Assignment -19

Task 1

1. **Write a program to read a text file and print the number of rows of data in the document.**

The number of rows present in the file is achieved by below commands.

First the text file is loaded into the variable ‘data’. Now the number of rows is achieved by using command **data.count()**

The below screenshot shows the data present in the file.

```
File  Edit  Format  View  Help
This is a BDHS Session
This is a BDHS Session
This is a BDHS Session
This is a BDHS Session
This is a BDHS Session
```

```scala
package RDD_Deep_Dive_Assign

import org.apache.spark.sql.SparkSession

object Num_Rows {

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

    println("hey scala")

    val spark = SparkSession
      .builder()
      .master( master = "local")
      .appName( name = "Number of Rows in a file ")
      .config("spark.some.config.option", "some-value")
      .getOrCreate()

    println("Spark Session Object created")


    val data = spark.sparkContext.textFile( path = "D:\\barath\\data.txt");

    println("Number of Rows->>"+data.count())
  }

}
```

The number of rows present in the file is '**22**' as shown below.

```
18/10/30 20:07:15 INFO Executor: Finished task 0.
Number of Rows->>5
18/10/30 20:07:15 INFO TaskSetManager: Finished ta
18/10/30 20:07:15 INFO TaskSchedulerImpl: Removed
18/10/30 20:07:15 INFO DAGScheduler: ResultStage (
```

2. **Write a program to read a text file and print the number of words in the document.**

The number of words present in the file is found using the commands shown in the below screenshot.

```scala
package RDD_Deep_Dive_Assign

import org.apache.spark.sql.SparkSession

object Num_words {

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

    println("hey scala")

    val spark = SparkSession
      .builder()
      .master( master = "local")
      .appName( name = "Number of words in a file ")
      .config("spark.some.config.option", "some-value")
      .getOrCreate()

    println("Spark Session Object created")

    val textFile = spark.sparkContext.textFile( path = "D:\\barath\\data.txt")

    println("loaded data file")
    val counts = textFile.flatMap(line => line.split( regex = " "))

      println("Number of words in the file is : " +counts.count())

  }
}
```

The output of the above code is shown in the below screenshot.

```
18/10/30 20:12:57 INFO TaskSchedulerImpl: Remove
18/10/30 20:12:57 INFO DAGScheduler: ResultStage
Number of words in the file is : 25
18/10/30 20:12:57 INFO DAGScheduler: Job 0 finis
18/10/30 20:12:57 INFO SparkContext: Invoking st
18/10/30 20:12:57 INFO SparkUI: Stopped Spark we
```

3. **We have a document where the word separator is -, instead of space. Write a spark code, to obtain the count of the total number of words present in the document.**

The data present in the file is as shown below.

data - Notepad

File Edit Format View Help

```
This-is-a-BDHS-Session
This-is-a-BDHS-Session
This-is-a-BDHS-Session
This-is-a-BDHS-Session
This-is-a-BDHS-Session
This-is-a-BDHS-Session
```

The number of words present in the file with '-'separated text is found using the code present in the below screenshot.

```
package RDD_Deep_Dive_Assign

import org.apache.spark.sql.SparkSession

object Num_words2 {

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

    println("hey scala")

    val spark = SparkSession
      .builder()
      .master( master = "local")
      .appName( name = "Number of words in a file ")
      .config("spark.some.config.option", "some-value")
      .getOrCreate()

    println("Spark Session Object created")

    val textFile = spark.sparkContext.textFile( path = "D:\\barath\\data.txt")

    val counts = textFile.flatMap(line => line.split( regex = "-"))

    println("Number of words in the file is : " +counts.count())

  }
}
```

The output of the above code is as shown in the below screenshot.

```
18/10/30 20:11:46 INFO DAGScheduler: Job 0 finis
Number of words in the file is : 25
18/10/30 20:11:46 INFO SparkContext: Invoking st
18/10/30 20:11:46 INFO SparkUI: Stopped Spark we
```

Task 2

Problem Statement 1:

1. **Read the text file, and create a tupled rdd.**
2. **Find the count of total number of rows present.**
3. **What is the distinct number of subjects present in the entire school**
4. **What is the count of the number of students in the school, whose name is Mathew and marks is 55**

1) I have created a tupled RDD with name as key and the subject, grades and the marks as values.

