Table of Contents

Creating Table in RDBMS:

1. Open My Sql Instance either in My sql editor or command line.

2. Create a table named customer using below query in test database:

```
CREATE TABLE test.customer (cust_id int, name varchar(50), age int, country varchar(100));
```

3. Insert data into table customer

```
INSERT INTO  test.customer VALUES(101,'John',25,'US');
INSERT INTO  test.customer VALUES(102,'Jack',35,'US');
INSERT INTO  test.customer VALUES(103,'Peter',28,'UK');
INSERT INTO  test.customer VALUES(104,'Katie',30,'US');
INSERT INTO  test.customer VALUES(105,'Daniel',22,'UK');
```

4. Check whether data is inserted in customer table or not

```
SELECT * FROM test.customer;
```

cust_id	name	age	country
101	John	25	US
102	Jack	35	US
103	Peter	28	UK
104	Katie	30	US
105	Daniel	22	UK

Importing RDBMS data into HDFS:

5. Check whether mysql-connectorXXXX.jar is present or not under /usr/lib/sqoop directory of your hadoop machine.

6. If not, then download and copy the mysql-connector jar to /usr/lib/sqoop directory.

7. Import customer table's data to HDFS using sqoop command.

```
$ sqoop-import --connect jdbc:mysql://192.168.1.121/test   \
--username root –password root      \
--table customer      \
--target-dir /etl/input/
```

Note: Please delete the customer directory from HDFS(if present) before running sqoop command.

8. Check whether data is imported to HDFS or not from customer table

```
$ hadoop fs -cat /etl/input/customer/part-m*
```

9. Import only those records of customer table where country is 'US'

```
$ sqoop-import --connect jdbc:mysql://192.168.1.121/test   \
--username root –password root      \
--table customer      \
--where  "country = 'US'"      \
--target-dir /etl/input/US/
```

10. Import only id, name and age from customer table belonging to UK and load to /etl/input/selected/customer path in HDFS

```
$ sqoop-import --connect jdbc:mysql://192.168.1.121/test   \
--username root –password root      \
--table customer      \
--query 'SELECT  id,name,age FROM test.customer WHERE country='UK''       \
--split-by id \
--target-dir /etl/input/UK/
```

Note: --split-by is used for slicing data to multiple parallel tasks. Usually done on primary key of table.

11. Defining number of mappers explicitly

```
$ sqoop-import --connect jdbc:mysql://192.168.1.121/test   \
--username root –password root      \
--table customer      \
--target-dir /etl/input/         \
-m 5
```

12. Import customer table data to HDFS as avro data file

```
$ sqoop-import --connect jdbc:mysql://192.168.1.121/test   \
--username root –password root      \
--table customer      \
--target-dir /etl/input/
--as-avrodatafile
```

13. Import the data of customer table to hive table

```
$ sqoop-import --connect jdbc:mysql://192.168.1.121/test   \
--username root –password root      \
--table customer      \
--hive-import
```

14. Check whether data is loaded to Hive table correctly

```
Select * from customer;
```

15. Add 1 row to your customer table in mysql

```
INSERT INTO  test.customer VALUES(106,'Vipul',25,'IN');
```

16. Check whether data is inserted in customer table or not

```
SELECT * FROM test.customer
```

cust_id	name	age	country
101	John	25	US
102	Jack	35	US
103	Peter	28	UK
104	Katie	30	US
105	Daniel	22	UK
106	Vipul	25	IN

17. Append the newly added row(incremental import) in customer table to /etl/input/ directory of HDFS

```
$ sqoop-import --connect jdbc:mysql://192.168.1.121/test   \
--username root –password root      \
--table customer      \
--incremental append          \
--check-column id    \
--last-value 105       \
--target-dir /etl/input/
```

18. Import the data of customer table to Hbase

```
$ sqoop-import --connect jdbc:mysql://192.168.1.121/test   \
-Dsqoop.hbase.add.row.key=true     \
--username root –password root      \
--table customer      \
--hbase-table customer       \
```

```
--column-family market
```

Note: HBase table and column family should exist before importing using sqoop.

Exporting HDFS data into RDBMS:

1. Create a new table in mysql with name customer_new in test db in which we will export data from HDFS

```
CREATE TABLE test.customer_new (cust_id int, name varchar(50), age int, country varchar(100));
```

2. Check whether data is present in HDFS path /etl/input/customer/ or not

```
$ hadoop fs -cat /etl/input/customer/part*
```

3. Export data from HDFS path /etl/input/customer/ to customer_new table in RDBMS under test db

```
$ sqoop-export --connect jdbc:mysql://192.168.1.121/test
 --username root --password root          \
--table customer_new   \
--export-dir /etl/input/customer/part* -m 2
```

4. Check whether data is exported to customer_new table of mysql or not

```
SELECT * FROM customer_new;
```

5. Export the data from HDFS to mysql table customer_new: Update existing records and add newly added records.

```
$ sqoop-export --connect jdbc:mysql://192.168.1.121/test
 --username root --password root          \
--table customer_new   \
--update-key id \
--update-mode allowinsert  --export-dir /etl/input/customer/part*
```