What is Sqoop?

- Sqoop is a tool used for data transfer between RDBMS (like MySQL, Oracle SQL etc.) and Hadoop (Hive, HDFS, and HBASE etc.)
- It is used to import data from RDBMS to Hadoop and export data from Hadoop to RDBMS.

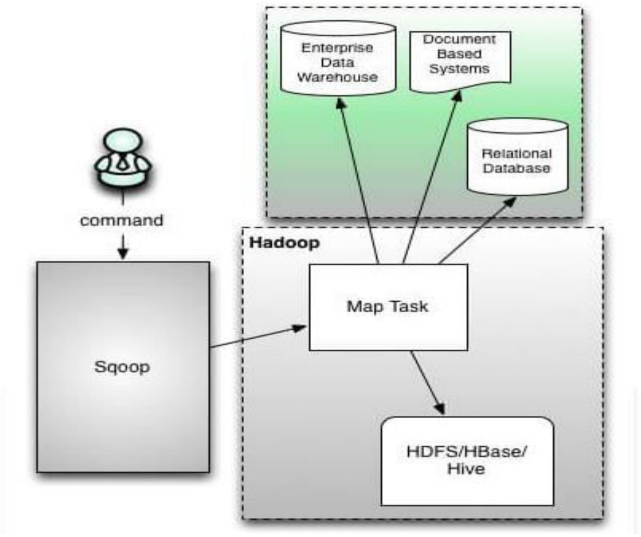
Why is Sqoop used?

- Big data developer's work start once the data is in Hadoop system like HDFS, Hive or Hbase. They do their magical stuff to find all the golden information hidden on such a huge amount of data.
- Before Sqoop developers used to write code to import and export data between Hadoop and RDBMS.
- Sqoop uses MapReduce mechanism for its operations like import and export work and work on parallel mechanism as well as fault tolerance.
- In Sqoop we need to mention the source, target and the rest of the work will be done by the Sqoop tool.

Features of Sqoop: Sqoop is robust, easily usable and has community support and contribution.

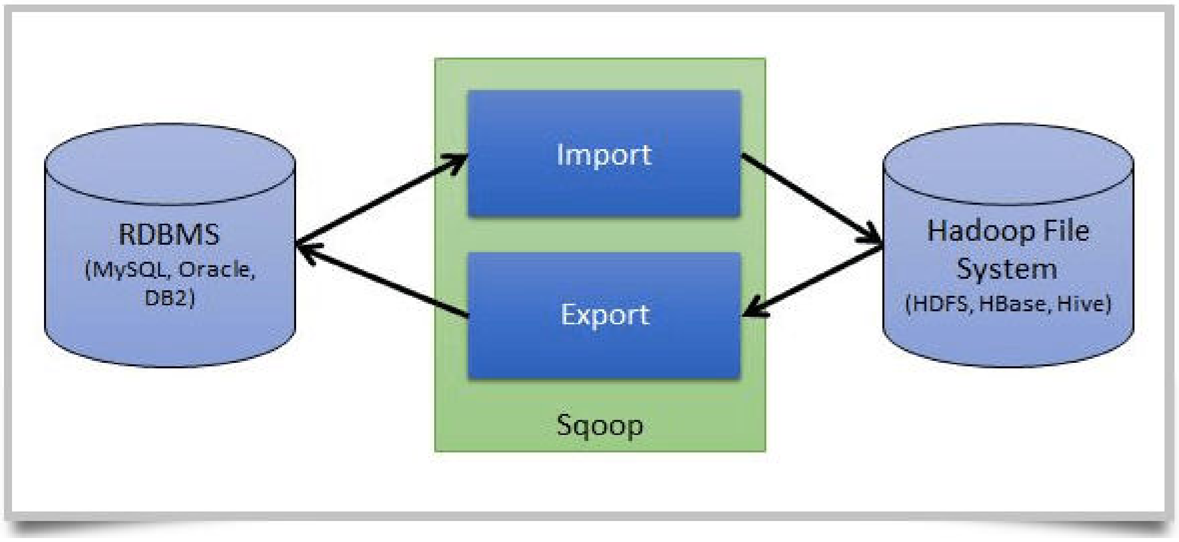
- Full Load
- Incremental Load
- Parallel import/export
- Import results of SQL query
- Compression
- Connectors for all major RDBMS Databases
- Load data directly into Hive/HBase

Sqoop Architecture:



When Sqoop starts functioning, only mapper job will run and reducer is not required because complete import and export process doesn't require any aggregation and so there is no need of reducers in Sqoop.

Main functions of Sqoop are: **Import & Export**.



Sqoop Import:

- The Sqoop import tool will import each table of the RDBMS in Hadoop and each row of the table will be considered as a record in the HDFS.
- All records are stored as text data in text files or as binary data in Avro and Sequence files.

Sqoop Export:

- The Sqoop export tool will export Hadoop files back to RDBMS tables.
- The records in the HDFS files will be the rows of a table and delimited with a user-specified delimiter.

Sqoop Commands:

1) sqoop version:

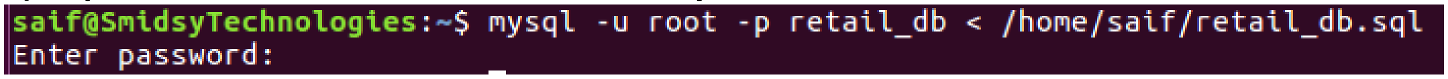
2) Sqoop provides 'list-databases' command to show all the databases in your MySQL or any other RDBMS.

3) Sqoop provides list-tables to see all the tables in DB

4) Sqoop eval command allows user to quickly run user defined SQL queries and get the output on console.

5) --boundary-query: By default sqoop will use query select min(), max() to find out boundaries for creating splits. In some cases this query is not the most optimal so you can specify any arbitrary query returning two numeric columns using --boundary-query argument.
Reason to use: If --split-by is not giving you the optimal performance you can use this to improve the performance further.
e.g. --boundary-query "SELECT min(id), max(id) from table"

6) Import database data from flat file into MySQL DB.

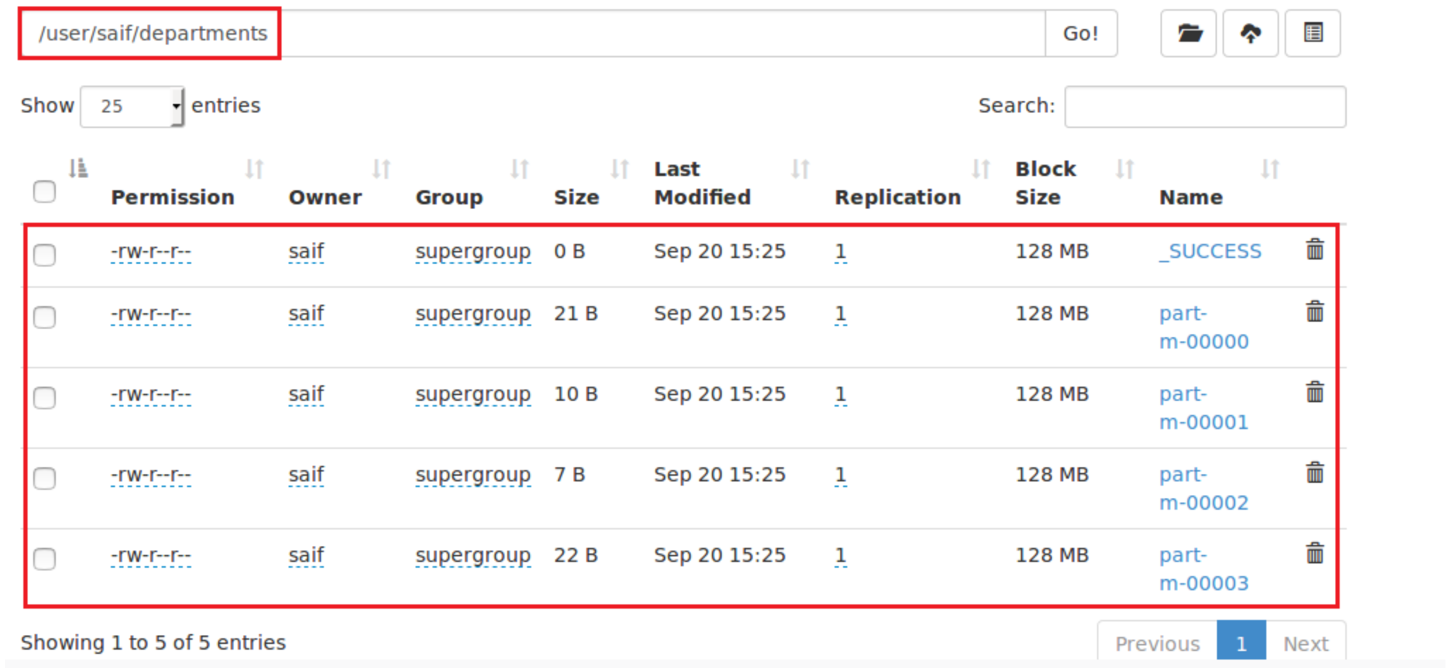


```
saif@SmidsyTechnologies:~$ mysql -u root -p retail_db < /home/saif/retail_db.sql
Enter password:
```

7) Import table data which has PK into HDFS.

Note:
1) BoundingValsQuery: SELECT MIN(`department_id`), MAX(`department_id`) FROM `departments`
2) number of splits:4 (Default)
3) Map-Reduce Framework
Map input records=6
Map output records=6

Weburl Path: http://localhost:50070/
Path: /user/saif/



Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-rw-r--r--	saif	supergroup	0 B	Sep 20 15:25	1	128 MB	_SUCCESS
-rw-r--r--	saif	supergroup	21 B	Sep 20 15:25	1	128 MB	part-m-00000
-rw-r--r--	saif	supergroup	10 B	Sep 20 15:25	1	128 MB	part-m-00001
-rw-r--r--	saif	supergroup	7 B	Sep 20 15:25	1	128 MB	part-m-00002
-rw-r--r--	saif	supergroup	22 B	Sep 20 15:25	1	128 MB	part-m-00003

Showing 1 to 5 of 5 entries

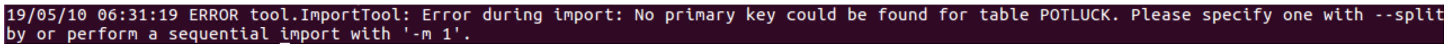
Import control arguments:

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)
--as-parquetfile	Imports data to Parquet Files
--boundary-query <statement>	Boundary query to use for creating splits
--columns <col, col, col …>	Columns to import from table
--delete-target-dir	Delete the import target directory if it exists
--direct	Use direct connector if exists for the database
--fetch-size <n>	Number of entries to read from database at once.
--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. Cannot be used with --autoreset-to-one-mapper option.
--autoreset-to-one-mapper	Import should use one mapper if a table has no primary key and no split-by column is provided. Cannot be used with --split-by <col> option.
--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

8) Import table data which don't have PK into HDFS.
1) Sqoop by default uses 4 concurrent map tasks to import data in Hadoop.
2) While performing the parallel imports Sqoop needs criteria by which it can split the workload. Sqoop uses the splitting column to split the workload.
3) By default, Sqoop will identify the primary key column (if present) in a table to use as the splitting column.
4) The low and high values of splitting column are retrieved from databases and the map task operate on evenly sized components of total range.

Error:

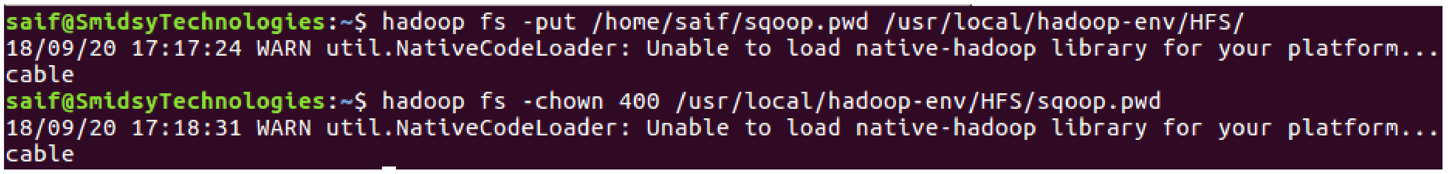


```
19/05/10 06:31:19 ERROR tool.ImportTool: Error during import: No primary key could be found for table POTLUCK. Please specify one with --split
by or perform a sequential import with '-m 1'.
```

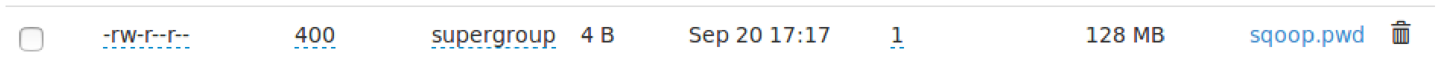
There are 2 ways to import MySQL table data into HDFS if table doesn't have primary key.
1) Add -m 1: -m 1 it tells to use sequential import with 1 mapper.
2) Add --split-by: Sqoop creates split based on values in a particular column of the table which is specified by the user in --split-by, throughout the import command.

9) Protecting your password:
Typing your password into command line is insecure. There are two methods other than specifying the password on the command line with --password parameter.
Method 1: The first option is to use the parameter -P that will instruct Sqoop to read the password from standard input.
Method 2: Save your password in a file and specify the path to this file with the parameter --password-file. The file containing the password can either be on local FS or HDFS.

Put the file from LFS to HDFS.

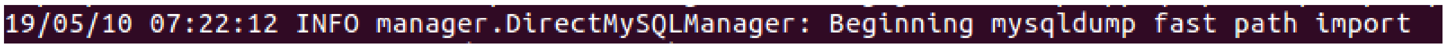


```
saif@SmidsyTechnologies:~$ hadoop fs -put /home/saif/sqoop.pwd /usr/local/hadoop-env/HFS/
18/09/20 17:17:24 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform...
cable
saif@SmidsyTechnologies:~$ hadoop fs -chown 400 /usr/local/hadoop-env/HFS/sqoop.pwd
18/09/20 17:18:31 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform...
cable
```



```
-rw-r--r--    400    supergroup    4 B    Sep 20 17:17    1    128 MB    sqoop.pwd
```

10) Speed up Transfers: --direct mode allow to extract the data quickly.

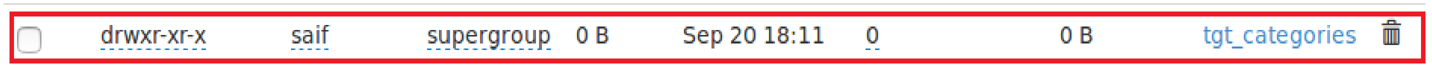


```
19/05/10 07:22:12 INFO manager.DirectMySQLManager: Beginning mysqldump fast path import
```

Note: Limitations of faster import.
1) Sqoop can only perform --direct mode imports from **PostgreSQL, Oracle and Netezza**.
2) Binary formats like **Sequence File** or **Avro** won't work with direct mode import.
3) In case of MySQL, when using --direct parameter with import query, Sqoop will take advantage of MySQL's native utility like **mysqldump** and **mysqlimport**, rather than using the **JDBC** interface for transferring data.

11) Importing the data to target directory:
--target-dir specify the directory on HDFS where Sqoop should import your data. The only requirement is that this directory must not exist prior to running the Sqoop command.
By default, Sqoop will create a directory with the same name as the imported table inside your home directory on HDFS and import all data there.

Path: /user/saif

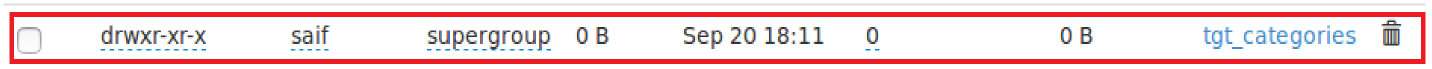


```
drwxr-xr-x   saif   supergroup   0 B   Sep 20 18:11   0   0 B   tgt_categories
```

12) --delete-target-dir: This deletes your existing directory with all files within it & creates a new directory. With this argument you can overcome the error: File exist during sqoop import.

13) --append: Add data to an existing file in HDFS.

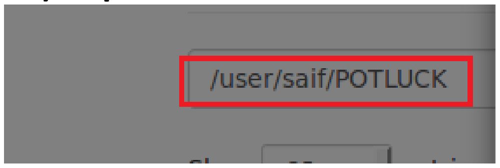
14) Importing the data inside parent directory:
By default, Sqoop imports data to your **home directory** on HDFS, the --warehouse-dir parameter allows you to specify only parent directory.



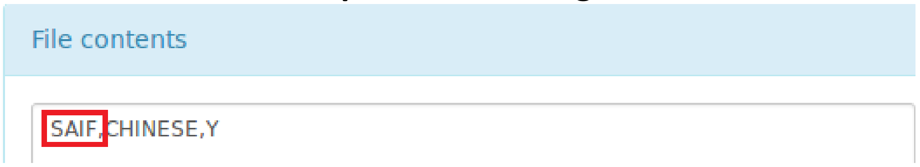
```
drwxr-xr-x   saif   supergroup   0 B   Sep 20 18:11   0   0 B   tgt_categories
```

15) Import all rows of a table from MySQL, but specific columns of table.
--columns allow to specify which columns to import.

16) Import subset of table data: --where allows us to Import rows on a given condition



/user/saif/POTLUCK



File contents

```
SAIF,CHINESE,Y
```

17) Import all tables of a MySQL DB into HDFS: import-all-tables

Import all tables from database at once using one command rather than importing the tables one by one.

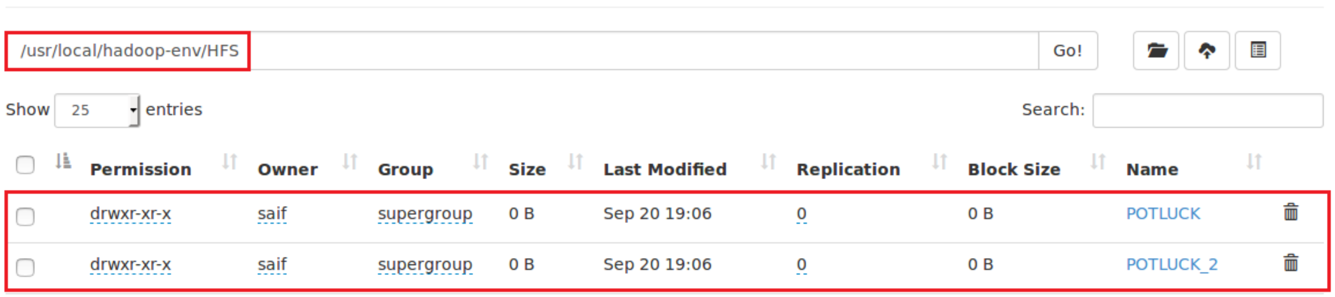
--target-dir parameter is not allowed

--warehouse-dir parameter is allowed

Note:

1) Each table must have a single column primary key or --autoreset-to-one-mapper option must be used.

2) We must intend to import all columns of each table.



Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
drwxr-xr-x	saif	supergroup	0 B	Sep 20 19:06	0	0 B	POTLUCK
drwxr-xr-x	saif	supergroup	0 B	Sep 20 19:06	0	0 B	POTLUCK_2

Note: We can use –m 1 or --split-by columns as well but that will apply on all tables of database irrespective of tables having primary key or not. But --autoreset-to-one-mapper will be applied to only those tables who don't have a primary key.

18) exclude some tables: --exclude-tables accepts a comma-separated list of table names that should be excluded from the bulk import.

--target-dir parameter is not allowed

--warehouse-dir parameter is allowed

19) Compressing Imported Data: Map reduce already has support for compression, Sqoop simply reuses its powerful abilities to provide compression options.

--compress output files will be compressed using GZip codec, with a .gz extension

--compression-codec output files will be compressed using BZip2 codec, with a .bz2 extension.

Note: -text command to view zip file data in HDFS.

20) Import MySQL data into HDFS in various binary file format

Sqoop supports three different file formats, one of those is text and the other two are binary. By default, Sqoop import the data in text files that are in human readable format, platform independent and simplest structure.

The binary formats are Avro and Hadoop's SequenceFile. These binary formats provide the most precise representation possible of the imported data.

21) --fields-terminated-by: You can change the default delimiter [,] to any delimiter as output.

22) Incremental Imports:
Sqoop supports two types of incremental imports: **append and lastmodified**.
You can use the --incremental argument to specify the type of incremental import to perform.

--append mode:
You should specify the append mode when importing a table, where new rows are continually added with increasing row id values. You must specify the column containing the row's id with --check-column. Sqoop imports rows where the check column has a value greater than the one specified with --last-value.

--lastmodified mode:
An alternate table update strategy supported by Sqoop is called lastmodified mode. This should be used when rows of the source table are updated, and each such update will set the value of a last-modified column to the current timestamp. Rows where the check column holds a timestamp more recent than the timestamp specified with --last-value are imported.

