How can I convert a string column in PySpark to an array column?

Authored by

stats writer

June 24, 2024

To convert a string column in PySpark to an array column, you can use the split function and specify the delimiter for the string. This will split the string into an array of substrings, which can then be converted into an array column. This process is useful for manipulating and analyzing data that is stored in string format, and allows for easier access and manipulation of individual elements within the string. Additionally, the resulting array column can be used for further data processing and analysis in PySpark.

To convert a string column (StringType) to an array column (ArrayType) in PySpark, you can use the `split()` function from the `pyspark.sql.functions` module. This function splits a string on a specified delimiter like space, comma, pipe e.t.c and returns an array.

In this article, I will explain converting String to Array column using split() function on DataFrame and SQL query.

Split() function syntax

PySpark SQL `split()` is grouped under Array Functions in PySpark SQL Functions class with the below syntax.

```
pyspark.sql.functions.split(str, pattern, limit=-1)
```

The split() function takes the DataFrame column of type String as the first argument and string delimiter as the second argument you want to split on. You can also use the pattern as a delimiter. This function returns `pyspark.sql.Column` of type Array.

Before we start with usage, first, let's create a DataFrame with a string column with text separated with comma delimiter

```
# Import
from pyspark.sql import SparkSession

# Create SparkSession
spark = SparkSession.builder
.appName('SparkByExamples.com')
.getOrCreate()

# Data
data =

columns=
```

# Create DataFrame
df=spark.createDataFrame(data,columns)
df.printSchema()

As you notice, we have a name column with firstname, middle name, and lastname separated by commas.

```
# Output:
root
|-- name: string (nullable = true)
|-- dob_year: string (nullable = true)
|-- gender: string (nullable = true)
|-- salary: integer (nullable = false)
```

PySpark Convert String to Array Column

Let's import the `pyspark.sql.functions import split` and use the `split()` function with select() to split the string column `name` by comma delimiter and create an array. The select() method just returns the array column.

```
# Import
from pyspark.sql.functions import split, col
```

# using split()
df2 = df.select(split(col("name"),",").alias("NameArray"))
df2.printSchema()
df2.show()

This results in the below output. As you see in the below schema, NameArray column is an array type.

```
# Output:
root
|-- NameArray: array (nullable = true)
| |-- element: string (containsNull = true)
```

+----------------------+
|NameArray |

```
+----------------------+
| |
||
| |
| |
| |
+----------------------+
```

Using split() on withColumn()

You can utilize the `split()` function within the `withColumn()` method to create a new column with array on the DataFrame. If you do not need the original column, use drop() to remove the column.

```
from pyspark.sql.functions import split

# Splitting the "name" column into an array of first name, middle name, and last name
df = df.withColumn("name_array", split(df, ",s*"))

# Displaying the updated DataFrame
df.show(truncate=False)
```

Output:

```
# Output
+--------------------+-------+------+------+-----------------------+
|name |dob_year|gender|salary|name_array |
+--------------------+--------+------+------+-----------------------+
|James, A, Smith |2018 |M |3000 | |
|Michael, Rose, Jones|2010 |M |4000 | |
|Robert,K,Williams |2010 |M |4000 | |
|Maria,Anne,Jones |2005 |F |4000 | |
|Jen,Mary,Brown |2010 | |-1 | |
+--------------------+--------+------+------+-----------------------+
```

Convert String to Array Column using SQL Query

Alternatively, you can write the same example using the SQL query. First, create a table using `createOrReplaceTempView()` and spark.sql() to run the SQL query.

```
# Run SQL query
df.createOrReplaceTempView("PERSON")
spark.sql("select SPLIT(name,',') as NameArray from PERSON")
.show()
```

This displays the output the same as the above.

Complete Example

The following is a complete example of splitting a String-type column based on a delimiter or patterns and converting it into an Array-type column.

Find an example at PySpark-Examples GitHub project for reference.

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

data =

columns=
df=spark.createDataFrame(data,columns)
df.printSchema()
df.show(truncate=False)

from pyspark.sql.functions import split, col
df2 = df.select(split(col("name"),",").alias("NameArray"))
.drop("name")
df2.printSchema()
df2.show()

df.createOrReplaceTempView("PERSON")
spark.sql("select SPLIT(name,',') as NameArray from PERSON")
.show()

Conclusion

The `split()` function is used to transform the string column type into an array type. This method is used with the `withColumn()` or `select()` to create a new array column where each string element

is separated into an array based on the delimiter. This approach is useful for transforming comma-separated values or other delimited strings into array structures for further processing.

Happy Learning !!

Related Articles