Data Governance and Security project using python

Introduction:

These Days, with data being produced, organizations obtain large volumes of data for better service. Without it, the data cannot be effectively managed and thus cannot be turned into value. Data governance is designed to create a set of rules and process to ensure that data can be trusted, secure, easily found and ready to use.

Data governance centres on determining who is responsible for data, the cleanliness of that data, preventing unauthorized access and maintaining compliance with laws and regulations to name a few. With a structured data governance strategy in place, organizations can then minimize the risks associated with poor data management practices leading to less trust in their data and use it intelligently as a strategic asset for competitive advantage.

Task:

Python's readability, ease of use, and large library ecosystem are what make it so popular in the data science and analytics community. In terms of data quality management, Python provides a variety of tools and libraries that make it easier to clean, validate, and enrich data.

1) Data Preparation:
   - Country Code: Represents the country (e.g., "USA" for the United States, "IND" for India).
   - Series Code: Represents different ESG indicators (e.g., CO2 emissions, renewable energy usage, social welfare index).
   - Description: A brief explanation of each Series Code.

2) Objectives:
   - Visualize and compare ESG indicators across different countries.
   - Allow users to filter by specific ESG categories (Environment, Social, Governance).
   - Enable users to view detailed descriptions of each Series Code.

3) Steps to Complete the Task:

   - Load the Data into python
   - Import the required libraries like numpy,pandas,matplotlib,seaborn
   - Display the dataframes using pandas
   - Do data manipulations
   - Make updates and do transpose the data
   - Take different counties data with ESG indicators from updated data (Sweden,finland,Norway)
   - Display a bar chart and line chart to comparing the selected ESG indicator across multiple countries.

PROJECT CODE

```
import numpy as np

import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

ESGData = pd.read_csv('C:/Users/rocki/OneDrive/Documents/ESGData.csv')

ESGData.head()
```

	Country Name	Country Code	Indicator Name	Indicator Code	1960	1961	1962	1963	1964	1965	...	2012	2013	2014	2015
0	Arab World	ARB	Access to clean fuels and technologies for coo...	EG.CFT.ACCS.ZS	NaN	NaN	NaN	NaN	NaN	NaN	...	83.120303	83.533457	83.897596	84.171599
1	Arab World	ARB	Access to electricity (% of population)	EG.ELC.ACCS.ZS	NaN	NaN	NaN	NaN	NaN	NaN	...	87.512260	88.129881	87.275323	88.720097
2	Arab World	ARB	Adjusted savings: natural resources depletion ...	NY.ADJ.DRES.GN.ZS	NaN	NaN	NaN	NaN	NaN	NaN	...	12.850205	11.641062	10.437876	6.277652
3	Arab World	ARB	Adjusted savings: net forest depletion (% of GNI)	NY.ADJ.DFOR.GN.ZS	NaN	NaN	NaN	NaN	NaN	NaN	...	0.061942	0.055593	0.079402	0.086177
4	Arab World	ARB	Agricultural land (% of land area)	AG.LND.AGRI.ZS	NaN	27.835643	27.826564	27.845522	27.847925	27.866972	...	36.472300	36.534503	36.607475	36.624759

```
col = ESGData.columns

print(col)
```

```
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', '2050'],
      dtype='object')
```

```
print(ESGData.loc[32])
```

```
Country Name                                              Arab World
Country Code                                                     ARB
Indicator Name    Labor force participation rate, total (% of to...
Indicator Code                                          SL.TLF.ACTI.ZS
1960                                                           NaN
                                          ...
2017                                                     50.637404
2018                                                     50.310959
2019                                                      50.39837
2020                                                           NaN
2050                                                           NaN
Name: 32, Length: 66, dtype: object
```

```
list = ['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',
'2019', '2020', '2050']

updated = ESGData.drop(list, axis = 1)
```

updated.head()

