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 71 :

Write down a Spark script using Python,

In which it read a file "Content.txt" (On hdfs) with following content.

After that split each row as (key, value), where key is first word in line and entire line as

value.

Filter out the empty lines.

And save this key value in "problem86" as Sequence file(On hdfs)

Part 2 : Save as sequence file , where key as null and entire line as value. Read back the

stored sequence files.

Content.txt

Hello this is ABCTECH.com

This is XYZTECH.com

Apache Spark Training

This is Spark Learning Session Spark is faster than MapReduce

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

Solution :

Step 1 :

Import SparkContext and SparkConf

from pyspark import SparkContext, SparkConf

Step 2:

#load data from hdfs

contentRDD = sc.textFile(MContent.txt")

Step 3:

#filter out non-empty lines

nonemptyjines = contentRDD.filter(lambda x: len(x) > 0)

Step 4:

#Split line based on space (Remember : It is mandatory to convert is in tuple} words =

nonempty_lines.map(lambda x: tuple(x.split(\\'\\', 1)))

words.saveAsSequenceFile("problem86")

Step 5: Check contents in directory problem86 hdfs dfs -cat problem86/part*

Step 6 : Create key, value pair (where key is null)

nonempty_lines.map(lambda line: (None, Mne}).saveAsSequenceFile("problem86_1")

Step 7 : Reading back the sequence file data using spark. seqRDD =

sc.sequenceFile("problem86_1")

Step 8 : Print the content to validate the same.

for line in seqRDD.collect():

print(line)

QUESTION 2

Problem Scenario 13 : 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.

Create a table in retailedb with following definition.

CREATE table departments_export (department_id int(11), department_name varchar(45),

created_date T1MESTAMP DEFAULT NOWQ);

2.

Now import the data from following directory into departments_export table,

/user/cloudera/departments new

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

Solution :

Step 1 : Login to musql db

mysql --user=retail_dba -password=cloudera

show databases; use retail_db; show tables;

step 2 : Create a table as given in problem statement.

CREATE table departments_export (departmentjd int(11), department_name varchar(45),

created_date T1MESTAMP DEFAULT NOW());

show tables;

Step 3 : Export data from /user/cloudera/departmentsnew to new table departments_export

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

-username retaildba \

--password cloudera \

--table departments_export \

-export-dir /user/cloudera/departments_new \

-batch

Step 4 : Now check the export is correctly done or not. mysql -user*retail_dba password=cloudera

show databases;

use retail _db;

show tables;

select\\' from departments_export;

QUESTION 3

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 4

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(_ + _)

QUESTION 5

Problem Scenario 89 : You have been given below patient data in csv format, patientID,name,dateOfBirth,lastVisitDate 1001,Ah Teck,1991-12-31,2012-01-20 1002,Kumar,2011-10-29,2012-09-20 1003,Ali,2011-01-30,2012-10-21 Accomplish following activities.

1.

Find all the patients whose lastVisitDate between current time and \\'2012-09-15\\'

2.

Find all the patients who born in 2011

3.

Find all the patients age

4.

List patients whose last visited more than 60 days ago

5.

Select patients 18 years old or younger

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

Solution :

Step 1:

hdfs dfs -mkdir sparksql3

hdfs dfs -put patients.csv sparksql3/

Step 2 : Now in spark shell

// SQLContext entry point for working with structured data

val sqlContext = neworg.apache.spark.sql.SQLContext(sc)

// this is used to implicitly convert an RDD to a DataFrame.

import sqlContext.implicits._

// Import Spark SQL data types and Row.

import org.apache.spark.sql._

// load the data into a new RDD

val patients = sc.textFilef\\'sparksqIS/patients.csv")

// Return the first element in this RDD

patients.first()

//define the schema using a case class

case class Patient(patientid: Integer, name: String, dateOfBirth:String , lastVisitDate:

String)

// create an RDD of Product objects

val patRDD = patients.map(_.split(M,M)).map(p => Patient(p(0).tolnt,p(1),p(2),p(3)))

patRDD.first()

patRDD.count()}

// change RDD of Product objects to a DataFrame val patDF = patRDD.toDF()

// register the DataFrame as a temp table patDF.registerTempTable("patients"}

// Select data from table

val results = sqlContext.sql(.....SELECT* FROM patients \\'.....)

// display dataframe in a tabular format

results.show()

//Find all the patients whose lastVisitDate between current time and \\'2012-09-15\\'

val results = sqlContext.sql(.....SELECT * FROM patients WHERE

TO_DATE(CAST(UNIX_TIMESTAMP(lastVisitDate, \\'yyyy-MM-dd\\') AS TIMESTAMP))

BETWEEN \\'2012-09-15\\' AND current_timestamp() ORDER BY lastVisitDate.....)

results.showQ

/.Find all the patients who born in 2011

val results = sqlContext.sql(.....SELECT * FROM patients WHERE

YEAR(TO_DATE(CAST(UNIXJTIMESTAMP(dateOfBirth, \\'yyyy-MM-dd\\') AS

TIMESTAMP))) = 2011 .....)

results. show()

//Find all the patients age

val results = sqlContext.sql(.....SELECT name, dateOfBirth, datediff(current_date(),

TO_DATE(CAST(UNIX_TIMESTAMP(dateOfBirth, \\'yyyy-MM-dd\\') AS TIMESTAMP}}}/365

AS age

FROM patients

Mini >

results.show() //List patients whose last visited more than 60 days ago -- List patients whose last visited more than 60 days ago val results = sqlContext.sql(.....SELECT name, lastVisitDate FROM patients WHERE

datediff(current_date(), TO_DATE(CAST(UNIX_TIMESTAMP[lastVisitDate, \\'yyyy-MM-dd\\')

AS T1MESTAMP))) > 60.....);

results. showQ;

-- Select patients 18 years old or younger

SELECT\\' FROM patients WHERE TO_DATE(CAST(UNIXJTIMESTAMP(dateOfBirth,

\\'yyyy-MM-dd\\') AS TIMESTAMP}) > DATE_SUB(current_date(),INTERVAL 18 YEAR);

val results = sqlContext.sql(.....SELECT\\' FROM patients WHERE

TO_DATE(CAST(UNIX_TIMESTAMP(dateOfBirth, \\'yyyy-MM--dd\\') AS TIMESTAMP)) >

DATE_SUB(current_date(), T8*365).....);

results. showQ;

val results = sqlContext.sql(.....SELECT DATE_SUB(current_date(), 18*365) FROM

patients.....);

results.show();