

PYSPARK – EXEMPLOS COM IMAGENS

AUTOR : RENAN DA SILVA RAMOS



SELECT + ORDER BY - (PYSPARK)

```
1 # 3) ORDENE OS (ESTADOS) DO MAIOR PARA O MENOR (DESCRECENTE) => ORDEM ALFABÉTICA (ORDER BY OU SORT)
2
3 df.select('estado').orderBy(col('estado').desc()).show()
4
5 #df.select('estado').sort(col('estado').desc()).show() # MESMA COISA
```

► (1) Spark Jobs

```
+-----+
|          estado|
+-----+
|        São Paulo|
|        São Paulo|
|        São Paulo|
|Rio Grande do Sul|
|Rio Grande do Sul|
|           Ceará|
+-----+
```

FILTER / WHERE – FILTROS

OBS => SÓ BUSCAM COISAS EM COLUNAS DIFERENTES

OBS2 => SE QUISER BUSCAR 2 ITENS NA MESMA COLUNA USAR (ISIN)

```
1  ## TAMBEM POSSO USAR O (WHERE) COM OS OPERADOS ==> (AND = &) e (OR = Or)
2
3  # 4) ENCONTRE CARROS (FLEX) DO ANO DE (2016)
4
5  #df.filter(col('Flex') == 'Sim').where(col('Ano') == 2016).display()
6
7  # desse jeito posso colocar (AND = & / ou OR = | )
8  #df.where((col('Flex') == 'Sim') & (col('Ano') == 2016)).display()
9
10 # ESSE JEITO É MELHOR DE VER ==> POREM NAO PERMITE (OR ) ==> É SÓMENTE (AND)
11 df.where(col('Flex') == "Sim").where(col('Ano') == 2016).display()
12
```

► (3) Spark Jobs

	cod ▲	Carro ▲	Marca ▲	Ano ▲	Preco ▲	Flex ▲	
1	5	ASX	Mitsubishi	2016	65000	Sim	

Showing all 1 rows.

FILTRO / WHERE (2) - NOVAS MANEIRAS DE USAR

```
1 # 5) ENCONTRE O CARRO QUE COMECE COM A LETRA ("C") (NÃO SEJA FLEX) E SEJA DO (ANO = 2004)
2
3 #df.where((col('Carro').like('C%') & (col('Flex') == "Não") & (col('Ano') == 2004))).display()
4
5 #df.filter(col('Carro').like('C%')).filter(col('Flex') == "Não").filter(col('Ano') == 2004).display()
6
7 df.where(col('Carro').like('C%')).where(col('Flex') == "Não").where(col('Ano') == 2004).display()
```

► (3) Spark Jobs

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

```
# 6) TRAGA CARROS QUE COMECEM COM A LETRA (T) E (P)

# ERRO PROPOSITAL ABAIXO==> PQ SE ELE NAO ACHAR O PRIMEIRO (FILTRO) ELE JA PARA ==> ENTAO PRECISO USAR (WHERE) COM OPERADORES LOGICOS

#df.filter(col('Carro').like('T%')).filter(col('Carro').like('P%')).display()

# AQUI ELE VAI ACHAR OU UM OUTRO ==> (| = OR )

df.where((col('Carro').like('T%') | (col('Carro').like('P%')))).display() # no caso só VAI ACHAR O ("P") ==> (t) não tem
```

(3) Spark Jobs

	cod	Carro	Marca	Ano	Preco	Flex
1	4	Palio	Fiat	1998	8900	Não

ISIN – FILTRO MAIS APROFUNDADO

```
1 # 3) ENCONTRE OS DADOS DOS CARROS (Civic, Palio)
2
3 df.filter(col('Carro').isin('Civic','Palio')).display()
4 # NÃO FUNCIONA ABAIXO => (O WHERE SÓ PEGA FILTROS DIFERENTES)
5 #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

GROUP BY – AGRUPAMENTO POR ITEM

OBS => SÓ PODE USAR COM (AGG + COUNT)

```
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)
```

município
Anápolis
Anápolis
Anápolis
Pato Branco
Pato Branco
Tauá



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

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

(2) Spark Jobs

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

LIKE - (%A , %A% , %A) - FILTRANDO VALORES INCERTOS

```
1 # 3) TRAGA OS CARROS QUE SÃO (FLEX) E A MARCA COMECE COM A LETRA ("v")
2
3 #df.filter(col('Flex') == 'Sim').filter(col('Marca').like('v%')).display()
4
5 df.where((col('flex') == 'Sim') & (col('Marca').like('v%'))).display()
```

► (3) Spark Jobs

	cod	Carro	Marca	Ano	Preco	Flex
1	2	Gol	volkswagen	2020	46320	Sim

Showing all 1 rows.



Command took 0.32 seconds -- by silvamosrenan@gmail.com at 13/08/2022 20:46:54 on pyspark

Cmd 7

```
1 # 4) TRAGA CARROS QUE (NÃO SÃO FLEX) COM A LETRA FINAL ("a")
2
3
4 df.filter(col('Flex') == 'Não').filter(col('Carro').like('%a')).display()
```

► (3) Spark Jobs

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

STARTSWITH – LETRA CERTA QUE (COMEÇA)

```
# 8) TRAGA CARROS QUE COMECEM COM A LETRA (C) E TENHAM (ANO) IGUAL A 2008 E PODEM SER (FLEX) (OU) NÃO

##==> AQUI N VAI ACHAR NADA, PQ É FLEX|
#df.filter(col('Carro').startswith('C')).where(col('Ano') == 2008).where(col('Flex') == 'Não').display()

# USANDO OPERADORES ==> (AND = &) e (OR = |)

df.where((col('Carro').startswith('C') & (col('Ano') == 2008) | (col('Flex') == "Não" ))).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

DISTINCT / DROPDUPLICATES – VALORES UNICOS

```
1 # 2) TRAZENDO APENAS VALORES (UNICOS) DAS COLUNAS (DEPARTAMENTO E SALARIO) ==> REMOVOU VALORES QUE SE REPETEM NAS DUAS COLUNAS
2
3 df_drop = df.dropDuplicates(["Departamento", "Salario"]).display()
4
5 # df_dist = df.distinct(["Departamento","Salario"]).display() ==> NÃO PODE USAR DISTINCT PARA FAZER O MESMO..
6
7 # O DISTINCT ==> OLHA EM TODAS AS LINHAS E COLUNAS E (REMOVE OS VALORES DUPLICADOS )
8
9 # O DROPDUPLICATES ==> FAZ O MESMO, POREM, VOCÊ PODE SELECIONAR QUAIS (COLUNAS) QUER REMOVER OS VALORES DUPLICADOS.
```

► (2) Spark Jobs

	Nome ▲	Departamento ▲	Salario ▲
1	Maria	Financas	3000
2	Scott	Financas	3300
3	Jonatas	Financas	3900
4	Caio	Marketing	2000
5	Jefferson	Marketing	3000
6	James	Vendas	3000
7	Robert	Vendas	4100

Showing all 8 rows.

BETWEEN - INTERVALOS

```
+-----+-----+
|               start|value|
+-----+-----+
| 2017-04-13 12:00:...|  1.0|
| 2017-04-14 00:00:...|  1.1|
+-----+-----+
```

```
test_df.filter(F.col("start").between('2017-04-13', '2017-04-14')).show()
```

```
+-----+-----+
|               start|value|
+-----+-----+
| 2017-04-13 12:00:...|  1.0|
+-----+-----+
```

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

Região	Sigla	Contratado
Sudeste	SP	Não
Sudeste	SP	Não
Sudeste	SP	Não
Sul	RS	Sim
Sul	RS	Sim
Nordeste	CE	Sim

```
# 6) CRIE UMA COLUNA (CONTRATADO) SOMENTE PARA PESSOAS QUE SÃO DA (REGIÃO) ==> (SUL) E (NORDESTE)
```

```
df = df.withColumn('Contratado' , when(col('Região').isin('Sul','Nordeste') , lit('Sim')).otherwise('Não')).display()
```



WHEN+OTHERWISE – 2 – (IF /ELSE)

```
df = df.withColumn('Região' , when(col('estado') == 'São Paulo' , lit('Sudeste'))\
    .when(col('estado') == 'Rio Grande do Sul' , lit('Sul'))\
    .when(col('estado') == 'Ceará' , lit('Nordeste')).otherwise('não-possui'))\
    .withColumn('Sigla' , when(col('estado') == 'São Paulo' , lit('SP'))\
    .when(col('estado') == 'Rio Grande do Sul' , lit('RS'))\
    .when(col('estado') == 'Ceará' , lit('CE')).otherwise('não-possui')).display()
```

