In [1]:
```
import pandas as pd
```

In [2]:
```
#importing the datasets
```

In [3]:
```
population = pd.read_csv("C:\\Users\\Dipen\\Downloads\\country_population.csv")
fertility_rate = pd.read_csv("C:\\Users\\Dipen\\Downloads\\fertility_rate.csv")
life_expectancy = pd.read_csv("C:\\Users\\Dipen\\Downloads\\life_expectancy.csv")
country = pd.read_csv("C:\\Users\\Dipen\\Downloads\\Metadata_Country.csv")
```

In [4]:
```
country
```

Out[4]:

	Country Code	Region	IncomeGroup	SpecialNotes	TableName	Unnamed: 5
0	ABW	Latin America & Caribbean	High income	SNA data for 2000-2011 are updated from offici...	Aruba	NaN
1	AFG	South Asia	Low income	Fiscal year end: March 20; reporting period fo...	Afghanistan	NaN
2	AGO	Sub-Saharan Africa	Lower middle income	NaN	Angola	NaN
3	ALB	Europe & Central Asia	Upper middle income	NaN	Albania	NaN
4	AND	Europe & Central Asia	High income	WB-3 code changed from ADO to AND to align wit...	Andorra	NaN
...	...	...	...	...	...	...
258	XKX	Europe & Central Asia	Lower middle income	WB-3 code changed from KSV to XKX to align wit...	Kosovo	NaN
259	YEM	Middle East & North Africa	Lower middle income	Based on official government statistics and In...	Yemen, Rep.	NaN
260	ZAF	Sub-Saharan Africa	Upper middle income	Fiscal year end: March 31; reporting period fo...	South Africa	NaN
261	ZMB	Sub-Saharan Africa	Lower middle income	The base year is 2010. National accounts data ...	Zambia	NaN
262	ZWE	Sub-Saharan Africa	Low income	Fiscal year end: June 30; reporting period for...	Zimbabwe	NaN

263 rows × 6 columns

In [5]:
```
country1 = country[['Country Code','Region']]# gives the dataset with 2 column(Country
```

In [6]:
```
country1
```

Out[6]:

	Country Code	Region
0	ABW	Latin America & Caribbean
1	AFG	South Asia
2	AGO	Sub-Saharan Africa
3	ALB	Europe & Central Asia
4	AND	Europe & Central Asia

...	...	...
258	XKX	Europe & Central Asia
259	YEM	Middle East & North Africa
260	ZAF	Sub-Saharan Africa
261	ZMB	Sub-Saharan Africa
262	ZWE	Sub-Saharan Africa

263 rows × 2 columns

In [7]:
```
#checking for null values
country1.isna().sum()
```

Out[7]:
```
Country Code     0
Region          46
dtype: int64
```

In [8]:
```
population
```

Out[8]:

	Country Name	Country Code	Indicator Name	Indicator Code	1960	1961	1962	1963	1964	
0	Aruba	ABW	Population, total	SP.POP.TOTL	54211.0	55438.0	56225.0	56695.0	57032.0	
1	Afghanistan	AFG	Population, total	SP.POP.TOTL	8996351.0	9166764.0	9345868.0	9533954.0	9731361.0	
2	Angola	AGO	Population, total	SP.POP.TOTL	5643182.0	5753024.0	5866061.0	5980417.0	6093321.0	
3	Albania	ALB	Population, total	SP.POP.TOTL	1608800.0	1659800.0	1711319.0	1762621.0	1814135.0	
4	Andorra	AND	Population, total	SP.POP.TOTL	13411.0	14375.0	15370.0	16412.0	17469.0	
...	...	...	...	...	...	...	...	...	...	
259	Kosovo	XKX	Population, total	SP.POP.TOTL	947000.0	966000.0	994000.0	1022000.0	1050000.0	
260	Yemen, Rep.	YEM	Population, total	SP.POP.TOTL	5172135.0	5260501.0	5351799.0	5446063.0	5543339.0	
261	South Africa	ZAF	Population, total	SP.POP.TOTL	17456855.0	17920673.0	18401608.0	18899275.0	19412975.0	1
262	Zambia	ZMB	Population, total	SP.POP.TOTL	3044846.0	3140264.0	3240587.0	3345145.0	3452942.0	
263	Zimbabwe	ZWE	Population, total	SP.POP.TOTL	3747369.0	3870756.0	3999419.0	4132756.0	4269863.0	

264 rows × 61 columns

In [9]:
```
#updating the population dataset(removing unnecessary columns)
population.drop(['Country Name','Indicator Name','Indicator Code'],axis =1,inplace=True)
population.head()
```

Out[9]:

Country Code	1960	1961	1962	1963	1964	1965	1966	1967	1968

0	ABW	54211.0	55438.0	56225.0	56695.0	57032.0	57360.0	57715.0	58055.0	58386.0
1	AFG	8996351.0	9166764.0	9345868.0	9533954.0	9731361.0	9938414.0	10152331.0	10372630.0	10604346.0
2	AGO	5643182.0	5753024.0	5866061.0	5980417.0	6093321.0	6203299.0	6309770.0	6414995.0	6523791.0
3	ALB	1608800.0	1659800.0	1711319.0	1762621.0	1814135.0	1864791.0	1914573.0	1965598.0	2022272.0
4	AND	13411.0	14375.0	15370.0	16412.0	17469.0	18549.0	19647.0	20758.0	21890.0

5 rows × 58 columns

In [10]:
```
population.isna().sum()
```

Out[10]:
```
Country Code    0
1960            4
1961            4
1962            4
1963            4
1964            4
1965            4
1966            4
1967            4
1968            4
1969            4
1970            4
1971            4
1972            4
1973            4
1974            4
1975            4
1976            4
1977            4
1978            4
1979            4
1980            4
1981            4
1982            4
1983            4
1984            4
1985            4
1986            4
1987            4
1988            4
1989            4
1990            2
1991            2
1992            3
1993            3
1994            3
1995            2
1996            2
1997            2
1998            1
1999            1
2000            1
2001            1
2002            1
2003            1
2004            1
2005            1
2006            1
2007            1
2008            1
2009            1
```

```
2010            1
2011            1
2012            2
2013            2
2014            2
2015            2
2016            2
dtype: int64
```

In [11]:
```
population.shape
```

Out[11]:
```
(264, 58)
```

In [12]:
```
#removing the null values
population.dropna(axis =0,inplace=True)
```

In [13]:
```
population.shape
```

Out[13]:
```
(258, 58)
```

In [14]:
```
#Using this we go through over each element in the range and converts each integer to a
#The result is a list of strings representing the years from 1960 to 2016.
years = [str(i) for i in range(1960,2017)]
print(years)
```

```
['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']
```

In [15]:
```
# melt
#The melt() method reshapes the DataFrame into a long table with one row for each each c

population1 = pd.melt(population,
        id_vars='Country Code',
        value_vars=years,
        var_name='Year',
        value_name='Population')
```

In [16]:
```
population1
```

Out[16]:

	Country Code	Year	Population
0	ABW	1960	54211.0
1	AFG	1960	8996351.0
2	AGO	1960	5643182.0
3	ALB	1960	1608800.0
4	AND	1960	13411.0
...	...	...	...
14701	XKX	2016	1816200.0
14702	YEM	2016	27584213.0
14703	ZAF	2016	56015473.0
14704	ZMB	2016	16591390.0
14705	ZWE	2016	16150362.0

14706 rows × 3 columns

In [17]:
```
population1.shape
```

Out[17]:
```
(14706, 3)
```

In [18]:
```
population1.head()
```

Out[18]:

	Country Code	Year	Population
0	ABW	1960	54211.0
1	AFG	1960	8996351.0
2	AGO	1960	5643182.0
3	ALB	1960	1608800.0
4	AND	1960	13411.0

In [19]:
```
country1.head()
```

Out[19]:

	Country Code	Region
0	ABW	Latin America & Caribbean
1	AFG	South Asia
2	AGO	Sub-Saharan Africa
3	ALB	Europe & Central Asia
4	AND	Europe & Central Asia

In [20]:
```
#merging the datasets(country1 and population1)
```

In [21]:
```
merg1 = pd.merge(country1,population1,how='left',on='Country Code')
```

In [22]:
```
merg1.head()
```

Out[22]:

	Country Code	Region	Year	Population
0	ABW	Latin America & Caribbean	1960	54211.0
1	ABW	Latin America & Caribbean	1961	55438.0
2	ABW	Latin America & Caribbean	1962	56225.0
3	ABW	Latin America & Caribbean	1963	56695.0
4	ABW	Latin America & Caribbean	1964	57032.0

In [23]:
```
fertility_rate.head()
```

Out[23]:

	Country Name	Country Code	Indicator Name	Indicator Code	1960	1961	1962	1963	1964	1965	...	2007	2008	2009
0	Aruba	ABW	Fertility rate, total (births	SP.DYN.TFRT.IN	4.820	4.655	4.471	4.271	4.059	3.842	...	1.763	1.764	1.769

			per woman)											
1	Afghanistan	AFG	Fertility rate, total (births per woman)	SP.DYN.TFRT.IN	7.450	7.450	7.450	7.450	7.450	7.450	...	6.460	6.254	6.038
2	Angola	AGO	Fertility rate, total (births per woman)	SP.DYN.TFRT.IN	7.478	7.524	7.563	7.592	7.611	7.619	...	6.368	6.307	6.238
3	Albania	ALB	Fertility rate, total (births per woman)	SP.DYN.TFRT.IN	6.489	6.401	6.282	6.133	5.960	5.773	...	1.668	1.650	1.646
4	Andorra	AND	Fertility rate, total (births per woman)	SP.DYN.TFRT.IN	NaN	NaN	NaN	NaN	NaN	NaN	...	1.180	1.250	1.190

