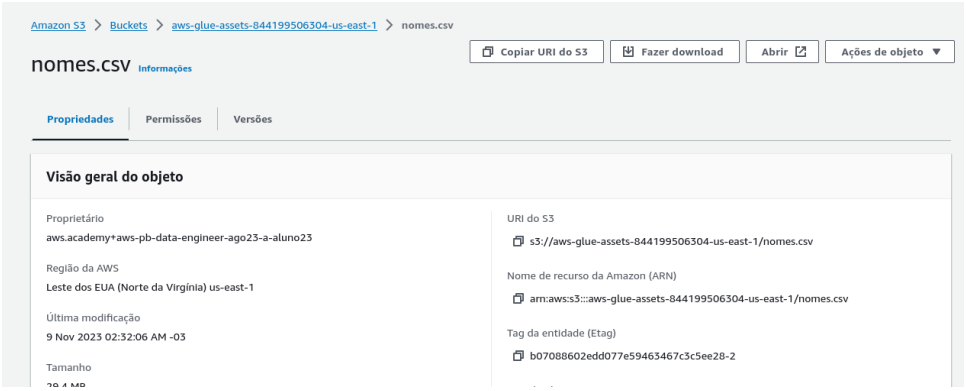
LabAwsGlue

Objetivo: construir um job Glue.

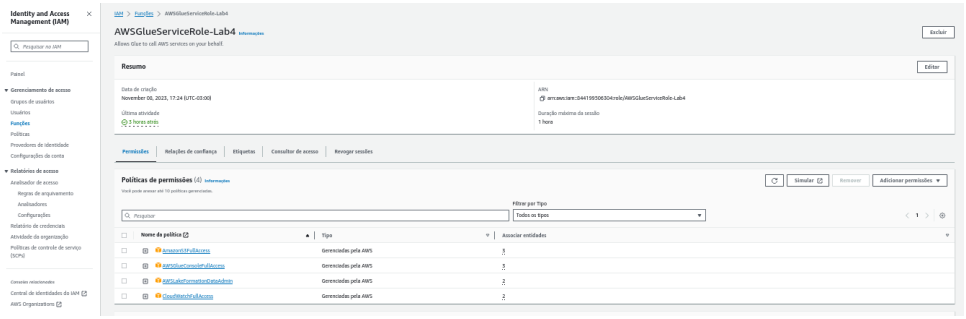
Atenção: Esse material segue as orientações da documentação da documentação “LabAWSGlueSteps” presente nesse diretorio.

- Os dados de origem foram exportados para dentro do S3:

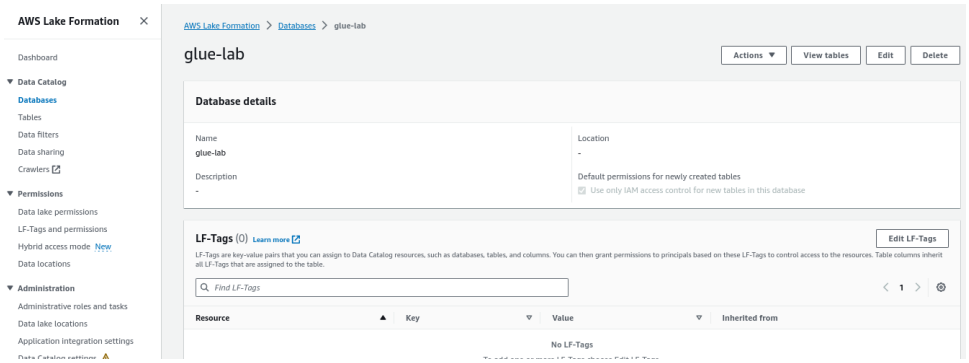


- Para a configuração da conta para ultilizar o AWS glue foi necessário criar um usuário IAM novo:

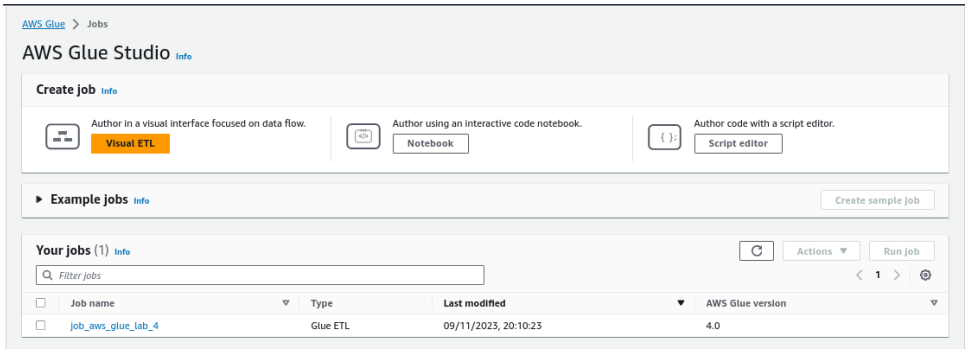
- A IAM Role para os jobs do aws glue foi criada:

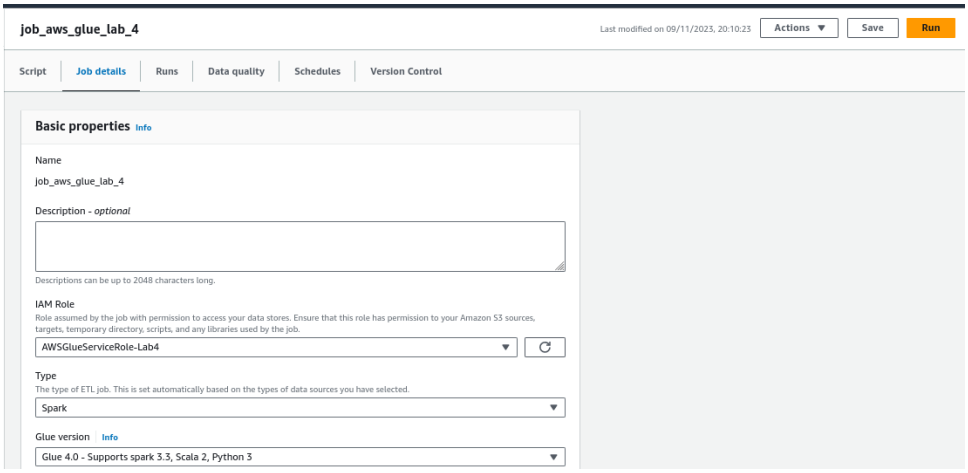


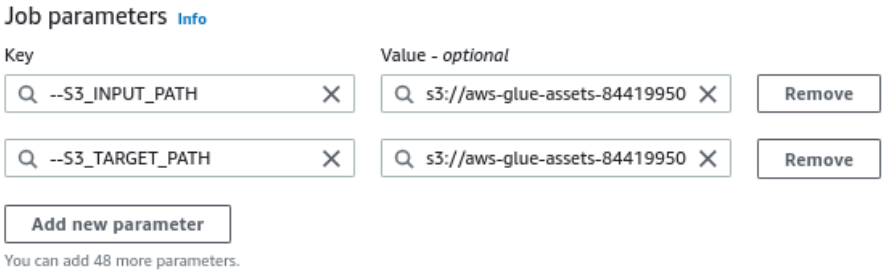
- A configuração das permissões no AWS Lake Formation foi feita:



- O novo job no AWS Glue foi criado:

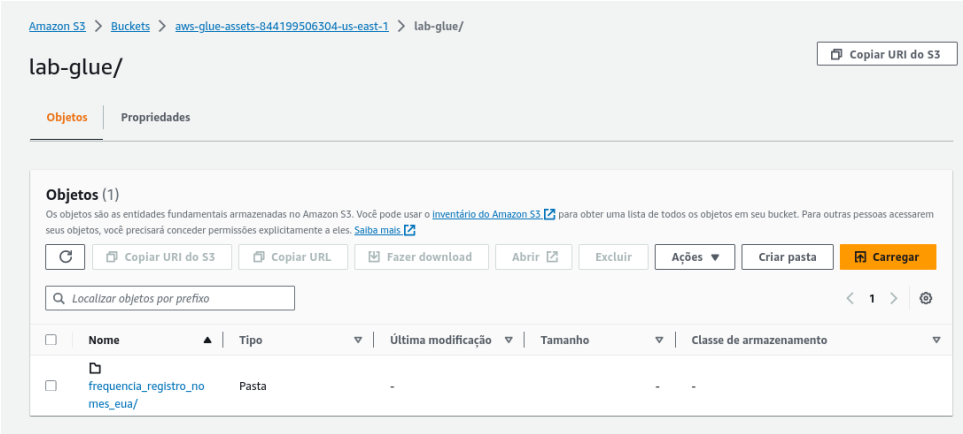






Atenção: os caminhos de entrada e saida já foram configurados como pode ser visto na imagem acima.

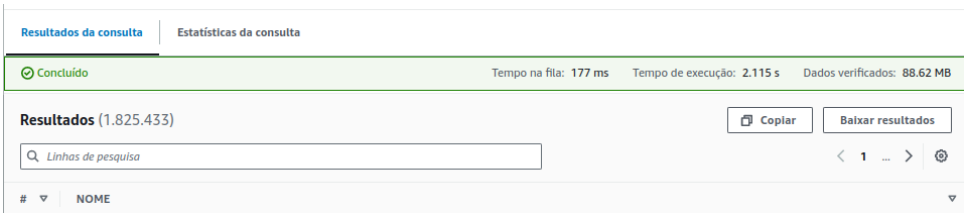
- O caminho de destino foi configurado:



Agora vamos construir um job Glue nos moldes dos exemplos anteriores. Seguem os passos para desenvolver:

- Ler o arquivo nomes.csv no S3 (lembre-se de realizar upload do arquivo antes).
- Imprima o schema do dataframe gerado no passo anterior.

- Escrever o código necessário para alterar a caixa dos valores da coluna nome para
- MAIÚSCULO.

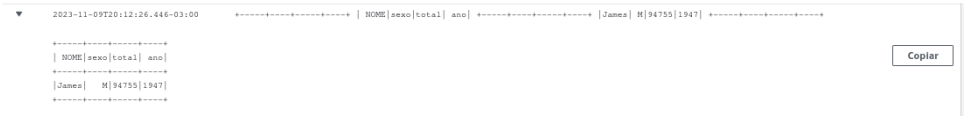


- Imprimir a contagem de linhas presentes no dataframe.

- Imprimir a contagem de nomes, agrupando os dados do dataframe pelas colunas ano e sexo.
- Ordene os dados de modo que o ano mais recente apareça como primeiro registro do dataframe.

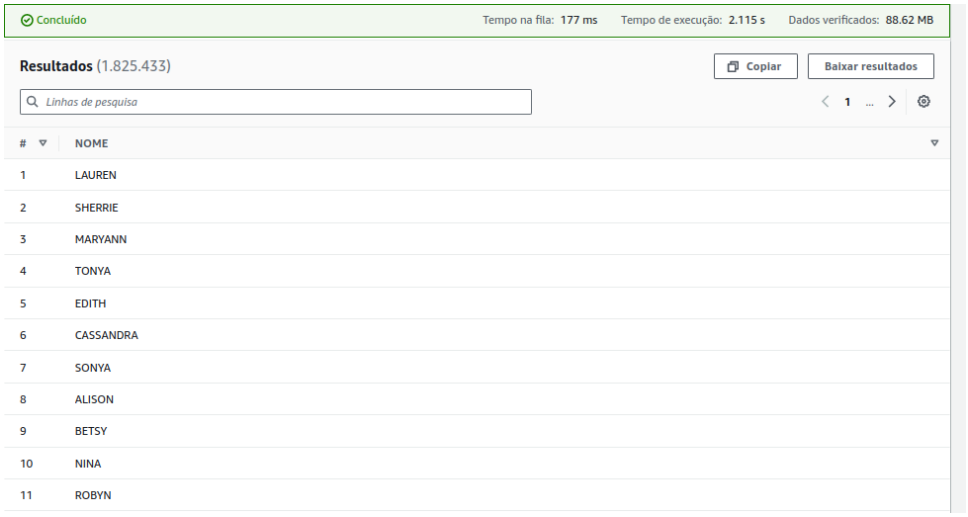
- Apresentar qual foi o nome feminino com mais registros e em que ano ocorreu.

- Apresentar qual foi o nome masculino com mais registros e em que ano ocorreu.



- Apresentar o total de registros (masculinos e femininos) para cada ano presente no dataframe.
- Considere apenas as primeiras 10 linhas, ordenadas pelo ano, de forma crescente.

- Escrever o conteúdo do dataframe com os valores de nome em maiúsculo no S3.



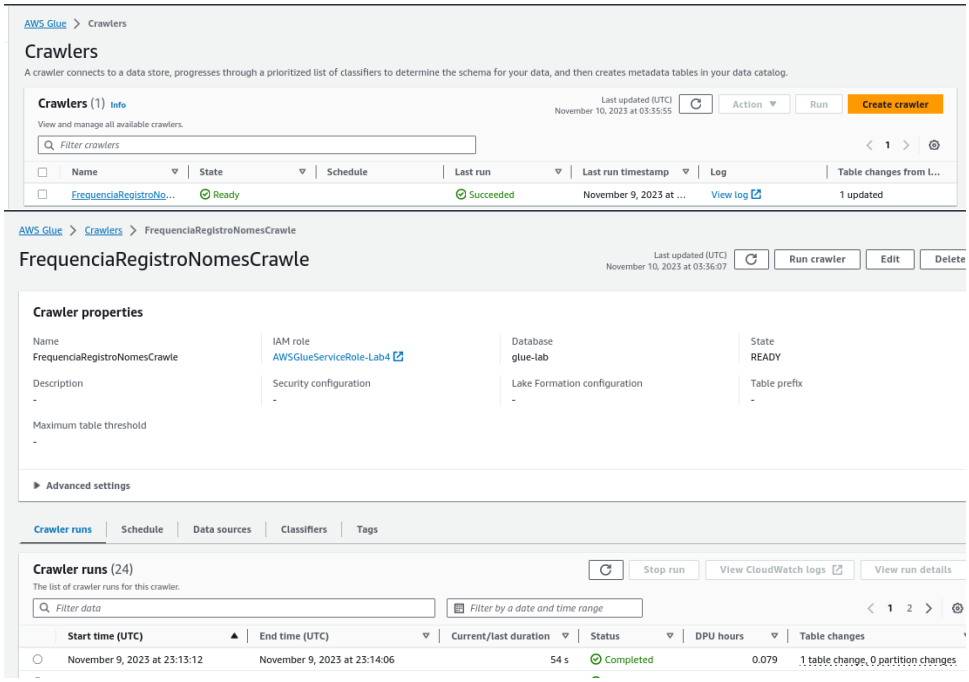
- Código geral usado dentro do ambiente AWS Glue:

```
import sys
from awsglue.transforms import *
from awsglue.utils import getResolvedOptions
from pyspark.context import SparkContext
from awsglue.context import GlueContext
from awsglue.job import Job
from awsglue.dynamicframe import DynamicFrame
from pyspark.sql.functions import count, col, sum, max, upper
from pyspark.sql.types import IntegerType

args=getResolvedOptions(sys.argv['JOB_NAME','S3_INPUT_PATH','S3_TARGET_PATH'])
sc = SparkContext()
glueContext = GlueContext(sc)
spark = glueContext.spark_session
job = Job(glueContext)
job.init(args['JOB_NAME'], args)

source_file = args['S3_INPUT_PATH']
target_path = args['S3_TARGET_PATH']
```

```
dynamicFrame = glueContext.create_dynamic_frame.from_options(
    connection_type="s3",
    connection_options={"paths": [source_file]},
    format="csv",
    format_options={
        "withHeader": True,
        # "optimizePerformance": True,
    },
)

spark_df = dynamicFrame.toDF()

spark_df = spark_df.withColumnRenamed("nome", "NOME")

spark_df = spark_df.withColumn("ano", spark_df["ano"].cast(IntegerType()))
spark_df = spark_df.withColumn("total", spark_df["total"].cast(IntegerType()))

dynamicFrame.printSchema()

row_count = spark_df.count()
print("Numero de linhas no DataFrame: ", row_count)

result=spark_df.groupBy("ano",
"sexo").agg(count("NOME").alias("count")).orderBy(col("ano").desc())
result.show()

spark_df = spark_df.orderBy(col("ano").desc())

f_data = spark_df.filter(col("sexo") == "F").orderBy(col("total").desc()).limit(1)
f_data.show()
```

```
m_data = spark_df.filter(col("sexo") == "M").orderBy(col("total").desc()).limit(1)
m_data.show()

agg_result = spark_df.groupBy("ano").agg(sum("total").alias("registros")).orderBy("ano")
agg_result.show(10)

spark_df = spark_df.withColumn("NOME", upper(spark_df["NOME"]))

dynamicFrame = DynamicFrame.fromDF(spark_df.repartition(1), glueContext, "dynamic_frame")

glueContext.write_dynamic_frame.from_options(
   frame=dynamicFrame,
   connection_type="s3",
   connection_options={"path": target_path},
   format="json",
)

job.commit()
```

- Criando crawler:


AWS Glue > Crawlers
Crawlers
A crawler connects to a data store, progresses through a prioritized list of classifiers to determine the schema for your data, and then creates metadata tables in your data catalog.
Crawlers (1) Info
Last updated (UTC) November 10, 2023 at 03:35:55
Action
Run
Create crawler
View and manage all available crawlers.
Filter crawlers
1
Name
State
Schedule
Last run
Last run timestamp
Log
Table changes from l...
FrequenciaRegistroNo...
Ready
Succeeded
November 9, 2023 at ...
View log
1 updated

AWS Glue > Crawlers > FrequenciaRegistroNomesCrawle
FrequenciaRegistroNomesCrawle
Last updated (UTC) November 10, 2023 at 03:36:07
Run crawler
Edit
Delete
Crawler properties
Name
FrequenciaRegistroNomesCrawle
IAM role
AWSGlueServiceRole-Lab4
Database
glue-lab
State
READY
Description
-
Security configuration
-
Lake Formation configuration
-
Table prefix
-
Maximum table threshold
-
Advanced settings
Crawler runs
Schedule
Data sources
Classifiers
Tags
Crawler runs (24)
The list of crawler runs for this crawler.
Stop run
View CloudWatch logs
View run details
Filter data
Filter by a date and time range
1
2
Start time (UTC)
End time (UTC)
Current/last duration
Status
DPU hours
Table changes
November 9, 2023 at 23:13:12
November 9, 2023 at 23:14:06
54 s
Completed
0.079
1 table change, 0 partition changes