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 23 : You have been given log generating service as below. Start_logs (It will generate continuous logs) Tail_logs (You can check , what logs are being generated) Stop_logs (It will stop the log service) Path where logs are generated using above service : /opt/gen_logs/logs/access.log Now write a flume configuration file named flume3.conf , using that configuration file dumps logs in HDFS file system in a directory called flumeflume3/%Y/%m/%d/%H/%M Means every minute new directory should be created). Please us the interceptors to provide timestamp information, if message header does not have header info. And also note that you have to preserve existing timestamp, if message contains it. Flume channel should have following property as well. After every 100 message it should be committed, use non-durable/faster channel and it should be able to hold maximum 1000 events.

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

Solution : Step 1 : Create flume configuration file, with below configuration for source, sink and channel. #Define source , sink , channel and agent, agent1 .sources = source1 agent1 .sinks = sink1 agent1.channels = channel1 # Describe/configure source1 agent1 .sources.source1.type = exec agentl.sources.source1.command = tail -F /opt/gen logs/logs/access.log #Define interceptors agent1 .sources.source1.interceptors=i1 agent1 .sources.source1.interceptors.i1.type=timestamp agent1 .sources.source1.interceptors.i1.preserveExisting=true ## Describe sink1 agent1 .sinks.sink1.channel = memory-channel agent1 .sinks.sink1.type = hdfs agent1 .sinks.sink1.hdfs.path = flume3/%Y/%m/%d/%H/%M agent1 .sinks.sjnkl.hdfs.fileType = 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.source1.channels = channel1 agent1.sinks.sink1.channel = channel1 Step 2 : Run below command which will use this configuration file and append data in hdfs. Start log service using : start_logs Start flume service: flume-ng agent -conf /home/cloudera/flumeconf -conf-file /home/cloudera/flumeconf/flume3.conf -Dflume.root.logger=DEBUG,INFO,console –name agent1 Wait for few mins and than stop log service. stop logs

QUESTION 2

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)

QUESTION 3

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 4

Problem Scenario 37 : ABCTECH.com has done survey on their Exam Products feedback using a web based form. With the following free text field as input in web ui. Name: StringSubscription Date: String Rating : String And servey data has been saved in a file called spark9/feedback.txt Christopher|Jan 11, 2015|5 Kapil|11 Jan, 2015|5 Thomas|6/17/2014|5 John|22-08-2013|5 Mithun|2013|5 Jitendra||5 Write a spark program using regular expression which will filter all the valid dates and save in two separate file (good record and bad record)

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

Solution :

Step 1 : Create a file first using Hue in hdfs.

Step 2 : Write all valid regular expressions sysntex for checking whether records are having

valid dates or not.

val regl =......(\d+)\s(\w{3})(,)\s(\d{4}).......r//11 Jan, 2015 val reg2 =......(\d+)(U)(\d+)(U)(\d{4})......s II 6/17/2014

val reg3 =......(\d+)(-)(\d+)(-)(\d{4})""".r//22-08-2013

val reg4 =......(\w{3})\s(\d+)(,)\s(\d{4})......s II Jan 11, 2015

Step 3 : Load the file as an RDD.

val feedbackRDD = sc.textFile("spark9/feedback.txt"}

Step 4 : As data are pipe separated , hence split the same. val feedbackSplit =

feedbackRDD.map(line => line.split(\\'|\\'))

Step 5 : Now get the valid records as well as , bad records.

val validRecords = feedbackSplit.filter(x =>

(reg1.pattern.matcher(x(1).trim).matches|reg2.pattern.matcher(x(1).trim).matches|reg3.patt

ern.matcher(x(1).trim).matches | reg4.pattern.matcher(x(1).trim).matches))

val badRecords = feedbackSplit.filter(x =>

!(reg1.pattern.matcher(x(1).trim).matches|reg2.pattern.matcher(x(1).trim).matches|reg3.pat

tern.matcher(x(1).trim).matches | reg4.pattern.matcher(x(1).trim).matches))

Step 6 : Now convert each Array to Strings

val valid =vatidRecords.map(e => (e(0),e(1),e(2)))

val bad =badRecords.map(e => (e(0),e(1),e(2)))

Step 7 : Save the output as a Text file and output must be written in a single tile,

valid.repartition(1).saveAsTextFile("spark9/good.txt")

bad.repartition(1).saveAsTextFile("sparkS7bad.txt")

QUESTION 5

Problem Scenario 24 : You have been given below comma separated employee information.

Data Set:

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

Requirements:

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 flumemaleemployee (Create hive table as well tor given data).

2.

While importing, make sure only male employee data is stored.

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

Step 1 : Create hive table for flumeemployee.\\' CREATE TABLE flumemaleemployee (

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 flume4.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.sourcel.port = 44444 #Define interceptors agent1.sources.source1.interceptors=il agent1 .sources.source1.interceptors.i1.type=regex_filter agent1 .sources.source1.interceptors.i1.regex=female agent1 .sources.source1.interceptors.i1.excludeEvents=true ## Describe sink1 agent1 .sinks, sinkl.channel = memory-channel agent1.sinks.sink1.type = hdfs agent1 .sinks, sinkl. hdfs. path = /user/hive/warehouse/flumemaleemployee hdfs-agent.sinks.hdfs-write.hdfs.writeFormat=Text agentl .sinks.sink1.hdfs.fileType = Data Stream # Now we need to define channel1 property. agent1.channels.channel1.type = memory agent1.channels.channell.capacity = 1000 agent1.channels.channel1.transactionCapacity = 100 # Bind the source and sink to the channel agent1 .sources.source1.channels = channel1 agent1 .sinks.sink1.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/flume4.conf --name agentl 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 6

Problem Scenario 83 : In Continuation of previous question, please accomplish following

activities.

1.

Select all the records with quantity >= 5000 and name starts with \\'Pen\\'

2.

Select all the records with quantity >= 5000, price is less than 1.24 and name starts with \\'Pen\\'

3.

Select all the records witch does not have quantity >= 5000 and name does not starts with \\'Pen\\'

4.

Select all the products which name is \\'Pen Red\\', \\'Pen Black\\'

5.

Select all the products which has price BETWEEN 1.0 AND 2.0 AND quantity BETWEEN 1000 AND 2000.

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

Solution :

Step 1 : Select all the records with quantity >= 5000 and name starts with \\'Pen\\'

val results = sqlContext.sql(......SELECT * FROM products WHERE quantity >= 5000 AND

name LIKE \\'Pen %.......)

results.show()

Step 2 : Select all the records with quantity >= 5000 , price is less than 1.24 and name

starts with \\'Pen\\'

val results = sqlContext.sql(......SELECT * FROM products WHERE quantity >= 5000 AND

price

results. showQ

Step 3 : Select all the records witch does not have quantity >= 5000 and name does not

starts with \\'Pen\\'

val results = sqlContext.sql(\\'.....SELECT * FROM products WHERE NOT (quantity >= 5000

AND name LIKE \\'Pen %\\')......)

results. showQ

Step 4 : Select all the products wchich name is \\'Pen Red\\', \\'Pen Black\\'

val results = sqlContext.sql(\\'.....SELECT\\' FROM products WHERE name IN (\\'Pen Red\\',

\\'Pen Black\\')......)

results. showQ

Step 5 : Select all the products which has price BETWEEN 1.0 AND 2.0 AND quantity

BETWEEN 1000 AND 2000.

val results = sqlContext.sql(......SELECT * FROM products WHERE (price BETWEEN 1.0

AND 2.0) AND (quantity BETWEEN 1000 AND 2000)......)

results. show()

QUESTION 7

Problem Scenario 11 : 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 departments table in a directory called departments.

2.

Once import is done, please insert following 5 records in departments mysql table.

Insert into departments(10, physics);

Insert into departments(11, Chemistry);

Insert into departments(12, Maths);

Insert into departments(13, Science);

Insert into departments(14, Engineering);

3.

Now import only new inserted records and append to existring directory . which has been created in first step.

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

Solution :

Step 1 : Clean already imported data. (In real exam, please make sure you dont delete

data generated from previous exercise).

hadoop fs -rm -R departments

Step 2 : Import data in departments directory.

sqoop import \

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

--username=retail_dba \

-password=cloudera \

-table departments \

"target-dir/user/cloudera/departments

Step 3 : Insert the five records in departments table.

mysql -user=retail_dba --password=cloudera retail_db

Insert into departments values(10, "physics"); Insert into departments values(11,

"Chemistry"); Insert into departments values(12, "Maths"); Insert into departments values(13, "Science"); Insert into departments values(14, "Engineering"); commit;

select\\' from departments;

Step 4 : Get the maximum value of departments from last import, hdfs dfs -cat

/user/cloudera/departments/part* that should be 7

Step 5 : Do the incremental import based on last import and append the results.

```
sqoop import \

--connect "jdbc:mysql://quickstart.cloudera:330G/retail_db" \

~username=retail_dba \

-password=cloudera \

-table departments \

--target-dir /user/cloudera/departments \

-append \

-check-column "department_id" \

-incremental append \

-last-value 7
```

Step 6 : Now check the result.

hdfs dfs -cat /user/cloudera/departments/part"

QUESTION 8

Problem Scenario 78 : 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 , order_customer_id, order_status)

Columns of ordeMtems table : (order_item_td , order_item_order_id ,

order_item_product_id,

order_item_quantity,order_item_subtotal,order_item_product_price)

Please accomplish following activities.

