## **Sqoop Import**

## **Running Simple Sqoop Command**

Below Sqoop import command will ingest data from table customers in MySQL database. It will use MySQL JDBC connector to connect to database using username and password provided in the Sqoop command. It will ingest the data parallelly using 4 mappers that will run as MapReduce program in Yarn cluster and will load the data into default warehouse directory. Below Sqoop command will generate a sub folder inside default warehouse directory with the same name as table name and all output part files will be generated inside that subfolder.

sqoop import \

--connect jdbc:mysql://localhost/retail\_db \

--username root \

--password pikachu123 \

--table customers

**Managing Warehouse Dir**

Default warehouse directory can be changed using “warehouse-dir” flag.

sqoop import \

--connect jdbc:mysql://localhost/retail\_db \

--username root \

--password pikachu123 \

--table customers \

--warehouse-dir /user/sqoop/new-warehouse

**Managing Target Dir**

Instead of generating output inside warehouse directory, we can define an output location called as target directory under which Sqoop command will load output files. Target directory can be defined by using “—target-dir” flag

sqoop import \

--connect jdbc:mysql://localhost/retail\_db \

--username root \

--password pikachu123 \

--table customers \

--target-dir /user/sqoop/my-target

**Import Data as Parquet File**

By default, data is generated as text file in which columns are separated by commas but Sqoop command can also be used to generate parquet file. To import the data of table as parquet file, you need to use “—as-parquetfile” flag in the Sqoop command

sqoop import \

--connect jdbc:mysql://localhost/retail\_db \

--username root \

--password pikachu123 \

--table customers \

--target-dir /user/customer/parquet \

--as-parquetfile

**To view the data in parquet file in JSON format:**  
  
hadoop jar ./parquet-tools-1.10.0.jar cat --json /user/parquet/7bef434e-6f2c-4dec-a8e9-5b401dc0a883.parquet  
  
**To get meta data of the parquet file:**  
  
hadoop jar ./parquet-tools-1.10.0.jar meta /user/parquet/7bef434e-6f2c-4dec-a8e9-5b401dc0a883.parquet

**Import Data as Avro File**

To import the data of table as avro file, you need to use “—as-avrodatafile” flag in the Sqoop command

sqoop import \

--connect jdbc:mysql://localhost/retail\_db \

--username root \

--password pikachu123 \

--table customers \

--target-dir /user/customer/parquet \

--as-avrodatafile  
  
**To view the transferred file contents:**  
hdfs dfs -cat /user/avro/part-m-00002.avro  
  
**To view data in JSON format:**  
  
run the following command:  
hadoop jar ./avro-tools-1.8.2.jar tojson /user/avro/part-m-00002.avro  
  
**To check schema of the avro file:**  
hadoop jar ./avro-tools-1.8.2.jar getschema /user/avro/part-m-00002.avro

**Import Data as Sequence File**

To import the data of table as sequence file, you need to use “—as-sequencefile” flag in the Sqoop command

sqoop import \

--connect jdbc:mysql://localhost/retail\_db \

--username root \

--password pikachu123 \

--table customers \

--target-dir /user/customer/parquet \

--as-sequencefile

**Import Data as Gzip compressed File**

To import the data in gzip compression, you need to use “--compress” flag in the Sqoop command

sqoop import \

--connect jdbc:mysql://localhost/retail\_db \

--username root \

--password pikachu123 \

--table customers \

--target-dir /user/customer/gzip \

--compress

**Import Data as Snappy compressed File**

To import the data in snappy compression, you need to use “--compress” and “—compression-codec” flag in the Sqoop command

sqoop import \

--connect jdbc:mysql://localhost/retail\_db \

--username root \

--password pikachu123 \

--table customers \

--target-dir /user/customer/snappy \

--compress

--compression-codec snappy

**Import Data as Bzip2 compressed File**

To import the data in bzip compression, you need to use “--compress” and “—compression-codec” flag in the Sqoop command

