**[training@localhost ~]$ sqoop list-databases --connect jdbc:mysql://localhost --username root**

information\_schema  
b2sqoop  
batch1  
batch2  
bossini  
hivemetastore  
movielens  
mysql  
oct2017  
oozie  
sqoopb2  
training

**[training@localhost ~]$ sqoop list-tables --connect jdbc:mysql://localhost/oct2017 --username root**

employees  
employees\_exp\_stg  
employees\_export

**[training@localhost ~]$ sqoop import --connect jdbc:mysql://localhost/oct2017 -username root --table employees**

17/11/16 16:34:02 INFO manager.MySQLManager: Preparing to use a MySQL streaming resultset.  
17/11/16 16:34:02 INFO tool.CodeGenTool: Beginning code generation  
17/11/16 16:34:03 INFO manager.SqlManager: Executing SQL statement: SELECT t.\* FROM `employees` AS t LIMIT 1  
17/11/16 16:34:03 INFO orm.CompilationManager: HADOOP\_HOME is /usr/lib/hadoop  
17/11/16 16:34:03 INFO orm.CompilationManager: Found hadoop core jar at: /usr/lib/hadoop/hadoop-core.jar  
17/11/16 16:34:03 ERROR orm.CompilationManager: Could not rename /tmp/sqoop-training/compile/ab15ad69de1b525a618ff34e392a399a/employees.java to /home/training/./employees.java  
java.io.IOException: Destination '/home/training/./employees.java' already exists  
 at org.apache.commons.io.FileUtils.moveFile(FileUtils.java:1811)  
 at com.cloudera.sqoop.orm.CompilationManager.compile(CompilationManager.java:229)  
 at com.cloudera.sqoop.tool.CodeGenTool.generateORM(CodeGenTool.java:85)  
 at com.cloudera.sqoop.tool.ImportTool.importTable(ImportTool.java:369)  
 at com.cloudera.sqoop.tool.ImportTool.run(ImportTool.java:455)  
 at com.cloudera.sqoop.Sqoop.run(Sqoop.java:146)  
 at org.apache.hadoop.util.ToolRunner.run(ToolRunner.java:65)  
 at com.cloudera.sqoop.Sqoop.runSqoop(Sqoop.java:182)  
 at com.cloudera.sqoop.Sqoop.runTool(Sqoop.java:221)  
 at com.cloudera.sqoop.Sqoop.runTool(Sqoop.java:230)

at com.cloudera.sqoop.Sqoop.main(Sqoop.java:239)

17/11/16 16:34:03 INFO orm.CompilationManager: Writing jar file: /tmp/sqoop-training/compile/ab15ad69de1b525a618ff34e392a399a/employees.jar

17/11/16 16:34:03 WARN manager.MySQLManager: It looks like you are importing from mysql.

17/11/16 16:34:03 WARN manager.MySQLManager: This transfer can be faster! Use the --direct

17/11/16 16:34:03 WARN manager.MySQLManager: option to exercise a MySQL-specific fast path.

17/11/16 16:34:03 INFO manager.MySQLManager: Setting zero DATETIME behavior to convertToNull (mysql)

17/11/16 16:34:03 INFO mapreduce.ImportJobBase: Beginning import of employees

17/11/16 16:34:05 INFO db.DataDrivenDBInputFormat: BoundingValsQuery: SELECT MIN(`emp\_no`), MAX(`emp\_no`) FROM `employees`

17/11/16 16:34:07 INFO mapred.JobClient: Running job: job\_201711161133\_0004

17/11/16 16:34:08 INFO mapred.JobClient: map 0% reduce 0%

17/11/16 16:35:02 INFO mapred.JobClient: map 25% reduce 0%

17/11/16 16:35:03 INFO mapred.JobClient: map 50% reduce 0%

17/11/16 16:35:06 INFO mapred.JobClient: map 75% reduce 0%

17/11/16 16:35:08 INFO mapred.JobClient: map 100% reduce 0%

17/11/16 16:35:08 INFO mapred.JobClient: Job complete: job\_201711161133\_0004

17/11/16 16:35:08 INFO mapred.JobClient: Counters: 12

17/11/16 16:35:08 INFO mapred.JobClient: Job Counters

17/11/16 16:35:08 INFO mapred.JobClient: SLOTS\_MILLIS\_MAPS=95747

17/11/16 16:35:08 INFO mapred.JobClient: Total time spent by all reduces waiting after reserving slots (ms)=0

17/11/16 16:35:08 INFO mapred.JobClient: Total time spent by all maps waiting after reserving slots (ms)=0

17/11/16 16:35:08 INFO mapred.JobClient: Launched map tasks=4

17/11/16 16:35:08 INFO mapred.JobClient: SLOTS\_MILLIS\_REDUCES=0

17/11/16 16:35:08 INFO mapred.JobClient: FileSystemCounters

17/11/16 16:35:08 INFO mapred.JobClient: HDFS\_BYTES\_READ=441

17/11/16 16:35:08 INFO mapred.JobClient: FILE\_BYTES\_WRITTEN=264432

17/11/16 16:35:08 INFO mapred.JobClient: HDFS\_BYTES\_WRITTEN=285

17/11/16 16:35:08 INFO mapred.JobClient: Map-Reduce Framework

17/11/16 16:35:08 INFO mapred.JobClient: Map input records=8

17/11/16 16:35:08 INFO mapred.JobClient: Spilled Records=0

17/11/16 16:35:08 INFO mapred.JobClient: Map output records=8

17/11/16 16:35:08 INFO mapred.JobClient: SPLIT\_RAW\_BYTES=441

17/11/16 16:35:09 INFO mapreduce.ImportJobBase: Transferred 285 bytes in 64.8982 seconds (4.3915 bytes/sec)

17/11/16 16:35:09 INFO mapreduce.ImportJobBase: Retrieved 8 records.

**[training@localhost ~]$ hadoop fs -ls employees**

Found 6 items

-rw-r--r-- 1 training supergroup 0 2017-11-16 16:35 /user/training/employees/\_SUCCESS

drwxr-xr-x - training supergroup 0 2017-11-16 16:34 /user/training/employees/\_logs

-rw-r--r-- 1 training supergroup 72 2017-11-16 16:34 /user/training/employees/part-m-00000

-rw-r--r-- 1 training supergroup 72 2017-11-16 16:35 /user/training/employees/part-m-00001

-rw-r--r-- 1 training supergroup 71 2017-11-16 16:35 /user/training/employees/part-m-00002

-rw-r--r-- 1 training supergroup 70 2017-11-16 16:35 /user/training/employees/part-m-00003

**[training@localhost ~]$ hadoop fs -cat /user/training/employees/part-m-00000**

119,1989-08-04,iii,ooo,F,2010-03-11

120,1989-08-05,jjj,ppp,F,2010-04-11

**[training@localhost ~]$ hadoop fs -cat /user/training/employees/part-m-00001**

121,1989-08-06,kkk,ttt,F,2010-05-11

122,1989-08-07,lll,uuu,F,2010-06-11

**[training@localhost ~]$ hadoop fs -cat /user/training/employees/part-m-00002**

123,1989-08-08,mmm,ooo,F,2010-03-11

124,1989-08-09,nnn,pp,F,2010-04-11

**[training@localhost ~]$ hadoop fs -cat /user/training/employees/part-m-00003**

125,1989-08-10,000,1tt,F,2010-05-11

126,1989-08-11,pp,uu,F,2010-06-11

Launched map tasks=4 the above are the 4 map tasks. In Sqoop, the default no of mapper launched is 4.

**[training@localhost ~]$ sqoop help**

usage: sqoop COMMAND [ARGS]

Available commands:

**codegen** Generate code to interact with database records

**create-hive-table** Import a table definition into Hive

**eval** Evaluate a SQL statement and display the results

**export**  Export an HDFS directory to a database table

**help** List available commands

**import** Import a table from a database to HDFS

**import-all-tables** Import tables from a database to HDFS

**job** Work with saved jobs

**list-databases** List available databases on a server

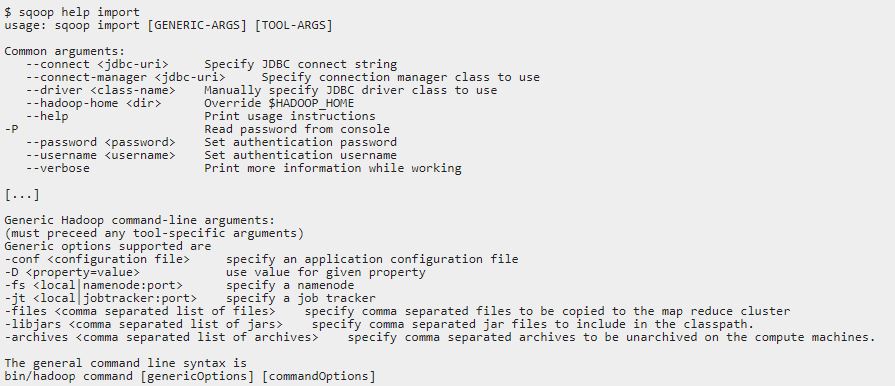
**list-tables** List available tables in a database

**merge**  Merge results of incremental imports

**metastore** Run a standalone Sqoop metastore

**version**  Display version information

See **'sqoop help COMMAND'** for information on a specific command.



Generic hadoop commands are preceded by a single dash (-) character, whereas tool specified commands starts with two dashes (--)

It’s important that you **do not** use the URL localhost if you intend to use Sqoop with a distributed Hadoop cluster. The connect string you supply will be used on TaskTracker nodes throughout your MapReduce cluster; if you specify the literal name localhost, each node will connect to a different database (or more likely, no database at all). Instead, you should use the full hostname or IP address of the database host that can be seen by all your remote nodes.

| **Argument** | **Description** |
| --- | --- |
| --append | Append data to an existing dataset in HDFS |
| --as-avrodatafile | Imports data to Avro Data Files |
| --as-sequencefile | Imports data to SequenceFiles |
| --as-textfile | Imports data as plain text (default) |
| --boundary-query <statement> | Boundary query to use for creating splits |
| --columns <col,col,col…> | Columns to import from table |
| --direct | Use direct import fast path |
| --direct-split-size <n> | Split the input stream every *n* bytes when importing in direct mode |
| --inline-lob-limit <n> | Set the maximum size for an inline LOB |
| -m,--num-mappers <n> | Use *n* map tasks to import in parallel |
| -e,--query <statement> | Import the results of *statement*. |
| --split-by <column-name> | Column of the table used to split work units |
| --table <table-name> | Table to read |
| --target-dir <dir> | HDFS destination dir |
| --warehouse-dir <dir> | HDFS parent for table destination |
| --where <where clause> | WHERE clause to use during import |
| -z,--compress | Enable compression |
| --compression-codec <c> | Use Hadoop codec (default gzip) |
| --null-string <null-string> | The string to be written for a null value for string columns |
| --null-non-string <null-string> | The string to be written for a null value for non-string columns |

Sqoop can also import the result set of an arbitrary SQL query. Instead of using the --table, --columns and --where arguments, you can specify a SQL statement with the --query argument.

When importing a free-form query, you must specify a destination directory with --target-dir.

$ sqoop import \

--query 'SELECT a.\*, b.\* FROM a JOIN b on (a.id == b.id) WHERE $CONDITIONS' \

--split-by a.id --target-dir /user/foo/joinresults

Alternately, the query can be executed once and imported serially, by specifying a single map task with -m 1:

$ sqoop import \

--query 'SELECT a.\*, b.\* FROM a JOIN b on (a.id == b.id) WHERE $CONDITIONS' \

-m 1 --target-dir /user/foo/joinresults