

# ***BIG DATA ANALYTICS WITH SPARK LAB***

## **Program – I – Program on spark context creation and basic transformations**

```
1 import org.apache.spark.SparkContext
2 import org.apache.spark.SparkConf
3 import org.apache.spark.sql.SparkSession
4 object mapTest{
5   def main(args:Array[String])={
6     val spark = SparkSession.builder.appName("mapExample").master("local").getOrCreate()
7     val data = spark.read.text("/FileStore/tables/spark_test-2.txt").rdd
8     val mapFile = data.map(line => (line,line.length))
9     mapFile.foreach(println)
10  }
11 }
```

```
import org.apache.spark.SparkContext
import org.apache.spark.SparkConf
import org.apache.spark.sql.SparkSession
defined object mapTest
```

Command took 28.39 seconds -- by svgbob11@gmail.com at 2/22/2023, 11:47:10 AM on cluster22/02

```
1 val parallel = sc.parallelize(1 to 9)
2 val par2 = sc.parallelize(5 to 15)
3 parallel.union(par2).collect
```

► (1) Spark Jobs

```
parallel: org.apache.spark.rdd.RDD[Int] = ParallelCollectionRDD[32] at parallelize at command-4193893003589037:1
par2: org.apache.spark.rdd.RDD[Int] = ParallelCollectionRDD[33] at parallelize at command-4193893003589037:2
res7: Array[Int] = Array(1, 2, 3, 4, 5, 6, 7, 8, 9, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15)
```

Command took 0.67 seconds -- by svgbob11@gmail.com at 2/22/2023, 12:08:50 PM on cluster22/02

```
1 val parallel = sc.parallelize(1 to 9)
2 val par2 = sc.parallelize(5 to 15)
3 parallel.intersection(par2).distinct.collect
```

▶ (1) Spark Jobs

```
parallel: org.apache.spark.rdd.RDD[Int] = ParallelCollectionRDD[62] at parallelize at command-4193893003589038:1
par2: org.apache.spark.rdd.RDD[Int] = ParallelCollectionRDD[63] at parallelize at command-4193893003589038:2
res16: Array[Int] = Array(8, 9, 5, 6, 7)
```

Command took 1.07 seconds -- by svgbob11@gmail.com at 2/22/2023, 12:24:54 PM on cluster22/02

```
1 val names1 = sc.parallelize(List("abe", "abby", "apple")).map(a => (a, 1))
2 val names2 = sc.parallelize(List("apple", "beatty", "beatrice")).map(a => (a, 1))
3 names1.join(names2).collect
```

▶ (1) Spark Jobs

```
names1: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[50] at map at command-4193893003589041:1
names2: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[52] at map at command-4193893003589041:2
res12: Array[(String, (Int, Int))] = Array((apple,(1,1)))
```

Command took 1.08 seconds -- by svgbob11@gmail.com at 2/22/2023, 12:17:43 PM on cluster22/02

```
1 names1.leftOuterJoin(names2).collect
```

► (1) Spark Jobs

```
res13: Array[(String, (Int, Option[Int]))] = Array((abby,(1,None)), (apple,(1,Some(1))), (abe,(1,None)))
```

Command took 0.83 seconds -- by svgbob11@gmail.com at 2/22/2023, 12:17:53 PM on cluster22/02

Cmd 7

```
1 names1.rightOuterJoin(names2).collect
```

► (1) Spark Jobs

```
res14: Array[(String, (Option[Int], Int))] = Array((apple,(Some(1),1)), (beatty,(None,1)), (beatrice,(None,1)))
```

Command took 0.75 seconds -- by svgbob11@gmail.com at 2/22/2023, 12:18:06 PM on cluster22/02

Cmd 8

```
1 val rdd1 = spark.sparkContext.parallelize(Seq((1,"jan",2016),(3,"nov",2014),(16,"feb",2014)))
2 val rdd2 = spark.sparkContext.parallelize(Seq((5,"dec",2014),(17,"sep",2015)))
3 val rdd3 = spark.sparkContext.parallelize(Seq((6,"dec",2011),(16,"may",2015)))
4 val rddUnion = rdd1.union(rdd2).union(rdd3)
5 val a=rddUnion.take(100)
6 a.foreach(p=>{
7     println(p)
8 })
```

► (3) Spark Jobs

(1,jan,2016)

(3,nov,2014)

(16,feb,2014)

(5,dec,2014)

(17,sep,2015)

(6,dec,2011)

(16,may,2015)

rdd1: org.apache.spark.rdd.RDD[(Int, String, Int)] = ParallelCollectionRDD[73] at parallelize at command-4193893003589044:1

rdd2: org.apache.spark.rdd.RDD[(Int, String, Int)] = ParallelCollectionRDD[74] at parallelize at command-4193893003589044:2

rdd3: org.apache.spark.rdd.RDD[(Int, String, Int)] = ParallelCollectionRDD[75] at parallelize at command-4193893003589044:3

rddUnion: org.apache.spark.rdd.RDD[(Int, String, Int)] = UnionRDD[77] at union at command-4193893003589044:4

a: Array[(Int, String, Int)] = Array((1,jan,2016), (3,nov,2014), (16,feb,2014), (5,dec,2014), (17,sep,2015), (6,dec,2011), (16,may,2015))

Command took 1.34 seconds -- by svgbob11@gmail.com at 2/22/2023, 12:36:24 PM on cluster22/02

Cmd 9

```
1 val rdd1 = spark.sparkContext.parallelize(Seq((1,"jan",2016),(3,"nov",2014), (16,"feb",2014)))
2 val rdd2 = spark.sparkContext.parallelize(Seq((5,"dec",2014),(1,"jan",2016)))
3 val intersect = rdd1.intersection(rdd2)
4 val a=intersect.take(100)
5 a.foreach(p=>{
6     println(p)
7 })
```

▶ (3) Spark Jobs

(1,jan,2016)

rdd1: org.apache.spark.rdd.RDD[(Int, String, Int)] = ParallelCollectionRDD[78] at parallelize at command-4193893003589045:1

rdd2: org.apache.spark.rdd.RDD[(Int, String, Int)] = ParallelCollectionRDD[79] at parallelize at command-4193893003589045:2

intersect: org.apache.spark.rdd.RDD[(Int, String, Int)] = MapPartitionsRDD[85] at intersection at command-4193893003589045:3

a: Array[(Int, String, Int)] = Array((1,jan,2016))

Command took 0.82 seconds -- by svgbob11@gmail.com at 2/22/2023, 12:36:37 PM on cluster22/02

Cmd 10

```
1 val rdd1 = spark.sparkContext.parallelize(Seq((1,"jan",2016),(3,"nov",2014),(16,"feb",2014),(3,"nov",2014)))
2 val result = rdd1.distinct()
3 val a=result.take(100)
4 a.foreach(p=>{
5     println(p)
6 })
```

▶ (3) Spark Jobs

(3,nov,2014)

(16,feb,2014)

(1,jan,2016)

rdd1: org.apache.spark.rdd.RDD[(Int, String, Int)] = ParallelCollectionRDD[86] at parallelize at command-4193893003589046:1

result: org.apache.spark.rdd.RDD[(Int, String, Int)] = MapPartitionsRDD[89] at distinct at command-4193893003589046:2

a: Array[(Int, String, Int)] = Array((3,nov,2014), (16,feb,2014), (1,jan,2016))

Command took 0.69 seconds -- by svgbob11@gmail.com at 2/22/2023, 12:36:57 PM on cluster22/02

Cmd 11

```
1 val data = spark.sparkContext.parallelize(Array(('k',5),('s',3),('s',4),('p',7),('p',5),('t',8),('k',6)),3)
2 val group = data.groupByKey().collect()
3 group.foreach(println)
```

▶ (1) Spark Jobs

(s,CompactBuffer(3, 4))

(p,CompactBuffer(7, 5))

(t,CompactBuffer(8))

(k,CompactBuffer(5, 6))

data: org.apache.spark.rdd.RDD[(Char, Int)] = ParallelCollectionRDD[90] at parallelize at command-4193893003589048:1

group: Array[(Char, Iterable[Int])] = Array((s,CompactBuffer(3, 4)), (p,CompactBuffer(7, 5)), (t,CompactBuffer(8)), (k,CompactBuffer(5, 6)))

Command took 1.22 seconds -- by svgbob11@gmail.com at 2/22/2023, 12:37:15 PM on cluster22/02

```

1  val words = Array("one","two","two","four","five","six","six","eight","nine","ten")
2  val data = spark.sparkContext.parallelize(words).map(w => (w,1)).reduceByKey(_+_).collect()
3  data.foreach(println)

```

► (1) Spark Jobs

(nine,1)

(six,2)

(five,1)

(two,2)

(ten,1)

(one,1)

(four,1)

