SQOOP

Extra Interview

Sqoop is a tool used to transfer bulk data between Hadoop and external databases such as RDBMS Sqoop works with RDBMS & NOSQL \databases to import and export data
* Loading data in Sqoop is not event-driven
* Works with structured data sources and Sqoop connectors are used to fetch Data from them
* It imports data from RDBMS onto HDFS and exports it back to RDBMS

SQOOP common arguments

--connect <jdbc-url>	Specify JDBC connect string
--connection-manager <class-name>	Specify connection manager class to use
--driver <class-name>	Manually specify JDBC driver class to use
--hadoop-home <dir>	Override $HADOOP_HOME
--help	Print usage instructions
-p	Read password from console
--password <password>	Set authentication password
--username <username>	Set authentication username
--verbose	print more information while working
--connection-param-file <filename>	Optional properties file that provides connection parameters

SQOOP Import Control Arguments

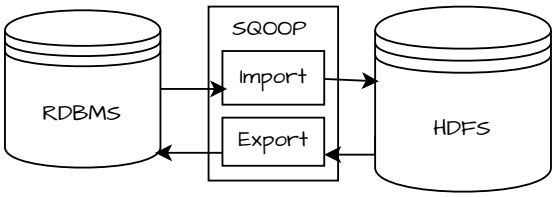
--append	Append data to an existing dataset in HDFS.
--as-avrodatafile	Imports data to Avro Data files
--as-sequencefile	Imports data to Sequence Files
--as-textfile	Imports data as plain text(default)
--boundary-query <statement>	Boundary query to use for creating splits
--columns <col, col, ...col...>	Columns to import the 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>	Use n map tasks to import in parallel
-m, --num-mappers <n>	Use n map tasks to import in parallel
-e, --query <statement>	Import the results of statemtent
--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-codex <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>	Append data to an existing dataset in HDFS

The generic import command for SQOOP is

```
sqoop import
        --connect jdbc:mysql:<HOST Database>
        --username <username>
        --password <password>
        --table <table name>
        --m <count>
        --target-dir <target dir in HDFS>
```

Issues before SQOOP

1. No direct import
(needed to use scripts)
2. These scripts we slower
(because of scripts)
3. No portion import
4. No Modified imports
5. No serialized data import
6. exports
7. Maintaining Data consistency
8,. Efficient utilization of resources
9. Loading bulk Data to Hadoop



Questions

1. What is Apache SQOOP?

-> Apache Sqoop is a command line tool designed to efficiently transfer bulk data between Hadoop and structured datastructures such as relational databses., It serves as a crucial bridge between these two worlds, enabling seamless data movement for analysis and processing.

2. What is the default number of mappers in SQOOP?

-> The default number is 4 mappers we can specify the split by column name or the primary key would be the column which the table would be split by.

3. Controlling imports/export?

-> where - can be applied over the sqoop commang using -where "condition"
this is only applicable for one table

sqoop import <connect> <--username> --<password> -table -where "condition"

-> query --query "select * from table where $conditions this can be a complex query which utilises the data from multiple tables

sqoop import <connect> --<username> --<password> --table --query

4. What is the default file format for Sqoop?

--> In Sqoop the default file format is Text file format(.txt)

5. What is SQOOP EVAL?

-> Sqoop EVAL tool allows you to quickly execute simple SQL queries against a Database, results are printed to the console. This allows us to preview import queries to ensure they import the data they expect

6. SQOOP incremental Imports?

-> Sqoop provides an incremental imports mode which can be used to retrieve only rows newer than some previously-imported set of row.
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 will add the latest changes to the HDFS this will only add the latest added rows, where as the lastmodified will add the new rows and also updates any changes that have happened in the previous data rather than importing the whole data we would only import the data which has been modified(updated)

--check-column (col)	Specifies the column to be examined when determining which rows to import
--incremental (mode)	Specifies how sqoop determines which rows are new. Legal values for **mode** include **append** and **lastmodified**
--last-value (value)	Specifies the maximum value of check column from the previous import

7. What is a SQOOP Job?

The Job tool allows us to create and work with saved jobs. Saved jobs remember the parameters used to specify a job, so they can be re-executed by invoking the job by its handle.
If a saved job is configured to perform an incremental import, state regarding the most recently imported rows is updated in the saved job to allow the job to continually import only the newest rows.
General syntax is : sqoop job (generic-args) (job-args) [--[subtool-name](subtool args)]

--create <job-id>	Define a new saved job with the specified job id(name). A second Sqoop command-line, seperated by a -- should be specified, this defnies a job
--delete <job-id>	Deletes a saved job
--exec <job-id>	Given a job defined with --create, run the saved job
--show <job-id>	show the parameters for the saved job
--list	List all saved jobs

when executing a job with a password directly you will be prompted to input the password everytime. If you want to do this automatically you can save your password in a file with no new line characters then read the password from the file.

8. SQOOP EXPORT?

The export tool exports a set of files from HDFS back to RDBMS. The target table must already exist in the database. The input files are read and parsed into a set of records according to the user-specified delimiters. The default operation is to transform these into a set of INSERT statements that injects the records into the database. In "Update Mode", sqoop will generate UPDATE statements that replace existing records in the database. Since SQOOP breaks down export process into multiple transactions, it is possible that a field export job may result in partial data being commited to the database. This can further lead to subsequent jobs failing due to insert collisions in some cases, or lead to duplicated data in others. You can overcome this problem by specifying a staging table via the **--staging-table** option which acts as an auxillary table that is used to stage exported data. The staged data in finally moved to the destination table in single transaction.
In order to use the staging facility, you must create the staging table prior to running the export job. This table must be structurally identical to the target table. This table should either be empty before the export job runs, or the **--clear-staging-table** option must be specified. If the staging table.

9. Can you import all tables in SQOOP?

The **import-all-tables** tool imports a set of tables from an RDBMS to HDFS. Data from each table is stored in a seperate directory in HDFS. For the **import-all-tables** tool to be useful the following conditions must be met

* Each table must have a single column primary key.
* You must intend to inmport all columns of each table.
* You must not intend to use non-default splitting column, nor impose any conditions via a WHERE caluse.

sqoop import-all-tables --connect <database> --username<> --password <> hadoop --warehous-dir 'location'
--exclude-tables <tab1>, <tab2>

10. Dates and Times in SQOOP?

Any DATE columns in a Database will be imported as TIMESTAMP in Sqoop, and Sqoop-generated code will store these values in java.sql.Timestamp fields.
In case of Hive since there is no one-to-one mapping between SQL types and Hive types. In general, SQL types that do not have direct mapping (DATE, TIME, TIMESTAMP) will be coerced to STRING in Hive. The NUMERIC & DECIMAL SQL types will be coerced to DOUBLE. In these cases SQOOP will emit a warning in its log messages informing you of the loss of precision.

11. MapColumn Java?

-> it is a Sqoop option that allows us to override the default mapping of SQL data types to Java data types during the import process. This is useful when the default mapping doesn't suit the specific needs.

--map-column-java "<col>=<type>"

12. What is SQOOP HCatalogue?

-> HCatalogue serves as a table and a storage management service for Apache Hadoop, which enables the users with different data processing tools such as Hive, Pig, MapReduce to read and write data on a grid with ease.

*By directly importing data into HCatalog tables, we eliminate the need for Hive table creation.

* HCatalog provides a structured way to manage and organize data in Hadoop.

* It improves query execution by providing metadata about data location and schema.

* HCatalog enables easy access to imported data using Hive, Pig, and other Hadoop tools.

13. Fetch Size?

-> specifies teh number of entries that Sqoop can import at a time "--fetch-size=<n>"

default is 1000, we can increase the n value based on the volume of the data that need to read. based on the available memory & bandwidth

14. Mappers?

Specifies number of map tasks that can run in parallel. Default is 4. To optimize performanace, set the number of map tasks to avalue lower than the maximum number of connections that the database supports

15. Split-By?

Specifies the column name based on which sqoop must split the work units

Note: If we donot specify a column name, sqoop splits the work units based on the primary key.

16. Direct Mode?

-> By default, the import process will use JDBC, some databases can perform imports in a more high-performance fashion by using data base specific data movement tools. By supplying the --direct argument, you are specifying that sqoop should attempt the direct import channel which is higher performant than using JDBC.

Sqoop only supports direct data drivers for MySQL and PostgreSQL so direct only works for these databases only.

17. Exports Performance?

-> when you export data, you can configure the batch argument along with the

Dsqoop.export.records.per.statement=1000

Dsqoop.export.statement.per.statement=1000

When you export data, you can configure the batch argument along with the above arguments to insert multiple rows with a single statement.

in the above case for every statement we would add 1000 values in that one insert command like this we willl have 1000 statements per cycle combning 1000 * 1000 = 1000000 (1M) records every time

18. Types of Serialized file formats?

-> * TextFile -- Readable and huge in size and default file format in SQOOP

* SequenceFile -- Java file format helps for faster processing for Mapreduce

* Parquet -- Columnar format and has predicate pushdown. Used for Target system for faster query and supports snappy compression of about 60-80% comporession percentage. useful where data reads are frequent faster querying

* ORC -- Columnar format, high on compression of about 90% due to ZLIB inbuilt used for legacy and historical data storage

* AVRO -- Supports 50-60% compression ratio an its a row file format, and its supports Schema evolution. Adaptable for changing data structures.

COMMANDS

#Import data from MySQL table orders into HDFS location /user/cloudera/orders #As Textfile format #Default field delimiter is comma

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

#Import data from MySQL table orders into HDFS location /user/cloudera/orders #Let sqoop delete target dir if exists

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

#Import data from MySQL table orders into HDFS location /user/cloudera/orders #Overriding the fields delimiter to $

```
sqoop import \
--connect jdbc:mysql://quickstart.cloudera:3306/retail_db \
--username retail_dba \
--password cloudera \
--table orders \
--delete-target-dir \
--as-textfile \
--target-dir=/user/cloudera/orders \
--fields-terminated-by '$'
```

Q. Free form import $CONDITIONS must be passed in the where clause and also split-by field must be specified

```
sqoop import \
--connect jdbc:mysql://quickstart.cloudera:3306/retail_db \
--username retail_dba \
--password cloudera \
--delete-target-dir \
--as-textfile \
--target-dir=/user/cloudera/orders \
--query "select * from orders where \$CONDITIONS" \
--split-by order_id
```

Q. Free form import with user-defined conditions

```
sqoop import \
--connect jdbc:mysql://quickstart.cloudera:3306/retail_db \
--username retail_dba \
--password cloudera \
--delete-target-dir \
--as-textfile \
--target-dir=/user/cloudera/orders \
--query "select * from orders where \$CONDITIONS
and order_status='COMPLETE'" \
--split-by order_id
```

Q. Built-in validator Validator works with single table only... and cannot use where criteria

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

Q. Import mysql table data into HDFS using Sequence File format

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

Q. Import mysql table data into HDFS using Avro File format This will create sqoop_import_<<Table>>.avsc file under current directory where sqoop command is executed It is the schema file

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

Q. We can use the .avsc file to import data into Hive

```
hadoop fs -put sqoop_import_orders.avsc /user/cloudera
```

Q. Using boundary query, selected list of columns
When you specify list of columns, make sure there is no white-space between field name and comma

```
sqoop import \
--connect jdbc:mysql://quickstart.cloudera:3306/retail_db \
--username retail_dba \
--password cloudera \
--table orders \
--target-dir /user/cloudera/orders \
--delete-target-dir \
--boundary-query "select 100, 200 from orders limit 1" \
--columns order_id,order_status
```

Q. Incremental Load You need to specify the check-column (mostly the primary key column) and its last value

```
sqoop import \
--connect jdbc:mysql://quickstart.cloudera:3306/retail_db \
--username retail_dba \
--password cloudera \
--table orders \
--target-dir /user/cloudera/orders \
--append \
--where "order_id < 600" \
--check-column "order_id" \
--last-value 99 \
--incremental append
```

Q. Import data from MySQL table into Hive (default HDFS Location /user/hive/warehouse)
#For importing data into Hive, we need to explicitely specify the field delimiter #When we do import into Hive, the data will be first copied under default hdfs folder and then to hive #Hence it is must to delete the hdfs folder #For ex: if you're importing orders table into hive, data will first copied to hdfs folder /user/cloudera/orders #If your source table is not having any primary key defined, #you need to explicitely define the column which contains unique key using --split-by switch
#We cannot use --hive-import and --as-avrodatafile or --as-sequencefile together #Hive import doesn't compatible with Avro or Sequence format

