



Up-to-date Questions and Answers from authentic resources to improve knowledge and pass the exam at very first attempt. ----- Guaranteed.



CCA175 MCQs
CCA175 TestPrep
CCA175 Study Guide
CCA175 Practice Test
CCA175 Exam Questions



killexams.com

Cloudera

CCA175

CCA Spark and Hadoop Developer

ORDER FULL VERSION

<https://killexams.com/pass4sure/exam-detail/CCA175>



Question: 94

Now import the data from following directory into departments_export table, /user/cloudera/departments new

Answer: 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;
```

Question: 95

Data should be written as text to hdfs

Answer: Solution:

Step 1: Create directory mkdir /tmp/spooldir2

Step 2: Create flume configuration file, with below configuration for source, sink and channel and save it in flume8.conf.

```
agent1.sources = source1
```

```
agent1.sinks = sink1a sink1b agent1.channels = channel1a channel1b
```

```
agent1.sources.source1.channels = channel1a channel1b
```

```
agent1.sources.source1.selector.type = replicating
```

```
agent1.sources.source1.selector.optional = channel1b
```

```
agent1.sinks.sink1a.channel = channel1a
```

```
agent1.sinks.sink1b.channel = channel1b
```

```
agent1.sources.source1.type = spooldir
```

```
agent1.sources.source1.spoolDir = /tmp/spooldir2
```

```
agent1.sinks.sink1a.type = hdfs
```

```
agent1.sinks.sink1a.hdfs.path = /tmp/flume/primary
```

```
agent1.sinks.sink1a.hdfs.tilePrefix = events
```

```
agent1.sinks.sink1a.hdfs.fileSuffix = .log
```

```
agent1.sinks.sink1a.hdfs.fileType = Data Stream
```

```
agent1.sinks.sink1b.type = hdfs
```

```
agent1.sinks.sink1b.hdfs.path = /tmp/flume/secondary
```

```
agent1.sinks.sink1b.hdfs.filePrefix = events
```

```
agent1.sinks.sink1b.hdfs.fileSuffix = .log
```

```
agent1.sinks.sink1b.hdfs.fileType = Data Stream
```

```
agent1.channels.channel1a.type = file
```

```
agent1.channels.channel1b.type = memory
```

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/flume8.conf --name age
```

Step 5: Open another terminal and create a file in /tmp/spooldir2/

```
echo "IBM, 100, 20160104" >> /tmp/spooldir2/bb.txt
```

```
echo "IBM, 103, 20160105" >> /tmp/spooldir2/bb.txt mv /tmp/spooldir2/bb.txt /tmp/spooldir2/bb.txt
```

After few mins

```
echo "IBM.100.2, 20160104" >>/tmp/spooldir2/dr.txt
```

```
echo "IBM, 103.1, 20160105" >> /tmp/spooldir2/dr.txt mv /tmp/spooldir2/dr.txt /tmp/spooldir2/dr.txt
```

Question: 96

Data should be written as text to hdfs

Answer: 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
```

```
agent1 .sources.source2.channels = channel1 agent1 .sinks.sink1.channel = channel1
```

```
agent1 . sources.source1.type = spooldir
```

```
agent1 .sources.source1.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/flumeconf/flume7.conf --name agent1
```

Step 5: Open another terminal and create a file in /tmp/spooldir/

```
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: 97

Data should be written as text to hdfs

Answer: Solution:

Step 1: Create directory mkdir /tmp/spooldir2

Step 2: Create flume configuration file, with below configuration for source, sink and channel and save it in flume8.conf.

```
agent1 .sources = source1
```

```
agent1.sinks = sink1a sink1b agent1.channels = channel1a channel1b
```

```
agent1.sources.source1.channels = channel1a channel1b
```

```
agent1.sources.source1.selector.type = replicating
```

```
agent1.sources.source1.selector.optional = channel1b
```

```
agent1.sinks.sink1a.channel = channel1a
```

```
agent1 .sinks.sink1b.channel = channel1b
```

```
agent1.sources.source1.type = spooldir
```

```
agent1 .sources.source1.spoolDir = /tmp/spooldir2
```

```
agent1.sinks.sink1a.type = hdfs
```

```
agent1 .sinks.sink1a.hdfs.path = /tmp/flume/primary
```

```
agent1 .sinks.sink1a.hdfs.tilePrefix = events
```

```
agent1 .sinks.sink1a.hdfs.fileSuffix = .log
```

```
agent1 .sinks.sink1a.hdfs.fileType = Data Stream
```

```
agent1 .sinks.sink1b.type = hdfs
```

```
agent1 .sinks.sink1b.hdfs.path = /tmp/flume/secondary
```

```
agent1 .sinks.sink1b.hdfs.filePrefix = events
```

```
agent1.sinks.sink1b.hdfs.fileSuffix = .log
```

```
agent1 .sinks.sink1b.hdfs.fileType = Data Stream
```

```
agent1.channels.channel1a.type = file
```

```
agent1.channels.channel1b.type = memory
```

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/flume8.conf --name age
```

Step 5: Open another terminal and create a file in /tmp/spooldir2/

```
echo "IBM, 100, 20160104" >> /tmp/spooldir2/.bb.txt
```

```
echo "IBM, 103, 20160105" >> /tmp/spooldir2/.bb.txt mv /tmp/spooldir2/.bb.txt /tmp/spooldir2/bb.txt
```

After few mins

```
echo "IBM.100.2, 20160104" >>/tmp/spooldir2/.dr.txt
```

```
echo "IBM, 103.1, 20160105" >> /tmp/spooldir2/.dr.txt mv /tmp/spooldir2/.dr.txt /tmp/spooldir2/dr.txt
```

Question: 98

Data should be written as text to hdfs

Answer: 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 ABCTech m.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
```

Question: 99

Problem Scenario 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.

Import Single table categories (Subset data) to hive managed table, where category_id between 1 and 22

Answer: Solution:

Step 1: Import Single table (Subset data)

```
sqoop import --connect jdbc:mysql://quickstart:3306/retail_db -username=retail_dba -password=cloudera -
table=categories -where " ' category_id ' between 1 and 22" --hive-import --m 1
```

Note: Here the ' is the same you find on ~ key

This command will create a managed table and content will be created in the following directory.

/user/hive/warehouse/categories

Step 2: Check whether table is created or not (In Hive)

```
show tables;
```

```
select * from categories;
```

Question: 100

Data should be written as text to hdfs

Answer: 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
```

```
agent1 .sources.source2.channels = channel1 agent1 .sinks.sink1.channel = channel1
```

```
agent1 . sources.source1.type = spooldir
```

```
agent1 .sources.source1.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/flumeconf/flume7.conf --name agent1
```

Step 5: Open another terminal and create a file in /tmp/spooldir/

```
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: 101

Problem Scenario 21: You have been given log generating service as below.

startjogs (It will generate continuous logs)

tailjogs (You can check, what logs are being generated)

stopjogs (It will stop the log service)

Path where logs are generated using above service: /opt/gen_logs/logs/access.log

Now write a flume configuration file named flumel.conf, using that configuration file MCQs logs in HDFS file system in a directory called flumel. Flume channel should have following property as well. After every 100 message it should be committed, use non-durable/faster channel and it should be able to hold maximum 1000 events

Answer: Solution:

Step 1: Create flume configuration file, with below configuration for source, sink and channel.

#Define source, sink, channel and agent,

```
agent1.sources = source1
```

```
agent1.sinks = sink1
```

```
agent1.channels = channel1
```

Describe/configure source1

```
agent1.sources.source1.type = exec
```

```
agent1.sources.source1.command = tail -F /opt/gen logs/logs/access.log
```

Describe sink1

```
agent1.sinks.sink1.channel = memory-channel
```

```
agent1.sinks.sink1.type = hdfs
```

```
agent1.sinks.sink1.hdfs.path = flume1
```

```
agent1.sinks.sink1.hdfs.fileType = Data Stream
```

Now we need to define channel property.

```
agent1.channels.channel1.type = memory
```

```
agent1.channels.channel1.capacity = 1000
```

```
agent1.channels.channel1.transactionCapacity = 100
```

Bind the source and sink to the channel

```
agent1.sources.source1.channels = channel1
```

```
agent1.sinks.sink1.channel = channel1
```

Step 2: Run below command which will use this configuration file and append data in hdfs.

Start log service using: startjogs

Start flume service:

```
flume-ng agent -conf /home/cloudera/flumeconf -conf-file /home/cloudera/flumeconf/flume1.conf -Dflume.root.logger=DEBUG, INFO, console
```

Wait for few mins and then stop log service.

Stop_logs

Question: 102

Problem Scenario 23: You have been given log generating service as below.

Start_logs (It will generate continuous logs)

Tail_logs (You can check, what logs are being generated)

Stop_logs (It will stop the log service)

Path where logs are generated using above service: /opt/gen_logs/logs/access.log

Now write a flume configuration file named flume3.conf, using that configuration file MCQs logs in HDFS file system in a directory called flumeflume3/%Y/%m/%d/%H/%M

Means every minute new directory should be created). Please use the interceptors to provide timestamp information, if message header does not have header info.

And also note that you have to preserve existing timestamp, if message contains it. Flume channel should have following property as well. After every 100 message it should be committed, use non-durable/faster channel and it should be able to hold maximum 1000 events.

Answer: Solution:

Step 1: Create flume configuration file, with below configuration for source, sink and channel.

#Define source, sink, channel and agent,

```
agent1.sources = source1
```

```
agent1.sinks = sink1
```

```
agent1.channels = channel1
```

Describe/configure source1

```
agent1.sources.source1.type = exec
```

```
agent1.sources.source1.command = tail -F /opt/gen_logs/logs/access.log
```

#Define interceptors

```
agent1.sources.source1.interceptors=i1
```

```
agent1.sources.source1.interceptors.i1.type=timestamp
```

```
agent1.sources.source1.interceptors.i1.preserveExisting=true
```

Describe sink1

```
agent1 .sinks.sink1.channel = memory-channel
```

```
agent1 . sinks.sink1.type = hdfs
```

```
agent1 . sinks.sink1.hdfs.path = flume3/%Y/%m/%d/%H/%M
```

```
agent1 .sinks.sjnk1.hdfs.fileType = Data Stream
```

Now we need to define channel1 property.

```
agent1.channels.channel1.type = memory
```

```
agent1.channels.channel1.capacity = 1000
```

```
agent1.channels.channel1.transactionCapacity = 100
```

Bind the source and sink to the channel

```
Agent1.sources.source1.channels = channel1
```

```
agent1.sinks.sink1.channel = channel1
```

Step 2: Run below command which will use this configuration file and append data in hdfs.

Start log service using: start_logs

Start flume service:

```
flume-ng agent -conf /home/cloudera/flumeconf -conf-file /home/cloudera/flumeconf/flume3.conf -  
Dflume.root.logger=DEBUG, INFO, console Cname agent1
```

Wait for few mins and than stop log service.

stop logs

Question: 103

Problem Scenario 21: You have been given log generating service as below.

startjogs (It will generate continuous logs)

tailjogs (You can check, what logs are being generated)

stopjogs (It will stop the log service)

Path where logs are generated using above service: /opt/gen_logs/logs/access.log

Now write a flume configuration file named flumel.conf, using that configuration file MCQs logs in HDFS file system in a directory called flumel. Flume channel should have following property as well. After every 100 message it should be committed, use non-durable/faster channel and it should be able to hold maximum 1000 events

Answer: Solution:

Step 1: Create flume configuration file, with below configuration for source, sink and channel.

#Define source, sink, channel and agent,

```
agent1.sources = source1
```

```
agent1.sinks = sink1
```

```
agent1.channels = channel1
```

Describe/configure source1

```
agent1.sources.source1.type = exec
```

```
agent1.sources.source1.command = tail -F /opt/gen logs/logs/access.log
```

Describe sink1

```
agent1.sinks.sink1.channel = memory-channel
```

```
agent1.sinks.sink1.type = hdfs
```

```
agent1.sinks.sink1.hdfs.path = flume1
```

```
agent1.sinks.sink1.hdfs.fileType = Data Stream
```

Now we need to define channel1 property.

```
agent1.channels.channel1.type = memory
```

```
agent1.channels.channel1.capacity = 1000
```

```
agent1.channels.channel1.transactionCapacity = 100
```

Bind the source and sink to the channel

```
agent1.sources.source1.channels = channel1
```

```
agent1.sinks.sink1.channel = channel1
```

Step 2: Run below command which will use this configuration file and append data in hdfs.

Start log service using: startjogs

Start flume service:

```
flume-ng agent -conf /home/cloudera/flumeconf -conf-file /home/cloudera/flumeconf/flume1.conf-Dflume.root.logger=DEBUG, INFO, console
```

Wait for few mins and then stop log service.

Stop_logs

Question: 104

Now import data from mysql table departments to this hive table. Please make sure that data should be visible using below hive command, select" from departments_hive

Answer: Solution:

Step 1: Create hive table as said.

```
hive  
  
show tables;  
  
create table departments_hive(department_id int, department_name string);
```

Step 2: The important here is, when we create a table without delimiter fields. Then default delimiter for hive is ^A (01). Hence, while importing data we have to provide proper delimiter.

```
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 departments_hive  
  
-fields-terminated-by '01'
```

Step 3: Check-the data in directory.

```
hdfs dfs -ls /user/hive/warehouse/departments_hive  
  
hdfs dfs -cat/user/hive/warehouse/departmentshive/part'
```

Check data in hive table.

```
Select * from departments_hive;
```

Question: 105

Import departments table as a text file in /user/cloudera/departments.