5 rows × 61 columns

In [24]:
```
# melt - unpivot the dataset

fertility_rate1 = pd.melt(fertility_rate,
        id_vars='Country Code',
        value_vars=years,
        var_name='Year',
        value_name='Fertility_rate')
```

In [25]:
```
fertility_rate1.head()
```

Out[25]:

	Country Code	Year	Fertility_rate
0	ABW	1960	4.820
1	AFG	1960	7.450
2	AGO	1960	7.478
3	ALB	1960	6.489
4	AND	1960	NaN

In [26]:
```
life_expectancy.head()
```

Out[26]:

	Country Name	Country Code	Indicator Name	Indicator Code	1960	1961	1962	1963	1964	1965	...	2007	20
0	Aruba	ABW	Life expectancy at birth, total (years)	SP.DYN.LE00.IN	65.662	66.074	66.444	66.787	67.113	67.435	...	74.576	74.7
1	Afghanistan	AFG	Life expectancy at birth,	SP.DYN.LE00.IN	32.292	32.742	33.185	33.624	34.060	34.495	...	59.694	60.2

			total (years)										
2	Angola	AGO	Life expectancy at birth, total (years)	SP.DYN.LE00.IN	33.251	33.573	33.914	34.272	34.645	35.031	...	55.096	56.1
3	Albania	ALB	Life expectancy at birth, total (years)	SP.DYN.LE00.IN	62.279	63.298	64.187	64.911	65.461	65.848	...	75.656	75.9
4	Andorra	AND	Life expectancy at birth, total (years)	SP.DYN.LE00.IN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	N

5 rows × 61 columns

In [27]:
```
# melt - unpivot the dataset

Life_expectancy1 = pd.melt(life_expectancy,
        id_vars='Country Code',
        value_vars=years,
        var_name='Year',
        value_name='Life_expectancy')
```

In [28]:
```
Life_expectancy1.head()
```

Out[28]:

	Country Code	Year	Life_expectancy
0	ABW	1960	65.662
1	AFG	1960	32.292
2	AGO	1960	33.251
3	ALB	1960	62.279
4	AND	1960	NaN

In [29]:
```
# Merge the data into one dataframe
merg1 = pd.merge(country1, population1, how='left', on='Country Code')
merg2 = pd.merge(merg1, Life_expectancy1, how='left', on=['Country Code', 'Year'])
merg3 = pd.merge(merg2, fertility_rate1, how='left', on=['Country Code', 'Year'])

# Remove remaining lines with missing values
# They will appear if a country is in one dataset but not in another one
merg3.dropna(axis=0, inplace=True)
```

