Teste para Engenharia de Dados - Cognitivo AI

Autora: Caroline Inês Lisevski

A proposta desse trabalho é, a partir de um dataset no formato CSV, identificar os registros que estão deduplicados e realizar processos de modo a permanecer com a entrada mais recente de cada registro.

Os requisitos desse trabalho são:

-todas as operações devem ser realizadas usando o Spark;

-cada operação deverá ser executada no dataframe anterior;

-o arquivo final deverá estar em um formato colunar de alta perfomance de leitura;

-algumas variáveis deverão ser convertidas, conforme arquivo types_mapping.json.

O processamento será local por ter um dataset pequeno. Inicialmente serão importadas as bibliotecas necessárias para o desenvolvimento do trabalho.

In [1]:
```
# importando as bibliotecas necessárias
import pandas as pd
import findspark #usado para encontrar a instalação do Spark na máquina
findspark.init()
from pyspark.sql import SparkSession, functions
from pyspark.sql.types import TimestampType, IntegerType
from pyspark.sql.functions import col
import json
```

Primeiramente é necessário iniciar uma sessão no Spark e definir que o local de processamento é a máquina que estamos trabalhando ('local[*]')

In [2]:
```
# iniciar o spark
spark = SparkSession.builder.master('local[*]').getOrCreate()
```

Vamos usar a biblioteca Pandas para carregar o dataset e, na sequência, esse arquivo será convertido em Spark DataFrame.

In [3]:
```
# carregar o dataset e converter ele em spark dataframe
data = pd.read_csv("data/input/users/load.csv")
df = spark.createDataFrame(data)
```

Vamos olhar o Data Frame usando o comando show( ):

In [4]:
```
df.show()
```

```
+---+--------------------+--------------------+--------------+-----------
--------+---+--------------------+--------------------+
| id|                name|               email|         phone|
address|age|         create_date|         update_date|
+---+--------------------+--------------------+--------------+-----------
--------+---+--------------------+--------------------+
|  1|david.lynch@cogni...|         David Lynch|(11) 99999-9997|Mulholland D
rive,...| 72|2018-03-03 18:47:...|2018-03-03 18:47:...|
|  1|david.lynch@cogni...|         David Lynch|(11) 99999-9998|Mulholland D
rive,...| 72|2018-03-03 18:47:...|2018-04-14 17:09:...|
|  2|sherlock.holmes@c...|     Sherlock Holmes|(11) 94815-1623|221B Baker S
treet...| 34|2018-04-21 20:21:...|2018-04-21 20:21:...|
|  3|spongebob.squarep...|Spongebob Squarep...|(11) 91234-5678|124 Conch St
reet,...| 13|2018-05-19 04:07:...|2018-05-19 04:07:...|
|  1|david.lynch@cogni...|         David Lynch|(11) 99999-9999|Mulholland D
rive,...| 72|2018-03-03 18:47:...|2018-05-23 10:13:...|
|  3|spongebob.squarep...|Spongebob Squarep...|(11) 98765-4321|122 Conch St
reet,...| 13|2018-05-19 04:07:...|2018-05-19 05:08:...|
+---+--------------------+--------------------+--------------+-----------
--------+---+--------------------+--------------------+
```

Esse Data Frame possui cinco registros com as variáveis 'id', 'name', 'email', 'phone', 'address', 'age', 'create_date' and 'update_date'. Analisando os registros é possível observar que os dados da variável 'name' estão na variável 'email' e vice-versa. Nesse caso é suficiente apenas renomear as colunas de acordo com as entradas.

In [5]:
```
df = df.withColumnRenamed("name", "e_mail").withColumnRenamed("email", "nar

df.show()
```

```
+---+--------------------+--------------------+--------------+-----------
--------+---+--------------------+--------------------+
| id|              e_mail|                name|         phone|
address|age|         create_date|         update_date|
+---+--------------------+--------------------+--------------+-----------
--------+---+--------------------+--------------------+
|  1|david.lynch@cogni...|         David Lynch|(11) 99999-9997|Mulholland D
rive,...| 72|2018-03-03 18:47:...|2018-03-03 18:47:...|
|  1|david.lynch@cogni...|         David Lynch|(11) 99999-9998|Mulholland D
rive,...| 72|2018-03-03 18:47:...|2018-04-14 17:09:...|
|  2|sherlock.holmes@c...|     Sherlock Holmes|(11) 94815-1623|221B Baker S
treet...| 34|2018-04-21 20:21:...|2018-04-21 20:21:...|
|  3|spongebob.squarep...|Spongebob Squarep...|(11) 91234-5678|124 Conch St
reet,...| 13|2018-05-19 04:07:...|2018-05-19 04:07:...|
|  1|david.lynch@cogni...|         David Lynch|(11) 99999-9999|Mulholland D
rive,...| 72|2018-03-03 18:47:...|2018-05-23 10:13:...|
|  3|spongebob.squarep...|Spongebob Squarep...|(11) 98765-4321|122 Conch St
reet,...| 13|2018-05-19 04:07:...|2018-05-19 05:08:...|
+---+--------------------+--------------------+--------------+-----------
--------+---+--------------------+--------------------+
```

Agora vamos ver os tipos de variáveis do Data Frame

In [6]:
```
df.printSchema()
```

```
root
 |-- id: long (nullable = true)
 |-- e_mail: string (nullable = true)
 |-- name: string (nullable = true)
 |-- phone: string (nullable = true)
 |-- address: string (nullable = true)
 |-- age: long (nullable = true)
 |-- create_date: string (nullable = true)
 |-- update_date: string (nullable = true)
```

Algumas das variáveis precisam ser convertidas para tipos específicos. O arquivo types_mapping.json nos dá quais variáveis devem ser convertidas e para qual tipo.

In [7]:
```
#importanto o arquivo json com informações sobre as variáveis que precisam
with open('config/types_mapping.json') as mapping:
    dados = json.load(mapping)

print(dados)
```

```
{'age': 'integer', 'create_date': 'timestamp', 'update_date': 'timestamp'}
```

A variável 'age' deve ser convertida em 'integer' e as variáveis 'create_date' e 'update_date' para 'timestamp'.

In [8]:
```
#alterar a variável age como Integer
df = df.withColumn('age', df['age'].cast(IntegerType()))
#alterar as variáveis create_date e update_date como Timestamp
df = df.withColumn('create_date', df['create_date'].cast(TimestampType()))
df = df.withColumn('update_date', df['update_date'].cast(TimestampType()))
```

In [9]:
```
#verificando se as alterações nas variáveis foram realizadas
df.printSchema()
```

```
root
 |-- id: long (nullable = true)
 |-- e_mail: string (nullable = true)
 |-- name: string (nullable = true)
 |-- phone: string (nullable = true)
 |-- address: string (nullable = true)
 |-- age: integer (nullable = true)
 |-- create_date: timestamp (nullable = true)
 |-- update_date: timestamp (nullable = true)
```

Na sequência será feita a filtragem dos dados de modo que seja mantido o dado mais recente ('update_date') para cada 'id'.

In [10]:
```
#agrupando dados por id, ordenando update_date de maneira descendente e ma
#de update_date para cada id.

df1 = df.orderBy(col('id').asc(),col('update_date').desc()).drop_duplicates
df1.orderBy('id').show()
```

```
+---+--------------------+--------------------+--------------+-----------
--------+---+--------------------+--------------------+
| id|              e_mail|                name|         phone|
address|age|         create_date|         update_date|
+---+--------------------+--------------------+--------------+-----------
--------+---+--------------------+--------------------+
|  1|david.lynch@cogni...|         David Lynch|(11) 99999-9999|Mulholland D
rive,...| 72|2018-03-03 18:47:...|2018-05-23 10:13:...|
|  2|sherlock.holmes@c...|     Sherlock Holmes|(11) 94815-1623|221B Baker S
treet...| 34|2018-04-21 20:21:...|2018-04-21 20:21:...|
|  3|spongebob.squarep...|Spongebob Squarep...|(11) 98765-4321|122 Conch St
reet,...| 13|2018-05-19 04:07:...|2018-05-19 05:08:...|
+---+--------------------+--------------------+--------------+-----------
--------+---+--------------------+--------------------+
```

Agora esses dados serão exportados para um arquivo em formato colunar. Para esse trabalho o formato colunar escolhido foi o Parquet porque ele fornece uma compressão de dados eficiente e também porque o Spark possui suporte em sua biblioteca para o Parquet, o que não exige a instalação de mais bibliotecas.

In [14]:
```
#coverter o arquivo para formato colunar Parquet

df1.write.parquet("data/output/out.parquet")
```

In [15]:
```
#verificando se o arquivo foi salvo
df2 = spark.read.parquet("data/output/out.parquet")
df2.orderBy('id').show()
```

```
+---+--------------------+--------------------+--------------+-----------
--------+---+--------------------+--------------------+
| id|              e_mail|                name|         phone|
address|age|         create_date|         update_date|
+---+--------------------+--------------------+--------------+-----------
--------+---+--------------------+--------------------+
|  1|david.lynch@cogni...|         David Lynch|(11) 99999-9999|Mulholland D
rive,...| 72|2018-03-03 18:47:...|2018-05-23 10:13:...|
|  2|sherlock.holmes@c...|     Sherlock Holmes|(11) 94815-1623|221B Baker S
treet...| 34|2018-04-21 20:21:...|2018-04-21 20:21:...|
|  3|spongebob.squarep...|Spongebob Squarep...|(11) 98765-4321|122 Conch St
reet,...| 13|2018-05-19 04:07:...|2018-05-19 05:08:...|
+---+--------------------+--------------------+--------------+-----------
--------+---+--------------------+--------------------+
```

In [ ]:
```

```