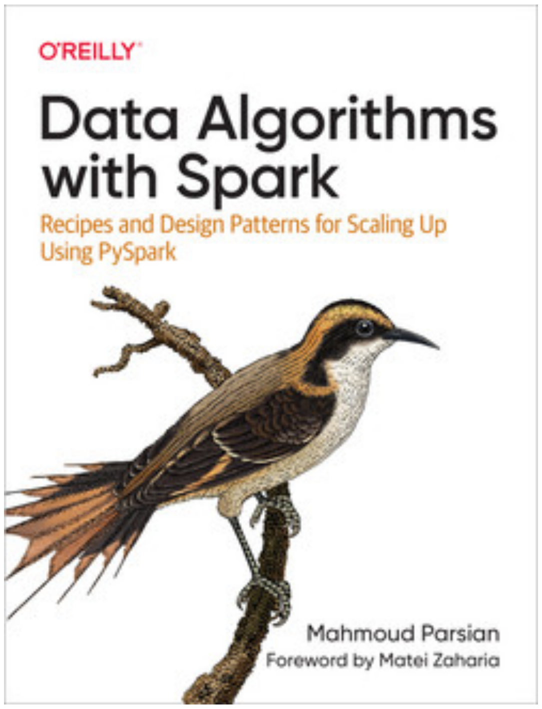
User-Defined Functions (UDF)

```
 @author: Mahmoud Parsian
          Ph.D. in Computer Science
          email: mahmoud.parsian@yahoo.com

Last updated: 2/25/2023
```

	"… This book will be a great resource for both readers looking to implement existing algorithms in a scalable fashion and readers who are developing new, custom algorithms using Spark. …" Dr. Matei Zaharia Original Creator of Apache Spark FOREWORD by Dr. Matei Zaharia

---

1. Introduction

This short article shows how to use Python user-defined functions in PySpark applications. To use a UDF, we need to do some basic tasks:

1. Create a UDF (user-defined-function) in Python
2. Register UDF

3. Use UDF in Spark SQL

2. What is a UDF?

User-Defined Functions (UDFs) are user-programmable functions that act on one row. Spark UDF (a.k.a User Defined Function) is the useful feature of Spark SQL & DataFrame which extends the Spark built in capabilities. UDF's are used to extend the functions of the Spark framework and re-use this function on several DataFrame.

3. Define a UDF in Python

Consider a function which triples its input:

```
# n : integer
def tripled(n):
  return 3 * n
#end-def
```

4. Register UDF

To register a UDF, we can use `SparkSession.udf.register()`. The `register()` function takes 3 parameters:

- 1st: the desired name for UDF to be used in SQL
- 2nd: the name of Python UDF function
- 3rd: the return data type of Python UDF function (if this parameter is missing, then it is assumed that it is `StringType()`

```
# "tripled_udf" : desired name to use in SQL
# tripled : defined Python function
# the last argument is the return type of UDF function
from pyspark.sql.types import IntegerType
spark.udf.register("tripled_udf", tripled, IntegerType())
```

Now, lets create a DataFrame and then apply the created UDF.

Create a sample DataFrame:

```
>>> data = [('alex', 20, 12000), ('jane', 30, 45000),
            ('rafa', 40, 56000), ('ted', 30, 145000),
            ('xo2', 10, 1332000), ('mary', 44, 555000)]
>>>
>>> column_names = ['name', 'age', 'salary']
>>> df = spark.createDataFrame(data, column_names)
>>>
>>> df
DataFrame[name: string, age: bigint, salary: bigint]
>>> df.printSchema()
root
 |-- name: string (nullable = true)
 |-- age: long (nullable = true)
 |-- salary: long (nullable = true)

>>>
>>> df.show()
+----+---+-------+
|name|age| salary|
+----+---+-------+
|alex| 20|  12000|
|jane| 30|  45000|
|rafa| 40|  56000|
| ted| 30| 145000|
| xo2| 10|1332000|
|mary| 44| 555000|
+----+---+-------+

>>> df.count()
6
>>> df2 = spark.sql("select * from people where salary > 67000")
>>> df2.show()
+----+---+-------+
|name|age| salary|
+----+---+-------+
| ted| 30| 145000|
| xo2| 10|1332000|
|mary| 44| 555000|
+----+---+-------+
```

5. Use UDF in SQL Query

```
>>> df.createOrReplaceTempView("people")
>>> df2 = spark.sql("select name, age, salary, tripled_udf(salary) as triple
>>> df2.show()
+----+---+-------+--------------+
|name|age| salary|tripled_salary|
+----+---+-------+--------------+
|alex| 20|  12000|         36000|
|jane| 30|  45000|        135000|
|rafa| 40|  56000|        168000|
| ted| 30| 145000|        435000|
| xo2| 10|1332000|       3996000|
|mary| 44| 555000|       1665000|
+----+---+-------+--------------+

>>>
```