How can PySpark’s JSON functions be used, and what are some examples of their usage?

Authored by

stats writer

June 24, 2024

PySpark's JSON functions are a set of built-in tools that allow users to easily manipulate and analyze JSON data within a Spark environment. These functions can be used to extract, transform, and load data from JSON files, as well as perform various operations on the data.

Some examples of how PySpark's JSON functions can be used include parsing JSON data into columns, converting JSON arrays into nested columns, and querying JSON data using SQL-like syntax. These functions can also be used to join JSON data with other data sources, filter and aggregate JSON data, and perform complex transformations on JSON objects.

Overall, PySpark's JSON functions offer a powerful and efficient way to work with JSON data in a Spark environment, making it easier for users to process and analyze large amounts of data in a variety of use cases.

In PySpark, the JSON functions allow you to work with JSON data within DataFrames. These functions help you parse, manipulate, and extract data from JSON columns or strings. These functions can also be used to convert JSON to a struct, map type, etc. I will explain the most used JSON SQL functions with Python examples in this article.

1. PySpark JSON Functions

JSON Functions	Description
from_json()	Converts JSON string into Struct type or Map type.
to_json()	Converts MapType or Struct type to JSON string.
json_tuple()	Extract the Data from JSON and create them as a new columns.
get_json_object()	Extracts JSON element from a JSON string based on json path specified.
schema_of_json()	Create schema string from JSON string

PySpark JSON Functions

1.1. Create DataFrame with Column containing JSON String

To explain these JSON functions first, let's create a DataFrame with a column containing JSON string.

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

```
jsonString="""{"Zipcode":704,"ZipCodeType":"STANDARD","City":"PARC PARQUE","State":"PR"}"""
```

df=spark.createDataFrame(,)
df.show(truncate=False)

//+---+--+
//|id |value |
//+---+--+
//|1 |{"Zipcode":704,"ZipCodeType":"STANDARD","City":"PARC PARQUE","State":"PR"}|
//+---+--+

2. PySpark JSON Functions Examples

2.1. from_json()

PySpark `from_json()` function is used to convert JSON string into Struct type or Map type. The below example converts JSON string to Map key-value pair. I will leave it to you to convert to struct type. Refer, Convert JSON string to Struct type column.

```
#Convert JSON string column to Map type
from pyspark.sql.types import MapType,StringType
from pyspark.sql.functions import from_json
df2=df.withColumn("value",from_json(df.value,MapType(StringType(),StringType()))
)
df2.printSchema()
df2.show(truncate=False)
```

//root
// |-- id: integer (nullable = false)
// |-- value: map (nullable = true)
// | |-- key: string
// | |-- value: string (valueContainsNull = true)

//+---+--+
//|id |value |
//+---+--+
//|1 ||
//+---+--+

2.2. to_json()

`to_json()` function is used to convert DataFrame columns MapType or Struct type to JSON string. Here, I am using df2 that created from above `from_json()` example.

```
from pyspark.sql.functions import to_json,col
df2.withColumn("value",to_json(col("value")))
.show(truncate=False)
```

```
//+---+--------------------------------------------------------------------------+
//|id |value |
//+---+--------------------------------------------------------------------------+
//|1 |{"Zipcode":"704","ZipCodeType":"STANDARD","City":"PARC PARQUE","State":"PR"}|
//+---+--------------------------------------------------------------------------+
```

2.3. json_tuple()

Function `json_tuple()` is used the query or extract the elements from JSON column and create the result as a new columns.

```
from pyspark.sql.functions import json_tuple
df.select(col("id"),json_tuple(col("value"),"Zipcode","ZipCodeType","City"))
.toDF("id","Zipcode","ZipCodeType","City")
.show(truncate=False)
```

```
//+---+-------+-----------+-----------+
//|id |Zipcode|ZipCodeType|City |
//+---+-------+-----------+-----------+
//|1 |704 |STANDARD |PARC PARQUE|
//+---+-------+-----------+-----------+
```

2.4. get_json_object()

`get_json_object()` is used to extract the JSON string based on path from the JSON column.

```
from pyspark.sql.functions import get_json_object
df.select(col("id"),get_json_object(col("value"),"$.ZipCodeType").alias("ZipCode
Type"))
```

```
.show(truncate=False)
```

//+---+-----------+
//|id |ZipCodeType|
//+---+-----------+
//|1 |STANDARD |
//+---+-----------+

2.5. schema_of_json()

Use schema_of_json() to create schema string from JSON string column.

```
from pyspark.sql.functions import schema_of_json,lit
schemaStr=spark.range(1)
.select(schema_of_json(lit("""{"Zipcode":704,"ZipCodeType":"STANDARD","City":"PA
RC PARQUE","State":"PR"}""")))
.collect()
print(schemaStr)
```

//struct<City:string,State:string,ZipCodeType:string,Zipcode:bigint>

3. Complete Example of PySpark JSON Functions

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

```
jsonString="""{"Zipcode":704,"ZipCodeType":"STANDARD","City":"PARC
PARQUE","State":"PR"}"""
```
df=spark.createDataFrame(,)
df.show(truncate=False)

#Convert JSON string column to Map type
from pyspark.sql.types import MapType,StringType
from pyspark.sql.functions import from_json
df2=df.withColumn("value",from_json(df.value,MapType(StringType(),StringType())))
df2.printSchema()
df2.show(truncate=False)

from pyspark.sql.functions import to_json,col

```
df2.withColumn("value",to_json(col("value")))
.show(truncate=False)

from pyspark.sql.functions import json_tuple
df.select(col("id"),json_tuple(col("value"),"Zipcode","ZipCodeType","City"))
.toDF("id","Zipcode","ZipCodeType","City")
.show(truncate=False)

from pyspark.sql.functions import get_json_object
df.select(col("id"),get_json_object(col("value"),"$.ZipCodeType").alias("ZipCodeType"))
.show(truncate=False)

from pyspark.sql.functions import schema_of_json,lit
schemaStr=spark.range(1)
.select(schema_of_json(lit("""{"Zipcode":704,"ZipCodeType":"STANDARD","City":"PARC
PARQUE","State":"PR"}""")))
.collect()
print(schemaStr)
```

Related Article

References