In [ ]:
```
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
```

In [ ]:
```
# Read and merge datasets
population_data = pd.read_csv('Cleaned Datasets/cleaned_population_data.csv')
education_2017 = pd.read_csv('Cleaned Datasets/CleanedEducationData1870_2017.cs
world_gdp = pd.read_csv('Cleaned Datasets/World_GDP_cleaned.csv')
education_wgdp = pd.merge(education_2017, world_gdp, on=['Code', 'Year'],how='i
education_wgdp = education_wgdp.rename(columns={"Country_x": "Country"})
data_fertility_rate = pd.read_csv('Cleaned Datasets/cleaned_fertility_data.csv
```

In [ ]:
```
data_with_population = pd.merge(education_wgdp, population_data, on=['Country'
df = pd.merge(data_with_population, data_fertility_rate, on=['Country', 'Year']
del df['Country_y']
del df['Unnamed: 0_x']
del df['Unnamed: 0_y']
del df['Region']
del df['_merge']
del df['Area in Square Kilometers']
df
```

Out[ ]:

	Country	Code	Year	avg_years_of_schooling	GDP	GENC	Population	Po
0	AFGHANISTAN	AFG	1965	0.29	1.006667e+09	AF	10997885	
1	AFGHANISTAN	AFG	1970	0.35	1.748887e+09	AF	12430623	
2	AFGHANISTAN	AFG	1975	0.62	2.366667e+09	AF	14132019	
3	AFGHANISTAN	AFG	1990	1.49	NaN	AF	13568282	6
4	AFGHANISTAN	AFG	1991	1.60	NaN	AF	13671918	6
...	...	...	...	...	...	...	...	
3719	ZIMBABWE	ZWE	2011	7.30	1.410192e+10	ZW	13268320	6
3720	ZIMBABWE	ZWE	2012	7.90	1.711485e+10	ZW	13322711	6
3721	ZIMBABWE	ZWE	2013	8.00	1.909102e+10	ZW	13504275	6
3722	ZIMBABWE	ZWE	2014	8.20	1.949552e+10	ZW	13791770	6
3723	ZIMBABWE	ZWE	2017	8.20	1.758489e+10	ZW	14735230	7

