



(https://databricks.com) Importando as bibliotecas

```
from pyspark.sql.functions import *
from pyspark.sql.functions import regexp_replace
```

Ponto de montagem camada bronze

```
dbutils.fs.mount(
    source = "wasbs://bronze@storageenem2000.blob.core.windows.net",
    mount_point = "/mnt/storageenem2000/bronze",
    extra_configs = {"fs.azure.account.key.storageenem2000.blob.core.windows.net":"ZP80pIRUCphDD5JuILTx1cIL6+yyLJXizbNHuLaylDipkV8EC5I"})

java.rmi.RemoteException: java.lang.IllegalArgumentException: requirement failed: Directory already mounted: /mnt/storageenem2000/bronze; nested exception is:
```

Ponto de montagem camada Silver

```
dbutils.fs.mount(
    source = "wasbs://silver@storageenem2000.blob.core.windows.net",
    mount_point = "/mnt/storageenem2000/silver",
    extra_configs = {"fs.azure.account.key.storageenem2000.blob.core.windows.net":"ZP80pIRUCphDD5JuILTx1cIL6+yyLJXizbNHuLaylDipkV8EC5I"})

Command skipped
```

Ponto de montagem camada Gold

```
dbutils.fs.mount(
    source = "wasbs://gold@storageenem2000.blob.core.windows.net",
    mount_point = "/mnt/storageenem2000/gold",
    extra_configs =
    {"fs.azure.account.key.storageenem2000.blob.core.windows.net":"ZP80pIRUCphDD5JuILTx1cIL6+yyLJXizbNHuLaylDipkV8EC5CVEGJzQhADqfKCSHVHCOTd0MpH+AStrHWSJg=="})

Command skipped
```

Criando um database

```
spark.sql("CREATE DATABASE IF NOT EXISTS enim")

Command skipped
```

Lendo o arquivo csv da camada bronze

```
df_enem_2022_bronze = spark.read.format('csv')\
    .options(header='true', infer_schema='true', delimiter=',')\
    .load('dbfs:/mnt/storageenem2000/bronze/MICRODADOS_ENEM_2022.csv')

df_enem_2021_bronze = spark.read.format('csv')\
    .options(header='true', infer_schema='true', delimiter=',')\
    .load('dbfs:/mnt/storageenem2000/bronze/MICRODADOS_ENEM_2021.csv')
```

Instanciando um daframe onde não existem valores nulos na nota da redação

```
#dataframe enem 2022
df_enem_2022_silver = df_enem_2022_bronze.filter(df_enem_2022_bronze.NU_NOTA_REDACAO.isNotNull())

#dataframe enem 2021
df_enem_2021_silver = df_enem_2021_bronze.filter(df_enem_2021_bronze.NU_NOTA_REDACAO.isNotNull())
```

Criando arquivo parquet

```
#Criando arquivo parquet no dir
df_enem_2021_silver.write.format('delta')\
    .mode('overwrite')\
    .save('/mnt/storageenem2000/silver/enem_2021')

df_enem_2022_silver.write.format('delta')\
    .mode('overwrite')\
    .save('/mnt/storageenem2000/silver/enem_2022')
```

Criando tabelas no database "enem"

```
#Usar database
spark.sql('USE enem')

#Criar tabela enem_2022
spark.sql("""
CREATE TABLE IF NOT EXISTS enem_2022
USING DELTA
LOCATION '/mnt/storageenem2000/silver/enem_2022'
""")

#Criar tabela enem_2021
spark.sql("""
CREATE TABLE IF NOT EXISTS enem_2021
USING DELTA
LOCATION '/mnt/storageenem2000/silver/enem_2021'
""")
```

DataFrame[]

```
#Query

df_resultado = spark.sql('SELECT * FROM enem_2021')

display(df_resultado)
```

Table								
	NU_INSCRIÇÃO	NU_ANO	TP_FAIXA_ETARIA	TP_SEXO	TP_ESTADO_CIVIL	TP_COR_RACA	TP_NACIONALIDA	
1	210054259052	2021	2	M	1	0	1	
2	210051551521	2021	1	M	1	3	1	
3	210051795446	2021	3	F	1	1	2	
4	210051313205	2021	6	M	1	2	1	
5	210052826675	2021	2	F	1	3	1	
6	210054392385	2021	3	F	1	2	1	
7	210054150052	2021	7	M	1	3	1	

2,783 rows | Truncated data

Camada Gold

```
#Camada gold
from pyspark.sql.functions import concat,to_date, year
```

Lendo os arquivo parquet

```
df_enem_2022_silver = spark.read.format('delta')\
    .load('/mnt/storageenem2000/silver/enem_2022')

df_enem_2021_silver = spark.read.format('delta')\
    .load('/mnt/storageenem2000/silver/enem_2021')
```

```
from pyspark.sql.functions import concat,to_date, year

# Transformar o type da coluna "NU_ANO"

df_enem_2022_silver = df_enem_2022_silver.withColumn("NU_ANO", to_date(df_enem_2022_silver['NU_ANO']))
```

Renomeando as colunas

```
df_enem_gold = df_enem_2022_silver.withColumnRenamed("NU_INSCRICAO","INSCRICAO")\
    .withColumnRenamed("TP_FAIXA_ETARIA","FAIXA_ETARIA")
```

```
df_enem_gold = df_enem_gold.withColumn('YEAR',year(df_enem_gold['NU_ANO']))
```

Modificando o type da coluna

```
from pyspark.sql.functions import col

df_enem_gold = df_enem_gold.withColumn('NU_ANO',col('NU_ANO').cast("integer"))
```

```
# Gravar os dados no Delta Lake, particionados por 'ano'
df_enem_gold.write.format("delta")\
    .mode('overwrite')\
    .option("overwriteSchema", "true")\
    .partitionBy("YEAR")\
    .save("/mnt/storageenem2000/gold/enem_full")
```

```
# Use enim database
spark.sql('USE enim')

#Create table externa

spark.sql("""
DROP TABLE IF EXISTS enim_gold """)

spark.sql("""
CREATE TABLE IF NOT EXISTS enim_gold
USING DELTA
LOCATION '/mnt/storageenem2000/gold/enem_full'
""")
```

DataFrame[]

```
# Query
df_enem_gold = spark.sql("SELECT * FROM enim_gold")

display(df_enem_gold)
```

Table

	INSCRIÇÃO	NU_ANO	FAIXA_ETARIA	TP_SEXO	TP_ESTADO_CIVIL	TP_COR_RACA	TP_NACIONALIDADE
1	210055590133	null	3	F	1	4	1
2	210056186398	null	2	F	1	3	1
3	210054775303	null	10	M	1	2	1
4	210054964335	null	2	M	1	2	1
5	210057560475	null	3	F	1	3	1
6	210054987607	null	1	F	1	2	1
7	210057167105	null	3	F	1	3	1

2,785 rows | Truncated data

Criando select agregado

```
#Criar table agregada
from pyspark.sql.functions import count

#Selecionando as colunas que irei trabalhar
select_columns = ["YEAR","FAIXA_ETARIA","TP_SEXO"]

#Criando um dataframe somente com as colunas especificas mencionado acima
df_select_columns_gold = df_enem_gold.select(select_columns)

#agrupando as colunas selecionada e realizando a contagem
grouped_df = df_select_columns_gold.groupBy(select_columns).agg(count("*").alias("count"))
```

```

grouped_df = grouped_df.withColumn("FAIXA_ETARIA", regexp_replace("FAIXA_ETARIA", "^1$","Menor de 17 anos"))

grouped_df = grouped_df.withColumn("FAIXA_ETARIA", regexp_replace("FAIXA_ETARIA", "^2$","17 anos"))

grouped_df = grouped_df.withColumn("FAIXA_ETARIA", regexp_replace("FAIXA_ETARIA", "^3$","18 anos"))

grouped_df = grouped_df.withColumn("FAIXA_ETARIA", regexp_replace("FAIXA_ETARIA", "^4$","19 anos"))

grouped_df = grouped_df.withColumn("FAIXA_ETARIA", regexp_replace("FAIXA_ETARIA", "^5$","20 anos"))

grouped_df = grouped_df.withColumn("FAIXA_ETARIA", regexp_replace("FAIXA_ETARIA", "^6$","21 anos"))

grouped_df = grouped_df.withColumn("FAIXA_ETARIA", regexp_replace("FAIXA_ETARIA", "^7$","22 anos"))

grouped_df = grouped_df.withColumn("FAIXA_ETARIA", regexp_replace("FAIXA_ETARIA", "^8$","23 anos"))

grouped_df = grouped_df.withColumn("FAIXA_ETARIA", regexp_replace("FAIXA_ETARIA", "^9$","24 anos"))

grouped_df = grouped_df.withColumn("FAIXA_ETARIA", regexp_replace("FAIXA_ETARIA", "^10$","25 anos"))

grouped_df = grouped_df.withColumn("FAIXA_ETARIA", regexp_replace("FAIXA_ETARIA", "^11$","Entre 26 e 30 anos"))

grouped_df = grouped_df.withColumn("FAIXA_ETARIA", regexp_replace("FAIXA_ETARIA", "^12$","Entre 31 e 35 anos"))

grouped_df = grouped_df.withColumn("FAIXA_ETARIA", regexp_replace("FAIXA_ETARIA", "^13$","Entre 36 e 40 anos"))

grouped_df = grouped_df.withColumn("FAIXA_ETARIA", regexp_replace("FAIXA_ETARIA", "^14$","Entre 41 e 45 anos"))

grouped_df = grouped_df.withColumn("FAIXA_ETARIA", regexp_replace("FAIXA_ETARIA", "^15$","Entre 46 e 50 anos"))

grouped_df = grouped_df.withColumn("FAIXA_ETARIA", regexp_replace("FAIXA_ETARIA", "^16$","Entre 51 e 55 anos"))

grouped_df = grouped_df.withColumn("FAIXA_ETARIA", regexp_replace("FAIXA_ETARIA", "^17$","Entre 56 e 60 anos"))

grouped_df = grouped_df.withColumn("FAIXA_ETARIA", regexp_replace("FAIXA_ETARIA", "^18$","Entre 61 e 65 anos"))

grouped_df = grouped_df.withColumn("FAIXA_ETARIA", regexp_replace("FAIXA_ETARIA", "^19$","Entre 66 e 70 anos"))

grouped_df = grouped_df.withColumn("FAIXA_ETARIA", regexp_replace("FAIXA_ETARIA", "^20$","Maior 70"))

display(grouped_df)

```

Table Visualization 1

	YEAR	FAIXA_ETARIA	TP_SEXO	count	
1	2022	17 anos	F	377011	
2	2022	Entre 51 e 55 anos	M	4097	
3	2022	Menor de 17 anos	F	177669	
4	2022	21 anos	M	40787	
5	2022	Entre 31 e 35 anos	M	20732	
6	2022	Entre 61 e 65 anos	F	1082	
7	2022	24 anos	M	16253	

40 rows

Salvando no container

```

df_enem_gold.write.format("delta")\
.mode('overwrite')\
.option("overwriteSchema", "true")\
.partitionBy("YEAR")\
.save("/mnt/storageenem2000/gold/enem_faixa_etaria_agg")

```

Criando uma tabela apartir aqui delta

```
# Use enem database

spark.sql('USE enem')

#Create table externa

spark.sql("""
DROP TABLE IF EXISTS enem_faixa_etaria_agg """)

spark.sql("""
CREATE TABLE IF NOT EXISTS enem_faixa_etaria_agg
USING DELTA
LOCATION '/mnt/storageenem2000/gold/enem_faixa_etaria_agg'
""")
```

DataFrame[]

```
#Criar table agregada

from pyspark.sql.functions import count

#Selecionando as colunas que irei trabalhar
select_columns = ["YEAR","FAIXA_ETARIA","TP_SEXO","TP_ST_CONCLUSAO","TP_ESCOLA","NO_MUNICIPIO_PROVA","SG_UF_PROVA"]

#Criando um dataframe somente com as colunas especificas mencionado acima
df_select_columns_gold_2 = df_enem_gold.select(select_columns)

#agrupando as colunas selecionada e realizando a contagem
grouped_df_02 = df_select_columns_gold_2.groupBy(select_columns).agg(count("*").alias("count"))
```

display(grouped_df_02)

Table Visualization 1								
	YEAR	FAIXA_ETARIA	TP_SEXO	TP_ST_CONCLUSAO	TP_ESCOLA	NO_MUNICIPIO_PROVA	SG_UF	
1	2022	2	M	2	3	Recife	PE	
2	2022	6	F	1	1	Canarana	BA	
3	2022	3	F	1	1	Ouricuri	PE	
4	2022	5	F	2	2	Bom Jesus	PI	
5	2022	6	F	1	1	Lagoa Santa	MG	
6	2022	3	M	2	2	Goiania	GO	
7	2022	11	F	1	1	Curitiba	PR	

10,000 rows | Truncated data

Modificando os nomes das colunas

```
grouped_df_02 = grouped_df_02.withColumnRenamed("YEAR", "ANO")
grouped_df_02 = grouped_df_02.withColumnRenamed("TP_SEXO", "SEXO")
grouped_df_02 = grouped_df_02.withColumnRenamed("TP_ST_CONCLUSAO", "ANO_CONCLUSAO")
grouped_df_02 = grouped_df_02.withColumnRenamed("TP_ESCOLA", "TIPO_ESCOLA")
grouped_df_02 = grouped_df_02.withColumnRenamed("NO_MUNICIPIO_PROVA", "MUNICIPIO_PROVA")
grouped_df_02 = grouped_df_02.withColumnRenamed("SG_UF_PROVA", "UF_PROVA")
grouped_df_02 = grouped_df_02.withColumnRenamed("count", "TOTAL")
```

Modificando os valores

```
grouped_df_02 = grouped_df_02.withColumn("TIPO_ESCOLA", regexp_replace("TIPO_ESCOLA", "^1$","Não Respondeu"))

grouped_df_02 = grouped_df_02.withColumn("TIPO_ESCOLA", regexp_replace("TIPO_ESCOLA", "^2$","Pública"))

grouped_df_02 = grouped_df_02.withColumn("TIPO_ESCOLA", regexp_replace("TIPO_ESCOLA", "^3$","Privada"))
```

```
#Exibindo o dataframe
display(grouped_df_02)
```

Table Visualization 1

	ANO	FAIXA_ETARIA	SEXO	ANO_CONCLUSAO	TIPO_ESCOLA	MUNICIPIO_PROVA	UF_I
1	2022	2	M	2	Privada	Recife	PE
2	2022	6	F	1	Não Respondeu	Canarana	BA
3	2022	3	F	1	Não Respondeu	Ouricuri	PE
4	2022	5	F	2	Pública	Bom Jesus	PI
5	2022	6	F	1	Não Respondeu	Lagoa Santa	MG
6	2022	3	M	2	Pública	Goiania	GO
7	2022	11	F	1	Não Respondeu	Curitiba	PR

10,000 rows | Truncated data

```
display(df_enem_gold)
```

Table

	INSCRICAO	NU_ANO	FAIXA_ETARIA	TP_SEXO	TP_ESTADO_CIVIL	TP_COR_RACA	TP_NACIONALIDADE
1	210055590133	null	3	F	1	4	1
2	210056186398	null	2	F	1	3	1
3	210054775303	null	10	M	1	2	1
4	210054964335	null	2	M	1	2	1
5	210057560475	null	3	F	1	3	1
6	210054987607	null	1	F	1	2	1
7	210057167105	null	3	F	1	3	1

2,785 rows | Truncated data

```
select_columns = ["NU_ANO","FAIXA_ETARIA","TP_SEXO","TP_ST_CONCLUSAO","TP_ESCOLA","CO_MUNICIPIO_PROVA","SG_UF_PROVA"]

#Criando um dataframe somente com as colunas especificas mencionado acima
df_select_columns_gold_2 = df_enem_gold.select(select_columns)

#agrupando as colunas selecionada e realizando a contagem
grouped_df_02 = df_select_columns_gold_2.groupBy(select_columns).agg(count("*").alias("count"))
```

Modificando o nome das colunas

```
select_columns =
['INSCRIÇÃO', 'NU_NOTA_CN', 'NU_NOTA_CH', 'NU_NOTA_LC', 'NU_NOTA_MT', 'NU_NOTA_REDACAO', "FAIXA_ETARIA", "SG_UF_PROVA"]

#Criando um dataframe somente com as colunas selecionadas

df_notas_gold = df_enem_gold.select(select_columns)

df_notas_gold = df_notas_gold.withColumnRenamed("SG_UF_PROVA", "UF_PROVA")

df_notas_gold = df_notas_gold.withColumnRenamed('NU_NOTA_CN', 'NOTA_CIENCIA_NATUREZA')

df_notas_gold = df_notas_gold.withColumnRenamed('NU_NOTA_CH', 'NOTA_CIENCIAS_HUMANAS')

df_notas_gold = df_notas_gold.withColumnRenamed('NU_NOTA_LC', 'NOTA_LINGUAGEM_CODIGOS')

df_notas_gold = df_notas_gold.withColumnRenamed('NU_NOTA_MT', 'NOTA_MATEMATICA')

df_notas_gold = df_notas_gold.withColumnRenamed('NU_NOTA_REDACAO', 'NOTA_REDACAO')

display(df_notas_gold)
```

Table

	INSCRIÇÃO	NOTA_CIENCIA_NATUREZA	NOTA_CIENCIAS_HUMANAS	NOTA_LINGUAGEM_CODIGOS	NOTA_MATEMATICA
1	210055590133	458.8	491.6	445.3	532.8
2	210056186398	414.4	426.9	431.6	366.6
3	210054775303	601.7	662	606.5	688.2
4	210054964335	470.7	531	372.1	529.1
5	210057560475	438.9	503	478.9	469.4
6	210054987607	401.9	432.1	405.7	491
7	210057167105	449.3	484.3	524.7	461.8

10,000 rows | Truncated data

Removendo os valores nulos dos atributos

```
#Selecionando as colunas
cols =
['INSCRIÇÃO', 'NOTA_CIENCIA_NATUREZA', 'NOTA_CIENCIAS_HUMANAS', 'NOTA_LINGUAGEM_CODIGOS', 'NOTA_MATEMATICA', 'NOTA_REDACAO']

#Instancionando um dataframe sem o nulos
df_notas_semnulo_gold = df_notas_gold.na.drop(subset=cols)

#Exibindo o dataframe
display(df_notas_semnulo_gold)
```

Table

	INSCRIÇÃO	NOTA_CIENCIA_NATUREZA	NOTA_CIENCIAS_HUMANAS	NOTA_LINGUAGEM_CODIGOS	NOTA_MATEMATICA
1	210055590133	458.8	491.6	445.3	532.8
2	210056186398	414.4	426.9	431.6	366.6
3	210054775303	601.7	662	606.5	688.2
4	210054964335	470.7	531	372.1	529.1
5	210057560475	438.9	503	478.9	469.4
6	210054987607	401.9	432.1	405.7	491
7	210057167105	449.3	484.3	524.7	461.8

10,000 rows | Truncated data

```
#Modificando o tipo da coluna para inteiro,
#porque está dando ruim quando mudo uma string
#Ex: Quando mudo o numero 1 ele tbm faz alteração no numeros que contem 1x , x1...

from pyspark.sql.functions import col

df_enem_gold = df_enem_gold.withColumn('FAIXA_ETARIA', col('FAIXA_ETARIA').cast("integer"))
```

Especificando as Faixa Etaria

#Neste código, estamos usando o padrão ^1\$, onde ^ representa o início da linha e \$ representa o final da linha. Portanto, estamos garantindo que "1" seja #substituído apenas quando for uma correspondência exata na coluna "FAIXA_ETARIA". Isso resolverá o problema de substituição incorreta quando "1" está #contido em outros números, como "10".

```
from pyspark.sql.functions import regexp_replace

df_notas_semnulo_gold = df_notas_semnulo_gold.withColumn("FAIXA_ETARIA", regexp_replace("FAIXA_ETARIA", "^1$","Menor de 17 anos"))

df_notas_semnulo_gold = df_notas_semnulo_gold.withColumn("FAIXA_ETARIA", regexp_replace("FAIXA_ETARIA", "^2$","17 anos"))

df_notas_semnulo_gold = df_notas_semnulo_gold.withColumn("FAIXA_ETARIA", regexp_replace("FAIXA_ETARIA", "^3$","18 anos"))

df_notas_semnulo_gold = df_notas_semnulo_gold.withColumn("FAIXA_ETARIA", regexp_replace("FAIXA_ETARIA", "^4$","19 anos"))

df_notas_semnulo_gold = df_notas_semnulo_gold.withColumn("FAIXA_ETARIA", regexp_replace("FAIXA_ETARIA", "^5$","20 anos"))

df_notas_semnulo_gold = df_notas_semnulo_gold.withColumn("FAIXA_ETARIA", regexp_replace("FAIXA_ETARIA", "^6$","21 anos"))

df_notas_semnulo_gold = df_notas_semnulo_gold.withColumn("FAIXA_ETARIA", regexp_replace("FAIXA_ETARIA", "^7$","22 anos"))

df_notas_semnulo_gold = df_notas_semnulo_gold.withColumn("FAIXA_ETARIA", regexp_replace("FAIXA_ETARIA", "^8$","23 anos"))

df_notas_semnulo_gold = df_notas_semnulo_gold.withColumn("FAIXA_ETARIA", regexp_replace("FAIXA_ETARIA", "^9$","24 anos"))

df_notas_semnulo_gold = df_notas_semnulo_gold.withColumn("FAIXA_ETARIA", regexp_replace("FAIXA_ETARIA", "^10$","25 anos"))

df_notas_semnulo_gold = df_notas_semnulo_gold.withColumn("FAIXA_ETARIA", regexp_replace("FAIXA_ETARIA", "^11$","Entre 26 e 30 anos"))

df_notas_semnulo_gold = df_notas_semnulo_gold.withColumn("FAIXA_ETARIA", regexp_replace("FAIXA_ETARIA", "^12$","Entre 31 e 35 anos"))

df_notas_semnulo_gold = df_notas_semnulo_gold.withColumn("FAIXA_ETARIA", regexp_replace("FAIXA_ETARIA", "^13$","Entre 36 e 40 anos"))

df_notas_semnulo_gold = df_notas_semnulo_gold.withColumn("FAIXA_ETARIA", regexp_replace("FAIXA_ETARIA", "^14$","Entre 41 e 45 anos"))

df_notas_semnulo_gold = df_notas_semnulo_gold.withColumn("FAIXA_ETARIA", regexp_replace("FAIXA_ETARIA", "^15$","Entre 46 e 50 anos"))

df_notas_semnulo_gold = df_notas_semnulo_gold.withColumn("FAIXA_ETARIA", regexp_replace("FAIXA_ETARIA", "^16$","Entre 51 e 55 anos"))

df_notas_semnulo_gold = df_notas_semnulo_gold.withColumn("FAIXA_ETARIA", regexp_replace("FAIXA_ETARIA", "^17$","Entre 56 e 60 anos"))

df_notas_semnulo_gold = df_notas_semnulo_gold.withColumn("FAIXA_ETARIA", regexp_replace("FAIXA_ETARIA", "^18$","Entre 61 e 65 anos"))

df_notas_semnulo_gold = df_notas_semnulo_gold.withColumn("FAIXA_ETARIA", regexp_replace("FAIXA_ETARIA", "^19$","Entre 66 e 70 anos"))

df_notas_semnulo_gold = df_notas_semnulo_gold.withColumn("FAIXA_ETARIA", regexp_replace("FAIXA_ETARIA", "^20$","Maior 70"))
```

```
display(df_notas_semnulo_gold)
```

Table Visualization 1

	INSCRIÇÃO	NOTA_CIENCIA_NATUREZA	NOTA_CIENCIAS_HUMANAS	NOTA_LINGUAGEM_CODIGOS	NOTA_MATEMATICA
1	210055590133	458.8	491.6	445.3	532.8
2	210056186398	414.4	426.9	431.6	366.6
3	210054775303	601.7	662	606.5	688.2
4	210054964335	470.7	531	372.1	529.1
5	210057560475	438.9	503	478.9	469.4
6	210054987607	401.9	432.1	405.7	491
7	210057167105	449.3	484.3	524.7	461.8

10,000 rows | Truncated data

Media das notas por estado

```
df_media_notas_estado = df_notas_semnulo_gold.groupBy("UF_PROVA").agg(
    round(avg(col("NOTA_CIENCIA_NATUREZA")),2).alias("MEDIA_CIENCIA_NATUREZA"),
    round(avg(col("NOTA_CIENCIAS_HUMANAS")),2).alias("MEDIA_CIENCIA_HUMANAS"),
    round(avg(col("NOTA_LINGUAGEM_CODIGOS")),2).alias("MEDIA_LINGUAGEM_CODIGOS"),
    round(avg(col("NOTA_MATEMATICA")),2).alias("MEDIA_MATEMATICA"),
    round(avg(col("NOTA_REDACAO")),2).alias("MEDIA_REDACAO")
)

display(df_media_notas_estado)
```

Table Visualization 1

	UF_PROVA	MEDIA_CIENCIA_NATUREZA	MEDIA_CIENCIA_HUMANAS	MEDIA_LINGUAGEM_CODIGOS	MEDIA_MATEMATICA
1	SC	509.89	546.3	537.4	567.55
2	RO	480.78	511.63	500.53	514.92
3	PI	481.32	508.7	494.87	519.55
4	AM	473.93	501.22	491.69	496.22
5	RR	492.65	523.59	513.18	523.58
6	GO	492.2	525.51	514.56	536.63
7	TO	482.04	509.19	496.92	517.29

27 rows

Salvando em arquivo delta na camada gold