Assignment

Statistical Modeling with Python (Batch 3)

Name of the Participant: Saniat Muntasir

Assignment Title:

Exploring the Health and Economic Determinants of Life Expectancy in Bangladesh (2000-2022): A Statistical Modeling Approach Using Python.

Part 1: Data Cleaning

1. Check for missing values in the dataset.

```
df.isna().sum()
```

	0
year	0
life_expectancy	0
gdp	0
health_exp_per_capita	0
govt_health_exp_per_capita	0
private_health_exp_per_capita	4
out_of_pocket_exp_pct	0
unemployment	0
literacy_rate	0

dtype: int64

So, there are 4 missing values in the “private_health_exp_per_capita” column.

2. Handle missing data using appropriate techniques.

```
[23] df1 = df.copy()
```

```
[24] df1['private_health_exp_per_capita'].fillna(df1['private_health_exp_per_capita'].mean(),inplace=True)
```

```
/tmp/ipython-input-3762798118.py:1: FutureWarning: A value is trying to be set on a copy of a DataFrame
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate ob

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inpla


  df1['private_health_exp_per_capita'].fillna(df1['private_health_exp_per_capita'].mean(),inplace=True)
```

```
[26] df1.isna().sum()
```

	0
year	0
life_expectancy	0
gdp	0
health_exp_per_capita	0
govt_health_exp_per_capita	0
private_health_exp_per_capita	0
out_of_pocket_exp_pct	0
unemployment	0
literacy_rate	0

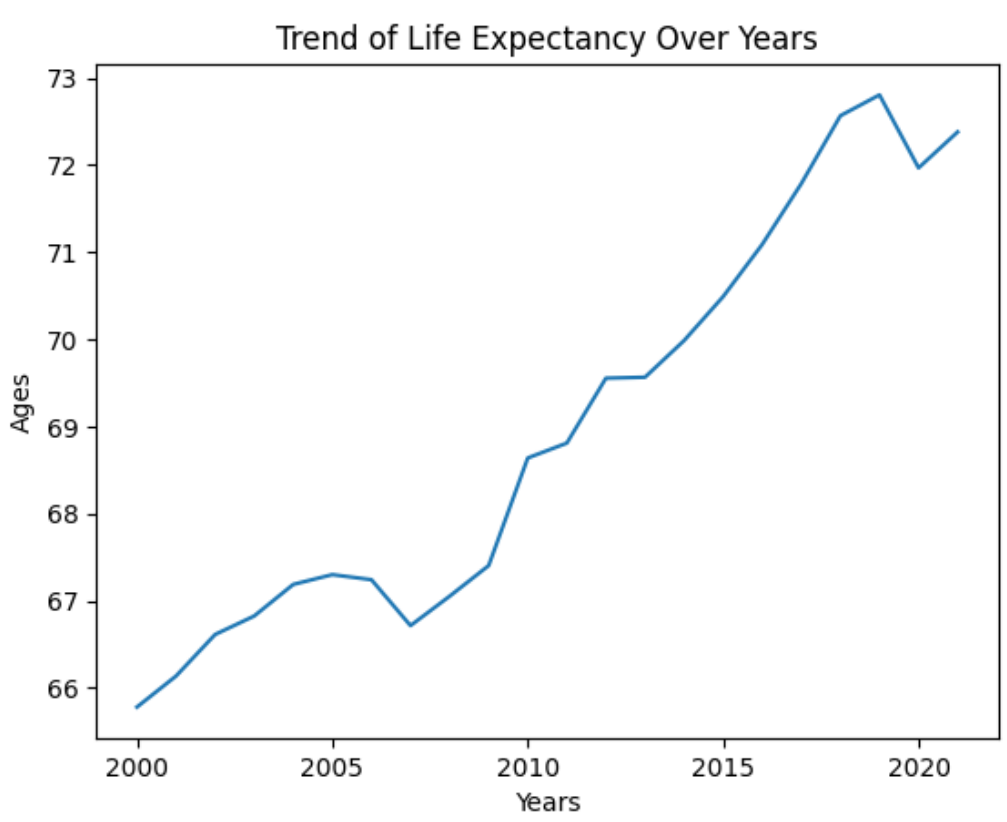
dtype: int64

To handle missing data, first we have to copy the file and assign into another data frame so that the first data frame stay intact. Then, we replace the null values with the average value of that column which is 18.948766122777783.

Part 2: Data Visualization

1. Line Plot: Trend of life_expectancy over years.

```
plt.plot(df1['year'], df1['life_expectancy'])
plt.xlabel("Years")
plt.ylabel("Ages")
plt.title("Trend of Life Expectancy Over Years")
plt.show()
```

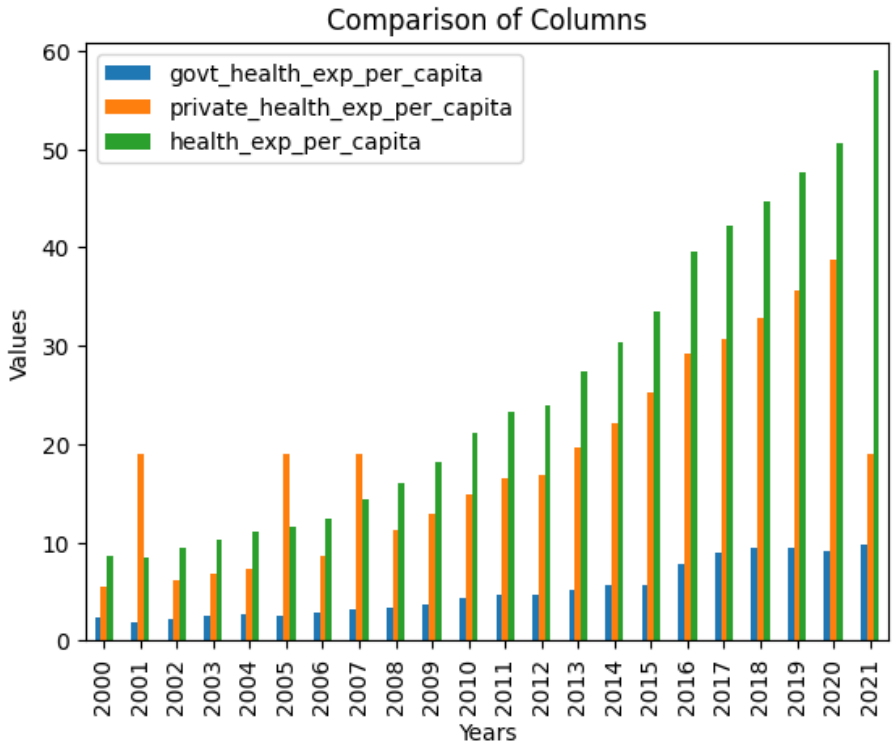


On the above Line Plot, the line shows the trends of life expectancy over the years.

2. Bar Chart: Compare govt_health_exp_per_capita, private_health_exp_per_capita and health_exp_per_capita for a chosen year.

```
plt.figure(figsize=(16, 9), dpi=300)

df1.plot(
    x='year',
    y=['govt_health_exp_per_capita', 'private_health_exp_per_capita', 'health_exp_per_capita'],
    kind='bar',
    xlabel='Years',
    ylabel='Values',
    title='Comparison of Columns'
)

plt.show()
```

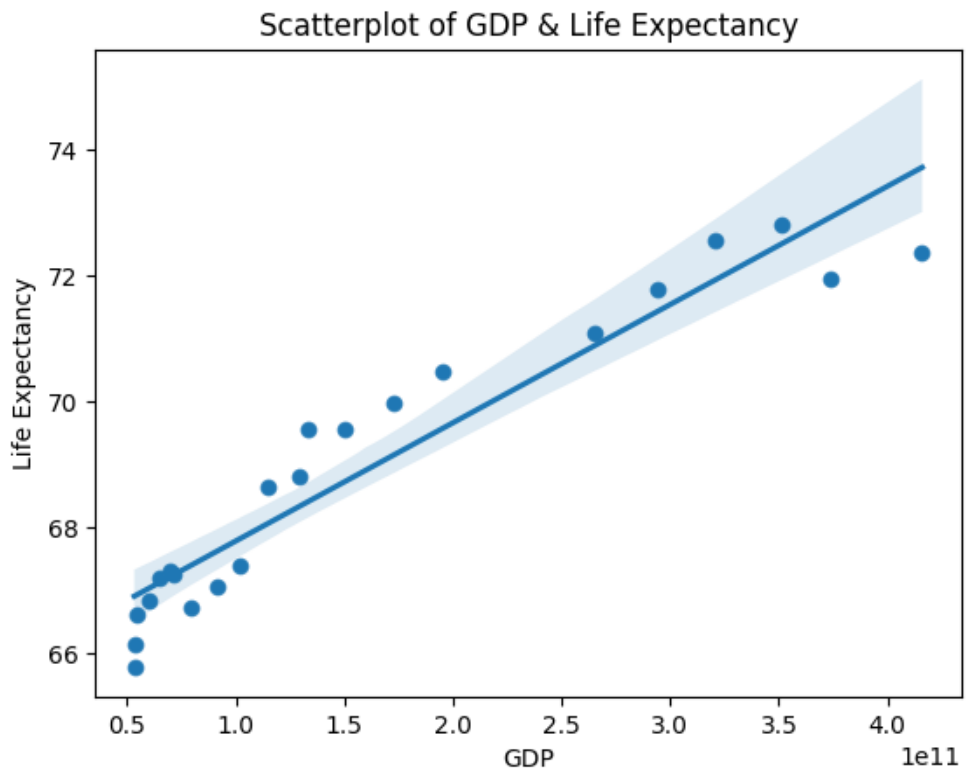
```
<Figure size 4800x2700 with 0 Axes>
```



