

✓ Configurações iniciais

✓ Instalação dos pacotes

```
1 !apt-get install openjdk-8-jdk-headless -qq > /dev/null
2 !wget -q http://archive.apache.org/dist/spark/spark-3.1.1/spark-3.1.1-bin-hadoop3.2.tgz
3 !tar xf spark-3.1.1-bin-hadoop3.2.tgz
4 !pip install -q findspark
```

✓ Criando o ambiente virtual do PySpark

```
1 import os
2 os.environ["JAVA_HOME"] = "/usr/lib/jvm/java-8-openjdk-amd64"
3 os.environ["SPARK_HOME"] = "/content/spark-3.1.1-bin-hadoop3.2"
```

✓ Criando a sessão PySpark

```
1 import findspark
2 findspark.init()
3 from pyspark.sql import SparkSession
4 spark = SparkSession.builder.master("local[*]").getOrCreate()
5 from pyspark.sql.functions import regexp_replace
6 spark.conf.set("spark.sql.repl.eagerEval.enabled", True) # Para deixar a visualização das tabelas mais amigável
7 spark
```

 **SparkSession - in-memory**

SparkContext

[Spark UI](#)

Version

v3.1.1

Master

local[*]


AppName

pyspark-shell

✓ 2 - Lendo um arquivo a partir do disco


```
1 df = spark.read.csv('/content/Estatistica Descritiva - Organização Gráfica.csv',header=True)
```

```
1 df.show()
```



	Matrícula	Nome	Cidade	Estado	País	Idade	Departamento	Cargo	Salário (R\$)	Escolaridade	Nota
1	1	Lucas	Atibaia	SP	Brasil	35	Compras	Gerente	25000	Superior	8
2	2	Ana	São Paulo	SP	Brasil	29	Vendas	Coordenador	12000	Superior	6
3	3	Luiza	Santos	SP	Brasil	38	Finanças	Gerente	28000	MBA	9
4	4	Fernando	Atibaia	SP	Brasil	36	Marketing	Diretor	40000	Mestrado	7
5	5	Sandra	Atibaia	SP	Brasil	28	Produção	Analista	23000	Superior	5
6	6	Douglas	Bragança	SP	Brasil	29	Finanças	Analista	11000	Superior	9
7	7	Eduardo	Extrema	MG	Brasil	30	Marketing	Gerente	12000	MBA	4
8	8	Ester	Itapeva	MG	Brasil	29	Compras	Analista	10000	Superior	2
9	9	Pedro	Extrema	MG	Brasil	30	Marketing	Analista	13000	Superior	1
10	10	Maria	Extrema	MG	Brasil	40	Produção	Analista	12000	MBA	7

```
1 df.printSchema()
```



```
root
|-- Matrícula: string (nullable = true)
|-- Nome: string (nullable = true)
|-- Cidade: string (nullable = true)
|-- Estado: string (nullable = true)
|-- País: string (nullable = true)
|-- Idade: string (nullable = true)
|-- Departamento: string (nullable = true)
|-- Cargo: string (nullable = true)
|-- Salário (R$): string (nullable = true)
|-- Escolaridade: string (nullable = true)
|-- Nota : string (nullable = true)
```

✓ Criando um arquivo a partir do zero

```

1 from pyspark.sql.types import *
2 from pyspark.sql.types import StructType,StructField,StringType,IntegerType,DoubleType # importando estrutura da tabela (structType)
3 schema = StructType([ \
4     StructField('nome',StringType(),True), \
5     StructField('idade',IntegerType(),True), \
6     StructField('altura',DoubleType(),True), \
7     StructField('peso', DoubleType(), True), \
8 ])

```

```

1
2 '''
3 Criação do objeto (nível Python) que contém as tuplas com os registros que farão
4 parte da tabela
5 '''
6 dados = [('Douglas',45,1.85,70.),
7     ('Daniela',7,1.23,22.),
8     ('Pedro',65,1.75,87.),
9     ('Maria',64,1.67,64.),
10    ('Eduardo',37,1.82,96.),
11    ('Ester',37,1.73,68.)
12 ]

```

```
1 df = spark.createDataFrame(schema=schema, data=dados)
```

```
1 df.show()
```

```

↗
+-----+-----+-----+
| nome|idade|altura|peso|
+-----+-----+-----+
|Douglas| 45| 1.85|70.0|
|Daniela| 7| 1.23|22.0|
| Pedro| 65| 1.75|87.0|
| Maria| 64| 1.67|64.0|
|Eduardo| 37| 1.82|96.0|
| Ester| 37| 1.73|68.0|
+-----+-----+-----+

