03.DataFrame

segunda-feira, 25 de setembro de 2023 12:38

DataFrames

- Tabelas com linhas e colunas
- Imutáveis
- Com schema conhecido
- Linhagem preservada
- Colunas podem ter tipos diferentes
- Existem análises comuns - Agrupar, ordenar e filtrar
- Spark pode otimizar estas análises atraves de planos de execução

Lazy Evaluation - O processo de transformação de fato só ocorre quando há uma ação

Tipos de dados de um DataFrame

Tipo
ByteType
ShortType
IntegerType
LongType
FloatType
DoubleType
DecimalType
StringType
BinaryType
BooleanType
TimestampType
DateType
ArrayType
MapType
StructType
StructField

Schema

- Pode deixar o spark inferir a partir de parte dos dados
- Você pode definir o schema
- Definir tem vantagens
  - Tipo correto
  - Sem overload (Processamento em excesso)

Criando DataFrame

```
>>> from pyspark.sql import SparkSession
>>> df1 = spark.createDataFrame([("Geronimo",20),("Eunice",30),("Caio",10)])
>>> df1.show
<bound method DataFrame.show of DataFrame[_1: string, _2: bigint]>
>>> df1.show()
+--------+---+
|      _1| _2|
+--------+---+
|Geronimo| 20|
|  Eunice| 30|
|    Caio| 10|
+--------+---+
```

Criando Schema

```
>>> schema = "Id INT, Nome STRING"
>>> dados = [[1,"Geronimo"],[2,"Eunice"]]
>>> df2 = spark.createDataFrame(dados, schema)
>>> df2.show()
+---+--------+
| Id|    Nome|
+---+--------+
|  1|Geronimo|
|  2|  Eunice|
+---+--------+

>>> df2.show(1)
+---+--------+
| Id|    Nome|
+---+--------+
|  1|Geronimo|
+---+--------+
```

Criando outro exemplo

```
>>> from pyspark.sql.functions import sum
>>> schema2 = "Produtos STRING, Vendas INT"
>>> vendas = [["Caneta",10],["Lápis",20],["Caneta",40]]
>>> df3 = spark.createDataFrame(vendas,schema2)
>>> df3.show()
+--------+------+
|Produtos|Vendas|
+--------+------+
|  Caneta|    10|
|   Lápis|    20|
|  Caneta|    40|
+--------+------+

>>>
```

Agrupando DataFrame por Produtos

```
>>> from pyspark.sql.functions import sum
>>> schema2 = "Produtos STRING, Vendas INT"
>>> vendas = [["Caneta",10],["Lápis",20],["Caneta",40]]
>>> df3 = spark.createDataFrame(vendas,schema2)
>>> df3.show()
+--------+------+
|Produtos|Vendas|
+--------+------+
|  Caneta|    10|
|   Lápis|    20|
|  Caneta|    40|
+--------+------+

>>> agrupado = df3.groupBy("Produtos").agg(sum("Vendas"))
>>> agrupado.show()
+--------+-----------+
|Produtos|sum(Vendas)|
+--------+-----------+
|  Caneta|         50|
|   Lápis|         20|
+--------+-----------+

>>>
```

Usando Select

```
>>> df3.select("Produtos").show()
+--------+
|Produtos|
+--------+
|  Caneta|
|   Lápis|
|  Caneta|
+--------+

>>> df3.select("Vendas","Produtos")
DataFrame[Vendas: int, Produtos: string]
>>> df3.select("Vendas","Produtos").show()
+------+--------+
|Vendas|Produtos|
+------+--------+
|    10|  Caneta|
|    20|   Lápis|
|    40|  Caneta|
+------+--------+

>>>
```

Usando select e criando coluna calculada

```
>>> df3.select("Vendas","Produtos",expr("Vendas*0.2")).show()
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'expr' is not defined
>>> from pyspark.sql.functions import expr
>>> df3.select("Produtos","Vendas",expr("Vendas * 0.2")).show()
+--------+------+--------------+
|Produtos|Vendas|(Vendas * 0.2)|
+--------+------+--------------+
|  Caneta|    10|           2.0|
|   Lápis|    20|           4.0|
|  Caneta|    40|           8.0|
+--------+------+--------------+

>>>
```

```
>>> df3.groupBy("Produtos").agg(sum("Vendas")).show()
+--------+----------+
|Produtos|sum(Vendas)|
+--------+----------+
|  Caneta|        50|
|   Lápis|        20|
+--------+----------+
```

Usando o comando Select no pyspark

```
>>> df3.select("Produtos").show()
+--------+
|Produtos|
+--------+
|  Caneta|
|   Lápis|
|  Caneta|
+--------+

>>> df3.select("Produtos","Vendas").show()
+--------+------+
|Produtos|Vendas|
+--------+------+
|  Caneta|    10|
|   Lápis|    20|
|  Caneta|    40|
+--------+------+

>>> df3.select("Vendas","Produtos").show()
+------+--------+
|Vendas|Produtos|
+------+--------+
|    10|  Caneta|
|    20|   Lápis|
|    40|  Caneta|
+------+--------+

>>>
```

Usando expressão para coluna calculada

```
>>> from pyspark.sql.functions import expr
>>> df3.select("Produtos","Vendas", expr("Vendas * 0.2")).show()
+--------+------+-------------+
|Produtos|Vendas|(Vendas * 0.2)|
+--------+------+-------------+
|  Caneta|    10|          2.0|
|   Lápis|    20|          4.0|
|  Caneta|    40|          8.0|
+--------+------+-------------+

>>>
```

Apresentando o schema e as colunas

```
>>> df3.schema
StructType(List(StructField(Produtos,StringType,true),StructField(Vendas,IntegerType,true
)))
>>> df3.columns
['Produtos', 'Vendas']
>>>
```