02_Basic_Part2

March 28, 2020

0.1 Load and show

[1]:
```
import pyspark
print(pyspark.__version__)
```

```
2.4.5
```

[2]:
```
from pyspark.sql import SparkSession
```

[3]:
```
spark = SparkSession.builder.appName("Basics02").getOrCreate()
```

[4]:
```
path = "Python-and-Spark-for-Big-Data-master/Spark_DataFrames/people.json"
df = spark.read.json(path)
```

[5]:
```
df.show()
```

```
+----+-------+
| age|   name|
+----+-------+
|null|Michael|
|  30|   Andy|
|  19| Justin|
+----+-------+
```

[6]:
```
df.columns
```

[6]:
```
['age', 'name']
```

[7]:
```
df.describe().show()
```

```
+-------+------------------+-------+
|summary|               age|   name|
+-------+------------------+-------+
|  count|                 2|      3|
|   mean|              24.5|   null|
| stddev|7.7781745930520225|   null|
|    min|                19|   Andy|
|    max|                30|Michael|
```

```
+-------+-----------------+-------+
```

[8]:
```
df.printSchema()
```

```
root
 |-- age: long (nullable = true)
 |-- name: string (nullable = true)
```

0.2 Select columns

[9]:
```
df["age"] # return a column object, helpful for filtering later
```

[9]:
```
Column<b'age'>
```

[10]:
```
type(df["age"])
```

[10]:
```
pyspark.sql.column.Column
```

[11]:
```
df.select("age") # return a data frame with a single column
```

[11]:
```
DataFrame[age: bigint]
```

[12]:
```
type(df.select("age"))
```

[12]:
```
pyspark.sql.dataframe.DataFrame
```

DataFrame object is more flexible than column object. It has more methods. For example, the show method

[13]:
```
df.select("age").show()
```

```
+----+
| age|
+----+
|null|
|  30|
|  19|
+----+
```

0.3 Select rows

[14]:
```
df.head(2) # return the first 2 rows
```

[14]:
```
[Row(age=None, name='Michael'), Row(age=30, name='Andy')]
```

[15]:
```
# each row is a row object
type(df.head(2)[0])
```

[15]:
```
pyspark.sql.types.Row
```

0.4 Select multiple columns

[16]:
```
df.select(["age", "name"]).show() # return a data frame of 2 columns
```

```
+----+-------+
| age|   name|
+----+-------+
|null|Michael|
|  30|   Andy|
|  19| Justin|
+----+-------+
```

0.5 Create new column and rename column

[17]:
```
df.withColumn("double_age", df["age"]*2).show()   #This doesn't change the␣
 ↪original data
```

```
+----+-------+----------+
| age|   name|double_age|
+----+-------+----------+
|null|Michael|      null|
|  30|   Andy|        60|
|  19| Justin|        38|
+----+-------+----------+
```

[18]:
```
df.show() # withColumn is not an inplace operation, the original data is␣
 ↪unchanged
```

```
+----+-------+
| age|   name|
+----+-------+
|null|Michael|
|  30|   Andy|
|  19| Justin|
+----+-------+
```

[19]:
```
df.withColumnRenamed("age", "new_age").show() #rename, also doesn't change the␣
 ↪original data frame
```

```
+-------+-------+
|new_age|   name|
+-------+-------+
|   null|Michael|
```

```
|     30|   Andy|
|     19| Justin|
+-------+-------+
```

0.6 Using sql with spark

[20]:
```
# first: regsiter the data frame as sql temporary view
df.createOrReplaceTempView("people")
```

[21]:
```
# using sql command directly, returns data frame
results = spark.sql("SELECT * from people")
```

[22]:
```
results.show()
```

```
+----+-------+
| age|   name|
+----+-------+
|null|Michael|
|  30|   Andy|
|  19| Justin|
+----+-------+
```

[23]:
```
new_results = spark.sql("SELECT * from people WHERE age=30")
```

[24]:
```
new_results.show()
```

```
+---+----+
|age|name|
+---+----+
| 30|Andy|
+---+----+
```