At the end of an incremental import, the value which should be specified as --last-value for a subsequent import is printed to the screen. When running a subsequent import, you should specify --last-value in this way to ensure you import only the new or updated data. This is handled automatically by creating an incremental import as a saved job, which is the preferred mechanism for performing a recurring incremental import.

Now, let's insert two more records in INC_IMP table in MySQL DB:

The following syntax is used for the incremental option in Sqoop import command.
--incremental <mode>
--check-column <column name>
--last value <last check column value>

23) Sqoop Job:
Imports and exports can be repeatedly performed by issuing the same command multiple times. Especially when using the incremental import capability, this is an expected scenario.

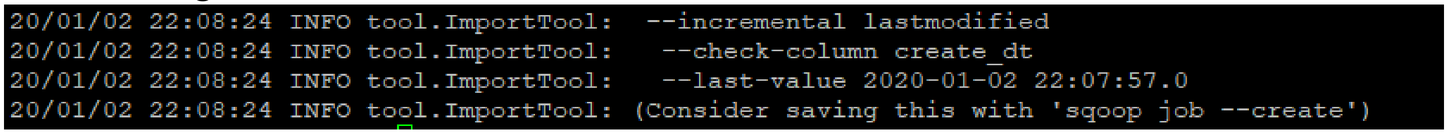
Sqoop allows you to define saved jobs which make this process easier. A saved job records the configuration information required to execute a Sqoop command at a later time.

By default, job descriptions are saved to a private repository stored in $HOME/.sqoop/. You can configure Sqoop to instead use a shared metastore, which makes saved jobs available to multiple users across a shared cluster.

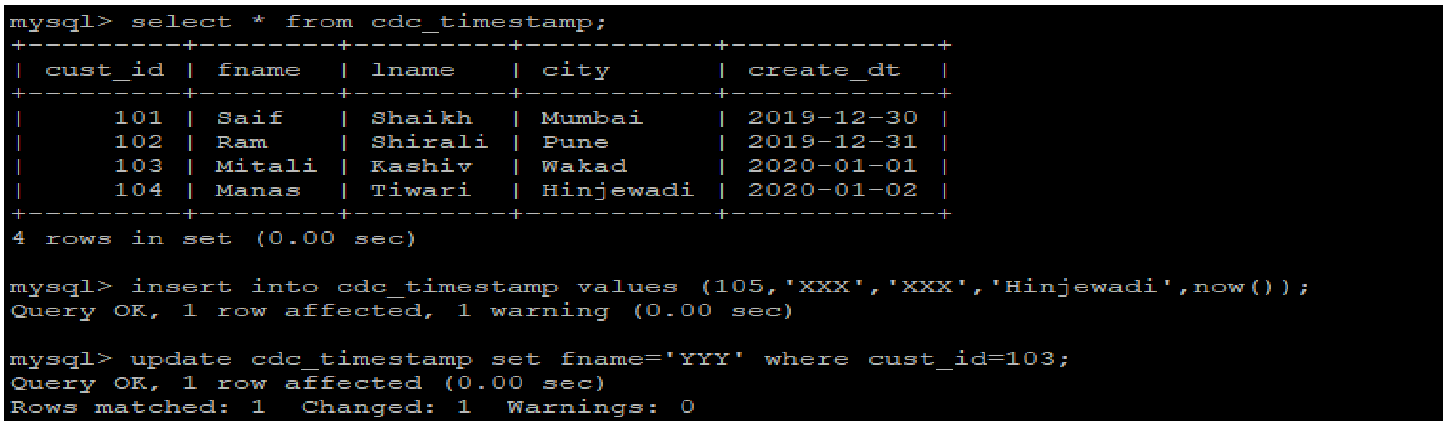
Argument	Description
--create <job-id>	Define a new saved job with the specified job-id (name)
--delete <job-id>	Delete a saved job.
--exec <job-id>	Given a job defined with --create, run the saved job.
--show <job-id>	Show the parameters for a saved job.
--list	List all saved jobs

lastmodified mode:

Before Change:

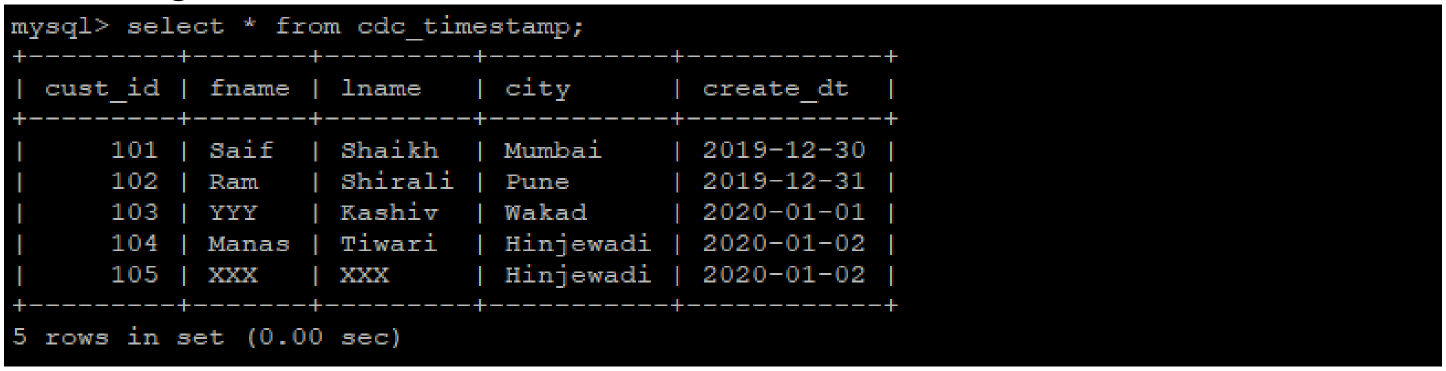


```
20/01/02 22:08:24 INFO tool.ImportTool:  --incremental lastmodified
20/01/02 22:08:24 INFO tool.ImportTool:   --check-column create_dt
20/01/02 22:08:24 INFO tool.ImportTool:   --last-value 2020-01-02 22:07:57.0
20/01/02 22:08:24 INFO tool.ImportTool: (Consider saving this with 'sqoop job --create')
```



```
mysql> select * from cdc_timestamp;
+---------+--------+---------+-----------+------------+
| cust_id | fname  | lname   | city      | create_dt  |
+---------+--------+---------+-----------+------------+
|     101 | Saif   | Shaikh  | Mumbai    | 2019-12-30 |
|     102 | Ram    | Shirali | Pune      | 2019-12-31 |
|     103 | Mitali | Kashiv  | Wakad     | 2020-01-01 |
|     104 | Manas  | Tiwari  | Hinjewadi | 2020-01-02 |
+---------+--------+---------+-----------+------------+
4 rows in set (0.00 sec)

mysql> insert into cdc_timestamp values (105,'XXX','XXX','Hinjewadi',now());
Query OK, 1 row affected, 1 warning (0.00 sec)

mysql> update cdc_timestamp set fname='YYY' where cust_id=103;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

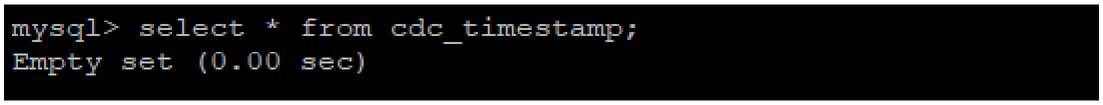
After Change:



```
mysql> select * from cdc_timestamp;
+---------+-------+---------+-----------+------------+
| cust_id | fname | lname   | city      | create_dt  |
+---------+-------+---------+-----------+------------+
|     101 | Saif  | Shaikh  | Mumbai    | 2019-12-30 |
|     102 | Ram   | Shirali | Pune      | 2019-12-31 |
|     103 | YYY   | Kashiv  | Wakad     | 2020-01-01 |
|     104 | Manas | Tiwari  | Hinjewadi | 2020-01-02 |
|     105 | XXX   | XXX     | Hinjewadi | 2020-01-02 |
+---------+-------+---------+-----------+------------+
5 rows in set (0.00 sec)
```

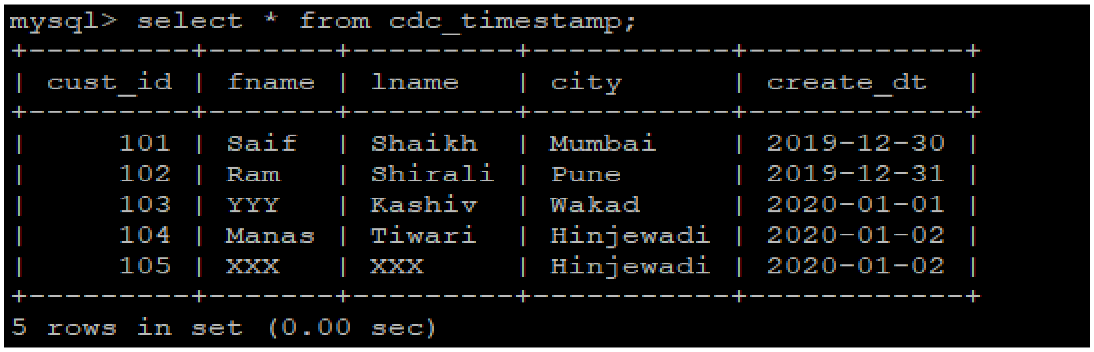
24) Sqoop Export:

Before:



```
mysql> select * from cdc_timestamp;
Empty set (0.00 sec)
```

After:

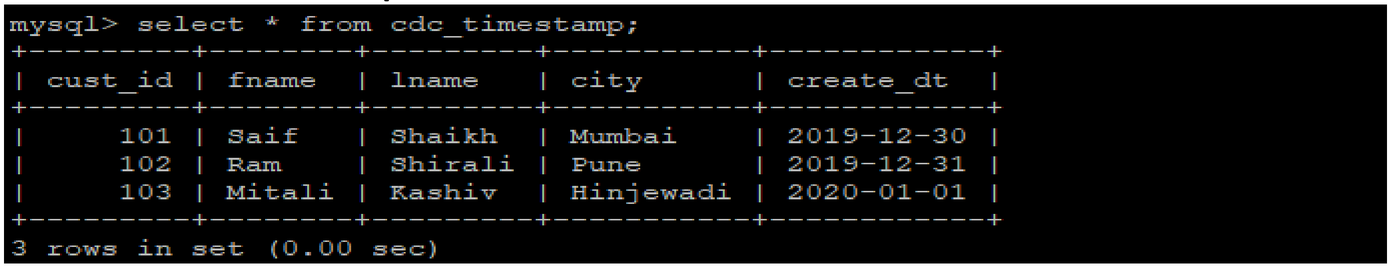


```
mysql> select * from cdc_timestamp;
+---------+-------+---------+-----------+------------+
| cust_id | fname | lname   | city      | create_dt  |
+---------+-------+---------+-----------+------------+
|     101 | Saif  | Shaikh  | Mumbai    | 2019-12-30 |
|     102 | Ram   | Shirali | Pune      | 2019-12-31 |
|     103 | YYY   | Kashiv  | Wakad     | 2020-01-01 |
|     104 | Manas | Tiwari  | Hinjewadi | 2020-01-02 |
|     105 | XXX   | XXX     | Hinjewadi | 2020-01-02 |
+---------+-------+---------+-----------+------------+
5 rows in set (0.00 sec)
```

Sqoop –null-string & --null-non-string:

Argument	Description
--null-string <null-string>	The string to be interpreted as null for string columns.
--null-non-string <null-string>	The string to be interpreted as null for non-string columns.

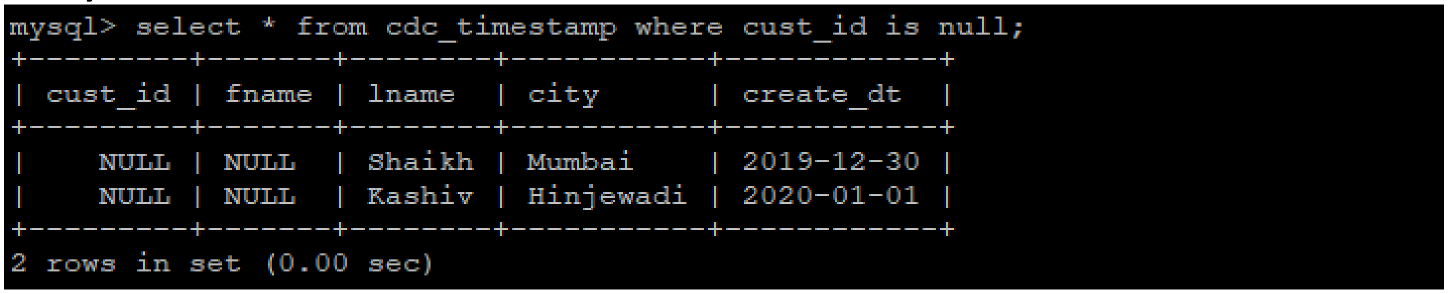
Table Preview before import:



```
mysql> select * from cdc_timestamp;
+---------+--------+---------+-----------+------------+
| cust_id | fname  | lname   | city      | create_dt  |
+---------+--------+---------+-----------+------------+
|     101 | Saif   | Shaikh  | Mumbai    | 2019-12-30 |
|     102 | Ram    | Shirali | Pune      | 2019-12-31 |
|     103 | Mitali | Kashiv  | Hinjewadi | 2020-01-01 |
+---------+--------+---------+-----------+------------+
3 rows in set (0.00 sec)
```

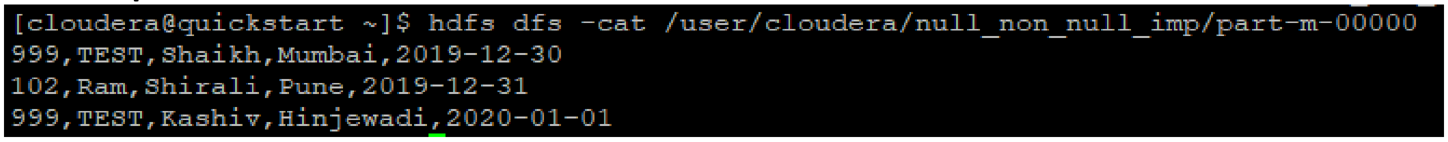
Update columns with NULL values:

Query table data for NULL Values:



```
mysql> select * from cdc_timestamp where cust_id is null;
+---------+-------+--------+-----------+------------+
| cust_id | fname | lname  | city      | create_dt  |
+---------+-------+--------+-----------+------------+
|    NULL | NULL  | Shaikh | Mumbai    | 2019-12-30 |
|    NULL | NULL  | Kashiv | Hinjewadi | 2020-01-01 |
+---------+-------+--------+-----------+------------+
2 rows in set (0.00 sec)
```

After import:

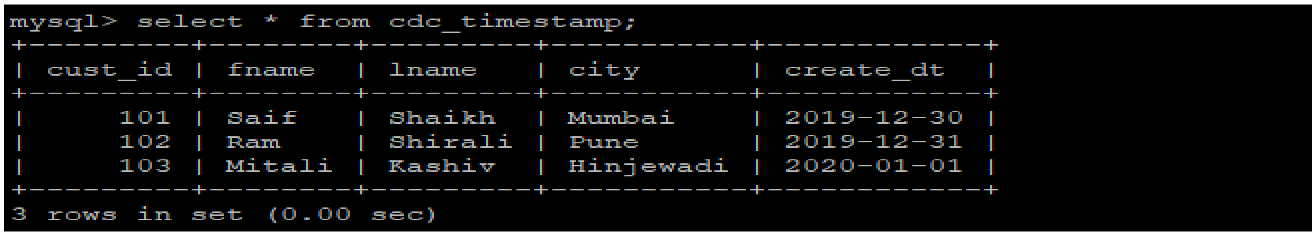


```
[cloudera@quickstart ~]$ hdfs dfs -cat /user/cloudera/null_non_null_imp/part-m-00000
999,TEST,Shaikh,Mumbai,2019-12-30
102,Ram,Shirali,Pune,2019-12-31
999,TEST,Kashiv,Hinjewadi,2020-01-01
```

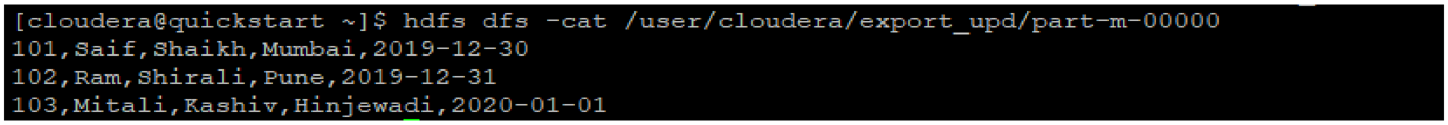
25) Sqoop --map-column-java:

--map-column-java <mapping>: Override mapping from SQL to Java type for configured columns.

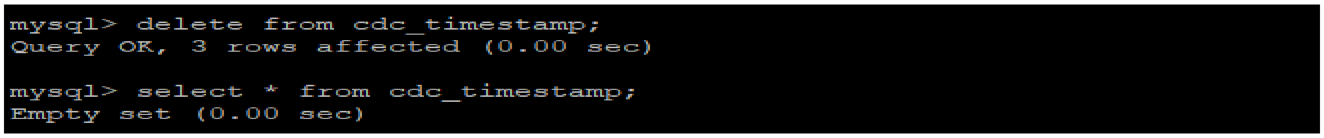
26) Sqoop export updateonly: --update-mode updateonly --update-key <column>



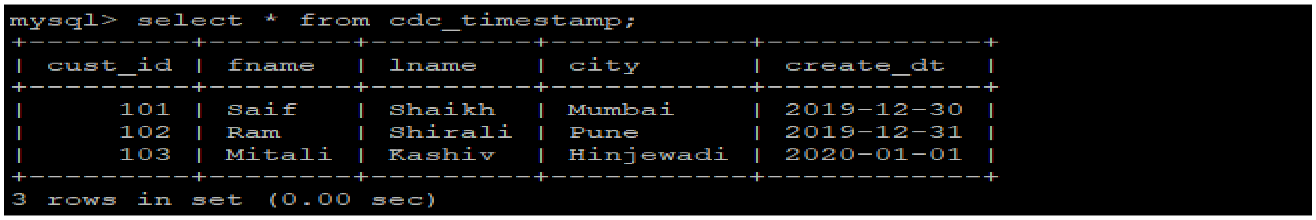
```
mysql> select * from cdc_timestamp;
+---------+--------+---------+-----------+------------+
| cust_id | fname  | lname   | city      | create_dt  |
+---------+--------+---------+-----------+------------+
|     101 | Saif   | Shaikh  | Mumbai    | 2019-12-30 |
|     102 | Ram    | Shirali | Pune      | 2019-12-31 |
|     103 | Mitali | Kashiv  | Hinjewadi | 2020-01-01 |
+---------+--------+---------+-----------+------------+
3 rows in set (0.00 sec)
```



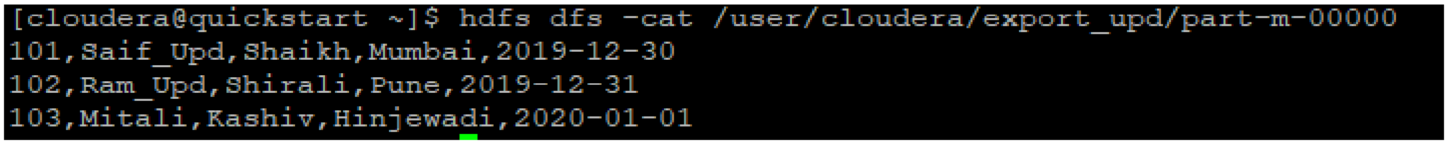
```
[cloudera@quickstart ~]$ hdfs dfs -cat /user/cloudera/export_upd/part-m-00000
101,Saif,Shaikh,Mumbai,2019-12-30
102,Ram,Shirali,Pune,2019-12-31
103,Mitali,Kashiv,Hinjewadi,2020-01-01
```



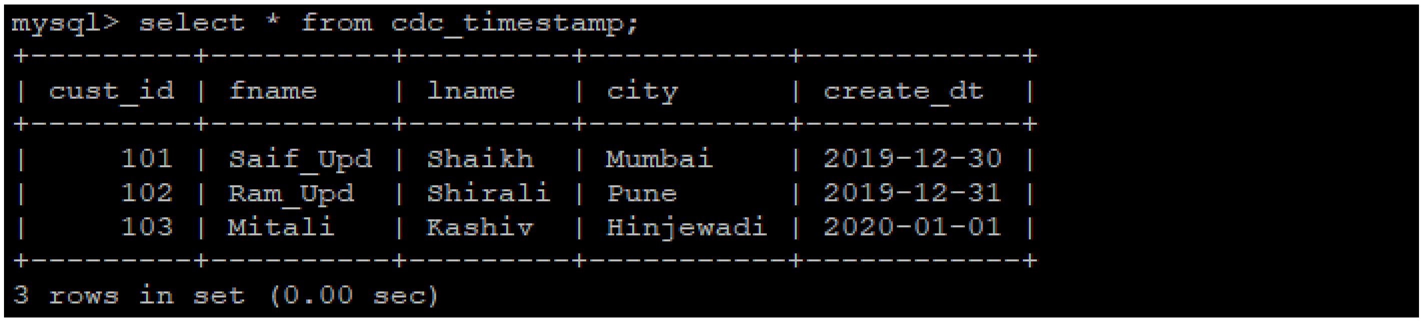
```
mysql> delete from cdc_timestamp;
Query OK, 3 rows affected (0.00 sec)

mysql> select * from cdc_timestamp;
Empty set (0.00 sec)
```



```
mysql> select * from cdc_timestamp;
+---------+--------+---------+-----------+------------+
| cust_id | fname  | lname   | city      | create_dt  |
+---------+--------+---------+-----------+------------+
|     101 | Saif   | Shaikh  | Mumbai    | 2019-12-30 |
|     102 | Ram    | Shirali | Pune      | 2019-12-31 |
|     103 | Mitali | Kashiv  | Hinjewadi | 2020-01-01 |
+---------+--------+---------+-----------+------------+
3 rows in set (0.00 sec)
```

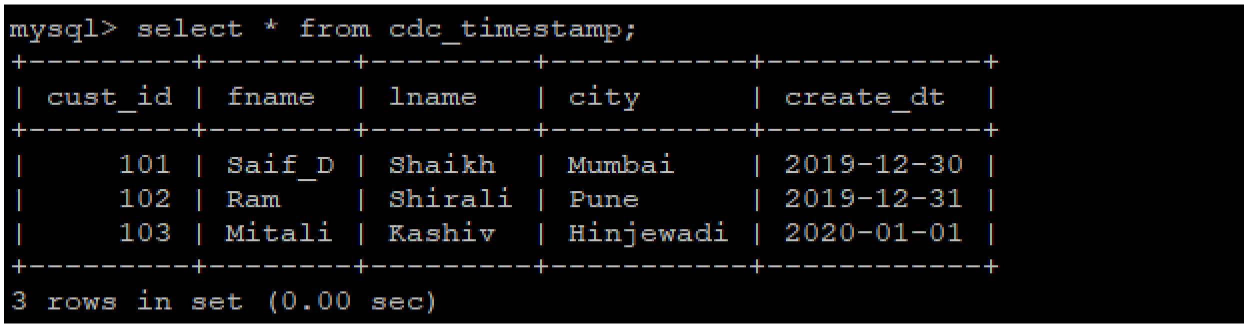


```
[cloudera@quickstart ~]$ hdfs dfs -cat /user/cloudera/export_upd/part-m-00000
101,Saif_Upd,Shaikh,Mumbai,2019-12-30
102,Ram_Upd,Shirali,Pune,2019-12-31
103,Mitali,Kashiv,Hinjewadi,2020-01-01
```

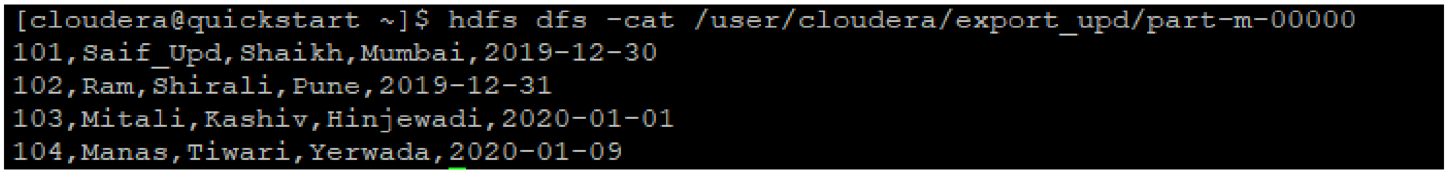


```
mysql> select * from cdc_timestamp;
+---------+----------+---------+-----------+------------+
| cust_id | fname    | lname   | city      | create_dt  |
+---------+----------+---------+-----------+------------+
|     101 | Saif_Upd | Shaikh  | Mumbai    | 2019-12-30 |
|     102 | Ram_Upd  | Shirali | Pune      | 2019-12-31 |
|     103 | Mitali   | Kashiv  | Hinjewadi | 2020-01-01 |
+---------+----------+---------+-----------+------------+
3 rows in set (0.00 sec)
```

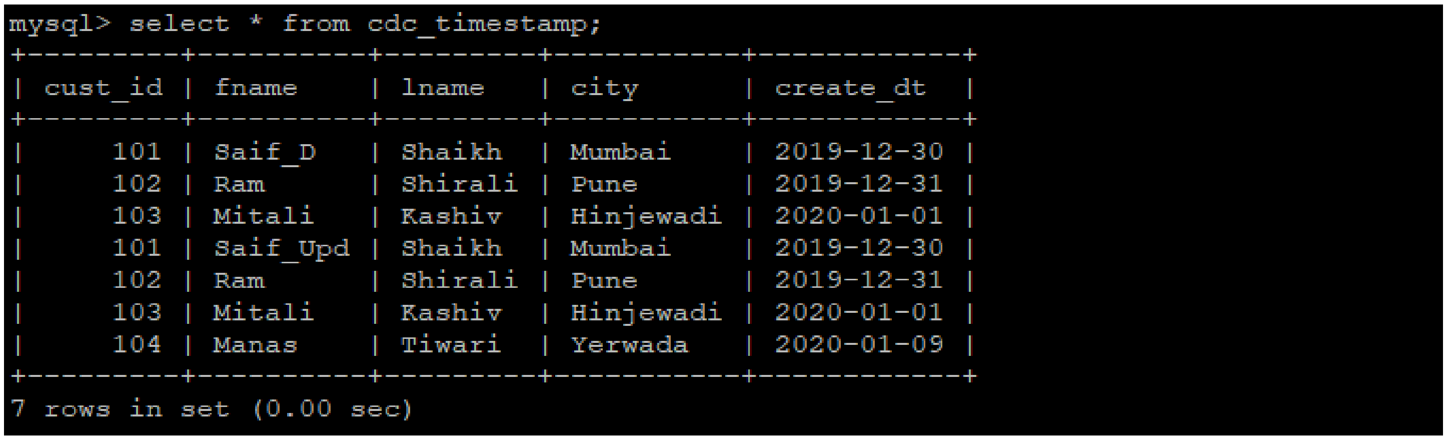
27) Sqoop export allowinsert: --update-mode allowinsert --update-key <column>



```
mysql> select * from cdc_timestamp;
+---------+--------+---------+-----------+------------+
| cust_id | fname  | lname   | city      | create_dt  |
+---------+--------+---------+-----------+------------+
|     101 | Saif_D | Shaikh  | Mumbai    | 2019-12-30 |
|     102 | Ram    | Shirali | Pune      | 2019-12-31 |
|     103 | Mitali | Kashiv  | Hinjewadi | 2020-01-01 |
+---------+--------+---------+-----------+------------+
3 rows in set (0.00 sec)
```



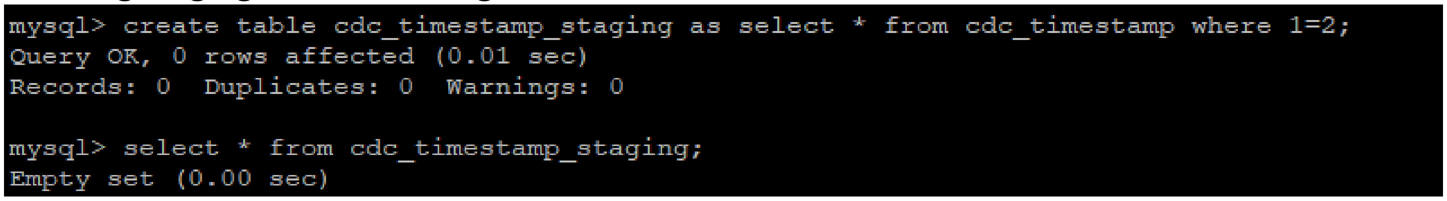
```
[cloudera@quickstart ~]$ hdfs dfs -cat /user/cloudera/export_upd/part-m-00000
101,Saif_Upd,Shaikh,Mumbai,2019-12-30
102,Ram,Shirali,Pune,2019-12-31
103,Mitali,Kashiv,Hinjewadi,2020-01-01
104,Manas,Tiwari,Yerwada,2020-01-09
```



```
mysql> select * from cdc_timestamp;
+---------+----------+---------+-----------+------------+
| cust_id | fname    | lname   | city      | create_dt  |
+---------+----------+---------+-----------+------------+
|     101 | Saif_D   | Shaikh  | Mumbai    | 2019-12-30 |
|     102 | Ram      | Shirali | Pune      | 2019-12-31 |
|     103 | Mitali   | Kashiv  | Hinjewadi | 2020-01-01 |
|     101 | Saif_Upd | Shaikh  | Mumbai    | 2019-12-30 |
|     102 | Ram      | Shirali | Pune      | 2019-12-31 |
|     103 | Mitali   | Kashiv  | Hinjewadi | 2020-01-01 |
|     104 | Manas    | Tiwari  | Yerwada   | 2020-01-09 |
+---------+----------+---------+-----------+------------+
7 rows in set (0.00 sec)
```

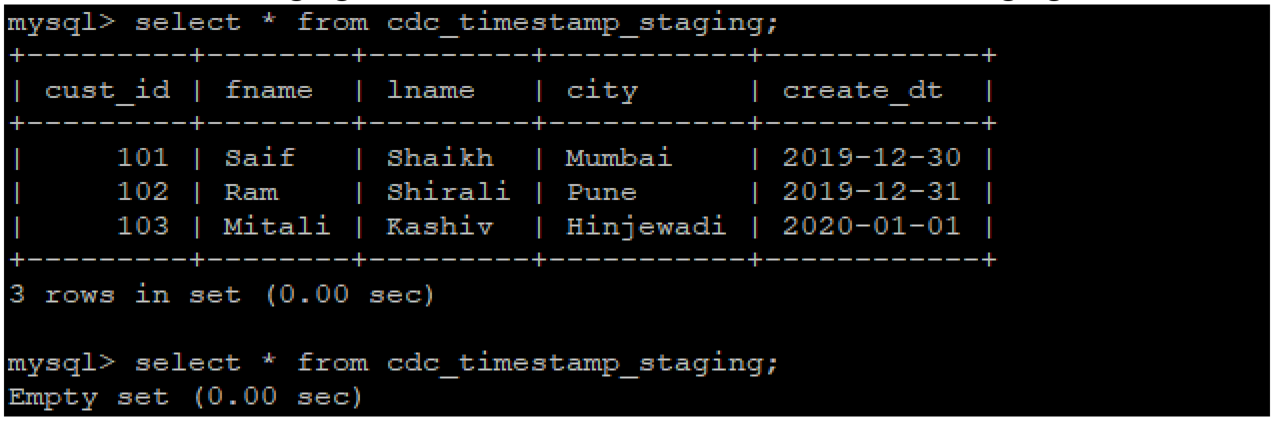
28) Sqoop Staging tables: --staging-table <table_name>

Creating staging table from original table:



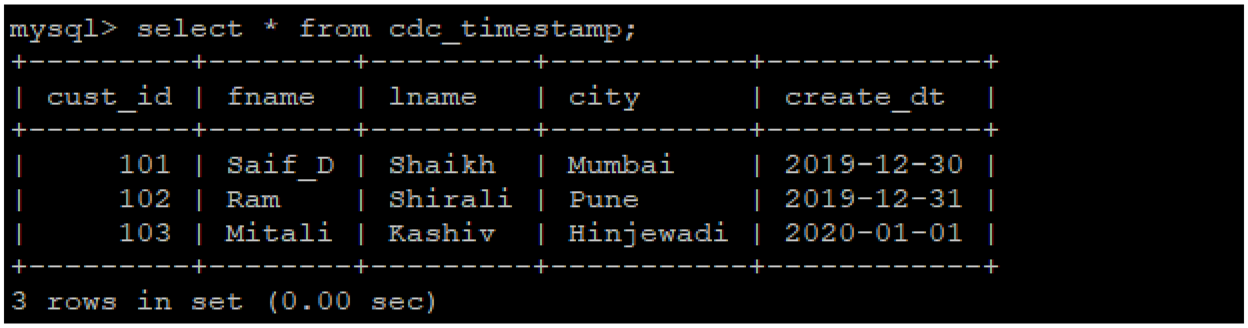
```
mysql> create table cdc_timestamp_staging as select * from cdc_timestamp where 1=2;
Query OK, 0 rows affected (0.01 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> select * from cdc_timestamp_staging;
Empty set (0.00 sec)
```

First the data is truncated from staging table, then the data is loaded to staging table. After that from staging table data is loaded to main table then staging table is truncated.



```
mysql> select * from cdc_timestamp_staging;
+---------+--------+---------+-----------+------------+
| cust_id | fname  | lname   | city      | create_dt  |
+---------+--------+---------+-----------+------------+
|     101 | Saif   | Shaikh  | Mumbai    | 2019-12-30 |
|     102 | Ram    | Shirali | Pune      | 2019-12-31 |
|     103 | Mitali | Kashiv  | Hinjewadi | 2020-01-01 |
+---------+--------+---------+-----------+------------+
3 rows in set (0.00 sec)

mysql> select * from cdc_timestamp_staging;
Empty set (0.00 sec)
```

Main table data:



```
mysql> select * from cdc_timestamp;
+---------+--------+---------+-----------+------------+
| cust_id | fname  | lname   | city      | create_dt  |
+---------+--------+---------+-----------+------------+
|     101 | Saif_D | Shaikh  | Mumbai    | 2019-12-30 |
|     102 | Ram    | Shirali | Pune      | 2019-12-31 |
|     103 | Mitali | Kashiv  | Hinjewadi | 2020-01-01 |
+---------+--------+---------+-----------+------------+
3 rows in set (0.00 sec)
```