# Mudanças climáticas:

## Processos ETL e consultas a bancos de dados públicos

**Grupo 5**

Iris Belo Gassner

Lucas Brandão Gomes

Lucas Zewe Uriarte

Nara Geciauskas Ramos Castillo

Tema: Mudanças Climáticas

Orientação: Sayure Paiva, Bismark William

SoulCode Academy

Turma BCW6 – Engenharia de Dados

Documentação incluindo todos os processos e passos para a realização do projeto final desenvolvido pelo Grupo 5, Turma BCW6 de Engenharia de Dados da SoulCode Academy

Dezembro

2021

# SUMÁRIO

# 1. INTRODUÇÃO

A discussão acerca das Mudanças Climáticas tem ganhado cada vez mais espaço em diversas áreas do conhecimento, por conta de sua abrangência, complexidade, multidisciplinaridade e, em especial, urgência. As pesquisas visando o entendimento de suas causas, impactos e formas de mitigação representam um dos maiores desafios da ciência atual.

Através deste projeto buscamos contribuir com os estudos e avanços na área por meio da análise e disponibilização de informação e insights acerca das variações de temperatura, emissões de CO2 e desmatamento a nível global, obtidos através do tratamento, manipulação e consultas a bancos de dados disponíveis publicamente na internet.

Esta documentação está dividida em três capítulos. No primeiro capítulo trazemos uma introdução em relação ao tema de pesquisa: mudanças climáticas, a apresentação dos data sets utilizados, além de informações práticas acerca do projeto, como tecnologias utilizadas, acessos e fluxo de trabalho.

No segundo capítulo apresentamos o passo a passo dos processos de ETL, divididos por datasets. Utilizamos um ambiente do Google Collaboratory para a transformação e manipulação de cada data set, bem como um ambiente específico para o processo de criação de Pipelines.

No capítulo 3 apresentamos os processos de consultas e visualizações de dados, desenvolvidos em quatro frentes: Plotagem Pandas; consultas em SparkSQL; consultas em BigQuery e visualizações no Data Studio. Neste capítulo trazemos um relatório acerca dos insights provenientes das consultas e visualizações.

## 1.1 – CONCEITOS E PROBLEMÁTICA

**Clima** é regido por um conjunto integrado de fenômenos que se fundem no tempo e no espaço, revelando uma unidade ou tipo passíveis de serem medidos em seu tamanho (extensão) e em seu ritmo (duração).

**Fenômeno climático é** constituído por um conjunto de elementos de naturezas diversas, atuando simultaneamente, nos mesmos espaços, em regime de trocas energéticas recíprocas e interdependentes. (AS ESCALAS DO CLIMA, Giacomini A., Boletim de Geografia Teórica 1993).

Sabe-se que as variações de temperatura no planeta são comuns em escalas temporais e espaciais, sendo influenciadas por uma extensa gama de variáveis, incluindo fatores termodinâmicos, correntes marinhas e atmosféricas, emissão de gases e número de indivíduos/seres vivos coexistindo nos mesmos espaços. Apesar de normais, existem fortes indicativos de que as mudanças e variações de temperatura vem se intensificando no planeta Terra nos últimos 150 anos.

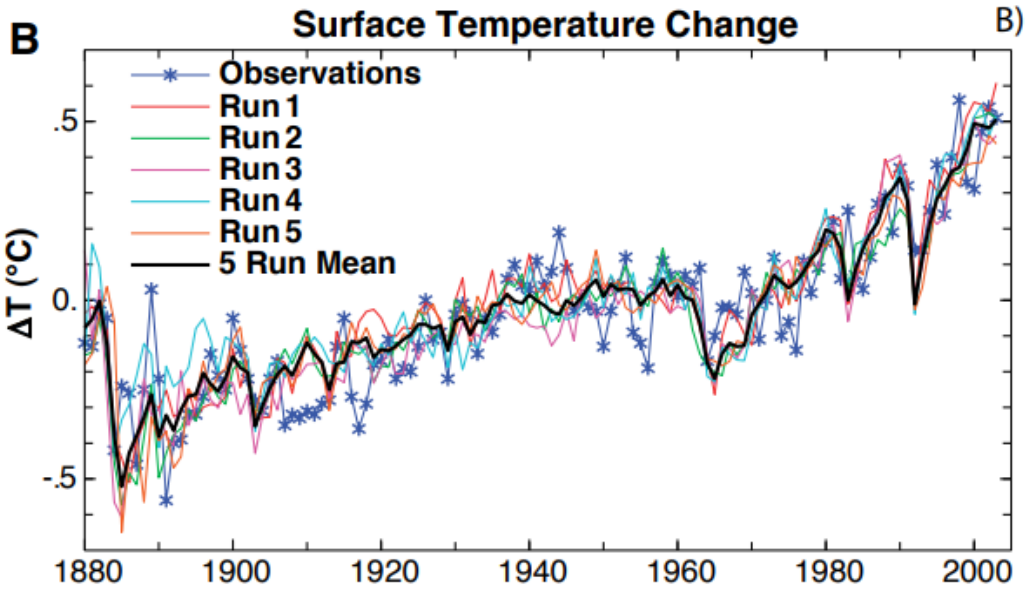



Figura 1: mudanças de temperatura na superfície do planeta Terra, observadas e simuladas
Disponível em: https://cetesb.sp.gov.br/wp-content/uploads/sites/36/2018/01/nobre_reid_veiga_fundamentos_2012.pdf

No gráfico acima é possível observar o comportamento da variação da temperatura na superfície do planeta, de modo que existem curvas que levam em consideração variáveis distintas, como por exemplo, a temperatura da superfície do oceano. Observa-se então a intensificação no aumento da temperatura média da terra no período posterior a 1900.

A emissão de gases de efeito estufa, GEE, é apontado como principal fator contribuinte para essa intensificação nas mudanças climáticas, sendo o gás carbônico, CO2, o maior responsável. O Painel Intergovernamental sobre Mudanças Climáticas (IPCC) afirma claramente em seu relatório de avaliação mais recente (AR5):

> As emissões antropogênicas de gases de efeito estufa aumentaram desde a era pré-industrial, impulsionadas em grande parte pelo crescimento econômico e populacional, e agora são maiores do que nunca. Isso levou a concentrações atmosféricas de dióxido de carbono, metano e óxido nitroso sem precedentes nos últimos 800.000 anos. **Seus efeitos, junto com os de outros fatores antropogênicos, foram detectados em todo o sistema climático e é extremamente provável que tenham sido a causa dominante do aquecimento observado desde meados do século XX.**

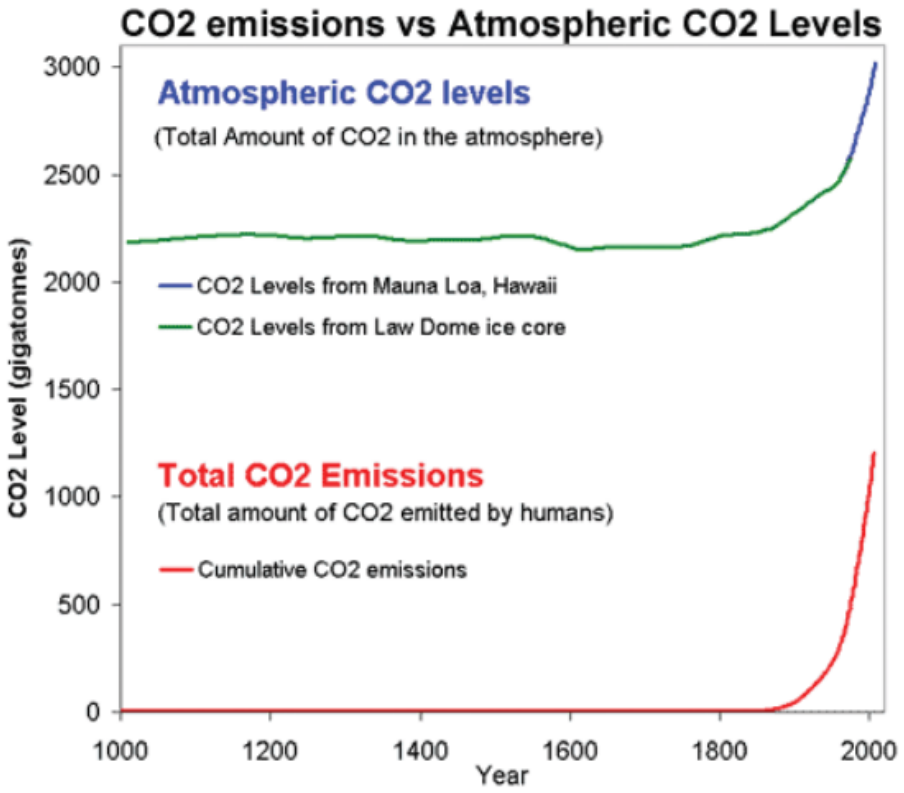



Figura 2: níveis de CO2 na atmosfera, acima em níveis totais considerando diferentes localidades, abaixo considerando somente emissões humanas em todo o planeta
Fonte: Cook, 2011.

No gráfico acima podemos observar duas curvas distintas representando os níveis de CO2 na atmosfera. A primeira curva, em verde e azul, expressa os níveis de CO2 na atmosfera nas localidades de Mauna Loa, no Havaí (azul) e Law Dome, no leste da Antártica (verde). Já na curva em vermelho, na parte de baixo do gráfico, observamos o total de emissões humanas de CO2. Ambos os níveis são expressos em gigatoneladas de CO2, correspondendo ao eixo Y, enquanto no eixo X temos uma linha do tempo avançando em um passo de 200 anos. Através deste gráfico podemos observar o grande impacto das atividades humanas nos últimos 150 anos em termos de emissão de CO2.

Estudos também sugerem que as concentrações de CO2 na atmosfera tem estrita relação com a variação da área de cobertura florestal no planeta. No gráfico abaixo vemos o comportamento das mudanças de área florestal de alguns países e também do mundo.

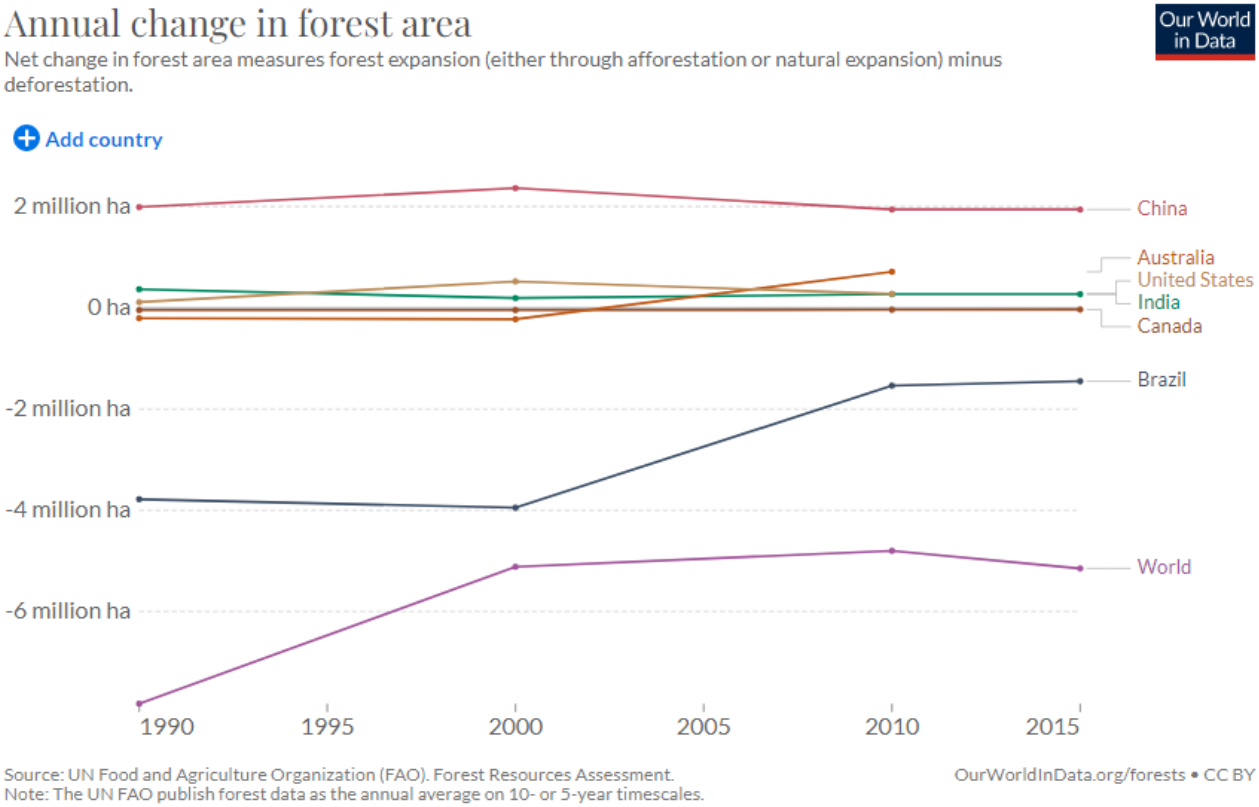



Figura 3: mudanças na área de cobertura florestal em países selecionados e em todo o mundo
Fonte: United Nations. Disponível em: https://ourworldindata.org/deforestation

## 1.2 – DATASETS UTILIZADOS

### DATASET 1 – TEMPERATURA MÉDIA

**Climate Change: Earth Surface Temperature Data - Exploring global temperatures since 1750**

Disponível em: https://www.kaggle.com/berkeleyearth/climate-change-earth-surface-temperature-data?select=GlobalLandTemperaturesByCountry.csv

Neste dataset contamos com diversas tabelas organizadas por Continentes, Países e maiores cidades do mundo. Optamos por utilizar somente a tabela referente aos países, propiciando um recorte interessante de todo o mundo, nem muito específico (como

nas medições por cidades), nem muito abrangente (medições por continente). A escolha pela tabela de países está alinhada aos outros datasets, que também trazem dados por países.

Os dados são resultado de uma compilação feita pela Berkeley Earth, do Lawrence Berkeley National Laboratory. São resultado da compilação de 1,6 bilhão de relatórios de temperatura e de 16 arquivos pré-existentes. Tendo em vista o objetivo do projeto, foi selecionado o banco de dados referente a média mensal de temperatura por país e por ano. As medições apresentam confiabilidade dentro do intervalo de confiança de 95%.

### DATASET 2 – EMISSÕES DE CO2

**CO2 emissions data - Data by country from 1750 to 2017**

Disponível em: https://www.kaggle.com/yoannboyere/co2-ghg-emissionsdata

Trata-se de um banco de dados em formato CSV, com registros anuais de CO2 por países e territórios a partir do ano de 1750. Este dataset é um compilado, tendo como fonte os dados coletados no portal "Our World in Data".

### DATASET 3 – COBERTURA FLORESTAL

**Global Forest Resources Assessment**

Disponível em: https://fra-data.fao.org/WO/fra2020/forestAreaChange/

Este dataset apresenta dados referentes a variações de cobertura vegetal no planeta, no período de 1985 a 2020. No site da FAO, Food and Agriculture Organization of the United Nations, é possível obter tais dados de acordo com aplicação de filtros desejados.

A FAO tem monitorado as florestas do mundo em intervalos de 5 a 10 anos desde 1946. As Avaliações de Recursos Florestais Globais (FRA) mais recentes são realizadas a cada cinco anos, fornecendo uma abordagem consistente para descrever as florestas do mundo e suas mudanças. A avaliação é baseada em duas fontes principais de dados: relatórios de países preparados por correspondentes nacionais e sensoriamento remoto.