1.

Copy "retail_db.orders" and "retail_db.order_items" table to hdfs in respective directory p92_orders and p92_order_items .

2.

Join these data using order_id in Spark and Python

3.

Calculate total revenue perday and per customer

4.

Calculate maximum revenue customer

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=p92_orders –m 1 sqoop import -connect jdbc:mysql://quickstart:3306/retail_db -username=retail_dba password=cloudera -table=order_items --target-dir=p92_order_orderitems --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 : Read the data from one of the partition, created using above command, hadoop fs -cat p92_orders/part-m-00000 hadoop fs -cat p92 orderitems/part-m-00000 Step 3 : Load these above two directory as RDD using Spark and Python (Open pyspark terminal and do following). orders = sc.textFile(Mp92_orders") orderitems = sc.textFile("p92_order_items") Step 4 : Convert RDD into key value as (orderjd as a key and rest of the values as a value)

#First value is orderjd

orders Key Value = orders.map(lambda line: (int(line.split(",")[0]), line))

#Second value as an Orderjd

orderltemsKeyValue = orderltems.map(lambda line: (int(line.split(",")[1]), line))

Step 5 : Join both the RDD using orderjd

joinedData = orderltemsKeyValue.join(ordersKeyValue)

#print the joined data

for line in joinedData.collect():

print(line)

#Format of joinedData as below.

#[Orderld, \\'All columns from orderltemsKeyValue\\', \\'All columns from ordersKeyValue\\']

ordersPerDatePerCustomer = joinedData.map(lambda line: ((line[1][1].split(",")[1], line[1][1].split(",M)[2]), float(line[1][0].split(",")[4]))) amountCollectedPerDayPerCustomer =

ordersPerDatePerCustomer.reduceByKey(lambda runningSum, amount: runningSum +

amount}

#(Out record format will be ((date,customer_id), totalAmount} for line in

amountCollectedPerDayPerCustomer.collect(): print(line)

#now change the format of record as (date,(customer_id,total_amount))

revenuePerDatePerCustomerRDD = amountCollectedPerDayPerCustomer.map(lambda

threeElementTuple: (threeElementTuple[0][0],

(threeElementTuple[0][1],threeElementTuple[1])))

for line in revenuePerDatePerCustomerRDD.collect():

print(line)

#Calculate maximum amount collected by a customer for each day

perDateMaxAmountCollectedByCustomer =revenuePerDatePerCustomerRDD.reduceByKey(lambda runningAmountTuple,newAmountTuple: (runningAmountTuple if runningAmountTuple[1] >= newAmountTuple[1] else newAmountTuple})

for line in perDateMaxAmountCollectedByCustomer\sortByKey().collect(): print(line)

QUESTION 9

Problem Scenario 62 : You have been given below code snippet.

val a = sc.parallelize(List("dogM, "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[(Int, String)] = Array((3,xdogx), (5,xtigerx), (4,xlionx), (3,xcatx), (7,xpantherx),

(5,xeaglex))

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

Solution : b.mapValuesf\\'x" + _ + "x").collect mapValues [Pair] : Takes the values of a RDD that consists of two-component tuples, and applies the provided function to transform each value. Tlien,.it.forms newtwo-componend tuples using the key and the transformed value and stores them in a new RDD.

QUESTION 10

Problem Scenario 70 : Write down a Spark Application using Python, In which it read a

file "Content.txt" (On hdfs) with following content. Do the word count and save the

results in a directory called "problem85" (On hdfs)

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 : Create an application with following code and store it in problem84.py

```
# Import SparkContext and SparkConf

from pyspark import SparkContext, SparkConf

# Create configuration object and set App name

conf = SparkConf().setAppName("CCA 175 Problem 85") sc = sparkContext(conf=conf)

#load data from hdfs

contentRDD = sc.textFile(MContent.txt")

#filter out non-empty lines

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

#Split line based on space

words = nonempty_lines.ffatMap(lambda x: x.split(\\'\\'}}

#Do the word count

wordcounts = words.map(lambda x: (x, 1)) \

reduceByKey(lambda x, y: x+y) \

map(lambda x: (x[1], x[0]}}.sortByKey(False}

for word in wordcounts.collect(): print(word)

#Save final data " wordcounts.saveAsTextFile("problem85")
```

step 2 : Submit this application

```
spark-submit -master yarn problem85.py
```

QUESTION 11

Problem Scenario 14 : 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 activities.

1.

Create a csv file named updated_departments.csv with the following contents in local file system. updated_departments.csv 2,fitness 3,footwear 12,fathematics 13,fcience 14,engineering 1000,management

2.

Upload this csv file to hdfs filesystem,

3.

Now export this data from hdfs to mysql retaildb.departments table. During upload make sure existing department will just updated and new departments needs to be inserted.

4.

Now update updated_departments.csv file with below content. 2,Fitness 3,Footwear 12,Fathematics 13,Science 14,Engineering 1000,Management 2000,Quality Check

5.

Now upload this file to hdfs.

6.

Now export this data from hdfs to mysql retail_db.departments table. During upload make sure existing department will just updated and no new departments needs to be inserted.

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

Solution :

Step 1 : Create a csv tile named updateddepartments.csv with give content.

Step 2 : Now upload this tile to HDFS.

Create a directory called newdata.

hdfs dfs -mkdir new_data

hdfs dfs -put updated_departments.csv newdata/

Step 3 : Check whether tile is uploaded or not. hdfs dfs -Is new_data

Step 4 : Export this file to departments table using sqoop.

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

-username retail_dba \

--password cloudera \

-table departments \

--export-dir new_data \

-batch \

-m 1 \

-update-key department_id \

-update-mode allowinsert

Step 5 : Check whether required data upsert is done or not. mysql --user=retail_dba password=cloudera

show databases;

use retail_db;

show tables;

select" from departments;

Step 6 : Update updated_departments.csv file.

Step 7 : Override the existing file in hdfs.

hdfs dfs -put updated_departments.csv newdata/

Step 8 : Now do the Sqoop export as per the requirement.

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

-username retail_dba\

--password cloudera \

--table departments \

--export-dir new_data \

--batch \

-m 1 \

--update-key-department_id \

-update-mode updateonly

Step 9 : Check whether required data update is done or not. mysql --user=retail_dba password=cloudera

show databases;

use retail db;

show tables;

select" from departments;

QUESTION 12

Problem Scenario 16 : 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_hive(department_id int, department_name string);

2.

Now import data from mysql table departments to this hive table. Please make sure that

data should be visible using below hive command, select" from departments_hive

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

Solution :

Step 1 : Create hive table as said.

hive

show tables;

create table departments_hive(department_id int, department_name string);

Step 2 : The important here is, when we create a table without delimiter fields. Then default

delimiter for hive is ^A (\001). Hence, while importing data we have to provide proper

delimiter.

sqoop import \

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

~username=retail_dba \

-password=cloudera \

--table departments \

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

-hive-import \

-hive-overwrite \

--hive-table departments_hive \

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

Step 3 : Check-the data in directory.

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

hdfs dfs -cat/user/hive/warehouse/departmentshive/part\\'

Check data in hive table.

Select * from departments_hive;

QUESTION 13

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

Solution :

step 1 : Create a file first using Hue in hdfs.

Step 2 : Load tile as an RDD

val rawlines = sc.textFile("spark10/sales.txt")

Step 3 : Create a case class, which can represent its column fileds. case class

Employee(dep: String, des: String, cost: Double, state: String)

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

val employees = rawlines.map(_.split(",")).map(row=>Employee(row(0), row{1),

row{2).toDouble, row{3)))

Step 5 : Create a row as we needed. All group by fields as a key and value as a count for

each employee as well as its cost, val keyVals = employees.map(em => ((em.dep,

em.des, em.state), (1 , em.cost)))

Step 6 : Group by all the records using reduceByKey method as we want summation as

well. For number of employees and their total cost, val results = keyVals.reduceByKey{

(a,b) => (a._1 + b._1, a._2 + b._2)} // (a.count + b.count, a.cost + b.cost)}

Step 7 : Save the results in a text file as below.

results.repartition(1).saveAsTextFile("spark10/group.txt")

QUESTION 14

Problem Scenario 20 : You have been given MySQL DB with following details. user=retail_dba password=cloudera database=retail_db table=retail_db.categories jdbc URL = jdbc:mysql://quickstart:3306/retail_db Please accomplish following activities.

