Cloudera.CCA175.v2017-08-25.q50

Exam Code:	CCA175
Exam Name:	CCA Spark and Hadoop Developer Exam
Certification Provider:	Cloudera
Free Question Number:	50
Version:	v2017-08-25
# of views:	1681
# of Questions views:	38576
https://www.freecram.net/torrent/Cloudera.CCA175.v2017-08-25.q50.html	

NEW QUESTION: 1
CORRECT TEXT
Problem Scenario 12 : 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_new (department_id int(11), department_name varchar(45), created_date T1MESTAMP DEFAULT NOW());
2 . Now isert records from departments table to departments_new
3 . Now import data from departments_new table to hdfs.
4 . Insert following 5 records in departmentsnew table. Insert into departments_new values(110, "Civil" , null); Insert into departments_new values(111, "Mechanical" , null); Insert into departments_new values(112, "Automobile" , null); Insert into departments_new values(113, "Pharma" , null);
Insert into departments_new values(114, "Social Engineering" , null);
5. Now do the incremental import based on created_date column.
Answer:
See the explanation for Step by Step Solution and configuration.
Explanation:
Solution :
Step 1 : Login to musql db
mysql --user=retail_dba -password=cloudera
show databases;

use retail db; show tables;
Step 2 : Create a table as given in problem statement.
CREATE table departments_new (department_id int(11), department_name varchar(45), createddate T1MESTAMP DEFAULT NOW()); show tables;
Step 3 : isert records from departments table to departments_new insert into departments_new select a.", null from departments a;
Step 4 : Import data from departments new table to hdfs.
sqoop import \
-connect jdbc:mysql://quickstart:330G/retail_db \
~ username=retail_dba \
-password=cloudera \
-table departments_new\
--target-dir /user/cloudera/departments_new \
--split-by departments
Stpe 5 : Check the imported data.
hdfs dfs -cat /user/cloudera/departmentsnew/part"
Step 6 : Insert following 5 records in departmentsnew table.
Insert into departments_new values(110, "Civil" , null);
Insert into departments_new values(111, "Mechanical" , null);
Insert into departments_new values(112, "Automobile" , null);
Insert into departments_new values(113, "Pharma" , null);
Insert into departments_new values(114, "Social Engineering" , null);
commit;
Stpe 7 : Import incremetal data based on created_date column.
sqoop import \
-connect jdbc:mysql://quickstart:330G/retail_db \
-username=retail_dba \
-password=cloudera \
--table departments_new\
-target-dir /user/cloudera/departments_new \
-append \
-check-column created_date \
-incremental lastmodified \
-split-by departments \
-last-value "2016-01-30 12:07:37.0"
Step 8 : Check the imported value.
hdfs dfs -cat /user/cloudera/departmentsnew/part"

NEW QUESTION: 2
CORRECT TEXT

Problem Scenario 92 : You have been given a spark scala application, which is bundled in jar named hadoopexam.jar.
Your application class name is com.hadoopexam.MyTask
You want that while submitting your application should launch a driver on one of the cluster node.
Please complete the following command to submit the application.
spark-submit XXX -master yarn \
YYY SSPARK HOME/lib/hadoopexam.jar 10
Answer:
See the explanation for Step by Step Solution and configuration.
Explanation:
Solution
XXX: -class com.hadoopexam.MyTask
YYY : --deploy-mode cluster

NEW QUESTION: 3
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: 4
CORRECT TEXT
Problem Scenario 33 : You have given a files as below.
spark5/EmployeeName.csv (id,name)
spark5/EmployeeSalary.csv (id,salary)
Data is given below:
EmployeeName.csv
E01,Lokesh
E02,Bhupesh
E03,Amit
E04,Ratan
E05,Dinesh
E06,Pavan
E07,Tejas
E08,Sheela
E09,Kumar
E10,Venkat
EmployeeSalary.csv
E01,50000
E02,50000
E03,45000
E04,45000
E05,50000
E06,45000
E07,50000
E08,10000
E09,10000
E10,10000
Now write a Spark code in scala which will load these two tiles from hdfs and join the same, and produce the (name.salary) values.
And save the data in multiple tile group by salary (Means each file will have name of employees with same salary). Make sure file name include salary as well.
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 EmployeeName.csv file from hdfs and create PairRDDs
val name = sc.textFile("spark5/EmployeeName.csv")
val namePairRDD = name.map(x=> (x.split(",")(0),x.split('V')(1)))
Step 3 : Load EmployeeSalary.csv file from hdfs and create PairRDDs
val salary = sc.textFile("spark5/EmployeeSalary.csv")
val salaryPairRDD = salary.map(x=> (x.split(",")(0),x.split(",")(1)))
Step 4 : Join all pairRDDS
val joined = namePairRDD.join(salaryPairRDD}
Step 5 : Remove key from RDD and Salary as a Key. val keyRemoved = joined.values
Step 6 : Now swap filtered RDD.
val swapped = keyRemoved.map(item => item.swap)
Step 7 : Now groupBy keys (It will generate key and value array) val grpByKey = swapped.groupByKey().collect()
Step 8 : Now create RDD for values collection
val rddByKey = grpByKey.map{case (k,v) => k->sc.makeRDD(v.toSeq)}
Step 9 : Save the output as a Text file.
rddByKey.foreach{ case (k,rdd) => rdd.saveAsTextFile("spark5/Employee"+k)}

NEW QUESTION: 5
CORRECT TEXT
Problem Scenario 88 : You have been given below three files
product.csv (Create this file in hdfs)
productID,productCode,name,quantity,price,supplierid
1001,PEN,Pen Red,5000,1.23,501
1002,PEN,Pen Blue,8000,1.25,501
1003,PEN,Pen Black,2000,1.25,501
1004,PEC,Pencil 2B,10000,0.48,502
1005,PEC,Pencil 2H,8000,0.49,502
1006,PEC,Pencil HB,0,9999.99,502
2001,PEC,Pencil 3B,500,0.52,501
2002,PEC,Pencil 4B,200,0.62,501
2003,PEC,Pencil 5B,100,0.73,501
2004,PEC,Pencil 6B,500,0.47,502
supplier.csv
supplierid,name,phone
501,ABC Traders,88881111

502,XYZ Company,88882222
503,QQ Corp,88883333
products_suppliers.csv
productID,supplierID
2001,501
2002,501
2003,501
2004,502
2001,503
Now accomplish all the queries given in solution.
1. It is possible that, same product can be supplied by multiple supplier. Now find each product, its price according to each supplier.
2. Find all the supllier name, who are supplying 'Pencil 3B'
3. Find all the products , which are supplied by ABC Traders.
Answer:
See the explanation for Step by Step Solution and configuration.
Explanation:
Solution :
Step 1 : It is possible that, same product can be supplied by multiple supplier. Now find each product, its price according to each supplier.
val results = sqlContext.sql(.....SELECT products.name AS Product Name', price, suppliers.name AS Supplier Name'
FROM products_suppliers
JOIN products ON products_suppliers.productID = products.productID JOIN suppliers ON products_suppliers.supplierID = suppliers.supplierID null t results.show()
Step 2 : Find all the supllier name, who are supplying 'Pencil 3B'
val results = sqlContext.sql(.....SELECT p.name AS 'Product Name", s.name AS "Supplier Name'
FROM products_suppliers AS ps
JOIN products AS p ON ps.productID = p.productID
JOIN suppliers AS s ON ps.supplierID = s.supplierID
WHERE p.name = 'Pencil 3B"",M)
results.show()
Step 3 : Find all the products , which are supplied by ABC Traders.
val results = sqlContext.sql(.....SELECT p.name AS 'Product Name", s.name AS "Supplier Name'
FROM products AS p, products_suppliers AS ps, suppliers AS s WHERE p.productID = ps.productID AND ps.supplierID = s.supplierID
AND s.name = 'ABC Traders".....)
results. show()

NEW QUESTION: 6
CORRECT TEXT
Problem Scenario GG : You have been given below code snippet.
val a = sc.parallelize(List("dog", "tiger", "lion", "cat", "spider", "eagle"), 2) val b = a.keyBy(_.length) val c = sc.parallelize(List("ant", "falcon", "squid"), 2) val d = c.keyBy(.length)
operation 1
Write a correct code snippet for operationl which will produce desired output, shown below.
Array[(Int, String)] = Array((4,lion))
Answer:
See the explanation for Step by Step Solution and configuration.
Explanation:
Solution :
b.subtractByKey(d).collect
subtractByKey [Pair] : Very similar to subtract, but instead of supplying a function, the key-component of each pair will be automatically used as criterion for removing items from the first RDD.

NEW QUESTION: 7
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: 8

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: 9

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: 10
Answer:
See the explanation for Step by Step Solution and configuration.
Explanation:
Solution :
Step 1 : Drop all the tables, which we have created in previous problems. Before implementing the solution.
Login to hive and execute following command.
show tables;
drop table categories;
drop table customers;
drop table departments;
drop table employee;
drop table ordeMtems;
drop table orders;
drop table products;
show tables;
Check warehouse directory. hdfs dfs -Is /user/hive/warehouse
Step 2 : Now we have cleaned database. Import entire retail db with all the required parameters as problem statement is asking.
sqoop import-all-tables \
-m3\
-connect jdbc:mysql://quickstart:3306/retail_db \

--username=retail_dba \
-password=cloudera \
-hive-import \
--hive-overwrite \
-create-hive-table \
--compress \
--compression-codec org.apache.hadoop.io.compress.SnappyCodec \
--outdir java_output
Step 3 : Verify the work is accomplished or not.
a. Go to hive and check all the tables hive
show tables;
select count(1) from customers;
b. Check the-warehouse directory and number of partitions,
hdfs dfs -Is /user/hive/warehouse
hdfs dfs -Is /user/hive/warehouse/categories
c. Check the output Java directory.
Is -Itr java_output/

NEW QUESTION: 11
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 "retail_db.order_items" 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)
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=order_items --target -dir=p90 ordeMtems --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 p90_order_items/part-m-00000
Step 3 : In pyspark, get the total revenue across all days and orders. entire TableRDD = sc.textFile("p90_order_items")
#Cast string to float
extractedRevenueColumn = entireTableRDD.map(lambda line: float(line.split(",")[4]))
Step 4 : Verify extracted data
for revenue in extractedRevenueColumn.collect():
print revenue
#use reduce'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

NEW QUESTION: 12
CORRECT TEXT
Problem Scenario 73 : You have been given data in json format as below.
{"first_name":"Ankit", "last_name":"Jain"}
{"first_name":"Amir", "last_name":"Khan"}
{"first_name":"Rajesh", "last_name":"Khanna"}
{"first_name":"Priynka", "last_name":"Chopra"}
{"first_name":"Kareena", "last_name":"Kapoor"}
{"first_name":"Lokesh", "last_name":"Yadav"}
Do the following activity
1 . create employee.json file locally.
2 . Load this file on hdfs
3 . Register this data as a temp table in Spark using Python.
4 . Write select query and print this data.
5 . Now save back this selected data in json format.
Answer:
See the explanation for Step by Step Solution and configuration.
Explanation:

Solution :
Step 1 : create employee.json tile locally.
vi employee.json (press insert) past the content.
Step 2 : Upload this tile to hdfs, default location hadoop fs -put employee.json
Step 3 : Write spark script
#lmport SQLContext
from pyspark import SQLContext
# Create instance of SQLContext sqIContext = SQLContext(sc)
# Load json file
employee = sqlContext.jsonFile("employee.json")
# Register RDD as a temp table employee.registerTempTablef'EmployeeTab"}
# Select data from Employee table
employeelnfo = sqlContext.sql("select * from EmployeeTab"}
#lterate data and print
for row in employeelnfo.collect():
print(row)
Step 4 : Write dataas a Text file employeelnfo.toJSON().saveAsTextFile("employeeJson1")
Step 5: Check whether data has been created or not hadoop fs -cat employeeJsonl/part"

NEW QUESTION: 13
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: 14
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: 15
CORRECT TEXT
Problem Scenario 14 : You have been given following mysql database details as well as other info.
user=retail_dba
password=cloudera
database=retail_db
jdbc URL = jdbc:mysql://quickstart:3306/retail_db
Please accomplish following activities.
1. Create a csv file named updated_departments.csv with the following contents in local file system.
updated_departments.csv
2 ,fitness
3 ,footwear
1 2,fathematics
1 3,fcience
1 4,engineering
1 000,management
2. Upload this csv file to hdfs filesystem,
3. Now export this data from hdfs to mysql retaildb.departments table. During upload make sure existing department will just updated and new departments needs to be inserted.
4. Now update updated_departments.csv file with below content.
2 ,Fitness
3 ,Footwear
1 2,Fathematics
1 3,Science
1 4,Engineering
1 000,Management
2 000,Quality Check
5. Now upload this file to hdfs.
6. Now export this data from hdfs to mysql retail_db.departments table. During upload make sure existing department will just updated and no new departments needs to be inserted.

Answer:
See the explanation for Step by Step Solution and configuration.
Explanation:
Solution :
Step 1 : Create a csv tile named updateddepartments.csv with give content.
Step 2 : Now upload this tile to HDFS.
Create a directory called newdata.
hdfs dfs -mkdir new_data
hdfs dfs -put updated_departments.csv newdata/
Step 3 : Check whether tile is uploaded or not. hdfs dfs -ls new_data
Step 4 : Export this file to departments table using sqoop.
sqoop export --connect jdbc:mysql://quickstart:3306/retail_db \
-username retail_dba \
--password cloudera \
-table departments \
--export-dir new_data \
-batch \
-m 1 \
-update-key department_id \
-update-mode allowinsert
Step 5 : Check whether required data upsert is done or not. mysql --user=retail_dba -password=cloudera show databases; use retail_db;
show tables;
select" from departments;
Step 6 : Update updated_departments.csv file.
Step 7 : Override the existing file in hdfs.
hdfs dfs -put updated_departments.csv newdata/
Step 8 : Now do the Sqoop export as per the requirement.
sqoop export --connect jdbc:mysql://quickstart:3306/retail_db \
-username retail_dba\
--password cloudera \
--table departments \
--export-dir new_data \
--batch \
-m 1 \
--update-key-department_id \
-update-mode updateonly
Step 9 : Check whether required data update is done or not. mysql --user=retail_dba -password=cloudera show databases; use retail db;
show tables;
select" from departments;

NEW QUESTION: 16
CORRECT TEXT
Problem Scenario 86 : In Continuation of previous question, please accomplish following activities.
1 . Select Maximum, minimum, average , Standard Deviation, and total quantity.
2 . Select minimum and maximum price for each product code.
3. Select Maximum, minimum, average , Standard Deviation, and total quantity for each product code, hwoever make sure Average and Standard deviation will have maximum two decimal values.
4. Select all the product code and average price only where product count is more than or equal to 3.
5. Select maximum, minimum , average and total of all the products for each code. Also produce the same across all the products.
Answer:
See the explanation for Step by Step Solution and configuration.
Explanation:
Solution :
Step 1 : Select Maximum, minimum, average , Standard Deviation, and total quantity.
val results = sqlContext.sql('.....SELECT MAX(price) AS MAX , MIN(price) AS MIN , AVG(price) AS Average, STD(price) AS STD, SUM(quantity) AS total_products FROM products.....) results. showQ
Step 2 : Select minimum and maximum price for each product code.
val results = sqlContext.sql(.....SELECT code, MAX(price) AS Highest Price', MIN(price) AS Lowest Price'
FROM products GROUP BY code.....)
results. showQ
Step 3 : Select Maximum, minimum, average , Standard Deviation, and total quantity for each product code, hwoever make sure Average and Standard deviation will have maximum two decimal values.
val results = sqlContext.sql(.....SELECT code, MAX(price), MIN(price), CAST(AVG(price} AS DECIMAL(7,2)) AS Average', CAST(STD(price) AS DECIMAL(7,2)) AS 'Std Dev\ SUM(quantity) FROM products
GROUP BY code.....)
results. showQ
Step 4 : Select all the product code and average price only where product count is more than or equal to 3.
val results = sqlContext.sql(.....SELECT code AS Product Code',
COUNTf) AS Count',
CAST(AVG(price) AS DECIMAL(7,2)) AS Average' FROM products GROUP BY code HAVING Count >=3"M") results. showQ

Step 5 : Select maximum, minimum , average and total of all the products for each code. Also produce the same across all the products.
val results = sqlContext.sql("""SELECT
code,
MAX(price),
MIN(pnce),
CAST(AVG(price) AS DECIMAL(7,2)) AS Average',
SUM(quantity)-
FROM products
GROUP BY code
WITH ROLLUP""")
results. show()

NEW QUESTION: 17
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, (lterable[String], lterable[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: 18
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/retail_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/retail_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: 19
CORRECT TEXT
Problem Scenario 23 : You have been given log generating service as below.
Start_logs (It will generate continuous logs)
Tail_logs (You can check , what logs are being generated)
Stop_logs (It will stop the log service)
Path where logs are generated using above service : /opt/gen_logs/logs/access.log

Now write a flume configuration file named flume3.conf , using that configuration file dumps logs in HDFS file system in a directory called flumeflume3/%Y/%m/%d/%H/%M Means every minute new directory should be created). Please us the interceptors to provide timestamp information, if message header does not have header info.
And also note that you have to preserve existing timestamp, if message contains it. Flume channel should have following property as well. After every 100 message it should be committed, use non-durable/faster channel and it should be able to hold maximum 1000 events.

Answer:

See the explanation for Step by Step Solution and configuration.

Explanation:

Solution :

Step 1 : Create flume configuration file, with below configuration for source, sink and channel.

```
#Define source , sink , channel and agent,
agent1 .sources = source1
agent1 .sinks = sink1
agent1.channels = channel1
# Describe/configure source1
agent1 .sources.source1.type = exec
agentl.sources.source1.command = tail -F /opt/gen logs/logs/access.log
#Define interceptors
agent1 .sources.source1.interceptors=i1
agent1 .sources.source1.interceptors.i1.type=timestamp
agent1 .sources.source1.interceptors.i1.preserveExisting=true
## Describe sink1
agent1 .sinks.sink1.channel = memory-channel
agent1 .sinks.sink1.type = hdfs
agent1 .sinks.sink1.hdfs.path = flume3/%Y/%m/%d/%H/%M
agent1 .sinks.sjnkl.hdfs.fileType = Data Stream
# Now we need to define channel1 property.
agent1.channels.channel1.type = memory
agent1.channels.channel1.capacity = 1000
agent1.channels.channel1.transactionCapacity = 100
# Bind the source and sink to the channel
Agent1.sources.source1.channels = channel1
agent1.sinks.sink1.channel = channel1
```

Step 2 : Run below command which will use this configuration file and append data in hdfs.

Start log service using : start_logs

Start flume service:

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

/home/cloudera/flumeconf/flume3.conf -Dflume.root.logger=DEBUG,INFO,console -name agent1
Wait for few mins and than stop log service.
stop logs

NEW QUESTION: 20
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 : Load tile as an RDD
val rawlines = sc.textFile("spark10/sales.txt")
Step 3 : Create a case class, which can represent its column fileds. case class Employee(dep: String, des: String, cost: Double, state: String)
Step 4 : Split the data and create RDD of all Employee objects.
val employees = rawlines.map(_.split(",")).map(row=>Employee(row(0), row{1), row{2).toDouble, row{3)))
Step 5 : Create a row as we needed. All group by fields as a key and value as a count for each employee as well as its cost, val keyVals = employees.map(em => ((em.dep, em.des, em.state), (1 , em.cost)))
Step 6 : Group by all the records using reduceByKey method as we want summation as well. For number of employees and their total cost, val results = keyVals.reduceByKey{ (a,b) => (a._1 + b._1, a._2 + b._2)} // (a.count + b.count, a.cost + b.cost)}
Step 7 : Save the results in a text file as below.
results.repartition(1).saveAsTextFile("spark10/group.txt")

NEW QUESTION: 21
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: 22
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: 23
CORRECT TEXT
Problem Scenario 24 : You have been given below comma separated employee information.
Data Set:
name,salary,sex,age
alok,100000,male,29
jatin,105000,male,32
yogesh,134000,male,39
ragini,112000,female,35
jyotsana,129000,female,39
valmiki,123000,male,29
Requirements:
Use the netcat service on port 44444, and nc above data line by line. Please do the following activities.
1. Create a flume conf file using fastest channel, which write data in hive warehouse directory, in a table called flumemaleemployee (Create hive table as well tor given data).
2. While importing, make sure only male employee data is stored.
Answer:
See the explanation for Step by Step Solution and configuration.
Explanation:
Step 1 : Create hive table for flumeemployee.'
CREATE TABLE flumemaleemployee
(
name string,
salary int,
sex string,
age int
)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ',';
step 2 : Create flume configuration file, with below configuration for source, sink and channel and save it in flume4.conf.
#Define source , sink, channel and agent.
agent1 .sources = source1

agent1 .sinks = sink1
agent1 .channels = channel1
Describe/configure source1
agent1 .sources.source1.type = netcat
agent1 .sources.source1.bind = 127.0.0.1
agent1.sources.sourcel.port = 44444
#Define interceptors
agent1.sources.source1.interceptors=il
agent1 .sources.source1.interceptors.i1.type=regex_filter
agent1 .sources.source1.interceptors.i1.regex=female
agent1 .sources.source1.interceptors.i1.excludeEvents=true
Describe sink1
agent1 .sinks, sinkl.channel = memory-channel
agent1.sinks.sink1.type = hdfs
agent1 .sinks, sinkl. hdfs. path = /user/hive/warehouse/flumemaleemployee hdfs-agent.sinks.hdfs-write.hdfs.writeFormat=Text agentl .sinks.sink1.hdfs.fileType = Data Stream
# Now we need to define channel1 property.
agent1.channels.channel1.type = memory
agent1.channels.channell.capacity = 1000
agent1.channels.channel1.transactionCapacity = 100
# Bind the source and sink to the channel
agent1 .sources.source1.channels = channel1
agent1 .sinks.sink1.channel = channel1
step 3 : Run below command which will use this configuration file and append data in hdfs. Start flume service:
flume-ng agent -conf /home/cloudera/flumeconf -conf-file /home/cloudera/flumeconf/flume4.conf --name agentl
Step 4 : Open another terminal and use the netcat service, nc localhost 44444
Step 5 : Enter data line by line.
alok,100000,male,29
jatin,105000,male,32
yogesh,134000,male,39
ragini,112000,female,35
jyotsana,129000,female,39
valmiki.123000.male.29
Step 6 : Open hue and check the data is available in hive table or not.
Step 7 : Stop flume service by pressing ctrl+c
Step 8 : Calculate average salary on hive table using below query. You can use either hive command line tool or hue. select avg(salary) from flumeemployee;

