CCA175 Q&As

CCA Spark and Hadoop Developer Exam

Pass Cloudera CCA175 Exam with 100% Guarantee

Free Download Real Questions & Answers **PDF** and **VCE** file from:

https://www.leads4pass.com/cca175.html

100% Passing Guarantee
100% Money Back Assurance

Following Questions and Answers are all new published by Cloudera Official Exam Center

QUESTION 1

Problem Scenario 50 : You have been given below code snippet (calculating an average

score}, with intermediate output.

type ScoreCollector = (Int, Double)

type PersonScores = (String, (Int, Double))

val initialScores = Array(("Fred", 88.0), ("Fred", 95.0), ("Fred", 91.0), ("Wilma", 93.0),

("Wilma", 95.0), ("Wilma", 98.0))

val wilmaAndFredScores = sc.parallelize(initialScores).cache()

val scores = wilmaAndFredScores.combineByKey(createScoreCombiner, scoreCombiner,

scoreMerger)

val averagingFunction = (personScore: PersonScores) => { val (name, (numberScores,

totalScore)) = personScore (name, totalScore / numberScores)

}

val averageScores = scores.collectAsMap(}.map(averagingFunction)

Expected output: averageScores: scala.collection.Map[String,Double] = Map(Fred ->

91.33333333333333, Wilma -> 95.33333333333333)

Define all three required function , which are input for combineByKey method, e.g.

(createScoreCombiner, scoreCombiner, scoreMerger). And help us producing required

results.

Correct Answer: See the explanation for Step by Step Solution and configuration.

Solution :

val createScoreCombiner = (score: Double) => (1, score)

val scoreCombiner = (collector: ScoreCollector, score: Double) => {

val (numberScores. totalScore) = collector (numberScores + 1, totalScore + score)

}

val scoreMerger= (collector-!: ScoreCollector, collector2: ScoreCollector) => { val

(numScoresl. totalScorel) = collector! val (numScores2, tota!Score2) = collector

(numScoresl + numScores2, totalScorel + totalScore2)

}

QUESTION 2

Problem Scenario 37 : ABCTECH.com has done survey on their Exam Products feedback using a web based form. With the following free text field as input in web ui. Name: StringSubscription Date: String Rating : String And servey data has been saved in a file called spark9/feedback.txt Christopher|Jan 11, 2015|5 Kapil|11 Jan, 2015|5 Thomas|6/17/2014|5 John|22-08-2013|5 Mithun|2013|5 Jitendra||5 Write a spark program using regular expression which will filter all the valid dates and save in two separate file (good record and bad record)

Correct Answer: See the explanation for Step by Step Solution and configuration.

Solution :

Step 1 : Create a file first using Hue in hdfs.

Step 2 : Write all valid regular expressions sysntex for checking whether records are having

valid dates or not.

val regl =.....(\d+)\s(\w{3})(,)\s(\d{4}).....r//11 Jan, 2015

val reg2 =.....(\d+)(U)(\d+)(U)(\d{4}).....s II 6/17/2014

val reg3 =.....(\d+)(-)(\d+)(-)(\d{4})""".r//22-08-2013

val reg4 =.....(\w{3})\s(\d+)(,)\s(\d{4}).....s II Jan 11, 2015

Step 3 : Load the file as an RDD.

val feedbackRDD = sc.textFile("spark9/feedback.txt"}

Step 4 : As data are pipe separated , hence split the same. val feedbackSplit =

feedbackRDD.map(line => line.split(\\'|\\'))

Step 5 : Now get the valid records as well as , bad records.

val validRecords = feedbackSplit.filter(x =>

(reg1.pattern.matcher(x(1).trim).matches|reg2.pattern.matcher(x(1).trim).matches|reg3.patt

ern.matcher(x(1).trim).matches | reg4.pattern.matcher(x(1).trim).matches))

val badRecords = feedbackSplit.filter(x =>

!(reg1.pattern.matcher(x(1).trim).matches|reg2.pattern.matcher(x(1).trim).matches|reg3.pat

tern.matcher(x(1).trim).matches | reg4.pattern.matcher(x(1).trim).matches))

Step 6 : Now convert each Array to Strings

val valid =vatidRecords.map(e => (e(0),e(1),e(2)))

val bad =badRecords.map(e => (e(0),e(1),e(2)))

Step 7 : Save the output as a Text file and output must be written in a single tile,

valid.repartition(1).saveAsTextFile("spark9/good.txt")

bad.repartition(1).saveAsTextFile("sparkS7bad.txt")

QUESTION 3

Problem Scenario 96 : Your spark application required extra Java options as below. XX:+PrintGCDetails-XX:+PrintGCTimeStamps Please replace the XXX values correctly ./bin/spark-submit --name "My app" --master local[4] --conf spark.eventLog.enabled=talse -conf XXX hadoopexam.jar

Correct Answer: See the explanation for Step by Step Solution and configuration.

Solution

XXX: Mspark.executoi\extraJavaOptions=-XX:+PrintGCDetails -XX:+PrintGCTimeStamps"

Notes: ./bin/spark-submit \

--class

--master \

--deploy-mode \

-conf = \

other options

\

[application-arguments]

Here, conf is used to pass the Spark related contigs which are required for the application

to run like any specific property(executor memory) or if you want to override the default

property which is set in Spark-default.conf.

QUESTION 4

Problem Scenario 62 : You have been given below code snippet.

val a = sc.parallelize(List("dogM, "tiger", "lion", "cat", "panther", "eagle"), 2)

val b = a.map(x => (x.length, x))

operation1

Write a correct code snippet for operationl which will produce desired output, shown below.

Array[(lnt, String)] = Array((3,xdogx), (5,xtigerx), (4,xlionx), (3,xcatx), (7,xpantherx),

(5,xeaglex))

Correct Answer: See the explanation for Step by Step Solution and configuration.

Solution : b.mapValuesf\\'x" + _ + "x").collect mapValues [Pair] : Takes the values of a RDD that consists of two-component tuples, and applies the provided function to transform each value. Tlien,.it.forms newtwo-componend tuples using the key and the transformed value and stores them in a new RDD.

QUESTION 5

Problem Scenario 39 : You have been given two files spark16/file1.txt 1,9,5 2,7,4 3,8,3 spark16/file2.txt 1,g,h 2,i,j 3,k,l Load these two tiles as Spark RDD and join them to produce the below results (l,((9,5),(g,h))) (2, ((7,4), (i,j))) (3, ((8,3), (k,l))) And write code snippet which will sum the second columns of above joined results (5+4+3).

Correct Answer: See the explanation for Step by Step Solution and configuration.

Solution :

Step 1 : Create tiles in hdfs using Hue.

Step 2 : Create pairRDD for both the files.

val one = sc.textFile("spark16/file1.txt").map{

_.split(",",-1) match {

case Array(a, b, c) => (a, (b, c))

} }

val two = sc.textFHe(Mspark16/file2.txt").map{

_.split(\\'7\-1) match {

case Array(a, b, c) => (a, (b, c))

} }

Step 3 : Join both the RDD. val joined = one.join(two)

Step 4 : Sum second column values.

val sum = joined.map {

case (_, ((_, num2), (_, _))) => num2.tolnt

}.reduce(_ + _)