sqoop import \

--connect jdbc:mysql://localhost/retail\_db \

--username root \

--password pikachu123 \

--table customers \

--target-dir /user/customer/bzip2 \

--compress

--compression-codec bzip2

**Import Data as lz4 compressed File**

To import the data in lz4 compression, you need to use “--compress” and “—compression-codec” flag in the Sqoop command

sqoop import \

--connect jdbc:mysql://localhost/retail\_db \

--username root \

--password pikachu123 \

--table customers \

--target-dir /user/customer/lz4 \

--compress

--compression-codec lz4

**Conditional Import**

You can include a where condition in the Sqoop command using “--where” flag to import data based on some condition.

sqoop import \

--connect jdbc:mysql://localhost/retail\_db \

--username root \

--password pikachu123 \

--table customers \

--target-dir /user/customer/conditional \

--where "customer\_id>100 "

**Selective Import**

You can also specify the selective columns in the Sqoop command using “--where” flag to import only data from those columns

sqoop import \

--connect jdbc:mysql://localhost/retail\_db \

--username root \

--password pikachu123 \

--table customers \

--target-dir /user/customer/selected \

--columns "customer\_id,customer\_fname,customer\_lname "

**Running SQL Query in Sqoop**

Sqoop is very flexible and allows you to run sql queries inside the sqoop command using “—query” flag. You need to provide “—spli-by” column explicitly

sqoop import \

--connect jdbc:mysql://localhost/retail\_db \

--username root \

--password pikachu123 \

--target-dir /user/customer/query \

--query 'Select customer\_id,customer\_fname,customer\_lname from customers where customer\_id>100 AND $CONDITIONS' \

--split-by customer\_id

**Split-By in Sqoop Command**

sqoop import \

--connect jdbc:mysql://localhost/retail\_db \

--username root \

--password pikachu123 \

--table products \

--split-by product\_id

**Fields delimiters**

You can explicitly specify the delimiters to be used in your output files to separate the fields. By default, if we don’t specify anything, it is comma.

sqoop import \

--connect jdbc:mysql://localhost/retail\_db \

--username root \

--password pikachu123 \

--table customers \

--target-dir /user/customers/delimited \

--fields-terminated-by '|'

**List Database**

Using sqoop, you can list the database in your relational database.

sqoop list-databases \

--connect jdbc:mysql://localhost/ \

--username root \

--password pikachu123

**List Tables**

Using sqoop, you can list the tables in your relational database.

sqoop list-tables \

--connect jdbc:mysql://localhost/retail\_db \

--username root \

--password pikachu123

**Incremental Append**

sqoop import \

--connect jdbc:mysql://localhost/retail\_db \

--username root \

--password pikachu123 \

--table orders \

--target-dir /orders \

--incremental append \

--check-column order\_id \

--last-value 100009

**Incremental Append Sqoop Job**

sqoop job –create order\_update\_job1 \

-- import \

--connect jdbc:mysql://localhost/retail\_db \

--username root \

--password pikachu123 \

--table orders \

--target-dir orders \

--incremental append \

--check-column order\_id \

--last-value 100009

**Hive Import**

sqoop import \

--connect jdbc:mysql://localhost/retail\_db \

--username root \

--password pikachu123 \

--table customers \

--hive-import \

--create-hive-table \

--hive-database default \

--hive-table customer\_mysql

## **Sqoop Export**

**HDFS to MySQL**

sqoop export \

--connect jdbc:mysql://localhost/export\_db \

--username root \

--password pikachu123 \

--table customer\_export \

--export-dir /user/customers/text

**HDFS to MySQL (Pipe Delimited File)**

sqoop export \

--connect jdbc:mysql://localhost/export\_db \

--username root \

--password pikachu123 \

--table customer\_delimited \

--export-dir /user/customers/delimited \

--input-fields-terminated-by '|'