How to Create a PySpark DataFrame from a List

Authored by

stats writer

February 4, 2026

Creating a DataFrame from a list in PySpark is a simple and efficient process. First, the list needs to be converted into a PySpark RDD (Resilient Distributed Dataset) using the "parallelize" function. Then, the RDD can be converted into a DataFrame by using the "toDF" function, which takes in the column names as parameters. This will create a DataFrame with each element in the list as a row, and the specified column names as the column headers. This method is useful for quickly converting a list of data into a structured and organized DataFrame in PySpark.

PySpark: Create DataFrame from List (With Examples)

You can use the following methods to create a DataFrame from a list in PySpark:

Method 1: Create DataFrame from List

from pyspark.sql.types import IntegerType

#define list of data
data =

#create DataFrame with one column
df = spark.createDataFrame(data, IntegerType())

Method 2: Create DataFrame from List of Lists

#define list of lists
data = ,

,

,

,

,

]

#define column names
columns =

#create DataFrame with three columns
df = spark.createDataFrame(data, columns)

The following examples show how to use each method in practice.

Example 1: Create PySpark DataFrame from List

The following code shows how to create a PySpark DataFrame from a single list:

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

from pyspark.sql.types import IntegerType

#define list of data
data =

```
#create DataFrame with one column
df = spark.createDataFrame(data, IntegerType())

#view DataFrame
df.show()

+-----+
|value|
+-----+
| 10|
| 15|
| 22|
| 27|
| 28|
| 40|
+-----+
```

The resulting DataFrame contains one column of integer values that came directly from the values in the list.

Note: In this example we specified that the column should be an integer, but you could instead use StringType, FloatType, etc. to specify a different data type. Refer to the PySpark for a complete list of data

types.

Also note that you can also use the withColumnRenamed function to rename the column in the DataFrame:

```
#rename column name to 'some_data'
df = df.withColumnRenamed('value', 'some_data')

#view updated DataFrame
df.show()

+---------+
|some_data|
+---------+
| 10|
| 15|
| 22|
| 27|
| 28|
| 40|
+---------+
```

Notice that the column name has been renamed to some_data.

Example 2: Create PySpark DataFrame from List of Lists

We can use the following syntax to create a PySpark DataFrame from a list of lists:

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

#define list of lists
data = ,
,
,
,
]

#define column names
columns =

#create DataFrame
df = spark.createDataFrame(data, columns)

#view DataFrame
df.show()

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

```
|team|conference|points|
+----+----------+------+
| A| East| 11|
| A| East| 8|
| A| East| 10|
| B| West| 6|
| B| West| 6|
| C| East| 5|
| C| East| 15|
| C| West| 31|
| D| West| 24|
+----+----------+------+
```

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