**Working on Parquet files in Spark**

Parquet is an open source file format for Hadoop/Spark and other Big data frameworks. Parquet stores nested data structures in a flat columnar format. Compared to any traditional approach where the data is stored in a row-oriented format, Parquet is more efficient in the terms of performance and storage.

Parquet can be used in any Hadoop ecosystems such as Spark, [Hive](https://acadgild.com/big-data/big-data-hadoop-administration-training-certification), Impala, and [Pig](https://acadgild.com/big-data/big-data-hadoop-administration-training-certification).

Parquet stores the binary data in a column-oriented way, where the values of each and every column are organized so that all the columns are adjacent, enabling better compression rate. It is especially good for the queries which read columns from a “wide” (with many columns) table since only needed columns are read and the IO(Input/Output) is minimized.

**When the data is processed with big data frameworks such as hadoop or spark, the cost of storing the data is more if the data is to be stored in HDFS** as HDFS has replication factor, **minimum 3 copies of each file will be maintained for fault tolerance**. So automatically the storage cost will increase, along with storage, processing cost will also increase as the data comes into CPU, Network IO etc., So  to **minimize all these costs, parquet is one of the choice for developers which efficiently stores the data and thereby increasing the performance.**

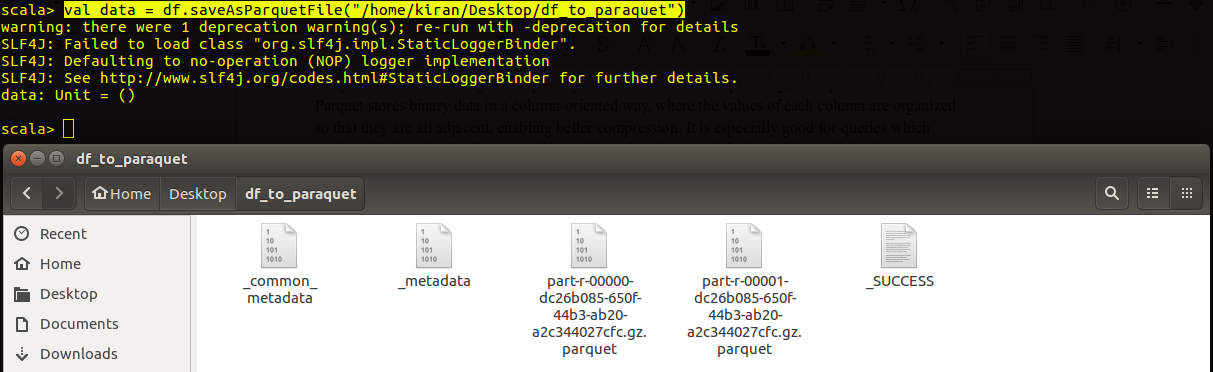
To work on Parquet files, we do not need to download any external jar files.

**Spark by default has provided support for Parquet files.**

We will now first convert the above data frame into a Parquet file. It is very simple in Spark. Just save the file as a Parquet file.

val data = df.saveAsParquetFile("/home/kiran/Desktop/df\_to\_paraquet")

Now in the specified path, you can see the files with .paraquet extension. The same is as shown in the below screen shot.

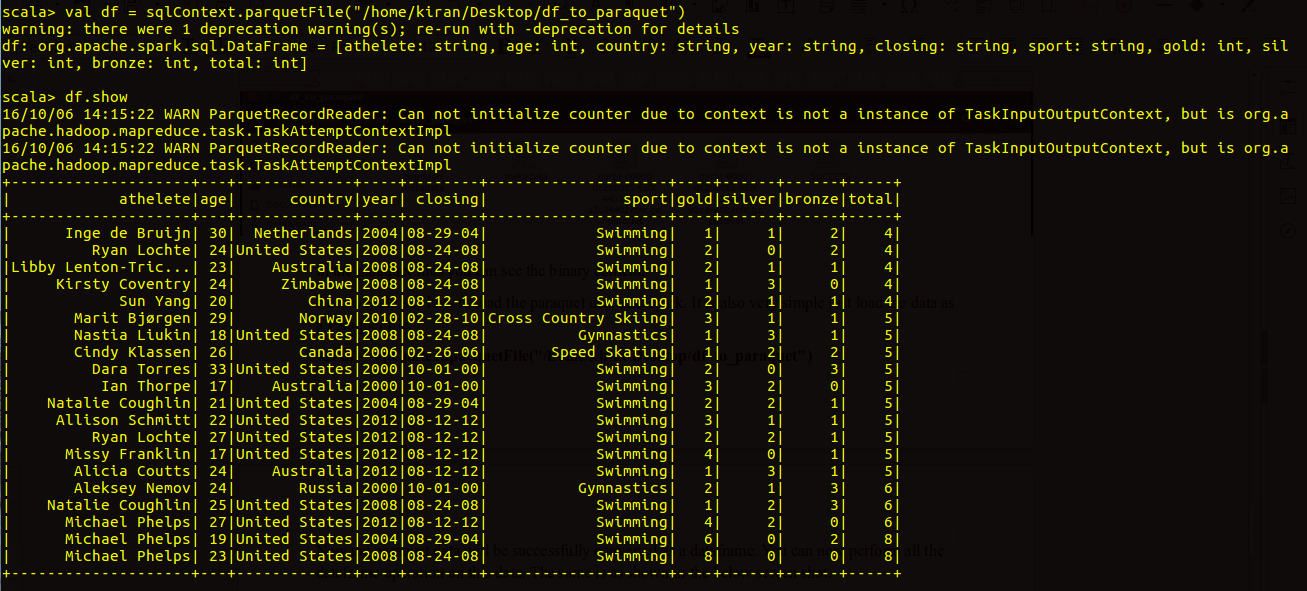


In the parquet files, you can see the binary content.

**Now we will see how to load the Parquet data into Spark. It is quite simple. Just load the data as a Parquet file.**

val df = sqlContext.parquetFile("/home/kiran/Desktop/df\_to\_paraquet")

Now the Parquet data will be successfully converted as a data frame. You can now perform all the data frame operation on this data. The same is as shown in the below screen shot.



#### **Advantages of using Parquet(** columnar formats.)

* Organizing by column allows for better compression, as data is more homogeneous. The **space savings** are very noticeable at the scale of a Hadoop cluster.
* **I/O will be reduced** as we can efficiently scan only a subset of the columns while reading the data. Better compression also **reduces the bandwidth required to read the input**.
* As we store data of the same type in each column, we can use **encoding better suited to the modern processors’ pipeline** by making instruction branching more predictable.

Reference :

[https://acadgild.com/blog/parquet-file-format-hadoop](https://acadgild.com/blog/parquet-file-format-hadoop\)

<https://acadgild.com/blog/avro-and-parquet-files-in-spark>