Apache Sqoop Exercise 2

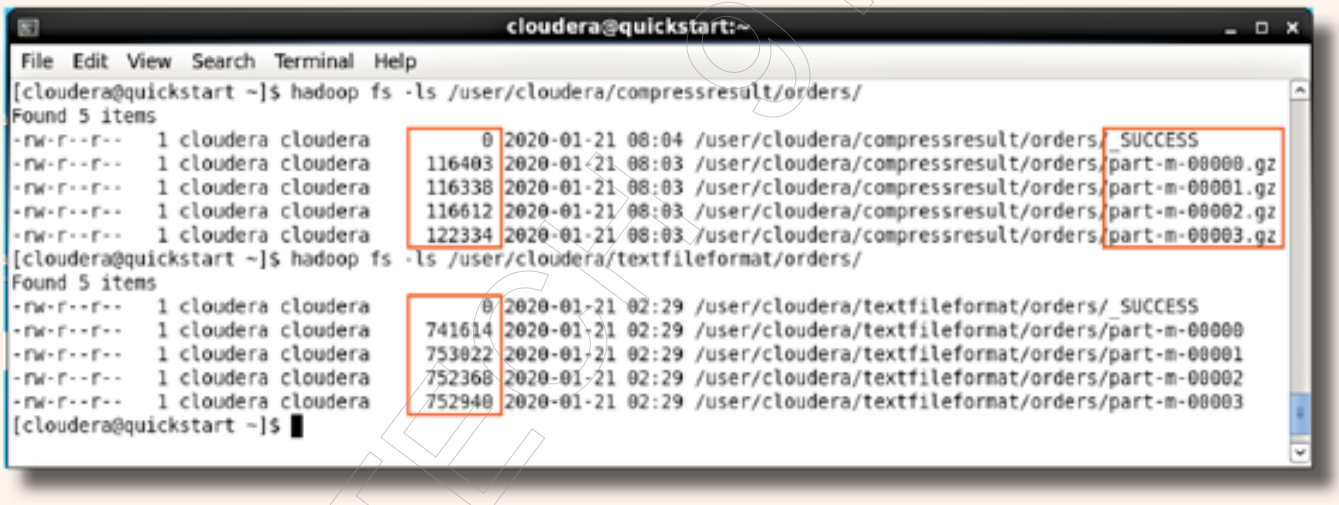
IMPORTANT

Copyright Infringement and Illegal Content Sharing Notice

Compression techniques:

You can compress your data by using the default (**gzip**) algorithm with the `-z` or `--compress` argument.

```
sqoop-import \
--connect jdbc:mysql://quickstart.cloudera:3306/retail_db \
--username retail_dba \
--password cloudera \
--table orders \
--compress \
--warehouse-dir /user/cloudera/compressresult
```



Note: we will see **.gz** extension)

We can specify any Hadoop compression codec using the `--compression-codec` argument:

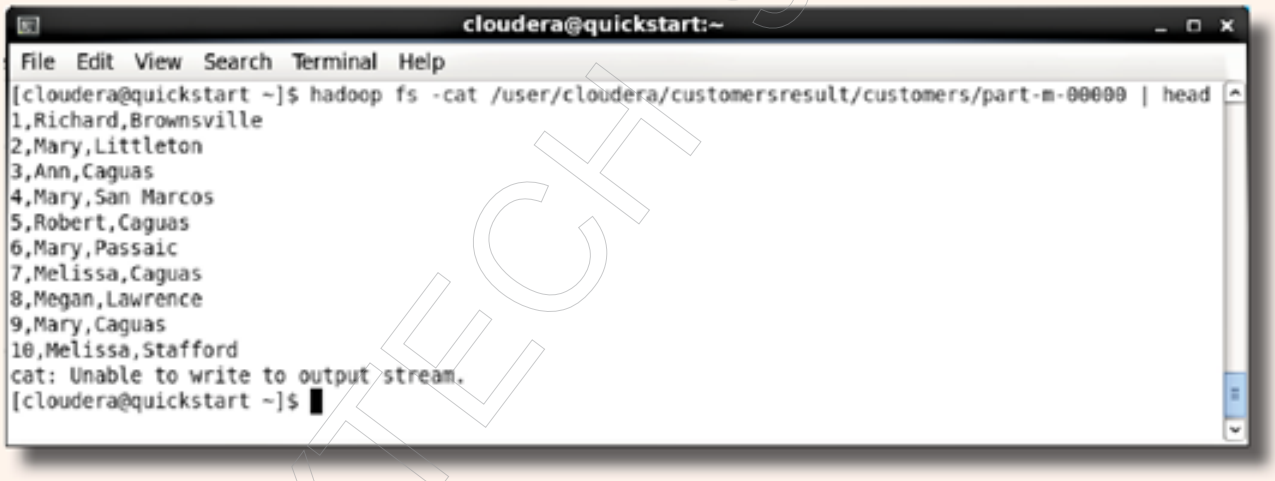
```
sqoop-import \
--connect jdbc:mysql://quickstart.cloudera:3306/retail_db \
--username retail_dba --password cloudera \
--table orders \
--compression-codec BZip2Codec \
--warehouse-dir /user/cloudera/bzipcompresult
```

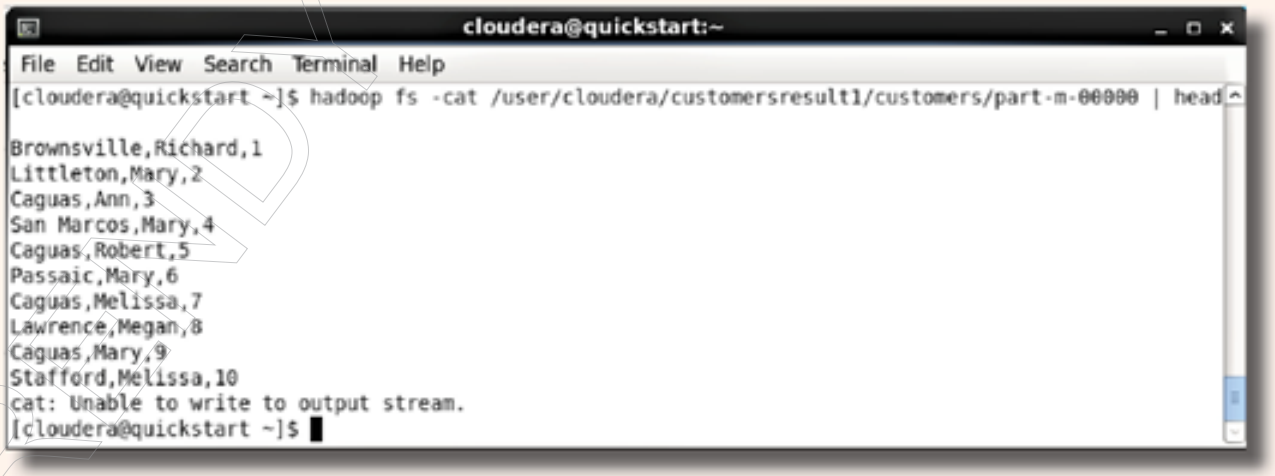
Import the data with selected columns:

We can select a subset of columns and control their ordering by using the `--columns` argument.

For example: `--columns customer_id,customer_fname`

```
sqoop-import \
--connect jdbc:mysql://quickstart.cloudera:3306/retail_db \
--username retail_dba \
--password cloudera \
--table customers \
--columns customer_id,customer_fname,customer_city \
--warehouse-dir /user/cloudera/customersresult
```





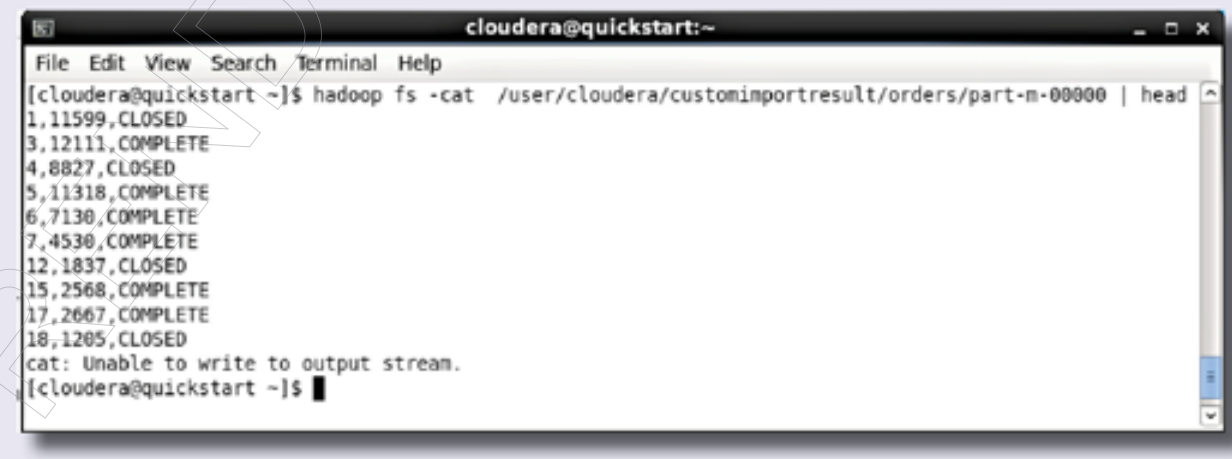
import with WHERE clause:

You can control which rows are imported by adding a SQL **WHERE** clause to the import statement.

```
sqoop-import \
--connect jdbc:mysql://quickstart.cloudera:3306/retail_db \
--username retail_dba \
--password cloudera \
--table orders \
--columns order_id,order_customer_id,order_status \
--where "order_status in ('complete','closed')" \
--warehouse-dir /user/cloudera/customimportresult
```

Chek the output of the imported customimportresult file:

```
hadoop fs -cat /user/cloudera/customimportresult/orders/part-m-00000
```



Sqoop import execution flow

- Get metadata of the table by running simple query.
- Build the POJO class with appropriate getters and setters.
- Compile the POJO class into jar file
- Run boundary vals query or boundary query to get min and max by split column (default is primary key column).
- Compute split size max - min.
- Divide it with number of mappers and compute splits.
- Submit map reduce job with number of mappers equal to 4 by default.
- Each map task will run select query on the source table with where condition based on the splits to read the data.
- Data will be written to the files in the location specified.