The Bar Chart shows the comparison of the columns throughout the year.

3. Scatter Plot: Relation between GDP and Life_expectancy.

```
sns.scatterplot(data=df1, x='gdp', y='life_expectancy')
sns.regplot(data=df1, x='gdp', y='life_expectancy')

plt.title("Scatterplot of GDP & Life Expectancy")
plt.xlabel("GDP")
plt.ylabel("Life Expectancy")
plt.show()
```

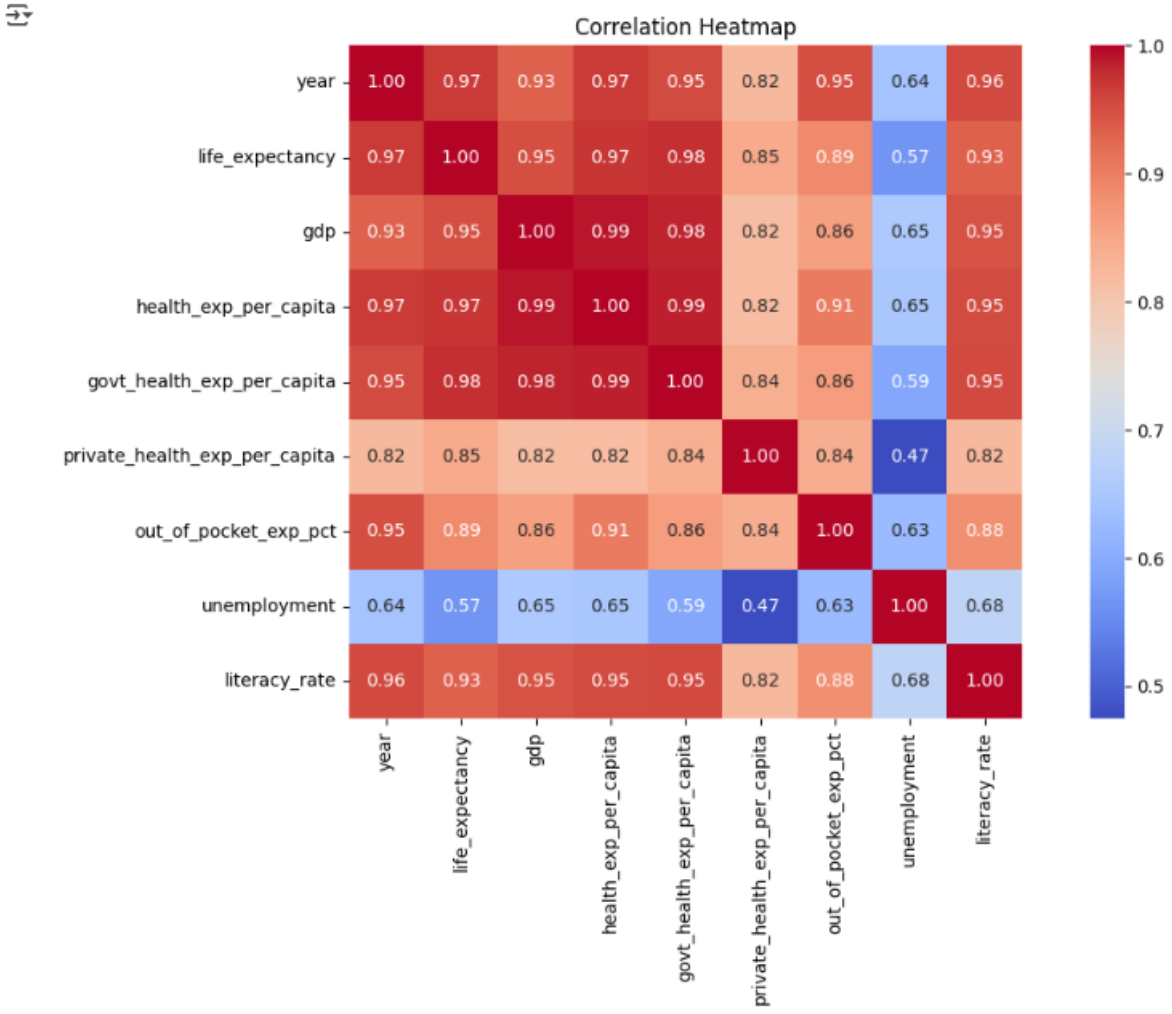


The Scatter Plot shows the relation between GDP and Life Expectancy & the Regression Line shows the prediction.

Part 3: Correlation and Regression Analysis

1. Generate a correlation matrix of all numeric variables.

```
plt.figure(figsize=(12,8))
numeric_df = df1.select_dtypes(include=['number'])
corr = numeric_df.corr()
sns.heatmap(corr, annot=True, fmt=".2f", cmap='coolwarm', square=True)
plt.title('Correlation Heatmap')
plt.tight_layout()
plt.show()
```



2. Fit a multiple linear regression model with life_expectancy as the dependent variable and the following as predictors:
    - gdp
    - health_exp_per_capita
    - unemployment
    - literacy_rate.

```
[23] import statsmodels.formula.api as smf
     model = smf.ols("life_expectancy ~ gdp + health_exp_per_capita + unemployment + literacy_rate", data=df).fit()

     print(model.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:        life_expectancy   R-squared:                       0.959
Model:                            OLS   Adj. R-squared:                  0.949
Method:                 Least Squares   F-statistic:                     98.94
Date:                Tue, 26 Aug 2025   Prob (F-statistic):           1.54e-11
Time:                        05:31:03   Log-Likelihood:                -14.174
No. Observations:                  22   AIC:                             38.35
Df Residuals:                      17   BIC:                             43.80
Df Model:                           4
Covariance Type:            nonrobust
===========================================================================================
                              coef    std err          t      P>|t|      [0.025      0.975]
-------------------------------------------------------------------------------------------
Intercept                  64.9087      1.987     32.675      0.000      60.717      69.100
gdp                    -9.666e-12   7.05e-12     -1.371      0.188   -2.45e-11    5.21e-12
health_exp_per_capita      0.2054      0.057      3.626      0.002       0.086       0.325
unemployment              -0.5237      0.285     -1.836      0.084      -1.125       0.078
literacy_rate              0.0443      0.043      1.035      0.315      -0.046       0.135
==============================================================================
Omnibus:                        1.379   Durbin-Watson:                   0.804
Prob(Omnibus):                  0.502   Jarque-Bera (JB):                1.233
Skew:                          -0.502   Prob(JB):                        0.540
Kurtosis:                       2.420   Cond. No.                     3.57e+12
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 3.57e+12. This might indicate that there are
strong multicollinearity or other numerical problems.
```

3. **Interpret The Result**
    - The model explains ~96% of life expectancy variation.
    - Health expenditure per capita is the most reliable predictor: countries that spend more on health per person tend to have higher life expectancy.
    - Unemployment might negatively affect life expectancy, but evidence is weaker.
    - GDP and literacy rate do not show a significant direct effect in this model — likely because their effects overlap with health spending (multicollinearity issue).

Part 4: Data Transformation

a) Apply log-transformation (log_gdp) = log(gdp) to reduce skewness and stabilize variance.
b) Refit your multiple linear regression model using the log-transformed variables.

```
df1['log_gdp'] = np.log(df1['gdp'])

model = smf.ols("life_expectancy ~ log_gdp + health_exp_per_capita + unemployment + literacy_rate",data=df1).fit()

print(model.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:        life_expectancy   R-squared:                       0.969
Model:                            OLS   Adj. R-squared:                  0.962
Method:                 Least Squares   F-statistic:                     133.2
Date:                Tue, 26 Aug 2025   Prob (F-statistic):           1.35e-12
Time:                        17:21:16   Log-Likelihood:                -11.016
No. Observations:                  22   AIC:                             32.03
Df Residuals:                      17   BIC:                             37.49
Df Model:                           4
Covariance Type:            nonrobust
==========================================================================================
                             coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------------------
Intercept                -10.6486     26.693     -0.399      0.695     -66.966      45.668
log_gdp                    3.1728      1.111      2.856      0.011       0.829       5.517
health_exp_per_capita      0.0167      0.047      0.358      0.725      -0.082       0.115
unemployment              -0.3475      0.257     -1.353      0.194      -0.889       0.194
literacy_rate             -0.0083      0.041     -0.203      0.842      -0.094       0.078
==============================================================================
Omnibus:                        6.238   Durbin-Watson:                   0.915
Prob(Omnibus):                  0.044   Jarque-Bera (JB):                1.747
Skew:                          -0.051   Prob(JB):                        0.417
Kurtosis:                       1.623   Cond. No.                     1.97e+04
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.97e+04. This might indicate that there are
strong multicollinearity or other numerical problems.
```

Part 5: Time Series Modeling of Life Expectancy

1. Convert life_expectancy to a time series object.

```
df1['year'] = pd.to_datetime(df1['year'], format='%Y')
df1.set_index('year', inplace=True)

df1['life_expectancy'] = pd.to_numeric(df1['life_expectancy'], errors='coerce')

life_expectancy_ts = df1['life_expectancy']

print(life_expectancy_ts.head())
print(life_expectancy_ts.dtypes)
```

```
year
2000-01-01    65
2001-01-01    66
2002-01-01    66
2003-01-01    66
2004-01-01    67
Name: life_expectancy, dtype: int64
int64
```

```
[ ] Start coding or generate with AI.
```

2. Conduct an ADF test to check for stationarity. Apply differencing if necessary.

```
[49] adf_result = sts.adfuller(df1['life_expectancy'])
     print(adf_result)
     print("ADF Statistic:", adf_result[0])
     print("p-value:", adf_result[1])

     if adf_result[1] < 0.05:
         print("Series is stationary")
     else:
         print("Series is not stationary - apply differencing")
```

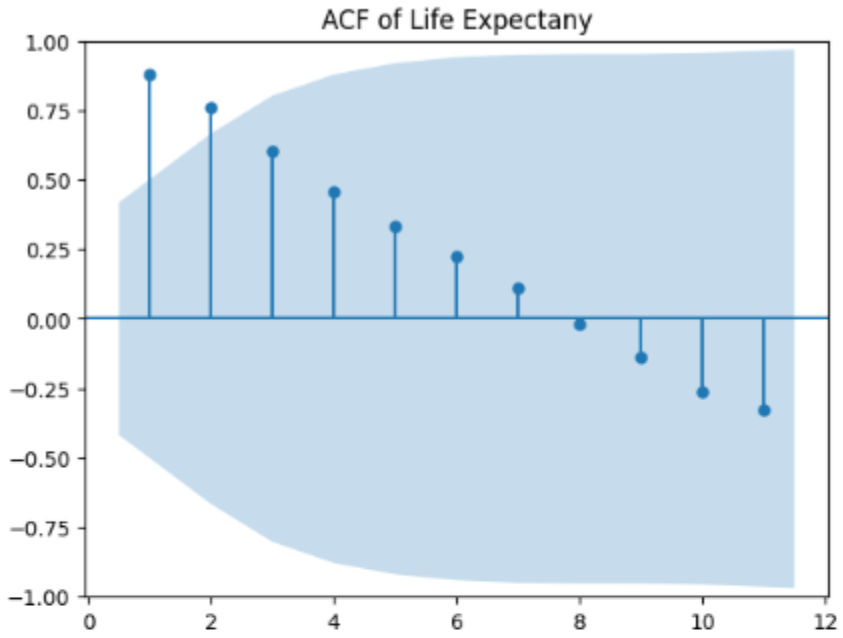
```
(np.float64(-4.368300731640477), np.float64(0.00033761503497393297), 9, 12, {'1%
ADF Statistic: -4.368300731640477
p-value: 0.00033761503497393297
Series is stationary
```

```
Start coding or generate with AI.
```

3. Generate and interpret ACF and PACF plots.

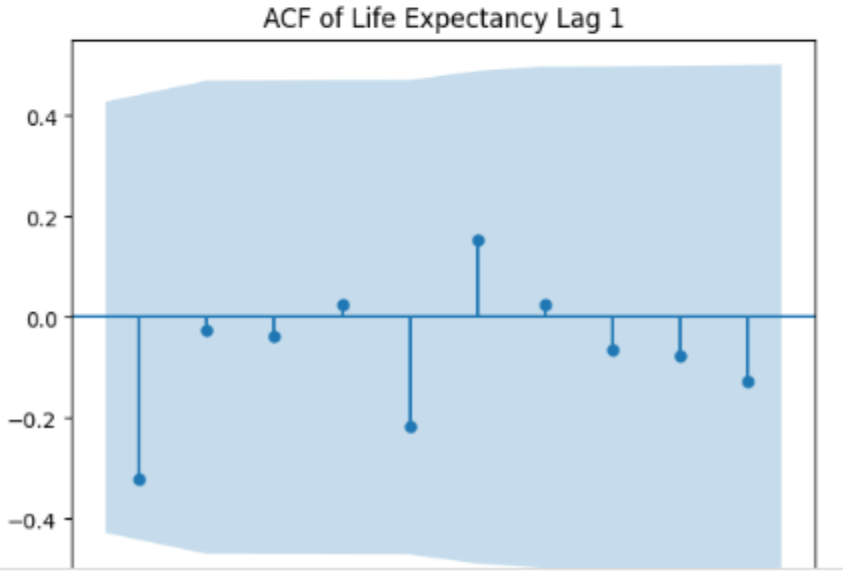
```
sgt.plot_acf(df['life_expectancy'],zero = False )
plt.title("ACF of Life Expectany")
```

```
Text(0.5, 1.0, 'ACF of Life Expectany')
```




```
[33] sgt.plot_acf(df1['life_expectancy_lag_1'].dropna(),zero = False )
     plt.title("ACF of Life Expectancy Lag 1")
     plt.axis('tight')
```

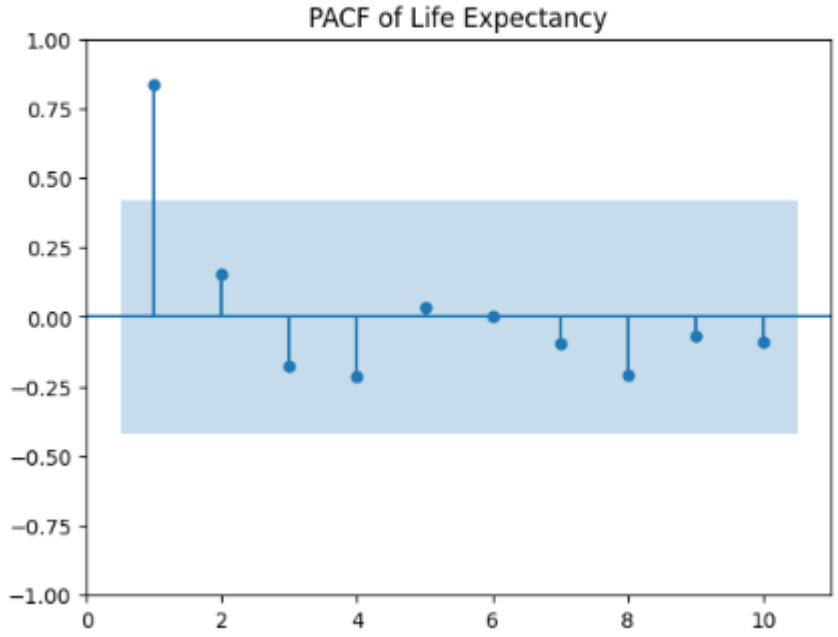
```
(np.float64(0.0),
 np.float64(11.0),
 np.float64(-0.5516244890623477),
 np.float64(0.5516244890623477))
```



Variables Terminal

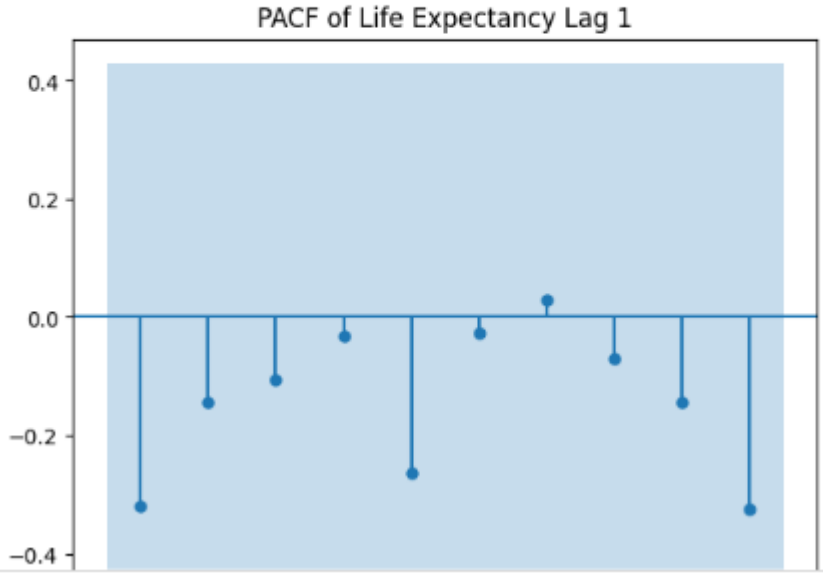
```
sgt.plot_pacf(df1['life_expectancy'],lags = 10,zero = False )
plt.title("PACF of Life Expectancy")
```

```
Text(0.5, 1.0, 'PACF of Life Expectancy')
```




```
sgt.plot_pacf(df1['life_expectancy_lag_1'].dropna(),lags = 10,zero = False )
plt.title("PACF of Life Expectancy Lag 1")
plt.axis('tight')
```

```
(np.float64(0.0),
 np.float64(11.0),
 np.float64(-0.4704691262093458),
 np.float64(0.4704691262093457))
```



