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 83 : In Continuation of previous question, please accomplish following

activities.

1.

Select all the records with quantity >= 5000 and name starts with \\'Pen\\'

2.

Select all the records with quantity >= 5000, price is less than 1.24 and name starts with \\'Pen\\'

3.

Select all the records witch does not have quantity >= 5000 and name does not starts with \\'Pen\\'

4.

Select all the products which name is \\'Pen Red\\', \\'Pen Black\\'

5.

Select all the products which has price BETWEEN 1.0 AND 2.0 AND quantity BETWEEN 1000 AND 2000.

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

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

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()

QUESTION 2

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 3

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 4

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 5

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

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

Solution :

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

Step 2 : Create sqlContext sqlContext = HiveContext(sc)

Step 3 : Query hive

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

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

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