How to Use Equivalent of Pandas value_counts()?

Authored by

stats writer

June 21, 2024

The groupBy() and count() functions in PySpark provide the equivalent functionality of Pandas' value_counts() function. These functions can be used together to group data by a specified column and count the number of occurrences in each group. This is useful for getting a quick summary of the distribution of values in a column within a PySpark DataFrame. Furthermore, the orderBy() function can be added to sort the values in descending order based on the count column.

PySpark: Use Equivalent of Pandas value_counts()

You can use the value_counts() function in pandas to count the occurrences of each unique value in a given column of a DataFrame.

You can use the following methods to replicate the value_counts() function in a PySpark DataFrame:

Method 1: Count Occurrences of Each Unique Value in Column

#count occurrences of each unique value in 'team' column
df.groupBy('team').count().show()

Method 2: Count Occurrences of Each Unique Value in Column and Sort Ascending

#count occurrences of each unique value in 'team' column and sort ascending
df.groupBy('team').count().orderBy('count').show()

Method 3: Count Occurrences of Each Unique Value in Column and Sort Descending

#count occurrences of each unique value in 'team' column and sort descending
df.groupBy('team').count().orderBy('count', ascending=False).show()

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| 30|
| B| Forward| 22|
| B| Forward| 22|
| B| Guard| 14|
| B| Guard| 10|
| C| Forward| 13|
| D| Forward| 7|
| D| Forward| 16|
+----+--------+------+
```

Example 1: Count Occurrences of Each Unique Value in Column

We can use the following syntax to count the number of occurrences of each unique value in the team column of the DataFrame:

```
#count occurrences of each unique value in 'team' column
df.groupBy('team').count().show()

+----+-----+
|team|count|
+----+-----+
| A| 2|
| B| 4|
| C| 1|
| D| 2|
+----+-----+
```

The output displays the count of each unique value in the team column.

By default, the rows are sorted in alphabetical order by the unique values in the team column.

Example 2: Count Occurrences of Each Unique Value in Column and Sort Ascending

We can use the following syntax to count the number of occurrences of each unique value in the team column of the DataFrame and sort by count ascending:

```
#count occurrences of each unique value in 'team' column and sort ascending
df.groupBy('team').count().orderBy('count').show()

+----+-----+
|team|count|
+----+-----+
|  C|    1|
|  A|    2|
|  D|    2|
|  B|    4|
+----+-----+
```

The output displays the count of each unique value in the team column, sorted by count in ascending order.

Example 3: Count Occurrences of Each Unique Value in Column and Sort Descending

We can use the following syntax to count the number of occurrences of each unique value in the team column of the DataFrame and sort by count descending:

```
#count occurrences of each unique value in 'team' column and sort descending
df.groupBy('team').count().orderBy('count', ascending=False).show()
```

```
+----+-----+
|team|count|
+----+-----+
|  B|    4|
|  A|    2|
|  D|    2|
|  C|    1|
+----+-----+
```

The output displays the count of each unique value in the team column, sorted by count in descending order.

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