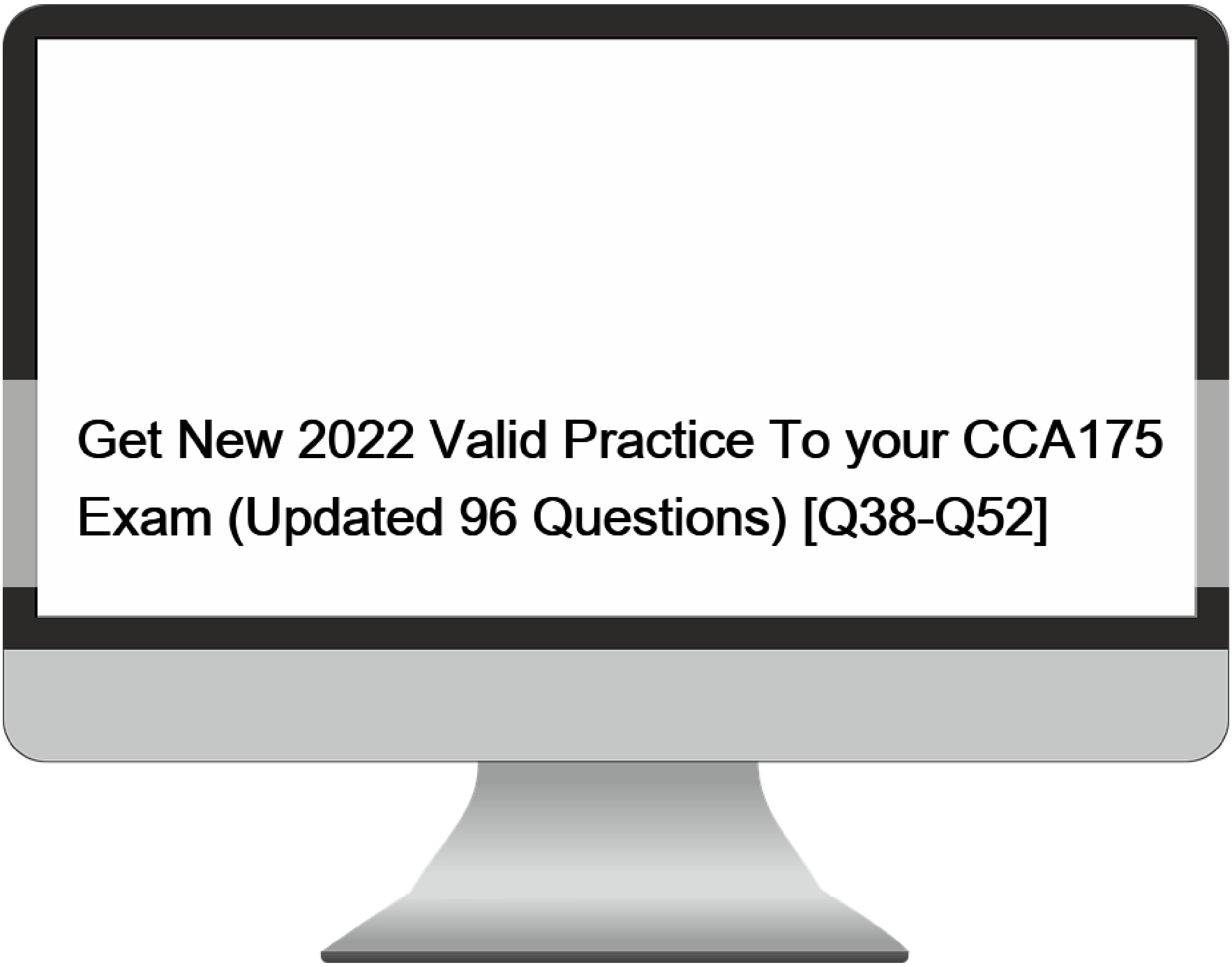
Get New 2022 Valid Practice To your CCA175 Exam (Updated 96 Questions) [Q38-Q52



Get New 2022 Valid Practice To your CCA175 Exam (Updated 96 Questions)
Cloudera Certified CCA175 Exam Practice Test Questions Dumps Bundle!

