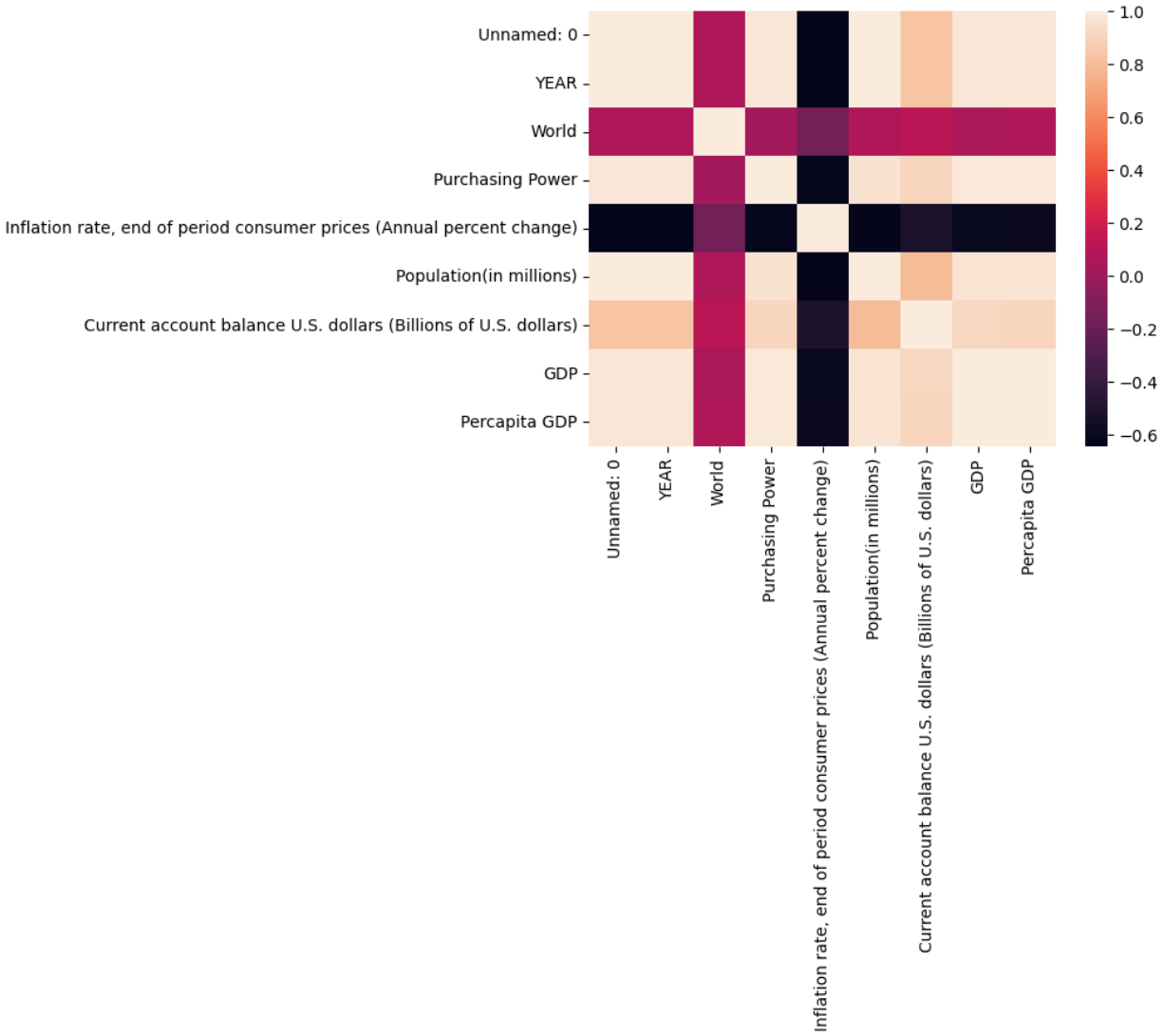
In [613]:
```
df.corr()
```

Out[613]:

	Unnamed: 0	YEAR	World	Purchasing Power	Inflation rate, end of period consumer prices (Annual percent change)	Population(in millions)	Current account balance U.S. dollars (Billions of U.S. dollars)	
Unnamed: 0	1.000000	1.000000	0.063291	0.977380	-0.642055	0.996897	0.827551	0.9
YEAR	1.000000	1.000000	0.063291	0.977380	-0.642055	0.996897	0.827551	0.9
World	0.063291	0.063291	1.000000	0.019050	-0.155893	0.066653	0.110453	0.0
Purchasing Power	0.977380	0.977380	0.019050	1.000000	-0.621182	0.960804	0.905191	0.9
Inflation rate, end of period consumer prices (Annual percent change)	-0.642055	-0.642055	-0.155893	-0.621182	1.000000	-0.628088	-0.522940	-0.6
Population(in millions)	0.996897	0.996897	0.066653	0.960804	-0.628088	1.000000	0.793068	0.9
Current account balance U.S. dollars (Billions of U.S. dollars)	0.827551	0.827551	0.110453	0.905191	-0.522940	0.793068	1.000000	0.9
GDP	0.976556	0.976556	0.044447	0.993374	-0.606028	0.962390	0.914349	1.0
Percapita GDP	0.975074	0.975074	0.070604	0.982618	-0.594032	0.963787	0.904934	0.9

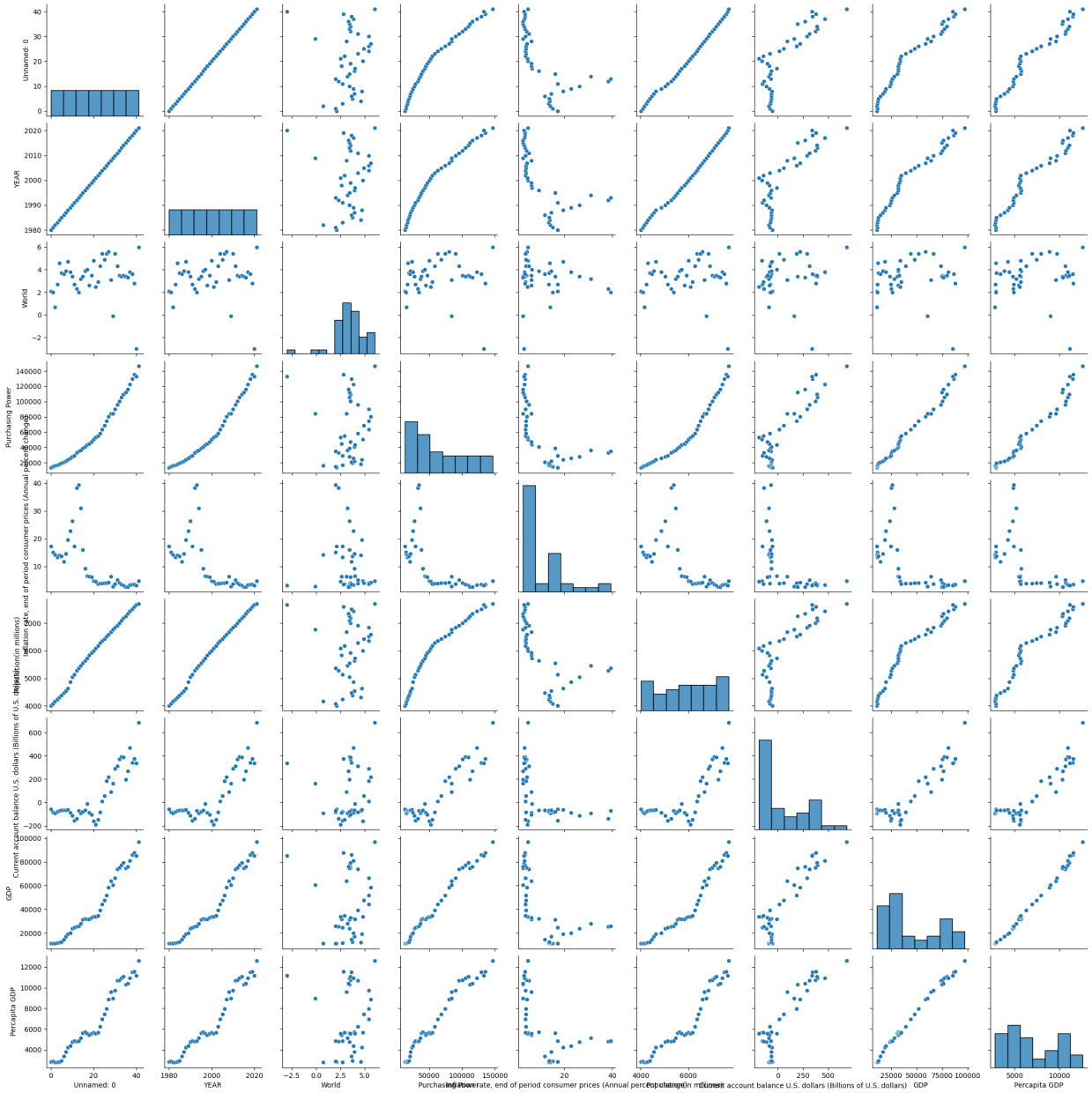
In [614]:
```
sns.heatmap(df.corr())
```

