WORKING WITH SCHEMA OF DATAFRAME

INFERRING A SCHEMA (Csv File)

INFERRING THE SCHEMA OF CSV FILE WITHOUT HEADER OPTION

If we do not ask to inferSchema for a csv file then all columns will be taken as same datatype.

scala> **val ratingsdf=spark.read.option("header","true").csv("/user/root/csvs/ratings.csv")**

scala> **ratingsdf.printSchema**

```
root
 |-- userId: string (nullable = true)
 |-- movieId: string (nullable = true)
 |-- rating: string (nullable = true)
 |-- timestamp: string (nullable = true)
```

We observe that all columns are taken as string, but if you look at data of ratings.csv we have numeric columns.

```
+------+-------+------+----------+
|userId|movieId|rating| timestamp|
+------+-------+------+----------+
|     1|      2|   3.5|1112486027|
|     1|     29|   3.5|1112484676|
|     1|     32|   3.5|1112484819|
|     1|     47|   3.5|1112484727|
|     1|     50|   3.5|1112484580|
|     1|    112|   3.5|1094785740|
|     1|    151|   4.0|1094785734|
|     1|    223|   4.0|1112485573|
|     1|    253|   4.0|1112484940|
|     1|    260|   4.0|1112484826|
|     1|    293|   4.0|1112484703|
|     1|    296|   4.0|1112484767|
|     1|    318|   4.0|1112484798|
|     1|    337|   3.5|1094785709|
|     1|    367|   3.5|1112485980|
|     1|    541|   4.0|1112484603|
|     1|    589|   3.5|1112485557|
|     1|    593|   3.5|1112484661|
|     1|    653|   3.0|1094785691|
|     1|    919|   3.5|1094785621|
+------+-------+------+----------+
```

Now let us see inferSchema

scala> **val ratingsdf=spark.read.option("header","true").option("inferSchema","true").csv("/user/root/csvs/ratings.csv")**

scala> **ratingsdf.printSchema**

```
root
 |-- userId: integer (nullable = true)
 |-- movieId: integer (nullable = true)
 |-- rating: double (nullable = true)
 |-- timestamp: integer (nullable = true)
```


However here it has not correctly inferred the schema, because the timestamp column has been taken as integer.

DEFINING SCHEMA MANUALY

On Scala

First read the devices.json file and see the schema, check what datatype the column release_dt is showing.

Now let us create schema explicitly

scala> **import org.apache.spark.sql.types._**

scala> **val devColumns = List(**

StructField("devnum",LongType),

StructField("make",StringType),

StructField("model",StringType),

StructField("release_dt",TimestampType),

StructField("dev_type",StringType))

scala> **val devSchema = StructType(devColumns)**

scala> **val devDF = spark.read.schema(devSchema).json("/user/root/jsons/devices.json")**

scala> **devDF.printSchema**

```
root
 |-- devnum: long (nullable = true)
 |-- make: string (nullable = true)
 |-- model: string (nullable = true)
 |-- release_dt: timestamp (nullable = true)
 |-- dev_type: string (nullable = true)
```

On Python

>>> **from pyspark.sql.types import ***

>>> **devColumns = [**

StructField("devnum",LongType()),

StructField("make",StringType()),

StructField("model",StringType()),

StructField("release_dt",TimestampType()),

StructField("dev_type",StringType())]

>>> **devschema = StructType(devColumns)**

>>> **devDF = spark.read.schema(devschema).json("/user/root/jsons/devices.json")**

>>> **devDF.printSchema()**

```
root
 |-- devnum: long (nullable = true)
 |-- make: string (nullable = true)
 |-- model: string (nullable = true)
 |-- release_dt: timestamp (nullable = true)
 |-- dev_type: string (nullable = true)
```

However, the other way round, if suppose we have to only change 1 column datatype then we can just change a column datatype instead of creating whole schema.

On Python

>>> **from pyspark.sql.types import ***

>>> **deviceDF=spark.read.json("/user/root/jsons/devices.json")**

>>> **devicecolfixed=deviceDF.withColumn("release_dt",deviceDF.release_dt.cast(TimestampType()))**

>>>**devicecolfixed.printSchema()**

On Scala

scala> **import org.apache.spark.sql.types._**

scala> **val deviceDF=spark.read.json("/user/root/jsons/devices.json")**

scala> **val devicecolfixed=deviceDF.withColumn("release_dt",$"release_dt".cast(TimestampType))**

scala> **devicecolfixed.printSchema**

RENAMING A COLUMN

```
scala>
empDF.withColumnRenamed("ENAME","EMPLOYEE_NAME").printSchema

root
 |-- COMM: long (nullable = true)
 |-- DEPTNO: long (nullable = true)
 |-- EMPNO: long (nullable = true)
 |-- EMPLOYEE_NAME: string (nullable = true)
 |-- HIREDATE: string (nullable = true)
 |-- JOB: string (nullable = true)
 |-- MGRCODE: long (nullable = true)
 |-- SALARY: long (nullable = true)
```

********** *Happy Learning* **********