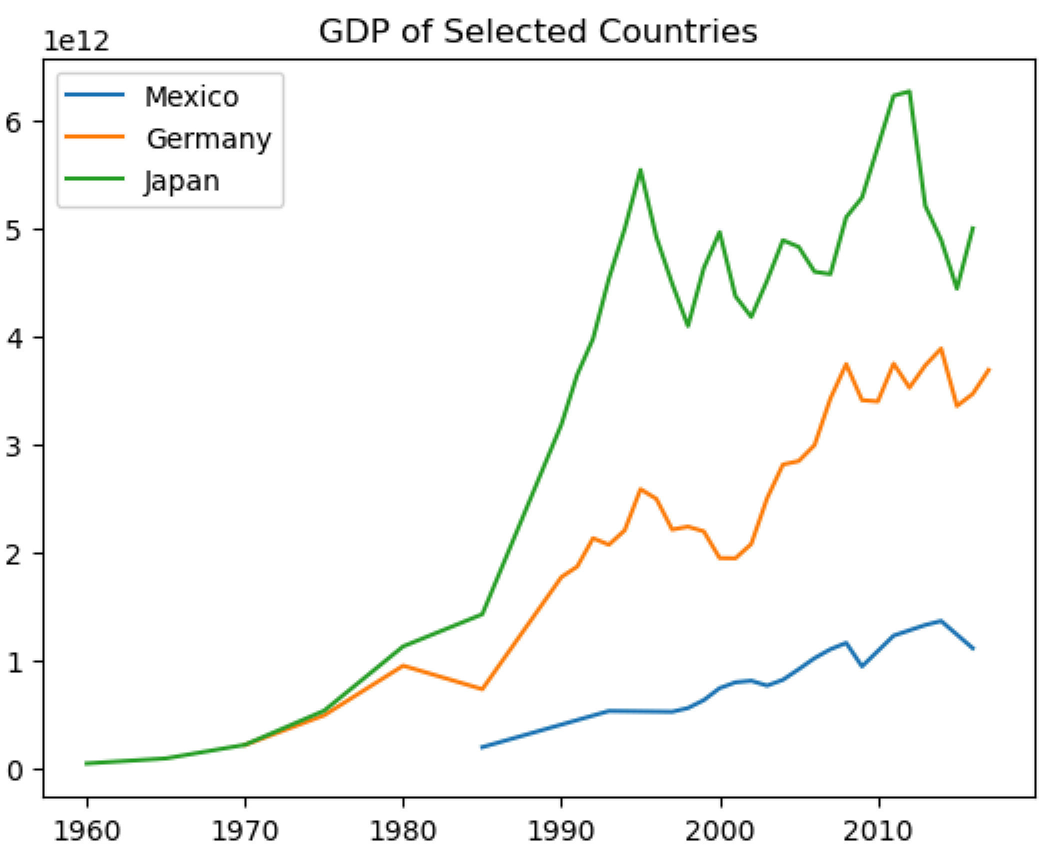
3724 rows × 56 columns

In [ ]:
```
df.to_csv("Cleaned Datasets/Merged_data.csv",index = False)
```

Our dataset is a combination of four datasets which contained data on GDP, education, population, and fertility rate. Each dataset was cleaned and pivoted individually in other notebooks and merged in the cells above. The datasets were merged on country code and country name. We used inner merges exclusively to help deal with missing values. Each row corresponds to a country and a year. Many columns of data were stored as strings, and we converted them to numeric values when appropriate, as well as other basic data cleaning steps. Using inner merges to deal with missing values results in a much smaller dataset, perhaps a third of the size as what we might have had with outer merges. Some loss of data is inevitable, but in the future we may check why exactly certain years and countries are being dropped and whether that data might be found through other sources. However, even after aggressively dropping missing data we still have enough countries with a full dataset to form a training and a testing group for our models.

We have plotted below the GDP available for a few countries

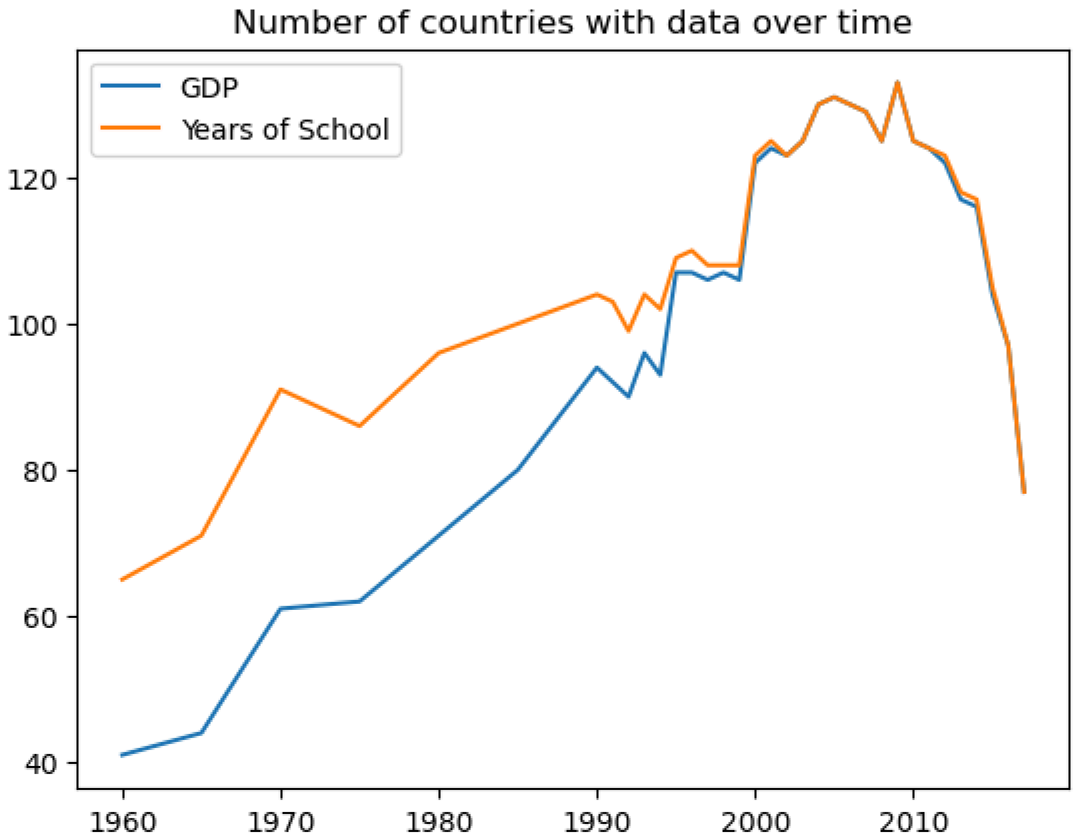
In [ ]:
```
mexicoData = df[df['Country'] == "MEXICO"]
germanyData = df[df['Country'] == "GERMANY"]
japanData = df[df['Country'] == "JAPAN"]
plt.plot(mexicoData['Year'], mexicoData['GDP'], label="Mexico")
plt.plot(germanyData['Year'], germanyData['GDP'], label="Germany")
plt.plot(japanData['Year'], japanData['GDP'], label="Japan")
plt.title("GDP of Selected Countries")
plt.legend()
plt.show()
```



