

How can I use the lit() function in PySpark to add a literal or constant value to a DataFrame?

Authored by
stats writer

June 24, 2024

RECOMMENDED CITATION

stats writer (2024). *How can I use the lit() function in PySpark to add a literal or constant value to a DataFrame?*. PSYCHOLOGICAL SCALES. Retrieved from <https://scales.arabpsychology.com/?p=151232>

The `lit()` function in PySpark allows users to add a literal or constant value to a DataFrame. This function takes in a value as an input and creates a new column in the DataFrame with that value repeated for each row. This can be useful for adding a default value to a column or creating a new column with a specified constant value. The `lit()` function simplifies and streamlines the process of adding a constant value to a DataFrame in PySpark.

PySpark SQL functions `lit()` and `typedLit()` are used to add a new column to DataFrame by assigning a literal or constant value. Both these functions return `Column type` as return type. `typedLit()` provides a way to be explicit about the data type of the constant value being added to a DataFrame, helping to ensure data consistency and type correctness of PySpark workflows.

Both of these are available in PySpark by importing `pyspark.sql.functions`

First, let's create a DataFrame.

```
# Imports
# prepare sample Data
import pyspark
from pyspark.sql import SparkSession

spark = SparkSession.builder.appName('SparkByExamples.com').getOrCreate()
data =
columns=
df = spark.createDataFrame(data = data, schema = columns)
```

lit() Function to Add Constant Column

PySpark `lit()` function is used to add constant or literal value as a new column to the DataFrame.

Creates a `Column` of literal value. The passed in object is returned directly if it is already a `Column`. If the object is a Scala Symbol, it is converted into a `Column` also. Otherwise, a new `Column` is created to represent the literal value

Let's take a look at some examples.

Example 1: Simple usage of lit() function

Let's see an example of how to create a new column with constant value using `lit()` [Spark SQL function](#). In the below snippet, we are creating a new column by adding a literal '1' to PySpark DataFrame.

```
# Usage of lit()
from pyspark.sql.functions import col,lit
df2 = df.select(col("EmpId"),col("Salary"),lit("1").alias("lit_value1"))
df2.show(truncate=False)
```

Output

```
+-----+-----+-----+
|EmpId|Salary|lit_value1|
+-----+-----+-----+
| 111| 50000| 1|
| 222| 60000| 1|
| 333| 40000| 1|
+-----+-----+-----+
```

Adding the same constant literal to all records in DataFrame may not be real-time useful so let's see another example.

Example 2 : lit() function with withColumn

The following example shows how to use pyspark lit() function using withColumn to derive a new column based on some conditions.

```
# Usage of lit() with withColumn()
from pyspark.sql.functions import when, lit, col
df3 = df2.withColumn("lit_value2", when((col("Salary") >=40000) & (col("Salary")
<= 50000),lit("100")).otherwise(lit("200")))
df3.show(truncate=False)
```

Below is the output for the above code snippet.

```
# Output
+-----+-----+-----+-----+
|EmpId|Salary|lit_value1|lit_value2|
+-----+-----+-----+-----+
| 111| 50000| 1| 100|
| 222| 60000| 1| 200|
| 333| 40000| 1| 100|
+-----+-----+-----+-----+
```

typedLit() Function

Difference between `lit()` and `typedLit()` is that the `typedLit()` function can handle collection types e.g.: Array, Dictionary(map), etc. Below is an example usage of `typedLit()`

```
# Usage of typedlit()
df4 = df4.withColumn("lit_value3", typedLit("flag", StringType()))
df4.show(truncate=False)
```

The above code adds a column `lit_value3` with the value being a string type flag.

Complete Example of How to Add Constant Column

```
import pyspark
from pyspark.sql import SparkSession

spark = SparkSession.builder.appName('SparkByExamples.com').getOrCreate()
data =
columns=
df = spark.createDataFrame(data = data, schema = columns)
df.printSchema()
df.show(truncate=False)

from pyspark.sql.functions import col,lit
df2 = df.select(col("EmpId"),col("Salary"),lit("1").alias("lit_value1"))
df2.show(truncate=False)
from pyspark.sql.functions import when
df3 = df2.withColumn("lit_value2", when(col("Salary") >=40000 & col("Salary") <=
50000,lit("100")).otherwise(lit("200")))
df3.show(truncate=False)
```

Frequently Asked Questions on lit()

Can `lit()` be used with different types of constant values?

`lit()` can be used with various constant values, including strings, integers, floats, and booleans.

How is `lit()` different from `expr()` in PySpark?

`lit()` is used to create a column with a constant literal value, while `expr()` is more versatile and can be used to express complex transformations and computations involving column expressions.

Can `lit()` be used to create a binary indicator column?

`lit()` is often used to create binary indicator columns by assigning a constant value of 1 or 0.

For example:

```
df = df.withColumn("is_active", lit(1))
```

Conclusion:

You have learned multiple ways to add a constant literal value to DataFrame using PySpark `lit()` function and have learned the difference between `lit` and `typedLit` functions.

When possible try to use predefined PySpark functions as they are a little bit more compile-time safety and perform better when compared to user-defined functions. If your application is critical on performance try to avoid using custom UDF functions as these are not guaranteed on performance.

Happy Learning !!

Related Articles