MVP Engenharia de Dados

Cássio Eduardo Kessler

Objetivo

O objetivo do trabalho é analisar a matriz elétrica brasileira a partir de dados de capacidade instalada de geração de energia elétrica fornecidos pelo Operador Nacional do Sistema Elétrico (ONS). Espera-se identificar as principais fontes de geração de energia elétrica que compõem a matriz nacional, bem como avaliar algumas questões pontuais, como a lista de usinas que foram desativadas em um determinado ano ou o montante de potência adicionada de cada fonte em um certo período.

Busca-se ainda avaliar, para algumas fontes, em quais estados está localizada a maior capacidade instalada. Por fim, será feita uma comparação da composição da matriz elétrica atual com a matriz de alguns momentos do passado, de forma a identificar as fontes que mais cresceram nos últimos anos. Assim, podemos observar a evolução da matriz, onde espera-se identificar um maior crescimento das fontes renováveis solar e eólica nos últimos anos. De forma a atender a esse objetivo, foram elaboradas as seguintes questões:

1. Qual o percentual de participação de cada fonte de geração na matriz elétrica brasileira atualmente?
2. Qual o montante de capacidade adicionada no ano de 2023, segregado por fonte?
3. Listar as usinas desativadas no ano de 2023.
4. Quais os 3 estados da federação com a maior capacidade de geração instalada da fonte solar?
5. Quais os 3 estados da federação com a maior capacidade de geração instalada da fonte eólica?
6. Demonstrar a evolução da matriz elétrica brasileira, comparando a composição atual com os anos de 2000, 2010 e 2020.

Plataforma

Foi utilizada a Plataforma Databricks, na sua versão de uso chamada Databricks Community Edition. Apesar dessa versão possuir limitação na qualidade e quantidade de máquinas no cluster, sua utilização é gratuita e seus recursos são suficientes para a realização deste trabalho.

Detalhamento

1. Busca pelos dados

Os dados utilizados são provenientes da Seção de Dados Abertos do Operador Nacional do Sistema Elétrico (ONS). Foi selecionado o seguinte conjunto de dados:

- Capacidade Instalada de Geração (https://dados.ons.org.br/dataset/capacidade-geracao)

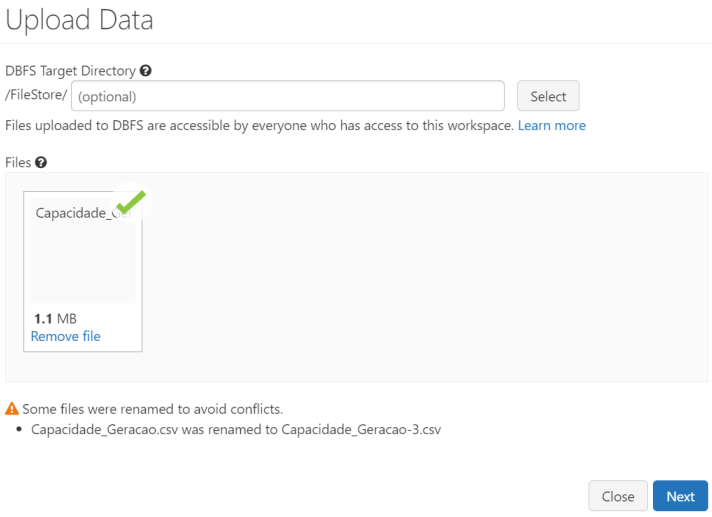
Este *dataset* apresenta dados de potência nominal de Unidades Geradoras de Usinas despachadas pelo ONS, que são aquelas classificadas com modalidade de operação Tipo I, Tipo II-A, Tipo II-B e Tipo II-C.

Estes dados possuem licença Creative Commons CC-BY, que permite que os reutilizadores distribuam, modifiquem, adaptem e desenvolvam o material sobre os dados, desde que seja atribuído o crédito apropriado ao criador (ONS) e que sejam informadas as alterações.

2. Coleta

O arquivo csv com os dados de interesse foi baixado manualmente no site do ONS (https://dados.ons.org.br/dataset/capacidade-geracao) para a máquina local.

Em seguida, foi realizado o upload arquivo csv para o Databricks File System (DBFS).



3. Modelagem

O modelo de dados utilizados foi o *flat*, com todos os dados em uma única tabela. Abaixo é descrito o Dicionário de Dados considerando os dados em seu estado bruto, sem qualquer modificação (camada bronze).

Coluna	Descrição	Tipo	Restrições
_id	Identificador único da entrada	Integer	Não permite valor nulo
id_subsistema	Código do Subsistema da Usina	String	Não permite valor nulo
nom_subsistema	Nome do Subsistema	String	Não permite valor nulo
id_estado	Sigla do Estado onde está localizada a usina	String	Não permite valor nulo. Deve ser a sigla de um dos 27 estados, além de "I" que representa o Paraguai em sua parcela de Itaipu
nom_estado	Nome do Estado	String	Não permite valor nulo
nom_modalidadeoperacao	Modalidade de Operação da Usina	String	Não permite valor nulo
nom_agenteproprietario	Agente Proprietário da Usina	String	Não permite valor nulo
nom_tipousina	Tipo da Usina	String	Não permite valor nulo
nom_usina	Nome da Usina	String	Não permite valor nulo
ceg	Código Único do Empreendimento de Geração (CEG), estabelecido pela ANEEL	String	Não permite valor nulo
nom_unidadegeradora	Nome da Unidade Geradora	String	Não permite valor nulo
cod_equipamento	Código do Equipamento – Unidade Geradora	String	Não permite valor nulo
num_unidadegeradora	Código Operacional da Unidade Geradora	String	Não permite valor nulo
nom_combustivel	Combustível da Unidade Geradora	String	Não permite valor nulo
dat_entradateste	Data da Liberação para Entrada em Comissionamento	Date	
dat_entradaoperacao	Data da Liberação para Entrada em Operação Comercial	Date	Não permite valor nulo
dat_desativacao	Data de Desativação da Unidade Geradora	Date	O conteúdo NULL (ou vazio) representa que a unidade geradora está ativa
val_potenciaefetiva	Potência Nominal da Unidade Geradora, conforme Documento Normativo da ANEEL, em MW	Float	Não permite valor nulo, valor deve ser maior do que zero

Após a primeira etapa de transformação (camada silver), descrita na próxima seção, a tabela resultante ficou com as seguintes características:

Coluna	Descrição	Tipo	Restrições
_id	Identificador único da entrada	Integer	Não permite valor nulo
id_estado	Sigla do Estado onde está localizada a usina	String	Não permite valor nulo. Deve ser a sigla de um dos 27 estados, além de "I" que representa o Paraguai em sua parcela de Itaipu
nom_tipousina	Tipo da Usina	String	Não permite valor nulo
nom_usina	Nome da Usina	String	Não permite valor nulo
ceg	Código Único do Empreendimento de Geração (CEG), estabelecido pela ANEEL	String	Não permite valor nulo
cod_equipamento	Código do Equipamento – Unidade Geradora	String	Não permite valor nulo
nom_combustivel	Combustível da Unidade Geradora	String	Não permite valor nulo
dat_entradaoperacao	Data da Liberação para Entrada em Operação Comercial	Date	Não permite valor nulo
dat_desativacao	Data de Desativação da Unidade Geradora	Date	O conteúdo NULL (ou vazio) representa que a unidade geradora está ativa
val_potenciaefetiva	Potência Nominal da Unidade Geradora, conforme Documento Normativo da ANEEL, em MW	Date	Não permite valor nulo, valor deve ser maior do que zero

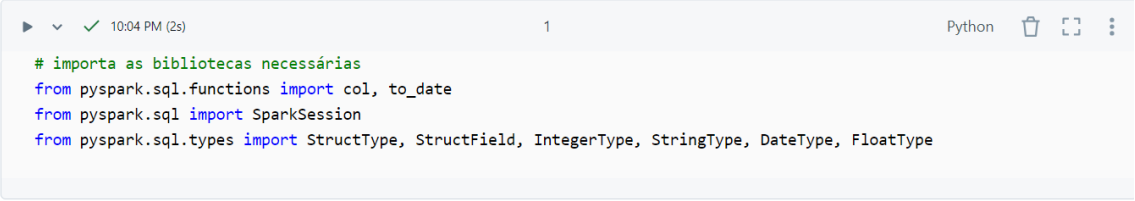
Por fim, foi criada a camada gold. Nesta tabela, foram mantidas apenas as linhas em que a data de desativação possuía valor nulo. Ou seja, foram mantidas apenas as usinas que permanecem ativas atualmente, excluindo aquelas que foram desativadas. Ressalta-se que as informações das usinas desativadas foram mantidas na camada silver para a realização de consultas de dados históricos, enquanto aos usuários da camada gold apenas as informações atuais são relevantes. A tabela abaixo apresenta as características dos dados da camada gold.

Coluna	Descrição	Tipo	Restrições
_id	Identificador único da entrada	Integer	Não permite valor nulo
id_estado	Sigla do Estado onde está localizada a usina	String	Não permite valor nulo. Deve ser a sigla de um dos 27 estados, além de "I" que

			representa o Paraguai em sua parcela de Itaipu
nom_tipousina	Tipo da Usina	String	Não permite valor nulo
nom_usina	Nome da Usina	String	Não permite valor nulo
ceg	Código Único do Empreendimento de Geração (CEG), estabelecido pela ANEEL	String	Não permite valor nulo
cod_equipamento	Código do Equipamento – Unidade Geradora	String	Não permite valor nulo
nom_combustivel	Combustível da Unidade Geradora	String	Não permite valor nulo
dat_entradaoperacao	Data da Liberação para Entrada em Operação Comercial	Date	Não permite valor nulo
dat_desativacao	Data de Desativação da Unidade Geradora	Date	O conteúdo NULL (ou vazio) representa que a unidade geradora está ativa. Na camada gold, todas as entradas possuem valor NULL
val_potenciaefetiva	Potência Nominal da Unidade Geradora, conforme Documento Normativo da ANEEL, em MW	Date	Não permite valor nulo, valor deve ser maior do que zero

4. Carga

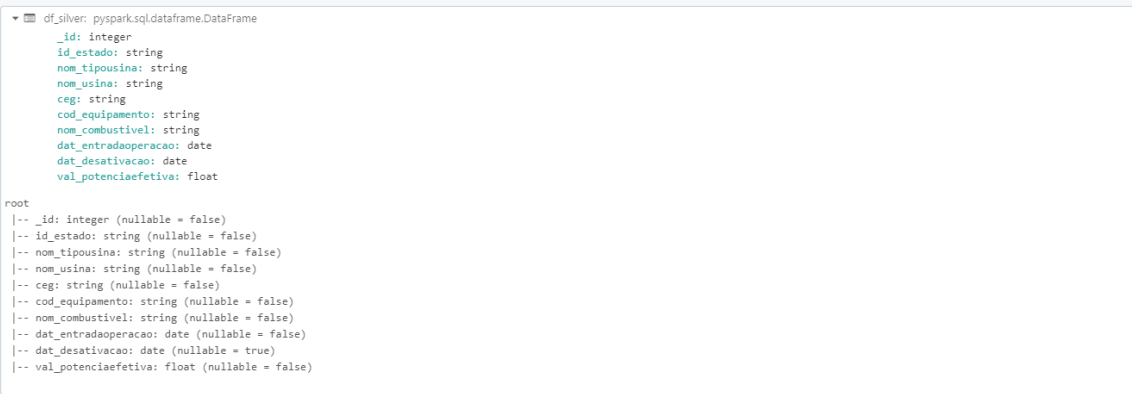
Inicialmente foram carregadas as bibliotecas necessárias e criada a camada bronze carregando o arquivo csv a partir do Data Brick File System (DBFS).



```
10:04 PM (28s)                                                    2

# Cria um DataFrame com o conteúdo do arquivo csv (bronze)
df_bronze = spark.read.format("csv").option("header", "true").load("dbfs:/FileStore/Capacidade_Geracao.csv")

(1) Spark Jobs

df_bronze: pyspark.sql.dataframe.DataFrame
      _id: string
      id_subsistema: string
      nom_subsistema: string
      id_estado: string
      nom_estado: string
      nom_modalidadeoperacao: string
      nom_agenteproprietario: string
      nom_tipousina: string
      nom_usina: string
      ceg: string
      nom_unidadegeradora: string
      cod_equipamento: string
      num_unidadegeradora: string
      nom_combustivel: string
      dat_entradateste: string
      dat_entradaoperacao: string
      dat_desativacao: string
      val_potenciaefetiva: string
```

Na etapa seguinte foi criada a camada Silver, na qual foram realizadas algumas transformações nos dados. Inicialmente, foram descartadas as colunas que não tinham relevância para as análises propostas. Em seguida foram corrigidos os tipos de dados de algumas colunas, pois estavam equivocadamente classificados como “string” na camada bronze. Também foi ajustado o esquema com as colunas que não podem aceitar valores nulos.

```
08:54 PM (2s)                                                     3                                                    Python

#Tratamento dos dados (camada silver)
spark = SparkSession.builder.appName("ManipularDataFrame").getOrCreate()
selected_columns = ["_id", "id_estado", "nom_tipousina", "nom_usina", "ceg", "cod_equipamento", "nom_combustivel", "dat_entradaoperacao", "dat_desativacao", "val_potenciaefetiva"]
df_silver = df_bronze.select(selected_columns)

# Converter as colunas "dat_entradaoperacao" e "dat_desativacao" para o formato data
df_silver = df_silver.withColumn("dat_entradaoperacao", to_date(col("dat_entradaoperacao"), "yyyy-MM-dd'T'HH:mm:ss")) \
                     .withColumn("dat_desativacao", to_date(col("dat_desativacao"), "yyyy-MM-dd'T'HH:mm:ss"))

# Converter a coluna "val_potenciaefetiva" para float e a coluna "_id" para integer
df_silver = df_silver.withColumn("val_potenciaefetiva", col("val_potenciaefetiva").cast("float")) \
                     .withColumn("_id",col("_id").cast("integer"))

# Definir o novo esquema com nullable = false as colunas que devem ter essa característica
new_schema = StructType([
    StructField("_id", IntegerType(), nullable=False),
    StructField("id_estado", StringType(), nullable=False),
    StructField("nom_tipousina", StringType(), nullable=False),
    StructField("nom_usina", StringType(), nullable=False),
    StructField("ceg", StringType(), nullable=False),
    StructField("cod_equipamento", StringType(), nullable=False),
    StructField("nom_combustivel", StringType(), nullable=False),
    StructField("dat_entradaoperacao", DateType(), nullable=False),
    StructField("dat_desativacao", DateType(), nullable=True),
    StructField("val_potenciaefetiva", FloatType(), nullable=False)
])

# Aplicar o novo esquema ao DataFrame
df_silver = spark.createDataFrame(df_silver.rdd, schema=new_schema)

# Exibir o esquema do novo DataFrame para verificar as alterações
df_silver.printSchema()
```



Em seguida, após a verificação da qualidade dos dados, que será mostrada na próxima seção, foi criada a tabela_silver, que será usada posteriormente em algumas das consultas.

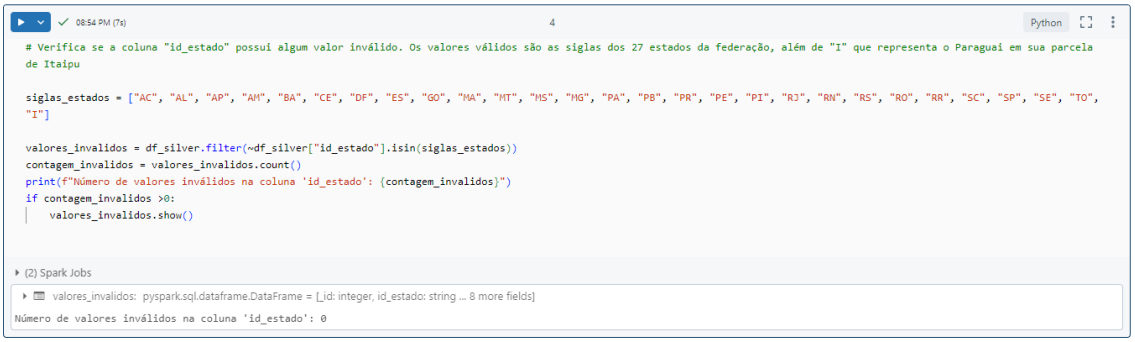
Por fim, foi criada a camada gold. Nesta tabela foram mantidos apenas os registros das usinas em que a campo da data de desativação estava em branco. Dessa forma, os usuários finais que utilizarão esta tabela poderão apenas fazer consultas referentes a usinas que estão em operação atualmente. Cabe ressaltar que os dados de usinas desativadas foram mantidas na camada silver para eventuais consultas de usuários específicos.

5. Análise

a. Qualidade de dados

Nesta etapa foi feita uma análise da qualidade dos dados, buscando identificar eventuais problemas.

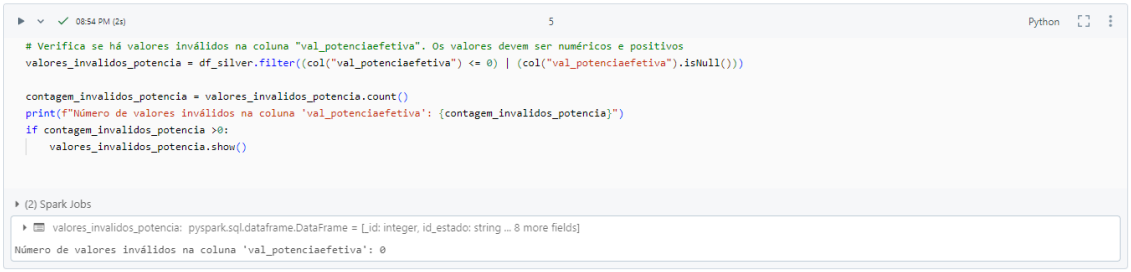
O primeiro teste realizado foi na coluna “id_estado”. Esta coluna identifica o estado onde a usina está instalada, de forma que os valores válidos são as siglas dos 27 estados brasileiros, além do código "I", que representa o Paraguai em sua parcela da usina hidrelétrica Itaipu. O teste indicou que não havia valores inválidos no conjunto de dados.



```
# Verifica se a coluna "id_estado" possui algum valor inválido. Os valores válidos são as siglas dos 27 estados da federação, além de "I" que representa o Paraguai em sua parcela de Itaipu

siglas_estados = ["AC", "AL", "AP", "AM", "BA", "CE", "DF", "ES", "GO", "MA", "MT", "MS", "MG", "PA", "PB", "PR", "PE", "PI", "RJ", "RN", "RS", "RO", "RR", "SC", "SP", "SE", "TO", "I"]

valores_invalidos = df_silver.filter(~df_silver["id_estado"].isin(siglas_estados))
contagem_invalidos = valores_invalidos.count()
print(f"Número de valores inválidos na coluna 'id_estado': {contagem_invalidos}")
if contagem_invalidos >0:
    valores_invalidos.show()
```

```
(2) Spark Jobs
valores_invalidos: pyspark.sql.dataframe.DataFrame = [_id: integer, id_estado: string ... 8 more fields]
Número de valores inválidos na coluna 'id_estado': 0
```

O teste seguinte verificou se havia algum dado inválido na coluna “val_potenciaefetiva”. Para esta coluna, são aceitos apenas valores numéricos maiores do que zero. Também não são aceitos valores em branco. Este teste também não identificou valores inválidos.



```
# Verifica se há valores inválidos na coluna "val_potenciaefetiva". Os valores devem ser numéricos e positivos
valores_invalidos_potencia = df_silver.filter((col("val_potenciaefetiva") <= 0) | (col("val_potenciaefetiva").isNull()))

contagem_invalidos_potencia = valores_invalidos_potencia.count()
print(f"Número de valores inválidos na coluna 'val_potenciaefetiva': {contagem_invalidos_potencia}")
if contagem_invalidos_potencia >0:
    valores_invalidos_potencia.show()
```

```
(2) Spark Jobs
valores_invalidos_potencia: pyspark.sql.dataframe.DataFrame = [_id: integer, id_estado: string ... 8 more fields]
Número de valores inválidos na coluna 'val_potenciaefetiva': 0
```

Para as demais colunas, o único requisito era que não seriam aceitos valores nulos. O teste realizado não identificou nenhum dado inválido.

```
# Verifica se há algum valor nula nas colunas que não dever ter valores nulos
colunas_nao_nulas = ["_id", "id_estado", "nom_tipousina", "nom_usina", "ceg", "cod_equipamento", "nom_combustivel", "dat_entradaoperacao", "val_potenciaefetiva"]

# Filtra as linhas que possuem valores nulos em alguma das colunas não nulas
valores_nulos = df_silver.filter(
    sum(col(coluna).isNull().cast("int") for coluna in colunas_nao_nulas) > 0
)

# Contagem dos valores nulos
contagem_nulos = valores_nulos.count()
print(f"Número de valores nulos nas colunas não nulas: {contagem_nulos}")

# Exibindo os valores nulos apenas se a contagem for maior que zero
if contagem_nulos > 0:
    valores_nulos.show()
```

