Spark DataFrame Creation

ddas.tech/spark-dataframe-creation/

September 26, 2022

Created By: Debasis Das (Sep 2022)

Table of Contents

In this post we will try different ways of creating a Spark DataFrame. Lets start with the import statements

```
import findspark
findspark.init()
import pandas as pd
import numpy as np
import random
import json

import pyspark
from pyspark.sql import SparkSession
from pyspark import SparkConf
from pyspark.sql.types import StructType, StructField, DateType, StringType,
IntegerType

conf = SparkConf().setMaster("local[3]").setAppName("SparkDataFrameCreation")
spark = SparkSession.builder.config(conf=conf).getOrCreate()
```

Sample Data Creation

Lets create some sample data and hold it in a pandas dataframe, we will be writing this data set into csv, json and parquet format and in subsequent code we will load a spark dataframe from the csv, json and parquet files

```
# Lets create some data and hold it in pandas dataframe
countries =
["USA","Mexico","Brazil","Canada","Australia","Japan","China","India","Singapore","Au
stria","Belgium","Finland"]
cars = ["BMW X5","BMW X4","BMW X6","BMW X7","Ford Explorer","Ford Focus","Ford
Expedition","Jeep Wrangler","Jeep Cherokee"]
weeks = []
for i in range(1,14):
    weeks.append(f"Week_{i}")

num_records = 1000
df = pd.DataFrame({"country":np.random.choice(countries,num_records),
                   "car":np.random.choice(cars,num_records),
                   "week":np.random.choice(weeks,num_records),
                   "units_sales":np.random.randint(100,size = num_records),
                   "used_new":np.random.choice(["Used","New"],num_records),
                   "price_per_unit":np.random.randint(low = 20000, high = 55000,size
= num_records)
                  })
print(df.head(10))
```

	country	car	week	units_sales	used_new	price_per_unit
0	USA	Jeep Cherokee	Week_5	93	Used	37819
1	Brazil	Jeep Cherokee	Week_12	63	New	47925
2	Singapore	Ford Expedition	Week_11	91	New	42226
3	Singapore	BMW X5	Week_12	55	New	29405
4	India	Jeep Wrangler	Week_6	44	Used	20281
5	Singapore	BMW X7	Week_2	13	Used	35900
6	Belgium	BMW X7	Week_1	13	New	32914
7	Belgium	Ford Explorer	Week_1	14	Used	32548
8	Finland	Ford Expedition	Week_3	25	New	28895
9	Japan	Jeep Cherokee	Week_6	1	Used	30955

```
# Now Lets write the data into multiple formats
# Write to CSV
df.to_csv("Data/SalesData.csv", index=False)
# Write to JSON
df.to_json("Data/SalesData.json", orient="records")
# Write to Parquet File
df.to_parquet('Data/SalesData.parquet')
```

Creating Spark DataFrames from csv, json and Parquet file formats

Spark DataFrame from CSV

```
sdf1 = spark.read \
        .format("csv") \
        .option("header", "true") \
        .option("mode", "FAILFAST") \
        .option("dateFormat", "M/d/y") \
        .load("data/SalesData.csv")
sdf1.show(5)
sdf1.printSchema()

+-------+---------------+-------+-----------+--------+--------------+
|country|            car|   week|units_sales|used_new|price_per_unit|
+-------+---------------+-------+-----------+--------+--------------+
| Canada|         BMW X4| Week_6|         16|     New|         53040|
|  India|         BMW X6| Week_3|         83|     New|         41132|
|  India|     Ford Focus|Week_11|         65|     New|         43563|
| Mexico|Ford Expedition|Week_12|         12|     New|         47848|
|  Japan|  Ford Explorer| Week_3|         89|     New|         31242|
+-------+---------------+-------+-----------+--------+--------------+
only showing top 5 rows

root
 |-- country: string (nullable = true)
 |-- car: string (nullable = true)
 |-- week: string (nullable = true)
 |-- units_sales: string (nullable = true)
 |-- used_new: string (nullable = true)
 |-- price_per_unit: string (nullable = true)

# As you can see from the print schema that everything was read as string.
# when it would have been great if the unit_sales and price_per_unit would be
inferred as integers.

salesDataSchemaStruct = StructType([
    StructField("country", StringType()),
    StructField("car", StringType()),
    StructField("week", StringType()),
    StructField("units_sales", IntegerType()),
    StructField("used_new", StringType()),
    StructField("price_per_unit", IntegerType())
])

sdf2 = spark.read \
        .format("csv") \
        .option("header", "true") \
        .schema(salesDataSchemaStruct) \
        .option("mode", "FAILFAST") \
        .option("dateFormat", "M/d/y") \
        .load("data/SalesData.csv")
sdf2.show(5)
sdf2.printSchema()
```

```
+-------+---------------+-------+-----------+--------+--------------+
|country|            car|   week|units_sales|used_new|price_per_unit|
+-------+---------------+-------+-----------+--------+--------------+
| Canada|         BMW X4| Week_6|         16|     New|         53040|
|  India|         BMW X6| Week_3|         83|     New|         41132|
|  India|     Ford Focus|Week_11|         65|     New|         43563|
| Mexico|Ford Expedition|Week_12|         12|     New|         47848|
|  Japan|  Ford Explorer| Week_3|         89|     New|         31242|
+-------+---------------+-------+-----------+--------+--------------+
only showing top 5 rows

root
 |-- country: string (nullable = true)
 |-- car: string (nullable = true)
 |-- week: string (nullable = true)
 |-- units_sales: integer (nullable = true)
 |-- used_new: string (nullable = true)
 |-- price_per_unit: integer (nullable = true)
```

Spark DataFrame from JSON

```
salesDataSchemaDDL = """country STRING, car STRING, week STRING, units_sales INT,
used_new STRING,price_per_unit INT"""
sdf3 = spark.read \
        .format("json") \
        .schema(salesDataSchemaDDL) \
        .load("Data/SalesData.json")
sdf3.show()
sdf3.printSchema()

root
 |-- country: string (nullable = true)
 |-- car: string (nullable = true)
 |-- week: string (nullable = true)
 |-- units_sales: integer (nullable = true)
 |-- used_new: string (nullable = true)
 |-- price_per_unit: integer (nullable = true)
```

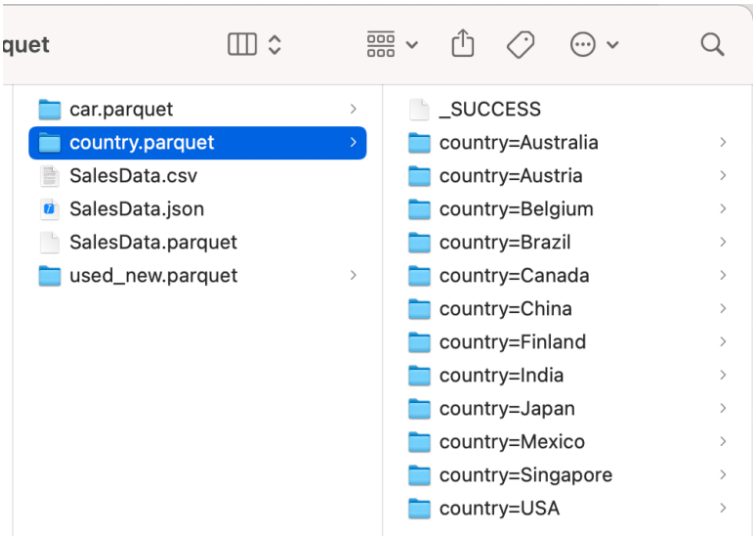
Spark DataFrame from Parquet

```
sdf4 = spark.read \
        .format("parquet") \
        .load("data/SalesData.parquet")
sdf4.show(5)
sdf4.printSchema()
```

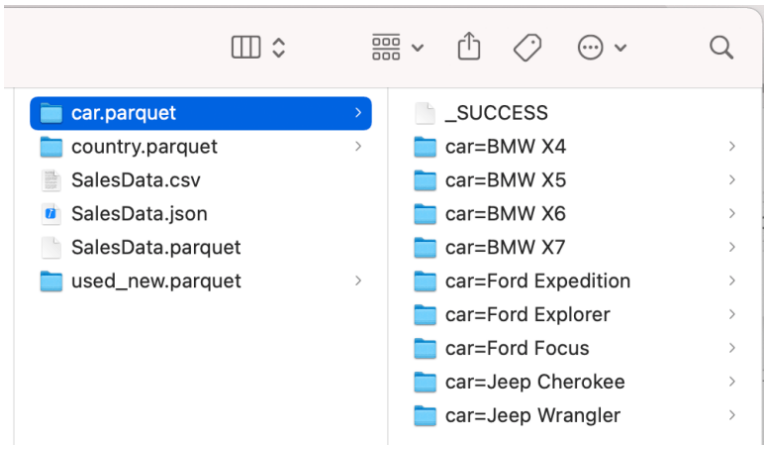
```
root
 |-- country: string (nullable = true)
 |-- car: string (nullable = true)
 |-- week: string (nullable = true)
 |-- units_sales: long (nullable = true)
 |-- used_new: string (nullable = true)
 |-- price_per_unit: long (nullable = true)
```

Partition and Write a Spark DataFrame to multiple parquet files

```
# Partition By Country
sdf4.write.partitionBy("country").parquet("Data/country.parquet")
# Partition By Car
sdf4.write.partitionBy("car").parquet("Data/car.parquet")
# Partition By Used vs New
sdf4.write.partitionBy("used_new").parquet("Data/used_new.parquet")
```



Partition By Country



Partition By Car

Load a Spark DataFrame from Partitioned Dataset

```
# Load a Spark DataFrame from a partioned data set
sdf5 = spark.read \
        .format("parquet") \
        .load("Data/country.parquet")
print(f"rows = {sdf5.count()} cols = {len(sdf5.columns)}")
sdf5.show(10)
sdf5.printSchema()
```

rows = 1000 cols = 6

```
+-------------+-------+-----------+--------+--------------+-------+
|          car|   week|units_sales|used_new|price_per_unit|country|
+-------------+-------+-----------+--------+--------------+-------+
|       BMW X4| Week_6|         16|     New|         53040| Canada|
|       BMW X4| Week_9|         15|     New|         52215| Canada|
|       BMW X7|Week_12|         71|    Used|         36289| Canada|
|       BMW X4| Week_7|         23|     New|         29140| Canada|
|       BMW X7|Week_11|         28|    Used|         33504| Canada|
|Jeep Wrangler| Week_4|         85|    Used|         47202| Canada|
|Ford Explorer| Week_1|          4|    Used|         51148| Canada|
|Jeep Wrangler|Week_10|         65|     New|         50406| Canada|
|       BMW X6| Week_5|         19|     New|         42587| Canada|
|       BMW X7| Week_9|         91|     New|         36257| Canada|
+-------------+-------+-----------+--------+--------------+-------+
only showing top 10 rows

root
 |-- car: string (nullable = true)
 |-- week: string (nullable = true)
 |-- units_sales: long (nullable = true)
 |-- used_new: string (nullable = true)
 |-- price_per_unit: long (nullable = true)
 |-- country: string (nullable = true)
```

Creating a Spark DataFrame from an Ordered List

```
employee_list = [("John", "Doe", 18, 2000),
                 ("Jane", "Doe", 20, 3000),
                 ("Mary", "Jane", 22, 4000),
                 ("Bill", "Smith", 19, 1000)
               ]
employee_df = spark.createDataFrame(employee_list).toDF("first_name", "last_name",
"age", "salary")
employee_df.show()
employee_df.printSchema()
```