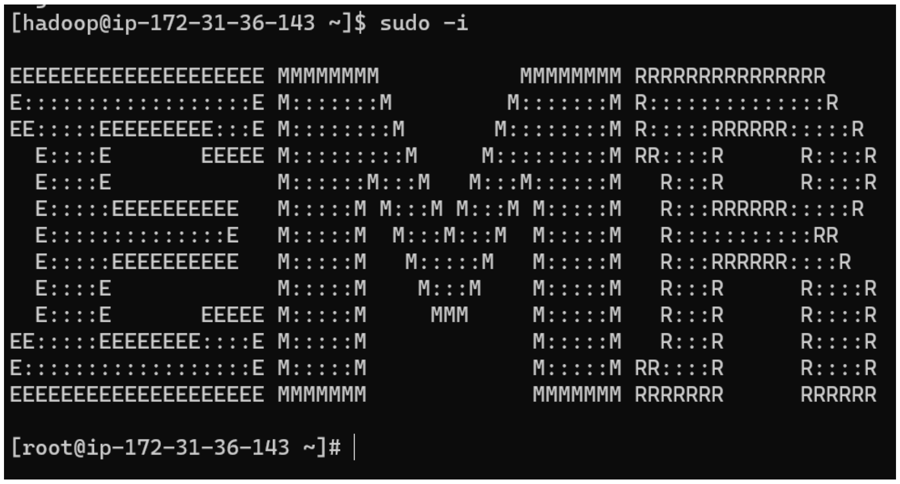
Running **HappyBase**

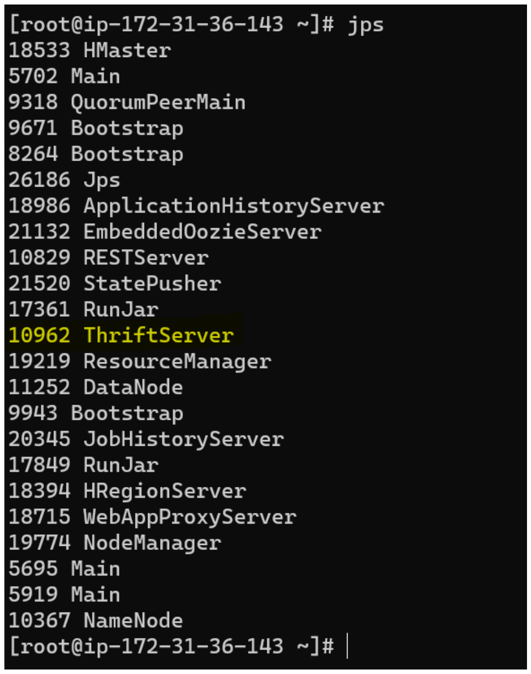
1. Login to your EMR instance using PuTTY in Windows or Shell command in Linux/Mac OS and switch to root user using the following command-

```
sudo -i
```



```
[hadoop@ip-172-31-36-143 ~]$ sudo -i

EEEEEEEEEEEEEEEEEEEE MMMMMMMM           MMMMMMMM RRRRRRRRRRRRRRR
E::::::::::::::::::E M:::::::M         M:::::::M R::::::::::::::R
EE:::::EEEEEEEEE:::E M::::::::M       M::::::::M R:::::RRRRRR:::::R
  E::::E       EEEEE M:::::::::M     M:::::::::M RR::::R      R::::R
  E::::E             M::::::M:::M   M:::M::::::M   R:::R      R::::R
  E:::::EEEEEEEEEE   M:::::M M:::M M:::M M:::::M   R:::RRRRRR:::::R
  E::::::::::::::E   M:::::M  M:::M:::M  M:::::M   R:::::::::::RR
  E:::::EEEEEEEEEE   M:::::M   M:::::M   M:::::M   R:::RRRRRR::::R
  E::::E             M:::::M    M:::M    M:::::M   R:::R      R::::R
  E::::E       EEEEE M:::::M     MMM     M:::::M   R:::R      R::::R
EE:::::EEEEEEEE::::E M:::::M             M:::::M   R:::R      R::::R
E::::::::::::::::::E M:::::M             M:::::M RR::::R      R::::R
EEEEEEEEEEEEEEEEEEEE MMMMMMM             MMMMMMM RRRRRRR      RRRRRR

[root@ip-172-31-36-143 ~]#
```