	Country Name	Country Code	Indicator Name	Indicator Code	1990	1991	1992	1993	1994	1995	...	200
0	Arab World	ARB	Access to clean fuels and technologies for coo...	EG.CFT.ACCS.ZS	NaN	NaN	NaN	NaN	NaN	NaN	...	81.80963
1	Arab World	ARB	Access to electricity (% of population)	EG.ELC.ACCS.ZS	NaN	NaN	NaN	NaN	NaN	NaN	...	84.2709
2	Arab World	ARB	Adjusted savings: natural resources depletion ...	NY.ADJ.DRES.GN.ZS	8.096032	8.368431	8.451037	8.247409	7.712396	8.064746	...	8.7213
			Adjusted savings: net									

new = updated.columns

new

```
Index(['Country Name', 'Country Code', 'Indicator Name', 'Indicator Code',
       '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'],
      dtype='object')
```

newlist = ['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', '2019', '2020', '2050']

updated = ESGData.drop(newlist, axis = 1)

updated = updated.transpose()

updated

	0	1	2	3	4	5	6	7	8
1990	NaN	NaN	8.096032	0.052709	31.534601	10.198358	NaN	NaN	NaN
1991	NaN	NaN	8.368431	0.087444	31.639377	10.910375	NaN	NaN	NaN
1992	NaN	NaN	8.451037	0.070674	31.741282	10.946288	NaN	NaN	NaN
1993	NaN	NaN	8.247409	0.054602	32.458657	10.524052	NaN	NaN	NaN
1994	NaN	NaN	7.712396	0.058116	33.133486	10.940366	NaN	NaN	NaN
1995	NaN	NaN	8.064746	0.080551	33.742542	9.844131	NaN	NaN	NaN
1996	NaN	NaN	9.571448	0.075840	34.385549	10.129188	NaN	NaN	NaN
1997	NaN	77.297851	8.379450	0.072735	34.978268	8.970577	NaN	NaN	NaN
1998	NaN	78.302779	5.703252	0.101829	35.711280	9.505661	NaN	NaN	NaN
1999	NaN	78.980861	7.278630	0.049495	36.329581	8.877752	NaN	NaN	NaN
2000	73.702495	78.479364	10.316367	0.031326	36.310596	7.453811	NaN	NaN	32.034514

sweden = updated[[14214, 14219, 14223, 14237, 14251]].reset_index(drop=True)

sweden = sweden.rename({14214: 'CO2 emissions (metric tons per capita)', 14219: 'Electricity production from coal sources (% of total)', 14223: 'Fertility rate, total (births per woman)', 14237: 'Life expectancy at birth, total (years)', 14251: 'Population ages 65 and above (% of total population)'}, axis=1)

print(sweden)

	CO2 emissions (metric tons per capita)	Electricity production from coal sources (% of total)	Fertility rate, total (births per woman)	Life expectancy at birth, total (years)	Population ages 65 and above (% of total population)
0	6.069368	1.050115	2.13	77.536829	17.823737
1	5.979201	1.667687	2.11	77.666829	17.823844
2	5.886699	1.753508	2.09	77.998780	17.780380
3	5.936305	1.912604	1.99	78.060488	17.702340
4	6.251746	2.203924	1.88	78.650244	17.606386
5	6.248525	1.982562	1.73	78.740488	17.508414
6	6.324031	3.066268	1.60	78.959024	17.464292
7	5.895099	1.838398	1.52	79.197561	17.427124
8	5.985460	1.883608	1.50	79.339024	17.393805
9	5.769667	2.028572	1.50	79.441463	17.353642
10	5.562430	1.697985	1.54	79.643902	17.303685

finland = updated[[7045, 7050, 7054, 7068, 7082]].reset_index(drop=True)

finland = finland.rename({7045: 'CO2 emissions (metric tons per capita)', 7050: 'Electricity production from coal sources (% of total)', 7054: 'Fertility rate, total (births per woman)', 7068: 'Life expectancy at birth, total (years)', 7082: 'Population ages 65 and above (% of total population)'}, axis=1)

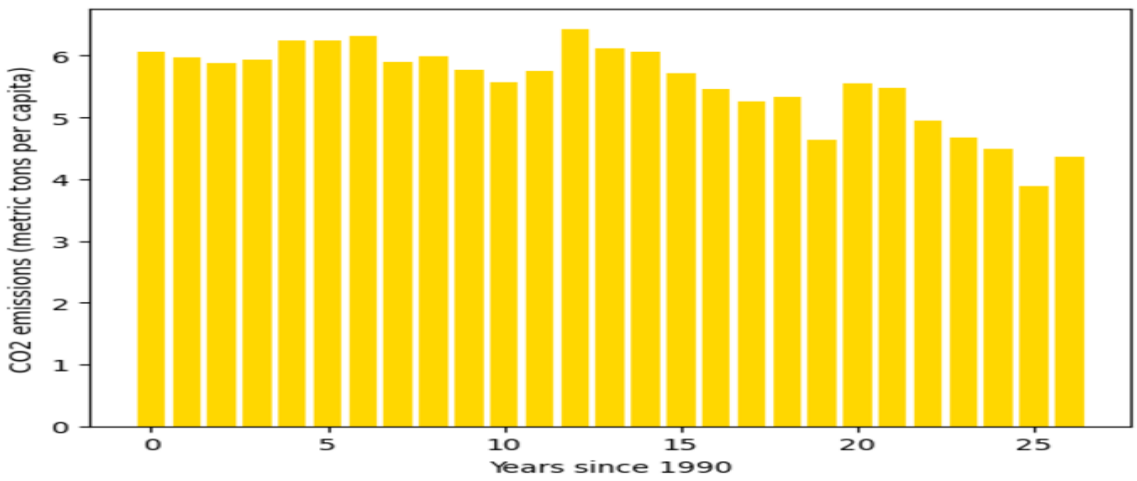
print(finland)

	CO2 emissions (metric tons per capita)	Electricity production from coal sources (% of total)	Fertility rate, total (births per woman)	Life expectancy at birth, total (years)	Population ages 65 and above (% of total population)
0	10.377169	18.460011	1.78	74.813171	13.435412
1	10.710477	17.624944	1.79	75.227561	13.641242
2	9.415521	14.108272	1.85	75.455366	13.828828
3	9.948375	17.117176	1.81	75.705122	13.998177
4	11.246028	22.265393	1.85	76.395610	14.148544
5	10.320143	18.194737	1.81	76.409512	14.281193
6	11.958636	22.360284	1.78	76.693415	14.435508
7	11.660578	19.039840	1.75	76.878537	14.581076
8	11.058284	12.249348	1.70	77.090732	14.720960
9	10.729516	13.851736	1.73	77.291220	14.857168
10	10.083843	13.107351	1.73	77.465854	14.993926

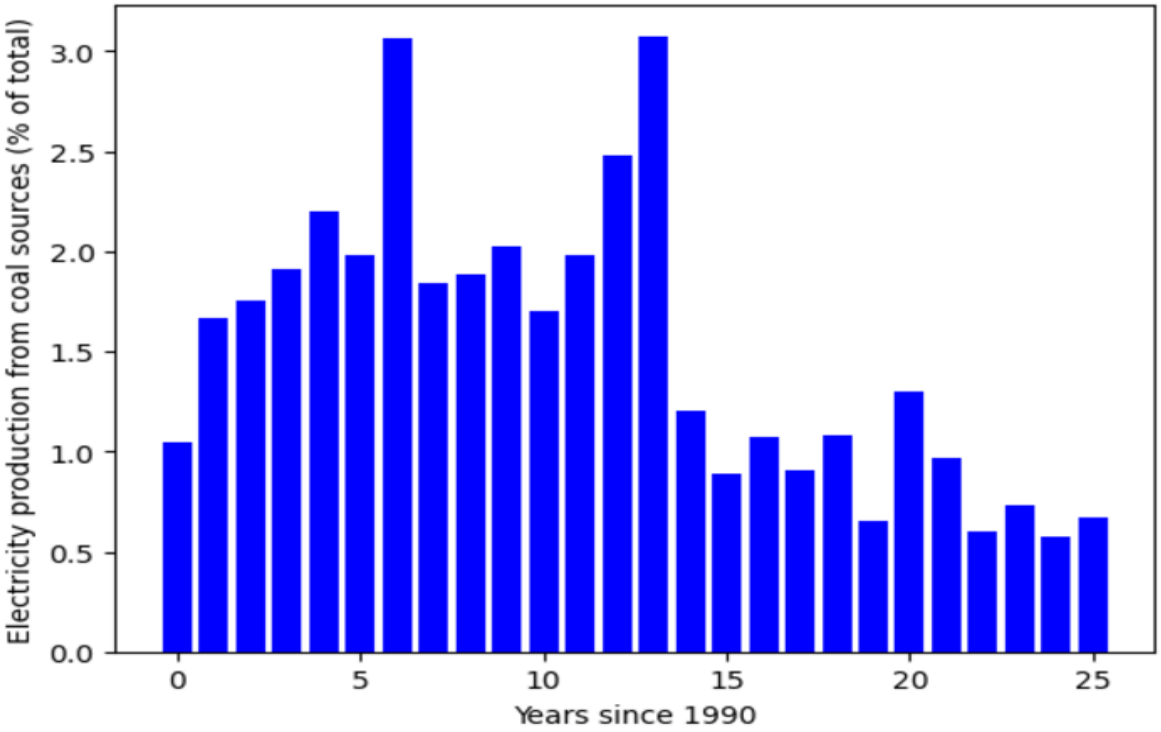
```
norway = updated[[11735, 11740, 11744, 11758, 11772]].reset_index(drop=True)

norway = norway.rename({11735: 'CO2 emissions (metric tons per capita)', 11740: 'Electricity production from coal sources (% of total)', 11744: 'Fertility rate, total (births per woman)', 11758: 'Life expectancy at birth, total (years)', 11772: 'Population ages 65 and above (% of total population)'}, axis=1)

print(norway)
```

