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

QUESTION 1

Problem Scenario 67 : You have been given below code snippet.

lines = sc.parallelize([\\'lts fun to have fun,\\',\\'but you have to know how.\\'])

M = lines.map(lambda x: x.replace(\\',7 \\').replace(\\'.\\',\\' \\'J.replaceC-V \\').lower())

r2 = r1.flatMap(lambda x: x.split())

r3 = r2.map(lambda x: (x, 1))

operation1

r5 = r4.map(lambda x:(x[1],x[0]))

r6 = r5.sortByKey(ascending=False)

r6.take(20)

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

[(2, \\'fun\\'), (2, \\'to\\'), (2, \\'have\\'), (1, its\\'), (1, \\'know1), (1, \\'how1), (1, \\'you\\'), (1, \\'but\\')]

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

Solution : r4 = r3.reduceByKey(lambda x,y:x+y)

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 GG : You have been given below code snippet.

val a = sc.parallelize(List("dog", "tiger", "lion", "cat", "spider", "eagle"), 2)

val b = a.keyBy(_.length)

val c = sc.parallelize(List("ant", "falcon", "squid"), 2)

val d = c.keyBy(.length)

operation 1

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

Array[(lnt, String)] = Array((4,lion))

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

Solution : b.subtractByKey(d).collect subtractByKey [Pair] : Very similar to subtract, but instead of supplying a function, the keycomponent of each pair will be automatically used as criterion for removing items from the first RDD.

QUESTION 4

Problem Scenario 89 : You have been given below patient data in csv format, patientID,name,dateOfBirth,lastVisitDate 1001,Ah Teck,1991-12-31,2012-01-20 1002,Kumar,2011-10-29,2012-09-20 1003,Ali,2011-01-30,2012-10-21 Accomplish following activities.

1.

Find all the patients whose lastVisitDate between current time and \\'2012-09-15\\'

2.

Find all the patients who born in 2011

3.

Find all the patients age

4.

List patients whose last visited more than 60 days ago

5.

Select patients 18 years old or younger

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

Solution :

Step 1:

hdfs dfs -mkdir sparksql3

hdfs dfs -put patients.csv sparksql3/

Step 2 : Now in spark shell

// SQLContext entry point for working with structured data

val sqlContext = neworg.apache.spark.sql.SQLContext(sc)

// 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 patients = sc.textFilef\\'sparksqIS/patients.csv")

// Return the first element in this RDD

patients.first()

//define the schema using a case class

case class Patient(patientid: Integer, name: String, dateOfBirth:String , lastVisitDate:

String)

// create an RDD of Product objects

val patRDD = patients.map(_.split(M,M)).map(p => Patient(p(0).tolnt,p(1),p(2),p(3)))

patRDD.first()

patRDD.count{}

// change RDD of Product objects to a DataFrame val patDF = patRDD.toDF()

// register the DataFrame as a temp table patDF.registerTempTable("patients"}

// Select data from table

val results = sqlContext.sql(.....SELECT* FROM patients \\'.....)

// display dataframe in a tabular format

results.show()

//Find all the patients whose lastVisitDate between current time and \\'2012-09-15\\'

val results = sqlContext.sql(.....SELECT * FROM patients WHERE

TO_DATE(CAST(UNIX_TIMESTAMP(lastVisitDate, \\'yyyy-MM-dd\\') AS TIMESTAMP))

BETWEEN \\'2012-09-15\\' AND current_timestamp() ORDER BY lastVisitDate.....)

results.showQ

/.Find all the patients who born in 2011

val results = sqlContext.sql(.....SELECT * FROM patients WHERE

YEAR(TO_DATE(CAST(UNIXJTIMESTAMP(dateOfBirth, \\'yyyy-MM-dd\\') AS

TIMESTAMP))) = 2011 .....)

results. show()

//Find all the patients age

val results = sqlContext.sql(.....SELECT name, dateOfBirth, datediff(current_date(),

TO_DATE(CAST(UNIX_TIMESTAMP(dateOfBirth, \\'yyyy-MM-dd\\') AS TIMESTAMP}}}/365

AS age

FROM patients

Mini >

results.show() //List patients whose last visited more than 60 days ago -- List patients whose last visited more than 60 days ago val results = sqlContext.sql(.....SELECT name, lastVisitDate FROM patients WHERE

datediff(current_date(), TO_DATE(CAST(UNIX_TIMESTAMP[lastVisitDate, \\'yyyy-MM-dd\\')

AS T1MESTAMP))) > 60.....);

results. showQ;

-- Select patients 18 years old or younger

SELECT\\' FROM patients WHERE TO_DATE(CAST(UNIXJTIMESTAMP(dateOfBirth,

\\'yyyy-MM-dd\\') AS TIMESTAMP}) > DATE_SUB(current_date(),INTERVAL 18 YEAR);

val results = sqlContext.sql(.....SELECT\\' FROM patients WHERE

TO_DATE(CAST(UNIX_TIMESTAMP(dateOfBirth, \\'yyyy-MM--dd\\') AS TIMESTAMP)) >

DATE_SUB(current_date(), T8*365).....);

results. showQ;

val results = sqlContext.sql(.....SELECT DATE_SUB(current_date(), 18*365) FROM

patients.....);

results.show();

QUESTION 5

Problem Scenario 13 : 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.

Create a table in retailedb with following definition.

CREATE table departments_export (department_id int(11), department_name varchar(45),

created_date T1MESTAMP DEFAULT NOWQ);

2.

Now import the data from following directory into departments_export table,

/user/cloudera/departments new

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

Solution :

Step 1 : Login to musql db

mysql --user=retail_dba -password=cloudera

show databases; use retail_db; show tables;

step 2 : Create a table as given in problem statement.

CREATE table departments_export (departmentjd int(11), department_name varchar(45),

created_date T1MESTAMP DEFAULT NOW());

show tables;

Step 3 : Export data from /user/cloudera/departmentsnew to new table departments_export

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

-username retaildba \

--password cloudera \

--table departments_export \

-export-dir /user/cloudera/departments_new \

-batch

Step 4 : Now check the export is correctly done or not. mysql -user*retail_dba password=cloudera

show databases;

use retail _db;

show tables;

select\\' from departments_export;