PROGRAM 9: TIMESERIES ANALYSIS USING PYTHON

REQUIREMENTS:

Mr. António Guterres, the UNO secretary general wants to go through the global analytical report of GDP availabl
e in his office. Kindly provide a bird view by implementing various time series models using available data.

IMPORTING LIBRARIES

In [2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import matplotlib.gridspec as gridspec
import plotly.express as px
```

IMPORTING DATASET AND DISPLAYING IT

In [3]:
```
data = pd.read_csv ('gdp.csv')
data.head()
```

Out[3]:

	Year	Afghanistan	Albania	Algeria	Angola	Antigua and Barbuda	Argentina	Armenia	Australia	Austria	...	United Kingdom	United States	Uruguay	Uzbekistan	Vanua
0	1960	4.13	NaN	39.0	NaN	NaN	7.60	NaN	13.0	23.2	...	20.2	4.97	13.8	NaN	Na
1	1961	4.45	NaN	46.2	NaN	NaN	5.99	NaN	12.4	23.0	...	19.9	4.90	14.2	NaN	Na
2	1962	4.88	NaN	19.8	NaN	NaN	4.69	NaN	13.9	23.4	...	19.4	4.81	11.3	NaN	Na
3	1963	9.17	NaN	24.7	NaN	NaN	7.89	NaN	13.0	23.4	...	19.3	4.87	12.0	NaN	Na
4	1964	8.89	NaN	25.1	NaN	NaN	5.56	NaN	14.9	23.4	...	18.7	5.10	11.9	NaN	Na

5 rows × 186 columns

HANDLING NaN VALUES

In [4]:
```
data = data.fillna(0)
data.head()
```

Out[4]:

	Year	Afghanistan	Albania	Algeria	Angola	Antigua and Barbuda	Argentina	Armenia	Australia	Austria	...	United Kingdom	United States	Uruguay	Uzbekistan	Vanua
0	1960	4.13	0.0	39.0	0.0	0.0	7.60	0.0	13.0	23.2	...	20.2	4.97	13.8	0.0	0
1	1961	4.45	0.0	46.2	0.0	0.0	5.99	0.0	12.4	23.0	...	19.9	4.90	14.2	0.0	0
2	1962	4.88	0.0	19.8	0.0	0.0	4.69	0.0	13.9	23.4	...	19.4	4.81	11.3	0.0	0
3	1963	9.17	0.0	24.7	0.0	0.0	7.89	0.0	13.0	23.4	...	19.3	4.87	12.0	0.0	0
4	1964	8.89	0.0	25.1	0.0	0.0	5.56	0.0	14.9	23.4	...	18.7	5.10	11.9	0.0	0

5 rows × 186 columns

Gross domestic product by country allows you to compare the economies of the world's nations. It measures everything produced by everyone in the country whether they are citizens or foreigners.

In [5]:
```
df1 = data[40:]
df2 = df1.drop(columns=['Year'])
cm = sns.light_palette("brown", as_cmap=True)
df2.style.background_gradient(cmap=cm)
```

Out[5]:

	Afghanistan	Albania	Algeria	Angola	Antigua and Barbuda	Argentina	Armenia	Australia	Austria	Azerbaijan	Bahamas	Bahrain	Bangladesh	Barbados
40	0	17.9	42.1	89.7	56.2	11	23.4	19.4	43.3	39	38	79.2	12.3	42.4
41	0	18.4	36.7	75.4	55.6	11.6	25.5	22.2	44.6	40.9	34.1	73.6	13.4	40.2
42	32.4	19.6	35.5	68.3	52.5	28.4	29.4	20.8	45.3	42.8	36.2	72.2	12.4	38.2
