Walid Belal
Sept 2020
Minor revisions by Michael McKee Sept 2020

Summary

In this lab you will use Sqoop to export/import data between HDFS and a relational database (MySQL). Relational databases are a source of data for data lakes, and applications like Sqoop are used to transfer transactional data from relational databases to HDFS for storage and analysis. Sometimes you might also need to import data from HDFS to relational databases to make some quick analysis and queries.

In this lab you will first (1) load data from files to a MySQL database. The data file will be provided to you by your instructor. You will also use Sqoop to view this MySQL data. Second (2), you will use Sqoop to import the data into HDFS as a one-time import. Three (3) and four (4), you will use Sqoop to perform two types of incremental imports. Five, (5), you will export data from HDFS to a MySQL database.

Connect to the VPN

Open **Cisco Any Connect Security Mobile Client**

Type *vpn.sheridancollege.ca* and connect

You will need to use your Sheridan credentials to connect NOTE: You need to be connected all the time to the VPN

Connect to the servers

This exercise will use the Sheridan private cloud Hadoop **Prod** cluster.

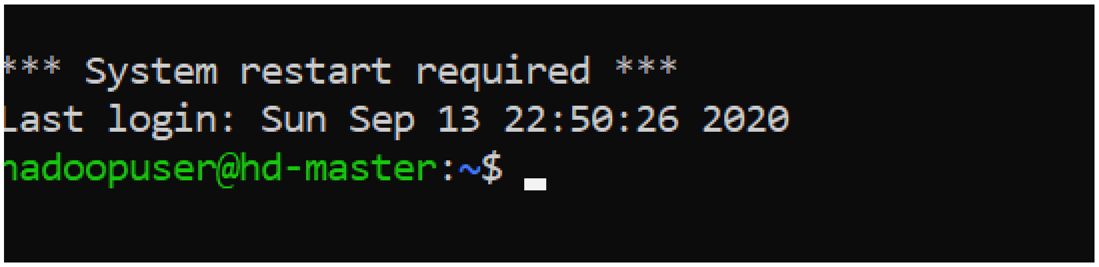
Use ssh from your laptop to connect to your **Prod** hd-master server then you can ssh from hd-master to the other servers

```
ssh hadoopuser@<router ip address>  -p 2221
```

the 2221 is the port that is used to forward you to the hd-master.

You will be asked for password. The default password is Sher1dan
It is advised to change the password on all the machines

Now you should get a prompt that looks like this



```
*** System restart required ***
Last login: Sun Sep 13 22:50:26 2020
hadoopuser@hd-master:~$ _
```

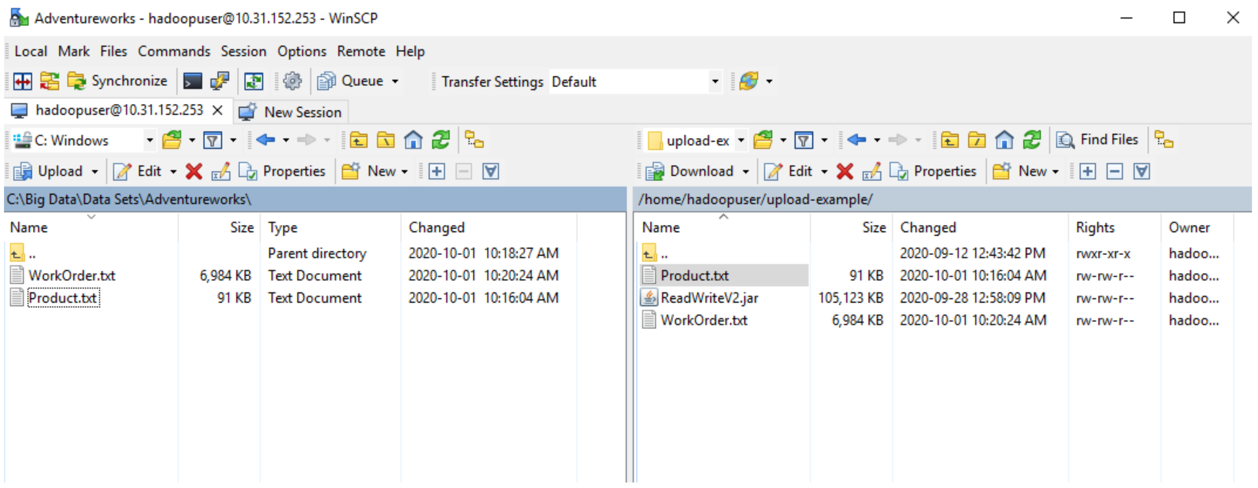
If you get it then you are inside hd_master

1. Copy product and work order tables

Download the files **WorkOrder.txt** and **Product.txt** from Slate. (Your instructor will give you directions on its location)

Open WinSCP and type the following configurations then press login Note that you need to use the IP address of the **Prod** router and the port is 2221. The default password is Sher1dan

Upload these files to your Prod hd-master node to **home/hadoopuser/upload-example**



Adventureworks - hadoopuser@10.31.152.253 - WinSCP
Local Mark Files Commands Session Options Remote Help
Synchronize
Queue
Transfer Settings Default
hadoopuser@10.31.152.253
New Session
C: Windows
upload-ex
Find Files
Upload
Edit
Properties
New
Download

C:\Big Data\Data Sets\Adventureworks\

Name	Size	Type	Changed
..		Parent directory	2020-10-01 10:18:27 AM
WorkOrder.txt	6,984 KB	Text Document	2020-10-01 10:20:24 AM
Product.txt	91 KB	Text Document	2020-10-01 10:16:04 AM

/home/hadoopuser/upload-example/

Name	Size	Changed	Rights	Owner
..		2020-09-12 12:43:42 PM	rwxr-xr-x	hadoo...
Product.txt	91 KB	2020-10-01 10:16:04 AM	rw-rw-r--	hadoo...
ReadWriteV2.jar	105,123 KB	2020-09-28 12:58:09 PM	rw-rw-r--	hadoo...
WorkOrder.txt	6,984 KB	2020-10-01 10:20:24 AM	rw-rw-r--	hadoo...

Copy the files to /var/liv/mysql-files

This is the only directory that SQL server is configured to allow imports from *sudo cp /home/hadoopuser/upload-example/WorkOrder.txt /var/lib/mysql-files*

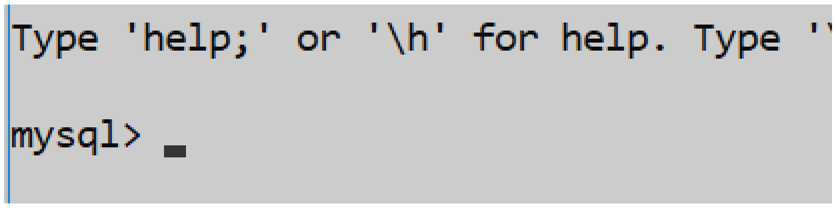
sudo cp /home/hadoopuser/upload-example/product.txt /var/lib/mysql-files

Create Tables in MySQL and Load them with data from the text files

Connect to MySQL on On hd-master

mysql -u root -p
the password is Sher1dan@

you will see the mysql> prompt



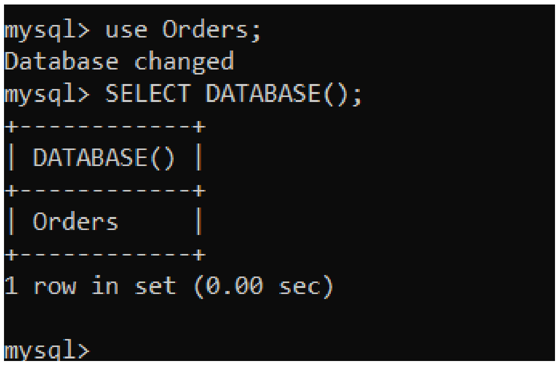
```
Type 'help;' or '\h' for help. Type '

mysql> _
```

CREATE DATABASE Orders;

Use Orders; Note: The ';' is important and terminates the command.

To make sure your current database is Orders, type the following:

SELECT DATABASE();

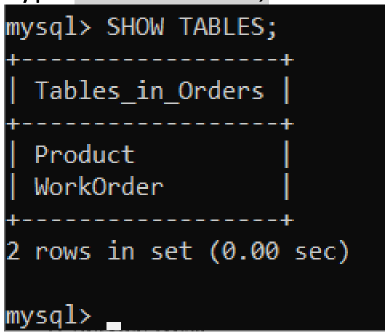

```
mysql> use Orders;
Database changed
mysql> SELECT DATABASE();
+------------+
| DATABASE() |
+------------+
| Orders     |
+------------+
1 row in set (0.00 sec)

mysql>
```

Now create the tables

```
CREATE TABLE Product
(ProdID INT PRIMARY KEY,
ProdName  VARCHAR(64),
ProdNumber VARCHAR(25),
Color VARCHAR(15),
ListPrice  DOUBLE(10,3),
Size VARCHAR(5));
```

```
CREATE TABLE WorkOrder
(WoID INT PRIMARY KEY,
ProdID INT,
OrderQty INT,
StockedQty INT,
ScrappedQty INT,
StartDate DATETIME,
EndDate DATETIME,
DueDate DATETIME,
ScrapReasonID INT,
ModifiedDate DATETIME,
FOREIGN KEY (ProdID) REFERENCES Product(ProdID));
```

Type *SHOW TABLES;* to make sure that the tables are selected


```
mysql> SHOW TABLES;
+------------------+
| Tables_in_Orders |
+------------------+
| Product          |
| WorkOrder        |
+------------------+
2 rows in set (0.00 sec)

mysql> _
```

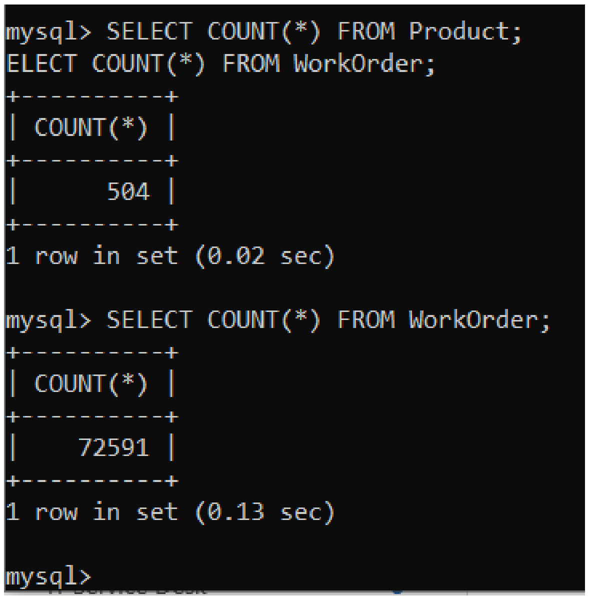
Now load the data from the text files to MySQL

```
LOAD DATA INFILE '/var/lib/mysql-files/product.txt' IGNORE
 INTO TABLE Product
FIELDS TERMINATED BY ','
IGNORE 1 LINES;
```

```
LOAD DATA INFILE '/var/lib/mysql-files/WorkOrder.txt' IGNORE
 INTO TABLE WorkOrder
FIELDS TERMINATED BY ','
IGNORE 1 LINES;
```

Type the below statements to make sure that the data was loaded

```
SELECT COUNT(*) FROM Product;
SELECT COUNT(*) FROM WorkOrder;
```



```
mysql> SELECT COUNT(*) FROM Product;
ELECT COUNT(*) FROM WorkOrder;
+----------+
| COUNT(*) |
+----------+
|      504 |
+----------+
1 row in set (0.02 sec)

mysql> SELECT COUNT(*) FROM WorkOrder;
+----------+
| COUNT(*) |
+----------+
|    72591 |
+----------+
1 row in set (0.13 sec)

mysql>
```

Quit from MySQL

```
quit
```

Use SQOOP to view the data

You can use Sqoop command lines to view the tables metadata and perform select statements

sqoop list-databases --connect jdbc:mysql://localhost/ --username root --password Sher1dan@

sqoop list-tables --connect jdbc:mysql://localhost/Orders --username root --password Sher1dan@

sqoop eval --connect jdbc:mysql://localhost/Orders --username root --password Sher1dan@ --query "SELECT COUNT(*) FROM WorkOrder"

2. Import data to Hadoop, one time

The following statement imports **WorkOrders** table to Hadoop

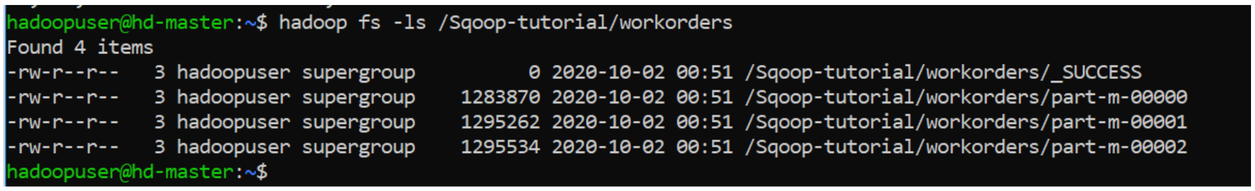
sqoop import --connect jdbc:mysql://localhost/Orders --username root --password Sher1dan@ --table WorkOrder --columns "WoID, ProdID, StartDate, EndDate" --where "WoID < 72000" --m 3 --target-dir /Sqoop-tutorial/workorders

Note: This is running a MapReduce job and may take over 2 minutes to run

With import you can import a subset of the columns using the **--column** parameter. You can also select a certain number of rows based on a condition. The condition is set using **--where** parameter.

Check the imported WorkOrders table

hadoop fs -ls /Sqoop-tutorial/workorders



```
hadoopuser@hd-master:~$ hadoop fs -ls /Sqoop-tutorial/workorders
Found 4 items
-rw-r--r--   3 hadoopuser supergroup          0 2020-10-02 00:51 /Sqoop-tutorial/workorders/_SUCCESS
-rw-r--r--   3 hadoopuser supergroup    1283870 2020-10-02 00:51 /Sqoop-tutorial/workorders/part-m-00000
-rw-r--r--   3 hadoopuser supergroup    1295262 2020-10-02 00:51 /Sqoop-tutorial/workorders/part-m-00001
-rw-r--r--   3 hadoopuser supergroup    1295534 2020-10-02 00:51 /Sqoop-tutorial/workorders/part-m-00002
hadoopuser@hd-master:~$
```

Notice that you have three data files in the folder. This is because you used 3 mappers to perform the import **--m 3**. Usually you will want to use more than one mapper in parallel when you are importing very large amounts of data.

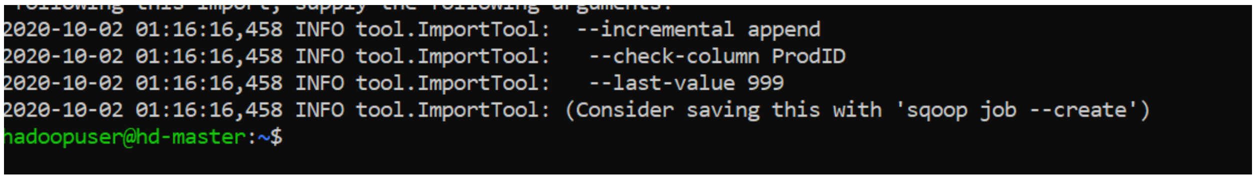
3. Import data to Hadoop, Incremental append

Extracting and loading data from RDMS to Hadoop and vice versa is where SQOOP is mainly used. Usually you will need to import data from tables to Hadoop in increments. For example, every night you will run a job that will import the new rows that were inserted to the database that day.

--incremental parameter indicates the type of increment. You will be using **append**.
--check column parameter is used which column will be used to determine what rows to append
--**last-value** parameter indicates what was the highest value of the check column in the last import. This value is set to 0 in the first import

sqoop import --connect jdbc:mysql://localhost/Orders --username root --password Sher1dan@ --table Product --columns "ProdID, ProdName, ProdNumber, Color, ListPrice, Size" --m 1 --target-dir /Sqoop-tutorial/products --incremental append --check-column ProdID --last-value "0"

At the end of the import Sqoop will write the last prodID in this import. You will be using this id as the value of the last-value parameter in the following import


```
2020-10-02 01:16:16,458 INFO tool.ImportTool:  --incremental append
2020-10-02 01:16:16,458 INFO tool.ImportTool:   --check-column ProdID
2020-10-02 01:16:16,458 INFO tool.ImportTool:   --last-value 999
2020-10-02 01:16:16,458 INFO tool.ImportTool: (Consider saving this with 'sqoop job --create')
hadoopuser@hd-master:~$
```

Now insert few more rows into **Product** table

mysql -u root -p

USE Orders;

INSERT INTO Product VALUES(1001,"Road-732 Green","BK-R19B-52","Green",539.99,52);
INSERT INTO Product VALUES(1002,"Road-733 Yellow","BK-R19B-53","Yellow",539.99,52);
INSERT INTO Product VALUES(1003,"Road-734 Red","BK-R19B-54","Red",539.99,52);
COMMIT;

Exit

Note if you want to redo the exercise then implement the following statements in mysql to reset the tables to their original state

USE Orders;

Delete from Product WHERE ProdID > 999;
COMMIT;

Now you will need to append the rows that you just inserted

```
sqoop import --connect jdbc:mysql://localhost/Orders --username root  --password Sher1dan@ --table Product --columns  "ProdID, ProdName, ProdNumber, Color, ListPrice, Size"  --m 1  --target-dir /Sqoop-tutorial/products --incremental append --check-column ProdID --last-value "999"
```

Note that in this import you are using 999 as the last-value. 999 is the highest value from the previous import

This method could be valid if the following:

1- ProdID is an incremental column. Every new row will have a higher ProdID than the previously inserted row. You can always use a timestamp column for this as timestamp will always increment.
2- None of the data is updated in the table. For example, if a product with id 4 was updated it will not be appended to HDFS.

Check the imported table

```
hadoop fs -ls /Sqoop-tutorial/products
```

You will find that you have two files 00000 from the first import and 00001 from the second

You can use tail argument to open the files and see if there are any duplicates

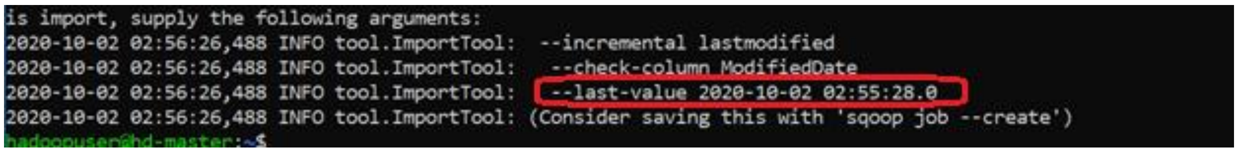
```
hadoop fs -tail /Sqoop-tutorial/products/part-m-00000
hadoop fs -tail /Sqoop-tutorial/products/part-m-00001
```

You should not find duplicates

4. Import data to Hadoop, Incremental modified

If rows are regularly updated in your database, then the append operation will not work as the updates will not be imported to Hadoop. In this case you need to use incremental lastmodified. Your check-column will be a datetime column or timestamp that will be updated with the current time whenever any modification happens to the row.

sqoop import --connect jdbc:mysql://localhost/Orders --username root --password Sher1dan@ --table WorkOrder --columns "WoID, ProdID, StartDate, EndDate, ModifiedDate" --m 1 --target-dir /Sqoop-tutorial/iworkorders --incremental lastmodified --check-column ModifiedDate



```
is import, supply the following arguments:
2020-10-02 02:56:26,488 INFO tool.ImportTool:  --incremental lastmodified
2020-10-02 02:56:26,488 INFO tool.ImportTool:   --check-column ModifiedDate
2020-10-02 02:56:26,488 INFO tool.ImportTool:   --last-value 2020-10-02 02:55:28.0
2020-10-02 02:56:26,488 INFO tool.ImportTool: (Consider saving this with 'sqoop job --create')
hadoopuser@hd-master:~$
```

You will see that there is date time for last-value. Please **write it down** because you will be using it in the next import

Now make some changes in the **WorkOrder** table

mysql -u root -p

USE Orders;

INSERT INTO WorkOrder VALUES(72592,1,10,12,2,NULL,NULL,NULL,2,now());
UPDATE WorkOrder SET StartDate = now() , ModifiedDate = now() WHERE WoID = 72587;
UPDATE WorkOrder SET StartDate = now() , ModifiedDate = now() WHERE WoID = 72584;

COMMIT;
exit

Now import the data again. Make sure to **use the date that your first import statement generated and not the one in the command line shown below**

Note if you want to redo the exercise then implement the following statements in mysql to reset the tables to their original state

Delete from WorkOrder WHERE WoID > 72591;
UPDATE WorkOrder SET StartDate = NULL , ModifiedDate = now() WHERE WoID = 72587;
UPDATE WorkOrder SET StartDate = NULL , ModifiedDate = now() WHERE WoID = 72584;

COMMIT;

```
sqoop import --connect jdbc:mysql://localhost/Orders --username root  --password Sher1dan@
--table WorkOrder --columns  "WoID, ProdID, StartDate, EndDate, ModifiedDate" --m 1  --
target-dir /Sqoop-tutorial/iworkorders --incremental lastmodified --check-column ModifiedDate
--last-value " 2020-10-02  02:55:28.0"   --merge-key WoID
```

In this statement you will find that I added two parameters
--last-value that has the date of the last import
-- *merge-key* that indicates which key should be used to merge updated rows. The merge key is usually your primary key

It will be very difficult to display the changes in Hadoop because it will order WoID alphanumerically so the updated rows will be in the middle of the file. You can copy the file from the target-dir **/Sqoop-tutorial/iworkorders** to the local file system and read it from there.

