How can I use fillna() and fill() in PySpark to replace NULL or None values?

Authored by

stats writer

June 24, 2024

The fillna() and fill() functions in PySpark allow for the replacement of NULL or None values in a dataset. These functions can be used to fill in missing values with a specified value, such as a numeric value or string, or to fill in missing values with the previous or next non-null value in the dataset. This can be useful for cleaning and preprocessing data before performing analysis or modeling. By utilizing these functions, users can effectively handle missing data and ensure the accuracy and completeness of their results.

In PySpark,`fillna()` from DataFrame class or `fill()` from `DataFrameNaFunctions` is used to replace NULL/None values on all or selected multiple columns with either **zero(0), empty string, space, or any constant literal** values.

While working on PySpark DataFrame we often need to replace null values since certain operations on null values return errors. Hence, we need to graciously handle nulls as the first step before processing. Also, while writing to a file, it's always best practice to replace null values, not doing this results in nulls on the output file.

As part of the cleanup, sometimes you may need to Drop Rows with NULL/None Values in PySpark DataFrame and Filter Rows by checking IS NULL/NOT NULL conditions.

In this article, I will use both `fill()` and `fillna()` to replace null/none values with an empty string, constant value, and zero(0) on Dataframe columns integer, string with Python examples.

Before we start, Let's read a CSV into PySpark DataFrame file, Note that the reading process automatically assigns null values for missing data.

The file we are using here is available at GitHub small_zipcode.csv

```
# Create SparkSession and read csv
from pyspark.sql import SparkSession
spark = SparkSession.builder
.master("local")
.appName(arabpsychology.com)
.getOrCreate()
```

```
filePath="resources/small_zipcode.csv"
df = spark.read.options(header='true', inferSchema='true')
.csv(filePath)
```

```
df.printSchema()
df.show(truncate=False)
```

This yields the below output. Note that the columns city, and population have null values.

```
# Output
+---+-------+--------+------------------+-----+----------+
|id |zipcode|type |city |state|population|
+---+-------+--------+------------------+-----+----------+
|1 |704 |STANDARD|null |PR |30100 |
|2 |704 |null |PASEO COSTA DEL SUR|PR |null |
|3 |709 |null |BDA SAN LUIS |PR |3700 |
|4 |76166 |UNIQUE |CINGULAR WIRELESS |TX |84000 |
|5 |76177 |STANDARD|null |TX |null |
+---+-------+--------+------------------+-----+----------+
```

Now, let's see how to replace these null values.

PySpark fillna() & fill() Syntax

PySpark provides DataFrame.fillna() and DataFrameNaFunctions.fill() to replace NULL/None values. These two are aliases of each other and returns the same results.

```
fillna(value, subset=None)
fill(value, subset=None)
```

PySpark Replace NULL/None Values with Zero (0)

PySpark `fill(value:Long)` signatures that are available in `DataFrameNaFunctions` is used to replace NULL/None values with numeric values either zero(0) or any constant value for all integer and long datatype columns of PySpark DataFrame or Dataset.

```
#Replace 0 for null for all integer columns
df.na.fill(value=0).show()
```

#Replace 0 for null on only population column
df.na.fill(value=0,subset=).show()

Above both statements yield the same output, since we have just an integer column `population` with null values Note that it replaces only Integer columns since our value is 0.

```
# Output
+---+-------+--------+-------------------+-----+----------+
|id |zipcode|type |city |state|population|
+---+-------+--------+-------------------+-----+----------+
|1 |704 |STANDARD|null |PR |30100 |
|2 |704 |null |PASEO COSTA DEL SUR|PR |0 |
|3 |709 |null |BDA SAN LUIS |PR |3700 |
|4 |76166 |UNIQUE |CINGULAR WIRELESS |TX |84000 |
|5 |76177 |STANDARD|null |TX |0 |
+---+-------+--------+-------------------+-----+----------+
```

PySpark Replace Null/None Value with Empty String

Now let's see how to replace NULL/None values with an empty string or any constant values String on all DataFrame String columns.

```
df.na.fill("").show(false)
```

Yields below output. This replaces all String type columns with empty/blank string for all NULL values.

```
# Output
+---+-------+--------+-------------------+-----+----------+
|id |zipcode|type |city |state|population|
+---+-------+--------+-------------------+-----+----------+
|1 |704 |STANDARD| |PR |30100 |
|2 |704 | |PASEO COSTA DEL SUR|PR |null |
|3 |709 | |BDA SAN LUIS |PR |3700 |
|4 |76166 |UNIQUE |CINGULAR WIRELESS |TX |84000 |
|5 |76177 |STANDARD| |TX |null |
+---+-------+--------+-------------------+-----+----------+
```

Now, let's replace NULLs on specific columns, below example replace a column type with an empty string and a column city with the value "unknown".

```
df.na.fill("unknown",)
.na.fill("",).show()
```

Yields below output. This replaces null values with an empty string for `type` column and replaces with a constant value "`unknown`" for `city` column.

```
# Output
+---+-------+--------+------------------+-----+----------+
|id |zipcode|type |city |state|population|
+---+-------+--------+------------------+-----+----------+
|1 |704 |STANDARD|unknown |PR |30100 |
|2 |704 | |PASEO COSTA DEL SUR|PR |null |
|3 |709 | |BDA SAN LUIS |PR |3700 |
|4 |76166 |UNIQUE |CINGULAR WIRELESS |TX |84000 |
|5 |76177 |STANDARD|unknown |TX |null |
+---+-------+--------+------------------+-----+----------+
```

Alternatively, write the above statement as below.

```
df.na.fill({"city": "unknown", "type": ""})
.show()
```

Complete Code

Below is the complete code with a python example. You can use it by copying it from here or using GitHub to download the source code.

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

filePath="resources/small_zipcode.csv"
df = spark.read.options(header='true', inferSchema='true')
.csv(filePath)

df.printSchema()
df.show(truncate=False)

df.fillna(value=0).show()
df.fillna(value=0,subset=).show()

```
df.na.fill(value=0).show()
df.na.fill(value=0,subset=).show()

df.fillna(value="").show()
df.na.fill(value="").show()

df.fillna("unknown",)
.fillna("",).show()

df.fillna({"city": "unknown", "type": ""})
.show()

df.na.fill("unknown",)
.na.fill("",).show()

df.na.fill({"city": "unknown", "type": ""})
.show()
```

Frequently Asked Questions on fillna() and fill()

What is the difference between fillna() and fill() in PySpark?

In PySpark both fillna() and fill() are used to replace missing or null values of a DataFrame. Functionally they both perform same. One can choose either of these based on preference. These are used mainly for handling missing data in PySpark.

What happens if I use `fillna()` on a non-existent column?

If you use `fillna()` on a column that doesn't exist in the DataFrame, it will not raise an error. The method will simply have no effect on the DataFrame.

Can I fill in missing values with different replacement values for different columns?

we can specify different replacement values for different columns when using `fillna()`.
Example
`df.fillna({"zipcode":0,"population":50000})` will fill in missing values in the "zipcode" column with 0 and in the "population" column with 50000.

Are there any performance considerations while using `fillna()` on large DataFrames?

Filling missing values with `fillna()` can be resource-intensive for large DataFrames. It's essential to consider the performance impact, especially when working with big data. Optimize your code

and use appropriate caching or storage strategies to improve performance.

Conclusion

In this PySpark article, you have learned how to replace null/None values with zero or an empty string on integer and string columns respectively using `fill()` and `fillna()` transformation functions.

Happy Learning !!

Related Articles

Reference: