```
# basic libraries
import pandas as pd

df = pd.read_csv("Life Expectancy Data.csv")

df.columns = df.columns.str.strip()

# quick checks so we know it loaded correctly
print("Shape (rows, columns):", df.shape)
display(df.head(3))
print("Columns:", df.columns.tolist())
```

```
Shape (rows, columns): (2938, 22)
```

	Country	Year	Status	Life expectancy	Adult Mortality	infant deaths	Alcohol	percentage expenditure	Hepatitis B	Measles	...	Polio	Total expenditure
0	Afghanistan	2015	Developing	65.0	263.0	62	0.01	71.279624	65.0	1154	...	6.0	8.16
1	Afghanistan	2014	Developing	59.9	271.0	64	0.01	73.523582	62.0	492	...	58.0	8.18
2	Afghanistan	2013	Developing	59.9	268.0	66	0.01	73.219243	64.0	430	...	62.0	8.13

3 rows × 22 columns

```
Columns: ['Country', 'Year', 'Status', 'Life expectancy', 'Adult Mortality', 'infant deaths', 'Alcohol', 'percentage expenditure
```

```
# summary statistics for all numeric columns
print("Summary of numeric columns:")
display(df.select_dtypes(include=["number"]).describe().T)

# check how many records exist for each year
print("\nRecord count by Year:")
print(df["Year"].value_counts().sort_index())
```

	count	mean	std	min	25%	50%	75%	max
Year	2938.0	2.007519e+03	4.613841e+00	2000.00000	2004.000000	2.008000e+03	2.012000e+03	2.015000e+03
Life expectancy	2928.0	6.922493e+01	9.523867e+00	36.30000	63.100000	7.210000e+01	7.570000e+01	8.900000e+01
Adult Mortality	2928.0	1.647964e+02	1.242921e+02	1.00000	74.000000	1.440000e+02	2.280000e+02	7.230000e+02
infant deaths	2938.0	3.030395e+01	1.179265e+02	0.00000	0.000000	3.000000e+00	2.200000e+01	1.800000e+03
Alcohol	2744.0	4.602861e+00	4.052413e+00	0.01000	0.877500	3.755000e+00	7.702500e+00	1.787000e+01
percentage expenditure	2938.0	7.382513e+02	1.987915e+03	0.00000	4.685343	6.491291e+01	4.415341e+02	1.947991e+04
Hepatitis B	2385.0	8.094046e+01	2.507002e+01	1.00000	77.000000	9.200000e+01	9.700000e+01	9.900000e+01
Measles	2938.0	2.419592e+03	1.146727e+04	0.00000	0.000000	1.700000e+01	3.602500e+02	2.121830e+05

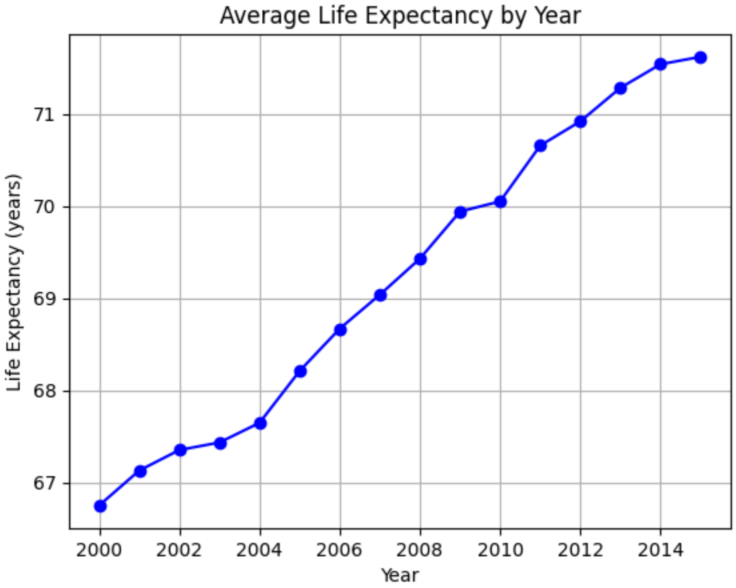
```
# Handling Missing Values

print("Missing values per column (top 10):")
print(df.isna().sum().sort_values(ascending=False).head(10))

# Fill numeric columns with their median value
num_cols = df.select_dtypes(include="number").columns
df[num_cols] = df[num_cols].fillna(df[num_cols].median())

cat_cols = df.select_dtypes(exclude="number").columns
for c in cat_cols:
    df[c] = df[c].fillna(df[c].mode()[0])


print("\nTotal missing values left:", df.isna().sum().sum())
```

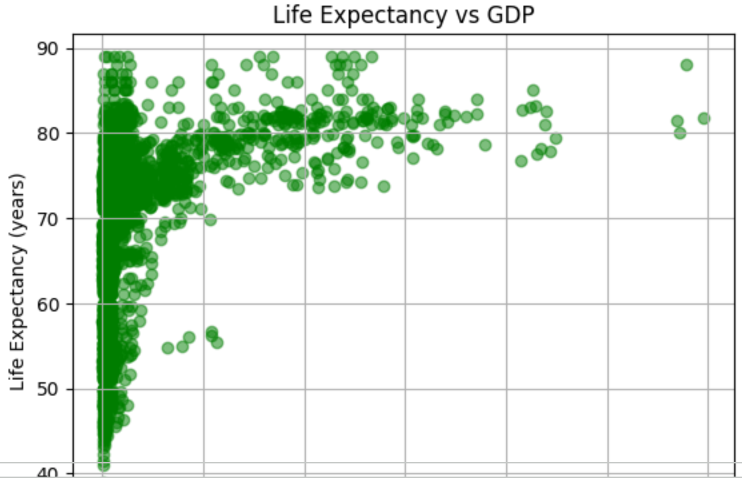
Income composition of resources	2771.0	6.275511e-01	2.109036e-01	0.00000	0.493000	6.770000e-01	7.790000e-01	9.480000e-01
Schooling	2775.0	1.199279e+01	3.358920e+00	0.00000	10.100000	1.230000e+01	1.430000e+01	2.070000e+01

```
Missing values per column (top 10):
Population                         652
Hepatitis B                        553
GDP                                448
Total expenditure                  226
Alcohol                            194
Income composition of resources    167
Schooling                          163
thinness  1-19 years                34
thinness 5-9 years                  34
BMI                                 34
dtype: int64

Total missing values left: 0
```

```
Record count by Year:
Year
2000    183
2001    183
2002    183
2003    183
2004    183
2005    183
2006    183
2007    183
2008    183
```

```
import matplotlib.pyplot as plt
# Plot 1 Average Life Expectancy by Year
if "Year" in df.columns and "Life expectancy" in df.columns:
    avg_by_year = df.groupby("Year")["Life expectancy"].mean()
    avg_by_year.plot(kind="line", marker='o', color='blue')
    plt.title("Average Life Expectancy by Year")
    plt.xlabel("Year")
    plt.ylabel("Life Expectancy (years)")
    plt.grid(True)
    plt.show()
# Plot 2 GDP vs Life Expectancy
if "GDP" in df.columns and "Life expectancy" in df.columns:
    plt.scatter(df["GDP"], df["Life expectancy"], alpha=0.5, color='green')
    plt.title("Life Expectancy vs GDP")
    plt.xlabel("GDP")
    plt.ylabel("Life Expectancy (years)")
    plt.grid(True)
    plt.show()

    print("These visuals help confirm that economic and social factors influence how long people live.")
```





```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error, r2_score

# target column
target = "Life expectancy"

# choose some easy numeric columns that exist in your dataset
candidate_features = ["Adult Mortality", "Alcohol", "BMI", "GDP", "Schooling", "infant deaths"]

# only keep the ones that actually exist
features = [c for c in candidate_features if c in df.columns]
print("Features used:", features)

# split into inputs (X) and output (y)
X = df[features].copy()
y = df[target].copy()

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

model = LinearRegression()
model.fit(X_train, y_train)

y_pred = model.predict(X_test)

mae = mean_absolute_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print("\n--- Model Results ---")
print("Mean Absolute Error (years):", round(mae, 2))
print("R^2 Score:", round(r2, 3))
```

```
Features used: ['Adult Mortality', 'Alcohol', 'BMI', 'GDP', 'Schooling', 'infant deaths']

--- Model Results ---
Mean Absolute Error (years): 3.34
R^2 Score: 0.731
```