CCA175$^{Q\&As}$

CCA Spark and Hadoop Developer Exam

Pass Cloudera CCA175 Exam with 100% Guarantee

QUESTION 1

Problem Scenario 5 : You have been given following mysql database details.

user=retail_dba password=cloudera database=retail_db jdbc URL = jdbc:mysql://quickstart:3306/retail_db Please accomplish following activities.

1.

List all the tables using sqoop command from retail_db

2.

Write simple sqoop eval command to check whether you have permission to read database tables or not.

3.

Import all the tables as avro files in /user/hive/warehouse/retail cca174.db

4.

Import departments table as a text file in /user/cloudera/departments.

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

Solution:

Step 1 : List tables using sqoop

sqoop list-tables --connect jdbc:mysql://quickstart:330G/retail_db --username retail dba password cloudera

Step 2 : Eval command, just run a count query on one of the table.

sqoop eval \

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

-username retail_dba \

-password cloudera \

--query "select count(1) from ordeMtems"

Step 3 : Import all the tables as avro file.

sqoop import-all-tables \

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

-username=retail_dba \

-password=cloudera \

-as-avrodatafile \

-warehouse-dir=/user/hive/warehouse/retail stage.db \

-ml

Step 4 : Import departments table as a text file in /user/cloudera/departments

sqoop import \

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

-username=retail_dba \

-password=cloudera \

-table departments \

-as-textfile \

-target-dir=/user/cloudera/departments

Step 5 : Verify the imported data.

hdfs dfs -Is /user/cloudera/departments

hdfs dfs -Is /user/hive/warehouse/retailstage.db

hdfs dfs -Is /user/hive/warehouse/retail_stage.db/products

QUESTION 2

Problem Scenario 46 : You have been given belwo list in scala (name,sex,cost) for each

work done.

List(("Deeapak" , "male", 4000), ("Deepak" , "male", 2000), ("Deepika" , "female",

2000),("Deepak" , "female", 2000), ("Deepak" , "male", 1000) , ("Neeta" , "female", 2000))

Now write a Spark program to load this list as an RDD and do the sum of cost for

combination of name and sex (as key)

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

Solution :

Step 1 : Create an RDD out of this list

val rdd = sc.parallelize(List(("Deeapak" , "male", 4000}, ("Deepak" , "male", 2000),

("Deepika" , "female", 2000),("Deepak" , "female", 2000), ("Deepak" , "male", 1000} ,

("Neeta" , "female", 2000}}}

Step 2 : Convert this RDD in pair RDD

val byKey = rdd.map({case (name,sex,cost) => (name,sex)->cost})

Step 3 : Now group by Key

val byKeyGrouped = byKey.groupByKey

Step 4 : Nowsum the cost for each group

val result = byKeyGrouped.map{case ((id1,id2),values) => (id1,id2,values.sum)}

Step 5 : Save the results result.repartition(1).saveAsTextFile("spark12/result.txt")

QUESTION 3

Problem Scenario 44 : You have been given 4 files , with the content as given below: spark11/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 spark11/file2.txt 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. spark11/file3.txt 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 spark11/file4.txt Apache Storm is focused on stream processing or what some call complex event processing. Storm implements a fault tolerant method for performing a computation or pipelining multiple computations on an event as it flows into a system. One might use

Storm to transform unstructured data as it flows into a system into a desired format

(spark11Afile1.txt)

(spark11/file2.txt)

(spark11/file3.txt)

(sparkl 1/file4.txt)

Write a Spark program, which will give you the highest occurring words in each file. With

their file name and highest occurring words.

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

Solution :

Step 1 : Create all 4 file first using Hue in hdfs.

Step 2 : Load all file as an RDD

val file1 = sc.textFile("sparkl1/filel.txt")

val file2 = sc.textFile("spark11/file2.txt")

val file3 = sc.textFile("spark11/file3.txt") val file4 = sc.textFile("spark11/file4.txt")

Step 3 : Now do the word count for each file and sort in reverse order of count.

val contentl = filel.flatMap(line => line.split(" ")).map(word => (word,1)).reduceByKey(_ +

_).map(item => item.swap).sortByKey(false).map(e=>e.swap)

val content.2 = file2.flatMap(line => line.splitf ")).map(word => (word,1)).reduceByKey(_

+

_).map(item => item.swap).sortByKey(false).map(e=>e.swap)

val content3 = file3.flatMap(line > line.split)" ")).map(word => (word,1)).reduceByKey(_

+

_).map(item => item.swap).sortByKey(false).map(e=>e.swap)

val content4 = file4.flatMap(line => line.split(" ")).map(word => (word,1)).reduceByKey(_ +

_).map(item => item.swap).sortByKey(false).map(e=>e.swap)

Step 4 : Split the data and create RDD of all Employee objects.

val filelword = sc.makeRDD(Array(file1.name+"->"+content1(0)._1+"-"+content1(0)._2)) val

file2word = sc.makeRDD(Array(file2.name+"->"+content2(0)._1+"-"+content2(0)._2)) val

file3word = sc.makeRDD(Array(file3.name+"->"+content3(0)._1+"-"+content3(0)._2)) val

file4word = sc.makeRDD(Array(file4.name+M->"+content4(0)._1+"-"+content4(0)._2))

Step 5: Union all the RDDS

val unionRDDs = filelword.union(file2word).union(file3word).union(file4word)

Step 6 : Save the results in a text file as below.

unionRDDs.repartition(1).saveAsTextFile("spark11/union.txt")

QUESTION 4

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

val a = sc.parallelize(List("dog", "tiger", "lion", "cat", "spider", "eagle"), 2)

val b = a.keyBy(_.length)

val c = sc.parallelize(List("ant", "falcon", "squid"), 2)

val d = c.keyBy(.length)

operation 1

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

Array[(Int, String)] = Array((4,lion))

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

Solution : b.subtractByKey(d).collect subtractByKey [Pair] : Very similar to subtract, but instead of supplying a function, the keycomponent of each pair will be automatically used as criterion for removing items from the first RDD.

QUESTION 5

Problem Scenario 76 : 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 : (orderid , order_date , ordercustomerid, order_status} ..... Please accomplish following activities.

1.

Copy "retail_db.orders" table to hdfs in a directory p91_orders.

2.

Once data is copied to hdfs, using pyspark calculate the number of order for each status.

3.

Use all the following methods to calculate the number of order for each status. (You need to know all these functions and its behavior for real exam)

-countByKey() -groupByKey()

-reduceByKey() -aggregateByKey()

-combineByKey()

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=p91_orders

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, hadoop fs

-cat p91_orders/part-m-00000

Step 3: countByKey #Number of orders by status allOrders = sc.textFile("p91_orders")

#Generate key and value pairs (key is order status and vale as an empty string keyValue =

alIOrders.map(lambda line: (line.split(",")[3], ""))

#Using countByKey, aggregate data based on status as a key

output=keyValue.countByKey()Jtems()

for line in output: print(line)

Step 4 : groupByKey

#Generate key and value pairs (key is order status and vale as an one

keyValue = allOrders.map(lambda line: (line.split)",")[3], 1))

#Using countByKey, aggregate data based on status as a key output=

keyValue.groupByKey().map(lambda kv: (kv[0], sum(kv[1]}}}

tor line in output.collect(): print(line}

Step 5 : reduceByKey

#Generate key and value pairs (key is order status and vale as an one

keyValue = allOrders.map(lambda line: (line.split(","}[3], 1))

#Using countByKey, aggregate data based on status as a key output=

keyValue.reduceByKey(lambda a, b: a + b)

tor line in output.collect(): print(line}

Step 6: aggregateByKey

#Generate key and value pairs (key is order status and vale as an one keyValue =

allOrders.map(lambda line: (line.split(",")[3], line}}

output=keyValue.aggregateByKey(0, lambda a, b: a+1, lambda a, b: a+b}

for line in output.collect(): print(line}

Step 7 : combineByKey

#Generate key and value pairs (key is order status and vale as an one

keyValue = allOrders.map(lambda line: (line.split(",")[3], line))

output=keyValue.combineByKey(lambda value: 1, lambda ace, value: acc+1, lambda ace,

value: acc+value)

tor line in output.collect(): print(line)

#Watch Spark Professional Training provided by www.ABCTECH.com to understand more

on each above functions. (These are very important functions for real exam)