Projeto Combustíveis

17/01/2023

SoulCode Academy - Engenharia de Dados - Turma BC 26

Integrantes

Augusto Tonelli

- Linkedin - GitHub

Caio Alves

- Linkedin - GitHub

Érica Marçal

- Linkedin - GitHub

Luan Sagara

- Linkedin - GitHub

Nayara Bernardo

- Linkedin - GitHub

DataRoot

Objetivo

Esse é o Projeto Final de Conclusão do Bootcamp de Engenharia de Dados promovido pela SoulCode Academy. O tema dado à equipe foi o de Combustíveis e os Datasets obtidos foram extraídos desse link.

O objetivo deste projeto é obter insights sobre o mercado de combustíveis no Brasil, analisando os valores das operações comerciais de importação e exportação, os valores de importação do etanol no período da safra, a receita gerada por tipos de produtos de importação e exportação, a produção por tempo, a produção de combustíveis no Brasil, o volume de produção por estado e o volume de produção por refinaria, além da produção, uso interno e venda externa.

Foram realizadas análise de dados públicos do Brasil sobre petróleo e derivados, biocombustíveis e gás natural no período de 2012 a 2021, em relação a variáveis de impacto:

- nos preços de revenda e margem de ganho;
- no volume de produção;
- nas importações e exportações.

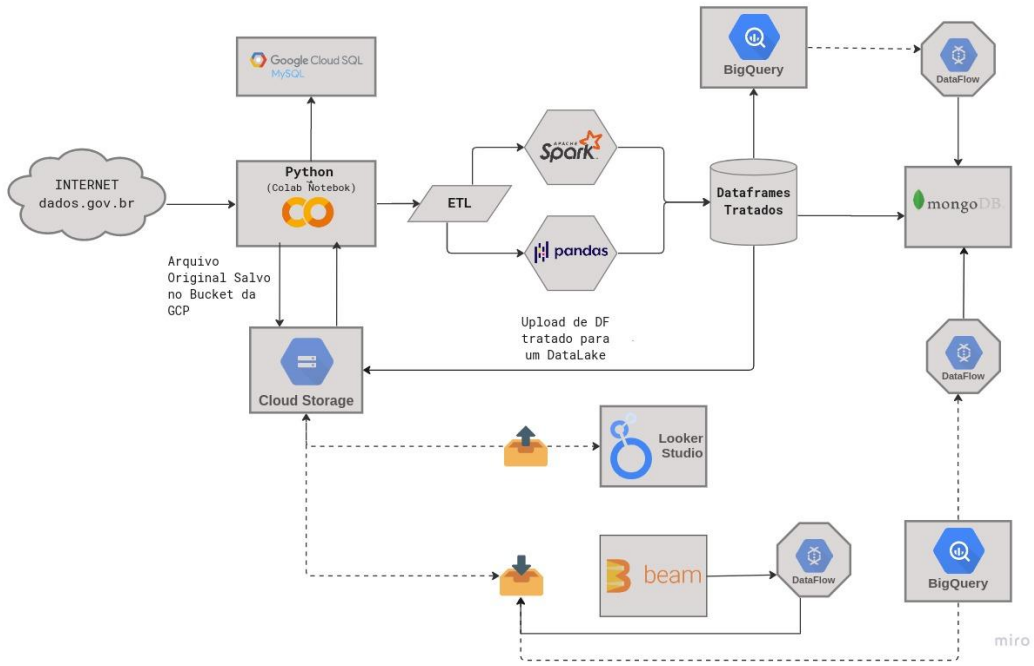
Insights

1. A entressafra da cana-de-açúcar alterou a importação e produção do Etanol;
2. A PPI em 2016 afetou o montante de importação de combustíveis;
3. Rio de Janeiro aparece em destaque em produção de Petróleo e Gás;
4. A pandemia não alterou a produção de petróleo e Gás Natural; entretanto, afetou a de derivados de petróleo

Requisitos atendidos neste projeto

- Foram utilizados 11 Datasets em formato csv e xlsx de fontes diferentes;
- Procedimento de ETL com análises através de Pandas, PySpark;
- Durante o processo de Transformação da nossa ETL, utilizamos do recurso de plotagem do Pandas para análise de alguns insights;
- Inserção, transformação e normalização dos dados por meio de uma Pipeline com modelo criado em Apache Beam usando o Dataflow para work e carregamento para um Data Lake;
- Armazenamento dos dados brutos em Cloud SQL (MySQL) e dos dados tratados no MongoDB, Cloud Storage (Data Lake) e no BigQuery (Data Warehouse);
- Utilização do modelo predefinido 'BigQuery to MongoDB' para enviar os DataSets da BigQuery para o MongoDB;
- Análises realizadas através do BigQuery em linguagem padrão SQL;
- Criação de Dashboard com dados tratados no Google Looker Studio;
- Levantar custos com a utilização do Google Cloud no período do projeto e possíveis otimizações de custo.
- Segue links do Colab Notebook - Principal, Colab Notebook - Pipeline e Dashboard

Workflow



Tecnologias utilizadas para realização do projeto:

- Google Cloud Platform (CGP)
- Cloud Storage
- Looker Studio
- BigQuery
- DataFlow
- Python
- Pandas
- PySpark
- Apache Beam
- MongoDB

Procedimentos

1. Extração

1.1. Instalação e importação de bibliotecas

1.1.1. Instalação das bibliotecas

```
[ ]  1  !pip install gcsfs
     2  !pip install pyspark
     3  !pip install pymongo
     4  !pip install pymysql
     5  !pip install mysql-connector-python
     6  pip install apache_beam[interactive]
     7  pip install apache_beam[gcp]
```

1.1.2. Importação das bibliotecas

```
[ ]  1  from google.cloud import storage
     2  import os
     3  import mysql.connector
     4  from sqlalchemy import create_engine
     5  import pandas as pd
     6  import matplotlib
     7  import glob
     8  import requests
     9  from pyspark.sql import SparkSession
    10  import pyspark.sql.functions as F
    11  from pyspark.sql.types import *
    12  from pymongo import MongoClient
    13  from apache_beam.io.textio import WriteToText
    14  import apache_beam as beam
```

