CASE PROPOSTO PELA PIB COMO MÉTODO DE AVALIAÇÃO PARA O PROCESSO DE ESTÁGIO

CANDIDATO: RODRIGO SANTOS DE SOUZA

SEGUE ABAIXO A ESTRUTURA DA SOLUÇÃO DESENVOLVIDA

- Setup do PySpark e das variáveis ambiente;
- Carregamento dos CSVs;
- Cálculo do PIB por região;
- Cálculo da população por região;
- Razão entre PIB e população por região;
- Razão entre PIB e população do Brasil;
- União do PIB per capita das regiões com Brasil.

PASSO A PASSO ABAIXO:

- SETUP DO PYSPARK E DAS VARIÁVEIS AMBIENTE

```
# SETUP DO PYSPARK E DAS VARIÁVEIS AMBIENTE

!pip install pyspark==3.3.1

# SETUP DO PYSPARK E DAS VARIÁVEIS AMBIENTE

!apt-get install openjdk-8-jdk-headless -qq > /dev/null

# SETUP DO PYSPARK E DAS VARIÁVEIS AMBIENTE

!wget -q https://dlcdn.apache.org/spark/spark-3.3.1/spark-3.3.1-bin-hadoop3.tgz
```

```
# SETUP DO PYSPARK E DAS VARIÁVEIS AMBIENTE

!tar xf spark-3.3.1-bin-hadoop3.tgz

# SETUP DO PYSPARK E DAS VARIÁVEIS AMBIENTE

!pip install -q findspark

# SETUP DO PYSPARK E DAS VARIÁVEIS AMBIENTE

import os
os.environ["JAVA_HOME"] = "/usr/lib/jvm/java-8-openjdk-amd64"
os.environ["SPARK_HOME"] = "/content/spark-3.3.1-bin-hadoop3"

# SETUP DO PYSPARK E DAS VARIÁVEIS AMBIENTE

import findspark
findspark.init()

# SETUP DO PYSPARK E DAS VARIÁVEIS AMBIENTE

from pyspark.sql import functions as F
from pyspark.sql import SparkSession

spark = SparkSession.builder \
    .master('local[*]') \
    .appName('pib-case-rodrigo-souza') \
    .config('spark.ui.port', '4050') \
    .getOrCreate()
```

CARREGAMENTO DOS CSV'S

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

df_state = spark.read.option('delimiter',
';').format('csv').load('/content/drive/MyDrive/Tabelas/De_para_UF-
Tabela 1.csv', header='true')
df_pib = spark.read.option('delimiter',
';').format('csv').load('/content/drive/MyDrive/Tabelas/PIB_municipio-
Tabela 1.csv', header='true')
df_population = spark.read.option('delimiter',
';').format('csv').load('/content/drive/MyDrive/Tabelas/Populacao_Esta
do-Tabela 1.csv', header='true')
df_region = spark.read.option('delimiter',
';').format('csv').load('/content/drive/MyDrive/Tabelas/UF_Regiao-
Tabela 1.csv', header='true')
```

```
df_state = df_state.distinct()  # O Distrito Federal, por alguma razão desconhecida, estava duplicado. O método ".distinct()" remove essa repetição.
```

```
Mounted at /content/drive
```

CÁLCULO DO PIB POR REGIÃO

```
# O método ".substring()" retorna os dois primeiros caracteres da coluna "Cod_Identificacao" que serão utilizados posteriormente.
df_pib = df_pib.withColumn('Cod_Identificacao',
F.substring(F.col('Cod_Identificacao'), 0, 2))

# O valor do PIB foi convertido para inteiro através do método ".cast("int")".
df_pib = df_pib.withColumn('PIB', F.col('PIB').cast('int'))

# Junção do dataframe de pib com estado e região através das colunas "Cod_Identificacao" e "Estado", respectivamente.
# pib -> estado (O método ".join()" ocorreu através da coluna "Cod_Identificacao").
# pib -> estado -> região (O método ".join()" ocorreu através da coluna "Estado").
# A coluna "Granularidade" foi renomeada para "Estado" a fim de fazer mais sentido semanticamente.

df_pib_per_region = df_pib.join(df_state, how='inner',
on='Cod_Identificacao').withColumnRenamed('Granularidade','Estado')\
                    .join(df_region, how='inner',
on='Estado').groupBy('Regiao').agg(F.sum('PIB').alias('PIB'))

df_pib_per_region.show() # O método ".show()" é utilizado para imprimir o resultado em tela.
```

```
+------+----------+
|Regiao|       PIB|
+------+----------+
|    NE| 522769299|
|     N| 207093641|
|     S| 620180421|
|    SE|2180987793|
|    CO| 354815818|
+------+----------+
```

CÁLCULO DA POPULAÇÃO POR REGIÃO