43	43.6	20.4	38.2	68.4	54.1	25.9	32.2	19.1	44.6	42	35.8	72.1	11.4	42.1
44	34	22	40.1	70.3	58.1	23.8	29.7	17.2	46.9	48.8	38.2	78.6	11.1	40.9
45	27.4	22.8	47.2	86	53.3	23.2	28.8	18.3	48.6	62.9	38.8	83.9	14.4	43.9
46	26.5	24.9	48.8	79.8	47.4	23	23.4	19.9	50.8	66.5	38.3	84.6	16.4	45.7
47	17.8	28.1	47.1	74	44.3	22.7	19.2	20.2	52.6	68.1	40.1	79.7	17	45.2
48	18	25.3	48	76.3	45.7	22.1	15	20.2	53.2	65.8	39.5	82.6	17.7	45.4
49	14.7	25.2	35.4	54.9	45.9	19.6	15.5	23	45.2	51.6	34.2	68.5	16.9	43
50	10	28	38.4	62.3	45.5	18.9	20.8	19.8	51.3	54.3	35	69.5	16	39.5
51	6.11	29.2	38.8	65.4	47.1	18.4	23.8	21.5	53.9	56.4	37.4	99.4	19.9	37.9
52	5.52	28.9	36.9	63.1	44.9	16.2	27.6	21.5	54	53	38.1	102	20.2	38.9
53	6.31	28.9	33.2	55.7	45.7	14.6	28.4	20	53.4	48.3	39.9	105	19.5	41.5
54	6.57	28.2	30.2	48	46.1	14.4	28.6	21.1	53.4	43.3	37.9	96.1	19	40.8
55	7	27.3	23.2	33.4	44.1	10.7	29.7	20	52.9	37.8	35	82.4	17.3	35.2
56	6.9	28.9	21	30	41.7	12.6	33.1	19.3	52.3	46.4	35.8	74	16.6	36.4
57	0	31.5	24	29.9	0	11.2	38.1	21.3	53.9	48.7	33.7	0	15	37

In [6]:
```
df3 = df2.describe()
cm = sns.light_palette("grey", as_cmap=True)
df3.style.background_gradient(cmap=cm)
```

Out[6]:

	Afghanistan	Albania	Algeria	Angola	Antigua and Barbuda	Argentina	Armenia	Australia	Austria	Azerbaijan	Bahamas	Bahrain	Bangladesh	Barba