	CO2 emissions (metric tons per capita)	Electricity production from coal sources (% of total)	Fertility rate, total (births per woman)	Life expectancy at birth, total (years)	Population ages 65 and above (% of total population)
0	7.429148	0.067428	1.93	76.537317	16.355954
1	7.526341	0.078688	1.92	76.980732	16.371021
2	7.467627	0.080265	1.88	77.184390	16.335332
3	8.151268	0.074341	1.86	77.151707	16.256045
4	7.818332	0.088139	1.87	77.689756	16.144269
5	7.671017	0.069534	1.87	77.736585	16.009476
6	7.615493	0.086917	1.89	78.150488	15.889342
7	8.195307	0.072402	1.86	78.142683	15.746397
8	8.524820	0.067851	1.81	78.329268	15.588065
9	9.117546	0.067873	1.85	78.282927	15.422114
10	8.834016	0.050522	1.85	78.634146	15.256435

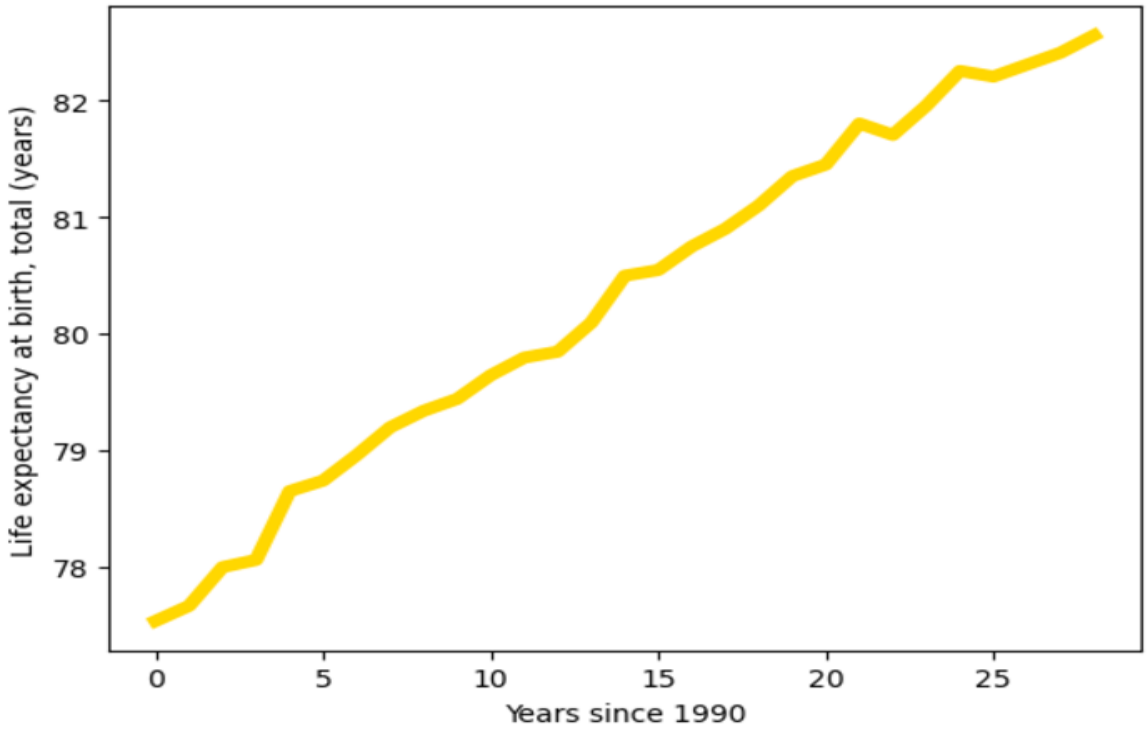
```
plt.bar(sweden.index, sweden['CO2 emissions (metric tons per capita)'],color = 'gold')

plt.ylabel('CO2 emissions (metric tons per capita)')

plt.xlabel('Years since 1990')

plt.show()
```



