# Bigdata Assignment 7

## Task 1

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

Code:

```
import org.apache.spark._
import org.apache.spark.SparkContext._
import org.apache.spark.sql._
import org.apache.log4j._
import org.apache.spark.rdd.MapPartitionsRDD

object assignment{

  def main(args: Array[String]){

    Logger.getLogger("org").setLevel(Level.ERROR)

    val spark = SparkSession
      .builder
      .appName("SparkSQL")
      .master("local[*]")
      .config("spark.sql.warehouse.dir", "file:///C:/temp") //
Necessary to work around a Windows bug in Spark 2.0.0; omit if you're
not on Windows.
      .getOrCreate()

      import spark.implicits._

    val fileContent =
spark.sparkContext.textFile("../C:/Users/Abhishek/Desktop/Acadgild/data
sets/assignment_7_dataset.txt")

    println("The number of rows in the document are:
",fileContent.count())
    val text =
spark.sparkContext.textFile("../C:/Users/Abhishek/Desktop/Acadgild/data
sets/assignment_7_dataset.txt")
    val words = text.flatMap(line=>line.split(","))
    val counts = words.map(word=>(word,1)).reduceByKey{case (a,b)=>a+b}
    counts.saveAsTextFile("assignment_7_dataset_output.txt")
    val output =
spark.sparkContext.textFile("../C:/Users/Abhishek/Desktop/Acadgild/data
sets/assignment_7_dataset_output.txt")
    println("The number of words in the file are: ",output.count)


  }
}
```

# Bigdata Assignment 7

**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

**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?

**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
Hint - Use Intersection Property

Code:

```
import org.apache.spark._
import org.apache.spark.SparkContext._
import org.apache.spark.sql._
import org.apache.log4j._

object assignment {

  case class Person(ID:Int, name:String, age:Int, numFriends:Int)

  def mapper(line:String): Person = {
    val fields = line.split(',')

    val person:Person = Person(fields(0).toInt, fields(1),
fields(2).toInt, fields(3).toInt)
    return person
  }

  /** Our main function where the action happens */
  def main(args: Array[String]) {

    // Set the log level to only print errors
    Logger.getLogger("org").setLevel(Level.ERROR)

    // Use new SparkSession interface in Spark 2.0
    val spark = SparkSession
      .builder
      .appName("SparkSQL")
```

```
      .master("local[*]")
      .config("spark.sql.warehouse.dir", "file:///C:/temp") //
Necessary to work around a Windows bug in Spark 2.0.0; omit if you're
not on Windows.
      .getOrCreate()

    // Convert our csv file to a DataSet, using our Person case
    // class to infer the schema.
    import spark.implicits._
    val lines =
spark.sparkContext.textFile("../C:/Users/Abhishek/Downloads/Acadgild/da
tasets/assignment_7_dataset.txt")
    val people = lines.map(mapper).toDS().cache()

    // There are lots of other ways to make a DataFrame.
    // For example, spark.read.json("json file path")
    // or sqlContext.table("Hive table name")

    println("Here is our inferred schema:")
    people.printSchema()

    println("Let's select the name column:")
    people.select("name").show()

    println("Filter out anyone over 21:")
    people.filter(people("age") < 21).show()

    println("Group by age:")
    people.groupBy("age").count().show()

    println("Make everyone 10 years older:")
    people.select(people("name"), people("age") + 10).show()

    spark.stop()
  }
}
```

**Task 3**

Dataset link

https://drive.google.com/open?id=1oWb_IxIzb5PkFgf6P6lwbHXubAMI-HvK

1) What is the distribution of the total number of air-travelers per year
2) What is the total air distance covered by each user per year
3) Which user has travelled the largest distance till date
4) What is the most preferred destination for all users.
5)Which route is generating the most revenue per year
6) What is the total amount spent by every user on air-travel per year
7) Considering age groups of < 20 , 20-35, 35 > ,Which age group is travelling the most every year.

# Bigdata Assignment 7

Code:

```
import org.apache.spark._
import org.apache.spark.SparkContext._
import org.apache.spark.sql._
import org.apache.log4j._

object assignment {

  case class Person(ID:Int, name:String, age:Int, numFriends:Int)

  def mapper(line:String): Person = {
    val fields = line.split(',')

    val person:Person = Person(fields(0).toInt, fields(1),
fields(2).toInt, fields(3).toInt)
    return person
  }

  /** Our main function where the action happens */
  def main(args: Array[String]) {

    // Set the log level to only print errors
    Logger.getLogger("org").setLevel(Level.ERROR)

    // Use new SparkSession interface in Spark 2.0
    val spark = SparkSession
      .builder
      .appName("SparkSQL")
      .master("local[*]")
      .config("spark.sql.warehouse.dir", "file:///C:/temp") //
Necessary to work around a Windows bug in Spark 2.0.0; omit if you're
not on Windows.
      .getOrCreate()

    // Convert our csv file to a DataSet, using our Person case
    // class to infer the schema.
    import spark.implicits._
    val lines =
spark.sparkContext.textFile("../C:/Users/Abhishek/Downloads/Acadgild/da
tasets/assignment_7_dataset.txt")
    val people = lines.map(mapper).toDS().cache()

    // There are lots of other ways to make a DataFrame.
    // For example, spark.read.json("json file path")
    // or sqlContext.table("Hive table name")

    println("Here is our inferred schema:")
    people.printSchema()

    println("Let's select the name column:")
    people.select("name").show()
```

```
    println("Filter out anyone over 21:")
    people.filter(people("age") < 21).show()

    println("Group by age:")
    people.groupBy("age").count().show()

    println("Make everyone 10 years older:")
    people.select(people("name"), people("age") + 10).show()

    spark.stop()
  }
}
```

**Task 4**

Dataset link

https://drive.google.com/open?id=1qIorA_mC6h4bruPtNOX_S44bPw4rb1Sa

Using spark-sql, Find:
1. What are the total number of gold medal winners every year
2. How many silver medals have been won by USA in each sport

Code:

Written along with Task 5 code

**Task 5**

Using udfs on dataframe
1. Change firstname, lastname columns into
Mr.first_two_letters_of_firstname<space>lastname
for example - michael, phelps becomes Mr.mi phelps
2. Add a new column called ranking using udfs on dataframe, where:
gold medalist, with age >= 32 are ranked as pro
gold medalists, with age <= 31 are ranked amateur
silver medalist, with age >= 32 are ranked as expert
silver medalists, with age <= 31 are ranked rookie

Code:

```
import org.apache.spark._
import org.apache.spark.SparkContext._
import org.apache.spark.sql._
import org.apache.log4j._

import org.apache.spark.sql.functions.udf

import org.apache.spark.sql.functions._
import scala.reflect.ClassTag

import org.apache.spark.sql.functions.{lit, struct}
```

# Bigdata Assignment 7

```
object sparkSql2 {

  //Case class to hold Sports Data

  case class Sports_Data (firstname:String, lastname:String, sports:String,
medal_type:String, age:Int, year:Int, country:String)



  def main(args:Array[String]): Unit = {
    // Set the log level to only print errors
    Logger.getLogger("org").setLevel(Level.ERROR)
    //Let us create a spark session object

    //Let us create a spark session object

    //Create a case class globally to be used inside the main method

    val spark = SparkSession

      .builder()

      .master("local")

      .appName("Spark SQL Assignment 20")

      .config("spark.some.config.option", "some-value")

      .getOrCreate()



    println("spark session object is created")



    //Read the Holiday Details from Local file

    val data = spark.sparkContext.textFile("E:/medaldataset.txt")
    import spark.implicits._

    //Remove Header

    val header = data.first()

    //Create Holdiays DF

    val SportsDF = data.filter(row => row != header).map(_.split(","))

      .map(x => Sports_Data(firstname = x(0), lastname = x(1), sports = x(2),
medal_type = x(3), age = x(4).toInt,
          year = x(5).toInt, country = x(6))).toDF()

    //Printing each row of Sports DF

    SportsDF.show()

    //Task 1.1 : What are the total number of gold medal winners every year

    //Need to group on year where medal type if gold
```

# Bigdata Assignment 7

```
    //Approach 1: Using Spark SQL Operations


SportsDF.filter("medal_type='gold'").groupBy("year").count().orderBy("year").s
how()



    //Approach 2: Using SQL Query

    SportsDF.createOrReplaceTempView("Sports_Table")

    spark.sql("Select year,count(year) as Winners from Sports_Table where
medal_type='gold' group by year order by year").show()


    //Task 1.2 How many silver medals have been won by USA in each sport

    //Need to group on sports where country is USA and medal_type is silver
    //Approach 1 : Using Spark SQL operations

    SportsDF.filter("country='USA' and
medal_type='silver'").groupBy("sports").count().show()



    //Approach 2: Using SQL Query

    spark.sql("Select sports,count(sports) as Winners from Sports_Table where
medal_type='silver' and country='USA' group by sports").show()

     //Task 2.1 :Using udfs on dataframe

    //1. Change firstname, lastname columns into

    //Mr.first_two_letters_of_firstname<space>lastname

    //for example - michael, phelps becomes Mr.mi phelps

    //write a basic function in scala

    def Name=(fname: String, lname: String)=>{

      var newName:String=null

      if (fname != null && lname != null) {

        newName="Mr.".concat(fname.substring(0, 2)).concat(" ")concat(lname)

      }

      newName

    }

    //first we have to create a UDF which returns the output as mentioned in
above use case
```

# Bigdata Assignment 7

```
    //Writing the UDF

    val Change_Name = udf(Name(_:String,_:String))


    //Approach 1 : For calling the Custom user define function without
registering

    SportsDF.withColumn("Name", Change_Name($"firstname", $"lastname")).show()


    //Approach 2: By registering the function

    spark.sqlContext.udf.register("Name", Name)


    spark.sql("Select Name(firstname,lastname) as changed_Name,
sports,medal_type,age,year,country from Sports_Table").show()

    //Task 2.2 2. Add a new column called ranking using udfs on dataframe,
where :

    //gold medalist, with age >= 32 are ranked as pro

    //gold medalists, with age <= 31 are ranked amateur

    //silver medalist, with age >= 32 are ranked as expert

    //silver medalists, with age <= 31 are ranked rookie


    //Write basic scala function for the required use case

    def ranking_recived =(medal_type:String,age:Int)=> {

      if(medal_type.equalsIgnoreCase("gold") && age>=32) "pro"

      else if(medal_type.equalsIgnoreCase("gold") && age <=31) "amateur"

      else if(medal_type.equalsIgnoreCase("silver") && age >= 32) "amateur"

      else if(medal_type.equalsIgnoreCase("silver") && age <= 31) "amateur"

      else ""

    }


    val Rankings = udf(ranking_recived(_:String,_:Int))


    //Approach 1: Without Registering the UDF and calling with Spark SQL
Operatios

    SportsDF.withColumn("Ranking",Rankings($"medal_type",$"age")).show()
```

# Bigdata Assignment 7

```
    //Approach 2:By Registering the function

    spark.sqlContext.udf.register("Rankings",ranking_recived)

    spark.sql("Select Rankings(medal_type,age) as changed_Name,
sports,medal_type,age,year,country from Sports_Table").show()


}
 }
```