HandOut.7: Delving Deep into Hbase

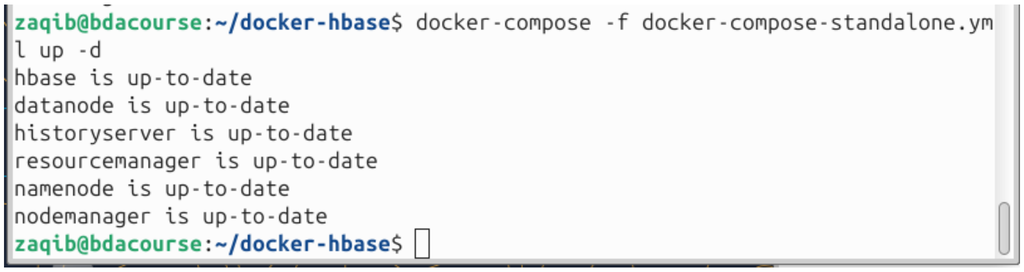
THIS LAB HAS BEEN WORKED ON BY BOTH MAHNOOR ADEEL AND ZUHA AQIB. BOTH OF US HAVE USED ZUHA'S VM FOR THIS LAB.

Installation

Access your docker working directory

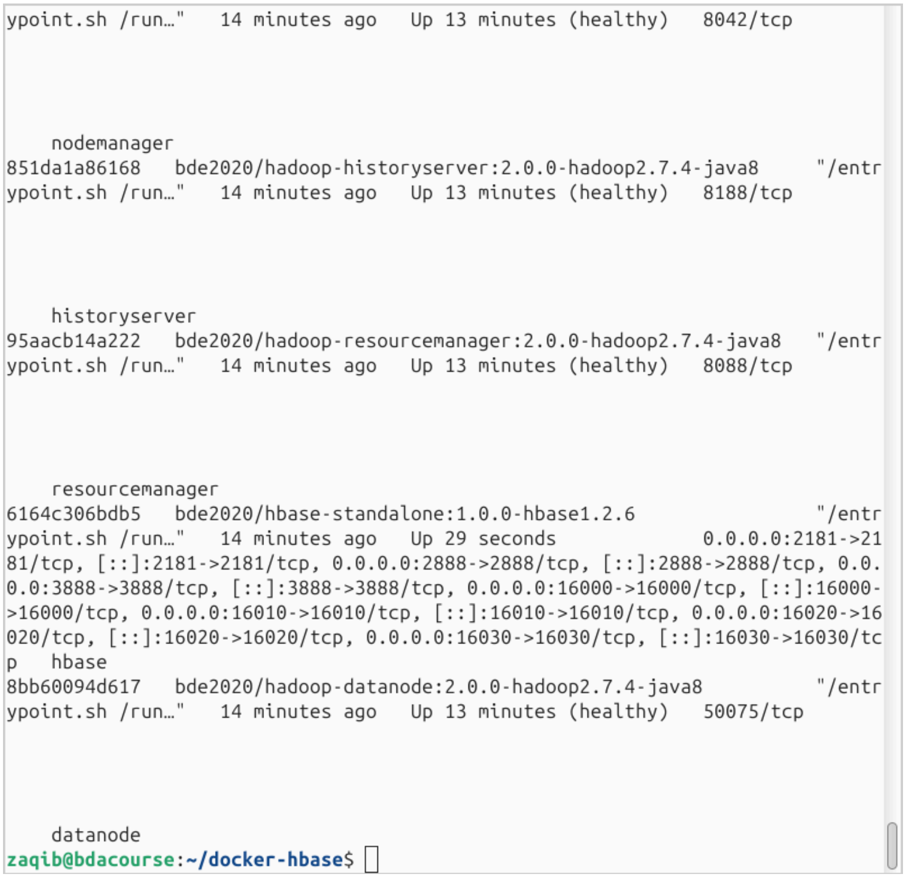
Execute: git clone https://github.com/big-data-europe/docker-hbase.git

Execute (Hbase in standalone mode): docker-compose -f docker-compose-standalone.yml up -d



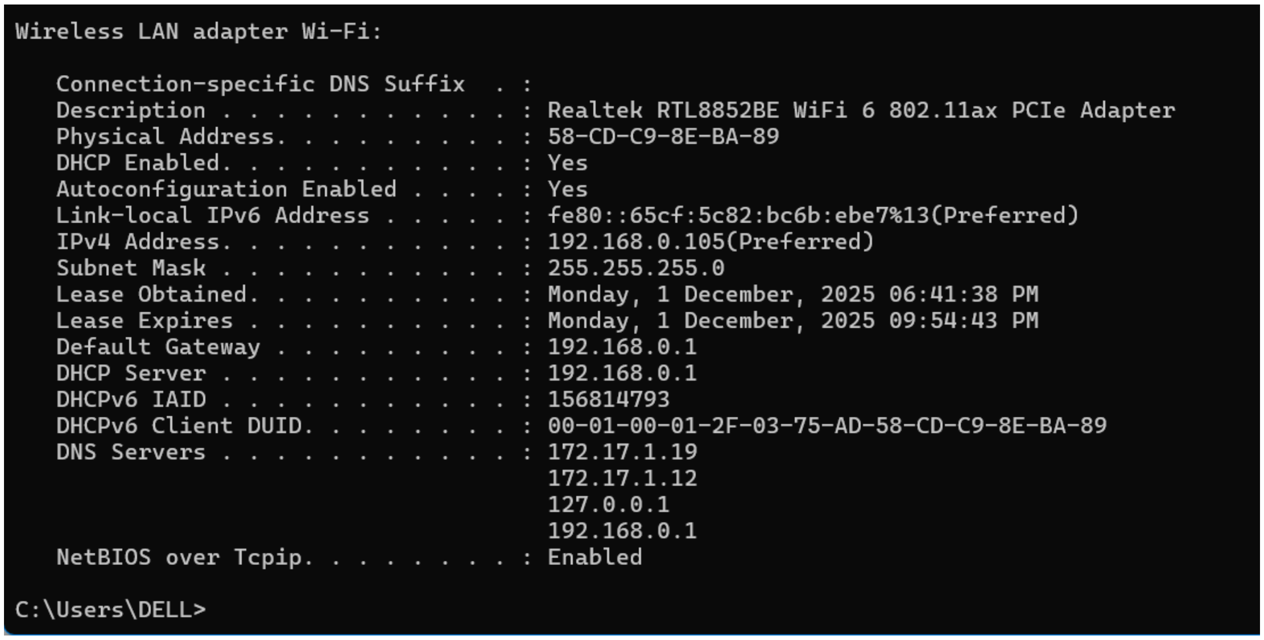
```
zaqib@bdacourse:~/docker-hbase$ docker-compose -f docker-compose-standalone.ym
l up -d
hbase is up-to-date
datanode is up-to-date
historyserver is up-to-date
resourcemanager is up-to-date
namenode is up-to-date
nodemanager is up-to-date
zaqib@bdacourse:~/docker-hbase$
```

Check all containers: docker ps



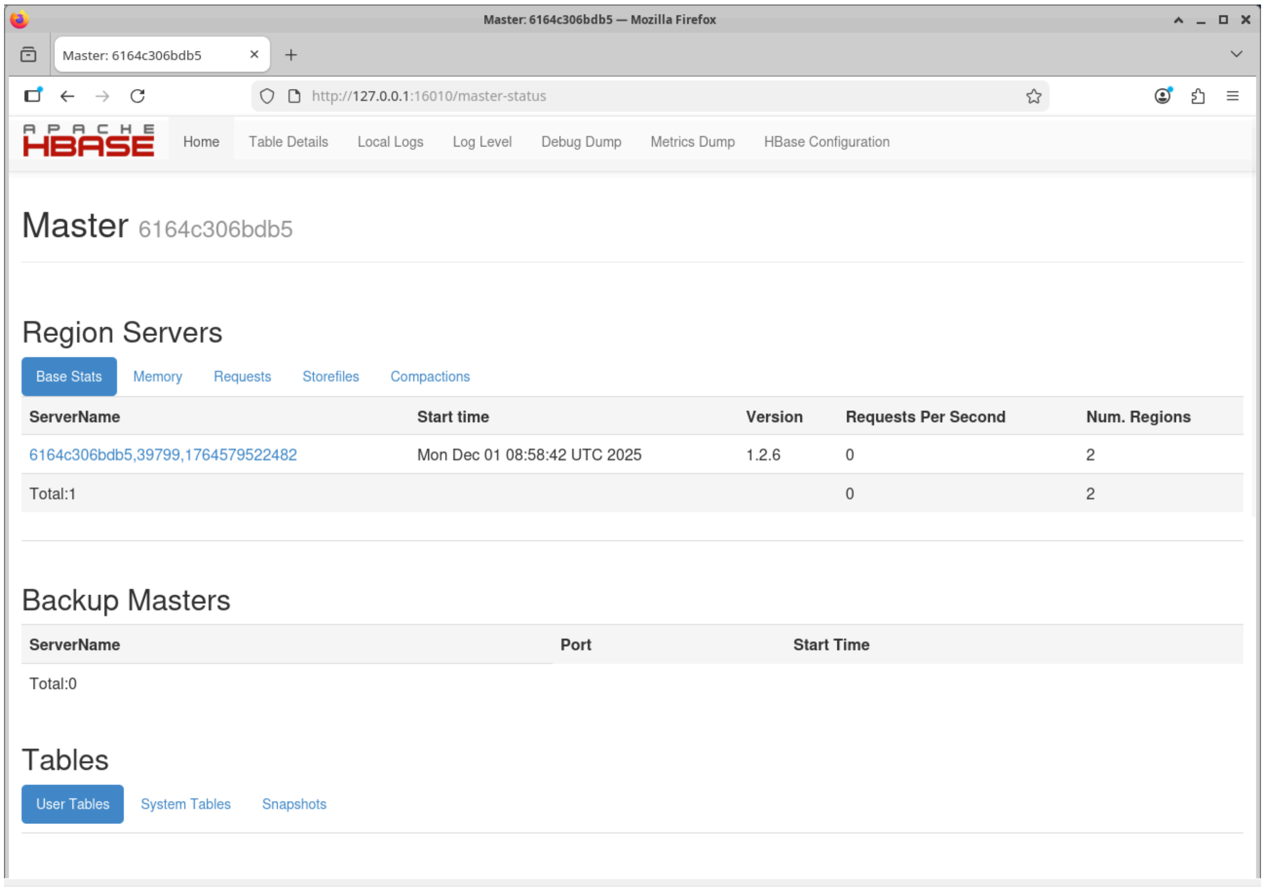
```
ypoint.sh /run…"   14 minutes ago   Up 13 minutes (healthy)   8042/tcp



    nodemanager
851da1a86168   bde2020/hadoop-historyserver:2.0.0-hadoop2.7.4-java8     "/entr
ypoint.sh /run…"   14 minutes ago   Up 13 minutes (healthy)   8188/tcp



    historyserver
95aacb14a222   bde2020/hadoop-resourcemanager:2.0.0-hadoop2.7.4-java8   "/entr
ypoint.sh /run…"   14 minutes ago   Up 13 minutes (healthy)   8088/tcp



    resourcemanager
6164c306bdb5   bde2020/hbase-standalone:1.0.0-hbase1.2.6                "/entr
ypoint.sh /run…"   14 minutes ago   Up 29 seconds             0.0.0.0:2181->21
81/tcp, [::]:2181->2181/tcp, 0.0.0.0:2888->2888/tcp, [::]:2888->2888/tcp, 0.0.
0.0:3888->3888/tcp, [::]:3888->3888/tcp, 0.0.0.0:16000->16000/tcp, [::]:16000-
>16000/tcp, 0.0.0.0:16010->16010/tcp, [::]:16010->16010/tcp, 0.0.0.0:16020->16
020/tcp, [::]:16020->16020/tcp, 0.0.0.0:16030->16030/tcp, [::]:16030->16030/tc
p   hbase
8bb60094d617   bde2020/hadoop-datanode:2.0.0-hadoop2.7.4-java8          "/entr
ypoint.sh /run…"   14 minutes ago   Up 13 minutes (healthy)   50075/tcp



    datanode
zaqib@bdacourse:~/docker-hbase$
```

Check your IP: ipconfig /all



```
Wireless LAN adapter Wi-Fi:

   Connection-specific DNS Suffix  . :
   Description . . . . . . . . . . . : Realtek RTL8852BE WiFi 6 802.11ax PCIe Adapter
   Physical Address. . . . . . . . . : 58-CD-C9-8E-BA-89
   DHCP Enabled. . . . . . . . . . . : Yes
   Autoconfiguration Enabled . . . . : Yes
   Link-local IPv6 Address . . . . . : fe80::65cf:5c82:bc6b:ebe7%13(Preferred)
   IPv4 Address. . . . . . . . . . . : 192.168.0.105(Preferred)
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Lease Obtained. . . . . . . . . . : Monday, 1 December, 2025 06:41:38 PM
   Lease Expires . . . . . . . . . . : Monday, 1 December, 2025 09:54:43 PM
   Default Gateway . . . . . . . . . : 192.168.0.1
   DHCP Server . . . . . . . . . . . : 192.168.0.1
   DHCPv6 IAID . . . . . . . . . . . : 156814793
   DHCPv6 Client DUID. . . . . . . . : 00-01-00-01-2F-03-75-AD-58-CD-C9-8E-BA-89
   DNS Servers . . . . . . . . . . . : 172.17.1.19
                                       172.17.1.12
                                       127.0.0.1
                                       192.168.0.1
   NetBIOS over Tcpip. . . . . . . . : Enabled

C:\Users\DELL>
```

[ToDo: Paste relevant snapshot at port 16010]



Master 6164c306bdb5

Region Servers

ServerName	Start time	Version	Requests Per Second	Num. Regions
6164c306bdb5,39799,1764579522482	Mon Dec 01 08:58:42 UTC 2025	1.2.6	0	2
Total:1			0	2

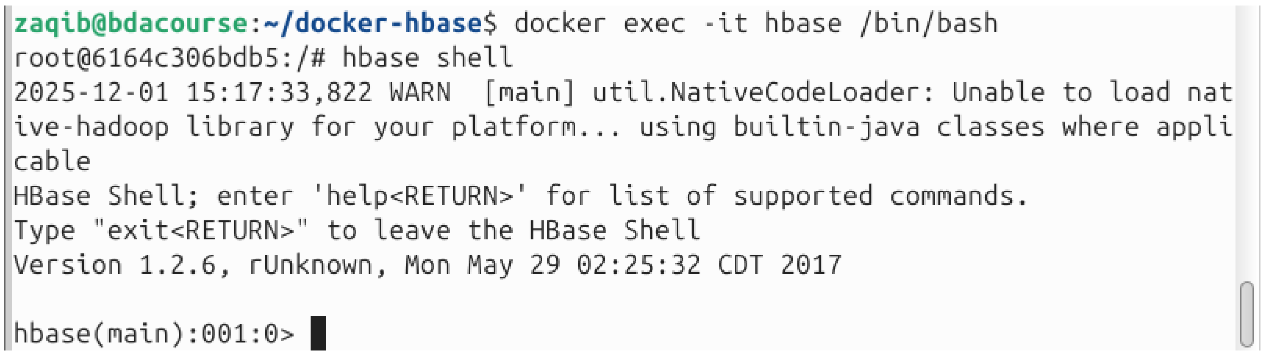
Backup Masters

ServerName	Port	Start Time
Total:0		

Tables

Execute: docker exec -it hbase /bin/bash

Execute: hbase shell



```
zaqib@bdacourse:~/docker-hbase$ docker exec -it hbase /bin/bash
root@6164c306bdb5:/# hbase shell
2025-12-01 15:17:33,822 WARN  [main] util.NativeCodeLoader: Unable to load nat
ive-hadoop library for your platform... using builtin-java classes where appli
cable
HBase Shell; enter 'help<RETURN>' for list of supported commands.
Type "exit<RETURN>" to leave the HBase Shell
Version 1.2.6, rUnknown, Mon May 29 02:25:32 CDT 2017

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

General Shell Commands

Starting HBase Shell: hbase shell

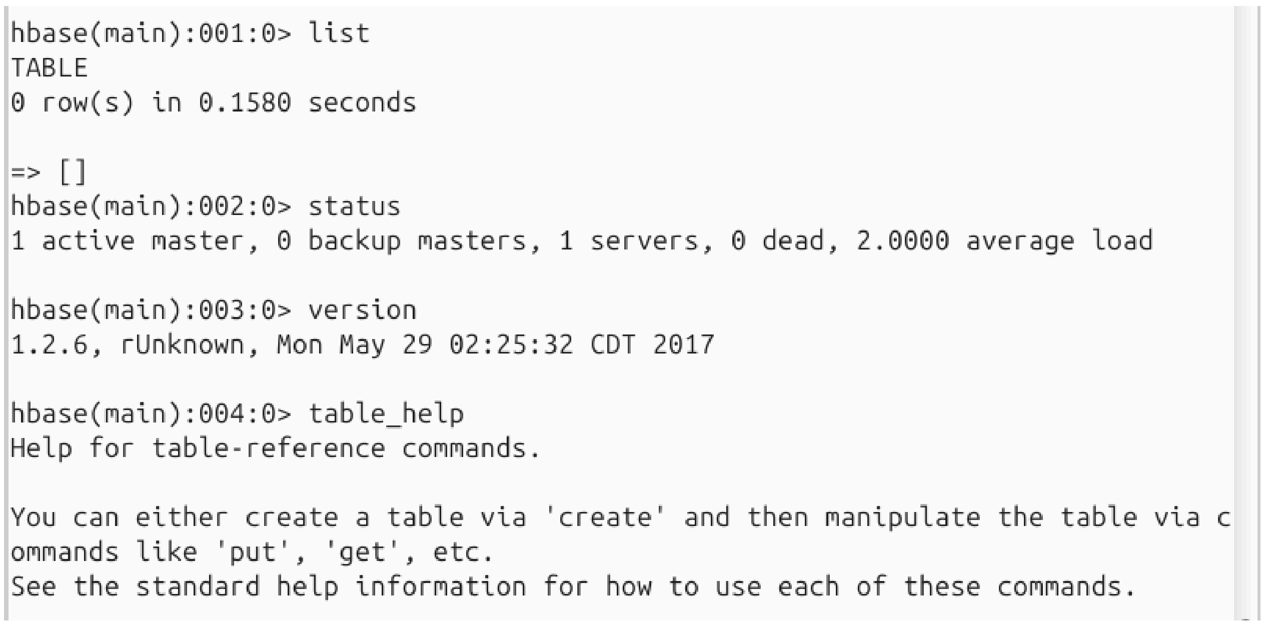
Exiting HBase Shell: exit

View existing tables in HBase: list

View existing servers in HBase: status

View version of HBase in use: version

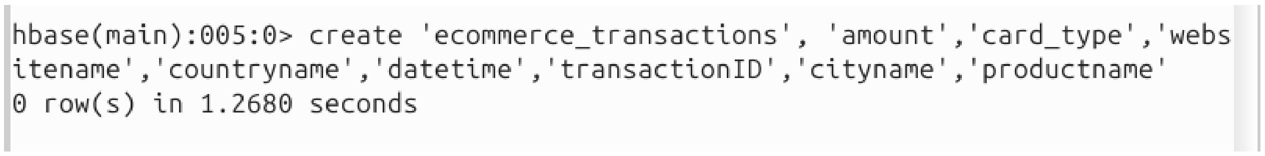
To open guide for tables in HBase: table_help



```
hbase(main):001:0> list
TABLE
0 row(s) in 0.1580 seconds

=> []
hbase(main):002:0> status
1 active master, 0 backup masters, 1 servers, 0 dead, 2.0000 average load

hbase(main):003:0> version
1.2.6, rUnknown, Mon May 29 02:25:32 CDT 2017

hbase(main):004:0> table_help
Help for table-reference commands.

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

Create table in HBase:

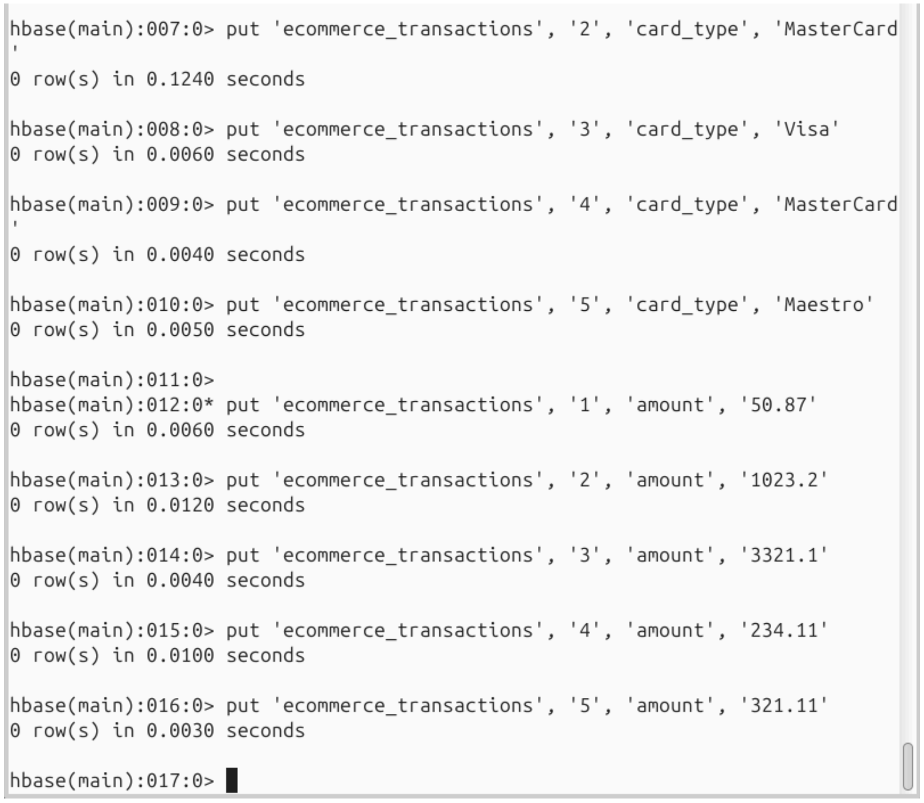
create'ecommerce_transactions,'amount','card_type', 'websitename', 'countryname', 'datetime', 'transactionID', 'cityname', 'productname'



```
hbase(main):005:0> create 'ecommerce_transactions', 'amount','card_type','webs
itename','countryname','datetime','transactionID','cityname','productname'
0 row(s) in 1.2680 seconds
```

Insert some rows for tables in HBase:

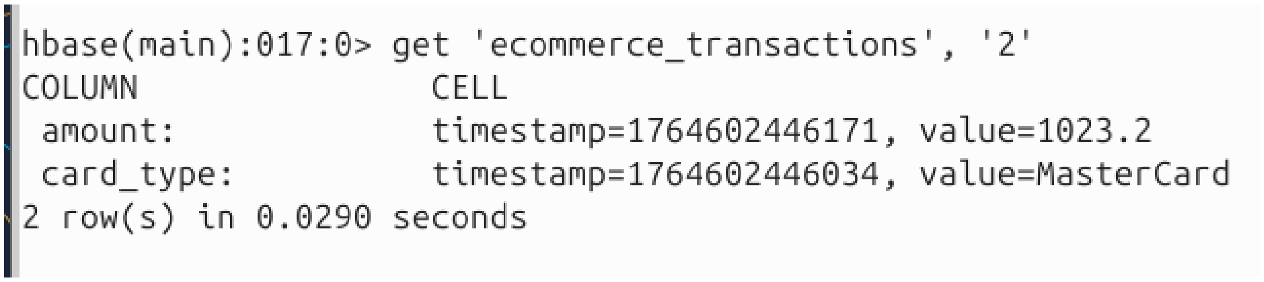
put 'ecommerce_transactions', '2', 'card_type', 'MasterCard'
put 'ecommerce_transactions', '3', 'card_type', 'Visa'
put 'ecommerce_transactions', '4', 'card_type', 'MasterCard' put 'ecommerce_transactions', '5', 'card_type', 'Maestro'
put 'ecommerce_transactions', '1', 'amount', '50.87'
put 'ecommerce_transactions', '2', 'amount', '1023.2'
put 'ecommerce_transactions', '3', 'amount', '3321.1'
put 'ecommerce_transactions', '4', 'amount', '234.11'
put 'ecommerce_transactions', '5', 'amount', '321.11'



```
hbase(main):007:0> put 'ecommerce_transactions', '2', 'card_type', 'MasterCard
'
0 row(s) in 0.1240 seconds

hbase(main):008:0> put 'ecommerce_transactions', '3', 'card_type', 'Visa'
0 row(s) in 0.0060 seconds

hbase(main):009:0> put 'ecommerce_transactions', '4', 'card_type', 'MasterCard
'
0 row(s) in 0.0040 seconds

hbase(main):010:0> put 'ecommerce_transactions', '5', 'card_type', 'Maestro'
0 row(s) in 0.0050 seconds

hbase(main):011:0>
hbase(main):012:0* put 'ecommerce_transactions', '1', 'amount', '50.87'
0 row(s) in 0.0060 seconds

hbase(main):013:0> put 'ecommerce_transactions', '2', 'amount', '1023.2'
0 row(s) in 0.0120 seconds

hbase(main):014:0> put 'ecommerce_transactions', '3', 'amount', '3321.1'
0 row(s) in 0.0040 seconds

hbase(main):015:0> put 'ecommerce_transactions', '4', 'amount', '234.11'
0 row(s) in 0.0100 seconds

hbase(main):016:0> put 'ecommerce_transactions', '5', 'amount', '321.11'
0 row(s) in 0.0030 seconds

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

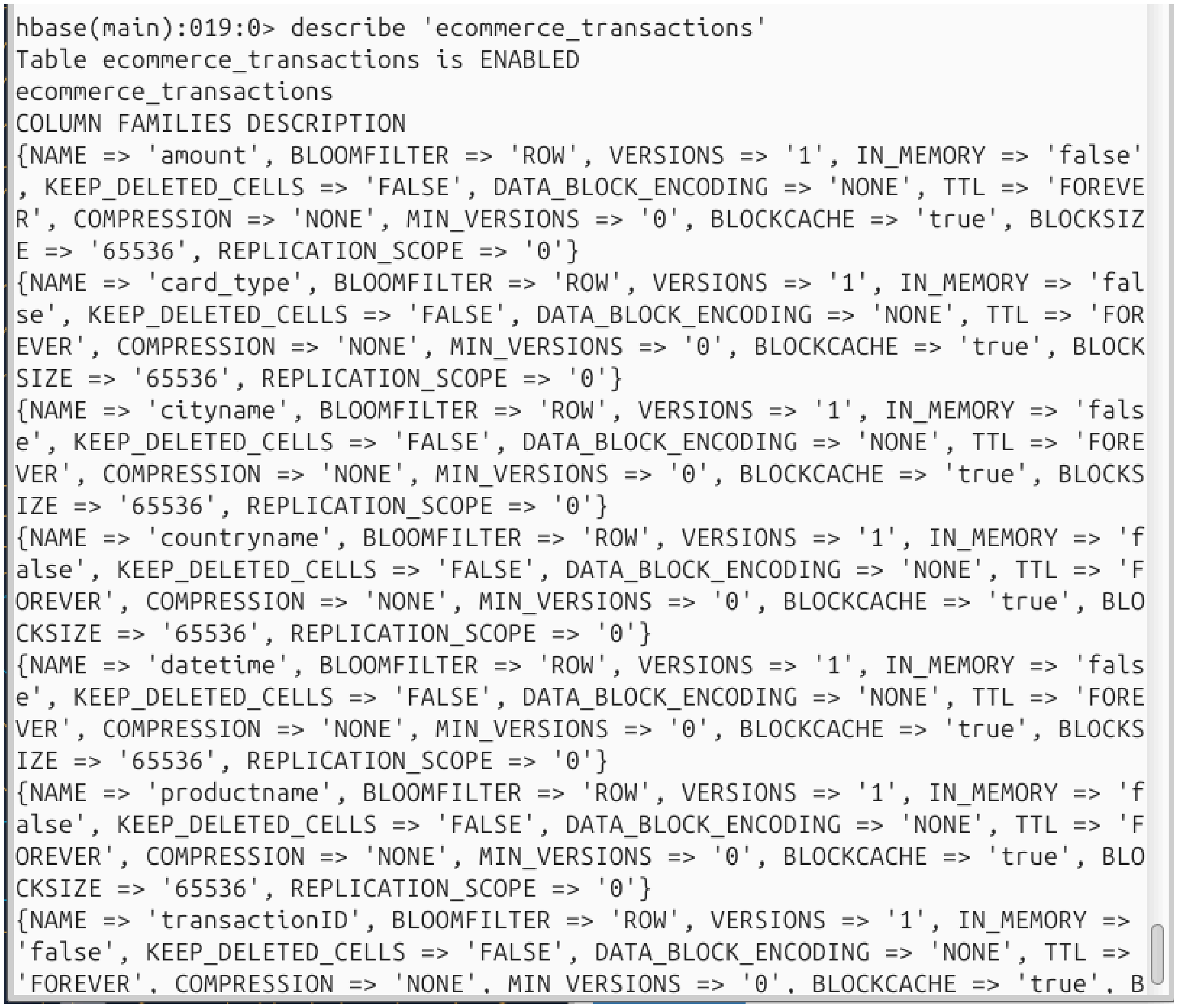
Get transactions for only row value 2 and 3 in HBase:

get 'ecommerce_transactions', '2' get 'ecommerce_transactions', '5'



```
hbase(main):017:0> get 'ecommerce_transactions', '2'
COLUMN                CELL
 amount:              timestamp=1764602446171, value=1023.2
 card_type:           timestamp=1764602446034, value=MasterCard
2 row(s) in 0.0290 seconds
```

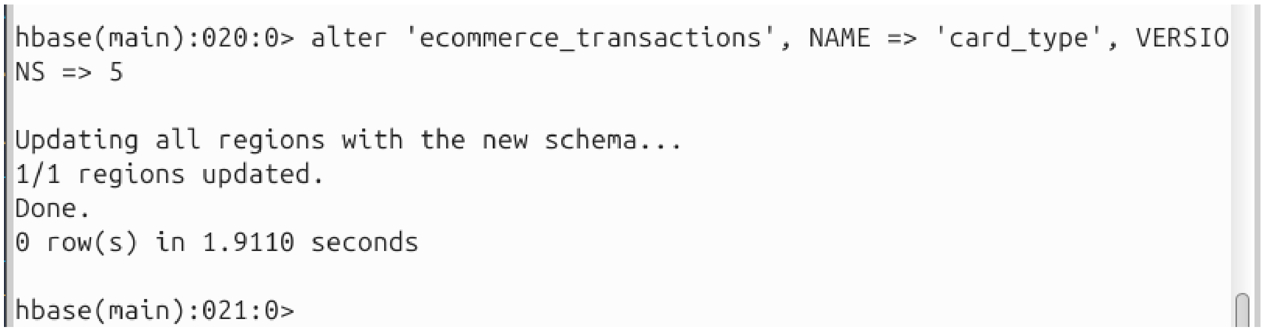
Get description of the table in HBase: describe 'ecommerce_transactions'



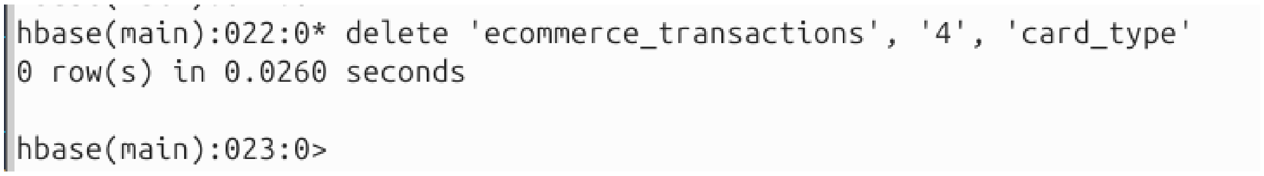
```
hbase(main):019:0> describe 'ecommerce_transactions'
Table ecommerce_transactions is ENABLED
ecommerce_transactions
COLUMN FAMILIES DESCRIPTION
{NAME => 'amount', BLOOMFILTER => 'ROW', VERSIONS => '1', IN_MEMORY => 'false'
, KEEP_DELETED_CELLS => 'FALSE', DATA_BLOCK_ENCODING => 'NONE', TTL => 'FOREVE
R', COMPRESSION => 'NONE', MIN_VERSIONS => '0', BLOCKCACHE => 'true', BLOCKSIZ
E => '65536', REPLICATION_SCOPE => '0'}
{NAME => 'card_type', BLOOMFILTER => 'ROW', VERSIONS => '1', IN_MEMORY => 'fal
se', KEEP_DELETED_CELLS => 'FALSE', DATA_BLOCK_ENCODING => 'NONE', TTL => 'FOR
EVER', COMPRESSION => 'NONE', MIN_VERSIONS => '0', BLOCKCACHE => 'true', BLOCK
SIZE => '65536', REPLICATION_SCOPE => '0'}
{NAME => 'cityname', BLOOMFILTER => 'ROW', VERSIONS => '1', IN_MEMORY => 'fals
e', KEEP_DELETED_CELLS => 'FALSE', DATA_BLOCK_ENCODING => 'NONE', TTL => 'FORE
VER', COMPRESSION => 'NONE', MIN_VERSIONS => '0', BLOCKCACHE => 'true', BLOCKS
IZE => '65536', REPLICATION_SCOPE => '0'}
{NAME => 'countryname', BLOOMFILTER => 'ROW', VERSIONS => '1', IN_MEMORY => 'f
alse', KEEP_DELETED_CELLS => 'FALSE', DATA_BLOCK_ENCODING => 'NONE', TTL => 'F
OREVER', COMPRESSION => 'NONE', MIN_VERSIONS => '0', BLOCKCACHE => 'true', BLO
CKSIZE => '65536', REPLICATION_SCOPE => '0'}
{NAME => 'datetime', BLOOMFILTER => 'ROW', VERSIONS => '1', IN_MEMORY => 'fals
e', KEEP_DELETED_CELLS => 'FALSE', DATA_BLOCK_ENCODING => 'NONE', TTL => 'FORE
VER', COMPRESSION => 'NONE', MIN_VERSIONS => '0', BLOCKCACHE => 'true', BLOCKS
IZE => '65536', REPLICATION_SCOPE => '0'}
{NAME => 'productname', BLOOMFILTER => 'ROW', VERSIONS => '1', IN_MEMORY => 'f
alse', KEEP_DELETED_CELLS => 'FALSE', DATA_BLOCK_ENCODING => 'NONE', TTL => 'F
OREVER', COMPRESSION => 'NONE', MIN_VERSIONS => '0', BLOCKCACHE => 'true', BLO
CKSIZE => '65536', REPLICATION_SCOPE => '0'}
{NAME => 'transactionID', BLOOMFILTER => 'ROW', VERSIONS => '1', IN_MEMORY =>
'false', KEEP_DELETED_CELLS => 'FALSE', DATA_BLOCK_ENCODING => 'NONE', TTL =>
'FOREVER', COMPRESSION => 'NONE', MIN_VERSIONS => '0', BLOCKCACHE => 'true', B
```

Alter is the command used to make changes to an existing table. This command can help change the maximum number of cells of a column family, set and delete table scope operators, and delete a column family from a table:

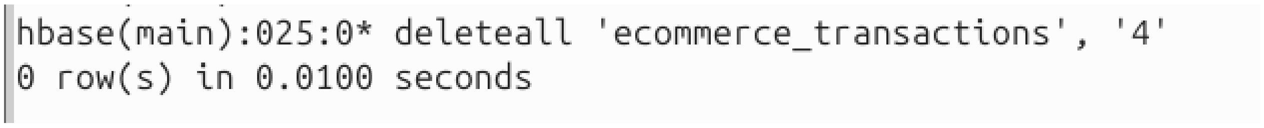
alter 'ecommerce_transactions', NAME => ‘card_type’, VERSIONS =>5


```
hbase(main):020:0> alter 'ecommerce_transactions', NAME => 'card_type', VERSIO
NS => 5

Updating all regions with the new schema...
1/1 regions updated.
Done.
0 row(s) in 1.9110 seconds

hbase(main):021:0>
```

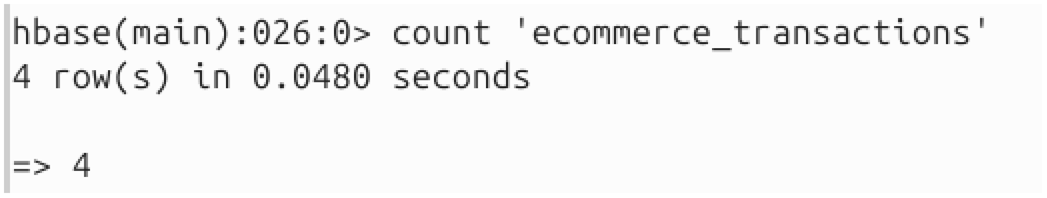
Deleting a cell in a table in HBase: delete 'ecommerce_transactions', ‘4’, ‘card_type’


```
hbase(main):022:0* delete 'ecommerce_transactions', '4', 'card_type'
0 row(s) in 0.0260 seconds

hbase(main):023:0>
```

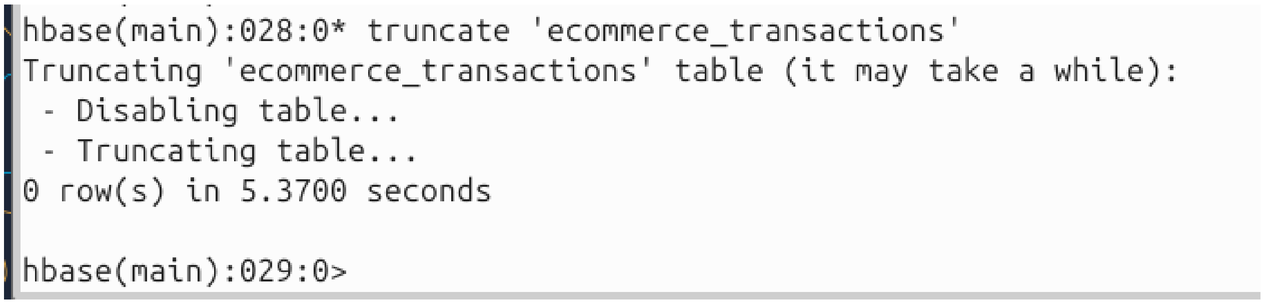
Deleting a row in a table in HBase: deleteall 'ecommerce_transactions', ‘4’


```
hbase(main):025:0* deleteall 'ecommerce_transactions', '4'
0 row(s) in 0.0100 seconds
```

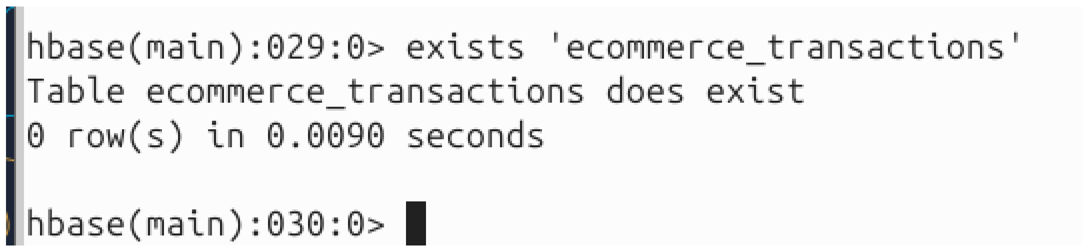
Count number of rows in a table in HBase: count 'ecommerce_transactions'


```
hbase(main):026:0> count 'ecommerce_transactions'
4 row(s) in 0.0480 seconds

=> 4
```

Disable, drop and recreate a table in HBase: truncate 'ecommerce_transactions'

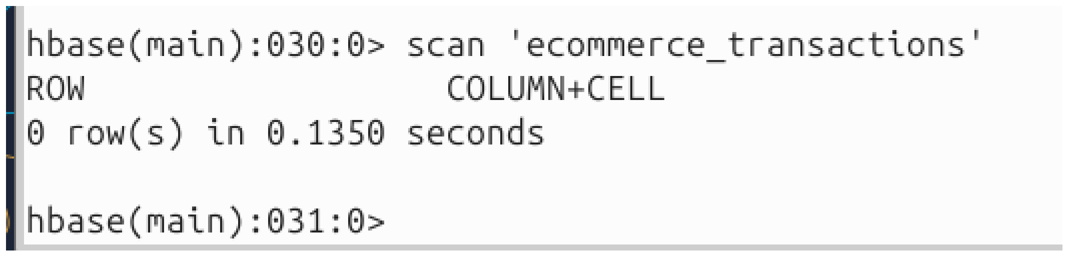

```
hbase(main):028:0* truncate 'ecommerce_transactions'
Truncating 'ecommerce_transactions' table (it may take a while):
 - Disabling table...
 - Truncating table...
0 row(s) in 5.3700 seconds

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

Check if a table is present in HBase: exists 'ecommerce_transactions'



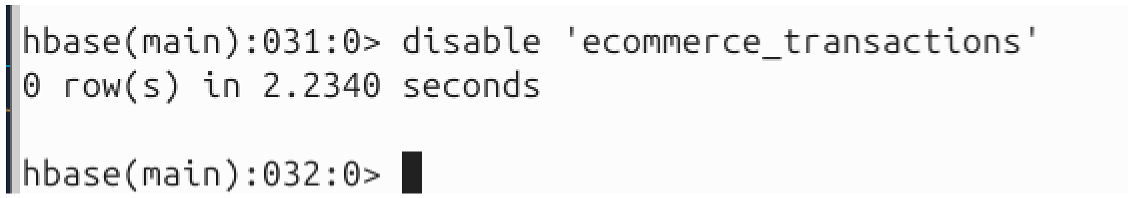
```
hbase(main):029:0> exists 'ecommerce_transactions'
Table ecommerce_transactions does exist
0 row(s) in 0.0090 seconds

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

View all rows in a table of HBase: scan ‘ecommerce_transactions’



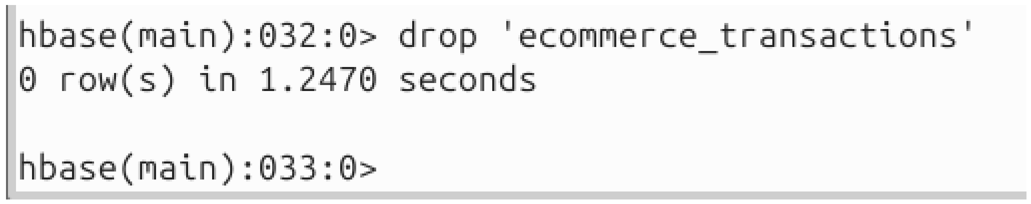
```
hbase(main):030:0> scan 'ecommerce_transactions'
ROW                    COLUMN+CELL
0 row(s) in 0.1350 seconds

hbase(main):031:0>
```

Disable a table of HBase: disable ‘ecommerce_transactions’



```
hbase(main):031:0> disable 'ecommerce_transactions'
0 row(s) in 2.2340 seconds

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

Drop a table of HBase: drop ‘ecommerce_transactions’



```
hbase(main):032:0> drop 'ecommerce_transactions'
0 row(s) in 1.2470 seconds

hbase(main):033:0>
```

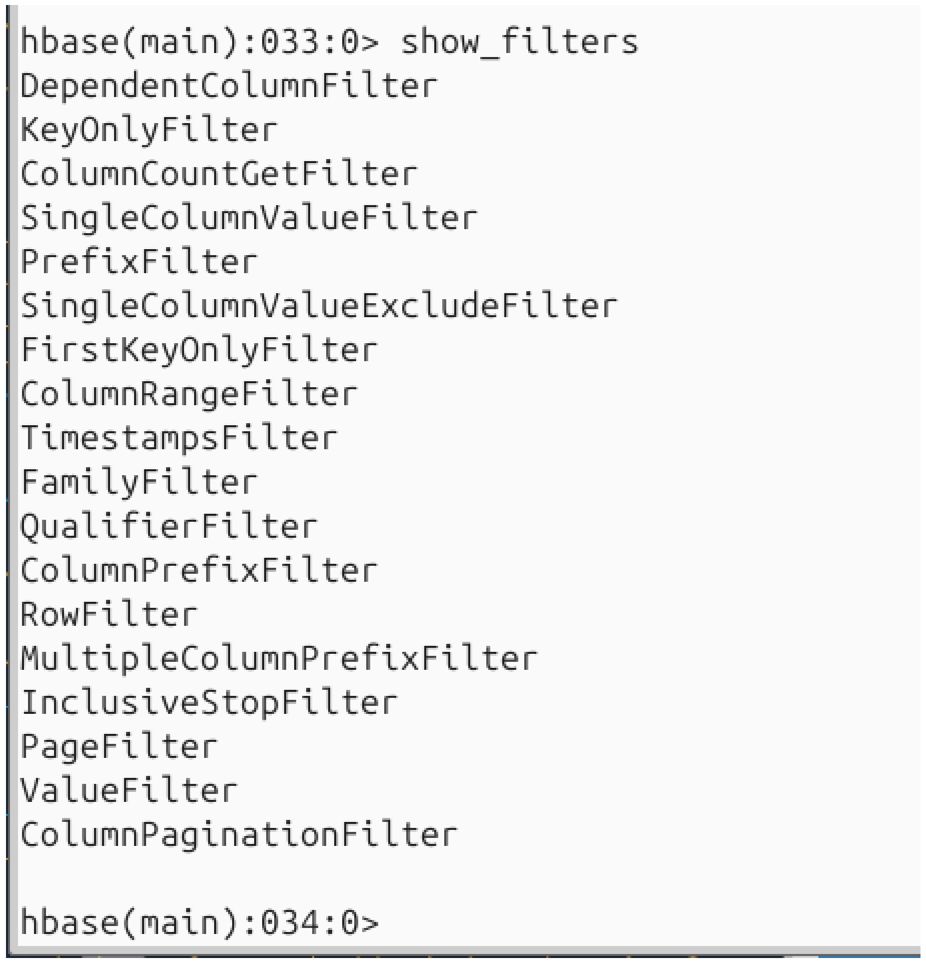
Note: drop won’t work till table has been disabled

Scan Filter Exercises

[ToDo: Paste snapshot of 2 filter outputs]

Using the data inserted in 'ecommerce_transactions' table, enhancing reads of data present in HBase using the scan filtering techniques. There are 6 most commonly used filters, but these can also be combined together for more specific searches.

Check for available filters in HBase to read data more specifically: show_filters (enter in HBase shell)



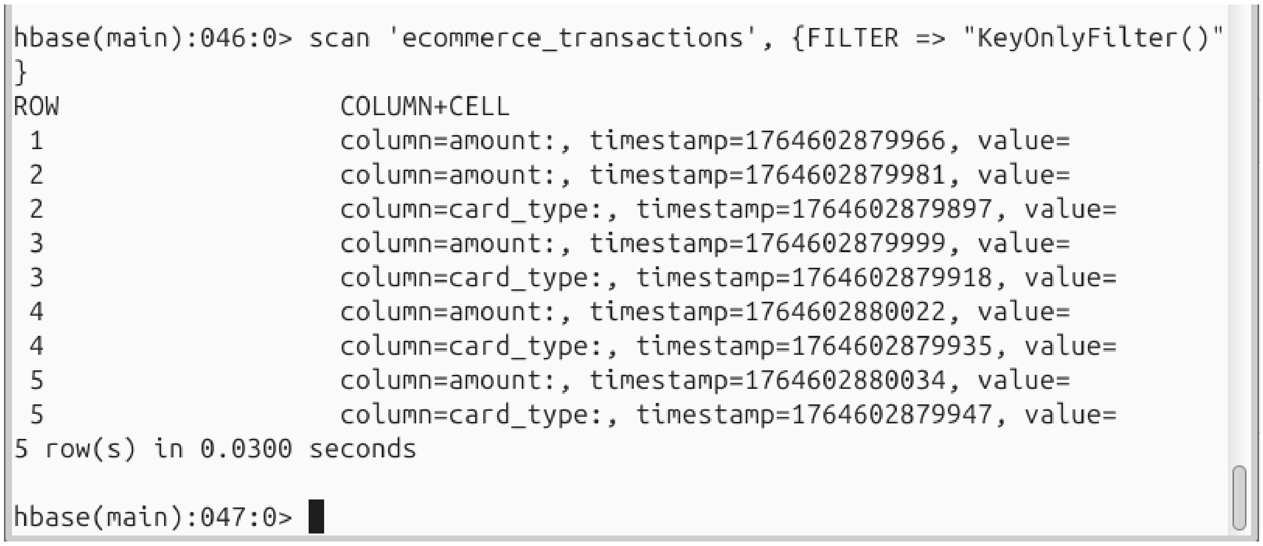
```
hbase(main):033:0> show_filters
DependentColumnFilter
KeyOnlyFilter
ColumnCountGetFilter
SingleColumnValueFilter
PrefixFilter
SingleColumnValueExcludeFilter
FirstKeyOnlyFilter
ColumnRangeFilter
TimestampsFilter
FamilyFilter
QualifierFilter
ColumnPrefixFilter
RowFilter
MultipleColumnPrefixFilter
InclusiveStopFilter
PageFilter
ValueFilter
ColumnPaginationFilter

hbase(main):034:0>
```

Using the data set we put earlier perform the following hands-on exercise:

RECREATED TABLE AND REINSERTED THE DATA AS IT WAS TRUNCATED/DROPPED BEFORE

Using KeyOnlyFilter get the names of columns and column family qualifiers without seeing their values (typically used for a dataset with very large values and we just want to see the attributes):

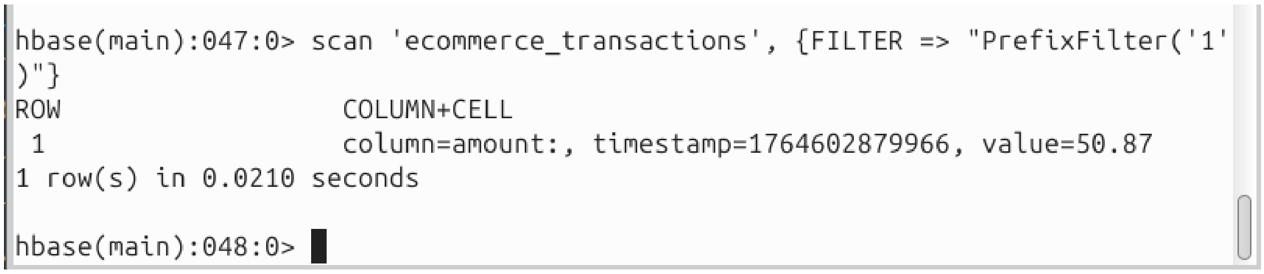
```
scan 'ecommerce_transactions', {FILTER => "KeyOnlyFilter()"}
```



```
hbase(main):046:0> scan 'ecommerce_transactions', {FILTER => "KeyOnlyFilter()"
}
ROW                       COLUMN+CELL
 1                        column=amount:, timestamp=1764602879966, value=
 2                        column=amount:, timestamp=1764602879981, value=
 2                        column=card_type:, timestamp=1764602879897, value=
 3                        column=amount:, timestamp=1764602879999, value=
 3                        column=card_type:, timestamp=1764602879918, value=
 4                        column=amount:, timestamp=1764602880022, value=
 4                        column=card_type:, timestamp=1764602879935, value=
 5                        column=amount:, timestamp=1764602880034, value=
 5                        column=card_type:, timestamp=1764602879947, value=
5 row(s) in 0.0300 seconds

hbase(main):047:0> 
```

Using the PrefixFilter to find any pattern associated with our row keys: Check which row key represents which website for transactions?

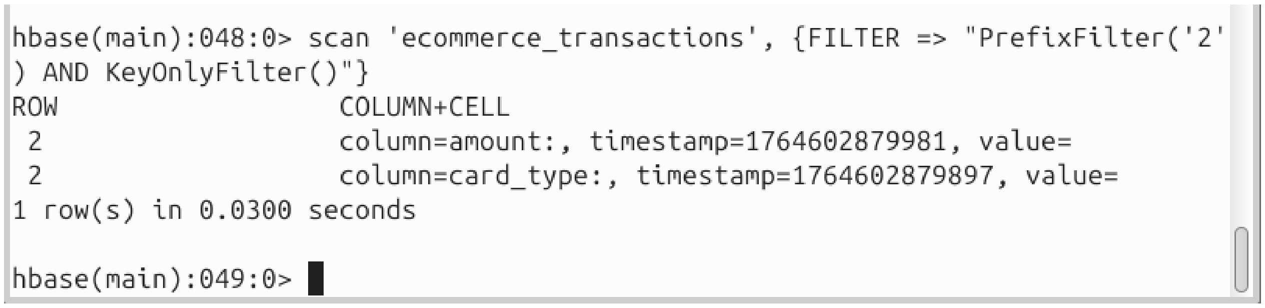
scan 'ecommerce_transactions', {FILTER => "PrefixFilter('1')"}



```
hbase(main):047:0> scan 'ecommerce_transactions', {FILTER => "PrefixFilter('1'
)"}
ROW                       COLUMN+CELL
 1                        column=amount:, timestamp=1764602879966, value=50.87
1 row(s) in 0.0210 seconds

hbase(main):048:0> 
```

Combining filters: Suppose now we want to see which column families are on row key 2? Then we combine the two filters such that:

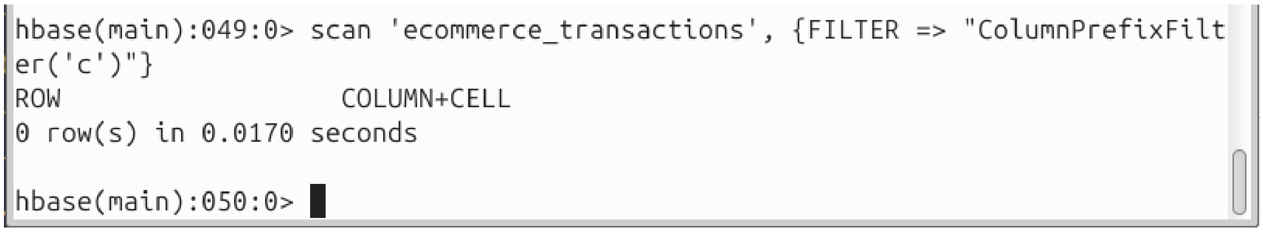
scan 'ecommerce_transactions',{FILTER => "PrefixFilter('2') AND KeyOnlyFilter()"}



```
hbase(main):048:0> scan 'ecommerce_transactions', {FILTER => "PrefixFilter('2'
) AND KeyOnlyFilter()"}
ROW                       COLUMN+CELL
 2                        column=amount:, timestamp=1764602879981, value=
 2                        column=card_type:, timestamp=1764602879897, value=
1 row(s) in 0.0300 seconds

hbase(main):049:0> 
```

Using the ColumnPrefixFilter, find the column families which start with 'c'

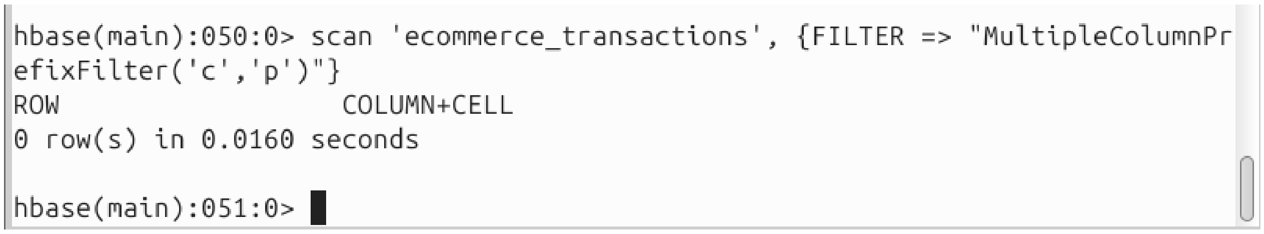
scan 'ecommerce_transactions', {FILTER => "ColumnPrefixFilter('c')"}



```
hbase(main):049:0> scan 'ecommerce_transactions', {FILTER => "ColumnPrefixFilt
er('c')"}
ROW                     COLUMN+CELL
0 row(s) in 0.0170 seconds

hbase(main):050:0> 
```

Similarly, specify two arguments in this filter and find all column families which start with 'c' or 'p'

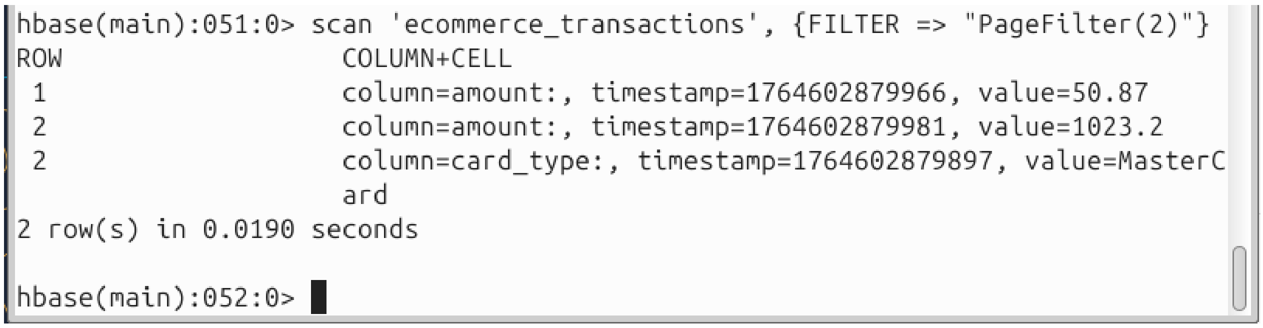
scan 'ecommerce_transactions', {FILTER => "MultipleColumnPrefixFilter('c','p')"}



```
hbase(main):050:0> scan 'ecommerce_transactions', {FILTER => "MultipleColumnPr
efixFilter('c','p')"}
ROW                     COLUMN+CELL
0 row(s) in 0.0160 seconds

hbase(main):051:0> 
```

Using the PageFilter, read first two rows of the HBase table 'ecommerce_transactions':

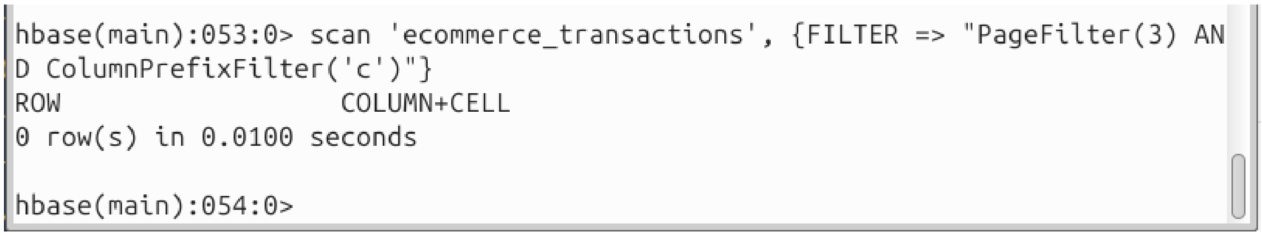
scan 'ecommerce_transactions', {FILTER => "PageFilter(2)"}



```
hbase(main):051:0> scan 'ecommerce_transactions', {FILTER => "PageFilter(2)"}
ROW                     COLUMN+CELL
 1                      column=amount:, timestamp=1764602879966, value=50.87
 2                      column=amount:, timestamp=1764602879981, value=1023.2
 2                      column=card_type:, timestamp=1764602879897, value=MasterC
                        ard
2 row(s) in 0.0190 seconds

hbase(main):052:0> 
```

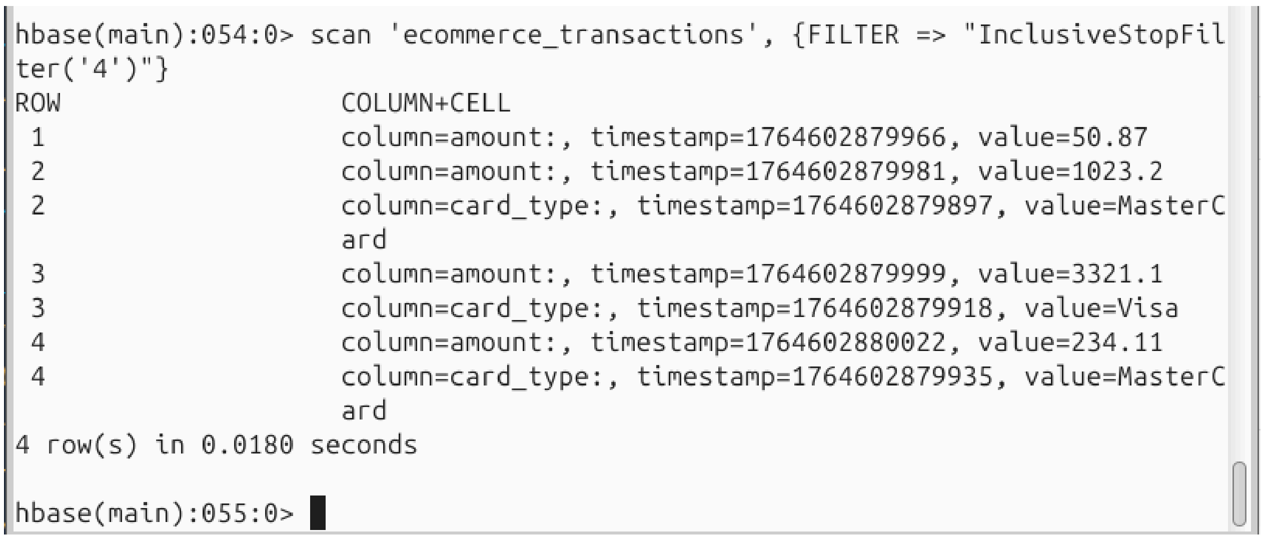
Combining this with another filter, get 3 rows with the column family starting with the letter 'c'

scan 'ecommerce_transactions', {FILTER => "PageFilter(3) AND ColumnPrefixFilter('c')"}



```
hbase(main):053:0> scan 'ecommerce_transactions', {FILTER => "PageFilter(3) AN
D ColumnPrefixFilter('c')"}
ROW                     COLUMN+CELL
0 row(s) in 0.0100 seconds

hbase(main):054:0>
```

Using the InclusiveStopFilter, scan from start till the specified row 4 (including 4 in it) scan 'ecommerce_transactions', {FILTER => "InclusiveStopFilter('4')"}

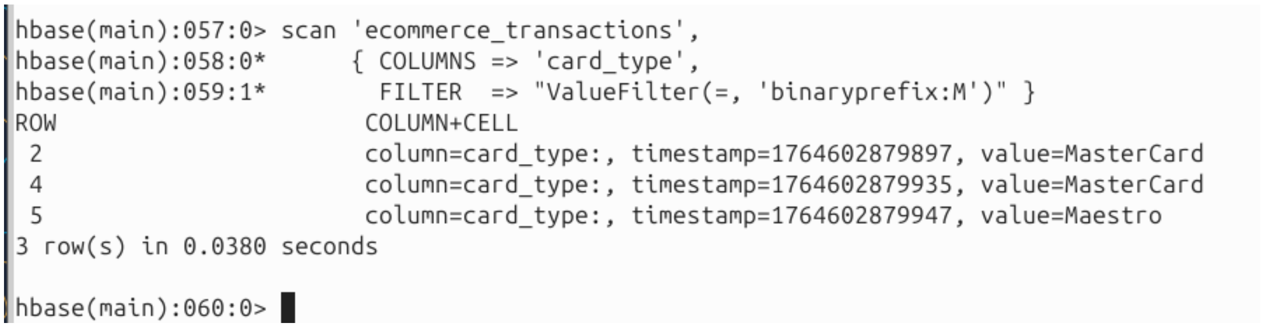


```
hbase(main):054:0> scan 'ecommerce_transactions', {FILTER => "InclusiveStopFil
ter('4')"}
ROW                    COLUMN+CELL
 1                     column=amount:, timestamp=1764602879966, value=50.87
 2                     column=amount:, timestamp=1764602879981, value=1023.2
 2                     column=card_type:, timestamp=1764602879897, value=MasterC
                       ard
 3                     column=amount:, timestamp=1764602879999, value=3321.1
 3                     column=card_type:, timestamp=1764602879918, value=Visa
 4                     column=amount:, timestamp=1764602880022, value=234.11
 4                     column=card_type:, timestamp=1764602879935, value=MasterC
                       ard
4 row(s) in 0.0180 seconds

hbase(main):055:0>
```

Now use the ValueFilter to do some basic search on values of the data. This works similar to the 'like' command of SQL. Find the card type starting with 'M'

scan 'ecommerce_transactions', {COLUMNS => 'card_type', FILTER => "ValueFilter(=,'binary prefix:M')"}

```
scan 'ecommerce_transactions',
   { COLUMNS => 'card_type',
     FILTER  => "ValueFilter(=, 'binaryprefix:M')" }   # // CHANGED
```



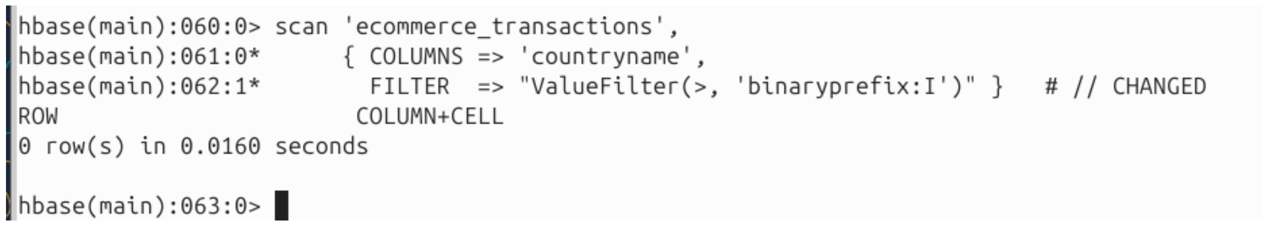
```
hbase(main):057:0> scan 'ecommerce_transactions',
hbase(main):058:0*      { COLUMNS => 'card_type',
hbase(main):059:1*        FILTER  => "ValueFilter(=, 'binaryprefix:M')" }
ROW                    COLUMN+CELL
 2                     column=card_type:, timestamp=1764602879897, value=MasterCard
 4                     column=card_type:, timestamp=1764602879935, value=MasterCard
 5                     column=card_type:, timestamp=1764602879947, value=Maestro
3 row(s) in 0.0380 seconds

hbase(main):060:0>
```

Return all countries starting with letters > 'I':

scan 'ecommerce_transactions', {COLUMNS => 'countryname', FILTER => "ValueFilter(>,'binary prefix:I')"}

```
scan 'ecommerce_transactions',
   { COLUMNS => 'countryname',
```

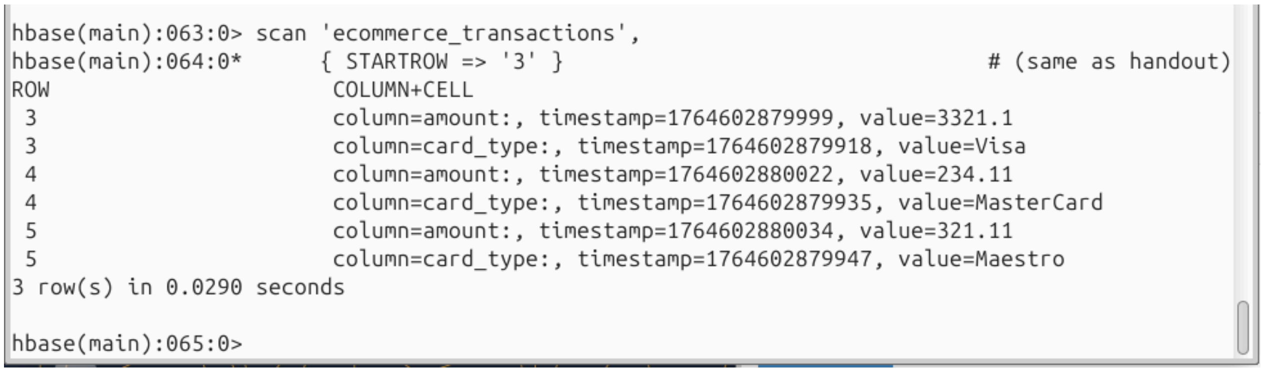
    FILTER => "ValueFilter(>, 'binaryprefix:I')" }   # // CHANGED



```
hbase(main):060:0> scan 'ecommerce_transactions',
hbase(main):061:0*       { COLUMNS => 'countryname',
hbase(main):062:1*         FILTER   => "ValueFilter(>, 'binaryprefix:I')" }    # // CHANGED
ROW                     COLUMN+CELL
0 row(s) in 0.0160 seconds

hbase(main):063:0>
```

Using the STARTROW, scan only rows from row key 3 onwards:

scan 'ecommerce_transactions', {STARTROW => '3'}

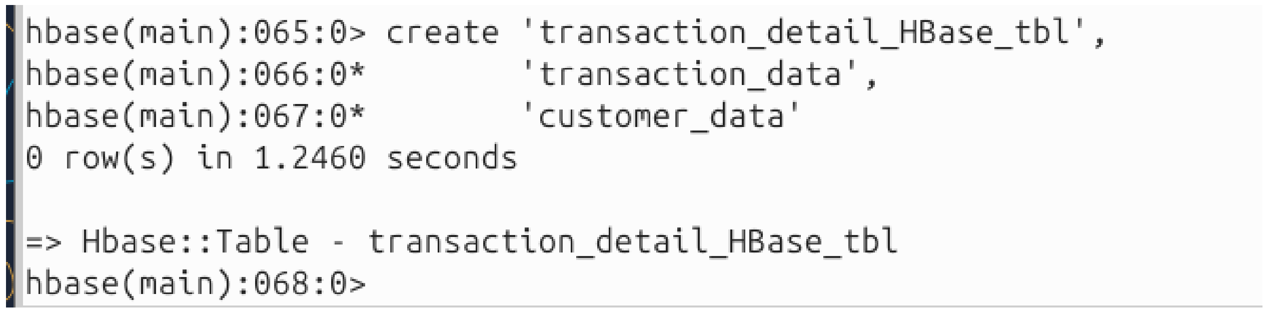


```
hbase(main):063:0> scan 'ecommerce_transactions',
hbase(main):064:0*       { STARTROW => '3' }                                # (same as handout)
ROW                      COLUMN+CELL
 3                       column=amount:, timestamp=1764602879999, value=3321.1
 3                       column=card_type:, timestamp=1764602879918, value=Visa
 4                       column=amount:, timestamp=1764602880022, value=234.11
 4                       column=card_type:, timestamp=1764602879935, value=MasterCard
 5                       column=amount:, timestamp=1764602880034, value=321.11
 5                       column=card_type:, timestamp=1764602879947, value=Maestro
3 row(s) in 0.0290 seconds

hbase(main):065:0>
```

A Use Case Combining Hive and HBase

Definition of use case: *To implement HBase for big data storage in NoSQL form and run that with integration to Hive HQL language in order to reuse the existing SQL queries* Starting HBase shell: hbase shell

Creating table in HBase for e-commerce transactions data: create 'transaction_detail_HBase_tbl','transaction_data','customer_data'



```
hbase(main):065:0> create 'transaction_detail_HBase_tbl',
hbase(main):066:0*         'transaction_data',
hbase(main):067:0*         'customer_data'
0 row(s) in 1.2460 seconds

=> Hbase::Table - transaction_detail_HBase_tbl
hbase(main):068:0>
```

Note: the column families are declared in the create table command in order to create it successfully in HBase shell.