In [30]:
```
merg3.head()
```

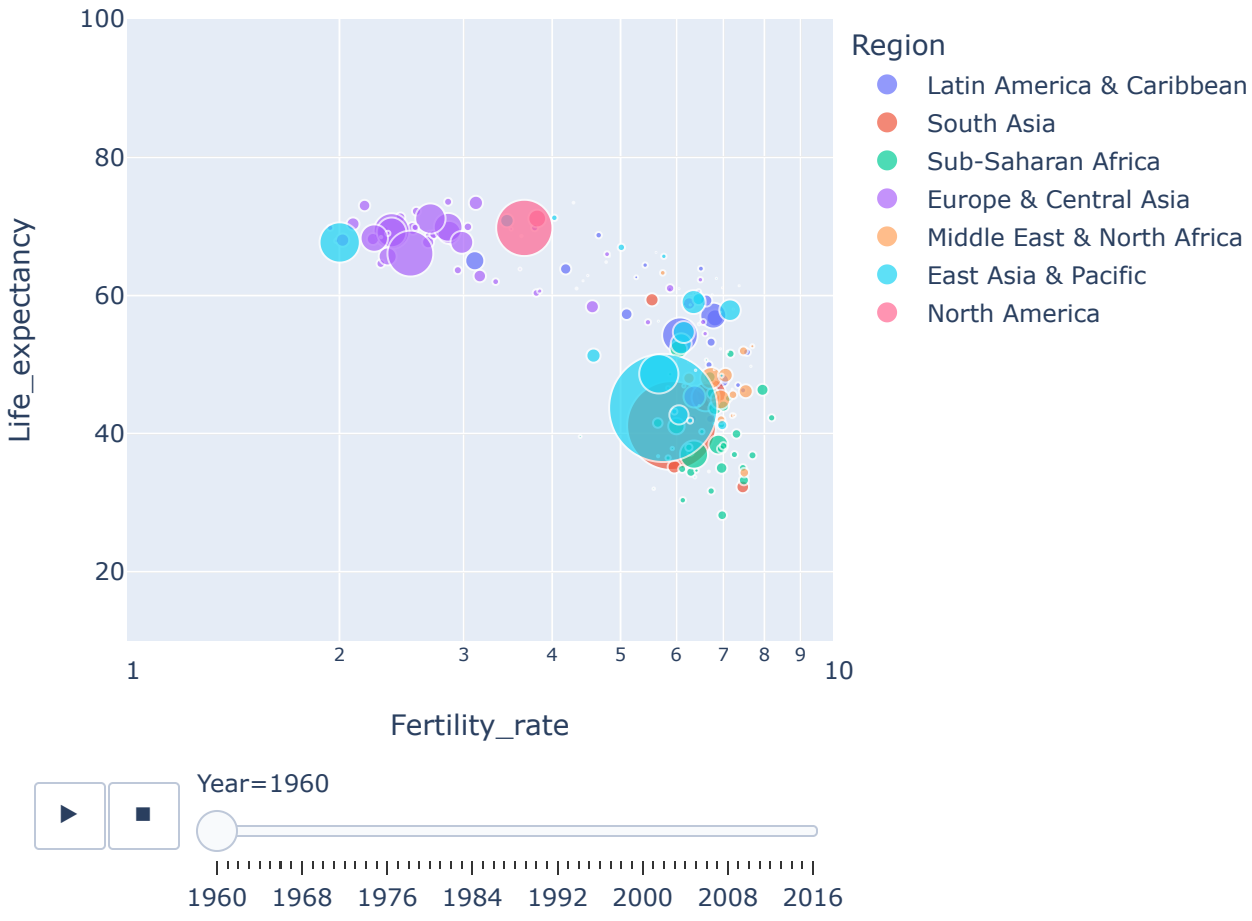
Out[30]:

	Country Code	Region	Year	Population	Life_expectancy	Fertility_rate
0	ABW	Latin America & Caribbean	1960	54211.0	65.662	4.820
1	ABW	Latin America & Caribbean	1961	55438.0	66.074	4.655
2	ABW	Latin America & Caribbean	1962	56225.0	66.444	4.471
3	ABW	Latin America & Caribbean	1963	56695.0	66.787	4.271

4	ABW	Latin America & Caribbean	1964	57032.0	67.113	4.059

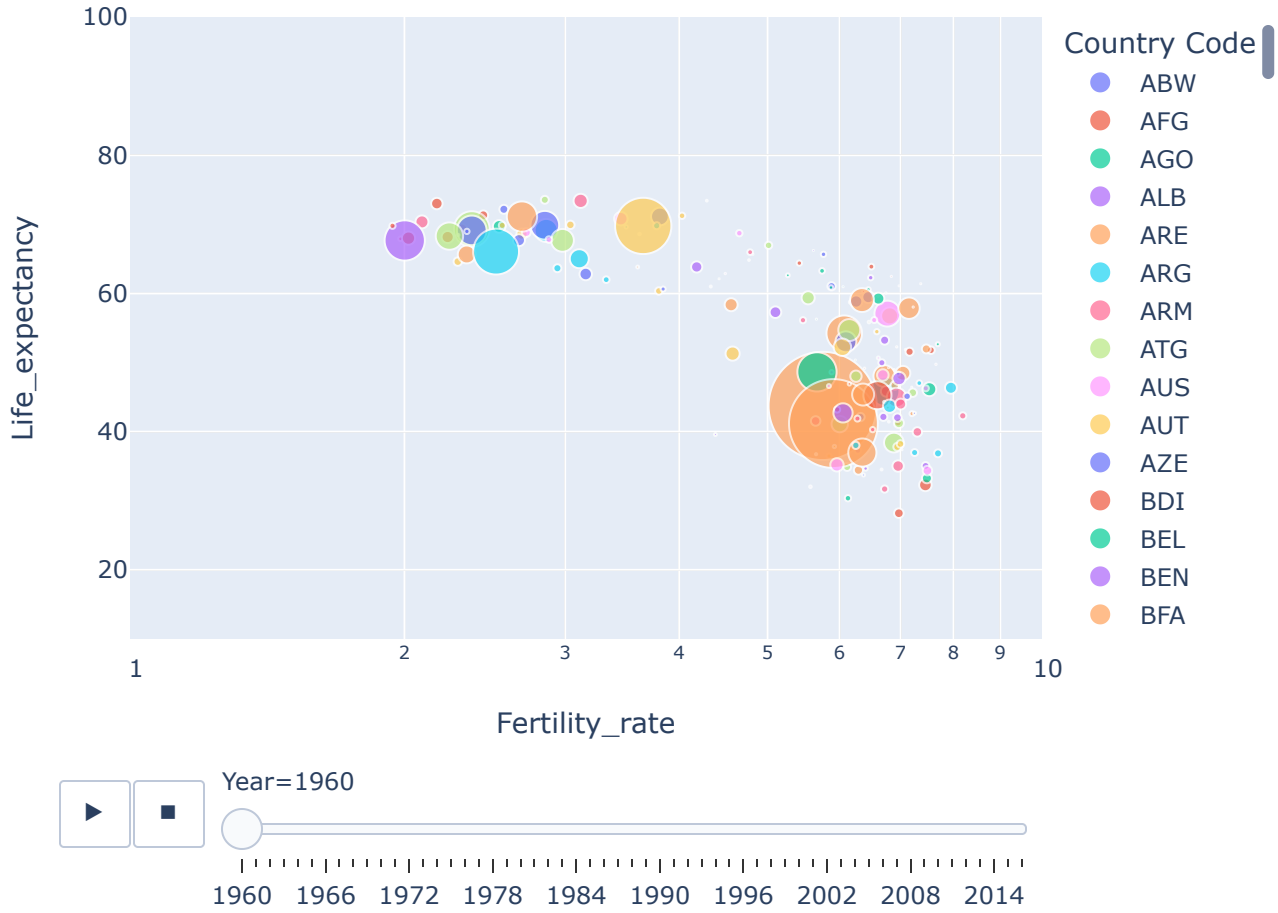
In [31]:
```
#importing the plot for clear visualization
import plotly.express as px
```

