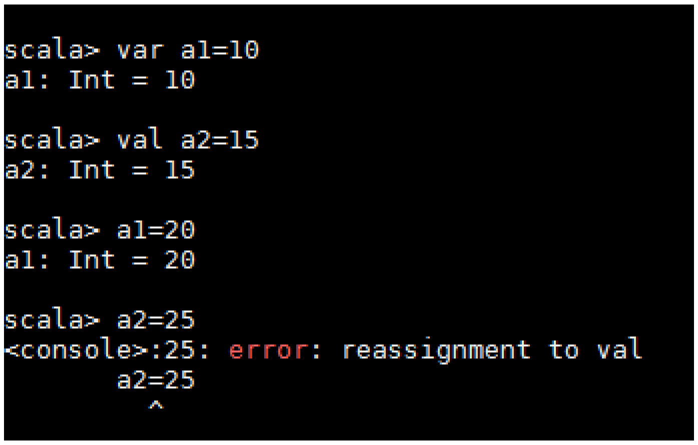
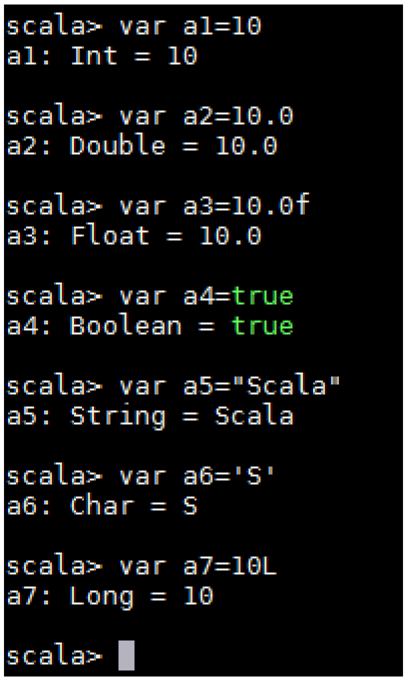
1) Creating Variables/Constants in Scala



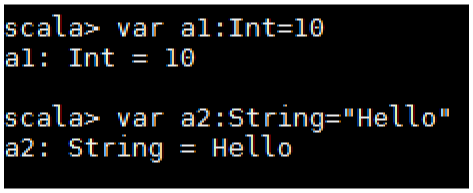
```
scala> var a1=10
a1: Int = 10

scala> val a2=15
a2: Int = 15

scala> a1=20
a1: Int = 20

scala> a2=25
<console>:25: error: reassignment to val
       a2=25
         ^
```

2) Different types of variables in Scala. This shows that Scala has the ability to infer the datatypes.



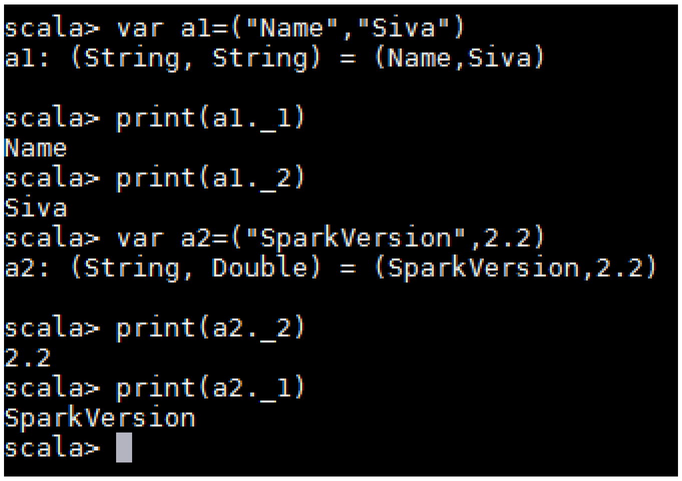
```
scala> var a1=10
a1: Int = 10

scala> var a2=10.0
a2: Double = 10.0

scala> var a3=10.0f
a3: Float = 10.0

scala> var a4=true
a4: Boolean = true

scala> var a5="Scala"
a5: String = Scala

scala> var a6='S'
a6: Char = S

scala> var a7=10L
a7: Long = 10

scala>
```

3) Create variables with explicit datatype.



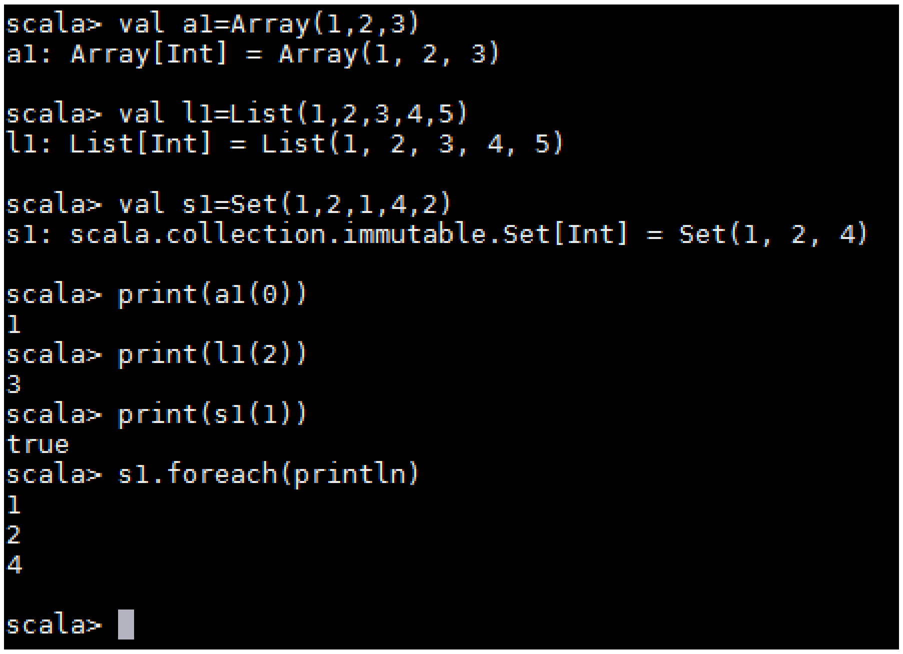
```
scala> var a1:Int=10
a1: Int = 10

scala> var a2:String="Hello"
a2: String = Hello
```

4) Create tuple/KeyValue and how to access the key or value.