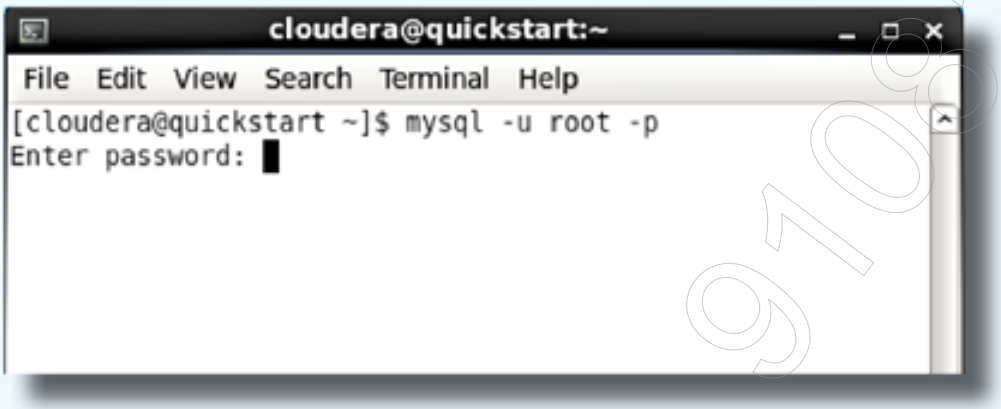
Sqoop boundary query

- Whenever we run sqoop import with number of mappers greater than 1, a **bounding vals query** will run.
- **Bounding Vals Query** is run on primary key field to get min and max value of it.
- Using the min and max split size is computed.
- We can customize the **bounding vals query** by using `--boundary-query`.
- Query should return 2 values as one row. These values will be used to compute the split size.
- We can hardcode the min and max values with an intention to remove outliers.

Insert one record in Mysql - orders table:

Login to MySQL:

```
mysql -u root -p
```



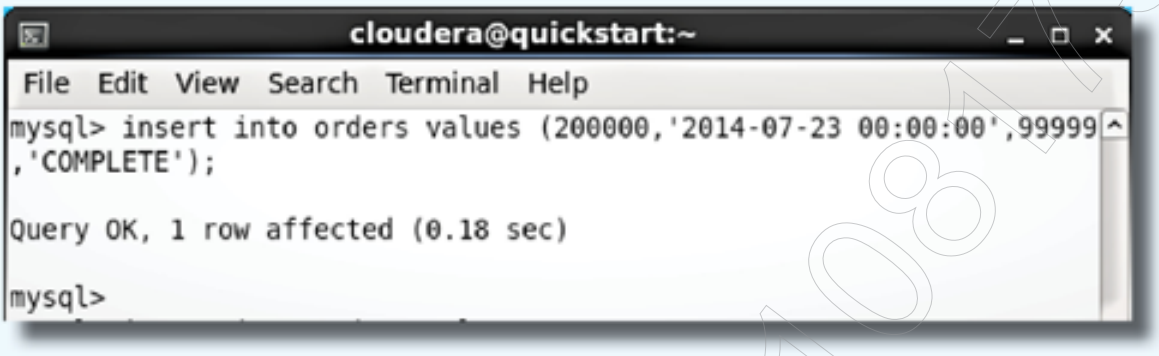

```
Enter password: cloudera
```

Use Database *retail_db*

```
use retail_db;
```

Insert one record in the existing *orders* table:

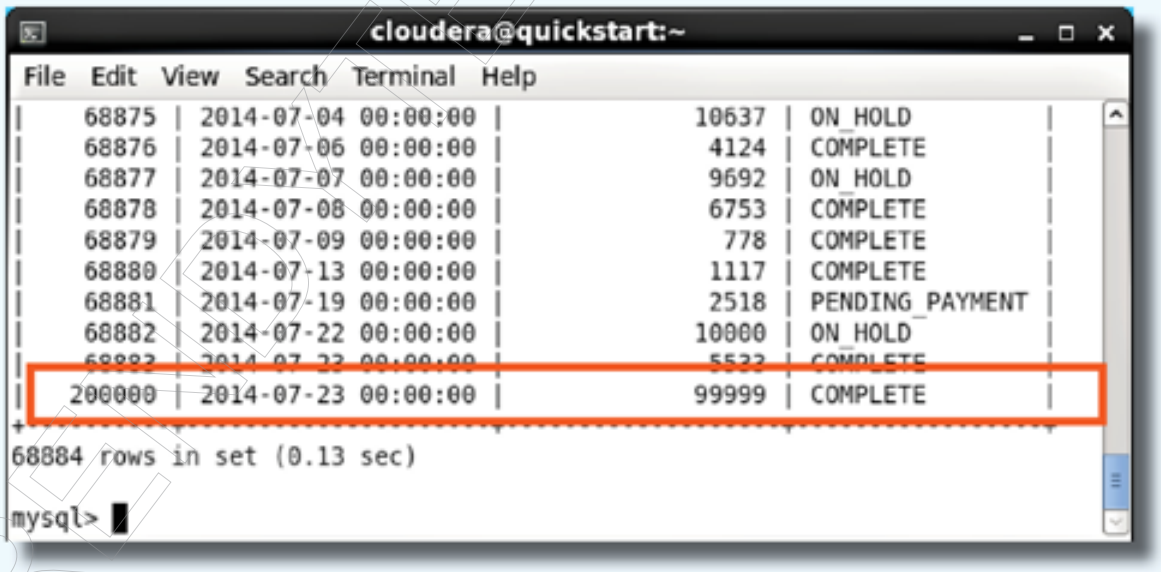
```
insert into orders values
(200000,'2014-07-23 00:00:00',99999,'COMPLETE');
```




```
commit;
```

Now check the *orders* table to confirm the insert record:

```
select * from orders;
```

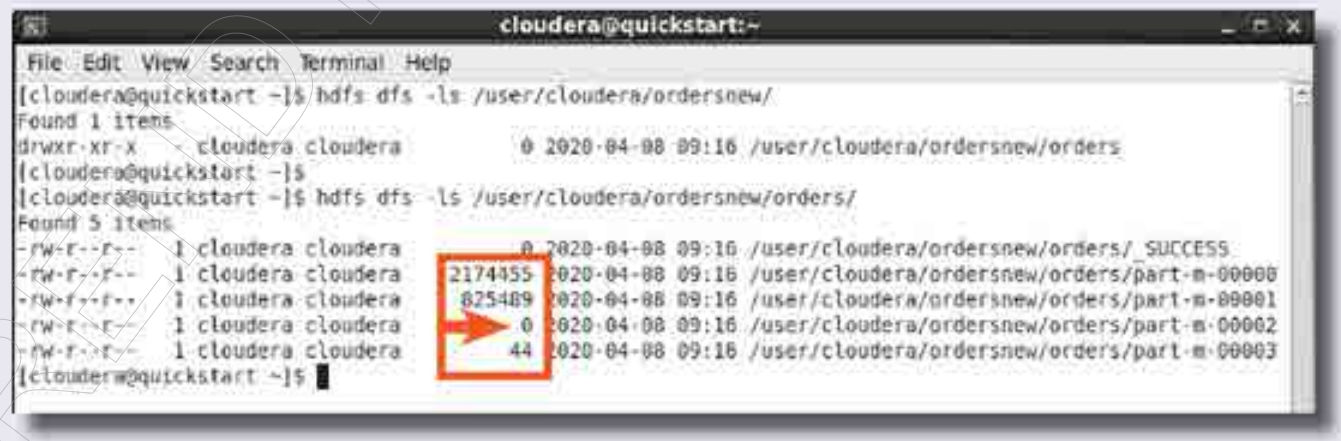


Now, import the updated *orders* table using Sqoop:

```
sqoop-import \
--connect jdbc:mysql://quickstart.cloudera:3306/retail_db \
--username retail_dba \
--password cloudera \
--table orders \
--warehouse-dir /user/cloudera/ordersnew
```

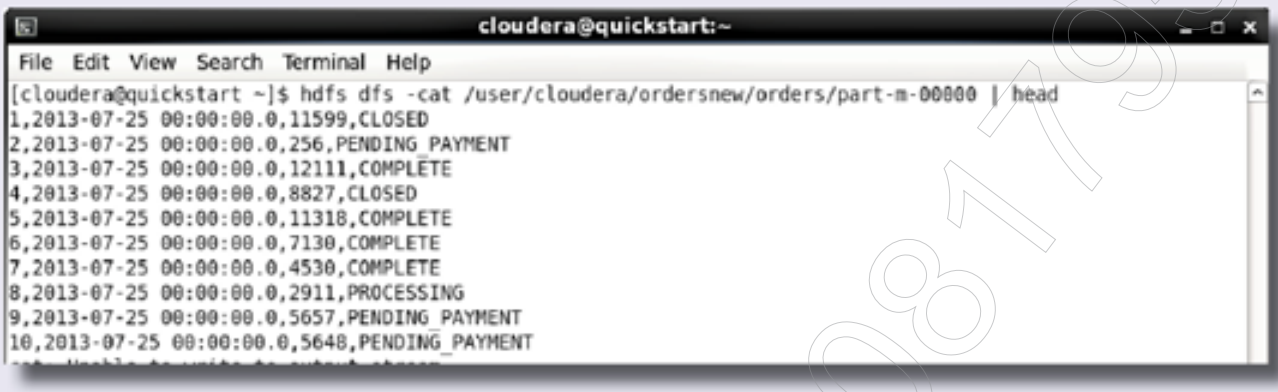
Check the output in hdfs:

```
hdfs dfs -ls /user/cloudera/ordersnew/orders/
```