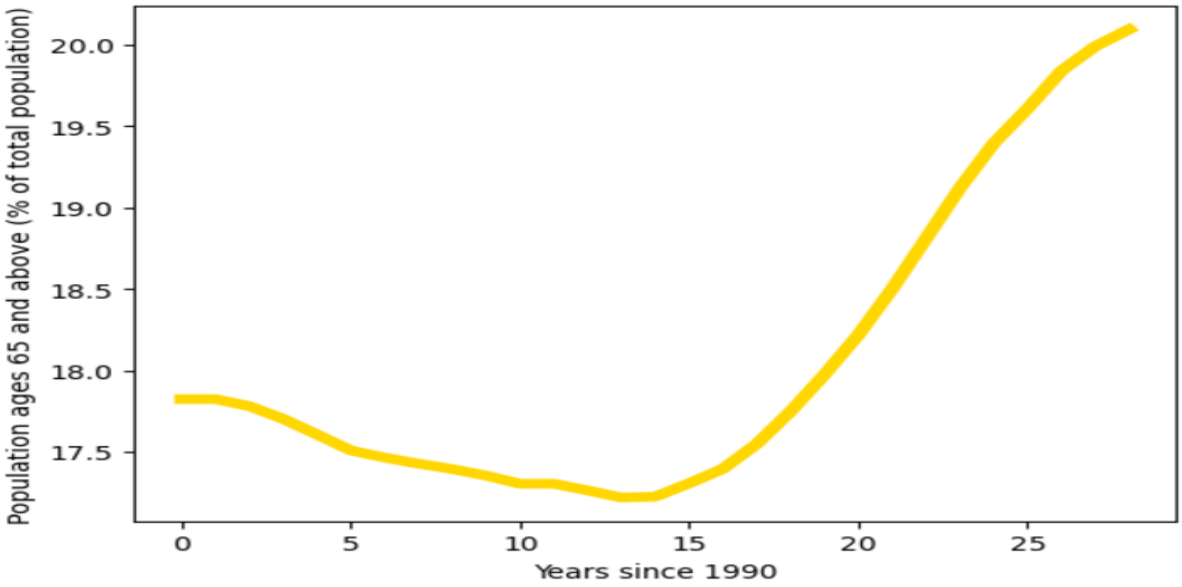
```
plt.bar(sweden.index, sweden['Electricity production from coal sources (% of total)'], color = 'b')

plt.ylabel('Electricity production from coal sources (% of total)')

plt.xlabel('Years since 1990')

plt.show()
```



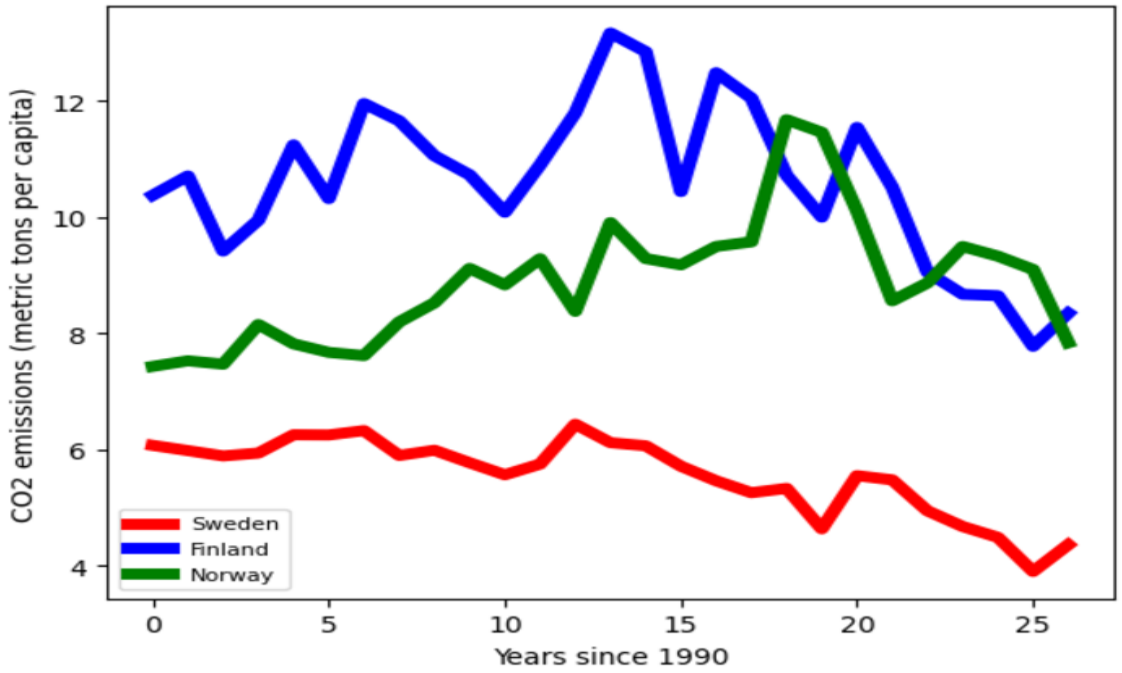

```
plt.plot(sweden.index, sweden['Life expectancy at birth, total (years)'], color = 'gold', linewidth = 5)

plt.ylabel('Life expectancy at birth, total (years)')

plt.xlabel('Years since 1990')

plt.show()
```