A critical aspect of our project data is handling null values, because for many countries, especially smaller and developing countries, data is not available. The following is a chart of non-null values over time:

In [ ]:
```
plt.plot(np.unique(df["Year"].values),df.groupby("Year")[["GDP","avg_years_of_s
plt.legend()
```

```
plt.title("Number of countries with data over time")
plt.show()
```



This graph shows the amount of data we have per year on GDP and Years of school. The most amount of data we have is in the 2000-2017 range. Since the focus of our project will be primarily on developed economies with lots of data available, we will mostly ignore data points with null values.

Our theoretical model will account for fixed effects by country and by year, which will allow us to remove variation that occurs across the globe in specific years (recessions, etc) as well as account for the specific conditions in a country that persist over the entire timeframe - such as culture, average health, etc. We would like to include country by year fixed effects, but that is obviously impossible. To combat that we will include a number of other control variables such as average years of education, infant mortality rate, and others in the hope that these covariates will help to explain variation in gdp not due to changes in demographics.

The main question we are interested in answering is how the demographic composition of a country impacts its growth rate. A first approximation is to just use the fertility rate in past decades to explain the gdp growth today. We may also explore more complicated models that use the entire age breakdown of a country into certain brackets to predict GDP growth. As a justification for how fertility rate impacts GDP growth, it seems logical to suppose that the fertility rate this year is negatively correlated with growth, since it removes parents from the labor force. However, fertility rate from twenty to sixty years ago should be correlated with more people of workign age, and presumably higher growth. Finally, births from over about 60 years ago likely contribute negatively to growth as those people work in lower numbers and while still consuming the country's resources. For an example of all three stages, consider Japan since World War 2.

Bellow is an example of data cleaning for population and fertility data.

In [ ]:
```
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
```

In [ ]:
```
#Read in the dataframe
df = pd.read_csv('Population_Data.csv')

#Make all Country Names uppercase for merging
df['Name'] = df["Name"].str.upper()
df = df.rename(columns={"Name": "Country"})

#Clean years, dropping missing, make the rest ints
df = df.dropna()
df["Year"] = df["Year"].astype(int)

#Deal with missing values, eliminate commas in large numbers
column_list = df.columns.tolist()
for col in column_list[4:]:
    df[col] = df[col].str.replace('--', '')
    df[col] = df[col].str.replace(',', '')

#Cast everything to floats
for col in column_list[4:]:
    df[col] = pd.to_numeric(df[col], errors='coerce')
```

In [ ]:
```
#Do some sampling to make sure everything is as it should be
df.sample(10)
```

Out[ ]:

	Country	Region	GENC	Year	Area in Square Kilometers	Population	Male Population	Fema Populatio
14283	NICARAGUA	2012,Nicaragua	NI	2012	119990	5818040	2856404.0	2961636
7747	WALLIS AND FUTUNA	1983,Wallis and Futuna	WF	1983	142	12398	NaN	Na
7093	BOLIVIA	1981,Bolivia	BO	1981	1083301	5545224	NaN	Na
9994	SOUTH SUDAN	1993,South Sudan	SS	1993	644329	4807785	NaN	Na
14791	TAJIKISTAN	2014,Tajikistan	TJ	2014	141510	8304941	4164932.0	4140009
6159	ALGERIA	1977,Algeria	DZ	1977	2381740	17152804	NaN	Na
15137	HONDURAS	2016,Honduras	HN	2016	111890	8804368	4290263.0	4514105
12032	SAN MARINO	2002,San Marino	SM	2002	61	28191	13661.0	14530
2591	GUATEMALA	1961,Guatemala	GT	1961	107159	4231703	NaN	Na
16055	IRAN	2020,Iran	IR	2020	1531595	84982541	43087120.0	41895421

10 rows × 54 columns

In [ ]:
```
#Count the missing values
missing_values_per_column = df.isna().sum()

print(missing_values_per_column)
print(len(df))
print("So approximately half the data is missing")
print("After 1990 approximately nothing is missing")
```

```
Country                                        0
Region                                         0
GENC                                           0
Year                                           0
Area in Square Kilometers                      0
Population                                     0
Male Population                             8066
Female Population                           8066
Annual Growth Rate %                        8098
Rate of Natural Increase                    8098
Population Density (People per Sq. Km.)        0
Dependency ratio                            8096
Youth dependency ratio                      8096
Old age dependency ratio                    8096
Median age, both sexes                      8097
Median age, females                         8097
Median age, males                           8097
Natural Increase                            8098
Sex ratio of the population                 8096
Total Fertility Rate                        8109
Age-specific Fertility Rate 15-19           8109
Age-specific Fertility Rate 20-24           8109
Age-specific Fertility Rate 25-29           8109
Age-specific Fertility Rate 30-34           8109
Age-specific Fertility Rate 35-39           8109
Age-specific Fertility Rate 40-44           8109
Age-specific Fertility Rate 45-49           8109
Crude Birth Rate                            8098
Gross Reproduction Rate                     8109
Births, both sexes                          8098
Sex Ratio at Birth                          8109
Births to mothers aged 15-19                8110
Births to mothers aged 20-24                8110
Births to mothers aged 25-29                8110
Births to mothers aged 30-34                8110
Births to mothers aged 35-39                8110
Births to mothers aged 40-44                8110
Births to mothers aged 45-49                8110
Life Expectancy at Birth, Both Sexes        8109
Life Expectancy at Birth, Males             8109
Life Expectancy at Birth, Females           8109
Infant Mortality Rate, Both Sexes           8109
Infant Mortality Rate, Males                8109
Infant Mortality Rate, Females              8109
Age 1-4 Mortality, Both Sexes               8109
Age 1-4 Mortality, Males                    8109
Age 1-4 Mortality, Females                  8109
Under Age 5 Mortality, Both Sexes           8109
Under Age 5 Mortality, Males                8109
Under Age 5 Mortality, Females              8109
Crude Death Rate                            8098
Deaths, both sexes                          8098
Net Migration Rate                          8098
Net international migrants, both sexes      8098
dtype: int64
16724
So approximately half the data is missing
After 1990 approximately nothing is missing
```

In [ ]:
```
#Save the Dataframe
df.to_csv('cleaned_population_data.csv')
```

In [ ]:
```
#Alternate Dataset
df = pd.read_csv('undesa_pd_2019_world_fertility_dataset.csv')

#Only want to keep measures using TFR - Total Fertility Rate
df = df[df["Indicator"] == "TFR"]

#Change Country Name to uppercase
df['Country or Area'] = df["Country or Area"].str.upper()
df = df.rename(columns={"Country or Area": "Country"})

#Keep the columns we want
df = df[['Country', 'Date', 'Value']]

#Rename Value to Fertility Rate
df = df.rename(columns={"Value": "Fertility Rate"})

#Send Years to an int
df['Date'] = np.floor(df['Date']).astype(int)
df = df.rename(columns={"Date": "Year"})

#Figure out how many are missing
missing_values_per_column = df.isna().sum()
print("No missing values!, of course, not every country has every year")

#Get rid of duplicates
df = df.drop_duplicates(subset=['Year', 'Country'])


#Save the Dataframe
df.to_csv('cleaned_fertility_data.csv')

#Look at a sample
df.sample(5)
```

```
No missing values!, of course, not every country has every year
/var/folders/7p/c4gmwp116656n0kd4wjd0x0c0000gn/T/ipykernel_50722/1231151261.p
y:2: DtypeWarning: Columns (14) have mixed types. Specify dtype option on impo
rt or set low_memory=False.
  df = pd.read_csv('undesa_pd_2019_world_fertility_dataset.csv')
```