Check the contents of mapper files:

```
hdfs dfs -cat /user/cloudera/ordersnew/orders/part-m-00000
```




```
hdfs dfs -cat /user/cloudera/ordersnew/orders/part-m-00001
```

```
[cloudera@quickstart ~]$ hdfs dfs -cat /user/cloudera/ordersnew/orders/part-m-00001 | head
50001,2014-06-02 00:00:00.0,10731,PENDING_PAYMENT
50002,2014-06-02 00:00:00.0,8037,CANCELED
50003,2014-06-02 00:00:00.0,9410,PENDING
50004,2014-06-02 00:00:00.0,7897,COMPLETE
50005,2014-06-02 00:00:00.0,9216,PENDING_PAYMENT
50006,2014-06-02 00:00:00.0,11020,COMPLETE
50007,2014-06-02 00:00:00.0,1573,PENDING
50008,2014-06-02 00:00:00.0,6296,COMPLETE
50009,2014-06-02 00:00:00.0,5082,PROCESSING
50010,2014-06-02 00:00:00.0,5061,COMPLETE
cat: Unable to write to output stream.
[cloudera@quickstart ~]$
```

```
hdfs dfs -cat /user/cloudera/ordersnew/orders/part-m-00002
```

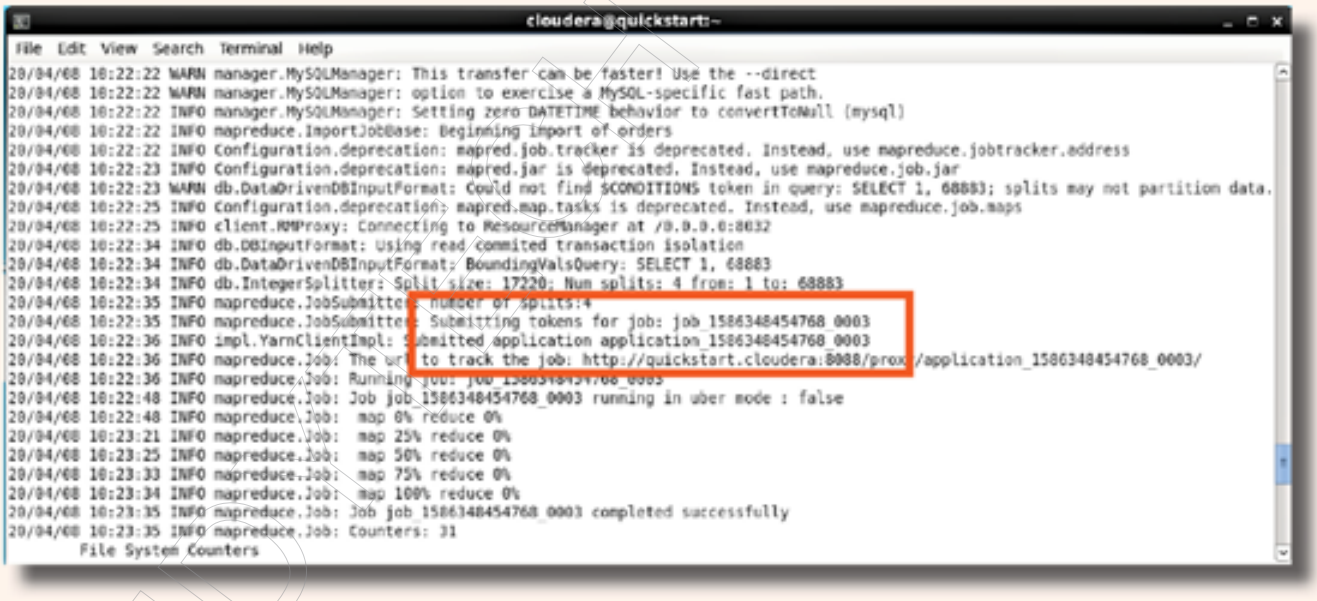

```
hdfs dfs -cat /user/cloudera/ordersnew/orders/part-m-00003
```

```
[cloudera@quickstart ~]$ hdfs dfs -cat /user/cloudera/ordersnew/orders/part-m-00003
200000,2014-07-23 00:00:00.0,99999,COMPLETE
[cloudera@quickstart ~]$
```

To solve the previous problem we need to customize the boundary query:

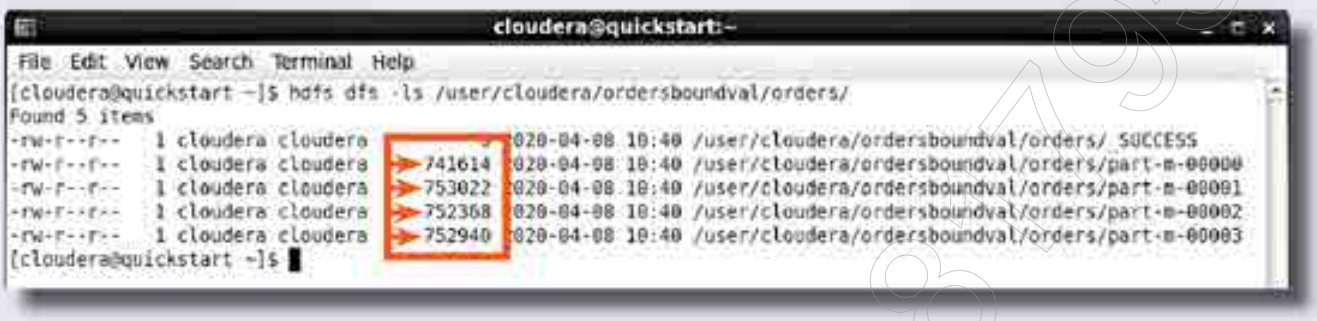
Example of boundary vals query where we can pass hardcoded **min** and **max** values using which splits will be computed.

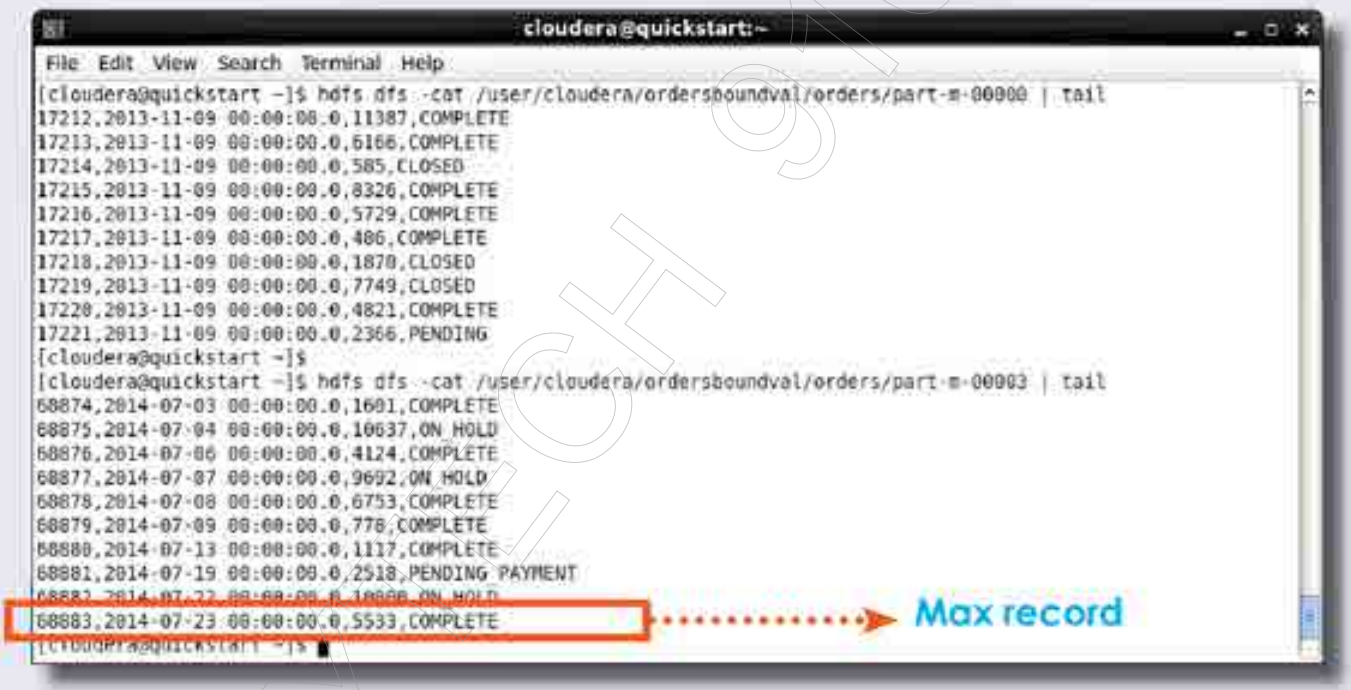
```
sqoop-import \
--connect jdbc:mysql://quickstart.cloudera:3306/retail_db \
--username retail_dba \
--password cloudera \
--table orders \
--boundary-query "SELECT 1, 68883" \
--warehouse-dir /user/cloudera/ordersboundval
```



Check the output in hdfs:

```
hdfs dfs -ls /user/cloudera/ordersboundval/orders/
```

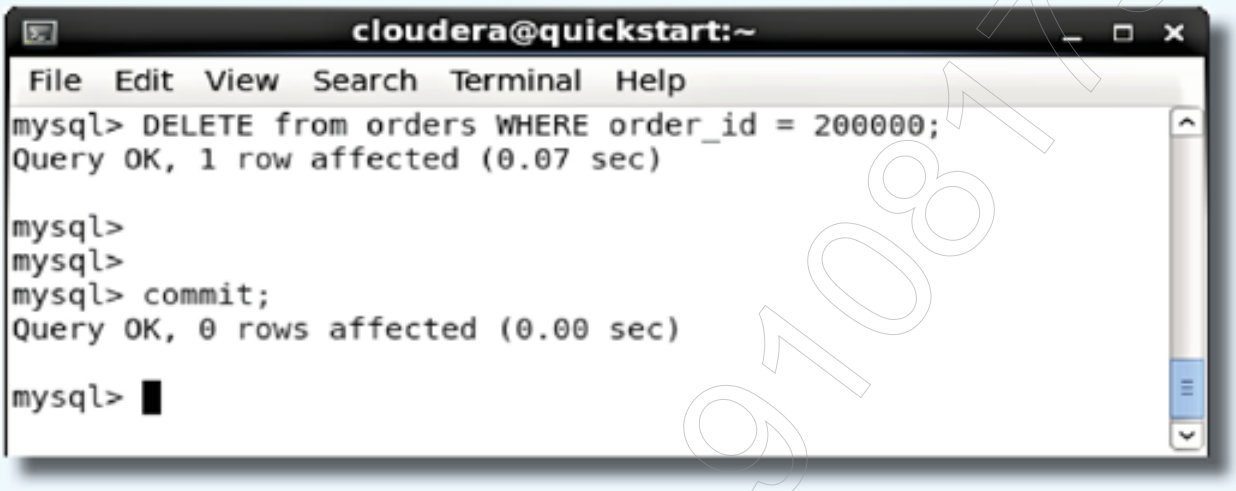




Now we can see with the above command all the mappers did the same amount of work, which is ideally good.

Now, delete the record which we have inserted in *orders* table in Mysql:

```
DELETE from orders WHERE order_id = 200000;
```



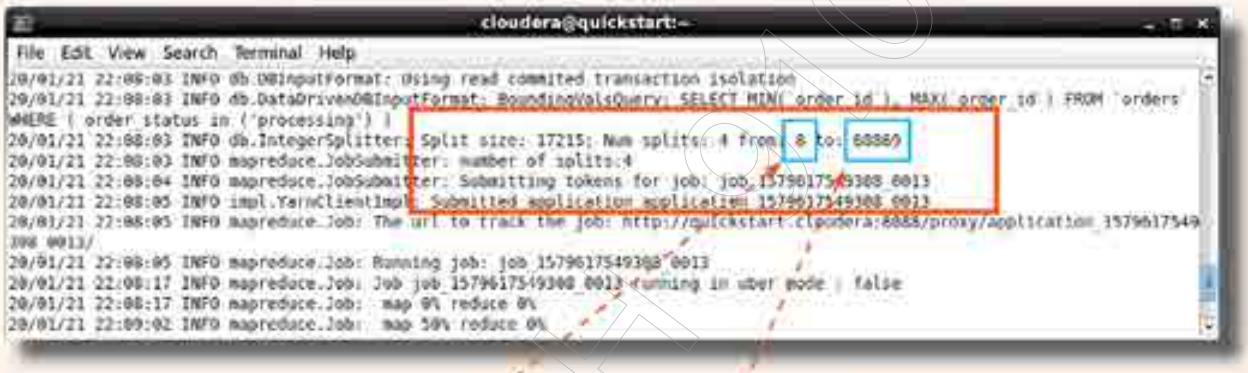

```
commit;
```

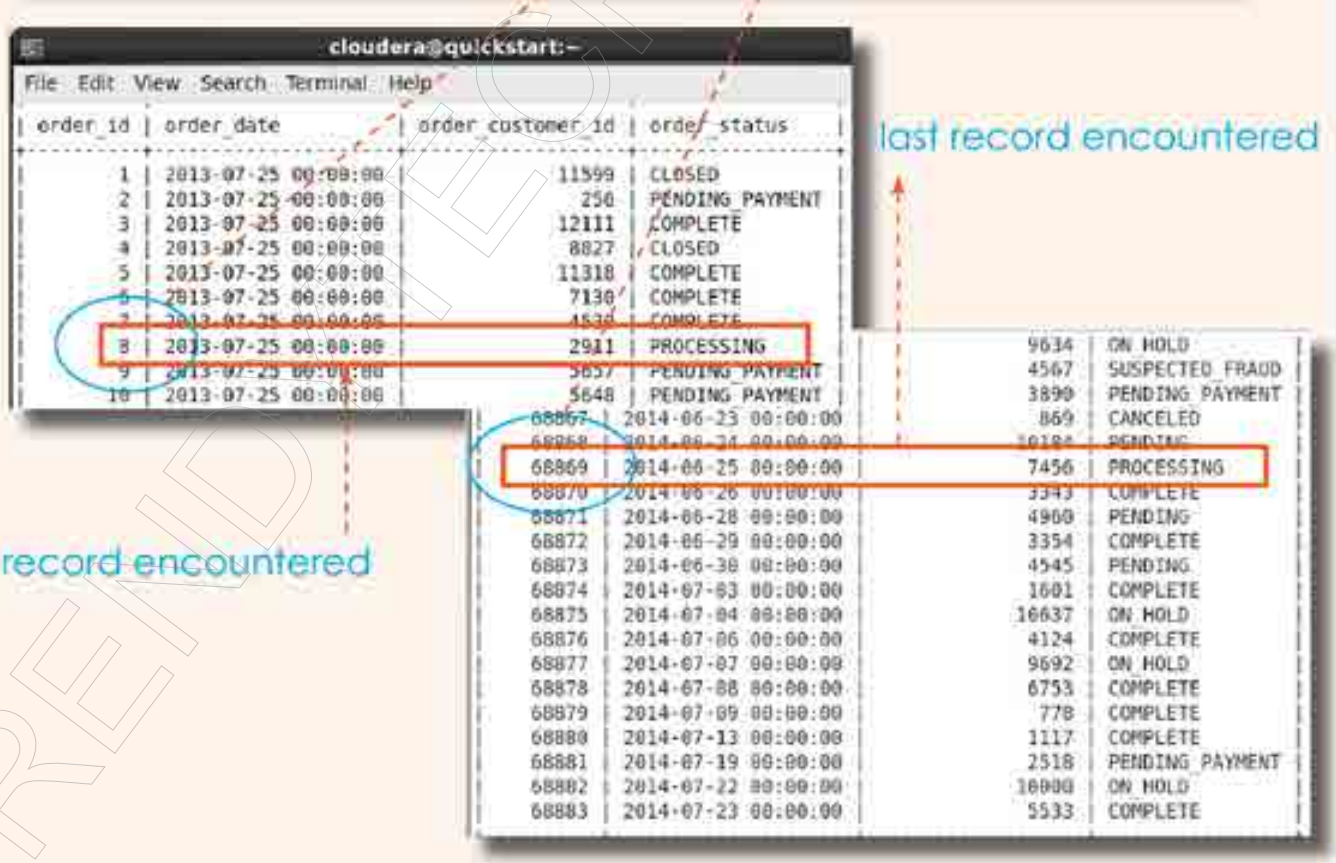
Let us see another example of boundary vals query where we can even customize the min and max values using non primary key column.

```
sqoop-import \
--connect jdbc:mysql://quickstart.cloudera:3306/retail_db \
--username retail_dba \
--password cloudera \
--table order_items \
--boundary-query "SELECT min(order_item_order_id),
max(order_item_order_id) FROM order_items WHERE order_
item_order_id ge 10000" \
--warehouse-dir /user/cloudera/bvqresult
```

WHERE clause also internally treated as BoundingValsQuery:

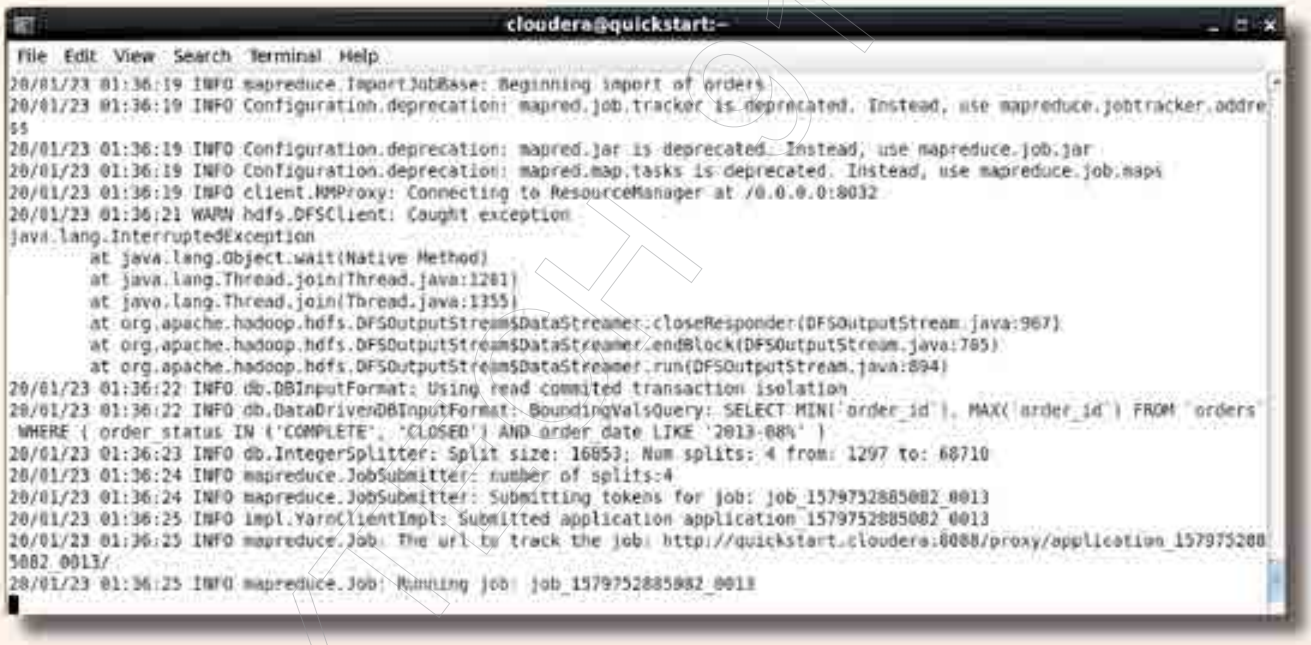
```
sqoop-import \
--connect jdbc:mysql://quickstart.cloudera:3306/retail_db \
--username retail_dba \
--password cloudera \
--table orders \
--columns order_id,order_customer_id,order_status \
--where "order_status in ('processing')" \
--warehouse-dir /user/cloudera/whereclauseresult
```





Example2: WHERE clause converted as BoundingValsQuery:

```
sqoop-import \
--connect jdbc:mysql://quickstart.cloudera:3306/retail_db \
--username retail_dba \
--password cloudera \
--table orders \
--where "order_status IN ('COMPLETE', 'CLOSED') AND
order_date LIKE '2013-08%'" \
--warehouse-dir /user/cloudera/bvqresult
```



Sqoop import using split by

Sqoop `--split-by` comes into picture when there is no primary key or the primary key column is not evenly distributed. When to use split-by:

1. When there is no primary key then we can use split-by to indicate the column based on which mappers should divide the work.

