PRINCIPAIS COMANDOS (SQL + PYSPARK)

PODEM SER FEITOS TANTO EM (SQL) QUANTO EM (PYSPARK)

AUTOR : RENAN DA SILVA RAMOS

READ – PYSPARK (LEITURA)

- DF = spark.read.format("csv").load("dbfs:.... ")
- DF = spark.read.csv("dbfs:.... " , header = True , encoding = "iso-8859-1" , sep = ",")
- DF = spark.read.json("resources/zipcodes.json")
- DF = spark.read.format('org.apache.spark.sql.json') \
- .load("resources/zipcodes.json")

WRITE – PYSPARK – (ESCRITA)

- Append = escrita que será adicionada ao texto anterior
- df.write.mode('append').parquet("/tmp/output/people.parquet")

- Overwrite = Escrita que sobrescreverá aquilo que já foi escrito.
- df.write.mode('overwrite').parquet("/tmp/output/people.parquet")

SELECT - (PYSPARK + SQL) - (SELECIONAR)

- df.createOrReplaceTempView("test") = DF PARA (VIEW)
- spark.sql(""" SELECT * FROM test """).show()

- DF = spark.select(col('coluna')).show()
- DF = spark.select('coluna').show()

DISTINCT – (PYSPARK + SQL) - (ÚNICOS)

- spark.sql("""SELECT DISTINCT coluna FROM tabela """).show()
- spark.sql("""SELECT DISTINCT coluna COUNT(col) as QNT FROM tabela """).show()

- DF = df.distinct()
- DF = df.dropDuplicates()

FILTER / WHERE - (PYSPARK) - (FILTRANDO)

OBS : (& = AND) - (| = OR) OBS2 : WHERE = FILTER

- DF = df.filter(col("state") == "OH")
- DF = df.filter("gender == 'M'").show()
- DF = df.filter((df.coluna1 == "OH") & (df.coluna2 == "M"))
- DF = df.filter((df.coluna1 == "OH") | (df.coluna2 == "M"))
- li=["OH","CA","DE"] => (ISIN) => PEGA CADA ELEMENTO DA LISTA
- DF = df.filter(df.coluna.isin(li)).show()
- DF = df.where(col('Flex') == 'Sim').where(col('Marca').like('v%')).display()

ISIN – (PYSPARK) FILTRO MAIS APROFUNDADO

```
# 3) ENCONTRE OS DADOS DOS CARROS (Civic, Palio)

df.filter(col('Carro').isin('Civic','Palio')).display()
# NÃO FUNCIONA ABAIXO => (O WHERE SÓ PEGA FILTROS DIFERENTES)
#df.filter(col('Carro') == 'Civic').filter(col('Carro') == 'Palio').display()
```

(3) Spark Jobs

	cod	Carro	Marca	Ano	Preco	Flex
1	4	Palio	Fiat	1998	8900	Não
2	7	Civic	Honda	2019	89990	Sim

```
# 6) MOSTRE TODOS OS CARROS ,
#QUE (NÃO) SÃO DA (MARCA = Fiat , Honda ,Chevrolet)
#utilizando (~ isin) = (~) = 'NÃO'

df.filter(~ col('Marca').isin('Fiat','Chevrolet','Honda')).display()
```

(3) Spark Jobs

	cod	Carro	Marca	Ano	Preco	Flex
1	2	Gol	volkswagen	2020	46320	Sim
2	5	ASX	Mitsubishi	2016	65000	Sim
3	6	Corolla	Toyota	2018	89990	Sim
4	8	Eco Sport	Ford	2017	69299	Sim

```
# 4) ENCONTRE CARROS DA (MARCA = CHEVROLET) QUE (NÃO) SEJA (FLEX)

df.filter(col('Marca') == 'Chevrolet').filter(col('Flex') == 'Não').display()

#abaixo tambem da
#df.filter(col('Marca').isin('Chevrolet')).filter(col('Flex').isin('Não')).display()
```

(3) Spark Jobs

	cod	Carro	Marca	Ano	Preco	Flex
1	1	Celta	Chevrolet	2004	18000	Não

```
# Encontre os dados sobre esses valores da lista
lista = ['Chevrolet','Fiat','Honda']

df.filter(col('marca').isin(lista)).display()
```

(3) Spark Jobs

	cod	Carro	Marca	Ano	Preco	Flex
1	1	Celta	Chevrolet	2004	18000	Não
2	3	Corsa Classic	Chevrolet	2008	15000	Sim
3	4	Palio	Fiat	1998	8900	Não
4	7	Civic	Honda	2019	89990	Sim
5	9	Onix	Chevrolet	2020	58990	Sim

WHEN-OTHERWISE – (IF/ELSE) - CONDICIONAIS

```
# 9) Crie uma coluna chamada (ADULTO) ==> para os que tem idade maior que 22

df = df.withColumn('Adulto' , when(col('idade') > 22 , lit('sim') ).otherwise('não'))

# df = df.withColumn('Adulto' , when(col('idade') > 22 , 'sim' ).otherwise('não')) # => tambem funciona

df.show()
```

(3) Spark Jobs

```
+----------+--------+-----------+-----------------+----------+-----+------+--------+------+
|cod_cliente|    nome|  municipio|           estado| data_nasc|idade|status|Paulista|Adulto|
+----------+--------+-----------+-----------------+----------+-----+------+--------+------+
|         1|    José|   Anápolis|        São Paulo|01/09/1900|  122|    OK|     Sim|   sim|
|         2|    Igor|   Anápolis|        São Paulo|11/09/1977|   45|    OK|     Sim|   sim|
|         3|Leonardo|   Anápolis|        São Paulo|21/12/2000|   22|    OK|     Sim|   não|
|         4|Humberto|Pato Branco|Rio Grande do Sul|13/11/1964|   58|    OK|     Não|   sim|
|         5|  Isaías|Pato Branco|Rio Grande do Sul|07/07/2002|   20|   NOT|     Não|   não|
|         6|   Lucas|       Tauá|            Ceará|05/09/1984|   38|    OK|     Não|   sim|
+----------+--------+-----------+-----------------+----------+-----+------+--------+------+
```

GROUP BY - (PYSPARK) - AGRUPANDO POR ITEM

OBS => SÓ PODE USAR COM (COUNT E AGG)

```
df.groupBy("department").count()
```

```
df.groupBy("department").min("salary")
```

```
df.groupBy("department").avg( "salary")
```

```
df.groupBy("department") \
    .agg(sum("salary").alias("sum_salary"), \
         avg("salary").alias("avg_salary"), \
         sum("bonus").alias("sum_bonus"), \
         max("bonus").alias("max_bonus") \
     ) \
    .show(truncate=False)
```

municipio
Anápolis
Anápolis
Anápolis
Pato Branco
Pato Branco
Tauá

```
# 1) EXIBA A QNT DE MUNICIPIOS (GROUP BY + COUNT)

df = df.groupBy('municipio').count().show()
```

▸ (2) Spark Jobs

```
+-----------+-----+
|  municipio|count|
+-----------+-----+
|   Anápolis|    3|
|Pato Branco|    2|
|       Tauá|    1|
+-----------+-----+
```

ORDER BY - (PYSPARK) - ORDENANDO

```
5  # TRANSFORMANDO DO MAIOR PARA O MENOR
6  #df = df.orderBy(col('anos').desc()).show()
7  df.select('anos').orderBy(col('anos').desc()).show()
8
```

▸ (1) Spark Jobs

```
+----+
|anos|
+----+
| 122|
|  58|
|  46|
|  45|
|  45|
|  38|
|  26|
|  22|
|  20|
+----+
```

REPLACE / REGEPX - (PYSPARK + SQL) - (SUBSTITUIR - VALORES)

- spark.sql(""" SELECT REPLACE(preco , '$' , 'R$') AS valor FROM tabela """)
- DF = df.withColumn('preco' , regexp_replace('preco' , '$' , 'R$'))

LPAD / RPAD - (PYSPARK) - (ADD . DIR/ESQ)

```
1  #5) NO CAMPO (COD_CLIENTE) ==> COLOCAR 3 ZEROS A ESQUERDA DO COD
2  dados = df.withColumn('cod_cliente' , lpad(df.cod_cliente, 4, '0'))
3
4  dados.show()
```

▸ (3) Spark Jobs

```
+----------+--------+----------+-----------------+----------+
|cod_cliente|    nome|  municipio|           estado| data_nasc|
+----------+--------+----------+-----------------+----------+
|      0001|    José|   Anápolis|        São Paulo|01-09-1900|
|      0002|    Igor|   Anápolis|        São Paulo|11-09-1977|
|      0003|Leonardo|   Anápolis|        São Paulo|21-12-2000|
|      0004|Humberto|Pato Branco|Rio Grande do Sul|13-11-1964|
|      0005|  Isaías|Pato Branco|Rio Grande do Sul|07-07-2002|
|      0006|   Lucas|       Tauá|            Ceará|05-09-1984|
+----------+--------+----------+-----------------+----------+
```

```
1  # 6) adicionar 2 (x) a direito do cod_cliente
2
3  test = df.withColumn('cod_cliente', rpad(df.cod_cliente, 3, 'x'))
4
5  test.show()
```