1.1.3. Definição do máximo de colunas a ser mostrada

```
[ ]  1  pd.set_option('display.max_columns',25)
```

1.2. Envio dos Datasets brutos ao MySQL

Criação de um dicionário contendo os nomes e links dos Datasets

```
[ ]  1  dataframes = {'dfbio_bruto':'https://www.gov.br/anp/pt-br/centrais-de-conteudo/dados-abertos/arquivos/arquivos-producao-de-biocombustiveis/producao-biodiesel-m3-2005-2021.csv',
     2  'dfetanol_bruto':'https://www.gov.br/anp/pt-br/centrais-de-conteudo/dados-abertos/arquivos/arquivos-producao-de-biocombustiveis/producao-etanol-anidro-hidratado-m3-2012-2022.csv',
     3  'dfpetroleo_bruto':'https://www.gov.br/anp/pt-br/centrais-de-conteudo/dados-abertos/arquivos/ppgn-el/producao-petroleo-m3-1997-2022.csv',
     4  'dfgas_bruto':'https://www.gov.br/anp/pt-br/centrais-de-conteudo/dados-abertos/arquivos/ppgn-el/producao-gas-natural-1000m3-1997-20[illegible].csv',
     5  'dfimpexppet_bruto':'https://www.gov.br/anp/pt-br/centrais-de-conteudo/dados-abertos/arquivos/ie/petroleo/importacoes-exportacoes-petroleo-2000-2022.csv',
     6  'dfimpexpgas_bruto':'https://www.gov.br/anp/pt-br/centrais-de-conteudo/dados-abertos/arquivos/ie/gn/importacao-gas-natural-2000-202.csv',
     7  'dfimpexpder_bruto':'https://www.gov.br/anp/pt-br/centrais-de-conteudo/dados-abertos/arquivos/ie/derivados/importacoes-exportacoes-derivados-2000-2022.csv',
     8  'dfimpexpeta_bruto':'https://www.gov.br/anp/pt-br/centrais-de-conteudo/dados-abertos/arquivos/ie/etanol/importacoes-exportacoes-etanol-2012-2022.csv',
     9  'dfprodder_bruto':'https://www.gov.br/anp/pt-br/centrais-de-conteudo/dados-abertos/arquivos/pppd/producao-derivados-petroleo-por-refinaria-m3-1990-2021.csv'}
```

```
[ ]  1  servidor = '34.69.19.129'
     2  nome_do_banco = 'projeto-final'
     3  usuario = 'root'
     4  senha = 'root'
     5
     6  # Criando uma SQLAlchemy engine para conectar com o MySQL
     7  engine = create_engine("mysql+pymysql://{user}:{pw}@{host}/{db}".format(host=servidor, db=nome_do_banco, user=usuario, pw=senha))
```

```
[ ]  1  for k,v in dataframes.items():
     2    df_generico = pd.read_csv(v,sep=';')
     3    df_generico.to_sql(k, engine, index=True, index_label="id", if_exists='replace')
```

```
[ ]  1  dfpreco1_bruto = pd.read_excel('https://www.gov.br/anp/pt-br/assuntos/precos-e-defesa-da-concorrencia/precos/precos-revenda-e-de-dist
     2  dfpreco2_bruto = pd.read_excel('https://www.gov.br/anp/pt-br/assuntos/precos-e-defesa-da-concorrencia/precos/precos-revenda-e-de-dist
     3  dfpreco1_bruto.to_sql('dfpreco0112_bruto', engine, index=True, index_label="id", if_exists='replace')
     4  dfpreco2_bruto.to_sql('dfpreco13_bruto', engine, index=True, index_label="id", if_exists='replace')
```

2. Transformação

2.1. Dataset de Produção de Biocombustíveis

2.1.1. Leitura dos Datasets

```
[ ]  1  dfbdi = pd.read_csv('https://www.gov.br/anp/pt-br/centrais-de-conteudo/dados-abertos/arquivos/arquivos-producao-de-biocombustiveis/producao-biodiesel-m3-2005-2021.csv', sep=';')
     2  dfeta = pd.read_csv('https://www.gov.br/anp/pt-br/centrais-de-conteudo/dados-abertos/arquivos/arquivos-producao-de-biocombustiveis/producao-etanol-anidro-hidratado-m3-2012-2022.csv', sep=';')
     3
```

2.1.2. Pré-análise dos Dataframes

```
[ ]  1  dfbdi.head(2)
```

2.1.3. Tratamentos

Drop da coluna de produtor para igualar os Dataframes. Não fará falta nas análises

```
[ ]  1  dfbdi.drop('PRODUTOR',axis=1,inplace=True)
```

União dos dois dataframes a fim de facilitar o tratamento

```
[ ]  1  dfbio = pd.concat([dfbdi,dfeta])
```

Check de valores nulos

```
[ ]  1  dfbio.isna().sum()
```

Renomeação de colunas

```
[ ]  1  dfbio.rename(columns={'ANO':'ano','MÊS':'mes','GRANDE REGIÃO':'regiao','UNIDADE DA FEDERAÇÃO':'uf','PRODUTO':'produto','PRODUÇÃO':'volume_m3'},inplace=True)
```

Retirada de acentuação da coluna estado

```
[ ]  1  dfbio['uf'] = dfbio.uf.str.normalize('NFKD').str.encode('ascii', errors='ignore').str.decode('utf-8')
```

Restrição do dataframe para o intervalo desejado

```
[ ]  1  dfbio = dfbio.loc[(dfbio['ano'] >= 2012) & (dfbio['ano'] <= 2021)]
```

Tratamento das inconsistências detectadas

```
[ ]  1  dfbio.volume_m3.replace(',','.',regex=True, inplace=True)
     2  dfbio.uf.replace({'CENTRO OESTE':'CENTRO-OESTE'},inplace=True)
     3  dfbio.uf.replace({'BRASILIA':'DISTRITO FEDERAL'},inplace=True)
```

Ajuste da coluna de data unindo as colunas de mês e ano

```
[ ]  1  dfbio.mes.replace(({'JAN':'01','FEV':'02','MAR':'03','ABR':'04','MAI':'05','JUN':'06','JUL':'07','AGO':'08',
     2                      'SET':'09','OUT':'10','NOV':'11','DEZ':'12'}),regex=True,inplace=True)
```

```
     1  # Troca do tipo para string, então união das colunas de data e finalmente transformação para o tipo datetime
     2  dfbio['ano'] = dfbio['ano'].astype(str)
     3  dfbio['data'] = dfbio['ano'] + '-' + dfbio['mes']
     4  dfbio['data'] = pd.to_datetime(dfbio['data'], format = '%Y-%m')
```

```
[ ]  1  # Drop das colunas ano e mes, pois não são mais necessárias
     2  dfbio.drop(['ano','mes'],axis=1,inplace=True)
```

Troca de tipo da coluna volume_m3 para float

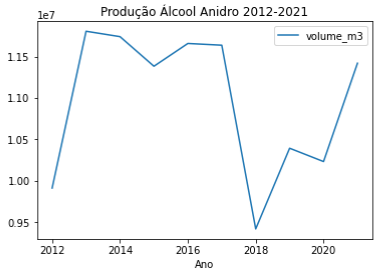
```
[ ]  1  dfbio['volume_m3'] = dfbio['volume_m3'].astype(float)
```

2.1.4. Plots em Pandas

Produção de álcool anidro durante o período 2012-2021

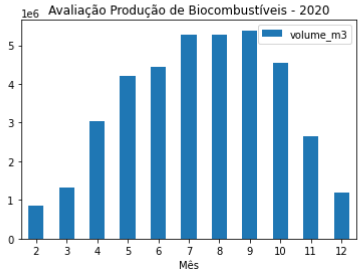
```
[ ]  1  ft1 = dfbio.loc[dfbio['produto'] == 'ANIDRO']
     2  ft1.groupby(dfbio['data'].dt.year).sum().plot.line(title ='Produção Álcool Anidro 2012-2021', xlabel = 'Ano')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f83c7989e50>
```




```
[ ]  1  ft = dfbio.loc[(dfbio['data'] > '2020-01-01') & (dfbio['data'] < '2020-12-31') ]
     2  ft.groupby(dfbio['data'].dt.month).sum().plot.bar(title ='Avaliação Produção de Biocombustíveis - 2020', xlabel='Mês',rot=0)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f83c75f7460>
```



2.2. Dataset de Produção de Petróleo e Gás Natural

2.2.1. Leitura dos Datasets

```
[ ]  1  dfpet = pd.read_csv('https://www.gov.br/anp/pt-br/centrais-de-conteudo/dados-abertos/arquivos/ppgn-el/producao-petroleo-m3-1997-2022.csv',sep=';')
     2  dfgas = pd.read_csv('https://www.gov.br/anp/pt-br/centrais-de-conteudo/dados-abertos/arquivos/ppgn-el/producao-gas-natural-1000m3-1997-2022.csv',sep=';')
```

2.2.2. Pré-análise dos Dataframes

```
[ ]  1  dfgas.head(2)
```

2.2.3. Tratamentos

Concatenação dos DFs

```
[ ]  1  dfgaspet = pd.concat([dfpet, dfgas])
```

Renomeação das colunas

```
[ ]  1  dfgaspet.rename(columns={'ANO':'ano','MÊS':'mes','GRANDE REGIÃO':'regiao','UNIDADE DA FEDERAÇÃO':'uf',
     2                           'PRODUTO':'produto','LOCALIZAÇÃO':'localizacao','PRODUÇÃO':'producao'},inplace=True)
```

Verificação dados de todas as colunas

```
[ ]  1  pd.unique(dfgaspet['producao'])
```

```
array(['2767443', '2891003', '2998305', ..., '2143', '2296,21325',
       '2124,99039'], dtype=object)
```

Restrição do dataframe para o intervalo desejado

```
[ ]  1  fano = (dfgaspet.ano > 2011) & (dfgaspet.ano < 2022)
     2  dfgaspet = dfgaspet.loc[fano]
```

Troca do tipo da coluna producao

```
[ ]  1  dfgaspet.producao.replace(',','.',regex=True,inplace=True)
```

```
[ ]  1  dfgaspet['producao'] = dfgaspet['producao'].astype(float)
```

Ajuste da coluna de data unindo as colunas de mês e ano

```
[ ]  1  dfgaspet.mes.replace(({'JAN':'01','FEV':'02','MAR':'03','ABR':'04','MAI':'05','JUN':'06',
     2                         'JUL':'07','AGO':'08','SET':'09','OUT':'10','NOV':'11','DEZ':'12'}),inplace=True)
```

```
[ ]  1  dfgaspet['ano'] = dfgaspet['ano'].astype(str)
     2  dfgaspet['data'] = dfgaspet['ano'] + '-' + dfgaspet['mes']
     3  dfgaspet['data'] = pd.to_datetime(dfgaspet['data'], format = '%Y-%m')
```

```
[ ]  1  dfgaspet.drop(['ano','mes'],axis=1,inplace=True)
```

Group By para verificar produtos

```
[ ]  1
     2  dfgaspet.groupby(['produto']).size().sort_values(ascending=False)
```

```
produto
PETRÓLEO       2640
GÁS NATURAL    2635
dtype: int64
```

2.3. Dataset de Produção de derivados do Petróleo

2.3.1. Leitura dos Datasets

```
[ ]  1  dfder = pd.read_csv('https://www.gov.br/anp/pt-br/centrais-de-conteudo/dados-abertos/arquivos/pppd/producao-derivados-petroleo-por-refinaria-m3-1990-2021.csv' ,sep=';')
```

2.3.2. Pré-análise dos Dataframes

```
[ ]  1  dfder.head(2)
```

```
[ ]  1  dfder.isna().sum()
```

2.3.3. Tratamentos

Renomeação das colunas

```
[ ]  1  dfder.rename(columns={'ANO':'ano','MÊS':'mes','UNIDADE DA FEDERAÇÃO':'uf','REFINARIA':'refinaria','PRODUTO':'produto','PRODUÇÃO':'produto','PRODUÇÃO':'producao'},inplace=True)
```

Procura de inconsistências com unique em cada coluna

```
[ ]  1  pd.unique(dfder['produto'])
```

```
array(['GASOLINA DE AVIAÇÃO', 'ÓLEO COMBUSTÍVEL', 'NAFTA',
       'OUTROS NÃO ENERGÉTICOS', 'ASFALTO', 'COQUE', 'OUTROS ENERGÉTICOS',
       'QUEROSENE ILUMINANTE', 'LUBRIFICANTE', 'QUEROSENE DE AVIAÇÃO',
       'ÓLEO DIESEL', 'SOLVENTE', 'PARAFINA', 'GLP', 'GASOLINA A'],
      dtype=object)
```

Filtragem de linhas com somente os produtos que interessam ao escopo

```
[ ]  1  dfder = dfder.loc[(dfder['produto'] == 'OUTROS ENERGÉTICOS') | (dfder['produto'] == 'COQUE') | (dfder['produto'] == 'GASOLINA A') |
     2                    (dfder['produto'] == 'ÓLEO COMBUSTÍVEL') | (dfder['produto'] == 'GASOLINA DE AVIAÇÃO') |
     3                    (dfder['produto'] == 'ÓLEO DIESEL') | (dfder['produto'] == 'GLP') | (dfder['produto'] == 'QUEROSENE DE AVIAÇÃO')]
```

Restrição do dataframe para o intervalo desejado

```
[ ]  1  # Filtro ano 2012 a 2021
     2  fano = (dfder.ano > 2011) & (dfder.ano < 2022)
     3  dfder = dfder.loc[fano]
```

Replace de vírgula por ponto para conseguir transformar a coluna para float

```
[ ]  1  dfder.replace([','],'.',regex=True,inplace=True)
```

```
[ ]  1  dfder['producao'] = dfder['producao'].astype(float)
```

Ajuste da coluna de data unindo as colunas de mês e ano

```
[ ]  1  # Mudança de letras para números do mês
     2  dfder.mes.replace(({'JAN':'01','FEV':'02','MAR':'03','ABR':'04','MAI':'05','JUN':'06',
     3                     'JUL':'07','AGO':'08','SET':'09','OUT':'10','NOV':'11','DEZ':'12'}),regex=True,inplace=True)
```

```
[ ]  1  # Transformação coluna ano para string
     2  dfder['ano'] = dfder['ano'].astype(str)
```

```
[ ]  1  # Junção colunas 'ano' e 'mes'
     2  dfder['data'] = dfder['ano'] + '-' + dfder['mes']
```

```
[ ]  1  # Transformação da coluna 'data' para DateTime
     2  dfder['data'] = pd.to_datetime(dfder['data'], format = '%Y-%m')
```

Organizando e dropando colunas de mes e ano, pois já existe a de data

```
[ ]  1  dfder = dfder[['data','uf','refinaria','produto','producao']]
```

2.4. Dataset de Importação e Exportação de Combustíveis

2.4.1. Leitura dos Datasets

```
[ ]
def baixar_arquivo(url, endereco=None):
    if endereco is None:
        endereco = os.path.basename(url.split("?")[0])
    resposta = requests.get(url, stream=True)
    if resposta.status_code == requests.codes.OK:
        with open(endereco, 'wb') as novo_arquivo:
            for parte in resposta.iter_content(chunk_size=256):
                novo_arquivo.write(parte)
        print("Download finalizado. Arquivo salvo em: {}".format(endereco))
    else:
        resposta.raise_for_status()
```

```
def selecionar_arquivos_para_baixar():
    import_export_01 = "https://www.gov.br/anp/pt-br/centrais-de-conteudo/dados-abertos/arquivos/ie/petroleo/importacoes-exportacoes-
    import_export_02 = "https://www.gov.br/anp/pt-br/centrais-de-conteudo/dados-abertos/arquivos/ie/gn/importacao-gas-natural-2000-20
    import_export_03 = "https://www.gov.br/anp/pt-br/centrais-de-conteudo/dados-abertos/arquivos/ie/derivados/importacoes-exportacoes
    import_export_04 = "https://www.gov.br/anp/pt-br/centrais-de-conteudo/dados-abertos/arquivos/ie/etanol/importacoes-exportacoes-et

    baixar_arquivo(import_export_01, "/content/import_export_01.csv")
    baixar_arquivo(import_export_02, "/content/import_export_02.csv")
    baixar_arquivo(import_export_03, "/content/import_export_03.csv")
    baixar_arquivo(import_export_04, "/content/import_export_04.csv")
```

```
[ ]
selecionar_arquivos_para_baixar()
```

```
Download finalizado. Arquivo salvo em: /content/import_export_01.csv
Download finalizado. Arquivo salvo em: /content/import_export_02.csv
Download finalizado. Arquivo salvo em: /content/import_export_03.csv
Download finalizado. Arquivo salvo em: /content/import_export_04.csv
```

Carregamento dos arquivos baixados e concantenados em um único Dataframe

```
[ ]
def carregar_dados_geral():
    caminho = r'/content'
    todos_os_arquivos = glob.glob(caminho + "/*.csv")
    li = []

    for nome_arquivo in todos_os_arquivos:
        df = pd.read_csv(nome_arquivo, encoding="UTF-8", sep=";")
        df.rename(columns={'IMPORTADO':'IMPORTADO / EXPORTADO', 'DISPÊNDIO':'DISPÊNDIO / RECEITA'}, inplace=True)
        li.append(df)

    dfimex = pd.concat(li)
    return dfimex
```

```
[ ]
dfimex = carregar_dados_geral()
```

2.4.2. Pré-análise do Dataframe

```
[ ]
dfimex.shape
```

```
(9584, 6)
```

```
[ ]
dfimex.isna().sum()
```

```
ANO                       0
MÊS                       0
PRODUTO                   0
OPERAÇÃO COMERCIAL        0
IMPORTADO / EXPORTADO    11
DISPÊNDIO / RECEITA       0
dtype: int64
```

Checagem de linhas com valores nulos

```
[ ]  1  dfimex[dfimex['IMPORTADO / EXPORTADO'].isna()]
```

```
[ ]  1  dfimex.groupby('PRODUTO').size()
```

2.4.3. Tratamentos

Filtragem de linhas com somente os produtos que interessam ao escopo

```
[ ]  1  dfimex = dfimex.loc[(dfimex['PRODUTO'] == 'COMBUSTÍVEIS PARA AERONAVES') |
     2                      (dfimex['PRODUTO'] == 'COMBUSTÍVEIS PARA NAVIOS') |
     3                      (dfimex['PRODUTO'] == 'COQUE') | (dfimex['PRODUTO'] == 'ETANOL ANIDRO') |
     4                      (dfimex['PRODUTO'] == 'ETANOL HIDRATADO') | (dfimex['PRODUTO'] == 'GASOLINA A') |
     5                      (dfimex['PRODUTO'] == 'GÁS NATURAL') | (dfimex['PRODUTO'] == 'PETRÓLEO') |
     6                      (dfimex['PRODUTO'] == 'QUEROSENE DE AVIAÇÃO') | (dfimex['PRODUTO'] == 'ÓLEO COMBUSTÍVEL') |
     7                      (dfimex['PRODUTO'] == 'ÓLEO DIESEL') | (dfimex['PRODUTO'] == 'GLP') |
     8                      (dfimex['PRODUTO'] == 'GASOLINA DE AVIAÇÃO')]
```

Transformação da coluna data unindo mes e ano

```
[ ]  1  dfimex['ANO'] = dfimex['ANO'].astype(str)
```

```
     1  dfimex.MÊS.replace(({'JAN':'01','FEV':'02','MAR':'03','ABR':'04','MAI':'05','JUN':'06',
     2                       'JUL':'07','AGO':'08','SET':'09','OUT':'10','NOV':'11','DEZ':'12'}),regex=True,inplace=True)
```

```
[ ]  1  # Junção colunas 'ano' e 'mes'
     2  dfimex['DATA'] = dfimex['ANO'] + '-' + dfimex['MÊS']
```

```
[ ]  1  # Transformação da coluna 'data' para DateTime
     2  dfimex['DATA'] = pd.to_datetime(dfimex['DATA'], format = '%Y-%m')
```

Renomeação das colunas

```
[ ]  1  dfimex.rename(columns={'IMPORTADO / EXPORTADO':'importado_exportado',
     2                         'DISPÊNDIO / RECEITA':'dispendio_receita',
     3                         'OPERAÇÃO COMERCIAL':'operacao_comercial',
     4                         'DATA':'data',
     5                         'PRODUTO':'produto'}, inplace=True)
```

Replace vírgula por ponto para conseguir transformar a coluna para float

```
[ ]  1  dfimex.replace([','],'.',regex=True,inplace=True)
```

```
[ ]  1  dfimex['importado_exportado'] = dfimex['importado_exportado'].astype(float)
```

```
[ ]  1  dfimex.drop(columns=['ANO', 'MÊS'], inplace=True)
```

Filtragem de linhas com somente os anos que interessam ao escopo

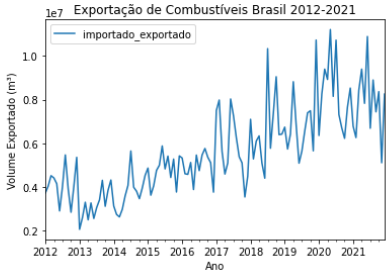
```
[ ]  1  dfimex = dfimex[(dfimex['data'].dt.year >= 2012) & (dfimex['data'].dt.year <= 2021)]
```

```
[ ]  1  #Reorganização das colunas
     2  dfimex = dfimex[['data','produto','operacao_comercial','importado_exportado','dispendio_receita']]
```

2.4.4. Plots em Pandas

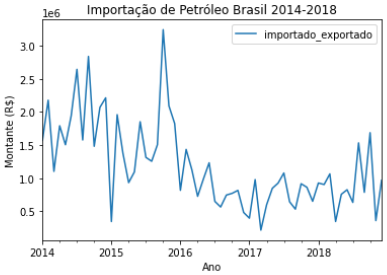
```
[ ]  1  ft = dfimex.loc[dfimex.operacao_comercial == 'EXPORTAÇÃO']
     2  ft.groupby('data').sum().plot.line(title='Exportação de Combustíveis Brasil 2012-2021',xlabel='Ano',ylabel='Volume Exportado (m³)')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f83c6a32e50>
```




```
[ ]  1  ft = dfimex.loc[(dfimex.operacao_comercial == 'IMPORTAÇÃO') & (dfimex.produto == 'PETRÓLEO') &
     2                  (dfimex.data.dt.year >= 2014) & (dfimex.data.dt.year <= 2018)]
     3  ft.groupby('data').sum().plot.line(title='Importação de Petróleo Brasil 2014-2018',
     4                                     xlabel='Ano',ylabel= 'Montante (R$)')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f83c8b9f490>
```



2.5. Dataset de Preços de Combustíveis

2.5.1. Leitura dos Datasets

```
[ ]  1  # Cria a sessão spark
     2  spark = (SparkSession.builder.
     3           master('local').
     4           appName('structtype').
     5           getOrCreate())
```

```
[ ]  1  spark
```

SparkSession - in-memory

SparkContext

Spark UI

Version
v3.3.1
Master
local
AppName
structtype

Leitura dos datasets com a aplicação do schema acima

```
[ ]  1  dfprecopand = pd.read_excel('https://www.gov.br/anp/pt-br/assuntos/precos-e-defesa-da-concorrencia/precos/precos-revenda-e-de-distrib
     2                              header = 12)
     3  dfpreco1 = spark.createDataFrame(dfprecopand, schema=schemat)
     4
     5  dfprecopand = pd.read_excel('https://www.gov.br/anp/pt-br/assuntos/precos-e-defesa-da-concorrencia/precos/precos-revenda-e-de-distrib
     6                              header = 17)
     7  dfpreco2 = spark.createDataFrame(dfprecopand, schema=schemat)
     8
```

2.5.2. Pré-análise do Dataframe

```
1  dfpreco1.show(2)
```

```
+------------+----------+------------+----------------+---------------+---------------------------+----------------+------------------+
|DATA INICIAL|DATA FINAL|      REGIÃO|          ESTADO|        PRODUTO|NÚMERO DE POSTOS PESQUISADOS|UNIDADE DE MEDIDA|PREÇO MÉDIO REVENDA|
+------------+----------+------------+----------------+---------------+---------------------------+----------------+------------------+
|  2004-05-09|2004-05-15|CENTRO OESTE|DISTRITO FEDERAL|ETANOL HIDRATADO|                        127|            R$/l|             1.288|
|  2004-05-09|2004-05-15|CENTRO OESTE|           GOIAS|ETANOL HIDRATADO|                        387|            R$/l|             1.162|
+------------+----------+------------+----------------+---------------+---------------------------+----------------+------------------+
only showing top 2 rows
```

```
1  dfpreco2.show(2)
```

```
+------------+----------+--------+-------+---------------+---------------------------+----------------+------------------+-------------
|DATA INICIAL|DATA FINAL|  REGIÃO| ESTADO|        PRODUTO|NÚMERO DE POSTOS PESQUISADOS|UNIDADE DE MEDIDA|PREÇO MÉDIO REVENDA|DESVIO PADRÃO
+------------+----------+--------+-------+---------------+---------------------------+----------------+------------------+-------------
|  2012-12-30|2013-01-05|   NORTE|   ACRE|ETANOL HIDRATADO|                         33|            R$/l|             2.553|
|  2012-12-30|2013-01-05|NORDESTE|ALAGOAS|ETANOL HIDRATADO|                         97|            R$/l|             2.308|
+------------+----------+--------+-------+---------------+---------------------------+----------------+------------------+-------------
only showing top 2 rows
```

2.5.3. Tratamentos

Unindo os dois Dataframes, pois têm as mesmas colunas

```
1  dfpreco = dfpreco1.union(dfpreco2)
```

```
1  backup = dfpreco
```

Seleção do período definido para o escopo do projeto

```
1  dfpreco = dfpreco.filter(F.col('DATA INICIAL') >= '2012-01-01')
```

Remoção uma data que não será usada

```
1  dfpreco = dfpreco.drop('DATA FINAL')
```

Renomeção das colunas

```
1  dfpreco = dfpreco.withColumnRenamed('DATA INICIAL','data')\
2  .withColumnRenamed('REGIÃO','regiao')\
3  .withColumnRenamed('ESTADO','estado')\
4  .withColumnRenamed('PRODUTO','produto')\
5  .withColumnRenamed('NÚMERO DE POSTOS PESQUISADOS','postos_pesquisados')\
6  .withColumnRenamed('UNIDADE DE MEDIDA','uni_medida')\
7  .withColumnRenamed('PREÇO MÉDIO REVENDA','media_rev')\
8  .withColumnRenamed('DESVIO PADRÃO REVENDA','desvio_rev')\
9  .withColumnRenamed('PREÇO MÍNIMO REVENDA','preco_min_rev')\
10 .withColumnRenamed('PREÇO MÁXIMO REVENDA','preco_max_rev')\
11 .withColumnRenamed('MARGEM MÉDIA REVENDA','margem_rev')\
12 .withColumnRenamed('COEF DE VARIAÇÃO REVENDA','coef_var_rev')\
13 .withColumnRenamed('PREÇO MÉDIO DISTRIBUIÇÃO','media_dist')\
14 .withColumnRenamed('DESVIO PADRÃO DISTRIBUIÇÃO','dp_dist')\
15 .withColumnRenamed('PREÇO MÍNIMO DISTRIBUIÇÃO','preco_min_dist')\
16 .withColumnRenamed('PREÇO MÁXIMO DISTRIBUIÇÃO','preco_max_dist')\
17 .withColumnRenamed('COEF DE VARIAÇÃO DISTRIBUIÇÃO','coef_var_dist')
```

Conversão de tipos de colunas

```
[ ]
# Inserção do nome de todas as colunas que deveriam ser float mas são string em uma lista
colunas = ['margem_rev', 'media_dist', 'dp_dist',
           'preco_min_dist', 'preco_max_dist', 'coef_var_dist']

# Substituição de todas as strings '-' por 0
for i in colunas:
    dfpreco = dfpreco.withColumn(i, F.regexp_replace(i, '-', '0'))

dfpreco = dfpreco.withColumn("margem_rev",F.col("margem_rev").cast(FloatType()))\
                 .withColumn("media_dist",F.col("media_dist").cast(FloatType()))\
                 .withColumn("dp_dist",F.col("dp_dist").cast(FloatType()))\
                 .withColumn("preco_min_dist",F.col("preco_min_dist").cast(FloatType()))\
                 .withColumn("preco_max_dist",F.col("preco_max_dist").cast(FloatType()))\
                 .withColumn("coef_var_dist",F.col("coef_var_dist").cast(FloatType()))
```

Checagem de inconsistências nas colunas

```
[ ]
dfpreco.select('margem_rev').distinct().show()
```

Tratamento de inconsistências

```
[ ]
#transforma 'kg' em 'Kg'
dfpreco = dfpreco.withColumn('uni_medida', F.regexp_replace('uni_medida', 'kg', 'Kg'))
```

```
[ ]
#transforma 'OLEO DIESEL' em 'ÓLEO DIESEL'
dfpreco = dfpreco.withColumn('produto', F.regexp_replace('produto', 'OLEO DIESEL', 'ÓLEO DIESEL'))
```

3. Carregamento

Conexão à Conta de Serviço atrelada ao Bucket utilizando as crendenciais dela com uma chave json

```
[ ]  1  # Conexão à Conta de Serviço atrelada ao bucket utilizando a chave json
     2  serviceAccount = '/content/sc-bc26-ed7-adb0dc2607d9.json'
     3  os.environ['GOOGLE_APPLICATION_CREDENTIALS'] = serviceAccount
```

3.1. Carregamento Dataframe de Produção de Biocombustível para o Cloud Storage

```
[ ]  1  dfbio.to_csv('gs://projeto-final-equipe4/arquivos_trat/dfbio_trat',
     2                    storage_options={'token':'/content/sc-bc26-ed7-adb0dc2607d9.json'})
```

3.2. Carregamento Dataframe de Produção de Petróleo e Gás Natural para o Cloud Storage

```
[ ]  1  dfgaspet.to_csv('gs://projeto-final-equipe4/arquivos_trat/dfproducaopetgas_trat',
     2                    storage_options={'token':'/content/sc-bc26-ed7-adb0dc2607d9.json'})
```

3.4. Carregamento Dataframe de Importações e Exportações de Combustíveis

```
[ ]  1  dfder.to_csv('gs://projeto-final-equipe4/arquivos_trat/derivados_trat',
     2                    storage_options={'token':'/content/sc-bc26-ed7-adb0dc2607d9.json'})
```

3.5. Carregamento Dataframe da Série Histórica de Preços de Combustíveis

```
[ ]  1  dfPandas=dfpreco.toPandas()
     2  dfPandas.to_csv('gs://projeto-final-equipe4/arquivos_trat/precos.csv', storage_options={'token':'/content/sc-bc26-ed7-adb0dc2607d9.json'})
```

3.6. Carregamento para o MongoDB

Conexão ao usuário MongoDB a partir de certificado X.509

```
[ ]  1
     2  uri = "mongodb+srv://projfinal-eq4.nvwuziz.mongodb.net/?authSource=%24external&authMechanism=MONGODB-X509&retryWrites=true&w=majority"
     3  client = MongoClient(uri,tls=True,tlsCertificateKeyFile='/content/X509-cert-1745820380525919486.pem')
     4
     5  db = client['projeto-final']
     6  colecao = db['tratados']
     7
```

Utilização de python puro para inserir e checar se os dados tratados foram enviados

```
[ ]  1  datasets = ['derivados_trat','dfbio_trat','dfproducaopetgas_trat','import-export_trat.csv','precos.csv']
     2
     3  for x in datasets:
     4    colecao = db[f'tratados/{x}']
     5    df = pd.read_csv(f'https://storage.googleapis.com/projeto-final-equipe4/arquivos_trat/{x}',sep=';')
     6    df_dict = df.to_dict('records')
     7    colecao.insert_many(df_dict)
     8    print(f'Database "{x}" adicionado ao mongoDB. Número de documentos criados: {colecao.count_documents({})}')
```

```
Database "derivados_trat" adicionado ao mongoDB. Número de documentos criados: 17264
Database "dfbio_trat" adicionado ao mongoDB. Número de documentos criados: 18051
Database "dfproducaopetgas_trat" adicionado ao mongoDB. Número de documentos criados: 5275
Database "import-export_trat.csv" adicionado ao mongoDB. Número de documentos criados: 2754
Database "precos.csv" adicionado ao mongoDB. Número de documentos criados: 77938
```

BigQuery Análises

1. Relação de volume exportação/importação e receita/custo - 2012-2021

RUN SAVE SHARE SCHEDULE MORE

```
SELECT operacao_comercial,SUM(importado_exportado) AS volume_m3,
                          SUM(dispendio_receita) AS receita_custo
FROM projetofinal.dfimport_export_trat
GROUP BY operacao_comercial
```

JOB INFORMATION | RESULTS | JSON | EXECUTIO

Row	operacao_comercial	volume_m3	receita_custo
1	EXPORTAÇÃO	675402800....	262908015...
2	IMPORTAÇÃO	506716844....	217025626...

2. Combustíveis mais exportados pelo Brasil entre 2012-2021

RUN SAVE SHARE SCHEDULE

```
SELECT produto,SUM(importado_exportado) AS volume_m3
FROM projetofinal.dfimport_export_trat
WHERE operacao_comercial = 'EXPORTAÇÃO'
GROUP BY produto
ORDER BY volume_m3 DESC
```

Row	produto	volume_m3
1	PETRÓLEO	520552223....
2	ÓLEO COMBUSTÍVEL	67936797.3...
3	COMBUSTÍVEIS PARA NAVIOS	21736058.0...
4	COMBUSTÍVEIS PARA AERONA...	19419826.5...
5	ETANOL ANIDRO	11618190.0...
6	GASOLINA A	11178197.6...
7	ETANOL HIDRATADO	8961126.089
8	COQUE	5664511.31...
9	QUEROSENE DE AVIAÇÃO	4624281.89...
10	ÓLEO DIESEL	3476330.85...
11	GLP	170569.903...
12	GASOLINA DE AVIAÇÃO	64687.3347...

3. Verificação dos maiores produtores de Gás Natural entre 2012-2021

RUN SAVE SHARE SCHEDU

```
SELECT uf AS estado,SUM(producao) AS volume_m3
FROM projetofinal.dfproducaopetgas_trat
WHERE produto LIKE '%GÁS%'
GROUP BY estado
ORDER BY volume_m3 DESC
```

Row	estado	volume_m3
1	RIO DE JANEIRO	186711506....
2	SÃO PAULO	52538408.2...
3	AMAZONAS	48666519.0...
4	ESPÍRITO SANTO	35557526.4...
5	BAHIA	26020364.7...
6	MARANHÃO	14794884.6...
7	SERGIPE	7397784.04...
8	ALAGOAS	4154039.48...
9	RIO GRANDE DO NORTE	3990015.61...
10	CEARÁ	259855.878...

4. Verificação dos maiores produtores de Petróleo entre 2012-2021

RUN SAVE SHARE SCHEDUL

```
SELECT uf AS estado,SUM(producao) AS volume_m3
FROM projetofinal.dfproducaopetgas_trat
WHERE produto LIKE 'PETRÓLEO'
GROUP BY estado
ORDER BY volume_m3 DESC
```

Row	estado	volume_m3
1	RIO DE JANEIRO	105828212...
2	ESPÍRITO SANTO	187386193....
3	SÃO PAULO	131863270....
4	RIO GRANDE DO NORTE	28083975.7...
5	BAHIA	20040462.3...
6	SERGIPE	15084994.9...
7	AMAZONAS	13481952.5...
8	CEARÁ	3129021.36...
9	ALAGOAS	2027334.74...
10	MARANHÃO	30750.1650...

5. Refinarias que mais produziram derivados entre 2012-2021

RUN

```
SELECT refinaria,SUM(producao) AS volume_m3
FROM projetofinal.dfderivados_trat
GROUP BY refinaria
ORDER BY volume_m3 DESC
```

Row	refinaria	volume_m3
1	REPLAN	203976221....
2	RLAM (ATUALMENTE MATARI...	129996649....
3	REVAP	127591946....
4	REPAR	103994602....
5	REDUC	101715007....

6. Estados que mais produziram derivados entre 2012-2021.

```
SELECT uf AS estado,SUM(producao) AS volume_m3
FROM projetofinal.dfderivados_trat
GROUP BY estado
ORDER BY volume_m3 DESC
```

Row	estado	volume_m3
1	SÃO PAULO	445866707....
2	BAHIA	130582591....
3	RIO DE JANEIRO	106359577....
4	PARANÁ	103994602....
5	RIO GRANDE DO SUL	90450142.4...
6	MINAS GERAIS	80800055.1...
7	PERNAMBUCO	28569108.6...
8	RIO GRANDE DO NORTE	18945225.1...
9	AMAZONAS	15608921.8...
10	CEARÁ	1847841.09...

7. Verificação da Bahia como maior produtor de Óleo Combustível

```
SELECT uf AS estado,SUM(producao) AS volume_m3
FROM projetofinal.dfderivados_trat
WHERE produto LIKE '%ÓLEO COMBUSTÍVEL%'
GROUP BY estado
ORDER BY volume_m3 DESC
```

Row	estado	volume_m3
1	BAHIA	41905075.1...
2	SÃO PAULO	37786255.5...
3	RIO DE JANEIRO	22908133.2...
4	RIO GRANDE DO NORTE	10266351.3...
5	PARANÁ	6965409.51...
6	MINAS GERAIS	5697707.51...
7	RIO GRANDE DO SUL	5578322.94...
8	PERNAMBUCO	4310569.60...
9	AMAZONAS	2345181.69...
10	CEARÁ	1303976.66...

DataFlow

Pipeline Utilizando Template criado no Apache Beam

Para a criação do Dataframe de 'dfcomb_etanol_trat', os dados foram inseridos, transformados e normalizados por meio de uma PIPELINE com modelo criado em Apache Beam usando o Dataflow para o work e enviados para um DataLake.

Neste Dataframe temos uma junção de dois Dataframes anteriormentes tratados "dfbio_trat" e "precos.csv" que estavam do Datalake Google Store.

```
[ ]  1  pip install --upgrade pip
     2  pip install apache_beam[interactive]
     3  pip install apache_beam[gcp]
     4  pip install gcsfs
```

```
[33]  1  import apache_beam as beam
      2  import os
      3  from apache_beam.options.pipeline_options import PipelineOptions
      4  from apache_beam.io.textio import WriteToText
      5
      6  colunas_bio = ['','regiao','uf','produto','volume_m3','data']
      7  colunas_preco = ['data','regiao','estado','produto','postosPes','uniMedida','mediaRev','desvioRev','menorRev','maiorRev','margemRev','coefRev'
      8
      9  def lista_dicionario(elemento, colunas):
     10    return dict(zip(colunas, elemento))
     11
     12  def trata_data(elemento):
     13    # Recebe um dicionario e cria um novo campo com ANO-MES -  Retorna o mesmo dicionario com novo campo
     14    elemento['ano_mes']= '-'.join(elemento['data'].split('-')[:2])
     15    return elemento
     16
     17  def chave_uf(elemento):
     18  #  Receber um dicionario -   Retorna uma tupla com estado e o elemento(UF, dicionario )
     19    chave = elemento['uf']
     20    return (chave, elemento)
     21
     22  def volume(elemento):
     23    #  Recebe um tupla ('SAO PAULO', [{},{}]) -   Retorna uma tupla ('SAO PAULO', 8.0)
     24
     25    uf, registros = elemento
     26    for registros in registros:
     27      yield (f"{uf}-{registros['ano_mes']}", float(registros['volume_m3']))
     28
[33] 29  def chave_estado(elemento):
     30    chave = elemento['estado']
     31    return (chave, elemento)
     32
     33  def mediaRev(elemento):
     34    estado, registros = elemento
     35    for registros in registros:
     36      yield (f"{estado}-{registros['ano_mes']}", float(registros['mediaRev']))
     37
     38  def arredonda(elemento):
     39    #Recebe uma tupla e retorna uma tupla com valor arredondado
     40    chave, valor = elemento
     41    return (chave, round(valor,2))
     42
     43  def filtra_campos_vazios(elemento):
     44    #Remove elementos que tenham chaves vazias] -   Receber uam tupla e retorna a mesma dupla sem campos vazios
     45    chave, dados = elemento
     46    if all([
     47        dados['volume_m3'],
     48        dados['Valor_MedRev']
     49        ]):
     50        return True
     51    return False
```

```python

def descompactar_elementos(elemento):
  #Receber uma tupla ('DISTRITO FEDERAL-2015-10', {'volume_m3': [4.0], 'Valor_MedRev': [11.67]})   Retorna uma tupla ('DISTRITO FEDERAL', '201
  chave, dados = elemento
  volume_m3 = dados['volume_m3'][0] #acessando o primeiro elemento dessa lista [0]
  Valor_MedRev = dados['Valor_MedRev'][0]
  uf, ano, mes = chave.split('-')
  return uf, ano, mes, str(volume_m3), str(Valor_MedRev)  #transformar em str para poder usar o join posteriomente

def preparar_csv(elemento, deliminator=','):
  #Recebe uma tupla e retorna uma string delimitada "DISTRITO FEDERAL;2015;10;4.0;11.67"
  return f"{deliminator}".join(elemento)

pipeline_options = {
    'project':'sc-bc26-ed7',
    'runner':'DataflowRunner',
    'region':'southamerica-east1',
    'staging_location':'gs://projeto-final-equipe4/beam/staging/',
    'temp_location':'gs://projeto-final-equipe4/beam/temp/',
    'template_location':'gs://projeto-final-equipe4/beam/models/modelo_batch'
}

serviceAccount = '/content/sc-bc26-ed7-adb0dc2607d9.json'
os.environ['GOOGLE_APPLICATION_CREDENTIALS'] = serviceAccount

pipeline_options = PipelineOptions.from_dictionary(pipeline_options)

p1 = beam.Pipeline(options=pipeline_options)

biocombustiveis = (
    p1
    |'Extrair do CSV Dataset biocombustiveis'>> beam.io.ReadFromText('gs://projeto-final-equipe4/arquivos_trat/dfbio_trat', skip_header_lines=
    |'Separador de dados Dataset biocombustiveis'>> beam.Map(lambda record: record.split(','))
    |'Filtro por produto Dataset biocombustiveis'>> beam.Filter(lambda record: str(record[3])== 'HIDRATADO')
    |'Tranformar lista para dicionario Dataset biocombustiveis'>>beam.Map(lista_dicionario, colunas_bio)
    |'Criar Campo ano_mes Dataset biocombustiveis'>>beam.Map(trata_data)
    |'Criar chave pelo uf Dataset biocombustiveis'>> beam.Map(chave_uf)
    |'Agrupar pelo uf Dataset biocombustiveis'>>beam.GroupByKey()
    |'Descompactar volume'>>beam.FlatMap(volume)
    |'Soma dos volumes pela chave Dataset biocombustiveis'>> beam.CombinePerKey(sum)
    |'Arredondar resultados'>>beam.Map(arredonda)
    # |'Imprimir o resultado'>> beam.Map(print)
)

precos = (
    p1
    |'Extrair do CSV Dataset Preços'>> beam.io.ReadFromText('gs://projeto-final-equipe4/arquivos_trat/precos.csv', skip_header_lines=1)
    |'Separador de dados Dataset Preços'>> beam.Map(lambda record: record.split(','))
    |'Filtro por produto Dataset Preços'>> beam.Filter(lambda record: str(record[3])== 'ETANOL HIDRATADO')
    |'Tranformar lista para dicionario Dataset Preços'>>beam.Map(lista_dicionario, colunas_preco)
    |'Criar Campo ano_mes Dataset Preços'>>beam.Map(trata_data)
    |'Criar chave pelo estado Dataset Preços'>> beam.Map(chave_estado)
    |'Agrupar pelo estado Dataset Preços'>>beam.GroupByKey()
    |'Descompactar volume Dataset Preços'>>beam.FlatMap(mediaRev)
    |'Media de preços pela chave'>> beam.combiners.Mean.PerKey()
    |'Arredondar resultados de preços'>>beam.Map(arredonda)
    # |'Imprimir o resultado Dataset Preços'>> beam.Map(print)
)

resultado = (
    ({'volume_m3':biocombustiveis,'Valor_MedRev':precos})
    |'Mesclar pcollections'>>beam.CoGroupByKey()
    |'Filtrar dados vazios'>>beam.Filter(filtra_campos_vazios)
    |'Descompactar elementos'>>beam.Map(descompactar_elementos)
    |'Preparar csv'>>beam.Map(preparar_csv, deliminator=',')
    #|'Imprimir o resultado da união'>> beam.Map(print)
    |'Load arquivo final CSV'>> beam.io.WriteToText('gs://projeto-final-equipe4/arquivos_trat/dfcomb_etanol_trat', file_name_suffix='.csv', he
)


p1.run()
```

projeto-comb-final	Batch	Jan 10, 2023, 5:52:37 PM	5 min 21 sec	Jan 10, 2023, 5:47:16 PM	Succeeded	2.43.0	2023-01-10_12_47_15-4708610437070038027	southar east1

