How can I use the flatMap() transformation in PySpark?

Authored by

stats writer

June 24, 2024

The flatMap() transformation is a useful function in PySpark that allows for the transformation of a collection of elements into a new collection. This function takes in a function as an argument and applies it to each element in the original collection. The main difference between flatMap() and map() is that flatMap() can return multiple elements for each input element, whereas map() only returns a single element. This can be useful when working with data that may have nested structures or when needing to create new collections from existing data.

PySpark `flatMap()` is a transformation operation that flattens the RDD/DataFrame (array/map DataFrame columns) after applying the function on every element and returns a new PySpark RDD/DataFrame. In this article, you will learn the syntax and usage of the PySpark flatMap() with an example.

First, let's create an RDD from the list.

```
data =
rdd=spark.sparkContext.parallelize(data)
for element in rdd.collect():
print(element)
```

This yields the below output

```
Project Gutenberg’s
Alice’s Adventures in Wonderland
Project Gutenberg’s
Adventures in Wonderland
Project Gutenberg’s
```

flatMap() Syntax

```
flatMap(f, preservesPartitioning=False)
```

flatMap() Example

Now, let's see with an example of how to apply a flatMap() transformation on RDD. In the below example, first, it splits each record by space in an RDD and finally flattens it. Resulting RDD consists of a single word on each record.

```
rdd2=rdd.flatMap(lambda x: x.split(" "))
for element in rdd2.collect():
print(element)
```

This yields below output.

```
Project
Gutenberg's
Alice's
Adventures
in
Wonderland
Project
Gutenberg's
Adventures
in
Wonderland
Project
Gutenberg's
```

Complete PySpark flatMap() example

Below is the complete example of flatMap() function that works with RDD.

```
from pyspark.sql import SparkSession
spark = SparkSession.builder.appName('SparkByExamples.com').getOrCreate()
```

data =
rdd=spark.sparkContext.parallelize(data)
for element in rdd.collect():
print(element)

#Flatmap
rdd2=rdd.flatMap(lambda x: x.split(" "))
for element in rdd2.collect():
print(element)

Using flatMap() transformation on DataFrame

Unfortunately, PySpark DataFame doesn't have flatMap() transformation however, DataFrame has explode() SQL function that is used to flatten the column. Below is a complete example.

```
import pyspark
from pyspark.sql import SparkSession
spark = SparkSession.builder.appName('pyspark-by-examples').getOrCreate()
```

arrayData = ,{'hair':'black','eye':'brown'}),
('Michael',,{'hair':'brown','eye':None}),
('Robert',,{'hair':'red','eye':''}),
('Washington',None,None),
('Jefferson',,{})]
df = spark.createDataFrame(data=arrayData, schema =)

from pyspark.sql.functions import explode
df2 = df.select(df.name,explode(df.knownLanguages))
df2.printSchema()
df2.show()

This example flattens the array column "`knownLanguages`" and yields below output

```
root
|-- name: string (nullable = true)
|-- col: string (nullable = true)

+---------+------+
| name| col|
+---------+------+
| James| Java|
| James| Scala|
| Michael| Spark|
| Michael| Java|
| Michael| null|
| Robert|CSharp|
| Robert| |
|Jefferson| 1|
|Jefferson| 2|
```

```
+---------+------+
```

Conclusion

In conclusion, you have learned how to apply a PySpark flatMap() transformation to flattens the array or map columns and also learned how to use alternatives for DataFrame.

Related Articles

Happy Learning !!