▸ (3) Spark Jobs

```
+----------+--------+----------+-----------------+----------+
|cod_cliente|    nome|  municipio|           estado| data_nasc|
+----------+--------+----------+-----------------+----------+
|      1xx|    José|   Anápolis|        São Paulo|01-09-1900|
|      2xx|    Igor|   Anápolis|        São Paulo|11-09-1977|
|      3xx|Leonardo|   Anápolis|        São Paulo|21-12-2000|
|      4xx|Humberto|Pato Branco|Rio Grande do Sul|13-11-1964|
|      5xx|  Isaías|Pato Branco|Rio Grande do Sul|07-07-2002|
|      6xx|   Lucas|       Tauá|            Ceará|05-09-1984|
+----------+--------+----------+-----------------+----------+
```

CAST - (PYSPARK + SQL) - (MUDANDO TIPOS)

- spark.sql(""" SELECT CAST(coluna AS INT) , CAST(col AS DOUBLE) FROM tabela """)
- DF = df.select(col('coluna').cast('int') , col('colun').cast('double'))
- DF = df.withColumn("salary", col("salary").cast("Integer")).show()

LIKE - (PYSPARK + SQL) - (A% / A%A / %A)

- spark.sql("SELECT * FROM tabela WHERE coluna LIKE 'a%' ")

- DF = df.filter(col('coluna').like("ab%")).show()
- DF = df.filter("carro like 'A%' ").display()
- DF = df.where(col('Carro').like('A%')).display()

BETWEEN - (PYSPARK + SQL) - (INTERVALOS)

- spark.sql(" SELECT * FROM carros WHERE preco BETWEEN 500 AND 800")
- DF = df.where(col('preco').between(500 , 600))

SUBSTRING - (SQL) - OBTENDO VALOR POR (POSIÇÃO)

```
# TESTANDO SUBSTRING

teste_substring = spark.sql("""SELECT modelo_carro,
SUBSTRING(modelo_carro, 2,3 ) mod_sub,
LEFT(modelo_carro, 1) mod_left ,
RIGHT(modelo_carro, 1) mod_right FROM teste """)

teste_substring.show(5)
```

```
+------------+-------+--------+---------+
|modelo_carro|mod_sub|mod_left|mod_right|
+------------+-------+--------+---------+
|      Avalon|    val|       A|        n|
|          RDX|     DX|       R|        X|
|         Golf|    olf|       G|        f|
|           EX|      X|       E|        X|
|       Escort|    sco|       E|        t|
```

SUBSTRING - (PYSPARK) - OBTENDO VALOR POR (POSIÇÃO)

```
spark_tab3.withColumn("mod_sub" , substring("modelo_carro" , 2 , 4)).\
                    withColumn("mod_left" , expr("LEFT(modelo_carro , 2)")).\
                              withColumn("mod_rigth" , expr("RIGHT(modelo_carro
```

```
+--------+-------------+--------+---------+-------+--------+---------+
|id_carro| modelo_carro|   preco|cod_marca|mod_sub|mod_left|mod_rigth|
+--------+-------------+--------+---------+-------+--------+---------+
|       1|       Avalon|78401.95|       54|   valo|      Av|       on|
|       2|          RDX|95987.38|        1|     DX|      RD|       DX|
|       3|         Golf|61274.55|       55|    olf|      Go|       lf|
|       4|           EX|84981.12|       23|      X|      EX|       EX|
|       5|       Escort|77466.89|       17|   scor|      Es|       rt|
```

SPLIT - (PYSPARK) - OBTENDO VALOR POR (CARACTER)

```
+---------+----------+--------+----------+
|firstname|middlename|lastname|dob       |
+---------+----------+--------+----------+
|James    |          |Smith   |1991-04-01|
|Michael  |Rose      |        |2000-05-19|
|Robert   |          |Williams|1978-09-05|
```

```
+---------+----------+--------+----------+----+-----+---+
|firstname|middlename|lastname|dob       |year|month|day|
+---------+----------+--------+----------+----+-----+---+
|James    |          |Smith   |1991-04-01|1991|04   |01 |
|Michael  |Rose      |        |2000-05-19|2000|05   |19 |
|Robert   |          |Williams|1978-09-05|1978|09   |05 |
```

```
DF = df.withColumn('year', split(col('dob'), '-').getItem(0)) \
     .withColumn('month', split(col('dob'), '-').getItem(1)) \
     .withColumn('day', split(col('dob'), '-').getItem(2))
```