NEW QUESTION: 24
CORRECT TEXT
Problem Scenario 82 : You have been given table in Hive with following structure (Which you have created in previous exercise).
productid int code string name string quantity int price float
Using SparkSQL accomplish following activities.
1 . Select all the products name and quantity having quantity <= 2000
2 . Select name and price of the product having code as 'PEN'
3 . Select all the products, which name starts with PENCIL
4 . Select all products which "name" begins with 'P\ followed by any two characters, followed by space, followed by zero or more characters
Answer:
See the explanation for Step by Step Solution and configuration.
Explanation:
Solution :
Step 1 : Copy following tile (Mandatory Step in Cloudera QuickVM) if you have not done it.
sudo su root
cp /usr/lib/hive/conf/hive-site.xml /usr/lib/sparkVconf/
Step 2 : Now start spark-shell
Step 3 ; Select all the products name and quantity having quantity <= 2000 val results = sqlContext.sql(.....SELECT name, quantity FROM products WHERE quantity
< = 2000.....)
results.showQ
Step 4 : Select name and price of the product having code as 'PEN'
val results = sqlContext.sql(.....SELECT name, price FROM products WHERE code = 'PEN.....)
results. showQ
Step 5 : Select all the products , which name starts with PENCIL
val results = sqlContext.sql(.....SELECT name, price FROM products WHERE upper(name) LIKE 'PENCIL%.....} results. showQ
Step 6 : select all products which "name" begins with 'P', followed by any two characters, followed by space, followed byzero or more characters
-- "name" begins with 'P', followed by any two characters,
- followed by space, followed by zero or more characters
val results = sqlContext.sql(.....SELECT name, price FROM products WHERE name LIKE 'P_ %.....)
results. show()

NEW QUESTION: 25
CORRECT TEXT

Problem Scenario 89 : You have been given below patient data in csv format,
patientID,name,dateOfBirth,lastVisitDate
1001,Ah Teck,1991-12-31,2012-01-20
1002,Kumar,2011-10-29,2012-09-20
1003,Ali,2011-01-30,2012-10-21
Accomplish following activities.
1 . Find all the patients whose lastVisitDate between current time and '2012-09-15'
2 . Find all the patients who born in 2011
3 . Find all the patients age
4 . List patients whose last visited more than 60 days ago
5 . Select patients 18 years old or younger
Answer:
See the explanation for Step by Step Solution and configuration.
Explanation:
Solution :
Step 1:
hdfs dfs -mkdir sparksql3
hdfs dfs -put patients.csv sparksql3/
Step 2 : Now in spark shell
// SQLContext entry point for working with structured data
val sqlContext = neworg.apache.spark.sql.SQLContext(sc)
// this is used to implicitly convert an RDD to a DataFrame.
import sqlContext.implicits._
// Import Spark SQL data types and Row.
import org.apache.spark.sql._
// load the data into a new RDD
val patients = sc.textFilef'sparksqlS/patients.csv")
// Return the first element in this RDD
patients.first()
//define the schema using a case class
case class Patient(patientid: Integer, name: String, dateOfBirth:String , lastVisitDate:
String)
// create an RDD of Product objects
val patRDD = patients.map(_.split(M,M)).map(p => Patient(p(0).tolnt,p(1),p(2),p(3)))
patRDD.first() patRDD.count(}
// change RDD of Product objects to a DataFrame val patDF = patRDD.toDF()
// register the DataFrame as a temp table patDF.registerTempTable("patients"}
// Select data from table
val results = sqlContext.sql(.....SELECT* FROM patients '.....)
// display dataframe in a tabular format
results.show()

//Find all the patients whose lastVisitDate between current time and '2012-09-15' val results = sqlContext.sql(.....SELECT * FROM patients WHERE TO_DATE(CAST(UNIX_TIMESTAMP(lastVisitDate, 'yyyy-MM-dd') AS TIMESTAMP)) BETWEEN '2012-09-15' AND current_timestamp() ORDER BY lastVisitDate.....) results.showQ
/.Find all the patients who born in 2011
val results = sqlContext.sql(.....SELECT * FROM patients WHERE YEAR(TO_DATE(CAST(UNIXJTIMESTAMP(dateOfBirth, 'yyyy-MM-dd') AS TIMESTAMP))) = 2011 .....)
results. show()
//Find all the patients age
val results = sqlContext.sql(.....SELECT name, dateOfBirth, datediff(current_date(), TO_DATE(CAST(UNIX_TIMESTAMP(dateOfBirth, 'yyyy-MM-dd') AS TIMESTAMP}}}/365 AS age
FROM patients
Mini >
results.show()
//List patients whose last visited more than 60 days ago
-- List patients whose last visited more than 60 days ago
val results = sqlContext.sql(.....SELECT name, lastVisitDate FROM patients WHERE datediff(current_date(), TO_DATE(CAST(UNIX_TIMESTAMP[lastVisitDate, 'yyyy-MM-dd') AS T1MESTAMP))) > 60.....);
results. showQ;
-- Select patients 18 years old or younger
SELECT' FROM patients WHERE TO_DATE(CAST(UNIXJTIMESTAMP(dateOfBirth, 'yyyy-MM-dd') AS TIMESTAMP}) > DATE_SUB(current_date(),INTERVAL 18 YEAR); val results = sqlContext.sql(.....SELECT' FROM patients WHERE TO_DATE(CAST(UNIX_TIMESTAMP(dateOfBirth, 'yyyy-MM--dd') AS TIMESTAMP)) > DATE_SUB(current_date(), T8*365).....);
results. showQ;
val results = sqlContext.sql(.....SELECT DATE_SUB(current_date(), 18*365) FROM patients.....); results.show();

NEW QUESTION: 26
CORRECT TEXT
Problem Scenario 11 : You have been given following mysql database details as well as other info.
user=retail_dba
password=cloudera
database=retail_db
jdbc URL = jdbc:mysql://quickstart:3306/retail_db

Please accomplish following.
1. Import departments table in a directory called departments.
2. Once import is done, please insert following 5 records in departments mysql table.
Insert into departments(10, physics);
Insert into departments(11, Chemistry);
Insert into departments(12, Maths);
Insert into departments(13, Science);
Insert into departments(14, Engineering);
3. Now import only new inserted records and append to existring directory . which has been created in first step.

Answer:

See the explanation for Step by Step Solution and configuration.

