#!/usr/bin/env python3

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

"""

Created on Sun Dec 3 16:45:32 2023

@author: praveen

"""

import numpy as np

import pandas

import seaborn as sns

import matplotlib.pyplot as plt

def main():

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

countriesToPlot = ["Switzerland", "Russia", "United States", "China", "Germany", "Singapore"]

#Returns two dataframes with years and countries as columns respectively

yearsDataframe, countryDataframe = readingData("climate.csv")

indCode = "SP.POP.GROW"

summary = summaryDescription(yearsDataframe, yearsToPlot, indCode)

#Returns the statistical summary of the data provided

print(summary.describe())

tab = summary.head(10)

list\_from\_df = tab.values.tolist()

data = {yearsToPlot[0]:list\_from\_df[0], yearsToPlot[1]:list\_from\_df[1], yearsToPlot[2]:list\_from\_df[2],yearsToPlot[3]:list\_from\_df[3], yearsToPlot[4]:list\_from\_df[4]}

countriesToTable = ["Switzerland", "United States", "China", "Germany", "Singapore"]

table\_data = pandas.DataFrame(data, index=countriesToTable)

#Returns the table of the summary data taken

print(table\_data)

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"]

#Creates heatmap for the data provided

createHeatmap(yearsDataframe, countriesToPlot, indicatorCode)

indicatorData = {'Countries':[], 'Indicators': [], 'Values': []}

countryCorr = ["India", "Germany", "Australia"]

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

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

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

if yearsDataframe["Country Name"][i] == countryCorr[k] and yearsDataframe["Indicator Code"][i] == indicatorCode[j]:

indicatorData['Countries'].append(countryCorr[k])

indicatorData['Indicators'].append(yearsDataframe["Indicator Name"][j])

indicatorData['Values'].append(yearsDataframe["2010"][i])

dataFrameIndicator = pandas.DataFrame(indicatorData)

heatmapData = dataFrameIndicator.pivot\_table(index='Countries', columns=['Indicators'], values='Values', aggfunc='sum')

correlationMatrix = heatmapData.corr()

#Creates a correlation heatmap between the features of 3 countries

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

plt.title("Correlation Heatmap of Country with Indicators as features")

plt.show()

#Gets the data for lineplot

dataResult = gettingData("EG.USE.ELEC.KH.PC" , yearsDataframe, yearsToPlot, countriesToPlot)

#Creates a lineplot for the data provided

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

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

countryName = dataResult[i]

data = dataResult[i][1:]

plt.plot(yearsToPlot, data, marker='s' , markersize=4 ,label = countryName[0])

plt.xlabel('Year')

plt.ylabel('Electric power consumption (kWh per capita)')

plt.title('Electric power consumption of different countries though time ')

plt.legend()

plt.show()

y\_axis1 = "Renewable electricity output (% of total electricity output)"

title1 = "Renewable electricity output of Countries over time"

yearsBar = ["1990", "2000","2005", "2010", "2015", "2020"]

dataBarChart1 = gettingData("EG.ELC.RNEW.ZS" , yearsDataframe, yearsBar, countriesToPlot)

#Plots bar chart for the given data and titles

barPlot(dataBarChart1, yearsBar, y\_axis1, title1)

y\_axis2 = "Renewable energy consumption (% of total final energy consumption)"

title2 = "Renewable energy consumption of different countries"

dataBarChart2 = gettingData("EG.FEC.RNEW.ZS" , yearsDataframe, yearsBar, countriesToPlot)

#Plots bar chart for the given data and titles

barPlot(dataBarChart2, yearsBar, y\_axis2, title2)

def summaryDescription(data, years, code):

'''

Cleans the input data and returns the final data with only the requested rows (country) and columns (years).

Parameters

----------

data : pandas DataFrame

Input data for cleaning the file.

years : list

The required years to process the data.

code : str

The indicator code string for selecting data

Returns

-------

data : pandas DataFrame

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

'''

columns = list(data.columns)

for year in years[:len(years) -1]:

columns.remove(year)

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

if data["Indicator Code"][i] != code:

data = data.drop(i)

else:

continue

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

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

return data

def readingData(inputFile):

'''

Reads data from the input filename (.csv) and returns two dataframes.

Parameters

----------

inputFile : str

Input filename (.csv) to read the data.

Returns

-------

inputData : pandas DataFrame

Data with years as columns.

dataTransposed : pandas DataFrame

Transposed data with years as columns.

'''

inputData = pandas.read\_csv(inputFile, skiprows=3)

dataTransposed = inputData.transpose()

return inputData, dataTransposed

def createHeatmap(data, countries, indicators):

'''

Generates a heatmap for specified countries and indicators based on input data.

Parameters

----------

data : pandas DataFrame

Input data for cleaning the file.

countries : list

The required countries in the data.

indicators : list

The indicator codes for the specific features in the data.

Returns

-------

None.

'''

selected\_data = {'Countries': [], 'Indicators': [], 'Values': []}

for country in countries:

for indicator in indicators:

for index, row in data.iterrows():

if row["Country Name"] == country and row["Indicator Code"] == indicator:

selected\_data['Countries'].append(country)

selected\_data['Indicators'].append(row["Indicator Name"])

selected\_data['Values'].append(row["2010"])

sorted\_data = pandas.DataFrame(selected\_data)

heatmap\_data = sorted\_data.pivot\_table(index='Countries', columns=['Indicators'], values='Values', aggfunc='sum')

#Creates heatmap for the data provided

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

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

plt.show()

def gettingData(indicators, source\_data, years, countries):

'''

Extracts and processes data based on the specified indicator, source data, years, and countries.

Parameters

----------

indicators : str

The indicator code for the specific feature in the data.

source\_data : pandas DataFrame

The input data to be processed.

years : list

The required years to process the data.

countries : list

The required countries in the data.

Returns

-------

finalResult : list

The output data with only the requested data from the given indicator.

'''

resultData = []

dataFrameGroup = {}

for year in years:

for index, row in source\_data.iterrows():

for country in countries:

if row["Indicator Code"] == indicators and row["Country Name"] == country:

resultData.append([row["Country Name"], row[year]])

for key, value in resultData:

if key not in dataFrameGroup:

dataFrameGroup[key] = [value]

else:

dataFrameGroup[key].append(value)

finalResult = [[key,\*values] for key, values in dataFrameGroup.items()]

return finalResult

def barPlot(data, years, yaxis, title):

'''

Parameters

----------

data : Data for plotting the bar graph

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

yaxis : y axis label for the graph

title : title for the plot

Returns

-------

None.

'''

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

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

colors = ['red', 'blue', 'yellow', 'green', 'magenta', 'black', 'cyan']

plt.bar(x\_axis-0.05, data[0][1:], 0.1, color=colors[0], label="Switzerland")

plt.bar(x\_axis+0.05, data[1][1:], 0.1, color=colors[1], label="China")

plt.bar(x\_axis+0.15, data[2][1:], 0.1, color=colors[2], label="Germany")

plt.bar(x\_axis+0.25, data[3][1:], 0.1, color=colors[3], label="Singapore")

plt.bar(x\_axis+0.35, data[4][1:], 0.1, color=colors[4], label="United States")

plt.xticks(x\_axis, years)

plt.xlabel('Year')

plt.title(title)

plt.ylabel(yaxis)

plt.legend()

plt.show()

main()