```

```
1 df.printSchema()
```

```

↗ root
 |-- nome: string (nullable = true)
 |-- idade: integer (nullable = true)
 |-- altura: double (nullable = true)
 |-- peso: double (nullable = true)

```

▼ Filtro com operadores lógicos AND (&) OR (|)

```

1 filtro01 = df.filter((df['idade']>=40)&(df['peso']>=50)) # em pyspark o operador lógico and deve ser usado pelo símbolo &
2 filtro01.show()

```

```

↗
+-----+-----+-----+
| nome|idade|altura|peso|
+-----+-----+-----+
|Douglas| 45| 1.85|70.0|
| Pedro| 65| 1.75|87.0|
| Maria| 64| 1.67|64.0|
+-----+-----+-----+

```

```

1 filtro02 = df.filter((df['idade']>=40)|(df['peso']>=50)) # em pyspark o operador lógico ou deve ser usado pelo símbolo |
2 filtro02.show()

```

```

↗
+-----+-----+-----+
| nome|idade|altura|peso|
+-----+-----+-----+
|Douglas| 45| 1.85|70.0|
| Pedro| 65| 1.75|87.0|
| Maria| 64| 1.67|64.0|
|Eduardo| 37| 1.82|96.0|
| Ester| 37| 1.73|68.0|
+-----+-----+-----+

```

```

1 filtro03 = df.filter('idade >= 40 AND peso >= 50')
2 filtro03.show()

```

```

↗
+-----+-----+-----+
| nome|idade|altura|peso|
+-----+-----+-----+
|Douglas| 45| 1.85|70.0|
| Pedro| 65| 1.75|87.0|
| Maria| 64| 1.67|64.0|
+-----+-----+-----+

```

```
1 filtro04 = df.filter('idade >= 40 OR peso >= 50')
```


	nome	idade	altura	massa
1	Douglas	45	1.85	70.0
2	Daniela	7	1.23	22.0
3	Pedro	65	1.75	87.0
4	Maria	64	1.67	64.0
5	Eduardo	37	1.82	96.0
6	Ester	37	1.73	68.0

```

1 # Criando uma nova coluna
2 df = df.withColumn('imc',df['massa']/df['altura']**2)

1 '''
2 Agora imagine que você deseja escolher uma determina linha por seu pseudo-índice
3 '''
4 tio = df.collect()[0]

```

```
1 print(tio)
```

```
Row(nome='Douglas', idade=45, altura=1.85, massa=70.0, imc=20.45288531775018)
```

```

1 '''
2 Agora imagine que você deseja extrair e armazena apenas o valor 'Douglas'
3 '''
4 doug = df.collect()[0][0]

```

```
1 print(doug)
```

```
Douglas
```

```
1 type(doug)
```

```
str
```

```
1 elemento = [('Tobias',18,1.77,80.,25.535446)]
```

```
1 nova_linha = spark.createDataFrame(elemento)
```

```
1 nova_linha.show()
```

```

+-----+-----+-----+-----+
|  _1|  _2|  _3|  _4|  _5|
+-----+-----+-----+-----+
|Tobias| 18|1.77|80.0|25.535446|
+-----+-----+-----+-----+

```

```

1 df = df.union(nova_linha)
2 df.show()

```

```

+-----+-----+-----+-----+-----+
| nome|idade|altura|massa| imc|
+-----+-----+-----+-----+-----+
|Douglas| 45| 1.85| 70.0| 20.45288531775018|
|Daniela| 7| 1.23| 22.0|14.541608830722454|
| Pedro| 65| 1.75| 87.0|28.408163265306122|
| Maria| 64| 1.67| 64.0|22.948115744558788|
|Eduardo| 37| 1.82| 96.0|28.982007003985025|
| Ester| 37| 1.73| 68.0|22.720438370810918|
| Tobias| 18| 1.77| 80.0| 25.535446|
+-----+-----+-----+-----+-----+

```

```

1 '''
2 Agora imagine que você deseja calcular as principais medidas descritivas do df
3 '''
4 df.describe().show()

```

```

+-----+-----+-----+-----+-----+-----+
|summary| nome| idade| altura| massa| imc|
+-----+-----+-----+-----+-----+-----+
| count| 7| 7| 7| 7| 7|
| mean| null| 39.0| 1.6885714285714286| 69.57142857142857| 23.369809219019068|
| stddev| null| 21.641010450839243| 0.21058874387682508| 23.831751529259886| 4.976893829941699|
| min| Daniela| 7| 1.23| 22.0| 14.541608830722454|
| max| Tobias| 65| 1.85| 96.0| 28.982007003985025|
+-----+-----+-----+-----+-----+-----+

