How can I use the PySpark withColumnRenamed function to rename a column on a DataFrame?

Authored by

stats writer

June 24, 2024

RECOMMENDED CITATION

stats writer (2024). *How can I use the PySpark withColumnRenamed function to rename a column on a DataFrame?*. PSYCHOLOGICAL SCALES. Retrieved from https://scales.arabpsychology.com/?p=150700

ARABPSYCHOLOGY.COM

The PySpark withColumnRenamed function is a useful tool for renaming columns on a DataFrame. This function allows users to specify the name of the column they want to rename, as well as the new desired name. By using this function, users can easily modify the column names on their DataFrame without having to create a new DataFrame. This feature is particularly helpful for data manipulation and organization, as it allows for more efficient and streamlined data analysis. With the PySpark withColumnRenamed function, users can easily and effectively rename columns on their DataFrame, making it a valuable tool in any data analysis workflow.

Use PySpark `withColumnRenamed()` to rename a DataFrame column, we often need to rename one column or multiple (or all) columns on PySpark DataFrame, you can do this in several ways. When columns are nested it becomes complicated.

Since DataFrame's are an immutable collection, you can't rename or update a column instead when using withColumnRenamed() it creates a new DataFrame with updated column names, In this PySpark article, I will cover different ways to rename columns with several use cases like rename nested column, all columns, selected multiple columns with Python/PySpark examples.

Refer to this page, If you are looking for a Spark with Scala example and rename pandas column with examples

PySpark withColumnRenamed - To rename DataFrame column namePySpark withColumnRenamed - To rename multiple columnsUsing StructType - To rename nested column on PySpark DataFrame Using Select - To rename nested columnsUsing withColumn - To rename nested columnsUsing col() function - To Dynamically rename all or multiple columnsUsing toDF() - To rename all or multiple columns

First, let's create our data set to work with.

```
dataDF =
```

Our base schema with nested structure.

```
from pyspark.sql.types import StructType,StructField, StringType, IntegerType
schema = StructType()),
StructField('dob', StringType(), True),
StructField('gender', StringType(), True),
StructField('gender', IntegerType(), True)
])
```

ARABPSYCHOLOGY.COM

Let's create the DataFrame by using parallelize and provide the above schema.

```
from pyspark.sql import SparkSession
spark = SparkSession.builder.appName('SparkByExamples.com').getOrCreate()
df = spark.createDataFrame(data = dataDF, schema = schema)
df.printSchema()
```

Below is our schema structure. I am not printing data here as it is not necessary for our examples. This schema has a nested structure.

```
root
 |-- name: struct (nullable = true)
 |    |-- firstname: string (nullable = true)
 |    |-- middlename: string (nullable = true)
 |    |-- lastname: string (nullable = true)
 |-- dob: string (nullable = true)
 |-- gender: string (nullable = true)
 |-- salary: integer (nullable = true)
```

1. PySpark withColumnRenamed - To rename DataFrame column name

PySpark has a `withColumnRenamed()` function on DataFrame to change a column name. This is the most straight forward approach; this function takes two parameters; the first is your existing column name and the second is the new column name you wish for.

PySpark withColumnRenamed() Syntax:

```
withColumnRenamed(existingName, newNam)
```

`existingName` - The existing column name you want to change

`newName` - New name of the column

Returns a new DataFrame with a column renamed.

Example

ARABPSYCHOLOGY.COM

```
df.withColumnRenamed("dob","DateOfBirth").printSchema()
```

The above statement changes column "dob" to "DateOfBirth" on PySpark DataFrame. Note that `withColumnRenamed` function returns a new DataFrame and doesn't modify the current DataFrame.

```
root
 |-- name: struct (nullable = true)
 |    |-- firstname: string (nullable = true)
 |    |-- middlename: string (nullable = true)
 |    |-- lastname: string (nullable = true)
 |-- DateOfBirth: string (nullable = true)
 |-- gender: string (nullable = true)
 |-- salary: integer (nullable = true)
```

2. PySpark withColumnRenamed - To rename multiple columns

To change multiple column names, we should chain `withColumnRenamed` functions as shown below. You can also store all columns to rename in a list and loop through to rename all columns, I will leave this to you to explore.

```
df2 = df.withColumnRenamed("dob","DateOfBirth")
.withColumnRenamed("salary","salary_amount")
df2.printSchema()
```

This creates a new DataFrame "df2" after renaming dob and salary columns.

