HBase Shell Commands

- Enter into your HBase directory and then access the hbase shell as you have seen in the video.

```
[root@ip-172-31-36-143 HBase]# hbase shell
HBase Shell
Use "help" to get list of supported commands.
Use "exit" to quit this interactive shell.
Version 1.4.13, rUnknown, Fri Apr 17 15:18:24 UTC 2020

hbase(main):001:0>
```

General Commands

- **Status Command-** This command shows the status of the cluster. It shows the number of servers present in the cluster, active server count and average load value.
  It can be 'simple', 'summary' or 'detailed'. The default is summary.
  Syntax: `status`

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

hbase(main):002:0> status 'simple'
active master:  ip-172-31-36-143.ec2.internal:16000 1627561471926
0 backup masters
1 live servers
    ip-172-31-36-143.ec2.internal:16020 1627561470227
        requestsPerSecond=0.0, numberOfOnlineRegions=2, usedHeapMB=33, maxHeapMB=6110, numberOfStores=2, numberOfS
torefiles=2, storefileUncompressedSizeMB=0, storefileSizeMB=0, memstoreSizeMB=0, storefileIndexSizeMB=0, readReque
stsCount=5, writeRequestsCount=4, rootIndexSizeKB=0, totalStaticIndexSizeKB=0, totalStaticBloomSizeKB=0, totalComp
actingKVs=0, currentCompactedKVs=0, compactionProgressPct=NaN, coprocessors=[MultiRowMutationEndpoint]
0 dead servers
Aggregate load: 0, regions: 2

hbase(main):003:0> status 'detailed'
version 1.4.13
0 regionsInTransition
active master:  ip-172-31-36-143.ec2.internal:16000 1627561471926
0 backup masters
master coprocessors: null
1 live servers
    ip-172-31-36-143.ec2.internal:16020 1627561470227
        requestsPerSecond=0.0, numberOfOnlineRegions=2, usedHeapMB=34, maxHeapMB=6110, numberOfStores=2, numberOfS
torefiles=2, storefileUncompressedSizeMB=0, storefileSizeMB=0, memstoreSizeMB=0, storefileIndexSizeMB=0, readReque
stsCount=5, writeRequestsCount=4, rootIndexSizeKB=0, totalStaticIndexSizeKB=0, totalStaticBloomSizeKB=0, totalComp
actingKVs=0, currentCompactedKVs=0, compactionProgressPct=NaN, coprocessors=[MultiRowMutationEndpoint]
        "hbase:meta,,1"
            numberOfStores=1, numberOfStorefiles=1, storefileUncompressedSizeMB=0, lastMajorCompactionTimestamp=0,
 storefileSizeMB=0, memstoreSizeMB=0, storefileIndexSizeMB=0, readRequestsCount=3, writeRequestsCount=2, rootIndex
SizeKB=0, totalStaticIndexSizeKB=0, totalStaticBloomSizeKB=0, totalCompactingKVs=0, currentCompactedKVs=0, compact
```

- **Version Command-** This command will display the currently used version of HBase.
  Syntax: `version`

```
hbase(main):004:0> version
1.4.13, rUnknown, Fri Apr 17 15:18:24 UTC 2020

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

- **Table Help Command-** This command provides a guide on how to use table referenced commands. It provides usage and syntax of various HBase shell commands.
  Syntax: `table_help`

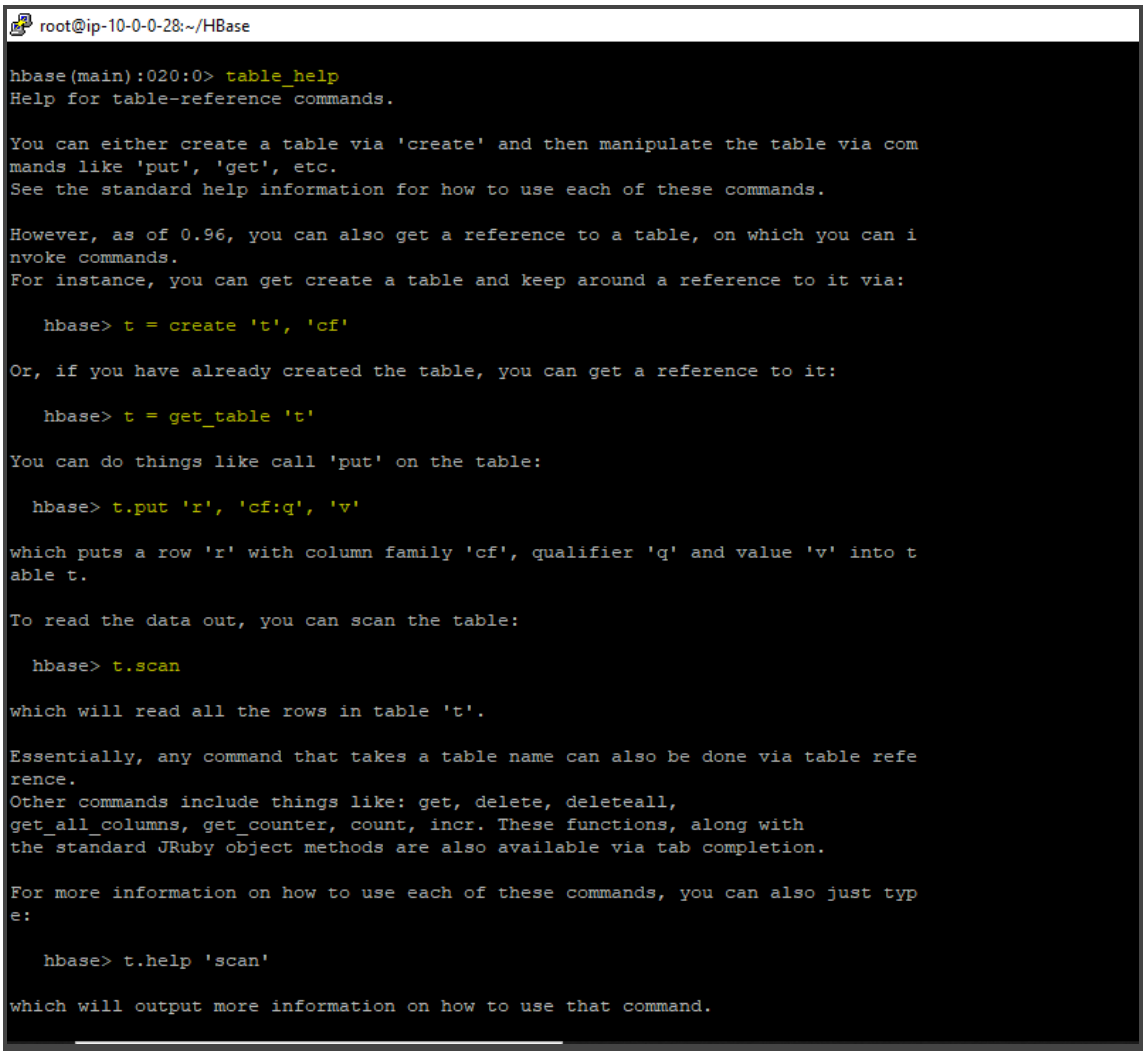

Table Management Commands

- **Create:** This command allows you to create a new table in HBase with a specified name.
  Syntax: `create '<table_name>', '<column_family_name>'`

```
hbase(main):022:0> create 'employee','personal data'
0 row(s) in 4.3560 seconds

=> Hbase::Table - employee
hbase(main):023:0>
```

- **List-** This command will display all the tables that are present in the HBase. This command is used to check whether your table is created or not.
  Syntax: `list`

```
hbase(main):023:0> list
TABLE
employee
1 row(s) in 0.0200 seconds

=> ["employee"]
hbase(main):024:0>
```

- **Describe-** This command gives the information about the column families of the mentioned table.
  Syntax: `describe '<table name>'`

```
hbase(main):025:0> describe 'employee'
Table employee is ENABLED
employee
COLUMN FAMILIES DESCRIPTION
{NAME => 'personal data', BLOOMFILTER => 'ROW', VERSIONS => '1', IN_MEMORY => 'f
OMPRESSION => 'NONE', MIN_VERSIONS => '0', BLOCKCACHE => 'true', BLOCKSIZE => '6
1 row(s) in 0.1340 seconds

hbase(main):026:0>
```

- **Disable-** This command disables the mentioned table. Tables need to be disabled before deleting or dropping.
  Syntax: `disable '<tablename>'`

Disable All- This command disables all tables matching the mentioned regex.
Syntax: `disable_all <"matching regex">`

```
hbase(main):026:0> disable 'employee'
0 row(s) in 2.2790 seconds

hbase(main):027:0>
```

- **Enable-** This command enables the mentioned table. Whichever table is disabled, to retrieve back to its previous state this command is used.
  Syntax: `enable '<tablename>'`

  Enable All- This command will enable all tables matching the mentioned regex.
  Syntax: `enable_all <"matching regex">`

```
hbase(main):027:0> enable 'employee'
0 row(s) in 1.2670 seconds

hbase(main):028:0>
```

- **Is Enabled-** This command verifies whether the mentioned table is enabled or not.
  Syntax: `is_enabled '<table_name>'`

```
hbase(main):002:0> is_enabled 'employee'
true
0 row(s) in 0.1130 seconds

hbase(main):003:0>
```

Is Disabled- This command verifies whether the mentioned table is disabled or not.
Syntax: `is_disabled '<table_name>'`

```
hbase(main):003:0> is_disabled 'employee'
false
0 row(s) in 0.0210 seconds

hbase(main):004:0>
```

- **Exists-** This command is used to verify whether a given table is present in the HBase storage or not.
  Syntax: `exists '<table_name>'`

```
hbase(main):031:0> exists 'employee'
Table employee does exist
0 row(s) in 0.0170 seconds

hbase(main):032:0>
```

- **Alter-** This command is used to alter the column family schema. This command can be used for operations like adding column families, updating the version number of column families. It can also be used to delete a column family by applying the delete method to it.

  Syntax:

  ```
  alter '<tablename>', NAME=>'<column familyname>', VERSIONS=><Number>
  ```

  or

  ```
  alter '<tablename>', 'delete'=> '<column familyname>'
  ```

  or

  ```
  alter '<tablename>',NAME=>'<column familyname>',METHOD=> 'delete'
  ```

  Following operations can be performed using alter command:

  - Add new column family to the mentioned table. Use describe command to check the updated status. Alter command can be used on multiple column families as well.

```
hbase(main):005:0> alter 'employee',NAME=>'Contact',VERSIONS=>2
Updating all regions with the new schema...
0/1 regions updated.
1/1 regions updated.
Done.
0 row(s) in 3.4840 seconds

hbase(main):006:0> describe 'employee'
Table employee is ENABLED
employee
COLUMN FAMILIES DESCRIPTION
{NAME => 'Contact', BLOOMFILTER => 'ROW', VERSIONS => '2', IN_MEMORY => 'false',
 KEEP_DELETED_CELLS => 'FALSE', DATA_BLOCK_ENCODING => 'NONE', TTL => 'FOREVER',
 COMPRESSION => 'NONE', MIN_VERSIONS => '0', BLOCKCACHE => 'true', BLOCKSIZE =>
'65536', REPLICATION_SCOPE => '0'}
{NAME => 'personal data', BLOOMFILTER => 'ROW', VERSIONS => '1', IN_MEMORY => 'f
alse', KEEP_DELETED_CELLS => 'FALSE', DATA_BLOCK_ENCODING => 'NONE', TTL => 'FOR
EVER', COMPRESSION => 'NONE', MIN_VERSIONS => '0', BLOCKCACHE => 'true', BLOCKSI
ZE => '65536', REPLICATION_SCOPE => '0'}
2 row(s) in 0.0310 seconds

hbase(main):007:0>
```

  - Deleting column family names from the table

```
hbase(main):010:0> alter 'employee','delete'=>'Contact'
Updating all regions with the new schema...
0/1 regions updated.
1/1 regions updated.
Done.
0 row(s) in 3.1590 seconds

hbase(main):011:0> describe 'employee'
Table employee is ENABLED
employee
COLUMN FAMILIES DESCRIPTION
{NAME => 'personal data', BLOOMFILTER => 'ROW', VERSIONS => '1', IN_MEMORY => 'f
alse', KEEP_DELETED_CELLS => 'FALSE', DATA_BLOCK_ENCODING => 'NONE', TTL => 'FOR
EVER', COMPRESSION => 'NONE', MIN_VERSIONS => '0', BLOCKCACHE => 'true', BLOCKSI
ZE => '65536', REPLICATION_SCOPE => '0'}
1 row(s) in 0.0200 seconds

hbase(main):012:0>
```

- **Alter Status-** This command gives the status of the alter command. It indicates the number of regions of the table that have received the updated schema.
  Syntax: `alter_status '<table_name>'`

```
hbase(main):012:0> alter_status 'employee'
1/1 regions updated.
Done.

hbase(main):013:0>
```

- **Drop-** This command drops the mentioned table. To drop a table it should be disabled first.
  Syntax: `drop '<table name>'`

  Drop All- This command drops all the tables matching the given regex.
  Syntax: `drop_all<"regex">`

```
hbase(main):028:0> disable 'employee'
0 row(s) in 2.2460 seconds

hbase(main):029:0> drop 'employee'
0 row(s) in 1.2530 seconds

hbase(main):030:0>
```

Data Manipulation Commands

- **Put-** This command is used to add a cell value in the mentioned table. Single put command can add a single cell value to the table.
  Syntax: `put '<table_name>', '<row_key>', '<column_value>', '<value>'`

```
hbase(main):013:0> put 'employee','employee1','personal data:Name','John'
0 row(s) in 0.0970 seconds

hbase(main):014:0> put 'employee','employee1','personal data:Age',25
0 row(s) in 0.0100 seconds
```

- **Get-** This command is used to fetch data from the HBase table.
  Syntax: `get '<table_name>', '<row_key>', {'<Additional Parameters>'}`

  Additional Parameters here include TIMERANGE, TIMESTAMP, VERSIONS and FILTERS.

```
hbase(main):016:0> get 'employee','employee1'
COLUMN                CELL
 personal data:Age    timestamp=1585553085267, value=25
 personal data:Name   timestamp=1585553067675, value=John
2 row(s) in 0.0160 seconds

hbase(main):017:0>
```

- **Get data based on the timestamp**

Syntax:

```
get '<table_name>', '<row_key>', {COLUMN => '<column_family_name>', TIMESTAMP => value}
```

```
hbase(main):004:0> scan 'employee'
ROW                   COLUMN+CELL
 employee1            column=personal data:Age, timestamp=1585669442476, value=2
                      5
 employee1            column=personal data:Name, timestamp=1585669428176, value=
                      John
1 row(s) in 0.0450 seconds

hbase(main):005:0> get 'employee','employee1',{COLUMN=>'personal data',TIMESTAMP
=>1585669428176}
COLUMN                CELL
 personal data:Name   timestamp=1585669428176, value=John
1 row(s) in 0.0150 seconds

hbase(main):006:0>
```

- **Count-** This command will return the number of rows present in the table.
  Syntax: `count '<table_name>'`

```
hbase(main):017:0> count 'employee'
1 row(s) in 0.0270 seconds

=> 1
hbase(main):018:0>
```

- **Delete-** This command is used to delete cell value in the mentioned table at the specified row or column.
  Syntax:
  `delete '<table_name>', '<row_key>', '<column_value>', <timestamp_value>`
  Note- timestamp_value is optional

```
hbase(main):022:0> delete 'employee','employee1','personal data:Age'
0 row(s) in 0.0040 seconds

hbase(main):023:0> get 'employee','employee1'
COLUMN                CELL
 personal data:Name   timestamp=1585553067675, value=John
1 row(s) in 0.0030 seconds

hbase(main):024:0>
```

- **Scan-** This command is used to view all the contents of the table created.

