How to Sum a Column in Spark Based on a Condition

Authored by

stats writer

January 2, 2026

Spark is a powerful tool that allows you to perform complex data manipulation and analysis. In this example, a Spark query is used to sum a column based on a condition. This is accomplished by using the SUM and WHERE clauses in the SELECT statement. This allows the user to get the total value for a particular column, but only for rows that meet a set of criteria. This is a great way to quickly analyze data and obtain meaningful results.

You can use the following methods to sum the values in a column of a PySpark DataFrame that meet a condition:

Method 1: Sum Values that Meet One Condition

```
from pyspark.sql.functions import sum

#sum values in points column for rows where team column is 'B'
df.filter(df.team=='B').agg(sum('points')).collect()
```

Method 2: Sum Values that Meet Multiple Conditions

```
from pyspark.sql.functions import sum

#sum values in points column for rows where team is 'B' and position is 'Guard'
df.filter((df.team=='B') & (df.position=='Guard')).agg(sum('points')).collect()
```

Method 3: Sum Values that Meet One of Several Conditions

```
from pyspark.sql.functions import sum

#sum values in points column for rows where team is 'B' or position is 'Guard'
df.filter((df.team=='B') | (df.position=='Guard')).agg(sum('points')).collect()
```

The following examples show how to use each method in practice with the following PySpark DataFrame that contains information about various basketball players:

```
from pyspark.sql import SparkSession
spark = SparkSession.builder.getOrCreate()

#define data
data = ,
,
,
,
```

,
,
,
]

#define column names
columns =

#create dataframe using data and column names
df = spark.createDataFrame(data, columns)

#view dataframe
df.show()

```
+----+--------+------+
|team|position|points|
+----+--------+------+
| A| Guard| 11|
| A| Guard| 8|
| A| Forward| 22|
| A| Forward| 22|
| B| Guard| 14|
| B| Guard| 14|
| B| Forward| 13|
| B| Forward| 7|
+----+--------+------+
```

Example 1: Sum Values that Meet One Condition

We can use the following syntax to sum the values in the **points** column where the corresponding value in the **team** column is equal to B:

from pyspark.sql.functions import sum

#sum values in points column for rows where team column is 'B'
df.filter(df.team=='B').agg(sum('points')).collect()

48

We can see that the sum of values in the **points** column for players on team B is **48**.

Example 2: Sum Values that Meet Multiple Conditions

We can use the following syntax to sum the values in the **points** column where the corresponding value in the **team** column is equal to B *and* the value in the **position** column is equal to Guard:

```
from pyspark.sql.functions import sum

#sum values in points column for rows where team is 'B' and position is 'Guard'
df.filter((df.team=='B') & (df.position=='Guard')).agg(sum('points')).collect()
```

```
28
```

We can see that the sum the values in the **points** column where the corresponding value in the **team** column is equal to B *and* the value in the **position** column is equal to Guard is **28**.

Example 3: Sum Values that Meet One of Several Conditions

We can use the following syntax to sum the values in the **points** column where the corresponding value in the **team** column is equal to B or the value in the **position** column is equal to Guard:

```
from pyspark.sql.functions import sum

#sum values in points column for rows where team is 'B' or position is 'Guard'
df.filter((df.team=='B') | (df.position=='Guard')).agg(sum('points')).collect()
```

```
67
```

We can see that the sum the values in the **points** column where the corresponding value in the **team** column is equal to B or the value in the **position** column is equal to Guard is **67**.

The following tutorials explain how to perform other common tasks in PySpark: