How can the MapType data type (Dict) be used in PySpark, and what are some examples of its usage?

Authored by

stats writer

June 24, 2024

The MapType data type, also known as Dict, is a type of data structure in PySpark that allows for the storage and manipulation of key-value pairs. This data type can be used in various ways in PySpark, such as creating dictionaries, mapping functions, and aggregating data.

One example of using the MapType data type in PySpark is in creating dictionaries to store and organize data. For instance, a dictionary can be created to represent a person's information, with the keys being the different attributes (e.g. name, age, gender) and the corresponding values being the data for each attribute.

Another example is using the MapType data type to map functions to a dataset. This allows for the efficient and effective application of functions to specific columns or rows of the dataset, making data processing and analysis more streamlined.

Additionally, the MapType data type can be used for aggregating data, such as calculating the mean or sum of values for a particular key in a dataset. This is useful for summarizing and analyzing large datasets.

Overall, the MapType data type is a versatile tool in PySpark for organizing, manipulating, and analyzing data in a structured and efficient manner.

PySpark `MapType` (also called map type) is a data type to represent Python Dictionary (`dict`) to store key-value pair, a MapType object comprises three fields, keyType (a `DataType`), valueType (a `DataType`) and valueContainsNull (a `BooleanType`).

What is PySpark MapType

PySpark MapType is used to represent map key-value pair similar to python Dictionary (Dict), it extends DataType class which is a superclass of all types in PySpark and takes two mandatory arguments `keyType` and `valueType` of type DataType and one optional boolean argument `valueContainsNull`. keyType and valueType can be any type that extends the DataType class. for e.g `StringType`, `IntegerType`, `ArrayType`, `MapType`, `StructType` (struct) e.t.c.

1. Create PySpark MapType

In order to use MapType data type first, you need to import it from `pyspark.sql.types.MapType` and use `MapType()` constructor to create a map object.

```
from pyspark.sql.types import StringType, MapType
mapCol = MapType(StringType(),StringType(),False)
```

MapType Key Points:

2. Create MapType From StructType

Let's see how to create a `MapType` by using PySpark StructType & StructField, `StructType()` constructor takes list of StructField, StructField takes a fieldname and type of the value.

```
from pyspark.sql.types import StructField, StructType, StringType, MapType
schema = StructType()
```

Now let's create a DataFrame by using above StructType schema.

```
from pyspark.sql import SparkSession
spark = SparkSession.builder.appName('SparkByExamples.com').getOrCreate()
dataDictionary =
df = spark.createDataFrame(data=dataDictionary, schema = schema)
df.printSchema()
df.show(truncate=False)
```

df.printSchema() yields the Schema and df.show() yields the DataFrame output.

```
root
|-- Name: string (nullable = true)
|-- properties: map (nullable = true)
| |-- key: string
| |-- value: string (valueContainsNull = true)

+----------+-----------------------------+
|Name |properties |
+----------+-----------------------------+
|James ||
|Michael | |
|Robert | |
|Washington| |
|Jefferson | |
+----------+-----------------------------+
```

3. Access PySpark MapType Elements

Let's see how to extract the key and values from the PySpark DataFrame Dictionary column. Here

I have used PySpark map transformation to read the values of `properties` (MapType column)

```
df3=df.rdd.map(lambda x:
(x.name,x.properties,x.properties))
.toDF()
df3.printSchema()
df3.show()
```

root
|-- name: string (nullable = true)
|-- hair: string (nullable = true)
|-- eye: string (nullable = true)

+----------+-----+-----+
| name| hair| eye|
+----------+-----+-----+
James	black	brown
Michael	brown	null
Robert	red	black
Washington	grey	grey
Jefferson	brown	
+----------+-----+-----+

Let's use another way to get the value of a key from Map using `getItem()` of `Column` type, this method takes a key as an argument and returns a value.

```
df.withColumn("hair",df.properties.getItem("hair"))
.withColumn("eye",df.properties.getItem("eye"))
.drop("properties")
.show()
```

df.withColumn("hair",df.properties)
.withColumn("eye",df.properties)
.drop("properties")
.show()

4. Functions

Below are some of the MapType Functions with examples.

4.1 - explode

```
from pyspark.sql.functions import explode
df.select(df.name,explode(df.properties)).show()
```

```
+----------+----+-----+
| name| key|value|
+----------+----+-----+
| James| eye|brown|
| James|hair|black|
| Michael| eye| null|
| Michael|hair|brown|
| Robert| eye|black|
| Robert|hair| red|
|Washington| eye| grey|
|Washington|hair| grey|
| Jefferson| eye| |
| Jefferson|hair|brown|
+----------+----+-----+
```

4.2 map_keys() - Get All Map Keys

```
from pyspark.sql.functions import map_keys
df.select(df.name,map_keys(df.properties)).show()
```

```
+----------+--------------------+
| name|map_keys(properties)|
+----------+--------------------+
| James| |
| Michael| |
| Robert| |
|Washington| |
| Jefferson| |
+----------+--------------------+
```

In case if you wanted to get all map keys as Python List. **WARNING**: **This runs very slow**.

```
from pyspark.sql.functions import explode,map_keys
```

```
keysDF = df.select(explode(map_keys(df.properties))).distinct()
keysList = keysDF.rdd.map(lambda x:x).collect()
print(keysList)
#
```

4.3 map_values() - Get All map Values

```
from pyspark.sql.functions import map_values
df.select(df.name,map_values(df.properties)).show()
```

```
+----------+----------------------+
| name|map_values(properties)|
+----------+----------------------+
| James| |
| Michael| |
| Robert| |
|Washington| |
| Jefferson| |
+----------+----------------------+
```

Conclusion

MapType is a map data structure that is used to store key key-value pairs similar to Python Dictionary (Dic), keys and values type of map should be of a type that extends DataType. Key won't accept null/None values whereas map of the key can have None/Null value.

Related Articles

References