Cloudera.CCA175.v2022-07-05.q34

Exam Code:	CCA175
Exam Name:	CCA Spark and Hadoop Developer Exam
Certification Provider:	Cloudera
Free Question Number:	34
Version:	v2022-07-05
# of views:	1729
# of Questions views:	1935
https://www.dumpsfiles.com/files/Cloudera/CCA175/Cloudera.CCA175.v2022-07-05.q34	

NEW QUESTION: 1
CORRECT TEXT
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
Answer:
See the explanation for Step by Step Solution and configuration.
Explanation:
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/

NEW QUESTION: 2
CORRECT TEXT
Problem Scenario 59 : You have been given below code snippet.
val x = sc.parallelize(1 to 20)
val y = sc.parallelize(10 to 30) operationl
z.collect
Write a correct code snippet for operationl which will produce desired output, shown below.
Array[lnt] = Array(16,12, 20,13,17,14,18,10,19,15,11)
Answer:
See the explanation for Step by Step Solution and configuration.
Explanation:
Solution :
val z = x.intersection(y)
intersection : Returns the elements in the two RDDs which are the same.

NEW QUESTION: 3
CORRECT TEXT
Problem Scenario 25 : You have been given below comma separated employee information. That needs to be added in /home/cloudera/flumetest/in.txt file (to do tail source) sex,name,city
1 ,alok,mumbai
1 ,jatin,chennai
1 ,yogesh,kolkata
2 ,ragini,delhi
2 ,jyotsana,pune
1,valmiki,banglore
Create a flume conf file using fastest non-durable channel, which write data in hive warehouse directory, in two separate tables called flumemaleemployee1 and flumefemaleemployee1
(Create hive table as well for given data}. Please use tail source with /home/cloudera/flumetest/in.txt file.
Flumemaleemployee1 : will contain only male employees data flumefemaleemployee1 :

Will contain only woman employees data

Answer:

See the explanation for Step by Step Solution and configuration.

Explanation:

Solution :

Step 1 : Create hive table for flumemaleemployeel and .'

```
CREATE TABLE flumemaleemployeel
(
sex_type int, name string, city string )
ROW FORMAT DELIMITED FIELDS TERMINATED BY ',';
CREATE TABLE flumefemaleemployeel
(
sex_type int, name string, city string
)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ',';
```

Step 2 : Create below directory and file mkdir /home/cloudera/flumetest/ cd /home/cloudera/flumetest/

Step 3 : Create flume configuration file, with below configuration for source, sink and channel and save it in flume5.conf.

```
agent.sources = tailsrc
agent.channels = mem1 mem2
agent.sinks = stdl std2
agent.sources.tailsrc.type = exec
agent.sources.tailsrc.command = tail -F /home/cloudera/flumetest/in.txt
agent.sources.tailsrc.batchSize = 1 agent.sources.tailsrc.interceptors = i1
agent.sources.tailsrc.interceptors.i1.type = regex_extractor
agent.sources.tailsrc.interceptors.il.regex = A(\\d} agent.sources.tailsrc.
interceptors. M.serializers = t1 agent.sources.tailsrc. interceptors, i1.serializers.t1. name =
type agent.sources.tailsrc.selector.type = multiplexing
agent.sources.tailsrc.selector.header = type agent.sources.tailsrc.selector.mapping.1 =
memi agent.sources.tailsrc.selector.mapping.2 = mem2 agent.sinks.std1.type = hdfs
agent.sinks.stdl.channel = mem1
agent.sinks.stdl.batchSize = 1
agent.sinks.std1.hdfs.path = /user/hive/warehouse/flumemaleemployeei
agent.sinks.stdl.rolllnterval = 0
agent.sinks.stdl.hdfs.tileType = Data Stream
agent.sinks.std2.type = hdfs
agent.sinks.std2.channel = mem2
agent.sinks.std2.batchSize = 1
```

agent.sinks.std2.hdfs.path = /user/hi ve/warehouse/flumefemaleemployee1
agent.sinks.std2.rolllnterval = 0 agent.sinks.std2.hdfs.tileType = Data Stream
agent.channels.mem1.type = memory agent.channels.meml.capacity = 100
agent.channels.mem2.type = memory agent.channels.mem2.capacity = 100
agent.sources.tailsrc.channels = mem1 mem2
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/flume5.conf --name agent
Step 5 : Open another terminal create a file at /home/cloudera/flumetest/in.txt.
Step 6 : Enter below data in file and save it.
l.alok.mumbai
1 jatin.chennai
1 ,yogesh,kolkata
2 ,ragini,delhi
2 ,jyotsana,pune
1,valmiki,banglore
Step 7 : Open hue and check the data is available in hive table or not.
Step 8 : Stop flume service by pressing ctrl+c

NEW QUESTION: 4
CORRECT TEXT
Problem Scenario 50 : You have been given below code snippet (calculating an average score}, with intermediate output.
type ScoreCollector = (Int, Double)
type PersonScores = (String, (Int, Double))
val initialScores = Array(("Fred", 88.0), ("Fred", 95.0), ("Fred", 91.0), ("Wilma", 93.0), ("Wilma", 95.0), ("Wilma", 98.0))
val wilmaAndFredScores = sc.parallelize(initialScores).cache()
val scores = wilmaAndFredScores.combineByKey(createScoreCombiner, scoreCombiner, scoreMerger) val averagingFunction = (personScore: PersonScores) => { val (name, (numberScores, totalScore)) = personScore (name, totalScore / numberScores)
}
val averageScores = scores.collectAsMap(}.map(averagingFunction)
Expected output: averageScores: scala.collection.Map[String,Double] = Map(Fred -> 91.33333333333333, Wilma -> 95.33333333333333)
Define all three required function , which are input for combineByKey method, e.g. (createScoreCombiner, scoreCombiner, scoreMerger). And help us producing required results.
Answer:
See the explanation for Step by Step Solution and configuration.

Explanation:
Solution :
val createScoreCombiner = (score: Double) => (1, score)
val scoreCombiner = (collector: ScoreCollector, score: Double) => {
val (numberScores. totalScore) = collector (numberScores + 1, totalScore + score)
}
val scoreMerger= (collector-!: ScoreCollector, collector2: ScoreCollector) => { val
(numScoresl. totalScorel) = collector! val (numScores2, tota!Score2) = collector
(numScoresl + numScores2, totalScorel + totalScore2)
}

NEW QUESTION: 5
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 --master yarn-cluster--num-executors 3
--driver-memory 512m XXX YYY lib/hadoopexam.jarZZZ
Answer:
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.

NEW QUESTION: 6
CORRECT TEXT

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.
Answer:
See the explanation for Step by Step Solution and configuration.
Explanation:
Solution :
seqOp = (lambda x,y: (x[0] + y['age'],x[1] + 1))
combOp = (lambda x,y: (x[0] + y[0], x[1] + y[1]))

NEW QUESTION: 7
CORRECT TEXT
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: String
Subscription 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)
Answer:

See the explanation for Step by Step Solution and configuration.
Explanation:
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.pattern.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.pattern.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")

NEW QUESTION: 8
CORRECT TEXT
Problem Scenario 40 : You have been given sample data as below in a file called spark15/file1.txt
3070811,1963,1096,,"US","CA",,1,
3022811,1963,1096,,"US","CA",,1,56
3033811,1963,1096,,"US","CA",,1,23
Below is the code snippet to process this tile.
val field= sc.textFile("spark15/f ilel.txt")
val mapper = field.map(x=> A)
mapper.map(x => x.map(x=> {B})).collect
Please fill in A and B so it can generate below final output
Array(Array(3070811,1963,109G, 0, "US", "CA", 0,1, 0)

,Array(3022811,1963,1096, 0, "US", "CA", 0,1, 56)
,Array(3033811,1963,1096, 0, "US", "CA", 0,1, 23)
)

Answer:

See the explanation for Step by Step Solution and configuration.

Explanation:

Solution :

A. x.split(","-1)

B. if (x. isEmpty) 0 else x

NEW QUESTION: 9

CORRECT TEXT

Problem Scenario 9 : 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.

2. Again import departments table same directory (However, directory already exist hence it should not overrride and append the results)

3. Also make sure your results fields are terminated by '|' and lines terminated by '\n\

Answer:

See the explanation for Step by Step Solution and configuration.

Explanation:

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:330G/retail_db \

--username=retail_dba \

-password=cloudera \

-table departments \

-target-dir=departments \

-fields-terminated-by '|' \
-lines-terminated-by '\n' \
-ml
Step 3 : Check imported data.
hdfs dfs -Is departments
hdfs dfs -cat departments/part-m-00000
Step 4 : Now again import data and needs to appended.
sqoop import \
-connect jdbc:mysql://quickstart:3306/retail_db \
--username=retail_dba \
-password=cloudera \
-table departments \
-target-dir departments \
-append \
-tields-terminated-by '|' \
-lines-termtnated-by '\n' \
-ml
Step 5 : Again Check the results
hdfs dfs -Is departments
hdfs dfs -cat departments/part-m-00001

NEW QUESTION: 10
CORRECT TEXT
Problem Scenario 34 : You have given a file named spark6/user.csv.
Data is given below:
user.csv
id,topic,hits
Rahul,scala,120
Nikita,spark,80
Mithun,spark,1
myself,cca175,180
Now write a Spark code in scala which will remove the header part and create RDD of values as below, for all rows. And also if id is myself" than filter out row.
Map(id -> om, topic -> scala, hits -> 120)
Answer:
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 user.csv file from hdfs and create PairRDDs val csv =

sc.textFile("spark6/user.csv")
Step 3 : split and clean data
val headerAndRows = csv.map(line => line.split(",").map(_.trim))
Step 4 : Get header row
val header = headerAndRows.first
Step 5 : Filter out header (We need to check if the first val matches the first header name)
val data = headerAndRows.filter(_(0) != header(O))
Step 6 : Splits to map (header/value pairs)
val maps = data.map(splits => header.zip(splits).toMap)
step 7: Filter out the user "myself
val result = maps.filter(map => mapf'id") != "myself")
Step 8 : Save the output as a Text file. result.saveAsTextFile("spark6/result.txt")

NEW QUESTION: 11
CORRECT TEXT
Problem Scenario 41 : You have been given below code snippet.
val aul = sc.parallelize(List (("a" , Array(1,2)), ("b" , Array(1,2)))) val au2 =
sc.parallelize(List (("a" , Array(3)), ("b" , Array(2))))
Apply the Spark method, which will generate below output.
Array[(String, Array[lnt])] = Array((a,Array(1, 2)), (b,Array(1, 2)), (a(Array(3)), (b,Array(2)))
Answer:
See the explanation for Step by Step Solution and configuration.
Explanation:
Solution:
au1.union(au2)

NEW QUESTION: 12
CORRECT TEXT
Problem Scenario 53 : You have been given below code snippet.
val a = sc.parallelize(1 to 10, 3)
operation1
b.collect
Output 1
Array[lnt] = Array(2, 4, 6, 8,10)
operation2
Output 2
Array[lnt] = Array(1,2, 3)
Write a correct code snippet for operation1 and operation2 which will produce desired output, shown above.
Answer:
See the explanation for Step by Step Solution and configuration.

Explanation:
Solution :
valb = a.filter(_%2==0)
a.filter(_ < 4).collect
filter
Evaluates a boolean function for each data item of the RDD and puts the items for which the function returned true into the resulting RDD.
When you provide a filter function, it must be able to handle all data items contained in the RDD. Scala provides so-called partial functions to deal with mixed data types (Tip: Partial functions to deal are very useful if you have some data which may be bad and you do not want to handle but for the good data (matching data) you want to apply some Kind of map function. The following article is good. It teaches you about partial functions in a very nice way and explains why case has to be used for partial functions:article)
Examples for mixed data without partial functions
val b = sc.parallelize(1 to 8)
b.filter(_ < 4)xollect
res15: ArrayjInt] = Array(1, 2, 3)
val a = sc.parallelize(List("cat'\ "horse", 4.0, 3.5, 2, "dog"))
a.filter(_<4).collect
error: value < is not a member of Any

NEW QUESTION: 13
CORRECT TEXT
Problem Scenario 51 : You have been given below code snippet.
val a = sc.parallelize(List(1, 2,1, 3), 1)
val b = a.map((_, "b"))
val c = a.map((_, "c"))
Operation_xyz
Write a correct code snippet for Operationxyz which will produce below output.
Output:
Array[(lnt, (Iterable[String], Iterable[String]))] = Array(
(2,(ArrayBuffer(b),ArrayBuffer(c))),
(3,(ArrayBuffer(b),ArrayBuffer(c))),
(1,(ArrayBuffer(b, b),ArrayBuffer(c, c)))
)
Answer:
See the explanation for Step by Step Solution and configuration.
Explanation:
Solution :
b.cogroup(c).collect
cogroup [Pair], groupWith [Pair]

A very powerful set of functions that allow grouping up to 3 key-value RDDs together using their keys.
Another example
val x = sc.parallelize(List((1, "apple"), (2, "banana"), (3, "orange"), (4, "kiwi")), 2) val y = sc.parallelize(List((5, "computer"), (1, "laptop"), (1, "desktop"), (4, "iPad")), 2)
x.cogroup(y).collect
Array[(Int, (Iterable[String], Iterable[String]))] = Array(
(4,(ArrayBuffer(kiwi),ArrayBuffer(iPad))),
(2,(ArrayBuffer(banana),ArrayBuffer())),
(3,(ArrayBuffer(orange),ArrayBuffer())),
(1 ,(ArrayBuffer(apple),ArrayBuffer(laptop, desktop))),
(5,{ArrayBuffer(),ArrayBuffer(computer))))

NEW QUESTION: 14
CORRECT TEXT
Problem Scenario 2 :
There is a parent organization called "ABC Group Inc", which has two child companies named Tech Inc and MPTech.
Both companies employee information is given in two separate text file as below. Please do the following activity for employee details.
Tech Inc.txt
1,Alok,Hyderabad
2,Krish,Hongkong
3,Jyoti,Mumbai
4 ,Atul,Banglore
5 ,Ishan,Gurgaon
MPTech.txt
6 ,John,Newyork
7 ,alp2004,California
8 ,tellme,Mumbai
9 ,Gagan21,Pune
1 0,Mukesh,Chennai
1 . Which command will you use to check all the available command line options on HDFS and How will you get the Help for individual command.
2. Create a new Empty Directory named Employee using Command line. And also create an empty file named in it Techinc.txt
3. Load both companies Employee data in Employee directory (How to override existing file in HDFS).
4. Merge both the Employees data in a Single tile called MergedEmployee.txt, merged tiles should have new line character at the end of each file content.

5. Upload merged file on HDFS and change the file permission on HDFS merged file, so that owner and group member can read and write, other user can read the file.
6. Write a command to export the individual file as well as entire directory from HDFS to local file System.

Answer:

See the explanation for Step by Step Solution and configuration.

Explanation:

Solution :

Step 1 : Check All Available command hdfs dfs

Step 2 : Get help on Individual command hdfs dfs -help get

Step 3 : Create a directory in HDFS using named Employee and create a Dummy file in it called e.g. Techinc.txt hdfs dfs -mkdir Employee

Now create an emplty file in Employee directory using Hue.

Step 4 : Create a directory on Local file System and then Create two files, with the given data in problems.

Step 5 : Now we have an existing directory with content in it, now using HDFS command line , overrid this existing Employee directory. While copying these files from local file System to HDFS. cd /home/cloudera/Desktop/ hdfs dfs -put -f Employee

Step 6 : Check All files in directory copied successfully hdfs dfs -ls Employee

Step 7 : Now merge all the files in Employee directory, hdfs dfs -getmerge -nl Employee MergedEmployee.txt

Step 8 : Check the content of the file. cat MergedEmployee.txt

Step 9 : Copy merged file in Employeed directory from local file ssytem to HDFS. hdfs dfs -put MergedEmployee.txt Employee/

Step 10 : Check file copied or not. hdfs dfs -ls Employee

Step 11 : Change the permission of the merged file on HDFS hdfs dfs -chmpd 664 Employee/MergedEmployee.txt

Step 12 : Get the file from HDFS to local file system, hdfs dfs -get Employee Employee_hdfs

NEW QUESTION: 15

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.

Answer:

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"
```

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 \
--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 \
--password=cloudera \
--table departments \
--as-avrodatafile \
--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/retail_db \
~ username=retail_dba \
--password=cloudera \
-table departments \
-as-parquetfile \
-target-dir=departments_parquet
verify imported data
hdfs dfs -cat departmentsparquet/part*

NEW QUESTION: 16
CORRECT TEXT
Problem Scenario 56 : You have been given below code snippet.
val a = sc.parallelize(l to 100. 3)
operation1
Write a correct code snippet for operationl which will produce desired output, shown below.
Array [Array [I nt]] = Array(Array(1, 2, 3,4, 5, 6, 7, 8, 9,10,11,12,13,14,15,16,17,18,19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33),
Array(34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 5 6, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66),
Array(67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 8 9, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100))
Answer:
See the explanation for Step by Step Solution and configuration.
Explanation:
Solution : a.glom.collect
glom
Assembles an array that contains all elements of the partition and embeds it in an RDD. Each returned array contains the contents of one panition

NEW QUESTION: 17
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 --class com.hadoopexam.MyTask XXX \ -deploy-mode cluster SSPARK_HOME/lib/hadoopexam.jar 10

Answer:

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://.... To submit with --deploy-mode cluster, the HOST:PORT should be configured to connect to the MesosClusterDispatcher.

yarn Connect to a YARN cluster in client or cluster mode depending on the value of -deploy-mode. The cluster location will be found based on the HADOOP CONF DIR or YARN CONF DIR variable.

NEW QUESTION: 18

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 ("a","the","an", "as", "a","with","this","these","is","are","in", "for", "to","and","The","of")
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.
Answer:
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("spark3/sparkdir1/file1.txt,spark3/sparkdir2/file2.txt,spark3/sparkdir3/file3.txt") //Load the text file
Step 3 : Now create split each line and create RDD of words.
val flatContent = content.flatMap(word=>word.split(" "))
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("a","theM,ManM, "as", "a","with","this","these","is","are'\"in'\ "for", "to","and","The","of"))
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("spark3/result")
Step 12 : Save compressed output.
import org.apache.hadoop.io.compress.GzipCodec
sortedOutput.saveAsTextFile("spark3/compressedresult", classOf[GzipCodec])

NEW QUESTION: 19
CORRECT TEXT
Problem Scenario 18 : 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. Create mysql table as below.
mysql --user=retail_dba -password=cloudera
use retail_db
CREATE TABLE IF NOT EXISTS departments_hive02(id int, department_name varchar(45), avg_salary int);
show tables;
2. Now export data from hive table departments_hive01 in departments_hive02. While exporting, please note following. wherever there is a empty string it should be loaded as a null value in mysql.
wherever there is -999 value for int field, it should be created as null value.
Answer:
See the explanation for Step by Step Solution and configuration.
Explanation:
Solution :
Step 1 : Create table in mysql db as well.
mysql ~user=retail_dba -password=cloudera
use retail_db
CREATE TABLE IF NOT EXISTS departments_hive02(id int, department_name varchar(45), avg_salary int);
show tables;
Step 2 : Now export data from hive table to mysql table as per the requirement.
sqoop export --connect jdbc:mysql://quickstart:3306/retail_db \
-username retaildba \
-password cloudera \
--table departments_hive02 \

-export-dir /user/hive/warehouse/departments_hive01 \
-input-fields-terminated-by '\001' \
--input-lines-terminated-by '\n' \
--num-mappers 1 \
-batch \
-Input-null-string "" \
-input-null-non-string -999
step 3 : Now validate the data,select * from departments_hive02;

NEW QUESTION: 20
CORRECT TEXT
Problem Scenario 22 : You have been given below comma separated employee information.
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
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 flumeemployee (Create hive table as well tor given data).
2. Write a hive query to read average salary of all employees.
Answer:
See the explanation for Step by Step Solution and configuration.
Explanation:
Solution :
Step 1 : Create hive table forflumeemployee.'
CREATE TABLE flumeemployee
(
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 flume2.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.source1.port = 44444
Describe sink1
agent1 .sinks.sink1.channel = memory-channel
agent1.sinks.sink1.type = hdfs
agent1 .sinks.sink1.hdfs.path = /user/hive/warehouse/flumeemployee
hdfs-agent.sinks.hdfs-write.hdfs.writeFormat=Text
agent1 .sinks.sink1.hdfs.tileType = 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.sourcel.channels = channell agent1 .sinks.sinkl.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/flume2.conf --name agent1
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;

NEW QUESTION: 21
CORRECT TEXT
Problem Scenario 80 : You have been given MySQL DB with following details.
user=retail_dba
password=cloudera

database=retail_db
table=retail_db.products
jdbc URL = jdbc:mysql://quickstart:3306/retail_db
Columns of products table : (product_id | product_category_id | product_name | product_description | product_price | product_image)
Please accomplish following activities.
1. Copy "retaildb.products" table to hdfs in a directory p93_products
2. Now sort the products data sorted by product price per category, use productcategoryid colunm to group by category

Answer:

See the explanation for Step by Step Solution and configuration.
Explanation:
Solution :
Step 1 : Import Single table .
sqoop import --connect jdbc:mysql://quickstart:3306/retail_db -username=retail_dba -password=cloudera -table=products --target-dir=p93