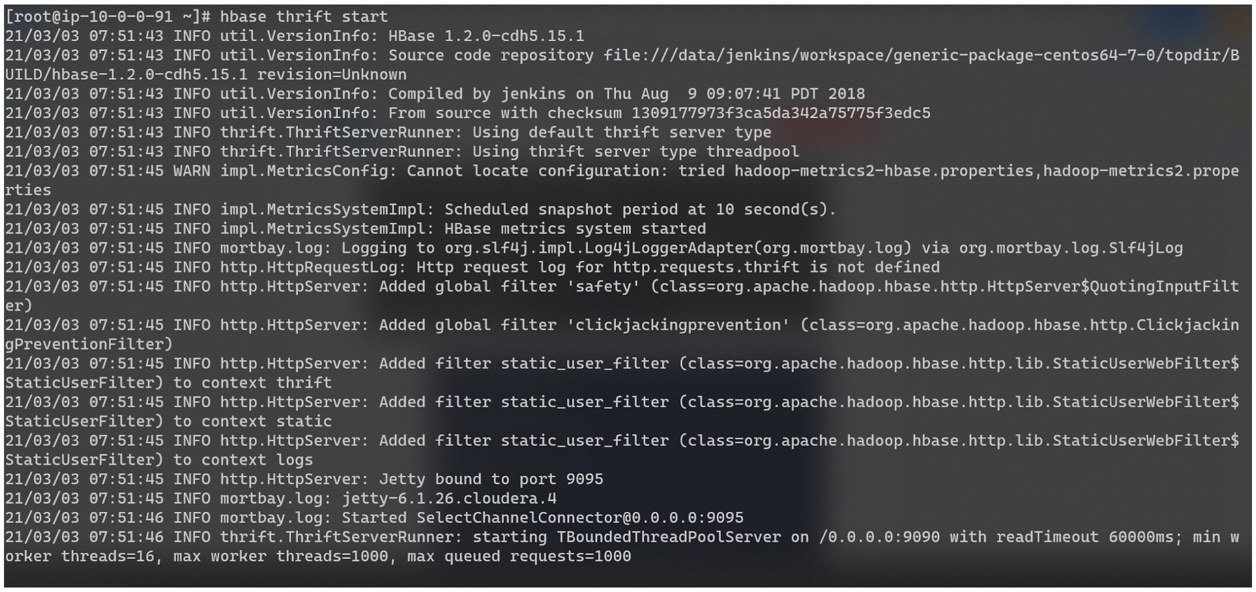
2. Make sure the thrift server is running.
   a. Run **jps** to list all the running daemons. By default, EMR should have the ThriftServer up and running already when you login to your EMR instance.



```
[root@ip-172-31-36-143 ~]# jps
18533 HMaster
5702 Main
9318 QuorumPeerMain
9671 Bootstrap
8264 Bootstrap
26186 Jps
18986 ApplicationHistoryServer
21132 EmbeddedOozieServer
10829 RESTServer
21520 StatePusher
17361 RunJar
10962 ThriftServer
19219 ResourceManager
11252 DataNode
9943 Bootstrap
20345 JobHistoryServer
17849 RunJar
18394 HRegionServer
18715 WebAppProxyServer
19774 NodeManager
5695 Main
5919 Main
10367 NameNode
[root@ip-172-31-36-143 ~]#
```

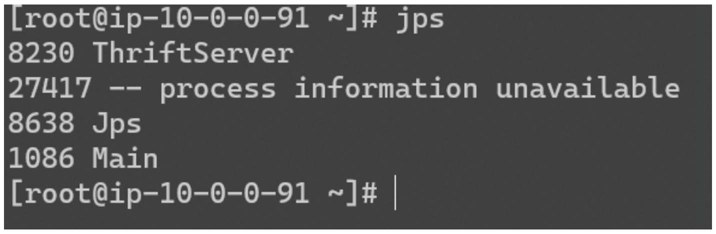
If the thrift server is not working by default, run the following command to start it

```
hbase thrift start
```



```
[root@ip-10-0-0-91 ~]# hbase thrift start
21/03/03 07:51:43 INFO util.VersionInfo: HBase 1.2.0-cdh5.15.1
21/03/03 07:51:43 INFO util.VersionInfo: Source code repository file:///data/jenkins/workspace/generic-package-centos64-7-0/topdir/B
UILD/hbase-1.2.0-cdh5.15.1 revision=Unknown
21/03/03 07:51:43 INFO util.VersionInfo: Compiled by jenkins on Thu Aug  9 09:07:41 PDT 2018
21/03/03 07:51:43 INFO util.VersionInfo: From source with checksum 1309177973f3ca5da342a75775f3edc5
21/03/03 07:51:43 INFO thrift.ThriftServerRunner: Using default thrift server type
21/03/03 07:51:43 INFO thrift.ThriftServerRunner: Using thrift server type threadpool
21/03/03 07:51:45 WARN impl.MetricsConfig: Cannot locate configuration: tried hadoop-metrics2-hbase.properties,hadoop-metrics2.prope
rties
21/03/03 07:51:45 INFO impl.MetricsSystemImpl: Scheduled snapshot period at 10 second(s).
21/03/03 07:51:45 INFO impl.MetricsSystemImpl: HBase metrics system started
21/03/03 07:51:45 INFO mortbay.log: Logging to org.slf4j.impl.Log4jLoggerAdapter(org.mortbay.log) via org.mortbay.log.Slf4jLog
21/03/03 07:51:45 INFO http.HttpRequestLog: Http request log for http.requests.thrift is not defined
21/03/03 07:51:45 INFO http.HttpServer: Added global filter 'safety' (class=org.apache.hadoop.hbase.http.HttpServer$QuotingInputFilt
er)
21/03/03 07:51:45 INFO http.HttpServer: Added global filter 'clickjackingprevention' (class=org.apache.hadoop.hbase.http.Clickjackin
gPreventionFilter)
21/03/03 07:51:45 INFO http.HttpServer: Added filter static_user_filter (class=org.apache.hadoop.hbase.http.lib.StaticUserWebFilter$
StaticUserFilter) to context thrift
21/03/03 07:51:45 INFO http.HttpServer: Added filter static_user_filter (class=org.apache.hadoop.hbase.http.lib.StaticUserWebFilter$
StaticUserFilter) to context static
21/03/03 07:51:45 INFO http.HttpServer: Added filter static_user_filter (class=org.apache.hadoop.hbase.http.lib.StaticUserWebFilter$
StaticUserFilter) to context logs
21/03/03 07:51:45 INFO http.HttpServer: Jetty bound to port 9095
21/03/03 07:51:45 INFO mortbay.log: jetty-6.1.26.cloudera.4
21/03/03 07:51:46 INFO mortbay.log: Started SelectChannelConnector@0.0.0.0:9095
21/03/03 07:51:46 INFO thrift.ThriftServerRunner: starting TBoundedThreadPoolServer on /0.0.0.0:9090 with readTimeout 60000ms; min w
orker threads=16, max worker threads=1000, max queued requests=1000
```

Verify if the thrift server has started. Run the **jps** command again

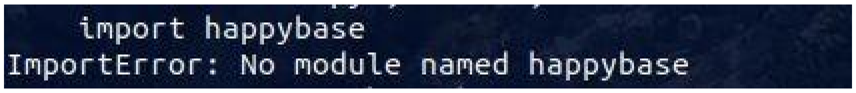


```
[root@ip-10-0-0-91 ~]# jps
8230 ThriftServer
27417 -- process information unavailable
8638 Jps
1086 Main
[root@ip-10-0-0-91 ~]#
```

Run the below command to test if the HappyBase package is present or not?

```
python -c "import happybase"
```

If you get an import error saying “**No module named happybase**”.



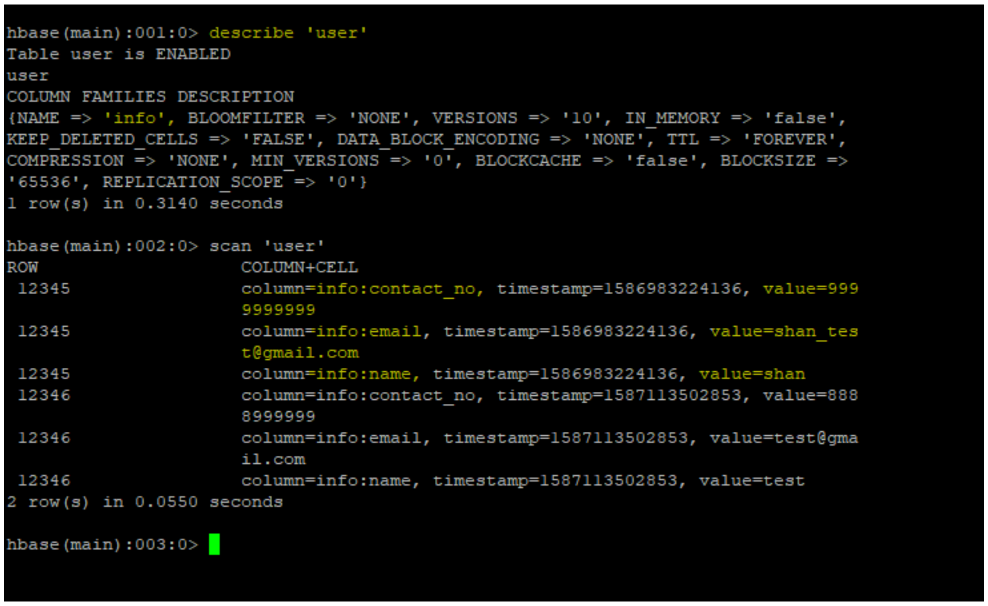
```
    import happybase
ImportError: No module named happybase
```

Then you need to install **happybase** by running the command below.

```
pip install happybase
```

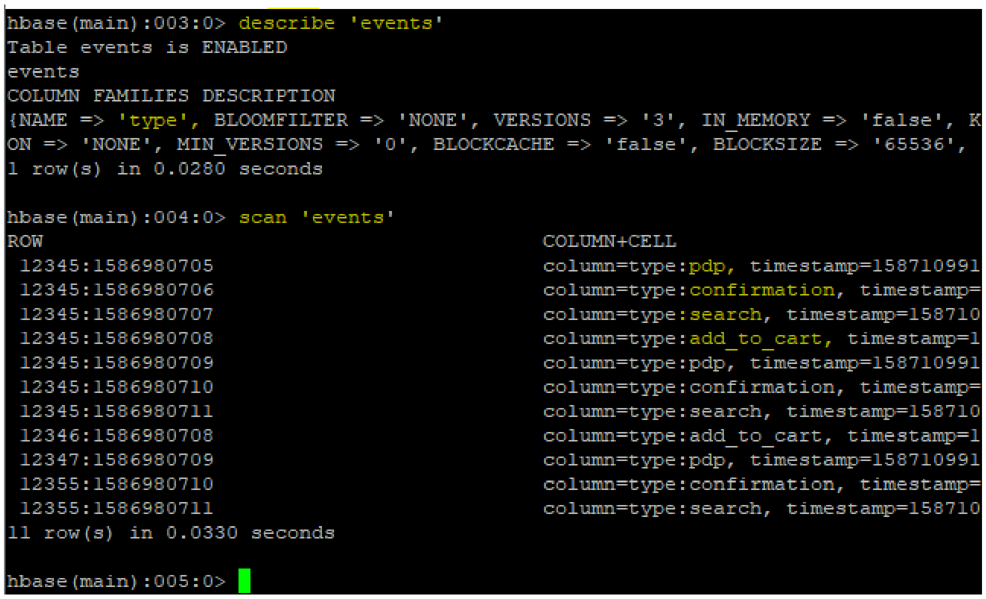
Use Case:
The events are flowing in real-time on a platform, where events can be of type search, product description page, add to cart, confirmation etc. We have to fetch and transform events of the user in real-time and store it in HBase which can be later used for analytics.
Note: To execute the python scripts given below, first of all, create the following tables in HBase using HBase shell commands and add relevant information to the tables using the put command.
Tables:

- User



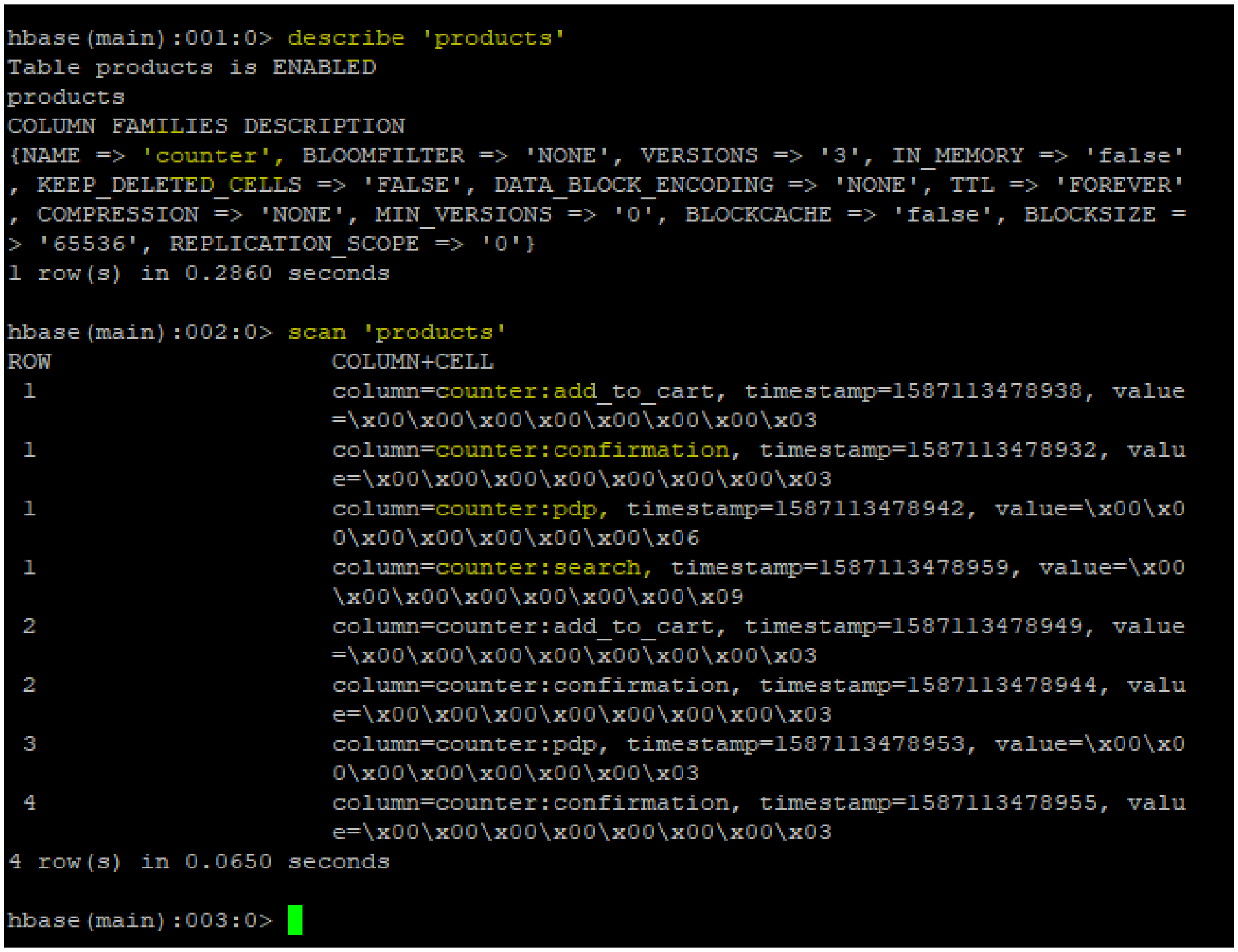
```
hbase(main):001:0> describe 'user'
Table user is ENABLED
user
COLUMN FAMILIES DESCRIPTION
{NAME => 'info', BLOOMFILTER => 'NONE', VERSIONS => '10', IN_MEMORY => 'false',
KEEP_DELETED_CELLS => 'FALSE', DATA_BLOCK_ENCODING => 'NONE', TTL => 'FOREVER',
COMPRESSION => 'NONE', MIN_VERSIONS => '0', BLOCKCACHE => 'false', BLOCKSIZE =>
'65536', REPLICATION_SCOPE => '0'}
1 row(s) in 0.3140 seconds

hbase(main):002:0> scan 'user'
ROW                      COLUMN+CELL
 12345                   column=info:contact_no, timestamp=1586983224136, value=999
                         9999999
 12345                   column=info:email, timestamp=1586983224136, value=shan_tes
                         t@gmail.com
 12345                   column=info:name, timestamp=1586983224136, value=shan
 12346                   column=info:contact_no, timestamp=1587113502853, value=888
                         8999999
 12346                   column=info:email, timestamp=1587113502853, value=test@gma
                         il.com
 12346                   column=info:name, timestamp=1587113502853, value=test
2 row(s) in 0.0550 seconds

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

- Events



```
hbase(main):003:0> describe 'events'
Table events is ENABLED
events
COLUMN FAMILIES DESCRIPTION
{NAME => 'type', BLOOMFILTER => 'NONE', VERSIONS => '3', IN_MEMORY => 'false', K
ON => 'NONE', MIN_VERSIONS => '0', BLOCKCACHE => 'false', BLOCKSIZE => '65536',
1 row(s) in 0.0280 seconds

hbase(main):004:0> scan 'events'
ROW                                   COLUMN+CELL
 12345:1586980705                     column=type:pdp, timestamp=158710991
 12345:1586980706                     column=type:confirmation, timestamp=
 12345:1586980707                     column=type:search, timestamp=158710
 12345:1586980708                     column=type:add_to_cart, timestamp=1
 12345:1586980709                     column=type:pdp, timestamp=158710991
 12345:1586980710                     column=type:confirmation, timestamp=
 12345:1586980711                     column=type:search, timestamp=158710
 12346:1586980708                     column=type:add_to_cart, timestamp=1
 12347:1586980709                     column=type:pdp, timestamp=158710991
 12355:1586980710                     column=type:confirmation, timestamp=
 12355:1586980711                     column=type:search, timestamp=158710
11 row(s) in 0.0330 seconds

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

- Products



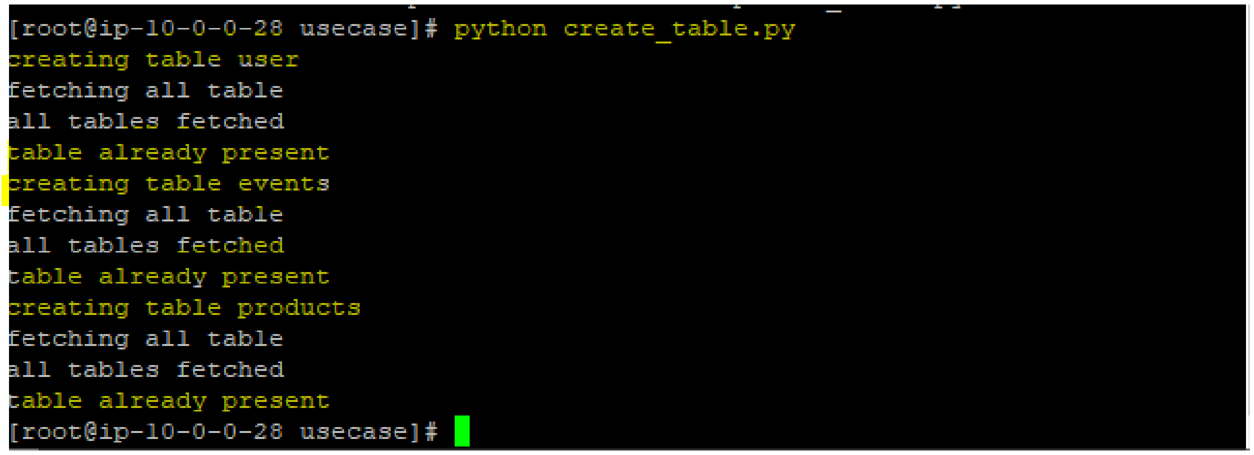
```
hbase(main):001:0> describe 'products'
Table products is ENABLED
products
COLUMN FAMILIES DESCRIPTION
{NAME => 'counter', BLOOMFILTER => 'NONE', VERSIONS => '3', IN_MEMORY => 'false'
, KEEP_DELETED_CELLS => 'FALSE', DATA_BLOCK_ENCODING => 'NONE', TTL => 'FOREVER'
, COMPRESSION => 'NONE', MIN_VERSIONS => '0', BLOCKCACHE => 'false', BLOCKSIZE =
> '65536', REPLICATION_SCOPE => '0'}
1 row(s) in 0.2860 seconds

hbase(main):002:0> scan 'products'
ROW                      COLUMN+CELL
 1                       column=counter:add_to_cart, timestamp=1587113478938, value
                         =\x00\x00\x00\x00\x00\x00\x00\x03
 1                       column=counter:confirmation, timestamp=1587113478932, valu
                         e=\x00\x00\x00\x00\x00\x00\x00\x03
 1                       column=counter:pdp, timestamp=1587113478942, value=\x00\x0
                         0\x00\x00\x00\x00\x00\x06
 1                       column=counter:search, timestamp=1587113478959, value=\x00
                         \x00\x00\x00\x00\x00\x00\x09
 2                       column=counter:add_to_cart, timestamp=1587113478949, value
                         =\x00\x00\x00\x00\x00\x00\x00\x03
 2                       column=counter:confirmation, timestamp=1587113478944, valu
                         e=\x00\x00\x00\x00\x00\x00\x00\x03
 3                       column=counter:pdp, timestamp=1587113478953, value=\x00\x0
                         0\x00\x00\x00\x00\x00\x03
 4                       column=counter:confirmation, timestamp=1587113478955, valu
                         e=\x00\x00\x00\x00\x00\x00\x00\x03
4 row(s) in 0.0650 seconds

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

Questions:
1. Create all the tables using happy base if not present

```
import happybase

#create connection
connection = happybase.Connection('localhost', port=9090 , autoconnect=False)

#open connection to perform operations
def open_connection():
    connection.open()

#close the opened connection
```

```
def close_connection():
    connection.close()

#list all tables in Hbase
def list_tables():
    print("fetching all table")
    open_connection()
    tables = connection.tables()
    close_connection()
    print("all tables fetched")
    return tables


#create a table by passing name and column family as a parameter
def create_table(name,cf):
    print("creating table " + name)
    tables = list_tables()
    if name not in tables:
       open_connection()
       connection.create_table(name, cf)
       close_connection()
       print("table created")

    else:
       print("table already present")

create_table('user', {'info' : dict(max_versions=5) })
create_table('events', {'type' : dict() })
create_table('products', {'counter' : dict() })
```

Output:



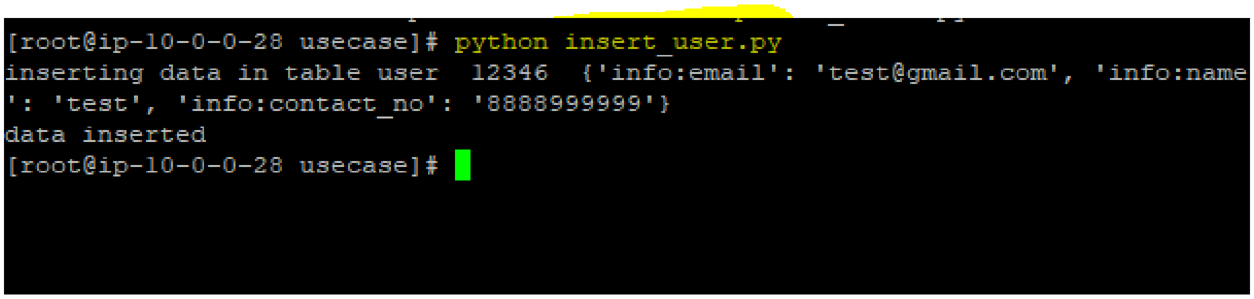
```
[root@ip-10-0-0-28 usecase]# python create_table.py
creating table user
fetching all table
all tables fetched
table already present
creating table events
fetching all table
all tables fetched
table already present
creating table products
fetching all table
all tables fetched
table already present
[root@ip-10-0-0-28 usecase]#
```

2. Insert data in table user

```
import happybase

#create connection
connection = happybase.Connection('localhost', port=9090 ,
autoconnect=False)

#open connection to perform operations
def open_connection():
    connection.open()

#close the opened connection
def close_connection():
    connection.close()

#get the pointer to a table
def get_table(name):
    open_connection()
    table = connection.table(name)
    close_connection()
    return table

#insert data in a table by passing key and data
```

```
def insert_data(table, key, data):
    print("inserting data in table " + table + "  " + key + "  " + 
str(data))
    table = get_table(table)
    open_connection()
    table.put(key, data)
    close_connection()
    print("data inserted")


insert_data('user' , '12346' ,{'info:name': 'test', 
'info:contact_no': '8888999999', 'info:email': 'test@gmail.com' })
```

Output:



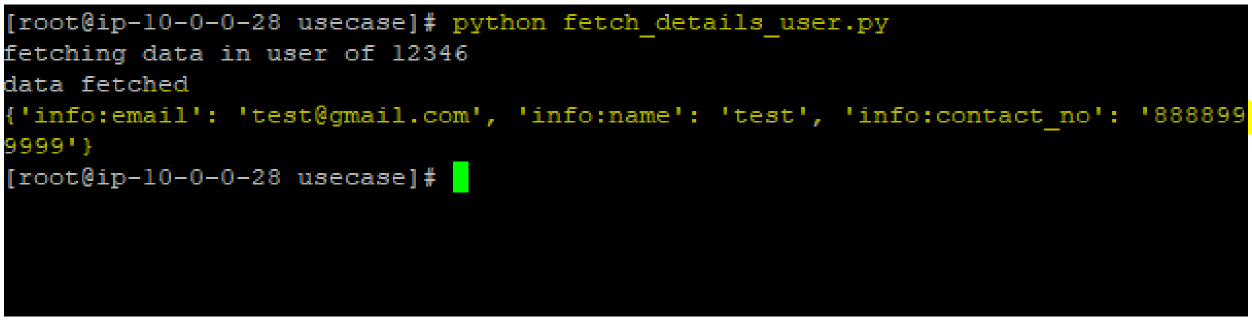
```
[root@ip-10-0-0-28 usecase]# python insert_user.py
inserting data in table user  12346  {'info:email': 'test@gmail.com', 'info:name
': 'test', 'info:contact_no': '8888999999'}
data inserted
[root@ip-10-0-0-28 usecase]# 
```

3. Fetch details of a user

```
import happybase

#create connection
connection = happybase.Connection('localhost', port=9090 , autoconnect=False)

#open connection to perform operations
def open_connection():
    connection.open()

#close the opened connection
def close_connection():
    connection.close()

#get the pointer to a table
```

```
def get_table(name):
    open_connection()
    table = connection.table(name)
    close_connection()
    return table

#fetch data from a table corresponding to a key
def get_data(table, key):
    print("fetching data in " + table + " of " + key)
    table = get_table(table)
    open_connection()
    data = table.row(key)
    close_connection()
    print("data fetched")
    return data

print(get_data('user', '12346'))
```

Output:



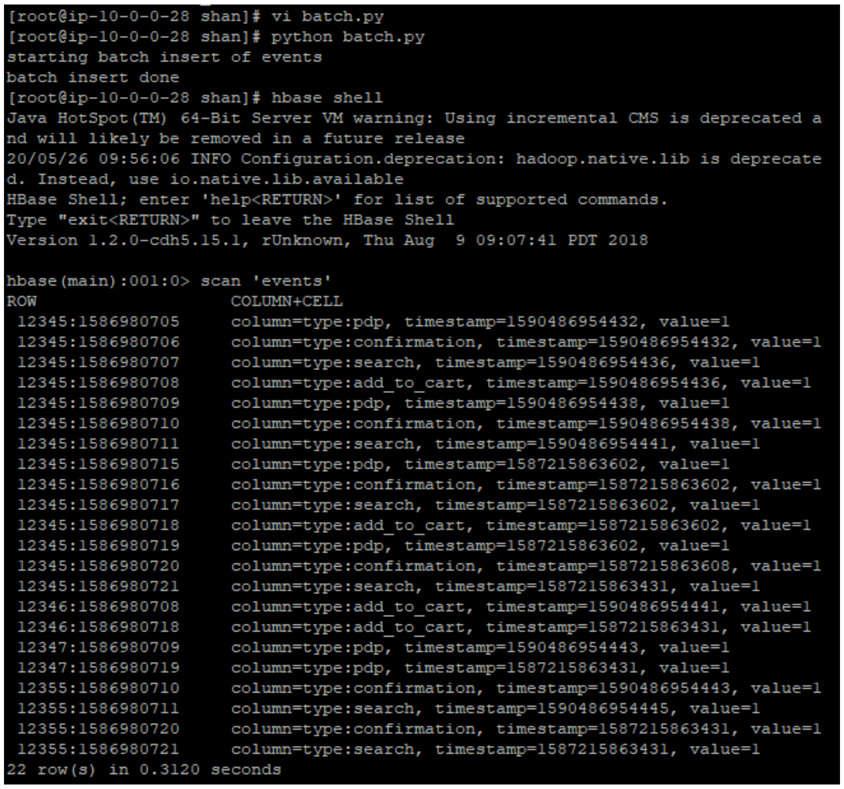
```
[root@ip-10-0-0-28 usecase]# python fetch_details_user.py
fetching data in user of 12346
data fetched
{'info:email': 'test@gmail.com', 'info:name': 'test', 'info:contact_no': '888899
9999'}
[root@ip-10-0-0-28 usecase]#
```

4. Batch insert events data
Create an input.txt file in the same directory where your python script is present.
Input:
epochTime,userID,productID,eventType
1586980705,12345,1,pdp
1586980706,12345,1,confirmation
1586980707,12345,1,search
1586980708,12345,1,add_to_cart
1586980709,12345,1,pdp
1586980710,12345,2,confirmation

```
1586980711,12345,1,search
1586980708,12346,2,add_to_cart
1586980709,12347,3,pdp
1586980710,12355,4,confirmation
1586980711,12355,1,search
```

```
import happybase

#create connection
connection = happybase.Connection('localhost', port=9090 ,
autoconnect=False)

#open connection to perform operations
def open_connection():
    connection.open()

#close the opened connection
def close_connection():
    connection.close()

#get the pointer to a table
def get_table(name):
    open_connection()
    table = connection.table(name)
    close_connection()
    return table

#batch insert data in events table
def batch_insert_data(filename):
    print("starting batch insert of events")
    file = open(filename, "r")
    table = get_table(filename)
    open_connection()
    i = 0
    with table.batch(batch_size=2) as b:
      for line in file:
         if i!=0:
            temp = line.strip().split(",")
            # this put() will result in two mutations (two cells)
            b.put(temp[1]+":"+temp[0] , { 'type:'+temp[3]: '1' })
         i+=1

    file.close()
    print("batch insert done")
    close_connection()

#Note: first create a file with name events and insert input data in
it
batch_insert_data('events')
```

Output:

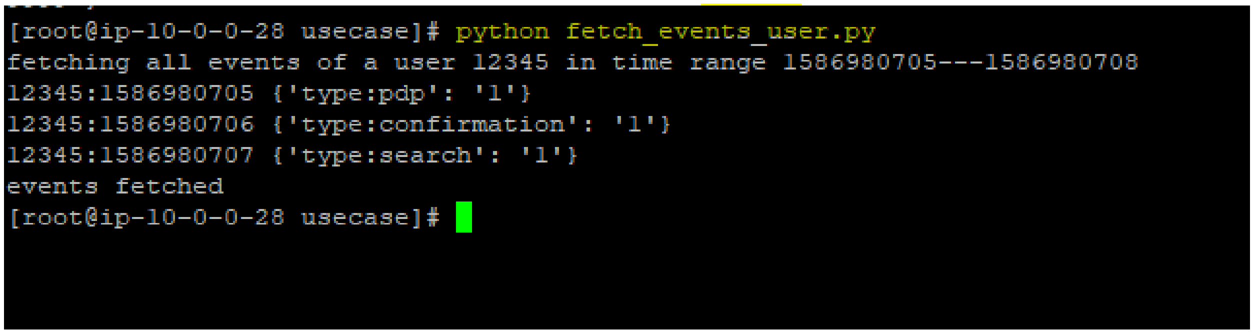


```
[root@ip-10-0-0-28 shan]# vi batch.py
[root@ip-10-0-0-28 shan]# python batch.py
starting batch insert of events
batch insert done
[root@ip-10-0-0-28 shan]# hbase shell
Java HotSpot(TM) 64-Bit Server VM warning: Using incremental CMS is deprecated a
nd will likely be removed in a future release
20/05/26 09:56:06 INFO Configuration.deprecation: hadoop.native.lib is deprecate
d. 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-cdh5.15.1, rUnknown, Thu Aug  9 09:07:41 PDT 2018

hbase(main):001:0> scan 'events'
ROW                      COLUMN+CELL
 12345:1586980705        column=type:pdp, timestamp=1590486954432, value=1
 12345:1586980706        column=type:confirmation, timestamp=1590486954432, value=1
 12345:1586980707        column=type:search, timestamp=1590486954436, value=1
 12345:1586980708        column=type:add_to_cart, timestamp=1590486954436, value=1
 12345:1586980709        column=type:pdp, timestamp=1590486954438, value=1
 12345:1586980710        column=type:confirmation, timestamp=1590486954438, value=1
 12345:1586980711        column=type:search, timestamp=1590486954441, value=1
 12345:1586980715        column=type:pdp, timestamp=1587215863602, value=1
 12345:1586980716        column=type:confirmation, timestamp=1587215863602, value=1
 12345:1586980717        column=type:search, timestamp=1587215863602, value=1
 12345:1586980718        column=type:add_to_cart, timestamp=1587215863602, value=1
 12345:1586980719        column=type:pdp, timestamp=1587215863602, value=1
 12345:1586980720        column=type:confirmation, timestamp=1587215863608, value=1
 12345:1586980721        column=type:search, timestamp=1587215863431, value=1
 12346:1586980708        column=type:add_to_cart, timestamp=1590486954441, value=1
 12346:1586980718        column=type:add_to_cart, timestamp=1587215863431, value=1
 12347:1586980709        column=type:pdp, timestamp=1590486954443, value=1
 12347:1586980719        column=type:pdp, timestamp=1587215863431, value=1
 12355:1586980710        column=type:confirmation, timestamp=1590486954443, value=1
 12355:1586980711        column=type:search, timestamp=1590486954445, value=1
 12355:1586980720        column=type:confirmation, timestamp=1587215863431, value=1
 12355:1586980721        column=type:search, timestamp=1587215863431, value=1
22 row(s) in 0.3120 seconds
```

5. Find events of a user in the time range
Note: While calling the function you need to mention the valid time range.