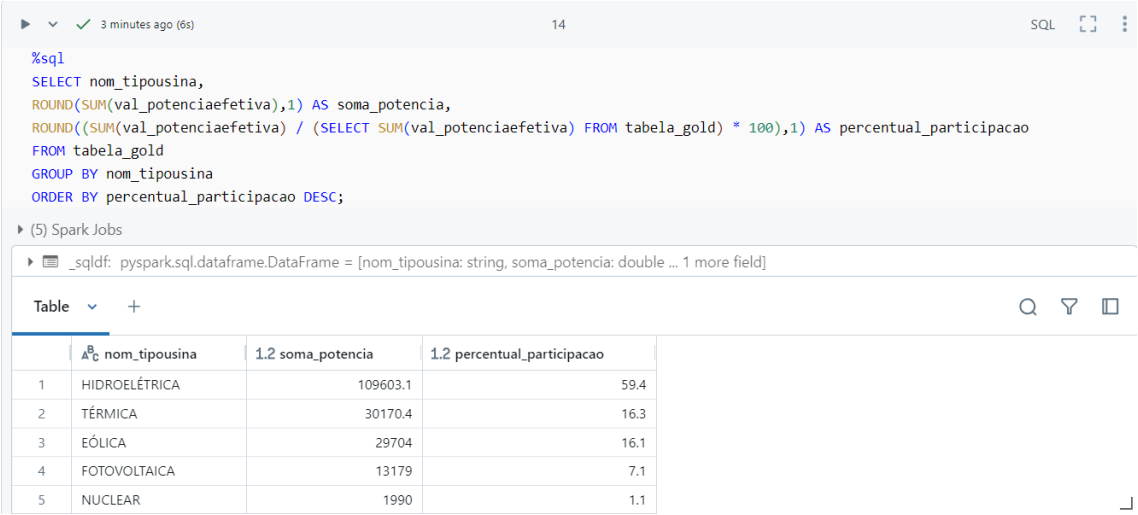
```
(1) Spark Jobs
valores_nulos: pyspark.sql.dataframe.DataFrame = [_id: integer, id_estado: string ... 8 more fields]
Número de valores nulos nas colunas não nulas: 0
```

Como foi demonstrado, não foi identificado nenhum problema a ser corrigido. Provavelmente isso se deve ao fato de que a fonte dos dados (ONS) já verifica e trata a qualidade dos dados antes de torná-los públicos.

b. Solução do problema

Por fim, buscou-se responder às questões elencadas no objetivo do trabalho. Para isso, foram realizadas consultas nos dados utilizando a linguagem SQL.

Questão 1: Qual o percentual de participação de cada fonte de geração na matriz elétrica brasileira atualmente?



```
%sql
SELECT nom_tipousina,
ROUND(SUM(val_potenciaefetiva),1) AS soma_potencia,
ROUND((SUM(val_potenciaefetiva) / (SELECT SUM(val_potenciaefetiva) FROM tabela_gold) * 100),1) AS percentual_participacao
FROM tabela_gold
GROUP BY nom_tipousina
ORDER BY percentual_participacao DESC;
```

(5) Spark Jobs

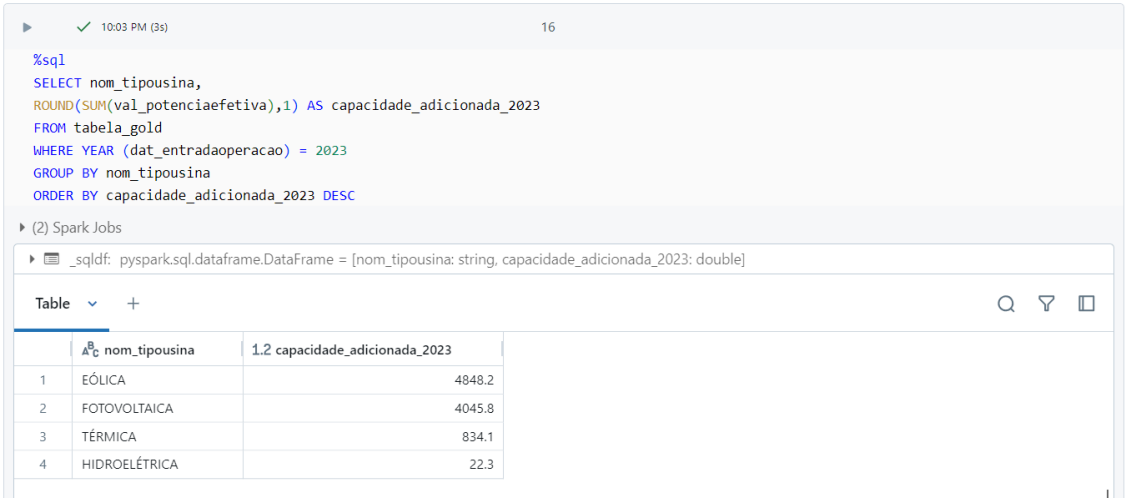
_sqldf: pyspark.sql.dataframe.DataFrame = [nom_tipousina: string, soma_potencia: double ... 1 more field]

Table

	nom_tipousina	soma_potencia	percentual_participacao
1	HIDROELÉTRICA	109603.1	59.4
2	TÉRMICA	30170.4	16.3
3	EÓLICA	29704	16.1
4	FOTOVOLTAICA	13179	7.1
5	NUCLEAR	1990	1.1

O resultado da consulta mostra que as usinas hidroelétricas são a principal fonte de geração de energia no Brasil, correspondendo a 59,4% do total de capacidade instalada. Em segundo lugar, a fonte térmica representa 16,3%. Esse resultado era esperado, uma vez que a matriz de energia brasileira é historicamente caracterizada como hidrotérmica. Em seguida, as fontes eólica (16,1%), fotovoltaica (7,1%) e nuclear (1,1%) completam a matriz.

Questão 2: Qual o montante de capacidade adicionada no ano de 2023, segregado por fonte?



```
%sql
SELECT nom_tipousina,
ROUND(SUM(val_potenciaefetiva),1) AS capacidade_adicionada_2023
FROM tabela_gold
WHERE YEAR (dat_entradaoperacao) = 2023
GROUP BY nom_tipousina
ORDER BY capacidade_adicionada_2023 DESC
```

(2) Spark Jobs

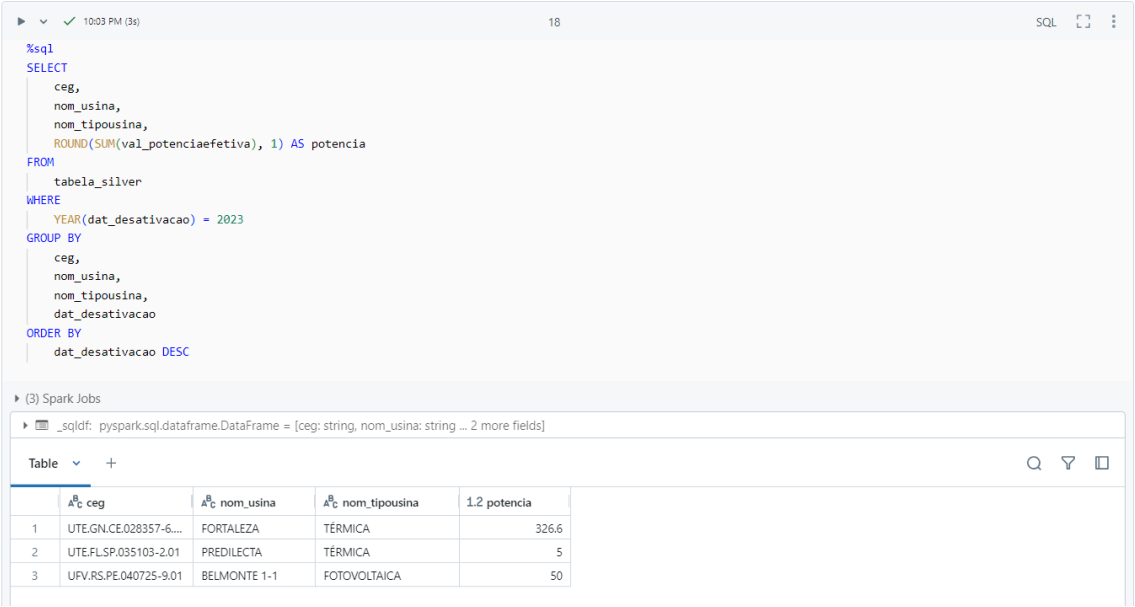
_sqldf: pyspark.sql.dataframe.DataFrame = [nom_tipousina: string, capacidade_adicionada_2023: double]

Table

	nom_tipousina	capacidade_adicionada_2023
1	EÓLICA	4848.2
2	FOTOVOLTAICA	4045.8
3	TÉRMICA	834.1
4	HIDROELÉTRICA	22.3

Nos últimos anos, tem se observado no setor elétrico brasileiro um forte crescimento das fontes renováveis eólica e solar. O resultado da consulta confirma este fato, mostrando que estas fontes foram as principais responsáveis pela expansão no ano de 2023.

Questão 3: Listar as usinas desativadas no ano de 2023.



```
%sql
SELECT
    ceg,
    nom_usina,
    nom_tipousina,
    ROUND(SUM(val_potenciaefetiva), 1) AS potencia
FROM
    tabela_silver
WHERE
    YEAR(dat_desativacao) = 2023
GROUP BY
    ceg,
    nom_usina,
    nom_tipousina,
    dat_desativacao
ORDER BY
    dat_desativacao DESC
```

(3) Spark Jobs

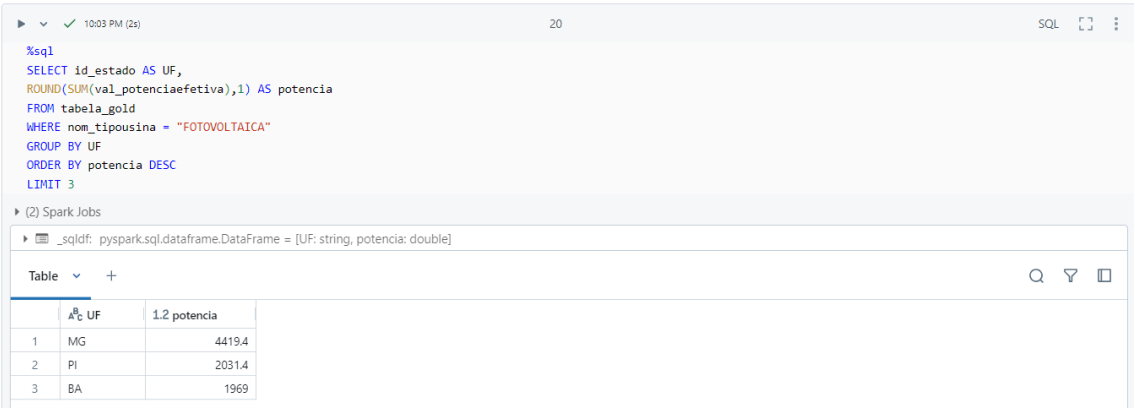
_sqldf: pyspark.sql.dataframe.DataFrame = [ceg: string, nom_usina: string ... 2 more fields]

Table

	ceg	nom_usina	nom_tipousina	potencia
1	UTE.GN.CE.028357-6....	FORTALEZA	TÉRMICA	326.6
2	UTE.FL.SP.035103-2.01	PREDILECTA	TÉRMICA	5
3	UFV.RS.PE.040725-9.01	BELMONTE 1-1	FOTOVOLTAICA	50

O resultado da consulta indicou apenas três usinas desativadas no ano de 2023, sendo duas termelétricas e uma solar fotovoltaica.

Questão 4: Quais os 3 estados da federação com a maior capacidade de geração instalada da fonte solar?



```
%sql
SELECT id_estado AS UF,
ROUND(SUM(val_potenciaefetiva),1) AS potencia
FROM tabela_gold
WHERE nom_tipousina = "FOTOVOLTAICA"
GROUP BY UF
ORDER BY potencia DESC
LIMIT 3
```

(2) Spark Jobs

_sqldf: pyspark.sql.dataframe.DataFrame = [UF: string, potencia: double]

Table

	UF	potencia
1	MG	4419.4
2	PI	2031.4
3	BA	1969

A consulta indicou que o estado de Minas Gerais é o que possui maior capacidade instalada da fonte solar fotovoltaica, seguido por Piauí e Bahia. De fato, esses estados possuem alta incidência solar e, por isso, são locais favoráveis para a instalação desta fonte.

Questão 5: Quais os 3 estados da federação com a maior capacidade de geração instalada da fonte eólica?

```
%sql
SELECT id_estado AS UF,
ROUND(SUM(val_potenciaefetiva),1) AS potencia
FROM tabela_gold
WHERE nom_tipousina = "EÓLICA"
GROUP BY UF
ORDER BY potencia DESC
LIMIT 3
```

(2) Spark Jobs

_sqldf: pyspark.sql.dataframe.DataFrame = [UF: string, potencia: double]

Table

	UF	1.2 potencia
1	RN	9744.4
2	BA	9472
3	PI	3921.8

Nesta questão, esperava-se que o resultado indicasse estados da região Nordeste, conhecida por seus ventos favoráveis à geração eólica. De fato, a consulta indicou que os estados com maior potência instalada eólica são Rio Grande do Norte, Bahia e Piauí.

Questão 6: Demonstrar a evolução da matriz elétrica brasileira, comparando a composição atual com os anos de 2000, 2010 e 2020.

```
%sql
SELECT
    CASE
        WHEN nom_tipousina = 'BOMBEAMENTO' THEN 'HIDROELÉTRICA'
        WHEN nom_tipousina = 'EOLIELÉTRICA' THEN 'EÓLICA'
        ELSE nom_tipousina
    END AS nom_tipousina,
    ROUND(SUM(CASE WHEN YEAR(dat_entradaoperacao) <= 2000 AND (dat_desativacao IS NULL OR YEAR(dat_desativacao) > 2000) THEN val_potenciaefetiva ELSE 0 END), 1) AS potencia_2000,
    ROUND(SUM(CASE WHEN YEAR(dat_entradaoperacao) <= 2010 AND (dat_desativacao IS NULL OR YEAR(dat_desativacao) > 2010) THEN val_potenciaefetiva ELSE 0 END), 1) AS potencia_2010,
    ROUND(SUM(CASE WHEN YEAR(dat_entradaoperacao) <= 2020 AND (dat_desativacao IS NULL OR YEAR(dat_desativacao) > 2020) THEN val_potenciaefetiva ELSE 0 END), 1) AS potencia_2020,
    ROUND(SUM(CASE WHEN YEAR(dat_entradaoperacao) <= 2024 AND (dat_desativacao IS NULL OR YEAR(dat_desativacao) > 2024) THEN val_potenciaefetiva ELSE 0 END), 1) AS potencia_2024
FROM
    tabela_silver
GROUP BY
    CASE
        WHEN nom_tipousina = 'BOMBEAMENTO' THEN 'HIDROELÉTRICA'
        WHEN nom_tipousina = 'EOLIELÉTRICA' THEN 'EÓLICA'
        ELSE nom_tipousina
    END
ORDER BY
    nom_tipousina;
```

(2) Spark Jobs

_sqldf: pyspark.sql.dataframe.DataFrame = [nom_tipousina: string, potencia_2000: double ... 3 more fields]

Table

	nom_tipousina	1.2 potencia_2000	1.2 potencia_2010	1.2 potencia_2020	1.2 potencia_2024
1	EÓLICA	0	567.8	16058.5	29704
2	FOTOVOLTAICA	0	0	2879	13179
3	HIDROELÉTRICA	65234.1	83347.8	109444.4	109603.1
4	NUCLEAR	640	1990	1990	1990
5	TÉRMICA	4925.9	17143.9	27322.7	30170.4

O resultado da consulta mostra a evolução da matriz elétrica ao longo dos anos. Em 2000, havia apenas hidroelétricas e térmicas, além de uma pequena parcela de nuclear. Entre 2000 e 2010, houve um crescimento relevante destas mesmas fontes (hidráulica e térmica), enquanto outras fontes eram pouco representativas ou até mesmo inexistentes, como a solar. Porém, mais recentemente, observa-se maior crescimento das fontes renováveis solar e eólica.