Entering some initial values to populate table:

[ToDo: Paste snapshot of below] put

'transaction_detail_HBase_tbl','1','transaction_data:transaction_amount','50.85' put

'transaction_detail_HBase_tbl','1','transaction_data:transaction_card_type','MasterCard' put

'transaction_detail_HBase_tbl','1','transaction_data:transaction_ecommerce_website_na
me','www.ebay.com' put
'transaction_detail_HBase_tbl','1','transaction_data:transaction_datetime','2019-05-
14 15:24:12' put

'transaction_detail_HBase_tbl','1','transaction_data:transaction_product_name','Laptop' put

'transaction_detail_HBase_tbl','1','customer_data:transaction_ci ty_name','Mumbai' put

'transaction_detail_HBase_tbl','1','customer_data:transaction_co untry_name','India' put

'transaction_detail_HBase_tbl','2','transaction_data:transaction_amount','259.12' put

'transaction_detail_HBase_tbl','2','transaction_data:transaction_card_type','MasterCard' put
'transaction_detail_HBase_tbl','2','transaction_data:transaction_ecommerce_website_na
me','www.amazon.com' put
'transaction_detail_HBase_tbl','2','transaction_data:transaction_datetime','2019-05-
14 15:24:13'

put 'transaction_detail_HBase_tbl','2','transaction_data:transaction_product_name','Wrist
Band' put 'transaction_detail_HBase_tbl','2','customer_data:transaction_ci ty_name','Pune' put

'transaction_detail_HBase_tbl','2','customer_data:transaction_co untry_name','India' put

'transaction_detail_HBase_tbl','3','transaction_data:transaction_amount','328.16' put

'transaction_detail_HBase_tbl','3','transaction_data:transaction

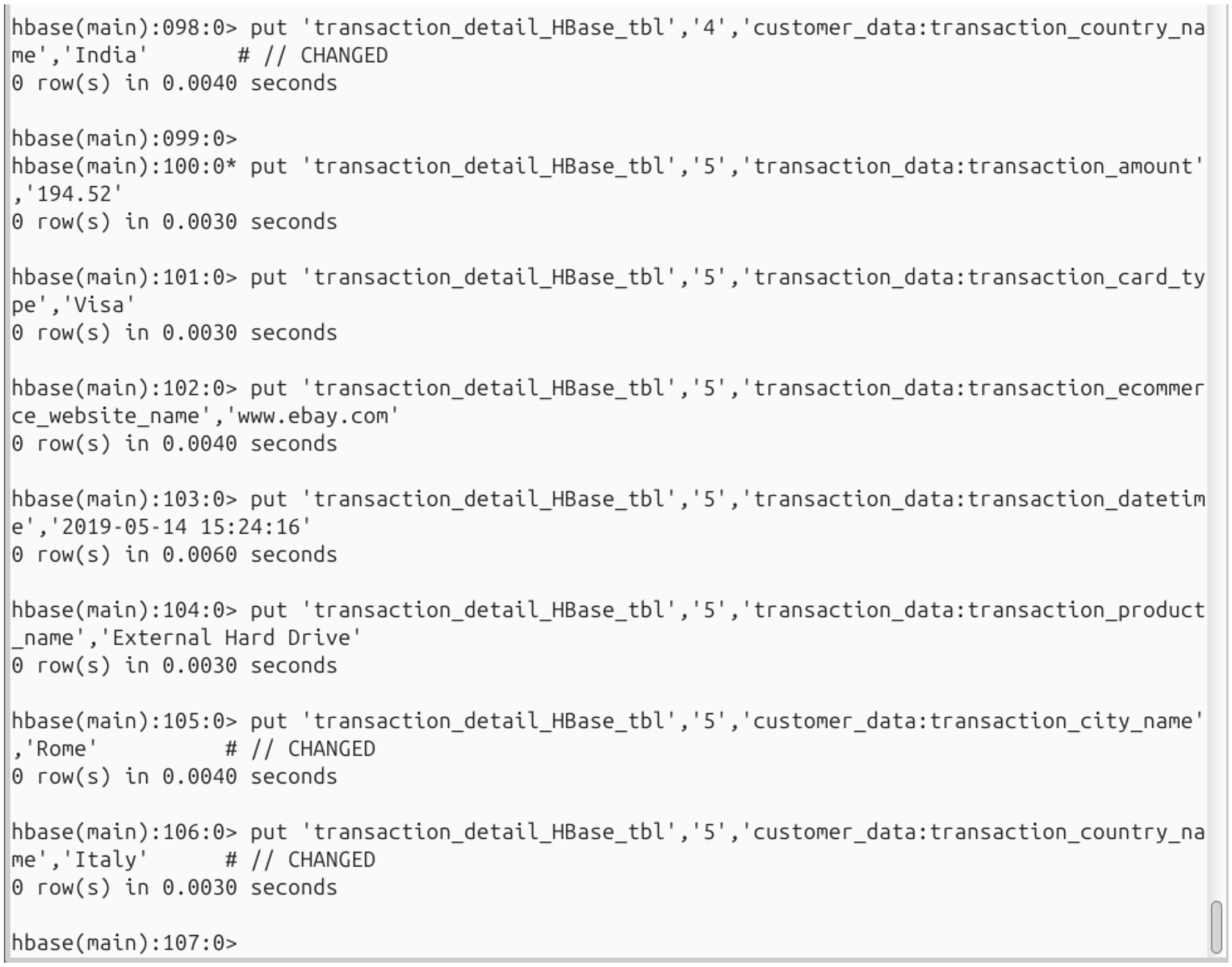
_card_type','MasterCard' put

'transaction_detail_HBase_tbl','3','transaction_data:transaction_ecommerce_website_na
me','www.flipkart.com' put
'transaction_detail_HBase_tbl','3','transaction_data:transaction_datetime','2019-05-
14 15:24:14'

put 'transaction_detail_HBase_tbl','3','transaction_data:transaction_product_name','TV Stand'

put 'transaction_detail_HBase_tbl','3','customer_data:transaction_ci ty_name','New York City'

```
put 'transaction_detail_HBase_tbl','3','customer_data:transaction_co untry_name','United
States' put 'transaction_detail_HBase_tbl','4','transaction_data:transaction_amount','399.06'
put 'transaction_detail_HBase_tbl','4','transaction_data:transaction_card_type','Visa'  put
'transaction_detail_HBase_tbl','4','transaction_data:transaction_ecommerce_website_na
me','www.snapdeal.com' put
'transaction_detail_HBase_tbl','4','transaction_data:transaction_datetime','2019-0514
15:24:15'
put 'transaction_detail_HBase_tbl','4','transaction_data:transaction_product_name','TV Stand'
put 'transaction_detail_HBase_tbl','4','customer_data:transaction_ci ty_name','New Delhi' put
'transaction_detail_HBase_tbl','4','customer_data:transaction_co untry_name','India' put
'transaction_detail_HBase_tbl','5','transaction_data:transaction_amount','194.52'  put
'transaction_detail_HBase_tbl','5','transaction_data:transaction_card_type','Visa'  put
'transaction_detail_HBase_tbl','5','transaction_data:transaction_ecommerce_website_na
me','www.ebay.com'  put
'transaction_detail_HBase_tbl','5','transaction_data:transaction_datetime','2019-05-
14 15:24:16' put
'transaction_detail_HBase_tbl','5','transaction_data:transaction_product_name','External
Hard Drive'  put
'transaction_detail_HBase_tbl','5','customer_data:transaction_city_name','Rome' put
'transaction_detail_HBase_tbl','5','customer_data:transaction_country_name','Italy'
```

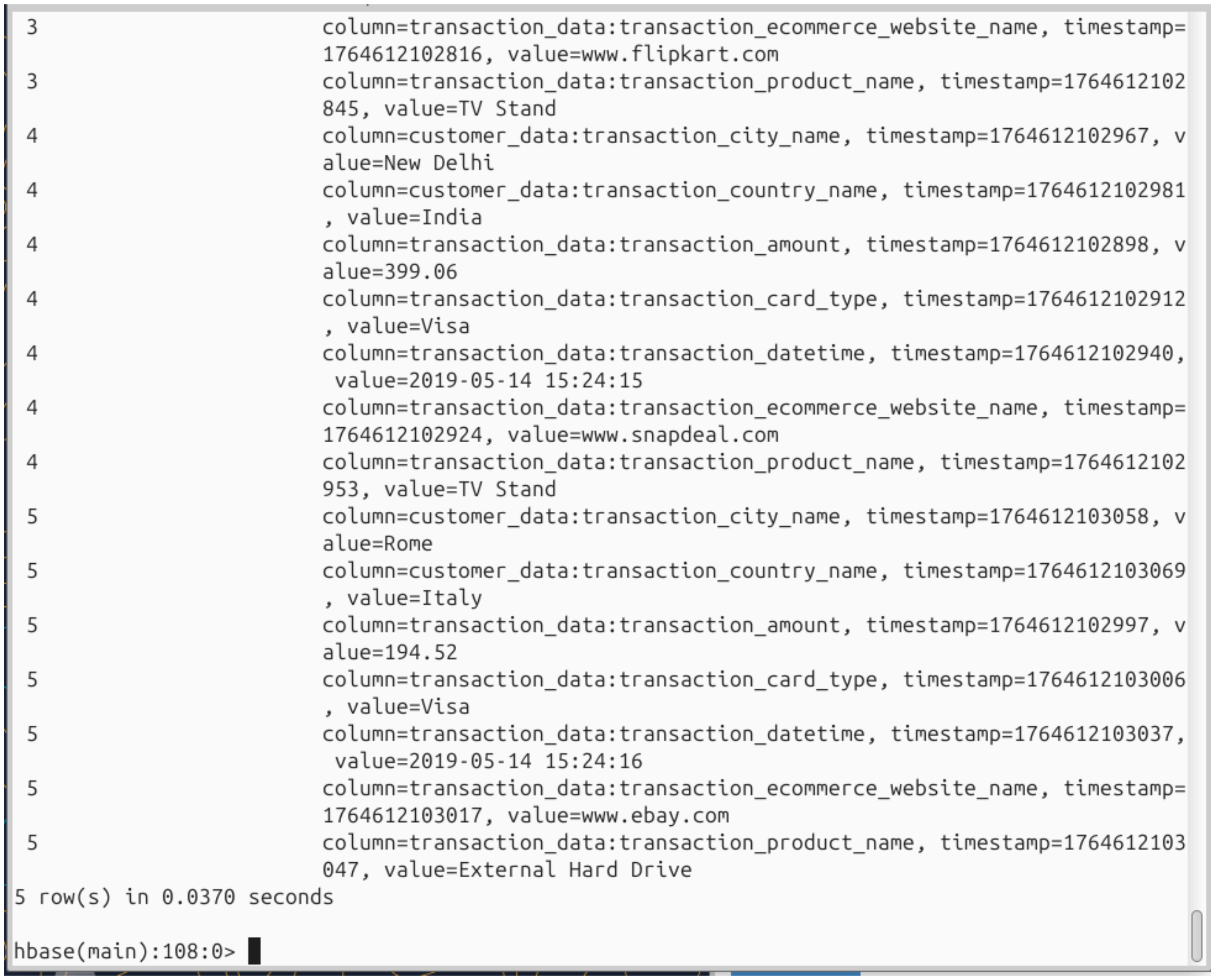


```
hbase(main):098:0> put 'transaction_detail_HBase_tbl','4','customer_data:transaction_country_na
me','India'       # // CHANGED
0 row(s) in 0.0040 seconds

hbase(main):099:0>
hbase(main):100:0* put 'transaction_detail_HBase_tbl','5','transaction_data:transaction_amount'
,'194.52'
0 row(s) in 0.0030 seconds

hbase(main):101:0> put 'transaction_detail_HBase_tbl','5','transaction_data:transaction_card_ty
pe','Visa'
0 row(s) in 0.0030 seconds

hbase(main):102:0> put 'transaction_detail_HBase_tbl','5','transaction_data:transaction_ecommer
ce_website_name','www.ebay.com'
0 row(s) in 0.0040 seconds

hbase(main):103:0> put 'transaction_detail_HBase_tbl','5','transaction_data:transaction_datetim
e','2019-05-14 15:24:16'
0 row(s) in 0.0060 seconds

hbase(main):104:0> put 'transaction_detail_HBase_tbl','5','transaction_data:transaction_product
_name','External Hard Drive'
0 row(s) in 0.0030 seconds

hbase(main):105:0> put 'transaction_detail_HBase_tbl','5','customer_data:transaction_city_name'
,'Rome'          # // CHANGED
0 row(s) in 0.0040 seconds

hbase(main):106:0> put 'transaction_detail_HBase_tbl','5','customer_data:transaction_country_na
me','Italy'      # // CHANGED
0 row(s) in 0.0030 seconds

hbase(main):107:0>
```

View the populated table to see the rows entered in the previous step:

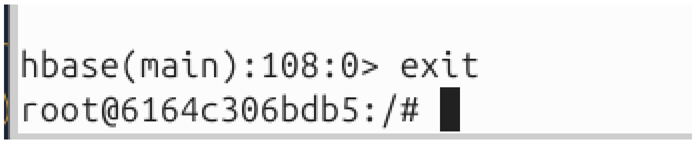
[ToDo: Paste snapshot of below]

scan 'transaction_detail_HBase_tbl'


```
 3                          column=transaction_data:transaction_ecommerce_website_name, timestamp=
                            1764612102816, value=www.flipkart.com
 3                          column=transaction_data:transaction_product_name, timestamp=1764612102
                            845, value=TV Stand
 4                          column=customer_data:transaction_city_name, timestamp=1764612102967, v
                            alue=New Delhi
 4                          column=customer_data:transaction_country_name, timestamp=1764612102981
                            , value=India
 4                          column=transaction_data:transaction_amount, timestamp=1764612102898, v
                            alue=399.06
 4                          column=transaction_data:transaction_card_type, timestamp=1764612102912
                            , value=Visa
 4                          column=transaction_data:transaction_datetime, timestamp=1764612102940,
                             value=2019-05-14 15:24:15
 4                          column=transaction_data:transaction_ecommerce_website_name, timestamp=
                            1764612102924, value=www.snapdeal.com
 4                          column=transaction_data:transaction_product_name, timestamp=1764612102
                            953, value=TV Stand
 5                          column=customer_data:transaction_city_name, timestamp=1764612103058, v
                            alue=Rome
 5                          column=customer_data:transaction_country_name, timestamp=1764612103069
                            , value=Italy
 5                          column=transaction_data:transaction_amount, timestamp=1764612102997, v
                            alue=194.52
 5                          column=transaction_data:transaction_card_type, timestamp=1764612103006
                            , value=Visa
 5                          column=transaction_data:transaction_datetime, timestamp=1764612103037,
                             value=2019-05-14 15:24:16
 5                          column=transaction_data:transaction_ecommerce_website_name, timestamp=
                            1764612103017, value=www.ebay.com
 5                          column=transaction_data:transaction_product_name, timestamp=1764612103
                            047, value=External Hard Drive
5 row(s) in 0.0370 seconds

hbase(main):108:0>
```

Once table has been populated, we exit the HBase shell and enter the Hive shell for integration purposes:

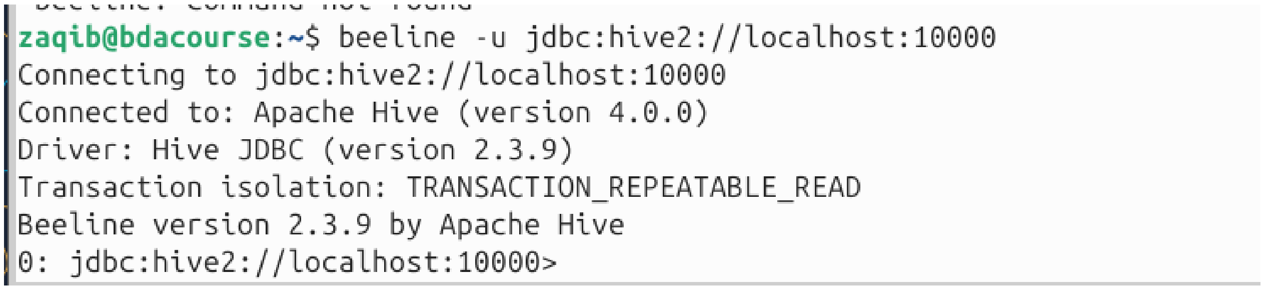
Exiting HBase Shell: exit


```
hbase(main):108:0> exit
root@6164c306bdb5:/#
```

Next we enter Hive shell to make a connecting table to our HBase table (Hive docker should also be running)

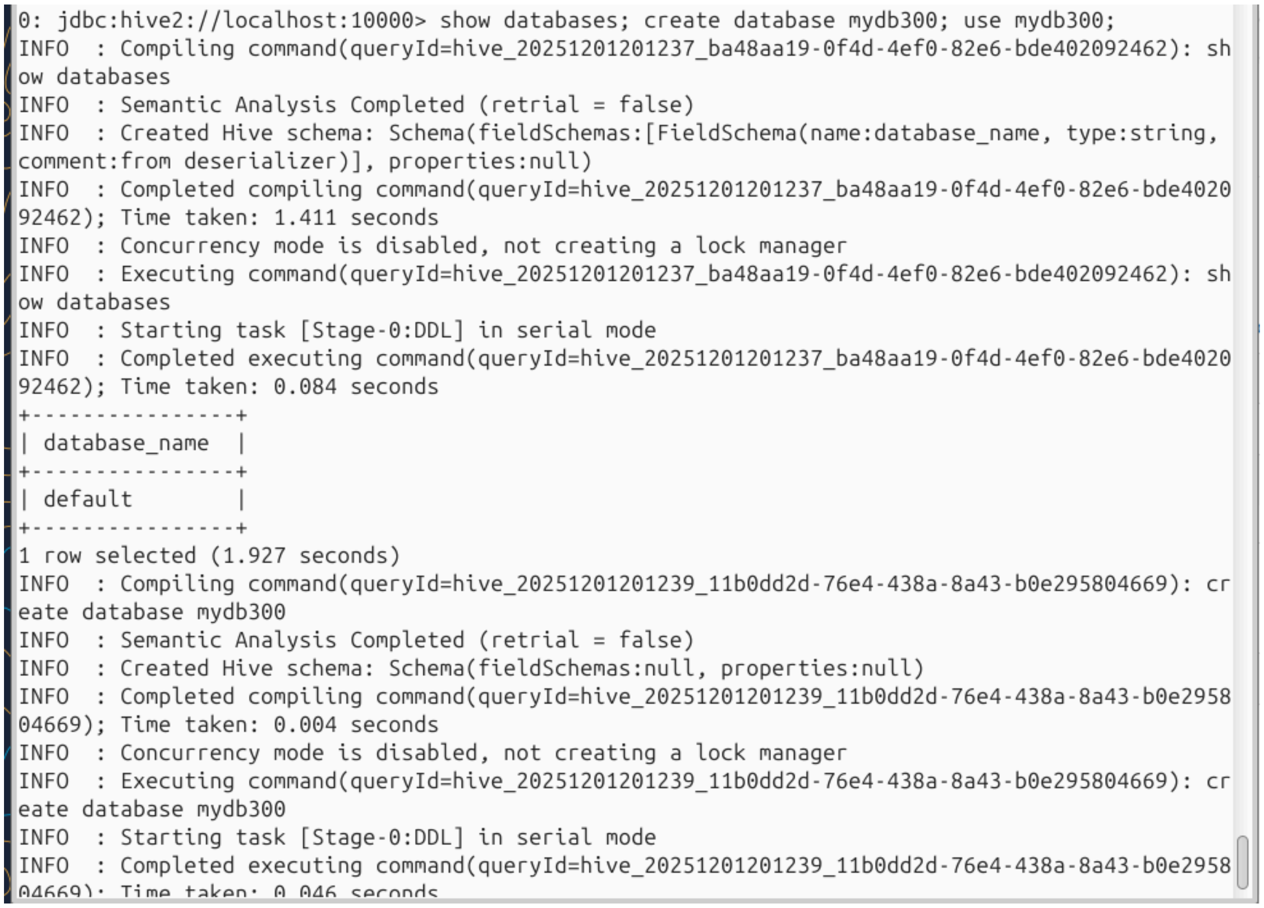
Enter Hive shell (execute the beeline command)

beeline -u jdbc:hive2://hive-server:10000



```
zaqib@bdacourse:~$ beeline -u jdbc:hive2://localhost:10000
Connecting to jdbc:hive2://localhost:10000
Connected to: Apache Hive (version 4.0.0)
Driver: Hive JDBC (version 2.3.9)
Transaction isolation: TRANSACTION_REPEATABLE_READ
Beeline version 2.3.9 by Apache Hive
0: jdbc:hive2://localhost:10000>
```

show databases; create database mydb300; use mydb300;



```
0: jdbc:hive2://localhost:10000> show databases; create database mydb300; use mydb300;
INFO  : Compiling command(queryId=hive_20251201201237_ba48aa19-0f4d-4ef0-82e6-bde402092462): sh
ow databases
INFO  : Semantic Analysis Completed (retrial = false)
INFO  : Created Hive schema: Schema(fieldSchemas:[FieldSchema(name:database_name, type:string,
comment:from deserializer)], properties:null)
INFO  : Completed compiling command(queryId=hive_20251201201237_ba48aa19-0f4d-4ef0-82e6-bde4020
92462); Time taken: 1.411 seconds
INFO  : Concurrency mode is disabled, not creating a lock manager
INFO  : Executing command(queryId=hive_20251201201237_ba48aa19-0f4d-4ef0-82e6-bde402092462): sh
ow databases
INFO  : Starting task [Stage-0:DDL] in serial mode
INFO  : Completed executing command(queryId=hive_20251201201237_ba48aa19-0f4d-4ef0-82e6-bde4020
92462); Time taken: 0.084 seconds
+----------------+
| database_name  |
+----------------+
| default        |
+----------------+
1 row selected (1.927 seconds)
INFO  : Compiling command(queryId=hive_20251201201239_11b0dd2d-76e4-438a-8a43-b0e295804669): cr
eate database mydb300
INFO  : Semantic Analysis Completed (retrial = false)
INFO  : Created Hive schema: Schema(fieldSchemas:null, properties:null)
INFO  : Completed compiling command(queryId=hive_20251201201239_11b0dd2d-76e4-438a-8a43-b0e2958
04669); Time taken: 0.004 seconds
INFO  : Concurrency mode is disabled, not creating a lock manager
INFO  : Executing command(queryId=hive_20251201201239_11b0dd2d-76e4-438a-8a43-b0e295804669): cr
eate database mydb300
INFO  : Starting task [Stage-0:DDL] in serial mode
INFO  : Completed executing command(queryId=hive_20251201201239_11b0dd2d-76e4-438a-8a43-b0e2958
04669); Time taken: 0.046 seconds
```

Now create an external table in hive which points to the table with data kept in HBase following an architecture such that:

[ToDo: Paste snapshot of below]

CREATE EXTERNAL TABLE transaction_detail_hive_tbl(transaction_id int, transaction_card_type string, transaction_ecommerce_website_name string, transaction_product_name string, transaction_datetime string, transaction_amount double, transaction_city_name string, transaction_country_name string) STORED BY 'org.apache.hadoop.hive.HBase.HBaseStorageHandler'

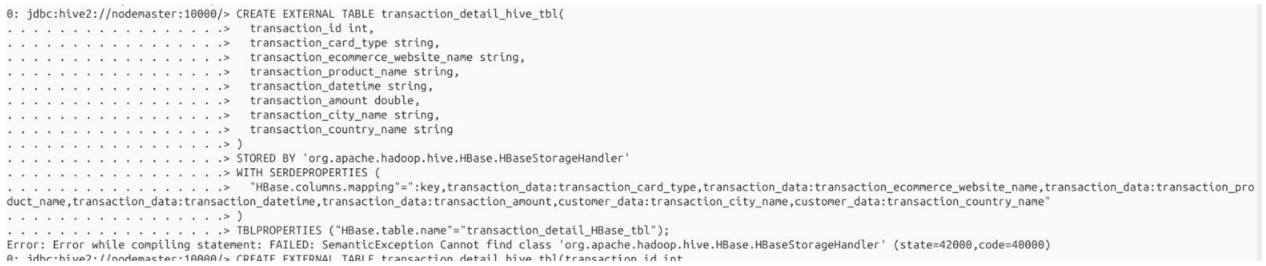
WITH SERDEPROPERTIES

("HBase.columns.mapping"=":key,transaction_data:transaction_card_type,transaction_data:transaction_ecommerce_website_name,transaction_data:transaction_product_name,transaction_data:transaction_datetime,transaction_data:transaction_amount,customer_data:tran saction_city_name,customer_data:transaction_country_name")

TBLPROPERTIES

("HBase.table.name"="transaction_detail_HBase_tbl");

For this command, we were initially getting errors that said HBaseStorageHandler class not found, this meant that the Hive server was not able to establish a connection with the Hbase.



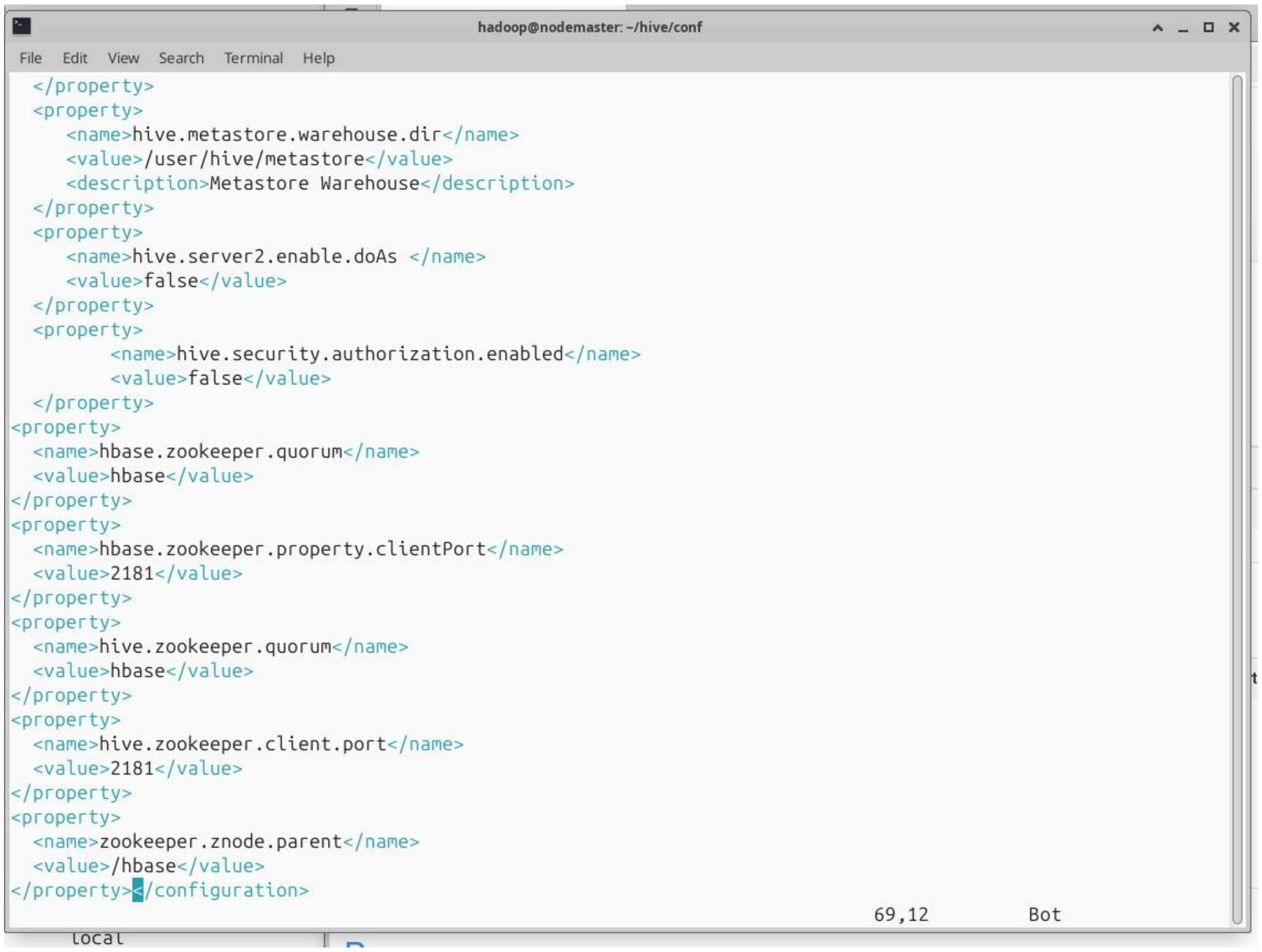
```
0: jdbc:hive2://nodemaster:10000/> CREATE EXTERNAL TABLE transaction_detail_hive_tbl(
. . . . . . . . . . . . . . . . .>   transaction_id int,
. . . . . . . . . . . . . . . . .>   transaction_card_type string,
. . . . . . . . . . . . . . . . .>   transaction_ecommerce_website_name string,
. . . . . . . . . . . . . . . . .>   transaction_product_name string,
. . . . . . . . . . . . . . . . .>   transaction_datetime string,
. . . . . . . . . . . . . . . . .>   transaction_amount double,
. . . . . . . . . . . . . . . . .>   transaction_city_name string,
. . . . . . . . . . . . . . . . .>   transaction_country_name string
. . . . . . . . . . . . . . . . .> )
. . . . . . . . . . . . . . . . .> STORED BY 'org.apache.hadoop.hive.HBase.HBaseStorageHandler'
. . . . . . . . . . . . . . . . .> WITH SERDEPROPERTIES (
. . . . . . . . . . . . . . . . .>   "HBase.columns.mapping"=":key,transaction_data:transaction_card_type,transaction_data:transaction_ecommerce_website_name,transaction_data:transaction_pro
duct_name,transaction_data:transaction_datetime,transaction_data:transaction_amount,customer_data:transaction_city_name,customer_data:transaction_country_name"
. . . . . . . . . . . . . . . . .> )
. . . . . . . . . . . . . . . . .> TBLPROPERTIES ("HBase.table.name"="transaction_detail_HBase_tbl");
Error: Error while compiling statement: FAILED: SemanticException Cannot find class 'org.apache.hadoop.hive.HBase.HBaseStorageHandler' (state=42000,code=40000)
0: jdbc:hive2://nodemaster:10000/> CREATE EXTERNAL TABLE transaction_detail_hive_tbl(transaction_id int
```

Few possible reasons for this were that Hive and Hbase containers were on different networks initially and Hive didn't have the HBase client jars and configuration available. We tried manually copying jars from the Hbbase container to the Hive container, fixed configurations by setting hbase-site.xml in Hive's class path and ensured Hive had correct zookeeper quorum, and made both containers available on the same network. However, Hive was still not able to connect to Hbase and this query kept failing. We were looking for some documentation for Hbase and Hive integration. We found a repository that had scripts for running containers for both.

Repository Link: https://github.com/kennonsr/hadoop_hbase_hive_spark_docker

We started all the containers and checked if everything was working fine. We recreated the table transaction_detail_HBase_tbl **in Hbase and populated it with the data given above.**

Initially, we tried running the external table query, but it gave the same result. We then tried the same, copied Hbase client jar and config files from Hbase to Hive, ran a script for that.



```
  </property>
  <property>
     <name>hive.metastore.warehouse.dir</name>
     <value>/user/hive/metastore</value>
     <description>Metastore Warehouse</description>
  </property>
  <property>
     <name>hive.server2.enable.doAs </name>
     <value>false</value>
  </property>
  <property>
         <name>hive.security.authorization.enabled</name>
         <value>false</value>
  </property>
<property>
  <name>hbase.zookeeper.quorum</name>
  <value>hbase</value>
</property>
<property>
  <name>hbase.zookeeper.property.clientPort</name>
  <value>2181</value>
</property>
<property>
  <name>hive.zookeeper.quorum</name>
  <value>hbase</value>
</property>
<property>
  <name>hive.zookeeper.client.port</name>
  <value>2181</value>
</property>
<property>
  <name>zookeeper.znode.parent</name>
  <value>/hbase</value>
</property></configuration>
                                                              69,12          Bot
```

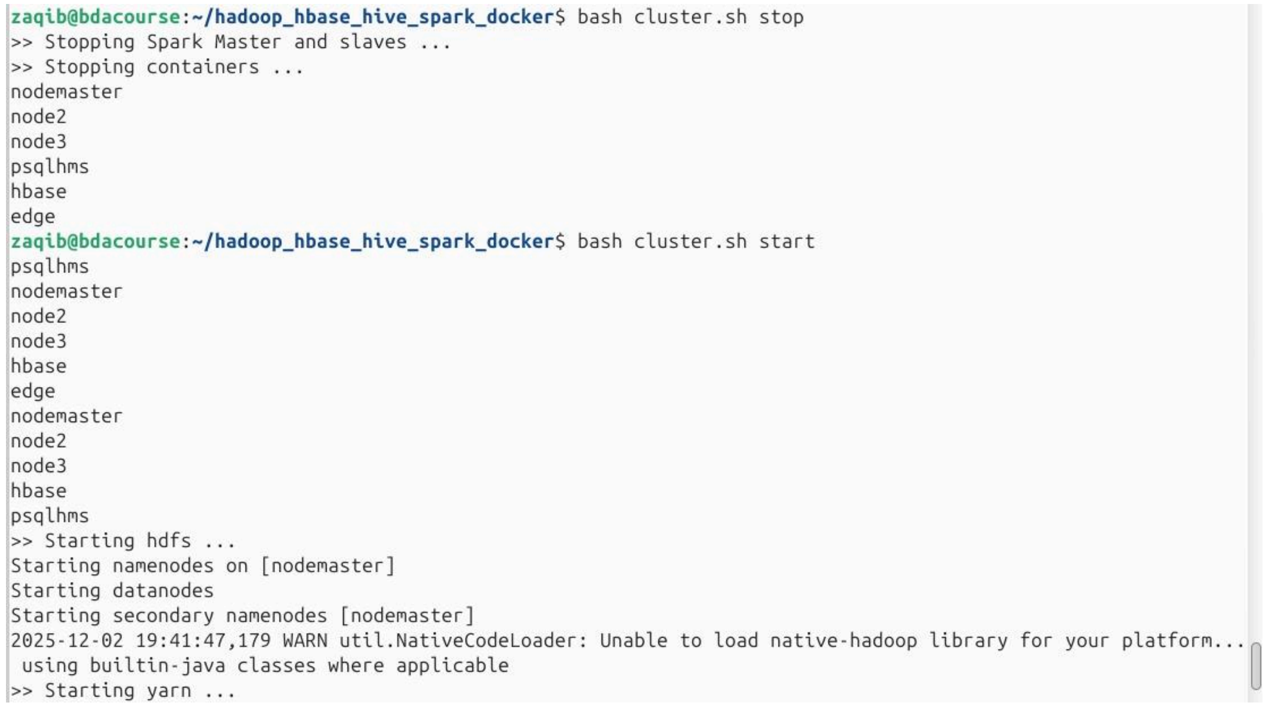
Next we added ZooKeeper configurations in `hive-site.xml` to connect Hive with HBase via ZooKeeper.

Additions to `hive-site.xml`:

```
<property>
  <name>hbase.zookeeper.quorum</name>
  <value>hbase</value>
</property>
<property>
  <name>hbase.zookeeper.property.clientPort</name>
  <value>2181</value>
</property>
<property>
  <name>hive.zookeeper.quorum</name>
  <value>hbase</value>
</property>
<property>
  <name>hive.zookeeper.client.port</name>
  <value>2181</value>
</property>
<property>
  <name>zookeeper.znode.parent</name>
  <value>/hbase</value>
```

```
</property>
```

Then we restarted all the containers to apply the changes.

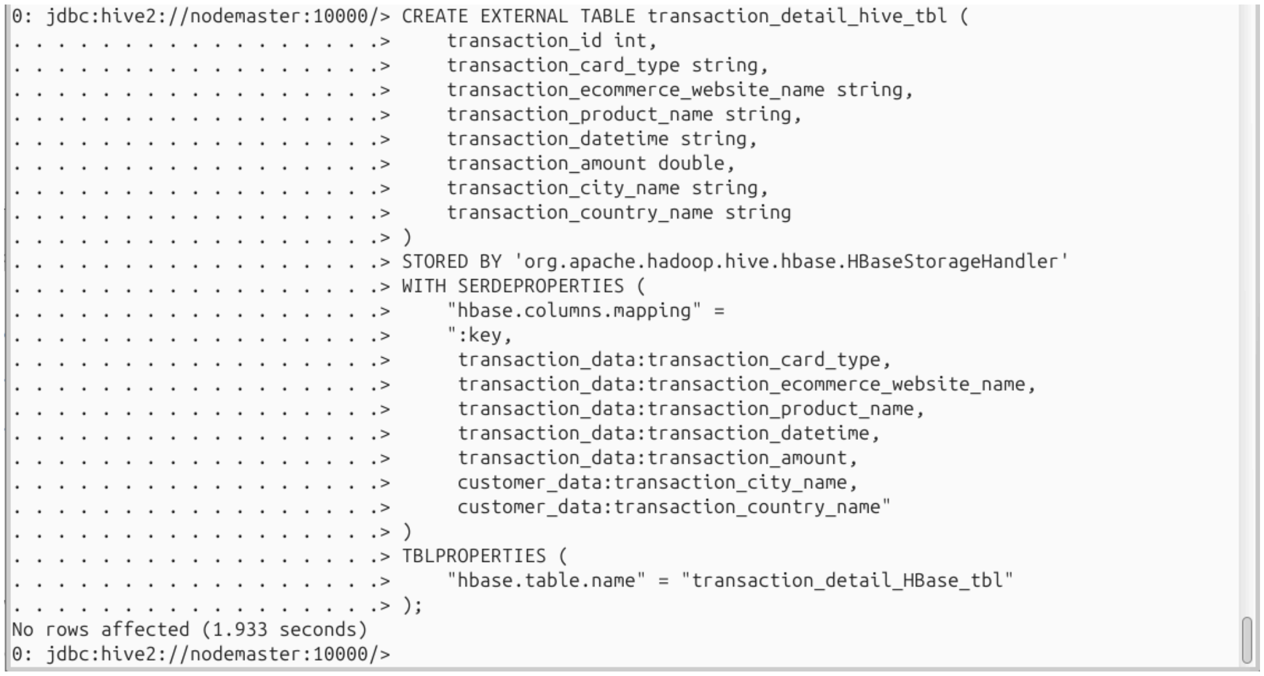


```
zaqib@bdacourse:~/hadoop_hbase_hive_spark_docker$ bash cluster.sh stop
>> Stopping Spark Master and slaves ...
>> Stopping containers ...
nodemaster
node2
node3
psqlhms
hbase
edge
zaqib@bdacourse:~/hadoop_hbase_hive_spark_docker$ bash cluster.sh start
psqlhms
nodemaster
node2
node3
hbase
edge
nodemaster
node2
node3
hbase
psqlhms
>> Starting hdfs ...
Starting namenodes on [nodemaster]
Starting datanodes
Starting secondary namenodes [nodemaster]
2025-12-02 19:41:47,179 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform...
 using builtin-java classes where applicable
>> Starting yarn ...
```

Finally we were able to run the command successfully and create an external table in Hive that mapped to the `transaction_detail_HBase_tbl` in Hbase using Hbase Storage Handler.

```
CREATE EXTERNAL TABLE transaction_detail_hive_tbl (
  transaction_id INT,
  transaction_card_type STRING,
  transaction_ecommerce_website_name STRING,
  transaction_product_name STRING,
  transaction_datetime STRING,
  transaction_amount DOUBLE,
  transaction_city_name STRING,
  transaction_country_name STRING
)
STORED BY 'org.apache.hadoop.hive.hbase.HBaseStorageHandler'
WITH SERDEPROPERTIES (
  "hbase.columns.mapping" = ":key,
  transaction_data:transaction_card_type,
  transaction_data:transaction_ecommerce_website_name,
  transaction_data:transaction_product_name,
  transaction_data:transaction_datetime,
  transaction_data:transaction_amount,
  customer_data:transaction_city_name,
  customer_data:transaction_country_name"
)
TBLPROPERTIES (
  "hbase.table.name" = "transaction_detail_HBase_tbl"
);
```

(`org.apache.hadoop.hive.HBase.HBaseStorageHandler` changed to `org.apache.hadoop.hive.hbase.HBaseStorageHandler`)



```
0: jdbc:hive2://nodemaster:10000/> CREATE EXTERNAL TABLE transaction_detail_hive_tbl (
. . . . . . . . . . . . . . . . .>     transaction_id int,
. . . . . . . . . . . . . . . . .>     transaction_card_type string,
. . . . . . . . . . . . . . . . .>     transaction_ecommerce_website_name string,
. . . . . . . . . . . . . . . . .>     transaction_product_name string,
. . . . . . . . . . . . . . . . .>     transaction_datetime string,
. . . . . . . . . . . . . . . . .>     transaction_amount double,
. . . . . . . . . . . . . . . . .>     transaction_city_name string,
. . . . . . . . . . . . . . . . .>     transaction_country_name string
. . . . . . . . . . . . . . . . .> )
. . . . . . . . . . . . . . . . .> STORED BY 'org.apache.hadoop.hive.hbase.HBaseStorageHandler'
. . . . . . . . . . . . . . . . .> WITH SERDEPROPERTIES (
. . . . . . . . . . . . . . . . .>     "hbase.columns.mapping" =
. . . . . . . . . . . . . . . . .>     ":key,
. . . . . . . . . . . . . . . . .>      transaction_data:transaction_card_type,
. . . . . . . . . . . . . . . . .>      transaction_data:transaction_ecommerce_website_name,
. . . . . . . . . . . . . . . . .>      transaction_data:transaction_product_name,
. . . . . . . . . . . . . . . . .>      transaction_data:transaction_datetime,
. . . . . . . . . . . . . . . . .>      transaction_data:transaction_amount,
. . . . . . . . . . . . . . . . .>      customer_data:transaction_city_name,
. . . . . . . . . . . . . . . . .>      customer_data:transaction_country_name"
. . . . . . . . . . . . . . . . .> )
. . . . . . . . . . . . . . . . .> TBLPROPERTIES (
. . . . . . . . . . . . . . . . .>     "hbase.table.name" = "transaction_detail_HBase_tbl"
. . . . . . . . . . . . . . . . .> );
No rows affected (1.933 seconds)
0: jdbc:hive2://nodemaster:10000/>
```

How this integration works:

A new table is created in Hive database using the create external table command which has all attributes and datatypes declared in Hive in a similar way as SQL tables.

This table is linked with HBase using a stored by operation using the storage handler for HBase as depicted in the architecture diagram above.

Furthermore, this command also adds serdeproperties where we specify all the column families and column family qualifiers that have to be connected to in HBase listing them in the same sequence as listed in the attributes declared after the create external table command.

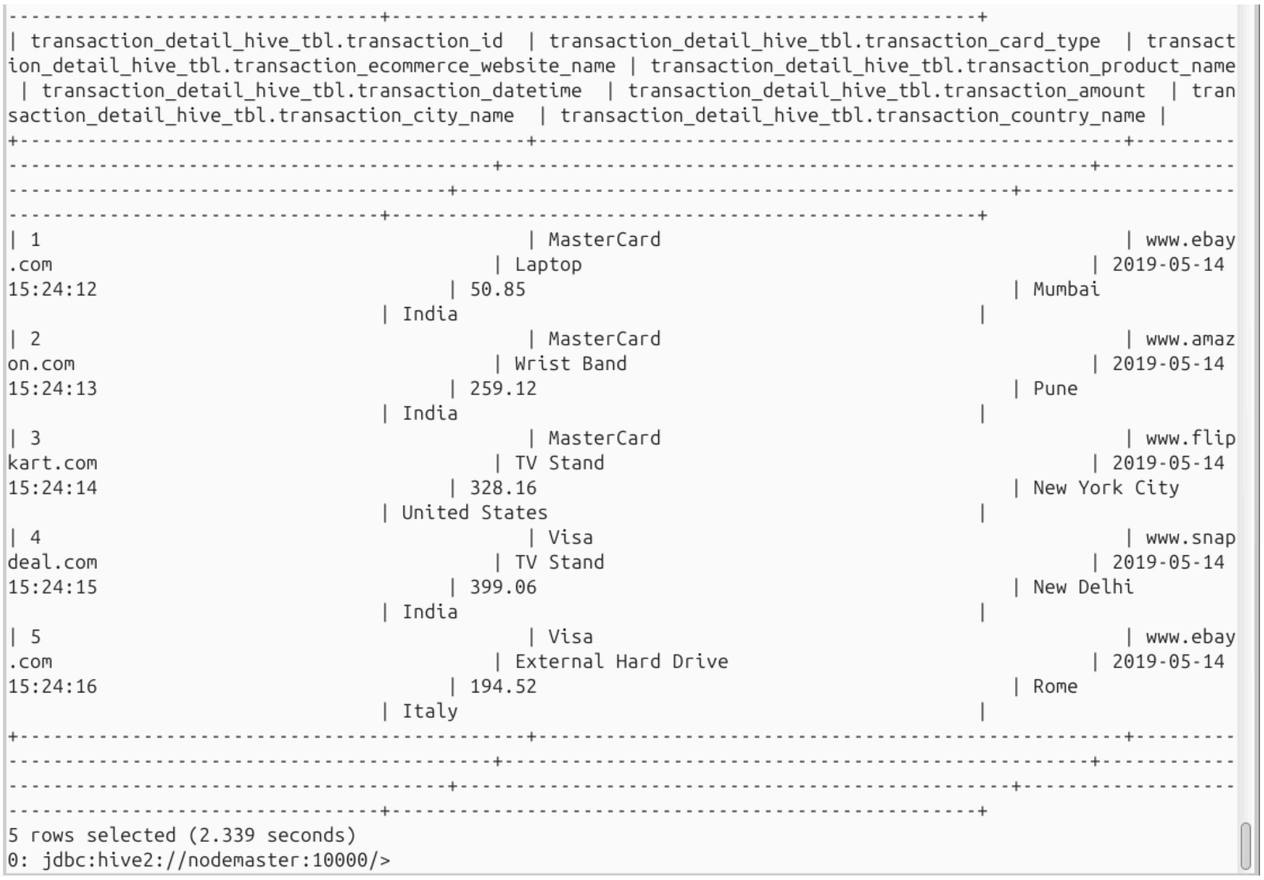
Lastly this command creates a link to the HBase table using tblproperties.

