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 68 : You have given a file as below. spark75/f ile1.txt File contain some text. As given Below spark75/file1.txt Apache Hadoop is an open-source software framework written in Java for distributed storage and distributed processing of very large data sets on computer clusters built from commodity hardware. All the modules in Hadoop are designed with a fundamental assumption that hardware failures are common and should be automatically handled by the framework The core of Apache Hadoop consists of a storage part known as Hadoop Distributed File System (HDFS) and a processing part called MapReduce. Hadoop splits files into large blocks and distributes them across nodes in a cluster. To process data, Hadoop transfers packaged code for nodes to process in parallel based on the data that needs to be processed. his approach takes advantage of data locality nodes manipulating the data they have access to to allow the dataset to be processed faster and more efficiently than it would be in a more conventional supercomputer architecture that relies on a parallel file system where computation and data are distributed via high-speed networking For a slightly more complicated task, lets look into splitting up sentences from our documents into word bigrams. A bigram is pair of successive tokens in some sequence. We will look at building bigrams from the sequences of words in each sentence, and then try to find the most frequently occuring ones. The first problem is that values in each partition of our initial RDD describe lines from the file rather than sentences. Sentences may be split over multiple lines. The glom() RDD method is used to create a single entry for each document containing the list of all lines, we can then join the lines up, then resplit them into sentences using "." as the separator, using flatMap so that every object in our RDD is now a sentence. A bigram is pair of successive tokens in some sequence. Please build bigrams from the sequences of words in each sentence, and then try to find the most frequently occuring ones.

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

Solution :

Step 1 : Create all three tiles in hdfs (We will do using Hue}. However, you can first create

in local filesystem and then upload it to hdfs.

Step 2 : The first problem is that values in each partition of our initial RDD describe lines

from the file rather than sentences. Sentences may be split over multiple lines.

The glom() RDD method is used to create a single entry for each document containing the

list of all lines, we can then join the lines up, then resplit them into sentences using "." as

the separator, using flatMap so that every object in our RDD is now a sentence.

sentences = sc.textFile("spark75/file1.txt") \ .glom() \

map(lambda x: " ".join(x)) \ .flatMap(lambda x: x.spllt("."))

Step 3 : Now we have isolated each sentence we can split it into a list of words and extract

the word bigrams from it. Our new RDD contains tuples

containing the word bigram (itself a tuple containing the first and second word) as the first

value and the number 1 as the second value. bigrams = sentences.map(lambda x:x.split())

\ .flatMap(lambda x: [((x[i],x[i+1]),1)for i in range(0,len(x)-1)])

Step 4 : Finally we can apply the same reduceByKey and sort steps that we used in the

wordcount example, to count up the bigrams and sort them in order of descending

frequency. In reduceByKey the key is not an individual word but a bigram.

freq_bigrams = bigrams.reduceByKey(lambda x,y:x+y)\

map(lambda x:(x[1],x[0])) \

sortByKey(False)

freq_bigrams.take(10)

QUESTION 2

Problem Scenario 52 : You have been given below code snippet.

val b = sc.parallelize(List(1,2,3,4,5,6,7,8,2,4,2,1,1,1,1,1))

Operation_xyz

Write a correct code snippet for Operation_xyz which will produce below output.

scalaxollection.Map[lnt,Long] = Map(5 -> 1, 8 -> 1, 3 -> 1, 6 -> 1, 1 -> S, 2 -> 3, 4 -> 2, 7 ->

1)

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

Solution : b.countByValue countByValue Returns a map that contains all unique values of the RDD and their respective occurrence counts. (Warning: This operation will finally aggregate the information in a single reducer.) Listing Variants def countByValue(): Map[T, Long]

QUESTION 3

Problem Scenario 47 : You have been given below code snippet, with intermediate output.

val z = sc.parallelize(List(1,2,3,4,5,6), 2)

// lets first print out the contents of the RDD with partition labels

def myfunc(index: Int, iter: lterator[(lnt)]): lterator[String] = {

iter.toList.map(x => "[partID:" + index + ", val: " + x + "]").iterator

}

//In each run , output could be different, while solving problem assume belowm output only.

z.mapPartitionsWithlndex(myfunc).collect

res28: Array[String] = Array([partlD:0, val: 1], [partlD:0, val: 2], [partlD:0, val: 3], [partlD:1,

val: 4], [partlD:1, val: S], [partlD:1, val: 6])

Now apply aggregate method on RDD z , with two reduce function , first will select

max value in each partition and second will add all the maximum values from all

partitions.

Initialize the aggregate with value 5. hence expected output will be 16.

Correct Answer: z.aggregate(5)(math.max(_, J, _ + _)

QUESTION 4

Problem Scenario 56 : You have been given below code snippet.

val a = sc.parallelize(l to 100. 3)

operation1

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

Array [Array [I nt]] = Array(Array(1, 2, 3,4, 5, 6, 7, 8, 9,10,11,12,13,14,15,16,17,18,19, 20,

21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33),

Array(34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55,

56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66),

Array(67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88,

89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100))

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

Solution : a.glom.collect glom Assembles an array that contains all elements of the partition and embeds it in an RDD. Each returned array contains the contents of one panition

---

QUESTION 5

Problem Scenario 16 : You have been given following mysql database details as well as other info. user=retail_dba password=cloudera database=retail_db jdbc URL = jdbc:mysql://quickstart:3306/retail_db Please accomplish below assignment.

1.

Create a table in hive as below.

create table departments_hive(department_id int, department_name string);

2.

Now import data from mysql table departments to this hive table. Please make sure that

data should be visible using below hive command, select" from departments_hive

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

Solution :

Step 1 : Create hive table as said.

hive

show tables;

create table departments_hive(department_id int, department_name string);

Step 2 : The important here is, when we create a table without delimiter fields. Then default

delimiter for hive is ^A (\001). Hence, while importing data we have to provide proper

delimiter.

sqoop import \

-connect jdbc:mysql://quickstart:3306/retail_db \

~username=retail_dba \

-password=cloudera \

--table departments \

--hive-home /user/hive/warehouse \

-hive-import \

-hive-overwrite \

--hive-table departments_hive \

--fields-terminated-by \\'\001\\'

Step 3 : Check-the data in directory.

hdfs dfs -Is /user/hive/warehouse/departments_hive

hdfs dfs -cat/user/hive/warehouse/departmentshive/part\\'

Check data in hive table.

Select * from departments_hive;