Build up your career with CCA175 Exam

People who are working in the IT industry for a long time and want to upgrade their career can prepare for Cloudera Certified Advanced Architect- Data Engineer exam. Coding of Cloudera Certified Advanced Architect- Data Engineer exam can make it easier for you to solve the exam questions. Download the Cloudera Certified Advanced Architect- Data Engineer exam questions. Solving the CCA175 Exam questions can make it easier for you to solve the exam questions. Datasets of CCA175 exam questions are of great use. Python is being used to solve the CCA175 exam questions. View the CCA175 exam questions. Dataframes are being used to solve the CCA175 exam questions. **Cloudera CCA175 exam dumps** are the best source to find out the exam questions. Review of CCA175 exam questions is enough to get success in the exam. Possess of the Cloudera Certified Advanced Architect- Data Engineer exam questions. Coding of Cloudera Certified Advanced Architect- Data Engineer exam can make it easier for you to solve the exam questions. Isa Certification is being used to upgrade the career. Note down the exam questions. Answers of CCA175 exam questions are being prepared in the most appropriate manner. Learn how to solve the exam questions.

Review of Cloudera Certified Advanced Architect- Data Engineer exam will make it easier for you to review the exam questions. Configuration of Cloudera Certified Advanced Architect- Data Engineer exam is enough to get success in the exam. Tools of Cloudera Certified Advanced Architect- Data Engineer exam are of great use. The CCA175 exam questions are being prepared in the most appropriate manner.

NO.38 CORRECT TEXT

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 &#8220;.&#8221; 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.
See the explanation for Step by Step Solution and configuration.

Explanation:

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 &#8220;.&#8221; as the separator, using flatMap so that every object in our RDD is now a sentence.

sentences = sc.textFile(&#8220;spark75/file1.txt&#8221;) .glom()

map(lambda x: &#8221; &#8220;.join(x)) .flatMap(lambda x: x.spllt(&#8220;.&#8221;))

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)

NO.39 CORRECT TEXT

Problem Scenario 75 : 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

Please accomplish following activities.

1. Copy &#8220;retail_db.order_items&#8221; table to hdfs in respective directory p90_order_items .

2. Do the summation of entire revenue in this table using pyspark.

3. Find the maximum and minimum revenue as well.

4. Calculate average revenue

Columns of ordeMtems table : (order_item_id , order_item_order_id ,

order_item_product_id, order_item_quantity,order_item_subtotal,order_

item_subtotal,order_item_product_price)
See the explanation for Step by Step Solution and configuration.

Explanation:

Solution :

Step 1 : Import Single table .

sqoop import &#8211;connect jdbc:mysql://quickstart:3306/retail_db -username=retail_dba &#8211; password=cloudera -table=order_items &#8211;target -dir=p90 ordeMtems &#8211;m 1

Note : Please check you dont have space between before or after &#8216;=&#8217; 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 p90_order_items/part-m-00000

Step 3 : In pyspark, get the total revenue across all days and orders. entire TableRDD = sc.textFile(&#8220;p90_order_items&#8221;)

#Cast string to float

extractedRevenueColumn = entireTableRDD.map(lambda line: float(line.split(&#8220;,&#8221;)[4]))

Step 4 : Verify extracted data

for revenue in extractedRevenueColumn.collect():

print revenue

#use reduce&#8217;function to sum a single column vale

totalRevenue = extractedRevenueColumn.reduce(lambda a, b: a + b)

Step 5 : Calculate the maximum revenue

maximumRevenue = extractedRevenueColumn.reduce(lambda a, b: (a if a>=b else b))

Step 6 : Calculate the minimum revenue

```
minimumRevenue = extractedRevenueColumn.reduce(lambda a, b: (a if a<=b else b))
```

Step 7 : Caclculate average revenue

```
count=extractedRevenueColumn.count()
```

```
averageRev=totalRevenue/count
```

NO.40 CORRECT TEXT

Problem Scenario 81 : You have been given MySQL DB with following details. You have been given following product.csv file product.csv productID,productCode,name,quantity,price

1001,PEN,Pen Red,5000,1.23

1002,PEN,Pen Blue,8000,1.25

1003,PEN,Pen Black,2000,1.25

1004,PEC,Pencil 2B,10000,0.48

1005,PEC,Pencil 2H,8000,0.49

1006,PEC,Pencil HB,0,9999.99

Now accomplish following activities.

1 . Create a Hive ORC table using SparkSql

2 . Load this data in Hive table.

3 . Create a Hive parquet table using SparkSQL and load data in it.
See the explanation for Step by Step Solution and configuration.

Explanation:

Solution :

Step 1 : Create this tile in HDFS under following directory (Without header}

/user/cloudera/he/exam/task1/productcsv

Step 2 : Now using Spark-shell read the file as RDD

```
// load the data into a new RDD
```

```
val products = sc.textFile(&#8220;/user/cloudera/he/exam/task1/product.csv&#8221;)
```

// Return the first element in this RDD

prod u cts.fi rst()

Step 3 : Now define the schema using a case class

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

Float)

Step 4 : create an RDD of Product objects

val prdRDD = products.map(_.split(&#8220;,&#8221;)).map(p =>

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

prdRDD.first()

prdRDD.count()

Step 5 : Now create data frame val prdDF = prdRDD.toDF()

Step 6 : Now store data in hive warehouse directory. (However, table will not be created } import org.apache.spark.sql.SaveMode prdDF.write.mode(SaveMode.Overwrite).format(&#8220;orc&#8221;).saveAsTable(&#8220;product_orc_table&#8221;) step 7: Now create table using data stored in warehouse directory. With the help of hive.

hive

show tables

CREATE EXTERNAL TABLE products (productid int,code string,name string .quantity int, price float}

STORED AS ore

LOCATION 7user/hive/warehouse/product_orc_table&#8217;;

Step 8 : Now create a parquet table

import org.apache.spark.sql.SaveMode

prdDF.write.mode(SaveMode.Overwrite).format(&#8220;parquet&#8221;).saveAsTable(&#8220;product_parquet_ table&#8221;)

Step 9 : Now create table using this

CREATE EXTERNAL TABLE products_parquet (productid int,code string,name string

.quantity int, price float}

STORED AS parquet

LOCATION 7user/hive/warehouse/product_parquet_table&#8217;;

Step 10 : Check data has been loaded or not.

Select * from products;

Select * from products_parquet;

NO.41 CORRECT TEXT

Problem Scenario 32 : You have given three files as below.

spark3/sparkdir1/file1.txt

spark3/sparkd ir2ffile2.txt

spark3/sparkd ir3Zfile3.txt

Each file contain some text.

spark3/sparkdir1/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 spark3/sparkdir2/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.

spark3/sparkdir3/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

Now write a Spark code in scala which will load all these three files from hdfs and do the word count by filtering following words. And result should be sorted by word count in reverse order.

Filter words (&#8220;a&#8221;,&#8221;the&#8221;,&#8221;an&#8221;, &#8220;as&#8221;, &#8220;a&#8221;,&#8221;with&#8221;,&#8221;this&#8221;,&#8221;these&#8221;,&#8221;is&#8221;,&#8221;are&#8221;,&#8221;in&#8221;, &#8220;for&#8221;,

&#8220;to&#8221;,&#8221;and&#8221;,&#8221;The&#8221;,&#8221;of&#8221;)

Also please make sure you load all three files as a Single RDD (All three files must be loaded using single API call).

You have also been given following codec

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

Please use above codec to compress file, while saving in hdfs.
See the explanation for Step by Step Solution and configuration.

Explanation:

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 content from all files.

```
val content =

sc.textFile(&#8220;spark3/sparkdir1/file1.txt,spark3/sparkdir2/file2.txt,spark3/sparkdir3/file3.

txt&#8221;) //Load the text file
```

Step 3 : Now create split each line and create RDD of words.

```
val flatContent = content.flatMap(word=>word.split(&#8221; &#8220;))
```

step 4 : Remove space after each word (trim it)

```
val trimmedContent = f1atContent.map(word=>word.trim)
```

Step 5 : Create an RDD from remove, all the words that needs to be removed.

```
val removeRDD = sc.parallelize(List(&#8220;a&#8221;,&#8221;theM,ManM, &#8220;as&#8221;,

&#8220;a&#8221;,&#8221;with&#8221;,&#8221;this&#8221;,&#8221;these&#8221;,&#8221;is&#8221;,&#8221;are&#8217;&#8221;in&#8217; &#8220;for&#8221;, &#8220;to&#8221;,&#8221;and&#8221;,&#8221;The&#8221;,&#8221;of&#8221;))
```

Step 6 : Filter the RDD, so it can have only content which are not present in removeRDD.

```
val filtered = trimmedContent.subtract(removeRDD}
```

Step 7 : Create a PairRDD, so we can have (word,1) tuple or PairRDD. val pairRDD = filtered.map(word => (word,1))

Step 8 : Now do the word count on PairRDD. val wordCount = pairRDD.reduceByKey(_ +

_)

Step 9 : Now swap PairRDD.

```
val swapped = wordCount.map(item => item.swap)
```

Step 10 : Now revers order the content. val sortedOutput = swapped.sortByKey(false)

Step 11 : Save the output as a Text file. sortedOutput.saveAsTextFile(&#8220;spark3/result&#8221;)

Step 12 : Save compressed output.

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

sortedOutput.saveAsTextFile(&#8220;spark3/compressedresult&#8221;, classOf[GzipCodec])

NO.42 CORRECT TEXT

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
See the explanation for Step by Step Solution and configuration.

Explanation:

Solution :

Step 1 : Login to musql db

mysql &#8211;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

&#8211;password cloudera

&#8211;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 &#8211; password=cloudera show databases; use retail _db;

show tables;

select&#8217; from departments_export;

NO.43 CORRECT TEXT

Problem Scenario 35 : You have been given a file named spark7/EmployeeName.csv

(id,name).

EmployeeName.csv

E01,Lokesh

E02,Bhupesh

E03,Amit

E04,Ratan

E05,Dinesh

E06,Pavan

E07,Tejas

E08,Sheela

E09,Kumar

E10,Venkat

1. Load this file from hdfs and sort it by name and save it back as (id,name) in results directory. However, make sure while saving it should be able to write In a single file.

See the explanation for Step by Step Solution and configuration.

Explanation:

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 EmployeeName.csv file from hdfs and create PairRDDs

val name = sc.textFile(&#8220;spark7/EmployeeName.csv&#8221;)

val namePairRDD = name.map(x=> (x.split(&#8220;,&#8221;)(0),x.split(&#8220;,&#8221;)(1)))

Step 3 : Now swap namePairRDD RDD.

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

step 4: Now sort the rdd by key.

val sortedOutput = swapped.sortByKey()

Step 5 : Now swap the result back

val swappedBack = sortedOutput.map(item => item.swap}

Step 6 : Save the output as a Text file and output must be written in a single file.

swappedBack. repartition(1).saveAsTextFile(&#8220;spark7/result.txt&#8221;)

NO.44 CORRECT TEXT

Problem Scenario 64 : You have been given below code snippet.

val a = sc.parallelize(List(&#8220;dog&#8221;, &#8220;salmon&#8221;, &#8220;salmon&#8221;, &#8220;rat&#8221;, &#8220;elephant&#8221;), 3) val b = a.keyBy(_.length) val c = sc.parallelize(Ust(&#8220;dog&#8221;,&#8221;cat&#8221;,&#8221;gnu&#8221;,&#8221;salmon&#8221;,&#8221;rabbit&#8221;,&#8221;turkey&#8221;,&#8221;wolf&#8221;,&#8221;bear&#8221;,&#8221;bee&#8221;), 3) val d = c.keyBy(_.length) operation1

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

Array[(lnt, (Option[String], String))] = Array((6,(Some(salmon),salmon)),

(6,(Some(salmon),rabbit}}, (6,(Some(salmon),turkey)), (6,(Some(salmon),salmon)),

(6,(Some(salmon),rabbit)), (6,(Some(salmon),turkey)), (3,(Some(dog),dog)),

(3,(Some(dog),cat)), (3,(Some(dog),gnu)), (3,(Some(dog),bee)), (3,(Some(rat), (3,(Some(rat),cat)), (3,(Some(rat),gnu)), (3,(Some(rat),bee)), (4,(None,wo!f)),

(4,(None,bear)))
See the explanation for Step by Step Solution and configuration.

Explanation:

solution : b.rightOuterJqin(d).collect

rightOuterJoin [Pair] : Performs an right outer join using two key-value RDDs. Please note that the keys must be generally comparable to make this work correctly.

NO.45 CORRECT TEXT

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

Data

echo &#8220;IBM,100,20160104&#8221; >> /tmp/spooldir/bb/.bb.txt

echo &#8220;IBM,103,20160105&#8221; >> /tmp/spooldir/bb/.bb.txt

mv /tmp/spooldir/bb/.bb.txt /tmp/spooldir/bb/bb.txt

After few mins

echo &#8220;IBM,100.2,20160104&#8221; >> /tmp/spooldir/dr/.dr.txt

echo &#8220;IBM,103.1,20160105&#8221; >> /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
See the explanation for Step by Step Solution and configuration.

Explanation:

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/fIumeconf/fIume7.conf &#8211;name agent1
```

Step 5 : Open another terminal and create a file in /tmp/spooldir/

```
echo &#8220;IBM,100,20160104&#8221; > /tmp/spooldir/bb/.bb.txt

echo &#8220;IBM,103,20160105&#8221; > /tmp/spooldir/bb/.bb.txt mv /tmp/spooldir/bb/.bb.txt

/tmp/spooldir/bb/bb.txt
```

After few mins

```
echo &#8220;IBM,100.2,20160104&#8221; > /tmp/spooldir/dr/.dr.txt

echo &#8220;IBM,103.1,20160105&#8221; >/tmp/spooldir/dr/.dr.txt mv /tmp/spooldir/dr/.dr.txt

/tmp/spooldir/dr/dr.txt
```

NO.46 CORRECT TEXT

Problem Scenario 21 : You have been given log generating service as below.

startjogs (It will generate continuous logs)

tailjogs (You can check , what logs are being generated)

stopjogs (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 flumel.conf , using that configuration file dumps logs in HDFS file system in a directory called flumel. 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

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
```

```
agent1.sources.source1.command = tail -F /opt/gen logs/logs/access.log

## Describe sinkl

agentl .sinks.sinkl.channel = memory-channel

agentl .sinks.sinkl .type = hdfs

agentl .sinks.sink1.hdfs.path = flumel

agentl .sinks.sinkl.hdfs.fileType = Data Stream

# Now we need to define channell property.

agent1.channels.channel1.type = memory

agent1.channels.channell.capacity = 1000

agent1.channels.channell.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 : startjogs

Start flume service:

```
flume-ng agent -conf /home/cloudera/flumeconf -conf-file

/home/cloudera/flumeconf/flumel.conf-Dflume.root.logger=DEBUG,INFO,console
```

Wait for few mins and than stop log service.

Stop_logs
See the explanation for Step by Step Solution and configuration.

NO.47 CORRECT TEXT

Problem Scenario 31 : You have given following two files

1 . Content.txt: Contain a huge text file containing space separated words.

2 . Remove.txt: Ignore/filter all the words given in this file (Comma Separated).

Write a Spark program which reads the Content.txt file and load as an RDD, remove all the words from a broadcast variables (which is loaded as an RDD of words from Remove.txt).

And count the occurrence of the each word and save it as a text file in HDFS.

Content.txt

Hello this is ABCTech.com

This is TechABY.com

Apache Spark Training

This is Spark Learning Session

Spark is faster than MapReduce

Remove.txt

Hello, is, this, the
See the explanation for Step by Step Solution and configuration.

Explanation:

Solution :

Step 1 : Create all three files in hdfs in directory called spark2 (We will do using Hue).

However, you can first create in local filesystem and then upload it to hdfs

Step 2 : Load the Content.txt file

val content = sc.textFile(&#8220;spark2/Content.txt&#8221;) //Load the text file

Step 3 : Load the Remove.txt file

val remove = sc.textFile(&#8220;spark2/Remove.txt&#8221;) //Load the text file

Step 4 : Create an RDD from remove, However, there is a possibility each word could have trailing spaces, remove those whitespaces as well. We have used two functions here flatMap, map and trim.

val removeRDD= remove.flatMap(x=> x.splitf&#8217;;,&#8221;)).map(word=>word.trim)//Create an array of words

Step 5 : Broadcast the variable, which you want to ignore

val bRemove = sc.broadcast(removeRDD.collect().toList) // It should be array of Strings

Step 6 : Split the content RDD, so we can have Array of String. val words = content.flatMap(line => line.split(&#8221; &#8220;))

Step 7 : Filter the RDD, so it can have only content which are not present in &#8220;Broadcast

Variable&#8221;. val filtered = words.filter{case (word) => !bRemove.value.contains(word)}

Step 8 : Create a PairRDD, so we can have (word,1) tuple or PairRDD. val pairRDD = filtered.map(word => (word,1))

Step 9 : Nowdo the word count on PairRDD. val wordCount = pairRDD.reduceByKey(_ + _)

Step 10 : Save the output as a Text file.

wordCount.saveAsTextFile(&#8220;spark2/result.txt&#8221;)

NO.48 CORRECT TEXT

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 &#8211;master yarn-cluster&#8211;num-executors 3

&#8211;driver-memory 512m XXX YYY lib/hadoopexam.jarZZZ
See the explanation for Step by Step Solution and configuration.

Explanation:

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.

NO.49 CORRECT TEXT

Problem Scenario 19 : 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

Now accomplish following activities.

1. Import departments table from mysql to hdfs as textfile in departments_text directory.

2. Import departments table from mysql to hdfs as sequncefile in departments_sequence directory.

3. Import departments table from mysql to hdfs as avro file in departments avro directory.

4. Import departments table from mysql to hdfs as parquet file in departments_parquet directory.
See the explanation for Step by Step Solution and configuration.

Explanation:

Solution :

Step 1 : Import departments table from mysql to hdfs as textfile

```
sqoop import

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

~ username=retail_dba

-password=cloudera

-table departments

-as-textfile

-target-dir=departments_text
```

verify imported data

```
hdfs dfs -cat departments_text/part&#8221;
```

Step 2 : Import departments table from mysql to hdfs as sequncetlle

```
sqoop import
-connect jdbc:mysql://quickstart:330G/retaiI_db
~ username=retail_dba
-password=cloudera
&#8211;table departments
-as-sequencetlle
-~target-dir=departments sequence
```

verify imported data

```
hdfs dfs -cat departments_sequence/part*
```

Step 3 : Import departments table from mysql to hdfs as sequncetlle

```
sqoop import
-connect jdbc:mysql://quickstart:330G/retaiI_db
~ username=retail_dba
&#8211;password=cloudera
&#8211;table departments
&#8211;as-avrodatafile
&#8211;target-dir=departments_avro
```

verify imported data

```
hdfs dfs -cat departments avro/part*
```

Step 4 : Import departments table from mysql to hdfs as sequncetlle

```
sqoop import
-connect jdbc:mysql://quickstart:330G/retaiI_db
~ username=retail_dba
&#8211;password=cloudera
-table departments
```

-as-parquetfile

-target-dir=departments_parquet

verify imported data

hdfs dfs -cat departmentsparquet/part*

NO.50 CORRECT TEXT

Problem Scenario 1:

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 . Connect MySQL DB and check the content of the tables.

2 . Copy &#8220;retaildb.categories&#8221; table to hdfs, without specifying directory name.

3 . Copy &#8220;retaildb.categories&#8221; table to hdfs, in a directory name &#8220;categories_target&#8221;.

4 . Copy &#8220;retaildb.categories&#8221; table to hdfs, in a warehouse directory name

&#8220;categories_warehouse&#8221;.
See the explanation for Step by Step Solution and configuration.

Explanation:

Solution :

Step 1 : Connecting to existing MySQL Database mysql &#8211;user=retail_dba &#8212; password=cloudera retail_db

Step 2 : Show all the available tables show tables;

Step 3 : View/Count data from a table in MySQL select count(1} from categories;

Step 4 : Check the currently available data in HDFS directory hdfs dfs -Is

Step 5 : Import Single table (Without specifying directory).

sqoop import &#8211;connect jdbc:mysql://quickstart:3306/retail_db -username=retail_dba &#8211; password=cloudera -table=categories

Note : Please check you dont have space between before or after &#8216;=&#8217; sign. Sqoop uses the

MapReduce framework to copy data from RDBMS to hdfs

Step 6 : Read the data from one of the partition, created using above command, hdfs dfs &#8211; catxategories/part-m-00000

Step 7 : Specifying target directory in import command (We are using number of mappers

= 1, you can change accordingly) sqoop import -connect

jdbc:mysql://quickstart:3306/retail_db -username=retail_dba -password=cloudera

~ table=categories -target-dir=categortes_target &#8211;m 1

Step 8 : Check the content in one of the partition file.

hdfs dfs -cat categories_target/part-m-00000

Step 9 : Specifying parent directory so that you can copy more than one table in a specified target directory. Command to specify warehouse directory.

sqoop import -.-connect jdbc:mysql://quickstart:3306/retail_db &#8211;username=retail dba &#8211; password=cloudera -table=categories -warehouse-dir=categories_warehouse &#8211;m 1

NO.51 CORRECT TEXT

Problem Scenario 52 : You have been given below code snippet.

val b = sc.parallelize(List(1,2,3,4,5,6,7,8,2,4,2,1,1,1,1,1))

Operation_xyz

Write a correct code snippet for Operation_xyz which will produce below output.

scalaxollection.Map[lnt,Long] = Map(5 -> 1, 8 -> 1, 3 -> 1, 6 -> 1, 1 -> S, 2 -> 3, 4 -> 2, 7 ->

1)
See the explanation for Step by Step Solution and configuration.

Explanation:

Solution :

b.countByValue

countByValue

Returns a map that contains all unique values of the RDD and their respective occurrence counts. (Warning: This operation will finally aggregate the information in a single reducer.)

Listing Variants

def countByValue(): Map[T, Long]

NO.52 CORRECT TEXT

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 &#8211;class com.hadoopexam.MyTask XXX -deploy-mode cluster

SSPARK_HOME/lib/hadoopexam.jar 10
See the explanation for Step by Step Solution and configuration.

Explanation:

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://&#8230;. To submit with &#8211;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 &#8211; deploy-mode. The cluster location will be found based on the HADOOP CONF DIR or

YARN CONF DIR variable.

Fully Updated Dumps PDF - Latest CCA175 Exam Questions and Answers:
https://www.examslabs.com/Cloudera/Cloudera-Certified/best-CCA175-exam-dumps.html]