In [32]:
```
#Scatter plot between Fertility_rate and Life_expectancy(Region wise)
px.scatter(merg3,
           x="Fertility_rate",
           y="Life_expectancy",
           animation_frame="Year",
           animation_group="Country Code",
           size="Population",
           color="Region",
           hover_name="Country Code",
           log_x=True,
           size_max=55,
           range_x=[1,10],
           range_y=[10,100])
```

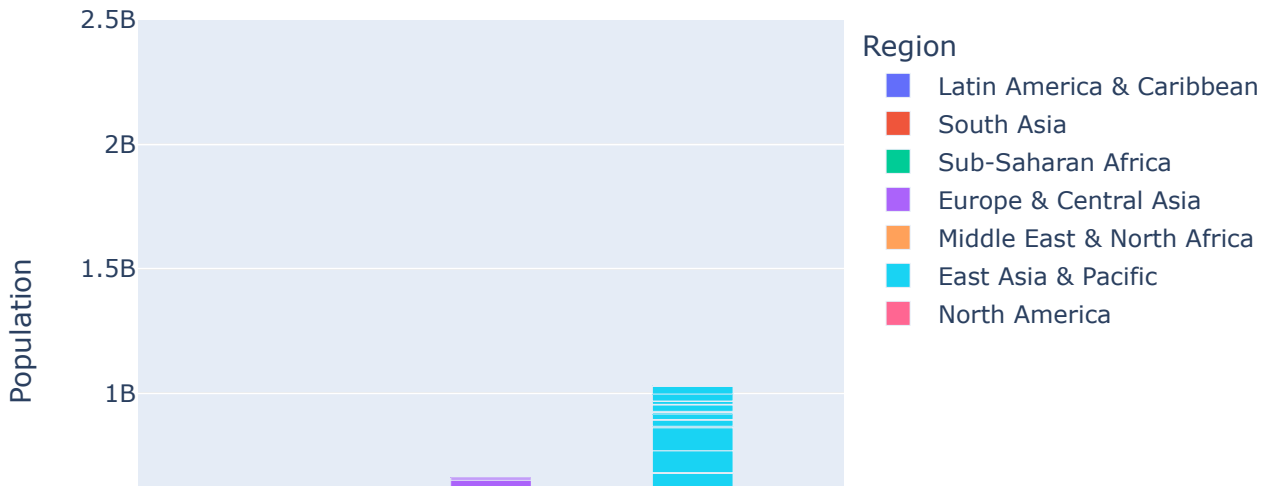


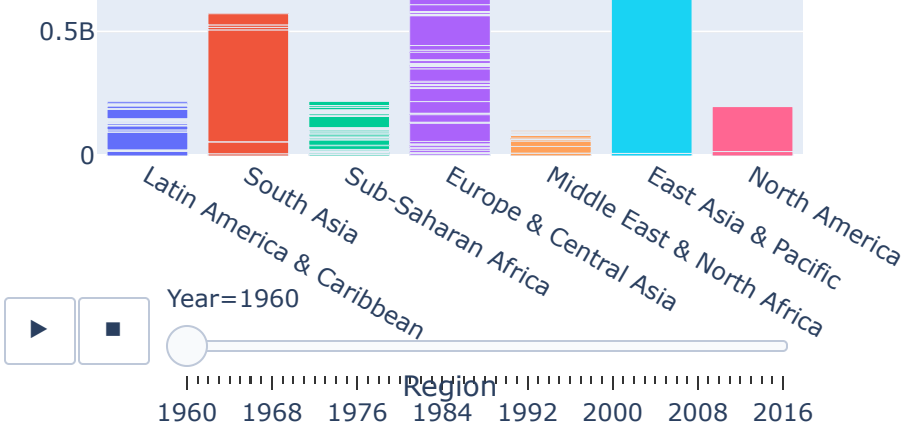
In [33]:
```
#Scatter plot between Fertility_rate and Life_expectancy(Country wise)
px.scatter(merg3,