DATAS - (PYSPARK + SQL) - FUNCÕES DE DATAS
(TO_TIMESTAMP / TO_DATE / DATE_FORMAT)

OBS => (DATE_FORMAT) ==> VOCÊ ESCOLHE

- spark.sql("SELECT TO_TIMESTAMP(coluna) data FROM tabela ")
- spark.sql("SELECT TO_DATE(coluna) data FROM tabela ")

```
+-------------------+
|     data_timestamp|
+-------------------+
|2021-07-05 10:00:00|
|2020-12-05 00:09:00|
|2017-02-23 16:24:00|
+-------------------+
```

- DF = df.withColumn('Coluna' , to_date('coluna')).show()
- DF = df.withColumn('Coluna' , to_timestamp('coluna')).show()
- DF = df.withColumn('Coluna' , date_format('coluna' , 'dd-MM-yyyy')).show()
- DF = df.withColumn('data_atual' , date_format(current_timestamp(), 'dd-MM-yyyy'))

```
+----------+
| data_date|
+----------+
|2021-07-05|
|2020-12-05|
|2017-02-23|
+----------+
```

DATAS-2 – (PYSPARK) – CALCULANDO PERÍODOS (DATEDIFF) – (ROUND + MONTHS_BETWEEN)

```
# 4) CRIE 3 COLUNAS ==> (DIAS_CORRIDOS) / (MESES_CORRIDOS) / (ANOS_CORRIDOS) ==>
#UTILIZANDO A COLUNA (DATE) COMO REFERENCIA COM A (DATA ATUAL)

# (current_date())  ===> data atual
# (datediff)   ===> calcula os dias
# (round + months , 0)   ===> qnt de meses sem (arredondar)
# (round + months , lit(12),0)   ==> qnt de anos sem (arredondar)
df = df.withColumn('Dias_corridos' , datediff( current_date() , col('date') ))\
        .withColumn('Meses_corridos' , round(months_between( current_date(), col('date')),0))\
        .withColumn('Anos_corridos' , round(months_between(current_date(),col('date'))/lit(12), 0)).display()
```

▸ (3) Spark Jobs

	id	date	Data_atual	Dias_corridos	Meses_corridos	Anos_corridos
1	1	2019-04-15	2022-08-16	1219	40	3
2	2	2020-05-20	2022-08-16	818	27	2
3	3	2021-06-25	2022-08-16	417	14	1
4	4	2022-06-28	2022-08-16	49	2	0

INNER JOIN - (PYSPARK + SQL) - UNIÃO COLUNAS

- spark.sql("SELECT * FROM tabela a INNER JOIN tabela b ON a.coluna = b.coluna")
- spark.sql("SELECT a.* , b.col FROM tabela a INNER JOIN tabela b ON a.coluna = b.coluna")

- DF = df.tab1.join(df.tab2 , (tab1.coluna == tab2.coluna) , "inner")
- DF = df.tab1.join(df.tab2 , (tab1.coluna == tab2.coluna) , "inner")\

.select(col('tab1.coluna').alias('qualquer') , col('tab2.coluna'))

EXISTS - (PYSPARK + SQL) - VALORES EXISTENTES EM AMBAS TABELAS

- spark.sql("SELECT * FROM tab1 a WHERE EXISTS (SELECT * FROM tab2 b WHERE a.col = b.col) ")
- spark.sql("SELECT * FROM tab1 a WHERE NOT EXISTS (SELECT * FROM tab2 b WHERE a.col = b.col) ")

- DF = df1.join(df2 , df1.col = df2.col , "leftsemi") => leftsemi = EXISTS
- DF = df1.join(df2 , df1.col = df2.col , "leftanti") => leftanti = NOT EXISTS

