pipeline

July 12, 2024

[3]:
```
import os
import requests
import tempfile
import pyspark.sql.functions as F
import pyspark.sql.functions as func
from pyspark.sql.types import *
from pyspark.sql import SparkSession
from pyspark.sql import Row
from pyspark.ml.feature import Bucketizer
from pyspark.sql.functions import col
from pyspark.sql.window import Window
from google.cloud import storage
from google.cloud import bigquery
import numpy as np
```

[4]:
```
# Definição do projeto na GCP
project_id = "pucmvp"

# Definindo a sessão Spark
spark = SparkSession.builder.appName("AnaliseCNH").getOrCreate()
```

1. Raw Ingestion - Camada Bronze / Upload do arquivo realizado manualmente para o Bucket

[5]:
```
#Lendo o dataset do Cloud Storage

#dframe = spark.read.csv("gs://pucmvpbucket/extracao/tabelao.csv",header=True,␣
↪inferSchema=True, sep=';')

dframe = spark.read.csv("gs://pucmvpbucket/extracao/dados_emissao_mvp_inicial.
↪csv",header=True, inferSchema=True, sep=',')
```

[6]:
```
# Printando schema do DataSet
dframe.printSchema()
```

```
root
 |-- SIT_INT_ID_SITE: integer (nullable = true)
```

```
 |-- CAR_INT_CD_STATUS: integer (nullable = true)
 |-- CAR_STR_NR_CPF: long (nullable = true)
 |-- CAR_STR_DS_NOMELINHA1: string (nullable = true)
 |-- CAR_DAT_DT_NASCIMENTO: string (nullable = true)
 |-- CAR_STR_ID_CATEGORIA: string (nullable = true)
 |-- CAR_STR_ID_TIPO: string (nullable = true)
 |-- CAR_STR_DS_UF: string (nullable = true)
 |-- CAR_DAT_DT_EMISSAO: string (nullable = true)
 |-- CAR_DAT_DT_PRIMEIRAHABILITACAO: string (nullable = true)
 |-- CAR_DAT_DT_VALIDADE: string (nullable = true)
 |-- CAR_DBL_NR_REGISTRO: double (nullable = true)
 |-- CAR_STR_DS_OBSERVACAOLINHA1: string (nullable = true)
 |-- CAR_STR_DS_OBSERVACAOLINHA2: string (nullable = true)
```

[7]:
```
# Imprimindo 2 linhas do Dataset Carregado
dframe.show(2)
```

```
+--------------+-----------------+-------------+--------------------+--------
-------------+--------------------+--------------+-------------+--------------
-----+------------------------------+--------------------+------------------+--
------------------------+---------------------------+
|SIT_INT_ID_SITE|CAR_INT_CD_STATUS|CAR_STR_NR_CPF|CAR_STR_DS_NOMELINHA1|CAR_DAT_
DT_NASCIMENTO|CAR_STR_ID_CATEGORIA|CAR_STR_ID_TIPO|CAR_STR_DS_UF|
CAR_DAT_DT_EMISSAO|CAR_DAT_DT_PRIMEIRAHABILITACAO| CAR_DAT_DT_VALIDADE|CAR_DBL_N
R_REGISTRO|CAR_STR_DS_OBSERVACAOLINHA1|CAR_STR_DS_OBSERVACAOLINHA2|
+--------------+-----------------+-------------+--------------------+--------
-------------+--------------------+--------------+-------------+--------------
-----+------------------------------+--------------------+------------------+--
------------------------+---------------------------+
|              8|               22|   41815564334| ANTONILSON FRAZAO…|
1966-11-24 00:00:…|                B   |              C|
MA|2024-07-08 00:00:…|         2007-07-10 00:00:…|2029-06-24 00:00:…|
4.13796069E9|                          C |                          D |
|              8|               22|   61983660302| LUCIANO SILVA DOS…|
1999-04-23 00:00:…|                B   |              P|
MA|2024-07-08 00:00:…|         2024-07-08 00:00:…|2025-07-07 00:00:…|
8.684790978E9|                          D |                          E |
+--------------+-----------------+-------------+--------------------+--------
-------------+--------------------+--------------+-------------+--------------
-----+------------------------------+--------------------+------------------+--
------------------------+---------------------------+
only showing top 2 rows
```