           x="Fertility_rate",
           y="Life_expectancy",
           animation_frame="Year",
           animation_group="Country Code",
           size="Population",
           color="Country Code",
           hover_name="Country Code",
```

```
            log_x=True,
            size_max=55,
            range_x=[1,10],
            range_y=[10,100])
```



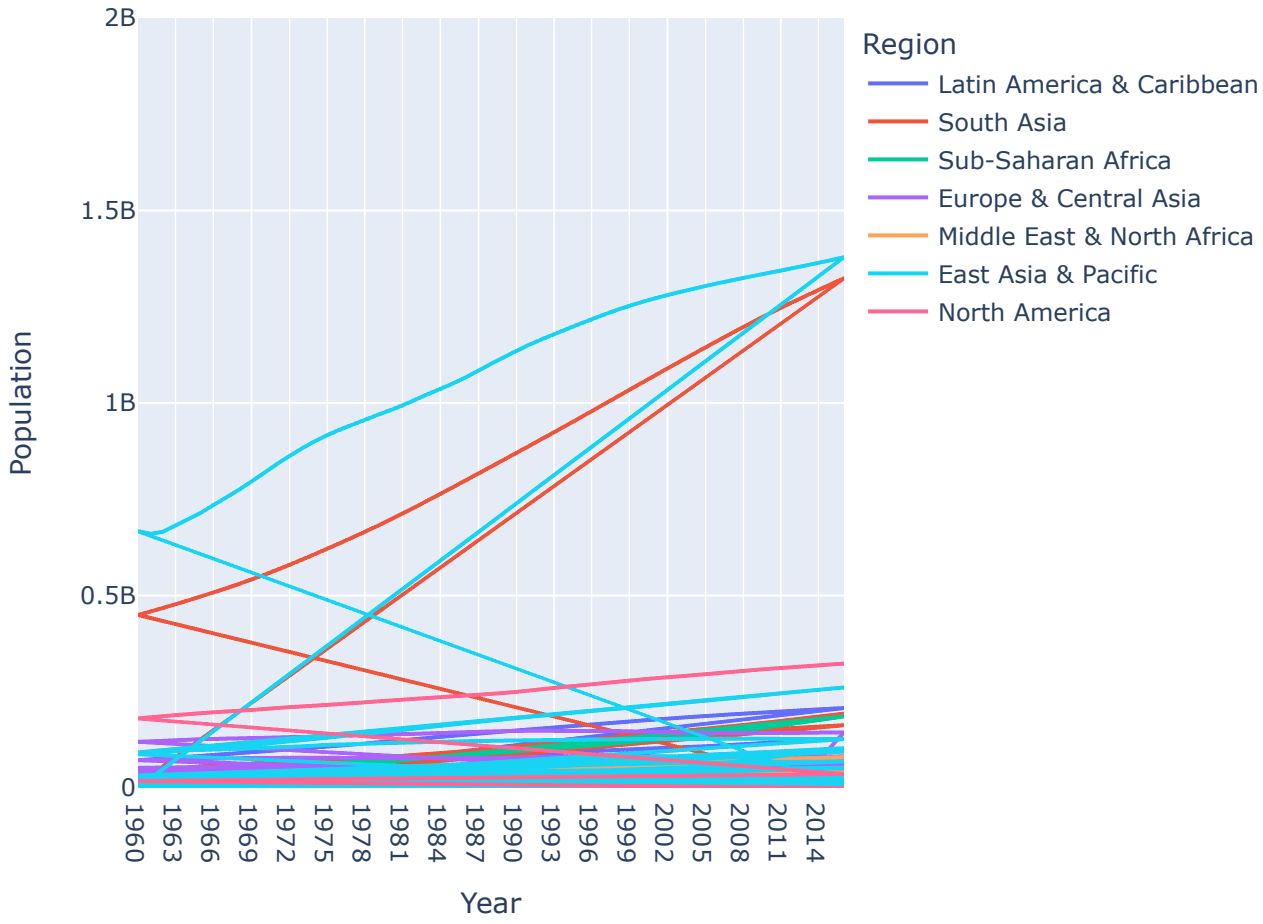
In [34]:
```
#bar chart of popultion
px.bar(merg3,
            x="Region",
            y="Population",
            animation_frame="Year",
            animation_group="Country Code",
            color="Region",
            range_y=[0,2500000000])
```





