distinct() vs dropDuplicates() no Apache Spark

Qual é a diferença entre distinct() e dropDuplicates() no Spark?

O API Spark vem com duas funções que podem ser utilizadas para remover duplicidades, distinct() e dropDuplicates(). Apesar dos dois métodos fazerem praticamente o mesmo

trabalho, há uma diferença que é bastante importante de utilização.

Neste artigo, vamos explorar como funcionam ambas funções e qual é a principal diferença. Além disso, vamos discutir quando utilizar uma em relação à outra.

Note-se que os exemplos que iremos utilizar para explorar os métodos foram construídos utilizando a API Python.

O método distinct()

distinct()

Retorna um novo DataFrame contendo as linhas distintas no DataFrame.

distinct() retornará as linhas distintas do DataFrame. Como exemplo, considere o seguinte DataFrame

>>> df.show()

+---+---------+---+

| id| name|age|

+---+---------+---+

| 1|Guilherme| 22|

| 1|Guilherme| 22|

| 1|Guilherme| 23|

| 2| Isaac| 12|

+---+---------+---+

O método não leva argumentos, portanto, todas as colunas são levadas em conta ao remover as duplicidades:

>>> df.distinct().show()

+---+---------+---+

| id| name|age|

+---+---------+---+

| 1|Guilherme| 23|

| 2| Isaac| 12|

| 1|Guilherme| 22|

+---+---------+---+

Agora, se precisar considerar apenas um subdataset ao eliminar duplicidades, então primeiro faça uma seleção da(s) colunas antes de efetuar um distinct() como podemos ver a seguir

>>> df.select(['id', 'name']).distinct().show()

+---+---------+

| id| name|

+---+---------+

| 2| Isaac|

| 1|Guilherme|

+---+---------+

Isto significa que o DataFrame retornado conterá apenas o subdataset que foi utilizado para eliminar as duplicidades. Se for esse o caso, então provavelmente distinct() não funcionará.

O método dropDuplicates()

Agora dropDuplicates() irá remover as duplicidades detectadas sobre um conjunto específico de colunas (se fornecidas), mas ao contrário do distinct() irá retornar todas as colunas do dataframe original.

Por exemplo, se quiser remover as duplicidades considerando todas as colunas, deve executar o seguinte comando

>>> df.dropDuplicates().show()

+---+---------+---+

| id| name|age|

+---+---------+---+

| 1|Guilherme| 23|

| 2| Isaac| 12|

| 1|Guilherme| 22|

+---+---------+---+

Portanto, dropDuplicates() é o melhor a se fazer caso queira remover duplicidade de um conjunto de colunas, mas ao mesmo tempo quer manter todas as colunas da estrutura original.

df.dropDuplicates(['id', 'name']).show()

+---+---------+---+

| id| name|age|

+---+---------+---+

| 2| Isaac| 12|

| 1|Guilherme| 22|

+---+---------+---+

Conclusão

Neste artigo exploramos duas funções úteis do API Spark, os métodos distinct() e dropDuplicates().

Ambos podem ser utilizados para eliminar linhas duplicadas de um DataFrame, no entanto, a diferença é que distinct() não leva qualquer argumento em consideração, enquanto que dropDuplicates() pode ser dado um subdataset

ao eliminar registos duplicados.

Isto significa que dropDuplicates() é a opção mais adequada quando se quer eliminar registos duplicados, considerando apenas um subdataset e ao mesmo tempo todas as colunas da DataFrame original devem ser retornadas.