



Hadoop Ecosystem Exam

Arturo Quintanilla
Software Engineer

This project was done using **cloudera-quickstart** for ease of use
15/05/2018.



Hands-on

Tasks to accomplish

1. Create a flume agent that transfers netcat output to a hive table.
2. Use bash and Python to generate a csv file with 1000 rows and has five columns, first column value should be unique, file name created should contain the date created
3. Use the file generated in question 2 and load into a mysql table.
4. Sqoop this table into Hive.
5. Load the same CSV file HDFS and create an external table which use this source
6. Integrate this table with HBase and show results through HBase.
7. Hive Workout
8. Spark RDD Exercises



Task #1

Create a flume agent that transfers netcat output to a hive table.

First we're going to create a directory to store our logged data.

```
hadoop fs -mkdir /user/cloudera/flume/
```

We use "hadoop fs" keywords in order to create a directory inside the hadoop file system and not the local cent OS system.

Task #1

Create a flume agent that transfers netcat output to a hive table.

A flume agent consist of the following components:

- Source – Defines where the data is coming from.
 - In this task we'll be using netcat to fetch data from a configured port.
- Channel – are pipes which establish connections between sources and sinks.
 - In this task we'll be using a memory.
- Sink – Defines the destination of the data pipelined from various sources.
 - In this task we'll be using hdfs to store our logged data.

TASK #1

Create a flume agent that transfers netcat output to a hive table.

Now let's create a flume agent which will be named "netcat.conf".

Each component will be given a name so the first lines of code in our configuration file will look something like this.

```
netcatagent.sources = netcat  
netcatagent.channels = memorychannel  
netcatagent.sinks = hdfs
```

TASK #1

Create a flume agent that transfers netcat output to a hive table.

Now lets configure our source component, so our source component consist of 3 attributes:

- type – The type of source to work with.
- bind – ip of source.
- Port – ip's open port to fetch data.

```
# For each one of the sources, the type is defined
netcatagent.sources.netcat.type = netcat
netcatagent.sources.netcat.bind = localhost
netcatagent.sources.netcat.port = 56565
```

Task #1

Create a flume agent that transfers netcat output to a hive table.

Coming up next we have our memory configuration which consists of 3 attributes.

- type – Type of channel
- capacity - This is the maximum capacity number of events of the channel.
- transactionCapacity - This is the max number of events stored in the channel per transaction

```
# The channel can be defined as follows.  
netcatagent.channels.memorychannel.type = memory  
netcatagent.channels.memorychannel.capacity = 1000  
netcatagent.channels.memorychannel.transactionCapacity = 100
```

Task #1

Create a flume agent that transfers netcat output to a hive table.

We're almost done configuring our agent now we need to configure our sink. Our sink consists of three attributes which are as follows:

- Type – Type of sink to use
- hdfs.path – path to store our logged files.
- writeFormat – file format

```
# Each sink's type must be defined
#netcatagent.sinks.hdfs.fileType = Datastream
netcatagent.sinks.hdfs.type = hdfs
netcatagent.sinks.hdfs.hdfs.path =
hdfs://localhost:8022/user/cloudera/flume/logs
netcatagent.sinks.hdfs.hdfs.writeFormat = Text
```


Task #1

Create a flume agent that transfers netcat output to a hive table.

Notice that we didn't have to do any fancy configuration in order to ingest data from flume to hive. The reason why this was done this way was because as long as our hive table is referencing a n specific path in our hadoop file system we will be able to access our ingested data. So that's what we did in this line of code.

```
# Each sink's type must be defined
#netcatagent.sinks.hdfs.fileType = Datastream
netcatagent.sinks.hdfs.type = hdfs
netcatagent.sinks.hdfs.hdfs.path =
hdfs://localhost:8022/user/cloudera/flume/logs
netcatagent.sinks.hdfs.hdfs.writeFormat = Text
```

Task #1

Create a flume agent that transfers netcat output to a hive table.

Finally lets bind all our components so our agent can work correctly.

```
#Bind the source and sink to the channel
netcatagent.sources.netcat.channels = memorychannel
netcatagent.sinks.hdfs.channel = memorychannel
```

Task #1

Create a flume agent that transfers netcat output to a hive table.

No let's create an external table in hive using HUE. Notice that we're referencing the path as our flume agent.

The screenshot shows the Hue web interface for Hive. The main editor displays a Hive query to create an external table named 'networkdata' with columns 'log_thrash String', 'id STRING', and 'content STRING'. The table is located at '/user/cloudera/flume/logs/'. The query is as follows:

```
1) CREATE EXTERNAL TABLE networkData (  
2) log_thrash String,  
3) id STRING,  
4) content STRING  
5) )  
6) ROW FORMAT DELIMITED  
7) FIELDS TERMINATED BY ','  
8) LOCATION '/user/cloudera/flume/logs/'
```

Below the editor, the 'Query History' section shows a list of recent queries. The first query, executed 'a few seconds ago', is the same CREATE EXTERNAL TABLE statement. Subsequent queries include DROP TABLE, SELECT, and ALTER TABLE commands for the 'networkdata' table.

The right sidebar shows the 'Tables' section with a search bar and the message 'No tables identified.' The bottom status bar indicates the user is 'cloudera@...' and the session is 'Hue - Edito... [Downloads] [Flume] [bin] cloudera@... cloudera@...'.

Task #1

Create a flume agent that transfers netcat output to a hive table.

Now let's execute our agent. Take into count that we need to be inside of `"/usr/lib/flume-ng"` directory. The command is as follows:

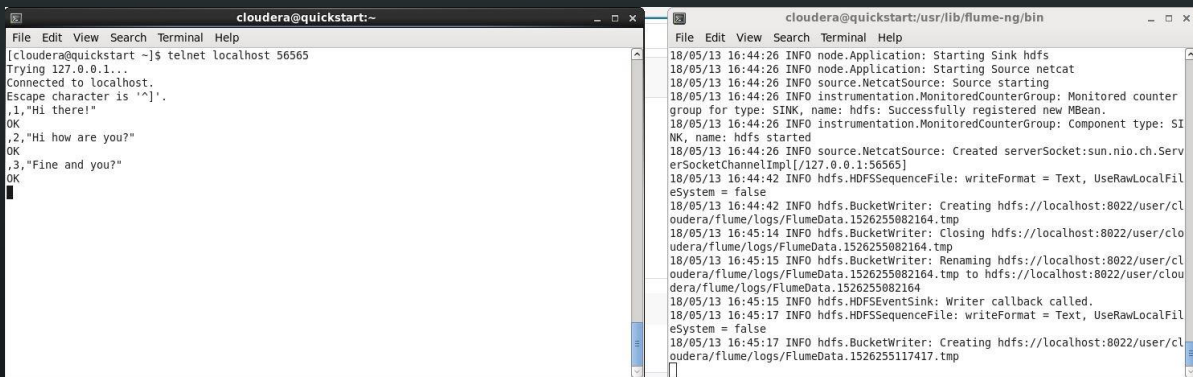
A terminal window titled 'cloudera@quickstart:/usr/lib/flume-ng/bin' is shown. The window has a menu bar with 'File', 'Edit', 'View', 'Search', 'Terminal', and 'Help'. The command prompt shows the user is in the 'bin' directory. The command entered is './flume-ng agent --conf conf --conf-file /home/cloudera/Downloads/Flume/netcat.conf --name netcatagent -Dflume.root.logger=INFO,console'. The command is partially visible on two lines.

```
cloudera@quickstart:/usr/lib/flume-ng/bin
File Edit View Search Terminal Help
[cloudera@quickstart bin]$ ./flume-ng agent --conf conf --conf-file /home/cloude
ra/Downloads/Flume/netcat.conf --name netcatagent -Dflume.root.logger=INFO,conso
le
```

Task #1

Create a flume agent that transfers netcat output to a hive table.

Now let's start ingesting data. In order to do that we first need to open a new terminal and type the following command "telnet 56565". Afterwards we'll be able to ingest data to our hive table. Just as shown below.



The image shows two terminal windows side-by-side. The left window, titled 'cloudera@quickstart:~', shows a telnet session to localhost 56565. The user connects and sends three messages: '1, "Hi there!"', '2, "Hi how are you?"', and '3, "Fine and you?"'. The right window, titled 'cloudera@quickstart:usr/lib/flume-ng/bin', shows the logs of a flume agent. The logs indicate that the agent is starting, the netcat source is starting, and the hdfs sink is successfully registered. The logs also show that the agent is writing data to the hdfs sink, with the file name 'hdfs://localhost:8022/user/cloudera/flume/logs/FlumeData.1526255082164.tmp'.

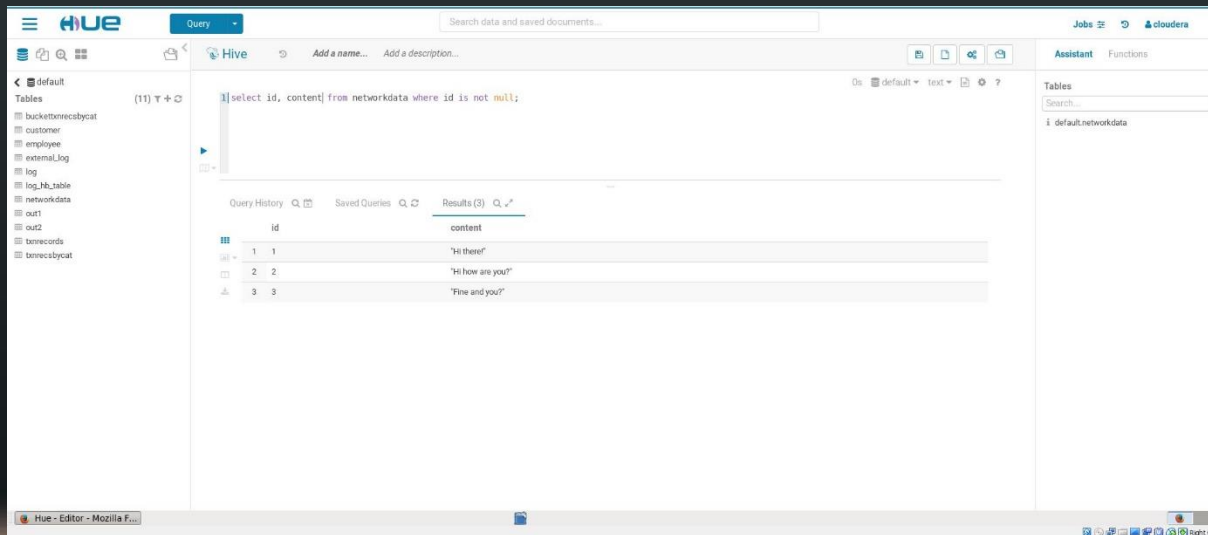
```
cloudera@quickstart:~  
File Edit View Search Terminal Help  
[cloudera@quickstart ~]$ telnet localhost 56565  
Trying 127.0.0.1...  
Connected to localhost.  
Escape character is '^['.  
,1,"Hi there!"  
OK  
,2,"Hi how are you?"  
OK  
,3,"Fine and you?"  
OK
```

```
cloudera@quickstart:usr/lib/flume-ng/bin  
File Edit View Search Terminal Help  
18/05/13 16:44:26 INFO node.Application: Starting Sink hdfs  
18/05/13 16:44:26 INFO node.Application: Starting Source netcat  
18/05/13 16:44:26 INFO source.NetcatSource: Source starting  
18/05/13 16:44:26 INFO instrumentation.MonitoredCounterGroup: Monitored counter  
group for type: SINK, name: hdfs: Successfully registered new MBean.  
18/05/13 16:44:26 INFO instrumentation.MonitoredCounterGroup: Component type: SI  
NK, name: hdfs started  
18/05/13 16:44:26 INFO source.NetcatSource: Created serverSocket:sun.nio.ch.Serv  
erSocketChannelImpl[/127.0.0.1:56565]  
18/05/13 16:44:42 INFO hdfs.HDFSSequenceFile: writeFormat = Text, UseRawLocalFil  
eSystem = false  
18/05/13 16:44:42 INFO hdfs.BucketWriter: Creating hdfs://localhost:8022/user/cl  
oudera/flume/logs/FlumeData.1526255082164.tmp  
18/05/13 16:45:14 INFO hdfs.BucketWriter: Closing hdfs://localhost:8022/user/cl  
oudera/flume/logs/FlumeData.1526255082164.tmp  
18/05/13 16:45:15 INFO hdfs.BucketWriter: Renaming hdfs://localhost:8022/user/cl  
oudera/flume/logs/FlumeData.1526255082164.tmp to hdfs://localhost:8022/user/cl  
oudera/flume/logs/FlumeData.1526255082164  
18/05/13 16:45:15 INFO hdfs.HDFSEventSink: Writer callback called.  
18/05/13 16:45:17 INFO hdfs.HDFSSequenceFile: writeFormat = Text, UseRawLocalFil  
eSystem = false  
18/05/13 16:45:17 INFO hdfs.BucketWriter: Creating hdfs://localhost:8022/user/cl  
oudera/flume/logs/FlumeData.1526255117417.tmp
```

Task #1

Create a flume agent that transfers netcat output to a hive table.

Finally let's view our ingested data from flume within our hive table.



The screenshot shows the Hue web interface for Hive. The query editor contains the following SQL query:

```
select id, content from networkdata where id is not null;
```

The query has been executed, and the results are displayed in a table with 3 rows and 2 columns: id and content.

id	content
1	"Hi there"
2	"Hi how are you?"
3	"Fine and you?"

The interface also shows a sidebar with a list of tables, including networkdata, and a right sidebar with a search bar and a list of tables.

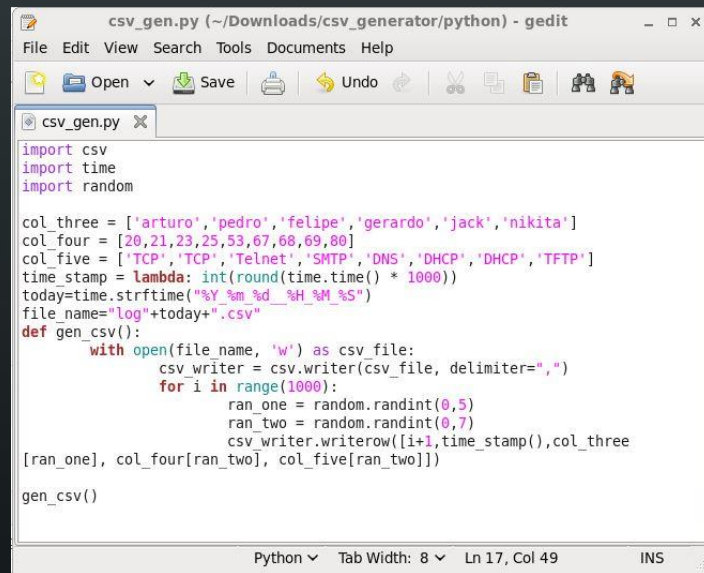
Task #2

Use bash and Python to generate a csv.

This is the python code I've developed. It's a pretty basic code which consists of records based on the following structure.

- The first column is a unique identifier.
- The second column is the current date in milliseconds.
- The three last rows are random indexes which are going to be used to access data from the three arrays I've hard coded.

On each iteration a new record will be added to the generated csv file.



```
csv_gen.py (~/Downloads/csv_generator/python) - gedit
File Edit View Search Tools Documents Help

import csv
import time
import random

col_three = ['arturo', 'pedro', 'felipe', 'gerardo', 'jack', 'nikita']
col_four = [20, 21, 23, 25, 53, 67, 68, 69, 80]
col_five = ['TCP', 'TCP', 'Telnet', 'SMTP', 'DNS', 'DHCP', 'DHCP', 'TFTP']
time_stamp = lambda: int(round(time.time() * 1000))
today = time.strftime("%Y_%m_%d_%H_%M_%S")
file_name = "log" + today + ".csv"

def gen_csv():
    with open(file_name, 'w') as csv_file:
        csv_writer = csv.writer(csv_file, delimiter=",")
        for i in range(1000):
            ran_one = random.randint(0, 5)
            ran_two = random.randint(0, 7)
            csv_writer.writerow([i+1, time_stamp(), col_three[ran_one], col_four[ran_two], col_five[ran_two]])

gen_csv()
```

Python Tab Width: 8 Ln 17, Col 49 INS

Task #2

Use bash and Python to generate a csv.

Now let's see some proof of work.

```
cloudera@quickstart:~/Downloads/csv_generator/python
File Edit View Search Terminal Help
[cloudera@quickstart python]$ python csv_gen.py
```



```
cloudera@quickstart:~/Downloads/csv_generator/python
File Edit View Search Terminal Help
978,1526365263548,arturo,68,DHCP
979,1526365263548,nikita,20,TCP
980,1526365263548,jack,53,DNS
981,1526365263548,jack,25,SMTP
982,1526365263548,felipe,67,DHCP
983,1526365263548,nikita,67,DHCP
984,1526365263548,nikita,23,Telnet
985,1526365263548,nikita,21,TCP
986,1526365263548,jack,67,DHCP
987,1526365263548,nikita,23,Telnet
988,1526365263548,gerardo,20,TCP
989,1526365263548,felipe,23,Telnet
990,1526365263548,nikita,68,DHCP
991,1526365263548,felipe,69,TFTP
992,1526365263548,felipe,53,DNS
993,1526365263548,jack,68,DHCP
994,1526365263548,nikita,53,DNS
995,1526365263548,nikita,68,DHCP
996,1526365263548,nikita,67,DHCP
997,1526365263548,gerardo,69,TFTP
998,1526365263548,nikita,53,DNS
999,1526365263548,felipe,67,DHCP
1000,1526365263548,gerardo,25,SMTP
[cloudera@quickstart python]$
```



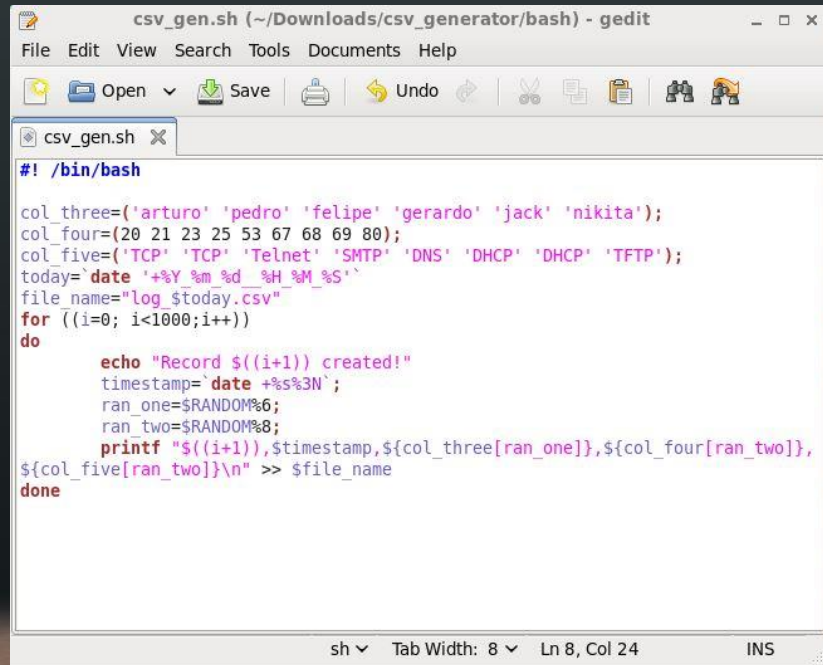
Task #2

Use bash and Python to generate a csv.

This is the bash script I've developed. It's a pretty basic script which consists of records based on the following structure.

- The first column is a unique identifier.
- The second column is the current date in milliseconds.
- The three last rows are random indexes which are going to be used to access data from the three arrays I've hard coded.

On each iteration a new record will be added to the generated csv file.



```
csv_gen.sh (~/Downloads/csv_generator/bash) - gedit
File Edit View Search Tools Documents Help

Open Save Undo

csv_gen.sh x
#!/bin/bash

col_three=('arturo' 'pedro' 'felipe' 'gerardo' 'jack' 'nikita');
col_four=(20 21 23 25 53 67 68 69 80);
col_five=('TCP' 'TCP' 'Telnet' 'SMTP' 'DNS' 'DHCP' 'DHCP' 'TFTP');
today='date +%Y %m %d %H %M %S'
file_name="log_$today.csv"
for ((i=0; i<1000;i++))
do
    echo "Record $((i+1)) created!"
    timestamp='date +%s%3N';
    ran_one=$RANDOM%6;
    ran_two=$RANDOM%8;
    printf "${(i+1)},$timestamp,${col_three[ran_one]},${col_four[ran_two]},
${col_five[ran_two]}\n" >> $file_name
done

sh Tab Width: 8 Ln 8, Col 24 INS
```

Task #2

Use bash and Python to generate a csv.

Now let's see some proof of work.

```
cloudera@quickstart:~/Downloads/csv_generator/bash
File Edit View Search Terminal Help
Record 978 created!
Record 979 created!
Record 980 created!
Record 981 created!
Record 982 created!
Record 983 created!
Record 984 created!
Record 985 created!
Record 986 created!
Record 987 created!
Record 988 created!
Record 989 created!
Record 990 created!
Record 991 created!
Record 992 created!
Record 993 created!
Record 994 created!
Record 995 created!
Record 996 created!
Record 997 created!
Record 998 created!
Record 999 created!
Record 1000 created!
[cloudera@quickstart bash]$
```

```
bash
File Edit View Places Help
csv_gen.sh log_2018_05_14_23_10_27.csv
bash 2 items, Free space: 42.5 GB
```

```
cloudera@quickstart:~/Downloads/csv_generator/bash
File Edit View Search Terminal Help
978,1526364628276,nikita,53,DNS
979,1526364628277,nikita,69,TFTP
980,1526364628278,gerardo,69,TFTP
981,1526364628279,felipe,69,TFTP
982,1526364628280,gerardo,20,TCP
983,1526364628281,gerardo,25,SMTP
984,1526364628282,felipe,25,SMTP
985,1526364628283,gerardo,25,SMTP
986,1526364628284,jack,53,DNS
987,1526364628285,nikita,23,Telnet
988,1526364628286,gerardo,25,SMTP
989,1526364628287,arturo,23,Telnet
990,1526364628288,nikita,25,SMTP
991,1526364628289,pedro,69,TFTP
992,1526364628290,arturo,67,DHCP
993,1526364628291,jack,20,TCP
994,1526364628292,pedro,69,TFTP
995,1526364628293,gerardo,53,DNS
996,1526364628294,jack,23,Telnet
997,1526364628294,jack,20,TCP
998,1526364628295,arturo,21,TCP
999,1526364628298,jack,23,Telnet
1000,1526364628299,felipe,67,DHCP
[cloudera@quickstart bash]$ cat log_2018_05_14_23_10_27.csv
```

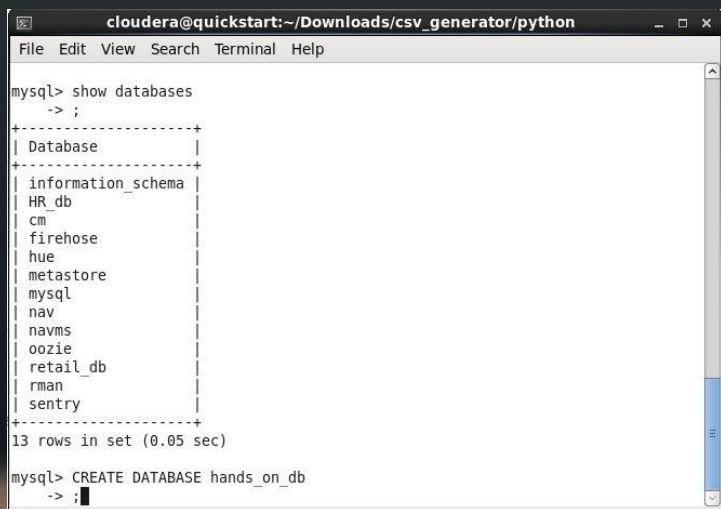


Task #3

Use the file generated in question 2 and load into a mysql table.

First of all we're going to open our terminal, start mysql and create a database.

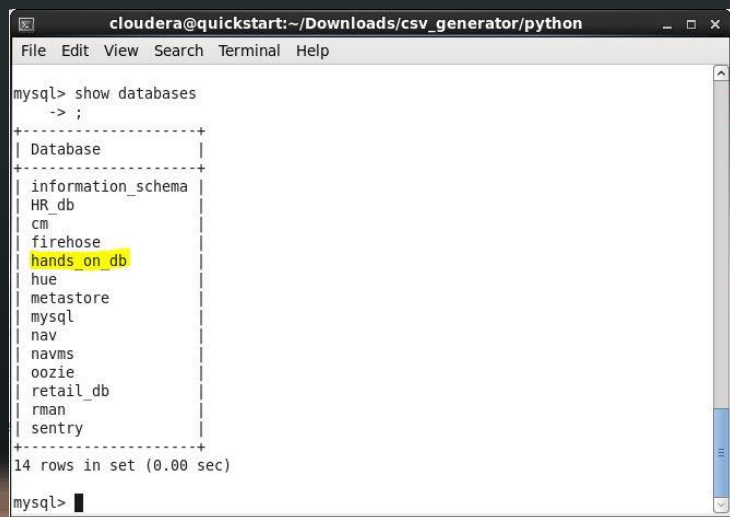
In cloudera to start mysql I used the following command: *"mysql -uroot -pcloudera"*



```
cloudera@quickstart:~/Downloads/csv_generator/python
File Edit View Search Terminal Help

mysql> show databases
-> ;
+-----+
| Database |
+-----+
| information_schema |
| HR_db      |
| cm         |
| firehose   |
| hue        |
| metastore  |
| mysql      |
| nav        |
| navms      |
| oozie       |
| retail_db  |
| rman       |
| sentry     |
+-----+
13 rows in set (0.05 sec)

mysql> CREATE DATABASE hands_on_db
-> ;
```



```
cloudera@quickstart:~/Downloads/csv_generator/python
File Edit View Search Terminal Help

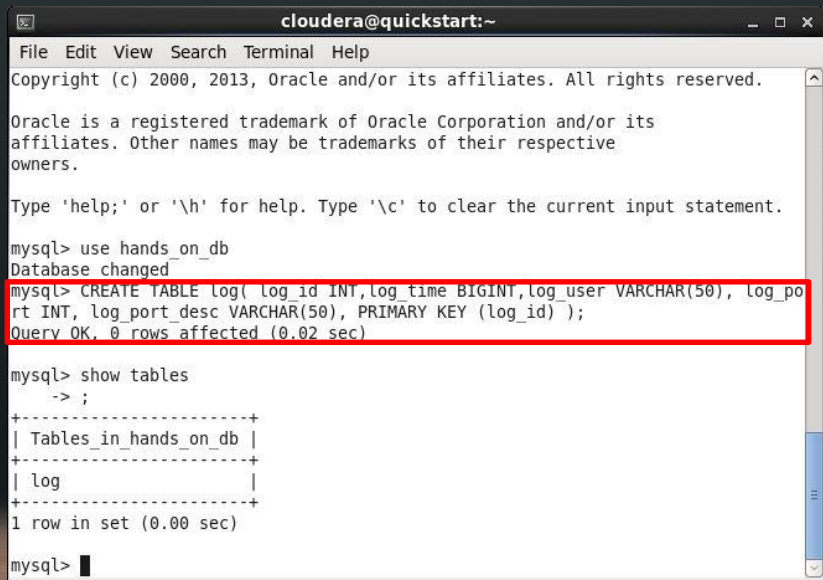
mysql> show databases
-> ;
+-----+
| Database |
+-----+
| information_schema |
| HR_db      |
| cm         |
| firehose   |
| hands_on_db |
| hue        |
| metastore  |
| mysql      |
| nav        |
| navms      |
| oozie       |
| retail_db  |
| rman       |
| sentry     |
+-----+
14 rows in set (0.00 sec)

mysql>
```

Task #3

Use the file generated in question 2 and load into a mysql table.

Now we'll create a table named "log" using the following command:

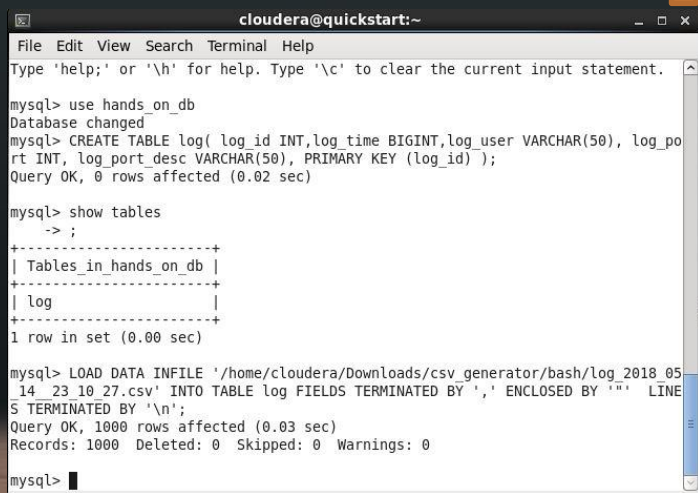
A terminal window titled 'cloudera@quickstart:~' with a menu bar (File, Edit, View, Search, Terminal, Help). It shows the MySQL command-line interface. The user has entered 'use hands_on_db' and 'show tables'. The output of 'show tables' is a table with one row: 'log'. The 'CREATE TABLE' command is highlighted with a red box.

```
cloudera@quickstart:~  
File Edit View Search Terminal Help  
Copyright (c) 2000, 2013, Oracle and/or its affiliates. All rights reserved.  
  
Oracle is a registered trademark of Oracle Corporation and/or its  
affiliates. Other names may be trademarks of their respective  
owners.  
  
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.  
  
mysql> use hands_on_db  
Database changed  
mysql> CREATE TABLE log( log_id INT,log_time BIGINT,log_user VARCHAR(50), log_po  
rt INT, log_port_desc VARCHAR(50), PRIMARY KEY (log_id) );  
Query OK, 0 rows affected (0.02 sec)  
  
mysql> show tables  
+-----+  
| Tables_in_hands_on_db |  
+-----+  
| log                     |  
+-----+  
1 row in set (0.00 sec)  
  
mysql>
```

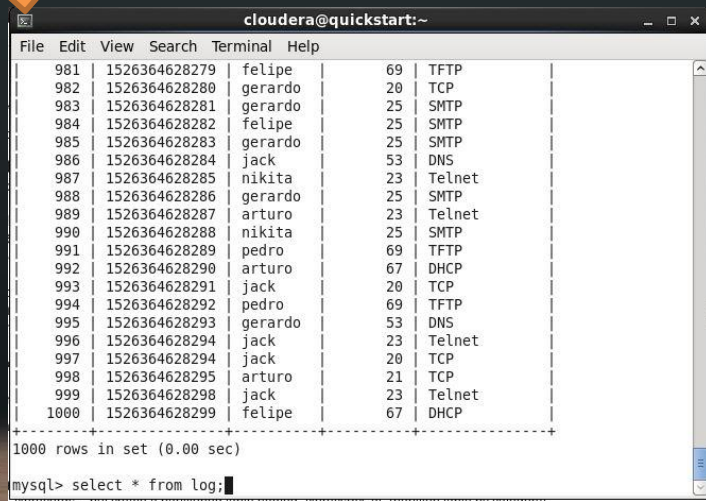
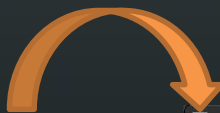
Task #3

Use the file generated in question 2 and load into a mysql table.

Finally we're going to import our data stored in our csv file to our log table in mysql and query data from it afterwards.



```
cloudera@quickstart:~  
File Edit View Search Terminal Help  
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.  
  
mysql> use hands_on_db  
Database changed  
mysql> CREATE TABLE log( log_id INT,log_time BIGINT,log_user VARCHAR(50), log_po  
rt INT, log_port_desc VARCHAR(50), PRIMARY KEY (log_id) );  
Query OK, 0 rows affected (0.02 sec)  
  
mysql> show tables  
+-----+  
| Tables_in_hands_on_db |  
+-----+  
| log                     |  
+-----+  
1 row in set (0.00 sec)  
  
mysql> LOAD DATA INFILE '/home/cloudera/Downloads/csv_generator/bash/log_2018_05  
_14_23_10_27.csv' INTO TABLE log FIELDS TERMINATED BY ',' ENCLOSED BY '"' LINE  
S TERMINATED BY '\n';  
Query OK, 1000 rows affected (0.03 sec)  
Records: 1000 Deleted: 0 Skipped: 0 Warnings: 0  
  
mysql>
```



```
cloudera@quickstart:~  
File Edit View Search Terminal Help  
981 | 1526364628279 | felipe | 69 | TFTP |  
982 | 1526364628280 | gerardo | 20 | TCP |  
983 | 1526364628281 | gerardo | 25 | SMTP |  
984 | 1526364628282 | felipe | 25 | SMTP |  
985 | 1526364628283 | gerardo | 25 | SMTP |  
986 | 1526364628284 | jack | 53 | DNS |  
987 | 1526364628285 | nikita | 23 | Telnet |  
988 | 1526364628286 | gerardo | 25 | SMTP |  
989 | 1526364628287 | arturo | 23 | Telnet |  
990 | 1526364628288 | nikita | 25 | SMTP |  
991 | 1526364628289 | pedro | 69 | TFTP |  
992 | 1526364628290 | arturo | 67 | DHCP |  
993 | 1526364628291 | jack | 20 | TCP |  
994 | 1526364628292 | pedro | 69 | TFTP |  
995 | 1526364628293 | gerardo | 53 | DNS |  
996 | 1526364628294 | jack | 23 | Telnet |  
997 | 1526364628294 | jack | 20 | TCP |  
998 | 1526364628295 | arturo | 21 | TCP |  
999 | 1526364628298 | jack | 23 | Telnet |  
1000 | 1526364628299 | felipe | 67 | DHCP |  
+-----+  
1000 rows in set (0.00 sec)  
  
mysql> select * from log;
```

Task #4

Sqoop this table into Hive.

In this next task we're going to import our created "log" table in mysql to hive. In order to do that we'll execute the following command. Which consist on the following:

1. Sqoop import (keyword for importing)
2. --connect (to define to which database to connect)
3. --username (user name for database)
4. --password (password for database)
5. --table (table to import)
6. --hive-import (special keyword to import to hive)



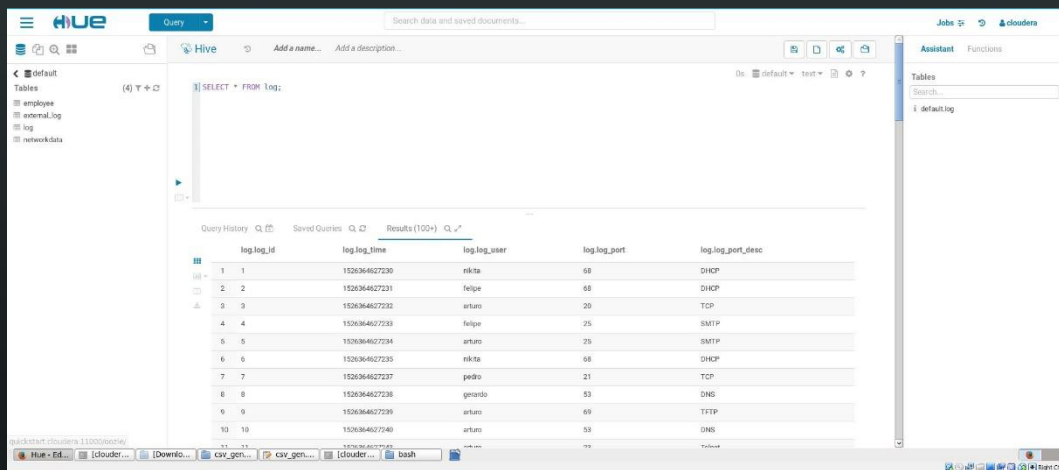
```
cloudera@quickstart:~/Downloads/csv_generator/python
File Edit View Search Terminal Help
[cloudera@quickstart python]$ sqoop import --connect "jdbc:mysql://quickstart.cloudera:3306/hands_on_db" --username=root --password=cloudera --table=log --hive-import
```

Task #4

Sqoop this table into Hive.

Finally let's showcase some proof of work. We're going to see the results of our import and query data from our hive table.

```
cloudera@quickstart:~/Downloads/csv_generator/python
File Edit View Search Terminal Help
CPU time spent (ms)=4600
Physical memory (bytes) snapshot=794894336
Virtual memory (bytes) snapshot=6288338944
Total committed heap usage (bytes)=671612928
File Input Format Counters
  Bytes Read=0
File Output Format Counters
  Bytes Written=32474
18/05/14 21:43:25 INFO mapreduce.ImportJobBase: Transferred 31.7129 KB in 72.5865 s
econds (447.3837 bytes/sec)
18/05/14 21:43:25 INFO mapreduce.ImportJobBase: Retrieved 1000 records.
18/05/14 21:43:25 INFO manager.SqlManager: Executing SQL statement: SELECT t.* FROM
`log` AS t LIMIT 1
18/05/14 21:43:25 INFO hive.HiveImport: Loading uploaded data into Hive
Logging initialized using configuration in jar:file:/usr/lib/hive/lib/hive-common-1
.1.0-cdh5.13.0.jar!/hive-log4j.properties
OK
Time taken: 4.508 seconds
Loading data to table default.log
Table default.log stats: [numFiles=4, totalSize=32474]
OK
Time taken: 1.046 seconds
[cloudera@quickstart python]$
```



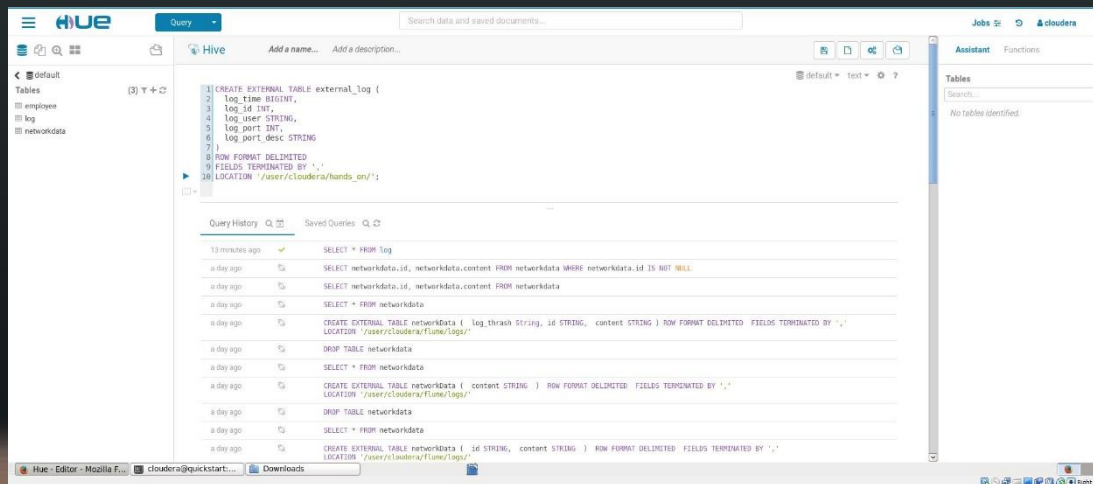
The screenshot shows the Hive web interface with a query result table. The query executed was `SELECT * FROM log;`. The results table has 5 columns: `log_log_id`, `log_log_time`, `log_log_user`, `log_log_port`, and `log_log_port_desc`. The table contains 10 rows of data.

log_log_id	log_log_time	log_log_user	log_log_port	log_log_port_desc
1	1526364627230	rikta	68	DHCP
2	1526364627231	felipe	68	DHCP
3	1526364627232	arturo	20	TCP
4	1526364627233	felipe	25	SMTP
5	1526364627234	arturo	25	SMTP
6	1526364627235	rikta	68	DHCP
7	1526364627237	pedro	21	TCP
8	1526364627238	gerardo	53	DNS
9	1526364627239	arturo	69	TFTP
10	1526364627240	arturo	53	DNS

Task #5

Load the same CSV file HDFS and create an external table which use th is source.

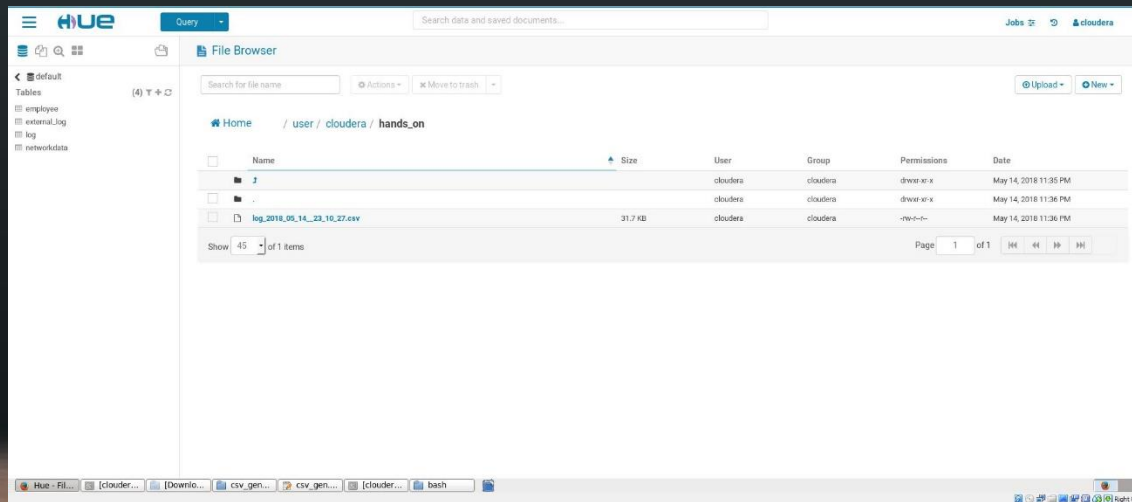
Previously we've seen how to import data from a csv to mysql and finally to a hive internal table. Now we'll be show casing how to "import" or access data stored in a csv file from a hive external table. First of all let's create an external table in hive.



Task #5

Load the same CSV file HDFS and create an external table which use this source.

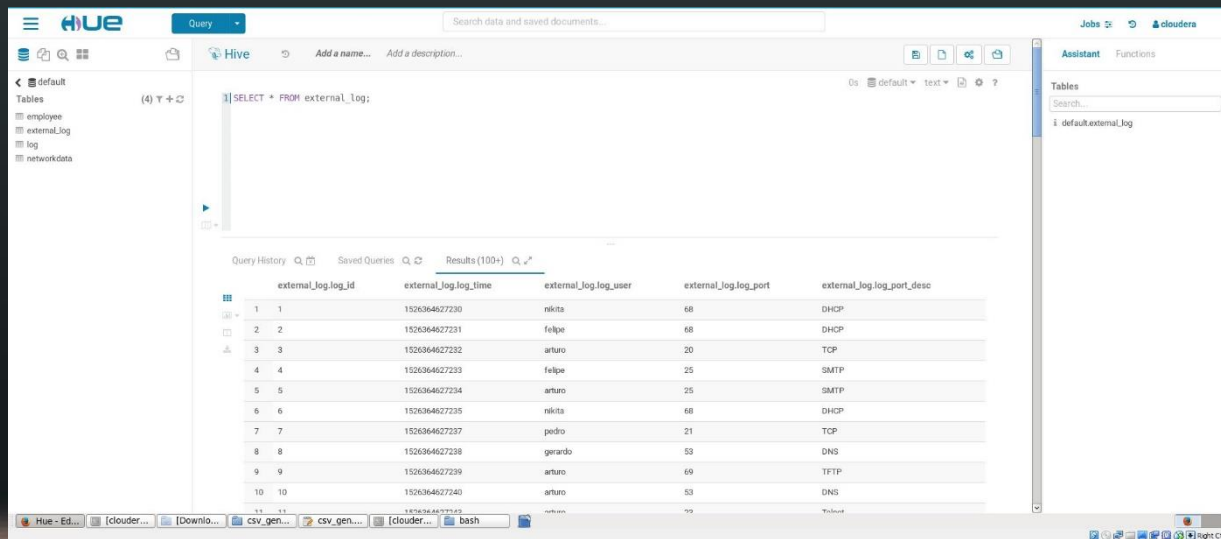
Now let's move our csv stored in our local file system to hdfs file system using HUE GUI so we can reference it from our recently created external table. In this case our path is "user/cloudera/hands_on". If you don't have "hands_on" folder please create it.



Task #5

Load the same CSV file HDFS and create an external table which use th is source.

Finally let's query the data referenced in our external table named "external_log".



The screenshot shows the Hue web interface with a Hive query executed. The query is `SELECT * FROM external_log;`. The results are displayed in a table with 5 columns: `external_log.log_id`, `external_log.log_time`, `external_log.log_user`, `external_log.log_port`, and `external_log.log_port_desc`. The table contains 10 rows of data. The interface also shows a sidebar with a file tree and a right sidebar with a table list.

	external_log.log_id	external_log.log_time	external_log.log_user	external_log.log_port	external_log.log_port_desc
1	1	1526364627230	nikita	68	DHCP
2	2	1526364627231	felipe	68	DHCP
3	3	1526364627232	arturo	20	TCP
4	4	1526364627233	felipe	25	SMTP
5	5	1526364627234	arturo	25	SMTP
6	6	1526364627235	nikita	68	DHCP
7	7	1526364627237	podro	21	TCP
8	8	1526364627238	gerardo	53	DNS
9	9	1526364627239	arturo	68	TFTP
10	10	1526364627240	arturo	53	DNS

Task #6

Integrate this table with HBase and show results through HBase.

In this task for ease of use we'll be creating an hbase table from our hive editor and then transfer data from our external_log table to our newly created hbase table.

The screenshot displays the Hue web interface. On the left, a sidebar shows a file tree with folders like 'employee', 'external_log', 'log', 'log_hb_table', and 'networkdata'. The main area is the 'Hive' query editor, which contains a SQL query to create a table named 'log_hb_table' with columns 'log_id', 'log_time', 'log_user', 'log_port', and 'log_port_desc'. The query is as follows:

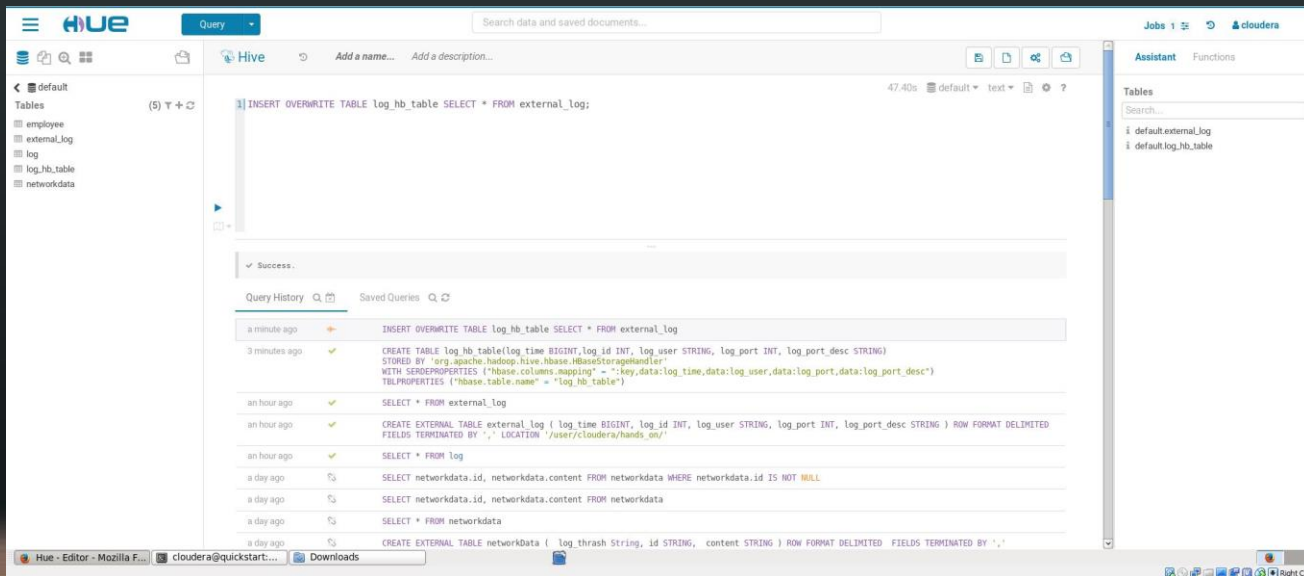
```
1 CREATE TABLE log_hb_table(log_id INT, log_time BIGINT, log_user STRING, log_port INT, log_port_desc STRING)
2 STORED BY 'org.apache.hadoop.hive.hbase.HBaseStorageHandler'
3 WITH SERDEPROPERTIES ("hbase.columns.mapping" = ":key,data:log_time,data:log_user,data:log_port,data:log_port_desc")
4 TBLPROPERTIES ("hbase.table.name" = "log_hb_table");
```

Below the query editor, a 'Query History' panel shows a list of recent queries. The most recent query is the same CREATE TABLE statement. Other queries include SELECT statements from 'external_log' and 'log_hb_table', and DROP TABLE statements for 'external_log' and 'log_hb_table'. The interface also includes a 'Jobs' tab on the right and a 'Tables' section with a search bar.

Task #6

Integrate this table with HBase and show results through HBase.

Now to transfer data from our external hive table to our hbase table we'll use the following command.



The screenshot shows the Hue web interface. The top navigation bar includes the Hue logo, a 'Query' dropdown, and a search bar. The left sidebar shows a 'Tables' list with 'employee', 'external_log', 'log', 'log_hb_table', and 'networkdata'. The main area displays a Hive query: `INSERT OVERWRITE TABLE log_hb_table SELECT * FROM external_log;`. Below the query, a 'Success.' message is shown. The 'Query History' section lists several queries with their execution times and status. The bottom of the interface shows the 'Assistant' and 'Functions' panels, and a 'Tables' list on the right side.

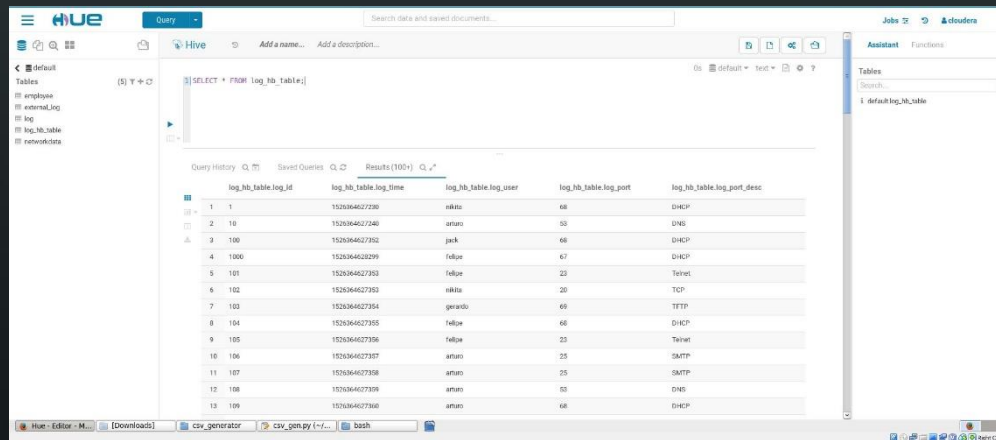
Query History:

Time	Status	Query
a minute ago	Success	INSERT OVERWRITE TABLE log_hb_table SELECT * FROM external_log
3 minutes ago	Success	CREATE TABLE log_hb_table(log_time BIGINT, log_id INT, log_user STRING, log_port INT, log_port_desc STRING) STORED BY 'org.apache.hadoop.hive.hbase.HBaseStorageHandler' WITH SERDEPROPERTIES ('hbase.columns.mapping' = ':key,data:log_time,data:log_user,data:log_port,data:log_port_desc') TBLPROPERTIES ('hbase.table.name' = 'log_hb_table')
an hour ago	Success	SELECT * FROM external_log
an hour ago	Success	CREATE EXTERNAL TABLE external_log (log_time BIGINT, log_id INT, log_user STRING, log_port INT, log_port_desc STRING) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' LOCATION '/user/cloudera/hands_on/'
an hour ago	Success	SELECT * FROM log
a day ago	Success	SELECT networkdata.id, networkdata.content FROM networkdata WHERE networkdata.id IS NOT NULL
a day ago	Success	SELECT networkdata.id, networkdata.content FROM networkdata
a day ago	Success	SELECT * FROM networkdata
a day ago	Success	CREATE EXTERNAL TABLE networkdata (log_thrash String, id STRING, content STRING) ROW FORMAT DELIMITED FIELDS TERMINATED BY ','

Task #6

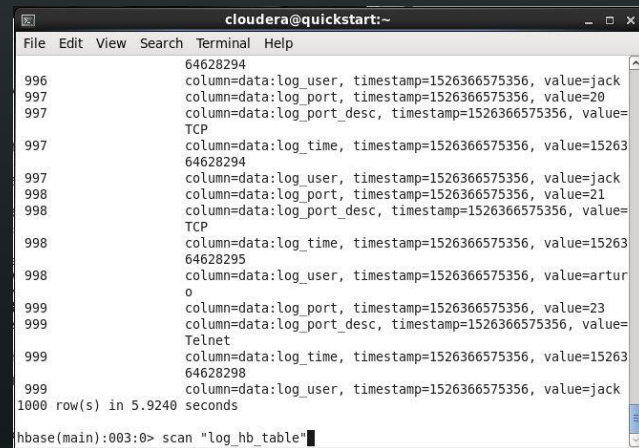
Integrate this table with HBase and show results through HBase.

Finally let's query out some data from our new table "log_hb_table" using hive editor and our hbase CLI.



The screenshot shows the Hue web interface for Hive. The query editor contains the SQL statement: `SELECT * FROM log_hb_table;`. The results pane displays a table with 100+ rows. The table has five columns: `log_hb_table.log_id`, `log_hb_table.log_time`, `log_hb_table.log_user`, `log_hb_table.log_port`, and `log_hb_table.log_port_desc`. The first 13 rows are visible in the screenshot.

	log_hb_table.log_id	log_hb_table.log_time	log_hb_table.log_user	log_hb_table.log_port	log_hb_table.log_port_desc
1	1	1526366277280	nikita	68	DHCP
2	10	1526366277280	arturo	50	DNS
3	100	1526366277352	jack	66	DHCP
4	1000	1526366282299	felipe	67	DHCP
5	101	1526366277353	felipe	23	Telnet
6	102	1526366277353	nikita	20	TCP
7	103	1526366277354	gerardo	69	TFTP
8	104	1526366277355	felipe	66	DHCP
9	105	1526366277356	felipe	23	Telnet
10	106	1526366277357	arturo	25	SMTP
11	107	1526366277358	arturo	25	SMTP
12	108	1526366277359	arturo	83	DNS
13	109	1526366277360	arturo	68	DHCP



The screenshot shows a terminal window on a Cloudera Quickstart VM. The user has executed the HBase command `scan "log_hb_table"`, which returned 1000 rows of data in 5.9240 seconds. The output shows the first few rows of the table, including columns for data, log_user, timestamp, log_port, log_port_desc, and value.

```
cloudera@quickstart:~  
File Edit View Search Terminal Help  
  
64628294  
column=data:log_user, timestamp=1526366575356, value=jack  
column=data:log_port, timestamp=1526366575356, value=20  
column=data:log_port_desc, timestamp=1526366575356, value=  
TCP  
column=data:log_time, timestamp=1526366575356, value=15263  
64628294  
column=data:log_user, timestamp=1526366575356, value=jack  
column=data:log_port, timestamp=1526366575356, value=21  
column=data:log_port_desc, timestamp=1526366575356, value=  
TCP  
column=data:log_time, timestamp=1526366575356, value=15263  
64628295  
column=data:log_user, timestamp=1526366575356, value=artur  
o  
column=data:log_port, timestamp=1526366575356, value=23  
column=data:log_port_desc, timestamp=1526366575356, value=  
Telnet  
column=data:log_time, timestamp=1526366575356, value=15263  
64628298  
column=data:log_user, timestamp=1526366575356, value=jack  
1000 row(s) in 5.9240 seconds  
  
hbase(main):003:0> scan "log_hb_table"
```

Task #7

Hive Workout

- Create a internal/managed table structure named "txnrecords" with below column/datatype for the data file txns.txt:
txnno INT, txndate STRING, custno INT, amount DOUBLE, category STRING, product STRING, city STRING, state STRING, spendby STRING

- Load the table with data file placed in HDFS

Note : Below 2 tables are to be created as external tables with some user defined location

create another table structure with similar columns like table "txnrecords" - but create a partitioned table named "txnrecsByCat" (partition table by category)

create another table structure with similar columns like table "txnrecords" - but create a table with partition and 10 buckets named "buckettxnrecsByCat" (partition by category & buckets by state)

Note : For dynamic partition - you need to execute two below hive properties before inserts

```
set hive.exec.dynamic.partition.mode=nonstrict;
```

```
set hive.exec.dynamic.partition=true;
```

Insert table "txnrecsByCat" with data from previously created table "txnrecords" (Dynamic partitioning)

Insert table "buckettxnrecsByCat" with data from previously created table "txnrecords" (Dynamic partitioning)

Task #7

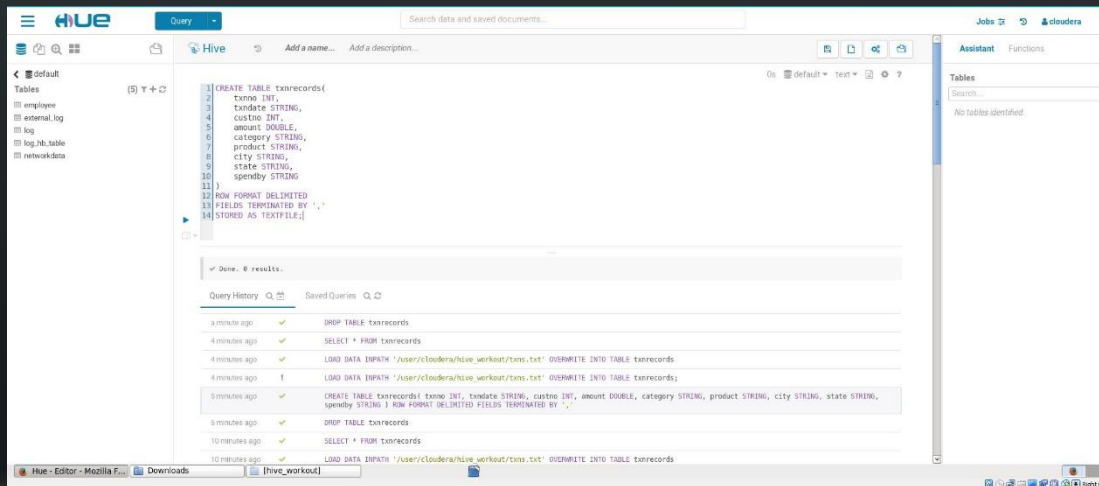
Hive Workout

- Create a internal/managed table structure named "customer" with below column/datatype for the data cust.txt :
custno string, firstname string, lastname string, age int,profession string
- Load the table with data file placed in HDFS
- Create a internal/managed table "out1" with below structure :
custno int,firstname string,age int,profession string,amount double,product string
- Insert the result of join between tables "txnrecords" & "customer" into table "out1"
- Create a internal/managed table "out2" with below structure :
custno int,firstname string,age int,profession string,amount double,product string, level string
- Populate first 6 columns with similar join results achieved in table "out1"
level column has to be populated based on below logic:
 - If age less than 30, level value = low
 - If age is between 30 & 50, level value = middle
 - If age is greater than 50, level value = old

Task #7

Hive workout

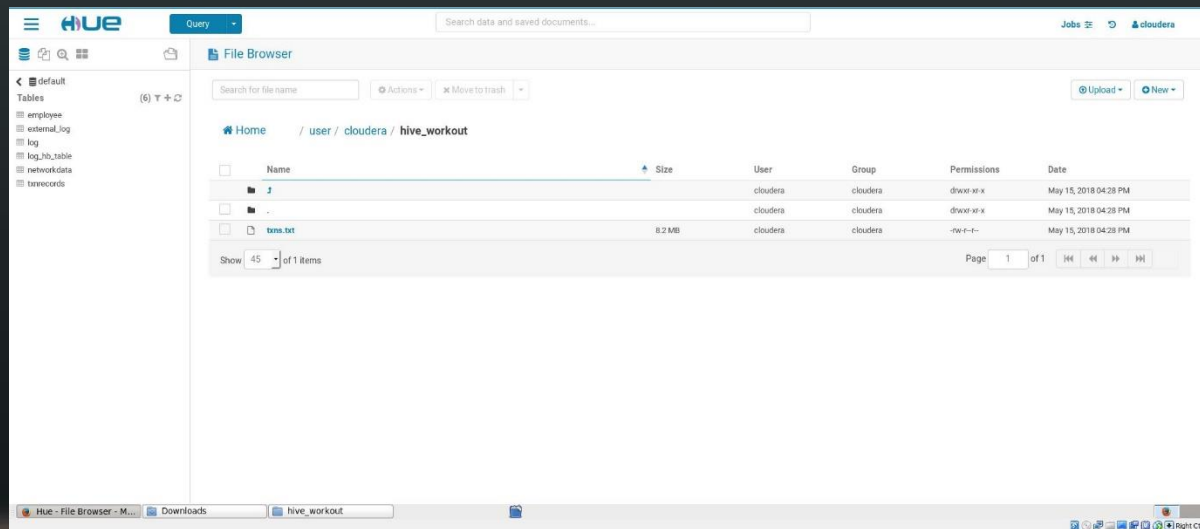
- Create a internal/managed table structure named "txnrecords" with below column/datatype for the data file txns.txt:
txno INT, txndate STRING, custno INT, amount DOUBLE, category STRING, product STRING, city STRING, state STRING, spendby STRING



Task #7

Hive workout

Loading our text file to hdfs



Task #7

Hive workout

Dump data from csv file to internal hive table.

The screenshot displays the Hue web interface for managing Hive queries. The main editor area contains a single query: `LOAD DATA INPATH '/user/cloudera/hive_workout/txns.txt' OVERWRITE INTO TABLE txnrecords;`. Below the editor, a success message indicates the query completed. The Query History panel shows a list of recent queries, including the current one, with their execution times and statuses. The left sidebar lists available tables, and the right sidebar shows the current table selected.

Hue Interface Details:

- Top Bar:** Hue logo, Query dropdown, Search data and saved documents...
- Left Sidebar:** default (6) T + ↻, Tables (employee, external_log, log, log_hb_table, networkdata, txnrecords)
- Editor:** Hive icon, Add a name..., Add a description..., Query text area with the Hive query.
- Right Sidebar:** Assistant, Functions, Tables (Search..., default.txnrecords)
- Bottom Panel:** Query History, Saved Queries, Query execution log showing success.

Query History Table:

Time Ago	Status	Query Text
a few seconds ago	Success	LOAD DATA INPATH '/user/cloudera/hive_workout/txns.txt' OVERWRITE INTO TABLE txnrecords
5 minutes ago	Success	select * from txnrecords
7 minutes ago	Success	CREATE TABLE txnrecords(txnno INT, txndate STRING, custno INT, amount DOUBLE, category STRING, product STRING, city STRING, state STRING, spendby STRING)
17 hours ago	Success	SELECT * FROM log_hb_table where log_id=1000
17 hours ago	Success	SELECT * FROM log_hb_table
17 hours ago	Success	INSERT OVERWRITE TABLE log_hb_table SELECT * FROM external_log
17 hours ago	Success	CREATE TABLE log_hb_table(log_id INT,log_time BIGINT, log_user STRING, log_port INT, log_port_desc STRING) STORED BY 'org.apache.hadoop.hive.hbase.HBaseStorageHandler' WITH SERDEPROPERTIES ("hbase.columns.mapping" = ":key,data:log time,data:log user,data:log port,data:log port desc")

Task #7

Hive workout

Finally query data from internal table.

The screenshot shows the Hue web interface for querying data. The top navigation bar includes the Hue logo, a 'Query' dropdown, and a search bar. The left sidebar shows a 'Tables' list with 'txnrecords' selected. The main query editor contains the SQL statement: `SELECT * FROM txnrecords;`. The results pane displays 13 rows of data from the 'txnrecords' table. The right sidebar shows the 'Assistant' and 'Functions' panels, with 'default.txnrecords' listed under 'Tables'.

	txnrecords.txnno	txnrecords.txndate	txnrecords.custno	txnrecords.amount	txnrecords.category	txnrecords.product	txnrecords.city	txnrecords.state
1	0	06-26-2011	4007024	40.329999999999998	Exercise & Fitness	Cardio Machine Accessories	Clarksville	Tennessee
2	1	05-26-2011	4006742	198.44	Exercise & Fitness	Weightlifting Gloves	Long Beach	California
3	2	06-01-2011	4009775	5.5800000000000001	Exercise & Fitness	Weightlifting Machine Accessories	Anaheim	California
4	3	06-05-2011	4002199	198.19	Gymnastics	Gymnastics Rings	Milwaukee	Wisconsin
5	4	12-17-2011	4002613	98.810000000000002	Team Sports	Field Hockey	Nashville	Tennessee
6	5	02-14-2011	4007991	193.63	Outdoor Recreation	Camping & Backpacking & Hiking	Chicago	Illinois
7	6	10-28-2011	4002190	27.890000000000001	Puzzles	Jigsaw Puzzles	Charleston	South Carolina
8	7	07-14-2011	4002964	96.010000000000005	Outdoor Play Equipment	Sandboxes	Columbus	Ohio
9	8	01-17-2011	4007361	10.44	Winter Sports	Snowmobiling	Des Moines	Iowa
10	9	05-17-2011	4004798	152.46000000000001	Jumping	Bungee Jumping	St. Petersburg	Florida
11	10	05-29-2011	4004646	180.28	Outdoor Recreation	Archery	Reno	Nevada
12	11	06-18-2011	4008071	121.39	Outdoor Play Equipment	Swing Sets	Columbus	Ohio
13	12	02-08-2011	4002473	41.520000000000003	Indoor Games	Bowling	San Francisco	California

Task #7

Hive workout

Load the table with data file placed in HDFS.

Note : Below 2 tables are to be created as external tables with some user defined location

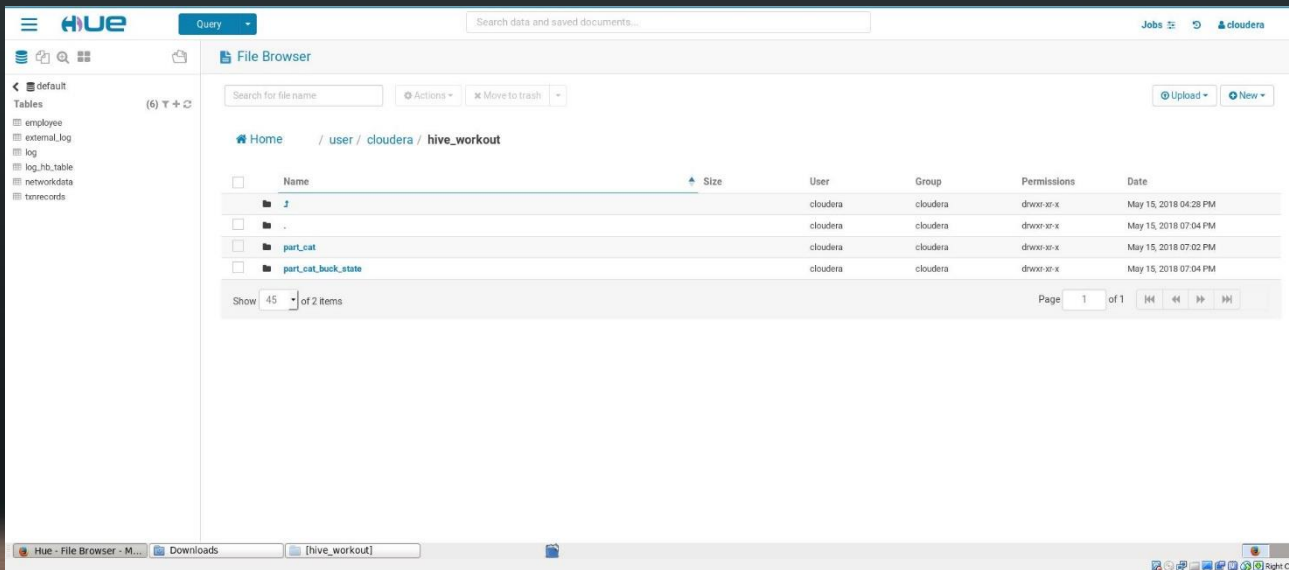
create another table structure with similar columns like table "txnrecords" - but create a partitioned table named "txnrecsByCat" (partition table by category)

create another table structure with similar columns like table 'txnrecords' - but create a table with partition and 10 buckets named "buckettxnrecsByCat"(partition by category & buckets by state)

Task #7

Hive workout

This is the directory in which we're going to store both of our table partitions.



Task #7

Hive workout

Let's create our "txnrecsByCat" table (partition table by category).

The screenshot displays the Hue web interface for managing Hadoop clusters. The central pane shows a Hive query that has been executed successfully. The query creates an external table named 'txnrecsByCat' with columns: txnno (INT), txndate (STRING), custno (INT), amount (DOUBLE), product (STRING), city (STRING), state (STRING), and spendby (STRING). The table is partitioned by the 'category' column and is located at '/user/cloudera/hive_workout/part_cat/'.

Below the query editor, a 'Success' message is displayed. A 'Query History' table lists the executed queries:

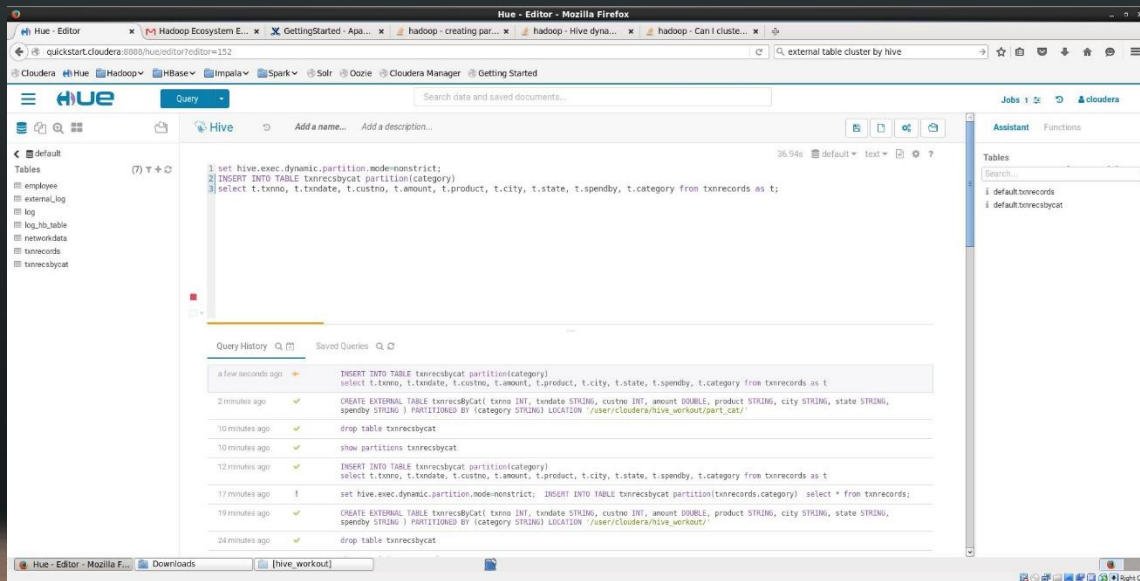
Time	Status	Query
a few seconds ago	✓	CREATE EXTERNAL TABLE txnrecsByCat(txnno INT, txndate STRING, custno INT, amount DOUBLE, product STRING, city STRING, state STRING, spendby STRING) PARTITIONED BY (category STRING) LOCATION '/user/cloudera/hive_workout/part_cat/';
8 minutes ago	✓	drop table txnrecsbycat
8 minutes ago	✓	show partitions txnrecsbycat
10 minutes ago	✓	INSERT INTO TABLE txnrecsbycat partition(category) select t.txnno, t.txndate, t.custno, t.amount, t.product, t.city, t.state, t.spendby, t.category from txnrecords as t
15 minutes ago	!	set hive.exec.dynamic.partition.mode=nonstrict; INSERT INTO TABLE txnrecsbycat partition(txnrecords.category) select * from txnrecords;
17 minutes ago	✓	CREATE EXTERNAL TABLE txnrecsByCat(txnno INT, txndate STRING, custno INT, amount DOUBLE, product STRING, city STRING, state STRING, spendby STRING) PARTITIONED BY (category STRING) LOCATION '/user/cloudera/hive_workout/';
22 minutes ago	✓	drop table txnrecsbycat

The right sidebar shows the 'Assistant' and 'Functions' tabs, with a 'Tables' section indicating 'No tables identified.'

Task #7

Hive workout

Let's insert our data from our source table (txnrecords) to our external partition table (txnrecsByCat).



Task #7

Hive workout

Finally let's query the different partitions that our table contains.

The screenshot shows the Hue web interface. At the top, there's a search bar and a 'Query' dropdown. The left sidebar shows a file tree with 'default' selected, containing tables like 'employee', 'external_log', 'log', 'log_hb_table', 'networkdata', 'txnrecords', and 'txnrecsbycat'. The main area displays the query `show partitions txnrecsbycat;` and the results of the query. The results are listed in a table with a 'partition' column and 15 rows of category names. The right sidebar shows the 'Assistant' and 'Functions' tabs, with 'Tables' selected, showing a search bar and the table 'default.txnrecsbycat'.

partition
1 category=Air Sports
2 category=Combat Sports
3 category=Dancing
4 category=Exercise & Fitness
5 category=Games
6 category=Gymnastics
7 category=Indoor Games
8 category=Jumping
9 category=Outdoor Play Equipment
10 category=Outdoor Recreation
11 category=Puzzles
12 category=Racquet Sports
13 category=Team Sports
14 category=Water Sports
15 category=Winter Sports

Task #7

Hive workout

Let's create our "buckettxnrecsByCat" table (partition by category & buckets by state)

The screenshot shows the Hue web interface with the Hive console. The left sidebar lists tables, including 'buckettxnrecsByCat'. The main console area displays the following SQL query:

```
1 CREATE EXTERNAL TABLE buckettxnrecsByCat(  
2   txnno INT,  
3   txndate STRING,  
4   custno INT,  
5   amount DOUBLE,  
6   product STRING,  
7   city STRING,  
8   state STRING,  
9   spendby STRING  
10 )  
11 PARTITIONED BY (category STRING)  
12 CLUSTERED BY (state) INTO 10 BUCKETS  
13 LOCATION '/user/cloudera/hive_workout/part_cat_buck_state/';
```

Below the query, a success message is displayed: "Success." The Query History section shows a list of recent queries, including the one that was just executed:

Time	Status	Query
a few seconds ago	✓	show partitions txnrecsbycat
a minute ago	✓	INSERT INTO TABLE txnrecsbycat partition(category) select t.txnno, t.txndate, t.custno, t.amount, t.product, t.city, t.state, t.spendby, t.category from txnrecords as t
2 minutes ago	✓	show partitions txnrecsbycat
3 minutes ago	✓	set hive.exec.dynamic.partition.mode=nonstrict
3 minutes ago	✓	CREATE EXTERNAL TABLE txnrecsByCat(txnno INT, txndate STRING, custno INT, amount DOUBLE, product STRING, city STRING, state STRING, spendby STRING) PARTITIONED BY (category STRING) LOCATION '/user/cloudera/hive_workout/part_cat/'
5 minutes ago	✓	DROP TABLE txnrecsbycat
5 minutes ago	✓	DROP TABLE buckettxnrecsbycat

The right sidebar shows the Assistant and Functions panels, with the Assistant panel displaying "No tables identified."

Task #7

Hive workout

Now let's insert data from our source table (txnrecords) to our external table (buckettxnrecsByCat)

The screenshot shows the Hue web interface for a Hive environment. The main query editor contains the following SQL code:

```
1 set hive.exec.dynamic.partition.mode=nonstrict;
2 set hive.enforce.bucketing = true;
3 INSERT INTO TABLE buckettxnrecsByCat partition(category)
4 select t.txno, t.txndate, t.custno, t.amount, t.product, t.city, t.state, t.spendby, t.category from txnrecords as t;
```

The left sidebar shows a list of tables, including 'buckettxnrecsByCat', 'employee', 'external_log', 'log', 'log_ib_table', 'networkdata', 'txnrecords', and 'txnrecsbycat'. The right sidebar shows a 'Query History' table with the following entries:

Time	Status	Query
a minute ago	✓	INSERT INTO TABLE buckettxnrecsByCat partition(category) select t.txno, t.txndate, t.custno, t.amount, t.product, t.city, t.state, t.spendby, t.category from txnrecords as t
2 minutes ago	✓	DESCRIBE FORMATTED buckettxnrecsByCat
7 minutes ago	✓	CREATE EXTERNAL TABLE buckettxnrecsByCat (txno INT, txndate STRING, custno INT, amount DOUBLE, product STRING, city STRING, state STRING, spendby STRING) PARTITIONED BY (category STRING) CLUSTERED BY (state) INTO 10 BUCKETS LOCATION '/user/cloudera/hive_workout/part_cat/';
an hour ago	✓	SELECT * FROM txnrecsbycat WHERE category='Air Sports' LIMIT 10;
an hour ago	✓	show partitions txnrecsbycat
an hour ago	✓	DESCRIBE FORMATTED txnrecsbycat
an hour ago	✓	show partitions txnrecsbycat
an hour ago	✓	INSERT INTO TABLE txnrecsbycat partition(category) select t.txno, t.txndate, t.custno, t.amount, t.product, t.city, t.state, t.spendby, t.category from txnrecords as t
an hour ago	✓	CREATE EXTERNAL TABLE txnrecsbycat (txno INT, txndate STRING, custno INT, amount DOUBLE, product STRING, city STRING, state STRING, spendby STRING) PARTITIONED BY (category STRING) LOCATION '/user/cloudera/hive_workout/part_cat/';
an hour ago	✓	drop table txnrecsbycat
an hour ago	✓	show partitions txnrecsbycat

Task #7

Hive workout

Now let's see the number of buckets our table has.

The screenshot shows the Hue web interface for a Hive table named 'buckettenrecsbycat'. The table's metadata is displayed in a table format with columns 'col_name', 'data_type', and 'comment'. The 'Num Buckets' is highlighted in yellow, showing a value of 16. The interface also includes a sidebar with a file tree, a top navigation bar, and a right-hand panel with 'Assistant' and 'Functions' tabs.

col_name	data_type	comment
20 CreateTime:	Tue May 15 20:33:10 PDT 2018	NULL
21 LastAccessTime:	UNKNOWN	NULL
22 Protect Mode:	None	NULL
23 Retention:	0	NULL
24 Location:	hdfs://quickstart.cloudera:8020/user/cloudera/hive_workout/part_cat_bucket_state	NULL
25 Table Type:	EXTERNAL_TABLE	NULL
26 Table Parameters:	NULL	NULL
27	EXTERNAL	TRUE
28 numPartitions	15	NULL
29 transient_lastDdlTime	1526441500	NULL
30	NULL	NULL
31 # Storage Information	NULL	NULL
32 SerDe Library:	org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe	NULL
33 InputFormat:	org.apache.hadoop.mapred.TextInputFormat	NULL
34 OutputFormat:	org.apache.hadoop.hive ql.io.HiveIgnoreKeyTextOutputFormat	NULL
35 Compressed:	No	NULL
36 Num Buckets:	16	NULL
37 Bucket Columns:	[state]	NULL
38 Sort Columns:	[]	NULL
39 Storage Desc Params:	NULL	NULL
40 serialization.format	1	NULL

Task #7

Hive workout

Finally let's see our generated partitions.

The screenshot shows the Hue web interface with a Hive query executed. The query is `show partitions bucketxnrecsbycat;`. The results are displayed in a table with 13 rows, each representing a partition. The table has a header row with the column name 'partition' and 13 data rows, each with a row number and a category name.

partition
1 category=Air Sports
2 category=Combat Sports
3 category=Dancing
4 category=Exercise & Fitness
5 category=Games
6 category=Gymnastics
7 category=Indoor Games
8 category=Jumping
9 category=Outdoor Play Equipment
10 category=Outdoor Recreation
11 category=Puzzles
12 category=Racquet Sports
13 category=Team Sports

Task #7

Hive workout

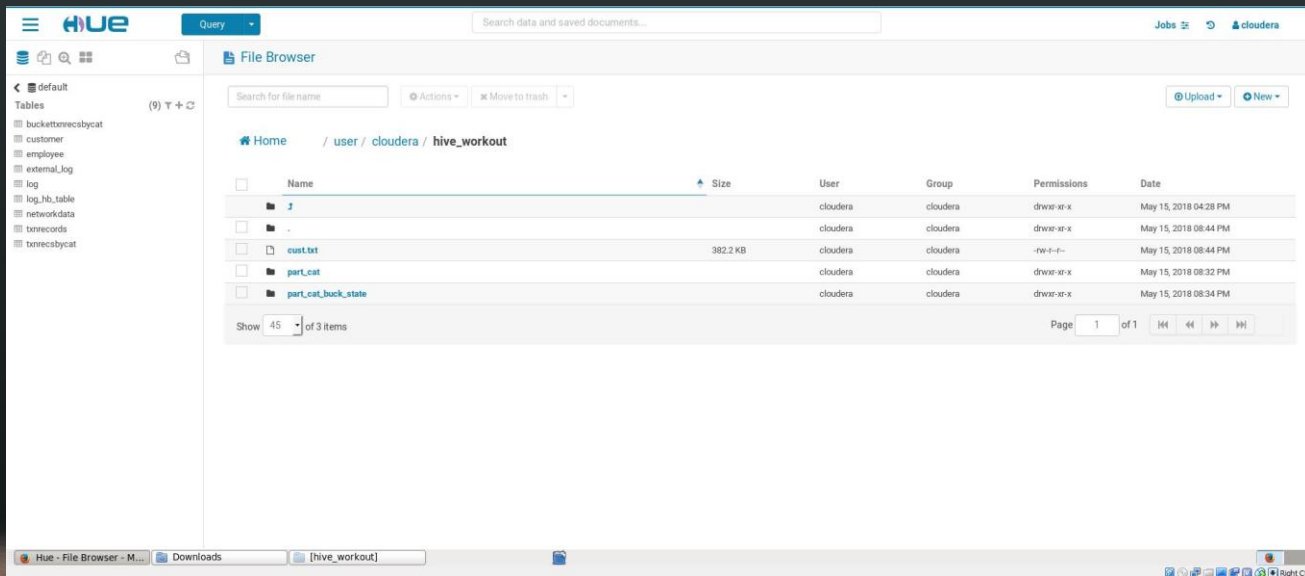
Create a internal/managed table structure named "customer" with below column/datatype for the data cust.txt :

custno string, firstname string, lastname string, age int,profession string

Task #7

Hive workout

Uploading our customer text file to HDFS.



Task #7

Hive workout

Creating our internal customer table.

The screenshot shows the Apache Hue web interface. On the left, a sidebar lists tables under the 'default' database, including 'buckettxnrscbycat', 'customer', 'employee', 'external_log', 'log', 'log_hb_table', 'networkdata', 'txnrrecords', and 'txnrscbycat'. The 'customer' table is highlighted. The main area displays a Hive query in a text editor:

```
1 CREATE TABLE customer(  
2   custno STRING,  
3   firstname STRING,  
4   lastname STRING,  
5   age INT,  
6   profession STRING  
7 )  
8 ROW FORMAT DELIMITED  
9 FIELDS TERMINATED BY ','  
10 STORED AS TEXTFILE;
```

Below the query editor, a success message is shown: "Success." The Query History panel lists several queries, with the most recent one being the 'CREATE TABLE customer' query, which was executed 'a few seconds ago'. The query history also shows subsequent queries like 'select * from txnrrecords', 'show partitions buckettxnrscbycat', 'DESCRIBE FORMATTED buckettxnrscbycat', and 'INSERT INTO TABLE buckettxnrscbycat partition(category) select t.txmno, t.txndate, t.custno, t.amount, t.product, t.city, t.state, t.spendby, t.category from txnrrecords as t'.

Task #7

Hive workout

Loading data to our customer table.

The screenshot displays the Hue web interface for managing Hive data. The top navigation bar includes the Hue logo, a 'Query' dropdown, and a search bar. The left sidebar shows a 'Tables' list with various database tables like 'buckettxnrecsbycat', 'customer', 'employee', etc. The main workspace is divided into three sections: a query editor at the top, a status bar in the middle, and a query history at the bottom.

The query editor contains the following Hive query:

```
1) LOAD DATA INPATH '/user/cloudera/hive_workout/cust.txt' OVERWRITE INTO TABLE customer;
```

The status bar shows a 'Success' message, indicating the query was executed successfully.

The query history section lists several previous queries, including:

- 3 minutes ago: CREATE TABLE customer(custno STRING, firstname STRING, lastname STRING, age INT, profession STRING) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' STORED AS TEXTFILE
- 5 minutes ago: select * from txnrecords
- 8 minutes ago: show partitions buckettxnrecsbycat
- 9 minutes ago: DESCRIBE FORMATTED buckettxnrecsbycat
- 11 minutes ago: INSERT INTO TABLE buckettxnrecsbycat partition(category) select t.txnno, t.txndate, t.custno, t.amount, t.product, t.city, t.state, t.spendby, t.category from txnrecords as t
- 12 minutes ago: CREATE EXTERNAL TABLE buckettxnrecsbycat(txnno INT, txndate STRING, custno INT, amount DOUBLE, product STRING, city STRING, state STRING, spendby STRING) PARTITIONED BY (category STRING) CLUSTERED BY (state) INTO 10 BUCKETS LOCATION '/user/cloudera/hive_workout/part_cat_buck_state/'
- 13 minutes ago: show partitions txnrecsbycat
- 14 minutes ago: INSERT INTO TABLE txnrecsbycat partition(category) select t.txnno, t.txndate, t.custno, t.amount, t.product, t.city, t.state, t.spendby, t.category from txnrecords as t

The bottom of the interface shows the operating system taskbar with the Hue editor, Mozilla Firefox, and a file explorer window named 'hive_workout'.

Task #7

Hive workout

Querying data from our customer table.

The screenshot shows the Hue web interface with a Hive query executed. The query is `select * from customer;`. The results are displayed in a table with 11 rows and 5 columns: `customer.custno`, `customer.firstname`, `customer.lastname`, `customer.age`, and `customer.profession`.

	customer.custno	customer.firstname	customer.lastname	customer.age	customer.profession
1	4000001	Kristina	Chung	55	Pilot
2	4000002	Paige	Chen	74	Teacher
3	4000003	Sherri	Melton	34	Firefighter
4	4000004	Gretchen	Hill	66	Computer hardware engineer
5	4000005	Karen	Puckett	74	Lawyer
6	4000006	Patrick	Song	42	Veterinarian
7	4000007	Elsie	Hamilton	43	Pilot
8	4000008	Hazel	Bender	63	Carpenter
9	4000009	Malcolm	Wagner	39	Artist
10	4000010	Dolores	McLaughlin	60	Writer
11	4000011	Francie	McNamara	47	Therapist

Task #7

Hive workout

Create a internal/managed table "out1" with below structure :
custno int,firstname string,age int,profession string,amount double,product string

The screenshot shows the Hue web interface for Hive. On the left, a sidebar lists various tables including 'out1', which is currently selected. The main area displays a Hive SQL query to create a table named 'out1' with the following schema: custno int, firstname string, age int, profession string, amount double, product string. The query is executed successfully, as indicated by a 'Success' message. Below the message, a 'Query History' table lists the executed queries, including the 'CREATE TABLE' statement and subsequent 'SELECT' and 'DESCRIBE' queries. The bottom of the interface shows the operating system taskbar with the Hue application running in a Mozilla Firefox browser window.

```
1 CREATE TABLE out1(
2   custno int,
3   firstname string,
4   age int,
5   profession string,
6   amount double,
7   product string
8 )
9 ROW FORMAT DELIMITED
10 FIELDS TERMINATED BY ','
11 STORED AS TEXTFILE
```

Success.

Query History	Saved Queries
a few seconds ago	CREATE TABLE out1(custno int, firstname string, age int, profession string, amount double, product string) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' STORED AS TEXTFILE
8 minutes ago	select * from customer
13 minutes ago	CREATE TABLE customer(custno STRING, firstname STRING, lastname STRING, age INT, profession STRING) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' STORED AS TEXTFILE
15 minutes ago	select * from txrecords
18 minutes ago	show partitions buckettxrecordsbycat
19 minutes ago	DESCRIBE FORMATTED buckettxrecordsbycat
20 minutes ago	INSERT INTO TABLE buckettxrecordsbycat partition(category) select t.txno, t.txdate, t.custno, t.amount, t.product, t.city, t.state, t.spendby, t.category from txrecords as t
21 minutes ago	CREATE EXTERNAL TABLE buckettxrecordsbyCat(txno INT, txdate STRING, custno INT, amount DOUBLE, product STRING, city STRING, state STRING, spendby STRING) PARTITIONED BY (category STRING) CLUSTERED BY (state) INTO 10 BUCKETS LOCATION '/user/cloudera/hive_workout/part_cat_buck_state/'

Task #7

Hive workout

Insert the result of join between tables "txnrecords" & "customer" into table "out1".

The screenshot shows the Hue web interface for Hive. The main query editor contains the following SQL:

```
1 INSERT INTO TABLE out1
2 SELECT c.custno, c.firstname, c.age, c.profession, txn.amount, txn.product
3 FROM txnrecords txn JOIN customer c
4 ON (txn.custno = c.custno);
```

The query has been executed successfully, as indicated by the "Success" message and the "Query History" table below it.

Time	Status	Query
1 minute ago	✓	INSERT INTO TABLE out1 SELECT c.custno, c.firstname, c.age, c.profession, txn.amount, txn.product FROM txnrecords txn JOIN customer c ON (txn.custno = c.custno)
2 minutes ago	!	INSERT INTO TABLE out1 SELECT c.custno, c.firstname, c.profession, txn.amount, txn.product FROM txnrecords txn JOIN customer c ON (txn.custno = c.custno)
9 minutes ago	✓	CREATE TABLE out1(custno int, firstname string, age int, profession string, amount double, product string) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' STORED AS TEXTFILE
16 minutes ago	✓	select * from customer
21 minutes ago	✓	CREATE TABLE customer(custno STRING, firstname STRING, lastname STRING, age INT, profession STRING) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' STORED AS TEXTFILE
23 minutes ago	✓	select * from txnrecords
26 minutes ago	✓	show partitions buckettxnrecsbycat
27 minutes ago	✓	DESCRIBE FORMATTED buckettxnrecsbycat

The left sidebar shows the database structure with tables like 'customer', 'txnrecords', and 'out1'. The right sidebar shows the 'Assistant' and 'Functions' panels.

Task #7

Hive workout

Querying data from out1 table.

The screenshot shows the Hue web interface for querying data. The query editor at the top contains the SQL statement: `select * from out1;`. Below the editor, the results are displayed in a table with 6 columns: `out1.custno`, `out1.firstname`, `out1.age`, `out1.profession`, `out1.amount`, and `out1.product`. The table contains 11 rows of data. On the left, a sidebar lists various tables and their schemas. On the right, a 'Tables' panel shows the current table being viewed.

	out1.custno	out1.firstname	out1.age	out1.profession	out1.amount	out1.product
1	4007024	Cameron	59	Actor	40.329999999999998	Cardio Machine Accessories
2	4006742	Gregory	36	Accountant	198.44	Weightlifting Gloves
3	4009775	Ruby	44	Designer	5.5800000000000001	Weightlifting Machine Accessories
4	4002199	Keith	44	Police officer	198.19	Gymnastics Rings
5	4002613	Hugh	43	Engineering technician	98.810000000000002	Field Hockey
6	4007591	Jennifer	54	Electrician	193.63	Camping & Backpacking & Hiking
7	4002190	Sheryl	62	Designer	27.890000000000001	Jigsaw Puzzles
8	4002964	Ken	67	Recreation and fitness worker	96.010000000000005	Sandboxes
9	4007361	Trent	52	Loan officer	10.44	Snowmobiling
10	4004798	Geoffrey	65	Chemist	152.46000000000001	Bungee Jumping
11	4007646	Ernie	64	Computer software engineer	180.78	Amateur

Task #7

Hive workout

Create a internal/managed table "out2" with below structure :
custno int,firstname string,age int,profession string,amount double,product string, level string

The screenshot shows the Hue web interface with a Hive query executed successfully. The query is as follows:

```
1 CREATE TABLE out2(  
2   custno int,  
3   firstname string,  
4   age int,  
5   profession string,  
6   amount double,  
7   product string,  
8   level string  
9 )  
10 ROW FORMAT DELIMITED  
11 FIELDS TERMINATED BY ','  
12 STORED AS TEXTFILE
```

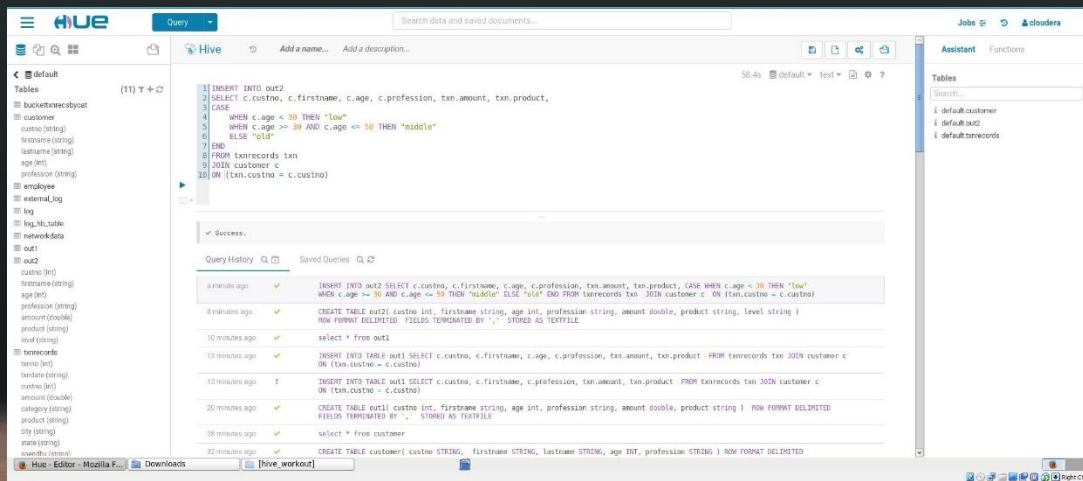
The Query History shows the following queries:

- a few seconds ago: CREATE TABLE out2(custno int, firstname string, age int, profession string, amount double, product string, level string) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' STORED AS TEXTFILE
- 2 minutes ago: select * from out1
- 5 minutes ago: INSERT INTO TABLE out1 SELECT c.custno, c.firstname, c.age, c.profession, tm.amount, tm.product FROM txrecords tm JOIN customer c ON (tm.custno = c.custno)
- 6 minutes ago: INSERT INTO TABLE out1 SELECT c.custno, c.firstname, c.profession, tm.amount, tm.product FROM txrecords tm JOIN customer c ON (tm.custno = c.custno)
- 12 minutes ago: CREATE TABLE out1(custno int, firstname string, age int, profession string, amount double, product string) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' STORED AS TEXTFILE
- 20 minutes ago: select * from customer
- 24 minutes ago: CREATE TABLE customer(custno STRING, firstname STRING, lastname STRING, age INT, profession STRING) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' STORED AS TEXTFILE
- 26 minutes ago: select * from txrecords

Task #7

Hive workout

- Populate first 6 columns with similar join results achieved in table "out1"
level column has to be populated based on below logic:
 - If age less than 30, level value = low
 - If age is between 30 & 50, level value = middle
 - If age is greater than 50, level value = old



Task #7

Hive workout

Querying data from out2 table.

The screenshot shows the Hue web interface for querying data in a Hive database. The query editor on the left contains the SQL query: `select * from out2;`. The results pane on the right displays a table with 11 rows and 7 columns. The columns are: `out2.custno`, `out2.firstname`, `out2.age`, `out2.profession`, `out2.amount`, `out2.product`, and `out2.level`. The data rows show various customer records, including Cameron, Gregory, Ruby, Keith, Hugh, Jennifer, Sheryl, Ken, Teet, Geoffrey, and Frank.

	out2.custno	out2.firstname	out2.age	out2.profession	out2.amount	out2.product	out2.level
1	4007024	Cameron	59	Actor	40.329999999999998	Cardio Machine Accessories	old
2	4006742	Gregory	36	Accountant	198.44	Weightlifting Gloves	middle
3	4009775	Ruby	44	Designer	5.5800000000000001	Weightlifting Machine Accessories	middle
4	4002199	Keith	44	Police officer	198.19	Gymnastics Rings	middle
5	4002613	Hugh	43	Engineering technician	98.8100000000000002	Field Hockey	middle
6	4007591	Jennifer	54	Electrician	193.63	Camping & Backpacking & Hiking	old
7	4002190	Sheryl	62	Designer	27.8900000000000001	Jigsaw Puzzles	old
8	4002964	Ken	67	Recreation and fitness worker	96.0100000000000005	Sandboxes	old
9	4007361	Teet	52	Loan officer	10.44	Snowmobiling	old
10	4004798	Geoffrey	65	Chemist	152.460000000000001	Bungee Jumping	old
11	4004646	Frank	64	Computer software engineer	180.28	Amateur	old

Task #8

Spark RDD Exercises

exercise 1

```
1 // using rdd_7 write the code to create an RDD that return an integer his square and cube.  
2 val rdd_7 = sc.parallelize( List(1,2,3,4,5,6, 7) )  
3 // your code here  
4 rdd_7.map(n => (n, n*n, n*n*n)).collect()
```

▶ (1) Spark Jobs

```
rdd_7: org.apache.spark.rdd.RDD[Int] = ParallelCollectionRDD[13] at parallelize at command-246765983207984:2  
res7: Array[(Int, Int, Int)] = Array((1,1,1), (2,4,8), (3,9,27), (4,16,64), (5,25,125), (6,36,216), (7,49,343))
```

Command took 0.35 seconds -- by art9518@gmail.com at 5/8/2018, 6:41:48 PM on My Cluster

Task #8

Spark RDD Exercises

Cmd 10

exercise 2

```
1 // using word count example as reference.
2 // write code that counts the words in a file and print an RDD, where each row in RDD has number of words and word.
3 val file_path = "dbfs:/databricks-datasets/README.md"
4
5 // load a text file into an RDD --> generating a bag of word with their respective word count
6 val textFile = sc.textFile( file_path )
7
8 val counts = textFile
9     .flatMap( line => line.split(" ") )
10    .map( word => (word, 1) )
11    .reduceByKey( _ + _ )
12    .map( _.swap )
13 counts.take( 10 )
```

► (1) Spark Jobs

```
file_path: String = dbfs:/databricks-datasets/README.md
textFile: org.apache.spark.rdd.RDD[String] = dbfs:/databricks-datasets/README.md MapPartitionsRDD[49] at textFile at command-246765983207985:6
counts: org.apache.spark.rdd.RDD[(Int, String)] = MapPartitionsRDD[53] at map at command-246765983207985:12
res21: Array[(Int, String)] = Array((1,Unless), (3,this), (1,under), (1,Please), (2,is), (1,Commons), (1,include), (1,-----), (1,CC), (5,data))

Command took 0.52 seconds -- by art9518@gmail.com at 5/8/2018, 7:03:42 PM on My Cluster
```

Task #8

Spark RDD Exercises

Cmd 11

exercise 3

```
1 // create an rdd that keep all the words that appear two times.
2 // tip: read filter transformation
3 // http://spark.apache.org/docs/2.1.1/programming-guide.html#transformations
4 val file_path = "dbfs:/databricks-datasets/README.md"
5
6 // load a text file into an RDD --> generating a bag of word with their respective word count
7 val textFile = sc.textFile( file_path )
8
9 val counts = textFile
10   .flatMap( line => line.split(" ") )
11   .map( word => (word, 1) )
12   .reduceByKey(_ + _)
13   .filter((item) => item._2 > 2)
14
15 counts.take( 10 )
```

► (2) Spark Jobs

file_path: String = dbfs:/databricks-datasets/README.md

textFile: org.apache.spark.rdd.RDD[String] = dbfs:/databricks-datasets/README.md MapPartitionsRDD[21] at textFile at command-246765983207986:7

counts: org.apache.spark.rdd.RDD[(String, Int)] = ShuffledRDD[24] at reduceByKey at command-246765983207986:12

filterCounts: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[25] at filter at command-246765983207986:14

res11: Array[(String, Int)] = Array((this,3), (data,5), (",",5), (for,3), (the,11), (to,5), (and,4))

Command took 0.76 seconds -- by art9518@gmail.com at 5/8/2018, 6:50:32 PM on My Cluster.

Task #8

Spark RDD Exercises

Cod 12

exercise 4

```
1 // create an RDD that groups the words by number of times they occur.
2 // each row must has number_of_times and a collection of words.
3 // tip read groupByKey
4 // http://spark.apache.org/docs/2.1.1/programming-guide.html#transformations
5 val file_path = "dbfs:/databricks-datasets/README.md"
6
7 // load a text file into an RDD --> generating a bag of word with their respective word count
8 val textFile = sc.textFile( file_path )
9
10 val counts = textFile
11   .flatMap( line => line.split(" ") )
12   .map( word => (word, 1) )
13   .reduceByKey(_ + _)
14   .map(_._swap)
15   .groupByKey()
16   .map(x => (x._1, x._2.toArray))
17
18 counts.take( 10 )
```

► (2) Spark Jobs

file_path: String = dbfs:/databricks-datasets/README.md

textFile: org.apache.spark.rdd.RDD[String] = dbfs:/databricks-datasets/README.md MapPartitionsRDD[1] at textFile at command-246765983207987:8

counts: org.apache.spark.rdd.RDD[(Int, Array[String])] = MapPartitionsRDD[7] at map at command-246765983207987:16

res0: Array[(Int, Array[String])] = Array((4,Array(and)), (2,Array(is, file, Databricks, within, be, by, hosted)), (11,Array(the)), (1,Array(Unless, under, Please, Commons, include, -----, (CC, otherwise, hos ted-datasets@databricks.com., using, send, Make, viewed, email, new, README, International, Databricks., allows, consumed, (e.g., can, -----, build, data,, Apache, how, 4.0, When, get, information, please, en sure, information., noted, users, Requests, want, repository,, Spark, The, it, url:, about, public., license,, -----, set)), at, following, pipelines, sure, which, Creative, README.md, To, Attribution, to:, includes, directory, 4.0)), you, BY, request,, a, given, contribute, source, datasets, Datasets, License, Contributions, contained, Hosted, or, license, of, request, an, licensed, making, additional, publish., [http://creativecommons.org/licenses/by/ 4.0/legalcode](http://creativecommons.org/licenses/by/4.0/legalcode))), (3,Array(this, for)), (5,Array(data, "", to)))

Command took 9.99 seconds -- by art9518@gmail.com at 5/15/2018, 11:32:55 PM on My Cluster

Resources

Useful Links

- <https://github.com/bobfreitas/flume-logs>
- https://stackoverflow.com/questions/30908641/save-flume-output-to-hive-table-with-hive-sink?utm_medium=organic&utm_source=google_rich_qa&utm_campaign=google_rich_qa
- https://stackoverflow.com/questions/18501551/how-to-use-flume-creating-a-task-to-load-data-from-hdfs-to-hive-automatic-by-tim?utm_medium=organic&utm_source=google_rich_qa&utm_campaign=google_rich_qa
- <https://www.quora.com/Why-do-we-use-row-format-delimited-and-fields-terminated-by-in-table-creation-in-Hive-and-in-Hadoop>



Resources

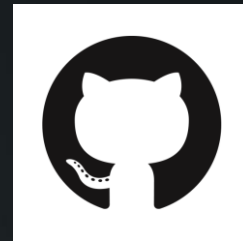
Useful Links

- <http://www.lopakalogic.com/articles/hadoop-articles/log-files-flume-hive/>
- <https://blogs.oracle.com/datawarehousing/flume-and-hive-for-log-analytics>
- <https://mevivs.wordpress.com/2010/11/24/hivehbase-integration/>
- <https://www.quora.com/How-can-I-transfer-data-from-Hive-external-table-to-Hbase>
- https://stackoverflow.com/questions/3635166/how-to-import-csv-file-to-mysql-table?utm_medium=organic&utm_source=google_rich_qa&utm_campaign=google_rich_qa
- https://stackoverflow.com/questions/32580356/creating-partition-in-external-table-in-hive?utm_medium=organic&utm_source=google_rich_qa&utm_campaign=google_rich_qa



Github

Repository



You can find this slides and more in my github repo:

