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

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

QUESTION 1

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 2

Problem Scenario 40 : You have been given sample data as below in a file called spark15/file1.txt 3070811,1963,1096,,"US","CA",,1, 3022811,1963,1096,,"US","CA",,1,56 3033811,1963,1096,,"US","CA",,1,23 Below is the code snippet to process this tile. val field= sc.textFile("spark15/f ilel.txt") val mapper = field.map(x=> A) mapper.map(x => x.map(x=> {B})).collect

Please fill in A and B so it can generate below final output

Array(Array(3070811,1963,109G, 0, "US", "CA", 0,1, 0)

,Array(3022811,1963,1096, 0, "US", "CA", 0,1, 56)

,Array(3033811,1963,1096, 0, "US", "CA", 0,1, 23)

)

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

Solution :

A. x.split(",”-1)

B. if (x. isEmpty) 0 else x

QUESTION 3

Problem Scenario 10 : 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 database named hadoopexam and then create a table named departments in it, with following fields. department_id int, department_name string

e.g. location should be hdfs://quickstart.cloudera:8020/user/hive/warehouse/hadoopexam.db/departments

2.

Please import data in existing table created above from retaidb.departments into hive table hadoopexam.departments.

3.

Please import data in a non-existing table, means while importing create hive table named hadoopexam.departments_new

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

Solution :

Step 1 : Go to hive interface and create database.

hive

create database hadoopexam;

Step 2. Use the database created in above step and then create table in it. use

hadoopexam; show tables;

Step 3 : Create table in it.

create table departments (department_id int, department_name string);

show tables;

desc departments;

desc formatted departments;

Step 4 : Please check following directory must not exist else it will give error, hdfs dfs -ls

/user/cloudera/departments

If directory already exists, make sure it is not useful and than delete the same.

This is the staging directory where Sqoop store the intermediate data before pushing in

hive table.

hadoop fs -rm -R departments

Step 5 : Now import data in existing table

sqoop import \

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

~username=retail_dba \

-password=cloudera \

--table departments \

-hive-home /user/hive/warehouse \

-hive-import \

-hive-overwrite \

-hive-table hadoopexam.departments

Step 6 : Check whether data has been loaded or not.

hive;

use hadoopexam;

show tables;

select" from departments;

desc formatted departments;

Step 7 : Import data in non-existing tables in hive and create table while importing.

sqoop import \

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

--username=retail_dba \

~password=cloudera \

-table departments \

-hive-home /user/hive/warehouse \

-hive-import \

-hive-overwrite \

-hive-table hadoopexam.departments_new \

-create-hive-table

Step 8 : Check-whether data has been loaded or not.

hive;

use hadoopexam;

show tables;

select" from departments_new;

desc formatted departments_new;

QUESTION 4

Problem Scenario 87 : You have been given below three files product.csv (Create this file in hdfs) productID,productCode,name,quantity,price,supplierid 1001,PEN,Pen Red,5000,1.23,501 1002,PEN,Pen Blue,8000,1.25,501

1003,PEN,Pen Black,2000,1.25,501 1004,PEC,Pencil 2B,10000,0.48,502 1005,PEC,Pencil 2H,8000,0.49,502 1006,PEC,Pencil HB,0,9999.99,502 2001,PEC,Pencil 3B,500,0.52,501 2002,PEC,Pencil 4B,200,0.62,501 2003,PEC,Pencil 5B,100,0.73,501 2004,PEC,Pencil 6B,500,0.47,502 supplier.csv supplierid,name,phone 501,ABC Traders,88881111 502,XYZ Company,88882222 503,QQ Corp,88883333 products_suppliers.csv productID,supplierID 2001,501 2002,501 2003,501 2004,502 2001,503 Now accomplish all the queries given in solution. Select product, its price , its supplier name where product price is less than 0.6 using SparkSQL

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

Solution :

Step 1:

hdfs dfs -mkdir sparksql2

hdfs dfs -put product.csv sparksq!2/

hdfs dfs -put supplier.csv sparksql2/

hdfs dfs -put products_suppliers.csv sparksql2/

Step 2 : Now in spark shell

// 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 products = sc.textFile("sparksql2/product.csv")

val supplier = sc.textFileC\\'sparksq^supplier.csv")

val prdsup = sc.textFile("sparksql2/products_suppliers.csv"}

// Return the first element in this RDD

products.fi rst()

supplier.first{).

prdsup.first()

//define the schema using a case class

case class Product(productid: Integer, code: String, name: String, quantity:lnteger, price:

Float, supplierid:lnteger)

case class Suplier(supplierid: Integer, name: String, phone: String)

case class PRDSUP(productid: Integer.supplierid: Integer)

// create an RDD of Product objects

val prdRDD = products.map(_.split(\\'\")).map(p =>

Product(p(0).tolnt,p(1),p(2),p(3).tolnt,p(4).toFloat,p(5).toint))

val supRDD = supplier.map(_.split(",")).map(p => Suplier(p(0).tolnt,p(1),p(2)))

val prdsupRDD = prdsup.map(_.split(",")).map(p => PRDSUP(p(0).tolnt,p(1}.tolnt}}

prdRDD.first()

prdRDD.count()

supRDD.first() supRDD.count()

prdsupRDD.first() prdsupRDD.count{}

// change RDD of Product objects to a DataFrame

val prdDF = prdRDD.toDF()

val supDF = supRDD.toDF()

val prdsupDF = prdsupRDD.toDF()

// register the DataFrame as a temp table prdDF.registerTempTablef\\'products")

supDF.registerTempTablef\\'suppliers")
```

prdsupDF.registerTempTablef\\'productssuppliers"}

//Select product, its price , its supplier name where product price is less than 0.6

val results = sqlContext.sql(.....SELECT products.name, price, suppliers.name as

sup_name FROM products JOIN suppliers ON products.supplierlD= suppliers.supplierlD

WHERE price

results. show()

QUESTION 5

Problem Scenario 56 : You have been given below code snippet.

val a = sc.parallelize(l to 100. 3)

operation1

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

Array [Array [I nt]] = Array(Array(1, 2, 3,4, 5, 6, 7, 8, 9,10,11,12,13,14,15,16,17,18,19, 20,

21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33),

Array(34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55,

56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66),

Array(67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88,

89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100))

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

Solution : a.glom.collect glom Assembles an array that contains all elements of the partition and embeds it in an RDD. Each returned array contains the contents of one panition