# How can I apply a function to a column in PySpark?

Authored by

**stats writer**

June 24, 2024

To apply a function to a column in PySpark, you can use the "withColumn" method. This allows you to create a new column by applying a user-defined function to an existing column. The syntax for this is: dataframe.withColumn("new_column_name", function(column_name)). This will create a new column with the specified name and apply the specified function to the values in the original column. This method is useful for data transformation and manipulation in PySpark.

How to apply a function to a column in PySpark? By using withColumn(), sql(), select() you can apply a built-in function or custom function to a column. In order to apply a custom function, first you need to create a function and register the function as a UDF. Recent versions of PySpark provide a way to use Pandas API hence, you can also use `pyspark.pandas.DataFrame.apply()`.

Explain PySpark Pandas UDF with Examples

Let's create a PySpark DataFrame.

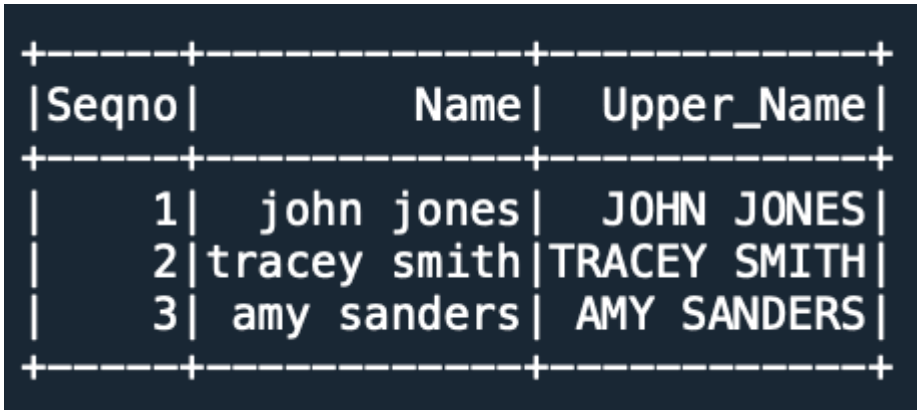
```
from pyspark.sql import SparkSession

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

df = spark.createDataFrame(data=data,schema=columns)

df.show(truncate=False)
```

## 1. PySpark apply Function using withColumn()

PySpark withColumn() is a transformation function that is used to apply a function to the column. The below example applies an `upper()` function to column `df.Name`.

```
# Apply function using withColumn
from pyspark.sql.functions import upper
df.withColumn("Upper_Name", upper(df.Name))
.show()
```

Yields below output.



```
+-----+-------------+-------------+
|Seqno|         Name|   Upper_Name|
+-----+-------------+-------------+
|    1|   john jones|   JOHN JONES|
|    2|tracey smith|TRACEY SMITH|
|    3|  amy sanders|  AMY SANDERS|
+-----+-------------+-------------+
```

## 2. Apply Function using select()

The select() is used to select the columns from the PySpark DataFrame while selecting the columns you can also apply the function to a column.

```
# Apply function using select
df.select("Seqno","Name", upper(df.Name))
.show()
```

Yields the same output as above.

## 3. Apply Function to SQL

You can also apply the function to the column while running the SQL query on the PySpark DataFrame. In order to use SQL, make sure you create a temporary view using createOrReplaceTempView().

```
# Apply function using sql()
df.createOrReplaceTempView("TAB")
spark.sql("select Seqno, Name, UPPER(Name) from TAB")
.show()
```

Yields the same output as above.

## Using UDF

In this section, I will explain how to create a custom PySpark UDF function and apply this function

to a column.

PySpark UDF (a.k.a User Defined Function) is the most useful feature of Spark SQL & DataFrame that is used to extend the PySpark built-in capabilities. Note that UDFs are the most expensive operations hence use them only if you have no choice and when essential.

Following are the steps to apply a custom UDF function on an SQL query.

## 4.1 Create Custom Function

First, create a python function. Though upper() is already available in the PySpark SQL function, to make the example simple, I would like to create one.

```
# Create custom function
def upperCase(str):
return str.upper()
```

## 4.2 Register UDF

Create a udf function by wrapping the above function with udf().

```
# Convert function to udf
from pyspark.sql.functions import col, udf
from pyspark.sql.types import StringType
upperCaseUDF = udf(lambda x:upperCase(x),StringType())
```

## 4.3 Apply Custom UDF to Column

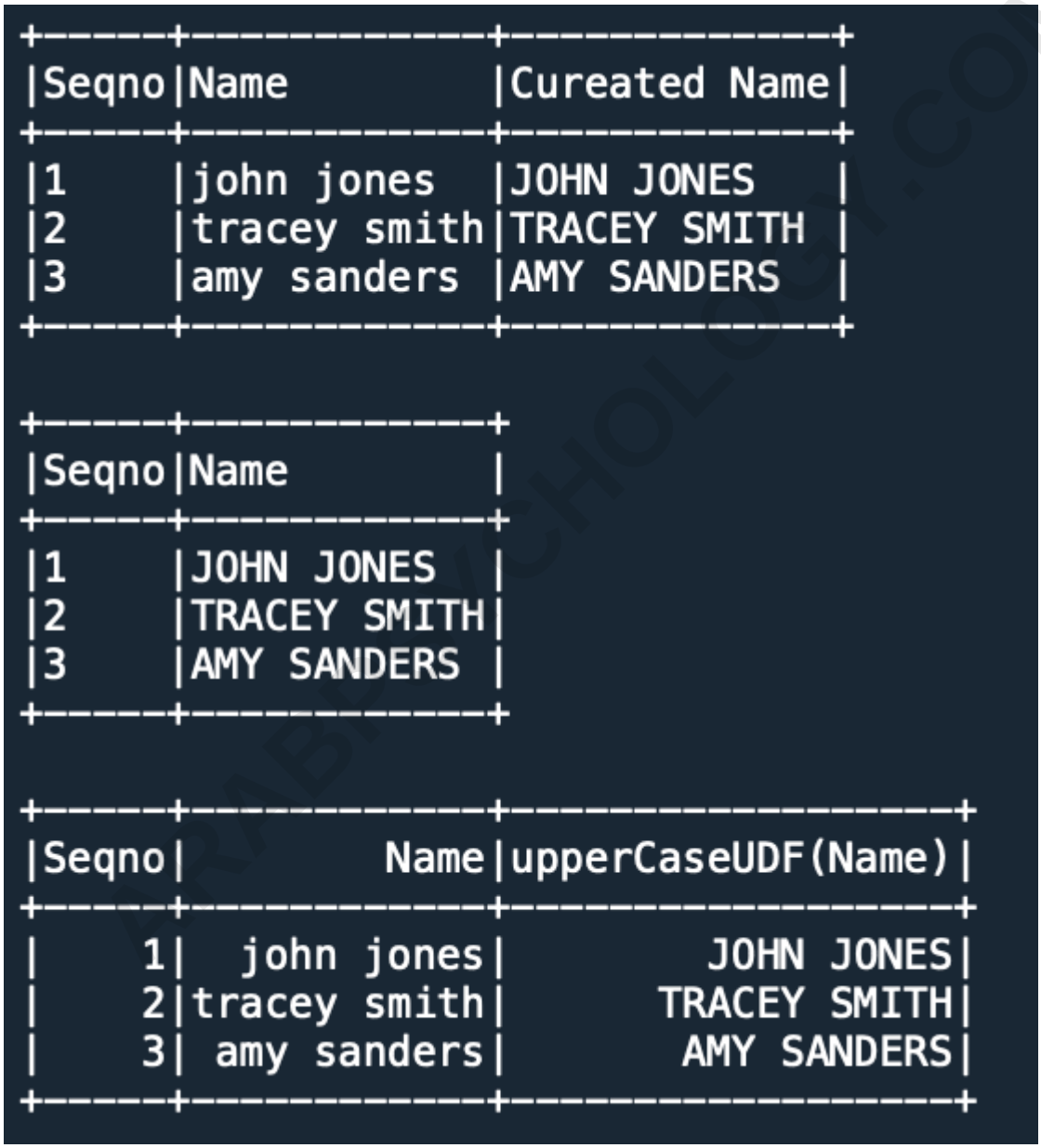
Finally apply the function to the column by using withColumn(), select() and sql().

```
# Custom UDF with withColumn()
df.withColumn("Cureated Name", upperCaseUDF(col("Name")))
.show(truncate=False)
```

```
# Custom UDF with select()
df.select(col("Seqno"),
upperCaseUDF(col("Name")).alias("Name") )
.show(truncate=False)
```

```
# Custom UDF with sql()
spark.udf.register("upperCaseUDF", upperCaseUDF)
df.createOrReplaceTempView("TAB")
spark.sql("select Seqno, Name, upperCaseUDF(Name) from TAB")
.show()
```

Yields below output.




## 5. PySpark Pandas apply()

PySpark DataFrame doesn't contain the apply() function however, we can leverage Pandas

DataFrame.apply() by running Pandas API over PySpark.

```
# Imports
import pyspark.pandas as ps
import numpy as np

technologies = ({
'Fee' :,
'Discount':
})
# Create a DataFrame
psdf = ps.DataFrame(technologies)
print(psdf)

def add(data):
return data + data

addDF = psdf.apply(add,axis=1)
print(addDF)
```

## 6. Complete Example

Following is the complete example of applying a function to a column using withColumn(), SQL(), select() e.t.c

```
from pyspark.sql import SparkSession

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

df = spark.createDataFrame(data=data,schema=columns)

df.show(truncate=False)

# Apply function using withColumn
from pyspark.sql.functions import upper
df.withColumn("Upper_Name", upper(df.Name))
.show()
```

```
# Apply function using select
df.select("Seqno","Name", upper(df.Name))
.show()

# Apply function using sql()
df.createOrReplaceTempView("TAB")
spark.sql("select Seqno, Name, UPPER(Name) from TAB")
.show()

# Create custom function
def upperCase(str):
return str.upper()

# Convert function to udf
from pyspark.sql.functions import col, udf
from pyspark.sql.types import StringType
upperCaseUDF = udf(lambda x:upperCase(x),StringType())

# Custom UDF with withColumn()
df.withColumn("Cureated Name", upperCaseUDF(col("Name")))
.show(truncate=False)

# Custom UDF with select()
df.select(col("Seqno"),
upperCaseUDF(col("Name")).alias("Name") )
.show(truncate=False)

# Custom UDF with sql()
spark.udf.register("upperCaseUDF", upperCaseUDF)
df.createOrReplaceTempView("TAB")
spark.sql("select Seqno, Name, upperCaseUDF(Name) from TAB")
.show()
```



## 6. Conclusion

In this article, you have learned how to apply a built-in function to a PySpark column by using withColumn(), select() and spark.sql(). Also learned how to create a custom UDF function and apply this function to the column.

## Related Articles