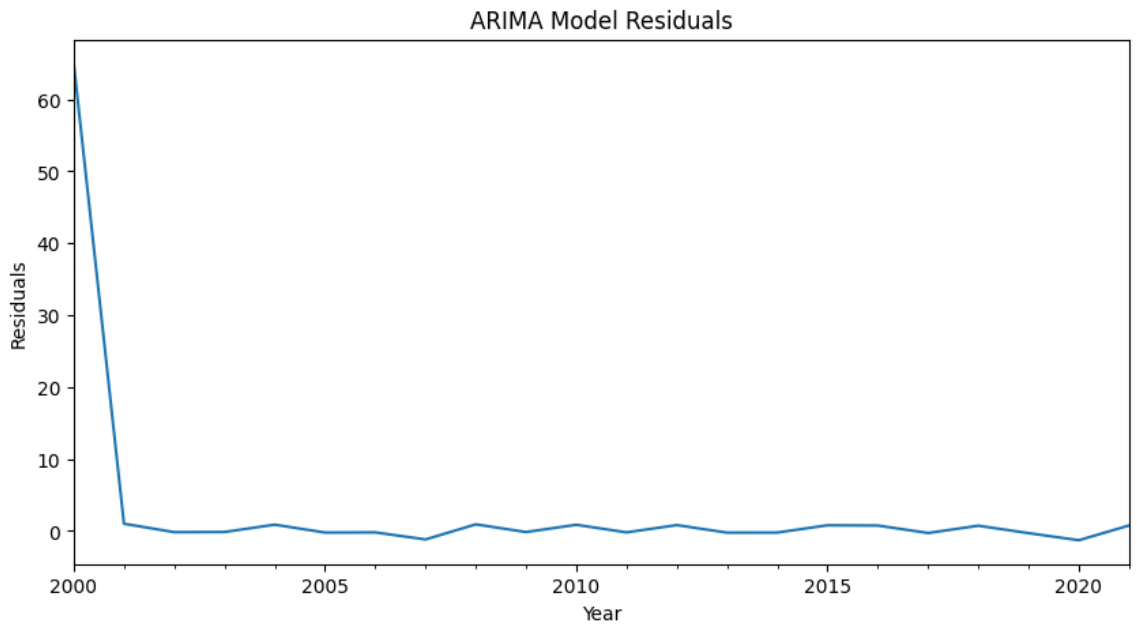
Variables Terminal

4. Build an ARIMA model and evaluate its residuals.

```
[46] model = ARIMA(life_expectancy_ts, order=(1,1,1)).fit()

     print(model.summary())
```

```
/usr/local/lib/python3.12/dist-packages/statsmodels/tsa/base/tsa_model.py:473: ValueWarning: No frequency information
  self._init_dates(dates, freq)
/usr/local/lib/python3.12/dist-packages/statsmodels/tsa/base/tsa_model.py:473: ValueWarning: No frequency information
  self._init_dates(dates, freq)
/usr/local/lib/python3.12/dist-packages/statsmodels/tsa/base/tsa_model.py:473: ValueWarning: No frequency information
  self._init_dates(dates, freq)
                               SARIMAX Results
==============================================================================
Dep. Variable:        life_expectancy   No. Observations:                   22
Model:                 ARIMA(1, 1, 1)   Log Likelihood                -21.859
Date:                Thu, 28 Aug 2025   AIC                             49.718
Time:                        16:39:50   BIC                             52.852
Sample:                  01-01-2000   HQIC                            50.398
                       - 01-01-2021
Covariance Type:                 opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1          1.0000      0.174      5.745      0.000       0.659       1.341
ma.L1         -0.9968     11.199     -0.089      0.929     -22.946      20.953
sigma2         0.4344      4.728      0.092      0.927      -8.832       9.700
===================================================================================
Ljung-Box (L1) (Q):                   2.44   Jarque-Bera (JB):                 1.05
Prob(Q):                              0.12   Prob(JB):                         0.59
Heteroskedasticity (H):               1.36   Skew:                            -0.42
Prob(H) (two-sided):                  0.70   Kurtosis:                         2.29
===================================================================================

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
/usr/local/lib/python3.12/dist-packages/statsmodels/base/model.py:607: ConvergenceWarning: Maximum Likelihood optimiza
  warnings.warn("Maximum Likelihood optimization failed to "
```