```

```

1 '''
2 Agora imagine que você deseja criar uma nova coluna com os nomes só com letras
3 em caixa baixa
4 '''
5 df = df.withColumn('minu',lower(col('nome')))
6 df.show()

```

```

+-----+-----+-----+-----+-----+-----+
| nome|idade|altura|massa| imc| minu|
+-----+-----+-----+-----+-----+-----+
|Douglas| 45| 1.85| 70.0| 20.45288531775018| douglas|
|Daniela| 7| 1.23| 22.0|14.541608830722454| daniela|
| Pedro| 65| 1.75| 87.0|28.408163265306122| pedro|
| Maria| 64| 1.67| 64.0|22.948115744558788| maria|
|Eduardo| 37| 1.82| 96.0|28.982007003985025| eduardo|
| Ester| 37| 1.73| 68.0|22.720438370810918| ester|
| Tobias| 18| 1.77| 80.0| 25.535446| tobias|
+-----+-----+-----+-----+-----+-----+

```

```

1 '''
2 Agora imagine que você deseja criar uma nova coluna com os nomes só com letras
3 em caixa alta
4 '''
5 df = df.withColumn('maiu', upper(col('nome')))
6 df.show()

```

```

+-----+-----+-----+-----+-----+-----+
| nome|idade|altura|massa|          imc|   minu|   maiu|
+-----+-----+-----+-----+-----+-----+
|Douglas|  45|  1.85|  70.0| 20.45288531775018|douglas|DOUGLAS|
|Daniela|  7|  1.23|  22.0|14.541608830722454|daniela|DANIELA|
|Pedro|  65|  1.75|  87.0|28.408163265306122|pedro|PEDRO|
|Maria|  64|  1.67|  64.0|22.948115744558788|maria|MARIA|
|Eduardo| 37|  1.82|  96.0|28.982007003985025|eduardo|EDUARDO|
|Ester|  37|  1.73|  68.0|22.720438370810918|ester|ESTER|
|Tobias| 18|  1.77|  80.0| 25.535446|tobias|TOBIAS|
+-----+-----+-----+-----+-----+-----+

```

```

1 '''
2 Medidas agregadas por coluna
3 Exemplo descobrir o menor valor da coluna idade
4 '''
5 idade_min = df.agg({'idade':'min'}) # como resultado teremos um df spark com uma linha e uma coluna
6 print(idade_min)

```

```

+-----+
|min(idade)|
+-----+
|          7|
+-----+

```

```

1 '''
2 Medidas agregadas por coluna
3 Exemplo descobrir o maior valor da coluna idade
4 '''
5 idade_max = df.agg({'idade':'max'}) # como resultado teremos um df spark com uma linha e uma coluna
6 print(idade_max)

```

```

+-----+
|max(idade)|
+-----+
|         65|
+-----+

```

```

1 '''
2 Medidas agregadas por coluna
3 Exemplo descobrir o maior valor da coluna idade
4 '''
5 idade_media = df.agg({'idade':'avg'}) # como resultado teremos um df spark com uma linha e uma coluna
6 print(idade_media)

```

```

+-----+
|avg(idade)|
+-----+
|        39.0|
+-----+

```

```

1 '''
2 Agora vamos supor que você deseja ordenar de forma crescente por nome
3 '''
4 df.orderBy('nome')

```

nome	idade	altura	massa	imc	minu	maiu
Daniela	7	1.23	22.0	14.541608830722454	daniela	DANIELA
Douglas	45	1.85	70.0	20.45288531775018	douglas	DOUGLAS
Eduardo	37	1.82	96.0	28.982007003985025	eduardo	EDUARDO
Ester	37	1.73	68.0	22.720438370810918	ester	ESTER
Maria	64	1.67	64.0	22.948115744558788	maria	MARIA
Pedro	65	1.75	87.0	28.408163265306122	pedro	PEDRO
Tobias	18	1.77	80.0	25.535446	tobias	TOBIAS

```

1 '''
2 Agora vamos supor que você deseja ordenar de forma decrescente por nome
3 '''
4 df.orderBy(df['nome'].desc())

```

	nome	idade	altura	massa	imc	minu	maiu
	Tobias	18	1.77	80.0	25.535446	tobias	TOBIAS
	Pedro	65	1.75	87.0	28.408163265306122	pedro	PEDRO
	Maria	64	1.67	64.0	22.948115744558788	maria	MARIA
	Ester	37	1.73	68.0	22.720438370810918	ester	ESTER
	Eduardo	37	1.82	96.0	28.982007003985025	eduardo	EDUARDO
	Douglas	45	1.85	70.0	20.45288531775018	douglas	DOUGLAS
	Daniela	7	1.23	22.0	14.541608830722454	daniela	DANIELA

```

1 '''
2 Agora imagine que você deseja agrupar por idade
3 '''
4 df.groupby('idade').agg({'idade':'count'})

