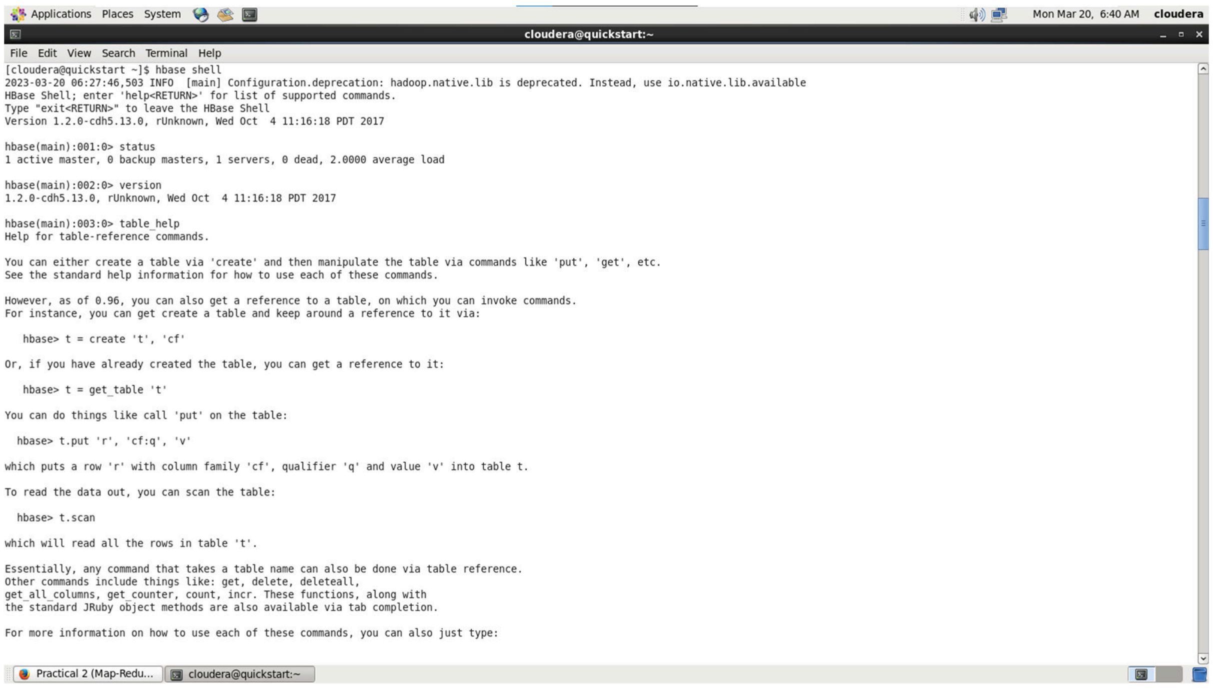
Practical 4: Install HBase and Use HBase Data Model to Store and Retrieve Databases

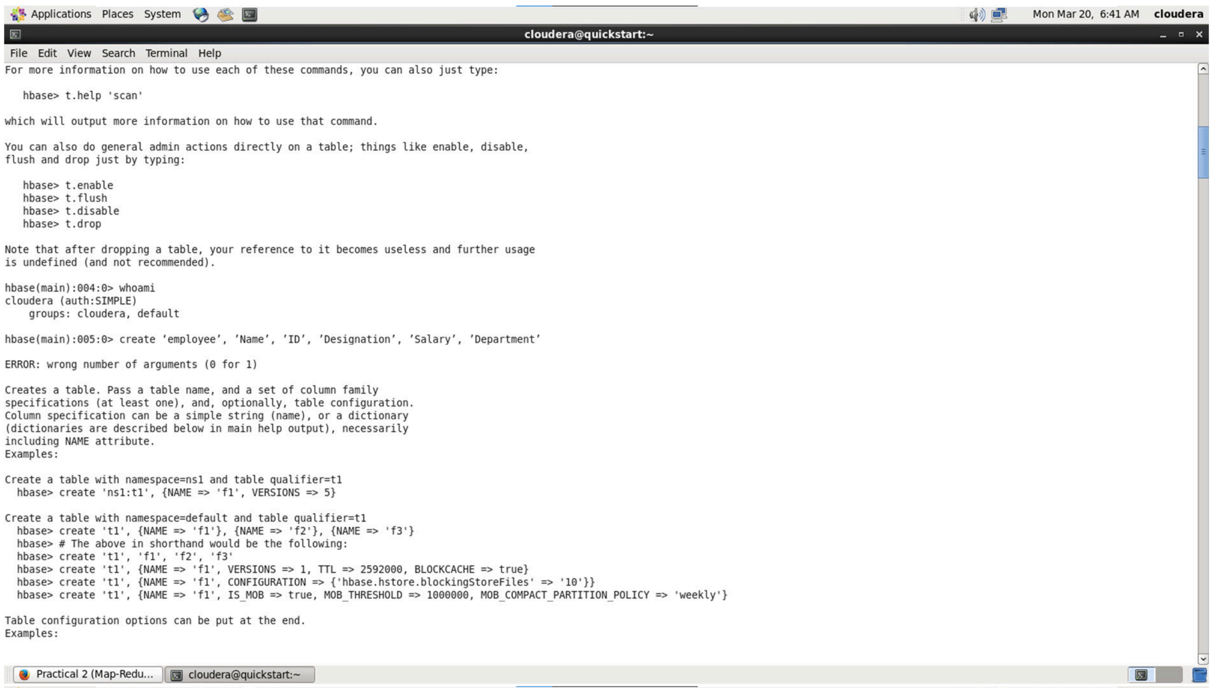
Commands:

```
hbase shell
status
version,
table_help
whoami
create ‘employee’, ’Name’, ’ID’, ’Designation’, ’Salary’, ’Department’
list
disable ‘employee’ (or is_disable ‘employee’)
scan ‘employee’
disable_all ‘e.*’
enable‘employee’ (or scan ‘is_enabled'employee')

//create new table
create‘student’, ‘name’, ‘age’, ‘course’
put ‘student’, ‘sharath’, ‘name:fullname’, ‘sharathkumar’
put ‘student’, ‘sharath’, ‘age:presentage’, ‘24’
put ‘student’, ‘sharath’, ‘course:pursuing’, ‘Hadoop’
put ‘student’, ‘shashank’, ‘name:fullname’, ‘shashank R
put ‘student’, ‘shashank’, ‘age:presentage’, ‘23’
put ‘student’, ‘shashank’, ‘course:pursuing’, ‘Java’

//Get Information
get ‘student’, ‘shashank’
get ‘student’, ‘sharath’
get ‘student’, ‘sharath’, ‘course’
get ‘student’, ‘shashank’, ‘course’
get ‘student’, ‘sharath’, ‘name’


scan ‘student’
Count ‘student’

//Alter
alter ‘student’, NAME=>’name’, VERSIONS=>5
put ‘student’, ‘shashank’, ‘name:fullname’, ‘shashank Rao’
scan ‘student’

//Delete
delete ‘student’, ‘shashank’, ‘name:fullname’
```

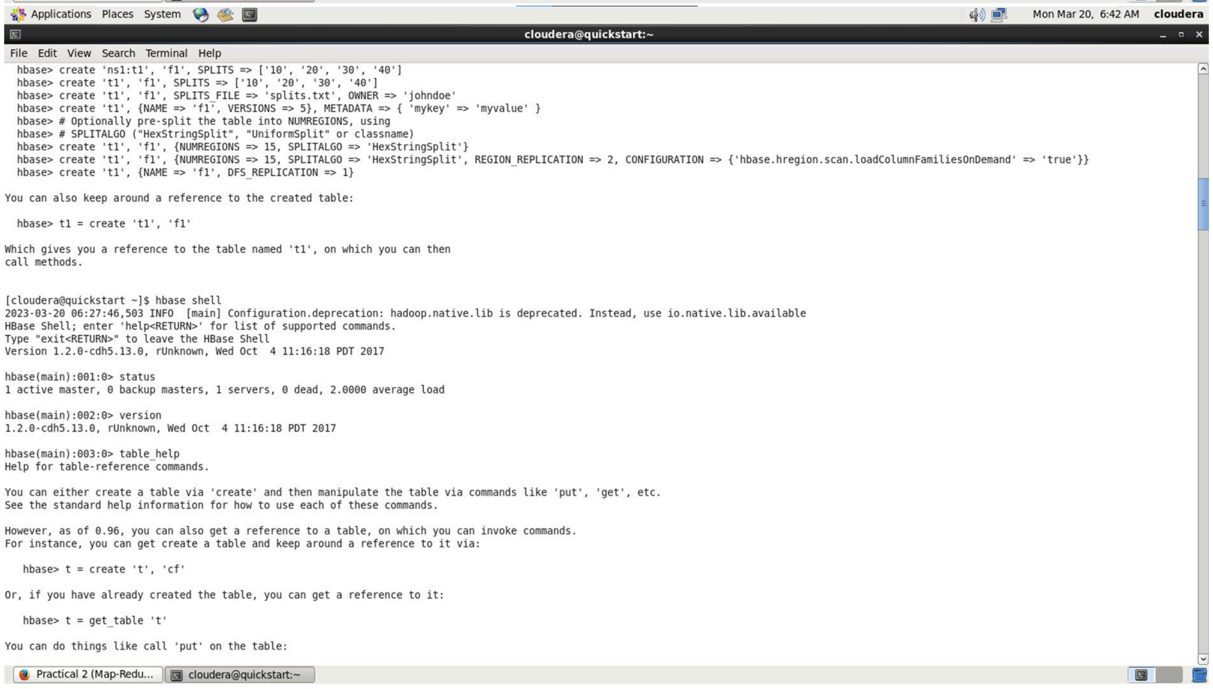
OUTPUT



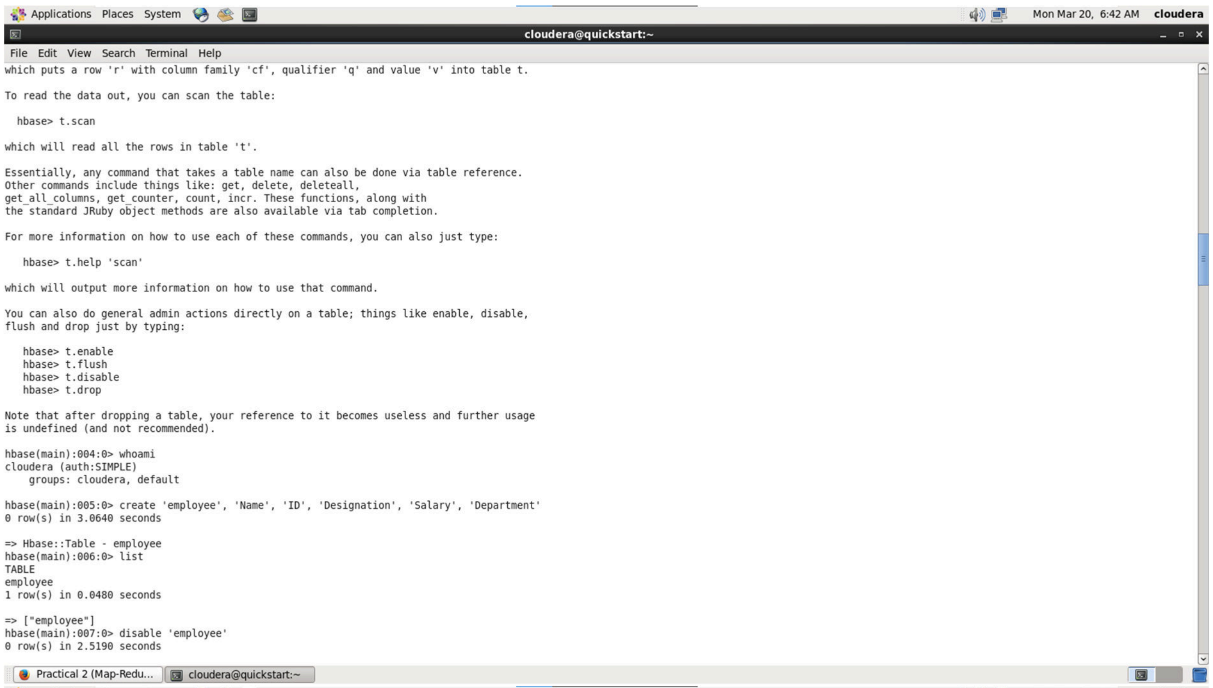
```
[cloudera@quickstart ~]$ hbase shell
2023-03-20 06:27:46,503 INFO  [main] Configuration.deprecation: hadoop.native.lib is deprecated. Instead, use io.native.lib.available
HBase Shell; enter 'help<RETURN>' for list of supported commands.
Type "exit<RETURN>" to leave the HBase Shell
Version 1.2.0-cdh5.13.0, rUnknown, Wed Oct  4 11:16:18 PDT 2017

hbase(main):001:0> status
1 active master, 0 backup masters, 1 servers, 0 dead, 2.0000 average load

hbase(main):002:0> version
1.2.0-cdh5.13.0, rUnknown, Wed Oct  4 11:16:18 PDT 2017

hbase(main):003:0> table_help
Help for table-reference commands.

You can either create a table via 'create' and then manipulate the table via commands like 'put', 'get', etc.
See the standard help information for how to use each of these commands.

However, as of 0.96, you can also get a reference to a table, on which you can invoke commands.
For instance, you can get create a table and keep around a reference to it via:

   hbase> t = create 't', 'cf'

Or, if you have already created the table, you can get a reference to it:

   hbase> t = get_table 't'

You can do things like call 'put' on the table:

  hbase> t.put 'r', 'cf:q', 'v'

which puts a row 'r' with column family 'cf', qualifier 'q' and value 'v' into table t.

To read the data out, you can scan the table:

  hbase> t.scan

which will read all the rows in table 't'.

Essentially, any command that takes a table name can also be done via table reference.
Other commands include things like: get, delete, deleteall,
get_all_columns, get_counter, count, incr. These functions, along with
the standard JRuby object methods are also available via tab completion.

For more information on how to use each of these commands, you can also just type:
```


Applications Places System
Mon Mar 20, 6:41 AM cloudera
cloudera@quickstart:~
File Edit View Search Terminal Help
Practical 2 (Map-Redu...
cloudera@quickstart:~

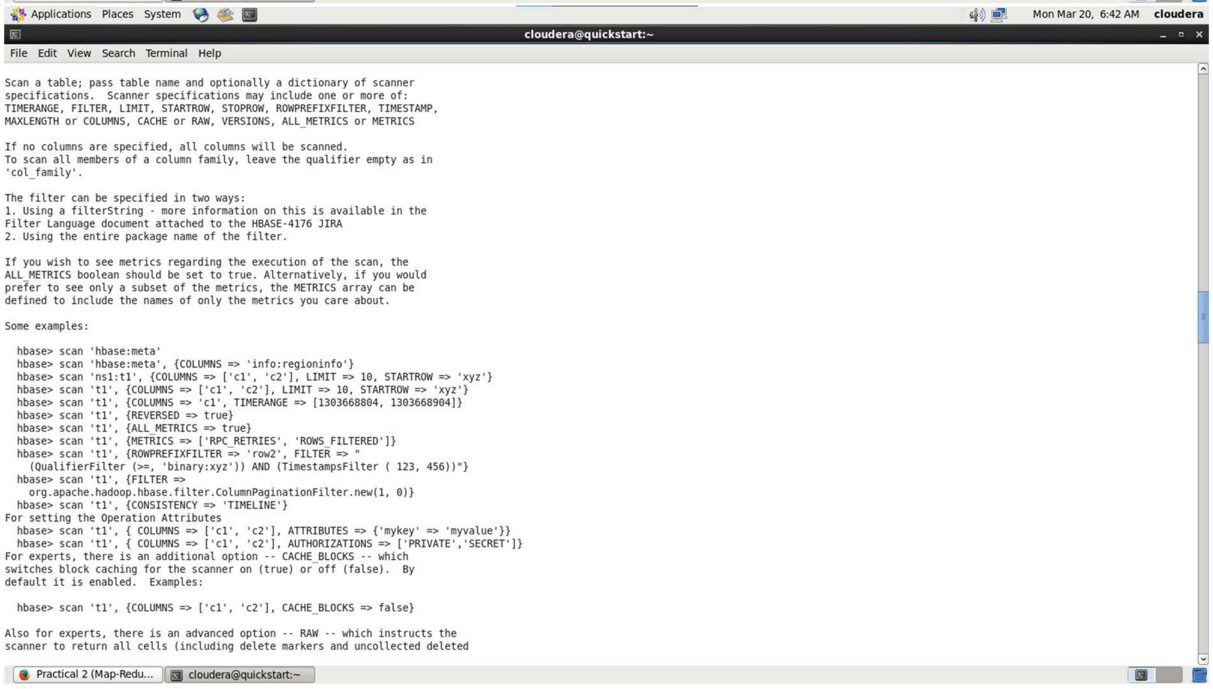
```
For more information on how to use each of these commands, you can also just type:

   hbase> t.help 'scan'

which will output more information on how to use that command.

You can also do general admin actions directly on a table; things like enable, disable,
flush and drop just by typing:

   hbase> t.enable
   hbase> t.flush
   hbase> t.disable
   hbase> t.drop

Note that after dropping a table, your reference to it becomes useless and further usage
is undefined (and not recommended).

hbase(main):004:0> whoami
cloudera (auth:SIMPLE)
    groups: cloudera, default

hbase(main):005:0> create ‘employee’, ’Name’, ’ID’, ’Designation’, ’Salary’, ’Department’

ERROR: wrong number of arguments (0 for 1)

Creates a table. Pass a table name, and a set of column family
specifications (at least one), and, optionally, table configuration.
Column specification can be a simple string (name), or a dictionary
(dictionaries are described below in main help output), necessarily
including NAME attribute.
Examples:

Create a table with namespace=ns1 and table qualifier=t1
  hbase> create 'ns1:t1', {NAME => 'f1', VERSIONS => 5}

Create a table with namespace=default and table qualifier=t1
  hbase> create 't1', {NAME => 'f1'}, {NAME => 'f2'}, {NAME => 'f3'}
  hbase> # The above in shorthand would be the following:
  hbase> create 't1', 'f1', 'f2', 'f3'
  hbase> create 't1', {NAME => 'f1', VERSIONS => 1, TTL => 2592000, BLOCKCACHE => true}
  hbase> create 't1', {NAME => 'f1', CONFIGURATION => {'hbase.hstore.blockingStoreFiles' => '10'}}
  hbase> create 't1', {NAME => 'f1', IS_MOB => true, MOB_THRESHOLD => 1000000, MOB_COMPACT_PARTITION_POLICY => 'weekly'}

Table configuration options can be put at the end.
Examples:
```


Applications Places System
Mon Mar 20, 6:42 AM cloudera
cloudera@quickstart:~
File Edit View Search Terminal Help
Practical 2 (Map-Redu...
cloudera@quickstart:~

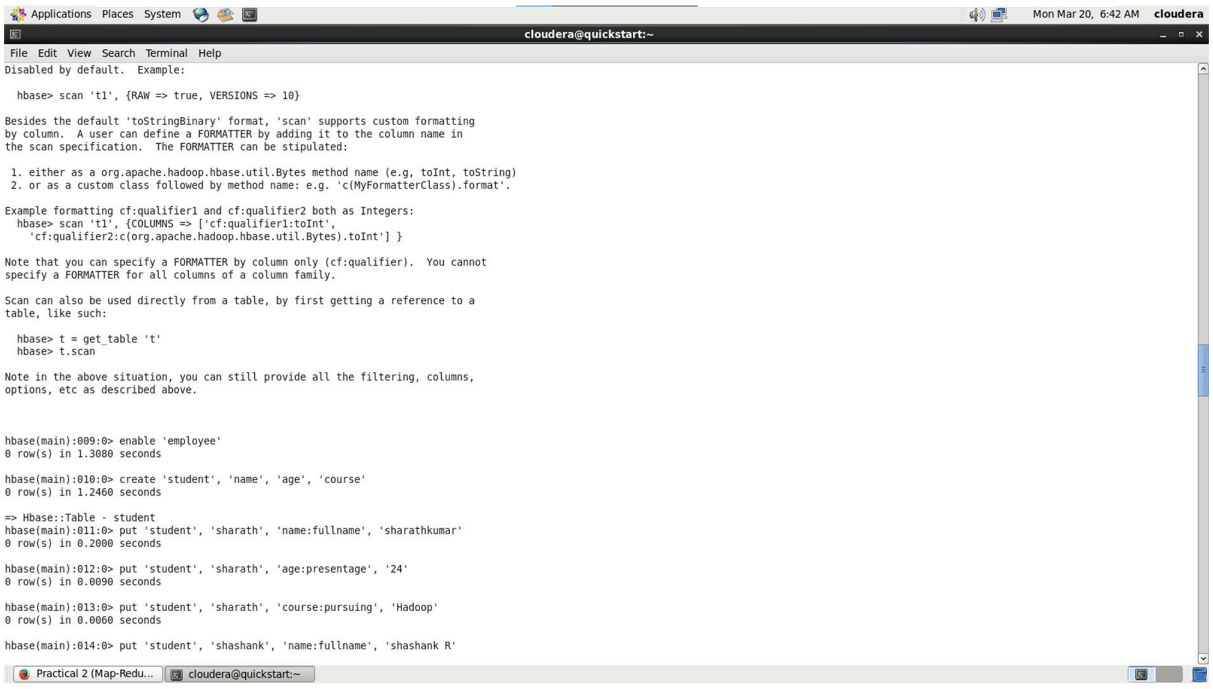
```
  hbase> create 'ns1:t1', 'f1', SPLITS => ['10', '20', '30', '40']
  hbase> create 't1', 'f1', SPLITS => ['10', '20', '30', '40']
  hbase> create 't1', 'f1', SPLITS_FILE => 'splits.txt', OWNER => 'johndoe'
  hbase> create 't1', {NAME => 'f1', VERSIONS => 5}, METADATA => { 'mykey' => 'myvalue' }
  hbase> # Optionally pre-split the table into NUMREGIONS, using
  hbase> # SPLITALGO ("HexStringSplit", "UniformSplit" or classname)
  hbase> create 't1', 'f1', {NUMREGIONS => 15, SPLITALGO => 'HexStringSplit'}
  hbase> create 't1', 'f1', {NUMREGIONS => 15, SPLITALGO => 'HexStringSplit', REGION_REPLICATION => 2, CONFIGURATION => {'hbase.hregion.scan.loadColumnFamiliesOnDemand' => 'true'}}
  hbase> create 't1', {NAME => 'f1', DFS_REPLICATION => 1}

You can also keep around a reference to the created table:

  hbase> t1 = create 't1', 'f1'

Which gives you a reference to the table named 't1', on which you can then
call methods.


[cloudera@quickstart ~]$ hbase shell
2023-03-20 06:27:46,503 INFO  [main] Configuration.deprecation: hadoop.native.lib is deprecated. Instead, use io.native.lib.available
HBase Shell; enter 'help<RETURN>' for list of supported commands.
Type "exit<RETURN>" to leave the HBase Shell
Version 1.2.0-cdh5.13.0, rUnknown, Wed Oct  4 11:16:18 PDT 2017

hbase(main):001:0> status
1 active master, 0 backup masters, 1 servers, 0 dead, 2.0000 average load

hbase(main):002:0> version
1.2.0-cdh5.13.0, rUnknown, Wed Oct  4 11:16:18 PDT 2017

hbase(main):003:0> table_help
Help for table-reference commands.

You can either create a table via 'create' and then manipulate the table via commands like 'put', 'get', etc.
See the standard help information for how to use each of these commands.

However, as of 0.96, you can also get a reference to a table, on which you can invoke commands.
For instance, you can get create a table and keep around a reference to it via:

   hbase> t = create 't', 'cf'

Or, if you have already created the table, you can get a reference to it:

   hbase> t = get_table 't'

You can do things like call 'put' on the table:
```


Applications Places System
Mon Mar 20, 6:42 AM cloudera
cloudera@quickstart:~
File Edit View Search Terminal Help
Practical 2 (Map-Redu...
cloudera@quickstart:~

```
which puts a row 'r' with column family 'cf', qualifier 'q' and value 'v' into table t.

To read the data out, you can scan the table:

  hbase> t.scan

which will read all the rows in table 't'.

Essentially, any command that takes a table name can also be done via table reference.
Other commands include things like: get, delete, deleteall,
get_all_columns, get_counter, count, incr. These functions, along with
the standard JRuby object methods are also available via tab completion.

For more information on how to use each of these commands, you can also just type:

   hbase> t.help 'scan'

which will output more information on how to use that command.

You can also do general admin actions directly on a table; things like enable, disable,
flush and drop just by typing:

   hbase> t.enable
   hbase> t.flush
   hbase> t.disable
   hbase> t.drop

Note that after dropping a table, your reference to it becomes useless and further usage
is undefined (and not recommended).

hbase(main):004:0> whoami
cloudera (auth:SIMPLE)
    groups: cloudera, default

hbase(main):005:0> create 'employee', 'Name', 'ID', 'Designation', 'Salary', 'Department'
0 row(s) in 3.0640 seconds

=> Hbase::Table - employee
hbase(main):006:0> list
TABLE
employee
1 row(s) in 0.0480 seconds

=> ["employee"]
hbase(main):007:0> disable 'employee'
0 row(s) in 2.5190 seconds
```

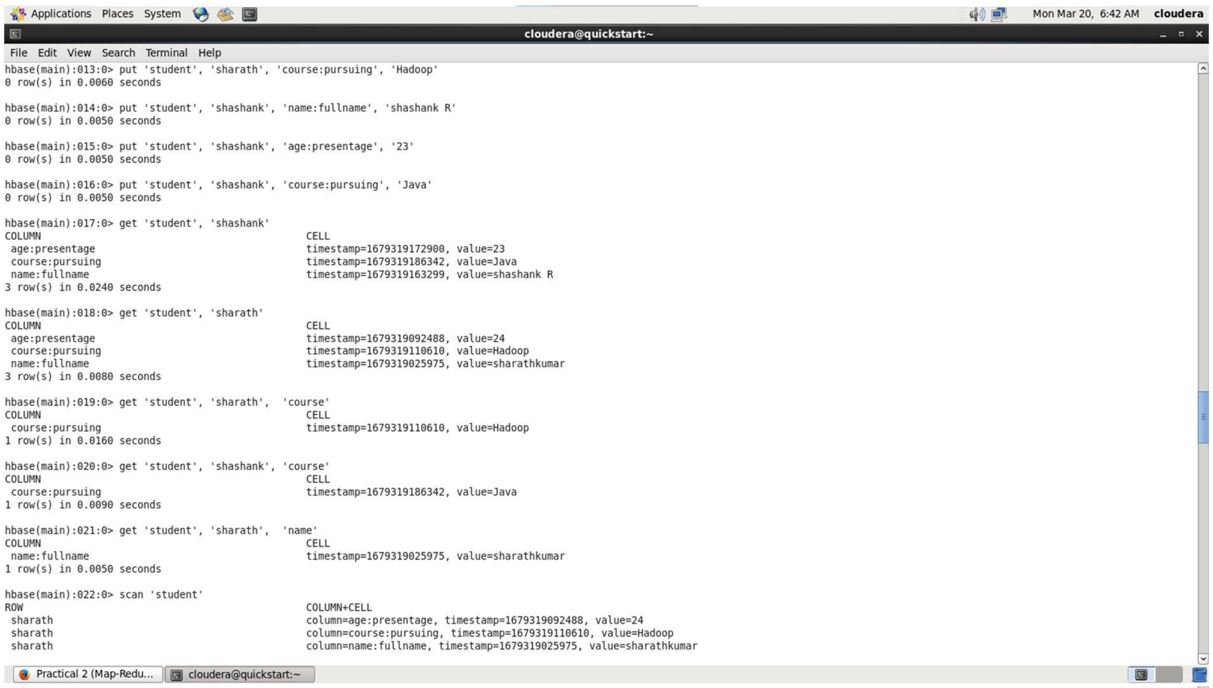

Applications Places System
Mon Mar 20, 6:42 AM cloudera
cloudera@quickstart:~
File Edit View Search Terminal Help
Practical 2 (Map-Redu...
cloudera@quickstart:~

```
Scan a table; pass table name and optionally a dictionary of scanner
specifications.  Scanner specifications may include one or more of:
TIMERANGE, FILTER, LIMIT, STARTROW, STOPROW, ROWPREFIXFILTER, TIMESTAMP,
MAXLENGTH or COLUMNS, CACHE or RAW, VERSIONS, ALL_METRICS or METRICS

If no columns are specified, all columns will be scanned.
To scan all members of a column family, leave the qualifier empty as in
'col_family'.

The filter can be specified in two ways:
1. Using a filterString - more information on this is available in the
Filter Language document attached to the HBASE-4176 JIRA
2. Using the entire package name of the filter.

If you wish to see metrics regarding the execution of the scan, the
ALL_METRICS boolean should be set to true. Alternatively, if you would
prefer to see only a subset of the metrics, the METRICS array can be
defined to include the names of only the metrics you care about.

Some examples:

  hbase> scan 'hbase:meta'
  hbase> scan 'hbase:meta', {COLUMNS => 'info:regioninfo'}
  hbase> scan 'ns1:t1', {COLUMNS => ['c1', 'c2'], LIMIT => 10, STARTROW => 'xyz'}
  hbase> scan 't1', {COLUMNS => ['c1', 'c2'], LIMIT => 10, STARTROW => 'xyz'}
  hbase> scan 't1', {COLUMNS => 'c1', TIMERANGE => [1303668804, 1303668904]}
  hbase> scan 't1', {REVERSED => true}
  hbase> scan 't1', {ALL_METRICS => true}
  hbase> scan 't1', {METRICS => ['RPC_RETRIES', 'ROWS_FILTERED']}
  hbase> scan 't1', {ROWPREFIXFILTER => 'row2', FILTER => "
    (QualifierFilter (>=, 'binary:xyz')) AND (TimestampsFilter ( 123, 456))"}
  hbase> scan 't1', {FILTER =>
    org.apache.hadoop.hbase.filter.ColumnPaginationFilter.new(1, 0)}
  hbase> scan 't1', {CONSISTENCY => 'TIMELINE'}
For setting the Operation Attributes
  hbase> scan 't1', { COLUMNS => ['c1', 'c2'], ATTRIBUTES => {'mykey' => 'myvalue'}}
  hbase> scan 't1', { COLUMNS => ['c1', 'c2'], AUTHORIZATIONS => ['PRIVATE','SECRET']}
For experts, there is an additional option -- CACHE_BLOCKS -- which
switches block caching for the scanner on (true) or off (false).  By
default it is enabled.  Examples:

  hbase> scan 't1', {COLUMNS => ['c1', 'c2'], CACHE_BLOCKS => false}

Also for experts, there is an advanced option -- RAW -- which instructs the
scanner to return all cells (including delete markers and uncollected deleted
```



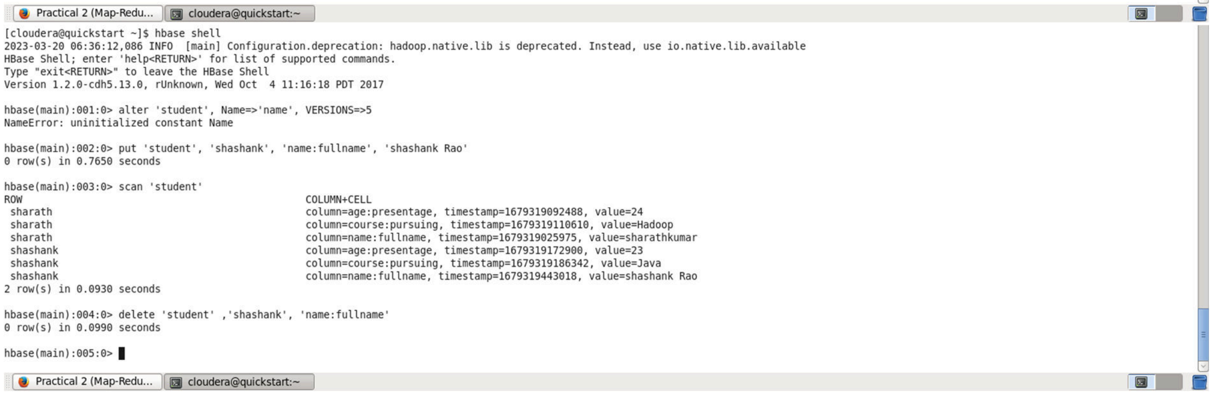
Applications Places System
Mon Mar 20, 6:42 AM cloudera
cloudera@quickstart:~
File Edit View Search Terminal Help
Practical 2 (Map-Redu...
cloudera@quickstart:~

```
Disabled by default.  Example:

  hbase> scan 't1', {RAW => true, VERSIONS => 10}

Besides the default 'toStringBinary' format, 'scan' supports custom formatting
by column.  A user can define a FORMATTER by adding it to the column name in
the scan specification.  The FORMATTER can be stipulated:

 1. either as a org.apache.hadoop.hbase.util.Bytes method name (e.g, toInt, toString)
 2. or as a custom class followed by method name: e.g. 'c(MyFormatterClass).format'.

Example formatting cf:qualifier1 and cf:qualifier2 both as Integers:
  hbase> scan 't1', {COLUMNS => ['cf:qualifier1:toInt',
    'cf:qualifier2:c(org.apache.hadoop.hbase.util.Bytes).toInt'] }

Note that you can specify a FORMATTER by column only (cf:qualifier).  You cannot
specify a FORMATTER for all columns of a column family.

Scan can also be used directly from a table, by first getting a reference to a
table, like such:

  hbase> t = get_table 't'
  hbase> t.scan

Note in the above situation, you can still provide all the filtering, columns,
options, etc as described above.



hbase(main):009:0> enable 'employee'
0 row(s) in 1.3080 seconds

hbase(main):010:0> create 'student', 'name', 'age', 'course'
0 row(s) in 1.2460 seconds

=> Hbase::Table - student
hbase(main):011:0> put 'student', 'sharath', 'name:fullname', 'sharathkumar'
0 row(s) in 0.2000 seconds

hbase(main):012:0> put 'student', 'sharath', 'age:presentage', '24'
0 row(s) in 0.0090 seconds

hbase(main):013:0> put 'student', 'sharath', 'course:pursuing', 'Hadoop'
0 row(s) in 0.0060 seconds

hbase(main):014:0> put 'student', 'shashank', 'name:fullname', 'shashank R'
```



Applications Places System
Mon Mar 20, 6:42 AM cloudera
cloudera@quickstart:~
File Edit View Search Terminal Help
Practical 2 (Map-Redu...
cloudera@quickstart:~

```
hbase(main):013:0> put 'student', 'sharath', 'course:pursuing', 'Hadoop'
0 row(s) in 0.0060 seconds

hbase(main):014:0> put 'student', 'shashank', 'name:fullname', 'shashank R'
0 row(s) in 0.0050 seconds

hbase(main):015:0> put 'student', 'shashank', 'age:presentage', '23'
0 row(s) in 0.0050 seconds

hbase(main):016:0> put 'student', 'shashank', 'course:pursuing', 'Java'
0 row(s) in 0.0050 seconds

hbase(main):017:0> get 'student', 'shashank'
COLUMN                                             CELL
 age:presentage                                    timestamp=1679319172900, value=23
 course:pursuing                                   timestamp=1679319186342, value=Java
 name:fullname                                     timestamp=1679319163299, value=shashank R
3 row(s) in 0.0240 seconds

hbase(main):018:0> get 'student', 'sharath'
COLUMN                                             CELL
 age:presentage                                    timestamp=1679319092488, value=24
 course:pursuing                                   timestamp=1679319110610, value=Hadoop
 name:fullname                                     timestamp=1679319025975, value=sharathkumar
3 row(s) in 0.0080 seconds

hbase(main):019:0> get 'student', 'sharath',  'course'
COLUMN                                             CELL
 course:pursuing                                   timestamp=1679319110610, value=Hadoop
1 row(s) in 0.0160 seconds

hbase(main):020:0> get 'student', 'shashank', 'course'
COLUMN                                             CELL
 course:pursuing                                   timestamp=1679319186342, value=Java
1 row(s) in 0.0090 seconds

hbase(main):021:0> get 'student', 'sharath',  'name'
COLUMN                                             CELL
 name:fullname                                     timestamp=1679319025975, value=sharathkumar
1 row(s) in 0.0050 seconds

hbase(main):022:0> scan 'student'
ROW                                                COLUMN+CELL
 sharath                                           column=age:presentage, timestamp=1679319092488, value=24
 sharath                                           column=course:pursuing, timestamp=1679319110610, value=Hadoop
 sharath                                           column=name:fullname, timestamp=1679319025975, value=sharathkumar
```



```
[cloudera@quickstart ~]$ hbase shell
2023-03-20 06:36:12,086 INFO  [main] Configuration.deprecation: hadoop.native.lib is deprecated. Instead, use io.native.lib.available
HBase Shell; enter 'help<RETURN>' for list of supported commands.
Type "exit<RETURN>" to leave the HBase Shell
Version 1.2.0-cdh5.13.0, rUnknown, Wed Oct  4 11:16:18 PDT 2017

hbase(main):001:0> alter 'student', Name=>'name', VERSIONS=>5
NameError: uninitialized constant Name

hbase(main):002:0> put 'student', 'shashank', 'name:fullname', 'shashank Rao'
0 row(s) in 0.7650 seconds

hbase(main):003:0> scan 'student'
ROW                                                COLUMN+CELL
 sharath                                           column=age:presentage, timestamp=1679319092488, value=24
 sharath                                           column=course:pursuing, timestamp=1679319110610, value=Hadoop
 sharath                                           column=name:fullname, timestamp=1679319025975, value=sharathkumar
 shashank                                          column=age:presentage, timestamp=1679319172900, value=23
 shashank                                          column=course:pursuing, timestamp=1679319186342, value=Java
 shashank                                          column=name:fullname, timestamp=1679319443018, value=shashank Rao
2 row(s) in 0.0930 seconds

hbase(main):004:0> delete 'student' ,'shashank', 'name:fullname'
0 row(s) in 0.0990 seconds

hbase(main):005:0> 
```