What is the difference between orderBy() and sort() in PySpark?

Authored by

stats writer

June 24, 2024

OrderBy() and sort() are two methods used in PySpark for sorting the data in a specified order. The main difference between these two methods is that orderBy() sorts the data in ascending or descending order based on one or more specified columns, while sort() sorts the data in a natural ascending order. Additionally, orderBy() can handle null values and allows for specifying different sorting orders for different columns, whereas sort() does not have these capabilities. In summary, orderBy() is more versatile and flexible for sorting data in PySpark compared to sort().

You can use either `sort()` or `orderBy()` function of PySpark DataFrame to sort DataFrame by ascending or descending order based on single or multiple columns. Both methods take one or more columns as arguments and return a new DataFrame after sorting. You can also do sorting using PySpark SQL sorting functions.

In this article, I will explain all these different ways using PySpark examples. Note that pyspark.sql.DataFrame.orderBy() is an alias for .sort()

How to sort DataFrame by using Scala

Before we start, first let's create a DataFrame.

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

simpleData =
columns=
Create SparkSession

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

This Yields below output.

```
root
|-- employee_name: string (nullable = true)
|-- department: string (nullable = true)
|-- state: string (nullable = true)
|-- salary: integer (nullable = false)
|-- age: integer (nullable = false)
|-- bonus: integer (nullable = false)
```

+-------------+----------+-----+------+---+-----+
|employee_name|department|state|salary|age|bonus|
+-------------+----------+-----+------+---+-----+
James	Sales	NY	90000	34	10000
Michael	Sales	NY	86000	56	20000
Robert	Sales	CA	81000	30	23000
Maria	Finance	CA	90000	24	23000
Raman	Finance	CA	99000	40	24000
Scott	Finance	NY	83000	36	19000
Jen	Finance	NY	79000	53	15000
Jeff	Marketing	CA	80000	25	18000
Kumar	Marketing	NY	91000	50	21000
+-------------+----------+-----+------+---+-----+

1.DataFrame sorting using the sort() function

PySpark DataFrame class provides `sort()` function to sort on one or more columns. `sort()` takes a Boolean argument for ascending or descending order. To specify different sorting orders for different columns, you can use the parameter as a list. For example, to sort by "column1" in ascending order and "column2" in descending order as shown below.

```
# Sorting different columns in different orders
```

df.sort("column1", "column2", ascending=)

By default, it sorts by ascending order. `pyspark.sql.functions` module provides methods `asc()` and `desc()` to sort in ascending and descending orders respectively.

Syntax

```
DataFrame.sort(*cols, **kwargs)
```

Parameters: **cols:** str, list, or Column, optional

Other Parameters: **ascending**: bool or list, optional

Example

```
# Sorting DataFrame using sort()
```

df.sort("department","state").show(truncate=False)
df.sort(col("department"),col("state")).show(truncate=False)

The above two examples return the same below output, the first one takes the DataFrame column name as a string and the next takes columns in Column type. This table is sorted by the first `department` column and then the `state` column.

```
+-------------+----------+-----+------+---+-----+
|employee_name|department|state|salary|age|bonus|
+-------------+----------+-----+------+---+-----+
|Maria |Finance |CA |90000 |24 |23000|
|Raman |Finance |CA |99000 |40 |24000|
|Jen |Finance |NY |79000 |53 |15000|
|Scott |Finance |NY |83000 |36 |19000|
|Jeff |Marketing |CA |80000 |25 |18000|
|Kumar |Marketing |NY |91000 |50 |21000|
|Robert |Sales |CA |81000 |30 |23000|
|James |Sales |NY |90000 |34 |10000|
|Michael |Sales |NY |86000 |56 |20000|
+-------------+----------+-----+------+---+-----+
```

2.DataFrame sorting using orderBy() function

PySpark DataFrame also provides `orderBy()` function to sort on one or more columns. By default, it orders by ascending.

Syntax

```
DataFrame.orderBy(*cols, **kwargs)
```

Parameters: **cols:** str, list, or Column, optional

Other Parameters : **ascending**: bool or list, optional

Example

```
# Sorting DataFrame using orderBy()
```

df.orderBy("department","state").show(truncate=False)
df.orderBy(col("department"),col("state")).show(truncate=False)

This returns the same output as the previous section.

3.Sort by Ascending (ASC)

The `sort()` method allows you to specify the sorting column(s) and the sorting order (ascending or descending). If you want to specify the ascending order/sort explicitly on DataFrame, you can use the `asc` method of the `Column` function. for example,

```
# Sort DataFrame with asc
```

df.sort(df.department.asc(),df.state.asc()).show(truncate=False)
df.sort(col("department").asc(),col("state").asc()).show(truncate=False)
df.orderBy(col("department").asc(),col("state").asc()).show(truncate=False)

The above three examples return the same output.

```
+-------------+----------+-----+------+---+-----+
|employee_name|department|state|salary|age|bonus|
+-------------+----------+-----+------+---+-----+
|Maria |Finance |CA |90000 |24 |23000|
|Raman |Finance |CA |99000 |40 |24000|
|Jen |Finance |NY |79000 |53 |15000|
|Scott |Finance |NY |83000 |36 |19000|
|Jeff |Marketing |CA |80000 |25 |18000|
|Kumar |Marketing |NY |91000 |50 |21000|
|Robert |Sales |CA |81000 |30 |23000|
|James |Sales |NY |90000 |34 |10000|
|Michael |Sales |NY |86000 |56 |20000|
+-------------+----------+-----+------+---+-----+
```

4.Sort by Descending (DESC)

If you want to specify the sorting by descending order on DataFrame, you can use the `desc` method of the `Column` function. for example. From our example, let's use desc on the state column.

```
# Sort DataFrame with desc
```

df.sort(df.department.asc(),df.state.desc()).show(truncate=False)
df.sort(col("department").asc(),col("state").desc()).show(truncate=False)
df.orderBy(col("department").asc(),col("state").desc()).show(truncate=False)

This yields the below output for all three examples.

```
# Output
+-------------+----------+-----+------+---+-----+
|employee_name|department|state|salary|age|bonus|
+-------------+----------+-----+------+---+-----+
|Scott |Finance |NY |83000 |36 |19000|
|Jen |Finance |NY |79000 |53 |15000|
|Raman |Finance |CA |99000 |40 |24000|
|Maria |Finance |CA |90000 |24 |23000|
|Kumar |Marketing |NY |91000 |50 |21000|
|Jeff |Marketing |CA |80000 |25 |18000|
|James |Sales |NY |90000 |34 |10000|
|Michael |Sales |NY |86000 |56 |20000|
|Robert |Sales |CA |81000 |30 |23000|
+-------------+----------+-----+------+---+-----+
```

Besides `asc()` and `desc()` functions, PySpark also provides `asc_nulls_first()` and `asc_nulls_last()` and equivalent descending functions.

5. Using Raw SQL

Below is an example of how to sort DataFrame using raw SQL syntax.

```
# Sort using spark SQL
```

df.createOrReplaceTempView("EMP")
spark.sql("select employee_name,department,state,salary,age,bonus from EMP ORDER BY department asc").show(truncate=False)

The above two examples return the same output as above.

6. Dataframe Sorting Complete Example

```
# Imports

import pyspark
from pyspark.sql import SparkSession
from pyspark.sql.functions import col, asc,desc

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

simpleData =
columns=

df = spark.createDataFrame(data = simpleData, schema = columns)

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

df.sort("department","state").show(truncate=False)
df.sort(col("department"),col("state")).show(truncate=False)

df.orderBy("department","state").show(truncate=False)
df.orderBy(col("department"),col("state")).show(truncate=False)

df.sort(df.department.asc(),df.state.asc()).show(truncate=False)
df.sort(col("department").asc(),col("state").asc()).show(truncate=False)
df.orderBy(col("department").asc(),col("state").asc()).show(truncate=False)

df.sort(df.department.asc(),df.state.desc()).show(truncate=False)
df.sort(col("department").asc(),col("state").desc()).show(truncate=False)
df.orderBy(col("department").asc(),col("state").desc()).show(truncate=False)

df.createOrReplaceTempView("EMP")
spark.sql("select employee_name,department,state,salary,age,bonus from EMP ORDER BY department asc").show(truncate=False)
```

This complete example is also available at PySpark sorting GitHub project for reference.

7. Frequently Asked Questions on sort() and OrderBy()

What is the difference between orderBy() and sort() in PySpark?

In PySpark, both `orderBy()` and `sort()` are methods used for sorting rows in a DataFrame, and they serve the same purpose. However, there is no significant difference in terms of functionality or sorting capability between these two methods. Both can be used to sort rows based on one or more columns in ascending or descending order.

How do I sort by specific order in PySpark?

We can use `sort()` or `orderBy()` methods to sort DataFrame by "ascending" or "descending" order based on single or multiple columns.
Example

```
df.orderBy(col("department").asc(),col("state").asc()).show(truncate=False)
```
or
```
df.sort(col("department").asc(),col("state").asc()).show(truncate=False)
```

What is the default order of ORDER BY in PySpark?

orderBy() from pyspark.sql.DataFrame is used to sort the DataFrame in ascending or descending order. By default, it sorts in ascending order. We can specify the order by passing a boolean parameter to the method.
Example

```
df.orderBy("department", desc=True)
```

8. Conclusion

Here you have learned how to Sort PySpark DataFrame columns using `sort()`, `orderBy()` and using SQL sort functions and used this function with PySpark SQL along with Ascending and Descending sorting orders.

Happy Learning !!

Related Articles