De forma a ilustrar melhor essa evolução, foi construído um gráfico:

11:43 PM (2s) 25

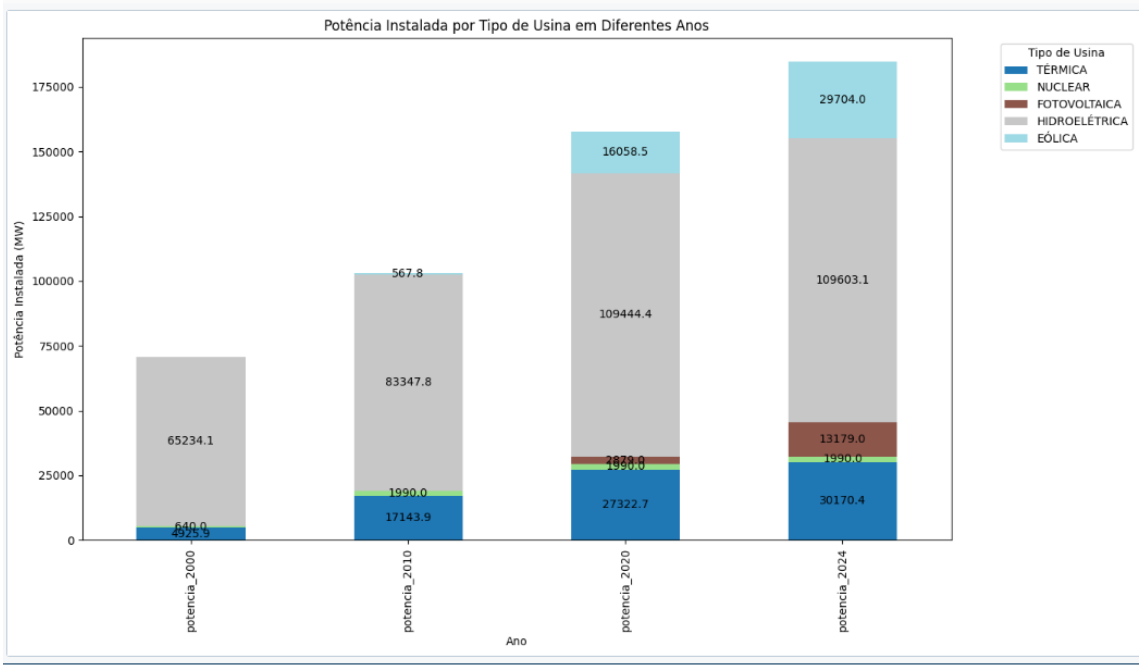
```
# Executar a consulta SQL e obter os resultados em um DataFrame do Spark
df_spark = spark.sql("""
SELECT
    CASE
        WHEN nom_tipousina = 'BOMBEAMENTO' THEN 'HIDROELÉTRICA'
        WHEN nom_tipousina = 'EOLIELÉTRICA' THEN 'EÓLICA'
        ELSE nom_tipousina
    END AS nom_tipousina,
    SUM(CASE WHEN YEAR(dat_entradaoperacao) <= 2000 AND (dat_desativacao IS NULL OR YEAR(dat_desativacao) > 2000) THEN val_potenciaefetiva ELSE 0 END) AS potencia_2000,
    SUM(CASE WHEN YEAR(dat_entradaoperacao) <= 2010 AND (dat_desativacao IS NULL OR YEAR(dat_desativacao) > 2010) THEN val_potenciaefetiva ELSE 0 END) AS potencia_2010,
    SUM(CASE WHEN YEAR(dat_entradaoperacao) <= 2020 AND (dat_desativacao IS NULL OR YEAR(dat_desativacao) > 2020) THEN val_potenciaefetiva ELSE 0 END) AS potencia_2020,
    SUM(CASE WHEN YEAR(dat_entradaoperacao) <= 2024 AND (dat_desativacao IS NULL OR YEAR(dat_desativacao) > 2024) THEN val_potenciaefetiva ELSE 0 END) AS potencia_2024
FROM tabela_silver
GROUP BY
    CASE
        WHEN nom_tipousina = 'BOMBEAMENTO' THEN 'HIDROELÉTRICA'
        WHEN nom_tipousina = 'EOLIELÉTRICA' THEN 'EÓLICA'
        ELSE nom_tipousina
    END
""")

# Converter o DataFrame do Spark para um DataFrame do Pandas
df_pandas = df_spark.toPandas()
```

▸ (2) Spark Jobs

▸ df_spark: pyspark.sql.dataframe.DataFrame = [nom_tipousina: string, potencia_2000: double ... 3 more fields]

11:43 PM (1s) 26

```
# Configurar os dados
years = ['2000', '2010', '2020', '2024']
df_pandas.set_index('nom_tipousina', inplace=True)
df_pandas = df_pandas.T

# Criar o gráfico
ax = df_pandas.plot(kind='bar', stacked=True, figsize=(14, 8), colormap='tab20')

# Adicionar rótulos e título
ax.set_xlabel('Ano')
ax.set_ylabel('Potência Instalada (MW)')
ax.set_title('Potência Instalada por Tipo de Usina em Diferentes Anos')
ax.legend(title='Tipo de Usina', bbox_to_anchor=(1.05, 1), loc='upper left')

# Adicionar os valores acima das barras
def add_values(bars):
    for bar in bars:
        for rect in bar:
            height = rect.get_height()
            if height > 0:
                ax.annotate(f'{height:.1f}',
                            xy=(rect.get_x() + rect.get_width() / 2, rect.get_y() + height / 2),
                            xytext=(0, 0),
                            textcoords="offset points",
                            ha='center', va='center')

# Chamar a função para cada pilha de barras
bars = [rects for rects in ax.containers]
add_values(bars)

# Ajustar layout
plt.tight_layout()

# Mostrar o gráfico
plt.show()
```



Em resumo, o objetivo do trabalho foi plenamente alcançado, com todas as questões propostas respondidas. As consultas realizadas permitiram uma série de análises sobre a matriz elétrica brasileira, gerando informações valiosas para profissionais do setor. Além disso, as ferramentas e conceitos aplicados neste estudo podem ser facilmente replicados em outros conjuntos de dados, demonstrando a relevância e a importância do aprendizado adquirido nesta disciplina.