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.pass4itsure.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 63 : You have been given below code snippet.

val a = sc.parallelize(List("dog", "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((4,lion), (3,dogcat), (7,panther), (5,tigereagle))

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

Solution :

b.reduceByKey(_ + _).collect

reduceByKey JPair] : This function provides the well-known reduce functionality in Spark.

Please note that any function f you provide, should be commutative in order to generate

reproducible results.

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.splIt("."))

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 74 : You have been given MySQL DB with following details.

user=retail_dba

password=cloudera

database=retail_db

table=retail_db.orders

table=retail_db.order_items

jdbc URL = jdbc:mysql://quickstart:3306/retail_db

Columns of order table : (orderjd , order_date , ordercustomerid, order status}

Columns of orderjtems table : (order_item_td , order_item_order_id ,

order_item_product_id,

order_item_quantity,order_item_subtotal,order_item_product_price)

Please accomplish following activities.

1.

Copy "retaildb.orders" and "retaildb.orderjtems" table to hdfs in respective directory p89_orders and p89_order_items .

2.

Join these data using orderjd in Spark and Python

3.

Now fetch selected columns from joined data Orderld, Order date and amount collected on this order.

4.

Calculate total order placed for each date, and produced the output sorted by date.

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

Solution: Step 1 : Import Single table . sqoop import --connect jdbc:mysql://quickstart:3306/retail_db -username=retail_dba password=cloudera -table=orders --target-dir=p89_orders - -m1 sqoop import --connect jdbc:mysql://quickstart:3306/retail_db -username=retail_dba password=cloudera -table=order_items ~target-dir=p89_ order items -m 1 Note : Please check you dont have space between before or after \\'=\\' sign. Sqoop uses the MapReduce framework to copy data from RDBMS to hdfs Step 2 : Read the data from one of the partition, created using above command, hadoopfs-cat p89_orders/part-m-00000 hadoop fs -cat p89_order_items/part-m-00000 Step 3 : Load these above two directory as RDD using Spark and Python (Open pyspark terminal and do following). orders = sc.textFile("p89_orders") orderitems = sc.textFile("p89_order_items") Step 4 : Convert RDD into key value as (orderjd as a key and rest of the values as a value) #First value is orderjd ordersKeyValue = orders.map(lambda line: (int(line.split(",")[0]), line)) #Second value as an Orderjd orderltemsKeyValue = orderltems.map(lambda line: (int(line.split(",")[1]), line)) Step 5 : Join both the RDD using orderjd joinedData = orderltemsKeyValue.join(ordersKeyValue) #print the joined data

tor line in joinedData.collect():

print(line)

Format of joinedData as below.

[Orderld, \\'All columns from orderltemsKeyValue\\', \\'All columns from orders Key Value\\']

Step 6 : Now fetch selected values Orderld, Order date and amount collected on this order.

revenuePerOrderPerDay = joinedData.map(lambda row: (row[0](row[1][1].split(",")[1](

f!oat(row[1][0].split(\\'\M}[4]}}}

#printthe result

for line in revenuePerOrderPerDay.collect():

print(line)

Step 7 : Select distinct order ids for each date.

#distinct(date,order_id)

distinctOrdersDate = joinedData.map(lambda row: row[1][1].split(\\'\")[1] + "," +

str(row[0])).distinct()

for line in distinctOrdersDate.collect(): print(line)

Step 8 : Similar to word count, generate (date, 1) record for each row. newLineTuple =

distinctOrdersDate.map(lambda line: (line.split(",")[0], 1))

Step 9 : Do the count for each key(date), to get total order per date. totalOrdersPerDate =

newLineTuple.reduceByKey(lambda a, b: a + b}

#print results

for line in totalOrdersPerDate.collect():

print(line)

step 10 : Sort the results by date sortedData=totalOrdersPerDate.sortByKey().collect()

#print results

for line in sortedData:

print(line)

QUESTION 4

Problem Scenario 67 : You have been given below code snippet.

lines = sc.parallelize([\\'Its fun to have fun,\\',\\'but you have to know how.\\'])

M = lines.map(lambda x: x.replace(\\',7 \\').replace(\\'.\\',\\' \\'J.replaceC-V \\').lower())

r2 = r1.flatMap(lambda x: x.split())

r3 = r2.map(lambda x: (x, 1))

operation1

r5 = r4.map(lambda x:(x[1],x[0]))

r6 = r5.sortByKey(ascending=False)

r6.take(20)

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

[(2, \\'fun\\'), (2, \\'to\\'), (2, \\'have\\'), (1, its\\'), (1, \\'know1), (1, \\'how1), (1, \\'you\\'), (1, \\'but\\')]

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

Solution : r4 = r3.reduceByKey(lambda x,y:x+y)

QUESTION 5

Problem Scenario 2 :

There is a parent organization called "ABC Group Inc", which has two child companies

named Tech Inc and MPTech.

Both companies employee information is given in two separate text file as below. Please do

the following activity for employee details.

Tech Inc.txt 1,Alok,Hyderabad 2,Krish,Hongkong 3,Jyoti,Mumbai 4,Atul,Banglore 5,Ishan,Gurgaon MPTech.txt 6,John,Newyork 7,alp2004,California 8,tellme,Mumbai 9,Gagan21,Pune 10,Mukesh,Chennai

1.

Which command will you use to check all the available command line options on HDFS and How will you get the Help for individual command.

2.

Create a new Empty Directory named Employee using Command line. And also create an empty file named in it Techinc.txt

3.

Load both companies Employee data in Employee directory (How to override existing file in HDFS).

4.

Merge both the Employees data in a Single tile called MergedEmployee.txt, merged tiles should have new line character at the end of each file content.

5.

Upload merged file on HDFS and change the file permission on HDFS merged file, so that owner and group member can read and write, other user can read the file.

6.

Write a command to export the individual file as well as entire directory from HDFS to local file System.

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

Solution :

Step 1 : Check All Available command hdfs dfs Step 2 : Get help on Individual command hdfs dfs -help get Step 3 : Create a directory in HDFS using named Employee and create a Dummy file in it called e.g. Techinc.txt hdfs dfs -mkdir Employee Now create an emplty file in Employee directory using Hue. Step 4 : Create a directory on Local file System and then Create two files, with the given data in problems. Step 5 : Now we have an existing directory with content in it, now using HDFS command line , overrid this existing Employee directory. While copying these files from local file System to HDFS. cd /home/cloudera/Desktop/ hdfs dfs -put -f Employee Step 6 : Check All files in directory copied successfully hdfs dfs -Is Employee Step 7 : Now merge all the files in Employee directory, hdfs dfs -getmerge -nl Employee MergedEmployee.txt Step 8 : Check the content of the file. cat MergedEmployee.txt Step 9 : Copy merged file in Employeed directory from local file ssytem to HDFS. hdfs dfs put MergedEmployee.txt Employee/ Step 10 : Check file copied or not. hdfs dfs -Is Employee Step 11 : Change the permission of the merged file on HDFS hdfs dfs -chmpd 664 Employee/MergedEmployee.txt Step 12 : Get the file from HDFS to local file system, hdfs dfs -get Employee Employee_hdfs