2. We can use split-by when your primary key has lot of outliers, then we can use any other column as split-by column (to get a better performance).

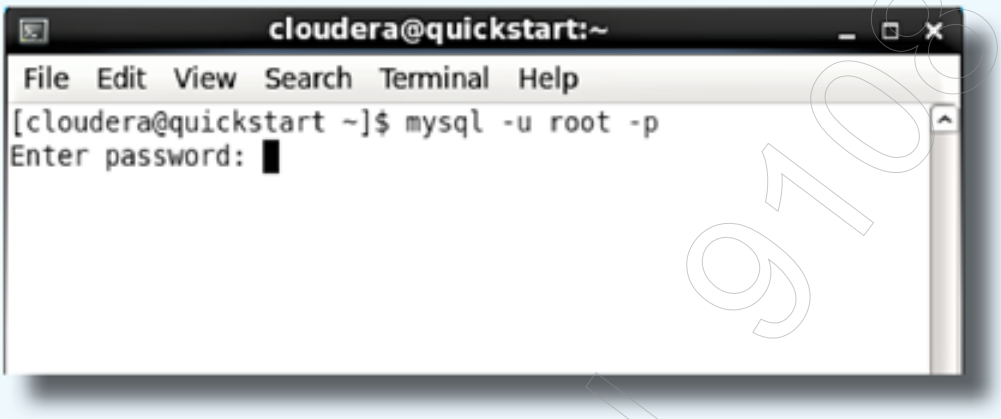
Understanding Sqoop SPLIT-BY

- Let us understand what happens when we run typical sqoop import and when `--split-by` argument should be used.
- Whenever we run sqoop import with number of mappers greater than 1, a bounding vals query will run.
- Bounding Vals Query is run on primary key field to get min and max value of it.
- Using the min and max split size is computed.
- If there is no primary key or unique key, import will fail unless number of mappers is set to 1 or specify a field using split-by.
- It is better practice to use an indexed field which do not contain null values as part of split-by.

Create a table orders_no_pk in MySQL without any primary key:

Login to MySQL:

```
mysql -u root -p
```



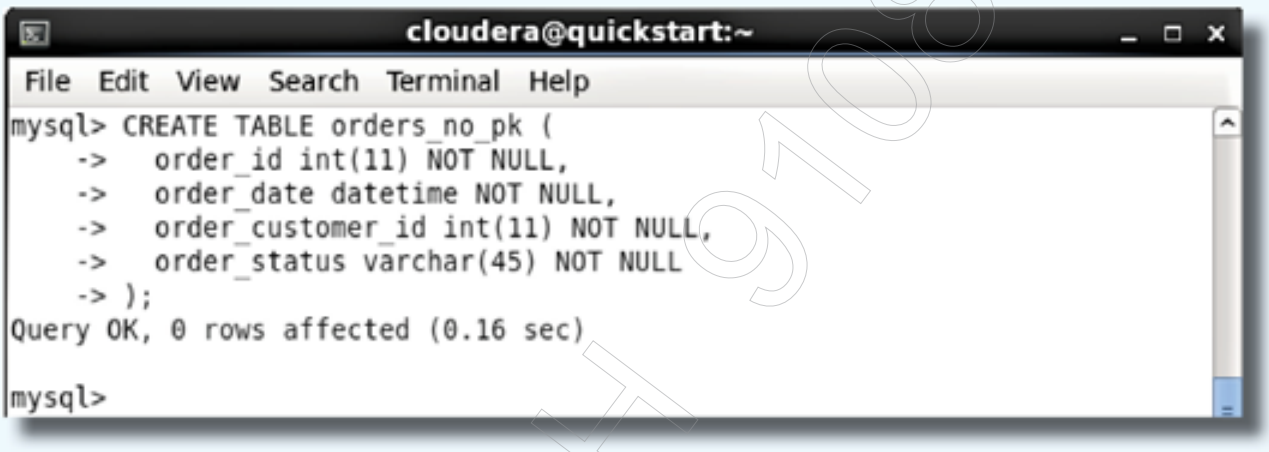

```
Enter password: cloudera
```

Use Database *retail_db*

```
use retail_db;
```

Create a *orders_no_pk* table in *retail_db* database:

```
CREATE TABLE orders_no_pk (
  order_id int(11) NOT NULL,
  order_date datetime NOT NULL,
  order_customer_id int(11) NOT NULL,
  order_status varchar(45) NOT NULL
);
```



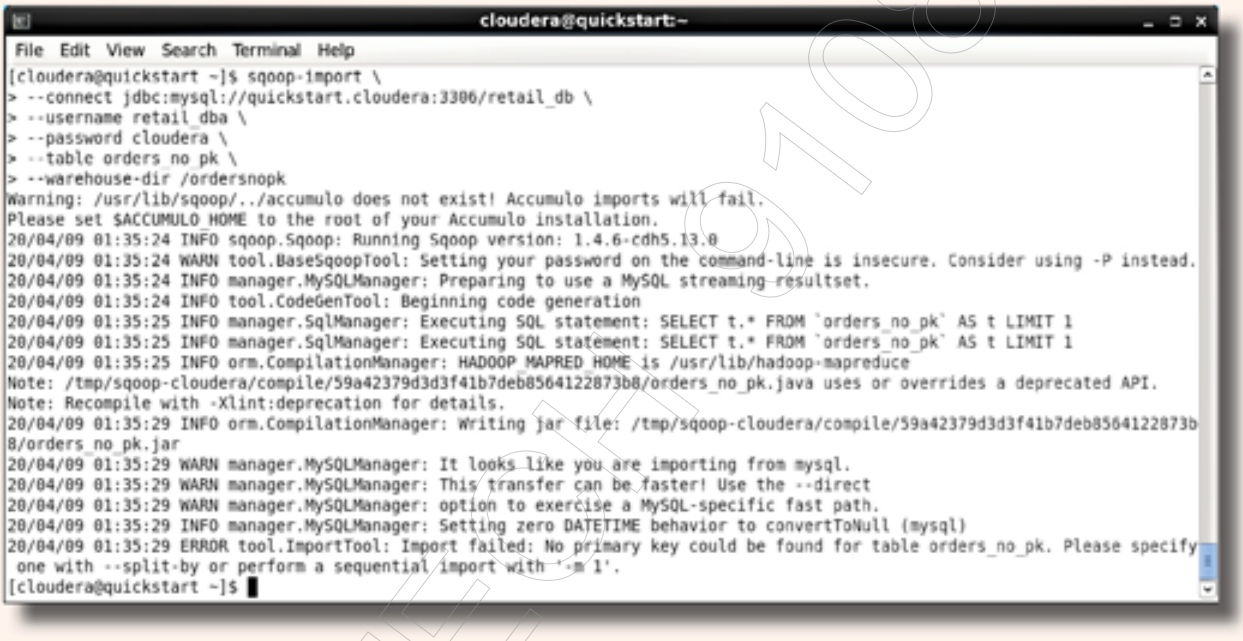
Copying data from *orders* table to *orders_no_pk* table:

```
insert into orders_no_pk select order_id, order_date,
order_customer_id, order_status from orders;
```


```
commit;
```

Now, import the *orders_no_pk* table into HDFS using Sqoop:

```
sqoop-import \
--connect jdbc:mysql://quickstart.cloudera:3306/retail_db \
--username retail_dba \
--password cloudera \
--table orders_no_pk \
--warehouse-dir /ordersnopk
```



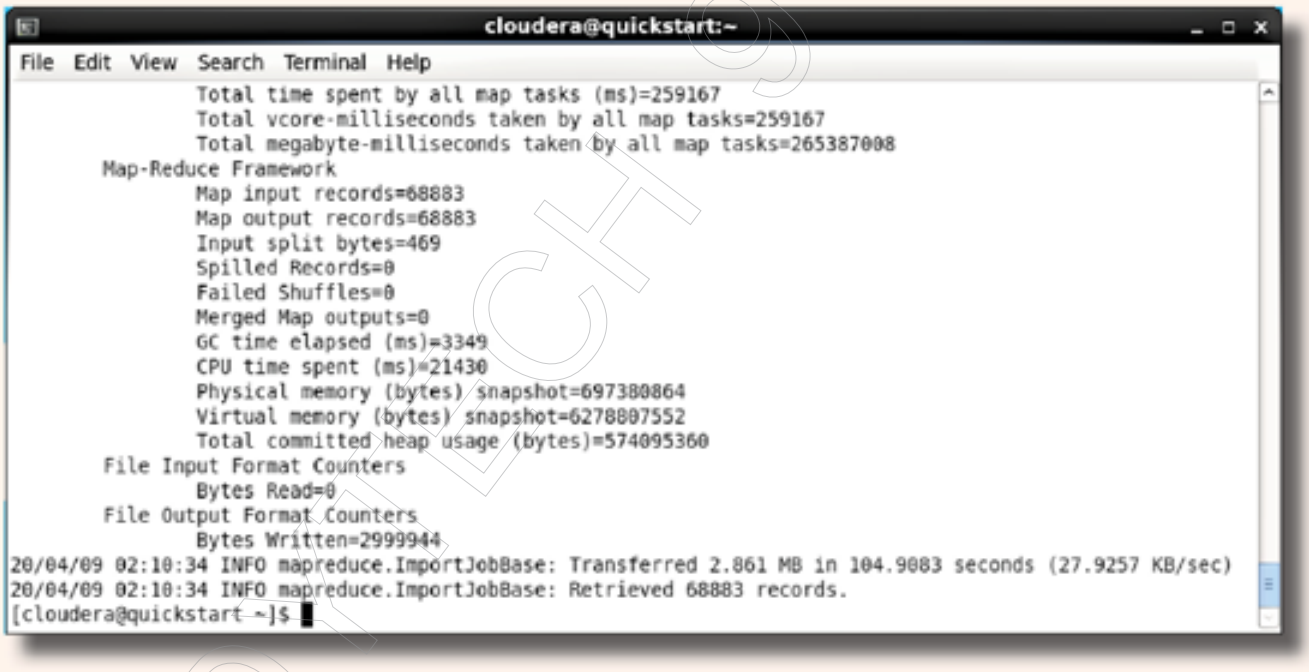
JOB FAILS

Above import fails because the orders_no_pk table doesn't have Primary key and Sqoop doesn't know how to divide records among Mappers.