Out[614]: <AxesSubplot: >



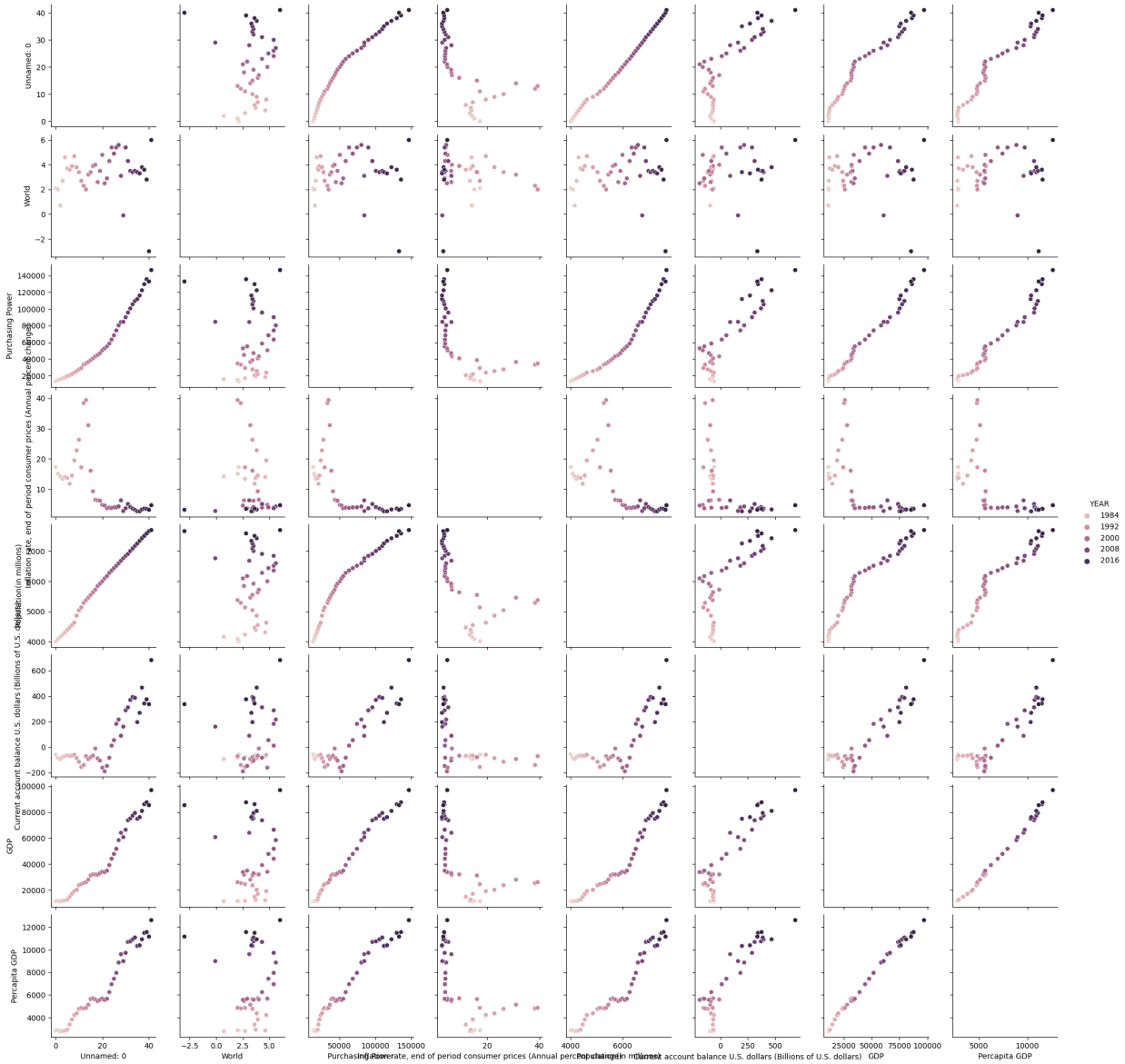
In [615]:
```
sns.pairplot(df)
```

Out[615]:
```
<seaborn.axisgrid.PairGrid at 0x2e76e6d30>
```



In [616]:
```
sns.pairplot(df,hue = 'YEAR')
```

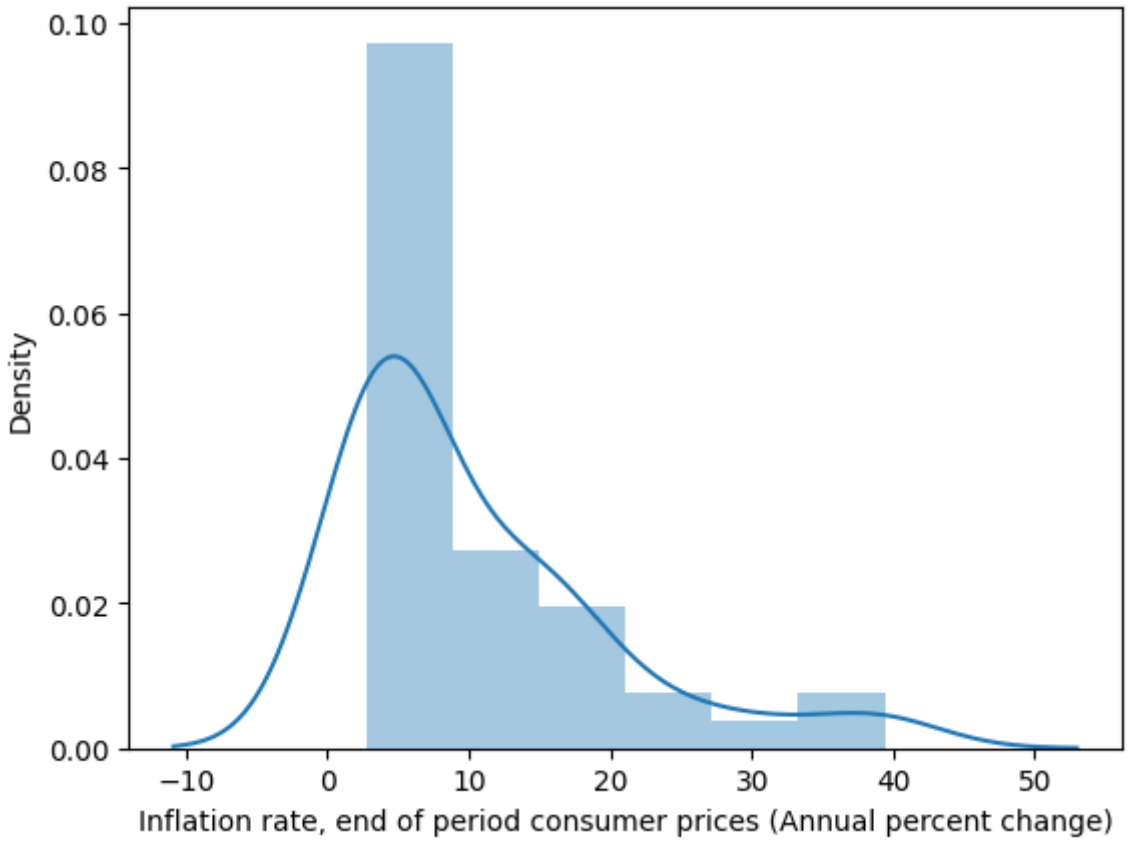
Out[616]: <seaborn.axisgrid.PairGrid at 0x2e9d5f2e0>



In [617]:
```
sns.distplot(df['Inflation rate, end of period consumer prices (Annual perc
```

```
/var/folders/19/1fkgq1fd0kx3kpcpbwd2phj00000gn/T/ipykernel_58472/2205912832.py:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751 (https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751)

  sns.distplot(df['Inflation rate, end of period consumer prices (Annual percent change)'])
```

Out[617]:
```
<AxesSubplot: xlabel='Inflation rate, end of period consumer prices (Annual percent change)', ylabel='Density'>
```



PREDICTION

In [618]:
```
X = df[['YEAR', 'Purchasing Power', 'Population(in millions)', 'Current acc
y = df['Inflation rate, end of period consumer prices (Annual percent chang
```

In [619]:
```
from sklearn.model_selection import train_test_split
```

In [620]:
```
x_train, x_test, y_train, y_test = train_test_split(X, y, test_size = 0.096
```

In [621]:
```
X
```

Out[621]:

	YEAR	Purchasing Power	Population(in millions)	Current account balance U.S. dollars (Billions of U.S. dollars)	GDP	Percapita GDP
0	1980	13400.209	4012.401	-57.235	11238.265	2862.333
1	1981	14930.382	4086.032	-83.002	11498.276	2876.220
2	1982	15909.301	4161.918	-92.745	11286.618	2772.215
3	1983	16934.331	4236.276	-76.953	11589.045	2797.085
4	1984	18321.225	4311.430	-69.007	12001.655	2846.673
5	1985	19579.824	4389.140	-64.732	12531.040	2920.230
6	1986	20669.482	4468.327	-67.732	14776.085	3376.821
7	1987	21981.840	4548.844	-65.442	17013.016	3820.734
8	1988	23788.830	4629.496	-59.458	19131.406	4222.994
9	1989	25630.201	4861.069	-85.635	20127.802	4365.323
10	1990	27659.306	5040.977	-114.323	23663.342	4763.550
11	1991	29295.591	5134.794	-154.599	24504.027	4851.567
12	1992	33270.148	5284.804	-139.277	25339.142	4802.195
13	1993	34719.277	5374.388	-69.766	26051.093	4855.191
14	1994	36509.056	5457.858	-92.485	27998.304	5131.819
15	1995	38640.040	5549.490	-78.080	31211.513	5626.243
16	1996	40857.982	5635.594	-66.214	32079.199	5694.385
17	1997	43242.683	5722.188	-9.909	31990.920	5592.904
18	1998	44926.541	5843.652	-88.193	31835.015	5448.350
19	1999	47125.482	5921.097	-104.878	32960.333	5566.836
20	2000	50498.195	5999.588	-160.066	34053.083	5671.185
21	2001	52878.461	6094.637	-188.910	33801.014	5558.803
22	2002	55218.306	6171.644	-146.456	34915.511	5653.864
23	2003	58589.628	6278.040	-82.434	39210.202	6245.003
24	2004	63335.218	6356.212	12.633	44134.371	6943.046
25	2005	68426.916	6435.031	55.248	47811.372	7429.368
26	2006	74266.032	6514.383	182.975	51783.269	7948.537
27	2007	80392.859	6595.180	217.264	58461.908	8863.739
28	2008	84371.972	6680.705	89.656	64162.470	9603.535
29	2009	84583.956	6760.972	161.390	60782.123	8989.602
30	2010	90151.344	6841.923	288.896	66484.526	9717.228
31	2011	95701.819	6904.905	312.796	73773.480	10683.647

	YEAR	Purchasing Power	Population(in millions)	Current account balance U.S. dollars (Billions of U.S. dollars)	GDP	Percapita GDP
32	2012	100691.800	7003.035	370.624	75196.673	10737.727
33	2013	105648.957	7085.831	394.241	77365.520	10918.342
34	2014	109595.081	7170.665	386.824	79429.015	11076.939
35	2015	111857.084	7252.775	196.854	74944.460	10333.211
36	2016	116168.540	7337.708	269.823	76211.252	10386.247
37	2017	122351.472	7423.245	467.424	81036.151	10916.540
38	2018	129708.966	7503.469	343.193	86209.627	11489.302
39	2019	135641.445	7583.444	375.861	87654.342	11558.645
40	2020	132936.143	7659.052	337.694	85440.667	11155.514
41	2021	146607.921	7694.524	683.256	97076.276	12616.281

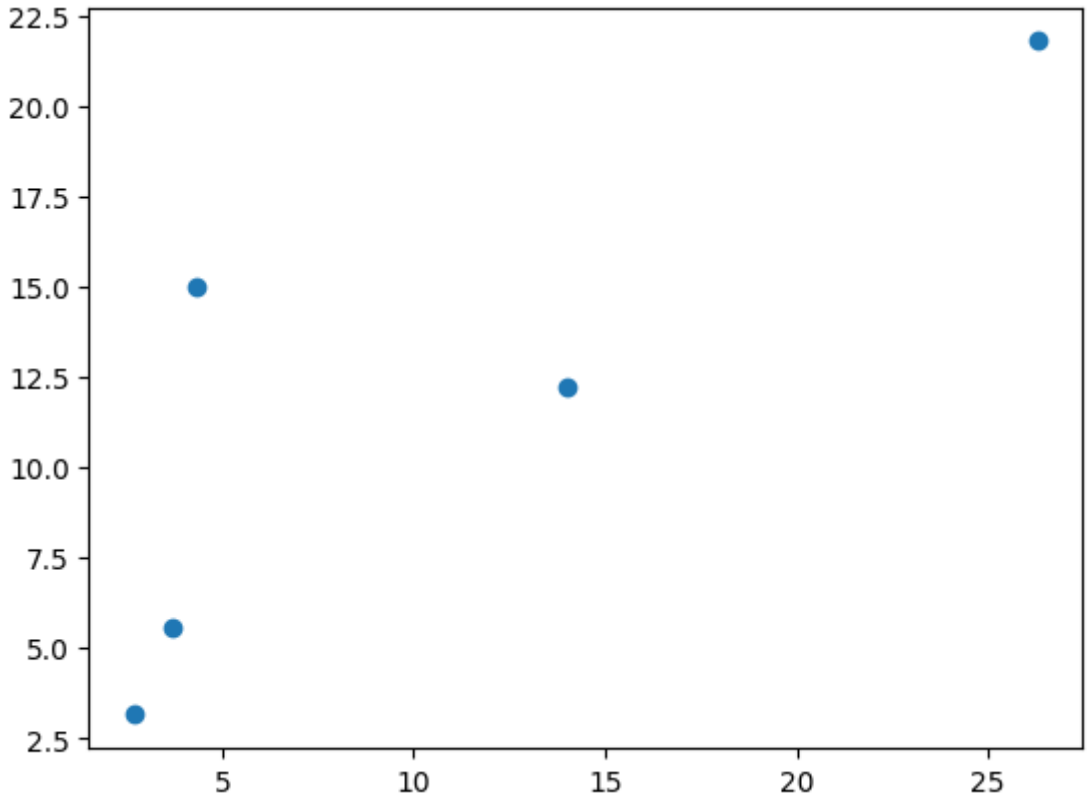
In [622]:
```
regr = linear_model.LinearRegression()
regr.fit(x_train, y_train)
```

Out[622]:
```
▾ LinearRegression

LinearRegression()
```

In [623]:
```
predictions = regr.predict(x_test)
```

In [624]:
```
plt.scatter(y_test, predictions)
```

Out[624]:
```
<matplotlib.collections.PathCollection at 0x2f55eedc0>
```



In [625]:
```
regr.coef_
regr.score(x_test, y_test)
```

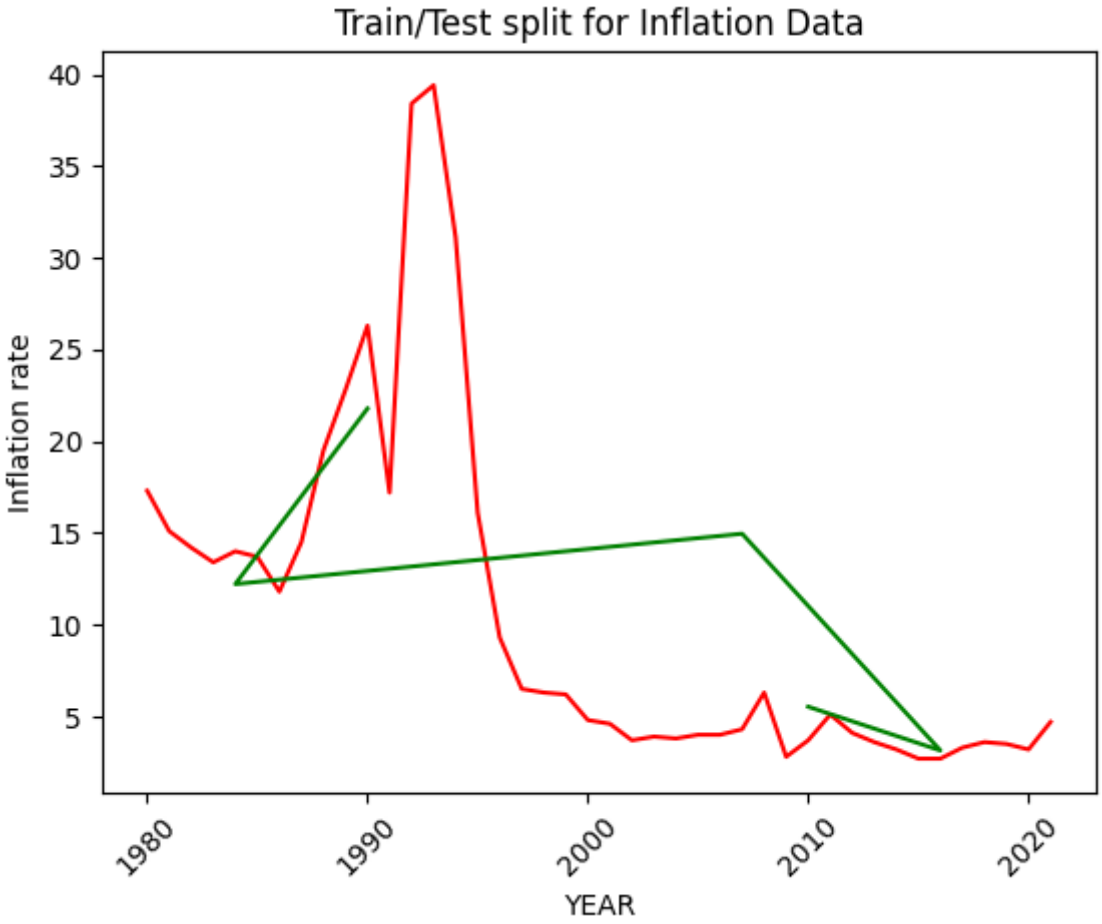
Out[625]:
```
0.6541693440929648
```

In [626]:
```
# predict the y values
y_pred=regr.predict(x_test)
# a data frame with actual and predicted values of y
evaluate = pd.DataFrame({"Predicted": y_pred.flatten()})
evaluate.head()
```

Out[626]:

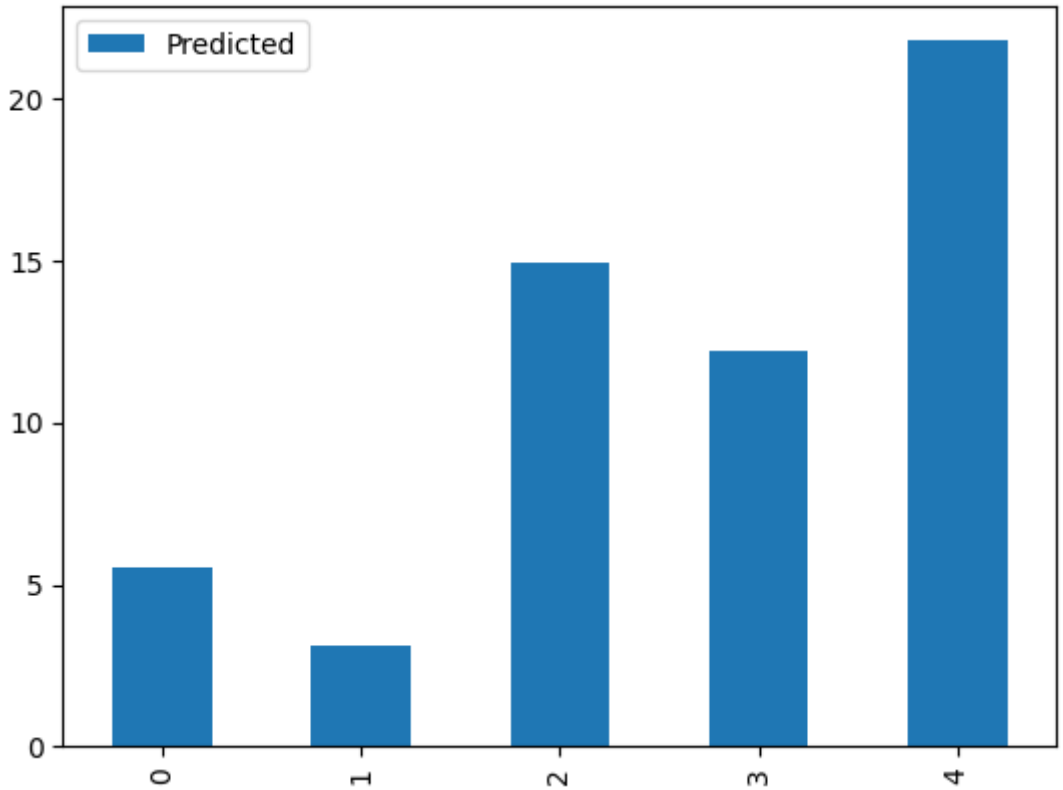
	Predicted
0	5.529925
1	3.143321
2	14.957774
3	12.214296
4	21.781460

In [627]:
```
# plt.plot(x_train['YEAR'],y_train, color = "black")
plt.plot(X['YEAR'],y, color = "red")
plt.plot(x_test['YEAR'],evaluate['Predicted'], color='green', label = 'Pred
plt.ylabel('Inflation rate')
plt.xlabel('YEAR')
plt.xticks(rotation=45)
plt.title("Train/Test split for Inflation Data")
plt.show()
```



In [628]:
```
evaluate.head(10).plot(kind = "bar")
```

Out[628]: <AxesSubplot: >



In [571]:
```
X = df[['YEAR', 'Purchasing Power', 'Population(in millions)', 'Current acc
y = df['GDP']
```

In [572]:
```
x_train, x_test1, y_train, y_test = train_test_split(X, y, test_size = 0.09
```

In [573]:
```
regr2 = linear_model.LinearRegression()
regr2.fit(x_train, y_train)
```

