CCA175 Q&As

CCA Spark and Hadoop Developer Exam

Pass Cloudera CCA175 Exam with 100% Guarantee

Free Download Real Questions & Answers **PDF** and **VCE** file from:

https://www.pass4itsure.com/cca175.html

100% Passing Guarantee
100% Money Back Assurance

Following Questions and Answers are all new published by Cloudera Official Exam Center

QUESTION 1

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

other info.

Please accomplish following.

1.

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

2.

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

3.

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

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

Solutions:

Step 1 : Clean the hdfs file system, if they exists clean out.

hadoop fs -rm -R departments

hadoop fs -rm -R categories

hadoop fs -rm -R products

hadoop fs -rm -R orders

hadoop fs -rm -R order_items

hadoop fs -rm -R customers

Step 2 : Now import the department table as per requirement.

sqoop import \

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

-username=retail_dba \

-password=cloudera \

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

order_items.order_item_order_id where \SCONDITlONS" \

-target-dir /user/cloudera/order_join \

-split-by order_id \

--num-mappers 2

Step 3 : Check imported data.

hdfs dfs -Is order_join

hdfs dfs -cat order_join/part-m-00000

hdfs dfs -cat order_join/part-m-00001

QUESTION 2

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

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

Solution

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

Notes: ./bin/spark-submit \

--class

--master \

--deploy-mode \

-conf = \

other options

\

[application-arguments]

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

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

property which is set in Spark-default.conf.

QUESTION 3

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

1003,PEN,Pen Black,2000,1.25,501 1004,PEC,Pencil 2B,10000,0.48,502 1005,PEC,Pencil 2H,8000,0.49,502 1006,PEC,Pencil HB,0,9999.99,502 2001,PEC,Pencil 3B,500,0.52,501 2002,PEC,Pencil 4B,200,0.62,501 2003,PEC,Pencil 5B,100,0.73,501 2004,PEC,Pencil 6B,500,0.47,502 supplier.csv supplierid,name,phone 501,ABC

Traders,88881111 502,XYZ Company,88882222 503,QQ Corp,88883333 products_suppliers.csv productID,supplierID 2001,501 2002,501 2003,501 2004,502 2001,503 Now accomplish all the queries given in solution. Select product, its price , its supplier name where product price is less than 0.6 using SparkSQL

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

Solution :

Step 1:

hdfs dfs -mkdir sparksql2

hdfs dfs -put product.csv sparksq!2/

hdfs dfs -put supplier.csv sparksql2/

hdfs dfs -put products_suppliers.csv sparksql2/

Step 2 : Now in spark shell

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

import sqlContext.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:lnteger)

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

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

// create an RDD of Product objects

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

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

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

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

prdRDD.first()

prdRDD.count()

supRDD.first() supRDD.count()

prdsupRDD.first() prdsupRDD.count{}

// change RDD of Product objects to a DataFrame

val prdDF = prdRDD.toDF()

val supDF = supRDD.toDF()

val prdsupDF = prdsupRDD.toDF()

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

supDF.registerTempTablef\\'suppliers")

prdsupDF.registerTempTablef\\'productssuppliers"}

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

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

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

WHERE price

results. show()

QUESTION 4

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,

56, 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,

89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100))

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

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

QUESTION 5

Problem Scenario 91 : 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 tile locally.

2.

Load this tile 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.

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

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

val employee = sqlContext.read.json("/user/cloudera/employee.json")

employee.write.parquet("employee. parquet")

val parq_data = sqlContext.read.parquet("employee.parquet")

parq_data.registerTempTable("employee")

val allemployee = sqlContext.sql("SELeCT\\' FROM employee")

all_employee.show()

import org.apache.spark.sql.SaveMode prdDF.write..format("orc").saveAsTable("product

ore table"}

//Change the codec. sqlContext.setConf("spark.sql.parquet.compression.codec","snappy")
employee.write.mode(SaveMode.Overwrite).parquet("employee.parquet")

QUESTION 6

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

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

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

Solution :

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

Step 2 : Show all the available tables show tables;

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

mandatory)

sqoop job -create sqoopjob \ -- import \

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

-username=retail_dba \

-password=cloudera \

-table categories \

-target-dir categories_targetJob \

-fields-terminated-by \\'|\\' \ -lines-terminated-by \\'\n\\' Step 4 : List all the Sqoop Jobs sqoop job --list Step 5 : Show details of the Sqoop Job sqoop job --show sqoopjob Step 6 : Execute the sqoopjob sqoopjob --exec sqoopjob Step 7 :

Check the output of import job hdfs dfs -ls categories_target_job hdfs dfs -cat categories_target_job/part*

QUESTION 7

Problem Scenario 35 : You have been given a file named spark7/EmployeeName.csv (id,name). EmployeeName.csv E01,Lokesh E02,Bhupesh E03,Amit E04,Ratan E05,Dinesh E06,Pavan E07,Tejas E08,Sheela E09,Kumar E10,Venkat

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

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

Solution:

Step 1 : Create file in hdfs (We will do using Hue). However, you can first create in local

filesystem and then upload it to hdfs.

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

val name = sc.textFile("spark7/EmployeeName.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 sort the rdd by key.

val sortedOutput = swapped.sortByKey()

Step 5 : Now swap the result back

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

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

swappedBack. repartition(1).saveAsTextFile("spark7/result.txt")

QUESTION 8

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

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

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")

QUESTION 9

Problem Scenario 76 : You have been given MySQL DB with following details. user=retail_dba password=cloudera database=retail_db table=retail_db.orders table=retail_db.order_items jdbc URL = jdbc:mysql://quickstart:3306/retail_db Columns of order table : (orderid , order_date , ordercustomerid, order_status} ..... Please accomplish following activities.

1.

Copy "retail_db.orders" table to hdfs in a directory p91_orders.

2.

Once data is copied to hdfs, using pyspark calculate the number of order for each status.

3.

Use all the following methods to calculate the number of order for each status. (You need to know all these functions and its behavior for real exam)

-countByKey() -groupByKey()

-reduceByKey() -aggregateByKey()

-combineByKey()

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

Solution :

Step 1 : Import Single table

sqoop import --connect jdbc:mysql://quickstart:3306/retail_db --username=retail dba password=cloudera -table=orders

--target-dir=p91_orders

Note : Please check you dont have space between before or after \\'=\\' sign. Sqoop uses the

MapReduce framework to copy data from RDBMS to hdfs

Step 2 : Read the data from one of the partition, created using above command, hadoop fs

-cat p91_orders/part-m-00000

Step 3: countByKey #Number of orders by status allOrders = sc.textFile("p91_orders")

#Generate key and value pairs (key is order status and vale as an empty string keyValue =

aIIOrders.map(lambda line: (line.split(",")[3], ""))

#Using countByKey, aggregate data based on status as a key

output=keyValue.countByKey()Jtems()

for line in output: print(line)

Step 4 : groupByKey

#Generate key and value pairs (key is order status and vale as an one

keyValue = allOrders.map(lambda line: (line.split)",")[3], 1))

#Using countByKey, aggregate data based on status as a key output=

keyValue.groupByKey().map(lambda kv: (kv[0], sum(kv[1]}}}

tor line in output.collect(): print(line}

Step 5 : reduceByKey

#Generate key and value pairs (key is order status and vale as an one

keyValue = allOrders.map(lambda line: (line.split(","}[3], 1))

#Using countByKey, aggregate data based on status as a key output=

keyValue.reduceByKey(lambda a, b: a + b)

tor line in output.collect(): print(line}

Step 6: aggregateByKey

#Generate key and value pairs (key is order status and vale as an one keyValue =

allOrders.map(lambda line: (line.split(",")[3], line}}

output=keyValue.aggregateByKey(0, lambda a, b: a+1, lambda a, b: a+b}

for line in output.collect(): print(line}

Step 7 : combineByKey

#Generate key and value pairs (key is order status and vale as an one

keyValue = allOrders.map(lambda line: (line.split(",")[3], line))

output=keyValue.combineByKey(lambda value: 1, lambda ace, value: acc+1, lambda ace,

value: acc+value)

tor line in output.collect(): print(line)

#Watch Spark Professional Training provided by www.ABCTECH.com to understand more

on each above functions. (These are very important functions for real exam)

QUESTION 10

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.