Explanation:

Solution :

Step 1 : Clean already imported data. (In real exam, please make sure you dont delete data generated from previous exercise).

```
hadoop fs -rm -R departments
```

Step 2 : Import data in departments directory.

```
sqoop import \
--connect jdbc:mysql://quickstart:3306/retail_db \
--username=retail_dba \
-password=cloudera \
-table departments \
"target-dir/user/cloudera/departments
```

Step 3 : Insert the five records in departments table.

```
mysql -user=retail_dba --password=cloudera retail_db
```

Insert into departments values(10, "physics"); Insert into departments values(11, "Chemistry"); Insert into departments values(12, "Maths"); Insert into departments values(13, "Science"); Insert into departments values(14, "Engineering"); commit; select' from departments;

Step 4 : Get the maximum value of departments from last import, hdfs dfs -cat /user/cloudera/departments/part* that should be 7

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

```
sqoop import \
--connect "jdbc:mysql://quickstart.cloudera:330G/retail_db" \
~ username=retail_dba \
-password=cloudera \
-table departments \
--target-dir /user/cloudera/departments \
-append \
-check-column "department_id" \
```

-incremental append \
-last-value 7
Step 6 : Now check the result.
hdfs dfs -cat /user/cloudera/departments/part"

NEW QUESTION: 27
CORRECT TEXT
Problem Scenario 87 : You have been given below three files
product.csv (Create this file in hdfs)
productID,productCode,name,quantity,price,supplierid
1 001,PEN,Pen Red,5000,1.23,501
1 002,PEN,Pen Blue,8000,1.25,501
1003,PEN,Pen Black,2000,1.25,501
1004,PEC,Pencil 2B,10000,0.48,502
1005,PEC,Pencil 2H,8000,0.49,502
1006,PEC,Pencil HB,0,9999.99,502
2001,PEC,Pencil 3B,500,0.52,501
2002,PEC,Pencil 4B,200,0.62,501
2003,PEC,Pencil 5B,100,0.73,501
2004,PEC,Pencil 6B,500,0.47,502
supplier.csv
supplierid,name,phone
501,ABC Traders,88881111
502,XYZ Company,88882222
503,QQ Corp,88883333
products_suppliers.csv
productID,supplierID
2001,501
2002,501
2003,501
2004,502
2001,503
Now accomplish all the queries given in solution.
Select product, its price , its supplier name where product price is less than 0.6 using SparkSQL
Answer:
See the explanation for Step by Step Solution and configuration.
Explanation:
Solution :
Step 1:
hdfs dfs -mkdir sparksql2

hdfs dfs -put product.csv sparksq!2/
hdfs dfs -put supplier.csv sparksql2/
hdfs dfs -put products_suppliers.csv sparksql2/
Step 2 : Now in spark shell
// this Is used to Implicitly convert an RDD to a DataFrame.
import sqlContext.implicits._
// Import Spark SQL data types and Row.
import org.apache.spark.sql._
// load the data into a new RDD
val products = sc.textFile("sparksql2/product.csv")
val supplier = sc.textFileC'sparksq^supplier.csv")
val prdsup = sc.textFile("sparksql2/products_suppliers.csv"}
// Return the first element in this RDD
products.fi rst()
supplier.first{).
prdsup.first()
//define the schema using a case class
case class Product(productid: Integer, code: String, name: String, quantity:lnteger, price: Float, supplierid:Integer)
case class Suplier(supplierid: Integer, name: String, phone: String)
case class PRDSUP(productid: Integer.supplierid: Integer)
// create an RDD of Product objects
val prdRDD = products.map(_.split('\")).map(p => Product(p(0).tolnt,p(1),p(2),p(3).tolnt,p(4).toFloat,p(5).toint))
val supRDD = supplier.map(_.split(",")).map(p => Suplier(p(0).tolnt,p(1),p(2))) val prdsupRDD = prdsup.map(_.split(",")).map(p => PRDSUP(p(0).tolnt,p(1}.tolnt}}
prdRDD.first() prdRDD.count() supRDD.first() supRDD.count()
prdsupRDD.first() prdsupRDD.count(}
// change RDD of Product objects to a DataFrame
val prdDF = prdRDD.toDF()
val supDF = supRDD.toDF()
val prdsupDF = prdsupRDD.toDF()
// register the DataFrame as a temp table prdDF.registerTempTablef'products")
supDF.registerTempTablef'suppliers") prdsupDF.registerTempTablef'productssuppliers"}
//Select product, its price , its supplier name where product price is less than 0.6 val results = sqlContext.sql(.....SELECT products.name, price, suppliers.name as sup_name FROM products JOIN suppliers ON products.supplierlD= suppliers.supplierlD
WHERE price < 0.6.....]
results. show()

NEW QUESTION: 28

CORRECT TEXT
Problem Scenario 58 : You have been given below code snippet.
val a = sc.parallelize(List("dog", "tiger", "lion", "cat", "spider", "eagle"), 2) val b = a.keyBy(_.length) operation1
Write a correct code snippet for operationl which will produce desired output, shown below.
Array[(lnt, Seq[String])] = Array((4,ArrayBuffer(lion)), (6,ArrayBuffer(spider)), (3,ArrayBuffer(dog, cat)), (5,ArrayBuffer(tiger, eagle}}}
Answer:
See the explanation for Step by Step Solution and configuration.
Explanation:
Solution :
b.groupByKey.collect
groupByKey [Pair]
Very similar to groupBy, but instead of supplying a function, the key-component of each pair will automatically be presented to the partitioner.
Listing Variants
def groupByKeyQ: RDD[(K, lterable[V]}]
def groupByKey(numPartittons: lnt): RDD[(K, lterable[V])]
def groupByKey(partitioner: Partitioner): RDD[(K, lterable[V])]

NEW QUESTION: 29
CORRECT TEXT
Problem Scenario 28 : You need to implement near real time solutions for collecting information when submitted in file with below
Data
echo "IBM,100,20160104" >> /tmp/spooldir2/.bb.txt
echo "IBM,103,20160105" >> /tmp/spooldir2/.bb.txt
mv /tmp/spooldir2/.bb.txt /tmp/spooldir2/bb.txt
After few mins
echo "IBM,100.2,20160104" >> /tmp/spooldir2/.dr.txt
echo "IBM,103.1,20160105" >> /tmp/spooldir2/.dr.txt
mv /tmp/spooldir2/.dr.txt /tmp/spooldir2/dr.txt
You have been given below directory location (if not available than create it) /tmp/spooldir2 .
As soon as file committed in this directory that needs to be available in hdfs in /tmp/flume/primary as well as /tmp/flume/secondary location.
However, note that/tmp/flume/secondary is optional, if transaction failed which writes in this directory need not to be rollback.
Write a flume configuration file named flumeS.conf and use it to load data in hdfs with following additional properties .
1 . Spool /tmp/spooldir2 directory

2 . File prefix in hdfs sholuld be events
3 . File suffix should be .log
4 . If file is not committed 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/spooldir2

Step 2 : Create flume configuration file, with below configuration for source, sink and channel and save it in flume8.conf.

agent1 .sources = source1
agent1.sinks = sink1a sink1bagent1.channels = channel1a channel1b
agent1.sources.source1.channels = channel1a channel1b
agent1.sources.source1.selector.type = replicating
agent1.sources.source1.selector.optional = channel1b
agent1.sinks.sink1a.channel = channel1a
agent1 .sinks.sink1b.channel = channel1b
agent1.sources.source1.type = spooldir
agent1 .sources.sourcel.spoolDir = /tmp/spooldir2
agent1.sinks.sink1a.type = hdfs
agent1 .sinks, sink1a.hdfs. path = /tmp/flume/primary
agent1 .sinks.sink1a.hdfs.tilePrefix = events
agent1 .sinks.sink1a.hdfs.fileSuffix = .log
agent1 .sinks.sink1a.hdfs.fileType = Data Stream
agent1 .sinks.sink1b.type = hdfs
agent1 .sinks.sink1b.hdfs.path = /tmp/flume/secondary
agent1 .sinks.sink1b.hdfs.filePrefix = events
agent1.sinks.sink1b.hdfs.fileSuffix = .log
agent1 .sinks.sink1b.hdfs.fileType = Data Stream
agent1.channels.channel1a.type = file
agent1.channels.channel1b.type = memory

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/flume8.conf --name age

Step 5 : Open another terminal and create a file in /tmp/spooldir2/

echo "IBM,100,20160104" > /tmp/spooldir2/.bb.txt
echo "IBM,103,20160105" > /tmp/spooldir2/.bb.txt mv /tmp/spooldir2/.bb.txt /tmp/spooldir2/bb.txt

After few mins

echo "IBM.100.2,20160104" >/tmp/spooldir2/.dr.txt
echo "IBM,103.1,20160105" > /tmp/spooldir2/.dr.txt mv /tmp/spooldir2/.dr.txt /tmp/spooldir2/dr.txt

NEW QUESTION: 30
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: 31
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[Int] = Array(2, 4, 6, 8,10)
operation2
Output 2
Array[Int] = 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: Arrayjlnt] = 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: 32
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 -Is /user/hive/warehouse/retail_stage.db/products

NEW QUESTION: 33
CORRECT TEXT
Problem Scenario 16 : You have been given following mysql database details as well as other info.
user=retail_dba
password=cloudera
database=retail_db
jdbc URL = jdbc:mysql://quickstart:3306/retail_db
Please accomplish below assignment.
1. Create a table in hive as below.
create table departments_hive(department_id int, department_name string);
2. Now import data from mysql table departments to this hive table. Please make sure that data should be visible using below hive command, select" from departments_hive
Answer:
See the explanation for Step by Step Solution and configuration.
Explanation:
Solution :
Step 1 : Create hive table as said.

hive
show tables;
create table departments_hive(department_id int, department_name string);
Step 2 : The important here is, when we create a table without delimiter fields. Then default delimiter for hive is ^A (\001). Hence, while importing data we have to provide proper delimiter.
sqoop import \
-connect jdbc:mysql://quickstart:3306/retail_db \
~ username=retail_dba \
-password=cloudera \
--table departments \
--hive-home /user/hive/warehouse \
-hive-import \
-hive-overwrite \
--hive-table departments_hive \
--fields-terminated-by '\001'
Step 3 : Check-the data in directory.
hdfs dfs -Is /user/hive/warehouse/departments_hive
hdfs dfs -cat/user/hive/warehouse/departmentshive/part'
Check data in hive table.
Select * from departments_hive;

NEW QUESTION: 34
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: 35
CORRECT TEXT
Problem Scenario 15 : You have been given following mysql database details as well as other info.
user=retail_dba
password=cloudera

database=retail_db
jdbc URL = jdbc:mysql://quickstart:3306/retail_db
Please accomplish following activities.
1. In mysql departments table please insert following record. Insert into departments values(9999, '"Data Science"1);
2. Now there is a downstream system which will process dumps of this file. However, system is designed the way that it can process only files if fields are enlcosed in(') single quote and separate of the field should be (-} and line needs to be terminated by : (colon).
3. If data itself contains the " (double quote } than it should be escaped by \.
4. Please import the departments table in a directory called departments_enclosedby and file should be able to process by downstream system.
Answer:
See the explanation for Step by Step Solution and configuration.
Explanation:
Solution :
Step 1 : Connect to mysql database.
mysql --user=retail_dba -password=cloudera
show databases; use retail_db; show tables;
Insert record
Insert into departments values(9999, '"Data Science"');
select" from departments;
Step 2 : Import data as per requirement.
sqoop import \
-connect jdbc:mysql;//quickstart:3306/retail_db \
~ username=retail_dba \
--password=cloudera \
-table departments \
-target-dir /user/cloudera/departments_enclosedby \
-enclosed-by V -escaped-by \\ -fields-terminated-by--' -lines-terminated-by :
Step 3 : Check the result.
hdfs dfs -cat/user/cloudera/departments_enclosedby/part"

NEW QUESTION: 36
CORRECT TEXT
Problem Scenario 30 : You have been given three csv files in hdfs as below.
EmployeeName.csv with the field (id, name)
EmployeeManager.csv (id, manager Name)
EmployeeSalary.csv (id, Salary)
Using Spark and its API you have to generate a joined output as below and save as a text tile (Separated by comma) for final distribution and output must be sorted by id.
ld,name,salary,managerName

EmployeeManager.csv
E01,Vishnu
E02,Satyam
E03,Shiv
E04,Sundar
E05,John
E06,Pallavi
E07,Tanvir
E08,Shekhar
E09,Vinod
E10,Jitendra
EmployeeName.csv
E01,Lokesh
E02,Bhupesh
E03,Amit
E04,Ratan
E05,Dinesh
E06,Pavan
E07,Tejas
E08,Sheela
E09,Kumar
E10,Venkat
EmployeeSalary.csv
E01,50000
E02,50000
E03,45000
E04,45000
E05,50000
E06,45000
E07,50000
E08,10000
E09,10000
E10,10000

Answer:

See the explanation for Step by Step Solution and configuration.

Explanation:

Solution :

Step 1 : Create all three files in hdfs in directory called sparkl (We will do using Hue}.
However, you can first create in local filesystem and then

Step 2 : Load EmployeeManager.csv file from hdfs and create PairRDDs

val manager = sc.textFile("spark1/EmployeeManager.csv")

val managerPairRDD = manager.map(x=> (x.split(",")(0),x.split(",")(1)))
Step 3 : Load EmployeeName.csv file from hdfs and create PairRDDs
val name = sc.textFile("spark1/EmployeeName.csv")
val namePairRDD = name.map(x=> (x.split(",")(0),x.split('\")(1)))
Step 4 : Load EmployeeSalary.csv file from hdfs and create PairRDDs
val salary = sc.textFile("spark1/EmployeeSalary.csv")
val salaryPairRDD = salary.map(x=> (x.split(",")(0),x.split(",")(1)))
Step 4 : Join all pairRDDS
val joined = namePairRDD.join(salaryPairRDD}.join(managerPairRDD}
Step 5 : Now sort the joined results, val joinedData = joined.sortByKey()
Step 6 : Now generate comma separated data.
val finalData = joinedData.map(v=> (v._1, v._2._1._1, v._2._1._2, v._2._2))
Step 7 : Save this output in hdfs as text file.
finalData.saveAsTextFile("spark1/result.txt")

NEW QUESTION: 37
CORRECT TEXT
Problem Scenario 43 : You have been given following code snippet.
val grouped = sc.parallelize(Seq(((1,"twoM), List((3,4), (5,6)))))
val flattened = grouped.flatMap {A =>
groupValues.map { value => B }
}
You need to generate following output.
Hence replace A and B
Array((1,two,3,4),(1,two,5,6))
Answer:
See the explanation for Step by Step Solution and configuration.
Explanation:
Solution :
A case (key, groupValues)
B (key._1, key._2, value._1, value._2)

NEW QUESTION: 38
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: 39
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: 40
Answer:
See the explanation for Step by Step Solution and configuration.
Explanation:
Solution :
Step 1 : Select all the records with quantity >= 5000 and name starts with 'Pen' val results = sqlContext.sql(.....SELECT * FROM products WHERE quantity >= 5000 AND name LIKE 'Pen %.....) results.show()
Step 2 : Select all the records with quantity >= 5000 , price is less than 1.24 and name starts with 'Pen' val results = sqlContext.sql(.....SELECT * FROM products WHERE quantity >= 5000 AND price < 1.24 AND name LIKE 'Pen %.....) results. showQ
Step 3 : Select all the records witch does not have quantity >= 5000 and name does not starts with 'Pen' val results = sqlContext.sql('.....SELECT * FROM products WHERE NOT (quantity >= 5000
AND name LIKE 'Pen %').....)
results. showQ
Step 4 : Select all the products wchich name is 'Pen Red', 'Pen Black'
val results = sqlContext.sql('.....SELECT' FROM products WHERE name IN ('Pen Red', 'Pen Black').....)
results. showQ
Step 5 : Select all the products which has price BETWEEN 1.0 AND 2.0 AND quantity BETWEEN 1000 AND 2000.
val results = sqlContext.sql(.....SELECT * FROM products WHERE (price BETWEEN 1.0 AND 2.0) AND (quantity BETWEEN 1000 AND 2000).....)
results. show()

NEW QUESTION: 41
CORRECT TEXT
Problem Scenario 60 : You have been given below code snippet.

val a = sc.parallelize(List("dog", "salmon", "salmon", "rat", "elephant"}, 3} val b = a.keyBy(_.length) val c = sc.parallelize(List("dog","cat","gnu","salmon","rabbit","turkey","woif","bear","bee"), 3) val d = c.keyBy(_.length) operation1
Write a correct code snippet for operationl which will produce desired output, shown below.
Array[(lnt, (String, String))] = Array((6,(salmon,salmon)), (6,(salmon,rabbit)), (6,(salmon,turkey)), (6,(salmon,salmon)), (6,(salmon,rabbit)), (6,(salmon,turkey)), (3,(dog,dog)), (3,(dog,cat)), (3,(dog,gnu)), (3,(dog,bee)), (3,(rat,dog)), (3,(rat,cat)), (3,(rat,gnu)), (3,(rat,bee)))

Answer:

See the explanation for Step by Step Solution and configuration.
Explanation:
solution:
b.join(d).collect
join [Pair]: Performs an inner join using two key-value RDDs. Please note that the keys must be generally comparable to make this work. keyBy : Constructs two-component tuples
(key-value pairs) by applying a function on each data item. The result of the function becomes the data item becomes the key and the original value of the newly created tuples.

NEW QUESTION: 42

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: 43
CORRECT TEXT
Problem Scenario 65 : You have been given below code snippet.
val a = sc.parallelize(List("dog", "cat", "owl", "gnu", "ant"), 2)
val b = sc.parallelize(1 to a.count.tolnt, 2)
val c = a.zip(b)
operation1
Write a correct code snippet for operationl which will produce desired output, shown below.
Array[(String, Int)] = Array((owl,3), (gnu,4), (dog,1), (cat,2>, (ant,5))
Answer:
See the explanation for Step by Step Solution and configuration.
Explanation:
Solution : c.sortByKey(false).collect
sortByKey [Ordered] : This function sorts the input RDD's data and stores it in a new RDD.
"The output RDD is a shuffled RDD because it stores data that is output by a reducer
which has been shuffled. The implementation of this function is actually very clever.
First, it uses a range partitioner to partition the data in ranges within the shuffled RDD.
Then it sorts these ranges individually with mapPartitions using standard sort mechanisms.

NEW QUESTION: 44
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
Answer:
See the explanation for Step by Step Solution and configuration.

NEW QUESTION: 45
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: 46
CORRECT TEXT
Problem Scenario 3: 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. Import data from categories table, where category=22 (Data should be stored in categories subset)
2. Import data from categories table, where category>22 (Data should be stored in categories_subset_2)
3. Import data from categories table, where category between 1 and 22 (Data should be stored in categories_subset_3)
4. While importing catagories data change the delimiter to '|' (Data should be stored in categories_subset_S)
5. Importing data from catagories table and restrict the import to category_name,category id columns only with delimiter as '|'
6 . Add null values in the table using below SQL statement ALTER TABLE categories modify category_department_id int(11); INSERT INTO categories values (eO.NULL.'TESTING');
7. Importing data from catagories table (In categories_subset_17 directory) using '|' delimiter and categoryjd between 1 and 61 and encode null values for both string and non string columns.
8. Import entire schema retail_db in a directory categories_subset_all_tables

Answer:

See the explanation for Step by Step Solution and configuration.

Explanation:

Solution:

Step 1: Import Single table (Subset data} Note: Here the ' is the same you find on - key

```
sqoop import --connect jdbc:mysql://quickstart:3306/retail_db --username=retail_dba -password=cloudera -table=categories ~warehouse-dir= categories_subset --where \'category_id\'=22 --m 1
```

Step 2 : Check the output partition

```
hdfs dfs -cat categoriessubset/categories/part-m-00000
```

Step 3 : Change the selection criteria (Subset data)

```
sqoop import --connect jdbc:mysql://quickstart:3306/retail_db --username=retail_dba -password=cloudera -table=categories ~warehouse-dir= categories_subset_2 --where \'category_id\'\>22 -m 1
```

Step 4 : Check the output partition

```
hdfs dfs -cat categories_subset_2/categories/part-m-00000
```

Step 5 : Use between clause (Subset data)

```
sqoop import --connect jdbc:mysql://quickstart:3306/retail_db --username=retail_dba -password=cloudera -table=categories ~warehouse-dir=categories_subset_3 --where "\'category_id\' between 1 and 22" --m 1
```

Step 6 : Check the output partition

```
hdfs dfs -cat categories_subset_3/categories/part-m-00000
```

Step 7 : Changing the delimiter during import.

sqoop import --connect jdbc:mysql://quickstart:3306/retail_db --username=retail dba -password=cloudera -table=categories -warehouse-dir=:categories_subset_6 --where "/'categoryjd /' between 1 and 22" -fields-terminated-by='|' -m 1
Step 8 : Check the.output partition
hdfs dfs -cat categories_subset_6/categories/part-m-00000
Step 9 : Selecting subset columns
sqoop import --connect jdbc:mysql://quickstart:3306/retail_db --username=retail_dba -password=cloudera -table=categories --warehouse-dir=categories subset col -where "/'category id/' between 1 and 22" -fields-terminated-by=T -columns=category name,category id --m 1
Step 10 : Check the output partition
hdfs dfs -cat categories_subset_col/categories/part-m-00000
Step 11 : Inserting record with null values (Using mysql} ALTER TABLE categories modify category_department_id int(11); INSERT INTO categories values ^NULL/TESTING'); select" from categories;
Step 12 : Encode non string null column
sqoop import --connect jdbc:mysql://quickstart:3306/retail_db --username=retail dba -password=cloudera -table=categories --warehouse-dir=categortes_subset_17 -where "\"category_id\" between 1 and 61" -fields-terminated-by=,|' --null-string-N' -null-non-string=,N' --m 1
Step 13 : View the content
hdfs dfs -cat categories_subset_17/categories/part-m-00000
Step 14 : Import all the tables from a schema (This step will take little time) sqoop import-all-tables -connect jdbc:mysql://quickstart:3306/retail_db -- username=retail_dba -password=cloudera -warehouse-dir=categories_si
Step 15 : View the contents
hdfs dfs -Is categories_subset_all_tables
Step 16 : Cleanup or back to originals.
delete from categories where categoryid in (59,60);
ALTER TABLE categories modify category_department_id int(11) NOTNULL;
ALTER TABLE categories modify category_name varchar(45) NOT NULL;
desc categories;

NEW QUESTION: 47
CORRECT TEXT
Problem Scenario 77 : You have been given MySQL DB with following details.
user=retail_dba
password=cloudera
database=retail_db
table=retail_db.orders
table=retail_db.order_items
jdbc URL = jdbc:mysql://quickstart:3306/retail_db
Columns of order table : (orderid , order_date , order_customer_id, order_status)
Columns of ordeMtems table : (order_item_id , order_item_order_ld , order_item_product_id, order_item_quantity,order_item_subtotal,order_ item_product_price)
Please accomplish following activities.
1. Copy "retail_db.orders" and "retail_db.order_items" table to hdfs in respective directory p92_orders and p92 order items .
2 . Join these data using orderid in Spark and Python
3 . Calculate total revenue perday and per order
4. Calculate total and average revenue for each date. - combineByKey -aggregateByKey
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_items -m1
Note : Please check you dont have space between before or after '=' sign. Sqoop uses the MapReduce framework to copy data from RDBMS to hdfs
Step 2 : Read the data from one of the partition, created using above command, hadoop fs -cat p92_orders/part-m-00000 hadoop fs -cat p92_order_items/part-m-00000
Step 3 : Load these above two directory as RDD using Spark and Python (Open pyspark terminal and do following). orders = sc.textFile("p92_orders") orderItems = sc.textFile("p92_order_items")
Step 4 : Convert RDD into key value as (orderjd as a key and rest of the values as a value)
First value is orderjd

ordersKeyValue = orders.map(lambda line: (int(line.split(",")[0]), line))
# Second value as an Orderjd
orderItemsKeyValue = orderItems.map(lambda line: (int(line.split(",")[1]), line))
Step 5 : Join both the RDD using orderjd
joinedData = orderItemsKeyValue.join(ordersKeyValue)
#print the joined data
for line in joinedData.collect():
print(line)
Format of joinedData as below.
[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.
//Retruned row will contain ((order_date,order_id),amout_collected)
revenuePerDayPerOrder = joinedData.map(lambda row: ((row[1][1].split(M,M)[1],row[0]}, float(row[1][0].split(",")[4])))
#print the result
for line in revenuePerDayPerOrder.collect():
print(line)
Step 7 : Now calculate total revenue perday and per order
A. Using reduceByKey
totalRevenuePerDayPerOrder = revenuePerDayPerOrder.reduceByKey(lambda runningSum, value: runningSum + value)
for line in totalRevenuePerDayPerOrder.sortByKey().collect(): print(line)
#Generate data as (date, amount_collected) (Ignore ordeMd)
dateAndRevenueTuple = totalRevenuePerDayPerOrder.map(lambda line: (line[0][0], line[1])) for line in dateAndRevenueTuple.sortByKey().collect(): print(line)
Step 8 : Calculate total amount collected for each day. And also calculate number of days.
Generate output as (Date, Total Revenue for date, total_number_of_dates)
Line 1 : it will generate tuple (revenue, 1)
Line 2 : Here, we will do summation for all revenues at the same time another counter to maintain number of records.
#Line 3 : Final function to merge all the combiner
totalRevenueAndTotalCount = dateAndRevenueTuple.combineByKey(\
lambda revenue: (revenue, 1), \
lambda revenueSumTuple, amount: (revenueSumTuple[0] + amount, revenueSumTuple[1] + 1), \
lambda tuple1, tuple2: (round(tuple1[0] + tuple2[0], 2}, tuple1[1] + tuple2[1]) \ for line in totalRevenueAndTotalCount.collect(): print(line)
Step 9 : Now calculate average for each date
averageRevenuePerDate = totalRevenueAndTotalCount.map(lambda threeElements: (threeElements[0], threeElements[1][0]/threeElements[1][1]}}
for line in averageRevenuePerDate.collect(): print(line)

Step 10 : Using aggregateByKey
#line 1 : (Initialize both the value, revenue and count)
#line 2 : runningRevenueSumTuple (Its a tuple for total revenue and total record count for each date)
# line 3 : Summing all partitions revenue and count
totalRevenueAndTotalCount = dateAndRevenueTuple.aggregateByKey(\
(0,0), \
lambda runningRevenueSumTuple, revenue: (runningRevenueSumTuple[0] + revenue, runningRevenueSumTuple[1] + 1), \ lambda tupleOneRevenueAndCount, tupleTwoRevenueAndCount:
(tupleOneRevenueAndCount[0] + tupleTwoRevenueAndCount[0], tupleOneRevenueAndCount[1] + tupleTwoRevenueAndCount[1]) \
)
for line in totalRevenueAndTotalCount.collect(): print(line)
Step 11 : Calculate the average revenue per date
averageRevenuePerDate = totalRevenueAndTotalCount.map(lambda threeElements: (threeElements[0], threeElements[1][0]/threeElements[1][1]))
for line in averageRevenuePerDate.collect(): print(line)

NEW QUESTION: 48
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: 49
CORRECT TEXT
Problem Scenario 85 : In Continuation of previous question, please accomplish following activities.
1. Select all the columns from product table with output header as below. productID AS ID code AS Code name AS Description price AS 'Unit Price'
2. Select code and name both separated by ' -' and header name should be Product Description'.
3. Select all distinct prices.
4 . Select distinct price and name combination.
5 . Select all price data sorted by both code and productID combination.
6 . count number of products.
7 . Count number of products for each code.
Answer:
See the explanation for Step by Step Solution and configuration.
Explanation:
Solution :
Step 1 : Select all the columns from product table with output header as below. productID AS ID code AS Code name AS Description price AS "Unit Price'
val results = sqlContext.sql(.....SELECT productID AS ID, code AS Code, name AS Description, price AS Unit Price' FROM products ORDER BY ID""""
results.show()
Step 2 : Select code and name both separated by ' -' and header name should be "Product Description.
val results = sqlContext.sql(.....SELECT CONCAT(code,' -', name) AS Product Description, price FROM products""") results.showQ
Step 3 : Select all distinct prices.
val results = sqlContext.sql(.....SELECT DISTINCT price AS Distinct Price" FROM products.....) results.show()
Step 4 : Select distinct price and name combination.
val results = sqlContext.sql(.....SELECT DISTINCT price, name FROM products""")
results. showQ
Step 5 : Select all price data sorted by both code and productID combination.
val results = sqlContext.sql('.....SELECT' FROM products ORDER BY code, productID'.....)
results.show()
Step 6 : count number of products.
val results = sqlContext.sql(.....SELECT COUNT(') AS 'Count' FROM products.....)
results.show()
Step 7 : Count number of products for each code.

val results = sqlContext.sql(.....SELECT code, COUNT('} FROM products GROUP BY code.....) results. showQ val results = sqlContext.sql(.....SELECT code, COUNT('} AS count FROM products
GROUP BY code ORDER BY count DESC.....)
results. showQ

NEW QUESTION: 50
Answer:
See the explanation for Step by Step Solution and configuration.
Explanation:
Solutions :
Step 1 : Clean the hdfs tile 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_itmes
hadoop fs -rm -R customers
Step 2 : Now import the department table as per requirement.
sqoop import \
-connect jdbc:mysql://quickstart:3306/retail_db \
--username=retail_dba \
-password=cloudera \
-table departments \
-target-dir /user/cloudera/departments \
-m2\
-boundary-query "select 1, 25 from departments" \
-columns department_id,department_name
Step 3 : Check imported data.
hdfs dfs -Is departments
hdfs dfs -cat departments/part-m-00000
hdfs dfs -cat departments/part-m-00001