Note : Please check you dont have space between before or after '=' sign. Sqoop uses the MapReduce framework to copy data from RDBMS to hdfs
Step 2 : Step 2 : Read the data from one of the partition, created using above command, hadoop fs -cat p93_products/part-m-00000
Step 3 : Load this directory as RDD using Spark and Python (Open pyspark terminal and do following}. productsRDD = sc.textFile(Mp93_products")
Step 4 : Filter empty prices, if exists
#filter out empty prices lines
Nonempty_lines = productsRDD.filter(lambda x: len(x.split(",")[4]) > 0)
Step 5 : Create data set like (categroyld, (id,name,price)
mappedRDD = nonempty_lines.map(lambda line: (line.split(",")[1], (line.split(",")[0], line.split(",")[2], float(line.split(",")[4])))) tor line in mappedRDD.collect(): print(line)
Step 6 : Now groupBy the all records based on categoryld, which a key on mappedRDD it will produce output like (categoryld, iterable of all lines for a key/categoryld)
groupByCategroyld = mappedRDD.groupByKey() for line in groupByCategroyld.collect(): print(line)
step 7 : Now sort the data in each category based on price in ascending order.
# sorted is a function to sort an iterable, we can also specify, what would be the Key on which we want to sort in this case we have price on which it needs to be sorted.
groupByCategroyld.map(lambda tuple: sorted(tuple[1], key=lambda tupleValue: tupleValue[2])).take(5)
Step 8 : Now sort the data in each category based on price in descending order.
sorted is a function to sort an iterable, we can also specify, what would be the Key on which we want to sort in this case we have price which it needs to be sorted.
on groupByCategroyld.map(lambda tuple: sorted(tuple[1], key=lambda tupleValue:

tupleValue[2] , reverse=True)).take(5)

NEW QUESTION: 22
CORRECT TEXT
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
Answer:
See the explanation for Step by Step Solution and configuration.
Explanation:
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

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.
#[OrderId, 'All columns from orderItemsKeyValue', '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)

NEW QUESTION: 23
CORRECT TEXT
Problem Scenario 4: 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.
Import Single table categories (Subset data} to hive managed table , where category_id between 1 and 22
Answer:
See the explanation for Step by Step Solution and configuration.
Explanation:

Solution :
Step 1 : Import Single table (Subset data)
sqoop import --connect jdbc:mysql://quickstart:3306/retail_db -username=retail_dba -password=cloudera -table=categories -where "\'category_id\' between 1 and 22" --hive-import --m 1
Note: Here the ' is the same you find on ~ key
This command will create a managed table and content will be created in the following directory.
/user/hive/warehouse/categories
Step 2 : Check whether table is created or not (In Hive)
show tables;
select * from categories;

NEW QUESTION: 24
CORRECT TEXT
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".
Answer:
See the explanation for Step by Step Solution and configuration.
Explanation:
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*

NEW QUESTION: 25
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 "." 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.
Answer:
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 "." 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)

NEW QUESTION: 26
CORRECT TEXT
Problem Scenario 90 : You have been given below two files
course.txt
id,course
1 ,Hadoop
2 ,Spark
3 ,HBase
fee.txt
id,fee
2,3900
3,4200
4,2900
Accomplish the following activities.
1. Select all the courses and their fees , whether fee is listed or not.
2. Select all the available fees and respective course. If course does not exists still list the fee

3. Select all the courses and their fees , whether fee is listed or not. However, ignore records having fee as null.

Answer:

See the explanation for Step by Step Solution and configuration.

Explanation:

Solution :

Step 1:

hdfs dfs -mkdir sparksql4

hdfs dfs -put course.txt sparksql4/

hdfs dfs -put fee.txt sparksql4/

Step 2 : Now in spark shell

// load the data into a new RDD

val course = sc.textFile("sparksql4/course.txt")

val fee = sc.textFile("sparksql4/fee.txt")

// Return the first element in this RDD

course.fi rst()

fee.fi rst()

//define the schema using a case class case class Course(id: Integer, name: String) case class Fee(id: Integer, fee: Integer)

// create an RDD of Product objects

val courseRDD = course.map(_.split(",")).map(c => Course(c(0).tolnt,c(1))) val feeRDD =fee.map(_.split(",")).map(c => Fee(c(0}.tolnt,c(1}.tolnt)) courseRDD.first()

courseRDD.count(}

feeRDD.first()

feeRDD.countQ

// change RDD of Product objects to a DataFrame val courseDF = courseRDD.toDF(} val feeDF = feeRDD.toDF{)

// register the DataFrame as a temp table courseDF. registerTempTable("course") feeDF. registerTempTablef'fee")

// Select data from table

val results = sqlContext.sql(.....SELECT' FROM course """)

results. showQ

val results = sqlContext.sql(.....SELECT' FROM fee.....)

results. showQ

val results = sqlContext.sql(.....SELECT * FROM course LEFT JOIN fee ON course.id = fee.id.....) results-showQ val results ="sqlContext.sql(.....SELECT * FROM course RIGHT JOIN fee ON course.id = fee.id "MM) results. showQ val results = sqlContext.sql(.....SELECT' FROM course LEFT JOIN fee ON course.id = fee.id where fee.id IS NULL" results. show()

NEW QUESTION: 27

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 "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
Answer:
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/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

NEW QUESTION: 28
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.
Answer:
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("/user/cloudera/he/exam/task1/product.csv")
// 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:Integer, price: Float)
Step 4 : 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))
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("orc").saveAsTable("product_orc_table")
