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

"""

Created on Tue Dec 01 11:45:58 2023

@author: aishw

"""

import pandas as pd

import matplotlib.pyplot as plt

import numpy as np

import seaborn as sns

def read\_data(filename):

'''

Parameters

----------

filename : input filename(.csv) fo read the data.

Returns

-------

data : data with years as columns.

data\_trans : data with years as columns.

'''

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

data\_trans = data.transpose()

return data, data\_trans

def getData(indicator, sourceData, year, countries):

'''

Parameters

----------

indicator : the indicator code for the specific feature in the data.

sourceData : the input data to be processed.

year : the required years to process the data.

countries : the required countries in the data.

Returns

-------

result : the output data with only the requested data from the indicator given.

'''

requestedData = []

groupData = {}

for p in range(0, len(year)):

for i in range(0, len(sourceData)):

for j in range(0, len(countries)):

if (sourceData["Indicator Code"][i] == indicator) and (sourceData["Country Name"][i] == countries[j]):{

requestedData.append([sourceData["Country Name"][i],sourceData[year[p]][i]])

}

for key, value in requestedData:

if key not in groupData:

groupData[key] = [value]

else:

groupData[key].append(value)

result = [[key, \*values] for key, values in groupData.items()]

return result

def plotLineChart(data, years):

'''

Parameters

----------

data : Data for plotting the line graph

years : the axis labels and years data for multiple plots.

Returns

-------

None.

'''

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

for i in range(0, len(data)):

name\_data = data[i]

plot\_data = data[i][1:]

plt.plot(years, plot\_data, marker='o' , markersize=4 ,label = name\_data[0])

plt.legend()

plt.xlabel('Year')

plt.title('Renewable energy consumption of different countries ')

plt.ylabel('Renewable energy consumption (% of total final energy consumption)')

plt.show()

def plotBarGraph(data, years):

'''

Parameters

----------

data : Data for plotting the bar graph

years : the axis labels and years data for multiple plots.

Returns

-------

None.

'''

plt.figure(figsize=(12, 6))

X\_axis = np.arange(len(years))

plt.bar(X\_axis-0.05, data[0][1:], 0.1, color = 'r',label = "Australia" )

plt.bar(X\_axis+0.05, data[1][1:], 0.1, color = 'b',label = "Canada")

plt.bar(X\_axis+0.15, data[2][1:], 0.1, color = 'g',label = "France")

plt.bar(X\_axis+0.25, data[3][1:], 0.1, color = 'y',label = "United Kingdom")

plt.bar(X\_axis+0.55, data[4][1:], 0.1, color = 'm',label = "India")

plt.bar(X\_axis+0.45, data[5][1:], 0.1, color = 'black',label = "Japan")

plt.bar(X\_axis+0.35, data[6][1:], 0.1, color = 'c',label = "United states" )

plt.xticks(X\_axis, years)

plt.xlabel('Year')

plt.title('Population Growth of countries through modernizing ')

plt.ylabel('Urban population growth (annual %)')

plt.legend()

plt.show()

def dataCleaning(dataInput, years):

'''

Parameters

----------

dataInput : Input data for cleaning the file.

years : the required years to process the data.

Returns

-------

dataInput : The final data with only the requested rows(country) and columns(years)

'''

columns = list(dataInput.columns)

for i in range(0, len(years)):

columns.remove(years[i])

for i in range(0, len(dataInput)):

if dataInput["Indicator Code"][i] != "SP.POP.GROW":

dataInput = dataInput.drop(i)

else:

continue

for i in range(0, len(columns)):

dataInput = dataInput.drop(columns[i], axis=1)

return dataInput

def heatMap(data, country, indicators):

'''

Parameters

----------

data : Input data for cleaning the file.

country: the required countries in the data.

indicator : the indicator code for the specific feature in the data.

Returns

-------

None.

'''

requiredData = {'Country': [],

'Feature':[],

'Value':[]

}

for k in range(0, len(country)):

for j in range(0, len(indicators)):

for i in range(0, len(data)):

if data["Country Name"][i] == country[k] and data["Indicator Code"][i] == indicators[j]:

requiredData['Country'].append(country[k])

requiredData['Feature'].append(data["Indicator Name"][i])

requiredData['Value'].append(data["2010"][i])

sortedData = pd.DataFrame(requiredData)

print(sortedData)

heatMap = sortedData.pivot\_table(index='Country', columns=['Feature'], values='Value', aggfunc='sum')

sns.heatmap(heatMap, annot=True, cmap='viridis')

plt.title("Heatmap of Countries with comparison between different features")

plt.show()

countriesPlotting = ["Australia", "Canada", "United Kingdom", "United States", "India", "France", "Japan"]

yearsPlotting = ["1970", "1980", "1990", "2000", "2010", "2020"]

yearsSummary = ["1970", "1980", "2000", "2020"]

# Retrieving data with 2 dataframes (one with years as columns and other with countries)

years\_as\_col, country\_as\_col = read\_data("climate.csv")

# Summary of data for the specified years and country

summarydata = dataCleaning(years\_as\_col, yearsSummary)

#Returns the statistical summary of the data provided

summary = summarydata.describe()

print("Summary of the data")

print(summary)

# HeatMap for different countries with the specified features

country = ["India", "Australia", "United Kingdom", "United States"]

indicatorCode = ["SP.URB.TOTL.IN.ZS", "AG.LND.ARBL.ZS", "SH.DYN.MORT", "EN.ATM.CO2E.LF.ZS", "EG.ELC.ACCS.ZS", "AG.LND.FRST.ZS"]

#Provides the heat map of the data provided

heatMap(years\_as\_col, country, indicatorCode)

# Plots line chart for population growth of countries

dataPopulation = getData("SP.POP.GROW" , years\_as\_col, yearsPlotting, countriesPlotting)

plotLineChart(dataPopulation, yearsPlotting)

# Plots line chart Renewable Energy consumption for different countries

dataRenewable = getData("EG.FEC.RNEW.ZS" , years\_as\_col, yearsPlotting, countriesPlotting) #Renewable energy consumption (% of total final energy consumption)

plotLineChart(dataRenewable, yearsPlotting)

# Plots line chart CO2 Emissions of countries

dataCo2 = getData("EN.ATM.CO2E.KT" , years\_as\_col, yearsPlotting, countriesPlotting) #CO2 emissions (kt)

plotLineChart(dataCo2, yearsPlotting)

# Plots line chart Electric power consumption of different countries

dataPower = getData("EG.USE.ELEC.KH.PC" , years\_as\_col, yearsPlotting, countriesPlotting) # Electric power consumption (kWh per capita)

plotLineChart(dataPower, yearsPlotting)

# Plots Bar graph for Electric power consumption

dataPowerBar = getData("EG.USE.ELEC.KH.PC" , years\_as\_col, yearsPlotting, countriesPlotting) #Electric power consumption (kWh per capita)

plotBarGraph(dataPowerBar, yearsPlotting)