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 40 : You have been given sample data as below in a file called spark15/file1.txt 3070811,1963,1096,,"US","CA",,1, 3022811,1963,1096,,"US","CA",,1,56 3033811,1963,1096,,"US","CA",,1,23 Below is the code snippet to process this tile. val field= sc.textFile("spark15/f ilel.txt") val mapper = field.map(x=> A) mapper.map(x => x.map(x=> {B})).collect

Please fill in A and B so it can generate below final output

Array(Array(3070811,1963,109G, 0, "US", "CA", 0,1, 0)

,Array(3022811,1963,1096, 0, "US", "CA", 0,1, 56)

,Array(3033811,1963,1096, 0, "US", "CA", 0,1, 23)

)

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

Solution :

A. x.split(",”-1)

B. if (x. isEmpty) 0 else x

QUESTION 2

Problem Scenario 34 : You have given a file named spark6/user.csv. Data is given below: user.csv id,topic,hits Rahul,scala,120 Nikita,spark,80 Mithun,spark,1 myself,cca175,180 Now write a Spark code in scala which will remove the header part and create RDD of values as below, for all rows. And also if id is myself" than filter out row. Map(id -> om, topic -> scala, hits -> 120)

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

Solution :

Step 1 : Create file in hdfs (We will do using Hue). However, you can first create in local

filesystem and then upload it to hdfs.

Step 2 : Load user.csv file from hdfs and create PairRDDs val csv =

sc.textFile("spark6/user.csv")

Step 3 : split and clean data

val headerAndRows = csv.map(line => line.split(",").map(_.trim))

Step 4 : Get header row

val header = headerAndRows.first

Step 5 : Filter out header (We need to check if the first val matches the first header name)

val data = headerAndRows.filter(_(0) != header(O))

Step 6 : Splits to map (header/value pairs)

val maps = data.map(splits => header.zip(splits).toMap)

step 7: Filter out the user "myself

val result = maps.filter(map => mapf\\'id") != "myself")

Step 8 : Save the output as a Text file. result.saveAsTextFile("spark6/result.txt")

QUESTION 3

Problem Scenario 38 : You have been given an RDD as below,

val rdd: RDD[Array[Byte]]

Now you have to save this RDD as a SequenceFile. And below is the code snippet.

import org.apache.hadoop.io.compress.GzipCodec

rdd.map(bytesArray => (A.get(), new B(bytesArray))).saveAsSequenceFile(\\'7output/path",classOt[GzipCodec]) What would be the correct replacement for A and B in above snippet.

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

Solution :

A. NullWritable

B. BytesWritable

QUESTION 4

Problem Scenario 33 : You have given a files as below. spark5/EmployeeName.csv (id,name) spark5/EmployeeSalary.csv (id,salary) Data is given below: 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 Now write a Spark code in scala which will load these two tiles from hdfs and join the same, and produce the (name.salary) values. And save the data in multiple tile group by salary (Means each file will have name of employees with same salary). Make sure file name include salary as well.

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

Solution :

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

in local filesystem and then upload it to hdfs.

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

val name = sc.textFile("spark5/EmployeeName.csv")

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

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

val salary = sc.textFile("spark5/EmployeeSalary.csv")

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

Step 4 : Join all pairRDDS

val joined = namePairRDD.join(salaryPairRDD}

Step 5 : Remove key from RDD and Salary as a Key. val keyRemoved = joined.values

Step 6 : Now swap filtered RDD.

val swapped = keyRemoved.map(item => item.swap)

Step 7 : Now groupBy keys (It will generate key and value array) val grpByKey =

swapped.groupByKey().collect()

Step 8 : Now create RDD for values collection

val rddByKey = grpByKey.map{case (k,v) => k->sc.makeRDD(v.toSeq)}

Step 9 : Save the output as a Text file.

rddByKey.foreach{ case (k,rdd) => rdd.saveAsTextFile("spark5/Employee"+k)}

QUESTION 5

Problem Scenario 36 : You have been given a file named spark8/data.csv (type,name). data.csv 1,Lokesh 2,Bhupesh 2,Amit 2,Ratan 2,Dinesh 1,Pavan 1,Tejas 2,Sheela 1,Kumar 1,Venkat

1. Load this file from hdfs and save it back as (id, (all names of same type)) in results directory. However, make sure while saving it should be

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

Solution :

Step 1 : Create file in hdfs (We will do using Hue). However, you can first create in local

filesystem and then upload it to hdfs.

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

val name = sc.textFile("spark8/data.csv")

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

Step 3 : Now swap namePairRDD RDD.

val swapped = namePairRDD.map(item => item.swap)

Step 4 : Now combine the rdd by key.

val combinedOutput = namePairRDD.combineByKey(List(_), (x:List[String], y:String) => y ::

x, (x:List[String], y:List[String]) => x ::: y)

Step 5 : Save the output as a Text file and output must be written in a single file.

:ombinedOutput.repartition(1).saveAsTextFile("spark8/result.txt")

QUESTION 6

Problem Scenario 93 : You have to run your Spark application with locally 8 thread or locally on 8 cores. Replace XXX with correct values. spark-submit --class com.hadoopexam.MyTask XXX \ -deploy-mode cluster SSPARK_HOME/lib/hadoopexam.jar 10

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

Solution

XXX: -master local[8]

Notes : The master URL passed to Spark can be in one of the following formats:

Master URL Meaning

local Run Spark locally with one worker thread (i.e. no parallelism at all}.

local[K] Run Spark locally with K worker threads (ideally, set this to the number of cores on

your machine).

local[*] Run Spark locally with as many worker threads as logical cores on your machine.

spark://HOST:PORT Connect to the given Spark standalone cluster master. The port must

be whichever one your master is configured to use, which is 7077 by default.

mesos://HOST:PORT Connect to the given Mesos cluster. The port must be whichever one

your is configured to use, which is 5050 by default. Or, for a Mesos cluster using

ZooKeeper, use mesos://zk://.... To submit with --deploy-mode cluster, the HOST:PORT

should be configured to connect to the MesosClusterDispatcher.

yarn Connect to a YARN cluster in client or cluster mode depending on the value of deploy-mode. The cluster location will be found based on the HADOOP CONF DIR or

YARN CONF DIR variable.

QUESTION 7

Problem Scenario 29 : Please accomplish the following exercises using HDFS command line options.

1.

Create a directory in hdfs named hdfs_commands.

2.

Create a file in hdfs named data.txt in hdfs_commands.

3.

Now copy this data.txt file on local filesystem, however while copying file please make sure file properties are not changed e.g. file permissions.

4.

Now create a file in local directory named data_local.txt and move this file to hdfs in hdfs_commands directory.

5.

Create a file data_hdfs.txt in hdfs_commands directory and copy it to local file system.

6.

Create a file in local filesystem named file1.txt and put it to hdfs

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

Solution : Step 1 : Create directory hdfs dfs -mkdir hdfs_commands Step 2 : Create a file in hdfs named data.txt in hdfs_commands. hdfs dfs -touchz hdfs_commands/data.txt Step 3 : Now copy this data.txt file on local filesystem, however while copying file please make sure file properties are not changed e.g. file permissions. hdfs dfs -copyToLocal -p hdfs_commands/data.txt/home/cloudera/Desktop/HadoopExam Step 4 : Now create a file in local directory named data_local.txt and move this file to hdfs in hdfs_commands directory. touch data_local.txt hdfs dfs -moveFromLocal /home/cloudera/Desktop/HadoopExam/dataJocal.txt hdfs_commands/ Step 5 : Create a file data_hdfs.txt in hdfs_commands directory and copy it to local file

system.

hdfs dfs -touchz hdfscommands/data hdfs.txt

hdfs dfs -getfrdfs_commands/data_hdfs.txt /home/cloudera/Desktop/HadoopExam/

Step 6 : Create a file in local filesystem named filel .txt and put it to hdfs

touch filel.txt

hdfs dfs -put/home/cloudera/Desktop/HadoopExam/file1.txt hdfs_commands/

QUESTION 8

Problem Scenario 22 : You have been given below comma separated employee information. name,salary,sex,age alok,100000,male,29 jatin,105000,male,32 yogesh,134000,male,39 ragini,112000,female,35 jyotsana,129000,female,39

valmiki,123000,male,29 Use the netcat service on port 44444, and nc above data line by line. Please do the following activities.

1.

Create a flume conf file using fastest channel, which write data in hive warehouse directory, in a table called flumeemployee (Create hive table as well tor given data).

2.

Write a hive query to read average salary of all employees.