5. Export from HDFS

Create a table that looks exactly like WorkOrder table

```
mysql -u root -p
```
the password is Sher1dan@

```
Use Orders;
```

Now create the tables

```
CREATE TABLE MyWorkOrder
(WoID INT PRIMARY KEY,
ProdID INT,
OrderQty INT,
StockedQty INT,
ScrappedQty INT,
StartDate DATETIME,
EndDate DATETIME,
DueDate DATETIME,
ScrapReasonID INT,
ModifiedDate DATETIME,
FOREIGN KEY (ProdID) REFERENCES Product(ProdID));
```

Exit

Make sure that you still have in HDFS the **/Sqoop-tutorial/workorders** directory (these files were created when you imported WorkOrder table).

If it does not exist, then redo the import again as follows. If it does exist, skip this **sqoop import** step and continue with the **sqoop export** step.

sqoop import --connect jdbc:mysql://localhost/Orders --username root --password Sher1dan@ --table WorkOrder --columns "WoID, ProdID, StartDate, EndDate" --where "WoID < 72000" --m 3 --target-dir /Sqoop-tutorial/workorders

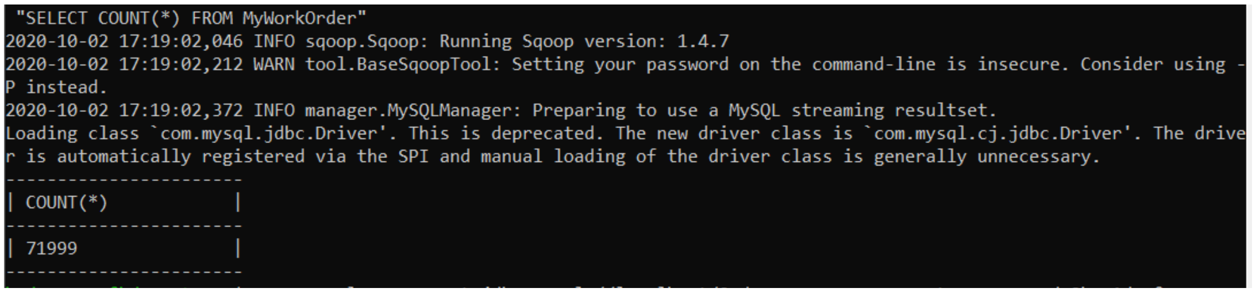
Export the files from HDFS to the MyWorkOrder table we just created.

sqoop export --connect jdbc:mysql://localhost/Orders --username root --password Sher1dan@ --table MyWorkOrder --columns "WoID, ProdID, StartDate, EndDate" --m 1 --export-dir /Sqoop-tutorial/workorders --input-fields-terminated-by '\0054'

The parameter ***input-fields-terminated-by*** is used to define the field separator in the file. 0054 is the octal representation of a comma. In the export you need to make sure that the column data types will accommodate the data in the file, or the export will fail.

You can use the below statements to check if the data was imported:

sqoop eval --connect jdbc:mysql://localhost/Orders --username root --password Sher1dan@ --query "SELECT COUNT(*) FROM MyWorkOrder"



```
 "SELECT COUNT(*) FROM MyWorkOrder"
2020-10-02 17:19:02,046 INFO sqoop.Sqoop: Running Sqoop version: 1.4.7
2020-10-02 17:19:02,212 WARN tool.BaseSqoopTool: Setting your password on the command-line is insecure. Consider using -
P instead.
2020-10-02 17:19:02,372 INFO manager.MySQLManager: Preparing to use a MySQL streaming resultset.
Loading class `com.mysql.jdbc.Driver'. This is deprecated. The new driver class is `com.mysql.cj.jdbc.Driver'. The drive
r is automatically registered via the SPI and manual loading of the driver class is generally unnecessary.
-----------------------
| COUNT(*)            |
-----------------------
| 71999               |
-----------------------
```

*sqoop eval --connect jdbc:mysql://localhost/Orders --username root --password Sher1dan@ --query "SELECT * FROM MyWorkOrder WHERE WoID < 100"*

Clean up Prod environment after you are done

1- Remove dir /Sqoop-tutorial/ directory in Hadoop HDFS
2- In MySQL, use DROP TABLE to remove Product, WorkOrder, and MyWorkOrder tables