#!/usr/bin/env python3

# -\*- coding: utf-8 -\*-

"""

Created on Tue Dec 12 20:25:44 2023

@author: kamalib

"""

#import important libraries

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

def read\_data(filename):

# Read the data from the file into a DataFrame, skipping unnecessary initial columns

data = pd.read\_csv(filename, skiprows=3)

# Drop unnecessary columns: Country Code, Indicator Name, Indicator Code

data.drop(['Country Code', 'Indicator Name', 'Indicator Code'], axis=1, inplace=True)

# Set 'Country Name' as the index

data.set\_index('Country Name', inplace=True)

# Transpose the DataFrame to have years as columns and countries as rows

transposed\_data = data.T

# Create two separate dataframes: one with years as columns and one with countries as columns

years\_df = transposed\_data.copy()

countries\_df = transposed\_data.transpose().copy()

return years\_df, countries\_df

# Read Urban Population

# Filename stores the path of the file to be read

filename = '//Users/kamalib/Downloads/Datasets/API\_SP.URB.TOTL.IN.ZS\_DS2\_en\_csv\_v2\_5996759.csv'

#Call the read\_data function for specific file

urban\_years\_df, urban\_countries\_df = read\_data(filename)

# Printing the first few rows of the dataframes

print("Years DataFrame:")

urban\_years\_df.head()

print("\nCountries DataFrame:")

urban\_countries\_df.head()

#Read Green House Emission

# Filename stores the path of the file to be read

filename = '//Users/kamalib/Downloads/Datasets/API\_EN.ATM.GHGT.KT.CE\_DS2\_en\_csv\_v2\_5995567.csv'

#Call the read\_data function for specific file

greenhouse\_years\_df, greenhouse\_countries\_df = read\_data(filename)

#Read Arable Land(%)

# Filename stores the path of the file to be read

filename = '//Users/kamalib/Downloads/Datasets/API\_AG.LND.ARBL.ZS\_DS2\_en\_csv\_v2\_5995308.csv'

#Call the read\_data function for specific file

arable\_land\_years\_df, arable\_land\_countries\_df = read\_data(filename)

#Forest Land(%)

# Filename stores the path of the file to be read

filename = '//Users/kamalib/Downloads/Datasets/API\_AG.LND.FRST.ZS\_DS2\_en\_csv\_v2\_5994693.csv'

#Call the read\_data function for specific file

forest\_land\_years\_df, forest\_land\_countries\_df = read\_data(filename)

#Agriculture Land(%)

# Filename stores the path of the file to be read

filename = '//Users/kamalib/Downloads/Datasets/API\_AG.LND.AGRI.ZS\_DS2\_en\_csv\_v2\_5995314.csv'

#Call the read\_data function for specific file

agr\_land\_years\_df, agr\_land\_countries\_df = read\_data(filename)

#Access to Electricity(%)

# Filename stores the path of the file to be read

filename = '//Users/kamalib/Downloads/Datasets/API\_EG.ELC.ACCS.ZS\_DS2\_en\_csv\_v2\_5995100.csv'

#Call the read\_data function for specific file

elc\_acc\_years\_df, elc\_acc\_countries\_df = read\_data(filename)

#Population Growth(%)

# Filename stores the path of the file to be read

filename = '//Users/kamalib/Downloads/Datasets/API\_SP.POP.GROW\_DS2\_en\_csv\_v2\_5995052.csv'

#Call the read\_data function for specific file

pop\_years\_df, pop\_countries\_df = read\_data(filename)

#GDP(%)

# Filename stores the path of the file to be read

filename = '//Users/kamalib/Downloads/Datasets/API\_NV.AGR.TOTL.ZS\_DS2\_en\_csv\_v2\_5995988.csv'

#Call the read\_data function for specific file

gdp\_years\_df, gdp\_countries\_df = read\_data(filename)

#electricity Consumption(kwh)

# Filename stores the path of the file to be read

filename = '//Users/kamalib/Downloads/Datasets/API\_EG.USE.ELEC.KH.PC\_DS2\_en\_csv\_v2\_5995551.csv'