Out[ ]:

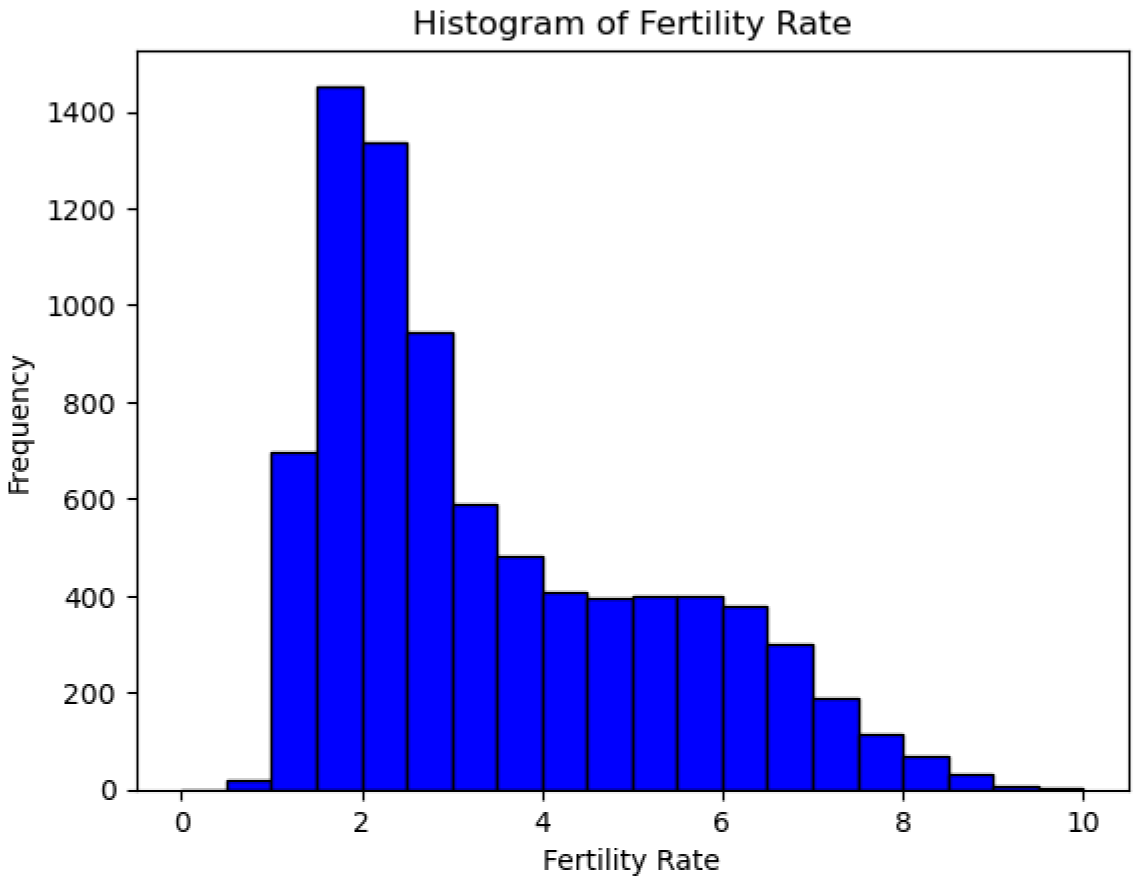
	Country	Year	Fertility Rate
22281	ECUADOR	2002	1.74
26411	FRENCH GUIANA	2009	3.50
11308	BURUNDI	2016	5.50
17355	COSTA RICA	1984	3.42
18683	CUBA	2004	1.54