Import the same *orders_no_pk* table into HDFS using split-by:

When there is no primary key **`--split-by`** is the ideal solution for import control.

```
sqoop-import \
--connect jdbc:mysql://quickstart.cloudera:3306/retail_db \
--username retail_dba \
--password cloudera \
--table orders_no_pk \
--split-by order_id \
--target-dir /ordersnopksplit
```



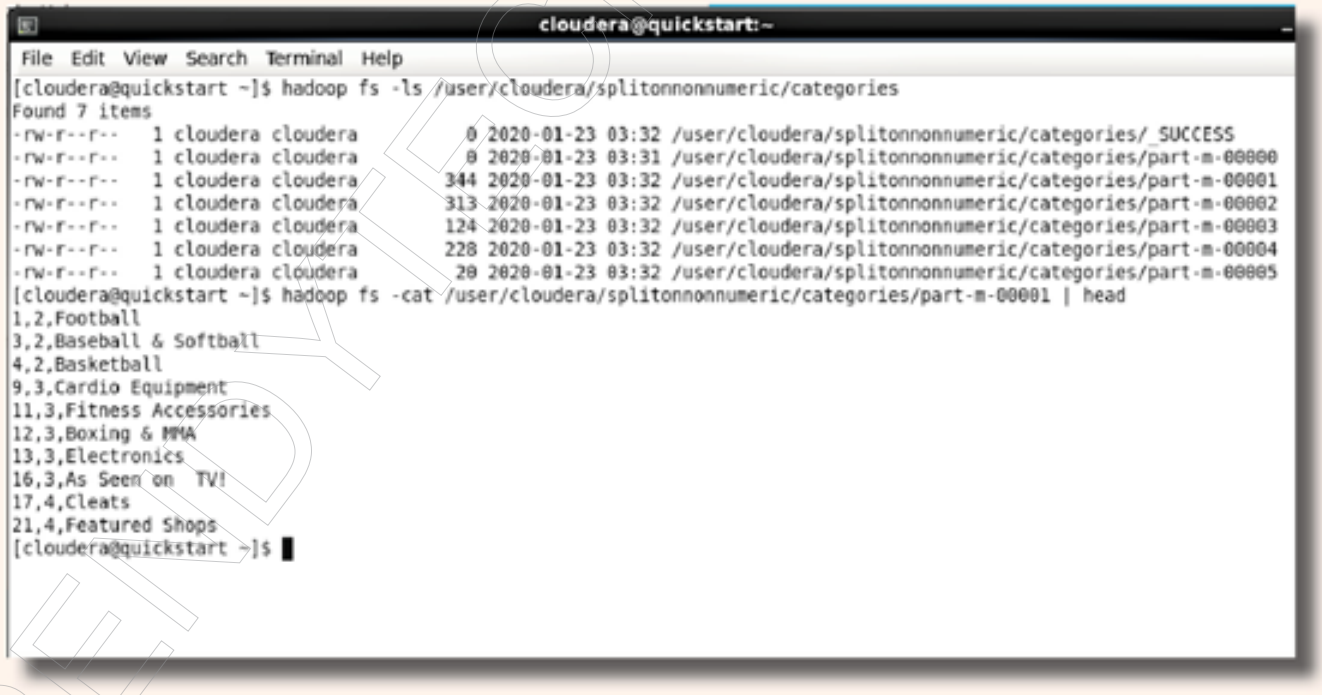
Note: Split column should have numeric values. It is not recommended to use split-by on a text column.

Dealing with SPLIT-BY or primary key on non numeric fields:

sqoop import using non numeric field fail with hint to use `org.apache.sqoop.splitter.allow_text_splitter=true`.

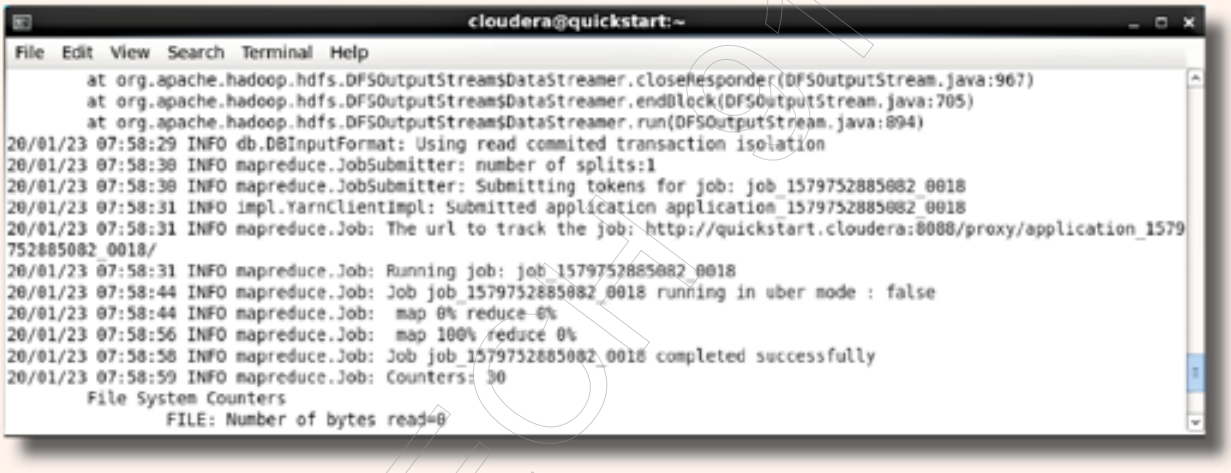
It is also applicable when we use non numeric field as part of **split-by** clause.

```
sqoop import \
-Dorg.apache.sqoop.splitter.allow_text_splitter=true \
--connect jdbc:mysql://quickstart.cloudera:3306/retail_db \
--username retail_dba \
--password cloudera \
--table categories \
--split-by "category_name" \
--warehouse-dir /user/cloudera/splitonnonnumeric \
--delete-target-dir
```



Importing table without primary key with single mapper:

```
sqoop import \
--connect jdbc:mysql://quickstart.cloudera:3306/retail_db \
--username retail_dba \
--password cloudera \
--table orders_no_pk \
--num-mappers 1 \
--warehouse-dir /user/cloudera/npkresult \
--split-by order_id \
--delete-target-dir
```

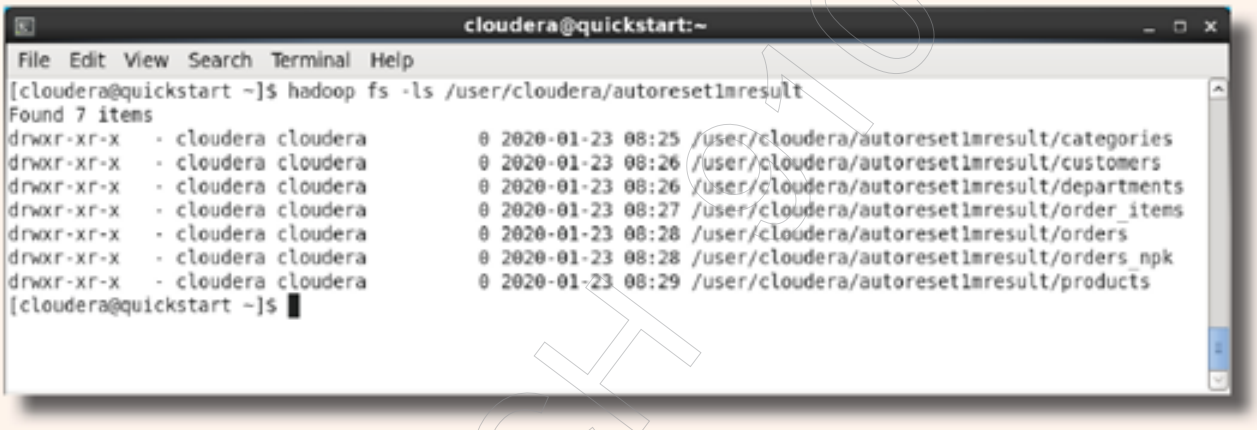


Sqoop autoreset to one mapper:

```
sqoop import \
--connect jdbc:mysql://quickstart.cloudera:3306/retail_db \
--username retail_dba \
--password cloudera \
--table orders_no_pk \
--warehouse-dir /user/cloudera/npkresult \
--delete-target-dir \
--autoreset-to-one-mapper \
--num-mappers 8
```

Sqoop autoreset to one: Example 2

```
sqoop import-all-tables \
--connect jdbc:mysql://quickstart.cloudera:3306/retail_db \
--username retail_dba \
--password cloudera \
--warehouse-dir /user/cloudera/autoreset1mresult \
--autoreset-to-one-mapper \
--num-mappers 2
```

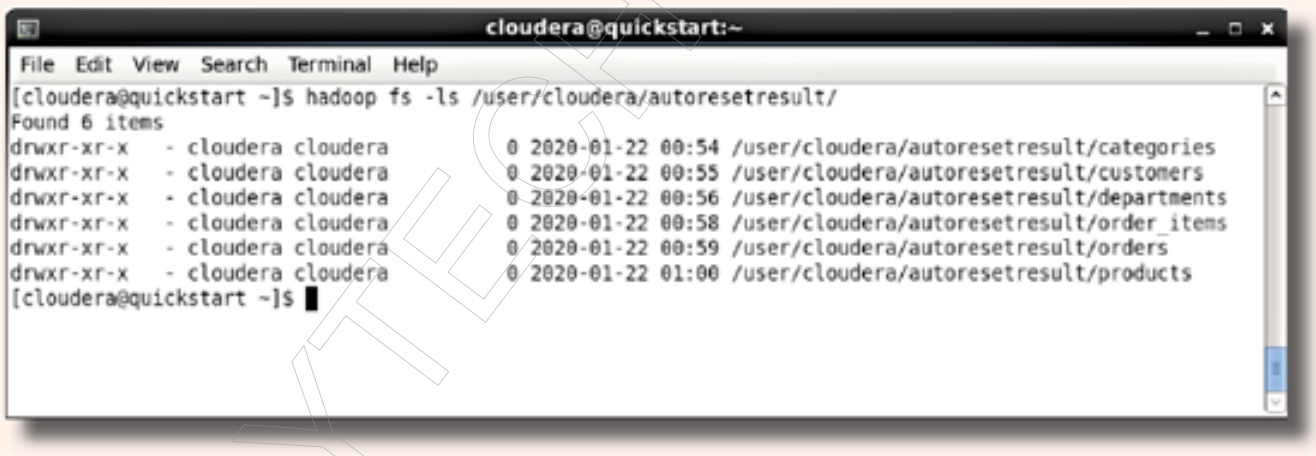


Import tables with or without primary key:

While importing all tables from RDBMS to HDFS, If a table does not have a primary key defined then import will fail.