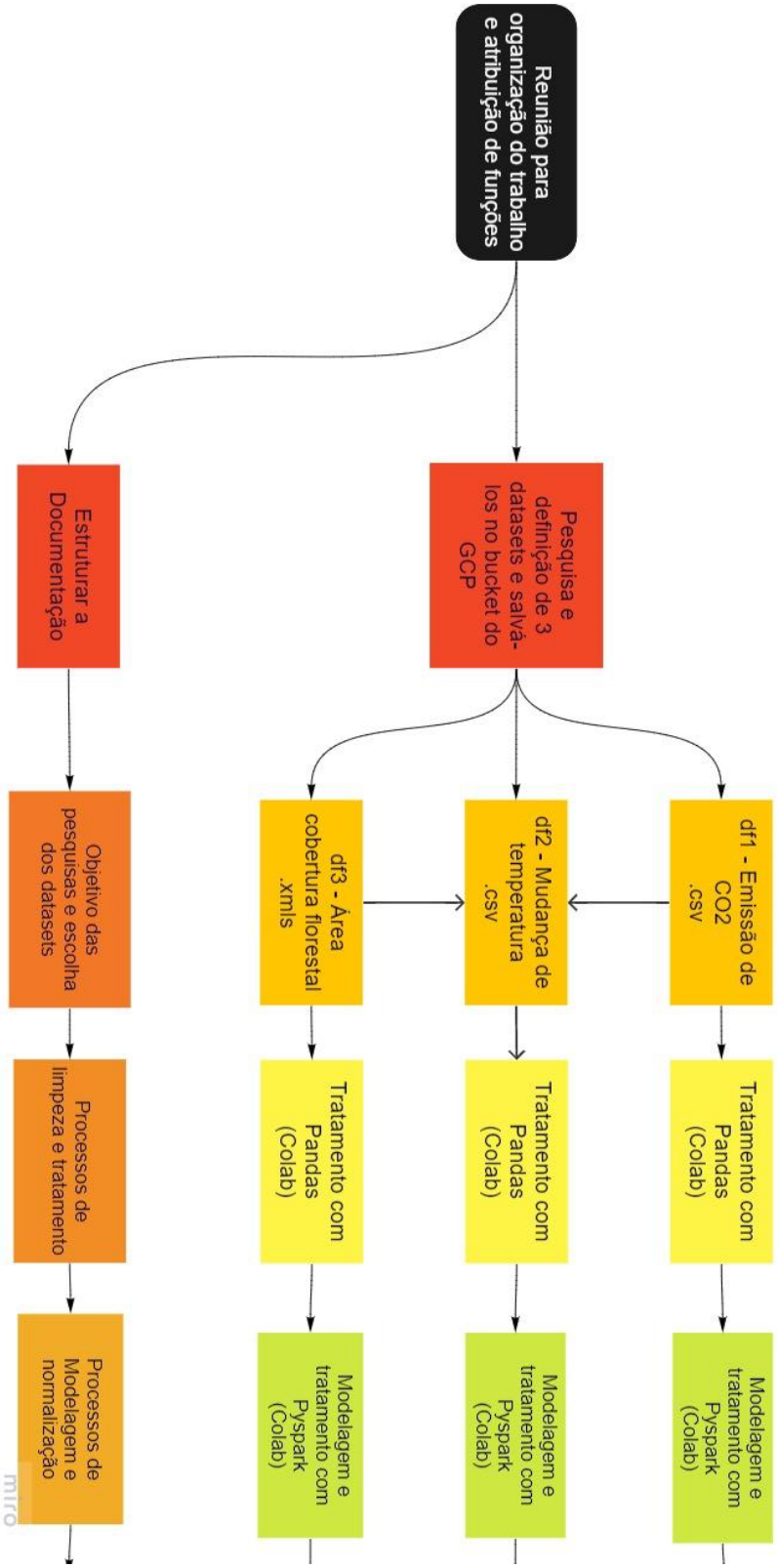
## 1.3 – ORGANIZAÇÃO E FERRAMENTAS

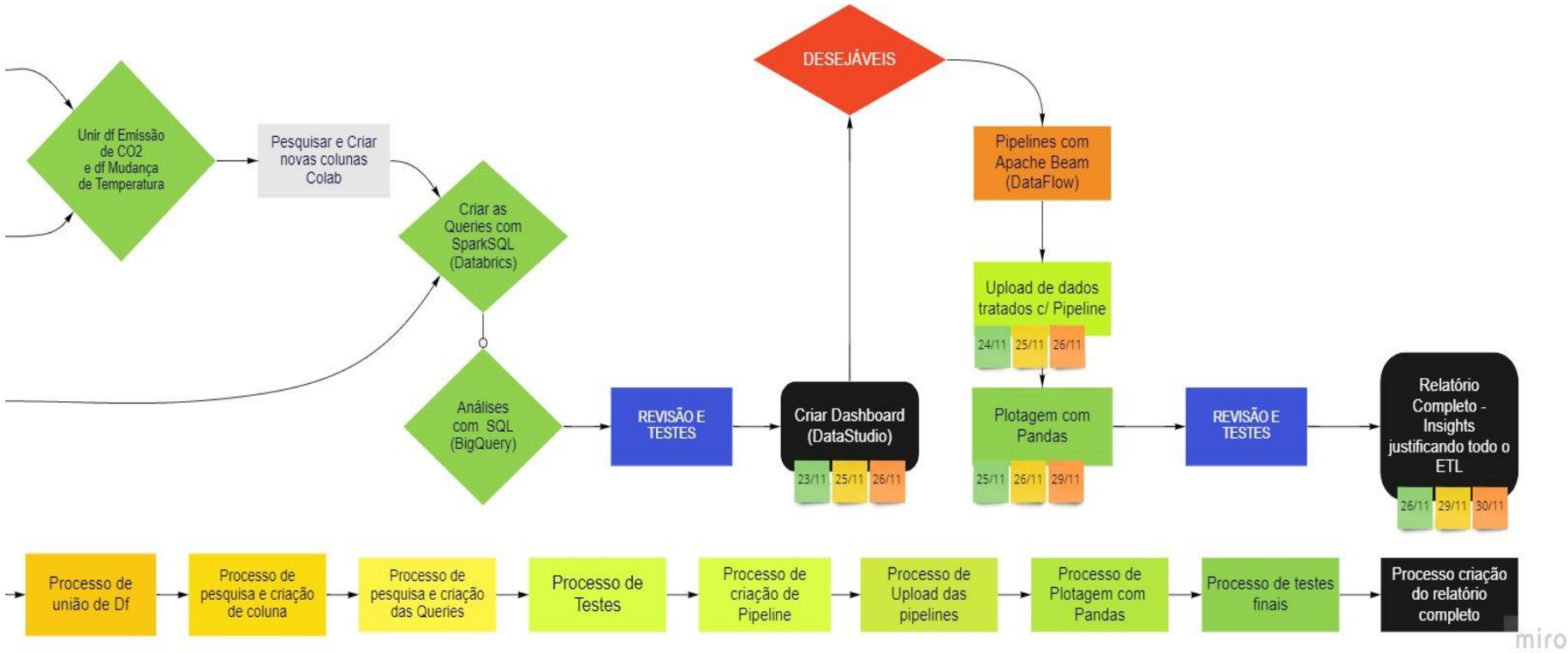
Organizamos o projeto de forma que várias frentes de trabalho pudessem ser executadas simultaneamente. Cada dataset foi tratado em um ambiente Colab próprio, por um ou dois integrantes, em paralelo. Também em paralelo fomos alimentando a documentação ao longo de toda a duração do projeto.

A parte de consultas e visualizações foi desenvolvida por diferentes integrantes ao longo do trabalho, contudo, para sua finalização, todos se concentraram nesse ponto, discutindo os insights e decidindo o que seria apresentado. Vale ainda ressaltar que os requisitos obrigatórios do trabalho foram priorizados, e os requisitos desejáveis foram contemplados posteriormente.

Foram utilizadas diversas ferramentas para organizar o fluxo de trabalho, bem como para o desenvolvimento dos processos ETL, das consultas e visualizações. Neste item vamos apresentar e trazer algumas informações práticas acerca de tais ferramentas, iniciando pelo diagrama de fluxo de trabalho, desenhado no software Miro.

**FLUXO DE TRABALHO**




Unir df Emissão de CO2 e df Mudança de Temperatura
Pesquisar e Criar novas colunas Colab
Criar as Queries com SparkSQL (Databrics)
Análises com SQL (BigQuery)
REVISÃO E TESTES
Criar Dashboard (DataStudio)
23/11
25/11
26/11
DESEJÁVEIS
Pipelines com Apache Beam (DataFlow)
Upload de dados tratados c/ Pipeline
24/11
25/11
26/11
Plotagem com Pandas
25/11
26/11
29/11
REVISÃO E TESTES
Relatório Completo - Insights justificando todo o ETL
26/11
29/11
30/11
Processo de união de Df
Processo de pesquisa e criação de coluna
Processo de pesquisa e criação das Queries
Processo de Testes
Processo de criação de Pipeline
Processo de Upload das pipelines
Processo de Plotagem com Pandas
Processo de testes finais
Processo criação do relatório completo
miro

## FERRAMENTAS E TECNOLOGIAS UTILIZADAS

### Organizacional

- Trello: Planejamento e gestão do projeto
- Google Meet: Plataforma para reuniões diárias.
- Google drive: armazenamento de arquivos

### Manipulação, tratamento e consultas aos bancos de dados:

- Google Collaboratory
    - Linguagem Python
    - Biblioteca Pandas: tratamento primário de dataset
    - Biblioteca PySpark: tratamento secundário e modelagem de dataset
    - sparkSQL: consultas SQL
    - ApacheBeam: criação de Pipeline
- Google Cloud Platform
    - Cloud Storage: armazenamento de arquivos necessários ao projeto
    - DataPrep API: visualização de dados
    - BigQuery: consultas SQL
    - DataFlow API: integração do Pipeline
    - DataStudio: visualização gráfica de dados e interpretações

## INFORMAÇÕES E ACESSOS

Para a realização deste trabalho criamos um projeto no Google Cloud Platform (GCP), chamado **ProjetoFinal-Grupo5**. O acesso ao projeto foi garantido a todos os alunos do grupo, bem como aos professores da SoulCode através de permissões do IAM.

### Projeto Google Cloud Platform

ID Projeto: projetofinal-grupo5

E-mails com permissão para acesso (donos e editores):

- lucaszeweuriarte@gmail.com – dono
- nara.castillo@usp.br – dono
- na.geciauskas@gmail.com – editor
- irisbelogassner@gmail.com – editor
- brandslucas@gmail.com – editor
- professores.bcw4@soulcodeacademy.org – editor

**Google Cloud Storage (GCS):** criamos um disco (*bucket*) vinculado ao projeto, para armazenar arquivos de entrada e saída, além de outros documentos. O acesso ao *bucket* foi garantido aos mesmos usuários através das configurações de permissão do próprio Google Cloud Storage. Já para conseguirmos acessar o bucket através do Google Colaboratory, criamos, nas opções do IAM, uma conta de serviço e uma chave vinculada.

Link para acesso ao bucket (para usuários cadastrados):
https://console.cloud.google.com/storage/browser/projeto-final-bucket-g5

O bucket conta com cinco pastas:

- A pasta **documentos** contém a chave de acesso da conta de serviço, a planilha de organização e programação do grupo, o arquivo de texto com a documentação do projeto, o arquivo com os slides utilizados na apresentação do projeto.
- A pasta **entrada** contém os datasets de temperatura, emissão de CO2, variação de cobertura florestal sem nenhum tratamento.
- A pasta **saida** contém os outputs dos dataframes tratados.
- As pastas **temp** e **template** foram criadas para utilização durante a execução da Pipeline, desenvolvida utilizando a tecnologia Apache Beam no Google Colaboratory.

**Links para notebooks Google Colaboratory**

Dividimos o trabalho de tratamento, manipulação e consultas aos datasets em diversos notebooks do Google Colaboratory. Os notebooks foram compartilhados com

todos os integrantes do grupo, bem como com os professores, podendo ser acessados pelos usuários autorizados através dos seguintes links:

- Colab para tratamento e manipulação do Dataset 1 – Temperatura média: https://colab.research.google.com/drive/1ynq9_awC8qnojinVTVkZM9gAIY-Dpdrg?usp=sharing
- Colab para tratamento e manipulação do Dataset 2 – Emissões de CO2: https://colab.research.google.com/drive/1DUxagxrfFRcUkv0jxwvhi64BDNMfVyuA?usp=sharing
- Colab para tratamento e manipulação do Dataset 3 – Cobertura florestal: https://colab.research.google.com/drive/1nCC8oEjTXl6BnzBx3TG5AVSkjoyG_TQF?usp=sharing
- Colab para criação, tratamento e manipulação do Dataset 4 – resultado da junção dos datasets 1 e 2 (temperatura e emissão CO2) + Plotagem de Pandas: https://colab.research.google.com/drive/177rpXxgOjEj-i9fSKzp87YG6MQQlsDOg?hl=pt-BR
- Colab para criação de Pipeline: https://colab.research.google.com/drive/1sN0I21I0DAxPZTKUMZuzAVNLN5LgWUHZ?usp=sharing
- Colab para consultas ao Dataset 4 utilizando o SparkSQL: https://colab.research.google.com/drive/1XmPRAq4NRz81AjVCBhTSo-erKThSLjv1?usp=sharing

**Data Studio**

Criamos um projeto no software Data Studio para visualizarmos consultas de BigQuery. Apenas algumas consultas foram escolhidas para esse fim, e os gráficos resultantes foram utilizados para apresentação do projeto. O projeto Data Studio está disponível através do link: https://datastudio.google.com/reporting/f92e1b36-ae3e-4482-834c-d46eae389874/page/p_cvrie86tpc

# 2. PROCESSOS ETL

## 2.1 – MÉTODO DE TRABALHO COMUM A TODOS OS DATASETS

Para todos os datasets tratados seguimos um mesmo roteiro, iniciando pela importação do dataset sem tratamento diretamente do *bucket* GCP, a partir do qual criamos um dataframe utilizando a biblioteca Pandas. Ainda com esta biblioteca, procedemos a tratamentos e verificação da integridade dos dados. Uma vez concluída esta etapa, criamos uma verificação através da biblioteca pandera, a fim de garantir que todos os dados correspondem aos tipos desejados.

Após a verificação com Pandera, criamos uma sessão Spark e um dataframe de PySpark a partir do dataframe já tratado em Pandas. Utilizando a ferramenta PySpark realizamos algumas modificações e modelagens mais profundas, como a criação de novas colunas, deixando o dataset pronto para as consultas SQL. Como última etapa, utilizamos uma função que permite carregar os dataframes como arquivos CSV diretamente para o *bucket* GCP.

Para ter acesso ao bucket através do Colab e instalar as bibliotecas necessárias, utilizamos os seguintes comandos em todos os Colabs:

- Instalação do pacote gcsfs para acesso de arquivos direto do GCS.

```
pip install gcsfs
```

- Instalação do pacote Pandera para validação de dados.

```
pip install pandera
```

- Importação de biblioteca para autenticação da conta de serviço GCP.

```
import os
```

- Criação de variável com o caminho para a chave relacionada à conta de serviço associada ao projeto. Através da biblioteca os, o arquivo de chave da conta de serviço é atribuído a variável serviceAccount. A chave de serviço em formato json foi carregada localmente no Google Colaboratory. Essa parte é essencial para que seja possível acessar o *bucket* diretamente do Colab.

```
serviceAccount ='/content/chave_projetofinal-grupo5-d2c4a9d78233.json'
os.environ['GOOGLE_APPLICATION_CREDENTIALS'] = serviceAccount
```

- Importação de bibliotecas a serem utilizadas na etapa de tratamento Pandas / Pandera.

```
import pandas as pd
import pandera as pa
from google.cloud import storage
```

- Instalação do pacote PySpark.

```
pip install pyspark
```

- Importação de módulos e de funções específicas do framework PySpark.

```
from pyspark.sql import SparkSession
# SparkSession é uma string de conexão, na qual serão definidos
# os parâmetros necessários para a manipulação do dataframe com PySpark

import pyspark.sql.functions as F
# importação da biblioteca de funções do pacote spark atribuindo F para o
# uso durante o código

from pyspark.sql.types import StructType,StructField, StringType,
IntegerType, FloatType, DoubleType, DateType, ArrayType, DoubleType,
BooleanType
# importação dos tipos de variáveis para futura estruturação e validação

from pyspark.sql.window import Window
# importação de funções do tipo window
```

- Criação de uma Spark Session.

```
spark = SparkSession.builder\
      .master("local")\
      #master: local, cluester, ou local onde irá rodar a sessão
      .appName("dataframe_temperatura")\
      #identificaçao do aplicação a ser criad
      .config("spark.ui.port","4050")\
      #configurção via API com apache spark, e definção da porta UI, que no
caso do colab é 4050
      .getOrCreate()
      #iniciar a sessão
```

### VERIFICAÇÃO DE PAÍSES

Desde o início do trabalho consideramos juntar pelo menos dois datasets, formando um banco de dados mais robusto para consultas. Levando-se em conta que todos os datasets contam com dados organizados por países e por anos, resolvemos justamente utilizar estas informações como referência para a junção. Sendo assim, fez-se necessária logo de início uma verificação, em cada dataset, dos países considerados e da grafia de seu nome.

A fim de padronizar as grafias e posteriormente classificar os países em relação aos continentes, regiões e níveis de desenvolvimento, utilizamos como referência dados disponíveis na página das Nações Unidas[1]. Após comparar os três datasets, verificar quais países apresentavam dados consistentes e corrigir problemas de grafia, chegamos à lista final de países considerada em nosso trabalho:

['Afghanistan', 'Albania', 'Algeria', 'Andorra', 'Angola', 'Anguilla', 'Antigua and Barbuda ', 'Argentina', 'Armenia', 'Aruba', 'Australia', 'Austria', 'Azerbaijan', 'Bahamas', 'Bahrain', 'Bangladesh', 'Barbados', 'Belarus', 'Belgium', 'Belize', 'Benin', 'Bhutan', 'Bolivia', 'Bosnia and Herzegovina', 'Botswana', 'Brazil', 'British Virgin Islands', 'Bulgaria', 'Burkina Faso', 'Myanmar', 'Burundi', "Cote d'Ivoire", 'Cambodia', 'Cameroon', 'Canada', 'Cape Verde', 'Cayman Islands', 'Central African Republic', 'Chad', 'Chile', 'China', 'Christmas Island', 'Colombia', 'Comoros', 'Democratic Republic of the Congo', 'Congo', 'Costa Rica', 'Croatia', 'Cuba', 'Curacao', 'Cyprus', 'Czech Republic', 'Denmark', 'Djibouti', 'Dominica', 'Dominican

[1] Dados disponíveis em https://unstats.un.org/unsd/methodology/m49/.

Republic', 'Ecuador', 'Egypt', 'El Salvador', 'Equatorial Guinea', 'Eritrea', 'Estonia', 'Ethiopia', 'Islas Malvinas', 'Faroe Islands', 'Micronesia', 'Fiji', 'Finland', 'France', 'French Guiana', 'French Polynesia', 'Gabon', 'Gambia', 'Georgia', 'Germany', 'Ghana', 'Greece', 'Greenland', 'Grenada', 'Guadeloupe', 'Guatemala', 'Guinea-Bissau', 'Guinea', 'Guyana', 'Haiti', 'Honduras', 'Hong Kong', 'Hungary', 'Iceland', 'India', 'Indonesia', 'Iran', 'Iraq', 'Ireland', 'Israel', 'Italy', 'Jamaica', 'Japan', 'Jordan', 'Kazakhstan', 'Kenya', 'Kiribati', 'Kuwait', 'Kyrgyzstan', 'Laos', 'Latvia', 'Lebanon', 'Lesotho', 'Liberia', 'Libya', 'Liechtenstein', 'Lithuania', 'Luxembourg', 'Macao', 'Macedonia', 'Madagascar', 'Malawi', 'Malaysia', 'Mali', 'Malta', 'Martinique', 'Mauritania', 'Mauritius', 'Mexico', 'Moldova', 'Mongolia', 'Montenegro', 'Montserrat', 'Morocco', 'Mozambique', 'Namibia', 'Nepal', 'Netherlands', 'New Caledonia', 'New Zealand', 'Nicaragua', 'Niger', 'Nigeria', 'Niue', 'North Korea', 'Norway', 'Oman', 'Pakistan', 'Palau', 'Palestine', 'Panama', 'Papua New Guinea', 'Paraguay', 'Peru', 'Philippines', 'Poland', 'Portugal', 'Qatar', 'Reunion', 'Romania', 'Russia', 'Rwanda', 'Saint Kitts and Nevis', 'Saint Lucia', 'Saint Vincent and the Grenadines', 'Samoa', 'Sao Tome And Principe', 'Saudi Arabia', 'Senegal', 'Serbia', 'Seychelles', 'Sierra Leone', 'Singapore', 'Slovakia', 'Slovenia', 'Solomon Islands', 'Somalia', 'South Africa', 'South Korea', 'Spain', 'Sri Lanka', 'Sudan', 'Suriname', 'Swaziland', 'Sweden', 'Switzerland', 'Syria', 'Taiwan', 'Tajikistan', 'Tanzania', 'Thailand', 'Timor Leste', 'Togo', 'Tonga', 'Trinidad and Tobago', 'Tunisia', 'Turkey', 'Turkmenistan', 'Turks and Caicos Islands', 'Uganda', 'Ukraine', 'United Arab Emirates', 'United Kingdom', 'United States', 'Uruguay', 'Uzbekistan', 'Venezuela', 'Vietnam', 'Yemen', 'Zambia', 'Zimbabwe']

## 2.2 – DATASET 1 – Temperatura média

### PANDAS E PANDERA

Após a importação de bibliotecas e configuração de chaves, procedemos aos seguintes passos com as bibliotecas Pandas e Pandera:

(a) Fizemos o download do dataset sem tratamento, diretamente do bucket, e criamos um dataframe de Pandas.

```
client = storage.Client()
bucket = client.get_bucket('projeto-final-bucket-g5')
#a variável bucket recebe o nome identico do bucket do cloud storage
bucket.blob('Temperatura_media_pais.csv')
#a variável bucket_blob recebe o nome do arquivo
```

```
path = 'gs://projeto-final-bucket-g5/entrada/Temperatura_media_pais.csv'
# a variável path recebe o caminho do arquivo (URI) do arquivo
df_orig = pd.read_csv(path, parse_dates=['dt'])
#leitura do arquivo e método parse-date que pega uma string de data
formatada customizada e a converte em uma string de dados formatada padrão
do esquema XML (YYYY-MM-DD)

df_orig
```

(b) Verificamos os tipos dos dados em cada coluna.

```
df_orig.dtypes
```

Saída do código:

```
dt                                datetime64[ns]
AverageTemperature                       float64
AverageTemperatureUncertainty            float64
Country                                   object
dtype: object
```

(c) Criamos um backup do dataframe.

```
df = df_orig.copy()

# a partir daqui, utilizamos o df e deixamos o df_orig como backup
```

(d) Eliminamos dados referentes a países que não estão de acordo com a lista adotada oficialmente, deixando somente países que apresentam dados em todos os datasets. Alguns dados de regiões e continentes também foram deletados.

```
# O código funciona da seguinte maneira: excluir todas as linhas nas quais
# a coluna Country apresentar o dado X

df.drop(df[(df['Country'] == 'Åland')].index, inplace=True)
df.drop(df[(df['Country'] == 'Africa')].index, inplace=True)
df.drop(df[(df['Country'] == 'American Samoa')].index, inplace=True)
df.drop(df[(df['Country'] == 'Asia')].index, inplace=True)
df.drop(df[(df['Country'] == 'Baker Island')].index, inplace=True)
df.drop(df[(df['Country'] == 'Bonaire, Saint Eustatius And Saba')].index,
inplace=True)
df.drop(df[(df['Country'] == 'Denmark')].index, inplace=True)
df.drop(df[(df['Country'] == 'Europa')].index, inplace=True)
df.drop(df[(df['Country'] == 'France')].index, inplace=True)
df.drop(df[(df['Country'] == 'French Southern And Antarctic Lands')].index,
inplace=True)
df.drop(df[(df['Country'] == 'Gaza Strip')].index, inplace=True)
```

```
df.drop(df[(df['Country'] == 'Guam')].index, inplace=True)
df.drop(df[(df['Country'] == 'Guernsey')].index, inplace=True)
df.drop(df[(df['Country'] == 'Heard Island And Mcdonald Islands')].index,
inplace=True)
df.drop(df[(df['Country'] == 'Isle Of Man')].index, inplace=True)
df.drop(df[(df['Country'] == 'Jersey')].index, inplace=True)
df.drop(df[(df['Country'] == 'Kingman Reef')].index, inplace=True)
df.drop(df[(df['Country'] == 'Mayotte')].index, inplace=True)
df.drop(df[(df['Country'] == 'Monaco')].index, inplace=True)
df.drop(df[(df['Country'] == 'Netherlands')].index, inplace=True)
df.drop(df[(df['Country'] == 'North America')].index, inplace=True)
df.drop(df[(df['Country'] == 'Northern Mariana Islands')].index,
inplace=True)
df.drop(df[(df['Country'] == 'Oceania')].index, inplace=True)
df.drop(df[(df['Country'] == 'Palmyra Atoll')].index, inplace=True)
df.drop(df[(df['Country'] == 'Puerto Rico')].index, inplace=True)
df.drop(df[(df['Country'] == 'Saint Barthélemy')].index, inplace=True)
df.drop(df[(df['Country'] == 'Saint Martin')].index, inplace=True)
df.drop(df[(df['Country'] == 'Saint Pierre And Miquelon')].index,
inplace=True)
df.drop(df[(df['Country'] == 'San Marino')].index, inplace=True)
df.drop(df[(df['Country'] == 'Sint Maarten')].index, inplace=True)
df.drop(df[(df['Country'] == 'South America')].index, inplace=True)
df.drop(df[(df['Country'] == 'South Georgia And The South Sandwich
Isla')].index, inplace=True)
df.drop(df[(df['Country'] == 'Svalbard And Jan Mayen')].index, inplace=True)
df.drop(df[(df['Country'] == 'United Kingdom')].index, inplace=True)
df.drop(df[(df['Country'] == 'Virgin Islands')].index, inplace=True)
df.drop(df[(df['Country'] == 'Western Sahara')].index, inplace=True)
```

(e) Renomeamos países de acordo com os nomes da lista oficial.

```
# Utilizando o comando loc, vamos procurar na coluna país por determinado
# nome e na sequência alterar esse nome na mesma coluna

df.loc[df.Country == 'Burma' , ['Country']] = 'Myanmar'
df.loc[df.Country == "Côte D'Ivoire" , ['Country']] = "Cote d'Ivoire"
df.loc[df.Country == 'Curaçao' , ['Country']] = 'Curacao'
df.loc[df.Country == 'Denmark (Europe)' , ['Country']] = 'Denmark'
df.loc[df.Country == 'Falkland Islands (Islas Malvinas)' , ['Country']] =
'Islas Malvinas'
df.loc[df.Country == 'Federated States Of Micronesia' , ['Country']] =
'Micronesia'
df.loc[df.Country == 'France (Europe)' , ['Country']] = 'France'
df.loc[df.Country == 'Netherlands (Europe)' , ['Country']] = 'Netherlands'
df.loc[df.Country == 'United Kingdom (Europe)' , ['Country']] = 'United
Kingdom'
df.loc[df.Country == 'Turks And Caicas Islands' , ['Country']] = 'Turks And
Caicos Islands'
```

(f) Traduzimos os nomes das colunas de inglês para português.

```
# Renomear colunas utilizando uma sintaxe de dicionário:
# Nome antigo: Nome novo

(
df.rename(columns={'dt': 'Data', 'AverageTemperature': 'TemperaturaMedia',
'AverageTemperatureUncertainty': 'CertitudeTempMedia', 'Country': 'País'},
inplace=True)
)
```

(g) Após uma segunda verificação, alteramos mais alguns nomes de países que escaparam à primeira verificação.

```
df.loc[df.País == 'Turks And Caicos Islands', ['País']] = 'Turks and Caicos
Islands'
df.loc[df.País == 'Trinidad And Tobago', ['País']] = 'Trinidad and Tobago'
df.loc[df.País == 'Saint Vincent And The Grenadines', ['País']] = 'Saint
Vincent and the Grenadines'
df.loc[df.País == 'Saint Kitts And Nevis', ['País']] = 'Saint Kitts and Nevis'
df.loc[df.País == 'Palestina', ['País']] = 'Palestine'
df.loc[df.País == 'Antigua And Barbuda', ['País']] = ' Antigua and Barbuda '
df.loc[df.País == 'Bosnia And Herzegovina', ['País']] = 'Bosnia and
Herzegovina'
df.loc[df.País == 'Congo (Democratic Republic Of The)', ['País']] =
'Democratic Republic of the Congo'
df.loc[df.País == 'Guinea Bissau', ['País']] = 'Guinea-Bissau'
df.loc[df.País == 'Macau', ['País']] = 'Macao'
```

(h) Verificamos os dados nulos em todo o dataset.

```
df.isnull().sum()
```

Saída do código

```
Data                        0
TemperaturaMedia        28253
CertitudeTempMedia      27514
País                        0
dtype: int64
```

Uma verificação mais detalhada dos dados nulos foi realizada posteriormente, utilizando a ferramenta DataPrep, disponível no Google Cloud Platform. Falaremos mais sobre isso no próximo item.

(i) Após essas modificações, criamos e executamos a validação dos dados utilizando o Pandera.

```
schema = pa.DataFrameSchema(
    columns = {
        "Data": pa.Column(pa.DateTime,nullable=False),
        "TemperaturaMedia": pa.Column(pa.Float,nullable=True),
        "CertitudeTempMedia": pa.Column(pa.Float,nullable=True),
        "País": pa.Column(pa.String,nullable=False)})

schema.validate(df)
```

(j) Carregamos o dataframe tratado em Pandas no formato CSV diretamente para o Google Cloud Storage.

```
client=storage.Client()
bucket=client.get_bucket('projeto-final-bucket-g5')
bucket.blob('saida/df_temp_media_pandas_tratamento_final.csv').upload_from_s
tring(df.to_csv(index=False), 'text/cvs')
```

**PYSPARK**

Após a instalação do PySpark e criação da SparkSession, procedemos à criação do dataframe de PySpark.

(a) Criamos o dataframe de PySpark utilizando o StructType para criar o esquema e configuramos o dataframe de Pandas como fonte de dados

```
esquema = (
   StructType()
     .add("Data",DateType(),True)
     .add("TemperaturaMedia",FloatType(),True)
     .add("CertitudeTempMedia",FloatType(),True)
     .add("País",StringType(),True)
)

df_spark = spark.createDataFrame(df, schema = esquema)
```

(b) Criamos um backup do dataframe PySpark.

```
# backup do DF original de PySpark
df_spark_backup = df_spark.alias('df_spark_backup')
```

(c) Criamos a coluna ‘Ano’ a partir da coluna ‘Data’, utilizando a função substring que permite seccionar a data em ano, mês e dia.

```
# Utilizamos a função withColumn para criar a coluna 'Ano' a partir da coluna
# ‘Data’, utilizando a posição dos valores a serem obtidos. Para tal usamos
# a função substring. Com a função cast, definimos o tipo de dado que a nova
# coluna irá receber
df_spark = df_spark.withColumn('Ano', F.substring(F.col('Data'), 1,
4).cast('Integer'))
```

(d) Após criarmos a coluna ‘Ano’, eliminamos a coluna ‘Data’, já que não utilizaríamos mais as informações de mês e dia. Falaremos mais sobre isso no passo (f) logo abaixo.

```
df_spark = df_spark.drop('Data')
```

(e) Após uma verificação no DataPrep dos dataframes 1 e 2 tratados em Pandas, chegamos à conclusão de que a maior parte dos nossos dados nulos eram de medições de temperatura (também de emissão de CO2, no caso do dataframe 2) dos séculos XVIII e XIX. Sendo assim, optamos por considerar somente os dados aferidos a partir de 1900 em ambos os dataframes. Como data mais recente, ficamos com o ano de 2013, último ano com dados coletados no dataset de temperatura.

```
df_spark = df_spark.where(df_spark.Ano > 1899)
```

(f) Eliminamos as linhas referentes ao território da Antártica, ao percebermos que os dados ainda estavam muito inconsistentes nessa região

```
df_spark = df_spark.where(df_spark.País != 'Antarctica')
```

(g) Calculamos a temperatura média anual a partir das medições de temperatura média mensal apresentadas no dataset original. Para tal, utilizamos um comando que conta com funções agregadas, funcionando da seguinte maneira: os dados são agrupados (groupBy) por países e por ano, sendo que os valores de

temperatura média mensal de cada país e de cada ano são submetidos a um cálculo de média (avg).

Importante salientar que tivemos o cuidado de testar esse método "à mão", somando as temperaturas mensais de alguns países e dividindo por 12 (meses). Fizemos também alguns testes com países que apresentavam dados nulos em determinados meses, descobrindo que estes são desconsiderados no cálculo de média. Exemplo: se um país tem somente dez medições de temperatura em um ano, sendo que em dois meses os valores de temperatura são nulos, o comando utilizado soma somente os dez valores válidos e divide por dez, dando a média mais fiel possível.

Utilizamos ainda uma função para ordenar (orderBy) o resultado pelo nome dos países em ordem alfabética e ano em ordem crescente.

Por fim, com esse comando ainda desconsideramos a coluna 'CertitudeTempMedia', presente até então no dataframe.

```
# Transformar as médias de temperatura mensais em médias anuais

df_spark2 = (
   df_spark.groupBy(F.col('País'), F.col('Ano')).avg('TemperaturaMedia')
           .orderBy('País', 'Ano')
)
```

(h) Renomeamos a coluna 'avg(TemperaturaMedia)', criada no passo anterior, para 'TemperaturaMediaAno'

```
df_spark2 = df_spark2.withColumnRenamed('avg(TemperaturaMedia)',
                                        'TemperaturaMediaAno')
```

(i) Após a operação de criação de médias anuais de temperatura, eliminamos também todas as linhas cujo valor na coluna 'Ano' correspondia a 2013, tendo em visto que nestas linhas o valor na coluna 'TemperaturaMediaAno' sempre correspondia a NaN. Portanto, nosso dataframe passou a contar com temperaturas médias anuais de 1900 a 2012.

```
df_spark2 = df_spark2.where(df_spark.Ano != 2013)
```

(j) Criamos uma nova coluna, chamada 'DiferencaTemp', que corresponde à diferença da temperatura média de um ano em relação ao ano anterior, no mesmo país. Utilizamos para tal uma função de partição (*window function*) chamada *lag*, que nos permite resgatar o valor da linha anterior em relação à coluna selecionada.

```
lag_window = Window.partitionBy('País').orderBy('País', 'Ano')
# criação do que chamamos de window, que é uma partição dos dados a qual será
# submetida ao uso da função 'lag' para a criação da coluna de DiferencaTemp

df_spark3 = (
    df_spark2.select('País', 'Ano', 'TemperaturaMediaAno')
          .withColumn('DiferencaTemp',
          F.col('TemperaturaMediaAno') - F.lag('TemperaturaMediaAno')
          .over(lag_window))
)
```

(k) Após estas modificações, consideramos que o dataframe de temperatura média anual estava pronto para a finalidade desejada, consistindo na junção com o dataframe de emissão de CO2 (que descreveremos no item 2.4). Sendo assim, fizemos a última operação neste Colab, salvando o dataframe como arquivo CSV no bucket GCS através da função *upload_blob*, utilizando o Google Drive como intermediário para salvar e fazer upload de arquivos.

```
#Definição de função, para poder salvar o arquivo pyspark diretamente para o
Bucket/Data Lake.
def upload_blob(bucket_name, source_file_name, destination_blob_name):
    """Uploads a file to the bucket."""
    # The ID of your GCS bucket
    # bucket_name = "your-bucket-name"
    # The path to your file to upload
    # source_file_name = "local/path/to/file"
    # The ID of your GCS object
    # destination_blob_name = "storage-object-name"

    storage_client = storage.Client()
    bucket = storage_client.bucket(bucket_name)
    blob = bucket.blob(destination_blob_name)

    blob.upload_from_filename(source_file_name)
```

```
    print("File {} uploaded to {}."
          .format(source_file_name, destination_blob_name))
upload_blob("projeto-final-bucket-g5",
"/content/drive/MyDrive/df_tempmedia_spark_tratado.csv",
"saida/df_tempmedia_pyspark_final.csv")
```

## 2.3 – DATASET 2 – Emissões de CO2

### PANDAS E PANDERA

Após instalar bibliotecas e configurar a chave para acesso ao bucket GCS, procedemos aos seguintes passos com as bibliotecas Pandas e Pandera:

(a) Fizemos o download do dataset sem tratamento, diretamente do bucket, e criamos um dataframe de Pandas.

```
df= pd.read_csv('gs://projeto-final-bucket-g5/entrada/Emissao_CO2_pais.csv')
```

(b) Verificamos os tipos de dados.

```
# verificação do tipo do dado
df.dtypes
```

Saída do código:

```
Entity                              object
Code                                object
Year                                 int64
Annual CO₂ emissions (tonnes )     float64
dtype: object
```

(c) Realizamos uma cópia de backup do dataframe Pandas.

```
df_backup = df.copy()
```

(d) Verificamos quantos e quais países estão contemplados no dataframe. Essa consulta se fez necessária para que pudéssemos chegar a nossa lista final de países contemplados em todos os dataframes.

```
#função pd.unique sendo utilizada para verificarmos as lista de países(Entity)
```

```
pd.unique(df['Entity'])
```

```
len(df['Entity'].unique())
```

Saída do código:

```
233
```

(e) Realizamos a tradução dos nomes das colunas do inglês para o português

```
(
df.rename(
      columns={'Entity': 'País', 'Code': 'Código',
               'Year': 'Ano',
               'Annual CO₂ emissions (tonnes )': 'Emissão CO₂ (ton/ano)'}
               , inplace=True)
)
```

(f) Filtramos o dataframe de modo a desconsiderar os dados a partir do ano de 2012, a fim de coincidir com a data fim da medição no dataset 1

```
filtro_anomax = df.Ano <= 2012
df = df.loc[filtro_anomax]
```

(g) Verificamos a relação de valores nulos, chegando a conclusão de que estes somente se encontravam na coluna 'Código'.

```
df.isna().sum()
```

Saída do código

```
País                        0
Código                   2162
Ano                         0
Emissão CO₂ (ton/ano)       0
dtype: int64
```

(h) A fim de melhor visualizar esses valores nulos, utilizamos o seguinte código:

```
#visualizando os valores NaN
filter = df['Código'].isna()
df.loc[filter]
```

Saída do código

| | País | Código | Ano | Emissão $CO_2$ (ton/ano) |
|---|---|---|---|---|
| 69 | Africa | NaN | 1751 | 0.0 |
| 70 | Africa | NaN | 1752 | 0.0 |
| 71 | Africa | NaN | 1753 | 0.0 |
| 72 | Africa | NaN | 1754 | 0.0 |
| 73 | Africa | NaN | 1755 | 0.0 |
| ... | ... | ... | ... | ... |
| 20343 | Wallis and Futuna Islands | NaN | 2008 | 21984.0 |
| 20344 | Wallis and Futuna Islands | NaN | 2009 | 29312.0 |
| 20345 | Wallis and Futuna Islands | NaN | 2010 | 29312.0 |
| 20346 | Wallis and Futuna Islands | NaN | 2011 | 25648.0 |
| 20347 | Wallis and Futuna Islands | NaN | 2012 | 25648.0 |

2162 rows × 4 columns

A partir dessa visualização percebemos que os valores nulos na coluna 'Código' se encontravam em linhas nas quais a coluna 'País' recebia como dado um nome de região, continente ou território. Não sendo propriamente um país, essas regiões não contam com um código de país, por essa razão tínhamos os valores nulos. Entendemos que essa inconsistência não seria um problema, já que tais regiões seriam excluídas de nosso dataframe, pois optamos por ficar somente com países.