Resource metrics

Current vCPUs	1
Total vCPU time	0.055 vCPU hr
Current memory	3.75 GB
Total memory time	0.207 GB hr
Current HDD PD	25 GB
Total HDD PD time	1.378 GB hr
Current SSD PD	0 B
Total SSD PD time	0 GB hr
Total Shuffle data processed	12.61 MB
Billable Shuffle data processed	3.15 MB

DataRoot

Pipeline Utilizando Template pré-definido: BigQuery to MongoDB

Name	Type	End time	Elapsed time	Start time	Status	SDK version	ID	Region
dfcomb-etanol-trat	Batch	Jan 10, 2023, 6:16:19 PM	4 min 59 sec	Jan 10, 2023, 6:11:20 PM	Succeeded	2.43.0	2023-01-10_13_11_20-6428002500645512000	us-east1
dfprecos-tratado	Batch	Jan 10, 2023, 5:03:10 PM	8 min 12 sec	Jan 10, 2023, 4:54:58 PM	Succeeded	2.43.0	2023-01-10_11_54_57-7890401685796579623	us-east1
dfimport-export-trat	Batch	Jan 10, 2023,	4 min 49 sec	Jan 10, 2023,	Succeeded	2.43.0	2023-01-10_11_20_08-11767284456004812462	us-east1

Job info

Resource metrics

Current vCPUs	1
Total vCPU time	0.027 vCPU hr
Current memory	3.75 GB
Total memory time	0.103 GB hr
Current HDD PD	25 GB
Total HDD PD time	0.686 GB hr
Current SSD PD	0 B
Total SSD PD time	0 GB hr
Total Shuffle data processed	200 B
Billable Shuffle data processed	50 B

Job info

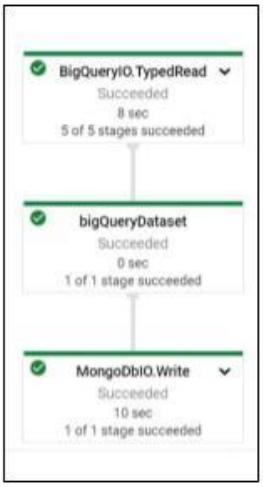
Resource metrics

Current vCPUs	1
Total vCPU time	0.089 vCPU hr
Current memory	3.75 GB
Total memory time	0.333 GB hr
Current HDD PD	25 GB
Total HDD PD time	2.217 GB hr
Current SSD PD	0 B
Total SSD PD time	0 GB hr
Total Shuffle data processed	200 B
Billable Shuffle data processed	50 B

Job info

Resource metrics

Current vCPUs	1
Total vCPU time	0.031 vCPU hr
Current memory	3.75 GB
Total memory time	0.116 GB hr
Current HDD PD	25 GB
Total HDD PD time	0.771 GB hr
Current SSD PD	0 B
Total SSD PD time	0 GB hr
Total Shuffle data processed	200 B
Billable Shuffle data processed	50 B



projeto_final_combustiveis_viabigquery

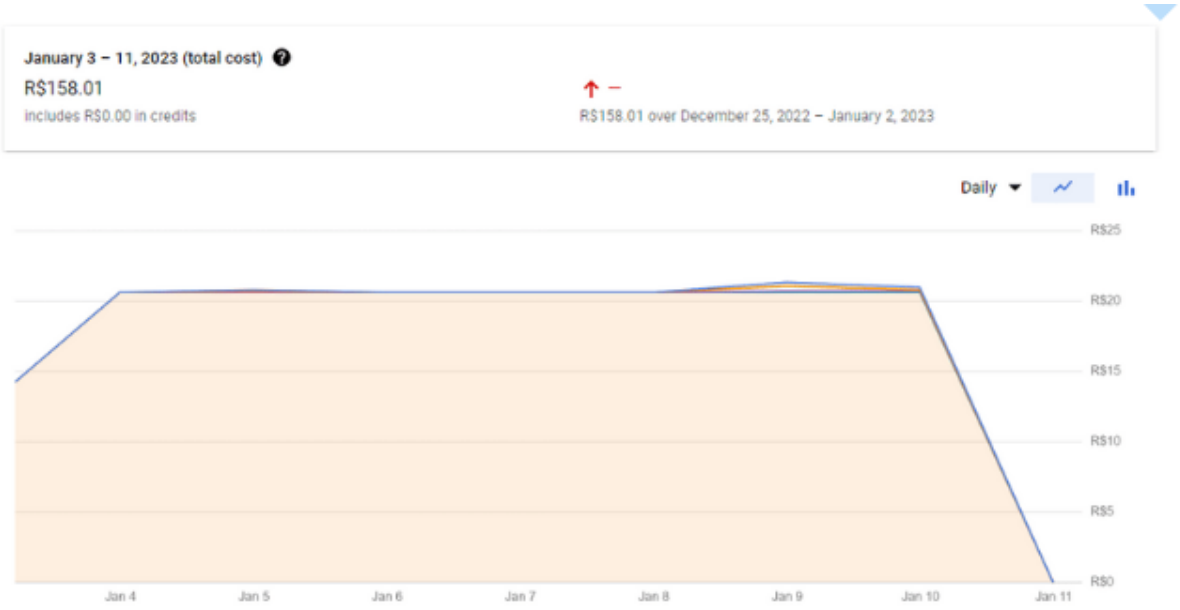
LOGICAL DATA SIZE: 33.8MB STORAGE SIZE: 9.29MB INDEX SIZE: 3.09MB TOTAL COLLECTIONS: 3

CREATE COLLECTION

Collection Name	Documents	Logical Data Size	Avg Document Size	Storage Size	Indexes	Index Size	Avg Index Size
dfcomb_etanol_trat	3106	365.96KB	121B	152KB	1	120KB	120KB
dfimport-export_trat	2754	669.87KB	250B	204KB	1	120KB	120KB
dfprecos_tratado	87034	32.79MB	396B	8.94MB	1	2.86MB	2.86MB

Levantamento de Custos

Nosso relatório de gastos liberado pela Google Cloud de custos de projetos utilizados no ambiente



Service	Cost	Discounts	Promotions and others	Subtotal
Cloud SQL	R$156.80	–	–	R$156.80
Cloud Dataflow	R$0.59	–	–	R$0.59
Networking	R$0.52	–	–	R$0.52
Cloud Storage	R$0.09	–	–	R$0.09
Compute Engine	R$0.00	–	–	R$0.00
Cloud Logging	R$0.00	–	–	R$0.00
BigQuery	R$0.00	–	–	R$0.00

DataRoot

Referências

- Pandas Documentation, 2007. Disponível em: <https://pandas.pydata.org/docs/index.html> Acesso em: 03 Jan. 2023.

- Apache Spark Documentation, 2022. Disponível em: <https://spark.apache.org/documentation.html> Acesso em: 03 Jan. 2023.

- Apache Beam, 2022. Disponível em: <https://beam.apache.org/documentation/programming-guide/#pardo> Acesso em: 07 Jan. 2023.

- Google Cloud Documentation, 2022. Disponível em: <https://cloud.google.com/docs> Acesso em: 05 Jan. 2023.