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';
Step 8 : Now create a parquet table
import org.apache.spark.sql.SaveMode
prdDF.write.mode(SaveMode.Overwrite).format("parquet").saveAsTable("product_parquet_table")
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';
Step 10 : Check data has been loaded or not.
Select * from products;
Select * from products_parquet;

NEW QUESTION: 29
CORRECT TEXT
Problem Scenario 74 : 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 : (orderjd , order_date , ordercustomerid, order status}
Columns of orderjtems 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 "retaildb.orders" and "retaildb.orderjtems" table to hdfs in respective directory p89_orders and p89_order_items .
2. Join these data using orderjd in Spark and Python
3. Now fetch selected columns from joined data Orderld, Order date and amount collected on this order.
4. Calculate total order placed for each date, and produced the output sorted by date.
Answer:
See the explanation for Step by Step Solution and configuration.
Explanation:
Solution:
Step 1 : Import Single table .
sqoop import --connect jdbc:mysql://quickstart:3306/retail_db -username=retail_dba - password=cloudera -table=orders --target-dir=p89_orders - -m1 sqoop import --connect jdbc:mysql://quickstart:3306/retail_db -username=retail_dba - password=cloudera - table=order_items ~target-dir=p89_ order items -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, hadoopfs -cat p89_orders/part-m-00000 hadoop fs -cat p89_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("p89_orders") orderitems = sc.textFile("p89_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
tor line in joinedData.collect():
print(line)
Format of joinedData as below.

[OrderId, 'All columns from orderItemsKeyValue', 'All columns from orders Key Value']
Step 6 : Now fetch selected values OrderId, Order date and amount collected on this order.
revenuePerOrderPerDay = joinedData.map(lambda row: (row[0](row[1][1].split(",")[1](f!
oat(row[1][0].split('\M}[4]}}}
#printthe result
for line in revenuePerOrderPerDay.collect():
print(line)
Step 7 : Select distinct order ids for each date.
#distinct(date,order_id)
distinctOrdersDate = joinedData.map(lambda row: row[1][1].split('\")[1] + "," +
str(row[0])).distinct() for line in distinctOrdersDate.collect(): print(line)
Step 8 : Similar to word count, generate (date, 1) record for each row. newLineTuple =
distinctOrdersDate.map(lambda line: (line.split(",")[0], 1))
Step 9 : Do the count for each key(date), to get total order per date. totalOrdersPerDate =
newLineTuple.reduceByKey(lambda a, b: a + b}
#print results
for line in totalOrdersPerDate.collect():
print(line)
step 10 : Sort the results by date sortedData=totalOrdersPerDate.sortByKey().collect()
#print results
for line in sortedData:
print(line)

NEW QUESTION: 30
CORRECT TEXT
Problem Scenario 10 : 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 database named hadoopexam and then create a table named departments in it, with following fields. department_id int, department_name string e.g. location should be hdfs://quickstart.cloudera:8020/user/hive/warehouse/hadoopexam.db/departments
2. Please import data in existing table created above from retaidb.departments into hive table hadoopexam.departments.
3. Please import data in a non-existing table, means while importing create hive table named hadoopexam.departments_new
Answer:
See the explanation for Step by Step Solution and configuration.

Explanation:
Solution :
Step 1 : Go to hive interface and create database.
hive
create database hadoopexam;
Step 2. Use the database created in above step and then create table in it. use hadoopexam; show tables;
Step 3 : Create table in it.
create table departments (department_id int, department_name string);
show tables;
desc departments;
desc formatted departments;
Step 4 : Please check following directory must not exist else it will give error, hdfs dfs -ls /user/cloudera/departments
If directory already exists, make sure it is not useful and than delete the same.
This is the staging directory where Sqoop store the intermediate data before pushing in hive table.
hadoop fs -rm -R departments
Step 5 : Now import data in existing table
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 hadoopexam.departments
Step 6 : Check whether data has been loaded or not.
hive;
use hadoopexam;
show tables;
select" from departments;
desc formatted departments;
Step 7 : Import data in non-existing tables in hive and create table while importing.
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 hadoopexam.departments_new \
-create-hive-table
Step 8 : Check-whether data has been loaded or not.
hive;
use hadoopexam;
show tables;
select" from departments_new;
desc formatted departments_new;

NEW QUESTION: 31
CORRECT TEXT
Problem Scenario 79 : You have been given MySQL DB with following details.
user=retail_dba
password=cloudera
database=retail_db
table=retail_db.orders
table=retail_db.order_items
jdbc URL = jdbc:mysql://quickstart:3306/retail_db
Columns of products table : (product_id | product categoryid | product_name | product_description | product_prtce | product_image)
Please accomplish following activities.
1 . Copy "retaildb.products" table to hdfs in a directory p93_products
2 . Filter out all the empty prices
3 . Sort all the products based on price in both ascending as well as descending order.
4 . Sort all the products based on price as well as product_id in descending order.
5 . Use the below functions to do data ordering or ranking and fetch top 10 elements top() takeOrdered() sortByKey()
Answer:
See the explanation for Step by Step Solution and configuration.
Explanation:
Solution :
Step 1 : Import Single table .
sqoop import --connect jdbc:mysql://quickstart:3306/retail_db -username=retail_dba -password=cloudera -table=products -target-dir=p93_products -m 1
Note : Please check you dont have space between before or after '=' sign. Sqoop uses the MapReduce framework to copy data from RDBMS to hdfs
Step 2 : Step 2 : Read the data from one of the partition, created using above command,
hadoop fs -cat p93_products/part-m-00000