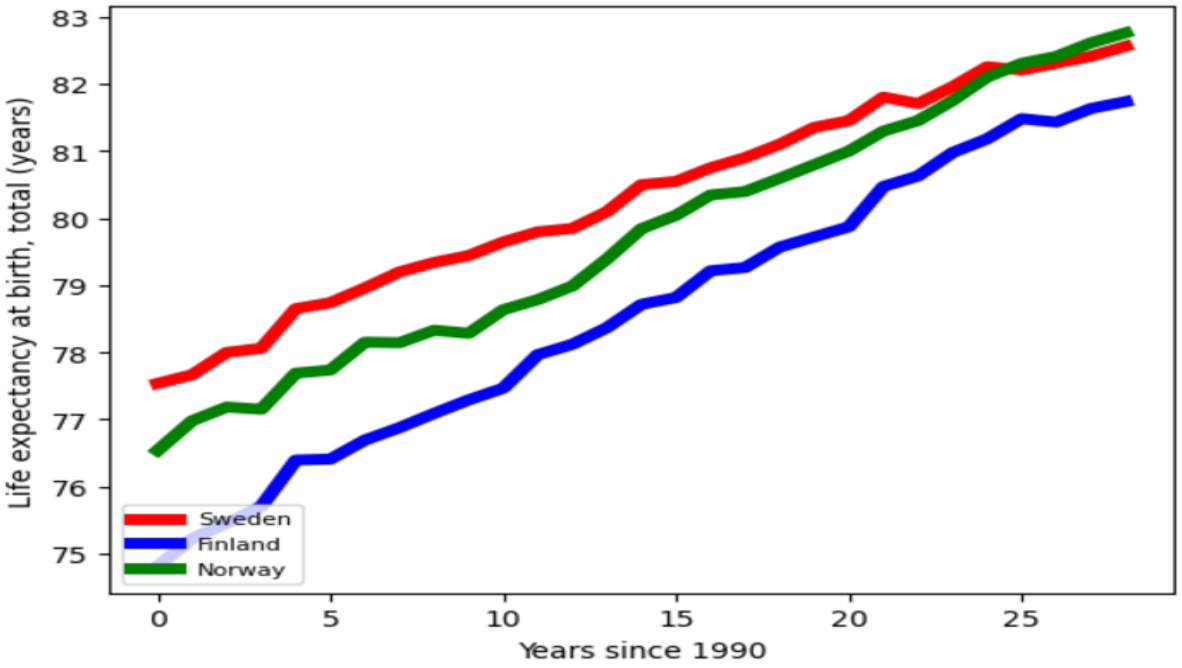

```
plt.plot(sweden.index, sweden['Population ages 65 and above (% of total population)'], color = 'gold', linewidth = 5)

plt.ylabel('Population ages 65 and above (% of total population)')

plt.xlabel('Years since 1990')

plt.show()
```