#Call the read\_data function for specific file

elc\_use\_years\_df, elc\_use\_countries\_df = read\_data(filename)

#Renewable electricity output(%)

# Filename stores the path of the file to be read

filename = '//Users/kamalib/Downloads/Datasets/API\_EG.ELC.RNEW.ZS\_DS2\_en\_csv\_v2\_5995544.csv'

#Call the read\_data function for specific file

rnw\_elc\_years\_df, rnw\_elc\_countries\_df = read\_data(filename)

#Renewable energy consumption(%)

# Filename stores the path of the file to be read

filename = '//Users/kamalib/Downloads/Datasets/API\_EG.ELC.RNEW.ZS\_DS2\_en\_csv\_v2\_5995544.csv'

#Call the read\_data function for specific file

rnw\_cnsmp\_years\_df, rnw\_cnsmp\_countries\_df = read\_data(filename)

#bargraph

def plot\_filtered\_data(countries\_df, selected\_countries, years, title):

# Sort the rows (countries) alphabetically

selected\_countries.sort()

# Filtering the data for the selected countries and years

filtered\_data = countries\_df[selected\_countries].loc[years]

# Plotting a bar graph

filtered\_data.T.plot(kind='bar', figsize=(10, 6)) # Transpose for correct plotting orientation

plt.title(title)

plt.xlabel('Country Name')

plt.legend(title='Year')

plt.xticks(rotation=45)

plt.tight\_layout()

plt.show()

# Selecting specified countries containing the greenhouse gas emissions data

selected\_countries = ['China', 'India', 'Japan', 'Indonesia', 'Bangladesh', 'Afghanistan', 'Philippines', 'Thailand', 'Pakistan', 'Malaysia']

# Five-year increments

years = ['1990', '1995', '2000', '2005', '2010', '2015', '2020']

plot\_filtered\_data(greenhouse\_years\_df, selected\_countries, years,'Greenhouse Gas Emissions (kt of CO2 equivalent)')

#display data in tabular format for the countries with their indicator values

def filter\_urban\_data(data, selected\_countries, years):

# Filtering the data for the selected countries and years

filtered\_data = data[selected\_countries].loc[years]

# Displaying the filtered data in tabular format

print(filtered\_data.T)

#Call the function to display urban population% for countries over the year

urban\_years = ['1980', '2000','2020']

filter\_urban\_data(urban\_years\_df, selected\_countries, urban\_years)

def plot\_arable\_land(data, selected\_countries, years, title):

if 'Country Name' in data.index:

# Filtering the data for the selected countries and years

filtered\_data = data.loc[selected\_countries, years]

else:

# Filtering the data for the selected countries and years

filtered\_data = data[selected\_countries]

filtered\_data = filtered\_data.loc[years]

# Plotting a line graph with dotted lines

plt.figure(figsize=(10, 6))

for country in selected\_countries:

plt.plot(filtered\_data.index.astype(int), filtered\_data.loc[:, country], linestyle='--', label=country)

plt.title(title)

plt.xlabel('Year')

plt.grid(True)

# Placing the legend outside the graph

plt.legend(title='Country', bbox\_to\_anchor=(1.05, 1), loc='upper left')

plt.tight\_layout()

plt.show()

elc\_years = ['2000','2002', '2004', '2006', '2008', '2010','2012', '2014', '2016', '2018', '2020']

plot\_arable\_land(elc\_acc\_years\_df, selected\_countries, elc\_years, 'Access to electricity Over Years')

rnw\_year = ['1990','2000', '2010','2015']

plot\_filtered\_data(rnw\_elc\_years\_df, selected\_countries, rnw\_year, 'Renewable electricity output (% of total electricity output)')

year = ['1960', '1965', '1970', '1975', '1980', '1985', '1990', '1995', '2000', '2005', '2010', '2015', '2020'] # Years of interest

plot\_arable\_land(arable\_land\_years\_df, selected\_countries, year, 'Arable Land Over Years')

plot\_arable\_land(forest\_land\_years\_df, selected\_countries, year, 'Forest Land Over Years')

plot\_arable\_land(agr\_land\_years\_df, selected\_countries, year, 'Agriculture Land Over Years')

def combine\_indicator\_data(\*indicator\_dfs):

