**[CCA EXAM CODE GUIDE](#_top)**

[**SQOOP**](#_SQOOP)

[**HIVE**](#_HIVE)

[**COMMON HDFS COMMANDS FROM COMMAND PROMPT IN LINUX**](#_COMMON_HDFS_COMMANDS)

[**SPARK**](#_SPARK_RDD’s)

# SQOOP

1. [SQOOP import a whole table](#_SQOOP_import_a)
2. [SQOOP IMPORT SPECIFIC COLUMNS](#_SQOOP_IMPORT_SPECIFIC)
3. [SQOOP EXPORT](#_SQOOP_EXPORT)

## SQOOP import a whole table

Sqoop import --connect mysql:jdbc://servername/database --username username --password password --table tablename --as-*filetyperequired* --target-dir *pathtofile* --delete-target-dir \*

## SQOOP IMPORT SPECIFIC COLUMNS

Sqoop import --connect mysql:jdbc://servername/database --username username --password password -m1 (or split by depending on performance definition) --table *tablename* --columns "columnlist" --target-dir *pathtofile* --delete-target-dir \*

## SQOOP EXPORT

Sqoop export --connect mysql:jdbc://servername/database --username username --password password --table tablename --export-dir *pathtofile*

# HIVE

1. [CREATING A TABLE FROM A FILE STORED IN HDFS](#_CREATING_A_TABLE)
2. [CREATING A PARTITIONED TABLE](#_CREATING_A_PARTITIONED)
3. [CREATE TABLE AS SELECT](#_CREATE_TABLE_AS)
4. [CREATE AVRO TABLE](#_CREATE_AVRO_TABLE)
5. [INSERT INTO TABLE](#_INSERT_INTO_TABLE)

Return To Start

1. [INSERT INTO PARTITIONED TABLE](#_INSERT_INTO_PARTITIONED)[DELETE TABLE](#_DELETE_TABLE)
2. [PARTITION ACTIVATOR](#_PARTITION_ACTIVATOR)

## CREATING A TABLE FROM A FILE STORED IN HDFS

create external\* table tablename (columns datatype)

stored as filetype (textfile, avro or parquet)

location '*/pathtofile*'

## CREATING A PARTITIONED TABLE

create external \* table *tablename* (columns datatype) partitioned by (column datatype)

row format delimited fields terminated by ‘*delimiter*’

stored as filetype

location ‘/*pathtofile*’

## CREATE TABLE AS SELECT

create table (Columns datatype)

row format delimited fields terminated by ‘*delimiter*’

stored as Filetype

location ‘pathtofile’

## CREATE AVRO TABLE

create external table *tablename*

row format delimited fields terminated by ‘*delimiter*’

stored as avro

location ‘pathtofile’

tblproperties (‘avro.schema.url’ = ‘pathtoschemafile.avsc’)

## INSERT INTO TABLE

Insert overwrite\* into table tablename (select query )

Return To Start

## INSERT INTO PARTITIONED TABLE

insert into table tablename (partition columnname)

select query (ensure the partitioned column is the last argument of the select query)

## DELETE TABLE

drop table tablename

## PARTITION ACTIVATOR

Set hive.exec.dynamic.partition=True

Set hive.exec.dynamic.partition.mode=Nonstrict

# COMMON HDFS COMMANDS FROM COMMAND PROMPT IN LINUX

1. [MAKING A DIRECTORY](#_MAKING_A_DIRECTORY)
2. [PUTTING A LOCAL FILE INTO HADOOP](#_PUTTING_A_LOCAL)
3. [EXTRACTING A SCHEMA FROM AN AVRO FILE](#_EXTRACTING_A_SCHEMA)
4. [CHANGING DIRECTORY](#_CHANGING_DIRECTORY)
5. [MOVING TO THE BEGINNING OF A COMMAND CODE](#_MOVING_TO_THE)
6. [LISTING THE COMMENTS OF A FOLDER](#_LISTING_THE_COMMENTS)
7. [MOVING A FILE](#_MOVING_A_FILE)

## MAKING A DIRECTORY

hdfs dfs –mkdir /pathtohdfsfile

## PUTTING A LOCAL FILE INTO HADOOP

hdfs dfs –get /pathtohdfsfile

## EXTRACTING A SCHEMA FROM AN AVRO FILE

avro-tools getschema filename

## CHANGING DIRECTORY

cd foldername

## MOVING TO THE BEGINNING OF A COMMAND CODE

CNTRL A

Return To Start

## LISTING THE COMMENTS OF A FOLDER

ls a\*

## MOVING A FILE

hdfs dfs –mv /currentfilepath /newdirectory

# SPARK RDD’s

1. [ACCESSING SPARK FROM COMMAND PROMPT IN LINUX](#_ACCESSING_SPARK_FROM)
2. [OPENING A LIST](#_OPENING_A_LIST)
3. [OPENING A TEXTFILE OR CSV FILE AS AN RDD](#_OPENING_A_TEXTFILE)
4. [MAPPING TRANSFORMATIONS](#_MAPPING_TRANSFORMATIONS)
5. [REDUCE BY KEY](#_REDUCE_BY_KEY)
6. [AGGREGATE BY KEY](#_AGGREGATE_BY_KEY)

## ACCESSING SPARK FROM COMMAND PROMPT IN LINUX

pyspark

## OPENING A LIST

var = sc.parrallelize(‘list’)

## OPENING A TEXTFILE OR CSV FILE AS AN RDD

var = sc.textFile(‘/pathtofile’)

## MAPPING TRANSFORMATIONS

.map(lambda x: transformation of x)

.flatMap(lambda x : transformation of c)

.mapValues(lambda x: algebraic expression of x)

Return To Start

## REDUCE BY KEY

.reduceByKey(lambda x,y : x+y)

Return To Start

## AGGREGATE BY KEY

.aggregateByKey((0,0), lambda x, y:(x[0] + y, x[1] + 1),lambda x, y: (x[0]+y[0], x[1]+x[1]))