2) The number of rows present in the file found using the command '**dataFile.count()**' as shown in the screenshot.

3) The distinct number of subjects present in the school has been found using the commands shown in the screenshot.

4) The number students in the school where name= '**Mathew**' and marks = '**55**' is found using the commands shown in the screenshot.

```
object Prbm_Stmt_1 {

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

    println("hey scala")

    val spark = SparkSession
      .builder()
      .master( master = "local")
      .appName( name = "Number of Rows in a file ")
      .config("spark.some.config.option", "some-value")
      .getOrCreate()

    println("Spark Session Object created")

    //Reading the text file, and create a tupled rdd
    val textFile = spark.sparkContext.textFile( path = "D:\\barath\\19_Dataset.txt")
    val arrayTuples = textFile.map(x=>x.split( regex = ",")).map(x => (x(0),x(1),x(2),x(3).toInt,x(4).toInt)).collect
    arrayTuples.foreach(println)


    //Find the count of total number of rows present
    println("Number of Rows->>"+textFile.count())


    //What is the distinct number of subjects present in the entire school
    val array = textFile.map(x=>x.split( regex = ",")).map(x => (x(1),1))
    val reduce = array.reduceByKey((x,y) => (x+y))
    reduce.foreach(println)

    //What is the count of the number of students in the school, whose name is Mathew and marks is 55
    val baseRDD = spark.sparkContext.textFile( path = "D:\\barath\\19_Dataset.txt").map(x => ((x.split( regex = ",")(0),x.split( regex = ",")(3).toInt),1))
    val RDDfilter = baseRDD.filter(x=>x._1._1 == "Mathew" && x._1._2 == 55)
    val RDDreduce = RDDfilter.reduceByKey((x,y)=> x+y).foreach(println)
  }
}
```

The output for tupled RDD is as shown below.

```
18/10/30 08:22:33 INFO TaskSchedulerImpl: Removed TaskSet 0.0, whose tasks have all completed, from pool
(Mathew,science,grade-3,45,12)
(Mathew,history,grade-2,55,13)
(Mark,maths,grade-2,23,13)
(Mark,science,grade-1,76,13)
(John,history,grade-1,14,12)
(John,maths,grade-2,74,13)
(Lisa,science,grade-1,24,12)
(Lisa,history,grade-3,86,13)
(Andrew,maths,grade-1,34,13)
(Andrew,science,grade-3,26,14)
(Andrew,history,grade-1,74,12)
(Mathew,science,grade-2,55,12)
(Mathew,history,grade-2,87,12)
(Mark,maths,grade-1,92,13)
(Mark,science,grade-2,12,12)
(John,history,grade-1,67,13)
(John,maths,grade-1,35,11)
(Lisa,science,grade-2,24,13)
(Lisa,history,grade-2,98,15)
(Andrew,maths,grade-1,23,16)
(Andrew,science,grade-3,44,14)
(Andrew,history,grade-2,77,11)
18/10/30 08:22:33 INFO SparkContext: Starting job: count at Prbm_Stmt_1.scala:27
```

The output for the number of rows in the file is shown in the below screenshot.

```
18/10/30 08:22:33 INFO HadoopRDD: Input split: file:/D:/barath
18/10/30 08:22:33 INFO Executor: Finished task 0.0 in stage 1.
Number of Rows->>22
18/10/30 08:22:33 INFO DAGScheduler: ResultStage 1 (count at P
18/10/30 08:22:33 INFO DAGScheduler: Job 1 finished: count at
```

The for the distinct number subjects present in the school is shown in the below screenshot.

```
18/10/30 08:22:34 INFO ShuffleBlockFetcherIterator: Started 0 remote fetches in 7 m
18/10/30 08:22:34 INFO Executor: Finished task 0.0 in stage 3.0 (TID 3). 1809 bytes
(maths,6)
(history,8)
(science,8)
18/10/30 08:22:34 INFO DAGScheduler: ResultStage 3 (foreach at Prbm_Stmt_1.scala:33
18/10/30 08:22:34 INFO DAGScheduler: Job 2 finished: foreach at Prbm_Stmt_1.scala:3
```