(eight,1)

words: Array[String] = Array(one, two, two, four, five, six, six, eight, nine, ten)

data: Array[(String, Int)] = Array((nine,1), (six,2), (five,1), (two,2), (ten,1), (one,1), (four,1), (eight,1))

Command took 0.90 seconds -- by svgbob11@gmail.com at 2/22/2023, 12:37:49 PM on cluster22/02

---

Cmd 13

```

1  val data = spark.sparkContext.parallelize(Seq(("maths",52), ("english",75), ("science",82), ("computer",65), ("maths",85)))
2  val sorted = data.sortByKey().collect()
3  sorted.foreach(println)

```

► (2) Spark Jobs

(computer,65)

(english,75)

(maths,52)

(maths,85)

(science,82)

data: org.apache.spark.rdd.RDD[(String, Int)] = ParallelCollectionRDD[95] at parallelize at command-4193893003589050:1

sorted: Array[(String, Int)] = Array((computer,65), (english,75), (maths,52), (maths,85), (science,82))

```

1 val data = spark.sparkContext.parallelize(Array(('A',1),('b',2),('c',3)))
2 val data2 = spark.sparkContext.parallelize(Array(('A',4),('A',6),('b',7),('c',3),('c',8)))
3 val result = data.join(data2).collect()
4 result.foreach(println)

```

► (1) Spark Jobs

(A,(1,4))

(A,(1,6))

(b,(2,7))

(c,(3,3))

(c,(3,8))

data: org.apache.spark.rdd.RDD[(Char, Int)] = ParallelCollectionRDD[99] at parallelize at command-4193893003589051:1

data2: org.apache.spark.rdd.RDD[(Char, Int)] = ParallelCollectionRDD[100] at parallelize at command-4193893003589051:2

result: Array[(Char, (Int, Int))] = Array((A,(1,4)), (A,(1,6)), (b,(2,7)), (c,(3,3)), (c,(3,8)))

Command took 0.94 seconds -- by svgbob11@gmail.com at 2/22/2023, 12:38:39 PM on cluster22/02

Cmd 15

```

1 val rdd1 = spark.sparkContext.parallelize(Array("jan", "feb", "mar", "april", "may", "jun"), 3)
2 val result = rdd1.coalesce(2)
3 println(result.collect().mkString(", "))

```

► (1) Spark Jobs

jan,feb,mar,april,may,jun

rdd1: org.apache.spark.rdd.RDD[String] = ParallelCollectionRDD[104] at parallelize at command-4193893003589052:1

result: org.apache.spark.rdd.RDD[String] = CoalescedRDD[105] at coalesce at command-4193893003589052:2

Command took 0.69 seconds -- by svgbob11@gmail.com at 2/22/2023, 12:39:06 PM on cluster22/02

Shift+Enter to run

Shift+Ctrl+Enter to run selected text

## Program – II - Program on spark collections and actions

```
import org.apache.spark.sql.SparkSession
val spark = SparkSession
    .builder
    .appName("SparkSQLExampleApp")
    .getOrCreate()

val df = spark.read.option("header",true).csv("/FileStore/tables/emp1.csv")
val data = List(("Apple",10,3), ("Banana", 20,4), ("Orange", 30,6),
("Grapes", 40,8))
val rdd = spark.sparkContext.parallelize(data)

//rdd.collect.foreach(println)
//rdd.take(2).foreach(println)

println("show")
df.show()
```

### O/P:

► (2) Spark Jobs

►  df: org.apache.spark.sql.DataFrame = [eid: string, e name: string]

show

eid	e name
101	Sravani
102	Aditya
103	Rahul
104	Roshini
105	Kavitha
106	Anjan Kumar
107	Sruthi
108	Mohan
109	Raja
110	Ramana

```
println("collect")
df.rdd.collect.foreach(println)
```

**O/P:**

▶ (1) Spark Jobs

collect

[101,Sravani]

[102,Aditya]

[103,Rahul]

[104,Roshini]

[105,Kavitha]

[106,Anjan Kumar]

[107,Sruthi]

[108,Mohan]

[109,Raja]

[110,Ramana]

Command took 1.01 seconds -- by adityasharma.p@hotmail.com at 3/8/2023, 10:29:59 AM on Cluster

```
println("take")  
df.rdd.take(2).foreach(println)
```

**O/P:**

▶ (1) Spark Jobs

take

[101,Sravani]

[102,Aditya]

Command took 0.86 seconds -- by adityasharma.p@hotmail.com at 3/8/2023, 10:30:02 AM on Cluster

```
println("first")  
df.rdd.first()
```