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

Q. You can use --delete-target-dir option to let sqoop delete the target hdfs folder

```
sqoop import \
--connect jdbc:mysql://quickstart.cloudera:3306/retail_db \
--username retail_dba \
--password cloudera \
--table orders \
--delete-target-dir \
--hive-import \
--hive-overwrite \
--hive-table orders \
--fields-terminated-by ','
```

Q. If you want to override $HIVE_HOME environment variable, you can use --hive-home switch

```
sqoop import \
--connect jdbc:mysql://quickstart.cloudera:3306/retail_db \
--username retail_dba \
--password cloudera \
--table orders \
--delete-target-dir \
--hive-home /home/cloudera/hive \
--hive-import \
--hive-overwrite \
--hive-table orders \
--fields-terminated-by ','
```

Q. when you specify --create-hive-table switch, and if the table already exists #import will fail

```
sqoop import \
--connect jdbc:mysql://quickstart.cloudera:3306/retail_db \
--username retail_dba \
--password cloudera \
--table orders \
--delete-target-dir \
--hive-database retail_db \
--hive-import \
--hive-overwrite \
--hive-table orders \
--create-hive-table \
--fields-terminated-by ','
```

Q. Export hive table data into MySQL #Make sure target mysql table exists and having same schema

```
sqoop export \
--connect jdbc:mysql://quickstart.cloudera:3306/retail_db \
--username retail_dba \
--password cloudera \
--table orders_test \
--export-dir /user/hive/warehouse/retail_db.db/orders
```

Q. Serial export

```
sqoop export \
--connect jdbc:mysql://quickstart.cloudera:3306/retail_db \
--username retail_dba \
--password cloudera \
--table orders_test \
--export-dir /user/hive/warehouse/retail_db.db/orders \
--m 1
```

Q. If your hive table columns are using delimiter other than ',' #you need to explicitely specify the delimiter using --input-fields-terminated-by switch

```
sqoop export \
--connect jdbc:mysql://quickstart.cloudera:3306/retail_db \
--username retail_dba \
--password cloudera \
--table orders_test \
--export-dir /user/hive/warehouse/retail_db.db/orders \
--input-fields-terminated-by ',' \
```

Q. Export specific columns #In this case, make sure all the other columns in the mysql table has default constraint

```
sqoop export \
#If you want to import the table under specific Hive
database #you can use --hive-database switch
sqoop import \
--connect jdbc:mysql://quickstart.cloudera:3306/retail_db \
--username retail_dba \
--password cloudera \
--table orders \
--delete-target-dir \
--hive-database retail_db \
--hive-import \
--hive-overwrite \
--hive-table orders \
--fields-terminated-by ','

--connect jdbc:mysql://quickstart.cloudera:3306/retail_db \
--username retail_dba \
--password cloudera \
--table orders_test \
--export-dir /user/hive/warehouse/retail_db.db/orders \
--input-fields-terminated-by ',' \
--columns order_id,order_status
```

Q. It is good to verify the target folder exists or not.

```
hadoop fs -ls /user/cloudera/categories
hadoop fs -ls /user/cloudera/customers
hadoop fs -ls /user/cloudera/departments
hadoop fs -ls /user/cloudera/order_items
hadoop fs -ls /user/cloudera/orders
hadoop fs -ls /user/cloudera/products
```

Q. Deleting the target folder

```
hadoop fs -rm -R /user/clouders/categories
hadoop fs -rm -R /user/clouders/customers
hadoop fs -rm -R /user/clouders/departments
hadoop fs -rm -R /user/clouders/order_items
hadoop fs -rm -R /user/clouders/orders
hadoop fs -rm -R /user/clouders/products
```

Q. Import all tables from MySQL database retail_db into HDFS location /user/cloudera/

```
sqoop import-all-tables \
--connect jdbc:mysql://quickstart.cloudera:3306/retail_db \
--username retail_dba \
--password cloudera \
--as-textfile \
--warehouse-dir=/user/cloudera
```

Q. Import all tables from MySQL database retail_db into HDFS location /user/cloudera #Avro file format
This will create .avsc files into the current local working folder

```
sqoop import-all-tables \
--connect jdbc:mysql://quickstart.cloudera:3306/retail_db \
--username retail_dba \
--password cloudera \
--as-avrodatafile \
--warehouse-dir=/user/cloudera
```

Q. It is good to verify the target folder exists or not. We're going to import all the tables into Hive database named retail_db #So, the hive tables will be copied into /user/hive/warehouse/retail_db.db folder

```
hadoop fs -ls /user/hive/warehouse/retail_db.db/categories
hadoop fs -ls /user/hive/warehouse/retail_db.db/customers
hadoop fs -ls /user/hive/warehouse/retail_db.db/departments
hadoop fs -ls /user/hive/warehouse/retail_db.db/order_items
hadoop fs -ls /user/hive/warehouse/retail_db.db/orders
hadoop fs -ls /user/hive/warehouse/retail_db.db/products
```

Q. Deleting the target folder
--delete-target-dir will not work with sqoop import-all-tables

```
hadoop fs -rm -R /user/hive/warehouse/retail_db.db/categories
hadoop fs -rm -R /user/hive/warehouse/retail_db.db/customers
hadoop fs -rm -R /user/hive/warehouse/retail_db.db/departments
hadoop fs -rm -R /user/hive/warehouse/retail_db.db/order_items
hadoop fs -rm -R /user/hive/warehouse/retail_db.db/orders
hadoop fs -rm -R /user/hive/warehouse/retail_db.db/products

hadoop fs -rm -R /user/cloudera/categories
hadoop fs -rm -R /user/cloudera/customers
hadoop fs -rm -R /user/cloudera/departments
hadoop fs -rm -R /user/cloudera/order_items
hadoop fs -rm -R /user/cloudera/orders
hadoop fs -rm -R /user/cloudera/products
```

Q. Import all tables from MySQL database retail_db into Hive default database #We cannot use --hive-import and --as-avrodatafile or --as-sequencefile together #Hive import doesn't compatible with Avro or Sequence format

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

Q. Import all tables from MySQL database retail_db into Hive database retail_db #When you are using --create-hive-table switch you have to make sure, the tables which you're going to import doesn't exists in retail_db database #If yes, you need to drop them and delete any hdfs folder with table exists, delete them as well

```
sqoop import-all-tables \
--connect jdbc:mysql://quickstart.cloudera:3306/retail_db \
--username retail_dba \
--password cloudera \
--fields-terminated-by ',' \
--hive-import \
--create-hive-table \
--hive-overwrite \
--hive-database retail_db
```

Q. Import all tables from MySQL database retail_db into Hive database retail_db #Removed --create-hive-table switch

```
sqoop import-all-tables \
--connect jdbc:mysql://quickstart.cloudera:3306/retail_db \
--username retail_dba \
--password cloudera \
--fields-terminated-by ',' \
--hive-import \
--create-hive-table \
--hive-overwrite \
--hive-database retail_db
```

```
sqoop import-all-tables \
-m 5 \
--connect "jdbc:mysql://quickstart.cloudera:3306/retail_db" \
--username=retail_dba \
--password=cloudera \
--as-avrodatafile \
--warehouse-dir=/user/hive/warehouse/retail_db.db

  #Verification Script (using sqoop eval)
  sqoop eval \
  --connect jdbc:mysql://quickstart.cloudera:3306/retail_db \
  --username retail_dba \
  --password cloudera \
  --query "select count(1) from orders"

  #Conditional Import
  sqoop import \
  --connect jdbc:mysql://quickstart.cloudera:3306/retail_db \
  --username retail_dba \
  --password cloudera \
  --table orders \
  --delete-target-dir \
  --as-textfile \
  --target-dir=/user/cloudera/orders \
  --where "order_status='COMPLETE'"

  #Conditional Import & Append to existing file
  sqoop import \
  --connect jdbc:mysql://quickstart.cloudera:3306/retail_db \
  --username retail_dba \
  --password cloudera \
  --table orders \
  --as-textfile \
  --target-dir=/user/cloudera/orders \
  --where "order_status='CANCELED'" \
  --append

  #Sqoop Import using Serial way
  #This will have impact in performance
  sqoop import \
  --connect jdbc:mysql://quickstart.cloudera:3306/retail_db \
  --username retail_dba \
  --password cloudera \
  --table orders \
  --delete-target-dir \
  --as-textfile \
  --target-dir=/user/cloudera/orders \
  --where "order_status='COMPLETE'" \
  --m 1
```

Practice Questions:
Q : Import orders table into hive.