Out[573]:
```
▾ LinearRegression

LinearRegression()
```

In [574]:
```
regr2.coef_
regr2.score(x_test1,y_test)
```

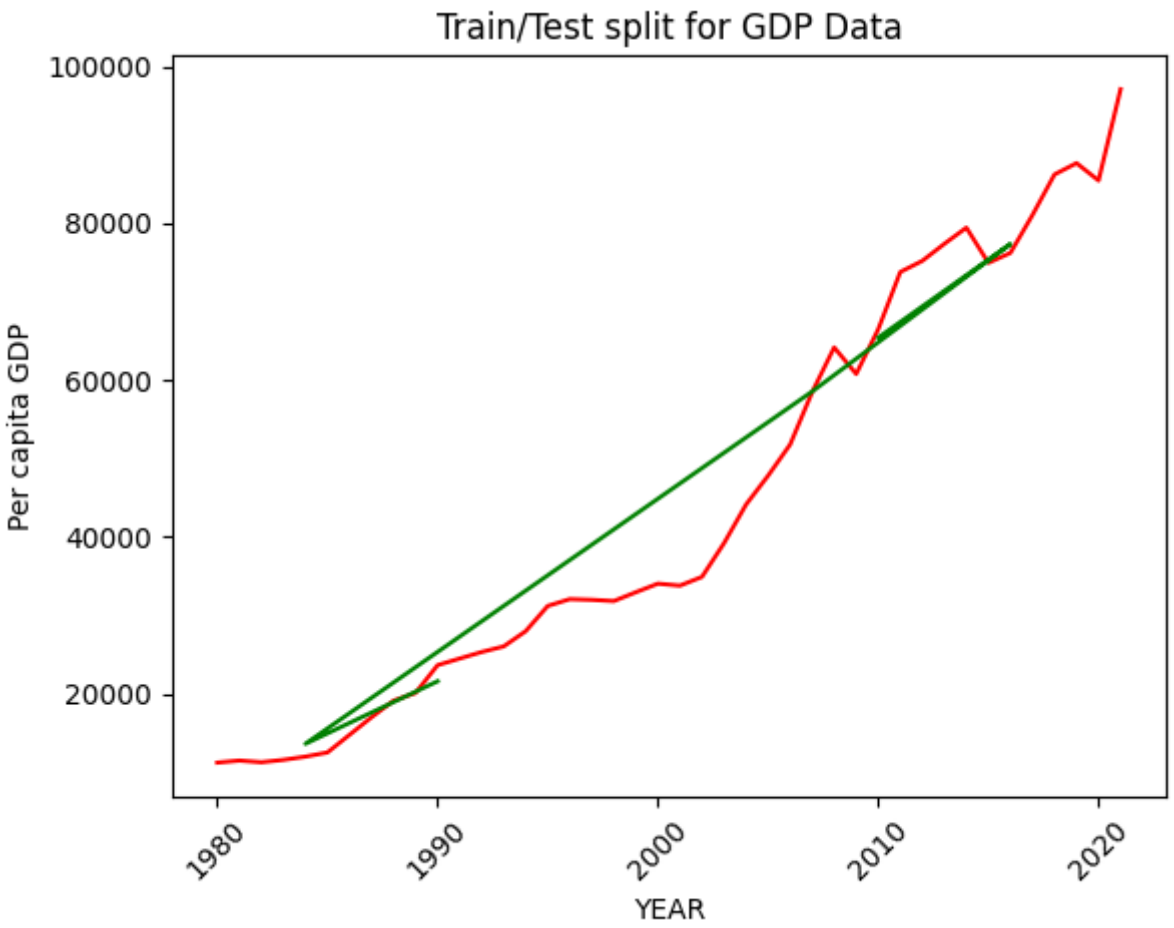
Out[574]:
```
0.9970257164576323
```

In [575]:
```
# predict the y values
y_pred=regr2.predict(x_test1)
# a data frame with actual and predicted values of y
evaluate2 = pd.DataFrame({"Actual": y_test.values.flatten(), "Predicted": y
evaluate2.head()
```

Out[575]:

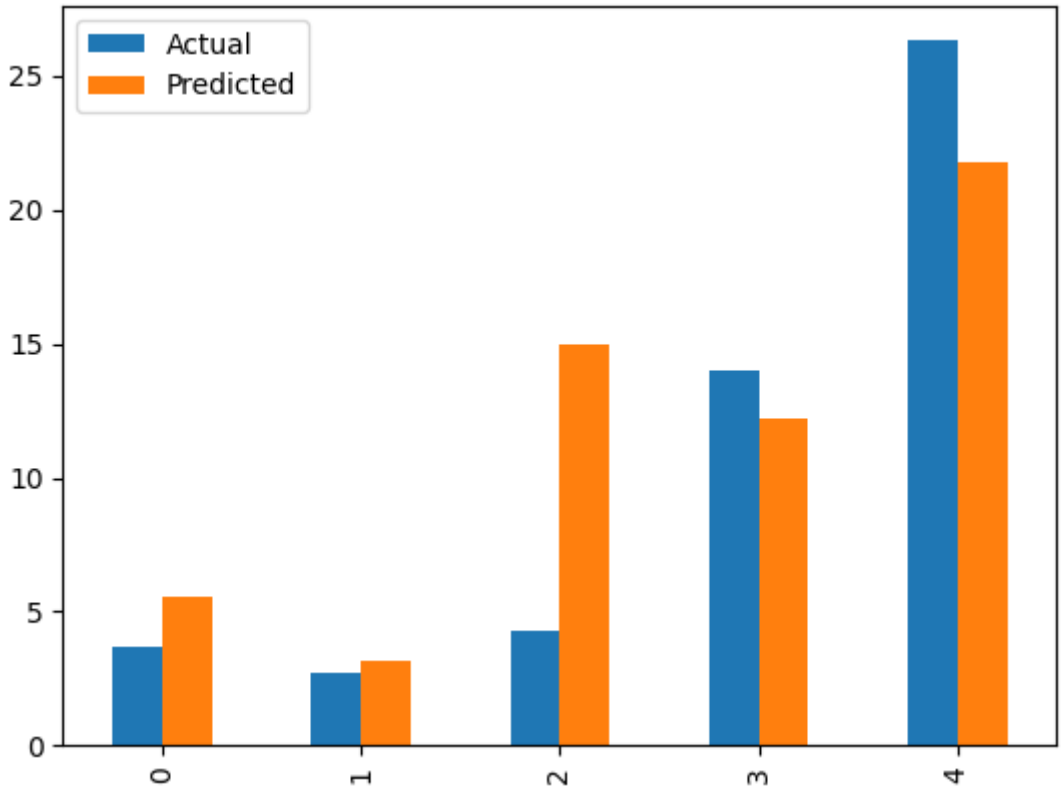
	Actual	Predicted
0	66484.526	65411.977035
1	76211.252	77341.329113
2	58461.908	58543.953076
3	12001.655	13636.058397
4	23663.342	21610.684863

In [576]:
```
# plt.plot(x_train['YEAR'],y_train, color = "black")
plt.plot(X['YEAR'],y, color = "red")
plt.plot(x_test1['YEAR'],evaluate2['Predicted'], color='green', label = 'Pr
plt.ylabel('Per capita GDP')
plt.xlabel('YEAR')
plt.xticks(rotation=45)
plt.title("Train/Test split for GDP Data")
plt.show()
```