3. Using PySpark StructType - To rename a nested column in Dataframe

Changing a column name on nested data is not straight forward and we can do this by creating a new schema with new DataFrame columns using StructType and use it using cast function as shown below.

```
schema2 = StructType()
```

ARABPSYCHOLOGY.COM

```
df.select(col("name").cast(schema2),
col("dob"), col("gender"),col("salary"))
.printSchema()
```

This statement renames `firstname` to `fname` and `lastname` to `lname` within name structure.

```
root
 |-- name: struct (nullable = true)
 |    |-- fname: string (nullable = true)
 |    |-- middlename: string (nullable = true)
 |    |-- lname: string (nullable = true)
 |-- dob: string (nullable = true)
 |-- gender: string (nullable = true)
 |-- salary: integer (nullable = true)
```

4. Using Select - To rename nested elements.

Let's see another way to change nested columns by transposing the structure to flat.

```
from pyspark.sql.functions import *
df.select(col("name.firstname").alias("fname"),
col("name.middlename").alias("mname"),
col("name.lastname").alias("lname"),
col("dob"),col("gender"),col("salary"))
.printSchema()
```

```
root
 |-- fname: string (nullable = true)
 |-- mname: string (nullable = true)
 |-- lname: string (nullable = true)
 |-- dob: string (nullable = true)
 |-- gender: string (nullable = true)
 |-- salary: integer (nullable = true)
```

ARABPSYCHOLOGY.COM

5. Using PySpark DataFrame withColumn - To rename nested columns

When you have nested columns on PySpark DatFrame and if you want to rename it, use `withColumn` on a data frame object to create a new column from an existing and we will need to drop the existing column. Below example creates a "fname" column from "name.firstname" and drops the "name" column

```
from pyspark.sql.functions import *
df4 = df.withColumn("fname",col("name.firstname"))
.withColumn("mname",col("name.middlename"))
.withColumn("lname",col("name.lastname"))
.drop("name")
df4.printSchema()
```

6. Using col() function - To Dynamically rename all or multiple columns

Another way to change all column names on Dataframe is to use `col()` function.

```
IN progress
```

7. Using toDF() - To change all columns in a PySpark DataFrame

When we have data in a flat structure (without nested) , use `toDF()` with a new schema to change all column names.

```
newColumns =
df.toDF(*newColumns).printSchema()
```

Source code

```
import pyspark
from pyspark.sql import SparkSession
from pyspark.sql.types import StructType,StructField, StringType, IntegerType
```

ARABPSYCHOLOGY.COM

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

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

dataDF =

schema = StructType()),
StructField('dob', StringType(), True),
StructField('gender', StringType(), True),
StructField('salary', IntegerType(), True)
])

df = spark.createDataFrame(data = dataDF, schema = schema)
df.printSchema()

# Example 1
df.withColumnRenamed("dob","DateOfBirth").printSchema()
# Example 2
df2 = df.withColumnRenamed("dob","DateOfBirth")
.withColumnRenamed("salary","salary_amount")
df2.printSchema()

# Example 3
schema2 = StructType()

df.select(col("name").cast(schema2),
col("dob"),
col("gender"),
col("salary"))
.printSchema()

Example 4
df.select(col("name.firstname").alias("fname"),
col("name.middlename").alias("mname"),
col("name.lastname").alias("lname"),
col("dob"),col("gender"),col("salary"))
.printSchema()

# Example 5
df4 = df.withColumn("fname",col("name.firstname"))
.withColumn("mname",col("name.middlename"))
.withColumn("lname",col("name.lastname"))

ARABPSYCHOLOGY.COM

```
.drop("name")
df4.printSchema()

#Example 7
newColumns =
df.toDF(*newColumns).printSchema()

# Example 6
'''
not working
old_columns = Seq("dob","gender","salary","fname","mname","lname")
new_columns = Seq("DateOfBirth","Sex","salary","firstName","middleName","lastName")
columnsList = old_columns.zip(new_columns).map(f=>{col(f._1).as(f._2)})
df5 = df4.select(columnsList:_*)
df5.printSchema()
'''
```

The complete code can be downloaded from GitHub

Conclusion:

This article explains different ways to rename all, a single, multiple, and nested columns on PySpark DataFrame.

I hope you like this article!!

Happy Learning.

Related Articles

ARABPSYCHOLOGY.COM