Task 1:

Given 2 Datasets employee.json and dept.json

We need to calculate the count of employees against each department. Use Structured API’s.

Sample output

|  |
| --- |
|  |

Solution:

Step1: Read both files and create 2 separate dataframes.

|  |
| --- |
| **object** sparkassignment\_q1 **extends** App {  Logger.getLogger("org").setLevel(Level.ERROR)  **val** sparkconf =**new** SparkConf()  sparkconf.set("spark.app.name","My\_Application")  sparkconf.set("spark.master","local[2]")    **val** spark = SparkSession.builder()  .config(sparkconf)  .getOrCreate()    **val** df1 = spark.read  .format("JSON")  .option("path","F:/TrendyTech\_Bigdata/12\_week\_spark\_structured\_API2/dept.json")  .load    **val** df2 = spark.read  .format("JSON")  .option("path","F:/TrendyTech\_Bigdata/12\_week\_spark\_structured\_API2/employee.json")  .load    df1.show()  df2.show() |

|  |
| --- |
|  |

Step 2: Renaming “Deptid” column in one of the column and performing broadcast join.

|  |
| --- |
| **val** df3 =df2.withColumnRenamed("deptid","emp\_deptid")    **val** joincondition=df1.col("deptid")===df3.col("emp\_deptid")  **val** dfcombinedInner= df1.join(broadcast(df3),joincondition,"inner")    dfcombinedInner.show() |

|  |
| --- |
|  |

Step 3: Grouping by and counting the employees per department.

|  |
| --- |
| **val** dfoutput= dfcombinedInner  .groupBy("deptName","deptid")  .agg(count("deptid").as("empcount"))    dfoutput.select("deptName","deptid","empcount").show() |

|  |
| --- |
|  |

Task 2 : Two Datasets movies.dat and ratings.dat has been given find the top movies(total views >1000 and avg. rating >=4.5). Use spark dataframe and broadcast join for processing.

Step 1: Creating spark session and loading RDD and converting it to dataframe.

|  |
| --- |
| **object** sparkassignment\_q2 **extends** App {    Logger.getLogger("org").setLevel(Level.ERROR)  **val** sparkconf =**new** SparkConf()  sparkconf.set("spark.app.name","My\_Application")  sparkconf.set("spark.master","local[2]")    **val** spark = SparkSession.builder()  .config(sparkconf)  .getOrCreate()  //=============================================================================================  **case** **class** **ratings\_Schema**(user\_id:Int,movie\_id:Int,rating:Int,unixtime:*String*)    **def** mapper(line:*String*):**ratings\_Schema**={  **val** row =line.split("::")  **var** output:**ratings\_Schema** = **ratings\_Schema**(row(0).toInt,row(1).toInt,row(2).toInt,row(3))  **return** output  }  **val** rdd1 = spark.sparkContext.textFile("F:/TrendyTech\_Bigdata/11\_week\_Spark\_structured\_API/Practical\_Data/ratings.dat")  **val** rdd2 =rdd1.map(mapper)  **import** spark.implicits.\_  **val** df1= rdd2.toDF()  df1.show(5) |

|  |
| --- |
|  |

Step 2: Calculating the average rating and total number of users for each movie.

|  |
| --- |
| **val** df11=df1.groupBy(col("movie\_id"))  .agg(count("rating").as("user\_views"),avg("rating").as("Avg\_Rating"))  // df1.printSchema() |

Step 3: loading movies file and converting RDD to dataframe.

|  |
| --- |
| **case** **class** **movies\_Schema**(movie\_id:Int,movie\_name:*String*,categories:*String*)    **def** mapper2(line:*String*):**movies\_Schema**={  **val** row =line.split("::")  **var** output:**movies\_Schema** = **movies\_Schema**(row(0).toInt,row(1),row(2))  **return** output  }  **val** rdd11 = spark.sparkContext.textFile("F:/TrendyTech\_Bigdata/11\_week\_Spark\_structured\_API/Practical\_Data/movies.dat")  **val** rdd22 =rdd11.map(mapper2)  **val** df2= rdd22.toDF()  df2.show(5) |
|  |

Step 4: Performing broadcast join

|  |
| --- |
| **val** df3 =df2.withColumnRenamed("movie\_id","movie\_number")    **val** joincondition=df11.col("movie\_id")===df3.col("movie\_number")  **val** df4= df11.join(broadcast(df3),joincondition,"inner").cache()  //df4.show() |

Step 5: Filter conditions and final answer

|  |
| --- |
| **val** df5 = df4.select("movie\_name","Avg\_Rating","user\_views")  .filter("user\_views>1000 and Avg\_Rating >= 4.5" )  .sort(col("Avg\_Rating").desc)  df5.show(**false**) |

|  |
| --- |
|  |

Task 3:

File A is a text file of size 1.2 GB in HDFS at location /loc/x. It contains match by match statistics of runs scored by all the batsman in the history of cricket. File B is a text file of size 1.2 MB present in local dir /loc/y. It contains list of batsmen playing in cricket world cup 2019.

Find the batsman participating in 2019 who has the best average of scoring runs in his career. Solve using Data frames or Datasets.

Step 1: Creating a spark session

|  |
| --- |
| **object** sparkassignment\_q3 **extends** App {    Logger.getLogger("org").setLevel(Level.ERROR)    **val** sparkconf =**new** SparkConf()  sparkconf.set("spark.app.name","My\_Application")  sparkconf.set("spark.master","local[2]")    **val** spark = SparkSession.builder()  .config(sparkconf)  .getOrCreate() |

Step 2: Loading File A and File B in RDD and modifying RDD to Data frame using case class.

|  |
| --- |
| **case** **class** **FileA\_Schema**(MatchNumber:Int, Batsman:*String*, Team:*String*, RunsScored:Int, StrikeRate:Float)    **def** mapper(line:*String*):**FileA\_Schema**={  **val** row =line.split(" ")  **var** output:**FileA\_Schema** = **FileA\_Schema**(row(0).toInt,row(1),row(2),row(3).toInt,row(4).toFloat)  **return** output  }  **val** rdd1 = spark.sparkContext.textFile("F:/TrendyTech\_Bigdata/12\_week\_Spark\_structured\_API2/File\_A.txt")  **val** rdd2 =rdd1.map(mapper)  **import** spark.implicits.\_  **val** df1= rdd2.toDF()  df1.show(5) |
|  |

|  |
| --- |
| **case** **class** **FileB\_Schema**( Batsman:*String*,Team:*String*)    **def** mapper1(line:*String*):**FileB\_Schema**={  **val** rows =line.split(" ")  **var** output:**FileB\_Schema** = **FileB\_Schema**(rows(0),rows(1))  **return** output  }  **val** rdd11 = spark.sparkContext.textFile("F:/TrendyTech\_Bigdata/12\_week\_Spark\_structured\_API2/File\_B.txt")  **val** rdd22 =rdd11.map(mapper1)  **import** spark.implicits.\_  **val** df11= rdd22.toDF()  df11.show(5) |

|  |
| --- |
|  |

Step 3: Performing broadcast join

|  |
| --- |
| **val** df3 =df11.withColumnRenamed("Batsman","Batsman\_2019")    **val** joincondition=df1.col("Batsman")===df3.col("Batsman\_2019")  **val** df4= df1.join(broadcast(df3),joincondition,"inner").cache()    df4.show() |

|  |
| --- |
|  |

Step 4: Performing groupBy() operation and printing final answer.

|  |
| --- |
| df4.groupBy(col("Batsman"))  .agg(avg("RunsScored").as("AvgScore"))  .sort(col("AvgScore").desc).show() |

|  |
| --- |
|  |