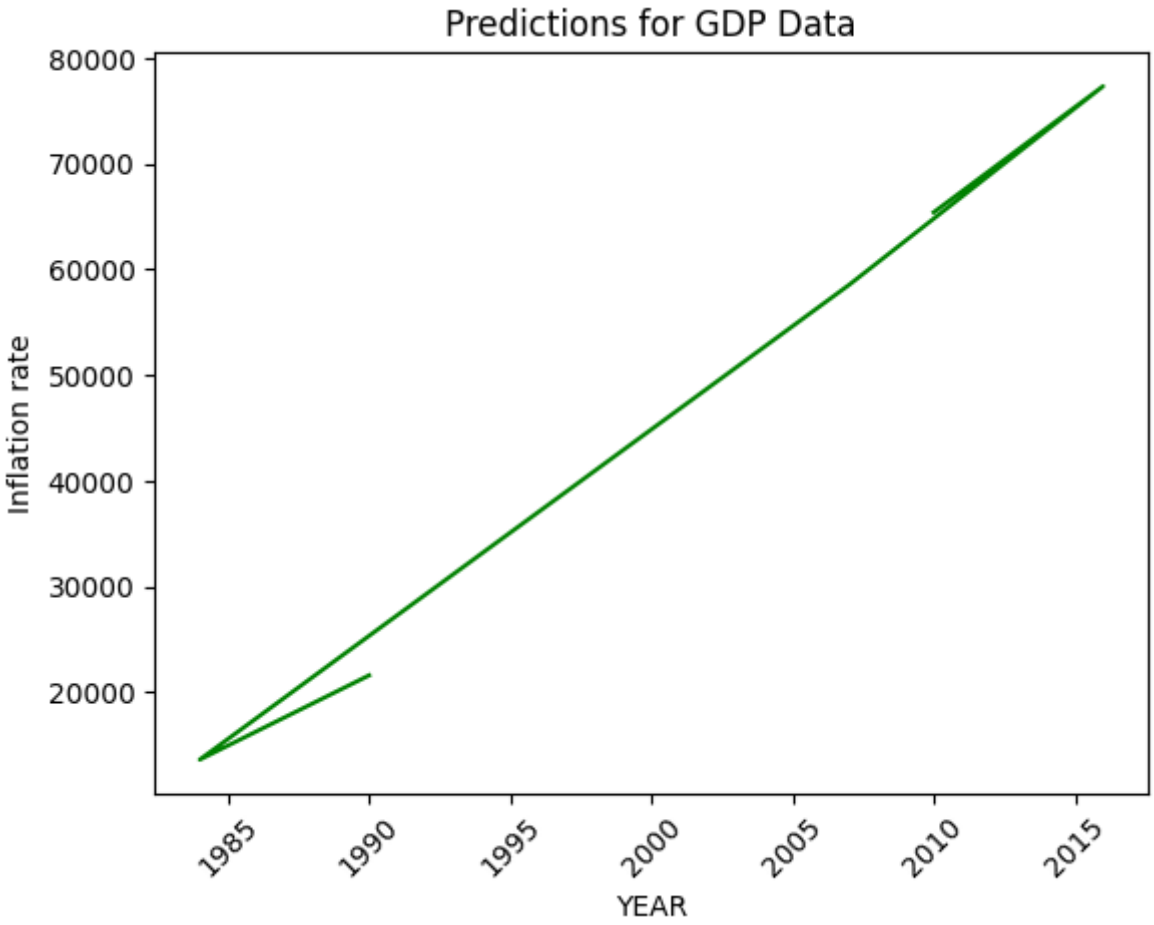


In [577]:
```
evaluate.head(10).plot(kind = "bar")
```

Out[577]: <AxesSubplot: >



In [579]:
```
plt.plot(x_test1['YEAR'],evaluate2['Predicted'], color='green', label = 'Pr
plt.ylabel('Inflation rate')
plt.xlabel('YEAR')
plt.xticks(rotation=45)
plt.title("Predictions for GDP Data")
plt.show()
```



In [580]:
```
!pip install pandas-datareader
```

```
Requirement already satisfied: pandas-datareader in /opt/miniconda3/lib/p
ython3.9/site-packages (0.10.0)
Requirement already satisfied: pandas>=0.23 in /opt/miniconda3/lib/python
3.9/site-packages (from pandas-datareader) (1.5.1)
Requirement already satisfied: lxml in /opt/miniconda3/lib/python3.9/site
-packages (from pandas-datareader) (4.9.1)
Requirement already satisfied: requests>=2.19.0 in /opt/miniconda3/lib/py
thon3.9/site-packages (from pandas-datareader) (2.27.1)
Requirement already satisfied: python-dateutil>=2.8.1 in /opt/miniconda3/
lib/python3.9/site-packages (from pandas>=0.23->pandas-datareader) (2.8.
2)
Requirement already satisfied: pytz>=2020.1 in /opt/miniconda3/lib/python
3.9/site-packages (from pandas>=0.23->pandas-datareader) (2022.5)
Requirement already satisfied: numpy>=1.20.3 in /opt/miniconda3/lib/pytho
n3.9/site-packages (from pandas>=0.23->pandas-datareader) (1.23.4)
Requirement already satisfied: six>=1.5 in /opt/miniconda3/lib/python3.9/
site-packages (from python-dateutil>=2.8.1->pandas>=0.23->pandas-dataread
er) (1.16.0)
Requirement already satisfied: certifi>=2017.4.17 in /opt/miniconda3/lib/
python3.9/site-packages (from requests>=2.19.0->pandas-datareader) (2022.
9.24)
Requirement already satisfied: charset-normalizer~=2.0.0 in /opt/minicond
a3/lib/python3.9/site-packages (from requests>=2.19.0->pandas-datareader)
(2.0.4)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /opt/miniconda3/l
ib/python3.9/site-packages (from requests>=2.19.0->pandas-datareader) (1.
26.9)
Requirement already satisfied: idna<4,>=2.5 in /opt/miniconda3/lib/python
3.9/site-packages (from requests>=2.19.0->pandas-datareader) (3.3)
```

In [581]:
```
!pip install statsmodels
from statsmodels.tsa.statespace.sarimax import SARIMAX
```

```
Requirement already satisfied: statsmodels in /opt/miniconda3/lib/python
3.9/site-packages (0.13.5)
Requirement already satisfied: patsy>=0.5.2 in /opt/miniconda3/lib/python
3.9/site-packages (from statsmodels) (0.5.3)
Requirement already satisfied: packaging>=21.3 in /opt/miniconda3/lib/pyt
hon3.9/site-packages (from statsmodels) (21.3)
Requirement already satisfied: scipy>=1.3 in /opt/miniconda3/lib/python3.
9/site-packages (from statsmodels) (1.9.3)
Requirement already satisfied: numpy>=1.17 in /opt/miniconda3/lib/python
3.9/site-packages (from statsmodels) (1.23.4)
Requirement already satisfied: pandas>=0.25 in /opt/miniconda3/lib/python
3.9/site-packages (from statsmodels) (1.5.1)
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in /opt/miniconda
3/lib/python3.9/site-packages (from packaging>=21.3->statsmodels) (3.0.9)
Requirement already satisfied: pytz>=2020.1 in /opt/miniconda3/lib/python
3.9/site-packages (from pandas>=0.25->statsmodels) (2022.5)
Requirement already satisfied: python-dateutil>=2.8.1 in /opt/miniconda3/
lib/python3.9/site-packages (from pandas>=0.25->statsmodels) (2.8.2)
Requirement already satisfied: six in /opt/miniconda3/lib/python3.9/site-
packages (from patsy>=0.5.2->statsmodels) (1.16.0)
```

In [582]:
```
ARMAmodel = SARIMAX(y_train, order = (1, 0, 1))
```

```
/opt/miniconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_mode
l.py:471: ValueWarning: An unsupported index was provided and will be ign
ored when e.g. forecasting.
  self._init_dates(dates, freq)
/opt/miniconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_mode
l.py:471: ValueWarning: An unsupported index was provided and will be ign
ored when e.g. forecasting.
  self._init_dates(dates, freq)
```

In [583]:
```
ARMAmodel = ARMAmodel.fit()
```

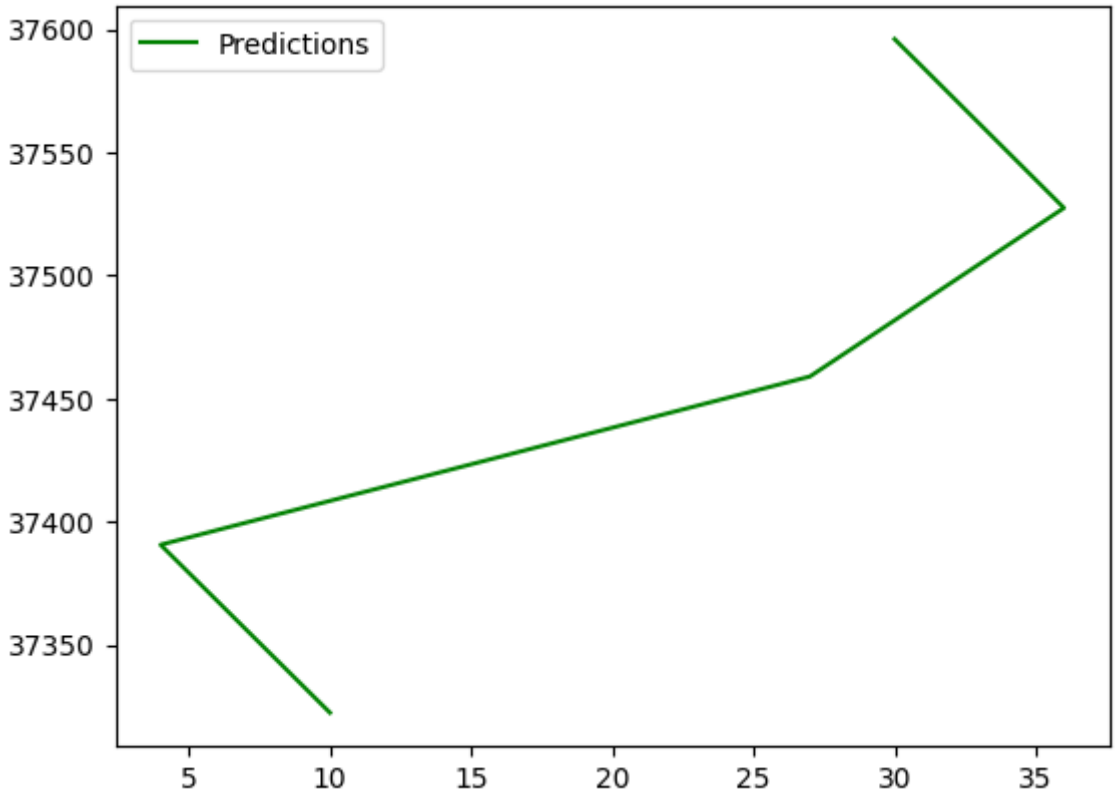
```
RUNNING THE L-BFGS-B CODE

           * * *

Machine precision = 2.220D-16
 N =            3     M =           10

At X0         0 variables are exactly at the bounds

At iterate    0    f=  1.33234D+01    |proj g|=  5.72894D-01

At iterate    5    f=  1.17073D+01    |proj g|=  3.73101D-03

At iterate   10    f=  1.17072D+01    |proj g|=  4.75220D-04

At iterate   15    f=  1.17072D+01    |proj g|=  1.74390D-05

           * * *

Tit   = total number of iterations
Tnf   = total number of function evaluations
Tnint = total number of segments explored during Cauchy searches
Skip  = number of BFGS updates skipped
Nact  = number of active bounds at final generalized Cauchy point
Projg = norm of the final projected gradient
F     = final function value

           * * *

   N    Tit     Tnf  Tnint  Skip  Nact     Projg        F
    3     16     23      1     0     0   2.779D-05   1.171D+01
  F =   11.707154575703392

CONVERGENCE: REL_REDUCTION_OF_F_<=_FACTR*EPSMCH


 This problem is unconstrained.
```

