***Manual de Pyspark***

O Pyspark é uma API do Python para processamento em big data, e todas as funções que você encontra no SQL estão disponíveis no Pyspark, para que você execute operações complexas de manipulação de dados.

Lembrando que também é possível usar apenas o SQL dentro do Pyspark, mas é altamente recomendado usar apenas a sintaxe do Pyspark, dado que esse tem melhor performance.

***Leitura de dados***

Para ler a tabela no Pyspark é necessário executar os comandos abaixo, sempre passando como parâmetro o ***nome do banco de dados, nome da tabela, e a partição selecionada.***

***Leitura de dados puxando a última partição sem biblioteca DataUtils***

Código para puxar última partição da tabela sem o DataUtils

***Transformando dados***

Para transformar os dados usando Pyspark, é necessário usar a estrutura conforme abaixo. Nesse código todos os elementos são contemplados: ***Case when, Cast, DateDiff, Date\_format, Translate, Truncate, Trim, FillNa, DropNa, orderBy, DropDuplicates entre outros***:

# Preenche valores nulos e remove linhas com nulos

df1\_cleaned = df1.fillna({'valor': '0', 'data': '2023-10-01'}).dropna()

# Realiza o SELECT com as transformações

result = df1\_cleaned.select( col("id").cast("bigint").alias("id\_bigint")

, trim(col("valor")).cast("decimal(10,2)").alias("valor\_decimal")

, when(col("descricao") == 'valor1', 'Descrição 1')

.when(col("descricao") == 'valor2', 'Descrição 2') .otherwise('Outro').alias("descricao\_transformada")

, translate(col("codigo"), '+', '').alias("codigo\_sem\_mais")

, date\_format(col("data"), 'yyyy-MM-dd').alias("data\_formatada")

, datediff(current\_date(), col("data")).alias("dias\_desde\_data") ) # Filtra os resultados onde

valor\_decimal é maior que 20 e ordena

filtered\_result = result.filter(col("valor\_decimal") > 20).orderBy(col("valor\_decimal")) # Faz um join com o segundo DataFrame

df2\_cleaned = df2.fillna({'valor': '0'})

joined\_result = filtered\_result.join(df2\_cleaned, "id\_bigint", "inner")

# Faz um union com outro

DataFrame final\_result = filtered\_result.union(df2\_cleaned.select("id", "valor", "descricao", "codigo", "data"))

***Salvando Tabela***

***Apagando Tabela e Pastinha do S3 pelo Glue***

Para apagar a tabela do catálogo de dados é necessário usar o comando abaixo, no entanto, importante lembrar que caso reutilize o espaço do S3 é necessário apagar o caminho também

# Apagando o s3 caminho

s3 = boto3.client('s3') objects = s3.list\_objects\_v2(Bucket=bucket\_name, Prefix=folder\_name) if 'Contents' in objects: keys = [{'Key': obj['Key']} for obj in objects['Contents']] s3.delete\_objects(Bucket=bucket\_name, Delete={'Objects': keys})

glue\_client = boto3.client('glue') try: # Apaga a tabela response = glue\_client.delete\_table( DatabaseName=database\_name, Name=table\_name ) print(f'Tabela {table\_name} apagada com sucesso do banco de dados {database\_name}.') except Exception as e: print(f'Erro ao apagar a tabela: {e}')