Lab: Importing RDBMS Data into HDFS

About this Lab

Objective:	Import data from a database into HDFS.
File locations:	/root/devph/labs/Lab3.1/
Successful outcome:	You will have imported data from MySQL into folders in HDFS.
Before you begin:	Your HDP 2.3 cluster should be up and running within your VM.
Related lesson:	***Inputting Data into HDFS***

Lab Steps

1) Create a Table in MySQL

a. If not already done, open a Terminal in your VM and type "ssh sandbox".
b. From the command prompt, change directories to `/root/devph/labs/Lab3.1/`:

```
# cd ~/devph/labs/Lab3.1/
```

c. View the contents of salaries.txt:

```
# tail salaries.txt
```

The comma-separated fields represent a gender, age, salary, and zip code.

d. Notice that there is a `salaries.sql` script that defines a new table in MySQL named salaries. For this script to work, you need to copy salaries.txt to the `/tmp` directory:

```
# cp salaries.txt /tmp
```

e. Now run the `salaries.sql` script using the following command:

```
# mysql test < salaries.sql
```

2) View the Table

a. To verify that the table is populated in MySQL, open the mysql prompt:

```
# mysql
```

b. Switch to the test database, which is where the salaries table was created:

```
mysql> use test;
```

c. Run the show tables command and verify that salaries is defined:

```
mysql> show tables;
+----------------+
| Tables_in_test |
+----------------+
| salaries       |
+----------------+
1 row in set (0.00 sec)
```

d. Select 10 items from the table to verify that it is populated:

```
mysql> select * from salaries limit 10;
+--------+------+--------+---------+----+
| gender | age  | salary | zipcode | id |
+--------+------+--------+---------+----+
| F      |   66 |  41000 |   95103 |  1 |
| M      |   40 |  76000 |   95102 |  2 |
| F      |   58 |  95000 |   95103 |  3 |
| F      |   68 |  60000 |   95105 |  4 |
| M      |   85 |  14000 |   95102 |  5 |
| M      |   14 |      0 |   95105 |  6 |
| M      |   52 |   2000 |   94040 |  7 |
| M      |   67 |  99000 |   94040 |  8 |
| F      |   43 |  11000 |   94041 |  9 |
| F      |   37 |  65000 |   94040 | 10 |
+--------+------+--------+---------+----+
```

e. Exit the mysql prompt:

```
mysql> exit
```

3) Import the Table into HDFS

a. Enter the following Sqoop command (all on a single line), which imports the salaries table in the test database into HDFS:

```
# sqoop import --connect jdbc:mysql://sandbox/test?user=root --
table salaries
```

b. A MapReduce job should start executing, and it may take a couple of minutes for the job to complete.

4) Verify the Import

a. View the contents of your HDFS folder:

```
# hdfs dfs -l
```

b. You should see a new folder named salaries. View its contents:

```
# hdfs dfs -ls salaries
Found 4 items
-rw-r--r--   1 root hdfs   272  salaries/part-m-00000
-rw-r--r--   1 root hdfs   241  salaries/part-m-00001
-rw-r--r--   1 root hdfs   238  salaries/part-m-00002
-rw-r--r--   1 root hdfs   272  salaries/part-m-00003
```

c. Notice there are four new files in the salaries folder named part-m-0000x. Why are there four of these files?

Answer: The MapReduce job that executed the Sqoop command used four `mappers`, so there are four output files (one from each `mapper`).

d. Use the cat command to view the contents of the files. For example:

```
# hdfs dfs -cat salaries/part-m-00000
```

Notice the contents of these files are the rows from the salaries table in MySQL. You have now successfully imported data from a MySQL database into HDFS. Notice that you imported the entire table with all of its columns. Next, you will import only specific columns of a table.

5) Specify Columns to Import

a. Using the --columns argument, write a Sqoop command that imports the salary and age columns (in that order) of the salaries table into a directory in HDFS named salaries2. In addition, set the -m argument to 1 so that the result is a single file.

Solution: The command you enter in the command line will look like this in the terminal window:

```
# sqoop import --connect jdbc:mysql://sandbox/test?user=root --
table salaries
--columns salary,age -m 1 --target-dir salaries2
```

Important
To make it easier to read, following is the same command as above, however we have broken it down into smaller chunks separated by a "\" at the end of the break point in each line. When you see this formatting in the lab, you should type it out as it appears above, and do not enter the \ characters unless specifically instructed to do so.

```
# sqoop import --connect jdbc:mysql://sandbox/test?user=root \
--table salaries \
--columns salary,age \
-m 1 \
--target-dir salaries2
```

b. After the import, verify you only have one part-m file in salaries2:

```
# hdfs dfs -ls salaries2
Found 1 items
-rw-r--r--   1 root hdfs   482  salaries2/part-m-00000
```

c. Verify that the contents of part-m-00000 are only the two columns you specified:

```
# hdfs dfs -cat salaries2/part-m-00000
The last few lines should look like the following:
69000.0,97
91000.0,48
0.0,1
48000.0,45
3000.0,39
14000.0,84
```

6) Importing from a Query

Write a Sqoop import command that imports the rows from salaries in MySQL whose salary column is greater than 90,000.00.

a. Use gender as the `--split-by` value, specify only two mappers, and import the data into the salaries3 folder in HDFS.

Tip
The Sqoop command will look similar to the ones you have been using throughout this lab, except you will use `--query` instead of `--table`. Recall that when you use a `--query` command you must also define a `--split-by` column, or define `-m` to be 1.

Also, do not forget to add `$CONDITIONS` to the `WHERE` clause of your query, as demonstrated earlier in this unit.

***Solution*:**
In the command below, the "\" at the beginning of line 3 just in front of $CONDITIONS" **is** part of the actual command and is required for it to function properly. All other \ symbols in the command should be ignored in the command line.

```
# sqoop import --connect jdbc:mysql://sandbox/test?user=root \
--query "select * from salaries s where s.salary > 90000.00 and \
\$CONDITIONS" \
--split-by gender \
-m 2 \
--target-dir salaries3


--This is how it should appear in the command line

# sqoop import --connect jdbc:mysql://sandbox/test?user=root --
query "select * from salaries s where s.salary > 90000.00 and
\$CONDITIONS" --split-by gender -m 2 --target-dir salaries3
```

b. To verify the result, view the contents of the files in salaries3. You should have only two output files.

```
# hdfs dfs -ls salaries3
```

c. View the contents of part-m-00000 and part-m-00001.

```
# hdfs dfs -cat salaries3/part-m-00000

# hdfs dfs -cat salaries3/part-m-00001
```

Notice that one file contains females, and the other file contains males. Why?

Answer: You used gender as the split-by column, so all records with the same gender are sent to the same mapper.

d. Verify that the output files contain only records whose salary is greater than 90,000.00.

Result

You have imported the data from MySQL to HDFS using the entire table, specific columns, and also using the result of a query.