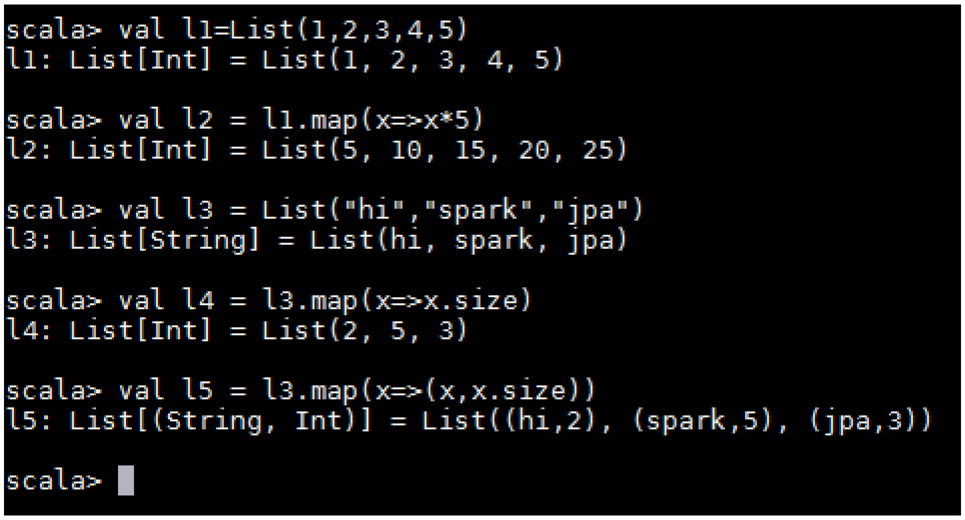
```
scala> var a1=("Name","Siva")
a1: (String, String) = (Name,Siva)

scala> print(a1._1)
Name
scala> print(a1._2)
Siva
scala> var a2=("SparkVersion",2.2)
a2: (String, Double) = (SparkVersion,2.2)

scala> print(a2._2)
2.2
scala> print(a2._1)
SparkVersion
scala>
```

5) Creating the collections Array/List/Set and accessing the elements in Scala



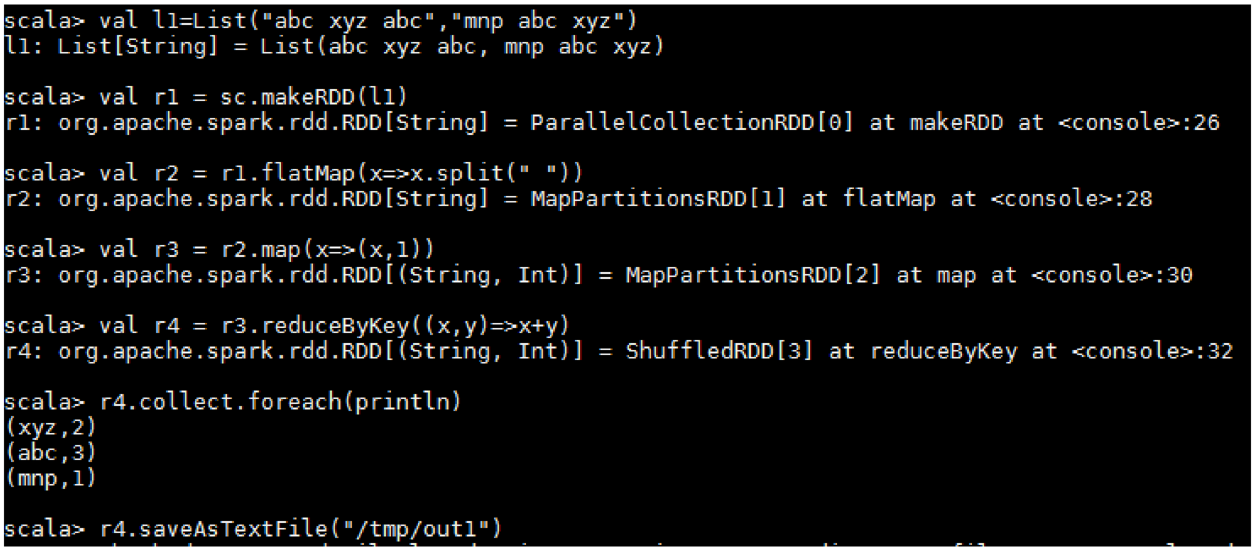
```
scala> val a1=Array(1,2,3)
a1: Array[Int] = Array(1, 2, 3)

scala> val l1=List(1,2,3,4,5)
l1: List[Int] = List(1, 2, 3, 4, 5)

scala> val s1=Set(1,2,1,4,2)
s1: scala.collection.immutable.Set[Int] = Set(1, 2, 4)

scala> print(a1(0))
1
scala> print(l1(2))
3
scala> print(s1(1))
true
scala> s1.foreach(println)
1
2
4

scala>
```

