**Important Instructions:**

* Please read the document thoroughly before you code.
* Please do not change the Business Requirements.

**Coverage:**

1. HDFS
2. Map Reduce
3. HIVE
4. PIG Script

For all the hands-on, please consider the following,

**UNIX User name: training**

**Home Directory: /home/training**

**SPARK**

**Requirement 1:**

We have the fruits details available in a CSV file named fruits.txt with column names in the 1st line. Write a Scala program using Spark to display the content of the file skipping the column headers.



**import** org.apache.spark.SparkContext

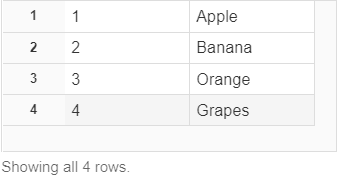
**import** org.apache.spark.SparkConf

**val** sc = SparkContext.getOrCreate()

**val** dataFrame = spark.read.format("csv").option("header","true").option("inferSchema", "true").load("/FileStore/tables/fruits.text")

**print**(dataFrame)

dataFrame.collect()

****

**Requirement 2:**

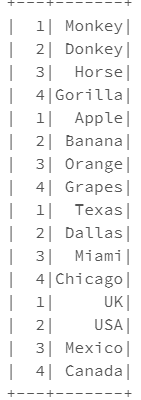
We have the four CSV files named fruits.txt, animal.txt, city.txt and country.txt in a directory named d:\demo\spark, with column names appearing as the 1st line. Write a Scala program using Spark to read all the files in the directory and display the content of the files skipping the column headers.

**val** dF = spark.read.format("csv").option("header","true").option("inferSchema", "true").load("/FileStore/tables/animal-1.text","/FileStore/tables/country-1.text","/FileStore/tables/fruits-2.text","/FileStore/tables/city-1.text")

dF.collect()

dF.show()

****

**Requirement 3:**

We need to count the number of words that starts with the letter ‘A’. To achieve this, we are supposed to write a Scala program using Spark. You are supposed to use RDD Transformation functions to implement this. Use the following Transformations functions -flatMap(), map(), filter() and reduceByKey() to achieve this.

File: Names.txt

Sample Data

Arpit

Sowmya

Jigyasa

Thiru

Mahesh

Arun

Saurav

Lalith

Keerthana

Nikita

Aiswarya

Nency

Kalyan

**val** data=sc.textFile("/FileStore/tables/Names-1.text")

data.collect()

**val** splitdata = data.flatMap(line => line.split(" "));

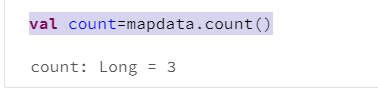
**val** output = splitdata.filter{word=> word.startsWith("A")}

output.collect()

**val** mapdata = output.map(word => (word,1));

mapdata.collect()

**val** count=mapdata.count()



**Requirement 4:**

We have the order details available in orders.txt. The requirement is to count the number of orders placed by the customer. You are supposed to write a Scala program using Spark to display the number of orders placed by each customer.



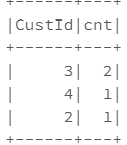
**val** order=sc.textFile("/FileStore/tables/orders.text")

order.collect()

**val** order = spark.read.format("csv").option("header","true").option("inferSchema", "true").load("/FileStore/tables/orders.text")

order.registerTempTable("order")

sqlContext.sql("SELECT CustId, COUNT(\*) AS cnt FROM order GROUP BY custId").show()



**Requirement 5:**

Assume you are getting a two-letter state code in a file named data.txt and the requirement is to transform it to full state name, (for example AP to ANDRAPRADESH, TN to TAMILNADU etc.) by doing a lookup to reference mapping.

You need to use the broadcast variable to cache this lookup info (which is available in state.txt) on each machine and tasks use this cached info while executing the transformations.

**val** states = Map(("BR","BIHAR"),("MH","MAHARASHTRA"),("TN","TAMILNADU"),("AP","ANDRAPRADESH"),("KL","KERALA"))

**val** broadcastStates = spark.sparkContext.broadcast(states)

**val** data = Seq(

("Amit","Mishra","BR"),

("Nitin","Kulkarni","MH"),

("Ram","Kumar","TN"),

("Kesav","Prasad","AP"),

("Biju","Joseph","KL"),

("Ravi","Teja","AP")

)

**val** rdd = spark.sparkContext.parallelize(data)

**val** rdd2 = rdd.map(f=>{

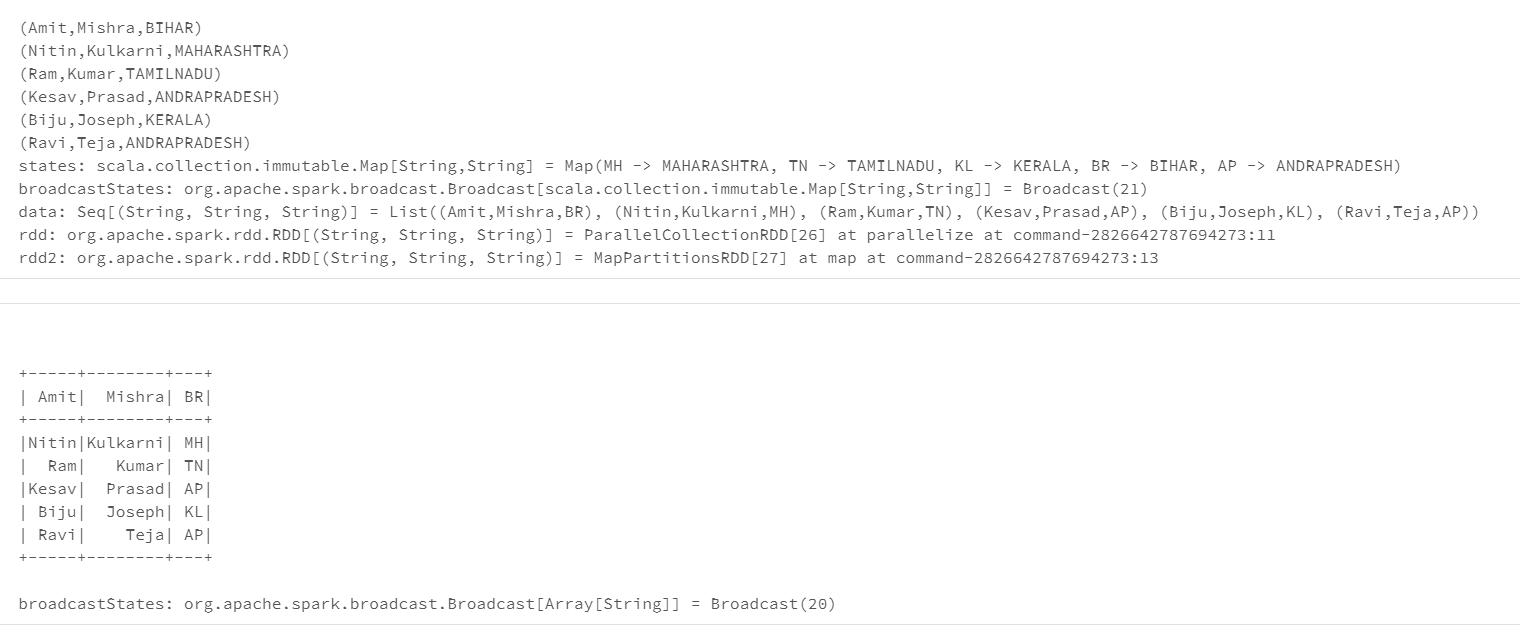
**val** state = f.\_3

**val** fullState = broadcastStates.value.get(state).get

(f.\_1,f.\_2,fullState)

})

**println**(rdd2.collect().mkString("\n"))



**Requirement 6:**

Introduce an array with few values. Write a Scala program to perform the total value of all elements of the array. You need to implement this problem by using a Named Accumulator.

**val** data=Array(1, 2, 3, 4)

**val** accum = sc.longAccumulator("My Accumulator")

sc.parallelize(data).foreach(x => accum.add(x))

accum.value



**Requirement 7:**

We have the population of few cities available in a sequence as follows,

Seq (("Chennai", "20000000"), ("Mumbai", "50000000"), ("Delhi", "40000000"))

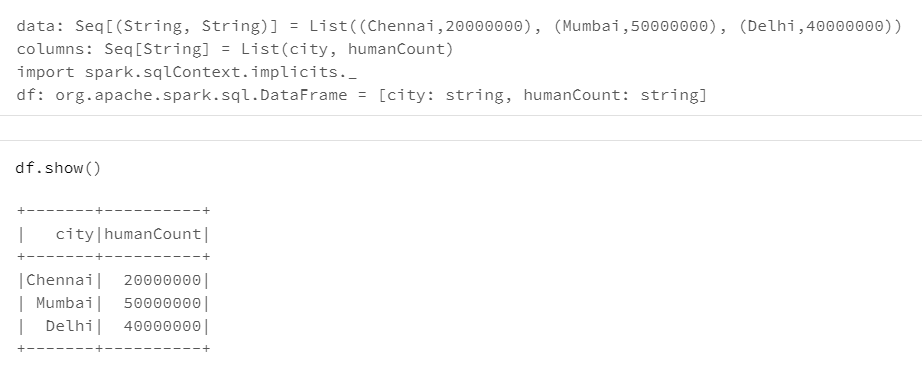
Write a Scala program using Spark to convert the above sequence into a Data Frame with two columns (City and humanCount).

**val** data =Seq (("Chennai", "20000000"), ("Mumbai", "50000000"), ("Delhi", "40000000"))

**val** columns = Seq("city","humanCount")

**import** spark.sqlContext.implicits.**\_**

**val** df = data.toDF(columns:**\_**\*)

****

**Requirement 8:**

We have the skillset of the candidates available in a sequence as follows,

Seq(

Row("Amit,,Mishra",List("Java","Scala","C++"),"UP"),

Row("Prabhu,Ram,",List("Spark","Java","C++"),"TN"),

Row("Ramesh,Kumar",List("CSharp","VB"),"TN")

)

You need to write a Scala program using Spark by converting the array of strings (skillset) into string column in a Dataframe.

**import** org.apache.spark.sql.types.{StringType, StructField, StructType, ArrayType}

**import** org.apache.spark.sql.Row

**import** scala.collection.JavaConversions.**\_**

**val** schema = StructType( Array(

StructField("Name", StringType,true),

StructField("Skillset", ArrayType(StringType),true),

StructField("State", StringType,true),

))

**val** data =Seq(

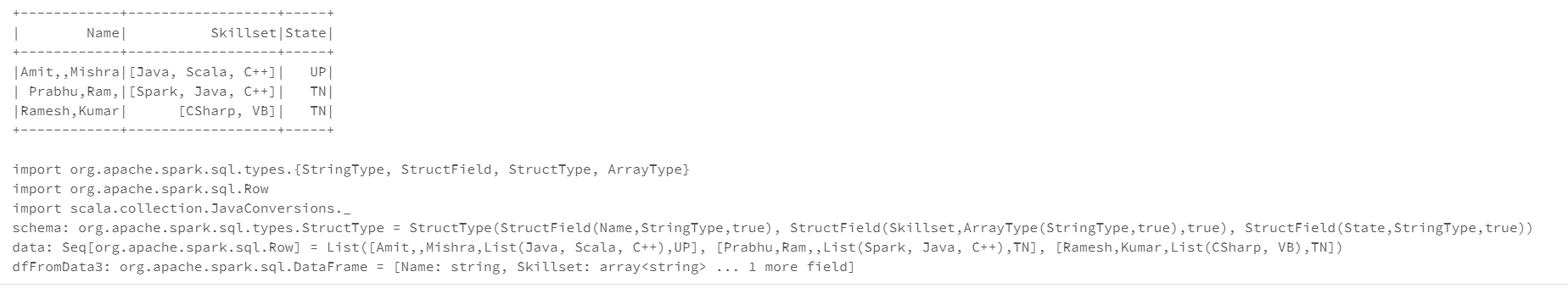
Row("Amit,,Mishra",List("Java","Scala","C++"),"UP"),

Row("Prabhu,Ram,",List("Spark","Java","C++"),"TN"),

Row("Ramesh,Kumar",List("CSharp","VB"),"TN"))

**var** dfFromData3 = spark.createDataFrame(spark.sparkContext.parallelize(data),schema)

dfFromData3.show()



**Requirement 9:**

Seq(

ROW(7369,"SMITH","CLERK",800,20)

ROW(7499,"ALLEN","SALESMAN",1600,30)

ROW(7521,"WARD","SALESMAN",1250,30)

ROW(7566,"JONES","MANAGER",2975,20)

ROW(7654,"MARTIN","SALESMAN",1400,30)

ROW(7698,"BLAKE",MANAGER",2850,30)

)

You need to write a Scala program using Spark to perform

1. Create a Data Frame from the above Sequence with the following column list
   1. EMPNO
   2. ENAME
   3. JOB
   4. SALARY
   5. DEPTNO
2. Update the salary column by adding $100
3. Create a new column named “Bonus” which is 20% of the salary

**import** org.apache.spark.sql.types.{StringType, StructField, StructType, ArrayType,IntegerType}

**import** org.apache.spark.sql.Row

**import** scala.collection.JavaConversions.**\_**

**val** schema = StructType( Array(

StructField("Empno", IntegerType,true),

StructField("Ename",StringType,true),

StructField("Job", StringType,true),

StructField("Sal", IntegerType,true),

StructField("Deptno", IntegerType,true),

))

**val** data =Seq(

Row(7369,"SMITH","CLERK",800,20),

Row(7499,"ALLEN","SALESMAN",1600,30),

Row(7521,"WARD","SALESMAN",1250,30),

Row(7566,"JONES","MANAGER",2975,20),

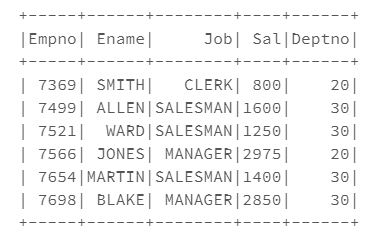
Row(7654,"MARTIN","SALESMAN",1400,30),

Row(7698,"BLAKE","MANAGER",2850,30)

)

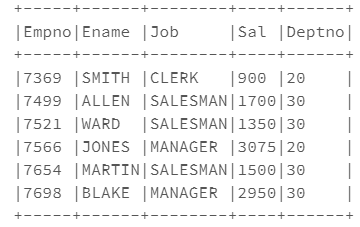
**var** dfFromData = spark.createDataFrame(spark.sparkContext.parallelize(data),schema)

dfFromData.show()

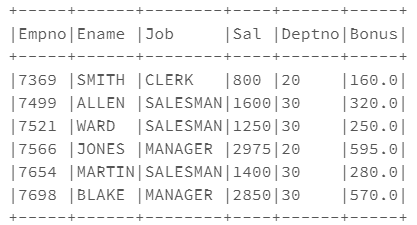


dfFromData.createOrReplaceTempView("EmpDetails")

dfFromData.withColumn("Sal",dfFromData("Sal")+100).show(false)



dfFromData.withColumn("Bonus",dfFromData("Sal")\*0.20).show(false)



**Requirement 10:**

Seq(

ROW(7369,"SMITH","CLERK",800,”Operations”)

ROW(7499,"ALLEN","SALESMAN",1600,”Marketing”)

ROW(7521,"WARD","SALESMAN",1250, ”Marketing”)

ROW(7566,"JONES","MANAGER",2975, ”Operations”)

ROW(7654,"MARTIN","SALESMAN",1400, ”Marketing”)

ROW(7698,"BLAKE",MANAGER",2850, ”Marketing”)

)

You need to write a Scala program using Spark to perform

1. Create a Data Frame from the above Sequence
2. Display the department wise sum of salary.
3. Display the department wise job wise sum of salary.
4. Display the department wise sum of salary where the sum of salary >= 5000.

**import** org.apache.spark.sql.types.{StringType, StructField, StructType, ArrayType,IntegerType}

**import** org.apache.spark.sql.Row

**import** scala.collection.JavaConversions.**\_**

**val** schema = StructType( Array(

StructField("Empno", IntegerType,true),

StructField("Ename",StringType,true),

StructField("Job", StringType,true),

StructField("Sal", IntegerType,true),

StructField("Dept", StringType,true),

))

**val** data =Seq(

Row(7369,"SMITH","CLERK",800,"Operations"),

Row(7499,"ALLEN","SALESMAN",1600,"Marketing"),

Row(7521,"WARD","SALESMAN",1250, "Marketing"),

Row(7566,"JONES","MANAGER",2975, "Operations"),

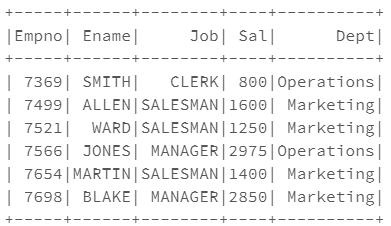
Row(7654,"MARTIN","SALESMAN",1400, "Marketing"),

Row(7698,"BLAKE","MANAGER",2850, "Marketing"),

)

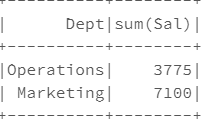
**var** dfFromData = spark.createDataFrame(spark.sparkContext.parallelize(data),schema)

dfFromData.show()

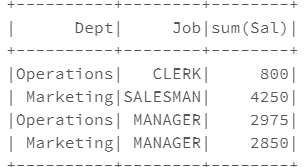


dfFromData.createOrReplaceTempView("EmpDept")

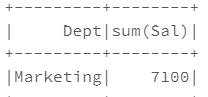
spark.sql("Select Dept,sum(Sal) from EmpDept group by Dept").show()



spark.sql("Select Dept,Job,sum(Sal) from EmpDept group by Dept,Job").show()



spark.sql("Select Dept,sum(Sal) from EmpDept group by Dept having(sum(sal)>=5000)").show()



**Requirement 11:**

Seq(

ROW(7369,"SMITH","CLERK", 800, 20)

ROW(7499,"ALLEN","SALESMAN", 1600, 30)

ROW(7521,"WARD","SALESMAN",1250,30)

ROW(7566,"JONES","MANAGER",2975,20)

ROW(7654,"MARTIN","SALESMAN",1400,30)

ROW(7698,"BLAKE",MANAGER",2850,30)

)

Seq(

ROW(10,”ACCOUNTING”,”NEWYORK”),

ROW(20,”RESEARCH”,”DALLAS”),

ROW(30,”SALES”,”CHICAGO”),

ROW(40,”OPERATIONS”,”BOSTON”)

)

You need to write a Scala program using Spark to perform

1. Combine EMP and DEPT data frames based on the Key column DepdId
2. Combine EMP and DEPT data frames based on the Key column DepdId but should include all the employee details
3. Combine EMP and DEPT data frames based on the Key column DepdId but should include all the dept details

**import** org.apache.spark.sql.types.{StringType, StructField, StructType, ArrayType,IntegerType}

**import** org.apache.spark.sql.Row

**import** scala.collection.JavaConversions.**\_**

**val** schema = StructType( Array(

StructField("Empno", IntegerType,true),

StructField("Ename",StringType,true),

StructField("Job", StringType,true),

StructField("Sal", IntegerType,true),

StructField("DeptId", IntegerType,true),

))

**val** data =Seq(

Row(7369,"SMITH","CLERK",800,20),

Row(7499,"ALLEN","SALESMAN",1600,30),

Row(7521,"WARD","SALESMAN",1250,30),

Row(7566,"JONES","MANAGER",2975,20),

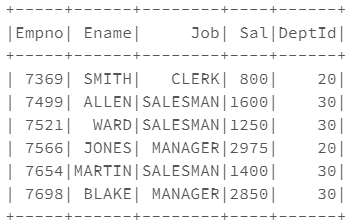
Row(7654,"MARTIN","SALESMAN",1400,30),

Row(7698,"BLAKE","MANAGER",2850,30)

)

**var** dfFromData1 = spark.createDataFrame(spark.sparkContext.parallelize(data),schema)

dfFromData1.show()



**import** org.apache.spark.sql.types.{StringType, StructField, StructType, ArrayType,IntegerType}

**import** org.apache.spark.sql.Row

**import** scala.collection.JavaConversions.**\_**

**val** schema= StructType( Array(

StructField("DeptId", IntegerType,true),

StructField("Dept",StringType,true),

StructField("Loc", StringType,true),

))

**val** data =Seq(

Row(10,"ACCOUNTING","NEWYORK"),

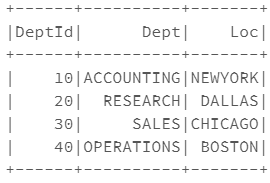
Row(20,"RESEARCH","DALLAS"),

Row(30,"SALES","CHICAGO"),

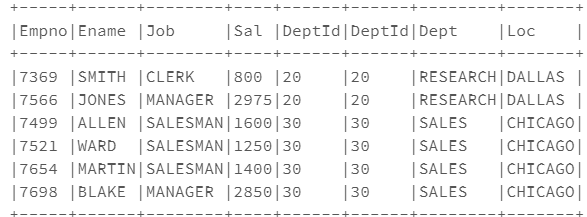
Row(40,"OPERATIONS","BOSTON"))

**var** dfFromData2 = spark.createDataFrame(spark.sparkContext.parallelize(data),schema)

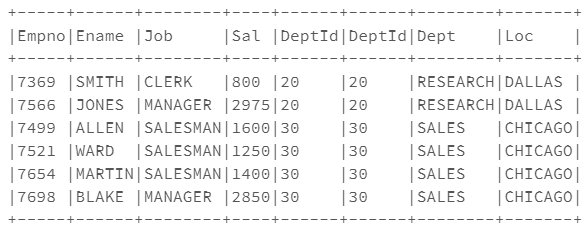
dfFromData2.show()



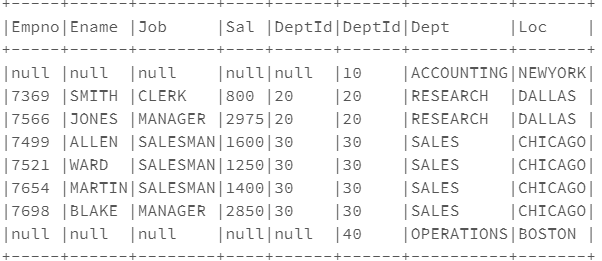
dfFromData1.join(dfFromData2,dfFromData1("DeptId") === dfFromData2("DeptId"),"inner").show(false)



dfFromData1.join(dfFromData2,dfFromData1("DeptId") === dfFromData2("DeptId"),"leftouter").show(false)



dfFromData1.join(dfFromData2,dfFromData1("DeptId") === dfFromData2("DeptId"),"rightouter").show(false)



**Requirement 12:**

Seq(

ROW(7369,"SMITH","CLERK",800,20)

ROW(7499,"ALLEN","SALESMAN",1600,30)

ROW(7521,"WARD","SALESMAN",1250,30)

ROW(7566,"JONES","MANAGER",2975,20)

ROW(7654,"MARTIN","SALESMAN",1400,30)

ROW(7698,"BLAKE",MANAGER",2850,30)

)

You need to write a Scala program using Spark to perform

1. Create a Data Frame from the above Sequence
2. Assign rank no for each employee based on the salary for every department using Windowing function.

**import** org.apache.spark.sql.types.{StringType, StructField, StructType, ArrayType,IntegerType}

**import** org.apache.spark.sql.Row

**import** scala.collection.JavaConversions.**\_**

**val** schema = StructType( Array(

StructField("Empno", IntegerType,true),

StructField("Ename",StringType,true),

StructField("Job", StringType,true),

StructField("Sal", IntegerType,true),

StructField("DeptId", IntegerType,true),

))

**val** data =Seq(

Row(7369,"SMITH","CLERK",800,20),

Row(7499,"ALLEN","SALESMAN",1600,30),

Row(7521,"WARD","SALESMAN",1250,30),

Row(7566,"JONES","MANAGER",2975,20),

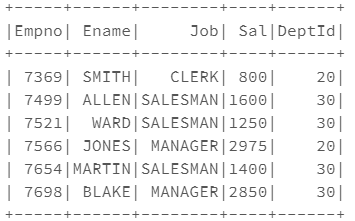
Row(7654,"MARTIN","SALESMAN",1400,30),

Row(7698,"BLAKE","MANAGER",2850,30)

)

**var** dfFromData1 = spark.createDataFrame(spark.sparkContext.parallelize(data),schema)

dfFromData1.show()



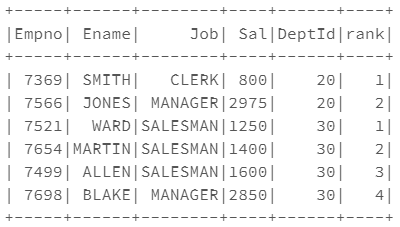
**import** org.apache.spark.sql.functions.**\_**

**import** org.apache.spark.sql.expressions.Window

//row\_number

**val** windowSpec = Window.partitionBy("DeptId").orderBy("Sal")

dfFromData1.withColumn("rank",row\_number.over(windowSpec)).show()



**Requirement 13:**

Seq(

ROW(7369,"SMITH","CLERK",800,20)

ROW(7499,"ALLEN","SALESMAN",1600,30)

ROW(7521,"WARD","SALESMAN",1250,30)

ROW(7566,"JONES","MANAGER",2975,20)

ROW(7654,"MARTIN","SALESMAN",1400,30)

ROW(7698,"BLAKE",MANAGER",2850,30)

)

You need to write a Scala program using Spark to perform

1. Create a Data Frame from the above Sequence
2. Display the Emp name and his salary along with the previous employee’s salary inside every department.
3. Display the Emp name and his salary along with the next employee’s salary inside every department.

**import** org.apache.spark.sql.types.{StringType, StructField, StructType, ArrayType,IntegerType}

**import** org.apache.spark.sql.Row

**import** scala.collection.JavaConversions.**\_**

**val** schema = StructType( Array(

StructField("Empno", IntegerType,true),

StructField("Ename",StringType,true),

StructField("Job", StringType,true),

StructField("Sal", IntegerType,true),

StructField("DeptId", IntegerType,true),

))

**val** data =Seq(

Row(7369,"SMITH","CLERK",800,20),

Row(7499,"ALLEN","SALESMAN",1600,30),

Row(7521,"WARD","SALESMAN",1250,30),

Row(7566,"JONES","MANAGER",2975,20),

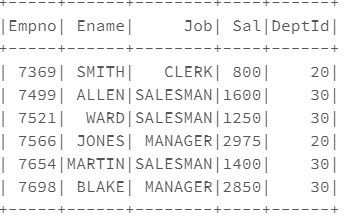
Row(7654,"MARTIN","SALESMAN",1400,30),

Row(7698,"BLAKE","MANAGER",2850,30)

)

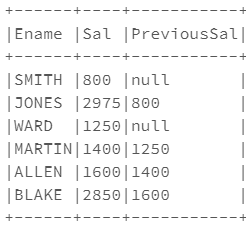
**var** dfFromData1 = spark.createDataFrame(spark.sparkContext.parallelize(data),schema)

dfFromData1.show()

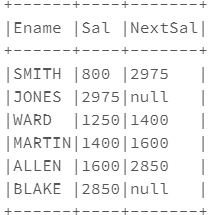


dfFromData1.createOrReplaceTempView("EmpDetails")

spark.sql("SELECT Ename,Sal,LAG(sal) OVER (PARTITION BY DeptId Order BY Sal) As PreviousSal FROM EmpDetails").show(false)



spark.sql("SELECT Ename,Sal,LEAD(sal) OVER (PARTITION BY DeptId Order BY Sal) As NextSal FROM EmpDetails").show(false)



**Requirement 14:**

We have the emp data available in a Sequence as Follows

Seq(

ROW(7369,"SMITH","CLERK",800,20)

ROW(7499,"ALLEN","SALESMAN",1600,30)

ROW(7521,"WARD","SALESMAN",1250,30)

ROW(7566,"JONES","MANAGER",2975,20)

ROW(7654,"MARTIN","SALESMAN",1400,30)

ROW(7698,"BLAKE",MANAGER",2850,30)

)

We have a HBase table with the following structure

Key Column Family : Person

empId Ename, Job, Sal, Deptno

You need to write a Scala program using Spark to perform

1. Create a Data Frame from the above Sequence
2. Store the data from the Data From into the HBase table.

Catalog for HBase Table EMP

def catalog =

s"""{

|"table":{"namespace":"default", "name":"emp"},

|"rowkey":"empId",

|"columns":{

|"key":{"cf":"rowkey", "col":"empId", "type":"int"},

|"eName":{"cf":"person", "col":"eName", "type":"string"},

|"job":{"cf":"person", "col":"job", "type":"string"},

|"sal":{"cf":"person", "col":"sal", "type":"string"},

|"deptno":{"cf":"person", "col":"deptno", "type":"string"},

|}

|}""".stripMargin