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

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"}

#Iterate 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"

QUESTION 11

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()

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

Solution :

Step 1 : Import Single table .

sqoop import --connect jdbc:mysql://quickstart:3306/retail_db -username=retail_dba password=cloudera -table=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]),linesplit(,,,,,)[

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]}}

QUESTION 12

Problem Scenario 17 : 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_hiveOl(department_id int,

department_name string, avg_salary int);

2.

Create another table in mysql using below statement CREATE TABLE IF NOT EXISTS

departments_hive01(id int, department_name varchar(45), avg_salary int);

3.

Copy all the data from departments table to departments_hive01 using insert into

departments_hive01 select a.*, null from departments a;

Also insert following records as below

insert into departments_hive01 values(777, "Not known",1000);

insert into departments_hive01 values(8888, null,1000);

insert into departments_hive01 values(666, null,1100);

4.

Now import data from mysql table departments_hive01 to this hive table. Please make

sure that data should be visible using below hive command. Also, while importing if null

value found for department_name column replace it with "" (empty string) and for id column

with -999 select * from departments_hive;

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

Solution :

Step 1 : Create hive table as below.

hive

show tables;

create table departments_hive01(department_id int, department_name string, avgsalary

int);

Step 2 : Create table in mysql db as well.

mysql -user=retail_dba -password=cloudera

use retail_db

CREATE TABLE IF NOT EXISTS departments_hive01(id int, department_name

varchar(45), avg_salary int);

show tables;

step 3 : Insert data in mysql table.

insert into departments_hive01 select a.*, null from departments a;

check data inserts

select\\' from departments_hive01;

Now iserts null records as given in problem. insert into departments_hive01 values(777,

"Not known",1000); insert into departments_hive01 values(8888, null,1000); insert into

departments_hive01 values(666, null,1100);

Step 4 : Now import data in hive as per requirement.

sqoop import \

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

~username=retail_dba \

--password=cloudera \

-table departments_hive01 \

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

--hive-import \

-hive-overwrite \

-hive-table departments_hive0l \

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

--null-string M"\

--null-non-strlng -999 \

-split-by id \

-m 1

Step 5 : Checkthe data in directory.

hdfs dfs -Is /user/hive/warehouse/departments_hive01

hdfs dfs -cat/user/hive/warehouse/departments_hive01/part"

Check data in hive table.

Select * from departments_hive01;

QUESTION 13

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

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

Solution :

Step 1 : Import Single table .

sqoop import --connect jdbc:mysql://quickstart:3306/retail_db -username=retail_dba password=cloudera -table=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)

QUESTION 14

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\

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

Solutions :

Step 1 : Clean the hdfs file system, if they exists clean out.

hadoop fs -rm -R departments

hadoop fs -rm -R categories

hadoop fs -rm -R products

hadoop fs -rm -R orders

hadoop fs -rm -R order_items

hadoop fs -rm -R customers

Step 2 : Now import the department table as per requirement.

sqoop import \

-connect jdbc:mysql://quickstart:330G/retaiI_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

QUESTION 15

Problem Scenario 55 : You have been given below code snippet.

val pairRDDI = sc.parallelize(List(("cat",2), ("cat", 5), ("book", 4),("cat", 12))) val

pairRDD2 = sc.parallelize(List(("cat",2), ("cup", 5), ("mouse", 4),("cat", 12)))

operation1

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

Array[(String, (Option[lnt], Option[lnt]))] = Array((book,(Some(4},None)),

(mouse,(None,Some(4))), (cup,(None,Some(5))), (cat,(Some(2),Some(2)),

(cat,(Some(2),Some(12))), (cat,(Some(5),Some(2))), (cat,(Some(5),Some(12))),

(cat,(Some(12),Some(2))), (cat,(Some(12),Some(12)))J

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

Solution : pairRDD1.fullOuterJoin(pairRDD2).collect

fullOuterJoin [Pair]

Performs the full outer join between two paired RDDs.

Listing Variants

def fullOuterJoin[W](other: RDD[(K, W)], numPartitions: Int): RDD[(K, (Option[V],

OptionfW]))]

def fullOuterJoin[W](other: RDD[(K, W}]}: RDD[(K, (Option[V], OptionfW]))]

def fullOuterJoin[W](other: RDD[(K, W)], partitioner: Partitioner): RDD[(K, (Option[V],

Option[W]))]