count	18	18	18	18	18	18	18	18	18	18	18	18	18	
mean	14.6006	25.3056	36.9333	62.8278	46.0111	18.2389	26.2333	20.2667	50.0111	50.9222	37	77.9667	15.9167	40.7
std	13.1213	4.1745	8.36484	17.9816	12.4843	5.65718	6.03558	1.39579	3.92936	9.71346	2.06483	22.4453	2.90056	3.15
min	0	17.9	21	29.9	0	10.7	15	17.2	43.3	37.8	33.7	0	11.1	3
25%	6.16	22.2	33.75	55.1	45.05	13.05	23.4	19.5	45.7	42.925	35.2	72.55	13.65	38.
50%	8.5	26.3	37.55	66.85	46	18.65	28	20.1	51.8	48.75	37.65	79.45	16.5	40
75%	24.375	28.725	41.6	75.05	53.1	22.925	29.625	21.25	53.35	55.875	38.275	84.425	17.6	42
max	43.6	31.5	48.8	89.7	58.1	28.4	38.1	23	54	68.1	40.1	105	20.2	4

The International Monetary Fund has measured the GDP of all countries in the world.

There are three ways to compare GDP between countries. The one you use depends on your purpose and how exchange rates and population would affect it.

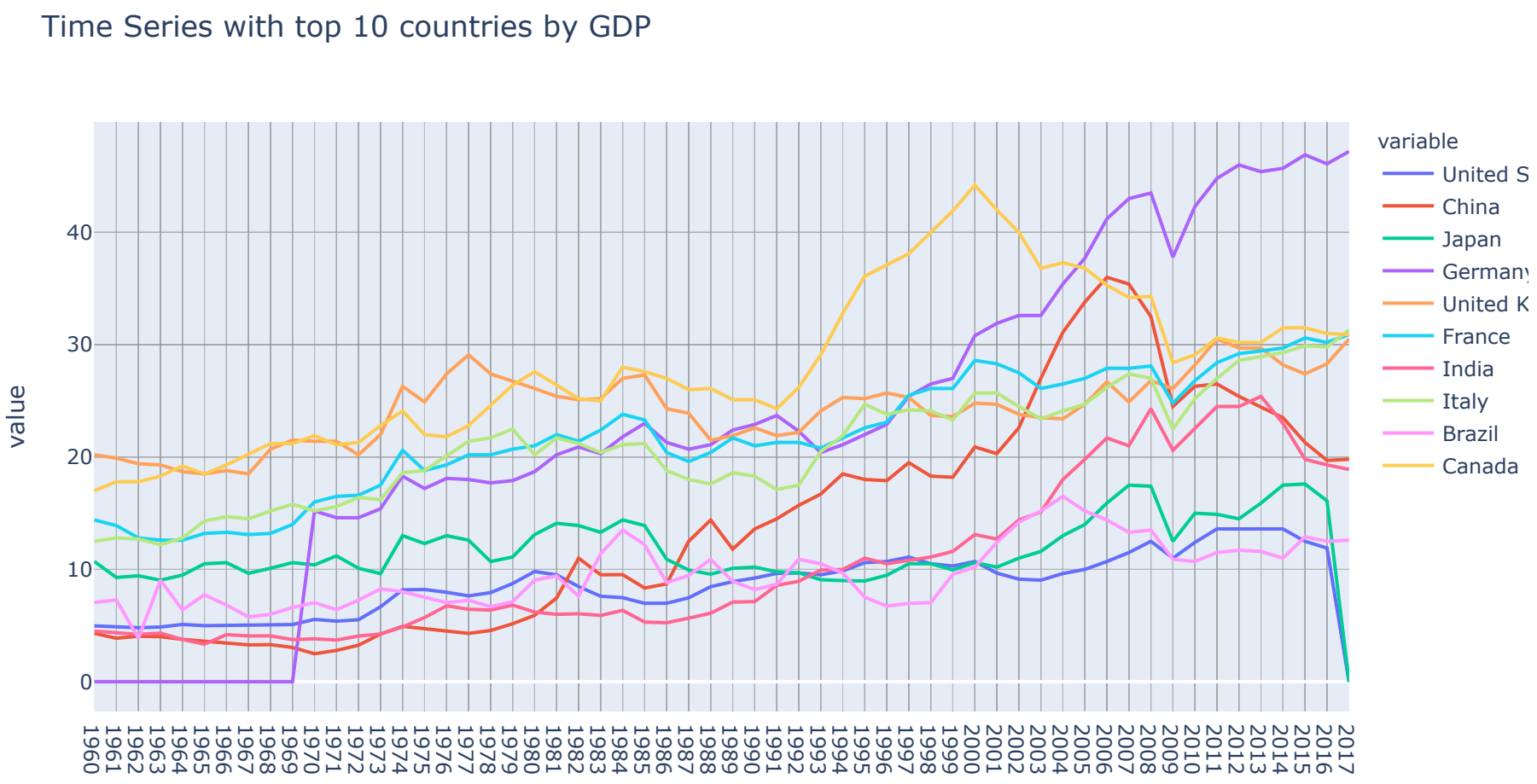
1. Official Exchange Rate

The IMF uses the most commonly agreed-upon measure, the official exchange rate. The country's government or central bank sets this rate.It tells you how much the bank will give you in exchange for one unit of your country's currency.

Time Series with top 10 countries by GDP

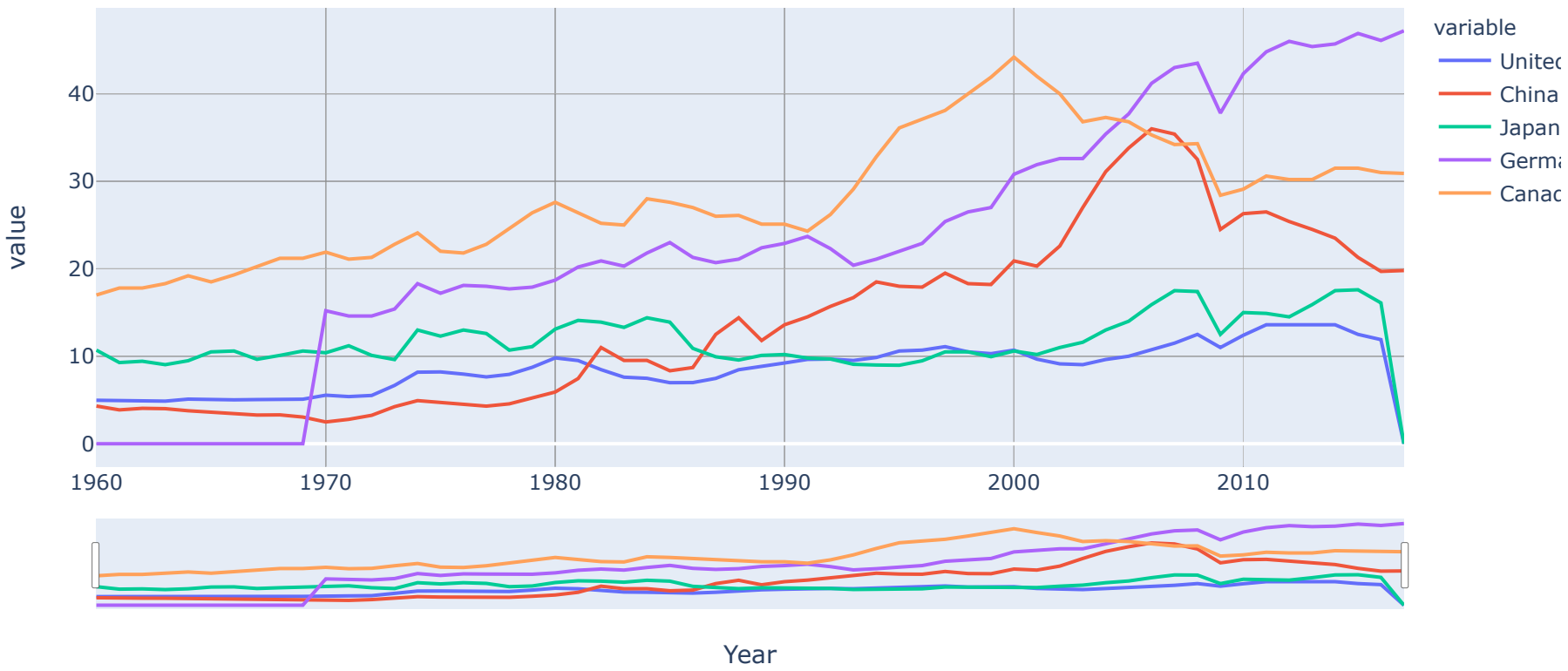
In [7]:
```
top10 = ['United States','China','Japan','Germany','United Kingdom', 'France','India','Italy','Brazil','Canada']
df = data
df1 = data
df2 = data
```

In [8]:
```
fig = px.line(df, x="Year", y=top10,
              hover_data={"Year": "%Y"},
              title='Time Series with top 10 countries by GDP')
fig.update_xaxes(
    dtick="M1",
    tickformat="Y",
    ticklabelmode="period")
fig.show()
```



Comparison within Top 10 countries by GDP

In [9]:
```
fig = px.line(df, x='Year', y=['United States','China','Japan','Germany','Canada'],
              title='Comparison within Top 10 countries by GDP')

fig.update_xaxes(
    rangeslider_visible=True,
    rangeselector=dict(
        buttons=list([
            dict(count=1, label="1m", step="month", stepmode="backward"),
            dict(count=6, label="6m", step="month", stepmode="backward"),
            dict(count=1, label="YTD", step="year", stepmode="todate"),
            dict(count=1, label="1y", step="year", stepmode="backward"),
            dict(step="all")
        ])
    )
)
fig.show()
```



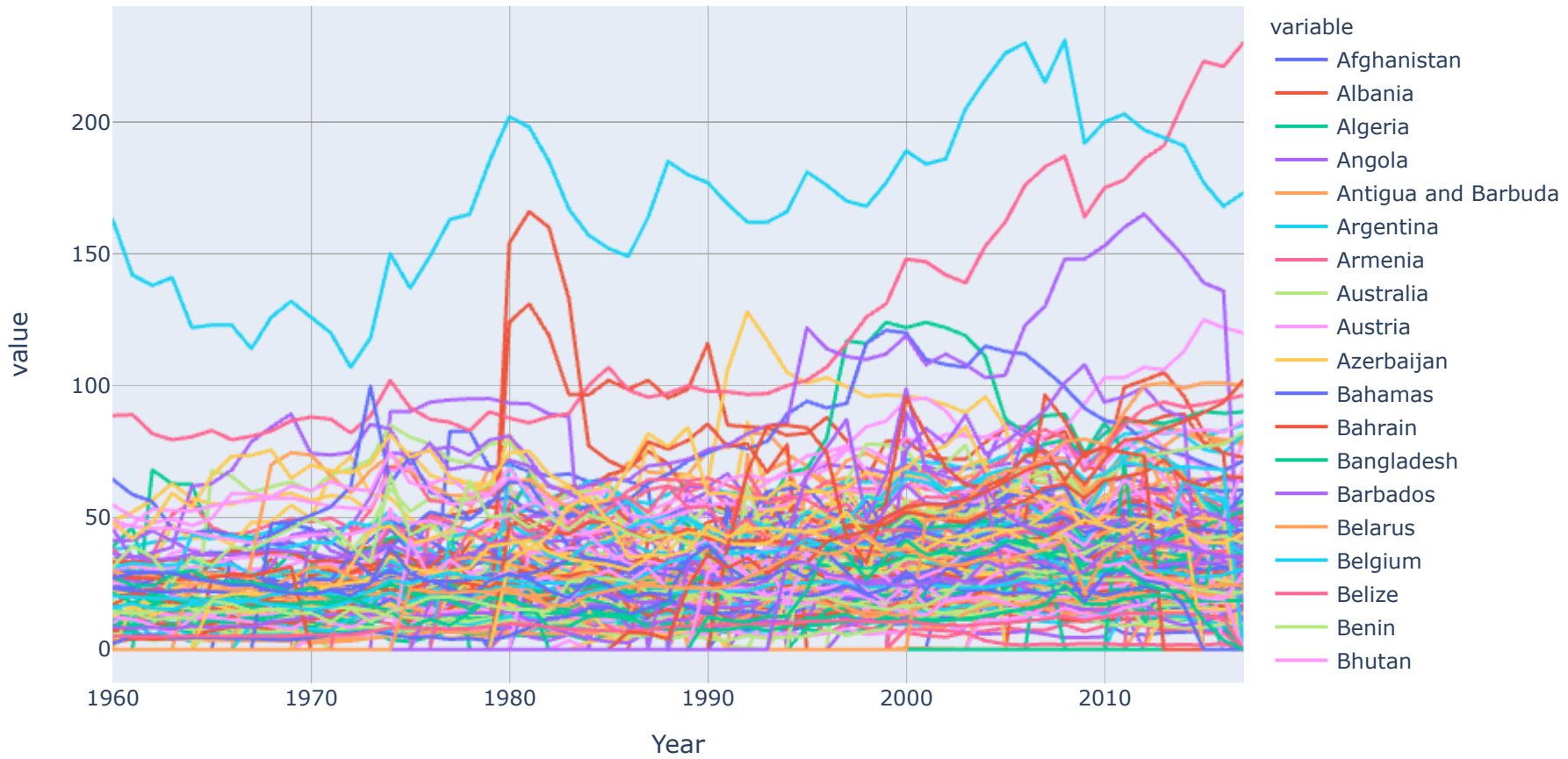
