```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
# Read the Dataset of the Population Growth
df1 = pd.read_csv("/content/API_SP.POP.GROW_DS2_en_csv_v2_5358698.csv")
df1.head()
```

	Country Name	Country Code	Indicator Name	Indicator Code	1960	1961	1962	1963	1
0	Aruba	ABW	Population growth (annual %)	SP.POP.GROW	NaN	2.179059	1.548572	1.389337	1.21
1	Africa Eastern and Southern	AFE	Population growth (annual %)	SP.POP.GROW	NaN	2.660180	2.732633	2.753248	2.80
2	Afghanistan	AFG	Population growth (annual %)	SP.POP.GROW	NaN	1.925952	2.014879	2.078997	2.13
3	Africa Western and Central	AFW	Population growth (annual %)	SP.POP.GROW	NaN	2.115789	2.145723	2.190827	2.21
4	Angola	AGO	Population growth (annual %)	SP.POP.GROW	NaN	1.558355	1.460738	1.410425	1.30

5 rows × 66 columns

```
# Read the Dataset of the Unemployement rate
df2 = pd.read_csv("/content/API_SL.UEM.TOTL.ZS_DS2_en_csv_v2_5358416.csv")
df2.head()
```

	Country Name	Country Code	Indicator Name	Indicator Code	1991	1992	1993	1994	1995	1996	...	201
0	Aruba	ABW	Unemployment, total (% of total labor force) (...	SL.UEM.TOTL.ZS	NaN	NaN	NaN	NaN	NaN	NaN	...	Nal
1	Africa Eastern and Southern	AFE	Unemployment, total (% of total labor force) (...	SL.UEM.TOTL.ZS	7.333336	7.318747	7.242706	7.160694	7.063796	7.055998	...	6.59935
2	Afghanistan	AFG	Unemployment, total (% of total labor force) (...	SL.UEM.TOTL.ZS	8.121000	8.168000	8.123000	8.111000	8.260000	8.165000	...	8.01900
3	Africa Western and Central	AFW	Unemployment, total (% of total labor force) (...	SL.UEM.TOTL.ZS	4.224595	4.335460	4.372125	4.366898	4.348996	4.379537	...	4.16755
4	Angola	AGO	Unemployment, total (% of total labor force) (...	SL.UEM.TOTL.ZS	4.489000	4.487000	4.531000	4.395000	4.304000	4.274000	...	8.06400

5 rows × 35 columns

Double-click (or enter) to edit

```
def convert_to_years(df1):
    id_vars = ['Country Name', 'Country Code', 'Indicator Name', 'Indicator Code']
    value_vars = df1.columns.difference(id_vars).tolist()
    df1_years = pd.melt(df1, id_vars=id_vars, value_vars=value_vars, var_name='Year', value_name='Value')
    df1_years['Year'] = pd.to_datetime(df1_years['Year'], format='%Y')
    return df1_years['Year']

def convert_to_dataframe(df1):
    years = convert_to_years(df1)
    return df1['Country Name'], years



Country_Name, Year = convert_2_datafram(df1)

print(Country_Name)
print(Year)
```

```
0                          Aruba
1      Africa Eastern and Southern
2                      Afghanistan
3       Africa Western and Central
4                          Angola
                  ...
261                       Kosovo
262                  Yemen, Rep.
263                 South Africa
264                       Zambia
265                     Zimbabwe
```

```
Name: Country Name, Length: 266, dtype: object
0       1960-01-01
1       1960-01-01
2       1960-01-01
3       1960-01-01
4       1960-01-01
           ...
16487   2021-01-01
16488   2021-01-01
16489   2021-01-01
16490   2021-01-01
16491   2021-01-01
Name: Year, Length: 16492, dtype: datetime64[ns]
```

To begin exploring the Population and Unemployment dataset, first check its available columns and then generate a summary using the .describe method.

```
df1.columns
```

```
Index(['Country Name', 'Country Code', 'Indicator Name', 'Indicator Code',
       '1960', '1961', '1962', '1963', '1964', '1965', '1966', '1967', '1968',
       '1969', '1970', '1971', '1972', '1973', '1974', '1975', '1976', '1977',
       '1978', '1979', '1980', '1981', '1982', '1983', '1984', '1985', '1986',
       '1987', '1988', '1989', '1990', '1991', '1992', '1993', '1994', '1995',
       '1996', '1997', '1998', '1999', '2000', '2001', '2002', '2003', '2004',
       '2005', '2006', '2007', '2008', '2009', '2010', '2011', '2012', '2013',
       '2014', '2015', '2016', '2017', '2018', '2019', '2020', '2021'],
      dtype='object')
```

```
# Calculate summary statistics
print(df1.describe())
```

```
             1960        1961        1962        1963        1964        1965  \
count   0.0  264.000000  264.000000  264.000000  264.000000  264.000000
mean    NaN    2.194004    2.286008    2.358841    2.303821    2.290847
std     NaN    1.380567    1.382939    1.521203    1.375586    1.419673
min     NaN   -1.015528   -1.510091   -1.845309   -2.110700   -2.354033
25%     NaN    1.370124    1.418289    1.484893    1.479080    1.412867
50%     NaN    2.204979    2.304936    2.394379    2.390442    2.381150
75%     NaN    2.845553    2.848907    2.885317    2.873235    2.847189
max     NaN   10.638254   11.774148   12.851885   12.147917   11.964503

             1966        1967        1968        1969  ...        2012  \
count  264.000000  264.000000  264.000000  264.000000  ...  265.000000
mean     2.253186    2.245973    2.265524    2.235044  ...    1.349639
std      1.426523    1.457580    1.698895    1.813731  ...    1.434792
min     -2.596081   -2.829547   -3.085539   -4.787105  ...   -5.280078
25%      1.295425    1.258571    1.263360    1.172316  ...    0.461311
50%      2.392027    2.339552    2.337629    2.340388  ...    1.255941
75%      2.804877    2.774092    2.808907    2.773119  ...    2.195008
max     11.988676   12.114861   12.612111   17.039974  ...    9.758169

             2013        2014        2015        2016        2017        2018  \
count  265.000000  265.000000  265.000000  265.000000  265.000000  265.000000
mean     1.401124    1.375687    1.323819    1.271352    1.198958    1.179653
std      1.508699    1.588856    1.402870    1.287458    1.250445    1.262902
min     -5.033810   -6.852118   -4.415744   -2.217280   -3.755484   -4.048391
25%      0.481191    0.473649    0.500452    0.468170    0.381500    0.362584
50%      1.236383    1.196115    1.140936    1.133234    1.149954    1.140549
75%      2.220753    2.196927    2.200322    2.162337    2.079584    2.032734
max      9.226496   11.794016    9.219918    7.212802    4.394554    4.556082

             2019        2020        2021
count  265.000000  265.000000  265.000000
mean     1.156251    1.054799    0.905508
std      1.205549    1.185017    1.264801
min     -2.904996   -2.984077   -4.170336
25%      0.371554    0.238041    0.180461
50%      1.074975    1.011272    0.902989
75%      1.978457    1.897864    1.796101
max      3.931356    3.727101    3.707424

[8 rows x 62 columns]
```

```
# Calculate summary statistics
print(df2.describe())
```

