ACESSANDO DADOS DE ARQUIVOS PARQUET DIRETAMENTO DO S3 BUCKET COM DATABRICKS, SPARK, PYTHON E SQL NA ENGENHARIA DE DADOS

JAIRO BERNARDES DA SILVA JÚNIOR
www.linkedin.com/in/jairobernardesjunior

Pós-Graduado em Big Data e Ciência de Dados
Pós-Graduado em Engenharia de Sistemas
Pós-Graduado em Gestão de TI

Graduado em Física

ACESSANDO DADOS DE ARQUIVOS PARQUET DIRETAMENTE DO S3 BUCKET COM DATABRICKS, SPARK, PYTHON E SQL NA ENGENHARIA DE DADOS

DESCRIÇÃO:

Tanto o projeto bigDVarejoPMC quanto o bigDGeneral_IPCA geraram arquivos parquet que foram armazenados em s3 buckets na AWS Cloud, sendo catalogados pelo Glue Data Catalog e disponibilizados para consultas através do AWS Athena e AWS Quicksight.

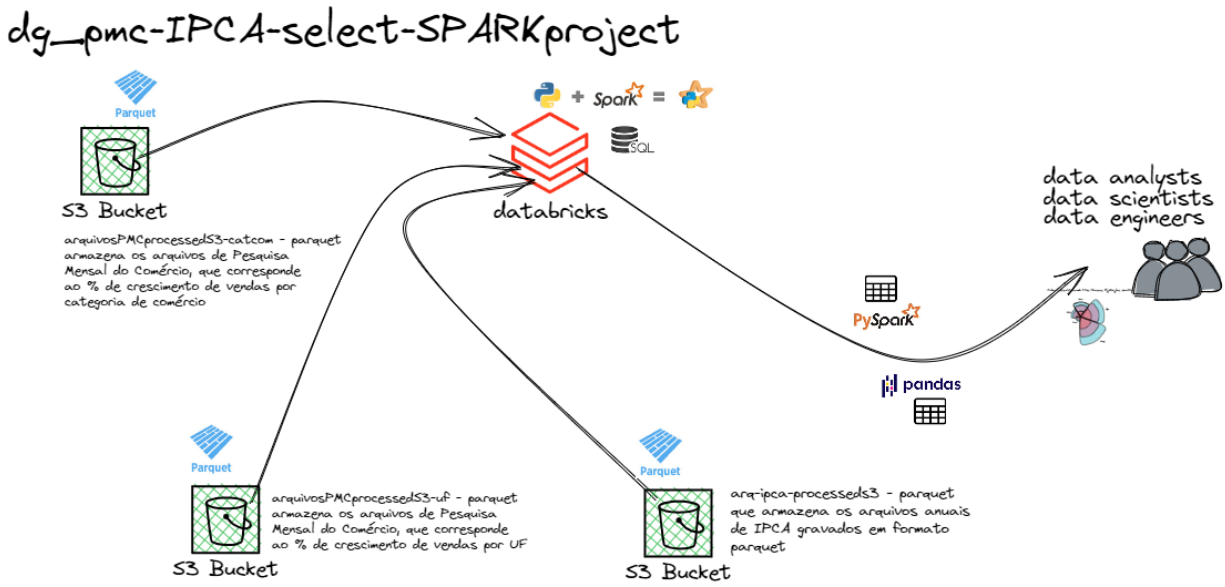
Agora surgiu a necessidade de se usar o Databricks com linguagem python e SQL, fazendo uso dos recursos do Spark utilizando o pyspark, acessando os dados que estão nos arquivos parquet gravados anteriormente e disponibilizados em buckets s3, sendo possível, o acesso direto a esses arquivos e carga em spark dataframes e pandas dataframes para posteriormente serem utilizados para geração de gráficos estatísticos e insights com o pyspark e pandas no código python.

Motivação:

Fazer o acesso dos arquivos parquet de pmc-uf, pmc-catcom e ipca diretamente dos s3 buckets da AWS que se encontram na nuvem, para geração de insights dos dados de vendas de varejo e ipca, sem a necessidade de usar o data catalog do Glue para a geração de metadados e acesso pelo athena e quickinsight, permitindo que o custo na geração de insights seja mais baixo, podendo ser feito tanto com o databricks na AWS quanto fora dela.

Aplicação Prática:

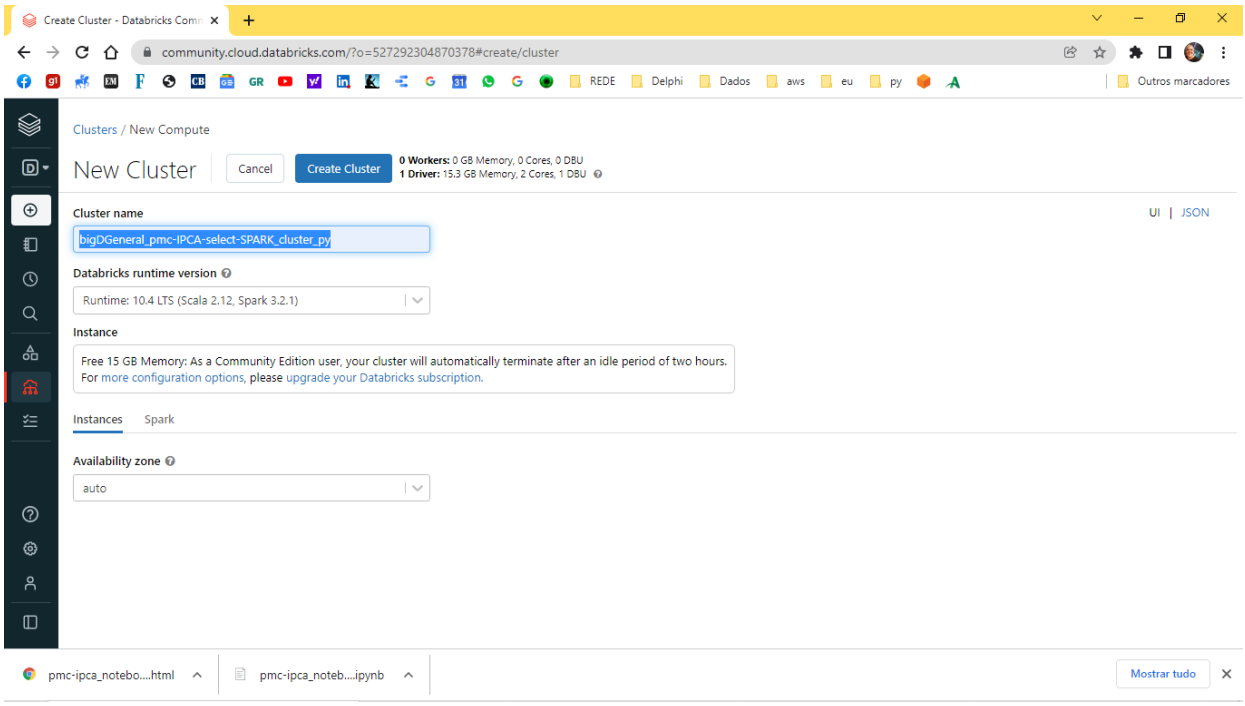
Gerar insights de crescimento de vendas de varejo e ipca fazendo o acesso diretamente no s3 bucket diminuindo custo, acessando os dados fazendo a conexão do armazenamento s3 da AWS diretamente pelo databricks de uma máquina local, usando python.



CRIAR UM CLUSTER NO DATABRICKS:

Foram usados os dados abaixo, conforme o projeto:

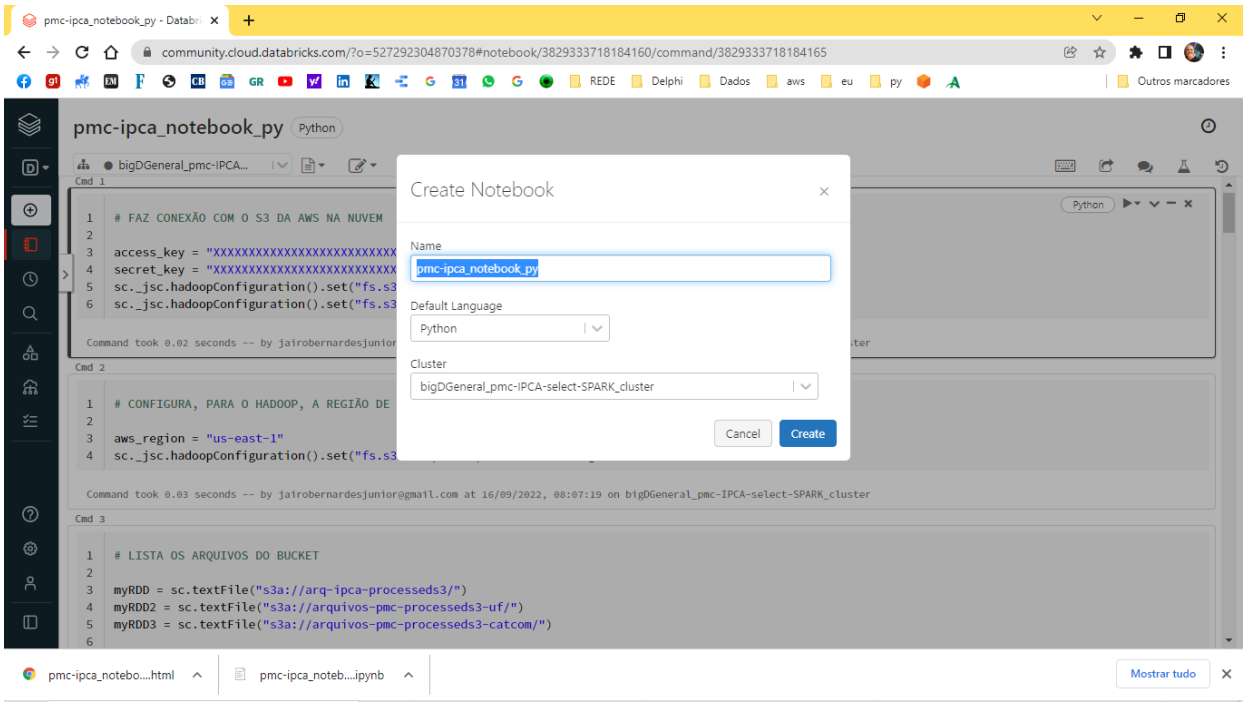
Cluster name = bigDGeneral_pmc-IPCA-select-SPARK_cluster_py
Databricks runtime version = Runtime: 10,4 LTS (Scala 2.12 Spark 3.2.1)



CRIAR UM NOTEBOOK NO DATABRICKS:

Foram usados os dados abaixo, conforme o projeto:

Name = pmc-ipca_notebook_py
Default Language = Python
Cluster = bigDGeneral_pmc-IPCA-select-SPARK_cluster_py (criado anteriormente)



ESCREVER O CÓDIGO NO NOTEBOOK CRIADO NO DATABRICKS:

O código está nos endereços:
https://github.com/jairo2016/bigDGeneral_pmc-IPCA-select-SPARKproject/tree/main/DATABRICKS_PY

https://databricks-prod-cloudfront.cloud.databricks.com/public/4027ec902e239c93eaaa8714f173bcfc/527292304870378/3829333718184160/4160994562188282/latest.html

CÓDIGO DATABRICKS NOTEBOOK:

```
# FAZ CONEXÃO COM O S3 DA AWS NA NUVEM

access_key = "xxxxxxxxxxxxxxxxxxxxxxxxxxxx"
secret_key = "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"
sc._jsc.hadoopConfiguration().set("fs.s3a.access.key", access_key)
sc._jsc.hadoopConfiguration().set("fs.s3a.secret.key", secret_key)
```

```
# CONFIGURA, PARA O HADOOP, A REGIÃO DE ACESSO PARA O S3 DA AWS NA NUVEM

aws_region = "us-east-1"
sc._jsc.hadoopConfiguration().set("fs.s3a.endpoint", "s3." + aws_region +
".amazonaws.com")
```

```
# LISTA OS ARQUIVOS DO BUCKET

myRDD = sc.textFile("s3a://arq-ipca-processeds3/")
myRDD2 = sc.textFile("s3a://arquivos-pmc-processeds3-uf/")
myRDD3 = sc.textFile("s3a://arquivos-pmc-processeds3-catcom/")

myRDD.count()
dbutils.fs.ls("s3a://arq-ipca-processeds3/")
```

```
# LISTA OS ARQUIVOS DO BUCKET

myRDD2.count()
dbutils.fs.ls("s3a://arquivos-pmc-processeds3-uf/")
```

```
# LISTA OS ARQUIVOS DO BUCKET

myRDD3.count()
dbutils.fs.ls("s3a://arquivos-pmc-processeds3-catcom/")
```

```
# LENDO O SCHEMA DOS ARQUIVOS PARQUET DE CADA BUCKET

spark.read.format("parquet").load("s3a://arq-ipca-processeds3/")
spark.read.format("parquet").load("s3a://arquivos-pmc-processeds3-uf/")
spark.read.format("parquet").load("s3a://arquivos-pmc-processeds3-catcom/")
```

```
# EXTRAINDO OS DADOS DE IPCA DOS ARQUIVOS PARQUET PARA O DATAFRAME PYSPARK

dfipca = spark.sql("SELECT ano, mes, avg(perc) as perc \
                           FROM parquet.`s3a://arq-ipca-processeds3/` \
                           where mes < 13 \
                           group by ano, mes order by ano, mes")

from pyspark.sql.functions import lit
#dfipca = dfipca.withColumn("id", lit("ipca"))
dfipca.show()
```

```
# TOTALIZANDO O PERCENTUAL DO IPCA POR ANO E GERANDO O GRÁFICO COM DISPLAY

dfipca_ano = dfipca.groupby('ano').sum('perc').sort('ano', ascending=False)

from pyspark.sql.functions import round, col
dfipca_ano = dfipca_ano.select('ano', round(col('sum(perc)'), 2).alias('perc'))

dfipca_ano.select('*').show()
display(dfipca_ano)
```

```
# CONVERTENDO DATAFRAME SPARK PARA DATAFRAM PANDAS GERANDO O GRÁFICO COM DISPLAY

dfPandas = dfipca_ano.toPandas()
display(dfPandas)
```

```
'''
EXTRAINDO OS DADOS DE PMC(pesquisa mensal de comércio por UF - ibge - percentual
de crescimento das vendas no varejo),
DOS ARQUIVOS PARQUET PARA O DATAFRAME PYSPARK
'''

dfpmcUF = spark.sql("SELECT uf, ano, mes, avg(m3mensal) as perc \
                           FROM parquet.`s3a://arquivos-pmc-processeds3-uf/` \
                           where mes < 13 \
                           group by uf, ano, mes order by uf, ano, mes")

from pyspark.sql.functions import lit
#dfpmcUF = dfpmcUF.withColumn("id", lit("uf"))
dfpmcUF.show(99999)
```