In [ ]:
```
#Make a histogram of fertility rates to make sure it looks reasonable
column_data = df['Fertility Rate']
plt.hist(column_data, bins=np.linspace(0, 10, 21), color='blue', edgecolor='bla

# Add labels and title
```

```
plt.xlabel('Fertility Rate')
plt.ylabel('Frequency')
plt.title('Histogram of Fertility Rate')

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

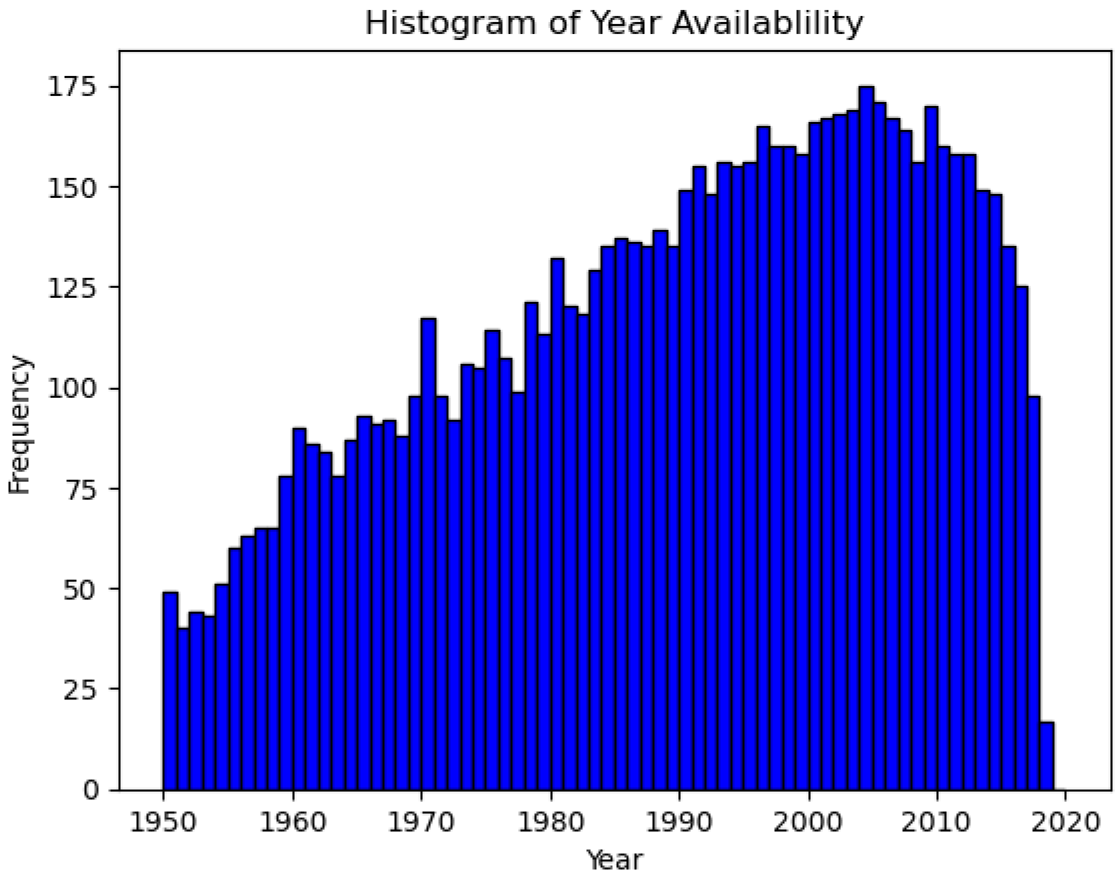


In [ ]:
```
#Now check which years tend to be available
#Make a histogram of fertility rates to make sure it looks reasonable
column_data = df['Year']
plt.hist(column_data, bins=np.linspace(1950, 2020, 71), color='blue', edgecolo

# Add labels and title
plt.xlabel('Year')
plt.ylabel('Frequency')
plt.title('Histogram of Year Availablility')

# Display the histogram
plt.show()

print(len(df))
```



8216

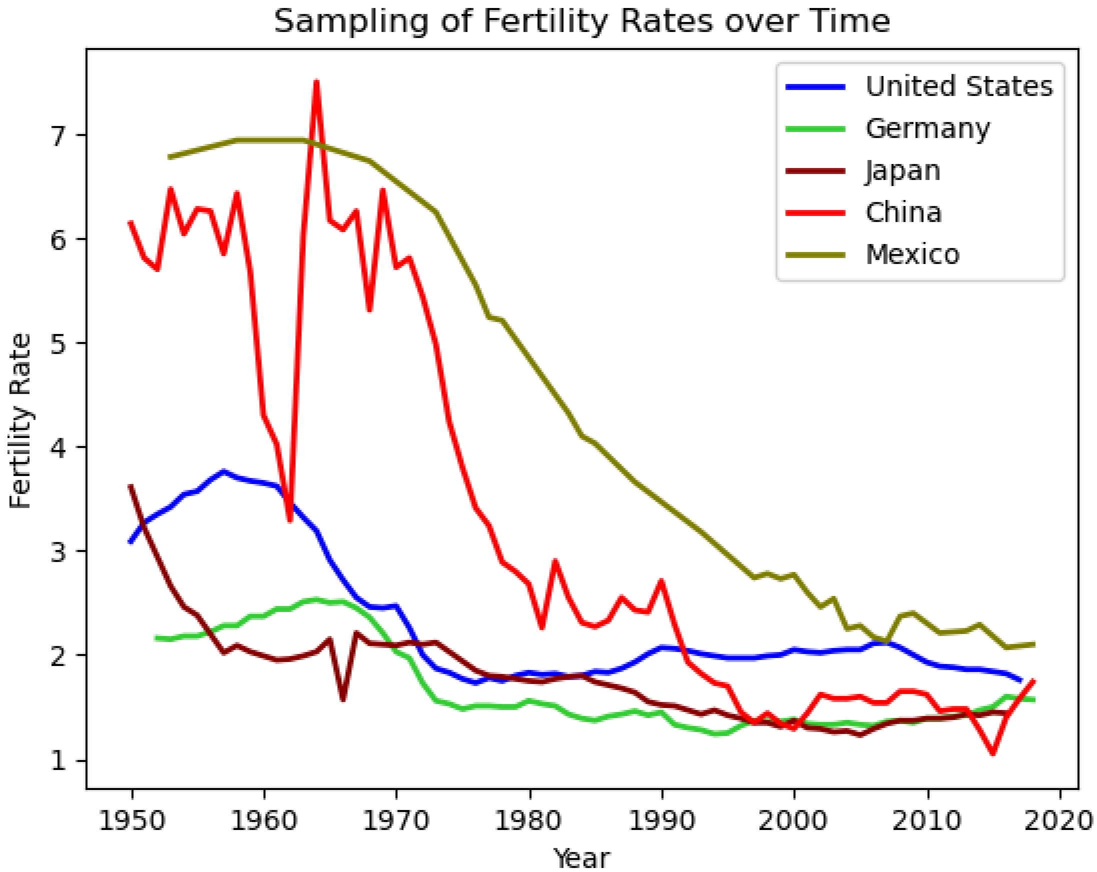
In [ ]:

```
#Graph the fertility rates of several countries
us = df[df['Country'] == "UNITED STATES OF AMERICA"]
germany = df[df['Country'] == "GERMANY"]
japan = df[df['Country'] == "JAPAN"]
china = df[df['Country'] == "CHINA"]
mexico = df[df['Country'] == "MEXICO"]

# Plot the values of 'Column_B' for the filtered DataFrame
plt.plot(us["Year"], us['Fertility Rate'], label='United States', color='blue'
plt.plot(germany["Year"], germany['Fertility Rate'], label='Germany', color='l
plt.plot(japan["Year"], japan['Fertility Rate'], label='Japan', color='darkred
plt.plot(china["Year"], china['Fertility Rate'], label='China', color='red', l
plt.plot(mexico["Year"], mexico['Fertility Rate'], label='Mexico', color='olive

# Add labels and title
plt.xlabel('Year')
plt.ylabel('Fertility Rate')
plt.title("Sampling of Fertility Rates over Time")
plt.legend()

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


Sampling of Fertility Rates over Time
United States
Germany
Japan
China
Mexico
Fertility Rate
Year
1
2
3
4
5
6
7
1950
1960
1970
1980
1990
2000
2010
2020