Q : You can use --delete-target-dir option to let sqoop delete the target hdfs folder

Q : create a dummy database in hive and import orders tables into that db. Take only orders greater than 100.

Q : select first 4 columns from the hive orders table and export to mysql.

Q : Import mysql orders tables into hive with $ as delimiter. export this table first 4 columns to mysql.

Q : Export the sequencefile stored in hdfs to mysql

Q : Export the avro file stored in hdfs to mysql

Q : Import data from a MySQL database into HDFS using Sqoop
Importing orders table data from retail_db into HDFS - textfile format with default delimiter, default mapper - even if we dont specify target-dir, the output will be written to default hdfs folder
Q : Importing orders table data from retail_db into HDFS
- textfile format with default delimiter, default mapper - Clean up target dir if it exists

Q : Importing orders table data from retail_db into HDFS
- textfile format with only one mapper (so, total files will be - Clean up target dir if it exists

Q : Importing orders table data from retail_db into HDFS
- textfile format with only one mapper (so, total files will be - Conditional Import

Q : Importing orders table data from retail_db into HDFS with append
- textfile format with only one mapper (so, total files will be - Conditional Import

Q : Importing orders table data from retail_db into HDFS
- textfile format with only one mapper (so, total files will be - Clean up target dir if it exists - Only specific columns (order_id, order_date, order_status) - When we use --columns switch, there shouldn't be any white-space char between columns

Q : Importing orders table data from retail_db into HDFS
- textfile format with freeform query - when using freeform query, we must specify target-dir
- and also, we must specify split-by switch to specify column provided if we dont pass --m 1

Q : Importing orders table data from retail_db into HDFS
- sequencefile - when using freeform query, we must specify target-dir - and also, we must specify split-by switch to specify column provided if we dont pass --m 1

Q : Importing orders table data from retail_db into HDFS
- avrodatafile (This option will create .avsc file (AVRO Schema file) in the local working folder - when using freeform query, we must specify target-dir - and also, we must specify split-by switch to specify column provided if we dont pass --m 1

Q : Export data to a MySQL database from HDFS using Sqoop
- Exporting orders data from HDFS to MySQL table orders_export_test

Q. Import all the records from the table order_items in the retail_db schema. Fields should be delimited by tabs Files should be compressed and located at /user/cloudera/classwork/retail_db/order_item

Q. Run this query on your retail_db schemacreate table order_item_2 as select * from order_items where 1 =2 ; alter table order_item_2 drop order_item_order_id;
Now export all records from /user/cloudera/classwork/retail_db/order_item into the database table order_item_2

Q. Import a table from mysql where there is not primary key in the table.
Split-by on a textual column.
user of --auto-reset-mapper

Q. Import all customer record from the customers table in retail_db mysql database into HDFS path "/user/cloudera/output/retail_db/customers_text".
Fields should be tab delimited and all records must be imported into a single text file

Q. Import all CLOSED customer orders from the orders table in retail_db mysql database into HDFS path "/user/cloudera/output/retail_db/orders_avro". Records must be imported in avro format.
All imported records must be CLOSED orders.

Q. Import all customer record from the customers table in retail_db mysql database into HDFS path "/user/cloudera/output/retail_db/customers_text".
Fields should be tab delimited and all records must be imported into a single text file

Q. Import all CLOSED customer orders from the orders table in retail_db mysql database into HDFS path "/user/cloudera/output/retail_db/orders_avro". Records must be imported in avro format.
All imported records must be CLOSED orders.

Q. Import all order_items from the order_items and product tables in retail_db mysql database into HDFS path "/user/cloudera/output/retail_db/denorm_order_items".
The imported dataset must contain the following columns in this order:
order_item_id, order_item_order_id, product_name, product_category_id, product_price, order_item_quantity, order_item_subtotal Records must be imported in parquet format.

Q. Export all record from HDFS location "/user/cloudera/rawdata/hist_forex/price_data" to the database table price_data in schema hist_forex;

Q. Import all the records from the table order_items in the retail_db schema.
Fields should be delimited by tabs
Files should be compressed and located at /user/cloudera/classwork/retail_db/order_item

Q. Run this query on your retail_db schema
create table order_item_2 as select * from order_items where 1 =2 ;
alter table order_item_2 drop order_item_order_id;
Now export all records from /user/cloudera/classwork/retail_db/order_item into the database table order_item_2