



**Nome dos Colaboradores:**

Greyce da Costa Silva  
Matheus Rodrigues Gonçalves de Gois  
Mayara Gomes Leite  
Vinicius Gomes Oliveira

**São Paulo, 23 de Fevereiro de 2021**

CONTROLE DE VERSÕES			
Autor	Versão	Data	Descrição
Greyce Silva	0	23/02/2021	Criação do documento.
Greyce Silva	1	01/03/2021	Atualização Capítulo 4
Mayara Leite	2	03/03/2021	Atualização 4.5, 4.6, 4.7 e 4.8
Greyce Silva	3	03/03/2021	Inclusão de nomes e Alteração Item 3.2

## 1. Introdução

Este documento tem o intuito de apresentar o projeto Integrado proposto pelo programa Blueshift Academy.

## 2. Solicitação

O projeto abrange os dados referentes a Emissões de Gases Poluentes e Causadores do efeito estufa, e tem o intuito de analisar quais países e/ou continentes mais emitem esses gases, em quais períodos, se os tratados como Kyoto e o acordo de Paris são de fatos efetivos e Possíveis relações entre Crescimentos Populacionais e PIB com estes países.

A base principal do projeto contém dados de 1751 a 2019, portanto todas as demais bases serão normalizadas para conter dados referente a este período de tempo, exceto por algumas visualizações genéricas.

O projeto será desenvolvido na Plataforma GCP, utilizando a linguagem Python para consumo de dados, tratativas e normatizações, assim como a inserção dos dataframes no bigquery, os Dashboards serão apresentados no Power Bi através da conexão com o Bigquery.

## 3. Premissas da Solução

### 3.1. Origem dos dados e especificações

Os arquivos foram consumidos de diversas formas distintas que devem ser citadas:

- Consumo de arquivo .csv encontrado no site ou no github e realizado a leitura através do pandas;
- Consumo de arquivo .csv no Kaggle através da API e realizado a leitura através do pandas;
- Não foi encontrado nenhum arquivo .csv ou de qualquer outra natureza sobre os dados demandados, porém foi encontrado tabelas no próprio site da UNCTC(United Nations Treaty Collections), foi realizado Web sCrapping utilizando a biblioteca BeautifulSoup para consumo dos dados.

Para informações mais detalhadas verificar especificações abaixo:

- 3.1.1. **Emissão de CO2 e CNH4 e outros gases por ano e Países** (Fonte: <https://ourworldindata.org/co2-and-other-greenhouse-gas-emissions>),

Modo de consumo de dados: Pandas, Read and Load Functions

Notebook:

1. kyoto\_Emission\_country\_year.ipynb, 2. kyoto\_Paris\_Emission\_country\_year.ipynb

Dataframes: Emission\_Country\_Year

Arquivos: Emission\_Country\_Year.csv

- 3.1.2. **Países que assinaram o tratado de kyoto** (Fonte: [https://treaties.un.org/Pages/ViewDetails.aspx?src=TREATY&mtdsg\\_no=XXVII-7-a&chapter=27&clang=en](https://treaties.un.org/Pages/ViewDetails.aspx?src=TREATY&mtdsg_no=XXVII-7-a&chapter=27&clang=en)),

Modo de consumo de dados: Web Scraping, BeautifulSoup;

Notebooks: 1. kyoto\_Emission\_country\_year.ipynb

Dataframes: Emission\_Country\_Year, Kyoto\_Agreement\_Countries

- 3.1.3. **Países que assinaram o Acordo de Paris** (Fonte: [https://treaties.un.org/pages/ViewDetails.aspx?src=TREATY&mtdsg\\_no=XXVII-7-d&chapter=27&clang=en](https://treaties.un.org/pages/ViewDetails.aspx?src=TREATY&mtdsg_no=XXVII-7-d&chapter=27&clang=en)),

Modo de consumo de dados: Web Scraping, BeautifulSoup;

Notebook: 2. kyoto\_Paris\_Emission\_country\_year.ipynb

Dataframes: Emission\_Country\_Year, Paris\_Agreement\_Countries

- 3.1.4. **Países por continente e Região** (Fonte: [Kaggle Datasets](#)),

Modo de consumo de dados: Pandas, Read and Load Functions;

Notebook: 3. Country\_Region\_Continent.ipynb

Dataframes: Country\_Continent\_Region

- 3.1.5. **Temperatura por País e Ano**(Fonte: [Kaggle Datasets](#)),

Modo de consumo de dados: Pandas, Read and Load Functions;

Notebook: 4. Temperatura.ipynb

Dataframes: MediaTempPais, TempPais, MediaTempGlobal, TempPais

- 3.1.6. **Crescimento Populacional por país e ano**  
(Fonte: <https://ourworldindata.org/world-population-growth>),

Modo de consumo de dados: Pandas, Read and Load Functions;

Notebook: 5. GrowthxEmission.ipynb

Dataframes: MediaGrowthPais, GrowthPais

- 3.1.7. **Crescimento PIB por país e ano**(Fonte: <https://ourworldindata.org/economic-growth>),

Modo de consumo de dados: Pandas, Read and Load Functions;

Notebook: 6. PiBxEmission.ipynb

Dataframes: PibPais, MediaPibPais

## 3.2. Desenvolvimento

O desenvolvimento se deu na plataforma GCP (Google Cloud Platform), utilizando os acessos que foram disponibilizados pela Blueshift. Segue abaixo todas as ferramentas utilizadas para o projeto:

**Cloud Scheduler:** Agendamento de tarefas para ligar e desligar a instância do Compute engine;

**Cloud Pub/Sub:** Gatilho para execução dos scripts do Cloud Functions;

**Cloud Functions:** script de iniciação e desligamento das instâncias do compute engine;

**CronTab:** Execução dos notebooks de 1 a 6 conforme agendamento.

**Compute Engine:** Criação da VM Debian 10;

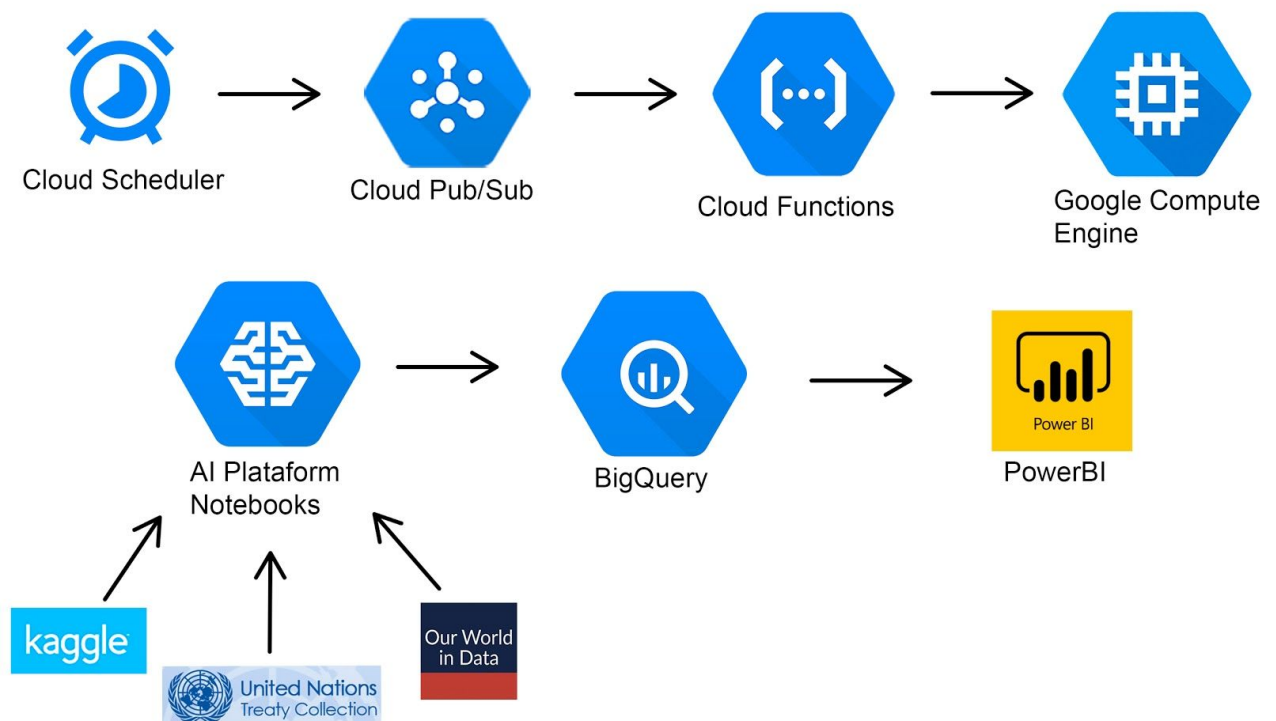
**AI Platform:** Utilização do Jupyter Notebook para Elaboração de códigos em Python;

**BigQuery:** Armazenamento de Dados;

**PowerBi:** Elaboração de Dashboards;

## 4. Arquitetura

Segue representação da Arquitetura do Projeto:



### 4.1. Compute Engine:

#### 4.1.1. VM Debian 10

Propriedades	Valor
Nome	projetointegrado
Série	N1
Tipo de Máquina	n1-standard-4 (4 vCPUs, 15 GB de memória)
Disco de Utilização	Debian 10
Tamanho do Disco	100 GB
Identidade e Acesso a API	Permitir acesso completo a todas as APIs do Cloud
Firewall	Padrão

#### 4.2. AI Platform:

Foi utilizado o Jupyter Lab ambiente TensorFlow:2.3 para desenvolvimento dos códigos em Python, foram utilizados os seguintes notebooks, conforme abaixo:

1. kyoto\_Emission\_country\_year.ipynb;
2. kyoto\_Paris\_Emission\_country\_year.ipynb;
3. Country\_Region\_Continent.ipynb;
4. Temperatura.ipynb;
5. GrowthxEmission.ipynb;
6. PiBxEmission.ipynb;

Link dos Notebooks:

<https://github.com/matheusgois97/projetointegrado/tree/main/Scripts>

#### 4.3. Bigquery

Foram armazenados os dados nas tabelas do bigquery, tanto os dados consumidos quanto dados após a tratativas, todas a tabelas com o final “\_in” são dados consumidos diretamente das fontes sem nenhum tratamento, segue abaixo esquemas de dados tratados que serão utilizados para a exibição do dashboard proposto, é importante ter em mente que não houve nenhum código executado no bigquery pois tudo foi executado no Jupyter notebook.

#### 4.3.1. Esquemas de Tabelas

##### 4.3.1.1. Tabela Emission\_Country\_Year

Nome do campo	Tipo	Modo
iso_code	STRING	NULLABLE
country	STRING	NULLABLE
year	INTEGER	NULLABLE
co2	FLOAT	NULLABLE
co2_per_capita	FLOAT	NULLABLE
share_global_co2	FLOAT	NULLABLE
cumulative_co2	FLOAT	NULLABLE
share_global_cumulative_co2	FLOAT	NULLABLE
co2_per_gdp	FLOAT	NULLABLE
cement_co2	FLOAT	NULLABLE
coal_co2	FLOAT	NULLABLE
flaring_co2	FLOAT	NULLABLE
gas_co2	FLOAT	NULLABLE
oil_co2	FLOAT	NULLABLE
other_industry_co2	FLOAT	NULLABLE
cement_co2_per_capita	FLOAT	NULLABLE
coal_co2_per_capita	FLOAT	NULLABLE
flaring_co2_per_capita	FLOAT	NULLABLE
gas_co2_per_capita	FLOAT	NULLABLE
oil_co2_per_capita	FLOAT	NULLABLE
other_co2_per_capita	FLOAT	NULLABLE
share_global_coal_co2	FLOAT	NULLABLE
share_global_oil_co2	FLOAT	NULLABLE
share_global_gas_co2	FLOAT	NULLABLE
share_global_flaring_co2	FLOAT	NULLABLE
share_global_cement_co2	FLOAT	NULLABLE
cumulative_coal_co2	FLOAT	NULLABLE

cumulative_oil_co2	FLOAT	NULLABLE
cumulative_gas_co2	FLOAT	NULLABLE
cumulative_flaring_co2	FLOAT	NULLABLE
cumulative_cement_co2	FLOAT	NULLABLE
share_global_cumulative_coal_co2	FLOAT	NULLABLE
share_global_cumulative_oil_co2	FLOAT	NULLABLE
share_global_cumulative_gas_co2	FLOAT	NULLABLE
share_global_cumulative_flaring_co2	FLOAT	NULLABLE
share_global_cumulative_cement_co2	FLOAT	NULLABLE
total_ghg	FLOAT	NULLABLE
ghg_per_capita	FLOAT	NULLABLE
methane	FLOAT	NULLABLE
methane_per_capita	FLOAT	NULLABLE
nitrous_oxide	FLOAT	NULLABLE
nitrous_oxide_per_capita	FLOAT	NULLABLE
population	FLOAT	NULLABLE
gdp	FLOAT	NULLABLE
Kyoto	BOOLEAN	NULLABLE
Paris	BOOLEAN	NULLABLE

#### 4.3.1.2. Tabela Country\_Continent\_Region

Nome do campo	Tipo	Modo
country	STRING	NULLABLE
code_3	STRING	NULLABLE
continent	STRING	NULLABLE
sub_region	STRING	NULLABLE
country_code	STRING	NULLABLE

#### 4.3.1.3. Tabela Temperature\_Country\_Year (fAverageTemperatureByCountry)



Nome do campo	Tipo	Modo
year	INTEGER	NULLABLE
iso_code	STRING	NULLABLE
country	STRING	NULLABLE
AverageTemperature	FLOAT	NULLABLE

4.3.1.4. Tabela Growth\_Country\_Year (MediaGrowthPais)

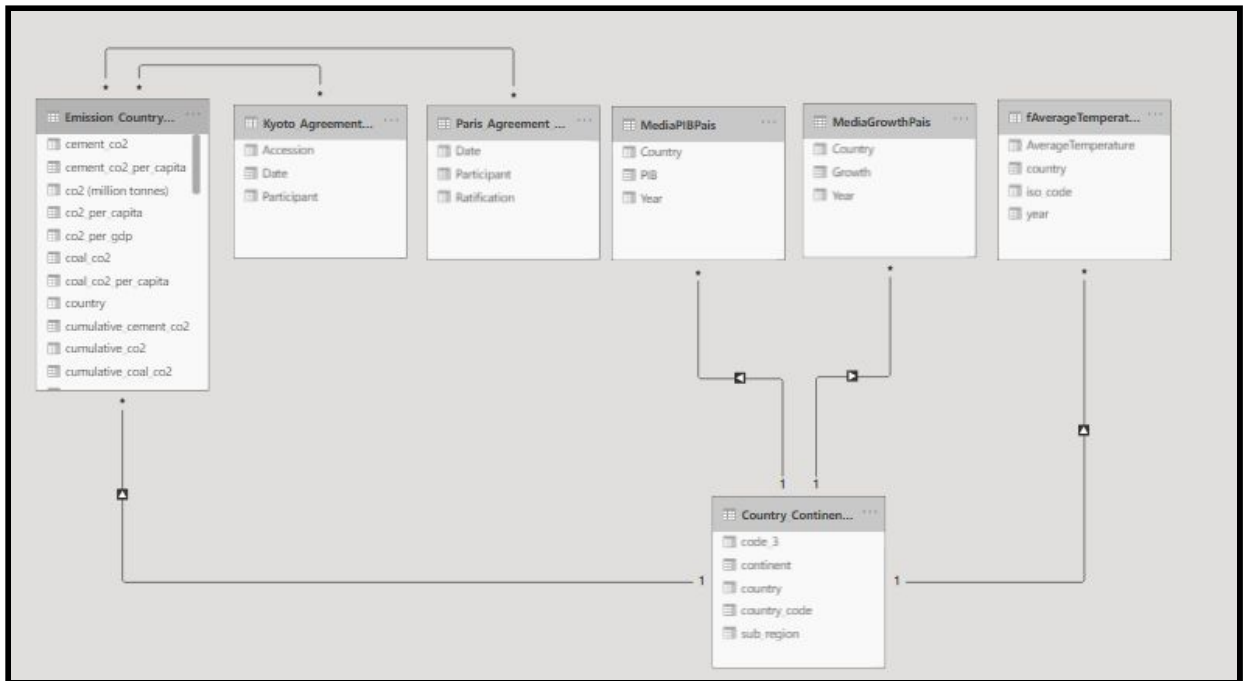
Nome do campo	Tipo	Modo
Year	INTEGER	NULLABLE
Country	STRING	NULLABLE
Growth	INTEGER	NULLABLE

4.3.1.5. Tabela PIB\_Country\_Year (MediaPIBPais)

Nome do campo	Tipo	Modo
Year	INTEGER	NULLABLE
Country	STRING	NULLABLE
PIB	INTEGER	NULLABLE

#### 4.4. Power BI

Segue relacionamentos entre todas as tabelas no power BI:





#### 4.5. Cloud Scheduler

Foram criados dois Jobs, sendo um para iniciar a instância do Compute Engine nomeado como “startInstanceScheduler” e o outro interromperá a execução da instância, “stopInstancesScheduler”. O objetivo das ações é garantir que a instância só estará ativa durante as execuções dos scripts Python, assim não será necessário ação humana, além de diminuir os custos com a ferramenta. Estes Jobs enviarão mensagens aos tópicos Pub/Sub nos horários agendados. Segue abaixo visualização dos jobs agendados no schedule:

	Cloud Scheduler	Jobs	<a href="#">+ CRIAR JOB</a>	<a href="#">ATUALIZAR</a>	<a href="#">EDITAR</a>	<a href="#">PAUSAR</a>
<a href="#">SCHEDULER JOBS</a>		CRON JOBS DO APP ENGINE 				
	Filtro	Filtrar jobs				
<input type="checkbox"/>	Nome ↑	Estado	Descrição	Frequência	Destino	
<input type="checkbox"/>	startInstanceScheduler	Ativado	Iniciar instância	0 1 * * * (America/Sao_Paulo)	Tópico : start-instance-event	
<input type="checkbox"/>	stopInstancesScheduler	Ativado	Interromper instância	0 2 * * * (America/Sao_Paulo)	Tópico : stop-instance-event	

#### 4.6. Cloud Pub/Sub

Esta ferramenta será usada como gatilho para executar os scripts criados no Cloud Functions. Foram criados dois tópicos que serão acionados pelos Jobs criados Cloud Scheduler, estes tópicos são os responsáveis por ativar a execução das funções de iniciar e interromper a instância. Os tópicos foram nomeados da seguinte maneira: “start-instance-event” (iniciar instância) e “stop-instance-event” (interromper instância). Segue abaixo visualização dos tópicos criados no Pub/Sub:

	Tópicos	<a href="#">+ CRIAR TÓPICO</a>	<a href="#">EXCLUIR</a>
	Filtro	Filter topics	
<input type="checkbox"/>	ID do tópico ↑	Chave de criptografia	Nome do tópico
<input type="checkbox"/>	start-instance-event	Google-managed	projects/blueshift-academy/topics/start-i...
<input type="checkbox"/>	stop-instance-event	Google-managed	projects/blueshift-academy/topics/stop-i...

#### 4.7. Cloud Functions

Ferramenta utilizada para automatização de atividades onde foram desenvolvidas as funções que iniciam e interrompem a instância. Estas funções possuem como gatilho os tópicos criados no Cloud Pub/Sub, estes que enviam o comando para executar a função. Nomeação das funções:

“startInstancePubSub” (iniciar instância) e “stopInstancePubSub” (interromper instância). Segue abaixo visualização dos funções no cloud functions:

Cloud Functions		Funções					
		+ CRIAR FUNÇÃO ATUALIZAR					
Filtro		Filtrar funções					
<input type="checkbox"/>	<input type="radio"/>	Nome ↑	Região	Gatilho	Tempo de execução	Memória alocada	Função executada
<input type="checkbox"/>	<input checked="" type="radio"/>	startInstancePubSub	us-central1	Tópico start-instance-event	Node.js 10	256 MIB	startInstancePubSub
<input type="checkbox"/>	<input checked="" type="radio"/>	stopInstancePubSub	us-central1	Tópico stop-instance-event	Node.js 10	256 MIB	stopInstancePubSub

Segue códigos configurados:

### *startInstancePubSub*

```
const Compute = require('@google-cloud/compute');
const compute = new Compute();

/**
 * Starts Compute Engine instances.
 *
 * Expects a PubSub message with JSON-formatted event data containing the
 * following attributes:
 * zone - the GCP zone the instances are located in.
 * label - the label of instances to start.
 *
 * @param {!object} event Cloud Function PubSub message event.
 * @param {!object} callback Cloud Function PubSub callback indicating
 * completion.
 */
exports.startInstancePubSub = async (event, context, callback) => {
  try {
    const payload = _validatePayload(
      JSON.parse(Buffer.from(event.data, 'base64').toString())
    );
    const options = {filter: `labels.${payload.label}`};
    const [vms] = await compute.getVMs(options);
    await Promise.all(
      vms.map(async instance => {
        if (payload.zone === instance.zone.id) {
          const [operation] = await compute
            .zone(payload.zone)
            .vm(instance.name)
            .start();
        }
      })
    );
  } catch (err) {
    console.error(err);
  }
  callback();
}
```

```
        // Operation pending
        return operation.promise();
    }
    })
);

// Operation complete. Instance successfully started.
const message = 'Successfully started instance(s)';
console.log(message);
callback(null, message);
} catch (err) {
    console.log(err);
    callback(err);
}
};

/**
 * Validates that a request payload contains the expected fields.
 *
 * @param {!object} payload the request payload to validate.
 * @return {!object} the payload object.
 */
const _validatePayload = payload => {
    if (!payload.zone) {
        throw new Error("Attribute 'zone' missing from payload");
    } else if (!payload.label) {
        throw new Error("Attribute 'label' missing from payload");
    }
    return payload;
};
```

### ***stopInstancePubSub***

```
const Compute = require('@google-cloud/compute');
const compute = new Compute();

/**
 * Stops Compute Engine instances.
 *
 * Expects a PubSub message with JSON-formatted event data containing the
 * following attributes:
 *   zone - the GCP zone the instances are located in.
 *   label - the label of instances to stop.
 *
 * @param {!object} event Cloud Function PubSub message event.
```

## Grupo A Projeto Integrado - BlueShift

```
* @param {!object} callback Cloud Function PubSub callback indicating
completion.
*/
exports.stopInstancePubSub = async (event, context, callback) => {
  try {
    const payload = _validatePayload(
      JSON.parse(Buffer.from(event.data, 'base64').toString())
    );
    const options = {filter: `labels.${payload.label}`};
    const [vms] = await compute.getVMs(options);
    await Promise.all(
      vms.map(async instance => {
        if (payload.zone === instance.zone.id) {
          const [operation] = await compute
            .zone(payload.zone)
            .vm(instance.name)
            .stop();

          // Operation pending
          return operation.promise();
        } else {
          return Promise.resolve();
        }
      })
    );

    // Operation complete. Instance successfully stopped.
    const message = 'Successfully stopped instance(s)';
    console.log(message);
    callback(null, message);
  } catch (err) {
    console.log(err);
    callback(err);
  }
};

/**
 * Validates that a request payload contains the expected fields.
 *
 * @param {!object} payload the request payload to validate.
 * @return {!object} the payload object.
 */
const _validatePayload = payload => {
  if (!payload.zone) {
    throw new Error("Attribute 'zone' missing from payload");
  } else if (!payload.label) {
    throw new Error("Attribute 'label' missing from payload");
  }
  return payload;
};
```

```
};
```

## 4.8. Crontab

Esta é uma ferramenta nativa do Linux utilizada para executar tarefas agendadas, foi usada para automatizar a execução dos Notebooks Python. Toda a manipulação e criação dos chamados cronjobs (atividades a serem executadas) foi feita diretamente no terminal da VM. Seguem códigos utilizados para agendamento das atividades:

```
10 4 * * * root /opt/conda/bin/jupyter nbconvert --execute
--clear-output /home/jupyter/'1.
kyoto_Emission_country_year(Finalizado).ipynb'

18 4 * * * root /opt/conda/bin/jupyter nbconvert
--execute --clear-output /home/jupyter/'2.
kyoto_Paris_Emission_country_year(Finalizado).ipynb'

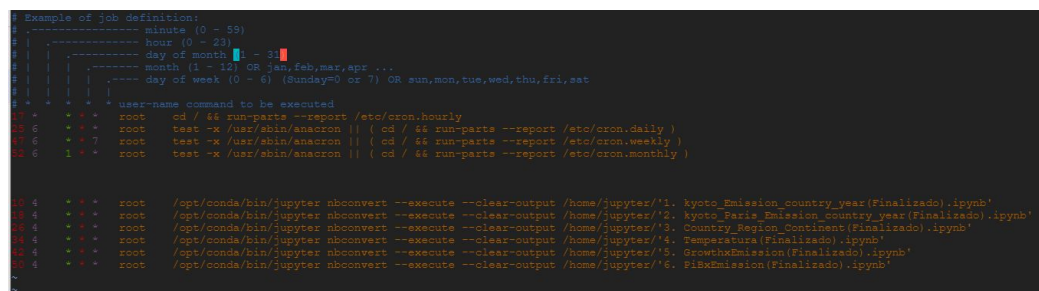
26 4 * * * root /opt/conda/bin/jupyter nbconvert --execute
--clear-output /home/jupyter/'3.
Country_Region_Continent(Finalizado).ipynb'

34 4 * * * root /opt/conda/bin/jupyter nbconvert --execute
--clear-output /home/jupyter/'4. Temperatura(Finalizado).ipynb'

42 4 * * * root /opt/conda/bin/jupyter nbconvert
--execute --clear-output /home/jupyter/'5.
GrowthxEmission(Finalizado).ipynb'

50 4 * * * root /opt/conda/bin/jupyter nbconvert --execute
--clear-output /home/jupyter/'6. PiBxEmission(Finalizado).ipynb'
```

O editor de Crontab utilizado foi Vim. Através do código “*sudo vim /etc/crontab*” é possível visualizar, editar e adicionar cronjobs, conforme abaixo:



```
Example of job definition:
# minute (0 - 59)
# hour (0 - 23)
# day of month (1 - 31)
# month (1 - 12) OR jan,feb,march,apr, ...
# day of week (0 - 6) (Sunday=0 or 7) OR sun,mon,tue,wed,thu,fri,sat
# * * * * * user-name command to be executed
17 * * * * root cd / && run-parts --report /etc/cron.hourly
17 6 * * * root test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.daily )
17 6 * * 7 root test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.weekly )
17 6 1 * * root test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.monthly )
10 4 * * * root /opt/conda/bin/jupyter nbconvert --execute --clear-output /home/jupyter/'1. kyoto_Emission_country_year(Finalizado).ipynb'
18 4 * * * root /opt/conda/bin/jupyter nbconvert --execute --clear-output /home/jupyter/'2. kyoto_Paris_Emission_country_year(Finalizado).ipynb'
26 4 * * * root /opt/conda/bin/jupyter nbconvert --execute --clear-output /home/jupyter/'3. Country_Region_Continent(Finalizado).ipynb'
34 4 * * * root /opt/conda/bin/jupyter nbconvert --execute --clear-output /home/jupyter/'4. Temperatura(Finalizado).ipynb'
42 4 * * * root /opt/conda/bin/jupyter nbconvert --execute --clear-output /home/jupyter/'5. GrowthxEmission(Finalizado).ipynb'
50 4 * * * root /opt/conda/bin/jupyter nbconvert --execute --clear-output /home/jupyter/'6. PiBxEmission(Finalizado).ipynb'
```

O problema desta ferramenta, é que caso a instância não esteja ativa, as atividades não serão executadas, e não seria viável mantê-la ativa sempre. Para resolver, foi implantada a automatização de início e interrupção da instância conforme descrito nos tópicos 4.5, 4.6 e 4.7. Os cronjobs foram

agendados para serem executados no intervalo de atividade da instância, porém devido a diferença de fuso horário entre a instância e o Cloud Scheduler (que pode ser agendado com o horário local, no caso, foi selecionado São Paulo) no crontab foi necessário agendar as atividades entre as 4h e 5h, que correspondem ao horário entre 1h e 2h em São Paulo.

## 5. DashBoards

### 5.1. Visualizações:

As visualizações apresentadas no item relatórios:

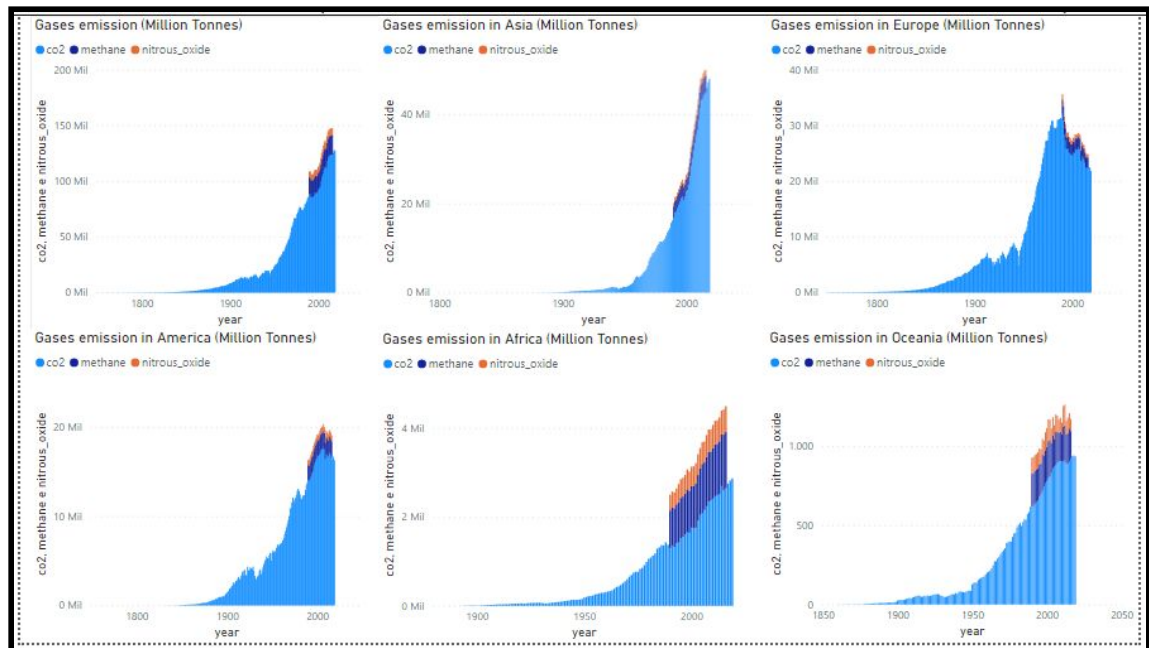
- Emissão de gases por País e Ano, segmentação por Região;
- verificação de impactos dos tratados de Kyoto e Acordo de Paris;
- Relação entre Emissões, PIB e Crescimento Populacional;
- Relação entre Emissão e Temperatura do Planeta;
- Emissão de gases por indústria e por País e ano.

### 5.2. Relatórios:

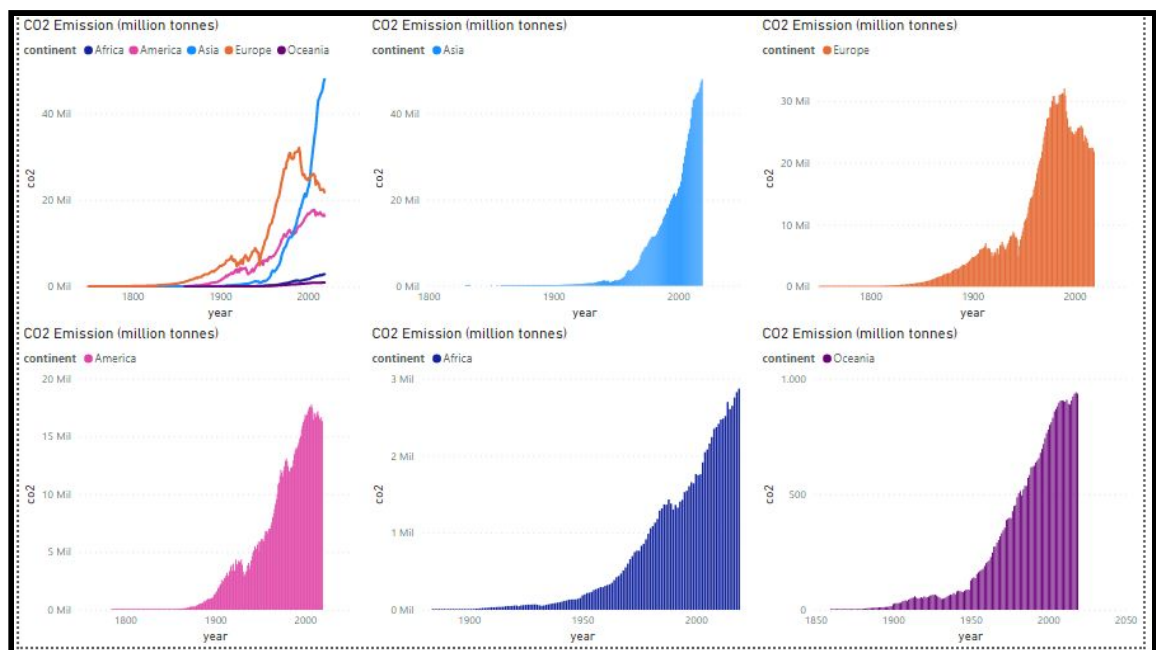
É importante ter em mente que todos os valores na tabela Emission\_Country\_Year estão em milhões de Toneladas, ou seja, quando nos gráficos aparecer Mi o resultado real seria Bilhões. Segue abaixo todos os gráficos contendo os dados importantes conforme contexto do trabalho:



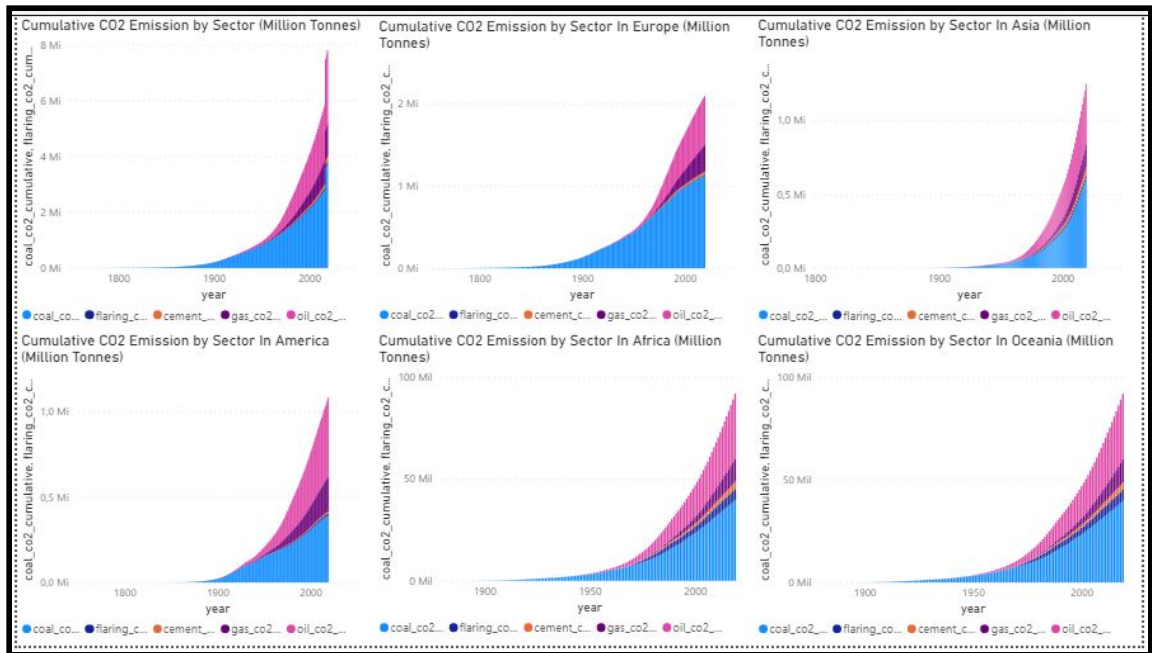
### 5.2.1. Emissão de CO<sub>2</sub>, Metano e Óxido Nitroso por ano e continente:



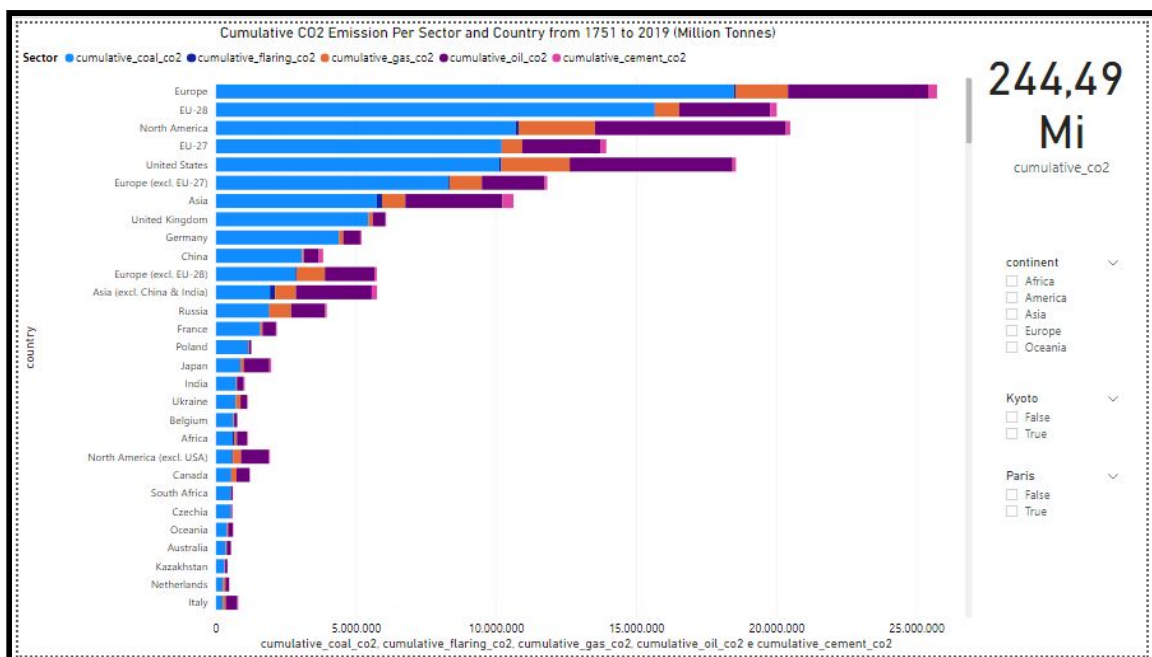
### 5.2.2. Emissão de CO<sub>2</sub> por ano e continente:



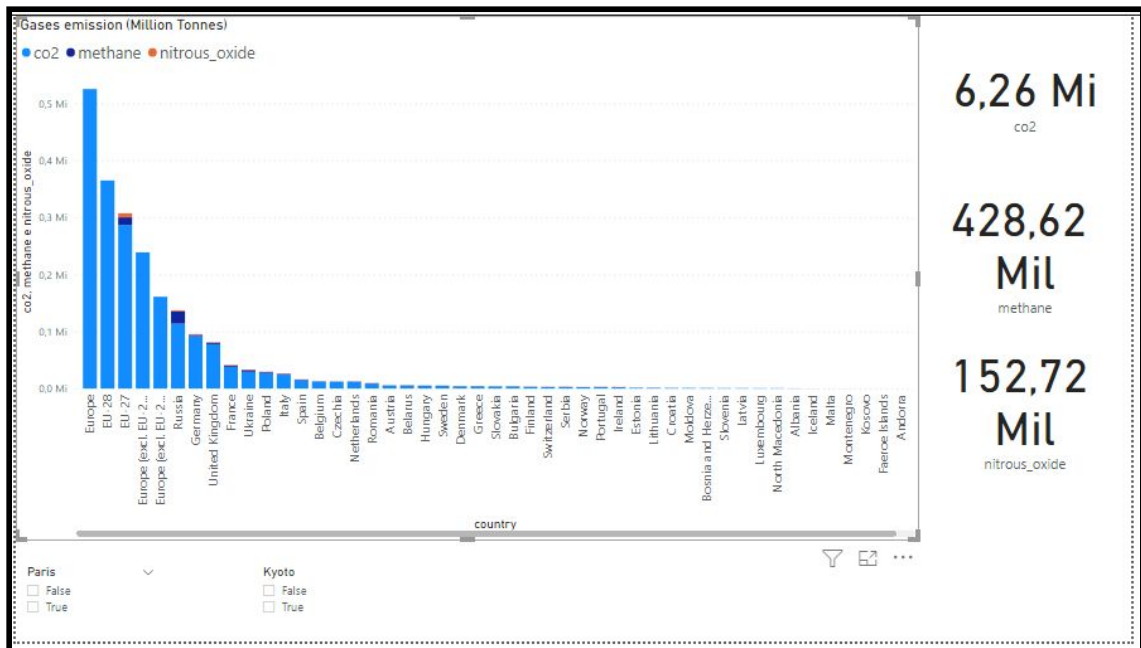
### 5.2.3. Emissão acumulada de CO2 por ano e Continente, para os setores conforme abaixo:



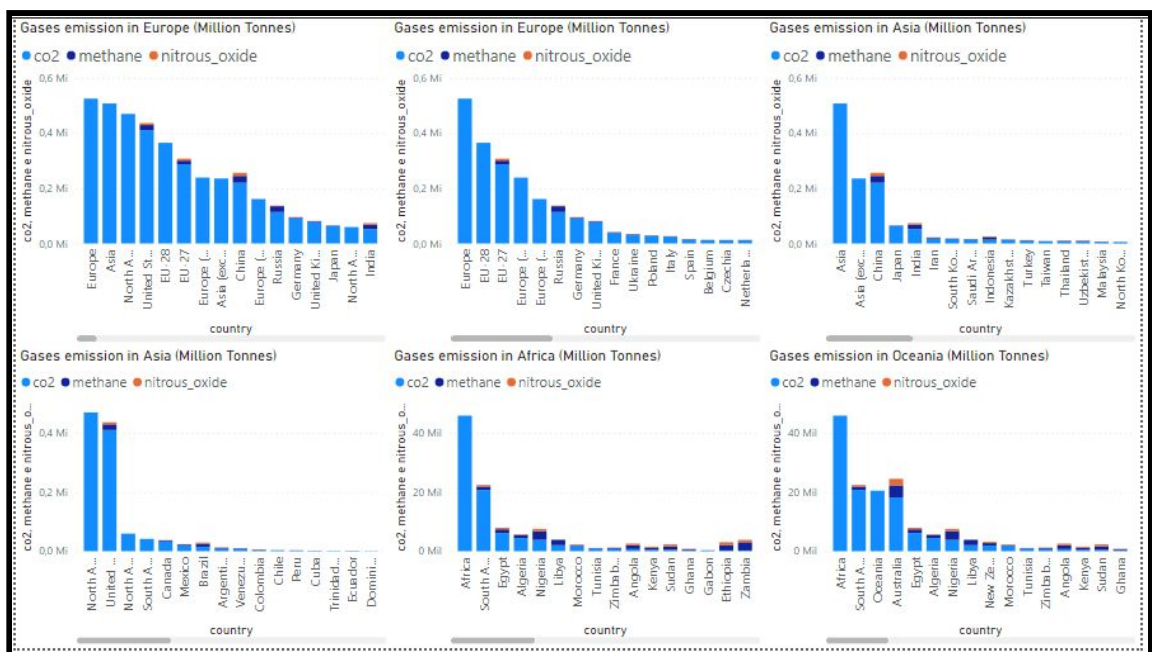
### 5.2.4. Emissão acumulada de CO2 por Setor e País/Região, Análise dos Protocolos de Kyoto e Acordo de Paris:



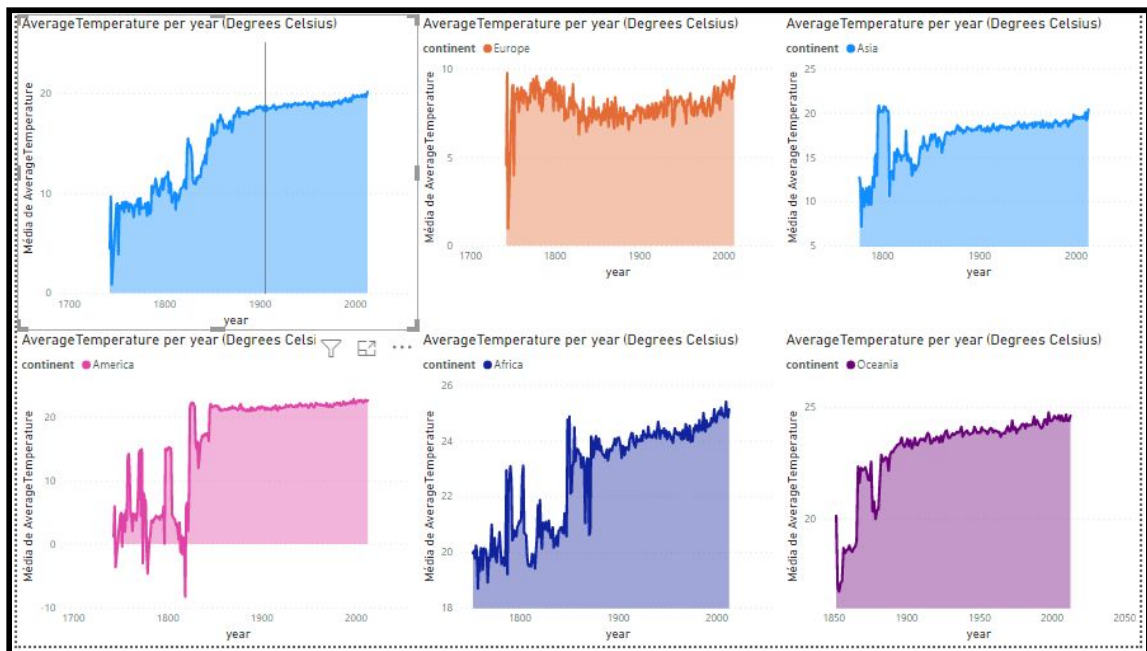
### 5.2.5. Emissão de CO<sub>2</sub>, Metano e Óxido Nitroso por ano e País/Região, Análise dos Protocolos de Kyoto e Acordo de Paris:



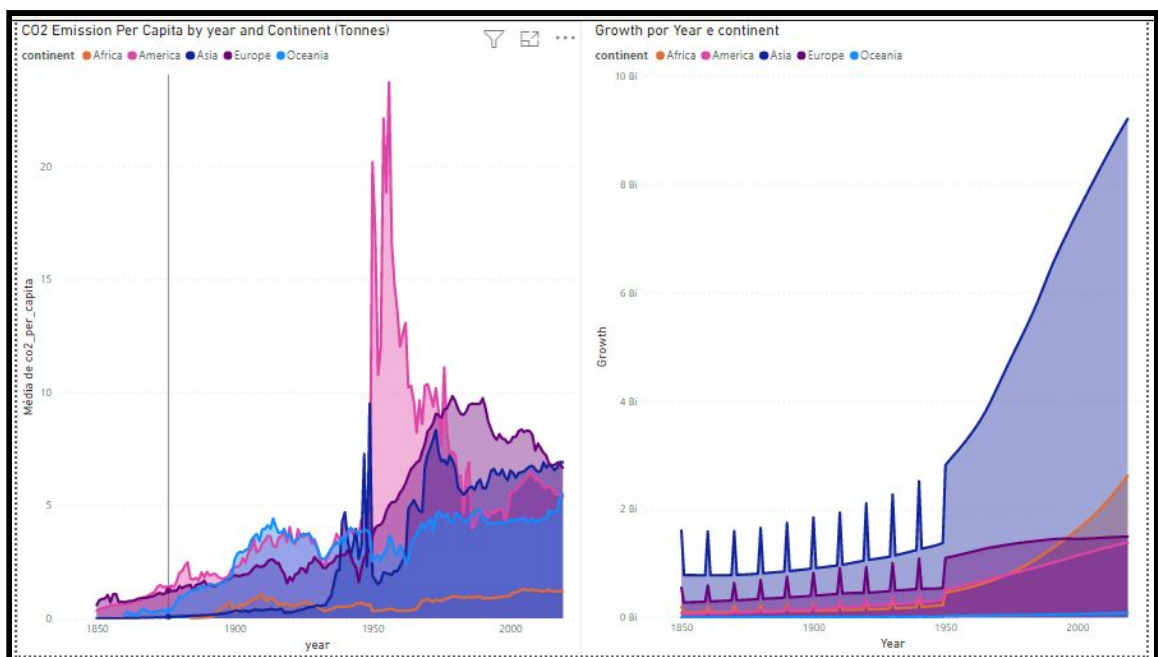
### 5.2.6. Emissão de CO<sub>2</sub>, Metano e Óxido Nitroso por País/Região:



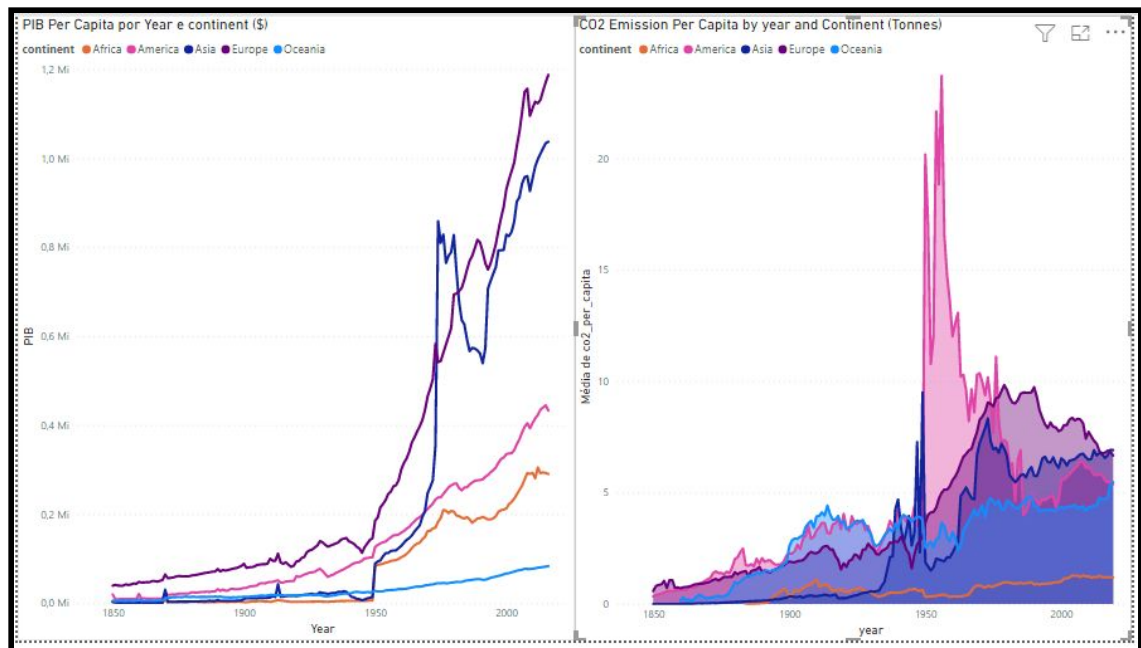
### 5.2.7. Variação de temperatura da terra.



### 5.2.8. Emissão de CO2 per capita e comparação com crescimento populacional por Continente:



### 5.2.9. Emissão de CO2 per capita e comparação com o PIB per capita:



## 6. Considerações finais

### 6.1. Lições Aprendidas

O projeto proposto inicialmente integrava na arquitetura o uso do Google Cloud Storage para o armazenamento dos arquivos tratados no Jupyter, porém foi concluído que não há a necessidade deste passo, uma vez que através da biblioteca `pandas_gbq` e o pacote `google.oauth2` é possível armazenar os data frames diretamente no Bigquery, sem a necessidade de códigos no bigquery. O projeto integra ainda funções de automatizações de notebooks que não haviam sido previstas anteriormente, sendo assim remove a necessidade de rodar os scripts manualmente e passa a ser executado conforme agendamento.

## 7. Bibliografia

<https://cloud.google.com/compute/docs/instances/transfer-files?hl=pt-br>  
<https://thihenos.medium.com/compute-engine-upload-storage-9a5c0a8c5d2>  
<https://googleapis.dev/python/storage/latest/client.html>  
[https://data.giss.nasa.gov/gistemp/graphs/graph\\_data/Global\\_Mean\\_Estimates\\_based\\_o\\_n\\_Land\\_and\\_Ocean\\_Data/graph.txt](https://data.giss.nasa.gov/gistemp/graphs/graph_data/Global_Mean_Estimates_based_o_n_Land_and_Ocean_Data/graph.txt)  
<https://stackoverflow.com/questions/21546739/load-data-from-txt-with-pandas>



[https://cloud.google.com/ai-platform/notebooks/docs/create-new?hl=pt#:~:text=Acesse%20a%20p%C3%A1gina%20AI%20Platform%20Notebooks%20no%20Console%20do%20Google%20Cloud.,-Acessar%20a%20p%C3%A1gina&text=Clique%20em%20add\\_boxNova%20inst%C3%A2ncia%20e%2C%20em%20seguida%2C%20selecione%20Personalizar,nome%20para%20a%20nova%20inst%C3%A2ncia](https://cloud.google.com/ai-platform/notebooks/docs/create-new?hl=pt#:~:text=Acesse%20a%20p%C3%A1gina%20AI%20Platform%20Notebooks%20no%20Console%20do%20Google%20Cloud.,-Acessar%20a%20p%C3%A1gina&text=Clique%20em%20add_boxNova%20inst%C3%A2ncia%20e%2C%20em%20seguida%2C%20selecione%20Personalizar,nome%20para%20a%20nova%20inst%C3%A2ncia).  
<https://3a9700bde2b600a2-dot-us-west1.notebooks.googleusercontent.com/lab?authuser=0>  
[https://googleapis.dev/python/storage/latest/blobs.html#google.cloud.storage.blob.Blob.download\\_as\\_bytes](https://googleapis.dev/python/storage/latest/blobs.html#google.cloud.storage.blob.Blob.download_as_bytes)  
[https://colab.research.google.com/github/corrieann/kaggle/blob/master/kaggle\\_api\\_in\\_colab.ipynb#scrollTo=ILRgc90dojon](https://colab.research.google.com/github/corrieann/kaggle/blob/master/kaggle_api_in_colab.ipynb#scrollTo=ILRgc90dojon)  
<https://stackoverflow.com/questions/56721927/how-to-load-data-to-jupyter-notebook-vm-from-google-cloud>  
[https://colab.research.google.com/github/corrieann/kaggle/blob/master/kaggle\\_api\\_in\\_colab.ipynb](https://colab.research.google.com/github/corrieann/kaggle/blob/master/kaggle_api_in_colab.ipynb)  
<https://medium.com/@charles2588/how-to-upload-download-files-to-from-notebook-in-my-local-machine-6a4e65a15767>  
<https://towardsdatascience.com/running-jupyter-notebook-in-google-cloud-platform-in-15-min-61e16da34d52>  
<https://stackoverflow.com/questions/61888615/how-to-connect-to-google-cloud-platform-data-storage-from-google-cloud-ai-jupyter>  
<https://medium.com/data-hackers/como-fazer-web-scraping-em-python-23c9d465a37f>  
<https://anderfernandez.com/en/blog/automate-python-script-google-cloud/>  
[https://cloud.google.com/scheduler/docs/configuring/cron-job-schedules?&\\_ga=2.7888951.-428690011.1612479549&\\_gac=1.124687480.1614726470.Cj0KCQiA4feBBhC9ARIsABp\\_nbUe8WMx3ulauMmLchhMFSY6MFgLqn8Y1N1m6Kdw\\_SPL37ByhrzNeZoaAq5LEALw\\_wcB#defining\\_the\\_job\\_schedule](https://cloud.google.com/scheduler/docs/configuring/cron-job-schedules?&_ga=2.7888951.-428690011.1612479549&_gac=1.124687480.1614726470.Cj0KCQiA4feBBhC9ARIsABp_nbUe8WMx3ulauMmLchhMFSY6MFgLqn8Y1N1m6Kdw_SPL37ByhrzNeZoaAq5LEALw_wcB#defining_the_job_schedule)