In [35]:
```
import matplotlib.pyplot as plt
import seaborn as sns
```

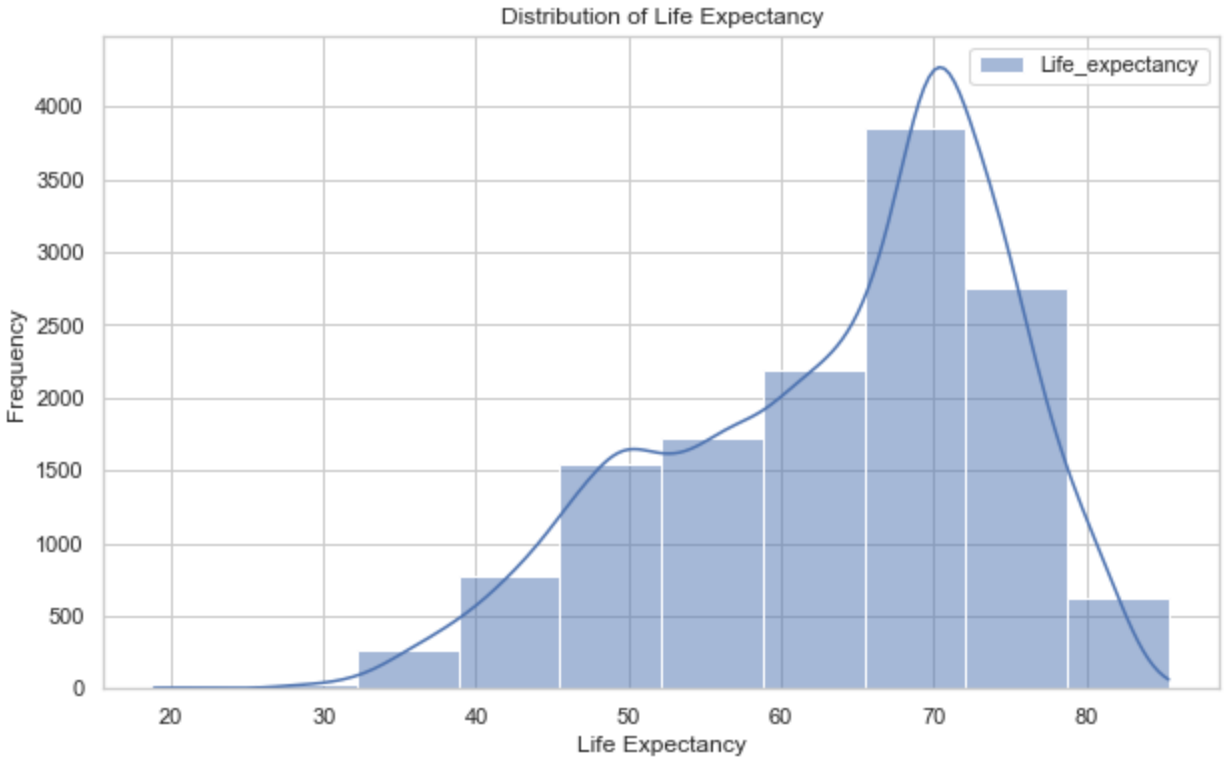
In [36]:
```
#line graph between year and population (region wise)
px.line(merg3,
            x="Year",
            y="Population",
            #animation_frame="Year",
            #animation_group="Country Code",
            color="Region",
            range_y=[0,2000000000])
```