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

Solution : Step 1 : Create hive table forflumeemployee.\\' CREATE TABLE flumeemployee (name string, salary int, sex string,age int) ROW FORMAT DELIMITED FIELDS TERMINATED BY \\',\\'; Step 2 : Create flume configuration file, with below configuration for source, sink and channel and save it in flume2.conf. #Define source , sink , channel and agent, agent1 .sources = source1 agent1 .sinks = sink1 agent1.channels = channel1 # Describe/configure source1 agent1.sources.source1.type = netcat agent1.sources.source1.bind = 127.0.0.1 agent1.sources.source1.port = 44444 ## Describe sink1 agent1 .sinks.sink1.channel = memory-channel agent1.sinks.sink1.type = hdfs agent1 .sinks.sink1.hdfs.path = /user/hive/warehouse/flumeemployee hdfs-agent.sinks.hdfs-write.hdfs.writeFormat=Text agent1 .sinks.sink1.hdfs.tileType = Data Stream # Now we need to define channel1 property. agent1.channels.channel1.type = memory agent1.channels.channel1.capacity = 1000 agent1.channels.channel1.transactionCapacity = 100 # Bind the source and sink to the channel Agent1 .sources.sourcel.channels = channell agent1 .sinks.sinkl.channel = channel1 Step 3 : 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/flume2.conf --name agent1 Step 4 : Open another terminal and use the netcat service. nc localhost 44444 Step 5 : Enter data line by line. alok,100000.male,29 jatin,105000,male,32 yogesh,134000,male,39 ragini,112000,female,35 jyotsana,129000,female,39 valmiki,123000,male,29 Step 6 : Open hue and check the data is available in hive table or not. step 7 : Stop flume service by pressing ctrl+c Step 8 : Calculate average salary on hive table using below query. You can use either hive command line tool or hue. select avg(salary) from flumeemployee;

QUESTION 9

Problem Scenario 17 : 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_hiveOl(department_id int,

department_name string, avg_salary int);

2.

Create another table in mysql using below statement CREATE TABLE IF NOT EXISTS

departments_hive01(id int, department_name varchar(45), avg_salary int);

3.

Copy all the data from departments table to departments_hive01 using insert into

departments_hive01 select a.*, null from departments a;

Also insert following records as below

insert into departments_hive01 values(777, "Not known",1000);

insert into departments_hive01 values(8888, null,1000);

insert into departments_hive01 values(666, null,1100);

4.

Now import data from mysql table departments_hive01 to this hive table. Please make sure that data should be visible using below hive command. Also, while importing if null value found for department_name column replace it with "" (empty string) and for id column with -999 select * from departments_hive;

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

Solution :

Step 1 : Create hive table as below.

hive

show tables;

create table departments_hive01(department_id int, department_name string, avgsalary int);

Step 2 : Create table in mysql db as well.

mysql -user=retail_dba -password=cloudera

use retail_db

CREATE TABLE IF NOT EXISTS departments_hive01(id int, department_name varchar(45), avg_salary int);

show tables;

step 3 : Insert data in mysql table.

insert into departments_hive01 select a.*, null from departments a;

check data inserts

select\\' from departments_hive01;

Now iserts null records as given in problem. insert into departments_hive01 values(777, "Not known",1000); insert into departments_hive01 values(8888, null,1000); insert into departments_hive01 values(666, null,1100);

Step 4 : Now import data in hive as per requirement.

sqoop import \

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

~username=retail_dba \

--password=cloudera \

-table departments_hive01 \

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

--hive-import \

-hive-overwrite \

-hive-table departments_hive0l \

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

--null-string M"\

--null-non-strlng -999 \

-split-by id \

-m 1

Step 5 : Checkthe data in directory.

hdfs dfs -Is /user/hive/warehouse/departments_hive01

hdfs dfs -cat/user/hive/warehouse/departments_hive01/part"

Check data in hive table.

Select * from departments_hive01;

QUESTION 10

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[lnt], Option[lnt]))] = 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]))]

QUESTION 11

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 -ls departments

hdfs dfs -cat departments/part-m-00000

hdfs dfs -cat departments/part-m-00001

QUESTION 12

Problem Scenario 86 : In Continuation of previous question, please accomplish following activities.

1.

Select Maximum, minimum, average , Standard Deviation, and total quantity.

2.

Select minimum and maximum price for each product code.

3.

Select Maximum, minimum, average , Standard Deviation, and total quantity for each product code, hwoever make sure Average and Standard deviation will have maximum two decimal values.

4.

Select all the product code and average price only where product count is more than or equal to 3.

5.

Select maximum, minimum , average and total of all the products for each code. Also produce the same across all the products.

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

Solution :

Step 1 : Select Maximum, minimum, average , Standard Deviation, and total quantity.

val results = sqlContext.sql(\\'.....SELECT MAX(price) AS MAX , MIN(price) AS MIN ,

AVG(price) AS Average, STD(price) AS STD, SUM(quantity) AS total_products FROM

products.....)

results. showQ

Step 2 : Select minimum and maximum price for each product code.

val results = sqlContext.sql(.....SELECT code, MAX(price) AS Highest Price\\', MIN(price)

AS Lowest Price\\'

FROM products GROUP BY code.....)

results. showQ

Step 3 : Select Maximum, minimum, average , Standard Deviation, and total quantity for

each product code, hwoever make sure Average and Standard deviation will have

maximum two decimal values.

val results = sqlContext.sql(.....SELECT code, MAX(price), MIN(price),

CAST(AVG(price} AS DECIMAL(7,2)) AS Average\\', CAST(STD(price) AS DECIMAL(7,2))

AS \\'Std Dev\ SUM(quantity) FROM products

GROUP BY code.....)

results. showQ

Step 4 : Select all the product code and average price only where product count is more

than or equal to 3.

val results = sqlContext.sql(.....SELECT code AS Product Code\\',

COUNTf) AS Count\\',

CAST(AVG(price) AS DECIMAL(7,2)) AS Average\\' FROM products GROUP BY code

HAVING Count >=3"M") results. showQ

Step 5 : Select maximum, minimum , average and total of all the products for each code.

Also produce the same across all the products.

val results = sqlContext.sql("""SELECT

code,

MAX(price),

MIN(pnce),

CAST(AVG(price) AS DECIMAL(7,2)) AS Average\\',

SUM(quantity)FROM products

GROUP BY code

WITH ROLLUP""")

results. show()

QUESTION 13

Problem Scenario 73 : You have been given data in json format as below.

{"first_name":"Ankit", "last_name":"Jain"}

{"first_name":"Amir", "last_name":"Khan"}

{"first_name":"Rajesh", "last_name":"Khanna"}

{"first_name":"Priynka", "last_name":"Chopra"}

{"first_name":"Kareena", "last_name":"Kapoor"}

{"first_name":"Lokesh", "last_name":"Yadav"}

Do the following activity

1.

create employee.json file locally.

2.

Load this file on hdfs

3.

Register this data as a temp table in Spark using Python.

4.

Write select query and print this data.

5.

Now save back this selected data in json format.

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

Solution :

Step 1 : create employee.json tile locally.

vi employee.json (press insert) past the content.

Step 2 : Upload this tile to hdfs, default location hadoop fs -put employee.json

Step 3 : Write spark script

#lmport SQLContext

from pyspark import SQLContext

#Create instance of SQLContext sqIContext = SQLContext(sc)

#Load json file

employee = sqlContext.jsonFile("employee.json")

#Register RDD as a temp table employee.registerTempTablef\\'EmployeeTab"}

#Select data from Employee table

employeelnfo = sqlContext.sql("select * from EmployeeTab"}

#lterate data and print

for row in employeelnfo.collect():

print(row)

Step 4 : Write dataas a Text file employeelnfo.toJSON().saveAsTextFile("employeeJson1") Step 5: Check whether data has been created or not hadoop fs -cat employeeJsonl/part"

QUESTION 14

Problem Scenario 49 : You have been given below code snippet (do a sum of values by

key}, with intermediate output.

val keysWithValuesList = Array("foo=A", "foo=A", "foo=A", "foo=A", "foo=B", "bar=C",

"bar=D", "bar=D")

val data = sc.parallelize(keysWithValuesl_ist}

//Create key value pairs

val kv = data.map(_.split("=")).map(v => (v(0), v(l))).cache()

val initialCount = 0;

val countByKey = kv.aggregateByKey(initialCount)(addToCounts, sumPartitionCounts)

Now define two functions (addToCounts, sumPartitionCounts) such, which will

produce following results.

Output 1

countByKey.collect

res3: Array[(String, Int)] = Array((foo,5), (bar,3))

import scala.collection._

val initialSet = scala.collection.mutable.HashSet.empty[String]

val uniqueByKey = kv.aggregateByKey(initialSet)(addToSet, mergePartitionSets)

Now define two functions (addToSet, mergePartitionSets) such, which will produce

following results.

Output 2:

uniqueByKey.collect

res4: Array[(String, scala.collection.mutable.HashSet[String])] = Array((foo,Set(B, A}},

(bar,Set(C, D}}}

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

Solution : val addToCounts = (n: Int, v: String) => n + 1 val sumPartitionCounts = (p1: Int, p2: Int} => p1 + p2 val addToSet = (s: mutable.HashSet[String], v: String) => s += v val mergePartitionSets = (p1: mutable.HashSet[String], p2: mutable.HashSet[String]) => p1 ++= p2

QUESTION 15

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)
```