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

- **Instant Download** After Purchase
- **100% Money Back** Guarantee
- **365 Days** Free Update
- **800,000+** Satisfied Customers

QUESTION 1

Problem Scenario 30 : You have been given three csv files in hdfs as below.

EmployeeName.csv with the field (id, name)

EmployeeManager.csv (id, manager Name)

EmployeeSalary.csv (id, Salary)

Using Spark and its API you have to generate a joined output as below and save as a text

tile (Separated by comma) for final distribution and output must be sorted by id.

ld,name,salary,managerName

EmployeeManager.csv

E01,Vishnu

E02,Satyam

E03,Shiv

E04,Sundar

E05,John

E06,Pallavi

E07,Tanvir

E08,Shekhar

E09,Vinod

E10,Jitendra

EmployeeName.csv

E01,Lokesh

E02,Bhupesh

E03,Amit

E04,Ratan E05,Dinesh E06,Pavan E07,Tejas E08,Sheela E09,Kumar E10,Venkat EmployeeSalary.csv E01,50000 E02,50000 E03,45000 E04,45000 E05,50000 E06,45000 E07,50000 E08,10000 E09,10000 E10,10000

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

Solution :

Step 1 : Create all three files in hdfs in directory called sparkl (We will do using Hue}.

However, you can first create in local filesystem and then

Step 2 : Load EmployeeManager.csv file from hdfs and create PairRDDs

val manager = sc.textFile("spark1/EmployeeManager.csv")

val managerPairRDD = manager.map(x=> (x.split(",")(0),x.split(",")(1)))

Step 3 : Load EmployeeName.csv file from hdfs and create PairRDDs

val name = sc.textFile("spark1/EmployeeName.csv")

val namePairRDD = name.map(x=> (x.split(",")(0),x.split(\\'\")(1)))

Step 4 : Load EmployeeSalary.csv file from hdfs and create PairRDDs

val salary = sc.textFile("spark1/EmployeeSalary.csv")

val salaryPairRDD = salary.map(x=> (x.split(",")(0),x.split(",")(1)))

Step 4 : Join all pairRDDS

val joined = namePairRDD.join(salaryPairRDD}.join(managerPairRDD}

Step 5 : Now sort the joined results, val joinedData = joined.sortByKey()

Step 6 : Now generate comma separated data.

val finalData = joinedData.map(v=> (v._1, v._2._1._1, v._2._1._2, v._2._2))

Step 7 : Save this output in hdfs as text file.

finalData.saveAsTextFile("spark1/result.txt")

QUESTION 2

Problem Scenario 7 : 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 department tables using your custom boundary query, which import departments between 1 to 25.

2.

Also make sure each tables file is partitioned in 2 files e.g. part-00000, part-00002

3.

Also make sure you have imported only two columns from table, which are department_id,department_name

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

Solutions :

Step 1 : Clean the hdfs tile system, if they exists clean out.

hadoop fs -rm -R departments

hadoop fs -rm -R categories

hadoop fs -rm -R products

hadoop fs -rm -R orders

hadoop fs -rm -R order_itmes

hadoop fs -rm -R customers

Step 2 : Now import the department table as per requirement.

sqoop import \

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

--username=retail_dba \

-password=cloudera \

-table departments \

-target-dir /user/cloudera/departments \

-m2\

-boundary-query "select 1, 25 from departments" \

-columns department_id,department_name

Step 3 : Check imported data.

hdfs dfs -Is departments

hdfs dfs -cat departments/part-m-00000

hdfs dfs -cat departments/part-m-00001

QUESTION 3

Problem Scenario 18 : 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 Now accomplish following activities.

1.

Create mysql table as below.

mysql --user=retail_dba -password=cloudera

use retail_db

CREATE TABLE IF NOT EXISTS departments_hive02(id int, department_name

varchar(45), avg_salary int);

show tables;

2.

Now export data from hive table departments_hive01 in departments_hive02. While

exporting, please note following. wherever there is a empty string it should be loaded as a null value in

mysql.

wherever there is -999 value for int field, it should be created as null value.

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

Solution :

Step 1 : Create table in mysql db as well.

mysql ~user=retail_dba -password=cloudera

use retail_db

CREATE TABLE IF NOT EXISTS departments_hive02(id int, department_name

varchar(45), avg_salary int);

show tables;

Step 2 : Now export data from hive table to mysql table as per the requirement.

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

-username retaildba \

-password cloudera \

--table departments_hive02 \

-export-dir /user/hive/warehouse/departments_hive01 \

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

--input-lines-terminated-by \\'\n\\' \

--num-mappers 1 \

-batch \

-Input-null-string "" \

-input-null-non-string -999

step 3 : Now validate the data,select * from departments_hive02;

QUESTION 4

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

QUESTION 5

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)