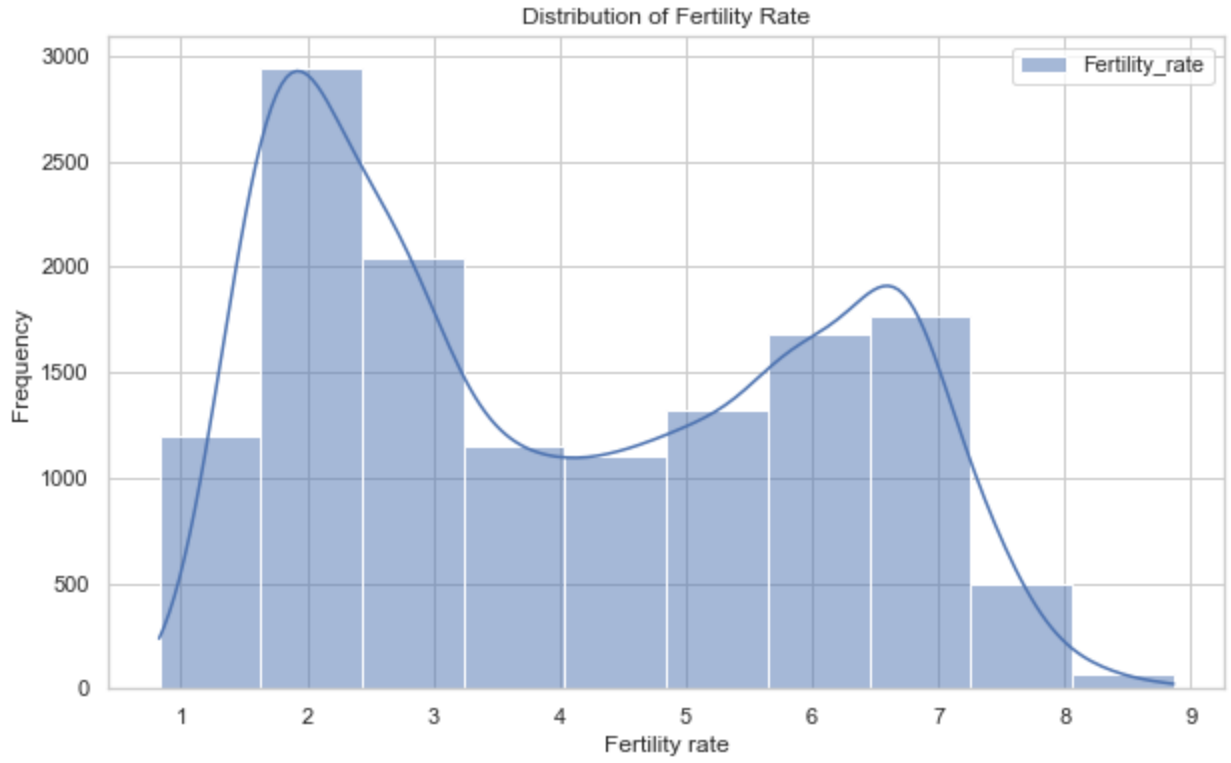
In [ ]:
```
```

In [37]:
```
# Distribution of Life Expectency
```

```
import seaborn as sns
import matplotlib.pyplot as plt


sns.set(style="whitegrid")
plt.figure(figsize=(10, 6))

# Plot the distribution of life expectancy
sns.histplot(Life_expectancy1, bins=10, kde=True, color='skyblue')

# Set plot labels and title
plt.xlabel('Life Expectancy')
plt.ylabel('Frequency')
plt.title('Distribution of Life Expectancy')

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

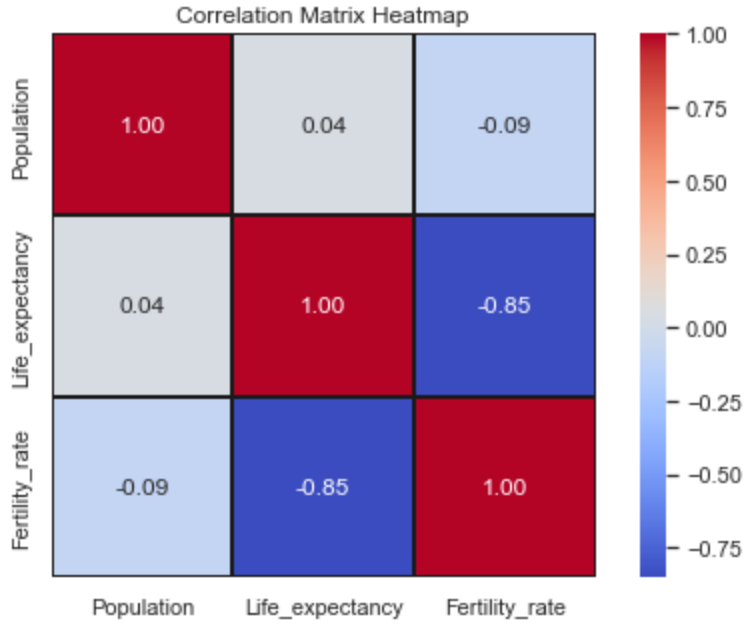


In [38]:
```
# distribution of Fertility Rate
sns.set(style="whitegrid")
plt.figure(figsize=(10, 6))

# Plot the distribution of fertility Rate
sns.histplot(fertility_rate1, bins=10, kde=True, color='skyblue')

# Set plot labels and title
plt.xlabel('Fertility rate')
plt.ylabel('Frequency')
plt.title('Distribution of Fertility Rate')

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



In [39]:
```
correlation_matrix = merg3.corr()
```

In [40]:
```
# correlation analysis
plt.figure(figsize=(8,5))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f", linewidths=.1,li
plt.title('Correlation Matrix Heatmap')
plt.show()
```



In [ ]: