# 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.pass2lead.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 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 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 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)

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 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")

**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 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

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

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 columnssqoop 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-nonstring=, 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;