```
import happybase

#create connection
connection = happybase.Connection('localhost', port=9090 ,autoconnect=False)

#open connection to perform operations
def open_connection():
    connection.open()

#close the opened connection
def close_connection():
    connection.close()

#get the pointer to a table
def get_table(name):
    open_connection()
    table = connection.table(name)
    close_connection()
    return table

#scan events table for a user by passing start and end timestamp
def scan_data(table, user_id, start, end):
    print("fetching all events of a user " + user_id + " in time range " +   start +
"---" + end)
    table = get_table(table)
    open_connection()
    for key,data in table.scan(row_start=user_id+":"+start, row_stop=user_id+":"+end):
        print(key,data)
    close_connection()
    print("events fetched")


scan_data('events','12345','1586980705','1586980708')
```

Output:



```
[root@ip-10-0-0-28 usecase]# python fetch_events_user.py
fetching all events of a user 12345 in time range 1586980705---1586980708
12345:1586980705 {'type:pdp': '1'}
12345:1586980706 {'type:confirmation': '1'}
12345:1586980707 {'type:search': '1'}
events fetched
[root@ip-10-0-0-28 usecase]#
```

6. Increment the count of events of a product and fetch the count of events of a product
Create an input.txt file in the same directory where your python script is present.

Input:

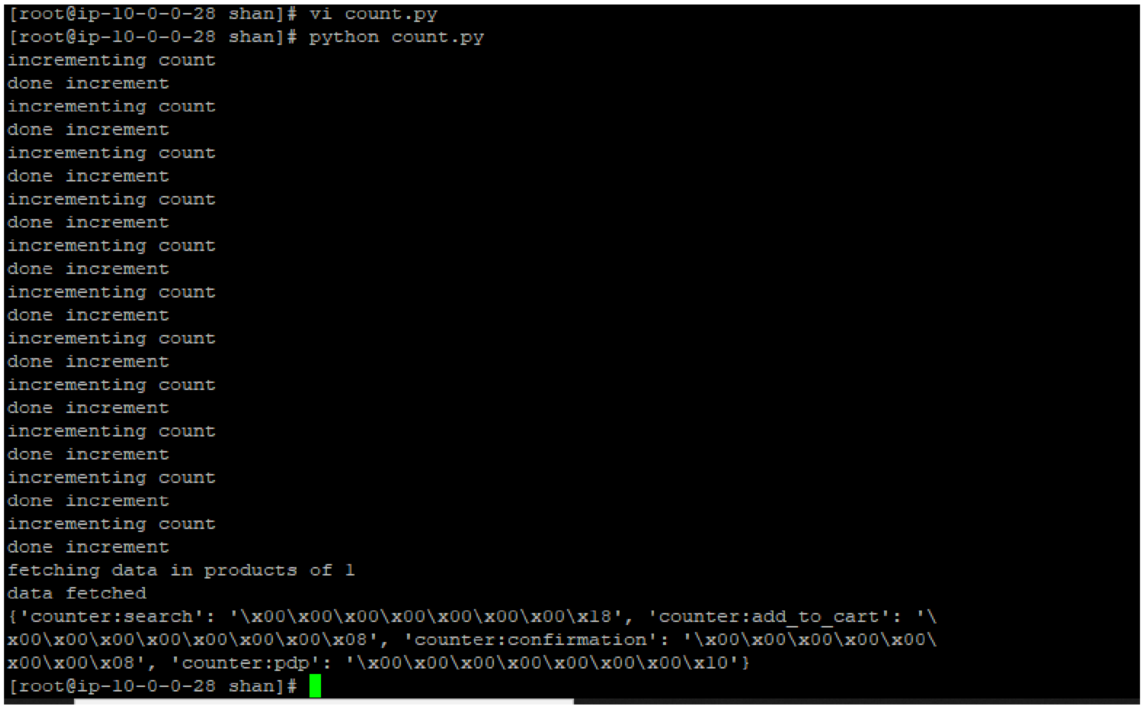
```
epochTime,userID,productID,eventType
1586980705,12345,1,pdp
1586980706,12345,1,confirmation
1586980707,12345,1,search
1586980708,12345,1,add_to_cart
1586980709,12345,1,pdp
1586980710,12345,2,confirmation
1586980711,12345,1,search
1586980708,12346,2,add_to_cart
1586980709,12347,3,pdp
1586980710,12355,4,confirmation
1586980711,12355,1,search
```

```
import happybase

#create connection
connection = happybase.Connection('localhost', port=9090 ,
autoconnect=False)

#open connection to perform operations
def open_connection():
    connection.open()
```

```
#close the opened connection
def close_connection():
    connection.close()


#get the pointer to a table
def get_table(name):
    open_connection()
    table = connection.table(name)
    close_connection()
    return table

def get_data(table, key):
    print("fetching data in " + table + " of " + key)
    table = get_table(table)
    open_connection()
    data = table.row(key)
    close_connection()
    print("data fetched")
    return data
# increment the count in products table by passing key and value
def increment_count(table,key,value):
    print("incrementing count")
    table = get_table(table)
    open_connection()
    table.counter_inc(key, 'counter:'+value)
    close_connection()
    print("done increment")
#Note: first make the file with name events and insert the input data
in file
file = open('events',"r")
i = 0
for line in file:
    if i !=0:
       temp = line.strip().split(",")
       increment_count('products',temp[2], temp[3])
    i+=1
file.close()
```

```
print(get_data('products','1'))
```

Output:



```
[root@ip-10-0-0-28 shan]# vi count.py
[root@ip-10-0-0-28 shan]# python count.py
incrementing count
done increment
incrementing count
done increment
incrementing count
done increment
incrementing count
done increment
incrementing count
done increment
incrementing count
done increment
incrementing count
done increment
incrementing count
done increment
incrementing count
done increment
incrementing count
done increment
incrementing count
done increment
fetching data in products of 1
data fetched
{'counter:search': '\x00\x00\x00\x00\x00\x00\x00\x18', 'counter:add_to_cart': '\
x00\x00\x00\x00\x00\x00\x00\x08', 'counter:confirmation': '\x00\x00\x00\x00\x00\
x00\x00\x08', 'counter:pdp': '\x00\x00\x00\x00\x00\x00\x00\x10'}
[root@ip-10-0-0-28 shan]#
```