Note: HBase table may have many columns however, in Hive we do not have to fetch all columns, but just the columns we need for analysis.

Once this is executed, the data can be fetched in Hive using similar queries to SQL.

View the HBase connected table in Hive:

[ToDo: Paste snapshot of below] select * from transaction_detail_hive_tbl;



```
--------------------------------+----------------------------------------------------+
| transaction_detail_hive_tbl.transaction_id  | transaction_detail_hive_tbl.transaction_card_type  | transact
ion_detail_hive_tbl.transaction_ecommerce_website_name | transaction_detail_hive_tbl.transaction_product_name
 | transaction_detail_hive_tbl.transaction_datetime  | transaction_detail_hive_tbl.transaction_amount  | tran
saction_detail_hive_tbl.transaction_city_name  | transaction_detail_hive_tbl.transaction_country_name |
+---------------------------------------------+----------------------------------------------------+--------
-----------------------------------------+-----------------------------------------------------+-----------
---------------------------------------+-------------------------------------------------+-----------------
--------------------------------+----------------------------------------------------+
| 1                                           | MasterCard                                         | www.ebay
.com                                         | Laptop                                              | 2019-05-14
15:24:12                               | 50.85                                           | Mumbai
                                | India                                              |
| 2                                           | MasterCard                                         | www.amaz
on.com                                       | Wrist Band                                          | 2019-05-14
15:24:13                               | 259.12                                          | Pune
                                | India                                              |
| 3                                           | MasterCard                                         | www.flip
kart.com                                     | TV Stand                                            | 2019-05-14
15:24:14                               | 328.16                                          | New York City
                                | United States                                      |
| 4                                           | Visa                                               | www.snap
deal.com                                     | TV Stand                                            | 2019-05-14
15:24:15                               | 399.06                                          | New Delhi
                                | India                                              |
| 5                                           | Visa                                               | www.ebay
.com                                         | External Hard Drive                                 | 2019-05-14
15:24:16                               | 194.52                                          | Rome
                                | Italy                                              |
+---------------------------------------------+----------------------------------------------------+--------
-----------------------------------------+-----------------------------------------------------+-----------
---------------------------------------+-------------------------------------------------+-----------------
--------------------------------+----------------------------------------------------+
5 rows selected (2.339 seconds)
0: jdbc:hive2://nodemaster:10000/>
```

Note: all this data is stored in HBase and is directly coming from it. Nothing is ever stored in Hive. Hive is only being used as an interface to query the data for simplification purposes.

Inserting Big Data in HBase programmatically:

[ToDo: Do all below with Java/Python]

As seen above, sample data was inserted using the ‘put’ command in the HBase table to demo the integration with Hive.

However, for big data, multiple put commands will make the data insertion process inefficient. A better approach would be to write a program for data insertion.

Example program code for inserting large data set in HBase (this code is in jRuby, but can be written in java or python as well)

Create a JRuby file named multi.rb through which we will insert multiple rows in an HBase table:

Comment: Setting up packages and function to initiate HBase insertion program import 'org.apache.hadoop.HBase.client.HTable' import 'org.apache.hadoop.HBase.client.Put'

def jbytes(*args) args.map {|arg| arg.to_s.to_java_bytes}

end

Comment: Creating HTable as an object called tables pointing to the transaction_detail_HBase_tbl

Comment: Creating an object called "row" with a row key specified table = HTable.new(@HBase.configuration, "transaction_detail_HBase_tbl") row = Put.new(*jbytes("200"))

Comment: Using the add method to put data into the "row" object row.add (*jbytes("customer_data", "card_type", "master card")) (*jbytes("customer_data", "card_type", "visa card")) (*jbytes("customer_data", "card_type", "multi card")) table.put(row)

Executing from linux command prompt:

hbase shell multi.rb

We created a java script `SingleInsertTest.java` that connects to HBase and inserts one single transaction record into the HBase table, `transaction_detail_HBase_tbl`.

It firstly creates configuration and connects to Hbase:

```
config = HBaseConfiguration.create();
connection = ConnectionFactory.createConnection(config);
```

Next, it gets access to the table:

```
table = connection.getTable(TableName.valueOf("transaction_detail_HBase_tbl"));
```

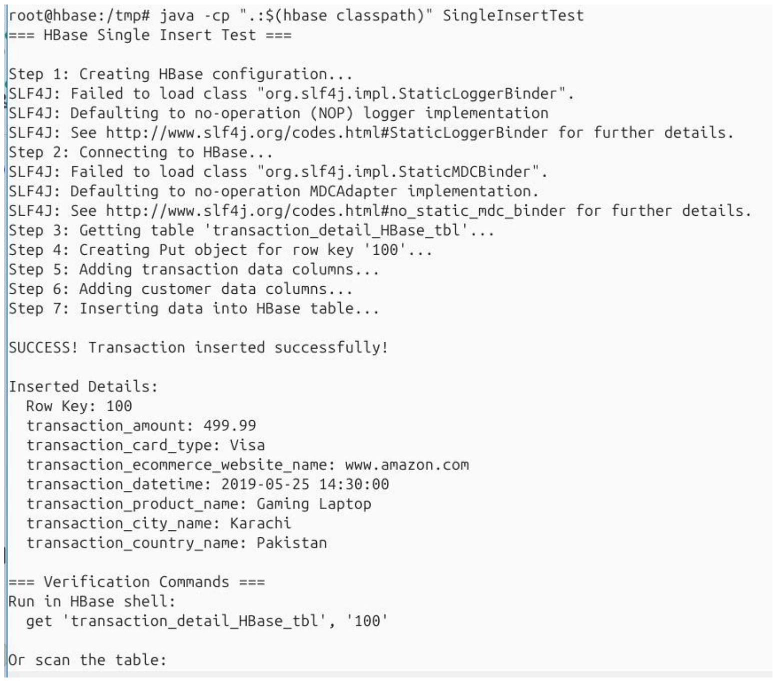
Next, it creates a new row (row key = 100) in the HBase table and inserts transaction and customer data into it.

```java
Put put = new Put(Bytes.toBytes("100"));

// transaction_data column family
put.addColumn(
     Bytes.toBytes("transaction_data"),
     Bytes.toBytes("transaction_amount"),
     Bytes.toBytes("499.99") );
put.addColumn(
     Bytes.toBytes("transaction_data"),
     Bytes.toBytes("transaction_card_type"),
     Bytes.toBytes("Visa") );
put.addColumn(
     Bytes.toBytes("transaction_data"),
     Bytes.toBytes("transaction_ecommerce_website_name"),
     Bytes.toBytes("www.amazon.com") );
put.addColumn(
     Bytes.toBytes("transaction_data"),
     Bytes.toBytes("transaction_datetime"),
     Bytes.toBytes("2019-05-25 14:30:00") );
put.addColumn(
     Bytes.toBytes("transaction_data"),
     Bytes.toBytes("transaction_product_name"),
     Bytes.toBytes("Gaming Laptop") );

// customer_data column family
put.addColumn(
     Bytes.toBytes("customer_data"),
     Bytes.toBytes("transaction_city_name"),
     Bytes.toBytes("Karachi") );
put.addColumn(
     Bytes.toBytes("customer_data"),
     Bytes.toBytes("transaction_country_name"),
     Bytes.toBytes("Pakistan") );

// Insert data
table.put(put);
```

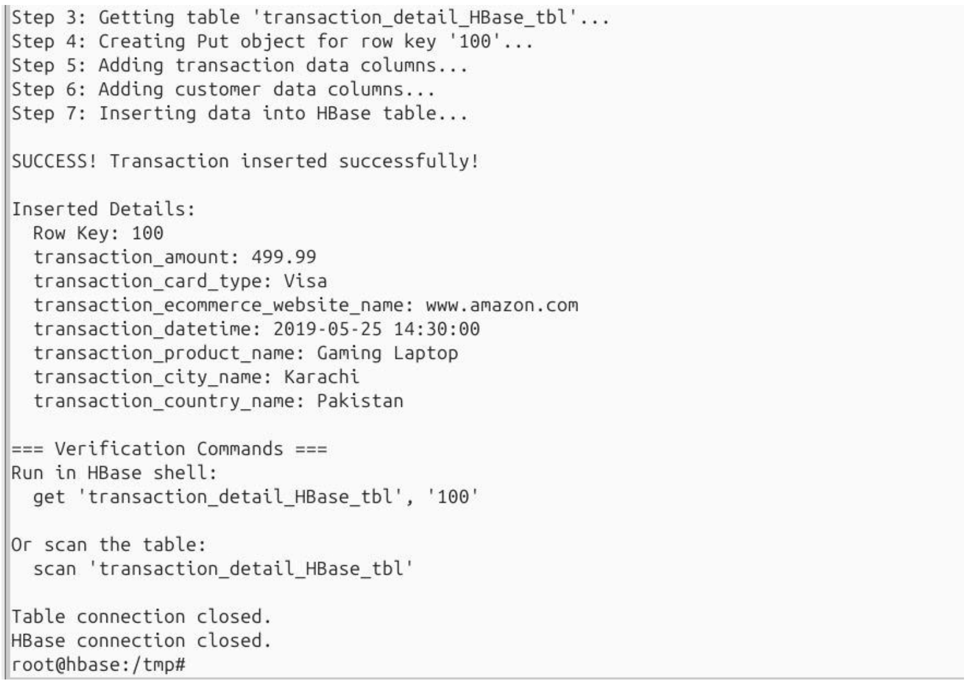
We copied this script into hbase container.

Compiled it: `javac -cp "$(hbase classpath)" SingleInsertTest.java`

Ran it: `java -cp ".:$(hbase classpath)" SingleInsertTest`



```
root@hbase:/tmp# java -cp ".:$(hbase classpath)" SingleInsertTest
=== HBase Single Insert Test ===

Step 1: Creating HBase configuration...
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
Step 2: Connecting to HBase...
SLF4J: Failed to load class "org.slf4j.impl.StaticMDCBinder".
SLF4J: Defaulting to no-operation MDCAdapter implementation.
SLF4J: See http://www.slf4j.org/codes.html#no_static_mdc_binder for further details.
Step 3: Getting table 'transaction_detail_HBase_tbl'...
Step 4: Creating Put object for row key '100'...
Step 5: Adding transaction data columns...
Step 6: Adding customer data columns...
Step 7: Inserting data into HBase table...

SUCCESS! Transaction inserted successfully!

Inserted Details:
  Row Key: 100
  transaction_amount: 499.99
  transaction_card_type: Visa
  transaction_ecommerce_website_name: www.amazon.com
  transaction_datetime: 2019-05-25 14:30:00
  transaction_product_name: Gaming Laptop
  transaction_city_name: Karachi
  transaction_country_name: Pakistan

=== Verification Commands ===
Run in HBase shell:
  get 'transaction_detail_HBase_tbl', '100'

Or scan the table:
```

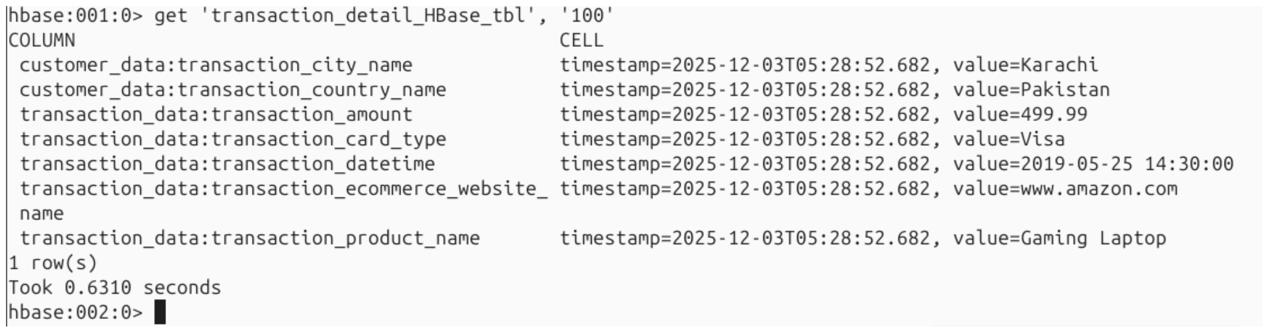


```
Step 3: Getting table 'transaction_detail_HBase_tbl'...
Step 4: Creating Put object for row key '100'...
Step 5: Adding transaction data columns...
Step 6: Adding customer data columns...
Step 7: Inserting data into HBase table...

SUCCESS! Transaction inserted successfully!

Inserted Details:
  Row Key: 100
  transaction_amount: 499.99
  transaction_card_type: Visa
  transaction_ecommerce_website_name: www.amazon.com
  transaction_datetime: 2019-05-25 14:30:00
  transaction_product_name: Gaming Laptop
  transaction_city_name: Karachi
  transaction_country_name: Pakistan

=== Verification Commands ===
Run in HBase shell:
  get 'transaction_detail_HBase_tbl', '100'

Or scan the table:
  scan 'transaction_detail_HBase_tbl'

Table connection closed.
HBase connection closed.
root@hbase:/tmp#
```

Next, we tested if the rows has been added correctly by running the query:

```
get 'transaction_detail_HBase_tbl', '100'
```

from the Hbase shell.



```
hbase:001:0> get 'transaction_detail_HBase_tbl', '100'
COLUMN                                          CELL
 customer_data:transaction_city_name            timestamp=2025-12-03T05:28:52.682, value=Karachi
 customer_data:transaction_country_name         timestamp=2025-12-03T05:28:52.682, value=Pakistan
 transaction_data:transaction_amount            timestamp=2025-12-03T05:28:52.682, value=499.99
 transaction_data:transaction_card_type         timestamp=2025-12-03T05:28:52.682, value=Visa
 transaction_data:transaction_datetime          timestamp=2025-12-03T05:28:52.682, value=2019-05-25 14:30:00
 transaction_data:transaction_ecommerce_website_ timestamp=2025-12-03T05:28:52.682, value=www.amazon.com
 name
 transaction_data:transaction_product_name      timestamp=2025-12-03T05:28:52.682, value=Gaming Laptop
1 row(s)
Took 0.6310 seconds
hbase:002:0>
```

Big data import in HBase - Importing data from a CSV file + Assignment: 2.2% of your grade

[Import your big data into Hbase and query that first through Hbase and then through Hive]

[Show all commands along with important snapshots]

Once the table has been created in HBase, then locate the CSV file on your computer. (transaction_detail.csv)

Make sure to remove the header row as attributes are declared in HBase.

[ToDo: Load the CSV file into the Hbase table and paste snapshot]

hbase org.apache.hadoop.hbase.mapreduce.ImportTsv - Dimporttsv.separator=',' - Dimporttsv.columns=HBASE_ROW_KEY,transaction_data:transaction_card_type,transaction_data:transaction_ecommerce_website_name,transaction_data:transaction_product_name,transaction_data:transaction_datetime,transaction_data:transaction_amount,customer_data:transaction_city_name,customer_data:transaction_country_name transaction_detail_HBase_tbl /user/hdfs/transaction_detail.csv

Firstly created a csv file with 100 rows, in the host machine with random data for the `transaction_detail_HBase_tbl` table. The rows in the csv had row-keys from 200-299. We also ensured the table did not have headers. The data was generated through an LLM.

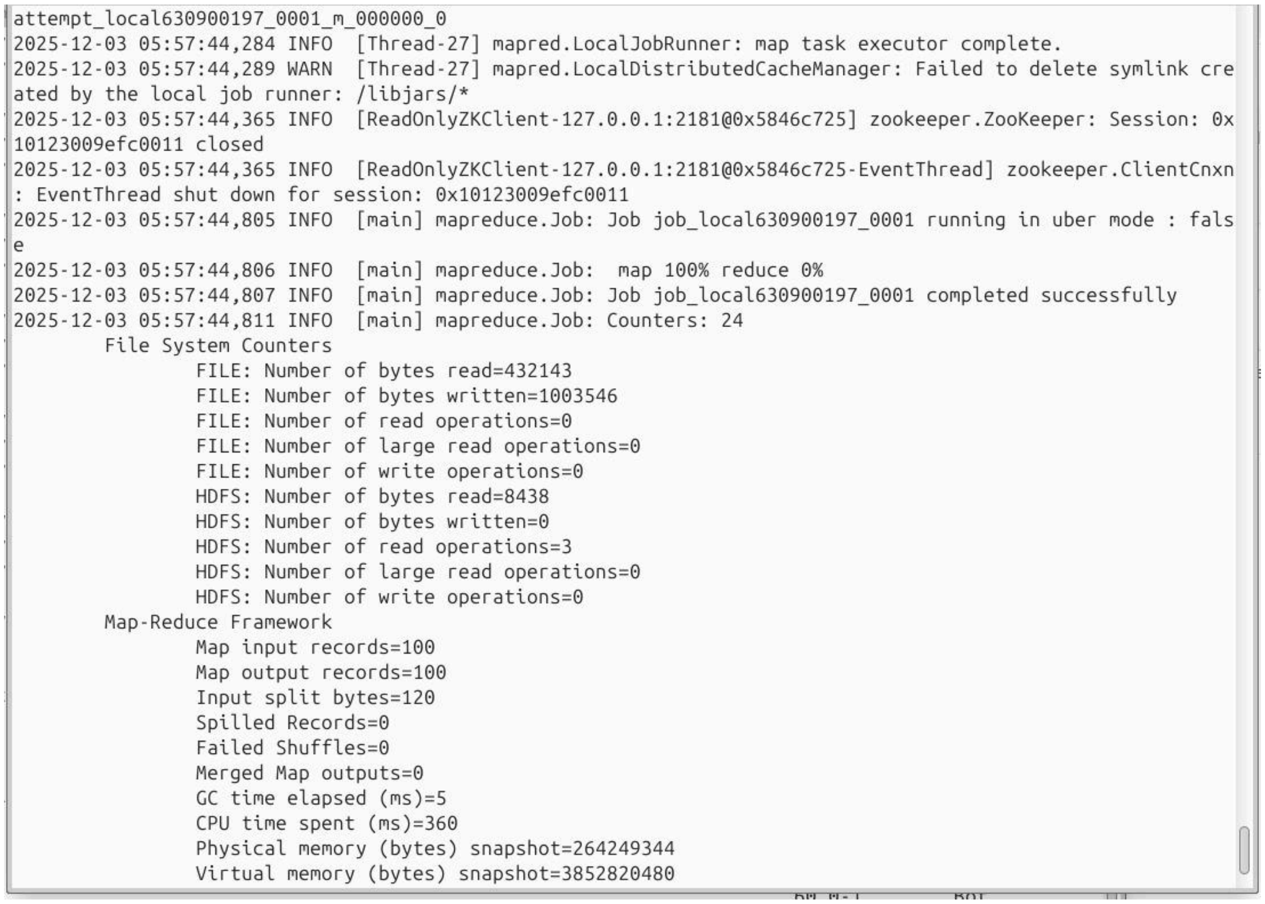
Copied the csv file into the Hbase container and then copied it to HDFS:

```
docker cp transaction_detail.csv hbase:/tmp/
```

```
hdfs dfs -mkdir -p /user/hdfs
hdfs dfs -put -f /tmp/transaction_detail.csv
/user/hdfs/transaction_detail.csv
```

Next we ran the `ImportTsv` command to load csv data into the table.

```
hbase org.apache.hadoop.hbase.mapreduce.ImportTsv \

-Dimporttsv.separator=',' \

-Dimporttsv.columns=HBASE_ROW_KEY,transaction_data:transaction_card_type,trans
action_data:transaction_ecommerce_website_name,transaction_data:transaction_pr
oduct_name,transaction_data:transaction_datetime,transaction_data:transaction_
amount,customer_data:transaction_city_name,customer_data:transaction_country_n
ame \

transaction_detail_HBase_tbl \

hdfs://nodemaster:9000/user/hdfs/transaction_detail.csv
```



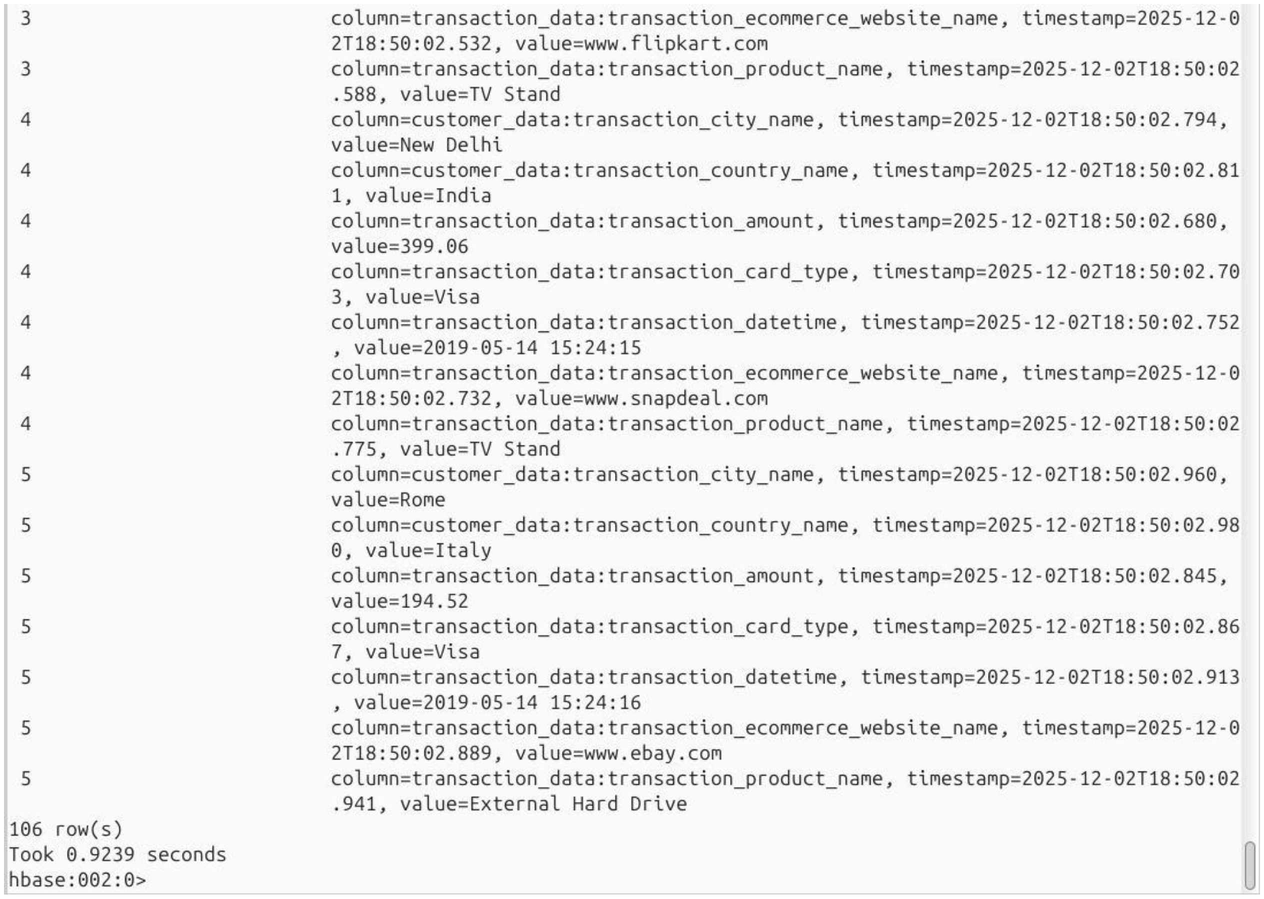
```
attempt_local630900197_0001_m_000000_0
2025-12-03 05:57:44,284 INFO  [Thread-27] mapred.LocalJobRunner: map task executor complete.
2025-12-03 05:57:44,289 WARN  [Thread-27] mapred.LocalDistributedCacheManager: Failed to delete symlink cre
ated by the local job runner: /libjars/*
2025-12-03 05:57:44,365 INFO  [ReadOnlyZKClient-127.0.0.1:2181@0x5846c725] zookeeper.ZooKeeper: Session: 0x
10123009efc0011 closed
2025-12-03 05:57:44,365 INFO  [ReadOnlyZKClient-127.0.0.1:2181@0x5846c725-EventThread] zookeeper.ClientCnxn
: EventThread shut down for session: 0x10123009efc0011
2025-12-03 05:57:44,805 INFO  [main] mapreduce.Job: Job job_local630900197_0001 running in uber mode : fals
e
2025-12-03 05:57:44,806 INFO  [main] mapreduce.Job:  map 100% reduce 0%
2025-12-03 05:57:44,807 INFO  [main] mapreduce.Job: Job job_local630900197_0001 completed successfully
2025-12-03 05:57:44,811 INFO  [main] mapreduce.Job: Counters: 24
        File System Counters
                FILE: Number of bytes read=432143
                FILE: Number of bytes written=1003546
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=8438
                HDFS: Number of bytes written=0
                HDFS: Number of read operations=3
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=0
        Map-Reduce Framework
                Map input records=100
                Map output records=100
                Input split bytes=120
                Spilled Records=0
                Failed Shuffles=0
                Merged Map outputs=0
                GC time elapsed (ms)=5
                CPU time spent (ms)=360
                Physical memory (bytes) snapshot=264249344
                Virtual memory (bytes) snapshot=3852820480
```

Next, we ran some queries from the Hbase shell to check if data has been loaded into the table.

