How can I use PySpark’s distinct function to drop duplicate rows in a dataset?

Authored by

stats writer

June 24, 2024

PySpark is a powerful tool for processing large datasets in a distributed computing environment. One of its functions, the "distinct" function, allows you to drop duplicate rows in a dataset. By using this function, you can easily remove any duplicate rows and ensure that your dataset contains only unique records. This can be particularly useful when dealing with messy data or when you want to perform analysis on a clean and accurate dataset. The distinct function in PySpark is a simple and efficient way to streamline your data and improve the accuracy of your analysis.

PySpark `distinct()` transformation is used to drop/remove the duplicate rows (all columns) from DataFrame and `dropDuplicates()` is used to drop rows based on selected (one or multiple) columns. `distinct()` and `dropDuplicates()` returns a new DataFrame. In this article, you will learn how to use distinct() and dropDuplicates() functions with PySpark example.

Before we start, first let's create a DataFrame with some duplicate rows and values on a few columns. We use this DataFrame to demonstrate how to get distinct multiple columns.

```
# Imports
from pyspark.sql import SparkSession
from pyspark.sql.functions import expr

# Create SparkSession
spark = SparkSession.builder.appName('SparkByExamples.com').getOrCreate()

# Prepare Data
data =

# Create DataFrame
columns=
df = spark.createDataFrame(data = data, schema = columns)
df.printSchema()
df.show(truncate=False)
```

Yields below output

```
# Output
+-------------+----------+------+
|employee_name|department|salary|
+-------------+----------+------+
|James |Sales |3000 |
|Michael |Sales |4600 |
|Robert |Sales |4100 |
```

```
|Maria |Finance |3000 |
|James |Sales |3000 |
|Scott |Finance |3300 |
|Jen |Finance |3900 |
|Jeff |Marketing |3000 |
|Kumar |Marketing |2000 |
|Saif |Sales |4100 |
+-------------+----------+------+
```

In the above table, record with employer name `James` has duplicate rows, As you notice we have 2 rows that have duplicate values on all columns and we have 4 rows that have duplicate values on `department` and `salary` columns.

1. Get Distinct Rows (By Comparing All Columns)

On the above DataFrame, we have a total of 10 rows with 2 rows having all values duplicated, performing distinct on this DataFrame should get us 9 after removing 1 duplicate row.

```
# Applying distinct() to remove duplicate rows
distinctDF = df.distinct()
print("Distinct count: "+str(distinctDF.count()))
distinctDF.show(truncate=False)
```

distinct() function on DataFrame returns a new DataFrame after removing the duplicate records. This example yields the below output.

```
# Output
Distinct count: 9
+-------------+----------+------+
|employee_name|department|salary|
+-------------+----------+------+
|James |Sales |3000 |
|Michael |Sales |4600 |
|Maria |Finance |3000 |
|Robert |Sales |4100 |
|Saif |Sales |4100 |
|Scott |Finance |3300 |
|Jeff |Marketing |3000 |
|Jen |Finance |3900 |
|Kumar |Marketing |2000 |
```

```
+-------------+----------+------+
```

Alternatively, you can also run dropDuplicates() function which returns a new DataFrame after removing duplicate rows.

```
# Applying dropDuplicates() to remove duplicates
df2 = df.dropDuplicates()
print("Distinct count: "+str(df2.count()))
df2.show(truncate=False)
```

2. PySpark Distinct of Selected Multiple Columns

PySpark doesn't have a distinct method that takes columns that should run distinct (drop duplicate rows on selected multiple columns) however, it provides another signature of `dropDuplicates()` transformation which takes multiple columns to eliminate duplicates.

Note that calling dropDuplicates() on DataFrame returns a new DataFrame with duplicate rows removed.

```
# Remove duplicates on selected columns using dropDuplicates()
dropDisDF = df.dropDuplicates()
print("Distinct count of department & salary : "+str(dropDisDF.count()))
dropDisDF.show(truncate=False)
```

Yields below output. If you notice the output, It dropped 2 records that are duplicates.

```
# Output
Distinct count of department & salary : 8
+-------------+----------+------+
|employee_name|department|salary|
+-------------+----------+------+
|Jen |Finance |3900 |
|Maria |Finance |3000 |
|Scott |Finance |3300 |
|Michael |Sales |4600 |
|Kumar |Marketing |2000 |
|Robert |Sales |4100 |
|James |Sales |3000 |
```

```
|Jeff |Marketing |3000 |
+-------------+----------+------+
```

3. Source Code to Get Distinct Rows

```
import pyspark
from pyspark.sql import SparkSession
from pyspark.sql.functions import expr
spark = SparkSession.builder.appName('SparkByExamples.com').getOrCreate()
```

data =
columns=
df = spark.createDataFrame(data = data, schema = columns)
df.printSchema()
df.show(truncate=False)

#Distinct
distinctDF = df.distinct()
print("Distinct count: "+str(distinctDF.count()))
distinctDF.show(truncate=False)

#Drop duplicates
df2 = df.dropDuplicates()
print("Distinct count: "+str(df2.count()))
df2.show(truncate=False)

#Drop duplicates on selected columns
dropDisDF = df.dropDuplicates()
print("Distinct count of department salary : "+str(dropDisDF.count()))
dropDisDF.show(truncate=False)
}

The complete example is available at GitHub for reference.

Frequently Asked Questions on distinct() and dropDuplicates()

How is `distinct()` different from `dropDuplicates()`?

`distinct()` and `dropDuplicates()` in PySpark are used to remove duplicate rows, but there is a subtle difference. `distinct()` considers all columns when identifying duplicates, while

`dropDuplicates()` allowing you to specify a subset of columns to determine uniqueness.

Can `distinct()` be used on specific columns only?

PySpark does not support specifying multiple columns with distinct() in order to remove the duplicates. We can use the dropDuplicates() transformation on specific columns to achieve the uniqueness of the columns.

Does `distinct()` maintain the original order of rows?

distinct() does not maintain the original order of the rows. To guarantee the original order we should perform additional sorting operations after distinct().

How does `distinct()` handle NULL values?

The `distinct()` function treats NULL values as equal, so if there are multiple rows with NULL values in all columns, only one of them will be retained after applying `distinct()`.

Can we use `distinct()` on a specific subset of rows based on a condition?

The `distinct()` function returns a new DataFrame with distinct rows, leaving the original DataFrame unchanged.So we can't use it on a specific subset of rows. If you want to modify the original DataFrame, you need to assign the result `distinct()` to a new variable or use the `inPlace` parameter if available.

Conclusion

In this PySpark SQL article, you have learned `distinct()` the method that is used to get the distinct values of rows (all columns) and also learned how to use `dropDuplicates()` to get the distinct and finally learned to use dropDuplicates() function to get distinct multiple columns.

Happy Learning !!

Related Articles: