---  
title: "Quarto Basics"  
format:  
 html:  
 code-fold: true  
jupyter: python3  
---

SyntaxError: ignored

import pyspark  
from pyspark import SparkContext  
from pyspark.sql import SparkSession  
#from pyspark.sql.functions import \*  
from pyspark.sql import functions as func  
import pandas as pd  
import seaborn as sns  
import numpy as np  
import matplotlib.pyplot as plt  
import plotly.express as px  
import plotly.io as pio  
pio.renderers.default = "plotly\_mimetype+notebook\_connected"  
  
sc = SparkContext.getOrCreate()  
spark = SparkSession.builder.appName("Python Spark").getOrCreate()

df\_annual\_co2 = spark.read\  
 .option("header", "true")\  
 .csv('Data/annual-co2-emissions-per-country.csv', inferSchema='true')  
  
df\_annual\_death = spark.read\  
 .option("header", "true")\  
 .csv('Data/annual-number-of-deaths-by-world-region.csv', inferSchema='true')  
  
df\_pib = spark.read\  
 .option("header", "true")\  
 .csv('Data/gdp(pib)-per-capita-maddison-2020.csv', inferSchema='true')  
  
df\_population = spark.read\  
 .option("header", "true")\  
 .csv('Data/population-world.csv', inferSchema='true')  
  
df\_nb\_wo\_clean\_cooking\_fuel = spark.read\  
 .option("header", "true")\  
 .csv('Data/number-without-clean-cooking-fuel.csv', inferSchema='true')  
  
df\_pop\_growth = spark.read\  
 .option("header", "true")\  
 .csv('Data/population-growth-rate-vs-median-age.csv', inferSchema='true')  
  
df\_co2\_target = spark.read\  
 .option("header", "true")\  
 .csv('Data/net-zero-target-set.csv', inferSchema='true')  
  
df\_continent = spark.read\  
 .option("header", "true")\  
 .csv('Data/countryContinent.csv', inferSchema='true')  
  
df\_iso\_alpha = spark.read\  
 .option("header", "true")\  
 .csv('Data/iso\_alpha.csv', inferSchema='true')

df\_annual\_co2.printSchema()  
df\_annual\_death.printSchema()  
df\_pib.printSchema()  
df\_population.printSchema()  
df\_nb\_wo\_clean\_cooking\_fuel.printSchema()  
df\_pop\_growth.printSchema()  
df\_co2\_target.printSchema()  
df\_continent.printSchema()

df\_annual\_co2.show(1)  
df\_annual\_death.show(1)  
df\_pib.show(1)  
df\_population.show(1)  
df\_nb\_wo\_clean\_cooking\_fuel.show(1)  
df\_pop\_growth.show(1)  
df\_co2\_target.show(1)  
df\_continent.show(1)

# Création des dataframes

#Join global en inner donc perte de données (seulement jusqu'à 2018)  
df\_all = df\_annual\_co2.join(df\_annual\_death, (df\_annual\_co2['Code'] == df\_annual\_death['Code']) \  
 & (df\_annual\_co2['Year'] == df\_annual\_death['Year']), 'inner')\  
 .select(df\_annual\_co2['Entity'].alias('Country'), df\_annual\_co2['Code'], df\_annual\_co2['Year'],\  
 df\_annual\_co2['Annual CO2 emissions'].alias('Annual\_CO2\_emissions'),\  
 df\_annual\_death['Deaths - sex: all - age: all - variant: estimates'].alias('Deaths'))\  
  
df\_all = df\_all.join(df\_pib, (df\_all['Code'] == df\_pib['Code']) \  
 & (df\_all['Year'] == df\_pib['Year']), 'inner')\  
 .select(df\_all['Country'], df\_all['Code'], df\_all['Year'], df\_all['Annual\_CO2\_emissions'],\  
 df\_all['Deaths'],df\_pib['GDP per capita'].alias('PIB\_per\_capita'))  
  
df\_all = df\_all.withColumn('PIB\_per\_capita', func.round(df\_all['PIB\_per\_capita'],2)) # vire les décimales  
  