The output for the question 4 has been shown in the below screenshot.

```
18/10/30 08:22:34 INFO ShuffleBlockFetche
18/10/30 08:22:34 INFO ShuffleBlockFetche
((Mathew,55),2)
18/10/30 08:22:34 INFO Executor: Finished
18/10/30 08:22:34 INFO TaskSetManager: Fi
```

Problem Statement 2:

1. **What is the count of students per grade in the school?**
2. **Find the average of each student (Note - Mathew is grade-1, is different from Mathew in some other grade!)**
3. **What is the average score of students in each subject across all grades?**
4. **What is the average score of students in each subject per grade?**
5. **For all students in grade-2, how many have average score greater than 50?**

1) The count of students per grade in the school is found using the commands shown in the screenshot.

2) The average of each student is found using the commands shown in the below screenshot.

3) The average score of students in each subject found using the commands shown in the below screenshot.

4) The average score of students in each subject per grade is found using the commands shown in the below screenshot.

5) The average score greater than 50 achieved by a student in grade-2 is found using the commands shown in the below screenshot.

```scala
package RDD_Deep_Dive_Assign

import org.apache.spark.sql.SparkSession

object Prbm_Stmt_2 {

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

      println("hey scala")

    val spark = SparkSession
      .builder()
      .master( master = "local")
      .appName( name = "Number of Rows in a file ")
      .config("spark.some.config.option", "some-value")
      .getOrCreate()

    println("Spark Session Object created")

    //What is the count of students per grade in the school
    val textFile = spark.sparkContext.textFile( path = "D:\\barath\\19_Dataset.txt")
    val array = textFile.map(x=>x.split( regex = ",")).map(x => (x(2),1)).reduceByKey((x,y) => x+y).foreach(println)

    //Find the average of each student (Note - Mathew is grade-1, is different from Mathew in some other grade!)
    val baserdd = spark.sparkContext.textFile( path = "D:\\barath\\19_Dataset.txt").map(x=> ((x.split( regex = ",")(0),x.split( regex = ",")(2)),x.split( regex = ",")(3).toInt))
    val rddmap = baserdd.mapValues(x=>(x, 1))
    val rddreduce = rddmap.reduceByKey((x,y) => (x._1+y._1, x._2+y._2))
    val stuAvg = rddreduce.mapValues{case(sum,count) => (1.0*sum)/count}
    stuAvg.foreach(println)

    //What is the average score of students in each subject across all grades
    val base= spark.sparkContext.textFile( path = "D:\\barath\\19_Dataset.txt").map(x=> ((x.split( regex = ",")(0),x.split( regex = ",")(1)),x.split( regex = ",")(3).toInt))
    val rdd = base.mapValues(x=>(x, 1))
    val rddred = rdd.reduceByKey((x,y) => (x._1+y._1, x._2+y._2))
    val stuAvgscore = rddred.mapValues{case(sum,count) => (1.0*sum)/count}
    stuAvgscore.foreach(println)
```

```scala
    //What is the average score of students in each subject per grade
    val baseRDD = spark.sparkContext.textFile( path = "D:\\barath\\19_Dataset.txt").map(x=>((x.split( regex = ",")(1),x.split( regex = ",")(2)),x.split( regex = ",")(3).toInt))
    val RDDmap = baseRDD.mapValues(x=>(x,1))
    val RDDre = RDDmap.reduceByKey((x,y)=>(x._1+y._1,x._2+y._2))
    val Avg_Grade = RDDre.mapValues{case(sum,count)=>(1.0*sum)/count}.foreach(println)

    //For all students in grade-2, how many have average score greater than 50
    val baseRD = spark.sparkContext.textFile( path = "D:\\barath\\19_Dataset.txt").map(x=>((x.split( regex = ",")(0),x.split( regex = ",")(2)),x.split( regex = ",")(3).toInt))
    val RDDma = baseRD.mapValues(x=>(x,1))
    val RDDreduc = RDDma.reduceByKey((x,y)=>(x._1+y._1,x._2+y._2))
    val RDDav = RDDreduc.mapValues{case(sum,count)=>(1.0*sum)/count}
    val RDDfilterma = RDDav.filter(x=>x._1._2 == "grade-2" && x._2>50).count()
    val RDDfilternap = RDDav.filter(x=>x._1._2 == "grade-2" && x._2>50).foreach(println)
  }

}
```

The output for the question 1 is shown in the below screenshot.

```
18/10/30 08:29:50 INFO ShuffleBlockFetcherIterator: Getting 1 non
18/10/30 08:29:50 INFO ShuffleBlockFetcherIterator: Started 0 rem
(grade-3,4)
(grade-1,9)
(grade-2,9)
18/10/30 08:29:50 INFO Executor: Finished task 0.0 in stage 1.0 (
18/10/30 08:29:50 INFO DAGScheduler: ResultStage 1 (foreach at Pr
18/10/30 08:29:50 INFO TaskSetManager: Finished task 0.0 in stage
```

The output for the question 2 is shown in the below screenshot.

```
18/10/30 08:29:50 INFO ShuffleBlockFetcherIterator:
18/10/30 08:29:50 INFO ShuffleBlockFetcherIterator:
((Lisa,grade-1),24.0)
((Mark,grade-2),17.5)
((Lisa,grade-2),61.0)
((Mathew,grade-3),45.0)
((Andrew,grade-2),77.0)
((Andrew,grade-1),43.666666666666664)
((Lisa,grade-3),86.0)
((John,grade-1),38.666666666666664)
((John,grade-2),74.0)
((Mark,grade-1),84.0)
((Andrew,grade-3),35.0)
((Mathew,grade-2),65.66666666666667)
18/10/30 08:29:50 INFO Executor: Finished task 0.0 i
18/10/30 08:29:50 INFO TaskSetManager: Finished task
18/10/30 08:29:50 INFO TaskSchedulerImpl: Removed Ta
```

The output for the question 3 is shown in the below screenshot.

```
18/10/30 08:29:50 INFO ShuffleBlockFetc
18/10/30 08:29:50 INFO ShuffleBlockFetc
((Lisa,history),92.0)
((Mark,maths),57.5)
((Andrew,science),35.0)
((Mark,science),44.0)
((Mathew,science),50.0)
((Andrew,maths),28.5)
((Mathew,history),71.0)
((John,maths),54.5)
((John,history),40.5)
((Lisa,science),24.0)
((Andrew,history),75.5)
18/10/30 08:29:50 INFO Executor: Finish
18/10/30 08:29:50 INFO TaskSetManager:
```

The output for the question 4 is shown in the below screenshot.

```
18/10/30 08:29:50 INFO TaskSetManager: Finish
18/10/30 08:29:50 INFO TaskSchedulerImpl: Rem
((history,grade-2),79.25)
((history,grade-3),86.0)
((maths,grade-1),46.0)
((science,grade-3),38.333333333333336)
((science,grade-1),50.0)
((science,grade-2),30.333333333333332)
((history,grade-1),51.666666666666664)
((maths,grade-2),48.5)
18/10/30 08:29:50 INFO DAGScheduler: ResultSt
18/10/30 08:29:50 INFO DAGScheduler: Job 3 fi
18/10/30 08:29:50 INFO MemoryStore: Block bro
18/10/30 08:29:50 INFO MemoryStore: Block bro
```

The output for the question 5 is shown in the below screenshot.

```
18/10/30 08:29:51 INFO Executor: Running task 0.0
18/10/30 08:29:51 INFO ShuffleBlockFetcherIterato
18/10/30 08:29:51 INFO ShuffleBlockFetcherIterato
((Lisa,grade-2),61.0)
((Andrew,grade-2),77.0)
((John,grade-2),74.0)
((Mathew,grade-2),65.66666666666667)
18/10/30 08:29:51 INFO Executor: Finished task 0.
18/10/30 08:29:51 INFO TaskSetManager: Finished t
18/10/30 08:29:51 INFO TaskSchedulerImpl: Removed
```

Problem Statement 3:

Are there any students in the college that satisfy the below criteria:

1. **Average score per student_name across all grades is same as average score per student_name per grade**

- To find the solution of above problem we will first calculate average of each student across all grades i.e. irrespective of grade. The commands are shown in the below screenshot.
- Now the second step of this problem is to find the average of each student per grade. We have used below code shown in the screenshot to find,
- Finally we are using intersection function between flatgradeAvg and flatnameAvg rdd's to find whether any common student is there.

```scala
package RDD_Deep_Dive_Assign

import org.apache.spark.sql.SparkSession

object prbm_stmt_3 {

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

    println("hey scala")

    val spark = SparkSession
      .builder()
      .master( master = "local")
      .appName( name = "Number of Rows in a file ")
      .config("spark.some.config.option", "some-value")
      .getOrCreate()

    //average of each student across all grades i.e. irrespective of grade
    val baseRDD1 = spark.sparkContext.textFile( path = "D:\\barath\\19_Dataset.txt").map(x => (x.split( regex = ",")(0), x.split( regex = ",")(3).toInt))
    val studAvg = baseRDD1.mapValues(x => (x, 1))
    val studReduce = studAvg.reduceByKey((x, y) => (x._1 + y._1, x._2 + y._2))
    val Avg_Stud = studReduce.mapValues { case (sum, count) => (1.0 * sum) / count }
    Avg_Stud.foreach(println)

    //average of each student per grade
    val baseRDD2 = spark.sparkContext.textFile( path = "D:\\barath\\19_Dataset.txt").map(x=>((x.split( regex = ",")(0),x.split( regex = ",")(2)),x.split( regex = ",")(3).toInt))
    val grade = baseRDD2.mapValues(x=>(x,1))
    val gradeReduce = grade.reduceByKey((x,y)=> (x._1+y._1,x._2+y._2))
    val gradeAvg = gradeReduce.mapValues{case(sum,count) => (1.0*sum)/count}
    gradeAvg.foreach(println)

    //intersection function
    val flatgradeAvg = gradeAvg.map(x=> x._1._1 + "," + x._2.toDouble)
    val flatAvg_Stud = Avg_Stud.map(x => x._1 + "," + x._2)
    val commanval = flatgradeAvg.intersection(flatAvg_Stud)
    print("Average score per student_name across all grades is same as average score per student_name per grade is : " +commanval.count() +"\n")
  }
}
```

Output for the first point is shown in the below screenshot.

```
18/10/30 08:35:17 INFO ShuffleBlockFetcherIt
18/10/30 08:35:17 INFO ShuffleBlockFetcherIt
(Mark,50.75)
(Andrew,46.333333333333336)
(Mathew,60.5)
(John,47.5)
(Lisa,58.0)
18/10/30 08:35:17 INFO Executor: Finished ta
18/10/30 08:35:17 INFO TaskSetManager: Finis
```

Output for the second point is shown in the below screenshot.

```
18/10/30 08:35:18 INFO ShuffleBlockFetcherI
18/10/30 08:35:18 INFO ShuffleBlockFetcherI
((Lisa,grade-1),24.0)
((Mark,grade-2),17.5)
((Lisa,grade-2),61.0)
((Mathew,grade-3),45.0)
((Andrew,grade-2),77.0)
((Andrew,grade-1),43.666666666666664)
((Lisa,grade-3),86.0)
((John,grade-1),38.666666666666664)
((John,grade-2),74.0)
((Mark,grade-1),84.0)
((Andrew,grade-3),35.0)
((Mathew,grade-2),65.66666666666667)
18/10/30 08:35:18 INFO Executor: Finished t
```

Finally, output for the question is shown in the below screenshot. The common students count is '**0**'

```
18/10/30 08:35:18 INFO DAGScheduler: ResultStage 8 (count at prbm_stmt_3.scala:36) finished in 0.037 s
18/10/30 08:35:18 INFO DAGScheduler: Job 2 finished: count at prbm_stmt_3.scala:36, took 0.230754 s
Average score per student_name across all grades is same as average score per student_name per grade is : 0
18/10/30 08:35:18 INFO SparkContext: Invoking stop() from shutdown hook
```