(i) Após uma conferência detalhada dos países presentes nos dois países e criação da lista de países utilizada em ambos, procedemos à exclusão de linhas referentes a países, regiões, continentes e territórios que não seriam mais considerados.

```
df.drop(df[(df['País'] == 'Africa')].index, inplace=True)
df.drop(df[(df['País'] == 'Americas (other)')].index, inplace=True)
df.drop(df[(df['País'] == 'Asia and Pacific (other)')].index, inplace=True)
df.drop(df[(df['País'] == 'Bonaire Sint Eustatius and Saba')].index,
inplace=True)
df.drop(df[(df['País'] == 'Bermuda')].index, inplace=True)
df.drop(df[(df['País'] == 'Brunei')].index, inplace=True)
df.drop(df[(df['País'] == 'EU-28')].index, inplace=True)
df.drop(df[(df['País'] == 'Europe (other)')].index, inplace=True)
df.drop(df[(df['País'] == 'Cook Islands')].index, inplace=True)
df.drop(df[(df['País'] == 'Gibraltar')].index, inplace=True)
```

```
df.drop(df[(df['País'] == 'International transport')].index, inplace=True)
df.drop(df[(df['País'] == 'Kyrgysztan')].index, inplace=True)
df.drop(df[(df['País'] == 'Middle East')].index, inplace=True)
df.drop(df[(df['País'] == 'Maldives')].index, inplace=True)
df.drop(df[(df['País'] == 'Marshall Islands')].index, inplace=True)
df.drop(df[(df['País'] == 'Nauru')].index, inplace=True)
df.drop(df[(df['País'] == 'Saint Helena')].index, inplace=True)
df.drop(df[(df['País'] == 'Saint Pierre and Miquelon')].index, inplace=True)
df.drop(df[(df['País'] == 'Sint Maarten (Dutch part)')].index, inplace=True)
df.drop(df[(df['País'] == 'Statistical differences')].index, inplace=True)
df.drop(df[(df['País'] == 'South Sudan')].index, inplace=True)
df.drop(df[(df['País'] == 'Tuvalu')].index, inplace=True)
df.drop(df[(df['País'] == 'Vanuatu')].index, inplace=True)
df.drop(df[(df['País'] == 'Wallis and Futuna Islands')].index, inplace=True)
df.drop(df[(df['País'] == 'Czechoslovakia')].index, inplace=True)
df.drop(df[(df['País'] == 'World')].index, inplace=True)
```

(j) Após a exclusão dos países, verificamos novamente os valores nulos e obtivemos como resposta 21 valores nulos na coluna de código referentes ao território da Antártica. Avaliamos então os dados neste território, não só no dataset 2, mas também no dataset 1 (temperatura média anual), chegando à conclusão de que tais dados eram bastante inconsistentes. Sendo assim, decidimos excluir as linhas referentes a essa região

```
df.isna().sum()
```

Saída do código

```
País                        0
Código                     21
Ano                         0
Emissão CO₂ (ton/ano)       0
dtype: int64
```

```
df.drop(df[(df['País'] == 'Antarctic Fisheries')].index, inplace=True)
```

(k) Realizamos a alteração de nomes de países a fim de normalizar todos os nomes de acordo com a lista de países empregada

```
df.loc[df['País'] == 'Timor' , ['País']] = 'Timor Leste'
df.loc[df['País'] == 'Republic of the Congo' , ['País']] = 'Congo'
df.loc[df['País'] == 'Faeroe Islands' , ['País']] = 'Faroe Islands'
df.loc[df['País'] == 'Falkland Islands' , ['País']] = 'Islas Malvinas'
df.loc[df['País'] == 'Micronesia (country)' , ['País']] = 'Micronesia'
```

```
df.loc[df['País'] == 'Congo (Democratic Republic of the)' , ['País']] =
'Democratic Republic of the Congo'
```

(l) Nesse momento, decidimos salvar o dataframe de Pandas diretamente no bucket com o intuito de melhor visualizar alguns dados, utilizando a ferramenta DataPrep, disponível no GCP.

```
client=storage.Client()
#a variável bucket recebe o nome do bucket do cloud storage
bucket=client.get_bucket('projeto-final-bucket-g5')
#caminho de destino, pasta saída, o qual será salvo o arquivo em formato csv
bucket.blob('saida/df_emissoes_pandas_tratamento1.csv').upload_from_string(d
f.to_csv(index=False), 'text/cvs')
```

(m) Com o DataPrep, confirmamos algo que já havia nos chamado a atenção anteriormente: a grande quantidade de valores zero na coluna 'Emissão $CO_2$ (ton/ano)' nos séculos XVIII e XIX, coincidindo com a mesma inconsistência observada no dataset 1. Sendo assim, definimos um intervalo de tempo entre os anos de 1900 e 2012, desconsiderando todas as linhas referentes aos anos anteriores.

```
filtro_anomin = df.Ano >= 1900
df = df.loc[filtro_anomin]
```

(n) Após esse último tratamento, executamos a validação de dados com o Pandera

```
schema = pa.DataFrameSchema(
      columns = {
      "País":pa.Column(pa.String),
      "Código":pa.Column(pa.String,pa.Check.str_length(3,3), nullable=True),
      "Ano":pa.Column(pa.Int),
      "Emissão CO₂ (ton/ano)":pa.Column(pa.Float, nullable=True)})

schema.validate(df)
```

(o) Salvamos o dataframe de Pandas tratado diretamente no bucket GCS

```
client=storage.Client()
bucket=client.get_bucket('projeto-final-bucket-g5')
bucket.blob('saida/df_emissoes_pandas_tratado_final.csv').upload_from_string
(df.to_csv(index=False), 'text/cvs')
```

**PYSPARK**

Após a instalação do PySpark e importação de funções necessárias, procedemos aos seguintes tratamentos em PySpark:

(a) Definimos o esquema (schema) dos cabeçalhos e colunas, atribuindo os tipos dos valores em cada coluna e configurando a condição de aceitar ou não os valores nulos. Criamos o dataframe de PySpark a partir do dataframe tratado de Pandas, com o esquema definido em StructType.

```
schema = StructType() \
      .add("País",StringType(),False) \
      .add("Código", StringType(), True) \
      .add("Ano", IntegerType(), False) \
      .add("Emissão CO₂ (ton/ano)", FloatType(), True)

dfCO2_spark = spark.createDataFrame(df, schema=schema)
```

(b) Assim como fizemos no dataset 1, criamos também uma coluna com a diferença de emissões de um ano para o anterior. Para tal, utilizamos novamente a função de janela 'lag', recuperando dados de uma linha anterior, agrupando por país.

```
lag_window = Window.partitionBy('País').orderBy('País','Ano')
#criação do que chamamos de window, que é uma partição dos dados a qual será
submetida ao uso da função 'lag' para criação da coluna de Diferença Emissões
(ton)

dfCO2_spark = (
        dfCO2_spark.select('País', 'Código','Ano', 'Emissão CO₂ (ton/ano)')
        .withColumn('Diferenca Emissões(ton)',
        F.col('Emissão CO₂ (ton/ano)') - F.lag('Emissão CO₂ (ton/ano)')
        .over(lag_window))
)
```

(c) Após essas alterações, o dataframe de emissões de CO2 estava pronto, restando apenas salvá-lo como arquivo CSV no bucket GCS. Utilizamos mais uma vez a função *upload_blob* para esse fim, iniciando pela montagem do drive, carregamento do arquivo no drive e posteriormente no bucket do GCS.

```
from google.colab import drive
drive.mount('/content/drive')
```

```
dfCO2_spark.write.format("csv") \
.option("header", "true") \
.option("inferschema", "true") \
.option("delimiter", ",") \
.save("/content/drive/MyDrive/dados_soulcode/dfCO2.csv")
```

```
def upload_blob(bucket_name, source_file_name, destination_blob_name):
    """Uploads a file to the bucket."""
    # The ID of your GCS bucket
    # bucket_name = "your-bucket-name"
    # The path to your file to upload
    # source_file_name = "local/path/to/file"
    # The ID of your GCS object
    # destination_blob_name = "storage-object-name"

    storage_client = storage.Client()
    bucket = storage_client.bucket(bucket_name)
    blob = bucket.blob(destination_blob_name)

    blob.upload_from_filename(source_file_name)

    print("File {} uploaded to {}."
          .format(source_file_name, destination_blob_name))
```

```
upload_blob("projeto-final-bucket-g5",
"/content/drive/MyDrive/dados_soulcode/dfCO2.csv/dfCO2pyspark.csv",
"saida/df_sparkCO2_emissoes_pyspark_final.csv")
```

## 2.4 – DATASET 3 – Cobertura Florestal

### PANDAS E PANDERA

Após instalar bibliotecas e configurar a chave para acesso ao bucket GCS, procedemos aos seguintes passos com as bibliotecas Pandas e Pandera:

(a) Criamos o dataframe de Pandas a partir do dataset em formato XLSX (tabela excel), utilizando a função *pd.read_excel*. Passamos os seguintes parâmetros: o

caminho do bucket no qual o dataset original se encontra; o nome da tabela que deve ser acessada dentro do arquivo excel, utilizando o código *sheet_name;* por último a especificação da importação do cabeçalho, informando qual a linha da tabela excel corresponde ao cabeçalho desejado. Alguns cabeçalhos acabaram sem nome, uma vez que os cabeçalhos originais, do arquivo excel, estavam organizados em mais de uma linha. Posteriormente reorganizamos e renomeamos tais colunas.

```
df = pd.read_excel('gs://projeto-final-bucket-
g5/entrada/Forest_extent_characteristics_changes.xlsx',
sheet_name="1a_Forest", header = [4])
```

(b) Eliminamos algumas colunas irrelevantes.

```
df.drop(['Unnamed: 2', 'Unnamed: 3', 'Unnamed: 4', 2020, 'Unnamed: 6',
'2020.1', 'Unnamed: 8', '2020.2', 'Unnamed: 10', 'Unnamed: 11'], axis=1,
inplace=True)
```

(c) Modificamos os nomes das colunas.

```
(
df.rename(columns=
      {'Unnamed: 0': 'Código', 'Unnamed: 1': 'País',
      1990: 'Área Florestal (em 1000 ha) em 1990',
      2000: 'Área Florestal (em 1000 ha) em 2000',
      2010: 'Área Florestal (em 1000 ha) em 2010',
      2015: 'Área Florestal (em 1000 ha) em 2015',
      '2020.3': 'Área Florestal (em 1000 ha) em 2020',
      '90-00': 'Variação 1990-2000',
      '90-10': 'Variação 1990-2010',
      '90-15': 'Variação 1990-2015',
      '90-20': 'Variação 1990-2020',
      '00-10': 'Variação 2000-2010',
      '00-15': 'Variação 2000-2015',
      '00-20': 'Variação 2000-2020',
      '10-15': 'Variação 2010-2015',
      '10-20': 'Variação 2010-2020',
      '15-20': 'Variação 2015-2020',
      '90-00.1': 'Var. percentual 1990-2000',
      '90-10.1': 'Var. percentual 1990-2010',
      '90-15.1': 'Var. percentual 1990-2015',
      '90-20.1': 'Var. percentual 1990-2020',
      '00-10.1': 'Var. percentual 2000-2010',
      '00-15.1': 'Var. percentual 2000-2015',
```

```
      '00-20.1': 'Var. percentual 2000-2020',
      '10-15.1': 'Var. percentual 2010-2015',
      '10-20.1': 'Var. percentual 2010-2020',
      '15-20.1': 'Var. percentual 2015-2020'}
      , inplace=True)
)
```

(d) Eliminamos dados de países e regiões que não constam na lista de países adotada para todos os datasets.

```
df.drop(df[(df['País'] == 'South America')].index, inplace=True)
df.drop(df[(df['País'] == 'World')].index, inplace=True)
df.drop(df[(df['País'] == 'Africa')].index, inplace=True)
df.drop(df[(df['País'] == 'Asia')].index, inplace=True)
df.drop(df[(df['País'] == 'North and Central America')].index, inplace=True)
df.drop(df[(df['País'] == 'Central America')].index, inplace=True)
df.drop(df[(df['País'] == 'Oceania')].index, inplace=True)
df.drop(df[(df['País'] == 'Europe')].index, inplace=True)
df.drop(df[(df['País'] == 'Western and Central Asia')].index, inplace=True)
df.drop(df[(df['País'] == 'South and Southeast Asia')].index, inplace=True)
df.drop(df[(df['País'] == 'Western and Central Africa')].index,
inplace=True)
df.drop(df[(df['País'] == 'Eastern and Southern Africa')].index,
inplace=True)
df.drop(df[(df['País'] == 'Northern Africa')].index, inplace=True)
df.drop(df[(df['País'] == 'East Asia')].index, inplace=True)
df.drop(df[(df['País'] == 'Caribbean')].index, inplace=True)
df.drop(df[(df['País'] == 'North America')].index, inplace=True)
df.drop(df[(df['País'] == 'Mayotte')].index, inplace=True)
df.drop(df[(df['País'] == 'South Sudan')].index, inplace=True)
df.drop(df[(df['País'] == 'Guernsey')].index, inplace=True)
df.drop(df[(df['País'] == 'Holy See')].index, inplace=True)
df.drop(df[(df['País'] == 'Jersey')].index, inplace=True)
df.drop(df[(df['País'] == 'Monaco')].index, inplace=True)
df.drop(df[(df['País'] == 'San Marino')].index, inplace=True)
df.drop(df[(df['País'] == 'Svalbard and Jan Mayen Islands')].index,
inplace=True)
df.drop(df[(df['País'] == 'Saint Martin (French part)')].index,
inplace=True)
df.drop(df[(df['País'] == 'Saint Barthélemy')].index, inplace=True)
df.drop(df[(df['País'] == 'Sint Maarten (Dutch part)')].index, inplace=True)
df.drop(df[(df['País'] == 'United States Virgin Islands')].index,
inplace=True)
df.drop(df[(df['País'] == 'American Samoa')].index, inplace=True)
df.drop(df[(df['País'] == 'Bonaire, Sint Eustatius and Saba')].index,
inplace=True)
df.drop(df[(df['País'] == 'Cook Islands')].index, inplace=True)
df.drop(df[(df['País'] == 'Guam')].index, inplace=True)
df.drop(df[(df['País'] == 'Norfolk Island')].index, inplace=True)
df.drop(df[(df['País'] == 'Pitcairn Islands')].index, inplace=True)
```

```
df.drop(df[(df['País'] == 'Tokelau')].index, inplace=True)
df.drop(df[(df['País'] == 'Wallis and Futuna Islands')].index, inplace=True)
df.drop(df[(df['País'] == 'Bermuda')].index, inplace=True)
df.drop(df[(df['País'] == 'Brunei Darussalam')].index, inplace=True)
df.drop(df[(df['País'] == 'Gibraltar')].index, inplace=True)
df.drop(df[(df['País'] == 'Maldives')].index, inplace=True)
df.drop(df[(df['País'] == 'Marshall Islands')].index, inplace=True)
df.drop(df[(df['País'] == 'Nauru')].index, inplace=True)
df.drop(df[(df['País'] == 'Saint Helena, Ascension and Tristan da
Cunha')].index, inplace=True)
df.drop(df[(df['País'] == 'Saint Pierre and Miquelon')].index, inplace=True)
df.drop(df[(df['País'] == 'Sint Maarten (Dutch part)')].index, inplace=True)
df.drop(df[(df['País'] == 'South Sudan')].index, inplace=True)
df.drop(df[(df['País'] == 'Tuvalu')].index, inplace=True)
df.drop(df[(df['País'] == 'Vanuatu')].index, inplace=True)
df.drop(df[(df['País'] == 'Wallis and Futuna Islands')].index, inplace=True)
df.drop(df[(df['País'] == 'Western Sahara')].index, inplace=True)
df.drop(df[(df['País'] == 'Northern Mariana Islands')].index, inplace=True)
df.drop(df[(df['País'] == 'Isle of Man')].index, inplace=True)
```

(e) Alteramos nomes de países seguindo a mesma lista.

```
df.loc[df['País'] == "Réunion" , ["País"]] = "Reunion"
df.loc[df['País'] == "United Republic of Tanzania" , ["País"]] = "Tanzania"
df.loc[df['País'] == "Cabo Verde" , ["País"]] = "Cape Verde"
df.loc[df['País'] == "Falkland Islands (Malvinas)" , ["País"]] = "Islas
Malvinas"
df.loc[df['País'] == "Côte d'Ivoire" , ["País"]] = "Cote d'Ivoire"
df.loc[df['País'] == "Democratic People's Republic of Korea" , ["País"]] =
"North Korea"
df.loc[df['País'] == "Republic of Korea" , ["País"]] = "South Korea"
df.loc[df['País'] == "Lao People's Democratic Republic" , ["País"]] = "Laos"
df.loc[df['País'] == "Cabo Verde" , ["País"]] = "Cape Verde"
df.loc[df['País'] == "Timor-Leste" , ["País"]] = "Timor Leste"
df.loc[df['País'] == "Vietnam" , ["País"]] = "Cote d'Ivoire"
df.loc[df['País'] == "Iran (Islamic Republic of)" , ["País"]] = "Iran"
df.loc[df['País'] == "Syrian Arab Republic" , ["País"]] = "Syria"
df.loc[df['País'] == "Czechia" , ["País"]] = "Czech Republic"
df.loc[df['País'] == "North Macedonia" , ["País"]] = "Macedonia"
df.loc[df['País'] == "Republic of Moldova" , ["País"]] = "Moldova"
df.loc[df['País'] == "Russian Federation" , ["País"]] = "Russia"
df.loc[df['País'] == "Curaçao" , ["País"]] = "Curacao"
df.loc[df['País'] == "Micronesia (Federated States of)" , ["País"]] =
"Micronesia"
df.loc[df['País'] == "Venezuela (Bolivarian Republic of)" , ["País"]] =
"Venezuela"
df.loc[df['País'] == "Bolivia (plurinational state of)" , ["País"]] =
"Bolivia"
```

(f) Validamos os dados utilizando a biblioteca Pandera.

```
# construção do esquema para validação dos dados

schema = pa.DataFrameSchema(
  columns = {
  "Código":pa.Column(pa.String,pa.Check.str_length(3,3)),
  "País":pa.Column(pa.String),
  "Área Florestal (em 1000 ha) em 1990":pa.Column(pa.Float, nullable=True),
  "Área Florestal (em 1000 ha) em 2000":pa.Column(pa.Float, nullable=True),
  "Área Florestal (em 1000 ha) em 2010":pa.Column(pa.Float, nullable=True),
  "Área Florestal (em 1000 ha) em 2015":pa.Column(pa.Float, nullable=True),
  "Área Florestal (em 1000 ha) em 2020":pa.Column(pa.Float, nullable=True),
  "Variação 1990-2000":pa.Column(pa.Float, nullable=True),
  "Variação 1990-2010":pa.Column(pa.Float, nullable=True),
  "Variação 1990-2015":pa.Column(pa.Float, nullable=True),
  "Variação 1990-2020":pa.Column(pa.Float, nullable=True),
  "Variação 2000-2010":pa.Column(pa.Float, nullable=True),
  "Variação 2000-2015":pa.Column(pa.Float, nullable=True),
  "Variação 2000-2020":pa.Column(pa.Float, nullable=True),
  "Variação 2010-2015":pa.Column(pa.Float, nullable=True),
  "Variação 2010-2020":pa.Column(pa.Float, nullable=True),
  "Variação 2015-2020":pa.Column(pa.Float, nullable=True),
  "Var. percentual 1990-2000":pa.Column(pa.Float, nullable=True),
  "Var. percentual 1990-2010":pa.Column(pa.Float, nullable=True),
  "Var. percentual 1990-2015":pa.Column(pa.Float, nullable=True),
  "Var. percentual 1990-2020":pa.Column(pa.Float, nullable=True),
  "Var. percentual 2000-2010":pa.Column(pa.Float, nullable=True),
  "Var. percentual 2000-2015":pa.Column(pa.Float, nullable=True),
  "Var. percentual 2000-2020":pa.Column(pa.Float, nullable=True),
  "Var. percentual 2010-2015":pa.Column(pa.Float, nullable=True),
  "Var. percentual 2010-2020":pa.Column(pa.Float, nullable=True),
  "Var. percentual 2015-2020":pa.Column(pa.Float, nullable=True)}
)

schema.validate(df)
```

(g) Salvamos o dataframe em formato CSV diretamente no bucket GCS.

```
client=storage.Client()
bucket=client.get_bucket('projeto-final-bucket-g5')
bucket.blob('saida/df_coberturaflorestal_pandas_tratado.csv').upload_from_st
ring(df.to_csv(index=False), 'text/cvs')
```

**PYSPARK**

(a) Para criar o dataframe de PySpark, configuramos o Esquema/Schema utilizando a função StructType e importamos os dados tratados do dataframe de Pandas.

```
schema = StructType() \
      .add("Código", StringType(),False) \
      .add("País", StringType(), False) \
      .add("Área Florestal (em 1000 ha) em 1990", FloatType(), True) \
      .add("Área Florestal (em 1000 ha) em 2000", FloatType(), True) \
      .add("Área Florestal (em 1000 ha) em 2010", FloatType(), True) \
      .add("Área Florestal (em 1000 ha) em 2015", FloatType(), True) \
      .add("Área Florestal (em 1000 ha) em 2020", FloatType(), True) \
      .add("Variação 1990-2000", FloatType(), True) \
      .add("Variação 1990-2010", FloatType(), True) \
      .add("Variação 1990-2015", FloatType(), True) \
      .add("Variação 1990-2020", FloatType(), True) \
      .add("Variação 2000-2010", FloatType(), True) \
      .add("Variação 2000-2015",FloatType(), True) \
      .add("Variação 2000-2020", FloatType(), True) \
      .add("Variação 2010-2015", FloatType(), True) \
      .add("Variação 2010-2020",FloatType(), True) \
      .add("Variação 2015-2020", FloatType(), True) \
      .add("Var. percentual 1990-2000", FloatType(), True) \
      .add("Var. percentual 1990-2010",FloatType(), True) \
      .add("Var. percentual 1990-2015", FloatType(), True) \
      .add("Var. percentual 1990-2020", FloatType(), True) \
      .add("Var. percentual 2000-2010",FloatType(), True) \
      .add("Var. percentual 2000-2015", FloatType(), True) \
      .add("Var. percentual 2000-2020", FloatType(), True) \
      .add("Var. percentual 2010-2015",FloatType(), True) \
      .add("Var. percentual 2010-2020", FloatType(), True) \
      .add("Var. percentual 2015-2020", FloatType(), True)

df_cflorestal_spark = spark.createDataFrame(df, schema=schema)
```

(b) Uma vez que não encontramos necessidade de fazer mais modificações no dataframe, utilizamos a função *upload_blob* para salvar o dataframe como um arquivo CSV no bucket GCS, passando pelo mesmo processo em relação aos demais dataframes.

```
from google.colab import drive
drive.mount('/content/drive')
```

```
df_cflorestal_spark.write.format("csv") \
.option("header", "true") \
.option("inferschema", "true") \
.option("delimiter", ",") \
.save("/content/drive/MyDrive/dados_soulcode/df_cflorestal.csv")
```

```
def upload_blob(bucket_name, source_file_name, destination_blob_name):
```

```
    """Uploads a file to the bucket."""

    # The ID of your GCS bucket
    # bucket_name = "your-bucket-name"
    # The path to your file to upload
    # source_file_name = "local/path/to/file"
    # The ID of your GCS object
    # destination_blob_name = "storage-object-name"

    storage_client = storage.Client()
    bucket = storage_client.bucket(bucket_name)
    blob = bucket.blob(destination_blob_name)

    blob.upload_from_filename(source_file_name)

    print("File {} uploaded to {}.".format(
            source_file_name, destination_blob_name))
```

```
upload_blob("projeto-final-bucket-g5",
"/content/drive/MyDrive/dados_soulcode/df_cflorestal.csv/df_cflorestal_spark
_tratado.csv", "saida/df_cflorestal_pyspark_final.csv")
```

## 2.5 – DATASET 4 – Junção de Temperatura Média e Emissões de CO2

Os datasets 1 e 2, contendo dados respectivamente acerca de temperaturas médias anuais e emissões de CO2 anuais, foram tratados em Pandas/Pandera e PySpark com o objetivo de se transformarem, posteriormente, em um único dataset. Para executar a função de junção (*join*) dos datasets, utilizamos mais uma vez o framework PySpark. Apresentamos a seguir os passos para a criação e tratamento deste dataset.