```

idade	count(idade)
65	1
7	1
37	2
18	1
64	1
45	1

```
1 df = spark.read.csv('/content/Estatistica Descritiva - Organização Gráfica.csv',header=True)
```

```
1 df.show()
```

	Matrícula	Nome	Cidade	Estado	País	Idade	Departamento	Cargo	Salário (R\$)	Escolaridade	Nota
1	1	Lucas	Atibaia	SP	Brasil	35	Compras	Gerente	25000	Superior	8
2	2	Ana	São Paulo	SP	Brasil	29	Vendas	Coordenador	12000	Superior	6
3	3	Luiza	Santos	SP	Brasil	38	Finanças	Gerente	28000	MBA	9
4	4	Fernando	Atibaia	SP	Brasil	36	Marketing	Diretor	40000	Mestrado	7
5	5	Sandra	Atibaia	SP	Brasil	28	Produção	Analista	23000	Superior	5
6	6	Douglas	Bragança	SP	Brasil	29	Finanças	Analista	11000	Superior	9
7	7	Eduardo	Extrema	MG	Brasil	30	Marketing	Gerente	12000	MBA	4
8	8	Ester	Itapeva	MG	Brasil	29	Compras	Analista	10000	Superior	2
9	9	Pedro	Extrema	MG	Brasil	30	Marketing	Analista	13000	Superior	1
10	10	Maria	Extrema	MG	Brasil	40	Produção	Analista	12000	MBA	7

```
1 df.printSchema()
```

```

root
 |-- Matrícula: string (nullable = true)
 |-- Nome: string (nullable = true)
 |-- Cidade: string (nullable = true)
 |-- Estado: string (nullable = true)
 |-- País: string (nullable = true)
 |-- Idade: string (nullable = true)
 |-- Departamento: string (nullable = true)
 |-- Cargo: string (nullable = true)
 |-- Salário (R$): string (nullable = true)
 |-- Escolaridade: string (nullable = true)
 |-- Nota : string (nullable = true)

```

```

1 '''
2 Neste bloco vamos converter idade em inteiro, matrículaem inteiro, salário em
3 double e nota em Double
4 '''
5 from pyspark.sql.types import *
6 from pyspark.sql.types import StructType,StructField,StringType,IntegerType,DoubleType
7 df = df \
8     .withColumn("Matrícula",
9                 df["Matrícula"]
10                    .cast(IntegerType())) \
11     .withColumn("Idade",
12                 df["Idade"]
13                    .cast(IntegerType())) \
14     .withColumn("Salário (R$)",
15                 df["Salário (R$)"]
16                    .cast(DoubleType())) \
17     .withColumn("Nota ",
18                 df["Nota "]
19                    .cast(DoubleType()))
20 df.printSchema()

```

```

root
 |-- Matrícula: integer (nullable = true)
 |-- Nome: string (nullable = true)
 |-- Cidade: string (nullable = true)
 |-- Estado: string (nullable = true)
 |-- País: string (nullable = true)
 |-- Idade: integer (nullable = true)
 |-- Departamento: string (nullable = true)
 |-- Cargo: string (nullable = true)

```

```

|-- Salário (R$): double (nullable = true)
|-- Escolaridade: string (nullable = true)
|-- Nota : double (nullable = true)