[8]:
```
# Quantidade de Registros e Colunas do Dataset
print((dframe.count(), len(dframe.columns)))
```

```
[Stage 3:>                                                          (0 + 1) / 1]
```

```
(10000, 14)
```

2. Filtered, Cleaned, Validated - Camada Silver

[9]:
```
# Remover linhas/dados CPF duplicados
dframe = dframe.dropDuplicates().na.fill(2)
```

[10]:
```
# Quantidade de Registros e Colunas do Dataset após apagar linhas/registros CPF␣
 ↪Duplicados
print((dframe.count(), len(dframe.columns)))
```

```
[Stage 6:>                                                          (0 + 1) / 1]
```

```
(7799, 14)
```

[11]:
```
from pyspark.sql.functions import col, count

# Mapeamento dos dados:
# Conversão de datatype >> dframe.withColumn("CAR_DBL_NR_REGISTRO",␣
 ↪col("CAR_DBL_NR_REGISTRO").cast("integer"))
# Conversão de nomeclatura >> dframe.withColumnRenamed("CAR_INT_CD_STATUS",␣
 ↪"CD_STATUS")
# Converter tipo de coluna double para integer / string para integer e datas␣
 ↪para Timestamp

dframe_filtrado = \
dframe.withColumn("CAR_DBL_NR_REGISTRO", col("CAR_DBL_NR_REGISTRO").
 ↪cast("integer")) \
.withColumnRenamed("CAR_DBL_NR_REGISTRO", "NR_REGISTRO") \
.withColumnRenamed("SIT_INT_ID_SITE", "ID_SITE") \
.withColumnRenamed("CAR_INT_CD_STATUS", "CD_STATUS") \
.withColumn("CAR_STR_NR_CPF", col("CAR_STR_NR_CPF").cast("integer")) \
.withColumnRenamed("CAR_STR_NR_CPF", "NR_CPF") \
.withColumnRenamed("CAR_STR_DS_NOMELINHA1", "DS_NOMELINHA1") \
.withColumn("CAR_DAT_DT_NASCIMENTO", col("CAR_DAT_DT_NASCIMENTO").
 ↪cast("timestamp")) \
.withColumnRenamed("CAR_DAT_DT_NASCIMENTO", "DT_NASCIMENTO") \
.withColumnRenamed("CAR_STR_ID_CATEGORIA", "DS_CATEGORIA") \
.withColumnRenamed("CAR_STR_ID_TIPO", "DS_TIPO") \
.withColumnRenamed("CAR_STR_DS_UF", "DS_UF") \
.withColumn("CAR_DAT_DT_EMISSAO", col("CAR_DAT_DT_EMISSAO").cast("timestamp")) \
```

```
.withColumnRenamed("CAR_DAT_DT_EMISSAO", "DT_EMISSAO") \
.withColumn("CAR_DAT_DT_PRIMEIRAHABILITACAO",␣
 ↪col("CAR_DAT_DT_PRIMEIRAHABILITACAO").cast("timestamp")) \
.withColumnRenamed("CAR_DAT_DT_PRIMEIRAHABILITACAO", "DT_PRIMEIRAHABILITACAO") \
.withColumn("CAR_DAT_DT_VALIDADE", col("CAR_DAT_DT_VALIDADE").
 ↪cast("timestamp")) \
.withColumnRenamed("CAR_DAT_DT_VALIDADE", "DT_VALIDADE") \
.withColumnRenamed("CAR_STR_DS_OBSERVACAOLINHA1", "DS_OBSERVACAOLINHA1") \
.withColumnRenamed("CAR_STR_DS_OBSERVACAOLINHA2", "DS_OBSERVACAOLINHA2")

# Printando Dataset com tipo e nome  alterados
dframe_filtrado.printSchema()
```

```
root
 |-- ID_SITE: integer (nullable = true)
 |-- CD_STATUS: integer (nullable = true)
 |-- NR_CPF: integer (nullable = true)
 |-- DS_NOMELINHA1: string (nullable = true)
 |-- DT_NASCIMENTO: timestamp (nullable = true)
 |-- DS_CATEGORIA: string (nullable = true)
 |-- DS_TIPO: string (nullable = true)
 |-- DS_UF: string (nullable = true)
 |-- DT_EMISSAO: timestamp (nullable = true)
 |-- DT_PRIMEIRAHABILITACAO: timestamp (nullable = true)
 |-- DT_VALIDADE: timestamp (nullable = true)
 |-- NR_REGISTRO: integer (nullable = true)
 |-- DS_OBSERVACAOLINHA1: string (nullable = true)
 |-- DS_OBSERVACAOLINHA2: string (nullable = true)
```

[12]:
```
# Eliminando coluna/dado Irrelevante para analise  NR_REGISTRO
dframe_filtrado = dframe_filtrado.drop("NR_REGISTRO")

# Printando schema sem o registro acima deletado NR_REGISTRO
dframe_filtrado.printSchema()
```

```
root
 |-- ID_SITE: integer (nullable = true)
 |-- CD_STATUS: integer (nullable = true)
 |-- NR_CPF: integer (nullable = true)
 |-- DS_NOMELINHA1: string (nullable = true)
 |-- DT_NASCIMENTO: timestamp (nullable = true)
 |-- DS_CATEGORIA: string (nullable = true)
 |-- DS_TIPO: string (nullable = true)
 |-- DS_UF: string (nullable = true)
 |-- DT_EMISSAO: timestamp (nullable = true)
 |-- DT_PRIMEIRAHABILITACAO: timestamp (nullable = true)
```

```
 |-- DT_VALIDADE: timestamp (nullable = true)
 |-- DS_OBSERVACAOLINHA1: string (nullable = true)
 |-- DS_OBSERVACAOLINHA2: string (nullable = true)
```

[13]:
```
# Gerando datagrid sem coluna acima deletada NR_REGISTRO
dframe_filtrado.show(2)

# Quantidade de Registros e Colunas do Dataset após deletar coluna NR_REGISTRO
print((dframe_filtrado.count(), len(dframe_filtrado.columns)))
```

```
+-------+---------+----------+--------------------+--------------------+---------
---+-------+-----+------------------+----------------------+------------------
+--------------------+--------------------+
|ID_SITE|CD_STATUS|    NR_CPF|       DS_NOMELINHA1|
DT_NASCIMENTO|DS_CATEGORIA|DS_TIPO|DS_UF|
DT_EMISSAO|DT_PRIMEIRAHABILITACAO|        DT_VALIDADE| DS_OBSERVACAOLINHA1|
DS_OBSERVACAOLINHA2|
+-------+---------+----------+--------------------+--------------------+---------
---+-------+-----+------------------+----------------------+------------------
+--------------------+--------------------+
|      9|       22|1538221864|VITORIA MARIA DE …|2002-06-03 00:00:00|
B|      C|   DF|2024-07-08 00:00:00|   2022-04-29 00:00:00|2031-04-28 00:00:00|A
…|EAR              …|
|      8|       22| 167997167|AIRTON KOCHHANN  …|1969-10-12 00:00:00|
AE  |      C|   MA|2024-07-09 00:00:00|   1992-03-18 00:00:00|2029-07-08
00:00:00|                null|                 EAR|
+-------+---------+----------+--------------------+--------------------+---------
---+-------+-----+------------------+----------------------+------------------
+--------------------+--------------------+
only showing top 2 rows

[Stage 15:>                                                         (0 + 1) / 1]
(7799, 13)
```

[14]:
```
# Tentativa de tratamento Cálculo IDADE para responder perguntas 4 e 5 .
# Consegui forçar apenas passando a String  lit("1982-08-06")), mas ai todos␣
 ↪ficavam com 41 anos.
# Não consegui buscar do dataframe a DT_Nascimento para cada registro.

from pyspark.sql.functions import to_timestamp
from pyspark.sql.functions import to_date, datediff, floor, current_date, lit,␣
 ↪concat_ws, col
```

```
age_column = floor(datediff(current_date(), to_date(concat_ws("-",␣
↪lit("1982-08-06")), "yyyy-MM-dd")) / 365)
#age_column = floor(datediff(current_date(), to_date(concat_ws("-",␣
↪lit(dframe_filtrado.select("DT_NASCIMENTO"))), "yyyy-MM-dd")) / 365)
#print(age_column)

# Add the new "Idade" column to the DataFrame
dframe_filtrado = dframe_filtrado.withColumn("Idade", age_column.cast("int"))
dframe_filtrado_idade = dframe_filtrado.select("Idade")
dframe_filtrado_idade.show(2)
```

```
+-----+
|Idade|
+-----+
|   41|
|   41|
+-----+
only showing top 2 rows
```

[15]:
```
# Armazenar a transformação no bucket Silver
# Defina o caminho para o arquivo Parquet no bucket
path_parquet = "gs://pucmvpbucket/transformacao/analise_tratada_3.parquet"

# Salve as novas colunas em Parquet no Cloud Storage
dframe_filtrado.write.format("parquet").option("path", path_parquet).save()
```

3. Carregamento de Dados / Business Level Aggregates - Camada Ouro

[16]:
```
# Autenticação no BigQuery
cliente_bq = bigquery.Client()

# Defina o nome do projeto e do conjunto de dados no BigQuery
dataset_id = "emissao_raw"
```

[ ]:
```
from pyspark.sql import SparkSession
# Imnportando tabela no BigQuery
dframe_filtrado.write.format("bigquery").option("temporaryGcsBucket", "").
↪option("writeMethod", "DIRECT").option("project", project_id).
↪option("dataset", dataset_id).option("table", "d_emissao_final").save()
```

[17]:
```
# Emissao de Ct's por UF
dframe_filtrado_1 = dframe_filtrado.groupBy(["DS_UF"]).agg(F.count("NR_CPF").
↪alias("QTD_CNH"))
dframe_filtrado_1.show()
```

```
+-----+-------+
|DS_UF|QTD_CNH|
+-----+-------+
|   GO|   1683|
|   RJ|   5361|
|   MA|    331|
|   DF|    424|
+-----+-------+
```

[20]:
```
# Count qtd linhas Dataframe
count_cl = dframe_filtrado.count()

# Porcentagem de Emissoes por UF
dframe_filtrado_2 = dframe_filtrado.groupBy(["DS_UF"]).count().withColumn('%␣
 ↪UF', func.round((func.col('count')/count_cl)*100,2)) \
.orderBy('count', ascending=False) \
.show()
```

```
+-----+-----+-----+
|DS_UF|count| % UF|
+-----+-----+-----+
|   RJ| 5361|68.74|
|   GO| 1683|21.58|
|   DF|  424| 5.44|
|   MA|  331| 4.24|
+-----+-----+-----+
```

[19]:
```
# Count qtd linhas Dataframe
count_cl = dframe_filtrado.count()

# Porcentagem de Emissoes por Tipo CNH, onde : C = Condutor  e  P =␣
 ↪Permissionário
dframe_filtrado_3 = dframe_filtrado.groupBy(["DS_TIPO"]).count().withColumn('%␣
 ↪Tipo CNH', func.round((func.col('count')/count_cl)*100,2)) \
.orderBy('count', ascending=False) \
.show()
```

```
[Stage 54:>                                                         (0 + 1) / 1]
+-------+-----+----------+
|DS_TIPO|count|% Tipo CNH|
+-------+-----+----------+
|      C| 6733|     86.33|
|      P| 1066|     13.67|
+-------+-----+----------+
```

[ ]:
```
# Stop SparkSession
spark.stop()
```

[ ]:
```

```