HBase Queries

1. `scan 'transaction_detail_HBase_tbl'`

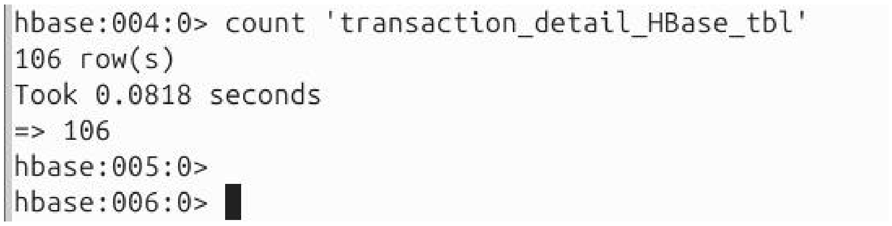
Displays all rows from the HBase table `transaction_detail_HBase_tbl`.



```
 3                           column=transaction_data:transaction_ecommerce_website_name, timestamp=2025-12-0
                             2T18:50:02.532, value=www.flipkart.com
 3                           column=transaction_data:transaction_product_name, timestamp=2025-12-02T18:50:02
                             .588, value=TV Stand
 4                           column=customer_data:transaction_city_name, timestamp=2025-12-02T18:50:02.794,
                             value=New Delhi
 4                           column=customer_data:transaction_country_name, timestamp=2025-12-02T18:50:02.81
                             1, value=India
 4                           column=transaction_data:transaction_amount, timestamp=2025-12-02T18:50:02.680,
                             value=399.06
 4                           column=transaction_data:transaction_card_type, timestamp=2025-12-02T18:50:02.70
                             3, value=Visa
 4                           column=transaction_data:transaction_datetime, timestamp=2025-12-02T18:50:02.752
                             , value=2019-05-14 15:24:15
 4                           column=transaction_data:transaction_ecommerce_website_name, timestamp=2025-12-0
                             2T18:50:02.732, value=www.snapdeal.com
 4                           column=transaction_data:transaction_product_name, timestamp=2025-12-02T18:50:02
                             .775, value=TV Stand
 5                           column=customer_data:transaction_city_name, timestamp=2025-12-02T18:50:02.960,
                             value=Rome
 5                           column=customer_data:transaction_country_name, timestamp=2025-12-02T18:50:02.98
                             0, value=Italy
 5                           column=transaction_data:transaction_amount, timestamp=2025-12-02T18:50:02.845,
                             value=194.52
 5                           column=transaction_data:transaction_card_type, timestamp=2025-12-02T18:50:02.86
                             7, value=Visa
 5                           column=transaction_data:transaction_datetime, timestamp=2025-12-02T18:50:02.913
                             , value=2019-05-14 15:24:16
 5                           column=transaction_data:transaction_ecommerce_website_name, timestamp=2025-12-0
                             2T18:50:02.889, value=www.ebay.com
 5                           column=transaction_data:transaction_product_name, timestamp=2025-12-02T18:50:02
                             .941, value=External Hard Drive
106 row(s)
Took 0.9239 seconds
hbase:002:0>
```

2. `count 'transaction_detail_HBase_tbl'`

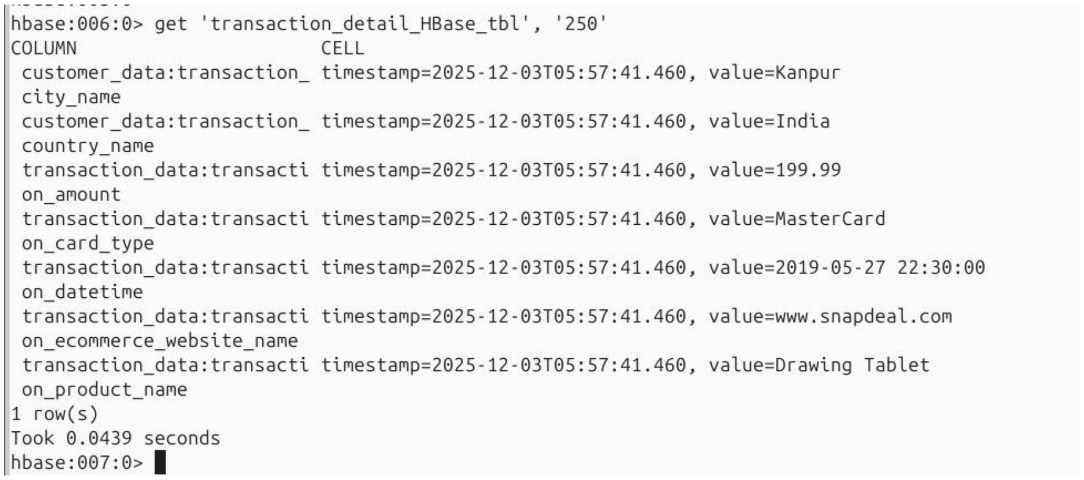
Counts and returns the total number of rows in the table.



```
hbase:004:0> count 'transaction_detail_HBase_tbl'
106 row(s)
Took 0.0818 seconds
=> 106
hbase:005:0>
hbase:006:0>
```

`3. get 'transaction_detail_HBase_tbl', '250'`

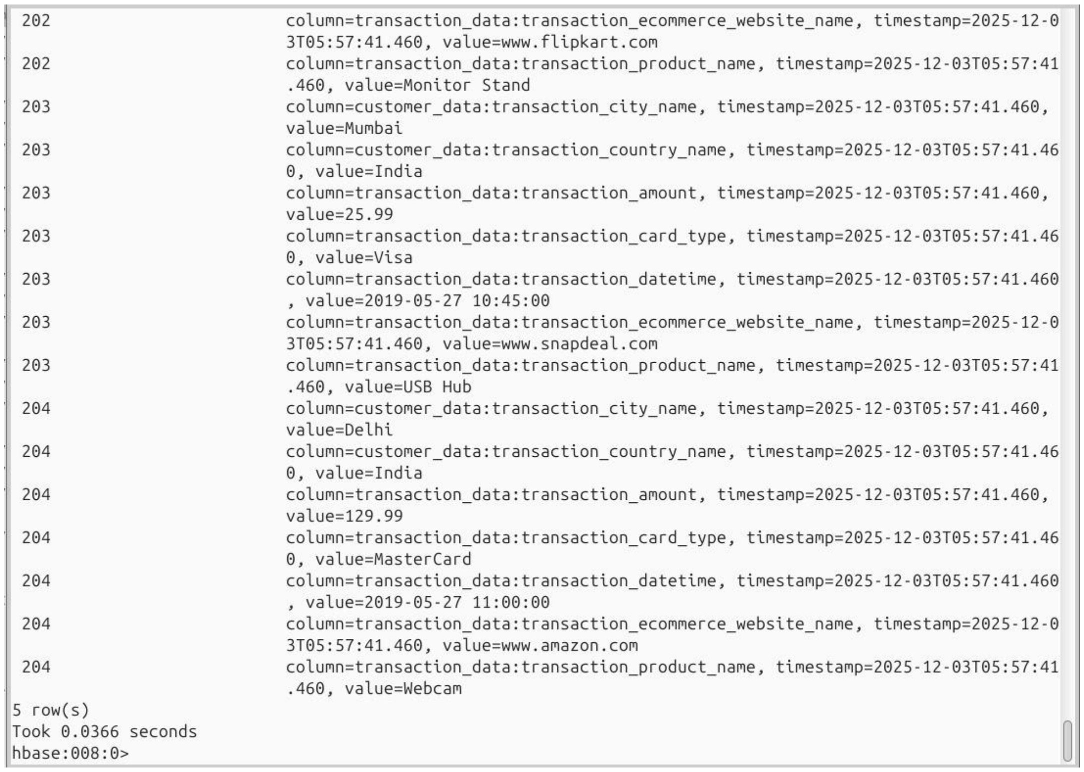
Retrieves all column values for the row with row key `'250'`.



```
hbase:006:0> get 'transaction_detail_HBase_tbl', '250'
COLUMN                      CELL
 customer_data:transaction_ timestamp=2025-12-03T05:57:41.460, value=Kanpur
 city_name
 customer_data:transaction_ timestamp=2025-12-03T05:57:41.460, value=India
 country_name
 transaction_data:transacti timestamp=2025-12-03T05:57:41.460, value=199.99
 on_amount
 transaction_data:transacti timestamp=2025-12-03T05:57:41.460, value=MasterCard
 on_card_type
 transaction_data:transacti timestamp=2025-12-03T05:57:41.460, value=2019-05-27 22:30:00
 on_datetime
 transaction_data:transacti timestamp=2025-12-03T05:57:41.460, value=www.snapdeal.com
 on_ecommerce_website_name
 transaction_data:transacti timestamp=2025-12-03T05:57:41.460, value=Drawing Tablet
 on_product_name
1 row(s)
Took 0.0439 seconds
hbase:007:0>
```

`4. scan 'transaction_detail_HBase_tbl', {STARTROW => '200', STOPROW => '205'}`

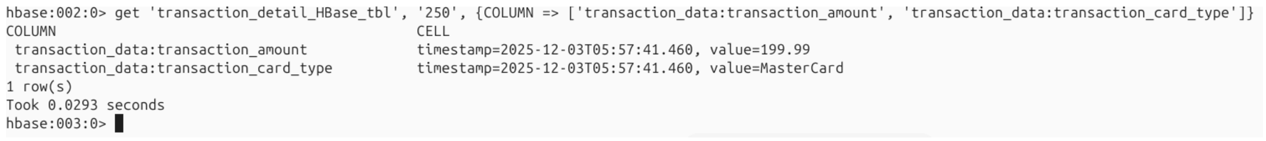
Retrieves all rows with row keys from `'200'` up to (but not including) `'205'`.



```
 202                          column=transaction_data:transaction_ecommerce_website_name, timestamp=2025-12-0
                              3T05:57:41.460, value=www.flipkart.com
 202                          column=transaction_data:transaction_product_name, timestamp=2025-12-03T05:57:41
                              .460, value=Monitor Stand
 203                          column=customer_data:transaction_city_name, timestamp=2025-12-03T05:57:41.460,
                              value=Mumbai
 203                          column=customer_data:transaction_country_name, timestamp=2025-12-03T05:57:41.46
                              0, value=India
 203                          column=transaction_data:transaction_amount, timestamp=2025-12-03T05:57:41.460,
                              value=25.99
 203                          column=transaction_data:transaction_card_type, timestamp=2025-12-03T05:57:41.46
                              0, value=Visa
 203                          column=transaction_data:transaction_datetime, timestamp=2025-12-03T05:57:41.460
                              , value=2019-05-27 10:45:00
 203                          column=transaction_data:transaction_ecommerce_website_name, timestamp=2025-12-0
                              3T05:57:41.460, value=www.snapdeal.com
 203                          column=transaction_data:transaction_product_name, timestamp=2025-12-03T05:57:41
                              .460, value=USB Hub
 204                          column=customer_data:transaction_city_name, timestamp=2025-12-03T05:57:41.460,
                              value=Delhi
 204                          column=customer_data:transaction_country_name, timestamp=2025-12-03T05:57:41.46
                              0, value=India
 204                          column=transaction_data:transaction_amount, timestamp=2025-12-03T05:57:41.460,
                              value=129.99
 204                          column=transaction_data:transaction_card_type, timestamp=2025-12-03T05:57:41.46
                              0, value=MasterCard
 204                          column=transaction_data:transaction_datetime, timestamp=2025-12-03T05:57:41.460
                              , value=2019-05-27 11:00:00
 204                          column=transaction_data:transaction_ecommerce_website_name, timestamp=2025-12-0
                              3T05:57:41.460, value=www.amazon.com
 204                          column=transaction_data:transaction_product_name, timestamp=2025-12-03T05:57:41
                              .460, value=Webcam
5 row(s)
Took 0.0366 seconds
hbase:008:0>
```

```
5. get 'transaction_detail_HBase_tbl', '250', {COLUMN =>
['transaction_data:transaction_amount',
'transaction_data:transaction_card_type']}
```

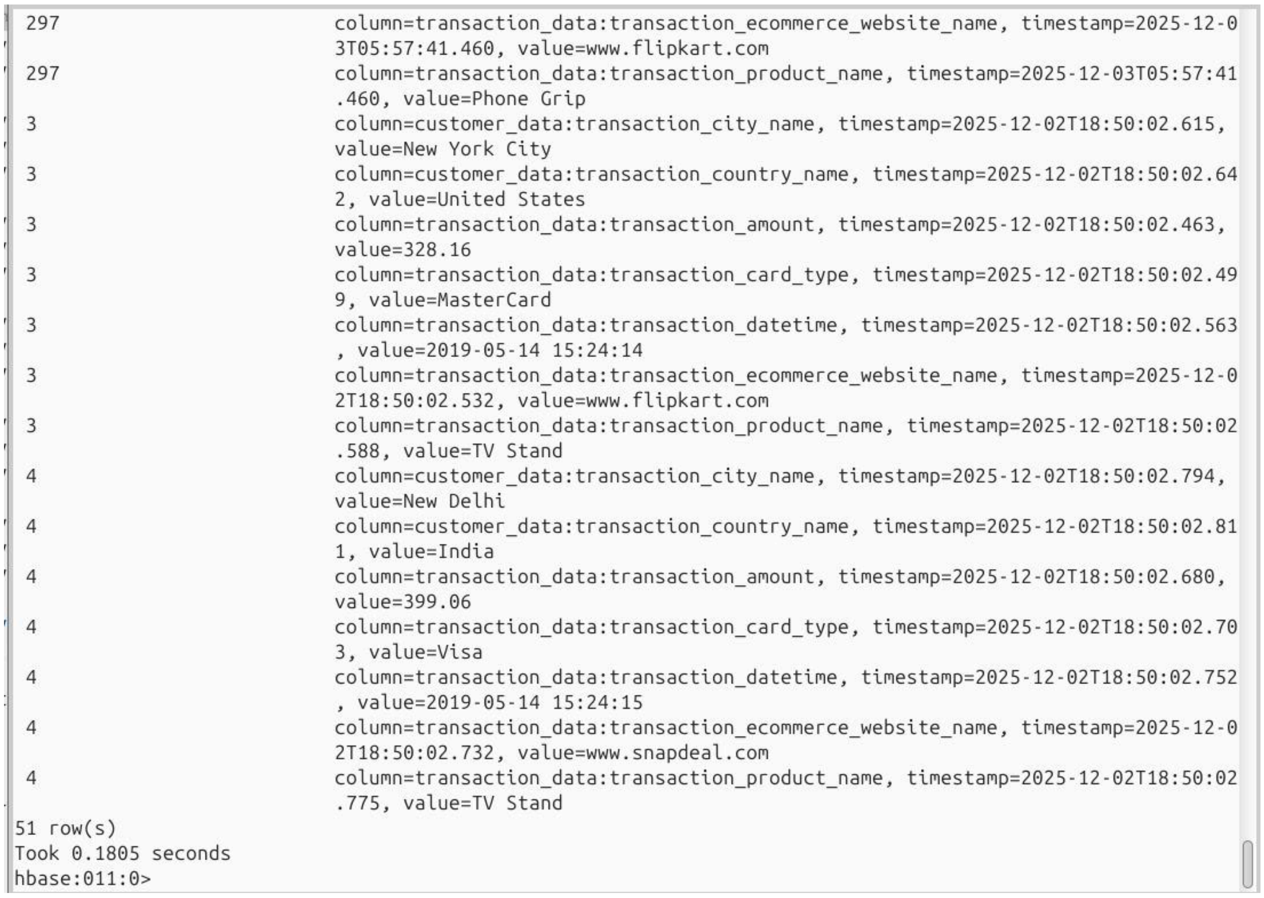
Retrieves only the `transaction_amount` and `transaction_card_type` columns from the `transaction_data` column family for the row with key `'300'`



```
hbase:002:0> get 'transaction_detail_HBase_tbl', '250', {COLUMN => ['transaction_data:transaction_amount', 'transaction_data:transaction_card_type']}
COLUMN                                         CELL
 transaction_data:transaction_amount           timestamp=2025-12-03T05:57:41.460, value=199.99
 transaction_data:transaction_card_type        timestamp=2025-12-03T05:57:41.460, value=MasterCard
1 row(s)
Took 0.0293 seconds
hbase:003:0>
```

```
6. scan 'transaction_detail_HBase_tbl', {
  FILTER => "SingleColumnValueFilter('transaction_data',
'transaction_amount', >, 'binary:300')"
}
```

Scans the table `transaction_detail_HBase_tbl` and returns only the rows where the `transaction_amount` in the `transaction_data` column family is greater than 300.



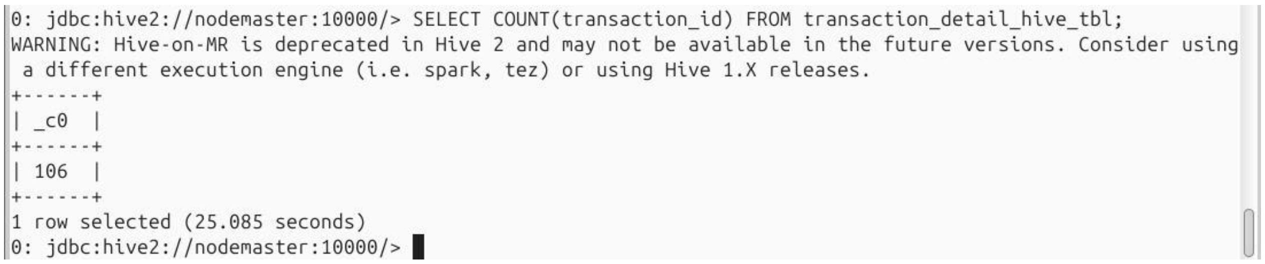
```
 297                          column=transaction_data:transaction_ecommerce_website_name, timestamp=2025-12-0
                              3T05:57:41.460, value=www.flipkart.com
 297                          column=transaction_data:transaction_product_name, timestamp=2025-12-03T05:57:41
                              .460, value=Phone Grip
 3                            column=customer_data:transaction_city_name, timestamp=2025-12-02T18:50:02.615,
                              value=New York City
 3                            column=customer_data:transaction_country_name, timestamp=2025-12-02T18:50:02.64
                              2, value=United States
 3                            column=transaction_data:transaction_amount, timestamp=2025-12-02T18:50:02.463,
                              value=328.16
 3                            column=transaction_data:transaction_card_type, timestamp=2025-12-02T18:50:02.49
                              9, value=MasterCard
 3                            column=transaction_data:transaction_datetime, timestamp=2025-12-02T18:50:02.563
                              , value=2019-05-14 15:24:14
 3                            column=transaction_data:transaction_ecommerce_website_name, timestamp=2025-12-0
                              2T18:50:02.532, value=www.flipkart.com
 3                            column=transaction_data:transaction_product_name, timestamp=2025-12-02T18:50:02
                              .588, value=TV Stand
 4                            column=customer_data:transaction_city_name, timestamp=2025-12-02T18:50:02.794,
                              value=New Delhi
 4                            column=customer_data:transaction_country_name, timestamp=2025-12-02T18:50:02.81
                              1, value=India
 4                            column=transaction_data:transaction_amount, timestamp=2025-12-02T18:50:02.680,
                              value=399.06
 4                            column=transaction_data:transaction_card_type, timestamp=2025-12-02T18:50:02.70
                              3, value=Visa
 4                            column=transaction_data:transaction_datetime, timestamp=2025-12-02T18:50:02.752
                              , value=2019-05-14 15:24:15
 4                            column=transaction_data:transaction_ecommerce_website_name, timestamp=2025-12-0
                              2T18:50:02.732, value=www.snapdeal.com
 4                            column=transaction_data:transaction_product_name, timestamp=2025-12-02T18:50:02
                              .775, value=TV Stand
51 row(s)
Took 0.1805 seconds
hbase:011:0>
```

[Run SQL queries on the data using Hive and show snapshot]

Hive Queries

1. `SELECT COUNT(transaction_id)`
   `FROM transaction_detail_hive_tbl;`

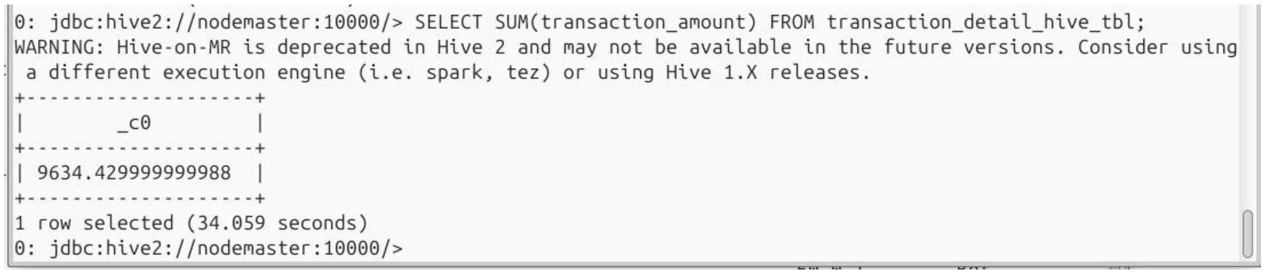
Counts the total number of `transaction_id` entries in the table.



```
0: jdbc:hive2://nodemaster:10000/> SELECT COUNT(transaction_id) FROM transaction_detail_hive_tbl;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using
 a different execution engine (i.e. spark, tez) or using Hive 1.X releases.
+------+
| _c0  |
+------+
| 106  |
+------+
1 row selected (25.085 seconds)
0: jdbc:hive2://nodemaster:10000/>
```

