DumpsCertify

Cloudera

CCA175

CCA Spark and Hadoop Developer Exam - Performance Base Scenarios

QUESTION & ANSWERS

DumpsCertify

Problem Scenario 90 : You have been given below two files course.txt
id,course
1,Hadoop
2,Spark
3,HBase

fee.txt id,fee
2,3900
3,4200
4,2900

Accomplish the following activities.
1. Select all the courses and their fees , whether fee is listed or not.
2. Select all the available fees and respective course. If course does not exists still list the fee
3. Select all the courses and their fees , whether fee is listed or not. However, ignore records having fee as null.

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

Step 1:
hdfs dfs -mkdir sparksql4
hdfs dfs -put course.txt sparksql4/
hdfs dfs -put fee.txt sparksql4/ Step 2 : Now in spark shell

```
val fee = sc.textFile("sparksql4/fee.txt")
// Return the first element in this RDD
course.fi rst()
fee.fi rst()
//define the schema using a case class case class Course(id: Integer, name: String) case class Fee(id: Integer, fee: Integer)
// create an RDD of Product objects
val courseRDD = course.map(_.split(",")).map(c => Course(c(0).tolnt,c(1))) val feeRDD =fee.map(_.split(",")).map(c =>
Fee(c(0}.tolnt,c(1}.tolnt)) courseRDD.first()
courseRDD.count(} feeRDD.first() feeRDD.countQ

// change RDD of Product objects to a DataFrame val courseDF = courseRDD.toDF(} val feeDF = feeRDD.toDF{)
// register the DataFrame as a temp table courseDF. registerTempTable("course") feeDF. registerTempTablef'fee")
// Select data from table
val results = sqlContext.sql(.....SELECT' FROM course """ )
results. showQ
val results = sqlContext.sql(.....SELECT' FROM fee.....)
results. showQ
val results = sqlContext.sql(.....SELECT * FROM course LEFT JOIN fee ON course.id =
fee.id.....)
results-showQ
val results ="sqlContext.sql(.....SELECT * FROM course RIGHT JOIN fee ON course.id =
fee.id "MM )
results. showQ
val results = sqlContext.sql(.....SELECT' FROM course LEFT JOIN fee ON course.id =
fee.id where fee.id IS NULL"
results. show()
```

Question No : 2 CORRECT TEXT

Problem Scenario 78 : You have been given MySQL DB with following details.

database=retail_db table=retail_db.orders table=retail_db.order_items
jdbc URL = jdbc:mysql://quickstart:3306/retail_db
Columns of order table : (orderid , order_date , order_customer_id, order_status) Columns of ordeMtems table : (order_item_td , order_item_order_id ,
order_item_product_id, order_item_quantity,order_item_subtotal,order_item_product_price)

Please accomplish following activities.

1. Copy "retail_db.orders" and "retail_db.order_items" table to hdfs in respective directory p92_orders and p92_order_items .
2. Join these data using order_id in Spark and Python
3. Calculate total revenue perday and per customer
4. Calculate maximum revenue customer

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

Step 1 : Import Single table .
sqoop import --connect jdbc:mysql://quickstart:3306/retail_db -username=retail_dba - password=cloudera -table=orders --target-dir=p92_orders –m 1
sqoop import -connect jdbc:mysql://quickstart:3306/retail_db -username=retail_dba - password=cloudera -table=order_items --target-dir=p92_order_orderitems --m 1

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 : Read the data from one of the partition, created using above command, hadoop fs -cat p92_orders/part-m-00000 hadoop fs -cat p92 orderitems/part-m-00000

```
#First value is orderjd
orders Key Value = orders.map(lambda line: (int(line.split(",")[0]), line))

#Second value as an Orderjd
orderltemsKeyValue = orderltems.map(lambda line: (int(line.split(",")[1]), line))

Step 5 : Join both the RDD using orderjd
joinedData = orderltemsKeyValue.join(ordersKeyValue)
#print the joined data
for line in joinedData.collect():
print(line)
#Format of joinedData as below.
#[Orderld, 'All columns from orderltemsKeyValue', 'All columns from ordersKeyValue'] ordersPerDatePerCustomer =
joinedData.map(lambda line: ((line[1][1].split(",")[1], line[1][1].split(",M)[2]), float(line[1][0].split(",")[4])))
amountCollectedPerDayPerCustomer = ordersPerDatePerCustomer.reduceByKey(lambda runningSum, amount:
runningSum + amount}
#(Out record format will be ((date,customer_id), totalAmount} for line in amountCollectedPerDayPerCustomer.collect():
print(line)

#now change the format of record as (date,(customer_id,total_amount)) revenuePerDatePerCustomerRDD =
amountCollectedPerDayPerCustomer.map(lambda threeElementTuple: (threeElementTuple[0][0],
(threeElementTuple[0][1],threeElementTuple[1])))
for line in revenuePerDatePerCustomerRDD.collect():
print(line)

#Calculate maximum amount collected by a customer for each day perDateMaxAmountCollectedByCustomer =
revenuePerDatePerCustomerRDD.reduceByKey(lambda runningAmountTuple, newAmountTuple:
(runningAmountTuple if runningAmountTuple[1] >= newAmountTuple[1] else newAmountTuple})
```

Question No : 3 CORRECT TEXT

Problem Scenario 88 : 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

2001,501
2002,501
2003,501
2004,502
2001,503
Now accomplish all the queries given in solution.

1. It is possible that, same product can be supplied by multiple supplier. Now find each product, its price according to each supplier.
2. Find all the supllier name, who are supplying 'Pencil 3B'
3. Find all the products , which are supplied by ABC Traders.

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

Step 1 : It is possible that, same product can be supplied by multiple supplier. Now find each product, its price according to each supplier.
val results = sqlContext.sql(.....SELECT products.name AS Product Name', price, suppliers.name AS Supplier Name'
FROM products_suppliers
JOIN products ON products_suppliers.productlD = products.productID JOIN suppliers ON
products_suppliers.supplierlD = suppliers.supplierlD
null t results.show()

Step 2 : Find all the supllier name, who are supplying 'Pencil 3B'
val results = sqlContext.sql(.....SELECT p.name AS 'Product Name", s.name AS "Supplier
Name'

JOIN suppliers AS s ON ps.supplierlD = s.supplierlD WHERE p.name = 'Pencil 3B"",M)
results.show()

Step 3 : Find all the products , which are supplied by ABC Traders.
val results = sqlContext.sql(.....SELECT p.name AS 'Product Name", s.name AS "Supplier Name'
FROM products AS p, products_suppliers AS ps, suppliers AS s WHERE p.productID = ps.productID AND ps.supplierlD = s.supplierlD AND s.name = 'ABC Traders".....)
results. show()

Question No : 4 CORRECT TEXT

Problem Scenario 63 : You have been given below code snippet.

val a = sc.parallelize(List("dog", "tiger", "lion", "cat", "panther", "eagle"), 2)
val b = a.map(x => (x.length, x))

operation1

Write a correct code snippet for operationl which will produce desired output, shown below. Array[(lnt, String}] = Array((4,lion), (3,dogcat), (7,panther), (5,tigereagle))

Answer: See the explanation for Step by Step Solution and configuration.
Explanation:
Solution :
b.reduceByKey(_ + _).collect
reduceByKey JPair] : This function provides the well-known reduce functionality in Spark. Please note that any function f you provide, should be commutative in order to generate reproducible results.

Problem Scenario 43 : You have been given following code snippet.

```
val grouped = sc.parallelize(Seq(((1,"twoM), List((3,4), (5,6)))))
val flattened = grouped.flatMap {A =>
groupValues.map { value => B }
}
```

You need to generate following output. Hence replace A and B Array((1,two,3,4),(1,two,5,6))

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

A case (key, groupValues)
B (key._1, key._2, value._1, value._2)

Question No : 6 CORRECT TEXT

Problem Scenario 87 : You have been given below three files

product.csv (Create this file in hdfs)
productID,productCode,name,quantity,price,supplierid

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

```
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.impIicits._
// 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(}
```

// 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 < 0.6.....]

results. show()

Question No : 7 CORRECT TEXT

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);

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;

Question No : 8 CORRECT TEXT

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.

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

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 \SCONDITlONS" \
-target-dir /user/cloudera/order_join \
-split-by order_id \
--num-mappers 2

Step 3 : Check imported data.

Question No : 9 CORRECT TEXT

Problem Scenario 27 : You need to implement near real time solutions for collecting information when submitted in file with below information.

Data
echo "IBM,100,20160104" >> /tmp/spooldir/bb/.bb.txt echo "IBM,103,20160105" >> /tmp/spooldir/bb/.bb.txt mv /tmp/spooldir/bb/.bb.txt /tmp/spooldir/bb/bb.txt
After few mins
echo "IBM,100.2,20160104" >> /tmp/spooldir/dr/.dr.txt echo "IBM,103.1,20160105" >> /tmp/spooldir/dr/.dr.txt mv /tmp/spooldir/dr/.dr.txt /tmp/spooldir/dr/dr.txt Requirements:
You have been given below directory location (if not available than create it) /tmp/spooldir . You have a finacial subscription for getting stock prices from BloomBerg as well as
Reuters and using ftp you download every hour new files from their respective ftp site in directories /tmp/spooldir/bb and /tmp/spooldir/dr respectively.
As soon as file committed in this directory that needs to be available in hdfs in
/tmp/flume/finance location in a single directory.

Write a flume configuration file named flume7.conf and use it to load data in hdfs with following additional properties .

1. Spool /tmp/spooldir/bb and /tmp/spooldir/dr

Answer: See the explanation for Step by Step Solution and configuration.
Explanation:
Solution :
Step 1 : Create directory mkdir /tmp/spooldir/bb mkdir /tmp/spooldir/dr Step 2 : Create flume configuration file, with below configuration for agent1.sources = source1 source2
agent1 .sinks = sink1
agent1.channels = channel1

agent1 .sources.source1.channels = channel1
agentl .sources.source2.channels = channell agent1 .sinks.sinkl.channel = channell agent1 .sources.source1.type = spooldir
agent1 .sources.sourcel.spoolDir = /tmp/spooldir/bb agent1 .sources.source2.type = spooldir
agent1 .sources.source2.spoolDir = /tmp/spooldir/dr agent1 .sinks.sink1.type = hdfs
agent1 .sinks.sink1.hdfs.path = /tmp/flume/finance agent1-sinks.sink1.hdfs.filePrefix = events
agent1.sinks.sink1.hdfs.fileSuffix = .log
agent1 .sinks.sink1.hdfs.inUsePrefix = _
agent1 .sinks.sink1.hdfs.fileType = Data Stream agent1.channels.channel1.type = file
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/fIumeconf/fIume7.conf --name agent1

echo "IBM,100,20160104" » /tmp/spooldir/bb/.bb.txt
echo "IBM,103,20160105" » /tmp/spooldir/bb/.bb.txt mv /tmp/spooldir/bb/.bb.txt
/tmp/spooldir/bb/bb.txt

After few mins
echo "IBM,100.2,20160104" » /tmp/spooldir/dr/.dr.txt
echo "IBM,103.1,20160105" »/tmp/spooldir/dr/.dr.txt mv /tmp/spooldir/dr/.dr.txt
/tmp/spooldir/dr/dr.txt

Question No : 10 CORRECT TEXT

Problem Scenario 1:

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".

Explanation: 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 -Is

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

DumpsCertify

GUARANTEED SUCCESS

100%

https://www.dumpscertify.com/CCA175-exam-dumps.html