HBase shell commands

Creating a namespace

```
hbase shell
hbase>create_namespace 'empns'
```

Create a new hbase table in the above namespace

```
create 'empns:emp','personaldata','professionaldata'
```

Insert data in to the table

```
put 'empns:emp','1','personaldata:name','ravi'
put 'empns:emp','1', 'personaldata:city','hyderabad'
put 'empns:emp','1', 'professionaldata:desig','manager'
put 'empns:emp','1', 'professionaldata:age',35

put 'empns:emp', '2','personaldata:name', 'kiran'
put 'empns:emp', '2', 'personaldata:city', 'banglore'
put 'empns:emp', '2', 'professionaldata:desig', 'Analyst'
put 'empns:emp', '2', 'professionaldata:age', 75
```

Look in to the HDFS structure(Exit hbase shell)

```
hdfs dfs -ls /hbase/data/<namespace>
hdfs dfs -ls /hbase/data/<namespace>/<table Name>
hdfs dfs -ls /hbase/data/<namespace>/<table Name>/regionid/<column
family>

Note: You will not find any data inside as the data is still in memory
hbase shell
flush 'empns:emp'
exit
hdfs dfs -ls /hbase/data/<namespace>/<table Name>/regionid/<column
family>
```

Reading the HFile (outside of hbase shell)

```
$ hbase hfile <complete hdfs path of hfile> -e
```

Listing HBase namespaces/tables from shell

```
$ hbase shell
list                    # lists all the tables available in default namespace
list_namespace   # lists all the namespaces
list_namespace_tables <namespace name>  # lists all the tables in the name space
```

Reading data from hbase shell

```
hbase> scan 'empns:emp'                    #scans all the table
hbase>scan 'empns:emp' , {LIMIT=>1}
hbase>scan 'empns:emp' ,
{LIMIT=>1,COLUMN=>['professionaldata:age']}

hbase> get 'ns:table name', 'row-key'  #Retrieves only row key info
hbase> get 'ns:table name', 'row-key', 'f1 ' # Retrieves info f1
column family
hbase> get 'ns:table name', 'row-key', 'f1:colname' # Retrieves info
for f1 column family and col name
```

Scanning with filters

```
hbase>import
org.apache.hadoop.hbase.filter.SingleColumnValueFilter
import org.apache.hadoop.hbase.filter.CompareFilter
import org.apache.hadoop.hbase.filter.BinaryComparator

//Similar to == in SQL
hbase> scan 'empns:emp', { FILTER =>
SingleColumnValueFilter.new(Bytes.toBytes('personaldata'),
Bytes.toBytes('name'),
CompareFilter::CompareOp.valueOf('EQUAL'),BinaryComparator.n
ew(Bytes.toBytes('kiran')))}

//Similar to >= in SQL
```

```
scan 'empns:emp', { FILTER =>
SingleColumnValueFilter.new(Bytes.toBytes('professionaldata'),
Bytes.toBytes('age'),
CompareFilter::CompareOp.valueOf('GREATER_OR_EQUAL'),Binary
Comparator.new(Bytes.toBytes('50')))}

// check a specific value on all columns and returns the one that
matches.
 scan 'empns:emp', {FILTER => "ValueFilter
(=,'binaryprefix:manager') "}
```

Design a Data model for people giving multiple comments on a stock symbol

Columns:

stock_symbol, commented, userid, displayname, active, comment

Sample data

stock_symbol	commentid	userid	displayname	Active	comment
EPAM	Comment1	User1	Tiger woods	True	This is a great stock
EPAM	Comment2	User1	Tiger woods	True	This is increasing
BHD	Comment1	User2	The Ex-Governator	True	This stock is undervalued
BHD	Comment2	User2	The Ex-Governator	false	This stock is going nowhere.
APL	Comment1	User3	Federer	True	This stock will eventually make money.
APL	Comment1	User3	Federer	True	Could be a big winner

Create table

```
create "stockcommentbysymbol","info"
```

Insert data in to table

```
put "stockcommentbysymbol","EPAM:comment1","info:userid","user1"
put "stockcommentbysymbol","EPAM:comment1","info:user_display_name","GreatUser"
put "stockcommentbysymbol","EPAM:comment1","info:active","True"
put "stockcommentbysymbol","EPAM:comment1","info:comment","This is a great stock"


put "stockcommentbysymbol","EPAM:comment1","info:userid","user1"
put "stockcommentbysymbol","EPAM:comment1","info:user_display_name","GreatUser"
put "stockcommentbysymbol","EPAM:comment1","info:active","True"
put "stockcommentbysymbol","EPAM:comment1","info:comment","This is a great stock"


put "stockcommentbysymbol","BHD:comment1","info:userid","user2"
put "stockcommentbysymbol","BHD:comment1","info:user_display_name","The Ex-Governator"
put "stockcommentbysymbol","BHD:comment1","info:active","True"
put "stockcommentbysymbol","BHD:comment1","info:comment","This stock is undervalued"

put "stockcommentbysymbol","BHD:comment2","info:userid","user2"
put "stockcommentbysymbol","BHD:comment2","info:user_display_name","The Ex-Governator"
put "stockcommentbysymbol","BHD:comment2","info:active","False"
put "stockcommentbysymbol","BHD:comment2","info:comment","This stock is going nowhere."
```

```
put "stockcommentbysymbol","APL:comment1","info:userid","user3"
put
"stockcommentbysymbol","APL:comment1","info:user_display_name","Fe
derer"
put "stockcommentbysymbol","APL:comment1","info:active","True"
put "stockcommentbysymbol","APL:comment1","info:comment","This
stock will eventually make money."

put "stockcommentbysymbol","APL:comment2","info:userid","user3"
put
"stockcommentbysymbol","APL:comment2","info:user_display_name","Fe
derer"
put "stockcommentbysymbol","APL:comment2","info:active","False"
put "stockcommentbysymbol","APL:comment2","info:comment","Could
be a big winner"
```

Query patterns:

1. Show all the comments for all the stocks

```
scan 'stockcommentbysymbol'
```

2. Get all the comments for EPAM

```
scan 'stockcommentbysymbol', {FILTER =>
"PrefixFilter('EPAM:')"}
```