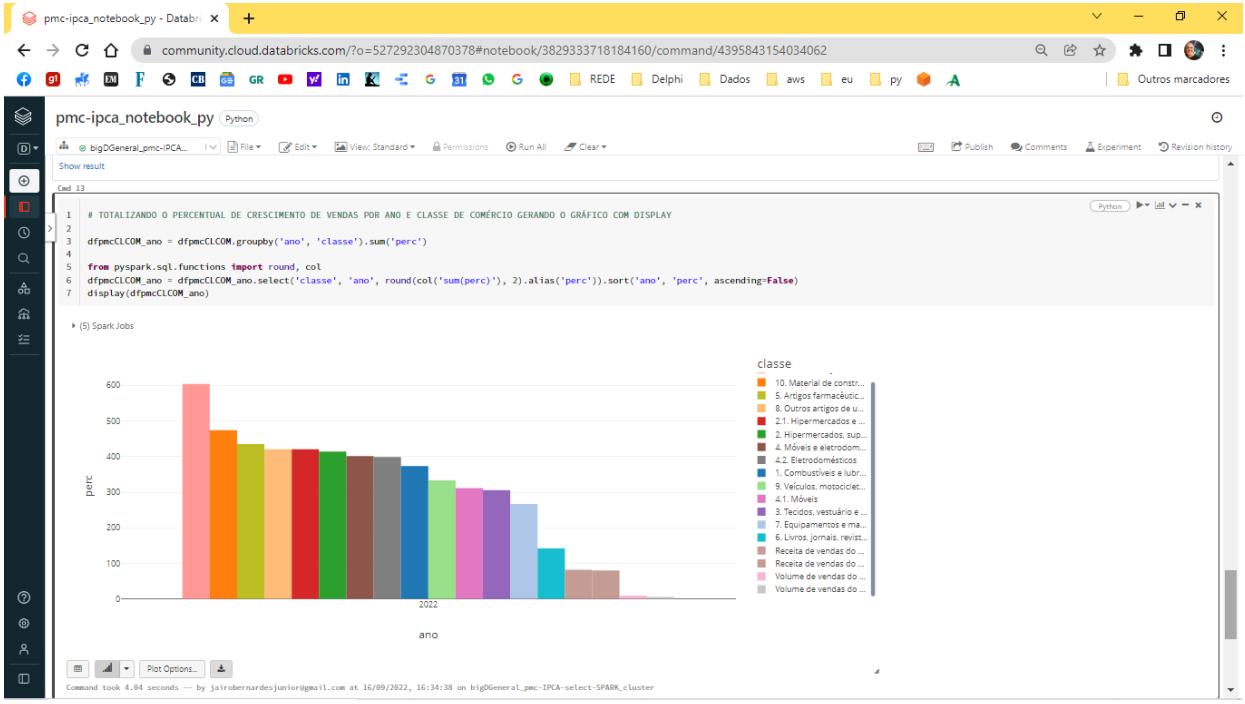
```
# TOTALIZANDO O PERCENTUAL DE CRESCIMENTO DE VENDAS POR ANO E UF GERANDO O
GRÁFICO COM DISPLAY

dfpmcUF_ano = dfpmcUF.groupby('ano', 'uf').sum('perc')

from pyspark.sql.functions import round, col
dfpmcUF_ano = dfpmcUF_ano.select('uf', 'ano', round(col('sum(perc)'),
2).alias('perc')).sort('ano', 'perc', ascending=False)
display(dfpmcUF_ano)
```

ACESSANDO DADOS DE ARQUIVOS PARQUET DIRETAMENTE DO S3 BUCKET COM DATABRICKS, SPARK, PYTHON E SQL NA ENGENHARIA DE DADOS

```
'''
EXTRAINDO OS DADOS DE PMC(pesquisa mensal de comércio por categoria de comércio
- ibge - percentual de crescimento das vendas no varejo),
DOS ARQUIVOS PARQUET PARA O DATAFRAME PYSPARK
'''

dfpmcCLCOM = spark.sql("SELECT classe_comercio as classe, ano, mes,
avg(m3mensal) as perc \
                        FROM parquet.`s3a://arquivos-pmc-processeds3-
catcom/` \
                        where mes < 13 and m3mensal > 0 \
                        group by classe_comercio, ano, mes order by
classe_comercio, ano, mes")

from pyspark.sql.functions import lit
#dfpmcCLCOM = dfpmcCLCOM.withColumn("id", lit("clcom"))
dfpmcCLCOM.show(99999)
```

```
# TOTALIZANDO O PERCENTUAL DE CRESCIMENTO DE VENDAS POR ANO E CLASSE DE COMÉRCIO
GERANDO O GRÁFICO COM DISPLAY

dfpmcCLCOM_ano = dfpmcCLCOM.groupby('ano', 'classe').sum('perc')

from pyspark.sql.functions import round, col
dfpmcCLCOM_ano = dfpmcCLCOM_ano.select('classe', 'ano', round(col('sum(perc)'),
2).alias('perc')).sort('ano', 'perc', ascending=False)
display(dfpmcCLCOM_ano)
```

OUTPUTS:



ACESSANDO DADOS DE ARQUIVOS PARQUET DIRETAMENTE DO S3 BUCKET COM DATABRICKS, SPARK, PYTHON E SQL NA ENGENHARIA DE DADOS