6) Applying map method on collections



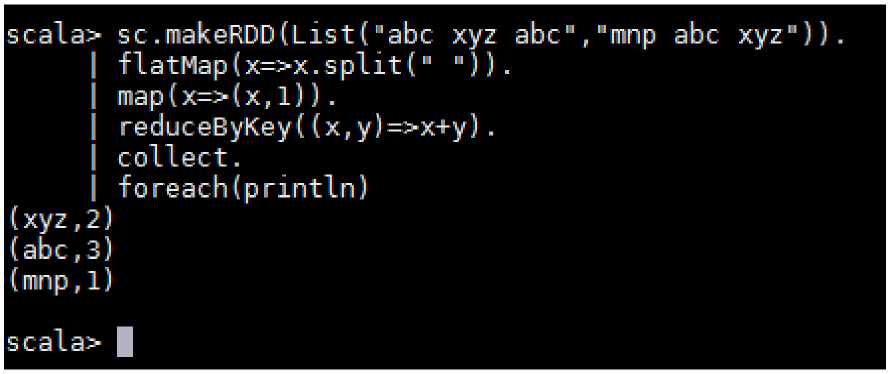
```
scala> val l1=List(1,2,3,4,5)
l1: List[Int] = List(1, 2, 3, 4, 5)

scala> val l2 = l1.map(x=>x*5)
l2: List[Int] = List(5, 10, 15, 20, 25)

scala> val l3 = List("hi","spark","jpa")
l3: List[String] = List(hi, spark, jpa)

scala> val l4 = l3.map(x=>x.size)
l4: List[Int] = List(2, 5, 3)

scala> val l5 = l3.map(x=>(x,x.size))
l5: List[(String, Int)] = List((hi,2), (spark,5), (jpa,3))

scala>
```

7) Wordcount Example on Spark



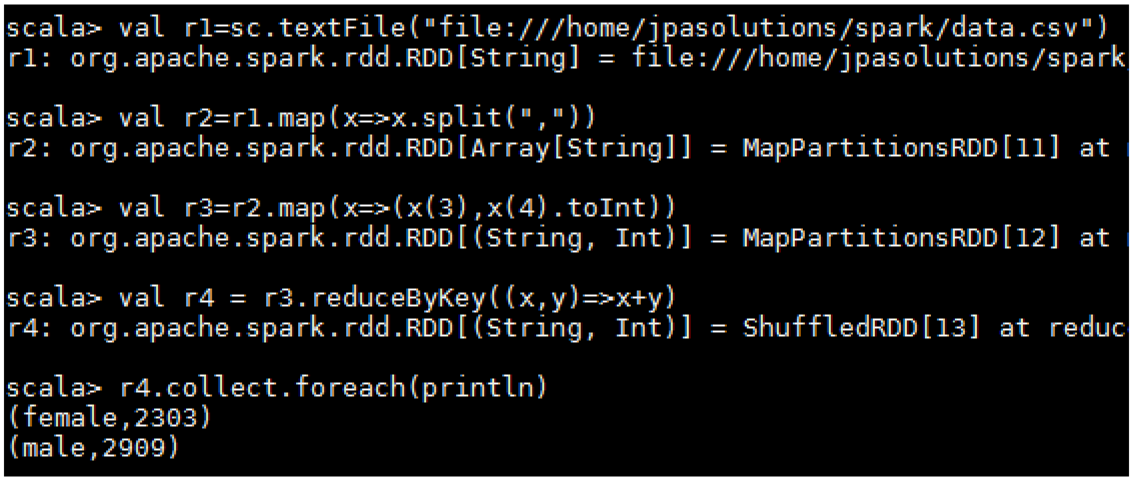
```
scala> val l1=List("abc xyz abc","mnp abc xyz")
l1: List[String] = List(abc xyz abc, mnp abc xyz)

scala> val r1 = sc.makeRDD(l1)
r1: org.apache.spark.rdd.RDD[String] = ParallelCollectionRDD[0] at makeRDD at <console>:26

scala> val r2 = r1.flatMap(x=>x.split(" "))
r2: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[1] at flatMap at <console>:28

scala> val r3 = r2.map(x=>(x,1))
r3: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[2] at map at <console>:30

scala> val r4 = r3.reduceByKey((x,y)=>x+y)
r4: org.apache.spark.rdd.RDD[(String, Int)] = ShuffledRDD[3] at reduceByKey at <console>:32

scala> r4.collect.foreach(println)
(xyz,2)
(abc,3)
(mnp,1)

scala> r4.saveAsTextFile("/tmp/out1")
```

8) Wordcount example on spark in single line

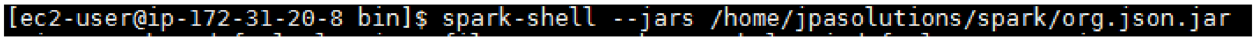


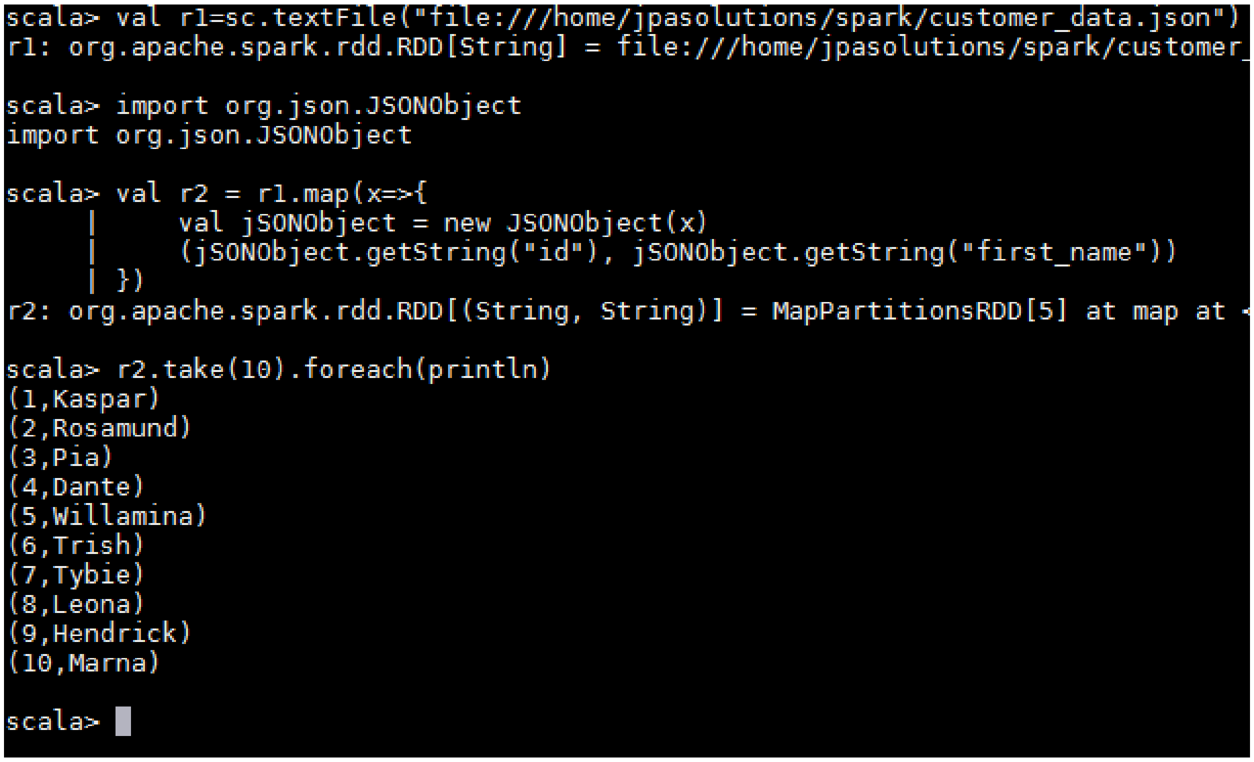
```
scala> sc.makeRDD(List("abc xyz abc","mnp abc xyz")).
     | flatMap(x=>x.split(" ")).
     | map(x=>(x,1)).
     | reduceByKey((x,y)=>x+y).
     | collect.
     | foreach(println)
(xyz,2)
(abc,3)
(mnp,1)

scala>
```

9) Get the sum by gender on the data.csv file

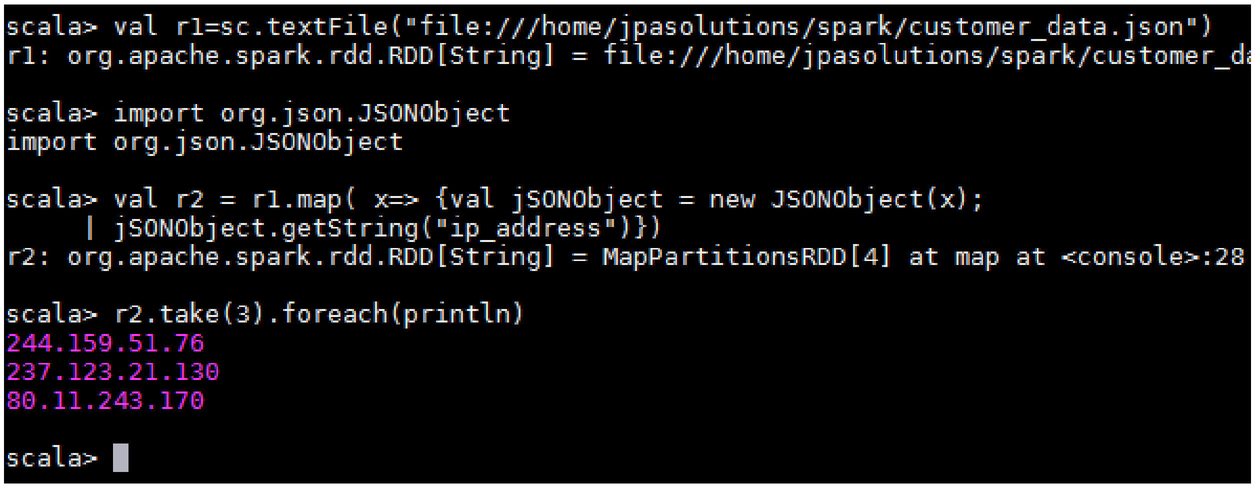

```
scala> val r1=sc.textFile("file:///home/jpasolutions/spark/data.csv")
r1: org.apache.spark.rdd.RDD[String] = file:///home/jpasolutions/spark

scala> val r2=r1.map(x=>x.split(","))
r2: org.apache.spark.rdd.RDD[Array[String]] = MapPartitionsRDD[11] at

scala> val r3=r2.map(x=>(x(3),x(4).toInt))
r3: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[12] at

scala> val r4 = r3.reduceByKey((x,y)=>x+y)
r4: org.apache.spark.rdd.RDD[(String, Int)] = ShuffledRDD[13] at reduc

scala> r4.collect.foreach(println)
(female,2303)
(male,2909)
```