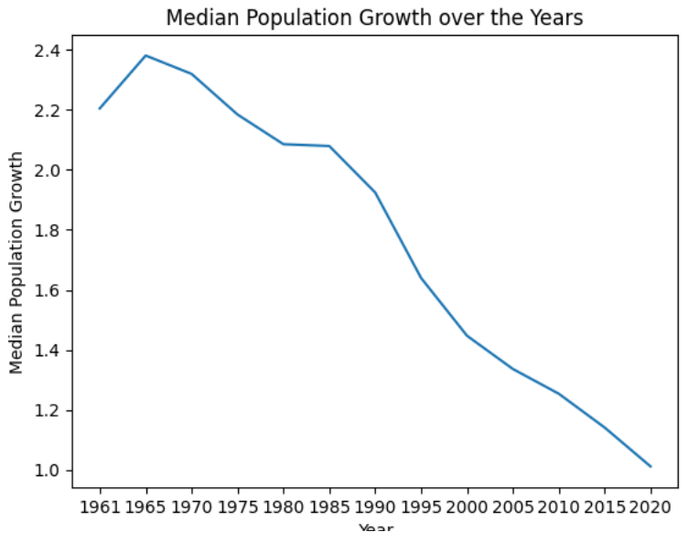
```
             1991        1992        1993        1994        1995        1996  \
count  235.000000  235.000000  235.000000  235.000000  235.000000  235.000000
mean     7.200893    7.538033    7.957919    8.132981    8.204283    8.361057
std      5.559648    5.853788    5.895106    5.817520    5.894885    5.930076
min      0.600000    0.661000    0.637000    0.645000    0.647000    0.640000
25%      3.005000    3.251000    3.719000    3.992487    4.018328    4.098335
50%      5.815639    5.919194    6.123000    6.550999    7.063796    7.141000
75%      9.764504   10.176000   10.842606   11.095000   10.980617   11.329000
max     29.886999   30.014999   29.745001   30.000000   35.599998   38.799999

             1997        1998        1999        2000  ...        2012  \
count  235.000000  235.000000  235.000000  235.000000  ...  235.000000
mean     8.259042    8.308067    8.441300    8.319262  ...    7.991842
std      5.741922    5.692938    5.672374    5.731190  ...    5.608217
min      0.610000    0.613000    0.614000    0.611000  ...    0.480000
25%      4.092500    4.198602    4.353729    4.133000  ...    4.059724
50%      7.090541    7.048398    6.930000    6.597600  ...    6.712282
75%     11.015313   11.259500   11.711000   11.319535  ...   10.248132
max     36.000000   34.500000   32.400002   32.200001  ...   31.200001

             2013        2014        2015        2016        2017        2018  \
count  235.000000  235.000000  235.000000  235.000000  235.000000  235.000000
mean     8.014396    7.856910    7.766160    7.641989    7.397113    7.141719
std      5.680118    5.538581    5.391755    5.284049    5.132107    5.060204
min      0.250000    0.200000    0.170000    0.150000    0.140000    0.110000
25%      4.080076    4.160410    4.305000    4.187633    3.965525    3.822140
```

```
50%      6.512784    6.250916    6.490000    6.010000    5.873657    5.620000
75%     10.140000   10.125765    9.835000    9.655000    9.360000    9.015000
max     29.139999   28.379999   27.690001   26.197001   26.059999   26.260000

             2019        2020        2021
count  235.000000  235.000000  233.000000
mean     6.981379    8.091862    7.789410
std      4.933526    5.168388    5.166205
min      0.100000    0.140000    0.170000
25%      3.765450    4.525500    4.370000
50%      5.540000    6.786562    6.333154
75%      8.736000   10.197892    9.582000
max     26.315001   28.048000   28.770000

[8 rows x 31 columns]
```

```
# Assuming the dataset is stored in a pandas DataFrame called 'df1'
# Extract the year columns for which you want to calculate the median
year_columns = ['1961', '1962', '1963', '1964', '1965', '1966', '1967', '1968',
                '1969', '1970', '1971', '1972', '1973', '1974', '1975', '1976', '1977',
                '1978', '1979', '1980', '1981', '1982', '1983', '1984', '1985', '1986',
                '1987', '1988', '1989', '1990', '1991', '1992', '1993', '1994', '1995',
                '1996', '1997', '1998', '1999', '2000', '2001', '2002', '2003', '2004',
                '2005', '2006', '2007', '2008', '2009', '2010', '2011', '2012', '2013',
                '2014', '2015', '2016', '2017', '2018', '2019', '2020', '2021']

# Extract the data for the year columns
year_data = df1[year_columns]

# Calculate the correlation matrix
correlation_matrix = year_data.corr()

# Print the correlation matrix
print("Correlation Matrix:")
print(correlation_matrix)
```

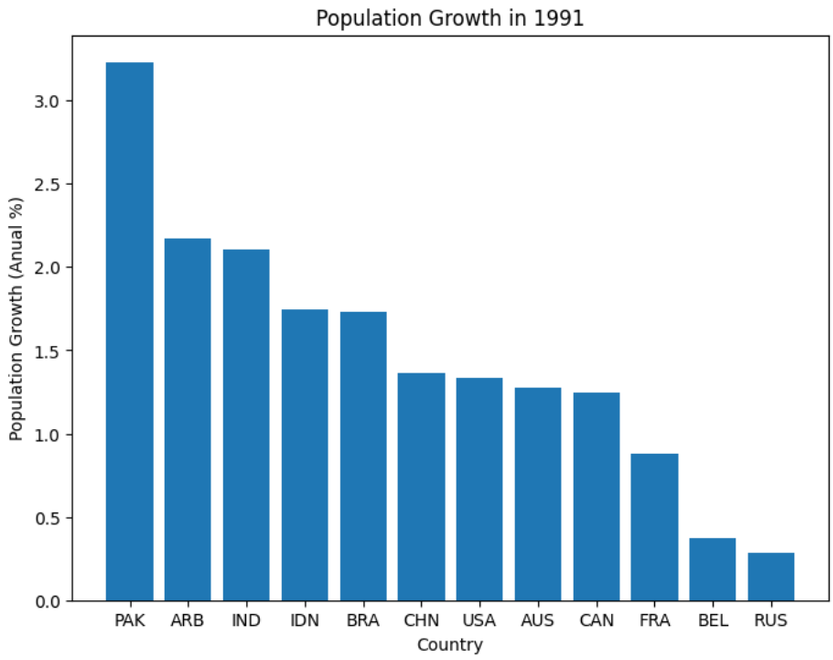
```
Correlation Matrix:
          1961      1962      1963      1964      1965      1966      1967  \
1961  1.000000  0.934272  0.872061  0.892730  0.882911  0.858947  0.835788
1962  0.934272  1.000000  0.883490  0.968772  0.928213  0.897871  0.882583
1963  0.872061  0.883490  1.000000  0.899643  0.941680  0.912546  0.887338
1964  0.892730  0.968772  0.899643  1.000000  0.975365  0.952028  0.915790
1965  0.882911  0.928213  0.941680  0.975365  1.000000  0.980808  0.943414
...        ...       ...       ...       ...       ...       ...       ...
2017  0.233092  0.256391  0.200452  0.268541  0.265896  0.287934  0.299063
2018  0.221472  0.227269  0.169091  0.235218  0.231505  0.248446  0.258337
2019  0.189970  0.192808  0.134704  0.200694  0.197175  0.211096  0.221238
2020  0.050867  0.038945 -0.004031  0.050707  0.055187  0.076820  0.085584
2021  0.062397  0.043013  0.008019  0.057645  0.068165  0.089006  0.098899

          1968      1969      1970  ...      2012      2013      2014  \
1961  0.767305  0.706939  0.711510  ...  0.292863  0.366969  0.314445
1962  0.764752  0.728972  0.757824  ...  0.322723  0.393985  0.338668
1963  0.877126  0.766887  0.754629  ...  0.267715  0.331530  0.280983
1964  0.799642  0.765688  0.793687  ...  0.338392  0.408899  0.356157
1965  0.864009  0.799501  0.809765  ...  0.334399  0.395216  0.355479
...        ...       ...       ...  ...       ...       ...       ...
2017  0.251790  0.249647  0.261056  ...  0.691043  0.665956  0.676306
2018  0.215836  0.214597  0.218093  ...  0.658333  0.608453  0.590084
2019  0.183132  0.187721  0.197535  ...  0.629834  0.548828  0.515956
2020  0.066479  0.087314  0.104729  ...  0.495372  0.431454  0.424845
2021  0.084272  0.102418  0.120986  ...  0.482328  0.441085  0.436744

          2015      2016      2017      2018      2019      2020      2021
1961  0.342960  0.323707  0.233092  0.221472  0.189970  0.050867  0.062397
1962  0.372219  0.352790  0.256391  0.227269  0.192808  0.038945  0.043013
1963  0.309978  0.290349  0.200452  0.169091  0.134704 -0.004031  0.008019
1964  0.390556  0.368049  0.268541  0.235218  0.200694  0.050707  0.057645
1965  0.389663  0.363985  0.265896  0.231505  0.197175  0.055187  0.068165
...        ...       ...       ...       ...       ...       ...       ...
2017  0.839146  0.952564  1.000000  0.945463  0.879442  0.756786  0.735087
2018  0.754799  0.875351  0.945463  1.000000  0.942672  0.783032  0.763159
2019  0.683077  0.814010  0.879442  0.942672  1.000000  0.908203  0.830424
2020  0.545131  0.663927  0.756786  0.783032  0.908203  1.000000  0.929233
2021  0.531823  0.631554  0.735087  0.763159  0.830424  0.929233  1.000000

[61 rows x 61 columns]
```

```
# Assuming the dataset is stored in a pandas DataFrame called 'df1'
# Extract the year columns for which you want to calculate the median
year_columns = ['1961', '1965', '1970', '1975', '1980', '1985','1990', '1995', '2000',
                '2005', '2010', '2015', '2020']

# Calculate the median for each year column
median_values = df1[year_columns].median()

# Create a line graph
plt.plot(year_columns, median_values)

# Add labels and title
plt.xlabel('Year')
plt.ylabel('Median Population Growth')
plt.title('Median Population Growth over the Years')

# Display the graph
plt.show()
```



To compare indicators across different countries over time and explore their interdependence, create a bar chart, line chart, and correlation matrix. To gain deeper insights not only among countries but also among indicators, use a Choropleth Ma and scatter Plot

```
# # Load the dataset
# df = pd.read_csv('/content/API_SP.POP.GROW_DS2_en_csv_v2_5358698.csv')

# create a list of countries to select
countries_to_select = ['USA', 'RUS', 'IND', 'PAK', 'AUS', 'BEL','CHN', 'IDN','CAN', 'FRA','ARB','BRA' ]

# filter the dataset by the selected countries
selected_df1 = df1[df1['Country Code'].isin(countries_to_select)]

# Select the columns of interest
selected_df1 = selected_df1[['Country Code', '1991']]

# Sort the data in descending order
selected_df1 = selected_df1.sort_values(by='1991', ascending=False)

# Create the bar chart
plt.figure(figsize=(8,6))
plt.bar(selected_df1['Country Code'], selected_df1['1991'])
plt.xlabel('Country')
plt.ylabel('Population Growth (Anual %)')
plt.title('Population Growth in 1991')
plt.show()
```

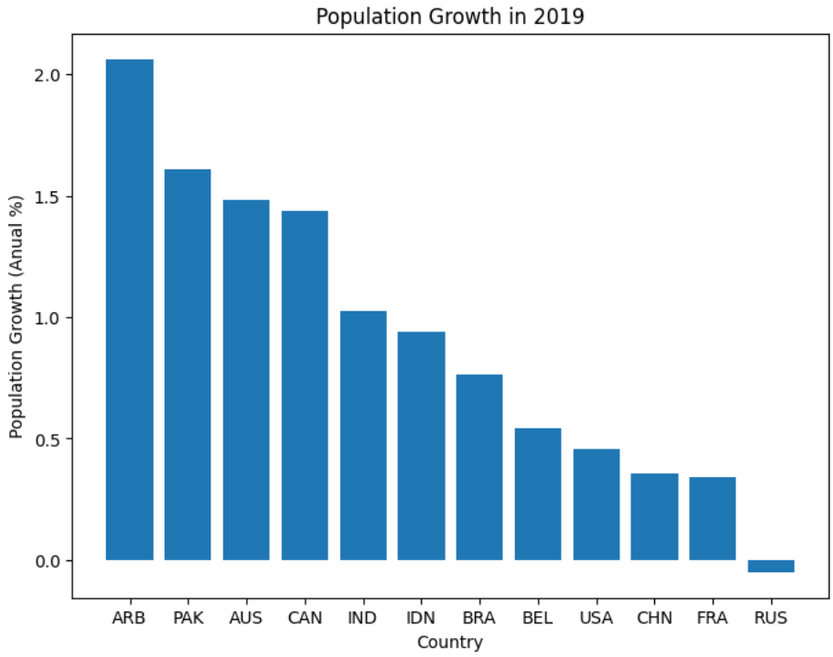



```
df1.columns
```

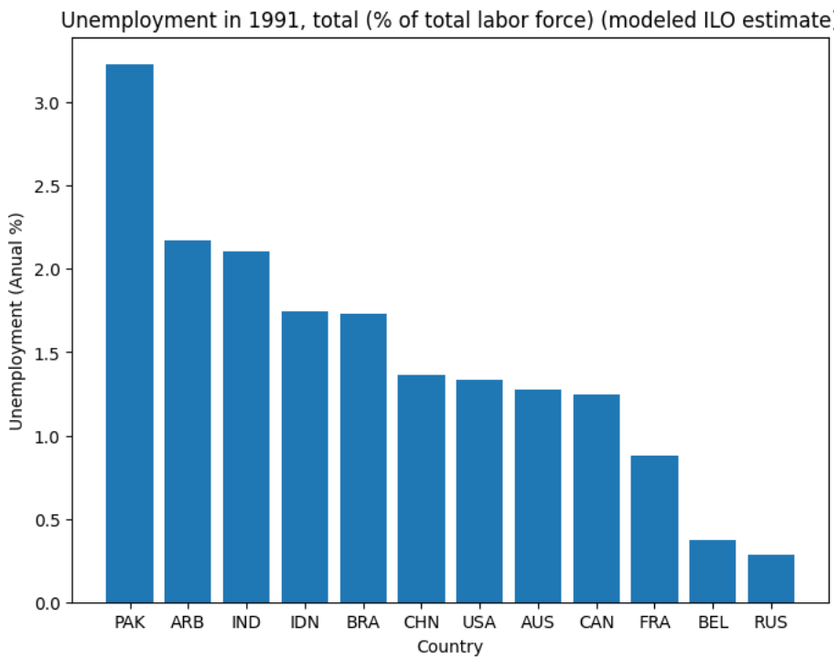
```
Index(['Country Name', 'Country Code', 'Indicator Name', 'Indicator Code',
       '1960', '1961', '1962', '1963', '1964', '1965', '1966', '1967', '1968',
       '1969', '1970', '1971', '1972', '1973', '1974', '1975', '1976', '1977',
       '1978', '1979', '1980', '1981', '1982', '1983', '1984', '1985', '1986',
       '1987', '1988', '1989', '1990', '1991', '1992', '1993', '1994', '1995',
       '1996', '1997', '1998', '1999', '2000', '2001', '2002', '2003', '2004',
       '2005', '2006', '2007', '2008', '2009', '2010', '2011', '2012', '2013',
       '2014', '2015', '2016', '2017', '2018', '2019', '2020', '2021'],
      dtype='object')
```

```
# filter the dataset by the selected countries
selected_df1 = df1[df1['Country Code'].isin(countries_to_select)]
# Select the columns of interest
```

```
selected_df1 = selected_df1[['Country Code', '2019']]

# Sort the data in descending order
selected_df1 = selected_df1.sort_values(by='2019', ascending=False)

# Create the bar chart
plt.figure(figsize=(8,6))
plt.bar(selected_df1['Country Code'], selected_df1['2019'])
plt.xlabel('Country')
plt.ylabel('Population Growth (Anual %)')
plt.title('Population Growth in 2019')
plt.show()
```

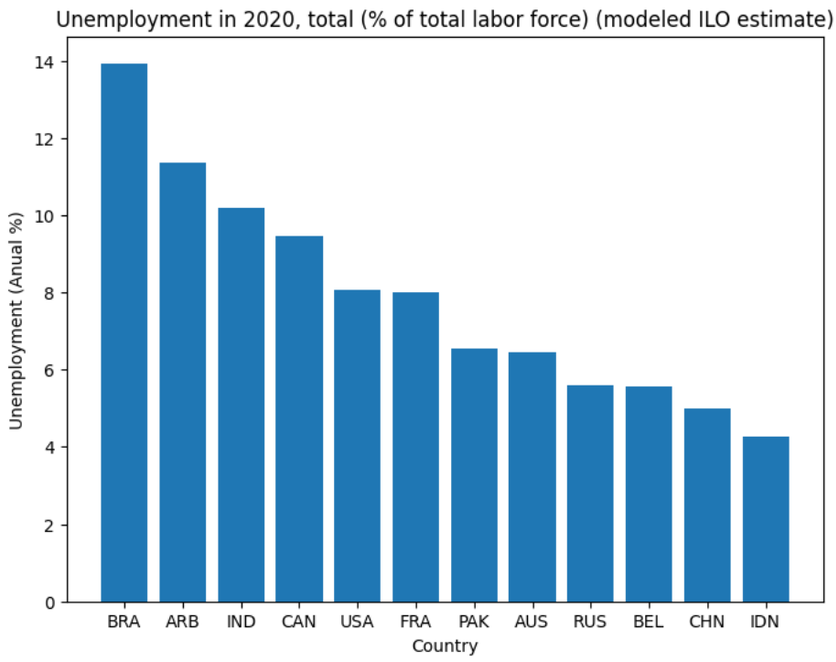



```
# create a list of countries to select
countries_to_select2 = ['USA', 'RUS', 'IND', 'PAK', 'AUS', 'BEL','CHN', 'IDN','CAN', 'FRA','ARB','BRA' ]

# filter the dataset by the selected countries
selected_df2 = df2[2['Country Code'].isin(countries_to_select2)]

# Select the columns of interest
selected_df2 = selected_df2[['Country Code', '1991']]

# Sort the data in descending order
selected_df2 = selected_df2.sort_values(by='1991', ascending=False)

# Create the bar chart
plt.figure(figsize=(8,6))
plt.bar(selected_df2['Country Code'], selected_df2['1991'])
plt.xlabel('Country')
plt.ylabel('Unemployment (Anual %)')
plt.title('Unemployment in 1991, total (% of total labor force) (modeled ILO estimate)')
plt.show()
```



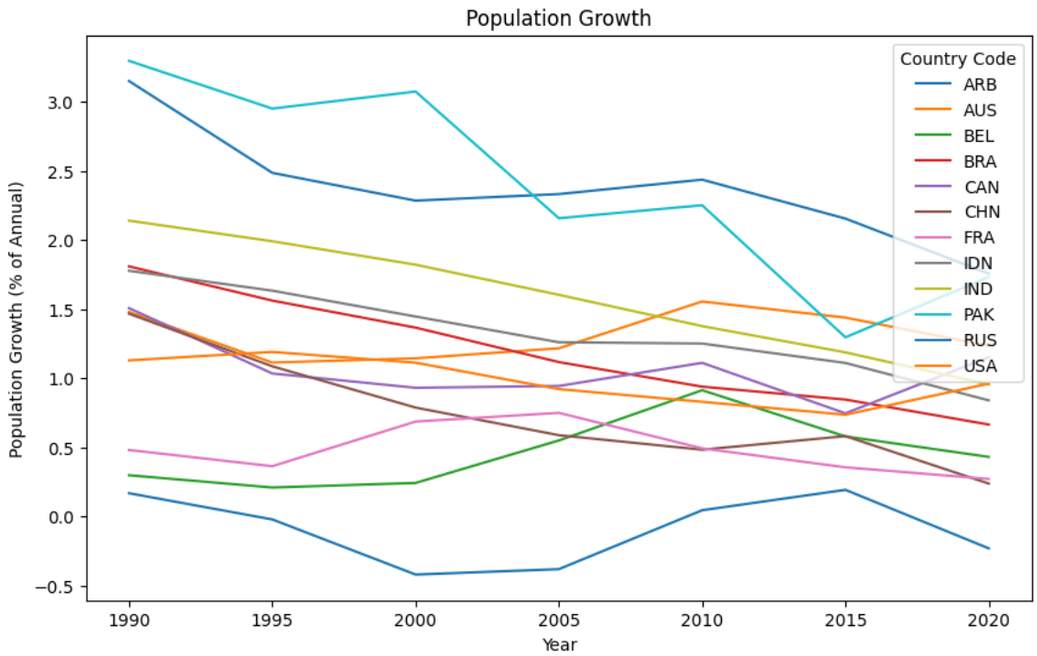

```
# filter the dataset by the selected countries
selected_df2 = df2[df2['Country Code'].isin(countries_to_select)]
```

```
# Select the columns of interest
selected_df2 = selected_df2[['Country Code', '2020']]

# Sort the data in descending order
selected_df2 = selected_df2.sort_values(by='2020', ascending=False)

# Create the bar chart
plt.figure(figsize=(8,6))
plt.bar(selected_df2['Country Code'], selected_df2['2020'])
plt.xlabel('Country')
plt.ylabel('Unemployment (Anual %)')
plt.title('Unemployment in 2020, total (% of total labor force) (modeled ILO estimate)')
plt.show()
```

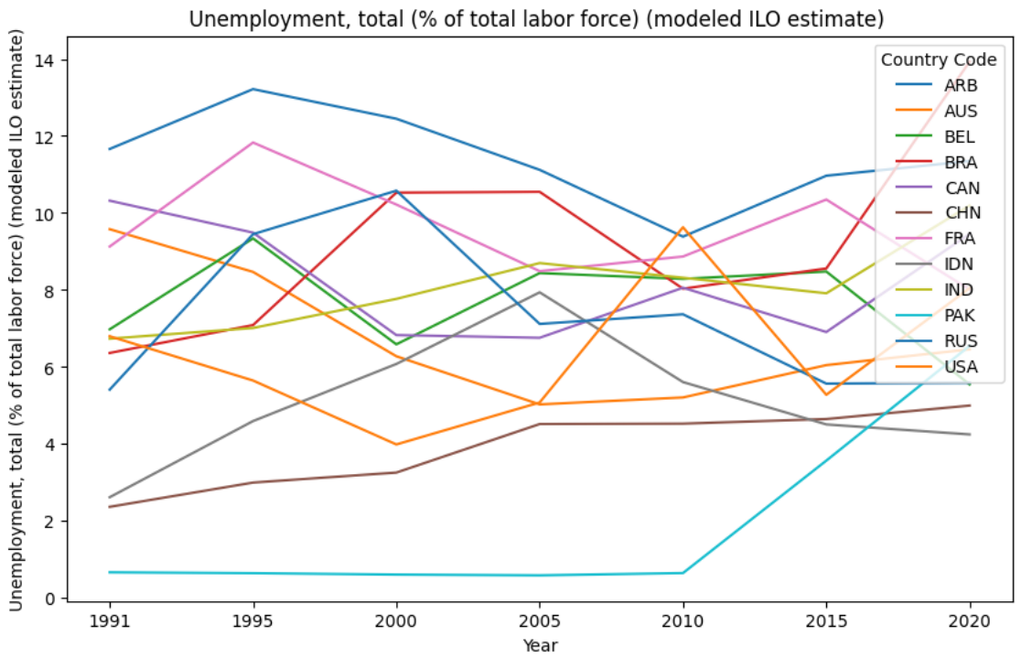


Double-click (or enter) to edit

```
# Select countries of interest
countries = ['USA', 'RUS', 'IND', 'PAK', 'AUS', 'BEL','CHN', 'IDN','CAN', 'FRA','ARB','BRA' ]

# Subset the data for these countries
subset = df1[df1["Country Code"].isin(countries)]

# Set the index to be the country names
subset.set_index("Country Code", inplace=True)

# Select columns of interest
cols = [ '1990', '1995', '2000','2005', '2010', '2015', '2020']
subset = subset[cols]

# Plot the data
subset.T.plot(kind='line', figsize=(10,6))
plt.title('Population Growth')
plt.xlabel('Year')
plt.ylabel('Population Growth (% of Annual)')
plt.show()
```



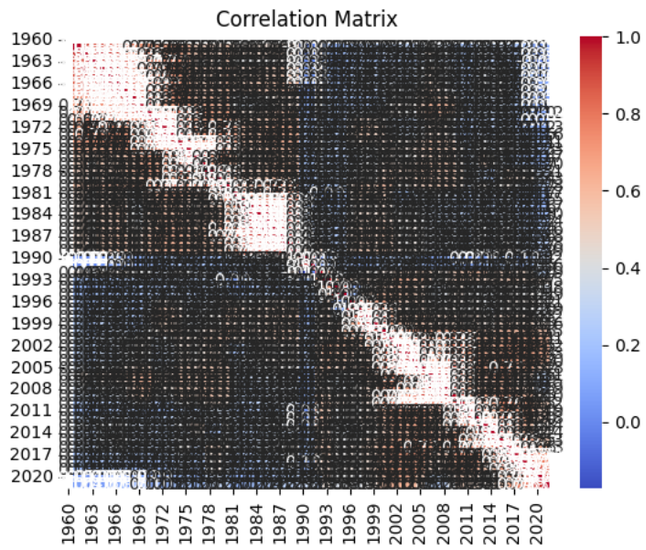
```
# Subset the data for these countries
subset1 = df2[df2["Country Code"].isin(countries)]

# Set the index to be the country names
subset1.set_index("Country Code", inplace=True)

# Select columns of interest
cols = [ '1991', '1995', '2000','2005', '2010', '2015', '2020']
subset1 = subset1[cols]

# Plot the data
subset1.T.plot(kind='line', figsize=(10,6))
plt.title('Unemployment, total (% of total labor force) (modeled ILO estimate) ')
plt.xlabel('Year')
plt.ylabel('Unemployment, total (% of total labor force) (modeled ILO estimate)')
plt.show()
```



Explore and understand any correlations (or lack of) between indicators. Does this vary between country, have any correlations or trends changed with time?

```
# Calculate correlation matrix
corr_matrix = df1.corr()

# Plot correlation matrix using heatmap
sns.heatmap(corr_matrix, cmap='coolwarm', annot=True, fmt='.2f')
plt.title('Correlation Matrix')
plt.show()
```

```
<ipython-input-49-5e1329f60ec9>:9: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a f
  corr_matrix = df1.corr()
```



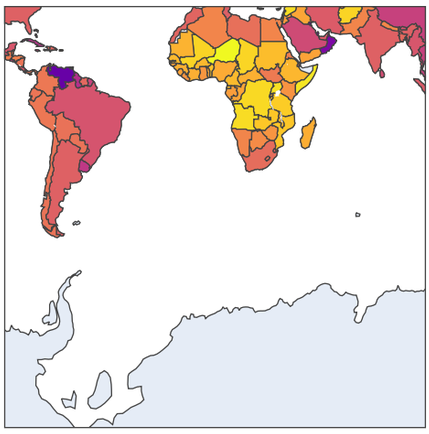

```
# Create Choropleth Map
fig = px.choropleth(df1, locations='Country Code', color='1980',
                    hover_name='Country Name',
                    projection='mercator',
                    title='World Population Growth in 1980')
```

```
fig.show()
```

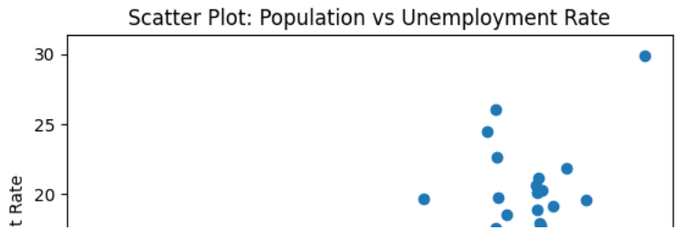
World Population Growth in 1980

```
# Create Choropleth Map
fig = px.choropleth(df1, locations='Country Code', color='2020',
                    hover_name='Country Name',
                    projection='mercator',
                    title='World Population Growth in 2020')
fig.show()
```

World Population Growth in 2020



```
# Extract the columns for population and unemployment rate
population_data = df1['1991']
unemployment_rate_data = df2['1991']

# Plot the scatter plot
plt.scatter(population_data, unemployment_rate_data)

# Add labels and title
plt.xlabel('Population')
plt.ylabel('Unemployment Rate')
plt.title('Scatter Plot: Population vs Unemployment Rate ')

# Show the plot
plt.show()
```




```
# Extract the columns for population and unemployment rate
population_data = df1['2020']
unemployment_rate_data = df2['2020']

# Plot the scatter plot
plt.scatter(population_data, unemployment_rate_data)

# Add labels and title
plt.xlabel('Population')
plt.ylabel('Unemployment Rate')
plt.title('Scatter Plot: Population vs Unemployment Rate')

# Show the plot
plt.show()
```