```
# O método ".filter()" foi utilizado objetivando filtrar apenas as
linhas que possuem o conteúdo "Total" na coluna "fx_idade"\ além de
remover as linhas que fazem referência a "Brasil" na coluna
"Granularidade".
# A coluna "Granularidade" foi renomeada para "Estado" a fim de
possibilitar para fazer o join com região

df_pop_per_state = df_population.filter((F.col('fx_idade') == 'Total')
& (F.col('Granularidade') != 'Brasil'))\
                    .withColumnRenamed('Granularidade',
'Estado').withColumn('Populacao', F.col('Populacao').cast('int'))

# Junção utilizando o método ".join()" das populações por região com
as regiões.
df_pop_per_region = df_pop_per_state.join(df_region, how='inner',
on='Estado').groupBy('Regiao').agg(F.sum('Populacao').alias('Populacao
'))

df_pop_per_region.show() # O método ".show()" é utilizado para
imprimir o resultado em tela.
```

```
+------+---------+
|Regiao|Populacao|
+------+---------+
|    NE| 53081950|
|     N| 15864454|
|     S| 27386891|
|    SE| 80364410|
|    CO| 14058094|
+------+---------+
```

RAZÃO ENTRE PIB E POPULAÇÃO POR REGIÃO

```
# Junção utilizando o método ".join()" das populações por região com
os PIBs por região.
df_regions = df_pop_per_region.join(df_pib_per_region, on='Regiao',
how='inner').select('Regiao',
F.col('PIB')/F.col('Populacao')).orderBy('Regiao')

df_regions.show() # O método ".show()" é utilizado para imprimir o
resultado em tela.
```

```
+------+------------------+
|Regiao| (PIB / Populacao)|
+------+------------------+
|    CO|25.239254908951384|
|     N|13.053940652480067|
|    NE| 9.848343909747099|
|     S|22.645156071202095|
|    SE|   27.138727118136|
+------+------------------+
```

RAZÃO ENTRE PIB E POPULAÇÃO DO BRASIL

```
# O método ".sum()" realiza a soma de todos os PIBs.
# O método ".select()" retorna a soma de todos os PIBs.
df_pib_total_brasil = df_pib.select(F.sum('PIB').alias('PIB'))

# O método ".filter()" foi utilizado objetivando filtrar apenas as
linhas que possuem o conteúdo "Total" na coluna "fx_idade" e as linhas
que fazem referência a "Brasil" na coluna "Granularidade".
df_pop_brasil = df_population.filter((F.col('fx_idade') == 'Total') &
(F.col('Granularidade') ==
'Brasil')).withColumnRenamed('Granularidade', 'Regiao')

# cross join é um produto cartesiano (todas as linhas com todas as
colunas). Como só existe uma linha em cada dataframe, simplesmente as
junta
df_brasil =
df_pop_brasil.crossJoin(df_pib_total_brasil).select(F.col('Regiao'),
F.col('PIB')/F.col('Populacao'))

df_brasil.show() # O método ".show()" é utilizado para imprimir o
resultado em tela.
```

```
+------+------------------+
|Regiao| (PIB / Populacao)|
+------+------------------+
|Brasil|20.370793403769603|
+------+------------------+
```

UNIÃO DE PIB PER CAPITA DAS REGIÕES COM BRASIL

```
# O método ".unionAll()" realiza a junção dos dois dataframes de
interesse que são o "df_regions" com o "df_brasil".
# Para o método acima funcionar sem problemas é necessário que as
colunas de ambos os dataframes possuam os mesmos nomes.
# O método ".withColumnRenamed()" foi utilizado a fim de renomear a
coluna "round((PIB / Populacao)" com um novo nome que é
"PIB_PER_CAPITA"
# O método ".withColumnRenamed()" foi utilizado, novamente, a fim de
renomear a coluna "Regiao" com um novo nome que é "REGIÃO"

finalSolution = df_regions.unionAll(df_brasil).select(F.col('Regiao'),
F.round(F.col('(PIB / Populacao)'), 2))\
     .withColumnRenamed('round((PIB / Populacao), 2)',
'PIB_PER_CAPITA').withColumnRenamed('Regiao', 'REGIÃO')

finalSolution.show() # O método ".show()" é utilizado para imprimir o
resultado em tela.
```

```
+------+--------------+
|REGIÃO|PIB_PER_CAPITA|
+------+--------------+
|    CO|         25.24|
|     N|         13.05|
|    NE|          9.85|
|     S|         22.65|
|    SE|         27.14|
|Brasil|         20.37|
+------+--------------+
```