# Assign indicator names to the DataFrames

indicator\_names = ['Arable Land', 'Forest Land', 'Agriculture Land', 'Urban', 'Populution Growth', 'Access to Electricity', 'Electrcity Consumption', 'Renewabale Electricity', 'Renewable Consumption', 'Greenhouse Emission', 'GDP']

named\_indicator\_dfs = zip(indicator\_names, indicator\_dfs)

# Create a list to hold modified DataFrames with indicator names

modified\_dfs = []

# Add indicator names to the DataFrames and append them to modified\_dfs list

for name, df in named\_indicator\_dfs:

df['Indicator Name'] = name

modified\_dfs.append(df)

# Merge the modified DataFrames on 'Country Name' index

combined\_data = pd.concat(modified\_dfs)

# Removing null values

combined\_data.dropna(axis=1, how='all', inplace=True)

combined\_data.dropna(axis=0, how='all', inplace=True)

return combined\_data

# Pass your indicator DataFrames as arguments below

combined\_data = combine\_indicator\_data(arable\_land\_countries\_df, forest\_land\_countries\_df, agr\_land\_countries\_df, urban\_countries\_df, pop\_countries\_df, elc\_acc\_countries\_df, elc\_use\_countries\_df, rnw\_elc\_countries\_df, rnw\_cnsmp\_countries\_df, greenhouse\_countries\_df, gdp\_countries\_df)

print(combined\_data)

def extract\_country\_data(country\_name, combined\_data):

country\_data = combined\_data[combined\_data.index == country\_name]

# Dropping unnecessary columns (0th columns)

country\_data.drop(country\_data.columns[0], axis=1, inplace=True)

# Set 'Indicator Name' as index and drop unnecessary columns

country\_data.set\_index('Indicator Name', inplace=True)

# Removing null values

country\_data.dropna(axis=1, how='all', inplace=True)

country\_data.dropna(axis=0, how='all', inplace=True)

return country\_data

# Extracting data for 'China'

china\_data = extract\_country\_data('China', combined\_data)

def calculate\_correlation\_heatmap(country\_data,name):

# Transpose the data to calculate correlation between indicators

transposed\_data = country\_data.T

# Calculate the correlation matrix between indicators

correlation\_matrix = transposed\_data.corr()

# Create a heatmap for correlation between indicators

plt.figure(figsize=(10, 8))

sns.heatmap(correlation\_matrix, annot=True, cmap='coolwarm')

plt.title('Correlation Heatmap for '+ name)

plt.show()

# Correlation function for 'China'

calculate\_correlation\_heatmap(china\_data, "China")

# Extracting data for 'India'

India\_data = extract\_country\_data('India', combined\_data)

# Correlation function for 'India'

calculate\_correlation\_heatmap(India\_data, "India")

# Extracting data for 'Afghanistan'

afg\_data = extract\_country\_data('Afghanistan', combined\_data)

# Correlation function for 'Afghanistan'

calculate\_correlation\_heatmap(afg\_data,"Afghanistan")

china\_data.describe()

India\_data.describe()

# Selecting specified countries containing the greenhouse gas emissions data

selected\_countries = ['Europe & Central Asia', 'South Asia', 'North America','Middle East & North Africa', 'South Africa', 'Arab World']

# Five-year increments

years = ['1990', '1995', '2000', '2005', '2010', '2015', '2020']

plot\_filtered\_data(greenhouse\_years\_df, selected\_countries, years,'Greenhouse Gas Emissions (kt of CO2 equivalent)')

# Selecting specified countries containing the greenhouse gas emissions data

selected\_countries = ['Europe & Central Asia', 'South Asia', 'North America','Middle East & North Africa', 'South Africa', 'Arab World']

# Five-year increments

years = ['1990', '1995', '2000', '2005', '2010', '2015', '2020']

plot\_filtered\_data(rnw\_elc\_years\_df, selected\_countries, years,'Renewable Electricity (%)')