Sqoop argument **`--autoreset-to-one-mapper`** uses one mapper if a table with no primary key is encountered.

```
sqoop-import-all-tables \
--connect jdbc:mysql://quickstart.cloudera:3306/retail_db \
--username retail_dba \
--password cloudera \
--autoreset-to-one-mapper \
--warehouse-dir /user/cloudera/autoresetresult
```

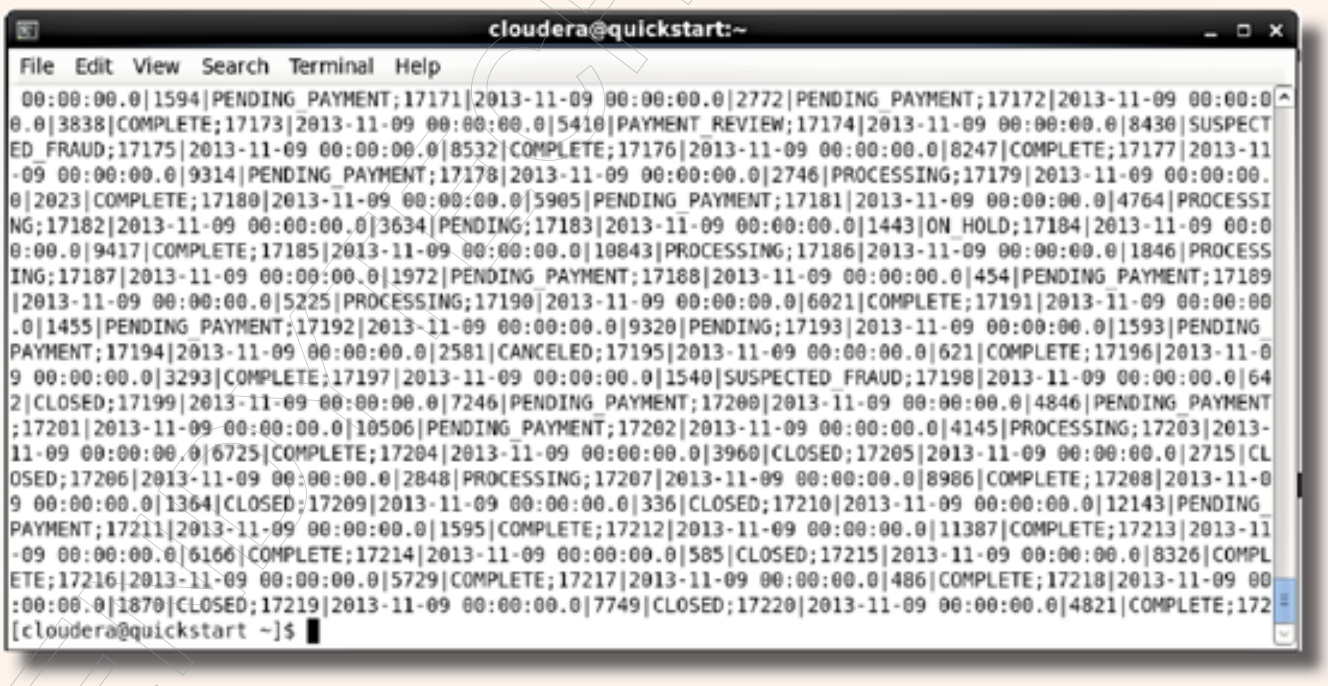


Delimiters

Delimiters may be specified by following arguments

- **--fields-terminated-by <char>**
- **--lines-terminated-by <char>**

```
sqoop-import \
--connect jdbc:mysql://quickstart.cloudera:3306/retail_db \
--username retail_dba \
--password cloudera \
--table orders \
--fields-terminated-by '|' \
--lines-terminated-by ';' \
--target-dir /user/cloudera/delimiterresult
```

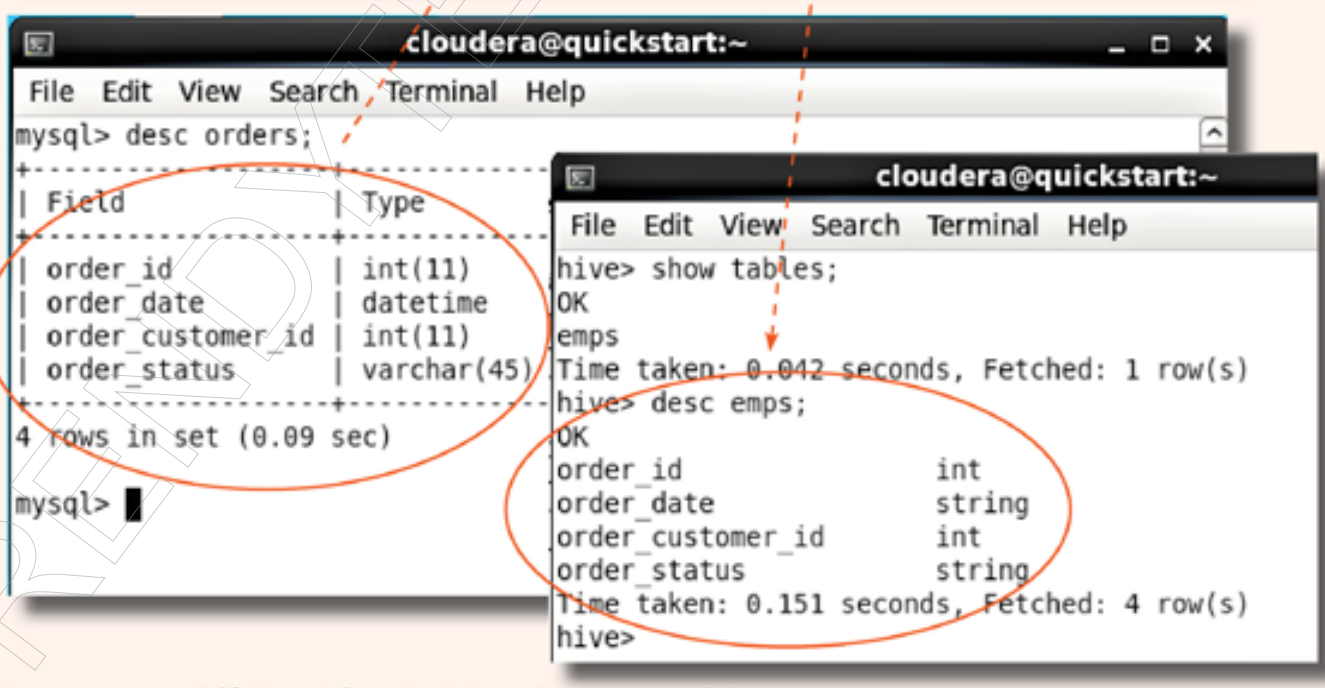


Create a Hive table based on a database table:

The `--create-hive-table` argument populates a Hive metastore with a definition for a table based on a database.

```
sqoop create-hive-table \
--connect jdbc:mysql://quickstart.cloudera:3306/retail_db \
--username retail_dba \
--password cloudera \
--table orders \
--hive-table emps \
--fields-terminated-by ','
```

created a empty table in hive based on metadata in mysql.



Note: error out if the hive table already exists.

Sqoop Verbose:

Run your Sqoop job with the `--verbose` flag to generate more logs and debugging information.

```
sqoop-import \
--connect jdbc:mysql://quickstart.cloudera:3306/retail_db \
--username retail_dba \
--password cloudera \
--table orders \
--verbose \
--target-dir /user/cloudera/verboseresult
```

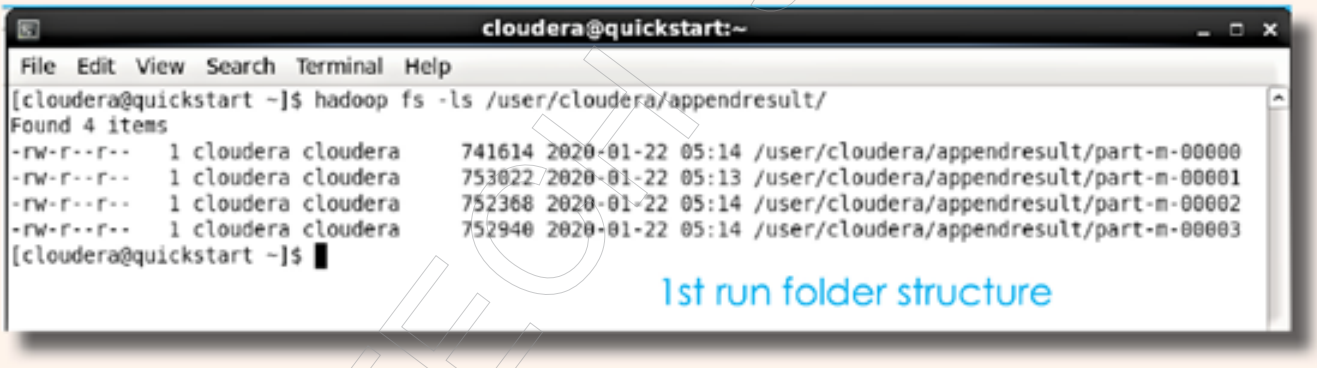
Note: We can see the bounding val queries framed properly.