Step 3 : Load this directory as RDD using Spark and Python (Open pyspark terminal and do following). productsRDD = sc.textFile("p93_products")
Step 4 : Filter empty prices, if exists
#filter out empty prices lines
nonemptyjines = productsRDD.filter(lambda x: len(x.split(",")[4]) > 0)
Step 5 : Now sort data based on product_price in order.
sortedPriceProducts=nonempty_lines.map(lambdaline:(float(line.split(",")[4]),line.split(",")[2])).sortByKey()
for line in sortedPriceProducts.collect(): print(line)
Step 6 : Now sort data based on product_price in descending order.
sortedPriceProducts=nonempty_lines.map(lambda line:
(float(line.split(",")[4]),line.split(",")[2])).sortByKey(False)
for line in sortedPriceProducts.collect(): print(line)
Step 7 : Get highest price products name.
sortedPriceProducts=nonemptyJines.map(lambda line : (float(line.split(",")[4]),line-split(,,,,,)[2]))-sortByKey(False).take(1) print(sortedPriceProducts)
Step 8 : Now sort data based on product_price as well as product_id in descending order.
#Dont forget to cast string #Tuple as key ((price,id),name)
sortedPriceProducts=nonemptyJines.map(lambda line : ((float(line
print(sortedPriceProducts)
Step 9 : Now sort data based on product_price as well as product_id in descending order, using top() function.
#Dont forget to cast string
#Tuple as key ((price,id),name)
sortedPriceProducts=nonemptyJines.map(lambda line: ((float(line.s^^
print(sortedPriceProducts)
Step 10 : Now sort data based on product_price as ascending and product_id in ascending order, using takeOrdered{) function.
#Dont forget to cast string
#Tuple as key ((price,id),name) sortedPriceProducts=nonemptyJines.map(lambda line: ((float(line.split(","}[4]},int(line.split(","}[0]}},line.split(","}[2]}}.takeOrdered(10, lambda tuple : (tuple[0][0],tuple[0][1]))
Step 11 : Now sort data based on product_price as descending and product_id in ascending order, using takeOrdered() function.
# Dont forget to cast string
# Tuple as key ((price,id},name)
# Using minus(-) parameter can help you to make descending ordering , only for numeric value.
sortedPrlceProducts=nonemptylines.map(lambda line:
((float(line.split(","}[4]},int(line.split(","}[0]}},line.split(","}[2]}}.takeOrdered(10, lambda tuple : (-tuple[0][0],tuple[0][1]}}

NEW QUESTION: 32
CORRECT TEXT
Problem Scenario 45 : You have been given 2 files , with the content as given Below
(spark12/technology.txt)
(spark12/salary.txt)
(spark12/technology.txt)
first,last,technology
Amit,Jain,java
Lokesh,kumar,unix
Mithun,kale,spark
Rajni,vekat,hadoop
Rahul,Yadav,scala
(spark12/salary.txt)
first,last,salary
Amit,Jain,100000
Lokesh,kumar,95000
Mithun,kale,150000
Rajni,vekat,154000
Rahul,Yadav,120000
Write a Spark program, which will join the data based on first and last name and save the joined results in following format, first Last.technology.salary
Answer:
See the explanation for Step by Step Solution and configuration.
Explanation:
Solution :
Step 1 : Create 2 files first using Hue in hdfs.
Step 2 : Load all file as an RDD
val technology = sc.textFile(Msparkl2/technology.txt").map(e => e.splitf',")) val salary = sc.textFile("spark12/salary.txt").map(e => e.split("."))
Step 3 : Now create Key.value pair of data and join them.
val joined = technology.map(e=>((e(0),e(1)),e(2))).join(salary.map(e=>((e(0),e(1)),e(2))))

Step 4 : Save the results in a text file as below.
joined.repartition(1).saveAsTextFile("spark12/multiColumn Joined.txt")

NEW QUESTION: 33
CORRECT TEXT
Problem Scenario 36 : You have been given a file named spark8/data.csv (type,name).
data.csv
1 ,Lokesh
2 ,Bhupesh
2 ,Amit
2 ,Ratan
2 ,Dinesh
1 ,Pavan
1 ,Tejas
2 ,Sheela
1 ,Kumar
1 ,Venkat
1. Load this file from hdfs and save it back as (id, (all names of same type)) in results directory. However, make sure while saving it should be
Answer:
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 data.csv file from hdfs and create PairRDDs
val name = sc.textFile("spark8/data.csv")
val namePairRDD = name.map(x=> (x.split(",")(0),x.split(",")(1)))
Step 3 : Now swap namePairRDD RDD.
val swapped = namePairRDD.map(item => item.swap)
Step 4 : Now combine the rdd by key.
val combinedOutput = namePairRDD.combineByKey(List(_), (x:List[String], y:String) => y ::
x, (x:List[String], y:List[String]) => x ::: y)
Step 5 : Save the output as a Text file and output must be written in a single file.
:ombinedOutput.repartition(1).saveAsTextFile("spark8/result.txt")

NEW QUESTION: 34
CORRECT TEXT
Problem Scenario 5 : You have been given following mysql database details.
user=retail_dba

password=cloudera
database=retail_db
jdbc URL = jdbc:mysql://quickstart:3306/retail_db
Please accomplish following activities.
1. List all the tables using sqoop command from retail_db
2. Write simple sqoop eval command to check whether you have permission to read database tables or not.
3 . Import all the tables as avro files in /user/hive/warehouse/retail cca174.db
4 . Import departments table as a text file in /user/cloudera/departments.
Answer:
See the explanation for Step by Step Solution and configuration.
Explanation:
Solution:
Step 1 : List tables using sqoop
sqoop list-tables --connect jdbc:mysql://quickstart:330G/retail_db --username retail dba -password cloudera
Step 2 : Eval command, just run a count query on one of the table.
sqoop eval \
--connect jdbc:mysql://quickstart:3306/retail_db \
-username retail_dba \
-password cloudera \
--query "select count(1) from ordeMtems"
Step 3 : Import all the tables as avro file.
sqoop import-all-tables \
-connect jdbc:mysql://quickstart:3306/retail_db \
-username=retail_dba \
-password=cloudera \
-as-avrodatafile \
-warehouse-dir=/user/hive/warehouse/retail stage.db \
-ml
Step 4 : Import departments table as a text file in /user/cloudera/departments sqoop import \
-connect jdbc:mysql://quickstart:3306/retail_db \
-username=retail_dba \
-password=cloudera \
-table departments \
-as-textfile \
-target-dir=/user/cloudera/departments
Step 5 : Verify the imported data.
hdfs dfs -Is /user/cloudera/departments
hdfs dfs -Is /user/hive/warehouse/retailstage.db

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