Answer: Solution:

Step 1: List tables using sqoop

```
sqoop list-tables --connect jdbc:mysql://quickstart:3306/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 orderItems"
```

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 -ls /user/cloudera/departments
```

```
hdfs dfs -ls /user/hive/warehouse/retailstage.db
```

```
hdfs dfs -ls /user/hive/warehouse/retail_stage.db/products
```

Question: 106

Problem Scenario 2:

There is a parent organization called "ABC Group Inc", which has two child companies named Tech Inc and MPTech.

Both companies employee information is given in two separate text file as below. Please do the following activity for employee details.

Tech Inc.txt

- 1, Alok, Hyderabad↵
- 2, Krish, Hongkong↵
- 3, Jyoti, Mumbai↵
- 4, Atul, Bangalore↵
- 5, Ishan, Gurgaon↵

MPTech.txt↵

- 6, John, Newyork↵
- 7, alp2004, California↵
- 8, Tellme, Mumbai↵
- 9, Gagan21, Pune↵
- 10, Mukesh, Chennai↵
- 1.↵

Which command will you use to check all the available command line options on HDFS and How will you get the Help for individual command.↵

2. Create a new Empty Directory named Employee using Command line. And also create an empty file named in it Techinc.txt↵
3. Load both companies Employee data in Employee directory (How to override existing file in HDFS).↵
4. Merge both the Employees data in a Single file called MergedEmployee.txt, merged files should have new line character at the end of each file content.↵
5. Upload merged file on HDFS and change the file permission on HDFS merged file, so that owner and group member can read and write, other user can read the file.↵
6. Write a command to export the individual file as well as entire directory from HDFS to local file System.↵

Answer: Solution:

Step 1: Check All Available command hdfs dfs

Step 2: Get help on Individual command hdfs dfs -help get

Step 3: Create a directory in HDFS using named Employee and create a Dummy file in it called e.g. Techinc.txt hdfs dfs -mkdir Employee

Now create an empty file in Employee directory using Hue.

Step 4: Create a directory on Local file System and then Create two files, with the given data in problems.

Step 5: Now we have an existing directory with content in it, now using HDFS command line, override this existing Employee directory. While copying these files from local file System to HDFS. `cd /home/cloudera/Desktop/ hdfs dfs -put -f Employee`

Step 6: Check All files in directory copied successfully `hdfs dfs -ls Employee`

Step 7: Now merge all the files in Employee directory, `hdfs dfs -getmerge -nl Employee MergedEmployee.txt`

Step 8: Check the content of the file. `cat MergedEmployee.txt`

Step 9: Copy merged file in Employee directory from local file system to HDFS. `hdfs dfs -put MergedEmployee.txt Employee/`

Step 10: Check file copied or not. `hdfs dfs -ls Employee`

Step 11: Change the permission of the merged file on HDFS `hdfs dfs -chmod 664 Employee/MergedEmployee.txt`

Step 12: Get the file from HDFS to local file system, `hdfs dfs -get Employee Employee_hdfs`

Question: 107

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 file (Separated by comma) for final distribution and output must be sorted by id.

id, 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

Answer: Solution:

Step 1: Create all three files in hdfs in directory called spark1 (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")
```

Killexams.com is a leading online platform specializing in high-quality certification exam preparation. Offering a robust suite of tools, including MCQs, practice tests, and advanced test engines, Killexams.com empowers candidates to excel in their certification exams. Discover the key features that make Killexams.com the go-to choice for exam success.



Exam Questions:

Killexams.com provides exam questions that are experienced in test centers. These questions are updated regularly to ensure they are up-to-date and relevant to the latest exam syllabus. By studying these questions, candidates can familiarize themselves with the content and format of the real exam.

Exam MCQs:

Killexams.com offers exam MCQs in PDF format. These questions contain a comprehensive collection of questions and answers that cover the exam topics. By using these MCQs, candidate can enhance their knowledge and improve their chances of success in the certification exam.

Practice Test:

Killexams.com provides practice test through their desktop test engine and online test engine. These practice tests simulate the real exam environment and help candidates assess their readiness for the actual exam. The practice test cover a wide range of questions and enable candidates to identify their strengths and weaknesses.

Guaranteed Success:

Killexams.com offers a success guarantee with the exam MCQs. Killexams claim that by using this materials, candidates will pass their exams on the first attempt or they will get refund for the purchase price. This guarantee provides assurance and confidence to individuals preparing for certification exam.

Updated Contents:

Killexams.com regularly updates its question bank of MCQs to ensure that they are current and reflect the latest changes in the exam syllabus. This helps candidates stay up-to-date with the exam content and increases their chances of success.