

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

Authored by  
**stats writer**

June 24, 2024

## RECOMMENDED CITATION

stats writer (2024). *How can the JSON functions in PySpark be used and what are some examples of their usage?*. PSYCHOLOGICAL SCALES. Retrieved from <https://scales.arabpsychology.com/?p=151252>

JSON functions in PySpark are designed to manipulate and transform data in JSON format. These functions can be used to extract, filter, and transform data within JSON objects, making it easier to analyze and process large amounts of data. For example, the "get\_json\_object" function can be used to extract a specific value from a JSON object, while the "json\_tuple" function can be used to create a tuple of values from a JSON object. Additionally, the "from\_json" function can be used to parse a JSON string into a specified schema, allowing for easier data analysis. Overall, the JSON functions in PySpark provide a convenient and efficient way to handle data in JSON format, making it an essential tool for data processing and analysis.

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.

Advertisements

## 1. PySpark JSON Functions

JSON Functions	Description
<code>from_json()</code>	Converts JSON string into Struct type or Map type.
<code>to_json()</code>	Converts MapType or Struct type to JSON string.
<code>json_tuple()</code>	Extract the Data from JSON and create them as a new columns.
<code>get_json_object()</code>	Extracts JSON element from a JSON string based on json path specified.
<code>schema_of_json()</code>	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 to query or extract the elements from a JSON column and create the result as a new column.

```

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 a 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":"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