How can JSON functions in PySpark be used and what are some examples of their implementation?

Authored by

stats writer

June 24, 2024

JSON (JavaScript Object Notation) is a popular data format for storing and transmitting data. PySpark, a powerful and efficient data processing framework, provides built-in functions for working with JSON data. These functions allow users to efficiently manipulate, query, and analyze JSON data within PySpark.

One of the main uses of JSON functions in PySpark is to convert JSON data into a structured format, such as a DataFrame, which can then be easily queried and analyzed. This is particularly useful when working with large datasets that are in JSON format.

Some examples of JSON functions in PySpark include "from_json", which converts a JSON string into a DataFrame, "to_json", which converts a DataFrame into a JSON string, and "get_json_object", which extracts a specific value from a JSON string.

Other useful JSON functions in PySpark include "explode", which splits a JSON array into separate rows, "json_tuple", which extracts multiple values from a JSON string, and "schema_of_json", which infers the schema of a JSON string and returns a StructType object.

Overall, the implementation of JSON functions in PySpark allows for efficient and flexible handling of JSON data, making it a valuable tool for data engineers and data scientists alike.

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("ZipCodeType"))
.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":"PARC 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