(a) Importamos os datasets 1 e 2 salvos no bucket GCS através da função *download_blob*. Para tal, iniciamos a função.

```
from google.cloud import storage

def download_blob(bucket_name, source_blob_name, destination_file_name):
    """Downloads a blob from the bucket."""
```

```
    # The ID of your GCS bucket
    # bucket_name = "your-bucket-name"
    # The ID of your GCS object
    # source_blob_name = "storage-object-name"
    # The path to which the file should be downloaded
    # destination_file_name = "local/path/to/file"

    storage_client = storage.Client()
    bucket = storage_client.bucket(bucket_name)

    # Construct a client side representation of a blob.
    # `Bucket.blob` differs from `Bucket.get_blob`s as it doesn't retrieve
    # any content from Google Cloud Storage.
    # As we don't need additional data, we'll use `Bucket.blob`
    blob = bucket.blob(source_blob_name)
    blob.download_to_filename(destination_file_name)

    print("Downloaded storage object {} from bucket {} to local file
{}.".format(source_blob_name, bucket_name, destination_file_name))
```

(b) Montamos o Google drive para poder acessar a pasta /MyDrive, onde os datasets serão salvos.

```
from google.colab import drive
drive.mount('/content/drive')
```

(c) Realizamos o download dos datasets de temperatura média e emissão de CO2 para o Google drive.

```
download_blob("projeto-final-bucket-g5",
"saida/df_tempmedia_pyspark_final.csv",
"/content/drive/MyDrive/dftempmedia.csv")
```

```
download_blob("projeto-final-bucket-g5",
"saida/df_sparkCO2_emissoes_pyspark_final.csv",
"/content/drive/MyDrive/dfco2.csv")
```

(d) Criamos os dataframes de PySpark de ambos os datasets

```
df_CO2 = (spark.read.format('csv')
  .option('inferschema', 'true')
  .option('header', 'true')
  .option('sep', ',')
  .load('/content/drive/MyDrive/dfco2.csv')
)
```

```
df_temp = (spark.read.format('csv')
  .option('inferschema', 'true')
  .option('header', 'true')
  .option('sep', ',')
  .load('/content/drive/MyDrive/dftempmedia.csv')
)
```

(e) Executamos a junção dos dataframes, através da função *join*. Esta função recebe como parâmetro o dataframe que será acoplado, as colunas que servirão de referências para fazer a junção, bem como o método de junção. Em nosso caso, escolhemos as colunas 'País' e 'Ano' como referências, já que os dataframes de temperatura e emissão de CO2 contam com os mesmos países e o mesmo recorte de tempo em anos (de 1900 a 2012). Como método de junção escolhemos o *inner join*, que considera apenas as linhas com valores válidos em ambos os dataframes, em relação às colunas 'País' e 'Ano'. Chamamos o novo dataframe de df_inner.

```
#o dataframe df_inner recebe a junção do df_temp e df_CO2.
# o parâmetro 'on' indica quais as colunas vão servir de interação
# o parâmetro 'how' recebe a indicação de qual modelo de join será executado.

df_inner = df_temp.join(df_CO2, on=['País', 'Ano'], how='inner')
```

(f) Alteramos um nome de país que estava desatualizado em nossa lista: Swaziland para Eswatini

```
#função regexp permite a modificação de valores na coluna País

df_inner2 = df_inner.withColumn('País', regexp_replace('País', 'Swaziland',
'Eswatini'))
```

(g) Imaginamos que poderíamos criar mais algumas colunas no df_inner que poderiam posteriormente enriquecer nossas consultas. Pensamos então em criar colunas classificando os países de acordo com sua localização geográfica e nível socioeconômico. Tal ideia resultou em três colunas: Continente, Região e Nível de Desenvolvimento. Para cria-las precisamos organizar os países em listas por continente, por regiões e por nível de desenvolvimento (desenvolvido / em

desenvolvimento), nos valendo mais uma vez dos dados disponíveis no portal das Nações Unidas para proceder a esta classificação.

```
# Lista países desenvolvidos

africa = ['Algeria', 'Angola', 'Benin', 'Botswana', 'Burkina Faso',
'Burundi', 'Cameroon', 'Cape Verde', 'Central African Republic', 'Chad',
'Comoros', 'Congo', 'Democratic Republic of the Congo', "Cote d'Ivoire",
'Djibouti', 'Egypt', 'Equatorial Guinea', 'Eritrea', 'Eswatini', 'Ethiopia',
'Gabon', 'Gambia', 'Ghana', 'Guinea', 'Guinea-Bissau', 'Kenya', 'Lesotho',
'Liberia', 'Libya', 'Madagascar', 'Malawi', 'Mali',
'Mauritania', 'Mauritius', 'Morocco', 'Mozambique', 'Namibia', 'Niger',
'Nigeria', 'Reunion', 'Rwanda', 'Sao Tome and Principe', 'Senegal',
'Seychelles', 'Sierra Leone', 'Somalia', 'South Africa', 'Sudan',
'Tanzania', 'Togo', 'Tunisia', 'Uganda', 'Zambia', 'Zimbabwe']

america = ['Anguilla', 'Antigua and Barbuda', 'Argentina', 'Aruba',
'Bahamas', 'Barbados', 'Belize', 'Bolivia', 'Brazil', 'British Virgin
Islands', 'Canada', 'Cayman Islands', 'Chile', 'Colombia', 'Costa Rica',
'Cuba', 'Curacao', 'Dominica', 'Dominican Republic', 'Ecuador', 'El
Salvador', 'Islas Malvinas', 'French Guiana', 'Greenland',
'Grenada', 'Guadeloupe', 'Guatemala', 'Guyana', 'Haiti', 'Honduras',
'Jamaica', 'Martinique', 'Mexico', 'Montserrat', 'Nicaragua', 'Panama',
'Paraguay', 'Peru', 'Saint Kitts and Nevis', 'Saint Lucia',
'Saint Vincent and the Grenadines', 'Suriname', 'Trinidad and Tobago',
'Turks and Caicos Islands', 'United States', 'Uruguay', 'Venezuela']

asia = ['Afghanistan', 'Armenia', 'Azerbaijan', 'Bahrain', 'Bangladesh',
'Bhutan', 'Cambodia', 'China', 'Cyprus', 'Georgia', 'Hong Kong', 'India',
'Indonesia', 'Iran', 'Iraq', 'Israel', 'Japan', 'Jordan', 'Kazakhstan',
'Kuwait', 'Kyrgyzstan', 'Laos', 'Lebanon', 'Macao', 'Malaysia', 'Mongolia',
'Myanmar', 'Nepal', 'North Korea', 'Oman', 'Pakistan', 'Palestine',
'Philippines', 'Qatar', 'Saudi Arabia', 'Singapore', 'South Korea', 'Sri
Lanka', 'Syria', 'Taiwan', 'Tajikistan', 'Thailand', 'Timor Leste',
'Turkey', 'Turkmenistan', 'United Arab Emirates', 'Uzbekistan', 'Vietnam',
'Yemen']

europa = ['Albania', 'Andorra', 'Austria', 'Belarus', 'Belgium', 'Bosnia and
Herzegovina', 'Bulgaria', 'Croatia', 'Czech Republic', 'Denmark', 'Estonia',
'Faroe Islands', 'Finland', 'France', 'Germany', 'Greece', 'Hungary',
'Iceland', 'Ireland', 'Italy', 'Latvia', 'Liechtenstein', 'Lithuania',
'Luxembourg', 'Macedonia', 'Malta', 'Moldova', 'Montenegro', 'Netherlands',
'Norway', 'Poland', 'Portugal', 'Romania', 'Russia', 'Serbia', 'Slovakia',
'Slovenia', 'Spain', 'Sweden', 'Switzerland', 'Ukraine', 'United Kingdom']

oceania = ['Australia', 'Christmas Island', 'Fiji', 'French Polynesia',
'Kiribati', 'Micronesia', 'New Caledonia','New Zealand', 'Niue', 'Palau',
'Papua New Guinea', 'Samoa', 'Solomon Islands', 'Tonga']
```

```
# Lista de países por regiões

# Africa

africa_norte = ['Algeria', 'Egypt', 'Libya', 'Morocco', 'Sudan', 'Tunisia']

africa_leste = ['Burundi', 'Comoros', 'Djibouti', 'Eritrea', 'Ethiopia',
'Kenya', 'Madagascar', 'Malawi', 'Mauritius', 'Mozambique', 'Reunion',
'Rwanda', 'Seychelles', 'Somalia', 'Uganda', 'Tanzania', 'Zambia',
'Zimbabwe']

africa_centro = ['Angola', 'Cameroon', 'Central African Republic', 'Chad',
'Congo', 'Democratic Republic of the Congo', 'Equatorial Guinea', 'Gabon',
'Sao Tome and Principe']

africa_sul = ['Botswana', 'Lesotho', 'Namibia', 'South Africa', 'Eswatini']

africa_oeste = ['Benin', 'Burkina Faso', 'Cape Verde', "Cote d'Ivoire",
'Gambia', 'Ghana', 'Guinea', 'Guinea-Bissau', 'Liberia', 'Mali',
'Mauritania', 'Niger', 'Nigeria', 'Senegal', 'Sierra Leone', 'Togo']

# America

america_caribe = ['Anguilla', 'Antigua and Barbuda', 'Aruba', 'Bahamas',
'Barbados', 'British Virgin Islands', 'Cayman Islands', 'Cuba', 'Curacao',
'Dominica', 'Dominican Republic', 'Grenada', 'Guadeloupe', 'Haiti',
'Jamaica', 'Martinique', 'Montserrat', 'Saint Kitts and Nevis', 'Saint
Lucia', 'Saint Vincent and the Grenadines', 'Trinidad and Tobago', 'Turks and
Caicos Islands']

america_central = ['Belize', 'Costa Rica', 'El Salvador', 'Guatemala',
'Honduras', 'Mexico', 'Nicaragua', 'Panama']

america_sul = ['Argentina', 'Bolivia', 'Brazil', 'Chile', 'Colombia',
'Ecuador', 'Islas Malvinas', 'French Guiana', 'Guyana','Paraguay', 'Peru',
'Suriname', 'Uruguay', 'Venezuela']

america_norte = ['Canada', 'Greenland', 'United States']

# Asia

asia_central = ['Kazakhstan', 'Kyrgyzstan', 'Tajikistan', 'Turkmenistan',
'Uzbekistan']

asia_leste = ['China', 'Hong Kong', 'Macao', 'Mongolia', 'North Korea', 'South
Korea', 'Japan', 'Taiwan']

asia_sudeste = ['Cambodia', 'Indonesia', 'Laos', 'Malaysia', 'Myanmar',
'Philippines', 'Singapore', 'Thailand', 'Timor Leste', 'Vietnam']

asia_sul = ['Afghanistan', 'Bangladesh', 'Bhutan', 'Iran', 'India', 'Nepal',
'Pakistan', 'Sri Lanka']
```

```
asia_oeste = ['Armenia', 'Azerbaijan', 'Bahrain', 'Cyprus', 'Georgia', 'Iraq',
'Israel', 'Jordan', 'Kuwait', 'Lebanon', 'Oman', 'Qatar', 'Saudi Arabia',
'Palestine', 'Syria', 'Turkey', 'United Arab Emirates', 'Yemen']

# Europa

europa_leste = ['Belarus', 'Bulgaria', 'Czech Republic', 'Hungary', 'Poland',
'Moldova', 'Romania', 'Russia', 'Slovakia', 'Ukraine']

europa_norte = ['Denmark', 'Estonia', 'Faroe Islands', 'Finland', 'Iceland',
'Ireland', 'Latvia', 'Lithuania', 'Norway', 'Sweden', 'United Kingdom']

europa_sul = ['Albania', 'Andorra', 'Bosnia and Herzegovina', 'Croatia',
'Greece', 'Italy', 'Malta', 'Montenegro', 'Macedonia', 'Portugal', 'Serbia',
'Slovenia', 'Spain']

europa_oeste = ['Austria', 'Belgium', 'France', 'Germany', 'Liechtenstein',
'Luxembourg', 'Netherlands', 'Switzerland']

# Oceania

australia_nova_zelandia = ['Australia', 'Christmas Island', 'New Zealand']

melanesia = ['Fiji', 'New Caledonia', 'Papua New Guinea', 'Solomon Islands']

micronesia = ['Kiribati', 'Micronesia', 'Palau']

polynesia = ['French Polynesia', 'Samoa', 'Niue', 'Tonga']
```

```
# Lista países desenvolvidos

paises_desenvolvidos = ['Albania', 'Andorra', 'Australia', 'Austria',
'Belarus', 'Belgium', 'Bosnia and Herzegovina', 'Bulgaria', 'Canada',
'Christmas Island', 'Croatia', 'Cyprus', 'Czech Republic', 'Denmark',
'Estonia', 'Faroe Islands', 'Finland', 'France', 'Germany','Greece',
'Greenland','Hungary', 'Iceland', 'Ireland', 'Israel', 'Italy', 'Japan',
'Latvia', 'Liechtenstein','Lithuania', 'Luxembourg', 'Macedonia', 'Malta' ,
'Moldova', 'Montenegro','Netherlands', 'New Zealand', 'Norway',
'Poland', 'Portugal', 'Romania', 'Russia','Serbia', 'Slovakia', 'Slovenia',
'Spain', 'Sweden', 'Switzerland', 'Ukraine', 'United Kingdom', 'United
States']
```

(h) A partir das listas, procedemos à criação das novas colunas. Para a criação da coluna continente utilizamos a função *withColumn* junto à função *when,* permitindo informar condições acerca dos valores que serão inseridos, e dentro da função *when* implementamos a função *isin,* passando como parâmetro cada

lista. O código funciona da seguinte maneira: se (when) o valor da coluna 'País', em determinada linha, estiver contido na lista europa, o valor a ser preenchido na nova coluna será 'Europa'. O mesmo código foi utilizado para todos os continentes, alterando somente os parâmetros da função *isin* e o valor a ser preenchido.

```
df_inner2 = (
      df_inner2.withColumn('Continente',
       when(F.col('País').isin(europa), 'Europa')
      .when(F.col('País').isin(asia), 'Ásia')
      .when(F.col('País').isin(africa), 'África')
      .when(F.col('País').isin(america), 'América')
      .when(F.col('País').isin(oceania), 'Oceania'))
)
```

(i) Procedemos da mesma maneira para a criação da coluna 'Região'.

```
df_inner2 = (
    df_inner2.withColumn('Região',
      when(F.col('País').isin(africa_norte), 'África Norte')
     .when(F.col('País').isin(africa_leste), 'África Leste')
     .when(F.col('País').isin(africa_centro), 'África Centro')
     .when(F.col('País').isin(africa_oeste), 'África Oeste')
     .when(F.col('País').isin(africa_sul), 'África Sul')
     .when(F.col('País').isin(america_caribe), 'América Caribe')
     .when(F.col('País').isin(america_sul), 'América do Sul')
     .when(F.col('País').isin(america_central), 'América Central')
     .when(F.col('País').isin(america_norte), 'América do Norte')
     .when(F.col('País').isin(asia_sul), 'Ásia Sul')
     .when(F.col('País').isin(asia_leste), 'Ásia Leste')
     .when(F.col('País').isin(asia_oeste), 'Ásia Oeste')
     .when(F.col('País').isin(asia_central), 'Ásia Central')
     .when(F.col('País').isin(asia_sudeste), 'Ásia sudeste')
     .when(F.col('País').isin(europa_leste), 'Europa Leste')
     .when(F.col('País').isin(europa_norte), 'Europa Norte')
     .when(F.col('País').isin(europa_sul), 'Europa Sul')
     .when(F.col('País').isin(europa_oeste), 'Europa Oeste')
     .when(F.col('País').isin(australia_nova_zelandia), 'Australia e Nova
Zelândia')
     .when(F.col('País').isin(melanesia), 'Melanesia')
     .when(F.col('País').isin(micronesia), 'Micronésia')
     .when(F.col('País').isin(polynesia), 'Polynesia'))
)
```

(j) Já para a criação da coluna 'Nivel desenvolvimento', utilizamos a função *when* junto à função *otherwise*, indicando que, caso o país na linha em questão esteja contido na lista países_desenvolvidos, a nova coluna receberá o valor 'Desenvolvido', caso contrário o valor recebido será 'Em desenvolvimento'.

```
df_inner2 = (
      df_inner2.withColumn('Nivel_desenvolvimento',
      when(F.col('País').isin(paises_desenvolvidos), 'Desenvolvido')
      .otherwise('Em desenvolvimento'))
)
```

(k) Após a criação das novas colunas, renomeamos todas as colunas buscando o mesmo padrão de nomenclatura.

```
df_inner2 = df_inner2.withColumnRenamed('País', 'Pais')
df_inner2 = df_inner2.withColumnRenamed('TemperaturaMediaAno',
'TempMediaAno')
df_inner2 = df_inner2.withColumnRenamed('DiferencaTemp', 'DifTempAno')
df_inner2 = df_inner2.withColumnRenamed('Código', 'Codigo')
df_inner2 = df_inner2.withColumnRenamed('Emissão CO₂ (ton/ano)',
'EmissaoCO2TonAno')
df_inner2 = df_inner2.withColumnRenamed('Diferenca Emissões(ton)',
'DifEmissaoCO2TonAno')
df_inner2 = df_inner2.withColumnRenamed('Nivel_desenvolvimento',
'NivelDesenvolvimento')
df_inner2 = df_inner2.withColumnRenamed('Região', 'Regiao')
df_inner2 = df_inner2.withColumnRenamed('DifTempAno', 'DifTempMediaAno')
```

(l) Reordenamos as colunas em uma sequência lógica.

```
df_inner3 = df_inner2.select(['Pais', 'Codigo', 'Continente', 'Regiao',
'NivelDesenvolvimento', 'Ano', 'TempMediaAno', 'DifTempMediaAno',
'EmissaoCO2TonAno', 'DifEmissaoCO2TonAno'])
```

(m) Por fim, ordenamos o data frame por ordem alfabética de Países e por ordem crescente de ano.

```
df_inner3=df_inner3.orderBy('Pais','Ano')
```

(n) Salvamos o dataframe PySpark como um arquivo CSV no Google Drive.

```
df_inner3.write.format("csv") \
      .option("header", "true") \
      .option("inferschema", "true") \
```

```
      .option("delimiter", ",") \
      .save("/content/drive/MyDrive/df_temp_co2_final.csv")
```

(o) Mais uma vez através da função *upload_blob*, salvamos o arquivo CSV diretamente no bucket GCS.

```
def upload_blob(bucket_name, source_file_name, destination_blob_name):
    """Uploads a file to the bucket."""
    # The ID of your GCS bucket
    # bucket_name = "your-bucket-name"
    # The path to your file to upload
    # source_file_name = "local/path/to/file"
    # The ID of your GCS object
    # destination_blob_name = "storage-object-name"

    storage_client = storage.Client()
    bucket = storage_client.bucket(bucket_name)
    blob = bucket.blob(destination_blob_name)
    blob.upload_from_filename(source_file_name)

    print("File {} uploaded to {}."
          .format(source_file_name, destination_blob_name))

upload_blob("projeto-final-bucket-g5",
"/content/drive/MyDrive/df_temp_co2_final.csv/DF_temp_co2_final.csv",
"saida/df_temp_co2_final.csv")
```

(p) Abaixo trazemos uma imagem das 10 primeiras linhas do dataframe tratado.

| Pais | Codigo | Continente | Regiao | NivelDesenvolvimento | Ano | TempMediaAno | DifTempMediaAno | EmissaoCO2TonAno | DifEmissaoCO2TonAno |
|---|---|---|---|---|---|---|---|---|---|
| Afghanistan | AFG | Ásia | Ásia Sul | Em desenvolvimento | 1949 | 13.350083380937576 | -0.9851666490236912 | 14656.0 | null |
| Afghanistan | AFG | Ásia | Ásia Sul | Em desenvolvimento | 1950 | 13.043499952182174 | -0.30658342875540257 | 84272.0 | 69616.0 |
| Afghanistan | AFG | Ásia | Ásia Sul | Em desenvolvimento | 1951 | 13.967749883731207 | 0.9242499315490331 | 91600.0 | 7328.0 |
| Afghanistan | AFG | Ásia | Ásia Sul | Em desenvolvimento | 1952 | 14.175416549046835 | 0.20766666531562805 | 91600.0 | 0.0 |
| Afghanistan | AFG | Ásia | Ásia Sul | Em desenvolvimento | 1953 | 14.650749941666922 | 0.47533339262008667 | 106256.0 | 14656.0 |
| Afghanistan | AFG | Ásia | Ásia Sul | Em desenvolvimento | 1954 | 13.691333214441935 | -0.9594167272249869 | 106256.0 | 0.0 |
| Afghanistan | AFG | Ásia | Ásia Sul | Em desenvolvimento | 1955 | 14.642583400011063 | 0.951250185569128 | 153888.0 | 47632.0 |
| Afghanistan | AFG | Ásia | Ásia Sul | Em desenvolvimento | 1956 | 14.191083321968714 | -0.4515000780423488 | 183200.0 | 29312.0 |
| Afghanistan | AFG | Ásia | Ásia Sul | Em desenvolvimento | 1957 | 12.77716659506162 | -1.4139167269070931 | 293120.0 | 109920.0 |
| Afghanistan | AFG | Ásia | Ásia Sul | Em desenvolvimento | 1958 | 14.716750005880991 | 1.9395834108193704 | 329760.0 | 36640.0 |
| Afghanistan | AFG | Ásia | Ásia Sul | Em desenvolvimento | 1959 | 14.15258331100146 | -0.5641666948795319 | 384571.4 | 54811.406 |

## 2.6 – PIPELINE | APACHE BEAM | PUB/SUB | DATAFLOW

O uso de Pipelines - que podem ser traduzidas, grosso modo, como "linhas de montagem" - serve para otimização e de processos e maior agilidade no tratamento de dados. Estes dados podem ser originados a partir de dispositivos móveis ou bases de dados, alimentos em sistemas em modo streaming ou em arquivos em lote - também chamados de arquivos tipo "batch".

Seja qual for a forma de obtenção/alimentação dos dados a serem tratados, uma vez que se tenha em mente qual o tipo de tratamento que aqueles dados precisam e como eles precisam ser disponibilizados após passarem pelo tratamento, é possível já começar a montar as etapas deste tratamento em um modelo/template de Pipeline customizado ou avaliar se é possível já utilizar algum dos modelos já disponibilizados em plataformas como a Google Cloud em sua API Dataflow.

Tendo em mente os propósitos e diretrizes deste trabalho, buscamos fazer um recorte dentro de um dataset já tratado nosso - o dataset de Cobertura Florestal. Primeiramente fizemos um novo tratamento em Pyspark deste dataset, pois era necessária a criação de uma coluna "Continente" para melhor manipulação e transformação dos dados. Então, definimos o tratamento a ser aplicado ao dataset Cobertura Florestal - fazer um recorte por Continente, selecionando o continente "Ásia"; dentro deste Continente, selecionar a coluna "País"; na segunda etapa de seleção e agrupamento, selecionar a Variação da Área Florestal no Período de 1990 a 2010 nos Países do Continente Asiático.

Consideramos este tratamento com os recortes mencionados interessante dentro da temática analisada pelo nosso grupo devido a enorme variedade entre os países dentro do continente asiático considerando diversos quesitos, tais quais níveis de emissão de CO2, médias de Variação de Temperatura e níveis de Desenvolvimento.

Para que fosse possível realizar as manipulações de dados, tratamentos e conseguinte disponibilização destes utilizamos as APIs Pub/Sub e Dataflow da Google Cloud Platform e o Apache Beam para a construção do template da nossa Pipeline.

A seguir estão dispostos os códigos utilizados nas várias etapas destes processos.

**Download do Dataset**

Importação de biblioteca para autenticação da conta de serviço GCP.

```
import os
```

Criação de variável com o caminho para a chave relacionada à conta de serviço associada ao projeto. Através da biblioteca os, o arquivo de chave da conta de serviço é atribuído a variável serviceAccount. A chave de serviço em formato json foi carregada localmente no Google Colaboratory. Essa parte é essencial para que seja possível acessar o bucket diretamente do Colab.

```
service_account_key = r"/content/chave_projetofinal-grupo5-d2c4a9d78233.json"
os.environ["GOOGLE_APPLICATION_CREDENTIALS"] = service_account_key
```

Instalações de pacotes.

```
pip install apache-beam[interactive]
pip install apache-beam[gcp]
pip install google-cloud-pubsub
```

Importação de módulos e de funções para a execução da primeira etapa do Pipeline com Apache Beam com Dataflow e integração de Pub/Sub.

```
import csv          #importação do módulo módulo csv, perimitindo uso de classes
                    # para ler e gravar dados no formato .csv

import time         #importação do módulo time, para utilização da função time.sleep,
                    # a fim de proporcionar melhor visualização entre
                    # uma mensagem e a próxima

from google.cloud import pubsub_v1  #importação do módulo pubsub_v1 da biblioteca
                                    # google.cloud, para que utilizemos
                                    # as funcionalidades do Pub/Sub
                                    # no ambiente do Google Colaboratory
```

Definição e execução da função utilizada para salvar o arquivo tratado no bucket GCP – dataframe de Cobertura Florestal, acrescida de coluna “Continente” em tratamento

breve e simples no PySpark, exclusivamente para uso do dataframe na transformação de dados com a Pipeline.

```
from google.cloud import storage

def download_blob(bucket_name, source_blob_name, destination_file_name):
    storage_client = storage.Client()
    bucket = storage_client.bucket(bucket_name)
    blob = bucket.blob(source_blob_name)
    blob.download_to_filename(destination_file_name)

    print(
        "Downloaded storage object {} from bucket {} to local file {}.".format(
            source_blob_name, bucket_name, destination_file_name
        )
    )
```

Montagem do Drive na pasta local para download do arquivo no bucket.

```
from google.colab import drive
drive.mount('/content/drive')
```

Download do arquivo.

```
download_blob("projeto-final-bucket-g5", \
              "saida/df_cobertura_florestal_pipeline.csv", \
              "/content/drive/MyDrive/dados_soulcode/pipeline/df_cobertura_florestal_pipeline.csv")
```

## PRIMEIRA ETAPA - PUBLICAÇÃO DOS DADOS DO DATASET NO TÓPICO PUB/SUB DEFINIDO

**Definições básicas para execução da Primeira Etapa.**

Tópico Pub/Sub

```
topico = 'projects/projetofinal-grupo5/topics/topicov1'  #definição do caminho para
                                                         #publicação da leitura
                                                         # do arquivo dentro do Pub/Sub
```

Uso de função para publicação no tópico.

```
publisher = pubsub_v1.PublisherClient()                # uso de função  de publicação
                                                       # a ser utilizada do módulo  Pub/Sub
```

Declaração do caminho do arquivo a ser enviado ao tópico.

```
entrada = r"/content/drive/MyDrive/dados_soulcode/pipeline/df_cobertura_florestal_pipeline.csv"

                                                       #definição do caminho do arquivo a ser
                                                       #enviado ao tópico Pub/Sub já definido
```

Definição de função que mostra a Primeira Etapa sendo executada.

```
with open(entrada, 'rb') as file:                      #definição de função que mostra a evolução
                                                       # da leitura do arquivo e subsequente publicação
                                                       # e publicação das mensagens no tópico
    for row in file:
        print('Publicando dados no tópico')
        publisher.publish(topico,row)
        time.sleep(2)
```

## SEGUNDA ETAPA - MONTAGEM DO TEMPLATE DA PIPELINE

Importação de módulos e de funções para a Segunda Etapa - montagem do Template a ser utilizado na API Dataflow, no ambiente GCP.

```
import apache_beam as beam
import os
from apache_beam.options.pipeline_options import PipelineOptions #importação da função
                                                                 #PipelineOptions para
                                                                 #definição dos parâmetros
                                                                 #para ordenação da
                                                                 #montagem do nosso template
from apache_beam import window   #importação de função window, para definição de janelas de
                                 #tempo realtivas a recorete/captura de dados,
                                 #essencial quandose trabalha com pipelines de dados em streaming
```

Definição dos parâmetros da Pipeline e definições de funções a serem usadas para composição do template da Pipeline.

```
pipeline_options = {
    'project': 'projetofinal-grupo5',   #ID do Projeto
    'runner': 'DataflowRunner',         #executor do Projeto
    'region': 'southamerica-east1',     #região onde o projeto será executado
    'staging_location': 'gs://projeto-final-bucket-g5/temp',
                        #staging é a área de preparação/
                        #espaço de armazenamento necessário p/ execução do trabalho
    'temp_location': 'gs://projeto-final-bucket-g5/temp', #local de processamento temporário,
                                                          # arquivos apagadosdepois
    'template_location': 'gs://projeto-final-bucket-g5/template/projeto_final_g5_TEMPLATE',
                        #local onde o template será armazenado
    'save_main_session': True, #declaração para que sessão seja salva
    'streaming': True #declaração de que se trata de uma pipeline de streaming}


pipeline_options = PipelineOptions.from_dictionary(pipeline_options)
p1 = beam.Pipeline(options=pipeline_options)
```

Definição de caminhos para buscar arquivos para tratamento e armazenar arquivos tratados.

```
subscription = 'projects/projetofinal-grupo5/subscriptions/assinaturatp1'
  #definição do local de onde serão buscados os arquivos para inserção na pipeline,
  #quando o template dela for utlizado no Dataflow
saida = 'projects/projetofinal-grupo5/topics/topicov2'
  #definição de local onde os arquivos tratados pelos processos da Pipeline serão
  #publicados após o tratamento
```

Definição de função usada na Pipeline, para leitura de arquivo em UTF-8 e uso de vírgula como separador.

```
class separar_linhas(beam.DoFn):
  def process(self,record):
    return [record.decode("utf-8").split(',')]
```

Leitura do arquivo na Pipeline.

```
pcollection_entrada = (
    p1  | 'Read from pubsub topic' >> beam.io.ReadFromPubSub(subscription= subscription)
)
 #definição do arquivo a ser lido na pipeline construída
```

Primeira etapa de tratamento de dados na Pipeline.

```
Paises_Asia = (
    pcollection_entrada
    | "Separar por Vírgulas Países" >> beam.ParDo(separar_linhas())
    #uso da função separadora de linhas
    | "Continente Ásia1" >> beam.Filter(lambda record: record[0] == 'Ásia')
    #recorte, dentro do dataset, por Continente
    | "Paises Ásia1" >> beam.Filter(lambda record: record[1])
    #recorte, dentro do dataset, dos Países específicos dentro do Continente
    | "Agregação de colunas1" >> beam.Map(lambda record : (record[0], record[1]))
    #agregação das colunas Continente e País
    | "Window1" >> beam.WindowInto(window.SlidingWindows(10,5))
    #definição de parâmetros da janela
    | "Combinar os dados1" >> beam.GroupByKey()
    #definição da forma de combinar os dados para posterior manipulação
)
```

Segunda etapa de tratamento de dados na Pipeline.

```
Variacao_Area_Florestal_1990_2010 = (
    pcollection_entrada
    | "Separar por Vírgulas Area Florestal" >> beam.ParDo(separar_linhas())
    #uso da função separadora de linhas
    | "Continente Ásia2" >> beam.Filter(lambda record: record[0] == 'Ásia')
    #recorte, dentro do dataset, por Continente
    | "Variacao_Area_Florestal_1990_2010" >> beam.Filter(lambda record: float(record[9]))
    #recorte, dentro do dataset, da Variaçao de Área Florestal
    # dentro de um período de tempo, dentro do Continente
    | "Agregação de colunas2" >> beam.Map(lambda record : (record[0], float(record[9])))
    #agregação das colunas Continente e Variaçao de Área Florestal 1990-2010
    | "Window2" >> beam.WindowInto(window.SlidingWindows(10,5))
    #definição de parâmetros da janela
    | "Combinar os dados2" >> beam.GroupByKey()
    #definição da forma de combinar os dados para posterior manipulação
)
```

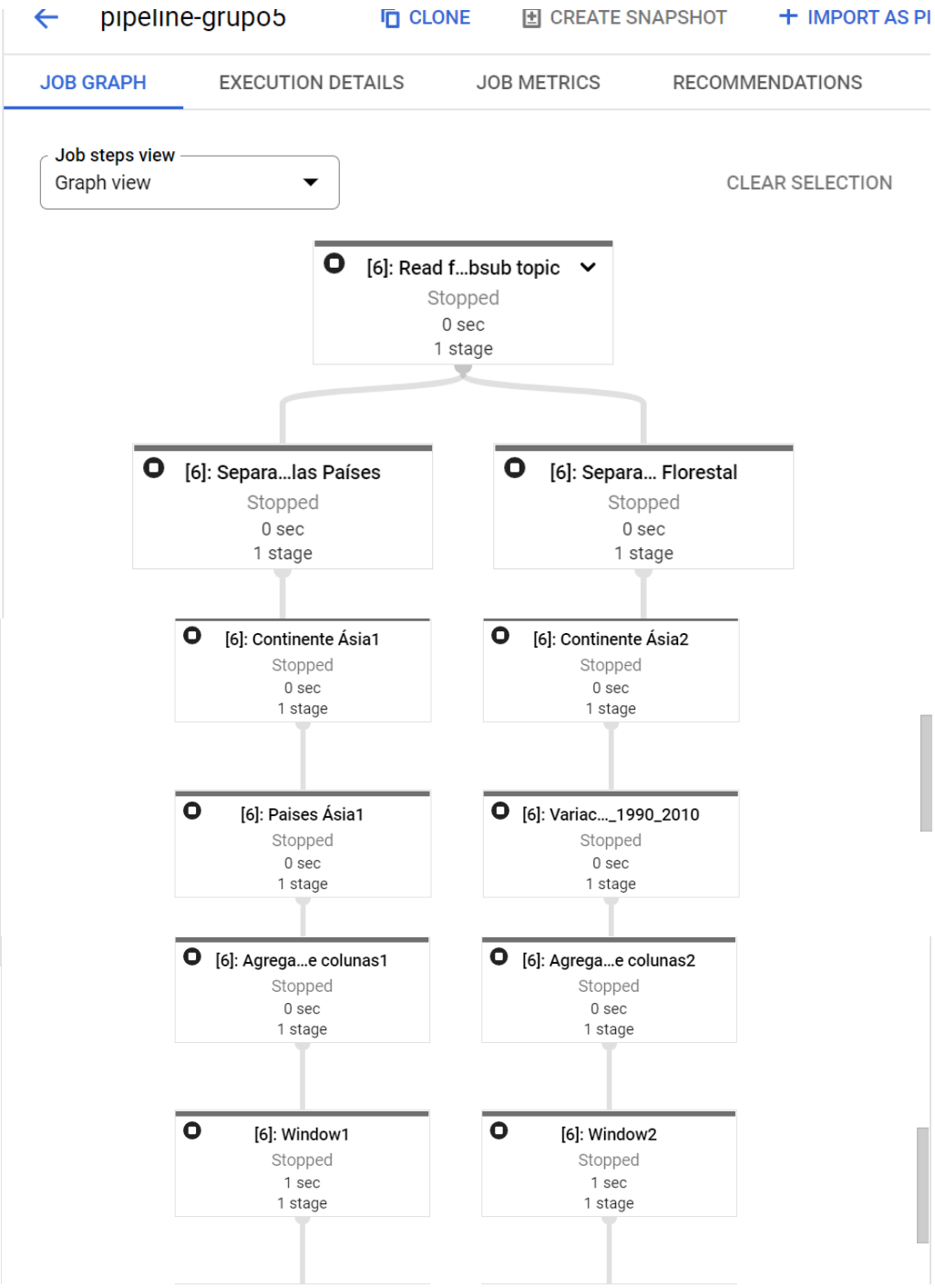
Etapa final da Pipeline - junção dos dados tratados em um Tabela Final.

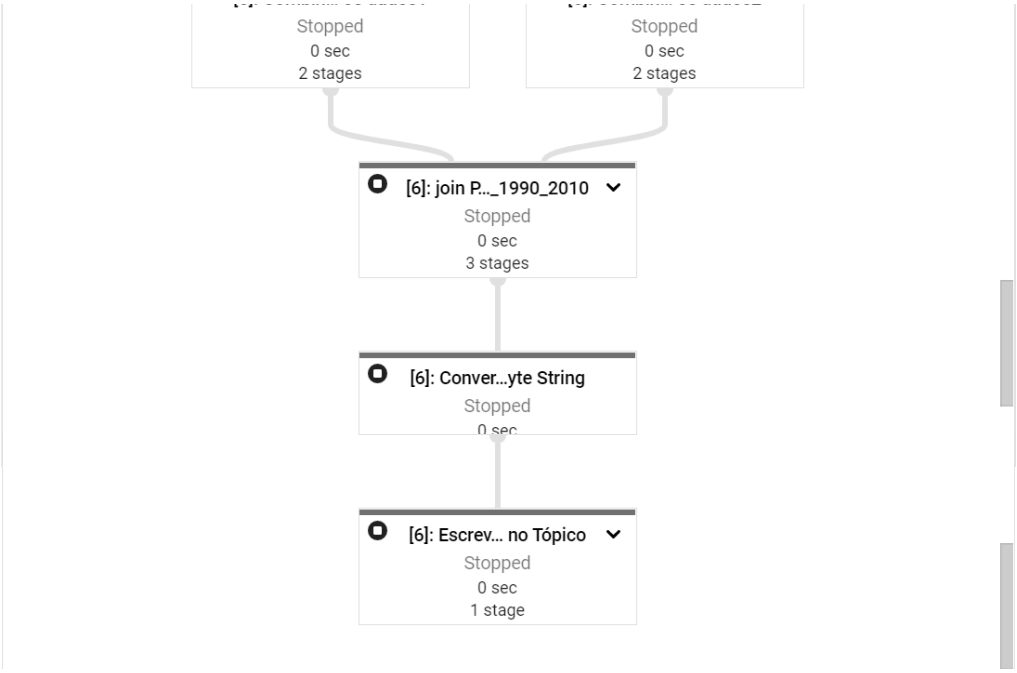
```
Tabela_Asia_Var_CFlorestal_90_10 = (
    {'Paises_Asia' : Paises_Asia, 'Variacao_Area_Florestal_1990_2010': Variacao_Area_Florestal_1990_2010}
                                    #junção dos dadoo tratados nas outra etapa em uma Tabela Final
    | "join Paises_Asia, Variacao_Area_Florestal_1990_2010" >> beam.CoGroupByKey()
                                    #definição de função pra junção
    | 'Converting to byte String' >> beam.Map(lambda row: (''.join(str(row)).encode('utf-8')) )
                                    #definição do formato de leitura do aquivo final
    | "Escrever no Tópico" >> beam.io.WriteToPubSub(saida)
                                    #definição do local de armazenamento da Tabela Final
)
```

Funções para execução da montagem do template da Pipeline.

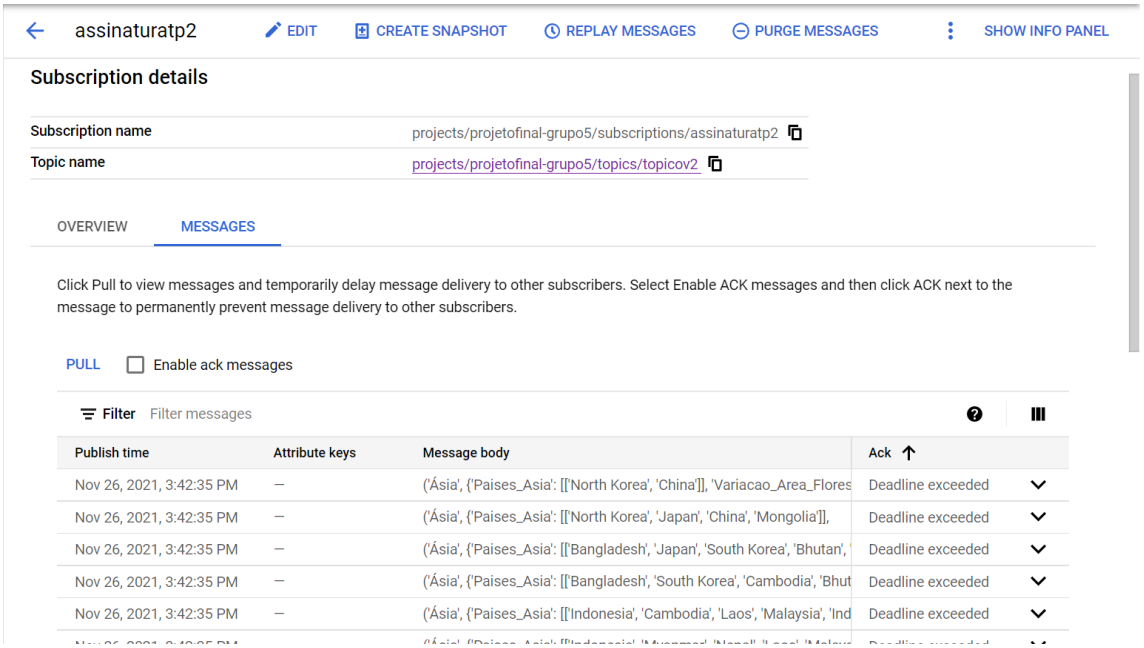
```
result = p1.run()
result.wait_until_finish()
```

## TERCEIRA ETAPA - EXECUÇÃO DA PIPELINE COM DATAFLOW






Dados tratados armazenados na assinatura "assinaturatp2" do tópico 2 "topicov2".



# 3. CONSULTAS E VISUALIZAÇÕES

As consultas aos bancos de dados, bem como os gráficos criados a partir destas consultas, foram realizadas utilizando quatro ferramentas diferentes: Spark SQL e BigQuery para consultas em SQL; Plotagem Pandas e Data Studio para visualizações gráficas.

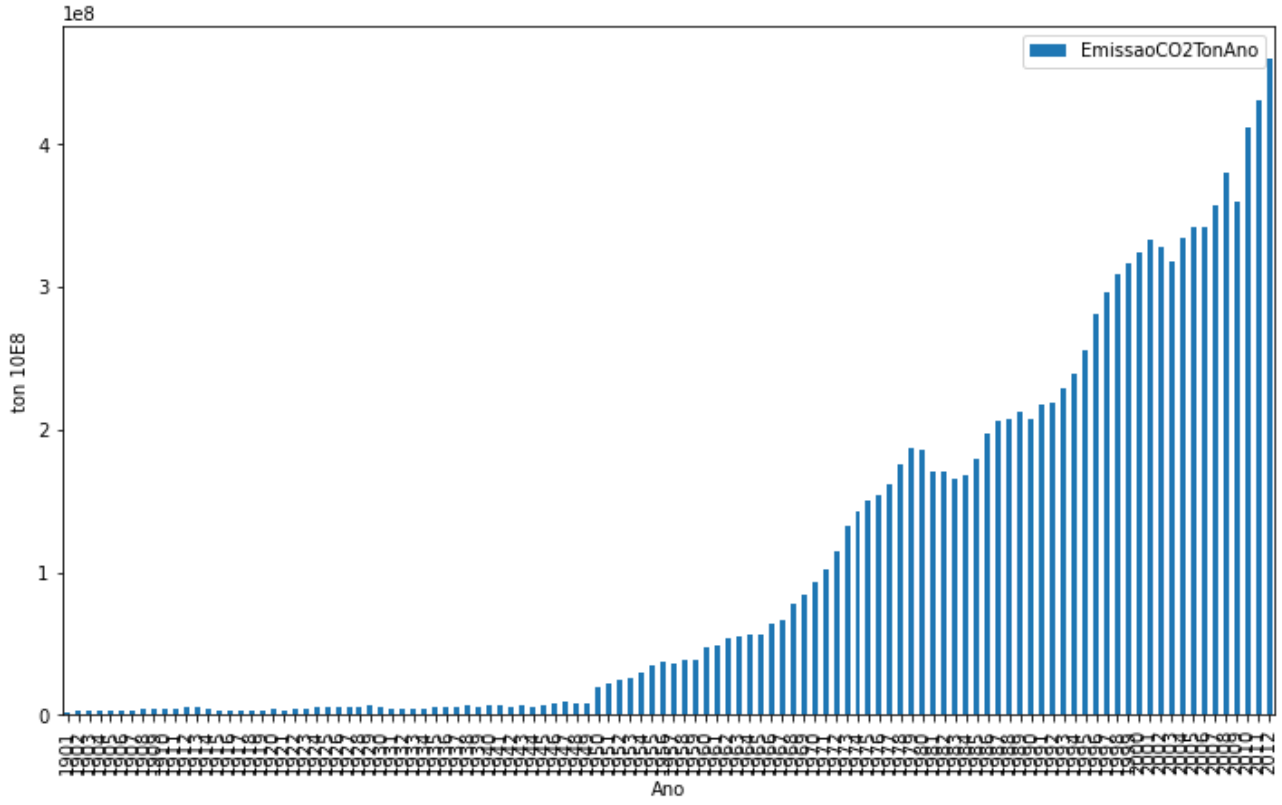
As ferramentas Spark SQL e Plotagem Pandas foram utilizadas como uma espécie de rascunho, compreendendo os primeiros testes em termos de consultas e gráficos. As consultas de SparkSQL foram reorganizadas e reformuladas posteriormente na plataforma BigQuery, de modo a trazer uma narrativa coerente, potencializando insights. A partir das consultas e tabelas criadas no BigQuery, desenvolvemos gráficos utilizando o Data Studio, dando mais consistência e ênfase em determinados pontos através de sua visualização.

Neste trabalho iremos apresentar a Plotagem em Pandas e as consultas no BigQuery. Optamos por não apresentar as consultas em SparkSQL por se tratar de um estágio inicial, sendo que as consultas mais relevantes foram replicadas no BigQuery. Já em relação ao Data Studio, achamos mais pertinente deixar o link para acesso diretamente na plataforma (no item 1.3 – ), por se tratar de uma ferramenta interativa.

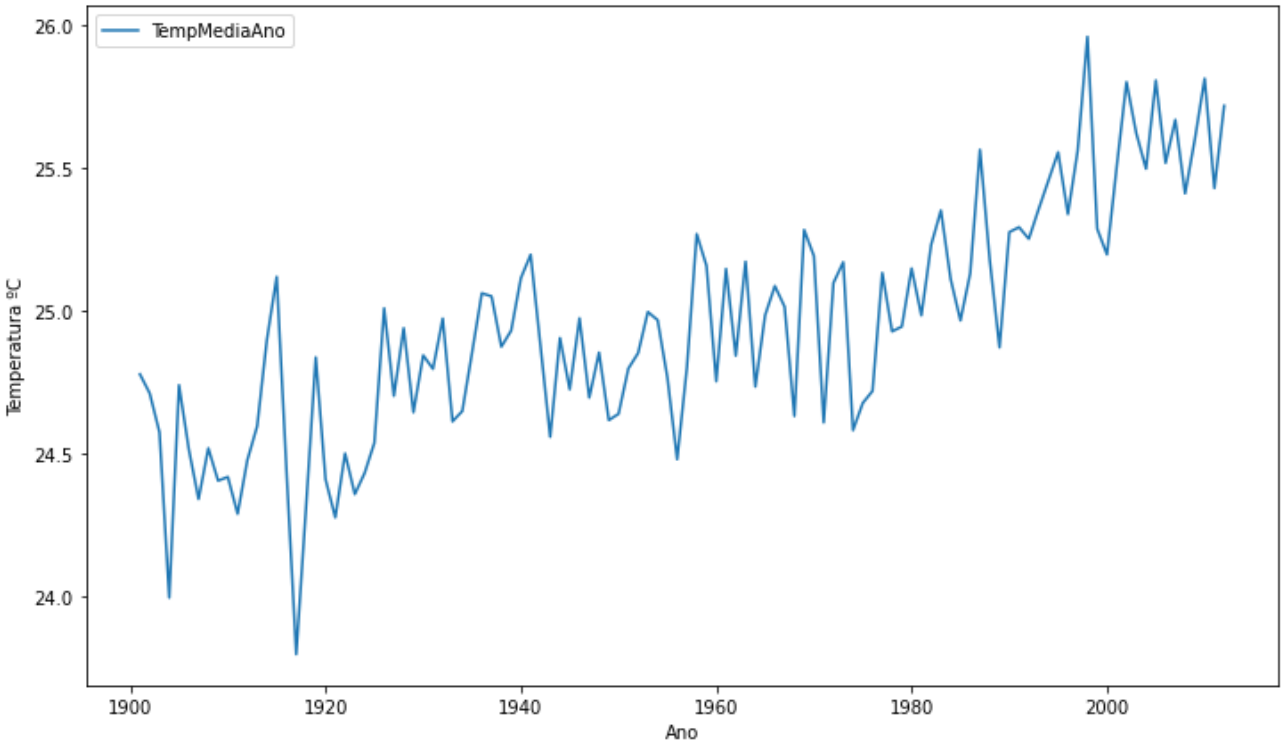
## 3.1 – PLOTAGEM PANDAS

A Plotagem em Pandas foi realizada a partir do dataset 4 (junção de temperatura + CO2), imediatamente após a criação e tratamento deste dataset, utilizando o mesmo ambiente Google Collaboratory. O intuito foi de pré-visualizar algumas tendências, e trazemos aqui esse recorte feito com um filtro para o Brasil.

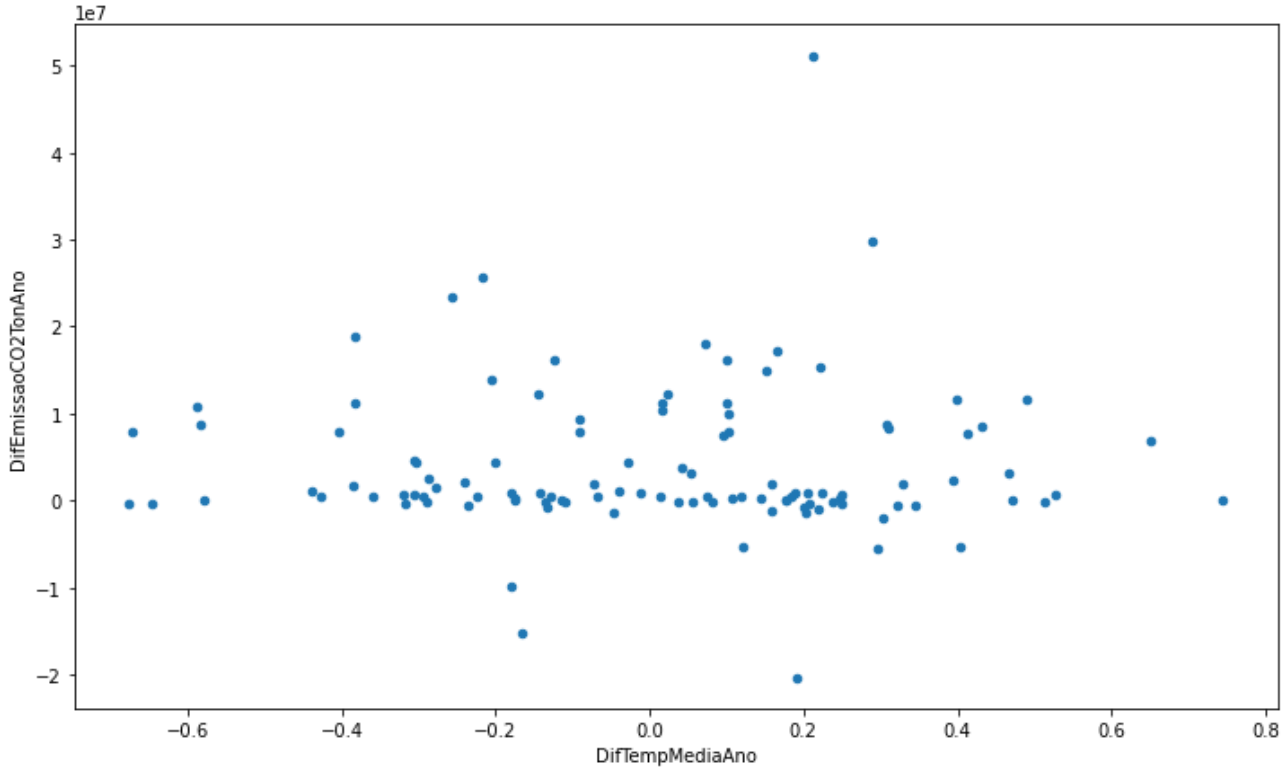
A primeira plotagem, em gráfico de barras, traz a evolução da emissão de carbono, anualmente, no período de 1900 a 2012.




Em seguida a plotagem, em gráfico de linha, trazendo a evolução da temperatura média no país no mesmo períodoo.



Por último, o gráfico de dispersão foi utilizado para observar uma possível correlação entre as duas variáveis, emissão de CO2 da temperatura média.




De mesmo modo esse caminho de análise também será reproduzido na etapa de consultas, observando mais variáveis.

## 3.2 – BIG QUERY

As consultas aqui foram pensadas de modo a construir uma sequência que traga informações que venham a agregar nos estudos acerca do tema, voltadas para as possíveis correlações das variáveis que trouxemos para o centro do projeto.

Desse modo, as queries foram construídas em blocos: primeiramente análises das temperaturas médias e suas variações em variados contextos; em seguida, o mesmo foi aplicado para as emissões de CO2; já no terceiro bloco de consultas, os códigos foram escritos a fim de trazer visualizações da relação destas duas variáveis. Por fim foram feitas

análises do dataset de cobertura florestal bem como suas possíveis relações com o aumento da temperatura em determinados contextos.

**Temperatura**

1. Esta query nos permite observar os países, com seu respectivo continente e região, com os maiores valores de temperatura média dentro do período observado (1900 a 2012), ordenados de forma decrescente. aqui vemos os países mais quentes

```
CREATE OR REPLACE TABLE `projetofinal-grupo5.mudancas_climaticas.query1` AS (
SELECT
  Pais,
  Continente,
  Regiao,
  AVG(TempMediaAno) AS TempMedia,
FROM
  `projetofinal-grupo5.mudancas_climaticas.temp_co2`
GROUP BY
  Pais,
  Continente,
  Regiao
ORDER BY
  TempMedia DESC
)
```

2. Esta query nos permite observar os países, com seu respectivo continente e região, com os menores (ordenados de forma crescente) valores de temperatura média dentro do período observado (1900 a 2012).

```
CREATE OR REPLACE TABLE `projetofinal-grupo5.mudancas_climaticas.query2` AS (
SELECT
  Pais,
  Continente,
  Regiao,
  SUM(DifTempMediaAno) AS TempMedia,
FROM
  `projetofinal-grupo5.mudancas_climaticas.temp_co2`
GROUP BY
  Pais,
  Continente,
  Regiao
ORDER BY
  TempMedia
)
```

As querys um e dois são interessantes porque nos mostram onde estão localizados os países mais quentes e mais frios e também sugere futuramente relação com os aumentos de temperaturas que serão observados.

3. Esta query nos permite observar os países e regiões em que houve maior variação de temperatura acumulada dentro do período observado, apresentando dados com valores bastante significativos as cinco primeiras linhas retornam valores de variação de temperatura acima de 2 ºC positivos em um período de 113 anos

```
CREATE OR REPLACE TABLE
  `projetofinal-grupo5.mudancas_climaticas.query3` AS(
  WITH
    temp AS(
    SELECT
      Pais,
      Continente,
      Regiao,
      SUM(DifTempMediaAno) AS totalVarTempMedia,
    FROM
      `projetofinal-grupo5.mudancas_climaticas.temp_co2`
    GROUP BY
      Pais,
      Continente,
      Regiao
    ORDER BY
      totalVarTempMedia DESC
    LIMIT
      20)
  SELECT
    Regiao,
    COUNT(Regiao) AS rankingRegiao
  FROM
    temp
  GROUP BY
    Regiao
  ORDER BY
    rankingRegiao DESC
   )
```

3.1 Esta query traz a ordem de países com maior variação de temperatura, agrupados por região e ranqueados. Verificou-se que o Oriente Médio e regiões do extremo norte na frente, induzindo a ideia de recortar dados da América do Norte e Oriente Médio posteriormente para avaliar relações.

```sql
CREATE OR REPLACE TABLE `projetofinal-grupo5.mudancas_climaticas.query3` AS(
SELECT
  Pais,
  Continente,
  Regiao,
  SUM(DifTempMediaAno) AS totalVarTempMedia,
FROM
  `projetofinal-grupo5.mudancas_climaticas.temp_co2`
GROUP BY
  Pais,
  Continente,
  Regiao
ORDER BY
  totalVarTempMedia DESC
)
```

As querys 4 a 6 são voltadas para observar a emissão de CO2 relacionadas na ordem de continentes, regiões e países. Também foram feitos os recortes para o século XXI incluindo a taxa de variação da emissão de CO2 neste período nas querys 4-1 a 6-1.

**Emissão de CO2**

4. Soma de emissões de CO2 por continente.

```sql
CREATE OR REPLACE TABLE `projetofinal-grupo5.mudancas_climaticas.query4` AS (
SELECT
  Continente,
  SUM(EmissaoCO2TonAno) AS EmissaoCO2Continente
FROM
  `projetofinal-grupo5.mudancas_climaticas.temp_co2`
GROUP BY
  Continente
ORDER BY
  EmissaoCO2Continente DESC
)
```

4.1 Recorte da query anterior , trazendo a soma das emissões de CO2 por continente e também a variação das taxas anuais de emissão de CO2, no período referente ao século XXI

```sql
CREATE TABLE `projetofinal-grupo5.mudancas_climaticas.query4-1` AS (
SELECT
  Continente,
  SUM(EmissaoCO2TonAno) AS EmissaoCO2ContinenteSecXXI,
  AVG(DifEmissaoCO2TonAno) AS TaxaEmissaoCO2SecXXI
FROM
  `projetofinal-grupo5.mudancas_climaticas.temp_co2`
WHERE Ano > 2000
GROUP BY
  Continente
ORDER BY
  EmissaoCO2ContinenteSecXXI DESC
)
```

5. Soma de emissões de CO2 por região

```sql
CREATE OR REPLACE TABLE `projetofinal-grupo5.mudancas_climaticas.query5` AS (
SELECT
  Regiao,
  SUM(EmissaoCO2TonAno) AS EmissaoCO2Regiao
FROM
  `projetofinal-grupo5.mudancas_climaticas.temp_co2`
GROUP BY
  Regiao
ORDER BY
  EmissaoCO2Regiao DESC
)
```

5.1 Esta query foi feita sobre um recorte da query anterior, trazendo o período do século XXI, onde foi adicionado a média da taxa de emissão de CO2 dos países agrupados por região.

```sql
CREATE OR REPLACE TABLE `projetofinal-grupo5.mudancas_climaticas.query5-1` AS (
SELECT
  Regiao,
  SUM(EmissaoCO2TonAno) AS EmissaoCO2Regiao,
  AVG(DifEmissaoCO2TonAno) AS TaxaEmissaoCO2SecXXI
FROM
  `projetofinal-grupo5.mudancas_climaticas.temp_co2`
WHERE Ano>2000
GROUP BY
  Regiao
ORDER BY
  EmissaoCO2Regiao DESC
)
```

6. Soma de emissões de CO2 por país

```
CREATE OR REPLACE TABLE `projetofinal-grupo5.mudancas_climaticas.query6` AS (
SELECT
  Pais,
  SUM(EmissaoCO2TonAno) AS EmissaoCO2Pais
FROM
  `projetofinal-grupo5.mudancas_climaticas.temp_co2`
GROUP BY
  Pais
ORDER BY
  EmissaoCO2Pais DESC
)
```

6.1 Recorte criado em relação a query anterior, onde foi acrescentada a média da taxa de emissão de CO2 de cada país no período do século XXI.

```
CREATE OR REPLACE TABLE `projetofinal-grupo5.mudancas_climaticas.query6-1` AS (
SELECT
  Pais,
  SUM(EmissaoCO2TonAno) AS EmissaoCO2Pais,
  AVG(DifEmissaoCO2TonAno) AS TaxaEmissaoCO2SecXXI
FROM
  `projetofinal-grupo5.mudancas_climaticas.temp_co2`
WHERE ANO > 2000
GROUP BY
  Pais
ORDER BY
  EmissaoCO2Pais DESC
)
```

7. Soma de emissão de CO2 por nível de desenvolvimento. Esta query traz a quantidade de emissão de CO2 por parte dos países desenvolvidos e em desenvolvimento, além da variação de emissão de CO2 no período de 1900 a 2012.

   A tabela resultante nos permite observar que, embora representem apenas um quarto (aproximadamente) do total de países listados do grupo analisado, os países desenvolvidos ainda assim possuem mais do que o dobro de emissões de CO2 que os países em desenvolvimento.

```sql
CREATE OR REPLACE TABLE `projetofinal-grupo5.mudancas_climaticas.query7` AS (
SELECT
  NivelDesenvolvimento,
  COUNT(DISTINCT Pais) AS QtdPaises,
  SUM(EmissaoCO2TonAno) AS EmissaoCO2NivelDesenvolvimento,
  ROUND(SUM(DifEmissaoCO2TonAno),5) AS varEmissaoCO2
FROM
  `projetofinal-grupo5.mudancas_climaticas.temp_co2`
GROUP BY
  NivelDesenvolvimento
ORDER BY
  EmissaoCO2NivelDesenvolvimento DESC
 )
```

As próximas consultas, 7, 8, e 9 foram feitas com intuito de estreitar a possibilidade de relacionar a emissão de CO2 , as temperaturas médias e as variações de temperatura média.

8. Esta query traz tanto a emissão de CO2 como a variação de temperatura média dos países agrupados por continentes no período de 1900 a 2012 ordenado por emissão de CO2, permitindo que busquemos uma possível relação entre Emissão de CO2 e variação de temperatura média dos diferentes Países, dando atenção também à localização geográfica - Continente e Região no período observado 1900 a 2012.

```sql
CREATE OR REPLACE TABLE `projetofinal-grupo5.mudancas_climaticas.query8` AS(
SELECT
  Pais,
  Continente,
  Regiao,
  SUM(EmissaoCO2TonAno) AS SomaEmissaoCO2,
  ROUND(SUM(DifTempMediaAno),5) AS VarDifTempMedia
FROM
  `projetofinal-grupo5.mudancas_climaticas.temp_co2`
GROUP BY
  Pais,
  Continente,
  Regiao
ORDER BY
  SomaEmissaoCO2 DESC
)
```

9. Análoga à consulta número 8, porém ordenado pelas variações de temperatura.

```sql
CREATE OR REPLACE TABLE `projetofinal-grupo5.mudancas_climaticas.query9` AS (
SELECT
  Pais,
  Continente,
  Regiao,
  SUM(EmissaoCO2TonAno) AS SomaEmissaoCO2,
  ROUND(SUM(DifTempMediaAno),5) AS VarDifTempMedia
FROM
  `projetofinal-grupo5.mudancas_climaticas.temp_co2`
GROUP BY
  Pais,
  Continente,
  Regiao
ORDER BY
  VarDifTempMedia DESC
)
```

Após comparar os resultados das queries 8 e 9 podemos sugerir que as duas variáveis são diretamente proporcionais em muitos casos, é que o aumento mais acentuado da temperatura parece estar também relacionado à localização geográfica.

As queries de número 10 a 13 apresentam em comum o recorte por algumas regiões, trazendo os dados também de temperatura média, médias das variações de temperatura e emissões de CO2.

10. Oriente Médio. Observou-se um aumento significativo da temperatura, e valor não tão significativo em termos de CO2.

```sql
CREATE OR REPLACE TABLE `projetofinal-grupo5.mudancas_climaticas.query10` AS(
SELECT
  Pais,
  Regiao,
  ROUND(SUM(DifTempMediaAno),5) AS VarTotalTempMedia,
  ROUND(SUM(EmissaoCO2TonAno),5) AS SomaEmissaoCO2
FROM
  `projetofinal-grupo5.mudancas_climaticas.temp_co2`
GROUP BY
  Pais,
  Regiao
HAVING
  Regiao = 'Ásia Oeste'
ORDER BY
  SomaEmissaoCO2 DESC
)
```

11. América do Norte. Observou-se aumento significativo de temperatura quanto mais ao norte, emissão maior mais ao sul (USA) .

```
CREATE OR REPLACE TABLE `projetofinal-grupo5.mudancas_climaticas.query11` AS(
SELECT
  Pais,
  Regiao,
  SUM(DifTempMediaAno) AS totalDifTempMedia,
  SUM(EmissaoCO2TonAno) AS somaEmissaoCO2
FROM
  `projetofinal-grupo5.mudancas_climaticas.temp_co2`
GROUP BY
  Pais,
  Regiao
HAVING
  Regiao = 'América do Norte'
ORDER BY
  totalDifTempMedia DESC
 )
```

12. Europa. Observou-se aumento significativo de temperatura quanto mais ao norte, emissão maior mais ao sul.

```
CREATE OR REPLACE TABLE `projetofinal-grupo5.mudancas_climaticas.query12` AS(
WITH
  temp AS(
  SELECT
    Pais,
    Continente,
    Regiao,
    SUM(DifTempMediaAno) AS totalVarTempMedia,
    SUM(EmissaoCO2TonAno) AS SomaEmissao
  FROM
    `projetofinal-grupo5.mudancas_climaticas.temp_co2`
  GROUP BY
    Pais,
    Continente,
    Regiao
  ORDER BY
    totalVarTempMedia DESC )
SELECT
  Regiao,
  AVG(totalVarTempMedia) AS MediaVarTempRegiao,
  SUM(SomaEmissao) AS SomaEmissaoRegiao
FROM
  temp
WHERE
  Continente='Europa'
GROUP BY
  Regiao
ORDER BY
  MediaVarTempRegiao DESC)
```

13. Brasil. Não foi possível verificar uma relação entre as duas váriáveis no Brasil.

```sql
CREATE OR REPLACE TABLE `projetofinal-grupo5.mudancas_climaticas.query13` AS(
SELECT
  Pais
  Regiao,
  SUM(DifTempMediaAno) AS totalDifTempMedia,
  SUM(EmissaoCO2TonAno) AS somaEmissaoCO2,
  AVG(TempMediaAno) AS TempMedia
FROM
  `projetofinal-grupo5.mudancas_climaticas.temp_co2`
GROUP BY
  Pais,
  Regiao
HAVING
  Pais = 'Brazil'
ORDER BY
  totalDifTempMedia DESC
 )
```

**Cobertura florestal**

Nas consultas de número 14 a 16, o intuito foi avaliar apenas a tabela de cobertura florestal com resultados de cobertura florestal em hectares, variações em hectares, e percentuais em relação à área de cobertura no país.

14. Países com maior cobertura florestal em 2010 (ano referência para nosso estudo). Esta consulta nos permite avaliar se os países com maior cobertura florestal foram pouco ou muito afetados pelas mudanças climáticas.

```sql
CREATE OR REPLACE TABLE `projetofinal-grupo5.mudancas_climaticas.query14` AS (
SELECT
  Pais,
  AreaFlorestal_2010_1000ha_
FROM
  `projetofinal-grupo5.mudancas_climaticas.area-florestal`
ORDER BY
  AreaFlorestal_2010_1000ha_ DESC
  )
```

15. Países com menor cobertura florestal em 2010 (ano referência para nosso estudo). Esta consulta nos permite avaliar se os países com menor cobertura florestal foram pouco ou muito afetados pelas mudanças climáticas.

```sql
CREATE OR REPLACE TABLE `projetofinal-grupo5.mudancas_climaticas.query15` AS (
SELECT
  Pais,
  AreaFlorestal_2010_1000ha_
FROM
  `projetofinal-grupo5.mudancas_climaticas.area-florestal`
ORDER BY
  AreaFlorestal_2010_1000ha_
  )
```

16. Países que mais perderam cobertura florestal entre 1990 e 2010. Esta consulta nos permite avaliar se algum desses países aumentou a temperatura consideravelmente no período.

```sql
CREATE OR REPLACE TABLE `projetofinal-grupo5.mudancas_climaticas.query16` AS (
SELECT
  Pais,
  AreaFlorestal_1990_1000ha_,
  AreaFlorestal_2010_1000ha_,
  Variacao_1990_2010,
  VarPercentual_1990_2010
FROM
  `projetofinal-grupo5.mudancas_climaticas.area-florestal`
ORDER BY
  Variacao_1990_2010
  )
```

As consultas seguintes, de 17 a 21, tiveram o intuito de unir os dados da tabela de cobertura florestal com a tabela salva,de variação de temperatura e emissão de CO2, no recorte de período e território específicos, referentes às queries anteriores.

17. Soma CO2, média e variação de temperatura e cobertura florestal no Oriente Médio entre 1990 e 2010. É possível observar forte correlação entre a falta de cobertura florestal e altos níveis de emissão de CO2 com a variação sensível de temperatura.

```sql
CREATE OR REPLACE TABLE `projetofinal-grupo5.mudancas_climaticas.query17` AS(
WITH temp AS(
SELECT
  Pais,
  Regiao,
  ROUND(SUM(DifTempMediaAno),5) AS VarTotalTempMedia,
  ROUND(SUM(EmissaoCO2TonAno),5) AS SomaEmissaoCO2
FROM
  `projetofinal-grupo5.mudancas_climaticas.temp_co2`
WHERE Regiao='Ásia Oeste' AND Ano BETWEEN 1990 AND 2010
GROUP BY
  Pais,
  Regiao
ORDER BY
  VarTotalTempMedia DESC
)
SELECT
a.*,
b.AreaFlorestal_1990_1000ha_,
b.AreaFlorestal_2010_1000ha_,
b.Variacao_1990_2010,
b.VarPercentual_1990_2010
FROM temp AS a
INNER JOIN
  `projetofinal-grupo5.mudancas_climaticas.area-florestal` AS b
ON
  a.Pais = b.Pais
)
```

18. Soma CO2, média e variação de temperatura e cobertura florestal na China entre 1990 e 2010. Há uma possível forte correlação entre a falta de cobertura florestal e altos níveis de emissão de CO2 com a variação sensível de temperatura.

```sql
CREATE OR REPLACE TABLE `projetofinal-grupo5.mudancas_climaticas.query18` AS(
WITH temp AS(
SELECT
  Pais,
  ROUND(SUM(DifTempMediaAno),5) AS VarTotalTempMedia,
  ROUND(SUM(EmissaoCO2TonAno),5) AS SomaEmissaoCO2
FROM
  `projetofinal-grupo5.mudancas_climaticas.temp_co2`
WHERE Pais= 'China' AND Ano BETWEEN 1990 AND 2010
GROUP BY
  Pais,
  Regiao
)
SELECT
a.*,
b.AreaFlorestal_1990_1000ha_,
b.AreaFlorestal_2010_1000ha_,
b.Variacao_1990_2010,
b.VarPercentual_1990_2010
FROM temp AS a
INNER JOIN
  `projetofinal-grupo5.mudancas_climaticas.area-florestal` AS b
ON
  a.Pais = b.Pais
```

19. Soma CO2, média e variação de temperatura e cobertura florestal na Europa entre 1990 e 2010. É possível observar forte correlação entre a falta de cobertura florestal e altos níveis de emissão de CO2 com a variação sensível de temperatura.

```sql
CREATE OR REPLACE TABLE `projetofinal-grupo5.mudancas_climaticas.query19` AS(
WITH temp AS(
SELECT
  Pais,
  Regiao,
  ROUND(SUM(DifTempMediaAno),5) AS VarTotalTempMedia,
  ROUND(SUM(EmissaoCO2TonAno),5) AS SomaEmissaoCO2
FROM
  `projetofinal-grupo5.mudancas_climaticas.temp_co2`
WHERE Continente='Europa' AND Ano BETWEEN 1990 AND 2010
GROUP BY
  Pais,
  Regiao
ORDER BY
  VarTotalTempMedia DESC
)
SELECT
a.*,
b.AreaFlorestal_1990_1000ha_,
b.AreaFlorestal_2010_1000ha_,
b.Variacao_1990_2010,
b.VarPercentual_1990_2010
FROM temp AS a
INNER JOIN
  `projetofinal-grupo5.mudancas_climaticas.area-florestal` AS b
ON
  a.Pais = b.Pais
)
```

20. Soma CO2, média e variação de temperatura e cobertura florestal no Brasil entre 1990 a 2010.

```sql
CREATE OR REPLACE TABLE `projetofinal-grupo5.mudancas_climaticas.query20` AS(
WITH temp AS(
SELECT
  Pais,
  ROUND(SUM(DifTempMediaAno),5) AS VarTotalTempMedia,
  ROUND(SUM(EmissaoCO2TonAno),5) AS SomaEmissaoCO2
FROM
  `projetofinal-grupo5.mudancas_climaticas.temp_co2`
WHERE Pais= 'Brazil' AND Ano BETWEEN 1990 AND 2010
GROUP BY
  Pais,
  Regiao
)
SELECT
a.*,
b.AreaFlorestal_1990_1000ha_,
b.AreaFlorestal_2010_1000ha_,
b.Variacao_1990_2010,
b.VarPercentual_1990_2010
FROM temp AS a
INNER JOIN
  `projetofinal-grupo5.mudancas_climaticas.area-florestal` AS b
ON
  a.Pais = b.Pais
)
```

Baseados nos resultados das nossas queries, bem como nas visualizações no Data Studio, pudemos chegar aos seguintes insights:

1) Segundo nossos dados, a emissão de CO2 e a variação de temperatura aumentaram substancialmente ao longo de todo o século XX e início do século XXI. Pesquisas da área apontam que a emissão de CO2 é um dos fatores relevantes na intensificação das mudanças climáticas, sendo que nossos dados suportam esta tese.
2) A relação direta entre emissão de CO2 e variação de temperatura em um mesmo país pode ou não ser significativa. Temos casos como o da Rússia, por exemplo, no qual a emissão de CO2 e o aumento da temperatura seguiram uma tendência forte de alta no mundo, sendo este país o terceiro que mais emite CO2 e também o terceiro que mais registrou aumento de temperatura no período histórico considerado. No entanto, outros países não seguem essa regra, como é o caso da

China, a maior emissora de CO2 do planeta, que registrou um aumento leve de temperatura no período registrado. Isso nos leva a crer que outros fatores também são determinantes para as mudanças climáticas, e nossos dados sugerem que tais fatores incluem a localização geográfica e o desmatamento / reflorestamento.

3) No que diz respeito à localização geográfica, pudemos observar uma concentração de países com elevado aumento de temperatura em duas regiões do planeta:
   a) no extremo norte, incluindo a Groenlândia (maior aumento de temperatura no mundo no período pesquisado), a Rússia, Noruega e Canadá, entre outros
   b) na zona do Equador, especialmente no oeste da Ásia (Oriente Médio) e norte da África
4) Em relação à cobertura florestal e sua variação, observamos que os países na zona do Equador, que sofrem com aumento elevado da temperatura, contam com uma cobertura florestal muito pequena, a menor em todo o mundo. Saltou ainda aos nossos olhos a diferença entre variação de cobertura florestal no Brasil e na China, relacionados à emissão de CO2 e aumento da temperatura. A China, maior emissor de CO2 do mundo, é também o país que mais área reflorestou entre 1990 e 2010, fato que pode ter contribuído para um aumento bastante ameno de temperatura. Já o Brasil, apesar da baixa emissão de CO2, foi o país que mais desmatou no mesmo período, no qual sua temperatura se elevou quase três vezes mais do que a China. Sendo assim, nossos dados sugerem que o reflorestamento e desmatamento podem contribuir positivamente ou negativamente no que diz respeito às mudanças climáticas.