How can I select specific columns from a PySpark DataFrame?

Authored by

stats writer

June 24, 2024

In order to select specific columns from a PySpark DataFrame, one must use the "select" function and specify the columns by name or using a list. This function will return a new DataFrame with only the specified columns, allowing for efficient data manipulation and analysis. Additionally, the "select" function can be combined with other functions such as "filter" and "groupBy" to further refine the selection of columns. By using this method, users can easily access and manipulate the desired columns within their PySpark DataFrame.

In PySpark, `select()` function is used to select single, multiple, column by index, all columns from the list and the nested columns from a DataFrame, PySpark select() is a transformation function hence it returns a new DataFrame with the selected columns.

Create a Dataframe.

```
import pyspark
from pyspark.sql import SparkSession
```

spark = SparkSession.builder.appName('SparkByExamples.com').getOrCreate()

Data
data =

Column names
columns =

# Create DataFrame
df = spark.createDataFrame(data = data, schema = columns)
df.show(truncate=False)

1. Select Single & Multiple Columns From PySpark

You can select the single or multiple columns of the DataFrame by passing the column names you wanted to select to the `select()` function. Since DataFrame is immutable, this creates a new DataFrame with selected columns. show() function is used to show the Dataframe contents.

Below are ways to select single, multiple or all columns.

```
# Select columns by different ways
df.select("firstname","lastname").show()
df.select(df.firstname,df.lastname).show()
df.select(df,df).show()
```

# By using col() function
from pyspark.sql.functions import col
df.select(col("firstname"),col("lastname")).show()

# Select columns by regular expression
df.select(df.colRegex("`^.*name*`")).show()

2. Select All Columns From List

Sometimes you may need to select all DataFrame columns from a Python list. In the below example, we have all columns in the `columns` list object.

```
# Select All columns from List
df.select(*columns).show()
```

# Select All columns
df.select().show()
df.select("*").show()

3. Select Columns by Index

Using a python list features, you can select the columns by index.

```
#Selects first 3 columns and top 3 rows
df.select(df.columns).show(3)
```

#Selects columns 2 to 4 and top 3 rows
df.select(df.columns).show(3)

4. Select Nested Struct Columns from PySpark

If you have a nested struct (StructType) column on PySpark DataFrame, you need to use an explicit column qualifier in order to select. If you are new to PySpark and you have not learned StructType yet, I would recommend skipping the rest of the section or first Understand PySpark StructType before you proceed.

First, let's create a new DataFrame with a struct type.

```
# Create DataFrame with nested columns
data =
```

from pyspark.sql.types import StructType,StructField, StringType
schema = StructType()),
StructField('state', StringType(), True),
StructField('gender', StringType(), True)
])
df2 = spark.createDataFrame(data = data, schema = schema)
df2.printSchema()
df2.show(truncate=False) # shows all columns

Produces the following schema output. If you observe, the column `name` is of a struct type, containing the columns firstname, middlename, and lastname.

```
# output:
root
|-- name: struct (nullable = true)
| |-- firstname: string (nullable = true)
| |-- middlename: string (nullable = true)
| |-- lastname: string (nullable = true)
|-- state: string (nullable = true)
|-- gender: string (nullable = true)
```

+----------------------+-----+------+
|name |state|gender|
+----------------------+-----+------+
	OH	M
	NY	F
	OH	F
	NY	M
	NY	M
	OH	M
+----------------------+-----+------+

To select the struct column, just specify the column name in the select().

```
# Select struct column
df2.select("name").show(truncate=False)
```

This returns the struct column `name` as is.

```
# Output:
+----------------------+
|name |
+----------------------+
| |
| |
| |
| |
| |
||
+----------------------+
```

In order to select the specific column from a nested struct, you need to explicitly qualify the nested struct column name.

```
# Select child columns
df2.select("name.firstname","name.lastname").show(truncate=False)
```

This results the `firstname` and `lastname` as two seperate columns from the name struct column.

```
# Output:
+---------+--------+
|firstname|lastname|
+---------+--------+
|James |Smith |
|Anna | |
|Julia |Williams|
|Maria |Jones |
|Jen |Brown |
|Mike |Williams|
+---------+--------+
```

To get all columns from the struct column, specify name.*

```
# Select all child columns
df2.select("name.*").show(truncate=False)
```

Output:

```
# Output
+---------+----------+--------+
|firstname|middlename|lastname|
+---------+----------+--------+
|James |null |Smith |
|Anna |Rose | |
|Julia | |Williams|
|Maria |Anne |Jones |
|Jen |Mary |Brown |
|Mike |Mary |Williams|
+---------+----------+--------+
```

5. Complete Example

```
import pyspark
from pyspark.sql import SparkSession
```

spark = SparkSession.builder.appName('SparkByExamples.com').getOrCreate()

data =

columns =
df = spark.createDataFrame(data = data, schema = columns)
df.show(truncate=False)

df.select("firstname").show()

df.select("firstname","lastname").show()

#Using Dataframe object name
df.select(df.firstname,df.lastname).show()

# Using col function
from pyspark.sql.functions import col
df.select(col("firstname"),col("lastname")).show()

data =

from pyspark.sql.types import StructType,StructField, StringType

```
schema = StructType()),
StructField('state', StringType(), True),
StructField('gender', StringType(), True)
])

df2 = spark.createDataFrame(data = data, schema = schema)
df2.printSchema()
df2.show(truncate=False) # shows all columns

df2.select("name").show(truncate=False)
df2.select("name.firstname","name.lastname").show(truncate=False)
df2.select("name.*").show(truncate=False)
```

This example is also available at PySpark github project.

6. Conclusion

In this article, you have learned select() is a transformation function of the DataFrame and is used to select single, multiple columns, select all columns from the list, select by index, and finally select nested struct columns, you have also learned how to select nested elements from the DataFrame.

Related Articles

Happy Learning !!