10) Using customer_data.json file on spark


```
[ec2-user@ip-172-31-20-8 bin]$ spark-shell --jars /home/jpasolutions/spark/org.json.jar
```

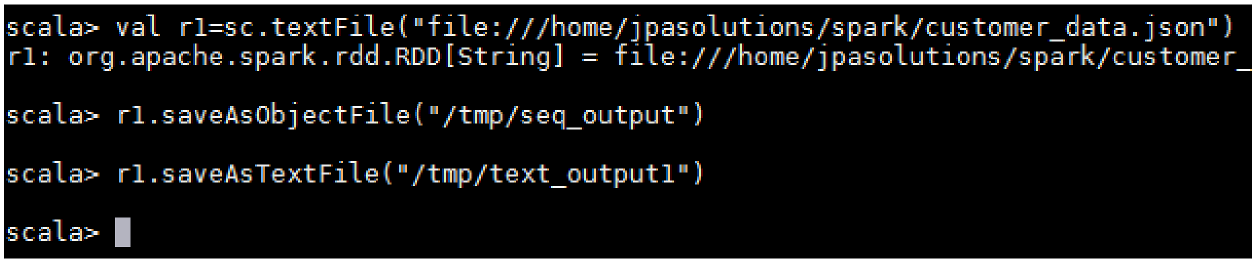

```
scala> val r1=sc.textFile("file:///home/jpasolutions/spark/customer_data.json")
r1: org.apache.spark.rdd.RDD[String] = file:///home/jpasolutions/spark/customer_

scala> import org.json.JSONObject
import org.json.JSONObject

scala> val r2 = r1.map(x=>{
     |     val jSONObject = new JSONObject(x)
     |     (jSONObject.getString("id"), jSONObject.getString("first_name"))
     | })
r2: org.apache.spark.rdd.RDD[(String, String)] = MapPartitionsRDD[5] at map at

scala> r2.take(10).foreach(println)
(1,Kaspar)
(2,Rosamund)
(3,Pia)
(4,Dante)
(5,Willamina)
(6,Trish)
(7,Tybie)
(8,Leona)
(9,Hendrick)
(10,Marna)

scala> 
```

11) Another example of reading the json data



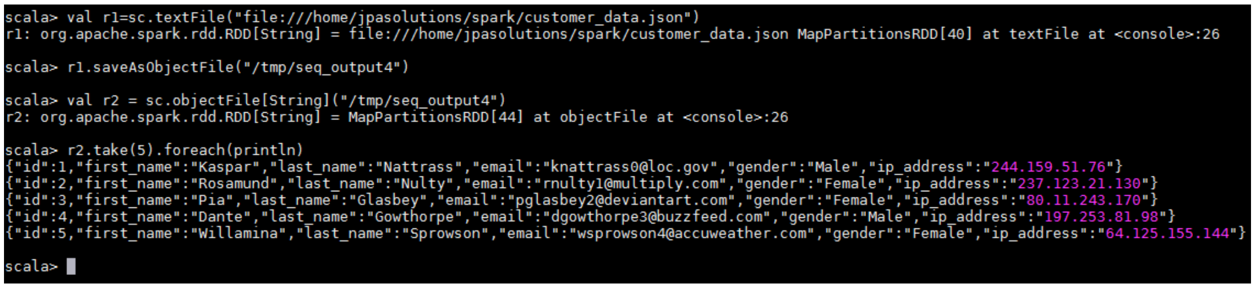
```
scala> val r1=sc.textFile("file:///home/jpasolutions/spark/customer_data.json")
r1: org.apache.spark.rdd.RDD[String] = file:///home/jpasolutions/spark/customer_d

scala> import org.json.JSONObject
import org.json.JSONObject

scala> val r2 = r1.map( x=> {val jSONObject = new JSONObject(x);
     | jSONObject.getString("ip_address")})
r2: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[4] at map at <console>:28

scala> r2.take(3).foreach(println)
244.159.51.76
237.123.21.130
80.11.243.170

scala> 
```

12) Writing the output as text file and sequence file



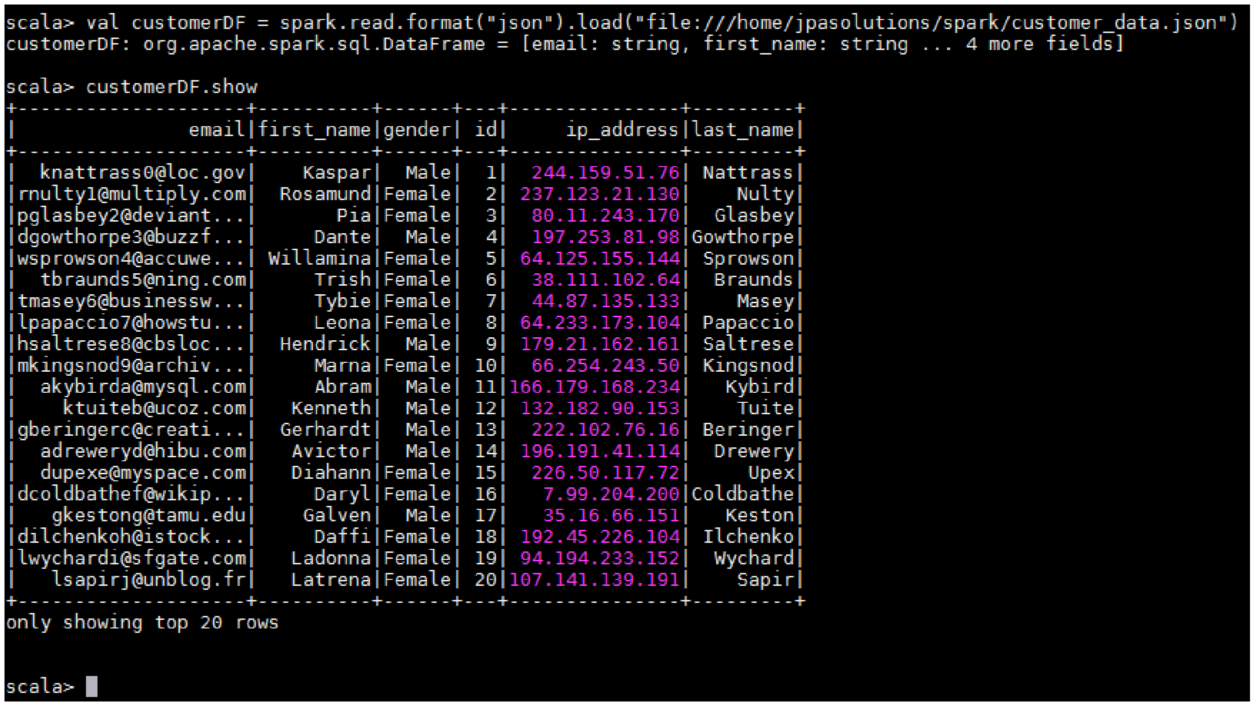
```
scala> val r1=sc.textFile("file:///home/jpasolutions/spark/customer_data.json")
r1: org.apache.spark.rdd.RDD[String] = file:///home/jpasolutions/spark/customer_

scala> r1.saveAsObjectFile("/tmp/seq_output")

scala> r1.saveAsTextFile("/tmp/text_output1")

scala> 
```

13) Reading the sequence file



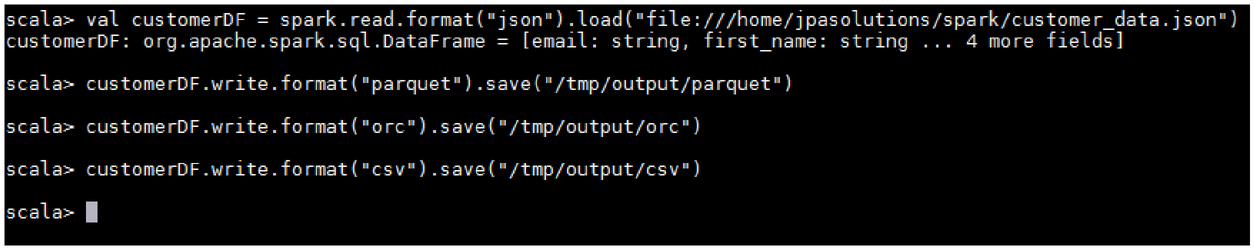
```
scala> val r1=sc.textFile("file:///home/jpasolutions/spark/customer_data.json")
r1: org.apache.spark.rdd.RDD[String] = file:///home/jpasolutions/spark/customer_data.json MapPartitionsRDD[40] at textFile at <console>:26

scala> r1.saveAsObjectFile("/tmp/seq_output4")

scala> val r2 = sc.objectFile[String]("/tmp/seq_output4")
r2: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[44] at objectFile at <console>:26

scala> r2.take(5).foreach(println)
{"id":1,"first_name":"Kaspar","last_name":"Nattrass","email":"knattrass0@loc.gov","gender":"Male","ip_address":"244.159.51.76"}
{"id":2,"first_name":"Rosamund","last_name":"Nulty","email":"rnulty1@multiply.com","gender":"Female","ip_address":"237.123.21.130"}
{"id":3,"first_name":"Pia","last_name":"Glasbey","email":"pglasbey2@deviantart.com","gender":"Female","ip_address":"80.11.243.170"}
{"id":4,"first_name":"Dante","last_name":"Gowthorpe","email":"dgowthorpe3@buzzfeed.com","gender":"Male","ip_address":"197.253.81.98"}
{"id":5,"first_name":"Willamina","last_name":"Sprowson","email":"wsprowson4@accuweather.com","gender":"Female","ip_address":"64.125.155.144"}

scala> 
```

14) Reading the customer data on Spark SQL.



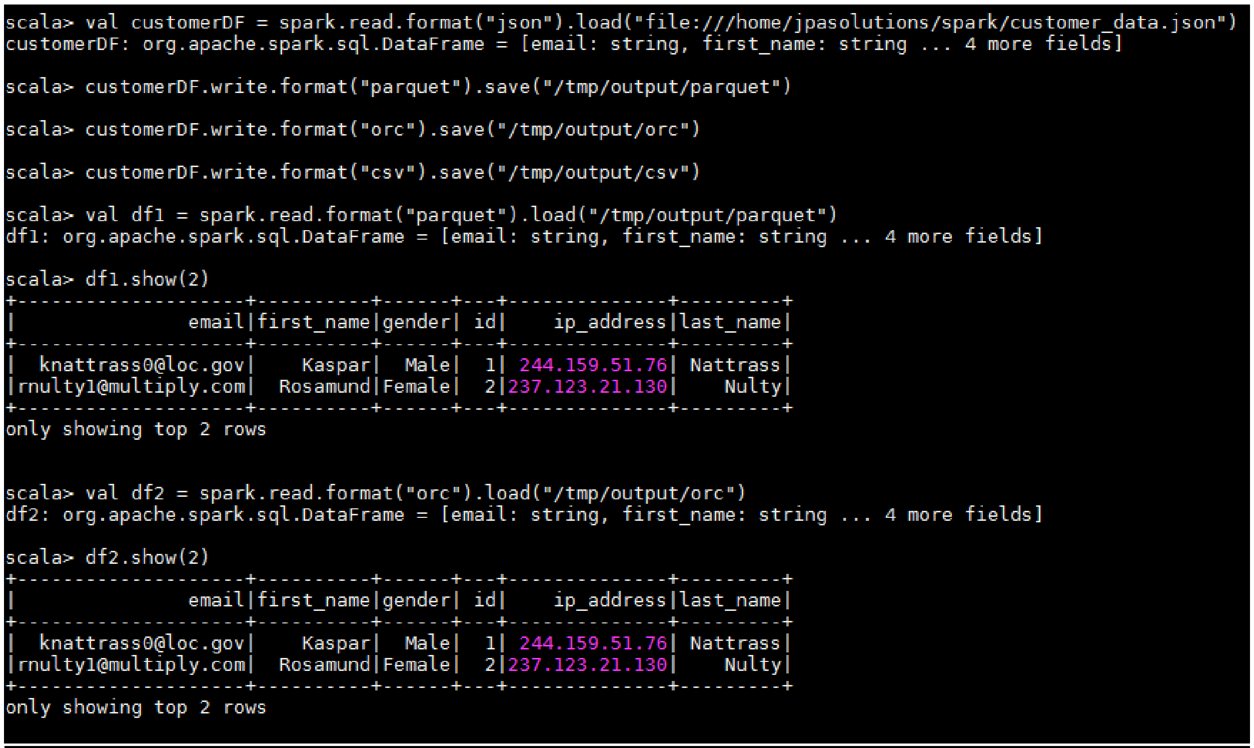
```
scala> val customerDF = spark.read.format("json").load("file:///home/jpasolutions/spark/customer_data.json")
customerDF: org.apache.spark.sql.DataFrame = [email: string, first_name: string ... 4 more fields]

scala> customerDF.show
+--------------------+----------+------+---+---------------+---------+
|               email|first_name|gender| id|     ip_address|last_name|
+--------------------+----------+------+---+---------------+---------+
|  knattrass0@loc.gov|    Kaspar|  Male|  1|  244.159.51.76| Nattrass|
|rnulty1@multiply.com|  Rosamund|Female|  2| 237.123.21.130|    Nulty|
|pglasbey2@deviant...|       Pia|Female|  3|  80.11.243.170|  Glasbey|
|dgowthorpe3@buzzf...|     Dante|  Male|  4|  197.253.81.98|Gowthorpe|
|wsprowson4@accuwe...| Willamina|Female|  5| 64.125.155.144| Sprowson|
|  tbraunds5@ning.com|     Trish|Female|  6|  38.111.102.64|  Braunds|
|tmasey6@businessw...|     Tybie|Female|  7|  44.87.135.133|    Masey|
|lpapaccio7@howstu...|     Leona|Female|  8| 64.233.173.104| Papaccio|
|hsaltrese8@cbsloc...|  Hendrick|  Male|  9| 179.21.162.161| Saltrese|
|mkingsnod9@archiv...|     Marna|Female| 10|  66.254.243.50| Kingsnod|
|  akybirda@mysql.com|     Abram|  Male| 11|166.179.168.234|   Kybird|
|    ktuiteb@ucoz.com|   Kenneth|  Male| 12| 132.182.90.153|    Tuite|
|gberingerc@creati...|  Gerhardt|  Male| 13|  222.102.76.16| Beringer|
|  adreweryd@hibu.com|   Avictor|  Male| 14| 196.191.41.114|  Drewery|
|  dupexe@myspace.com|   Diahann|Female| 15|  226.50.117.72|     Upex|
|dcoldbathef@wikip...|     Daryl|Female| 16|   7.99.204.200|Coldbathe|
|   gkestong@tamu.edu|    Galven|  Male| 17|   35.16.66.151|   Keston|
|dilchenkoh@istock...|     Daffi|Female| 18| 192.45.226.104| Ilchenko|
|lwychardi@sfgate.com|   Ladonna|Female| 19| 94.194.233.152|  Wychard|
|   lsapirj@unblog.fr|   Latrena|Female| 20|107.141.139.191|    Sapir|
+--------------------+----------+------+---+---------------+---------+
only showing top 20 rows


scala> 
```

15) Writing the json data into csv/parquet/orc format

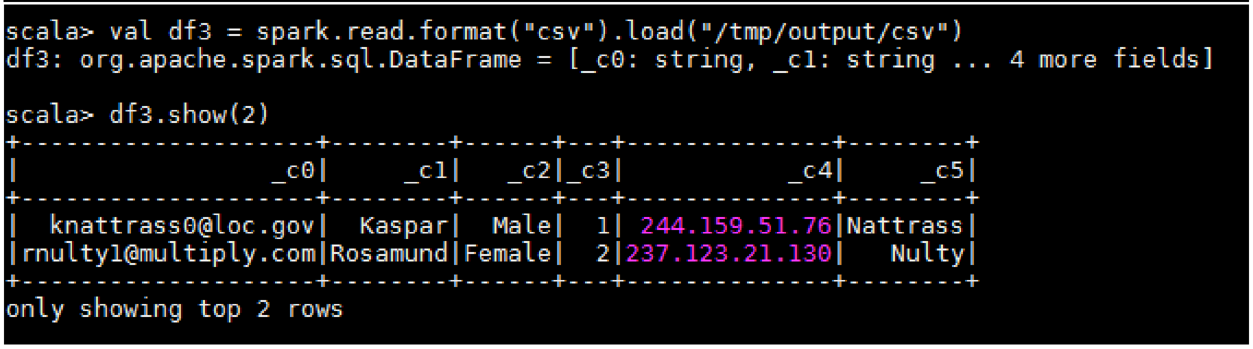


```
scala> val customerDF = spark.read.format("json").load("file:///home/jpasolutions/spark/customer_data.json")
customerDF: org.apache.spark.sql.DataFrame = [email: string, first_name: string ... 4 more fields]

scala> customerDF.write.format("parquet").save("/tmp/output/parquet")

scala> customerDF.write.format("orc").save("/tmp/output/orc")

scala> customerDF.write.format("csv").save("/tmp/output/csv")

scala> 
```

16) Reading the json/parquet/csv data

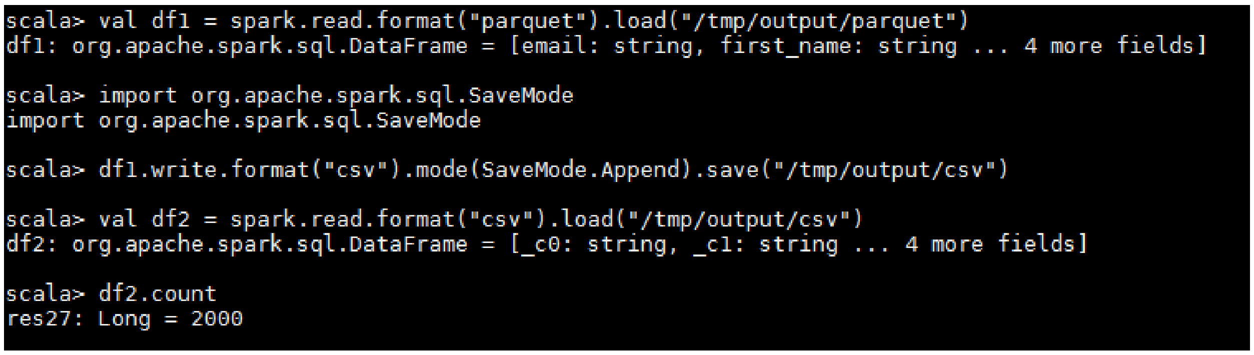


```
scala> val customerDF = spark.read.format("json").load("file:///home/jpasolutions/spark/customer_data.json")
customerDF: org.apache.spark.sql.DataFrame = [email: string, first_name: string ... 4 more fields]

scala> customerDF.write.format("parquet").save("/tmp/output/parquet")

scala> customerDF.write.format("orc").save("/tmp/output/orc")

scala> customerDF.write.format("csv").save("/tmp/output/csv")

scala> val df1 = spark.read.format("parquet").load("/tmp/output/parquet")
df1: org.apache.spark.sql.DataFrame = [email: string, first_name: string ... 4 more fields]

scala> df1.show(2)
+--------------------+----------+------+---+--------------+---------+
|               email|first_name|gender| id|    ip_address|last_name|
+--------------------+----------+------+---+--------------+---------+
|  knattrass0@loc.gov|    Kaspar|  Male|  1| 244.159.51.76| Nattrass|
|rnulty1@multiply.com|  Rosamund|Female|  2|237.123.21.130|    Nulty|
+--------------------+----------+------+---+--------------+---------+
only showing top 2 rows


scala> val df2 = spark.read.format("orc").load("/tmp/output/orc")
df2: org.apache.spark.sql.DataFrame = [email: string, first_name: string ... 4 more fields]

scala> df2.show(2)
+--------------------+----------+------+---+--------------+---------+
|               email|first_name|gender| id|    ip_address|last_name|
+--------------------+----------+------+---+--------------+---------+
|  knattrass0@loc.gov|    Kaspar|  Male|  1| 244.159.51.76| Nattrass|
|rnulty1@multiply.com|  Rosamund|Female|  2|237.123.21.130|    Nulty|
+--------------------+----------+------+---+--------------+---------+
only showing top 2 rows
```



```
scala> val df3 = spark.read.format("csv").load("/tmp/output/csv")
df3: org.apache.spark.sql.DataFrame = [_c0: string, _c1: string ... 4 more fields]

scala> df3.show(2)
+--------------------+--------+------+---+--------------+--------+
|                 _c0|     _c1|   _c2|_c3|           _c4|     _c5|
+--------------------+--------+------+---+--------------+--------+
|  knattrass0@loc.gov|  Kaspar|  Male|  1| 244.159.51.76|Nattrass|
|rnulty1@multiply.com|Rosamund|Female|  2|237.123.21.130|   Nulty|
+--------------------+--------+------+---+--------------+--------+
only showing top 2 rows
```

17) Write the data using the SaveMode



```
scala> val df1 = spark.read.format("parquet").load("/tmp/output/parquet")
df1: org.apache.spark.sql.DataFrame = [email: string, first_name: string ... 4 more fields]

scala> import org.apache.spark.sql.SaveMode
import org.apache.spark.sql.SaveMode

scala> df1.write.format("csv").mode(SaveMode.Append).save("/tmp/output/csv")

scala> val df2 = spark.read.format("csv").load("/tmp/output/csv")
df2: org.apache.spark.sql.DataFrame = [_c0: string, _c1: string ... 4 more fields]

scala> df2.count
res27: Long = 2000
```