**Closure** :

A closure is a function, whose return value depends on the value of one or more variables declared outside this function.

Snippet:

case class Field(…..)

// Funtion declaration 1

def convertToField (line : String) : Field {…..} ///Return Field object

// Funtion declaration 2

val convertToField2 = (line : String) => {….. return Field(….)}

///Return Field object

rdd.map(convertToField)

Here return of map depends of return of ‘convertToField’ method

Curring:

Technique of transforming a function that takes multiple arguments into a function that takes a single argument

Snippet Before currying:

def add(x:Int, y:Int) = x + y

add(1, 2) // 3

add(7, 3) // 10

Snippet after currying:

def add(x:Int) = (y:Int) => x + y

add(1)(2) // 3

add(7)(3) // 10

sddsd

**map function:**

‘map’ function applied the function definition passed on to it against each entry of rdd. It is a Transformation

Snippet:

Let Rdd be of type RDD<Int>

rdd.map.(\_.split(“,”))

‘\_.split(“,”)’ short hand to - split each entry of rdd (‘\_’ is a place holder)mapped by ‘map function)

NOTE: ‘map’ function support closure.

**reduce function**

‘reduce’ function return a single output after traversing through rdd. It is an Action

Snippet:

Let Rdd be of type RDD<Int>

rdd.reduce((v1, v2) => v1+v2 )

In the above example ‘reduce’ function take first entry of int RDD ‘v1’ add it to second entry ‘v2’ and so on so forth till the all the entries are traversed. The result of this snippet will be sum of all elements in rdd

**‘aggragate’ function**

‘aggragate’ function return two values