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 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)

QUESTION 2

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 3

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.

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

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)}

QUESTION 4

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 \SCONDITIONS" \

-target-dir /user/cloudera/order_join \

-split-by order_id \

--num-mappers 2

Step 3 : Check imported data.

hdfs dfs -ls order_join

hdfs dfs -cat order_join/part-m-00000

hdfs dfs -cat order_join/part-m-00001

QUESTION 5

Problem Scenario 6 : 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 Compression Codec : org.apache.hadoop.io.compress.SnappyCodec Please accomplish following.

1.

Import entire database such that it can be used as a hive tables, it must be created in default schema.

2.

Also make sure each tables file is partitioned in 3 files e.g. part-00000, part-00002, part00003

3.

Store all the Java files in a directory called java_output to evalute the further

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

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 -ls /user/hive/warehouse

hdfs dfs -ls /user/hive/warehouse/categories

c.

Check the output Java directory.

ls -ltr java_output/