2. Purchasing Power Parity

Purchasing power parity allows you to make more accurate comparisons of the economies of two countries. It compensates for exchange rates changes over time. It also accounts for government manipulation of exchange rates.

World's 10 Largest Economies Using Purchasing Power Parity

In [10]:
```
Largest10 = ['China','United States','India','Japan','Germany','Russia','Indonesia','Brazil','United Kingdom']
Last10Year = ['2007','2008','2009','2010','2011','2012','2013','2014','2015','2016']
```

In [22]:
```
fig = px.line(df, x=df1.index, y=df.columns)
fig.show()
```



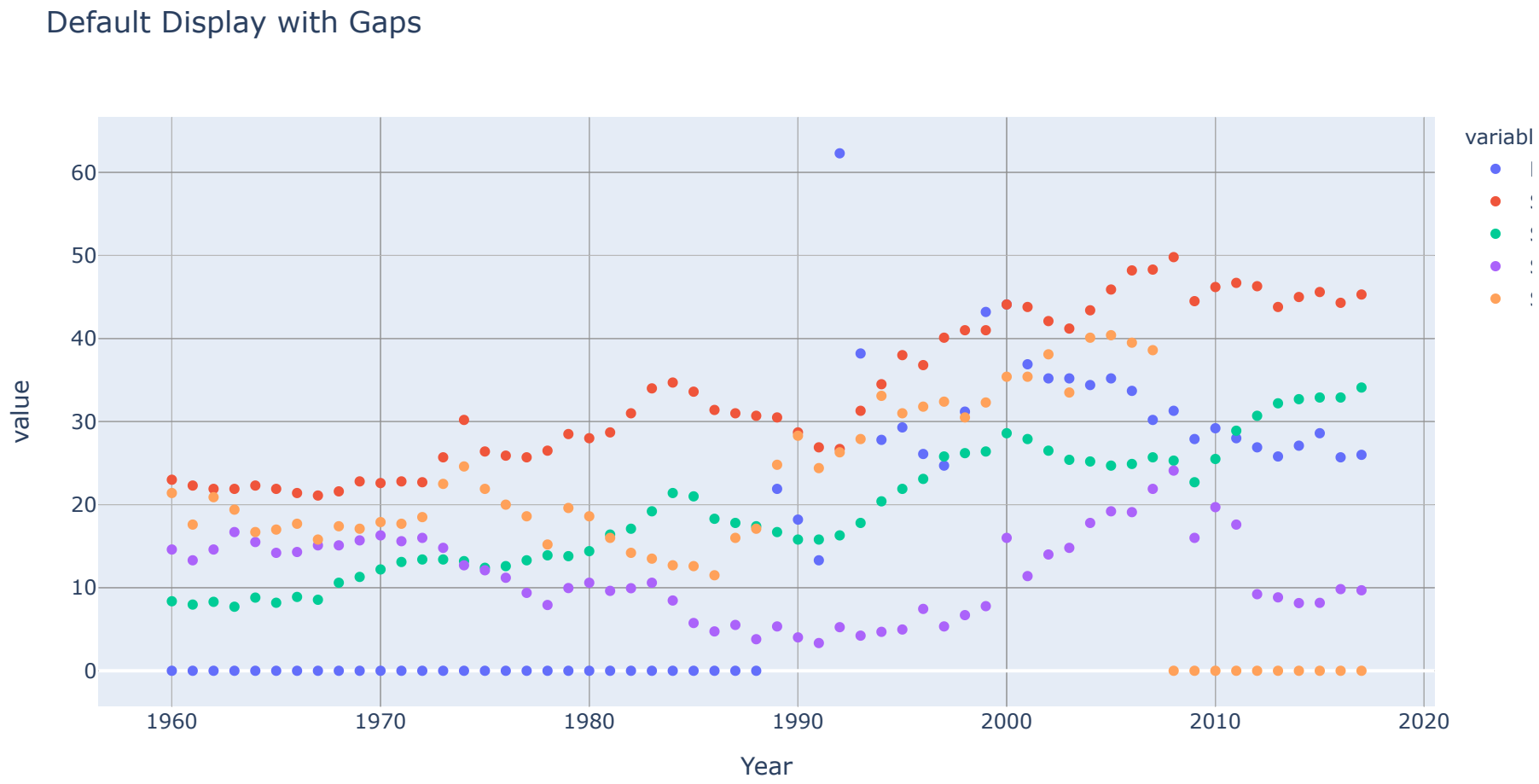
3. GDP per Capita

GDP per capita is a good way to compare the economic output of a country as experienced by its residents. It divides a country's economic output by its population. You can use GDP per capita to compare any country with another one.

In [24]:
```
PerCapita = ['Luxembourg','Switzerland','Macao','Norway','Ireland','Iceland','Qatar','Singapore','United States','Denmar
```

In [27]:
```
fig = px.line(df, x=df.index, y=df.columns, title='Time Series with GDP per Capita')

fig.update_xaxes(rangeslider_visible=True)
fig.show()
```

In [14]:
```