(3) Spark Jobs

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

```
df5.withColumn("new_column", when((col("code") == "a") | (col("code") == "d"), "A")
    .when((col("code") == "b") & (col("amt") == "4"), "B")
    .otherwise("A1")).show()
```


REGEXP REPLACE – SUBSTITUIR CARACTERE POR (DELIMITADOR)

```
1 # 8) NA COLUNA (DATA_NASC) TROQUE O SINAL (-) por (/)
2
3 # OBS ==> (REGEXP_REPLACE) ==> NA MAIORIA DAS VEZES SÓ FUNCIONA COM (withColumn) => não é bom colocar (filter;where)
4
5 df = df.withColumn('data_nasc', regexp_replace('data_nasc' , '\\-' , '/')) # ESSA (\\) => significa que ira pegar todos sinais
6
7 df.show()
8
9 # OBS => NÃO PODE FAZER DESTE JEITO ABAIXO, POIS ELE NÃO PERCORRE CHARACTER POR CHARACTER ==> ELE BUSCA A PALAVRA COMPLETA
10
11 #df = df.withColumn('data_nasc', when(col('data_nasc') == '-' , "/").otherwise('data_nasc')).show()
12
```

► (3) Spark Jobs

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

DROP - (PYSPARK) - EXCLUSÃO DE COLUNAS

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



```
# 2.1) UTILIZANDO A FUNÇÃO (DROP) => APAGUE A COLUNA (NOME) E (ESTADO)  
df.drop('nome','estado').display()
```

► (3) Spark Jobs

	cod_cliente	municipio	data_nasc
1	1	Anápolis	01-09-1900
2	2	Anápolis	11-09-1977
3	3	Anápolis	21-12-2000
4	4	Pato Branco	13-11-1964
5	5	Pato Branco	07-07-2002
6	6	Tauá	05-09-1984

DATAS - I

CAST + (DATE FORMAT) + (SUBSTRING)

```
# 4) CRIE UMA NOVA COLUNA PARA (IDADE), PEGANDO A DATA ATUAL - O (ANO) DA COLUNA (DATA_NAS)
```

```
data = df.withColumn('idade', date_format(current_timestamp(), 'yyyy')\
                                .cast('integer') - substring(col('data_nasc'), 7,4)\
                                .cast('integer'))
```

```
data.show()
```

(3) Spark Jobs

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

DATAS-2

```
1 # 5) CRIE UMA COLUNA (ANO) - PEGUE SOMENTE O (ANO) DA COLUNA (DATA_NASC) ==> EXISTE BASICAMENTE 3 FORMAS ==> (SPLIT) / (SUBSTRING) / (TO_DATE)
2
3 ## ==> COM (SPLIT)
4
5 #df.withColumn('Ano' , split(col('data_nasc') , '-').getItem(2)).display()
6
7 ## ==> COM (SUBSTRING)
8
9 #df.withColumn('Ano' , substring('data_nasc' , 7 , 4)).display()
10
11
12 ## ==> COM (to_date()) ==> PRIMEIRO TRANSFORMA PARA (DATE) DEPOIS FALA SOMENTE O QUE QUER USANDO (DATE_FORMAT) => ano = 'yyyy'
13
14 df = df.withColumn('Ano' , to_date(col('data_nasc'), 'dd-MM-yyyy'))
15
16 df = df.withColumn('Ano' , date_format(col('Ano'), 'yyyy')).display()
17
18
```

► (3) Spark Jobs

	cod	nome	sobrenome	municipio	estado	data_nasc	Região	Local	Ano
1	1	José	Silva	Anápolis	São Paulo	01-09-1900	Sudeste	Anápolis - São Paulo - Sudeste	1900
2	2	Igor	Alves	Anápolis	São Paulo	11-09-1977	Sudeste	Anápolis - São Paulo - Sudeste	1977
3	3	Leonardo	Lopes	Anápolis	São Paulo	21-12-2000	Sudeste	Anápolis - São Paulo - Sudeste	2000
4	4	Humberto	Ferreira	Pato Branco	Rio Grande do Sul	13-11-1964	Sul	Pato Branco - Rio Grande do Sul - Sul	1964
5	5	Isaías	Pereira	Pato Branco	Rio Grande do Sul	07-07-2002	Sul	Pato Branco - Rio Grande do Sul - Sul	2002
6	6	Lucas	Santos	Tauá	Ceará	05-09-1984	Nordeste	Tauá - Ceará - Nordeste	1984

DATAS -3

```
1 # 4) CRIE 3 COLUNAS ==> (DIAS_CORRIDOS) / (MESES_CORRIDOS) / (ANOS_CORRIDOS) ==>
2 #UTILIZANDO A COLUNA (DATE) COMO REFERENCIA COM A (DATA ATUAL)
3
4 # (current_date()) ==> data atual
5 # (datediff) ==> calcula os dias
6 # (round + months , 0) ==> qnt de meses sem (arredondar)
7 # (round + months , lit(12),0) ==> qnt de anos sem (arredondar)
8 df = df.withColumn('Dias_corridos' , datediff( current_date() , col('date') ))\
9         .withColumn('Meses_corridos' , round(months_between( current_date(), col('date')),0))\
10        .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

CURRENT TIMESTAMP - DATA

```
1 # 8) CRIE UMA COLUNA COM A (DATA ATUAL)
2
3 dados = df.withColumn('data_atual' , date_format(current_timestamp(), 'dd-MM-yyyy'))
4
5 dados.show()
```

► (3) Spark Jobs

cod_cliente	nome	municipio	estado	data_nasc	data_atual
1	José	Anápolis	São Paulo	01-09-1900	16-08-2022
2	Igor	Anápolis	São Paulo	11-09-1977	16-08-2022
3	Leonardo	Anápolis	São Paulo	21-12-2000	16-08-2022
4	Humberto	Pato Branco	Rio Grande do Sul	13-11-1964	16-08-2022
5	Isaías	Pato Branco	Rio Grande do Sul	07-07-2002	16-08-2022
6	Lucas	Tauá	Ceará	05-09-1984	16-08-2022

UNION – UNINDO DATAFRAMES COM MESMO (SCHEMA)

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

LPAD + RPAD – ADD.CARACT (DIREIT/ESQUER)

```
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	município	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	município	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

OVERLAY – 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
7-	Cláudio	Belo Horizonte	Minas Gerais	07-09-1977	45	7-Cláudio
8-	Hélio	Contagem	Belo Horizonte	24-07-1976	46	8-Hélio
9-	Guilherme	Recife	Pernambuco	06-11-1996	26	9-Guilherme

CONCAT – UNINDO COLUNAS E CARACTERES

```
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|
|  IsaiásPereira|
|    LucasSantos|
+-----+

+-----+
| Nome_compl2|
+-----+
|    José Silva|
|    Igor Alves|
|  Leonardo Lopes|
|Humberto Ferreira|
|  Isaiás Pereira|
|    Lucas Santos|
```

SUBSTRING – OBTENDO VALOR POR (POSICAO)

```
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 , 2)"))
```

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 – 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))**

JOIN – UNINDO COLUNAS (COM ALGO EM COMUM)

```
empDF.join(deptDF, empDF.emp_dept_id == deptDF.dept_id, "inner") \
.show(truncate=False)
```

Emp Dataset

emp_id	name	superior_emp_id	year_joined	emp_dept_id	gender	salary
1	Smith	-1	2018	10	M	3000
2	Rose	1	2010	20	M	4000
3	Williams	1	2010	10	M	1000
4	Jones	2	2005	10	F	2000
5	Brown	2	2010	40		-1
6	Brown	2	2010	50		-1

salary	dept_name	dept_id
3000	Finance	10
4000	Marketing	20
1000	Finance	10
2000	Finance	10
-1	IT	40

ISNULL – BUSCANDO – VALORES (NULOS OU NÃO)

```
+-----+-----+-----+
| name | state | gender |
+-----+-----+-----+
| James | null  | M      |
| Anna  | NY    | F      |
| Julia | null  | null   |
+-----+-----+-----+
```



```
df.filter("state is NULL").show()
df.filter(df.state.isNull()).show()
df.filter(col("state").isNull()).show()
```



```
+-----+-----+-----+
| name | state | gender |
+-----+-----+-----+
| James | null  | M      |
| Julia | null  | null   |
+-----+-----+-----+
```

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