How can I replace specific column values in a PySpark DataFrame?

Authored by

stats writer

June 24, 2024

Replacing specific column values in a PySpark DataFrame can be achieved by using the `withColumn()` function. This function takes in two arguments - the name of the column to be updated and the new value to be assigned. It then creates a new column with the updated values. This can be further customized by using conditional statements within the function, allowing for specific values to be replaced based on certain criteria. This approach provides a simple and efficient way to replace specific column values in a PySpark DataFrame.

You can replace column values of PySpark DataFrame by using SQL string functions `regexp_replace(), translate(),` and `overlay()` with Python examples.

In this article, I will cover examples of how to replace part of a string with another string, replace all columns, change values conditionally, replace values from a python dictionary, replace column value from another DataFrame column e.t.c

First, let's create a PySpark DataFrame with some addresses and will use this DataFrame to explain how to replace column values.

```
# Imports
# Create sample Data
from pyspark.sql import SparkSession
spark =
SparkSession.builder.master("local").appName(arabpsychology.com).getOrCreate()
address =
df =spark.createDataFrame(address,)
df.show()
```

1. PySpark Replace String Column Values

By using PySpark SQL function `regexp_replace()` you can replace a column value with a string for another string/substring. `regexp_replace()` uses **Java regex** for matching, if the regex does not match it returns an empty string, the below example replaces the street name `Rd` value with `Road` string on `address` column.

```
#Replace part of string with another string
from pyspark.sql.functions import regexp_replace
df.withColumn('address', regexp_replace('address', 'Rd', 'Road'))
.show(truncate=False)
```

```
# Output
```

```
+---+------------------+-----+
|id |address |state|
+---+------------------+-----+
|1 |14851 Jeffrey Road|DE |
|2 |43421 Margarita St|NY |
|3 |13111 Siemon Ave |CA |
+---+------------------+-----+
```

2. Replace Column Values Conditionally

In the above example, we just replaced `Rd` with `Road`, but not replaced `St` and `Ave` values, let's see how to replace column values conditionally in PySpark Dataframe by using when().otherwise() SQL condition function.

```
#Replace string column value conditionally
from pyspark.sql.functions import when
df.withColumn('address',
when(df.address.endswith('Rd'),regexp_replace(df.address,'Rd','Road'))
.when(df.address.endswith('St'),regexp_replace(df.address,'St','Street'))
.when(df.address.endswith('Ave'),regexp_replace(df.address,'Ave','Avenue'))
.otherwise(df.address))
.show(truncate=False)
```

```
#+---+--------------------+-----+
#|id |address |state|
#+---+--------------------+-----+
#|1 |14851 Jeffrey Road |DE |
#|2 |43421 Margarita Street|NY |
#|3 |13111 Siemon Avenue |CA |
#+---+--------------------+-----+
```

3. Replace Column Value with Dictionary (map)

You can also replace column values from the **python dictionary (map)**. In the below example, we replace the string value of the `state` column with the full abbreviated name from a dictionary **key-value pair**, in order to do so I use PySpark map() transformation to loop through each row of DataFrame.

```
#Replace values from Dictionary
```

```
stateDic={'CA':'California','NY':'New York','DE':'Delaware'}
df2=df.rdd.map(lambda x:
(x.id,x.address,stateDic)
).toDF()
df2.show()

#+---+-----------------+----------+
#| id| address| state|
#+---+-----------------+----------+
#| 1| 14851 Jeffrey Rd| Delaware|
#| 2|43421 Margarita St| New York|
#| 3| 13111 Siemon Ave|California|
#+---+-----------------+----------+
```

4. Replace Column Value Character by Character

By using `translate()` string function you can **replace character by character of DataFrame column** value. In the below example, every character of **1** is replaced with **A**, **2** replaced with **B**, and **3** replaced with **C** on the `address` column.

```
#Using translate to replace character by character
from pyspark.sql.functions import translate
df.withColumn('address', translate('address', '123', 'ABC'))
.show(truncate=False)

#+---+-----------------+-----+
#|id |address |state|
#+---+-----------------+-----+
#|1 |A485A Jeffrey Rd |DE |
#|2 |4C4BA Margarita St|NY |
#|3 |ACAAA Siemon Ave |CA |
#+---+-----------------+-----+
```

5. Replace Column with Another Column Value

By using `expr()` and `regexp_replace()` you can **replace column value with a value from another DataFrame column**. In the below example, we match the value from `col2` in `col1` and replace with `col3` to create `new_column`. Use expr() to provide SQL like expressions and is used to refer to another column to perform operations.

```
#Replace column with another column
from pyspark.sql.functions import expr
df = spark.createDataFrame(
,
("col1", "col2","col3")
)
df.withColumn("new_column",
expr("regexp_replace(col1, col2, col3)")
.alias("replaced_value")
).show()

#+---------+----+----+----------+
#| col1|col2|col3|new_column|
#+---------+----+----+----------+
#|ABCDE_XYZ| XYZ| FGH| ABCDE_FGH|
#+---------+----+----+----------+
```

6. Replace All or Multiple Column Values

If you want to replace values on all or selected DataFrame columns, refer to How to Replace NULL/None values on all column in PySpark or How to replace empty string with NULL/None value

7. Using overlay() Function

Replace column value with a string value from another column.

```
#Overlay
from pyspark.sql.functions import overlay
df = spark.createDataFrame(, ("col1", "col2"))
df.select(overlay("col1", "col2", 7).alias("overlayed")).show()

#+---------+
#|overlayed|
#+---------+
#|ABCDE_FGH|
#+---------+
```

8. Complete Example

```
from pyspark.sql import SparkSession
spark = SparkSession.builder.master("local").appName(arabpsychology.com).getOrCreate()
```

address =

df =spark.createDataFrame(address,)
df.show()

#Replace string
from pyspark.sql.functions import regexp_replace
df.withColumn('address', regexp_replace('address', 'Rd', 'Road'))
.show(truncate=False)

#Replace string
from pyspark.sql.functions import when
df.withColumn('address',
when(df.address.endswith('Rd'),regexp_replace(df.address,'Rd','Road'))
.when(df.address.endswith('St'),regexp_replace(df.address,'St','Street'))
.when(df.address.endswith('Ave'),regexp_replace(df.address,'Ave','Avenue'))
.otherwise(df.address))
.show(truncate=False)

#Replace values from Dictionary
stateDic={'CA':'California','NY':'New York','DE':'Delaware'}
df2=df.rdd.map(lambda x:
(x.id,x.address,stateDic)
).toDF()
df2.show()

#Using translate
from pyspark.sql.functions import translate
df.withColumn('address', translate('address', '123', 'ABC'))
.show(truncate=False)

#Replace column with another column
from pyspark.sql.functions import expr
df = spark.createDataFrame(, ("col1", "col2","col3"))
df.withColumn("new_column",
expr("regexp_replace(col1, col2, col3)")
.alias("replaced_value")
).show()

```
#Overlay
from pyspark.sql.functions import overlay
df = spark.createDataFrame(, ("col1", "col2"))
df.select(overlay("col1", "col2", 7).alias("overlayed")).show()
```

Conclusion

In conclusion `regexp_replace()` function is used to replace a string in a DataFrame column with another value, `translate()` function to replace character by character of column values, `overlay()` function to overlay string with another column string from start position and number of characters. Finally, you have also learned how to replace column values from a dictionary using Python examples.

Happy Learning !!

Related Articles

References