1.

i) Upload the data using RDD.

uploadedRddData = spark.sparkContext.textFile("RDDBasicsTextFile.txt") OR

uploadedRddData = sc.textFile("RDDBasicsTextFile.txt")

uploadedRddDataWithoutHeader=uploadedRddData.zipWithIndex().filter( lambda x:x[1]>0).map(lambda x:x[0]).collect()

uploadedRddDataWithoutHeader.collect()

**Note-** Avoid using collect() , as the data is too small it’s fine to use here to see the results .

ii) Sort given data by priority

SortWithPriorityRdd= uploadedRddDataWithoutHeader.sortBy(lambda x:x[0])

SortWithPriorityRdd.collect()

2.

i) Place Low and high priority data into 2 seperate files.

uploadedRddDataWithoutHeader.map(lambda x:x.split(",")).filter(lambda x:x[0]=='Low').saveAsTextFile("LowPriorityData")

uploadedRddDataWithoutHeader.map(lambda x:x.split(",")).filter(lambda x:x[0]=='High').saveAsTextFile("HighPriorityData")

ii) Place records with less than 20 as qty into another file.

uploadedRddDataWithoutHeader.map(lambda x:x.split(",")).filter(lambda x:int(x[1])<20).coalesce(1).saveAsTextFile("qtylessthaa20Data")

3.

i) Create RDDs for priority & sales, qty & sales .

prioritySaleRdd=uploadedRddDataWithoutHeader.map(lambda x:x.split(",")).map(lambda x:(x[0],x[2]))

prioritySaleRdd.collect()

qtySaleRdd=uploadedRddDataWithoutHeader.map(lambda x:x.split(",")).map(lambda x:(x[1],x[2]))

qtySaleRdd.collect()

ii) Load sum of the sales aggregated by Priority into another file.

uploadedRddDataWithoutHeader.map(lambda x: (x[0],float(x[2]))).reduceByKey(lambda x,y :x+y).saveAsTextFile("salesAggByPrty")

4.

i) Load avg of the sales aggregated by qty into another file.

qtyAndSalesData=uploadedRddDataWithoutHeader.map(lambda x:x.split(",")).map(lambda x: (x[1] ,x[2]))

qtyAndSalesAggrData=qtyAndSalesData.aggregateByKey((0.0,0), lambda x,y : (float(x[0])+float(y) ,int( x[1])+1) ,lambda x,y : (float(x[0]+x[1]),int( y[0]+y[1])))

qtyAndSalesAggrData.map(lambda x:(x[0],x[1][0]/x[1][1])).collect()

ii) Display count of records in low and high priorities .

uploadedRddDataWithoutHeader.map(lambda x:x.split(",")).filter(lambda x:x[0]=='Low').count()

uploadedRddDataWithoutHeader.map(lambda x:x.split(",")).filter(lambda x:x[0]=='High').count()

5.

i) Load above data into an RDD with 4 partitions

uploadedRddDataPart4 = spark.sparkContext.textFile("RDDBasicsTextFile.txt",4)

ii) Display data along with partitions.

uploadedRddDataPart4.getNumPartitions()

uploadedRddDataPart4.glom().collect()

6.

i) Re-partition data to 6 partitions.

uploadedRddDataPart6=uploadedRddDataPart4.repartition(6)

uploadedRddDataPart6.getNumPartitions()

ii) Display whole data from all partitions and load into another text file.

uploadedRddDataPart6.saveAsTextFile("uploadedRddDataPart6")

uploadedRddDataPart6.glom().collect()

7. Consider the following text file.

priority,grade

Low,A

High,B

Null,K

i) Develop a RDD for above data with 2 partitions and display.

RDDBasicsTextFileJoinFile=sc.textFile("RDDBasicsTextFileJoinFile.txt",2)

RDDBasicsTextFileJoinFile.glom().collect()

ii) List out priority,quantity,sales ,grade and save into another file.

rddjoin1 =uploadedRddDataWithoutHeader.map(lambda x:x.split(",")).map(lambda x:(x[0],x))

rddjoin2=RDDBasicsTextFileJoinFileMap.map(lambda x:(x[0],x[1]))

dataWithGrade=rddjoin1.join(rddjoin2).map(lambda x:x[1])

dataWithGrade.saveAsTextFile("dataWithGrade")

8.

i) From above file, separate data by grades and load into 3 different files.

dataWithGrade.filter(lambda x:x[1]=='A').saveAsTextFile("dataWithGradeA")

dataWithGrade.filter(lambda x:x[1]=='B').saveAsTextFile("dataWithGradeB")

dataWithGrade.filter(lambda x:x[1]=='K').saveAsTextFile("dataWithGradeC")

ii) Is it possible to lessen the number of partitions? verify re-partition with above file by decreasing to 1 partition.

Yes , it's possible with coalesce(1) or repartition(1) function .

uploadedRddDataPart6.coalesce(1).glom().collect()