Syntax: `scan '<table_name>' {Optional parameters}`
The optional parameters in syntax include TIMERANGE, FILTER, TIMESTAMP, LIMIT, MAXLENGTH, COLUMNS, CACHE, STARTROW and STOPROW.

```
hbase(main):024:0> scan 'employee'
ROW                   COLUMN+CELL
 employee1            column=personal data:Name, timestamp=1585553067675, value=
                      John
1 row(s) in 0.0810 seconds

hbase(main):025:0>
```

- **Get data based on filters:** In HBase, fetching data based on a filtering condition is achieved by using Filters. In HBase, filters are like java methods which take two input parameters that are, a logical operator and a comparator. The logical operator specifies the type of the test, i.e. equals, less than, etc. The comparator is the number/value against which you wish to compare your record. Some commonly used filter functions are:
    - **Value Filter:** A ValueFilter takes a comparison operator and a comparator as the parameter. It compares each value with the comparator using the comparison operator. If the check is true, then the result is displayed on the console.
      Syntax:
      `"ValueFilter(<compareOp>, '<value_comparator>')"`

      Example- To fetch all details of all the employees having age greater than or equal to 25.

```
hbase(main):018:0> scan  'employee',{FILTER=>"ValueFilter(>=,'binary:25')"}
ROW                   COLUMN+CELL
 employee1            column=personal data:Age, timestamp=1585669442476, value=2
                      5
 employee1            column=personal data:Name, timestamp=1585669428176, value=
                      John
 employee2            column=personal data:Age, timestamp=1585669947952, value=2
                      6
 employee3            column=personal data:Name, timestamp=1585670594166, value=
                      Sam
 employee4            column=personal data:Age, timestamp=1585669983075, value=2
                      9
 employee5            column=personal data:Age, timestamp=1585670004136, value=3
                      2
5 row(s) in 0.0180 seconds
```

- **Qualifier Filter:** A QualifierFilter also takes two parameters: comparison operator and comparator. Each qualifier name is compared with the comparator using the compare operator, and if the comparison is true, it returns the key-values in that column.
  Syntax:
  `"QualifierFilter(<compareOp>, '<qualifier_comparator>')"`
  Example- Fetch the names of the employees from the table.

```
hbase(main):019:0> scan  'employee',{FILTER=>"QualifierFilter(=,'substring:Name'
)"}
ROW                   COLUMN+CELL
 employee1            column=personal data:Name, timestamp=1585669428176, value=
                      John
 employee3            column=personal data:Name, timestamp=1585670594166, value=
                      Sam
2 row(s) in 0.0170 seconds

hbase(main):020:0>
```

- **Family Filter:** A FamilyFilter is used to fetch key-values for a specified column family.
  Syntax:
  `"FamilyFilter(<compareOp>, '<family_comparator>')"`
  Example- Fetch data from a column family.

```
hbase(main):001:0> scan 'employee',{FILTER=>"FamilyFilter(=,'substring:personal
data')"}
ROW                   COLUMN+CELL
 employee1            column=personal data:Age, timestamp=1585669442476, value=2
                      5
 employee1            column=personal data:Name, timestamp=1585669428176, value=
                      John
 employee2            column=personal data:Age, timestamp=1585669947952, value=2
                      6
 employee3            column=personal data:Age, timestamp=1585669967387, value=2
                      0
 employee3            column=personal data:Name, timestamp=1585670594166, value=
                      Sam
 employee4            column=personal data:Age, timestamp=1585669983075, value=2
                      9
 employee5            column=personal data:Age, timestamp=1585670004136, value=3
                      2
5 row(s) in 0.3080 seconds

hbase(main):002:0>
```

- **Delete All-** This command will delete all the cells of the mentioned row.
  Syntax: `deleteall '<tablename>', '<rowkey>'`

```
hbase(main):027:0> deleteall 'employee','employee1'
0 row(s) in 0.0060 seconds

hbase(main):028:0> get 'employee','employee1'
COLUMN                CELL
0 row(s) in 0.0030 seconds

hbase(main):029:0>
```

- **Truncate-** This command deletes all the data from the table. This command performs three tasks- disable the table, drops it and then recreates it.
  Syntax: `truncate '<table_name>'`

```
hbase(main):029:0> truncate 'employee'
Truncating 'employee' table (it may take a while):
 - Disabling table...
 - Truncating table...
0 row(s) in 3.5980 seconds
```