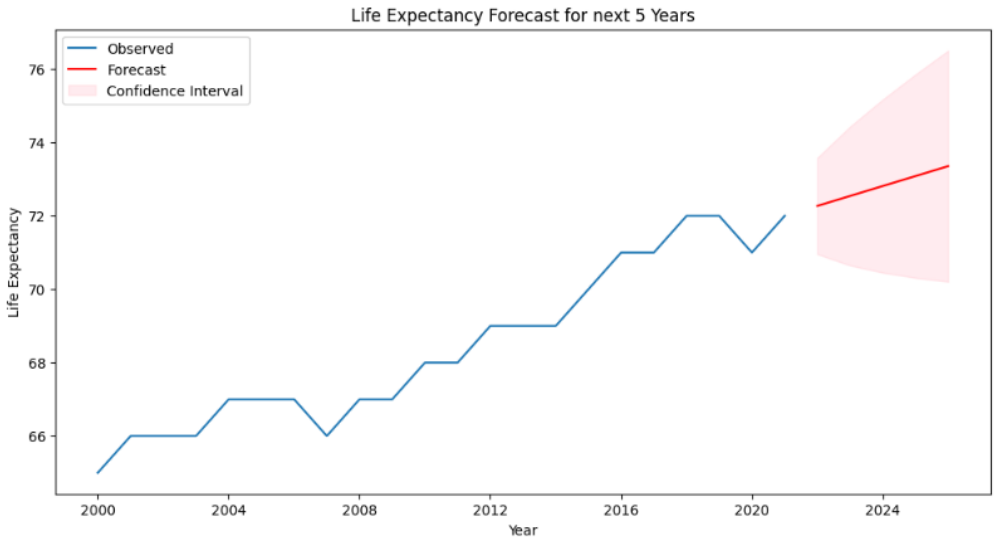
```
[47] plt.figure(figsize=(10,5))
     model.resid.plot()
     plt.title("ARIMA Model Residuals")
     plt.xlabel("Year")
     plt.ylabel("Residuals")
     plt.show()
```



5. Forecast the next 5 years of life expectancy and visualize the forecast.

```
model = ARIMA(life_expectancy_ts, order=(1,1,1)).fit()

forecast = model.get_forecast(steps=5)
forecast_mean = forecast.predicted_mean
forecast_ci = forecast.conf_int()

plt.figure(figsize=(12,6))
plt.plot(life_expectancy_ts, label="Observed")
plt.plot(forecast_mean.index, forecast_mean, label="Forecast", color="red")
plt.fill_between(forecast_ci.index,
                 forecast_ci.iloc[:,0],
                 forecast_ci.iloc[:,1],
                 color="pink", alpha=0.3, label="Confidence Interval")

plt.title("Life Expectancy Forecast for next 5 Years")
plt.xlabel("Year")
plt.ylabel("Life Expectancy")
plt.legend()
plt.show()

print("Forecasted Life Expectancy for next 5 years:")
print(forecast_mean)
```

```
/usr/local/lib/python3.12/dist-packages/statsmodels/tsa/base/tsa_model.py:473: ValueWarning: No frequency information was provided, so inferred frequency YS-JAN will be used.
  self._init_dates(dates, freq)
/usr/local/lib/python3.12/dist-packages/statsmodels/tsa/base/tsa_model.py:473: ValueWarning: No frequency information was provided, so inferred frequency YS-JAN will be used.
  self._init_dates(dates, freq)
/usr/local/lib/python3.12/dist-packages/statsmodels/tsa/base/tsa_model.py:473: ValueWarning: No frequency information was provided, so inferred frequency YS-JAN will be used.
  self._init_dates(dates, freq)
/usr/local/lib/python3.12/dist-packages/statsmodels/base/model.py:607: ConvergenceWarning: Maximum Likelihood optimization failed to converge. Check mle_retvals
  warnings.warn("Maximum Likelihood optimization failed to "
```




```
Forecasted Life Expectancy for next 5 years:
2022-01-01    72.271205
2023-01-01    72.542403
2024-01-01    72.813594
2025-01-01    73.084778
2026-01-01    73.355956
Freq: YS-JAN, Name: predicted_mean, dtype: float64
```