▶ (1) Spark Jobs

first

res13: org.apache.spark.sql.Row = [101,Sravani]

Command took 0.66 seconds -- by adityasharma.p@hotmail.com at 3/8/2023, 10:32:22 AM on Cluster

```
println("count")
```

```
df.rdd.count()
```

**O/P:**

▶ (1) Spark Jobs

count

res14: Long = 10

Command took 0.73 seconds -- by adityasharma.p@hotmail.com at 3/8/2023, 10:33:06 AM on Cluster

```
println("countByKey")
```

```
//df.rdd.countByKey(5)
```

```
val rdd1 =
```

```
sc.parallelize(Seq(("Spark",78),("Hive",95),("spark",15),("HBase",25),("spark",  
39),("BigData",78),("spark",49)))
```

```
rdd1.countByKey()
```

▶ (1) Spark Jobs

countByKey

rdd1: org.apache.spark.rdd.RDD[(String, Int)] = ParallelCollectionRDD[54] at parallelize at command-1386146738819546:3

res22: scala.collection.Map[String,Long] = Map(Hive -> 1, BigData -> 1, HBase -> 1, spark -> 3, Spark -> 1)

Command took 0.88 seconds -- by adityasharma.p@hotmail.com at 3/8/2023, 10:47:41 AM on Cluster

## Program – III - Program on RDD creation and some functions

```
//The following program is in Python
from pyspark.sql import SparkSession
spark:SparkSession =
SparkSession.builder.appName("SparkByExamples.com").getOrCreate()
#Create RDD from parallelize
data = [1,2,3,4,5,6,7,8,9,10,11,12]
rdd=spark.sparkContext.parallelize(data)
#Create empty RDD with partition
rdd2 = spark.sparkContext.parallelize([],10) #This creates 10 partitions

print("initial partition count:"+str(rdd.getNumPartitions()))
#Outputs: initial partition count
```

```
initial partition count:8
```

```
Command took 0.12 seconds -- by adityasharma.p@hotmail.com at 3/16/2023, 11:34:14 AM on Cluster
```

```
reparRdd = rdd.repartition(4)
print("re-partition count:"+str(reparRdd.getNumPartitions()))
#Outputs: "re-partition count:4
```

```
re-partition count:4
```

```
Command took 0.26 seconds -- by adityasharma.p@hotmail.com at 3/16/2023, 11:34:27 AM on Cluster
```

---

```
# Creates empty RDD with no partition
rdd = spark.sparkContext.emptyRDD
# rddString = spark.sparkContext.emptyRDD[String]

#Create empty RDD with partition
rdd2 = spark.sparkContext.parallelize([],10)

print("initial partition count:"+str(rdd.getNumPartitions()))
#Outputs: initial partition count
```

Cmd 5

```
1 print("initial partition count:"+str(rdd.getNumPartitions()))
2 #Outputs: initial partition count
```

⊕ **AttributeError:** 'function' object has no attribute 'getNumPartitions'

Command took 0.15 seconds -- by adityasharma.p@hotmail.com at 3/16/2023, 11:40:57 AM on Cluster

```
reparRdd = rdd.repartition(4)
print("re-partition count:"+str(reparRdd.getNumPartitions()))
#Outputs: "re-partition count:4
```

Cmd 6

```
1 reparRdd = rdd.repartition(4)
2 print("re-partition count:"+str(reparRdd.getNumPartitions()))
3 #Outputs: "re-partition count:4
```

⊕ **AttributeError:** 'function' object has no attribute 'repartition'

Command took 0.10 seconds -- by adityasharma.p@hotmail.com at 3/16/2023, 11:41:55 AM on Cluster

## **Transformations and Actions**

#Create RDD from external Data source

```
rdd2 = spark.sparkContext.textFile("/FileStore/tables/Project_Gutenberg_s.txt")
```

#Reads entire file into a RDD as single record.

```
rdd3 =
```

```
spark.sparkContext.wholeTextFiles("/FileStore/tables/Project_Gutenberg_s.txt")
```

```
rdd = spark.sparkContext.textFile("/FileStore/tables/Project_Gutenberg_s.txt")
```

```
rdd2 = rdd.flatMap(lambda x: x.split(" "))
```

```
rdd3 = rdd2.map(lambda x: (x,1))
```

```
rdd4 = rdd3.reduceByKey(lambda a,b: a+b)
```

```
rdd5 = rdd4.map(lambda x: (x[1],x[0])).sortByKey()
```

#Print rdd5 result to console

```
print(rdd5.collect())
```

▶ (3) Spark Jobs

```
[(9, 'Project'), (9, 'Gutenberg's'), (19, 'Alice's'), (19, 'in'), (19, 'Lewis'), (19, 'Carroll'), (19, 'Adventures'), (19, 'Wonderland'), (19, 'by'), (27, 'is'), (27, 'use'), (27, 'of'), (27, 'anyone'), (27, 'anywhere'), (27, 'at'), (27, 'no'), (27, 'This'), (27, 'eBook'), (27, 'for'), (27, 'the'), (27, 'cost'), (27, 'and'), (27, 'with')]
```

Command took 6.89 seconds -- by adityasharma.p@hotmail.com at 3/16/2023, 11:43:31 AM on Cluster

```
rdd6 = rdd5.filter(lambda x : 'an' in x[1])
print(rdd6.collect())
```

▶ (1) Spark Jobs

```
[(19, 'Wonderland'), (27, 'anyone'), (27, 'anywhere'), (27, 'and')]
```

Command took 0.33 seconds -- by adityasharma.p@hotmail.com at 3/16/2023, 11:47:06 AM on Cluster

# Action - count

```
print("Count : "+str(rdd6.count()))
```

▶ (1) Spark Jobs

Count : 4

Command took 0.29 seconds -- by adityasharma.p@hotmail.com at 3/16/2023, 11:47:12 AM on Cluster

# Action - first

```
firstRec = rdd6.first()
```

```
print("First Record : "+str(firstRec[0]) + "," + firstRec[1])
```

▶ (1) Spark Jobs

First Record : 19,Wonderland

Command took 0.27 seconds -- by adityasharma.p@hotmail.com at 3/16/2023, 11:52:34 AM on Cluster

```
# Action - max
datMax = rdd6.max()
print("Max Record : "+str(datMax[0]) + ","+ datMax[1])
```

▶ (1) Spark Jobs

Max Record : 27,anywhere

Command took 0.31 seconds -- by adityasharma.p@hotmail.com at 3/16/2023, 11:52:38 AM on Cluster

```
# Action - reduce
totalWordCount = rdd6.reduce(lambda a,b: (a[0]+b[0],a[1]))
print("dataReduce Record : "+str(totalWordCount[0]))
```

▶ (1) Spark Jobs

dataReduce Record : 100

Command took 0.29 seconds -- by adityasharma.p@hotmail.com at 3/16/2023, 11:52:41 AM on Cluster

```
# Action - take
data3 = rdd6.take(3)
for f in data3:
    print("data3 Key:"+ str(f[0]) +", Value:"+f[1])
```

▶ (1) Spark Jobs

data3 Key:19, Value:Wonderland

data3 Key:27, Value:anyone

data3 Key:27, Value:anywhere

Command took 0.23 seconds -- by adityasharma.p@hotmail.com at 3/16/2023, 11:52:44 AM on Cluster

```
# Action - collect
data = rdd6.collect()
for f in data:
```

```
print("Key:"+ str(f[0]) +", Value:"+f[1])
```

```
▶ (1) Spark Jobs
```

```
Key:19, Value:Wonderland
```

```
Key:27, Value:anyone
```

```
Key:27, Value:anywhere
```

```
Key:27, Value:and
```

```
Command took 0.22 seconds -- by adityasharma.p@hotmail.com at 3/16/2023, 11:52:48 AM on Cluster
```

```
rdd6.saveAsTextFile("/tmp/wordCount")
```

//The following program is in scala

```
%scala
```

```
import org.apache.spark.rdd.RDD
```

```
import org.apache.spark.sql.Session
```

```
//object WordCountExample {
```

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

```
    val spark:Session =  
    Session.builder.appName("SparkByExamples.com").getOrCreate()
```

```
    val sc = spark.sparkContext
```

```
    val rdd:RDD[String] =  
    sc.textFile("/FileStore/tables/Project_Gutenberg_s.txt")  
    println("initial partition count:"+rdd.getNumPartitions)
```

```
    val repartitioned = rdd.repartition(4)  
    println("re-partition count:"+repartitioned.getNumPartitions)
```

```
//rdd.coalesce(3)
```

```
rdd.collect().foreach(println)
```

```
// rdd flatMap transformation
```

```
val rdd2 = rdd.flatMap(f=>f.split(" "))
```

```

rdd2.foreach(f=>println(f))

//Create a Tuple by adding 1 to each word
val rdd3:RDD[(String,Int)]= rdd2.map(m=>(m,1))
rdd3.foreach(println)

//Filter transformation
val rdd4 = rdd3.filter(a=> a._1.startsWith("a"))
rdd4.foreach(println)

//ReduceBy transformation
val rdd5 = rdd3.reduceByKey(_ + _)
rdd5.foreach(println)

//Swap word,count and sortByKey transformation
val rdd6 = rdd5.map(a=>(a._2,a._1)).sortByKey()
println("Final Result")

//Action - foreach
rdd6.foreach(println)

//Action - count
println("Count : "+rdd6.count())

//Action - first
val firstRec = rdd6.first()
println("First Record : "+firstRec._1 + ", "+ firstRec._2)

//Action - max
val datMax = rdd6.max()
println("Max Record : "+datMax._1 + ", "+ datMax._2)

//Action - reduce
val totalWordCount = rdd6.reduce((a,b) => (a._1+b._1,a._2))
println("dataReduce Record : "+totalWordCount._1)
//Action - take
val data3 = rdd6.take(3)
data3.foreach(f=>{
  println("data3 Key:"+ f._1 +", Value:"+f._2)
})

```

```
//Action - collect
val data = rdd6.collect()
data.foreach(f=>{
  println("Key:"+ f._1 +", Value:"+f._2)
})
//Action - saveAsTextFile
rdd5.saveAsTextFile("C:\\Users\\Aditya
Sharma\\OneDrive\\Desktop\\wordCount")
```

**Output:**





```
#Convert PySpark RDD to DataFrame
```

```
#Using rdd.toDF() function
```

```
df = rdd.toDF()
```

```
df.printSchema()
```

```
df.show(truncate=False)
```

#toDF() has another signature that takes arguments to define column names as shown below.

```
deptColumns = ["dept_name", "dept_id"]
```

```
df2 = rdd.toDF(deptColumns)
```

```
df2.printSchema()
```

```
df2.show(truncate=False)
```

```
#Using PySpark createDataFrame() function
```

```
deptDF = spark.createDataFrame(rdd, schema = deptColumns)
```

```
deptDF.printSchema()
```

```
deptDF.show(truncate=False)
```

```
#Using createDataFrame() with StructType schema
```

```
from pyspark.sql.types import StructType, StructField, StringType
```

```
deptSchema = StructType([
```

```
    StructField('dept_name', StringType(), True),
```

```
    StructField('dept_id', StringType(), True)
```

```
])
```

```
deptDF1 = spark.createDataFrame(rdd, schema = deptSchema)
```

```
deptDF1.printSchema()
```

```
deptDF1.show(truncate=False)
```

**Output:**



>

► (5) Spark Jobs

► df: pyspark.sql.dataframe.DataFrame = [\_1: string, \_2: long]

```
root
|-- _1: string (nullable = true)
|-- _2: long (nullable = true)
```

```
+-----+-----+
|_1      |_2 |
+-----+-----+
|Finance  |10 |
|Marketing|20 |
|Sales    |30 |
|IT       |40 |
+-----+-----+
```

► (5) Spark Jobs

► df2: pyspark.sql.dataframe.DataFrame = [dept\_name: string, dept\_id: long]

```
root
|-- dept_name: string (nullable = true)
|-- dept_id: long (nullable = true)
```

```
+-----+-----+
|dept_name|dept_id|
+-----+-----+
|Finance  |10      |
|Marketing|20      |
|Sales    |30      |
|IT       |40      |
+-----+-----+
```

► (5) Spark Jobs

► deptDF: pyspark.sql.dataframe.DataFrame = [dept\_name: string, dept\_id: long]

```
root
|-- dept_name: string (nullable = true)
|-- dept_id: long (nullable = true)
```

```
+-----+-----+
|dept_name|dept_id|
+-----+-----+
|Finance  |10      |
|Marketing|20      |
|Sales    |30      |
|IT       |40      |
+-----+-----+
```

► (3) Spark Jobs

►  deptDF1: pyspark.sql.dataframe.DataFrame = [dept\_name: string, dept\_id: string]

```
root
|-- dept_name: string (nullable = true)
|-- dept_id: string (nullable = true)
```

```
+-----+-----+
|dept_name|dept_id|
+-----+-----+
|Finance  |10      |
|Marketing|20      |
|Sales    |30      |
|IT       |40      |
+-----+-----+
```

**//Program to perform actions**

```
from pyspark.sql import SparkSession
```

```
spark =
```

```
SparkSession.builder.appName('SparkByExamples.com').getOrCreate()
()
```

```
data=[("Z", 1),("A", 20),("B", 30),("C", 40),("B", 30),("B", 60)]
```

```
inputRDD = spark.sparkContext.parallelize(data)
```

```
listRdd = spark.sparkContext.parallelize([1,2,3,4,5,3,2])
```

**#aggregate**

```
seqOp = (lambda x, y: x + y)
```

```
combOp = (lambda x, y: x + y)
```

```
agg=listRdd.aggregate(0, seqOp, combOp)
```

```
print(agg) # output 20
```

### **#aggregate 2**

```
seqOp2 = (lambda x, y: (x[0] + y, x[1] + 1))  
combOp2 = (lambda x, y: (x[0] + y[0], x[1] + y[1]))  
agg2=listRdd.aggregate((0, 0), seqOp2, combOp2)  
print(agg2) # output (20,7)
```

```
agg2=listRdd.treeAggregate(0,seqOp, combOp)  
print(agg2) # output 20
```

### **#fold**

```
from operator import add  
foldRes=listRdd.fold(0, add)  
print(foldRes) # output 20
```

### **#reduce**

```
redRes=listRdd.reduce(add)  
print(redRes) # output 20
```

### **#treeReduce. This is similar to reduce**

```
add = lambda x, y: x + y  
redRes=listRdd.treeReduce(add)  
print(redRes) # output 20
```

### **#Collect**

```
data = listRdd.collect()  
print(data)
```

### **#count, countApprox, countApproxDistinct**

```
print("Count : "+str(listRdd.count()))  
#Output: Count : 20  
print("countApprox : "+str(listRdd.countApprox(1200)))  
#Output: countApprox : (final: [7.000, 7.000])  
print("countApproxDistinct : "+str(listRdd.countApproxDistinct()))  
#Output: countApproxDistinct : 5
```

```
print("countApproxDistinct :  
"+str(inputRDD.countApproxDistinct()))  
#Output: countApproxDistinct : 5
```

### **#countByValue, countByValueApprox**

```
print("countByValue : "+str(listRdd.countByValue()))
```

### **#first**

```
print("first : "+str(listRdd.first()))  
#Output: first : 1  
print("first : "+str(inputRDD.first()))  
#Output: first : (Z,1)
```

### **#top**

```
print("top : "+str(listRdd.top(2)))  
#Output: take : 5,4  
print("top : "+str(inputRDD.top(2)))  
#Output: take : (Z,1),(C,40)
```

### **#min**

```
print("min : "+str(listRdd.min()))  
#Output: min : 1  
print("min : "+str(inputRDD.min()))  
#Output: min : (A,20)
```

### **#max**

```
print("max : "+str(listRdd.max()))  
#Output: max : 5  
print("max : "+str(inputRDD.max()))  
#Output: max : (Z,1)
```

### **#take, takeOrdered, takeSample**

```
print("take : "+str(listRdd.take(2)))  
#Output: take : 1,2
```

```
print("takeOrdered : "+ str(listRdd.takeOrdered(2)))
```

## Output

```
20
(20, 7)
20
20
20
20
[1, 2, 3, 4, 5, 3, 2]
Count : 7
countApprox : 7
countApproxDistinct : 5
countApproxDistinct : 5
countByValue : defaultdict(<class 'int'>, {1: 1, 2: 2, 3: 2, 4: 1, 5: 1})
first : 1
first : ('Z', 1)
top : [5, 4]
top : [('Z', 1), ('C', 40)]
min : 1
min : ('A', 20)
max : 5
max : ('Z', 1)
take : [1, 2]
takeOrdered : [1, 2]
```

## **Program – V – Program on text file to create view with spark sql**

```
//By Using Spark with Json File
import org.apache.spark.sql.Session
val spark = Session
    .builder
    .appName("SparkSQLExampleApp")
    .getOrCreate()
// Path to data set
val
jsonFile="/databricks-datasets/learning-spark-v2/flights/summary-dat
a/json/*"

// Read and create a temporary view
// Infer schema (note that for larger files you may want to specify the
schema)
val df = spark.read.format("json")
    .option("inferSchema", "true")
    .option("header", "true")
    .load(jsonFile)
// Create a temporary view
df.createOrReplaceTempView("us_delay_flights_tbl1")

spark.sql("SELECT * FROM us_delay_flights_tbl1").show()
```



## Output:

► (1) Spark Jobs

DEST_COUNTRY_NAME	ORIGIN_COUNTRY_NAME	count
United States	Romania	15
United States	Croatia	1
United States	Ireland	344
Egypt	United States	15
United States	India	62
United States	Singapore	1
United States	Grenada	62
Costa Rica	United States	588
Senegal	United States	40
Moldova	United States	1
United States	Sint Maarten	325
United States	Marshall Islands	39
Guyana	United States	64
Malta	United States	1
Anguilla	United States	41
Bolivia	United States	30
United States	Paraguay	6
Algeria	United States	4

```
//By Using Spark with CSV File
import org.apache.spark.sql.Session
val spark = Session
    .builder
    .appName("SparkSQLExampleApp")
    .getOrCreate()

// Path to data set
val
csvFile="/databricks-datasets/learning-spark-v2/flights/departuredelay
s.csv"

// Read and create a temporary view
// Infer schema (note that for larger files you may want to specify the
schema)
val df = spark.read.format("csv")
    .option("inferSchema", "true")
    .option("header", "true")
    .load(csvFile)
// Create a temporary view
df.createOrReplaceTempView("us_delay_flights_tbl")

val schema = "date STRING, delay INT, distance INT, origin
STRING, destination STRING"

spark.sql("select * from us_delay_flights_tbl").show()
```

## Output:

► (1) Spark Jobs

date	delay	distance	origin	destination
1011245	6	602	ABE	ATL
1020600	-8	369	ABE	DTW
1021245	-2	602	ABE	ATL
1020605	-4	602	ABE	ATL
1031245	-4	602	ABE	ATL
1030605	0	602	ABE	ATL
1041243	10	602	ABE	ATL
1040605	28	602	ABE	ATL
1051245	88	602	ABE	ATL
1050605	9	602	ABE	ATL
1061215	-6	602	ABE	ATL
1061725	69	602	ABE	ATL
1061230	0	369	ABE	DTW
1060625	-3	602	ABE	ATL
1070600	0	369	ABE	DTW
1071725	0	602	ABE	ATL
1071230	0	369	ABE	DTW
1070625	0	602	ABE	ATL

## **Program – VI – Program on spark sql with sql operations**

```
// In Scala
import org.apache.spark.sql.functions._
// Set file paths
val delaysPath =
"/databricks-datasets/learning-spark-v2/flights/departuredelays.csv"
val airportsPath =
"/databricks-datasets/learning-spark-v2/flights/airport-codes-na.txt"

// Obtain airports data set
val airports = spark.read
.option("header", "true")
.option("inferSchema", "true")
.option("delimiter", "\t")
.csv(airportsPath)
airports.createOrReplaceTempView("airports_na")

// Obtain departure Delays data set
val delays = spark.read
.option("header", "true")
.csv(delaysPath)
.withColumn("delay", expr("CAST(delay as INT) as delay"))
.withColumn("distance", expr("CAST(distance as INT) as distance"))
delays.createOrReplaceTempView("departureDelays")

// Create temporary small table
val foo = delays.filter(
expr("""origin == 'SEA' AND destination == 'SFO' AND
date like '01010%' AND delay > 0"""))
foo.createOrReplaceTempView("foo")

spark.sql("SELECT * FROM airports_na LIMIT 10").show()
```

## Output:

► (1) Spark Jobs

City	State	Country	IATA
Abbotsford	BC	Canada	YXX
Aberdeen	SD	USA	ABR
Abilene	TX	USA	ABI
Akron	OH	USA	CAK
Alamosa	CO	USA	ALS
Albany	GA	USA	ABY
Albany	NY	USA	ALB
Albuquerque	NM	USA	ABQ
Alexandria	LA	USA	AEX
Allentown	PA	USA	ABE

```
spark.sql("SELECT * FROM departureDelays LIMIT 10").show()
```

## Output:

► (1) Spark Jobs

date	delay	distance	origin	destination
01011245	6	602	ABE	ATL
01020600	-8	369	ABE	DTW
01021245	-2	602	ABE	ATL
01020605	-4	602	ABE	ATL
01031245	-4	602	ABE	ATL
01030605	0	602	ABE	ATL
01041243	10	602	ABE	ATL
01040605	28	602	ABE	ATL
01051245	88	602	ABE	ATL
01050605	9	602	ABE	ATL

```
spark.sql("SELECT * FROM foo").show()
```

**Output:**

► (3) Spark Jobs

date	delay	distance	origin	destination
01010710	31	590	SEA	SFO
01010955	104	590	SEA	SFO
01010730	5	590	SEA	SFO