1. Write a Sqoop Job which will import "retaildb.categories" table to hdfs, in a directory name "categories_targetJob".

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

Solution :

Step 1 : Connecting to existing MySQL Database mysql -user=retail_dba -password=cloudera retail_db

Step 2 : Show all the available tables show tables;

Step 3 : Below is the command to create Sqoop Job (Please note that - import space is

mandatory)

sqoop job -create sqoopjob \ -- import \

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

-username=retail_dba \

-password=cloudera \

-table categories \

-target-dir categories_targetJob \

-fields-terminated-by \\'|\\' \ -lines-terminated-by \\'\n\\' Step 4 : List all the Sqoop Jobs sqoop job --list Step 5 : Show details of the Sqoop Job sqoop job --show sqoopjob Step 6 : Execute the sqoopjob sqoopjob --exec sqoopjob Step 7 : Check the output of import job hdfs dfs -Is categories_target_job hdfs dfs -cat categories_target_job/part*

QUESTION 15

Problem Scenario 87 : You have been given below three files product.csv (Create this file in hdfs) productID,productCode,name,quantity,price,supplierid 1001,PEN,Pen Red,5000,1.23,501 1002,PEN,Pen Blue,8000,1.25,501

1003,PEN,Pen Black,2000,1.25,501 1004,PEC,Pencil 2B,10000,0.48,502 1005,PEC,Pencil 2H,8000,0.49,502 1006,PEC,Pencil HB,0,9999.99,502 2001,PEC,Pencil 3B,500,0.52,501 2002,PEC,Pencil 4B,200,0.62,501 2003,PEC,Pencil 5B,100,0.73,501 2004,PEC,Pencil 6B,500,0.47,502 supplier.csv supplierid,name,phone 501,ABC Traders,88881111 502,XYZ Company,88882222 503,QQ Corp,88883333 products_suppliers.csv productID,supplierID 2001,501 2002,501 2003,501 2004,502 2001,503 Now accomplish all the queries given in solution. Select product, its price , its supplier name where product price is less than 0.6 using SparkSQL

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

Solution :

Step 1:

hdfs dfs -mkdir sparksql2

hdfs dfs -put product.csv sparksq!2/

hdfs dfs -put supplier.csv sparksql2/

hdfs dfs -put products_suppliers.csv sparksql2/

Step 2 : Now in spark shell

// this Is used to Implicitly convert an RDD to a DataFrame.

import sqlContext.impIicits._

// Import Spark SQL data types and Row.

import org.apache.spark.sql._

// load the data into a new RDD

val products = sc.textFile("sparksql2/product.csv")

val supplier = sc.textFileC\\'sparksq^supplier.csv")

val prdsup = sc.textFile("sparksql2/products_suppliers.csv"}

// Return the first element in this RDD

products.fi rst()

supplier.first{).

prdsup.first()

//define the schema using a case class

case class Product(productid: Integer, code: String, name: String, quantity:lnteger, price:

Float, supplierid:lnteger)

case class Suplier(supplierid: Integer, name: String, phone: String)

case class PRDSUP(productid: Integer.supplierid: Integer)

// create an RDD of Product objects

val prdRDD = products.map(_.split(\\'\")).map(p =>

Product(p(0).tolnt,p(1),p(2),p(3).tolnt,p(4).toFloat,p(5).toint))

val supRDD = supplier.map(_.split(",")).map(p => Suplier(p(0).tolnt,p(1),p(2)))

val prdsupRDD = prdsup.map(_.split(",")).map(p => PRDSUP(p(0).tolnt,p(1}.tolnt}}

prdRDD.first()

prdRDD.count()

supRDD.first() supRDD.count()

prdsupRDD.first() prdsupRDD.count(}

// change RDD of Product objects to a DataFrame

val prdDF = prdRDD.toDF()

val supDF = supRDD.toDF()

val prdsupDF = prdsupRDD.toDF()

// register the DataFrame as a temp table prdDF.registerTempTablef\\'products")

supDF.registerTempTablef\\'suppliers")

prdsupDF.registerTempTablef\\'productssuppliers"}

//Select product, its price , its supplier name where product price is less than 0.6

val results = sqlContext.sql(......SELECT products.name, price, suppliers.name as

sup_name FROM products JOIN suppliers ON products.supplierlD= suppliers.supplierlD

WHERE price

results. show()