Step 1: Start Hadoop system processes like NameNode, DataNode, etc and MySQL service.
Here is the command to start Hadoop system processes:
$ /usr/local/hadoop-2.6.0/sbin/start-all.sh
We can verify that they are running by using 'jps' command.
Here is the command to start MySQL service:
$ sudo service mysqld start

Step 2: Enter into MySQL prompt and create a new database.
We can use the following command to get into MySQL prompt:
$ sudo mysql
Once we get MySQL prompt, we can see a list existing databases via following command:
mysql> SHOW DATABASES;
Let's create a database named 'sample_db' with the following command:
mysql> CREATE DATABASE sample_db;

```
[acadgild@localhost bin]$ sudo mysql
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 7
Server version: 5.1.73 Source distribution

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> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| b1                 |
| metastore          |
| mysql              |
| test               |
+--------------------+
5 rows in set (0.00 sec)

mysql> CREATE DATABASE sample_db;
Query OK, 1 row affected (0.00 sec)

mysql> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| b1                 |
| metastore          |
| mysql              |
| sample_db          |
```

Step 3: Switch to the database created. Create a table by name 'employee' and insert few records into it.
Here is the command to switch to the database that we create in the previous step:
mysql> use sample_db;
Let's create a table by name 'employee' with following command:

```
mysql> CREATE TABLE employee
(emp_id int,
emp_name varchar(100),
emp_salary int,
years_of_exp int
);
```

The 'employee' schema has four fields: employee id, employee name, employee salary and years of experience.

```
mysql> use sample_db;
Database changed
mysql> CREATE TABLE employee
    -> (emp_id int,
    -> emp_name varchar(100),
    -> emp_salary int,
    -> years_of_exp int
    -> );
Query OK, 0 rows affected (0.01 sec)

mysql> SHOW TABLES;
+---------------------+
| Tables_in_sample_db |
+---------------------+
| employee            |
+---------------------+
1 row in set (0.00 sec)

mysql> DESCRIBE employee;
+--------------+--------------+------+-----+---------+-------+
| Field        | Type         | Null | Key | Default | Extra |
+--------------+--------------+------+-----+---------+-------+
| emp_id       | int(11)      | YES  |     | NULL    |       |
| emp_name     | varchar(100) | YES  |     | NULL    |       |
| emp_salary   | int(11)      | YES  |     | NULL    |       |
| years_of_exp | int(11)      | YES  |     | NULL    |       |
+--------------+--------------+------+-----+---------+-------+
4 rows in set (0.00 sec)

mysql>
```

Now let's insert few records into this table using INSERT command:

```
mysql> INSERT INTO employee VALUES(101,'Amitabh',20000,1);
mysql> INSERT INTO employee VALUES(102,'Shahrukh',10000,2);
mysql> INSERT INTO employee VALUES(103,'Akshay',11000,3);
mysql> INSERT INTO employee VALUES(104,'Anubhav',5000,4);
mysql> INSERT INTO employee VALUES(105,'Pawan',2500,5);
mysql> INSERT INTO employee VALUES(106,'Aamir',25000,1);
mysql> INSERT INTO employee VALUES(107,'Salman',17500,2);
```

We have inserted seven records with employee id in sequential manner starting from value '101'.

We can see the result by using SELECT command on this table:

```
SELECT * FROM employee;
```

```
mysql> INSERT INTO employee VALUES(101,'Amitabh',20000,1);
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO employee VALUES(102,Shahrukh,10000,2);
ERROR 1054 (42S22): Unknown column 'Shahrukh' in 'field list'
mysql> INSERT INTO employee VALUES(102,'Shahrukh',10000,2);
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO employee VALUES(103,'Akshay',11000,3);
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO employee VALUES(104,'Anubhav',5000,4);
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO employee VALUES(105,'Pawan',2500,5);
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO employee VALUES(106,'Aamir',25000,1);
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO employee VALUES(107,'Salman',17500,2);
Query OK, 1 row affected (0.00 sec)

mysql> SELECT * FROM employee;
+--------+----------+------------+--------------+
| emp_id | emp_name | emp_salary | years_of_exp |
+--------+----------+------------+--------------+
|    101 | Amitabh  |      20000 |            1 |
|    102 | Shahrukh |      10000 |            2 |
|    103 | Akshay   |      11000 |            3 |
|    104 | Anubhav  |       5000 |            4 |
|    105 | Pawan    |       2500 |            5 |
|    106 | Aamir    |      25000 |            1 |
|    107 | Salman   |      17500 |            2 |
+--------+----------+------------+--------------+
7 rows in set (0.00 sec)

mysql>
```

Step 4: Grant privileges to access tables outside of MySQL and commit all the operations performed till now.
Here are few commands to grant access to this table and other related commands.
mysql> grant all on *.* to 'root'@'localhost' with grant option;
mysql> flush privileges;
mysql> commit;
We can exit from MySQL now:
mysql> exit;

```
mysql> grant all on *.* to 'root'@'localhost' with grant option;
Query OK, 0 rows affected (0.00 sec)

mysql> flush privileges;
Query OK, 0 rows affected (0.00 sec)

mysql> commit;
Query OK, 0 rows affected (0.00 sec)
```

Step 5: Use Sqoop import command to import data from MySQL table into HDFS.
Here is the import command to achieve this:

$ sqoop import --connect jdbc:mysql://localhost/sample_db --table employee --username root -P --split-by 'years_of_exp' --target-dir '/sqoop_output' -m 2;

In the above command, connect parameter takes a JDBC URL to connect to a specific database, in this case 'sample_db'. We can specify login credentials (user name and password to MySQL) and table name subsequently. An interesting parameter here is **'split-by'**. Sqoop automatically splits the table data by primary key column which will split total number of records based on that column's data and performs transfer operation in parallel by assigning the task to several mappers. If primary key is not present for a table, we have to use split-by option followed by a column name for Sqoop to consider that field while splitting data.

In this example, we have mentioned number of mappers as 2 and we are splitting data based on the column 'years_of_exp'. If we observe the data inserted into employee table, we have four similar values in this column (1 and 2 in repeated manner). So one mapper will get these four data records and another mapper gets remaining records in the execution of the above command.

```
[acadgild@localhost bin]$ sqoop import --connect jdbc:mysql://localhost/sample_db --table employee --username root -P --split
-by 'years_of_exp' --target-dir '/sqoop_output' -m 2;
Warning: /usr/local/sqoop/../hcatalog does not exist! HCatalog jobs will fail.
Please set $HCAT_HOME to the root of your HCatalog installation.
Warning: /usr/local/sqoop/../accumulo does not exist! Accumulo imports will fail.
Please set $ACCUMULO_HOME to the root of your Accumulo installation.
Warning: /usr/local/sqoop/../zookeeper does not exist! Accumulo imports will fail.
Please set $ZOOKEEPER_HOME to the root of your Zookeeper installation.
2018-01-23 02:20:39,614 INFO  [main] sqoop.Sqoop: Running Sqoop version: 1.4.5
Enter password:
2018-01-23 02:20:41,479 INFO  [main] manager.MySQLManager: Preparing to use a MySQL streaming resultset.
2018-01-23 02:20:41,480 INFO  [main] tool.CodeGenTool: Beginning code generation
2018-01-23 02:20:42,157 INFO  [main] manager.SqlManager: Executing SQL statement: SELECT t.* FROM `employee` AS t LIMIT 1
2018-01-23 02:20:42,239 INFO  [main] manager.SqlManager: Executing SQL statement: SELECT t.* FROM `employee` AS t LIMIT 1
2018-01-23 02:20:42,285 INFO  [main] orm.CompilationManager: HADOOP_MAPRED_HOME is /usr/local/hadoop-2.6.0
```

Let's check the output by examining the contents of the directory /sqoop_output in HDFS.
$ Hadoop fs –ls /sqoop_output

```
[acadgild@localhost ~]$ hadoop fs -ls /sqoop_output
18/01/23 02:25:10 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java cl
asses where applicable
Found 3 items
-rw-r--r--   1 acadgild supergroup          0 2018-01-23 02:22 /sqoop_output/_SUCCESS
-rw-r--r--   1 acadgild supergroup         78 2018-01-23 02:22 /sqoop_output/part-m-00000
-rw-r--r--   1 acadgild supergroup         55 2018-01-23 02:22 /sqoop_output/part-m-00001
```

As we can see there are two files one from each mapper. Let's see the contents of these two files:

$ hadoop fs -cat /sqoop_output/part-m-00000

$ hadoop fs -cat /sqoop_output/part-m-00001

```
[acadgild@localhost ~]$ hadoop fs -cat /sqoop_output/part-m-00000
18/01/23 02:25:37 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java cl
asses where applicable
101,Amitabh,20000,1
102,Shahrukh,10000,2
106,Aamir,25000,1
107,Salman,17500,2
[acadgild@localhost ~]$ hadoop fs -cat /sqoop_output/part-m-00001
18/01/23 02:25:53 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java cl
asses where applicable
103,Akshay,11000,3
104,Anubhav,5000,4
105,Pawan,2500,5
[acadgild@localhost ~]$
```

All records from MySQL table 'employee' have been imported to HDFS successfully.