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

- **Instant Download** After Purchase
- **100% Money Back** Guarantee
- **365 Days** Free Update
- **800,000+** Satisfied Customers

QUESTION 1

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();

QUESTION 2

Problem Scenario 72 : You have been given a table named "employee2" with following detail. first_name string last_name string Write a spark script in python which read this table and print all the rows and individual column values.

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

Solution :

Step 1 : Import statements for HiveContext from pyspark.sql import HiveContext

Step 2 : Create sqIContext sqIContext = HiveContext(sc)

Step 3 : Query hive

employee2 = sqlContext.sql("select\\' from employee2")

Step 4 : Now prints the data for row in employee2.collect(): print(row)

Step 5 : Print specific column for row in employee2.collect(): print(row.fi rst_name)

QUESTION 3

Problem Scenario 96 : Your spark application required extra Java options as below. XX:+PrintGCDetails-XX:+PrintGCTimeStamps Please replace the XXX values correctly ./bin/spark-submit --name "My app" --master local[4] --conf spark.eventLog.enabled=talse -conf XXX hadoopexam.jar

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

Solution

XXX: Mspark.executoi\extraJavaOptions=-XX:+PrintGCDetails -XX:+PrintGCTimeStamps"

Notes: ./bin/spark-submit \

--class

--master \

--deploy-mode \

-conf = \

other options

\

[application-arguments]

Here, conf is used to pass the Spark related contigs which are required for the application

to run like any specific property(executor memory) or if you want to override the default

property which is set in Spark-default.conf.

QUESTION 4

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 5

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 6

Problem Scenario 48 : You have been given below Python code snippet, with intermediate

output.

We want to take a list of records about people and then we want to sum up their ages and

count them.

So for this example the type in the RDD will be a Dictionary in the format of {name: NAME,

age:AGE, gender:GENDER}.

The result type will be a tuple that looks like so (Sum of Ages, Count)

people = []

people.append({\\'name\\':\\'Amit\\', \\'age\\':45,\\'gender\\':\\'M\\'})

people.append({\\'name\\':\\'Ganga\\', \\'age\\':43,\\'gender\\':\\'F\\'})

people.append({\\'name\\':\\'John\\', \\'age\\':28,\\'gender\\':\\'M\\'})

people.append({\\'name\\':\\'Lolita\\', \\'age\\':33,\\'gender\\':\\'F\\'})

people.append({\\'name\\':\\'Dont Know\\', \\'age\\':18,\\'gender\\':\\'T\\'})

peopleRdd=sc.parallelize(people) //Create an RDD

peopleRdd.aggregate((0,0), seqOp, combOp) //Output of above line : 167, 5)

Now define two operation seqOp and combOp , such that

seqOp : Sum the age of all people as well count them, in each partition. combOp :

Combine results from all partitions.

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

Solution :

seqOp = (lambda x,y: (x[0] + y[\\'age\\'],x[1] + 1))

combOp = (lambda x,y: (x[0] + y[0], x[1] + y[1]))

QUESTION 7

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 8

Problem Scenario 15 : 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.

In mysql departments table please insert following record. Insert into departments values(9999, \\""Data Science"1);

2.

Now there is a downstream system which will process dumps of this file. However, system is designed the way that it can process only files if fields are enlcosed in(\\') single quote and separate of the field should be (-} and line needs to be terminated by : (colon).

3.

If data itself contains the " (double quote } than it should be escaped by \.

4.

Please import the departments table in a directory called departments_enclosedby and file should be able to process by downstream system.

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

Solution :

Step 1 : Connect to mysql database.

mysql --user=retail_dba -password=cloudera

show databases; use retail_db; show tables;

Insert record

Insert into departments values(9999, \\'"Data Science"\\');

select" from departments;

Step 2 : Import data as per requirement.

sqoop import \

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

~username=retail_dba \

--password=cloudera \

-table departments \

-target-dir /user/cloudera/departments_enclosedby \

-enclosed-by V -escaped-by \\ -fields-terminated-by--\\' -lines-terminated-by :

Step 3 : Check the result.

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

QUESTION 9

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 10

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 11

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 =

aIIOrders.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)

QUESTION 12

Problem Scenario 77 : 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_id , order_item_order_ld ,

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 orderid in Spark and Python

3.

Calculate total revenue perday and per order

4.

Calculate total and average revenue for each date. - combineByKey -aggregateByKey

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_items –m1

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_order_items/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("p92_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

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

#Second value as an Orderjd

orderItemsKeyValue = orderItems.map(lambda line: (int(line.split(",")[1]), line))

Step 5 : Join both the RDD using orderjd

joinedData = orderItemsKeyValue.join(ordersKeyValue)

#print the joined data

for line in joinedData.collect():

print(line)

Format of joinedData as below.

[Orderld, \\'All columns from orderItemsKeyValue\\', \\'All columns from orders Key Value\\']

Step 6 : Now fetch selected values Orderld, Order date and amount collected on this order.

//Retruned row will contain ((order_date,order_id),amout_collected)

revenuePerDayPerOrder = joinedData.map(lambda row: ((row[1][1].split(M,M)[1],row[0]},

float(row[1][0].split(",")[4])))

#print the result

for line in revenuePerDayPerOrder.collect():

print(line)

Step 7 : Now calculate total revenue perday and per order

A. Using reduceByKey

totalRevenuePerDayPerOrder = revenuePerDayPerOrder.reduceByKey(lambda

runningSum, value: runningSum + value)

for line in totalRevenuePerDayPerOrder.sortByKey().collect(): print(line)

#Generate data as (date, amount_collected) (Ignore ordeMd)

dateAndRevenueTuple = totalRevenuePerDayPerOrder.map(lambda line: (line[0][0],

line[1]))

for line in dateAndRevenueTuple.sortByKey().collect(): print(line)

Step 8 : Calculate total amount collected for each day. And also calculate number of days.

#Generate output as (Date, Total Revenue for date, total_number_of_dates)

#Line 1 : it will generate tuple (revenue, 1)

#Line 2 : Here, we will do summation for all revenues at the same time another counter to

maintain number of records.

#Line 3 : Final function to merge all the combiner

totalRevenueAndTotalCount = dateAndRevenueTuple.combineByKey(\

lambda revenue: (revenue, 1), \

lambda revenueSumTuple, amount: (revenueSumTuple[0] + amount, revenueSumTuple[1]

+ 1), \

lambda tuplel, tuple2: (round(tuple1[0] + tuple2[0], 2}, tuple1[1] + tuple2[1]) \

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

Step 9 : Now calculate average for each date

averageRevenuePerDate = totalRevenueAndTotalCount.map(lambda threeElements:

(threeElements[0], threeElements[1][0]/threeElements[1][1]}}

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

Step 10 : Using aggregateByKey

#line 1 : (Initialize both the value, revenue and count)

#line 2 : runningRevenueSumTuple (Its a tuple for total revenue and total record count for

each date)

#line 3 : Summing all partitions revenue and count

totalRevenueAndTotalCount = dateAndRevenueTuple.aggregateByKey(\

```
(0,0), \
lambda runningRevenueSumTuple, revenue: (runningRevenueSumTuple[0] + revenue,
runningRevenueSumTuple[1] + 1), \
lambda tupleOneRevenueAndCount, tupleTwoRevenueAndCount:
(tupleOneRevenueAndCount[0] + tupleTwoRevenueAndCount[0],
tupleOneRevenueAndCount[1] + tupleTwoRevenueAndCount[1]) \
)
for line in totalRevenueAndTotalCount.collect(): print(line)
Step 11 : Calculate the average revenue per date
averageRevenuePerDate = totalRevenueAndTotalCount.map(lambda threeElements:
(threeElements[0], threeElements[1][0]/threeElements[1][1]))
for line in averageRevenuePerDate.collect(): print(line)
```

QUESTION 13

Problem Scenario 8 : You have been given following mysql database details as well as

other info.

Please accomplish following.

1.

Import joined result of orders and order_items table join on orders.order_id = order_items.order_item_order_id.

2.

Also make sure each tables file is partitioned in 2 files e.g. part-00000, part-00002

3.

Also make sure you use orderid columns for sqoop to use for boundary conditions.

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

Solutions:

Step 1 : Clean the hdfs file 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_items

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 \

-query="select\\' from orders join order_items on orders.orderid =

order_items.order_item_order_id where \SCONDITIONS" \

-target-dir /user/cloudera/order_join \

-split-by order_id \

--num-mappers 2

Step 3 : Check imported data.

hdfs dfs -ls order_join

hdfs dfs -cat order_join/part-m-00000

hdfs dfs -cat order_join/part-m-00001

QUESTION 14

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 15

Problem Scenario 69 : Write down a Spark Application using Python, In which it read a file "Content.txt" (On hdfs) with following content. And filter out the word which is less than 2 characters and ignore all empty lines. Once doen store the filtered data in a directory called "problem84" (On hdfs) Content.txt Hello this is ABCTECH.com This is ABYTECH.com Apache Spark TrainingThis 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 84") 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(\\'\\'}}

#filter out all 2 letter words

finalRDD = words.filter(lambda x: len(x) > 2)

for word in finalRDD.collect():

print(word)

#Save final data finalRDD.saveAsTextFile("problem84M)

step 2 : Submit this application

spark-submit -master yarn problem84.py