Establecer el schema de nuestro csv (CASO DE USO CUANDO DESCONOCEMOS LAS COLUMNAS)

```
def main(args: Array[String]): Unit = {

    val spark: SparkSession = SparkSession.builder()
      .master("local[1]")
      .appName("json parquet")
      .getOrCreate()

    /*val  df=spark.read.format("csv")
      .option("delimiter",",")
      .load ("aerolineas.csv")
    df.show()*/
    // ffijar nosotros el schema para poder añadir los campos
    val schema = new StructType().add("ID",IntegerType,true)
      .add("descripcion",StringType,true)
    val  df=spark.read.format("csv").schema(schema)
      .option("delimiter",",")
      .load ("aerolineas.csv")
    df.show()


  }
}
```

Usar una clase para guardar los datos de un fichero

```
import org.apache.spark.sql.SparkSession
object schemacsv {
    case class Persona(nombre:String, edad:Long)
    def main(args: Array[String]): Unit = {
      val spark = SparkSession.builder().appName("MiApp").master("local[*]").getOrCreate()
      import spark.implicits._
      val cdatos = spark.sparkContext.textFile("Persona.txt")
        .map(_.split(",")).map(atributos => Persona(atributos(0),
        atributos(1).trim.toInt))
      cdatos.collect().foreach(println)
    }
  }
```

Ejercicio2. Leer el contenido de un fichero JSON

```
import org.apache.spark.SparkConf
import org.apache.spark.SparkContext

import org.apache.spark.sql.Dataset
import org.apache.spark.sql.DataFrame
import org.apache.spark.sql.SparkSession

object ejemplo {
  def main(args: Array[String]): Unit = {
    val spark = SparkSession.builder().appName("MiApp").master("local[*]").getOrCreate()
    val dataframe = spark.read.option("multiline", true).json("datoSsjson")
    //val dataframe= spark.read.json(spark.sparkContext.wholeTextFiles("datos.json").values)

    val res = dataframe.collect().foreach(println)
  }
}
```

EJERCICIO2. LISTAR TODO EL ARBOL, MOSTRAR EL ESQUEMA Y MOSTRAR UN CAMPO CONCRETO DEL JSON QUE SE OS FACILITA

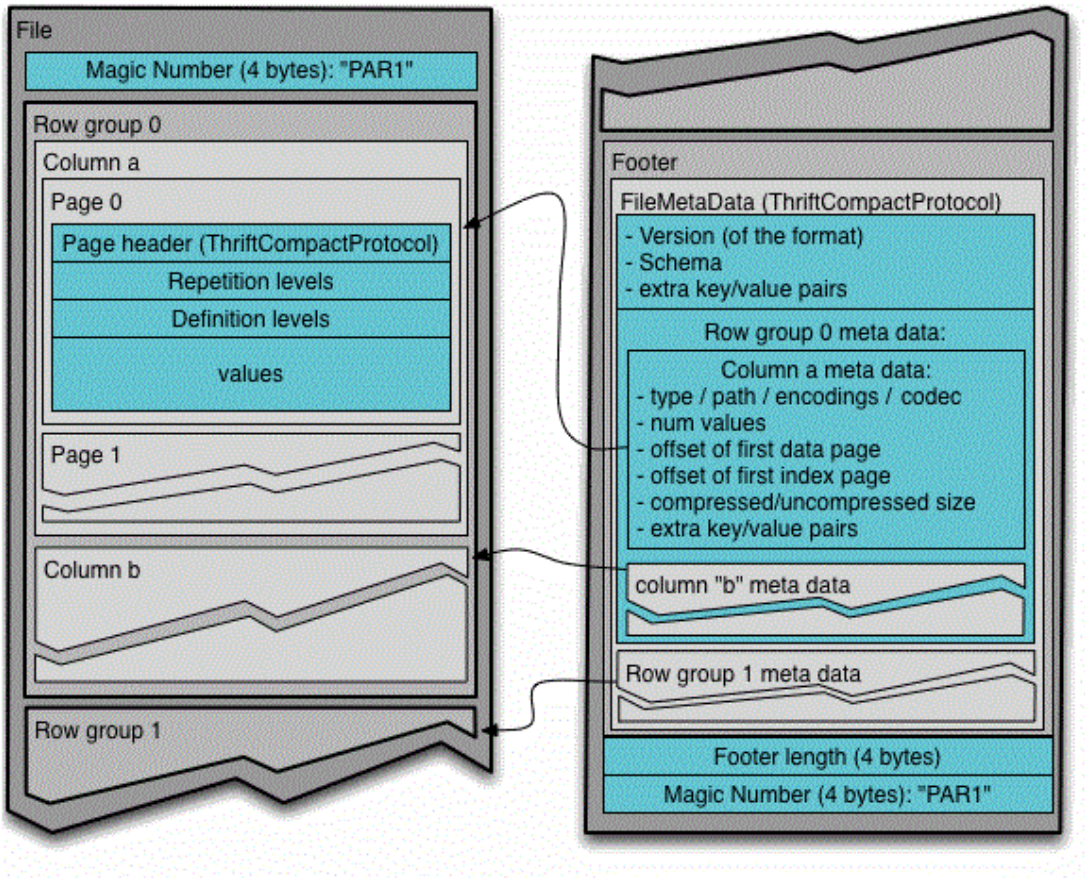
```
import org.apache.spark.sql.SparkSession

object ejemplo {
  def main(args: Array[String]): Unit = {
    val spark = SparkSession.builder().appName("MiApp").master("local[*]").getOrCreate()
    //val dataframe = spark.read.json("datosjson")
    val dataframe= spark.read.json(spark.sparkContext.wholeTextFiles("datosjson").values)

  dataframe.show()
dataframe.printSchema()
dataframe.select("type").show()
  } }
```

Formato: Parquet

El formato Parquet es un formato open-source de almacenamiento en columnas enfocado en la mejora del procesamiento **de** datos, modelado **de** datos y programación



El formato de Parquet está compuesto por tres piezas:

- **Row group**: es un conjunto de filas en formato columnar, con un tamaño entre 50Mb a 1Gb.
- **Column chunk**: son los datos de una columna en un grupo. Se puede leer de manera independiente para mejorar las lecturas.
- **Page**: Es donde finalmente se almacenan los datos debe ser lo suficiente grande para que la compresión sea eficiente.

Rendimiento de la consulta

Dependiendo del caso de uso de su negocio, Apache Parquet es una buena opción **si tiene que proporcionar funciones de búsqueda parcial, es decir, no consultar todas las columnas,** y no le preocupa el tiempo de escritura del archivo.

Ejercicio 3. EJEMPLO PARQUET CON UNA SEQ DE DATOS

```
import org.apache.spark.sql.SparkSession

import org.apache.spark.sql.types.{BooleanType, DoubleType, IntegerType, StringType,
StructType}



object schemacsv {

  def main(args: Array[String]): Unit = {

    val spark = SparkSession.builder().appName("MiApp").master("local[*]").getOrCreate()



    val data = Seq(("James ","","Smith","36636","M",3000),

      ("Michael ","Rose","","40288","M",4000),

      ("Robert ","","Williams","42114","M",4000),

      ("Maria ","Anne","Jones","39192","F",4000),

      ("Jen","Mary","Brown","","F",-1)

    )



    val columns = Seq("firstname","middlename","lastname","dob","gender","salary")

    import spark.sqlContext.implicits._
```

```
val df = data.toDF(columns:_*)

// ESCRIBIMOS

df.write.parquet("/tmp/output/people1.parquet")

// LEEMOS

val parqDF = spark.read.parquet("/tmp/output/people1.parquet")

// CONSUTA SOBRE PARQUET

print("Pintando conuslta ")

parqDF.createOrReplaceTempView("ParquetTable")

val parkSQL = spark.sql("select * from ParquetTable where salary >= 4000 ").show()
```

EJERCICIO 4. PASAR DE CSV A PARQUET Y A JSON

```
object schemacsv {

  def main(args: Array[String]): Unit = {

    val spark: SparkSession = SparkSession.builder()
      .master("local[1]")
      .appName("CONVERTIR ")
      .getOrCreate()

    spark.sparkContext.setLogLevel("ERROR")

    //read csv with options
    val df = spark.read.options(Map("inferSchema"->"true","delimiter"->",","header"->"true"))
      .csv("1800.csv")
    /*df.show()
```

```
   df.printSchema()
   //convert to parquet
   df.write.mode(SaveMode.Overwrite).parquet("/tmp/parquet/1800.parquet")
   print("parquet")
   val parqDF = spark.read.parquet("/tmp/parquet/1800.parquet").show()*/
   //convert to JSON
   df.write.mode(SaveMode.Overwrite).json("/tmp/json/zipcodes.json")
   val parqDF = spark.read.json("/tmp/json/1800.json").show()
  }}
```

DESDE EL TERMINAL VEREMOS LSO DATOS BIEN COMO JSON

EJERCICIO 5 DE JSON A PARQUET

```
object schemacsv {


  def main(args: Array[String]): Unit = {

    val spark: SparkSession = SparkSession.builder()
      .master("local[1]")
      .appName("json parquet")
      .getOrCreate()

    spark.sparkContext.setLogLevel("ERROR")

    import spark.implicits._

    val peopleDF = spark.read.json("datojson")

    // DataFrames can be saved as Parquet files, maintaining the schema information
    peopleDF.write.parquet("people1.parquet")

    // Read in the parquet file created above
    // Parquet files are self-describing so the schema is preserved
    // The result of loading a Parquet file is also a DataFrame
    val parquetFileDF = spark.read.parquet("people1.parquet")

    // Parquet files can also be used to create a temporary view and then used in SQL statements
    parquetFileDF.createOrReplaceTempView("parquetFile")
    val namesDF = spark.sql("SELECT nombre FROM parquetFile WHERE edad BETWEEN 10 AND
50")
// DOS MANERAS DE MOSTRAR LA INFO
    namesDF.show()
    // lanzarlos ala vez sino el parqiete cambia y debemos deponer otro
    namesDF.map(attributes =>"nombre: " + attributes(0)).show()

  }
}
```