```

```
1 df.show()
```

	Matrícula	Nome	Cidade	Estado	País	Idade	Departamento	Cargo	Salário (R\$)	Escolaridade	Nota
	1	Lucas	Atibaia	SP	Brasil	35	Compras	Gerente	25000.0	Superior	8.0
	2	Ana	São Paulo	SP	Brasil	29	Vendas	Coordenador	12000.0	Superior	6.0
	3	Luiza	Santos	SP	Brasil	38	Finanças	Gerente	28000.0	MBA	9.0
	4	Fernando	Atibaia	SP	Brasil	36	Marketing	Diretor	40000.0	Mestrado	7.0
	5	Sandra	Atibaia	SP	Brasil	28	Produção	Analista	23000.0	Superior	5.0
	6	Douglas	Bragança	SP	Brasil	29	Finanças	Analista	11000.0	Superior	9.0
	7	Eduardo	Extrema	MG	Brasil	30	Marketing	Gerente	12000.0	MBA	4.0
	8	Ester	Itapeva	MG	Brasil	29	Compras	Analista	10000.0	Superior	2.0
	9	Pedro	Extrema	MG	Brasil	30	Marketing	Analista	13000.0	Superior	1.0
	10	Maria	Extrema	MG	Brasil	40	Produção	Analista	12000.0	MBA	7.0

```

1 '''
2 Agora você vai aplicar seus conhecimentos de Pyspark para para as seguintes
3 transformações no DF:
4
5 1 - Renomear todas as colunas de forma que se mantenha os nomes, mas apenas com
6 letras minúsculas sem espaços, e/ou caracteres especiais
7
8 2 - Fazer livremente no mínimo 10 filtros que tragam recortes a partir de colunas
9 diferentes
10
11 3 - Considerando que todos os funcionários tem um desconto de previdência complementar
12 de 10 % do salário Bruto crie e calcule os valores de uma nova coluna chamada
13 previdencia.
14
15 4 - Considerando que todos os funcionários nesse mês específico terão uma bonificação
16 de 20% sobre o salário bruto, crie e calcule os valores de uma nova coluna chamada
17 bonus
18
19 5 - Crie uma nova coluna com todos os nomes dos funcionários em letras maiúsculas
20
21 6 - Crie uma nova coluna com os nomes dos funcionários todos em letras minúsculas
22
23 7 - Crie um novo dataframe chamado quant apenas com as colunas idade,
24 salário e notas
25
26 8 - calcule as principais medidas descritivas do df quant (describe)
27
28 9 - Ordene o df original por notas de forma crescente
29
30 10 - faça as contagens de frequências, usando groupby para as seguintes colunas:
31 a) Cargo
32 b) Cidade
33 c) Estado
34 d) Escolaridade
35 e) Departamento
36 '''

```

```
1 df.printSchema()
```

```

root
 |-- Matrícula: integer (nullable = true)
 |-- Nome: string (nullable = true)
 |-- Cidade: string (nullable = true)
 |-- Estado: string (nullable = true)
 |-- País: string (nullable = true)
 |-- Idade: integer (nullable = true)
 |-- Departamento: string (nullable = true)
 |-- Cargo: string (nullable = true)
 |-- Salário (R$): double (nullable = true)
 |-- Escolaridade: string (nullable = true)
 |-- Nota : double (nullable = true)

```

```

1 '''
2 1 - Renomear todas as colunas de forma que se mantenha os nomes, mas apenas com
3 letras minúsculas sem espaços, e/ou caracteres especiais
4 '''
5 print("atual: ", df.columns)
6 df = df.toDF(*('matricula','nome','cidade','estado','pais','idade',
7               'departamento', 'cargo', 'salario', 'escolaridade', 'nota'))
8 df.show()

```

```
atual: ['Matrícula', 'Nome', 'Cidade', 'Estado', 'País', 'Idade', 'Departamento', 'Cargo', 'Salário (R$)', 'Escolaridade', 'Nota ']
```

	matricula	nome	cidade	estado	pais	idade	departamento	cargo	salario	escolaridade	nota
	1	Lucas	Atibaia	SP	Brasil	35	Compras	Gerente	25000.0	Superior	8.0
	2	Ana	São Paulo	SP	Brasil	29	Vendas	Coordenador	12000.0	Superior	6.0
	3	Luiza	Santos	SP	Brasil	38	Finanças	Gerente	28000.0	MBA	9.0

4	Fernando	Atibaia	SP	Brasil	36	Marketing	Diretor	40000.0	Mestrado	7.0
5	Sandra	Atibaia	SP	Brasil	28	Produção	Analista	23000.0	Superior	5.0
6	Douglas	Bragança	SP	Brasil	29	Finanças	Analista	11000.0	Superior	9.0
7	Eduardo	Extrema	MG	Brasil	30	Marketing	Gerente	12000.0	MBA	4.0
8	Ester	Itapeva	MG	Brasil	29	Compras	Analista	10000.0	Superior	2.0
9	Pedro	Extrema	MG	Brasil	30	Marketing	Analista	13000.0	Superior	1.0
10	Maria	Extrema	MG	Brasil	40	Produção	Analista	12000.0	MBA	7.0

```

1 '''
2 2 - Fazer livremente no mínimo 10 filtros que tragam recortes a partir de colunas
3 diferentes
4 '''
5 from pyspark.sql.functions import col
6 f1 = df.filter(df['cidade']=='Atibaia')
7 f1.show()
8 f2 = df.filter(df['idade']>=40)
9 f2.show()
10 f3 = df.filter("idade >= 40 AND salario >= 35000")
11 f3.show()
12 f4 = df.filter("idade >= 40 OR salario >= 35000")
13 f4.show()
14 f5 = df.filter(col('nome').startswith('D'))
15 f5.show()
16 f6 = df.select('idade')
17 f6.show()
18 f7 = df.select('cargo')
19 f7.show()
20 f8 = df.select('cidade','estado')
21 f8.show()
22 f9 = df.select('cidade','estado')
23 f9.show()
24 f10 = df.select('escolaridade','salario')
25 f10.show()

```

```

+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|matricula| nome| cidade|estado| pais|idade|departamento| cargo|salario|escolaridade|nota|
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 1| Lucas|Atibaia| SP|Brasil| 35| Compras| Gerente|25000.0| Superior| 8.0|
| 4| Fernando|Atibaia| SP|Brasil| 36| Marketing| Diretor|40000.0| Mestrado| 7.0|
| 5| Sandra|Atibaia| SP|Brasil| 28| Produção|Analista|23000.0| Superior| 5.0|
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+

```

```

+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|matricula| nome| cidade|estado| pais|idade|departamento| cargo|salario|escolaridade|nota|
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 10|Maria|Extrema| MG|Brasil| 40| Produção|Analista|12000.0| MBA| 7.0|
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+

```

```

+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|matricula|nome|cidade|estado|pais|idade|departamento|cargo|salario|escolaridade|nota|
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+

```

```

+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|matricula| nome| cidade|estado| pais|idade|departamento| cargo|salario|escolaridade|nota|
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 4| Fernando|Atibaia| SP|Brasil| 36| Marketing| Diretor|40000.0| Mestrado| 7.0|
| 10| Maria|Extrema| MG|Brasil| 40| Produção|Analista|12000.0| MBA| 7.0|
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+

```

```

+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|matricula| nome| cidade|estado| pais|idade|departamento| cargo|salario|escolaridade|nota|
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 6|Douglas|Bragança| SP|Brasil| 29| Finanças|Analista|11000.0| Superior| 9.0|
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+

```

```

+-----+
|idade|
+-----+
| 35|
| 29|
| 38|
| 36|
| 28|
| 29|
| 30|
| 29|
| 30|
| 40|
+-----+

```

```

+-----+
| cargo|
+-----+
| Gerente|
| Coordenador|
| Gerente|
| Diretor|
| Analista|
| Analista|
| Gerente|
| Analista|

```

```

1 '''
2 3 - Considerando que todos os funcionários tem um desconto de previdência complementar

```


3 de 10 % do salário Bruto crie e calcule os valores de uma nova coluna chamada

4 previdencia.

5 '''

6 df = df.withColumn('previdencia',df['salario']*0.1)

7 df.show()

matricula	nome	cidade	estado	pais	idade	departamento	cargo	salario	escolaridade	nota	previdencia
1	Lucas	Atibaia	SP	Brasil	35	Compras	Gerente	25000.0	Superior	8.0	2500.0
2	Ana	São Paulo	SP	Brasil	29	Vendas	Coordenador	12000.0	Superior	6.0	1200.0
3	Luiza	Santos	SP	Brasil	38	Finanças	Gerente	28000.0	MBA	9.0	2800.0
4	Fernando	Atibaia	SP	Brasil	36	Marketing	Diretor	40000.0	Mestrado	7.0	4000.0
5	Sandra	Atibaia	SP	Brasil	28	Produção	Analista	23000.0	Superior	5.0	2300.0
6	Douglas	Bragança	SP	Brasil	29	Finanças	Analista	11000.0	Superior	9.0	1100.0
7	Eduardo	Extrema	MG	Brasil	30	Marketing	Gerente	12000.0	MBA	4.0	1200.0
8	Ester	Itapeva	MG	Brasil	29	Compras	Analista	10000.0	Superior	2.0	1000.0
9	Pedro	Extrema	MG	Brasil	30	Marketing	Analista	13000.0	Superior	1.0	1300.0
10	Maria	Extrema	MG	Brasil	40	Produção	Analista	12000.0	MBA	7.0	1200.0

1 '''

2 4 - Considerando que todos os funcionários nesse mês específico terão uma bonificação

3 de 20% sobre o salário bruto, crie e calcule os valores de uma nova coluna chamada

4 bonus

5 '''

6 df = df.withColumn('bonus',df['salario']*0.2)

7 df.show()

matricula	nome	cidade	estado	pais	idade	departamento	cargo	salario	escolaridade	nota	previdencia	bonus
1	Lucas	Atibaia	SP	Brasil	35	Compras	Gerente	25000.0	Superior	8.0	2500.0	5000.0
2	Ana	São Paulo	SP	Brasil	29	Vendas	Coordenador	12000.0	Superior	6.0	1200.0	2400.0
3	Luiza	Santos	SP	Brasil	38	Finanças	Gerente	28000.0	MBA	9.0	2800.0	5600.0
4	Fernando	Atibaia	SP	Brasil	36	Marketing	Diretor	40000.0	Mestrado	7.0	4000.0	8000.0
5	Sandra	Atibaia	SP	Brasil	28	Produção	Analista	23000.0	Superior	5.0	2300.0	4600.0
6	Douglas	Bragança	SP	Brasil	29	Finanças	Analista	11000.0	Superior	9.0	1100.0	2200.0
7	Eduardo	Extrema	MG	Brasil	30	Marketing	Gerente	12000.0	MBA	4.0	1200.0	2400.0
8	Ester	Itapeva	MG	Brasil	29	Compras	Analista	10000.0	Superior	2.0	1000.0	2000.0
9	Pedro	Extrema	MG	Brasil	30	Marketing	Analista	13000.0	Superior	1.0	1300.0	2600.0
10	Maria	Extrema	MG	Brasil	40	Produção	Analista	12000.0	MBA	7.0	1200.0	2400.0

1 '''

2 5 - Crie uma nova coluna com todos os nomes dos funcionários em letras maiúsculas

3 '''

4 from pyspark.sql.functions import *

5 df = df.withColumn('maiu',upper(col('nome')))

6 df.show()

matricula	nome	cidade	estado	pais	idade	departamento	cargo	salario	escolaridade	nota	previdencia	bonus	maiu
1	Lucas	Atibaia	SP	Brasil	35	Compras	Gerente	25000.0	Superior	8.0	2500.0	5000.0	LUCAS
2	Ana	São Paulo	SP	Brasil	29	Vendas	Coordenador	12000.0	Superior	6.0	1200.0	2400.0	ANA
3	Luiza	Santos	SP	Brasil	38	Finanças	Gerente	28000.0	MBA	9.0	2800.0	5600.0	LUIZA
4	Fernando	Atibaia	SP	Brasil	36	Marketing	Diretor	40000.0	Mestrado	7.0	4000.0	8000.0	FERNANDO
5	Sandra	Atibaia	SP	Brasil	28	Produção	Analista	23000.0	Superior	5.0	2300.0	4600.0	SANDRA
6	Douglas	Bragança	SP	Brasil	29	Finanças	Analista	11000.0	Superior	9.0	1100.0	2200.0	DOUGLAS
7	Eduardo	Extrema	MG	Brasil	30	Marketing	Gerente	12000.0	MBA	4.0	1200.0	2400.0	EDUARDO
8	Ester	Itapeva	MG	Brasil	29	Compras	Analista	10000.0	Superior	2.0	1000.0	2000.0	ESTER
9	Pedro	Extrema	MG	Brasil	30	Marketing	Analista	13000.0	Superior	1.0	1300.0	2600.0	PEDRO
10	Maria	Extrema	MG	Brasil	40	Produção	Analista	12000.0	MBA	7.0	1200.0	2400.0	MARIA

1 '''

2 6 - Crie uma nova coluna com os nomes dos funcionários todos em letras minúsculas

3 '''

4 df = df.withColumn('minu',lower(col('nome')))

5 df.show()

matricula	nome	cidade	estado	pais	idade	departamento	cargo	salario	escolaridade	nota	previdencia	bonus	maiu	minu
1	Lucas	Atibaia	SP	Brasil	35	Compras	Gerente	25000.0	Superior	8.0	2500.0	5000.0	LUCAS	lucas
2	Ana	São Paulo	SP	Brasil	29	Vendas	Coordenador	12000.0	Superior	6.0	1200.0	2400.0	ANA	ana
3	Luiza	Santos	SP	Brasil	38	Finanças	Gerente	28000.0	MBA	9.0	2800.0	5600.0	LUIZA	luiza
4	Fernando	Atibaia	SP	Brasil	36	Marketing	Diretor	40000.0	Mestrado	7.0	4000.0	8000.0	FERNANDO	fernando
5	Sandra	Atibaia	SP	Brasil	28	Produção	Analista	23000.0	Superior	5.0	2300.0	4600.0	SANDRA	sandra
6	Douglas	Bragança	SP	Brasil	29	Finanças	Analista	11000.0	Superior	9.0	1100.0	2200.0	DOUGLAS	douglas
7	Eduardo	Extrema	MG	Brasil	30	Marketing	Gerente	12000.0	MBA	4.0	1200.0	2400.0	EDUARDO	eduardo
8	Ester	Itapeva	MG	Brasil	29	Compras	Analista	10000.0	Superior	2.0	1000.0	2000.0	ESTER	ester
9	Pedro	Extrema	MG	Brasil	30	Marketing	Analista	13000.0	Superior	1.0	1300.0	2600.0	PEDRO	pedro
10	Maria	Extrema	MG	Brasil	40	Produção	Analista	12000.0	MBA	7.0	1200.0	2400.0	MARIA	maria

```

1 '''
2 7 - Crie um novo dataframe chamado quant apenas com as colunas idade,
3 salário e notas
4 '''
5 quanti = df.select('idade','salario','nota')
6 quanti.show()

```

```

+-----+-----+-----+
| idade|salario|nota|
+-----+-----+-----+
| 35|25000.0| 8.0|
| 29|12000.0| 6.0|
| 38|28000.0| 9.0|
| 36|40000.0| 7.0|
| 28|23000.0| 5.0|
| 29|11000.0| 9.0|
| 30|12000.0| 4.0|
| 29|10000.0| 2.0|
| 30|13000.0| 1.0|
| 40|12000.0| 7.0|
+-----+-----+-----+

```

```

1 '''
2 8 - calcule as principais medidas descritivas do df quant (describe)
3 '''
4 quanti.describe().show()

```

```

+-----+-----+-----+-----+
| summary|      idade|      salario|      nota|
+-----+-----+-----+-----+
| count|         10|          10|         10|
| mean|        32.4|      18600.0|         5.8|
| stddev|4.402019738458447|10002.221975363496|2.780887148615228|
| min|         28|      10000.0|         1.0|
| max|         40|      40000.0|         9.0|
+-----+-----+-----+-----+

```

```

1 '''
2 9 - Ordene o df original por notas de forma crescente
3 '''
4 df.orderBy('nota')

```

matricula	nome	cidade	estado	pais	idade	departamento	cargo	salario	escolaridade	nota	previdencia	bonus	maiu	minu
9	Pedro	Extrema	MG	Brasil	30	Marketing	Analista	13000.0	Superior	1.0	1300.0	2600.0	PEDRO	pedro
8	Ester	Itapeva	MG	Brasil	29	Compras	Analista	10000.0	Superior	2.0	1000.0	2000.0	ESTER	ester
7	Eduardo	Extrema	MG	Brasil	30	Marketing	Gerente	12000.0	MBA	4.0	1200.0	2400.0	EDUARDO	eduardo
5	Sandra	Atibaia	SP	Brasil	28	Produção	Analista	23000.0	Superior	5.0	2300.0	4600.0	SANDRA	sandra
2	Ana	São Paulo	SP	Brasil	29	Vendas	Coordenador	12000.0	Superior	6.0	1200.0	2400.0	ANA	ana
10	Maria	Extrema	MG	Brasil	40	Produção	Analista	12000.0	MBA	7.0	1200.0	2400.0	MARIA	maria
4	Fernando	Atibaia	SP	Brasil	36	Marketing	Diretor	40000.0	Mestrado	7.0	4000.0	8000.0	FERNANDO	fernando
1	Lucas	Atibaia	SP	Brasil	35	Compras	Gerente	25000.0	Superior	8.0	2500.0	5000.0	LUCAS	lucas
3	Luiza	Santos	SP	Brasil	38	Finanças	Gerente	28000.0	MBA	9.0	2800.0	5600.0	LUIZA	luiza
6	Douglas	Bragança	SP	Brasil	29	Finanças	Analista	11000.0	Superior	9.0	1100.0	2200.0	DOUGLAS	douglas

```

1 '''
2 10 - faça as contagens de frequências, usando groupby para as seguintes colunas:
3 a) Cargo
4 b) Cidade
5 c) Estado
6 d) Escolaridade
7 e) Departamento
8
9 '''
10 a = df.groupBy('cargo').agg({'cargo':'count'})
11 a.show()
12 b = df.groupBy('cidade').agg({'cidade':'count'})
13 b.show()
14 c = df.groupBy('estado').agg({'estado':'count'})
15 c.show()
16 d = df.groupBy('escolaridade').agg({'escolaridade':'count'})
17 d.show()
18 e = df.groupBy('departamento').agg({'departamento':'count'})
19 e.show()

```

```

+-----+-----+
| cargo|count(cargo)|
+-----+-----+
| Gerente|          3|
| Coordenador|        1|
| Diretor|         1|
| Analista|         5|
+-----+-----+

+-----+-----+
| cidade|count(cidade)|
+-----+-----+
| Santos|          1|
| Extrema|         3|
| São Paulo|        1|
+-----+-----+

```

	Atibaia	3
	Bragança	1
	Itapeva	1
+-----+		

+-----+		
	estado	count(estado)
+-----+		
	SP	6
	MG	4
+-----+		

+-----+		
	escolaridade	count(escolaridade)
+-----+		
	Superior	6
	Mestrado	1
	MBA	3
+-----+		

+-----+		
	departamento	count(departamento)
+-----+		
	Vendas	1
	Finanças	2
	Marketing	3
	Compras	2
	Produção	2
+-----+		