df\_all = df\_all.join(df\_population, (df\_all['Code'] == df\_population['Code']) \  
 & (df\_all['Year'] == df\_population['Year']), 'inner')\  
 .select(df\_all['Country'], df\_all['Code'], df\_all['Year'], df\_all['Annual\_CO2\_emissions'],\  
 df\_all['Deaths'],df\_all['PIB\_per\_capita'], df\_population['Population - Sex: all - Age: all - Variant: estimates'].alias('Population'))  
   
df\_all = df\_all.join(df\_pop\_growth, (df\_all['Code'] == df\_pop\_growth['Code']) \  
 & (df\_all['Year'] == df\_pop\_growth['Year']), 'inner')\  
 .select(df\_all['Country'], df\_all['Code'], df\_all['Year'], df\_all['Annual\_CO2\_emissions'],\  
 df\_all['Deaths'],df\_all['PIB\_per\_capita'], df\_all['Population'],\  
 df\_pop\_growth['Estimates, 1950 - 2020: Annually interpolated demographic indicators - Rate of natural increase (per 1,000 population)']\  
 .alias('Rate\_of\_natural\_population\_increase'))  
  
df\_all = df\_all.withColumn('Rate\_of\_natural\_population\_increase', func.round(df\_all['Rate\_of\_natural\_population\_increase'],2)) # vire les décimales  
  
df\_all = df\_all.join(df\_co2\_target, (df\_all['Code'] == df\_co2\_target['Code']) \  
 #& (df\_all['Year'] == df\_co2\_target['Year'])  
 , 'left')\  
 .select(df\_all['Country'], df\_all['Code'], df\_all['Year'], df\_all['Annual\_CO2\_emissions'],\  
 df\_all['Deaths'],df\_all['PIB\_per\_capita'], df\_all['Population'], df\_all['Rate\_of\_natural\_population\_increase'],\  
 df\_co2\_target['Year'].alias('Net\_zero\_target'))  
   
df\_all = df\_all.join(df\_continent, (df\_all['Code'] == df\_continent['code\_3']) #supprime les données 'World' \  
 #& (df\_all['Year'] == df\_co2\_target['Year'])  
 , 'inner')\  
 .select(df\_all['Country'], df\_all['Code'], df\_all['Year'], df\_all['Annual\_CO2\_emissions'],\  
 df\_all['Deaths'],df\_all['PIB\_per\_capita'], df\_all['Population'], df\_all['Rate\_of\_natural\_population\_increase'],\  
 df\_all['Net\_zero\_target'], df\_continent['continent'], df\_continent['sub\_region'] )  
  
df\_all = df\_all.join(df\_iso\_alpha, (df\_all['Code'] == df\_continent['code\_3']) #supprime les données 'World' \  
 #& (df\_all['Year'] == df\_co2\_target['Year'])  
 , 'inner')\  
 .select(df\_all['Country'], df\_all['Code'], df\_all['Year'], df\_all['Annual\_CO2\_emissions'],\  
 df\_all['Deaths'],df\_all['PIB\_per\_capita'], df\_all['Population'], df\_all['Rate\_of\_natural\_population\_increase'],\  
 df\_all['Net\_zero\_target'], df\_continent['continent'], df\_continent['sub\_region'] )  
  
  
  
###########################################  
#Join avec données jusqu'à 2020 : emmission C02, population, continent  
  
  
df\_pop = df\_annual\_co2.join(df\_population, (df\_annual\_co2['Code'] == df\_population['Code']) \  
 & (df\_annual\_co2['Year'] == df\_population['Year']), 'outer')\  
 .select(df\_annual\_co2['Entity'].alias('Country'), df\_annual\_co2['Code'], df\_annual\_co2['Year'],\  
 df\_annual\_co2['Annual CO2 emissions'].alias('Annual\_CO2\_emissions'),\  
 df\_population['Population - Sex: all - Age: all - Variant: estimates'].alias('Population'))  
   
df\_pop = df\_pop.join(df\_continent, (df\_pop['Code'] == df\_continent['code\_3']) #supprime les données 'World' \  
 #& (df\_all['Year'] == df\_co2\_target['Year'])  
 , 'outer')\  
 .select(df\_pop['Country'], df\_pop['Code'], df\_pop['Year'], df\_pop['Annual\_CO2\_emissions'],\  
 df\_pop['Population'], df\_continent['continent'], df\_continent['sub\_region'] )  
  
df\_pop = df\_pop.where(df\_all.Year>1949)  
  