UNION – (PYSPARK + SQL) - UNINDO DATAFRAMES EM COMUM

OBS => SE OS ESQUEMAS NÃO FOREM IGUAIS, ELE RETORNARÁ UM ERRO.

```
+-------------+----------+-----+------+---+-----+
|employee_name|department|state|salary|age|bonus|
+-------------+----------+-----+------+---+-----+
|James        |Sales     |NY   |90000 |34 |10000|
|Michael      |Sales     |NY   |86000 |56 |20000|
```

```
+-------------+----------+-----+------+---+-----+
|employee_name|department|state|salary|age|bonus|
+-------------+----------+-----+------+---+-----+
|James        |Sales     |NY   |90000 |34 |10000|
|Maria        |Finance   |CA   |90000 |24 |23000|
```

```
+-------------+----------+-----+------+---+-----+
|employee_name|department|state|salary|age|bonus|
+-------------+----------+-----+------+---+-----+
|James        |Sales     |NY   |90000 |34 |10000|
|Michael      |Sales     |NY   |86000 |56 |20000|
|Robert       |Sales     |CA   |81000 |30 |23000|
|Maria        |Finance   |CA   |90000 |24 |23000|
|James        |Sales     |NY   |90000 |34 |10000|
|Maria        |Finance   |CA   |90000 |24 |23000|
|Jen          |Finance   |NY   |79000 |53 |15000|
|Jeff         |Marketing |CA   |80000 |25 |18000|
|Kumar        |Marketing |NY   |91000 |50 |21000|
+-------------+----------+-----+------+---+-----+
```

OVERLAY - (PYSPARK) - UNINDO COLUNAS

OBS => NÃO PRECISA TER RELAÇÃO

```
1  ##3) CRIAR UMA COLUNA UNINDO A COLUNA (COD_CLIENTE + NOME)
2
3  df = df.withColumn('cod+nome_client' , overlay('cod_cliente','nome',10))
4
5  df.show()
```

▸ (3) Spark Jobs

```
+----------+---------+-------------+-----------------+----------+----+---------------+
|cod_cliente|     nome|     municipio|           estado| data_nasc|anos|cod+nome_client|
+----------+---------+-------------+-----------------+----------+----+---------------+
|         1-|     José|      Anápolis|        São Paulo|01-09-1900| 122|          1-José|
|         2-|     Igor|      Anápolis|        São Paulo|11-09-1977|  45|          2-Igor|
|         3-| Leonardo|      Anápolis|        São Paulo|21-12-2000|  22|     3-Leonardo|
|         4-| Humberto|   Pato Branco|Rio Grande do Sul|13-11-1964|  58|     4-Humberto|
|         5-|   Isaías|   Pato Branco|Rio Grande do Sul|07-07-2002|  20|       5-Isaías|
|         6-|    Lucas|         Tauá|            Ceará|05-09-1984|  38|        6-Lucas|
```

CONCAT – (PYSPARK) – (UNINDO COLUNAS)

```
1  # 1) CRIE UMA COLUNA (NOME COMPLETO) UNINDO A COLUNA (NOME) + (SOBRENOME) ==> FUNCAO SELECT+ (CONCAT)
2
3  df.select(concat('nome','sobrenome').alias('Nome_Completo')).show()
4
5  ## QUANDO QUER ADD (ALGUMA COISA ) NA UNIAO USA ==> CONCAT_WS
6
7  df.select(concat_ws(' ', 'nome', 'sobrenome').alias('Nome_compl2')).show()
```

▸ (6) Spark Jobs

```
+---------------+
|  Nome_Completo|
+---------------+
|      JoséSilva|
|      IgorAlves|
|  LeonardoLopes|
|HumbertoFerreira|
|  IsaíasPereira|
|    LucasSantos|
+---------------+

+----------------+
|     Nome_compl2|
+----------------+
|      José Silva|
|      Igor Alves|
|  Leonardo Lopes|
|Humberto Ferreira|
|  Isaías Pereira|
```

IS NULL - (PYSPARK + SQL) - VALORES NULOS

- spark.sql("SELECT * FROM tab1 WHERE coluna IS NULL")
- spark.sql("SELECT * FROM tab1 WHERE coluna IS NOT NULL")

- DF = df1.where(col('coluna').isNull())
- DF = df1.where(~ col('coluna').isNull()) => (~) = NÃO

AGG - (PYSPARK + SQL) - AGREGACOES (SUM , MAX , MIN ,AVG)

- spark.sql(" SELECT col , SUM(col) AS SOMA , MAX(col) AS MAXIMO , MIN(col) AS MINIMO , AVG(col) AS MÉDIA FROM tab1 GROUP BY col ")

- DF = df1.groupBy(col('coluna').agg(sum('col').alias('soma') , max('col') , min('col') , avg('col')))