fig = px.scatter(df, x='Year', y=['Russia','Sweden','Spain','Sudan','Syria'], range_x=['2015-12-01', '2016-01-15'],
                 title="Default Display with Gaps")
fig.show()
```

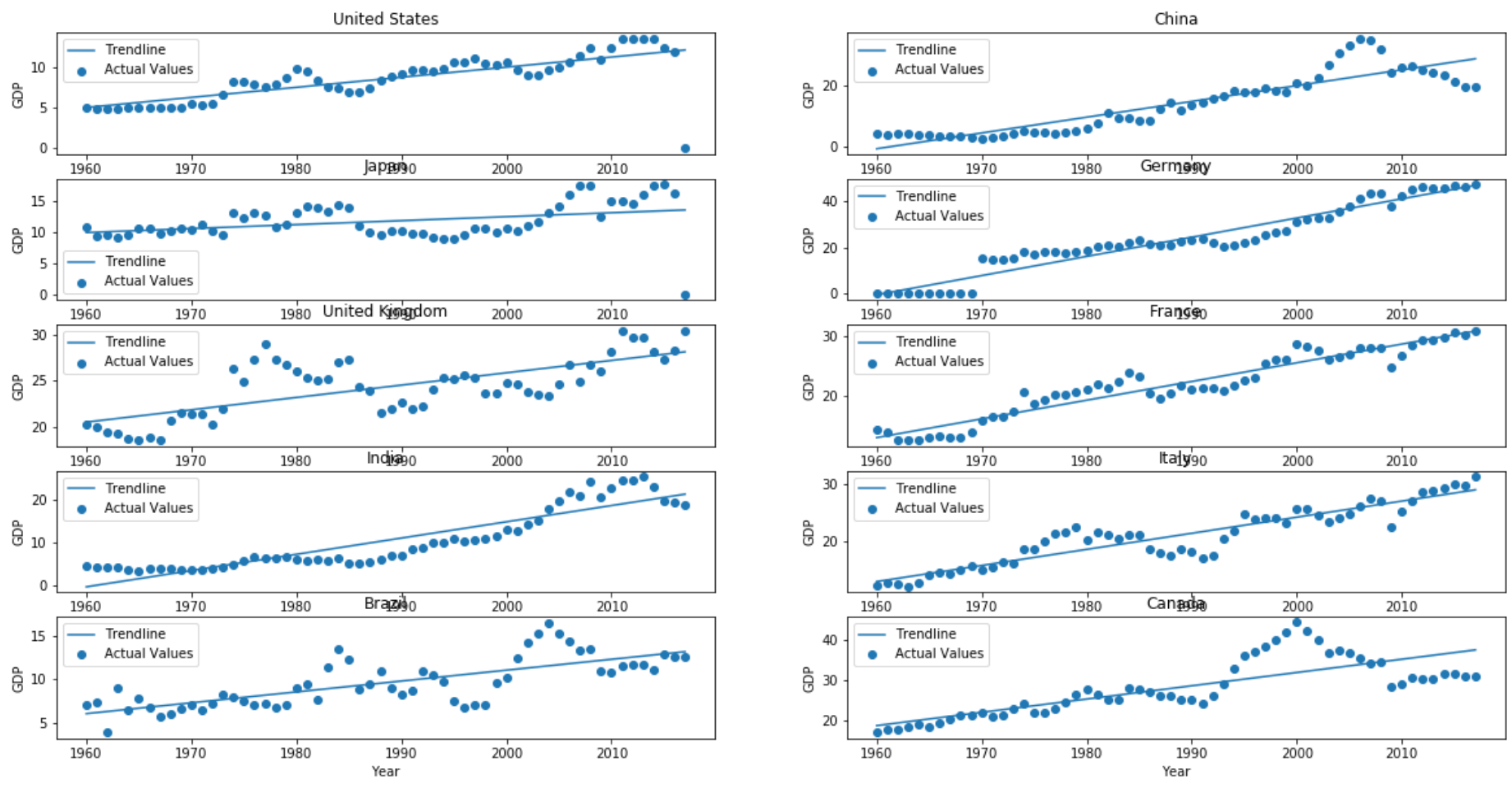


1.1 Time Varying Space Fixed

Each graph represents a country with different values of countries in x-axis.

In [15]:
```
from sklearn.linear_model import LinearRegression
def graph_plot(country):
    x=np.arange(1,len(df['Year'])+1)
    y=df[country]
    ts=pd.DataFrame({'Year':x,'GDP':y})
    X=ts[['Year']]
    Y=ts['GDP']
    lin=LinearRegression()
    lin.fit(X,Y)
    predy=lin.predict(X)
    plt.scatter(df['Year'],Y,label="Actual Values")
    plt.plot(df['Year'],predy,label="Trendline")
    plt.legend()
    plt.xlabel('Year')
    plt.ylabel('GDP')
    plt.title(country)
```

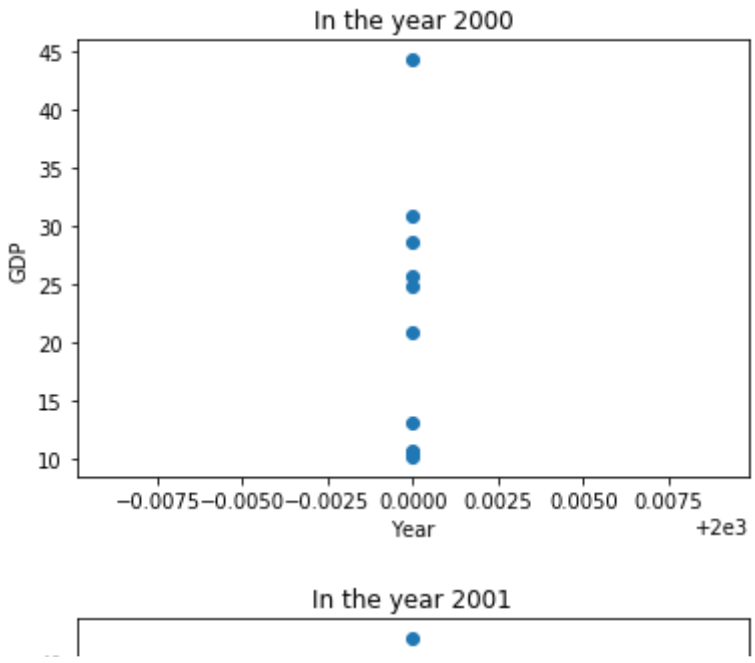
In [16]:
```
plt.figure(figsize=(20,10))
plt.subplot(5, 2, 1)
graph_plot('United States')
plt.subplot(5, 2, 2)
graph_plot('China')
plt.subplot(5, 2, 3)
graph_plot('Japan')
plt.subplot(5, 2, 4)
graph_plot('Germany')
plt.subplot(5, 2, 5)
graph_plot('United Kingdom')
plt.subplot(5, 2, 6)
graph_plot('France')
plt.subplot(5, 2, 7)
graph_plot('India')
plt.subplot(5, 2, 8)
graph_plot('Italy')
plt.subplot(5, 2, 9)
graph_plot('Brazil')
plt.subplot(5, 2, 10)
graph_plot('Canada')
plt.show()
```



In [17]:
```
df1.set_index("Year", inplace = True)
```

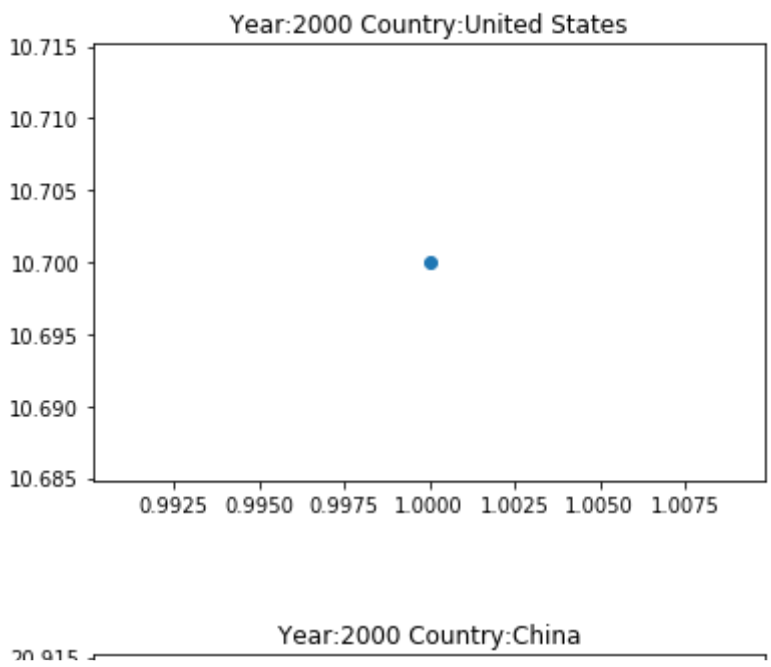
1.2 Time Fixed Space Varying

In [18]:
```
GDP_TVSF=df1.loc[2000:2017,top10]

years=GDP_TVSF.index
years

for year in years:
    y=GDP_TVSF.loc[year]
    x=[year]*10
    plt.scatter(x,y)
    plt.title('In the year '+str(year))
    plt.xlabel('Year')
    plt.ylabel('GDP')
    plt.show()
```



1.3 Space Fixed Time Fixed

In [19]:
```
import matplotlib.pyplot as plt
for i in range(0,len(years)):
    for j in range(0,len(top10)):
        plt.scatter([1],GDP_TVSF.iloc[i][j])
        plt.title('Year:'+str(years[i])+' Country:'+top10[j])
        plt.show()
        print()
```



1.4 Time Varying space Varying from 2000-2017

In [20]:
```
selcountries=df1.loc[2000:2017,top10]

c=[]
for country in top10:
    countrydata=selcountries[country]
    c.append(countrydata)
TVSV=pd.concat(c)

plt.figure(figsize=(10,5))
plt.plot(selcountries)
plt.xticks(range(2000,2018,1))
plt.title('Yearwise GDP(s) of Top 10 countries')
plt.xlabel('Year')
plt.ylabel('GDP')
plt.legend(top10)
plt.show()
```