  
##############################################  
  
df\_world\_cO2 = df\_annual\_co2.filter((df\_annual\_co2.Entity == 'World') | (df\_annual\_co2.Entity =='China')\  
 | (df\_annual\_co2.Entity =='United States') | (df\_annual\_co2.Entity =='India')\  
 | (df\_annual\_co2.Entity =='Russia')).select(df\_annual\_co2['Entity'].alias('Country'),\  
 df\_annual\_co2['Code'], df\_annual\_co2['Year'], df\_annual\_co2['Annual CO2 emissions'].alias('Annual\_CO2\_emissions'))

## Data cleaning

#Suppression de la ligne avec Code = DMA (contient une valeur null)  
cond = (df\_all.Code == 'DMA')  
df\_all = df\_all.filter(~cond)  
  
#Pareil pour 'World'  
cond = (df\_all.Code == 'OWID\_WRL')  
df\_all = df\_all.filter(~cond)

#On vérifie que les 'pays' Dominica et World ont bien disparu  
df\_all.filter(df\_all.Code =='DMA').show(truncate=False)  
df\_all.filter(df\_all.Code == 'OWID\_WRL').show()#.select(countDistinct(df\_all.Year)).show()

+-------+----+----+--------------------+------+--------------+----------+-----------------------------------+---------------+---------+----------+  
|Country|Code|Year|Annual\_CO2\_emissions|Deaths|PIB\_per\_capita|Population|Rate\_of\_natural\_population\_increase|Net\_zero\_target|continent|sub\_region|  
+-------+----+----+--------------------+------+--------------+----------+-----------------------------------+---------------+---------+----------+  
+-------+----+----+--------------------+------+--------------+----------+-----------------------------------+---------------+---------+----------+  
  
+-------+----+----+--------------------+------+--------------+----------+-----------------------------------+---------------+---------+----------+  
|Country|Code|Year|Annual\_CO2\_emissions|Deaths|PIB\_per\_capita|Population|Rate\_of\_natural\_population\_increase|Net\_zero\_target|continent|sub\_region|  
+-------+----+----+--------------------+------+--------------+----------+-----------------------------------+---------------+---------+----------+  
+-------+----+----+--------------------+------+--------------+----------+-----------------------------------+---------------+---------+----------+

## Visu en interne : data

#nombre total de pays répértoriés  
df\_all.select(func.countDistinct('Country')).show(truncate=False)

[Stage 1620:========================> (93 + 8) / 200]

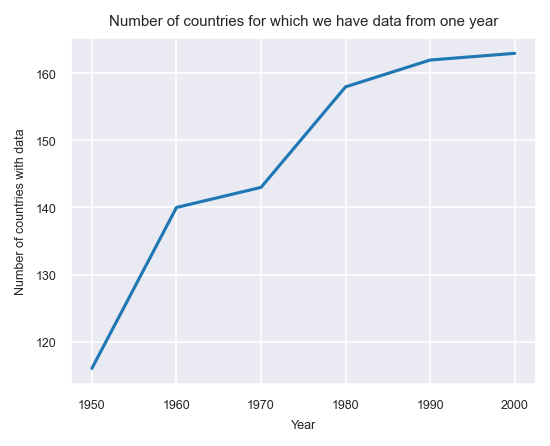
+-----------------------+  
|count(DISTINCT Country)|  
+-----------------------+  
|163 |  
+-----------------------+

#rechercher cb de pays sur 163 ont l'info pour une année donnée  
annee = 2000  
df\_all.filter(df\_all.Year==annee).select(func.count('Year')).collect()[0][0]  
  
#à partir de l'année  
#1950 : on a 117 pays avec toutes les infos  
#1960 : on a 141 pays avec toutes les infos  
#1970 : on a 144 pays avec toutes les infos  
#1980 : on a 159 pays avec toutes les infos  
#1990 : on a 163 pays avec toutes les infos  
#2000 : tous les pays : 164

216

#visualisation de l'évolution du nombre de pays pour lesquels nous avons toutes les informations#  
fig = plt.figure(figsize=(4,3), dpi = 150)  
annees = []  
pays\_w\_infos = []  
for annee in range(1950,2010,10):  
 info = df\_all.filter(df\_all.Year==annee).select(func.count('Year')).collect()[0][0]  
 annees.append(annee)  
 pays\_w\_infos.append(info)  
  
new\_lst = [list(x) for x in zip(annees, pays\_w\_infos)]  
  
columns = ["year","pays\_w\_infos"]  
df\_intro = spark.createDataFrame(data=new\_lst, schema = columns)  
#df\_intro.printSchema()  
#df\_intro.show(truncate=False)  
  
df\_intro\_pd = df\_intro.toPandas()  
sns.set\_style("darkgrid")  
sns.lineplot(data=df\_intro\_pd, x="year", y="pays\_w\_infos").set(title='Number of countries for which we have data from one year',\  
 ylabel='Number of countries with data', xlabel='Year')

[Text(0.5, 1.0, 'Number of countries for which we have data from one year'),  
 Text(0, 0.5, 'Number of countries with data'),  
 Text(0.5, 0, 'Year')]



#Vérification de la suppression de la l'unique ligne DMA en 2014  
df\_pop\_growth.filter((df\_pop\_growth.Year=='2014')&(df\_pop\_growth.Code=='DMA')).show(truncate=False)

+------+----+----+----------------------------------------------------------------------------------------------------------------------+------------------------------------------+---------+  
|Entity|Code|Year|Estimates, 1950 - 2020: Annually interpolated demographic indicators - Rate of natural increase (per 1,000 population)|UN Population Division (Median Age) (2017)|Continent|  
+------+----+----+----------------------------------------------------------------------------------------------------------------------+------------------------------------------+---------+  
+------+----+----+----------------------------------------------------------------------------------------------------------------------+------------------------------------------+---------+

#Aperçu France  
df\_all.filter(df\_all.Code == 'FRA').show(5, truncate=False)

+-------+----+----+--------------------+------+--------------+----------+-----------------------------------+---------------+---------+--------------+  
|Country|Code|Year|Annual\_CO2\_emissions|Deaths|PIB\_per\_capita|Population|Rate\_of\_natural\_population\_increase|Net\_zero\_target|continent|sub\_region |  
+-------+----+----+--------------------+------+--------------+----------+-----------------------------------+---------------+---------+--------------+  
|France |FRA |1950|202428862 |533995|8266.0 |41842356 |0.66 |2050 |Europe |Western Europe|  
|France |FRA |1951|228915868 |566424|8705.0 |42196660 |0.65 |2050 |Europe |Western Europe|  
|France |FRA |1952|227188014 |525413|8869.0 |42542396 |0.64 |2050 |Europe |Western Europe|  
|France |FRA |1953|214738732 |558275|9060.0 |42883300 |0.63 |2050 |Europe |Western Europe|  
|France |FRA |1954|226568034 |519916|9428.0 |43225644 |0.63 |2050 |Europe |Western Europe|  
+-------+----+----+--------------------+------+--------------+----------+-----------------------------------+---------------+---------+--------------+  
only showing top 5 rows

#nombre de pays dont on a la donnée objectif 0 emissions  
df\_all.filter(df\_all.Net\_zero\_target.isNotNull()).select(func.countDistinct("Country")).show(truncate=False)

[Stage 1693:> (0 + 1) / 1]

+-----------------------+  
|count(DISTINCT Country)|  
+-----------------------+  
|58 |  
+-----------------------+

Essayer de selectionner seulement les 50 premiers pays polluants pour la visu car bcp trop ne polluent pas et prennent de la place sur la visu pour rien

df = df\_all.filter(df\_all.Year == 2018).orderBy(func.desc('Annual\_CO2\_emissions')).select(df\_all.Country).rdd.map(lambda x: x[0]).collect()  
df[:5]

['China', 'United States', 'India', 'Russia', 'Japan']

## Transformation df\_all vers pandas pour visu

df\_all\_pd = df\_all.toPandas() #df plein  
#df\_all\_2000\_pd = df\_all\_2000.toPandas() #df test avec année min selectionnée au dessus

## Visu 1 : World C02 emission

df = df\_world\_cO2.toPandas()  
fig = px.line(df, x="Year", y='Annual\_CO2\_emissions', color='Country', \  
 title='Carbon dioxide (CO2) emissions from fossil, fuels and industry',\  
 labels={'Annual\_CO2\_emissions':'CO2 emission (billions of t)'})  
  
fig.update\_xaxes(range=[1820, 2022])  
fig.show()

Unable to display output for mime type(s): application/vnd.plotly.v1+json, text/html

df\_all.columns

['Country',  
 'Code',  
 'Year',  
 'Annual\_CO2\_emissions',  
 'Deaths',  
 'PIB\_per\_capita',  
 'Population',  
 'Rate\_of\_natural\_population\_increase',  
 'Net\_zero\_target',  
 'continent',  
 'sub\_region']

df\_all\_pd[df\_all\_pd['Country'] == 'France']

|  | Country | Code | Year | Annual\_CO2\_emissions | Deaths | PIB\_per\_capita | Population | Rate\_of\_natural\_population\_increase | Net\_zero\_target | continent | sub\_region |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 3290 | France | FRA | 1950 | 202428862 | 533995 | 8266.00 | 41842356 | 0.66 | 2050.0 | Europe | Western Europe |
| 3291 | France | FRA | 1951 | 228915868 | 566424 | 8705.00 | 42196660 | 0.65 | 2050.0 | Europe | Western Europe |
| 3292 | France | FRA | 1952 | 227188014 | 525413 | 8869.00 | 42542396 | 0.64 | 2050.0 | Europe | Western Europe |
| 3293 | France | FRA | 1953 | 214738732 | 558275 | 9060.00 | 42883300 | 0.63 | 2050.0 | Europe | Western Europe |
| 3294 | France | FRA | 1954 | 226568034 | 519916 | 9428.00 | 43225644 | 0.63 | 2050.0 | Europe | Western Europe |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 3354 | France | FRA | 2014 | 325885736 | 546921 | 36527.00 | 63588496 | 0.31 | 2050.0 | Europe | Western Europe |
| 3355 | France | FRA | 2015 | 329910376 | 582205 | 36827.00 | 63809768 | 0.28 | 2050.0 | Europe | Western Europe |
| 3356 | France | FRA | 2016 | 333578746 | 580585 | 37124.00 | 63989320 | 0.25 | 2050.0 | Europe | Western Europe |
| 3357 | France | FRA | 2017 | 336995589 | 594684 | 37895.00 | 64144092 | 0.22 | 2050.0 | Europe | Western Europe |
| 3358 | France | FRA | 2018 | 322371888 | 596919 | 38515.92 | 64277812 | 0.19 | 2050.0 | Europe | Western Europe |

df = df\_pop.toPandas()  
fig = px.bar(df, x='Year', y='Population', color='continent', hover\_data=['Country'],\  
 title = "Evolution de la population mondiale")  
  
  
fig.show()

Unable to display output for mime type(s): application/vnd.plotly.v1+json, text/html

df2018 = df\_all\_pd.query('Year == 2018')  
  
fig = px.histogram(df2018, x = 'Annual\_CO2\_emissions', color = 'continent', marginal = 'rug', hover\_name = 'Country',\  
 hover\_data = df2018.columns, \  
 title = "Distribution of C02 emission rates in 2018", labels={'Annual\_CO2\_emissions':'CO2 emission (billions of t)', 'count':'Nombre de pays'})  
fig.show()

Unable to display output for mime type(s): application/vnd.plotly.v1+json, text/html

CO2\_p\_hab = df2018['Annual\_CO2\_emissions']/df2018['Population']  
#df2018['CO2\_hab'] = CO2\_p\_hab  
fig = px.sunburst(df2018, path=['continent', 'Country'], color=CO2\_p\_hab,  
 hover\_data =df2018.columns, values='PIB\_per\_capita',  
 title = 'Emission en C02 par habitant, classement selon PIB par habitant')  
  
fig.show()

Unable to display output for mime type(s): application/vnd.plotly.v1+json, text/html

px.choropleth(df2018, locations="Code", color="Annual\_CO2\_emissions",  
 hover\_data=df2018.columns)

Unable to display output for mime type(s): application/vnd.plotly.v1+json, text/html

#df2018.sort\_values("CO2\_hab", ascending = False)  
df1 = df2018  
df1 = df1.loc[df1["Population"] >=5000000]  
df1.sort\_values("CO2\_hab", ascending = False)  
len(df1)

117

#selection des colonnes uniquement utilisées pour voir si ça marche mieux mais tester sans ça  
df\_all\_pd['Year'] = df\_all\_pd['Year'].astype('int')  
df\_all\_pd['Year'] = df\_all\_pd['Year'].astype('category')  
#df\_all\_pd = df\_all\_pd[["Country", "Year", "Annual\_CO2\_emissions"]]

df\_all\_pd.dtypes

Country object  
Code object  
Year category  
Annual\_CO2\_emissions int64  
Deaths int32  
PIB\_per\_capita float64  
Population int64  
Rate\_of\_natural\_population\_increase float64  
Net\_zero\_target float64  
dtype: object

len(df\_all\_pd)

10415

len(df\_all\_pd['Country'].unique())

163

## THEODORE

temp = df\_all.filter(df\_all.Year==2000).groupBy('continent').sum('Annual\_CO2\_emissions','Population').collect()  
  
lst\_all = []  
for annee in range(1950,2018,1):  
 lst\_temp = []  
 temp = df\_all.filter(df\_all.Year==annee).groupBy('continent').sum('Annual\_CO2\_emissions').collect()  
 lst\_temp.append(annee)  
 lst\_temp.append(temp[0][1])  
 lst\_temp.append(temp[1][1])  
 lst\_temp.append(temp[2][1])  
 lst\_temp.append(temp[3][1])  
 lst\_temp.append(temp[4][1])  
 lst\_all.append(lst\_temp)  
  
columns\_continent= ['Year','Europe','Afrique','Amérique','Oceanie','Asie']  
df\_continent = spark.createDataFrame(data=lst\_all, schema = columns\_continent)  
#pivot des données en pyspark  
df\_continent\_pd = df\_continent.withColumnRenamed("Amérique","Amerique")  
df\_continent\_pd = df\_continent\_pd.selectExpr(\  
 'Year', 'stack(5, "Europe", Europe, "Afrique", Afrique, "Amerique", Amerique, "Oceanie", Oceanie, "Asie", Asie) as (Continent, Annual\_CO2\_Emissions)')

df = df\_continent\_pd.toPandas()  
fig = px.line(df, x="Year", y='Annual\_CO2\_Emissions', color='Continent', title='CO2 emissions by continent since 1950',\  
 labels={'Annual\_CO2\_Emissions':'CO2 emission (billions of t)'})  
  
fig.show()

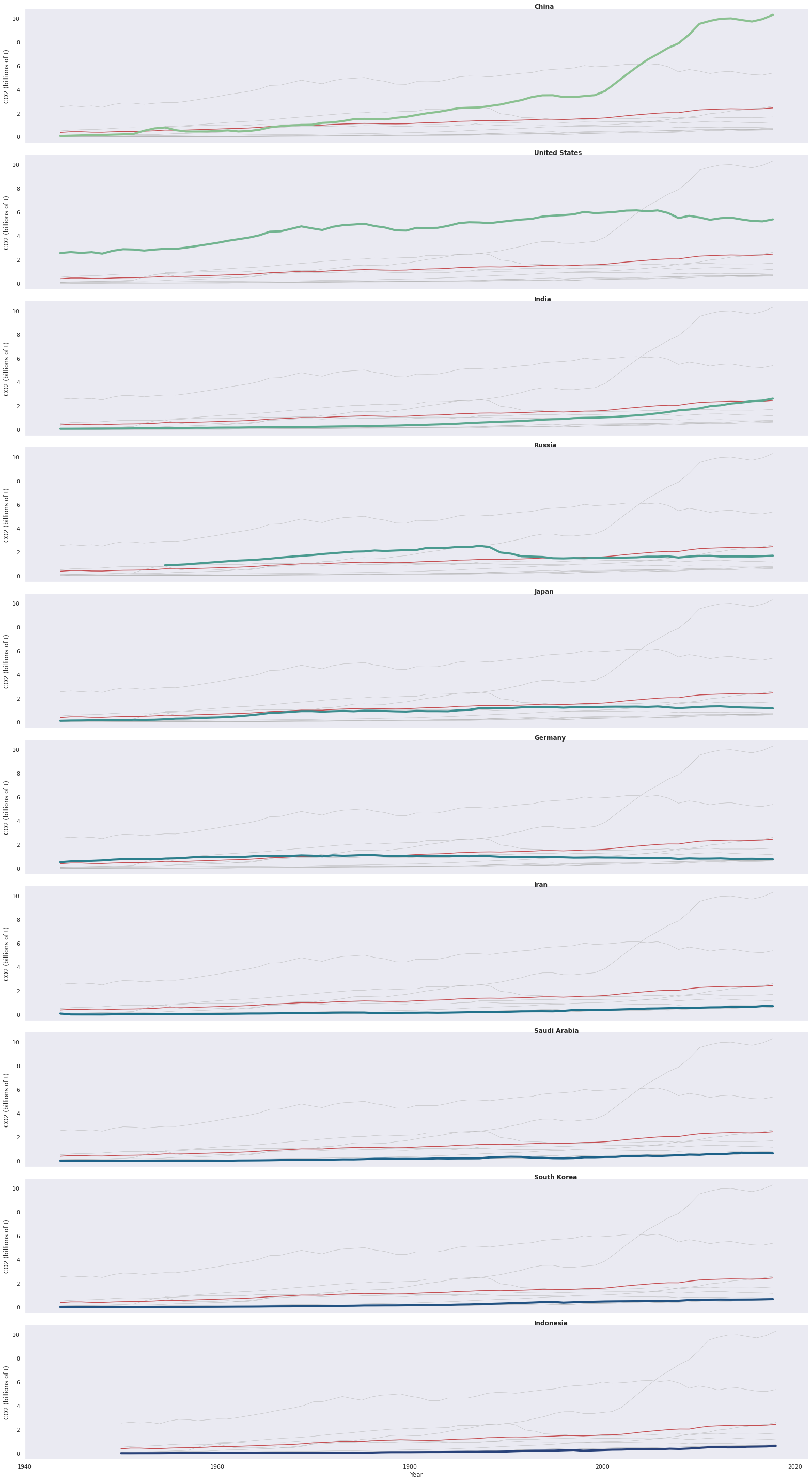
Unable to display output for mime type(s): application/vnd.plotly.v1+json, text/html

## Visu 2 = Emissions C02 pour les N pays les plus polluants

## Test visu sur nombre de pays limité avant d’appliquer sur all (à supprimer une fois all fonctionnel)

n = input("Please enter number n of most polluting countries...:\n") #nbr de pays les plus polluants à afficher  
n = int(n)  
  
#En pyspark, liste des n pays les plus émetteurs de C02 en 2018  
c\_list = df\_all.filter(df\_all.Year == 2018).orderBy(func.desc('Annual\_CO2\_emissions')).select(df\_all.Country).rdd.map(lambda x: x[0]).collect()  
c\_list = c\_list[:n]  
  
#En pyspark avant conversion pandas, sélectionner seulement les pays compris dans n  
df\_list\_pd = df\_all[df\_all.Country.isin(c\_list)].orderBy(func.desc('Annual\_CO2\_emissions')).select(df\_all.Country, df\_all.Year, (df\_all.Annual\_CO2\_emissions / 1000000000).alias("Annual\_CO2\_emissions")).toPandas()  
  
  
#####Partie plot  
  
df = df\_list\_pd  
  
sns.set\_theme(style="dark")  
  
# Plot each year's time series in its own facet  
g = sns.relplot(  
 data=df,  
 x="Year", y="Annual\_CO2\_emissions", col="Country", hue="Country",  
 kind="line", palette="crest", linewidth=4, zorder=5,  
 col\_wrap=1, height=4, aspect=5.5, legend=False  
)  
  
# Iterate over each subplot to customize further  
for country, ax in g.axes\_dict.items():  
  
 # Add the title as an annotation within the plot  
 ax.text(.65, 1, country, transform=ax.transAxes, fontweight="bold")  
  
 # Plot l'émission des n pays en fond  
 sns.lineplot(  
 data=df, x="Year", y="Annual\_CO2\_emissions", units="Country",  
 estimator=None, color="0.7", linewidth=0.5, ax=ax, legend='full'  
 )  
  
 #Plot la moyenne entre les n pays sélectionnés  
 sns.lineplot(  
 data=df, x="Year", y="Annual\_CO2\_emissions",  
 estimator=np.mean, color="r", linewidth=1.5, ax=ax, ci=None  
 )  
  
  
# Reduce the frequency of the x axis ticks  
ax.set\_xticks(ax.get\_xticks()[::2])  
  
# Tweak the supporting aspects of the plot  
g.set\_titles("")  
#g.set\_title(f"Emission en C02 depuis 1950 des {n} pays les plus polluants en 2018 \n \n \n")  
g.set\_axis\_labels("Year", "CO2 (billions of t)")  
g.tight\_layout()

Please enter number n of most polluting countries...:  
10



# Using graph\_objects  
import plotly.graph\_objects as go  
  
import pandas as pd  
df = pd.read\_csv('https://raw.githubusercontent.com/plotly/datasets/master/finance-charts-apple.csv')  
  
fig = go.Figure([go.Scatter(x=df['Date'], y=df['AAPL.High'])])  
fig.show()

Unable to display output for mime type(s): text/html

!pip install dash

Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/  
Collecting dash  
 Downloading dash-2.7.0-py3-none-any.whl (9.9 MB)  
 |████████████████████████████████| 9.9 MB 5.2 MB/s   
Requirement already satisfied: Flask>=1.0.4 in /usr/local/lib/python3.7/dist-packages (from dash) (1.1.4)  
Collecting dash-core-components==2.0.0  
 Downloading dash\_core\_components-2.0.0-py3-none-any.whl (3.8 kB)  
Requirement already satisfied: plotly>=5.0.0 in /usr/local/lib/python3.7/dist-packages (from dash) (5.5.0)  
Collecting dash-html-components==2.0.0  
 Downloading dash\_html\_components-2.0.0-py3-none-any.whl (4.1 kB)  
Collecting dash-table==5.0.0  
 Downloading dash\_table-5.0.0-py3-none-any.whl (3.9 kB)  
Requirement already satisfied: click<8.0,>=5.1 in /usr/local/lib/python3.7/dist-packages (from Flask>=1.0.4->dash) (7.1.2)  
Requirement already satisfied: Jinja2<3.0,>=2.10.1 in /usr/local/lib/python3.7/dist-packages (from Flask>=1.0.4->dash) (2.11.3)  
Requirement already satisfied: itsdangerous<2.0,>=0.24 in /usr/local/lib/python3.7/dist-packages (from Flask>=1.0.4->dash) (1.1.0)  
Requirement already satisfied: Werkzeug<2.0,>=0.15 in /usr/local/lib/python3.7/dist-packages (from Flask>=1.0.4->dash) (1.0.1)  
Requirement already satisfied: MarkupSafe>=0.23 in /usr/local/lib/python3.7/dist-packages (from Jinja2<3.0,>=2.10.1->Flask>=1.0.4->dash) (2.0.1)  
Requirement already satisfied: six in /usr/local/lib/python3.7/dist-packages (from plotly>=5.0.0->dash) (1.15.0)  
Requirement already satisfied: tenacity>=6.2.0 in /usr/local/lib/python3.7/dist-packages (from plotly>=5.0.0->dash) (8.1.0)  
Installing collected packages: dash-table, dash-html-components, dash-core-components, dash  
Successfully installed dash-2.7.0 dash-core-components-2.0.0 dash-html-components-2.0.0 dash-table-5.0.0