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 79 : 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 products table : (product_id | product categoryid | product_name | product_description | product_prtce | product_image) Please accomplish following activities.

1.

Copy "retaildb.products" table to hdfs in a directory p93_products

2.

Filter out all the empty prices

3.

Sort all the products based on price in both ascending as well as descending order.

4.

Sort all the products based on price as well as product_id in descending order.

5.

Use the below functions to do data ordering or ranking and fetch top 10 elements top() takeOrdered() sortByKey()

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=products -target-dir=p93_products -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 : Step 2 : Read the data from one of the partition, created using above command,

hadoop fs -cat p93_products/part-m-00000

Step 3 : Load this directory as RDD using Spark and Python (Open pyspark terminal and

do following). productsRDD = sc.textFile("p93_products")

Step 4 : Filter empty prices, if exists

#filter out empty prices lines

nonemptyjines = productsRDD.filter(lambda x: len(x.split(",")[4]) > 0)

Step 5 : Now sort data based on product_price in order.

sortedPriceProducts=nonempty_lines.map(lambdaline:(float(line.split(",")[4]),line.split(",")[2]

)).sortByKey()

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

Step 6 : Now sort data based on product_price in descending order.

sortedPriceProducts=nonempty_lines.map(lambda line:

(float(line.split(",")[4]),line.split(",")[2])).sortByKey(False)

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

Step 7 : Get highest price products name.

sortedPriceProducts=nonemptyJines.map(lambda line : (float(line.split(",")[4]),linesplit(,,,,,)[

2]))-sortByKey(False).take(1)

print(sortedPriceProducts)

Step 8 : Now sort data based on product_price as well as product_id in descending order.

#Dont forget to cast string #Tuple as key ((price,id),name)

sortedPriceProducts=nonemptyJines.map(lambda line : ((float(line

print(sortedPriceProducts)

Step 9 : Now sort data based on product_price as well as product_id in descending order,

using top() function.

#Dont forget to cast string

#Tuple as key ((price,id),name)

sortedPriceProducts=nonemptyJines.map(lambda line: ((float(line.s^^

print(sortedPriceProducts)

Step 10 : Now sort data based on product_price as ascending and product_id in ascending

order, using takeOrdered{} function.

#Dont forget to cast string

#Tuple as key ((price,id),name) sortedPriceProducts=nonemptyJines.map(lambda line:

((float(line.split(","}[4]},int(line.split(","}[0]}},line.split(","}[2]}}.takeOrdered(10, lambda tuple :

(tuple[0][0],tuple[0][1]))

Step 11 : Now sort data based on product_price as descending and product_id in

ascending order, using takeOrdered() function.

#Dont forget to cast string

#Tuple as key ((price,id},name)

#Using minus(-) parameter can help you to make descending ordering , only for numeric

value.

sortedPrlceProducts=nonemptylines.map(lambda line:

((float(line.split(",")[4]},int(line.split(",")[0]}},line.split(",")[2]}}.takeOrdered(10, lambda tuple :

(-tuple[0][0],tuple[0][1]}}

QUESTION 2

Problem Scenario 95 : You have to run your Spark application on yarn with each executor Maximum heap size to be 512MB and Number of processor cores to allocate on each executor will be 1 and Your main application required three values as input arguments V1 V2 V3. Please replace XXX, YYY, ZZZ ./bin/spark-submit -class com.hadoopexam.MyTask --master yarn-cluster--num-executors 3 --driver-memory 512m XXX YYY lib/hadoopexam.jarZZZ

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

Solution

XXX: -executor-memory 512m YYY: -executor-cores 1 ZZZ : V1 V2 V3 Notes : spark-submit on yarn options Option Description archives Comma-separated list of archives to be extracted into the working directory of each executor. The path must be globally visible inside your cluster; see Advanced Dependency Management. executor-cores Number of processor cores to allocate on each executor. Alternatively, you can use the spark.executor.cores property, executor-memory Maximum heap size to allocate to each executor. Alternatively, you can use the spark.executor.memory-property. num-executors Total number of YARN containers to allocate for this application. Alternatively, you can use the spark.executor.instances property. queue YARN queue to submit to. For more information, see Assigning Applications and Queries to Resource Pools. Default: default.

QUESTION 3

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

#Import SQLContext

from pyspark import SQLContext

#Create instance of SQLContext sqlContext = 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 4

Problem Scenario 27 : You need to implement near real time solutions for collecting information when submitted in file with below information.

Data

echo "IBM,100,20160104" >> /tmp/spooldir/bb/.bb.txt echo "IBM,103,20160105" >> /tmp/spooldir/bb/.bb.txt mv /tmp/spooldir/bb/.bb.txt /tmp/spooldir/bb/bb.txt After few mins echo "IBM,100.2,20160104" >> /tmp/spooldir/dr/.dr.txt echo "IBM,103.1,20160105" >> /tmp/spooldir/dr/.dr.txt mv /tmp/spooldir/dr/.dr.txt /tmp/spooldir/dr/dr.txt

Requirements:

You have been given below directory location (if not available than create it) /tmp/spooldir .

You have a finacial subscription for getting stock prices from BloomBerg as well as

Reuters and using ftp you download every hour new files from their respective ftp site in

directories /tmp/spooldir/bb and /tmp/spooldir/dr respectively.

As soon as file committed in this directory that needs to be available in hdfs in

/tmp/flume/finance location in a single directory.

Write a flume configuration file named flume7.conf and use it to load data in hdfs with

following additional properties .

1.

Spool /tmp/spooldir/bb and /tmp/spooldir/dr

2.

File prefix in hdfs sholuld be events

3.

File suffix should be .log

4.

If file is not commited and in use than it should have _ as prefix.

5.

Data should be written as text to hdfs

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

Solution : Step 1 : Create directory mkdir /tmp/spooldir/bb mkdir /tmp/spooldir/dr Step 2 : Create flume configuration file, with below configuration for agent1.sources = source1 source2 agent1 .sinks = sink1 agent1.channels = channel1 agent1 .sources.source1.channels = channel1 agentl .sources.source2.channels = channell agent1 .sinks.sinkl.channel = channell agent1 .sources.source1.type = spooldir agent1 .sources.sourcel.spoolDir = /tmp/spooldir/bb agent1

.sources.source2.type = spooldir agent1 .sources.source2.spoolDir = /tmp/spooldir/dr agent1 .sinks.sink1.type = hdfs agent1 .sinks.sink1.hdfs.path = /tmp/flume/finance agent1-sinks.sink1.hdfs.filePrefix = events agent1.sinks.sink1.hdfs.fileSuffix = .log agent1 .sinks.sink1.hdfs.inUsePrefix = _ agent1 .sinks.sink1.hdfs.fileType = Data Stream agent1.channels.channel1.type = file Step 4 : 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/flume7.conf --name agent1 Step 5 : Open another terminal and create a file in /tmp/spooldir/ echo "IBM,100,20160104" » /tmp/spooldir/bb/.bb.txt echo "IBM,103,20160105" » /tmp/spooldir/bb/.bb.txt mv /tmp/spooldir/bb/.bb.txt /tmp/spooldir/bb/bb.txt After few mins echo "IBM,100.2,20160104" » /tmp/spooldir/dr/.dr.txt echo "IBM,103.1,20160105" »/tmp/spooldir/dr/.dr.txt mv /tmp/spooldir/dr/.dr.txt /tmp/spooldir/dr/dr.txt

QUESTION 5

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.