File Type	Packages	Read	Write
JSON		val git = sqlContext.read.json(path)	git.write.json("/user/pratap/git_parquet/")
Parquet		val git = sqlContext.read.parquet(path)	git.write.parquet("/user/pratap/git_parquet/")
Avro	import com.databricks.spark.avro._	val df = sqlContext.read.avro(path)	df.write.format("com.databricks.spark.avro").save("/user/pratap/tmp/avro")
XML	spark-shell --packages com.databricks:spark-xml_2.10:0.3.5	val df = sqlContext.read.format("com.databricks.spark.xml").option("rowTag", "Transaction").load("/user/pratap/samplexml.xml")	df.write.format("com.databricks.spark.xml").option("rowTag", "Transaction") .save("/user/pratap/newtrans.xml")
CSV	spark-shell --packages com.databricks:spark-csv_2.10:1.5.0 spark-shell --packages com.databricks:spark-csv_2.10:1.5.0 --jars .ivy2/jars/hadoop-lzo-0.4.20-SNAPSHOT.jar import com.hadoop.compression.lzo.LzoCodec	val git = sqlContext.read.format("csv").option("header", "true").option("inferSchema", "true").load(path) val git = sqlContext.read.format("com.databricks.spark.csv").option("header", "true").option("codec","org.apache.hadoop.io.compress.DefaultCodec").load("/user/pratap/tmp/csv/compressed/part-00000.deflate")	**write csv:** git.repartition(1).write.format("com.databricks.spark.csv").option("header", "true").save("/user/pratap/tmp/csv") **write compressed** git.repartition(1).write.format("com.databricks.spark.csv").option("header", "true").option("codec","com.hadoop.compression.lzo.LzoCodec").save("/user/pratap/tmp/csv/compressed10") git.repartition(1).write.format("com.databricks.spark.csv").option("header", "true").option("codec","org.apache.hadoop.io.compress.DefaultCodec").save("/user/pratap/tmp/csv/compressed") GzipCodec, DefaultCodec, Lz4Codec,SnappyCodec, BZip2Codec
ORC	import org.apache.spark.sql.hive.orc._ import org.apache.spark.sql._	val hiveContext = new org.apache.spark.sql.hive.HiveContext(sc) val git = hiveContext.read.orc(path)	hiveContext.createDataFrame(git.rdd,git.schema).write.orc("hdfs://ybolcldrmstr.yotabites.com:8020/user/pratap/orc")
Seq.	import org.apache.hadoop.io.Text import org.apache.hadoop.io.IntWritable	val result = sc.sequenceFile("/user/pratap/tmp/seq/part-00000", classOf[Text], classOf[IntWritable])	data.coalesce(1).saveAsSequenceFile("/user/pratap/tmp/seq")