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.geekcert.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 11 : 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 following.

1.

Import departments table in a directory called departments.

2.

Once import is done, please insert following 5 records in departments mysql table.

Insert into departments(10, physics);

Insert into departments(11, Chemistry);

Insert into departments(12, Maths);

Insert into departments(13, Science);

Insert into departments(14, Engineering);

3.

Now import only new inserted records and append to existring directory . which has been created in first step.

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

Solution :

Step 1 : Clean already imported data. (In real exam, please make sure you dont delete

data generated from previous exercise).

hadoop fs -rm -R departments

Step 2 : Import data in departments directory.

sqoop import \

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

--username=retail_dba \

-password=cloudera \

-table departments \

"target-dir/user/cloudera/departments

Step 3 : Insert the five records in departments table.

mysql -user=retail_dba --password=cloudera retail_db

Insert into departments values(10, "physics"); Insert into departments values(11,

"Chemistry"); Insert into departments values(12, "Maths"); Insert into departments

values(13, "Science"); Insert into departments values(14, "Engineering"); commit;

select\\' from departments;

Step 4 : Get the maximum value of departments from last import, hdfs dfs -cat

/user/cloudera/departments/part* that should be 7

Step 5 : Do the incremental import based on last import and append the results.

sqoop import \

--connect "jdbc:mysql://quickstart.cloudera:330G/retail_db" \

~username=retail_dba \

-password=cloudera \

-table departments \

--target-dir /user/cloudera/departments \

-append \

-check-column "department_id" \

-incremental append \

-last-value 7

Step 6 : Now check the result.

hdfs dfs -cat /user/cloudera/departments/part"

QUESTION 2

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 3

Problem Scenario 26 : You need to implement near real time solutions for collecting information when submitted in file

with below information. You have been given below directory location (if not available than create it) /tmp/nrtcontent. Assume your departments upstream service is continuously committing data in this directory as a new file (not stream of data, because it is near real time solution). As soon as file committed in this directory that needs to be available in hdfs in /tmp/flume location Data

echo "I am preparing for CCA175 from ABCTECH.com" > /tmp/nrtcontent/.he1.txt mv /tmp/nrtcontent/.he1.txt /tmp/nrtcontent/he1.txt After few mins echo "I am preparing for CCA175 from TopTech.com" > /tmp/nrtcontent/.qt1.txt mv /tmp/nrtcontent/.qt1.txt /tmp/nrtcontent/qt1.txt

Write a flume configuration file named flumes.conf and use it to load data in hdfs with following additional properties.

1.

Spool /tmp/nrtcontent

2.

File prefix in hdfs sholuld be events

3.

File suffix should be Jog

4.

If file is not commited and in use than it should have as prefix.

5.

Data should be written as text to hdfs

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

Solution : Step 1 : Create directory mkdir /tmp/nrtcontent Step 2 : Create flume configuration file, with below configuration for source, sink and channel and save it in flume6.conf. agent1 .sources = source1 agent1 .sinks = sink1 agent1.channels = channel1 agent1 .sources.source1.channels = channel1 agent1 .sinks.sink1.channel = channel1 agent1 .sources.source1.type = spooldir agent1 .sources.source1.spoolDir = /tmp/nrtcontent agent1 .sinks.sink1 .type = hdfs agent1 .sinks.sink1.hdfs.path = /tmp/flume agent1.sinks.sink1.hdfs.filePrefix = events agent1.sinks.sink1.hdfs.fileSuffix = .log agent1 .sinks.sink1.hdfs.inUsePrefix = _ agent1 .sinks.sink1.hdfs.fileType = Data Stream Step 4 : Run below command which will use this configuration file and append data in hdfs. Start flume service: flume-ng agent -conf /home/cloudera/flumeconf -conf-file /home/cloudera/flumeconf/flume6.conf --name agent1 Step 5 : Open another terminal and create a file in /tmp/nrtcontent echo "I am preparing for CCA175 from ABCTechm.com" > /tmp/nrtcontent/.he1.txt mv /tmp/nrtcontent/.he1.txt /tmp/nrtcontent/he1.txt After few mins echo "I am preparing for CCA175 from TopTech.com" > /tmp/nrtcontent/.qt1.txt mv /tmp/nrtcontent/.qt1.txt /tmp/nrtcontent/qt1.txt

QUESTION 4

Problem Scenario 51 : You have been given below code snippet.

val a = sc.parallelize(List(1, 2,1, 3), 1)

val b = a.map((_, "b"))

val c = a.map((_, "c"))

Operation_xyz

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

Output:

Array[(Int, (Iterable[String], Iterable[String]))] = Array(

(2,(ArrayBuffer(b),ArrayBuffer(c))),

(3,(ArrayBuffer(b),ArrayBuffer(c))),

(1,(ArrayBuffer(b, b),ArrayBuffer(c, c))))

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

Solution :

b.cogroup(c).collect

cogroup [Pair], groupWith [Pair]

A very powerful set of functions that allow grouping up to 3 key-value RDDs together using

their keys.

Another example

val x = sc.parallelize(List((1, "apple"), (2, "banana"), (3, "orange"), (4, "kiwi")), 2)

val y = sc.parallelize(List((5, "computer"), (1, "laptop"), (1, "desktop"), (4, "iPad")), 2)

x.cogroup(y).collect

Array[(Int, (Iterable[String], Iterable[String]))] = Array(

(4,(ArrayBuffer(kiwi),ArrayBuffer(iPad))),

(2,(ArrayBuffer(banana),ArrayBuffer())),

(3,(ArrayBuffer(orange),ArrayBuffer())),

(1 ,(ArrayBuffer(apple),ArrayBuffer(laptop, desktop))),

(5,{ArrayBuffer(),ArrayBuffer(computer))))

QUESTION 5

Problem Scenario 55 : You have been given below code snippet.

val pairRDDI = sc.parallelize(List(("cat",2), ("cat", 5), ("book", 4),("cat", 12))) val

pairRDD2 = sc.parallelize(List(("cat",2), ("cup", 5), ("mouse", 4),("cat", 12)))

operation1

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

Array[(String, (Option[Int], Option[Int]))] = Array((book,(Some(4},None)),

(mouse,(None,Some(4))), (cup,(None,Some(5))), (cat,(Some(2),Some(2)),

(cat,(Some(2),Some(12))), (cat,(Some(5),Some(2))), (cat,(Some(5),Some(12))),

(cat,(Some(12),Some(2))), (cat,(Some(12),Some(12)))J

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

Solution : pairRDD1.fullOuterJoin(pairRDD2).collect

fullOuterJoin [Pair]

Performs the full outer join between two paired RDDs.

Listing Variants

def fullOuterJoin[W](other: RDD[(K, W)], numPartitions: Int): RDD[(K, (Option[V],

OptionfW]))]

def fullOuterJoin[W](other: RDD[(K, W}]}: RDD[(K, (Option[V], OptionfW]))]

def fullOuterJoin[W](other: RDD[(K, W)], partitioner: Partitioner): RDD[(K, (Option[V],

Option[W]))]