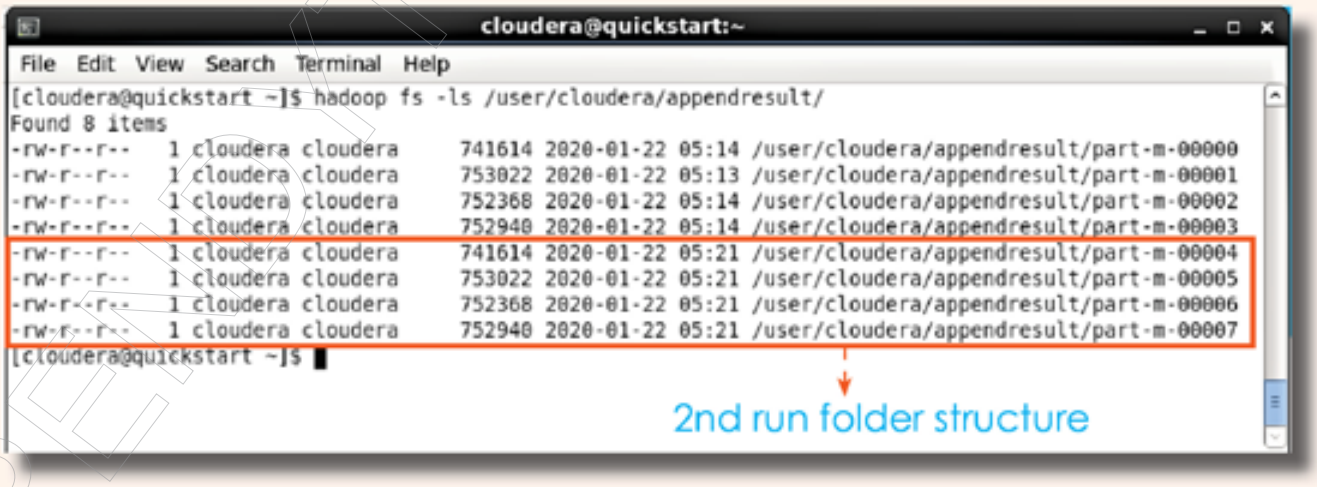
Sqoop append:

By default, imports go to a new target location. If the destination directory already exists in HDFS, Sqoop will refuse to import.

The `--append` argument will append data to an existing dataset in HDFS.

```
sqoop-import \
--connect jdbc:mysql://quickstart.cloudera:3306/retail_db \
--username retail_dba \
--password cloudera \
--table orders \
--target-dir /user/cloudera/appendresult \
--append
```





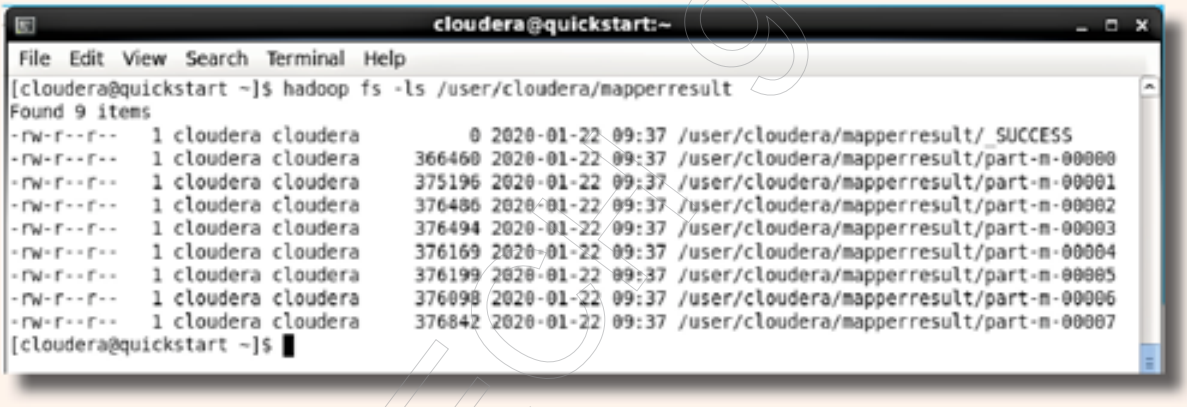
Delete target directory if exists:

```
sqoop-import \
--connect jdbc:mysql://quickstart.cloudera:3306/retail_db \
--username retail_dba \
--password cloudera \
--table orders \
--target-dir /user/cloudera/appendresult \
--delete-target-dir
```

Controlling Parallelism:

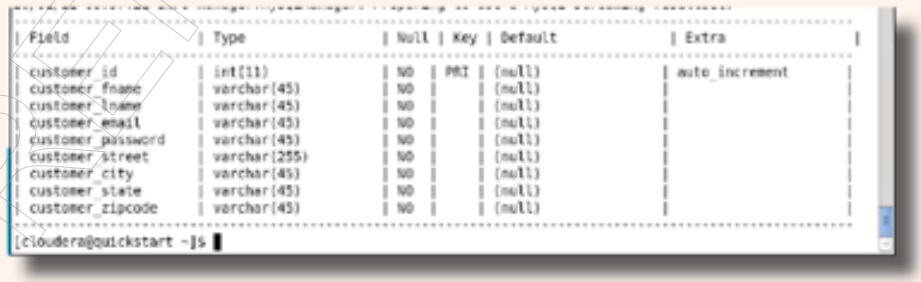
Specify the number of map tasks (parallel processes) to import data by using the `-m` or `--num-mappers` argument.

```
sqoop-import \
--connect jdbc:mysql://quickstart.cloudera:3306/retail_db \
--username retail_dba \
--password cloudera \
--table orders \
--target-dir /user/cloudera/mapperresult \
--delete-target-dir \
--num-mappers 8
```



Displaying schema of mysql table from terminal:

```
sqoop eval \
--connect jdbc:mysql://quickstart.cloudera:3306/retail_db \
--username retail_dba \
--password cloudera \
-e "DESCRIBE customers"
```

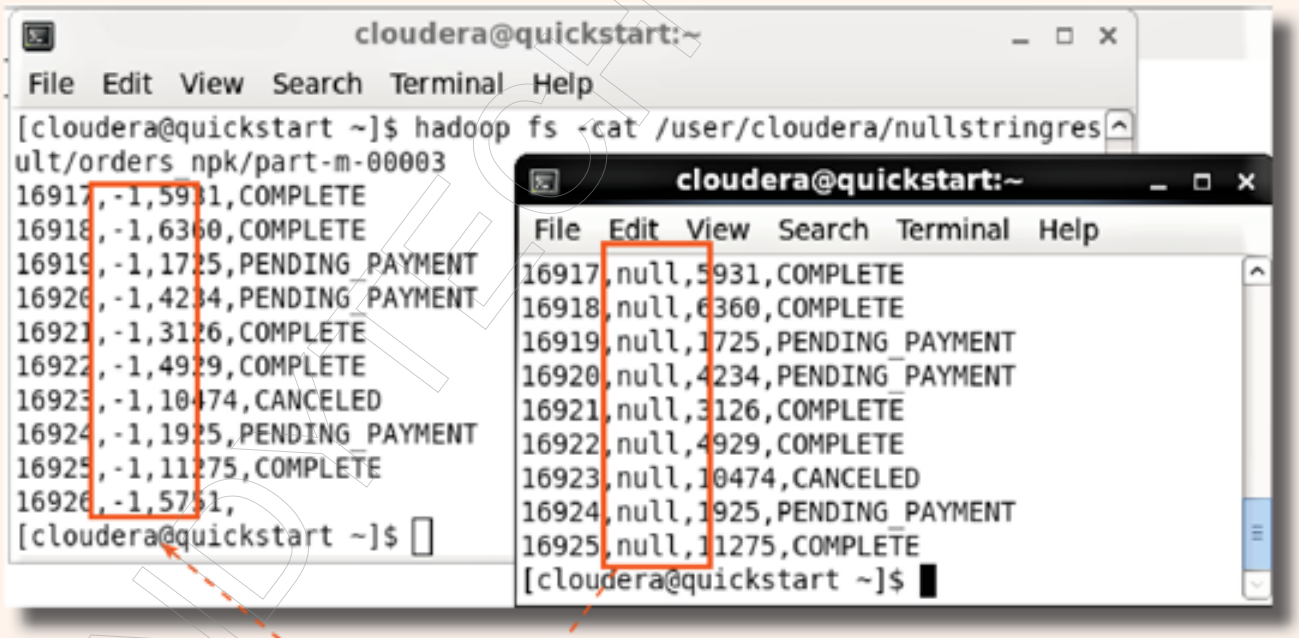


Dealing with nulls while importing data

When we import data into text files, we might have to explicitly deal with null values.

We can specify non string nulls using `--null-non-string` and string nulls using `--null-string`.

```
sqoop import \
--connect jdbc:mysql://quickstart.cloudera:3306/retail_db \
--username retail_dba \
--password cloudera \
--table orders \
--warehouse-dir /user/cloudera/nullstringresult \
--delete-target-dir \
--null-non-string "-1"
```



Google REVIEWS

5 Star Google Rated Big Data Course

LEARN FROM THE EXPERT

9108179578

Call for more details

Follow US

Trainer	Mr. Sumit Mittal
LinkedIn	https://www.linkedin.com/in/bigdatabysumit/
Website	https://trendytech.in/courses/big-data-online-training/
Phone	9108179578
Email	trendytech.sumit@gmail.com
Youtube	TrendyTech
Twitter	@BigdataBySumit
Instagram	bigdatabysumit
Facebook	https://www.facebook.com/trendytech.in/