In [584]:
```
y_pred = ARMAmodel.get_forecast(len(y_test.index))
y_pred_df = y_pred.conf_int(alpha = 0.05)
y_pred_df["Predictions"] = ARMAmodel.predict(start = y_pred_df.index[0], en
y_pred_df.index = y_test.index
y_pred_out = y_pred_df["Predictions"]
```

```
/opt/miniconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_mode
l.py:834: ValueWarning: No supported index is available. Prediction resul
ts will be given with an integer index beginning at `start`.
  return get_prediction_index(
/opt/miniconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_mode
l.py:834: ValueWarning: No supported index is available. Prediction resul
ts will be given with an integer index beginning at `start`.
  return get_prediction_index(
```

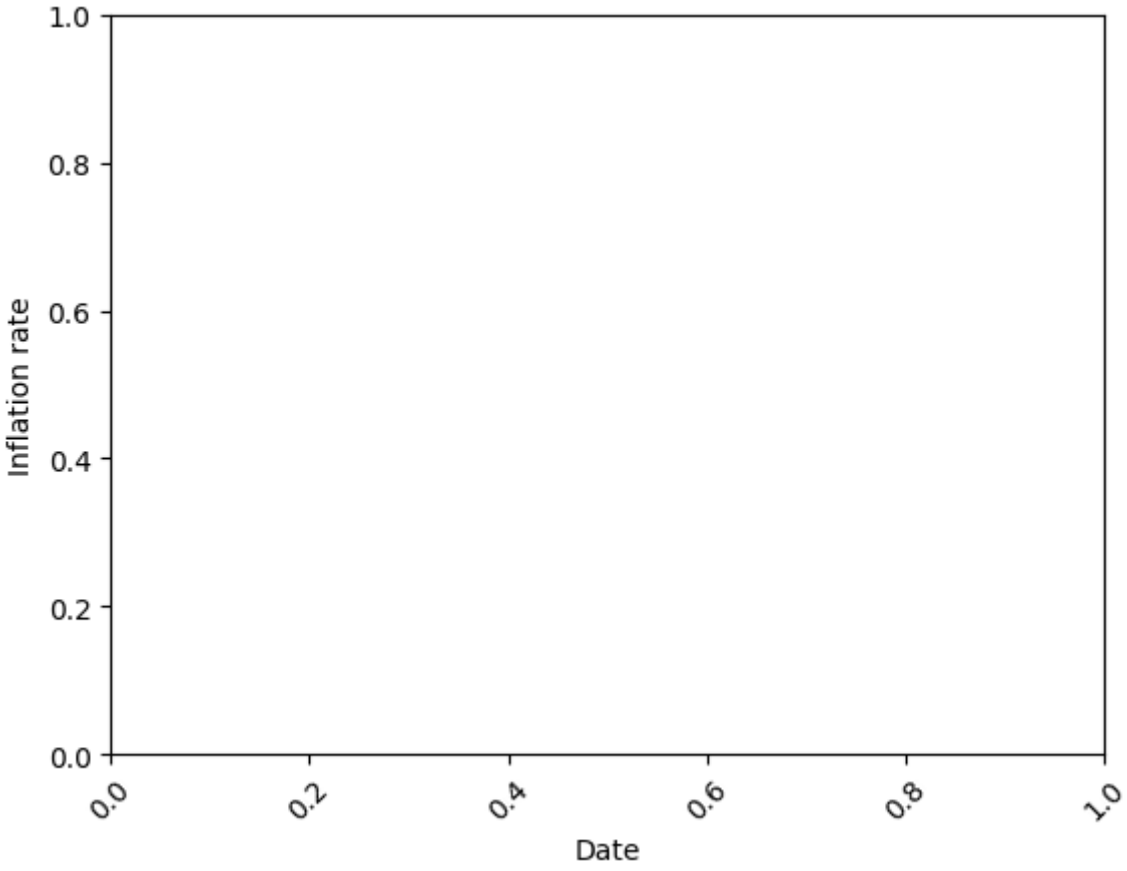
In [585]:
```
plt.plot(y_pred_out, color='green', label = 'Predictions')
plt.legend()
```

Out[585]: <matplotlib.legend.Legend at 0x2a5751d00>



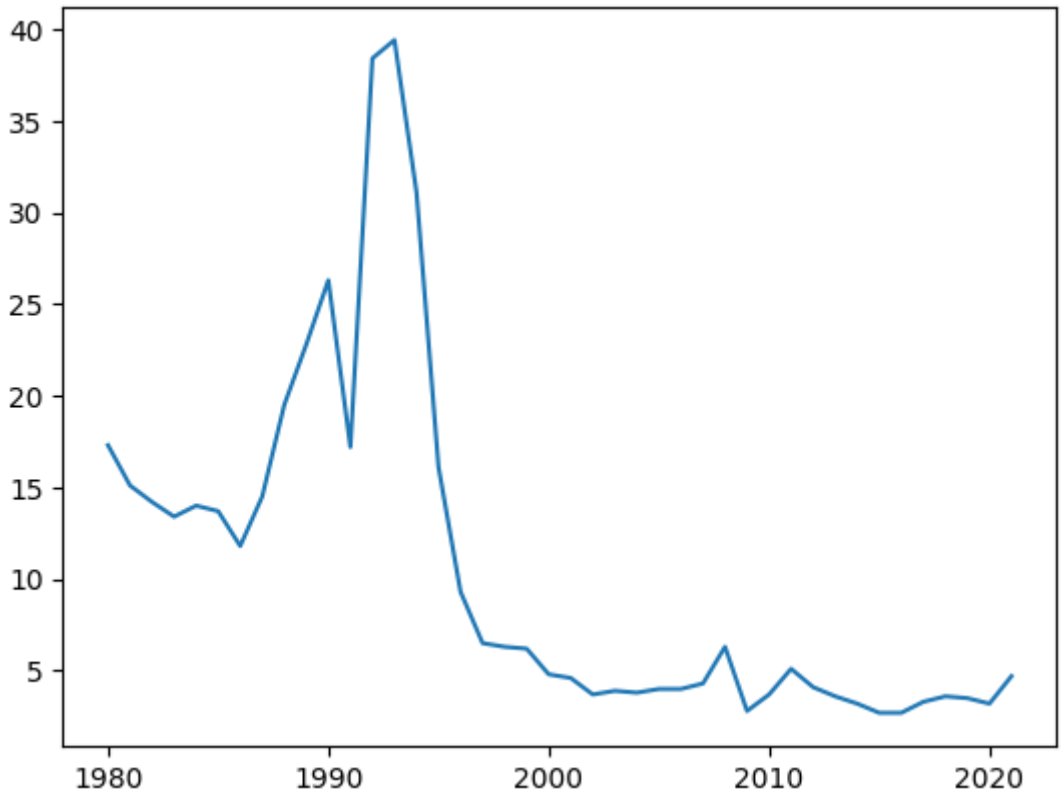
In [586]:
```
import matplotlib.pyplot as plt
import seaborn as sns
plt.ylabel('Inflation rate')
plt.xlabel('Date')
plt.xticks(rotation=45)
```