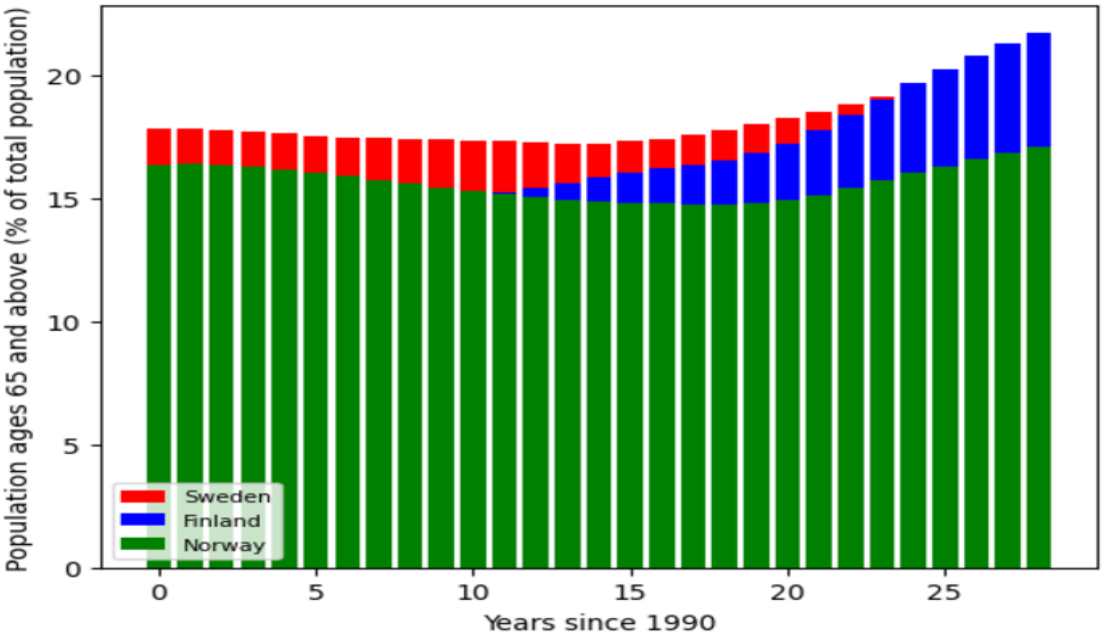
```
plt.plot(sweden.index, sweden['CO2 emissions (metric tons per capita)'], color = 'red', linewidth = 5, label = "Sweden")

plt.plot(finland.index, finland['CO2 emissions (metric tons per capita)'], color = 'blue', linewidth = 5, label = "Finland")

plt.plot(norway.index, norway['CO2 emissions (metric tons per capita)'], color = 'green', linewidth = 5, label = "Norway")

plt.ylabel('CO2 emissions (metric tons per capita)')

plt.xlabel('Years since 1990')

plt.legend(loc="lower left", prop={'size': 8})

plt.show()
```



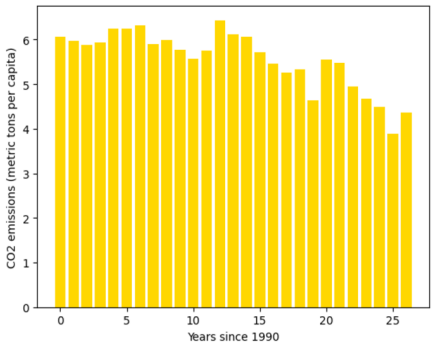

```
plt.plot(sweden.index, sweden['Life expectancy at birth, total (years)'], color = 'red', linewidth = 5, label = "Sweden")

plt.plot(finland.index, finland['Life expectancy at birth, total (years)'], color = 'blue', linewidth = 5, label = "Finland")

plt.plot(norway.index, norway['Life expectancy at birth, total (years)'], color = 'green', linewidth = 5, label = "Norway")

plt.ylabel('Life expectancy at birth, total (years)')

plt.xlabel('Years since 1990')

plt.legend(loc="lower left", prop={'size': 8})

plt.show()
```

