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.geekcert.com/cca175.html

100% Passing Guarantee
100% Money Back Assurance

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

- **Instant Download** After Purchase
- **100% Money Back** Guarantee
- **365 Days** Free Update
- **800,000+** Satisfied Customers

QUESTION 1

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.

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

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

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 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

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

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")

QUESTION 4

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

1.

Connect MySQL DB and check the content of the tables.

2.

Copy "retaildb.categories" table to hdfs, without specifying directory name.

3.

Copy "retaildb.categories" table to hdfs, in a directory name "categories_target".

4.

Copy "retaildb.categories" table to hdfs, in a warehouse directory name "categories_warehouse".

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 : View/Count data from a table in MySQL select count(1} from categories;

Step 4 : Check the currently available data in HDFS directory hdfs dfs -ls

Step 5 : Import Single table (Without specifying directory). sqoop import --connect jdbc:mysql://quickstart:3306/retail_db -username=retail_dba password=cloudera -table=categories 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 6 : Read the data from one of the partition, created using above command, hdfs dfs catxategories/part-m-00000 Step 7 : Specifying target directory in import command (We are using number of mappers =1, you can change accordingly) sqoop import -connect jdbc:mysql://quickstart:3306/retail_db -username=retail_dba -password=cloudera ~table=categories -target-dir=categortes_target --m 1 Step 8 : Check the content in one of the partition file. hdfs dfs -cat categories_target/part-m-00000 Step 9 : Specifying parent directory so that you can copy more than one table in a specified target directory. Command to specify warehouse directory. sqoop import -.-connect jdbc:mysql://quickstart:3306/retail_db --username=retail dba password=cloudera -table=categories -warehouse-dir=categories_warehouse --m 1

QUESTION 5

Problem Scenario 26 : You need to implement near real time solutions for collecting information when submitted in file with below information. You have been given below directory location (if not available than create it) /tmp/nrtcontent. Assume your departments upstream service is continuously committing data in this directory as a new file (not stream of data, because it is near real time solution). As soon as file committed in this directory that needs to be available in hdfs in /tmp/flume location Data

echo "I am preparing for CCA175 from ABCTECH.com" > /tmp/nrtcontent/.he1.txt mv /tmp/nrtcontent/.he1.txt /tmp/nrtcontent/he1.txt After few mins echo "I am preparing for CCA175 from TopTech.com" > /tmp/nrtcontent/.qt1.txt mv /tmp/nrtcontent/.qt1.txt /tmp/nrtcontent/qt1.txt

Write a flume configuration file named flumes.conf and use it to load data in hdfs with following additional properties.

1.

Spool /tmp/nrtcontent

2.

File prefix in hdfs sholuld be events

3.

File suffix should be Jog

4.

If file is not commited and in use than it should have as prefix.

5.

Data should be written as text to hdfs

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

Solution : Step 1 : Create directory mkdir /tmp/nrtcontent Step 2 : Create flume configuration file, with below configuration for source, sink and channel and save it in flume6.conf. agent1 .sources = source1 agent1 .sinks = sink1 agent1.channels = channel1 agent1 .sources.source1.channels = channel1 agent1 .sinks.sink1.channel = channel1 agent1 .sources.source1.type = spooldir agent1 .sources.source1.spoolDir = /tmp/nrtcontent agent1 .sinks.sink1 .type = hdfs agent1 .sinks.sink1.hdfs.path = /tmp/flume agent1.sinks.sink1.hdfs.filePrefix = events agent1.sinks.sink1.hdfs.fileSuffix = .log agent1 .sinks.sink1.hdfs.inUsePrefix = _ agent1 .sinks.sink1.hdfs.fileType =

Data Stream 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/flume6.conf --name agent1 Step 5 : Open another terminal and create a file in /tmp/nrtcontent echo "I am preparing for CCA175 from ABCTechm.com" > /tmp/nrtcontent/.he1.txt mv /tmp/nrtcontent/.he1.txt /tmp/nrtcontent/he1.txt After few mins echo "I am preparing for CCA175 from TopTech.com" > /tmp/nrtcontent/.qt1.txt mv /tmp/nrtcontent/.qt1.txt /tmp/nrtcontent/qt1.txt