Out[586]:
```
(array([0. , 0.2, 0.4, 0.6, 0.8, 1. ]),
 [Text(0.0, 0, '0.0'),
  Text(0.2, 0, '0.2'),
  Text(0.4, 0, '0.4'),
  Text(0.6000000000000001, 0, '0.6'),
  Text(0.8, 0, '0.8'),
  Text(1.0, 0, '1.0')])
```



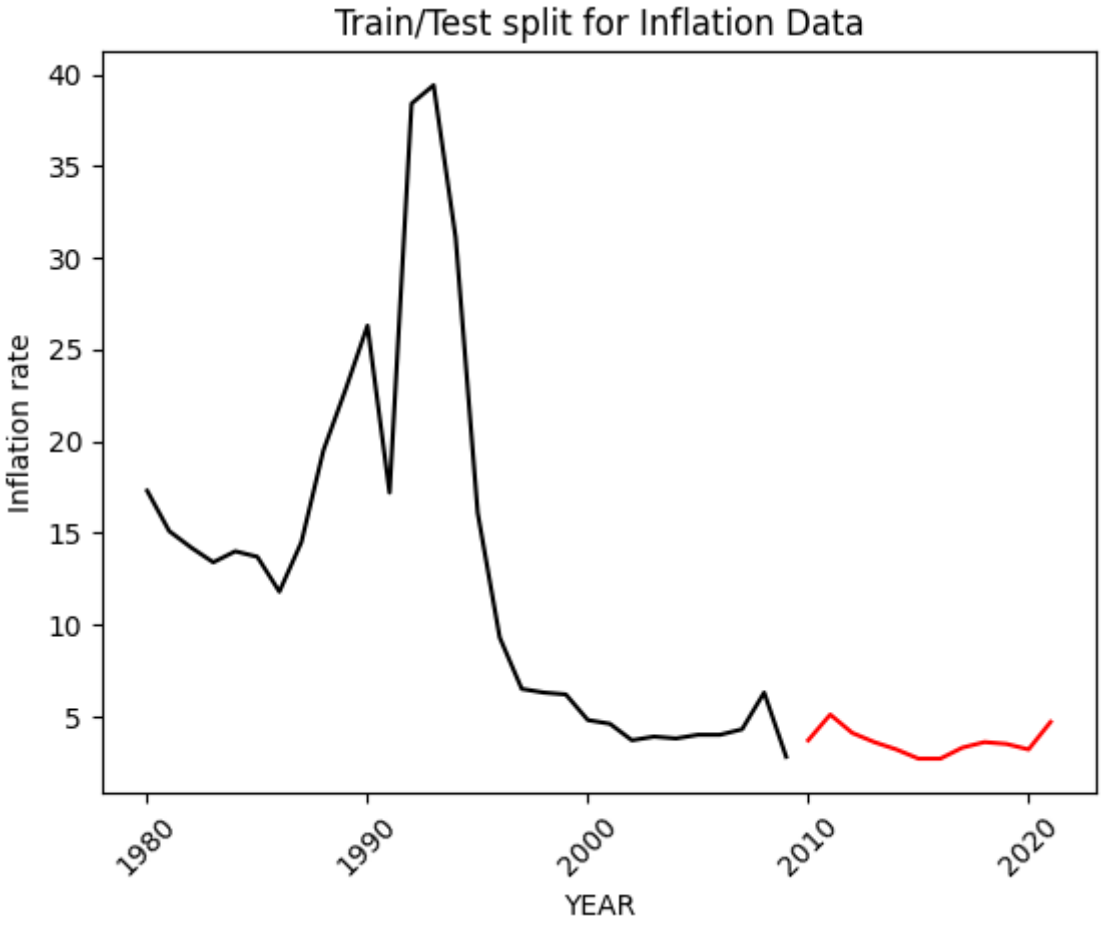
In [587]:
```
plt.plot(df['YEAR'], df['Inflation rate, end of period consumer prices (Ann
```

Out[587]:
```
[<matplotlib.lines.Line2D at 0x2e8223cd0>]
```



In [588]:
```
train = df[df['YEAR']<2010]
test = df[df['YEAR']>2009]
```

In [589]:
```
plt.plot(train['YEAR'],train['Inflation rate, end of period consumer prices
plt.plot(test['YEAR'],test['Inflation rate, end of period consumer prices (
plt.ylabel('Inflation rate')
plt.xlabel('YEAR')
plt.xticks(rotation=45)
plt.title("Train/Test split for Inflation Data")
plt.show()
```



In [590]:
```
from statsmodels.tsa.statespace.sarimax import SARIMAX
```

In [591]:
```
y = train['Inflation rate, end of period consumer prices (Annual percent ch
```

In [592]:
```
ARMAmodel = SARIMAX(y, order = (1, 0, 1))
```

In [593]:
```
ARMAmodel = ARMAmodel.fit()
```

```
RUNNING THE L-BFGS-B CODE

           * * *

Machine precision = 2.220D-16
 N =            3     M =           10

At X0         0 variables are exactly at the bounds

At iterate    0    f=  3.18217D+00    |proj g|=  1.55251D-02

At iterate    5    f=  3.17976D+00    |proj g|=  1.07553D-03

           * * *

Tit   = total number of iterations
Tnf   = total number of function evaluations
Tnint = total number of segments explored during Cauchy searches
Skip  = number of BFGS updates skipped
Nact  = number of active bounds at final generalized Cauchy point
Projg = norm of the final projected gradient
F     = final function value

           * * *

   N    Tit     Tnf  Tnint  Skip  Nact     Projg        F
    3      8     10      1     0     0   8.376D-06   3.180D+00
  F =   3.1797442645738645

CONVERGENCE: NORM_OF_PROJECTED_GRADIENT_<=_PGTOL

 This problem is unconstrained.
```

In [594]:
```
test
```

Out[594]:

	Unnamed: 0	YEAR	World	Purchasing Power	Inflation rate, end of period consumer prices (Annual percent change)	Population(in millions)	Current account balance U.S. dollars (Billions of U.S. dollars)	GDP	Percapita GDP
30	30	2010	5.4	90151.344	3.7	6841.923	288.896	66484.526	9717.228
31	31	2011	4.3	95701.819	5.1	6904.905	312.796	73773.480	10683.647
32	32	2012	3.5	100691.800	4.1	7003.035	370.624	75196.673	10737.727
33	33	2013	3.4	105648.957	3.6	7085.831	394.241	77365.520	10918.342
34	34	2014	3.5	109595.081	3.2	7170.665	386.824	79429.015	11076.939
35	35	2015	3.4	111857.084	2.7	7252.775	196.854	74944.460	10333.211
36	36	2016	3.3	116168.540	2.7	7337.708	269.823	76211.252	10386.247
37	37	2017	3.8	122351.472	3.3	7423.245	467.424	81036.151	10916.540
38	38	2018	3.6	129708.966	3.6	7503.469	343.193	86209.627	11489.302
39	39	2019	2.8	135641.445	3.5	7583.444	375.861	87654.342	11558.645
40	40	2020	-3.0	132936.143	3.2	7659.052	337.694	85440.667	11155.514
41	41	2021	6.0	146607.921	4.7	7694.524	683.256	97076.276	12616.281

In [595]:
```
y_pred = ARMAmodel.get_forecast(len(test['YEAR']))
y_pred_df = y_pred.conf_int(alpha = 0.05)
y_pred_df["Predictions"] = ARMAmodel.predict(start = y_pred_df.index[0], en
y_pred_df['YEAR'] = test['YEAR']
y_pred_out = y_pred_df["Predictions"]
```

In [596]:
```
print(y_pred_out)
```

```
30    2.244675
31    2.060737
32    1.891872
33    1.736844
34    1.594520
35    1.463858
36    1.343904
37    1.233779
38    1.132678
39    1.039861
40    0.954651
41    0.876423
Name: Predictions, dtype: float64
```

In [597]:
```
lt.plot(train['YEAR'],train['Inflation rate, end of period consumer prices
lt.plot(test['YEAR'],test['Inflation rate, end of period consumer prices (A
lt.plot(y_pred_df['YEAR'],y_pred_df["Predictions"], color='green', label =
lt.ylabel('Inflation rate')
lt.xlabel('YEAR')
lt.xticks(rotation=45)
lt.title("Train/Test split for Inflation Data")
lt.show()
```