import numpy as np  
import pandas as pd  
import matplotlib.pyplot as plt  
  
from pandas import Series, DataFrame  
from pandas.io.parsers import TextFileReader  
  
#Read CSV's into Project  
CD = pd.read\_csv('transformed\_data.csv')  
RD = pd.read\_csv('raw\_data.csv')  
EU = pd.read\_csv('states.csv')  
Countries = pd.read\_csv('countries\_of\_the\_world.csv')  
  
  
# Create Key to merge data together  
CD.sort\_values('DATE')  
CD['KEYCODE&DATE'] = CD['CODE'] + CD['DATE']  
RD.sort\_values('date')  
RD['KEYCODE&DATE'] = RD['iso\_code'] + RD['date']  
print(RD['KEYCODE&DATE'])  
print(CD['KEYCODE&DATE'])  
  
#Sort rows before creating new data series  
RD['DATE'] = pd.to\_datetime(RD['date'], format='%Y/%m/%d')  
CD['DATE'] = pd.to\_datetime(CD['DATE'], format='%Y/%m/%d')  
  
  
  
print(CD.columns)  
print(RD.columns)  
print(EU.columns)  
  
  
  
# Merge EU data into CD Data  
CD = (CD.merge(EU, left\_on='COUNTRY', right\_on='Country', how='inner'))  
print(CD.head)  
print(CD.columns)  
print(CD.isna().sum())  
  
CD.to\_csv('CD.csv')  
  
# Merge enlarged CD dataset with RD dataset  
RDCD = (RD.merge(CD, on='KEYCODE&DATE', how='outer', suffixes=('\_RD', '\_CD')))  
  
#Create new fields re Dates  
RDCD['DATE'] = pd.to\_datetime(RDCD['date'], format='%Y/%m/%d')  
print(RDCD['date'])  
# Create Key Fields  
RDCD['Week\_Num'] = RDCD['DATE'].dt.strftime('%U') #number of the week in the year  
RDCD['Month'] = RDCD['DATE'].dt.strftime('%b') #short name of the month  
RDCD['day'] = RDCD['DATE'].dt.strftime('%A') #day of the week  
RDCD['dayz'] = RDCD['DATE'].dt.strftime('%j') #number of the day in the year  
RDCD['Dates'] = RDCD['DATE'].dt.strftime('%Y%m%d') #dont think this is doing anything useful  
RDCD['Wednesdays'] = RDCD['day'] == 'Wednesday' #boolean set to confirm Wednesday as True  
  
# add columns with comparable measures  
RDCD['CasesxArea'] = RDCD['total\_cases'] / RDCD['Area (km²)'].round()  
RDCD['CasesxPop'] = RDCD['total\_cases'] / RDCD['population'].round()  
RDCD['DeathsxCases'] = (RDCD['total\_deaths'] / RDCD['total\_cases']).round(1)  
RDCD['AreaKM2'] =RDCD['Area (km²)'] #new field  
RDCD['PopKM2'] =RDCD ['population'] / RDCD['Area (km²)'] #new field  
#RDCD['Cases per AreaKM²'] = (RDCD['total\_cases'] / RDCD['Area (km²)']).round(2)  
#Change Govt'stringency\_index' measure to a comparable index (0-5) called Stringency\_Indexed  
RDCD['Stringency\_Indexed']=(RDCD['stringency\_index'] / 20).round()  
print(RDCD)  
  
# park/remove unneeded rows  
del RDCD['Unnamed: 9']  
del RDCD['Unnamed: 10']  
del RDCD['Unnamed: 11']  
del RDCD['Unnamed: 12']  
del RDCD['Unnamed: 13']  
del RDCD['GDP per capita ($, millions)']  
del RDCD['GDP ($, millions)']  
del RDCD['GDP (€, millions)']  
#del RDCD['Population Density']  
del RDCD['Population']  
del RDCD['Language']  
del RDCD['Currency Code']  
del RDCD['Currency']  
del RDCD['European Monetary Union']  
del RDCD['European Single Market']  
del RDCD['European Free Trade Agreement']  
del RDCD['European Parliament Seats']  
del RDCD['Council Votes']  
del RDCD['Accession Year']  
del RDCD['GDPCAP']  
del RDCD['POP']  
del RDCD['STI']  
del RDCD['TD']  
del RDCD['TC']  
del RDCD['HDI']  
del RDCD['COUNTRY']  
del RDCD['CODE']  
del RDCD['KEYCODE&DATE']  
#del RDCD['DATE\_RD']  
#del RDCD['DATE\_CD']  
  
# Create groups of Economic Blocks from Dictionary  
econ\_blocks = {  
 'Country': ['Austria', 'Belgium', 'Bulgaria', 'Croatia', 'Cyprus', 'Czech Republic', 'Denmark', 'Estonia',  
 'Finland', 'France', 'Germany', 'Greece', 'Hungary', 'Ireland', 'Italy', 'Latvia', 'Lithuania',  
 'Luxembourg', 'Malta', 'Netherlands', 'Poland', 'Portugal', 'Romania', 'Slovakia', 'Slovenia', 'Spain',  
 'Sweden', 'United Kingdom', 'Brazil', 'Russia', 'India', 'China', 'South Africa', 'United States'],  
 'Econ\_Block': ['EU', 'EU', 'EU', 'EU', 'EU', 'EU', 'EU', 'EU', 'EU', 'EU', 'EU', 'EU', 'EU', 'EU', 'EU', 'EU', 'EU',  
 'EU', 'EU', 'EU', 'EU', 'EU', 'EU', 'EU', 'EU', 'EU', 'EU', 'EU', 'BRICS', 'BRICS', 'BRICS', 'BRICS',  
 'BRICS', 'US']}  
#convert econ\_block to dataframe  
Econ\_BLocks = pd.DataFrame(econ\_blocks)  
  
# Merge Econ\_BLocks dataset to enhance RDCD dataset  
RDCD = (RDCD.merge(Econ\_BLocks, left\_on='location', right\_on='Country', how='outer'))  
  
# check for missing data  
#print(RDCD.isna().any())  
print(RDCD.isna().sum())  
  
  
#Replace Missing Values by filling 'na's with 'No\_Affiliation' identifier  
RDCD.fillna(0)  
RDCD['Econ\_Block'] = RDCD['Econ\_Block'].fillna('All\_Others')  
  
print(RDCD.isna().sum())  
print(RDCD)  
  
# Create a Weekly Dataset to using Wednesday as the filter to give a weekly reporting day  
RDCD=RDCD.query('day == "Wednesday"')  
  
  
print(RDCD)  
print(RDCD.isna().any())  
print(RDCD.isna().sum())  
print(RDCD.dtypes)  
  
  
#output this to csv  
RDCD.to\_csv('RDCD.csv')  
  
# import this as csv  
RDCD1 = pd.read\_csv('RDCD.csv')  
#mask = (RDCD['DATE\_CD'] > '2020-01-01') & (RDCD['DATE\_CD'] < '2020-09-30')  
#Date\_Range=RDCD.loc[mask]  
#print(Date\_Range)  
  
  
  
del RDCD1['Country\_y']  
del RDCD1['Unnamed: 0']  
  
  
  
# Calculate Weekly Change in Cases & Deaths from cumulative data within dataset  
RDCD1['Weekly\_Cases']=(RDCD1.groupby('iso\_code')['total\_cases'].diff())  
RDCD1['Weekly\_Deaths']=(RDCD1.groupby('iso\_code')['total\_deaths'].diff())  
  
  
  
print(RDCD1.dtypes)  
  
group=(RDCD1.groupby('iso\_code')['Week\_Num'].count())  
print(group)  
#select final date range to analyse  
RDCD1=RDCD1.query("DATE\_RD >= '2020-01-07' and DATE\_RD <='2020-09-30'")  
  
#output this to csv  
RDCD1.to\_csv('RDCD11.csv') #being used by subsetting .loc to create Key\_columns\_Only.csv  
  
RDCD2 = pd.read\_csv('RDCD11.csv')  
  
#RDCD2['Stringency\_Index\_5'] = RDCD2.loc[:, ['Stringency\_Indexed']==5]  
  
# Using .loc to create boolean of Stringency at 5 ad store as new column  
RDCD2.loc[:, 'Stringency\_Index\_5']= RDCD2['Stringency\_Indexed']>=4  
print(RDCD2.dtypes)  
  
#filter stringency levels >=4 by country  
RDCD2=RDCD2.query('Stringency\_Index\_5 == True')  
RDCD2.to\_csv('RDCD2.csv')  
RDCD3 = pd.read\_csv('RDCD2.csv')  
  
RDCD12 = pd.read\_csv('RDCD.csv')  
RDCD12a=RDCD12.query('Week\_Num== 39')  
RDCD12a.to\_csv('RDCD12a.csv')  
RDCD12b = pd.read\_csv('RDCD12a.csv')  
RDCD12c = RDCD12b.query('Econ\_Block == "EU"')  
RDCD12c.to\_csv('RDCD12c.csv')  
RDCD12c = pd.read\_csv('RDCD12c.csv')  
  
#create a pivot table  
RDCD4=(RDCD1.pivot\_table(values='DeathsxCases', index='Month', columns='Econ\_Block', aggfunc='sum',fill\_value=0, margins=True).iloc[:-1,:])  
RDCD4a=RDCD4.fillna(0)  
new\_order=['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep']  
RDCD4b=RDCD4a.reindex(new\_order, axis=0)  
RDCD4b.to\_csv('RDCD4b.csv')  
print(RDCD4b)  
  
RDCD5=(RDCD1.pivot\_table(values='Weekly\_Deaths', index='Month', columns='Econ\_Block', aggfunc='sum', fill\_value=0, margins=False))  
new\_order=['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep']  
RDCD5a=RDCD5.reindex(new\_order, axis=0)  
RDCD5a.to\_csv('RDCD5a.csv')  
print(RDCD5a)  
  
RDCD6=(RDCD1.pivot\_table(values='Weekly\_Cases', index=['Month'], columns=['Econ\_Block'], aggfunc='sum', fill\_value=0, margins=False))  
new\_order=['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep']  
RDCD6a=RDCD6.reindex(new\_order, axis=0)  
RDCD6a.to\_csv('RDCD6a.csv')  
print(RDCD6a)  
  
RDCD7=(RDCD1.pivot\_table(values=['Weekly\_Cases','DeathsxCases'] , index=['Week\_Num'], aggfunc='sum', fill\_value=0))  
RDCD7.to\_csv('RDCD7.csv')  
print(RDCD7)  
  
RDCD8=(RDCD1.pivot\_table(values='Weekly\_Cases', index='Week\_Num', columns='location', aggfunc='sum', fill\_value=0, margins=True, margins\_name='Grand\_Total\_Cases').iloc[:-1,:])  
RDCD8.to\_csv('RDCD8.csv')  
print(RDCD8)  
RDCD8a=(RDCD1.pivot\_table(values='Weekly\_Deaths', index='Week\_Num', columns='location', aggfunc='sum', fill\_value=0, margins=True, margins\_name='Grand\_Total\_Deaths').iloc[:-1,:])  
RDCD8a.to\_csv('RDCD8a.csv')  
print(RDCD8a)  
  
RDCD3 = pd.read\_csv('RDCD2.csv')  
RDCD9=(RDCD3.pivot\_table(values='Stringency\_Indexed', index='Week\_Num', columns='location', aggfunc='count', fill\_value=0, margins=True, margins\_name='Grand\_Total').iloc[:-1,:])  
RDCD9.to\_csv('RDCD9.csv')  
print(RDCD9)  
  
#new\_order=['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep']  
#RDCD9a=RDCD9.reindex(new\_order, axis=0)  
RDCD9e=(RDCD3.pivot\_table(values='Stringency\_Indexed' , columns='location', index='Week\_Num', aggfunc='count', fill\_value=0, margins=True))  
RDCD9e.to\_csv('RDCD9e.csv')  
print(RDCD9e)  
RDCD9f=(RDCD3.pivot\_table(values='Weekly\_Deaths', columns='location', index='Week\_Num', aggfunc='sum', fill\_value=0, margins=True))  
RDCD9e.to\_csv('RDCD9f.csv')  
print(RDCD9f)  
  
#Bubble Chart, cases per Area on YTD Wk39  
RDCD12h=(RDCD12c.pivot\_table(values=['PopKM2'], index='location', aggfunc='sum', fill\_value=0, margins=True, margins\_name='Grand\_Total').iloc[:-1,:])  
RDCD12h.to\_csv('RDCD12h.csv')  
print(RDCD12h)  
RDCD12i=(RDCD12c.pivot\_table(values=['CasesxArea'], index='location', aggfunc='sum', fill\_value=0, margins=True, margins\_name='Grand\_Total').iloc[:-1,:])  
RDCD12i.to\_csv('RDCD12i.csv')  
print(RDCD12i)  
  
  
  
RDCD12g=(RDCD12c.pivot\_table(values=['AreaKM2'], index='location', aggfunc='sum', fill\_value=0, margins=True, margins\_name='Grand\_Total').iloc[:-1,:])  
RDCD12g.to\_csv('RDCD12g.csv')  
print(RDCD12g)  
RDCD12j=(RDCD12c.pivot\_table(values=['population'], index='location', aggfunc='sum', fill\_value=0, margins=True, margins\_name='Grand\_Total').iloc[:-1,:])  
RDCD12j.to\_csv('RDCD12j.csv')  
print(RDCD12j)  
RDCD12k=(RDCD12c.pivot\_table(values=['Population Density'], index='location', aggfunc='sum', fill\_value=0, margins=True, margins\_name='Grand\_Total').iloc[:-1,:])  
RDCD12k.to\_csv('RDCD12k.csv')  
print(RDCD12k)  
  
RDCD12l=(RDCD12c.pivot\_table(values=['Stringency\_Indexed', 'total\_cases', 'Population Density', 'CasesxArea', 'CasesxPop', 'AreaKM2'], index='location', aggfunc='sum', fill\_value=0, margins=True, margins\_name='Grand\_Total').iloc[:-1,:])  
RDCD12l.to\_csv('RDCD12l.csv')  
print(RDCD12l)

RDCDString = pd.read\_csv('RDCD9f.csv')  
print(RDCDString)  
RDCDStringtote=RDCDString[['All']]  
RDCDStringtote.to\_csv('StringL5WkTot.csv')  
print(RDCDStringtote)