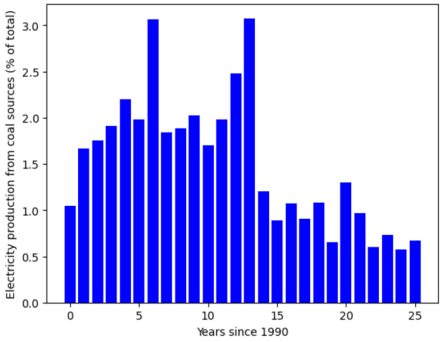


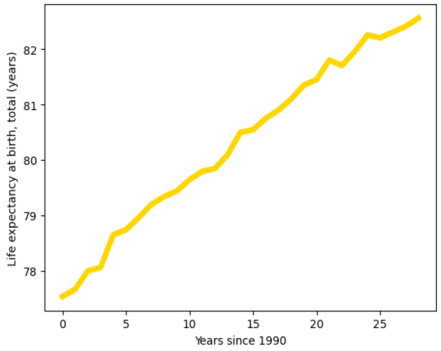

```
plt.bar(sweden.index, sweden['Population ages 65 and above (% of total population)'], color = 'red', linewidth = 5, label = "Sweden")

plt.bar(finland.index, finland['Population ages 65 and above (% of total population)'], color = 'blue', linewidth = 5, label = "Finland")

plt.bar(norway.index, norway['Population ages 65 and above (% of total population)'], color = 'green', linewidth = 5, label = "Norway")

plt.ylabel('Population ages 65 and above (% of total population)')

plt.xlabel('Years since 1990')

plt.legend(loc="lower left", prop={'size': 8})

plt.show()
```

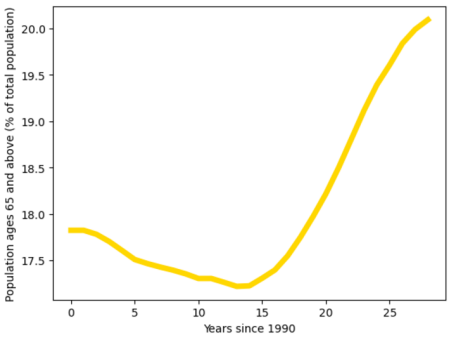


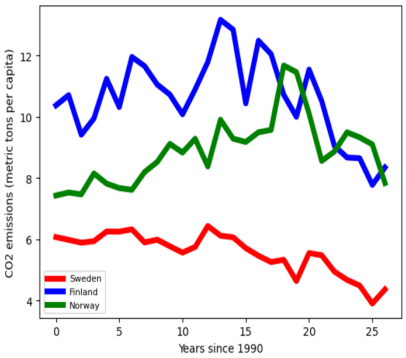
DASHBOARD

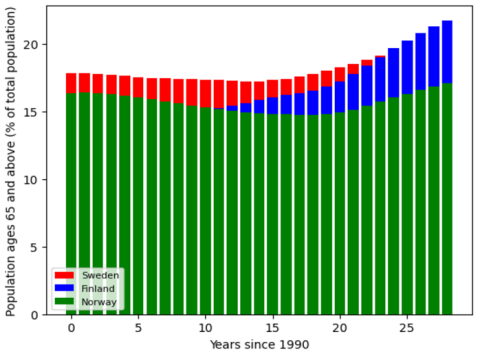

CO2 emissions (metric tons per capita)
Years since 1990


Electricity production from coal sources (% of total)
Years since 1990


Life expectancy at birth, total (years)
Years since 1990


Population ages 65 and above (% of total population)
Years since 1990


CO2 emissions (metric tons per capita)
Sweden
Finland
Norway
Years since 1990


Population ages 65 and above (% of total population)
Sweden
Finland
Norway
Years since 1990

CONCLUSION

The Data Governance and Security project using Python successfully demonstrates how data can be transformed into actionable insights through proper governance and visualization techniques. Here are the key takeaways:

Data Governance: The project effectively highlights the importance of managing and cleaning data for accurate analysis. By removing irrelevant columns and transposing datasets, the data is prepared for meaningful comparisons.

Visualization of ESG Indicators: Using Python's powerful libraries like Matplotlib and Seaborn, the project provides clear visual representations of key ESG indicators (e.g., CO2 emissions, life expectancy) for Sweden, Finland, and Norway. Bar charts and line charts are used to present trends and comparisons, making it easier for users to understand differences and patterns across countries.

Strategic Insights: The project allows for the comparison of environmental, social, and governance metrics, offering insights that can guide policies or strategies in sustainability and governance. For instance, users can observe trends in CO2 emissions reduction or increases in life expectancy over time.

Reproducibility and Scalability: The modular approach of the code ensures that it can be easily adapted for analyzing other countries or indicators. By leveraging Python's libraries, the project is scalable and can integrate additional ESG dimensions or datasets with minimal adjustments.

User-Friendly Features: The ability to filter and focus on specific ESG categories adds flexibility, making the tool suitable for various stakeholders, from analysts to policymakers.

The project underscores how data governance principles, combined with Python's data analysis capabilities, can unlock the value of data. By ensuring that the data is clean, secure, and structured, this project provides a solid framework for leveraging ESG data as a strategic asset for competitive advantage and informed decision-making.