2. `SELECT SUM(transaction_amount)`
   `FROM transaction_detail_hive_tbl;`

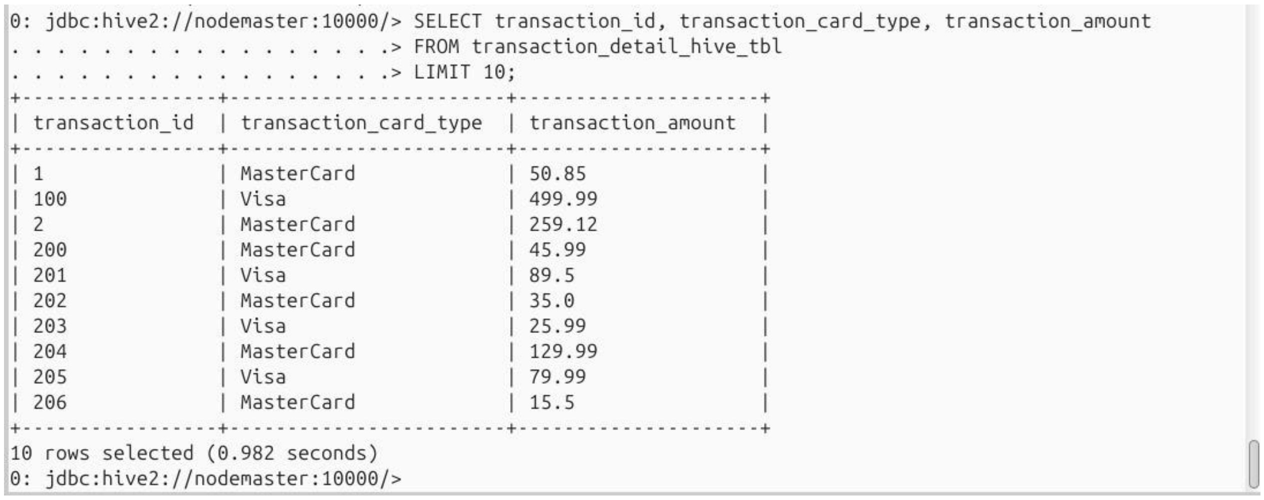
Calculates the total sum of the `transaction_amount` column across all rows in the table.



```
0: jdbc:hive2://nodemaster:10000/> SELECT SUM(transaction_amount) FROM transaction_detail_hive_tbl;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using
 a different execution engine (i.e. spark, tez) or using Hive 1.X releases.
+--------------------+
|        _c0         |
+--------------------+
| 9634.429999999988  |
+--------------------+
1 row selected (34.059 seconds)
0: jdbc:hive2://nodemaster:10000/>
```

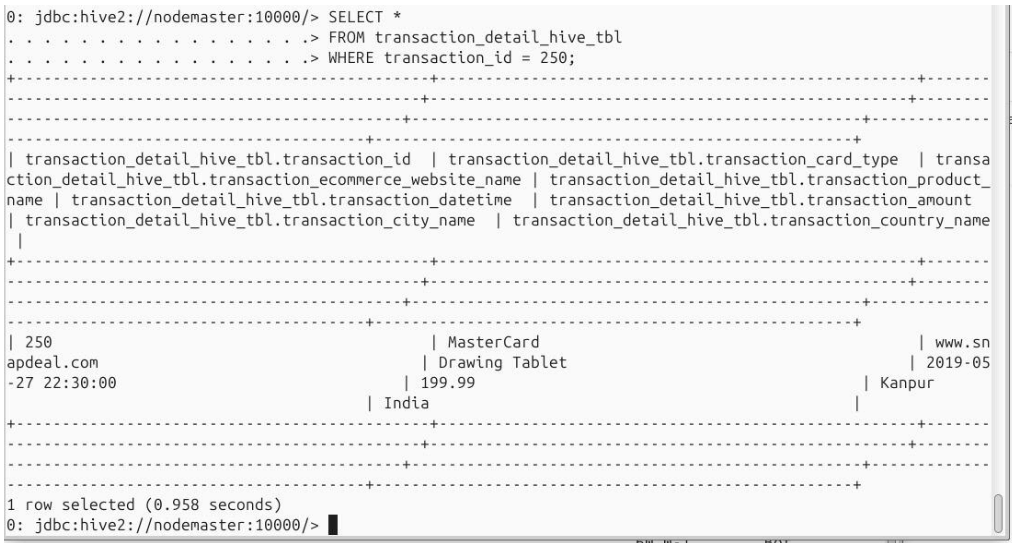
3. `SELECT transaction_id, transaction_card_type, transaction_amount`
   `FROM transaction_detail_hive_tbl`
   `LIMIT 10;`

Retrieves the first 10 rows of `transaction_id`, `transaction_card_type`, and `transaction_amount`.



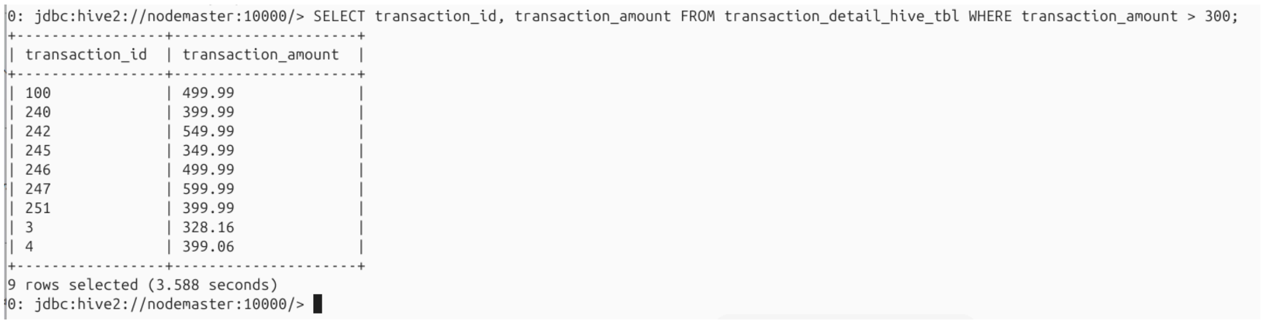
```
0: jdbc:hive2://nodemaster:10000/> SELECT transaction_id, transaction_card_type, transaction_amount
. . . . . . . . . . . . . . . . .> FROM transaction_detail_hive_tbl
. . . . . . . . . . . . . . . . .> LIMIT 10;
+-----------------+------------------------+---------------------+
| transaction_id  | transaction_card_type  | transaction_amount  |
+-----------------+------------------------+---------------------+
| 1               | MasterCard             | 50.85               |
| 100             | Visa                   | 499.99              |
| 2               | MasterCard             | 259.12              |
| 200             | MasterCard             | 45.99               |
| 201             | Visa                   | 89.5                |
| 202             | MasterCard             | 35.0                |
| 203             | Visa                   | 25.99               |
| 204             | MasterCard             | 129.99              |
| 205             | Visa                   | 79.99               |
| 206             | MasterCard             | 15.5                |
+-----------------+------------------------+---------------------+
10 rows selected (0.982 seconds)
0: jdbc:hive2://nodemaster:10000/>
```

```
4. SELECT *
   FROM transaction_detail_hive_tbl
   WHERE transaction_id = 250;
```
Retrieves all columns for the row where `transaction_id` is 250.



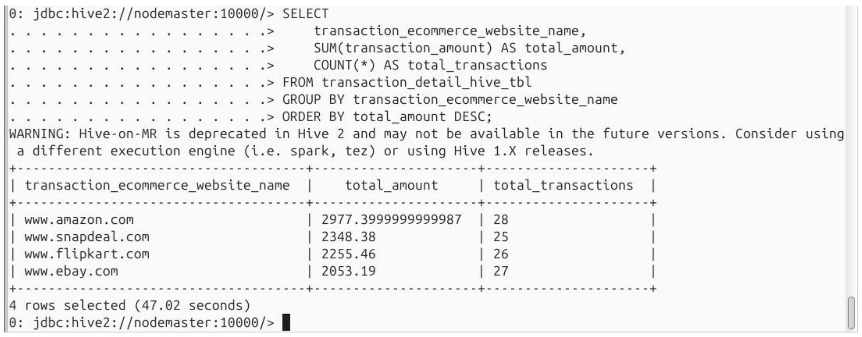
```
0: jdbc:hive2://nodemaster:10000/> SELECT *
. . . . . . . . . . . . . . . . .> FROM transaction_detail_hive_tbl
. . . . . . . . . . . . . . . . .> WHERE transaction_id = 250;
+---------------------------------------------+----------------------------------------------------+-------
---------------------------------------------+----------------------------------------------------+--------
-------------------------------------------+-------------------------------------------------+-------------
---------------------------------------+----------------------------------------------------+
| transaction_detail_hive_tbl.transaction_id  | transaction_detail_hive_tbl.transaction_card_type  | transa
ction_detail_hive_tbl.transaction_ecommerce_website_name | transaction_detail_hive_tbl.transaction_product_
name | transaction_detail_hive_tbl.transaction_datetime  | transaction_detail_hive_tbl.transaction_amount
| transaction_detail_hive_tbl.transaction_city_name  | transaction_detail_hive_tbl.transaction_country_name
 |
+---------------------------------------------+----------------------------------------------------+-------
---------------------------------------------+----------------------------------------------------+--------
-------------------------------------------+-------------------------------------------------+-------------
---------------------------------------+----------------------------------------------------+
| 250                                         | MasterCard                                         | www.sn
apdeal.com                                   | Drawing Tablet                                     | 2019-05
-27 22:30:00                               | 199.99                                          | Kanpur
                                       | India                                              |
+---------------------------------------------+----------------------------------------------------+-------
---------------------------------------------+----------------------------------------------------+--------
-------------------------------------------+-------------------------------------------------+-------------
---------------------------------------+----------------------------------------------------+
1 row selected (0.958 seconds)
0: jdbc:hive2://nodemaster:10000/>
```

```
5. SELECT *
   FROM transaction_detail_hive_tbl
   WHERE transaction_amount > 300;
```
Retrieves all rows where the `transaction_amount` is greater than 300.



```
0: jdbc:hive2://nodemaster:10000/> SELECT transaction_id, transaction_amount FROM transaction_detail_hive_tbl WHERE transaction_amount > 300;
+-----------------+---------------------+
| transaction_id  | transaction_amount  |
+-----------------+---------------------+
| 100             | 499.99              |
| 240             | 399.99              |
| 242             | 549.99              |
| 245             | 349.99              |
| 246             | 499.99              |
| 247             | 599.99              |
| 251             | 399.99              |
| 3               | 328.16              |
| 4               | 399.06              |
+-----------------+---------------------+
9 rows selected (3.588 seconds)
0: jdbc:hive2://nodemaster:10000/>
```

```
6. SELECT
       transaction_ecommerce_website_name,
       SUM(transaction_amount) AS total_amount,
       COUNT(*) AS total_transactions
   FROM transaction_detail_hive_tbl
   GROUP BY transaction_ecommerce_website_name
   ORDER BY total_amount DESC;
```
Groups transactions by e-commerce website, calculates the total transaction amount and count per website, and orders the results by total amount in descending order.



```
0: jdbc:hive2://nodemaster:10000/> SELECT
. . . . . . . . . . . . . . . . .>     transaction_ecommerce_website_name,
. . . . . . . . . . . . . . . . .>     SUM(transaction_amount) AS total_amount,
. . . . . . . . . . . . . . . . .>     COUNT(*) AS total_transactions
. . . . . . . . . . . . . . . . .> FROM transaction_detail_hive_tbl
. . . . . . . . . . . . . . . . .> GROUP BY transaction_ecommerce_website_name
. . . . . . . . . . . . . . . . .> ORDER BY total_amount DESC;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using
 a different execution engine (i.e. spark, tez) or using Hive 1.X releases.
+-------------------------------------+---------------------+---------------------+
| transaction_ecommerce_website_name  |    total_amount     | total_transactions  |
+-------------------------------------+---------------------+---------------------+
| www.amazon.com                      | 2977.3999999999987  | 28                  |
| www.snapdeal.com                    | 2348.38             | 25                  |
| www.flipkart.com                    | 2255.46             | 26                  |
| www.ebay.com                        | 2053.19             | 27                  |
+-------------------------------------+---------------------+---------------------+
4 rows selected (47.02 seconds)
0: jdbc:hive2://nodemaster:10000/>
```

Client API: The Basics

Topic: Basics of client API, CRUD operations, KeyValue Class, application for inserting data into database, application for retrieving data from database, and row locks.

To perform CRUD operations which stands for create, read, update, and delete, on HBase table we use Java client API for HBase. Since HBase has a Java Native API and it is written in Java thus it offers programmatic access to DML (Data Manipulation Language).

Class HBase Configuration: This class adds HBase configuration files to a Configuration. It belongs to the org.apache.hadoop.hbase package.

Method: To create a Configuration with HBase resources, we use the following method:

static org.apache.hadoop.conf.Configuration create()

Class HTable in HBase Client API:

An HBase internal class which represents an HBase table is HTable. Basically, to communicate with a single HBase table, we use this implementation of a table. It belongs to the org.apache.hadoop.hbase.client class.

Constructors: We can create an object to access an HBase table by using the following constructors:

HTable()

HTable(TableName tableName, ClusterConnection connection, ExecutorService pool) **Methods**:

void close() : To release all the resources of the HTable, we use this method.

void delete(Delete delete) : The method “void delete(Delete delete)” helps to delete the specified cells/row.

Result get(Get get) : This method retrieves certain cells from a given row.

org.apache.hadoop.conf.Conf iguration getConfiguration() : It returns the Configuration object used by this instance.

TableName getName() : This method returns the table name instance of the table.

HTableDescriptor getTableDescriptor() : It returns the table descriptor for the table. byte[]

getTableName() : This method returns the name of the table. void put(Put put): We can insert

data into the table, by using this method.

Class Put in HBase Client API:

To perform put operations for a single row, we use the class that belongs to the org.apache.hadoop.hbase.client package.

Constructors:

put(byte[] row): We can create a Put operation for the specified row, by using this constructor.

put(byte[] rowArray, int rowOffset, int rowLength): To make a copy of the passed-in row key to keep local, we use it.

put(byte[] rowArray, int rowOffset, int rowLength, long ts): We can make a copy of the passed-in row key to keep local, by using this constructor.

put(byte[] row, long ts): To create a Put operation for the specified row, using a given timestamp, we use it.

Methods

put add(byte[] family, byte[] qualifier, byte[] value): The method "Put add(byte[] family, byte[] qualifier, byte[] value)" adds the specified column and value to this Put operation.

put add(byte[] family, byte[] qualifier, long ts, byte[] value): With the specified timestamp, it adds the specified column and value, as its version to this Put operation.

put add(byte[] family, ByteBuffer qualifier, long ts, ByteBuffer value): This method adds the specified column and value, with the specified timestamp as its version to this Put operation.

put add(byte[] family, ByteBuffer qualifier, long ts, ByteBuffer value): With the specified timestamp, it adds the specified column and value, as its version to this Put operation.

The KeyValue class:

In our code we may have to deal with KeyValue instances directly. These instances contain the data as well as the coordinates of one specific cell. The coordinates are the row key, name of the column family, column qualifier, and timestamp.

The class provides a plethora of constructors that allow you to combine all of these in many variations. The fully specified constructor looks like this: KeyValue(byte[] row, int roffset, int rlength, byte[] family, int foffset, int flength, byte[] qualifier, int qoffset, int qlength, long timestamp, Type type, byte[] value, int voffset, int vlength)

Class Get in HBase Client API:

To perform Get operations on a single row, we use the class that belongs to the org.apache.hadoop.hbase.client package.

Constructor:

get(byte[] row): It is possible to create a Get operation for the specified row, by using this constructor.

Methods:

get addColumn(byte[] family, byte[] qualifier): To retrieve the column from the specific family with the specified qualifier, this method helps.

get addFamily(byte[] family): This one helps to retrieve all columns from the specified family.

Class Delete in HBase Client API Constructors delete(byte[] row): To create a delete operation

for the specified row, we use it.

delete(byte[] rowArray, int rowOffset, int rowLength): This constructor creates a Delete operation for the specified row and timestamp.

delete(byte[] rowArray, int rowOffset, int rowLength, long ts): This constructor performs Delete operation.

delete(byte[] row, long timestamp): Again this constructor performs Delete operation.

Methods delete addColumn(byte[] family, byte[] qualifier): This method helps to delete the latest version of the specified column.

delete addColumns(byte[] family, byte[] qualifier, long timestamp): To delete all versions of the specified column we use this method, especially, with a timestamp less than or equal to the specified timestamp.

delete addFamily(byte[] family): This method deletes all versions of all columns of the specified family.

delete addFamily(byte[] family, long timestamp): Again, with a timestamp less than or equal to the specified timestamp, this method also deletes all columns of the specified family.

Class Result in HBase Client API Constructor result(): With no KeyValue payload, it is possible to create an empty Result; returns null if you call raw Cells(), by using this constructor. **Methods**

byte[] getValue(byte[] family, byte[] qualifier): In order to get the latest version of the specified column, we use this method.

byte[] getRow(): Moreover, to retrieve the row key which corresponds to the row from which this Result was created, we use this method.

Row Locks

The region servers provide a row lock feature ensuring that only a client holding the matching lock can modify a row. In practice, though, most client applications do not provide an explicit lock, but rather rely on the mechanism in place that guards each operation separately.

When you send, for example, a put() call to the server with an instance of Put, created with the following constructor: Put(byte[] row) , Which is not providing a RowLock instance parameter, the servers will create a lock on our behalf, just for the duration of the call.

Example Application:

1. Inserting data into HBase Table Using Java API

import org.apache.hadoop.conf.Configuration; import

org.apache.hadoop.hbase.HBaseConfiguration; import

org.apache.hadoop.hbase.client.HTable; import org.apache.hadoop.hbase.client.Put; import

org.apache.hadoop.hbase.util.Bytes; import java.io.IOException; public class PutExample {

public static void main(String[] args) throws IOException {

Create the required configuration.

Configuration conf = HBaseConfiguration.create(); Instantiate a new client.

HTable table = new HTable(conf, "testtable"); Create Put with specific row.

Put put = new Put(Bytes.toBytes("row1"));

Add a column, whose name is “colfam1:qual1”, to the Put. put.add(Bytes.toBytes("colfam1"), Bytes.toBytes("qual1"), Bytes.toBytes("val1"));

Add another column, whose name is “colfam1:qual2”, to the Put.

put.add(Bytes.toBytes("colfam1"), Bytes.toBytes("qual2"), Bytes.toBytes("val2")); Store the row

with the column into the HBase table. table.put(put);

}

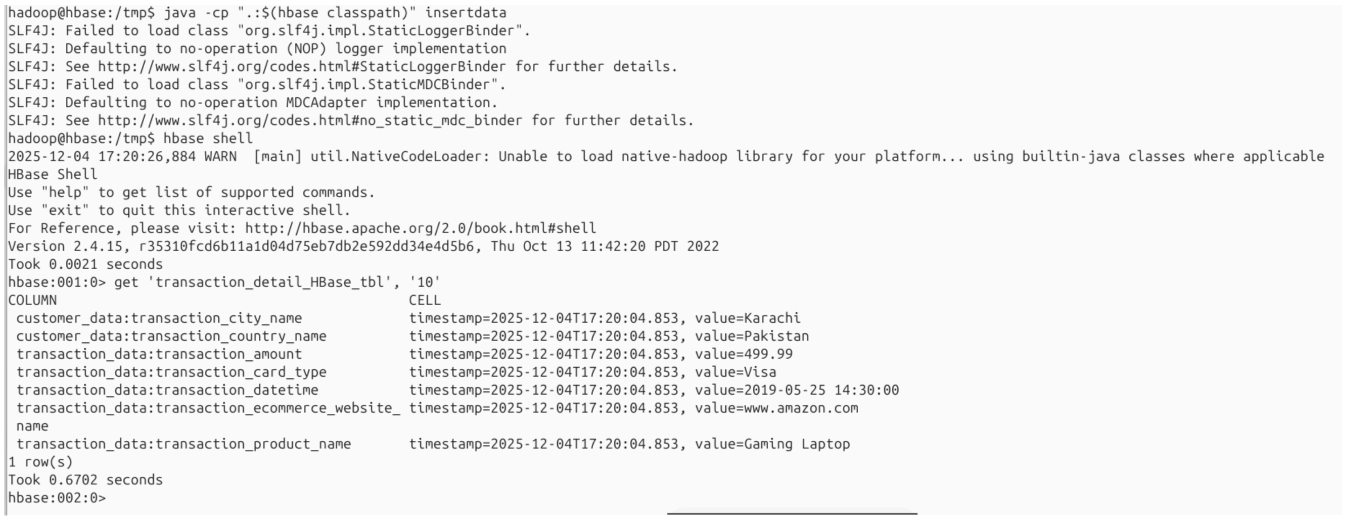
}

}

insertdata.java:

```
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.hbase.HBaseConfiguration;
import org.apache.hadoop.hbase.TableName;
import org.apache.hadoop.hbase.client.*;
import org.apache.hadoop.hbase.util.Bytes;
import java.io.IOException;

public class insertdata {
    public static void main(String[] args) {
        Configuration config = null;
        Connection connection = null;
        Table table = null;
        try {
            // Step 1: Create configuration
            config = HBaseConfiguration.create();

            // Step 2: Establish connection
            connection = ConnectionFactory.createConnection(config);

            // Step 3: Get table reference
            table =
connection.getTable(TableName.valueOf("transaction_detail_HBase_tbl"));

            // Step 4: Create Put object with row key '10'
            Put put = new Put(Bytes.toBytes("10"));

            // transaction_data column family
            put.addColumn(
                Bytes.toBytes("transaction_data"),
                Bytes.toBytes("transaction_amount"),
                Bytes.toBytes("499.99")
            );

            put.addColumn(
                Bytes.toBytes("transaction_data"),
                Bytes.toBytes("transaction_card_type"),
                Bytes.toBytes("Visa")
            );

            put.addColumn(
                Bytes.toBytes("transaction_data"),
                Bytes.toBytes("transaction_ecommerce_website_name"),
                Bytes.toBytes("www.amazon.com")
            );

            put.addColumn(
                Bytes.toBytes("transaction_data"),
                Bytes.toBytes("transaction_datetime"),
                Bytes.toBytes("2019-05-25 14:30:00")
            );

            put.addColumn(
                Bytes.toBytes("transaction_data"),
                Bytes.toBytes("transaction_product_name"),
                Bytes.toBytes("Gaming Laptop")
            );

            // customer_data column family
            put.addColumn(
```

```java
                Bytes.toBytes("customer_data"),
                Bytes.toBytes("transaction_city_name"),
                Bytes.toBytes("Karachi")
            );

            put.addColumn(
                Bytes.toBytes("customer_data"),
                Bytes.toBytes("transaction_country_name"),
                Bytes.toBytes("Pakistan")
            );

            // Step 7: Insert data
            table.put(put);

        } catch (IOException e) {
            e.printStackTrace();
        } finally {
            // Step 8: Close resources
            try {
                if (table != null) {
                    table.close();
                }
                if (connection != null) {
                    connection.close();
                }
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }
}
```



```
hadoop@hbase:/tmp$ java -cp ".:$(hbase classpath)" insertdata
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
SLF4J: Failed to load class "org.slf4j.impl.StaticMDCBinder".
SLF4J: Defaulting to no-operation MDCAdapter implementation.
SLF4J: See http://www.slf4j.org/codes.html#no_static_mdc_binder for further details.
hadoop@hbase:/tmp$ hbase shell
2025-12-04 17:20:26,884 WARN  [main] util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
HBase Shell
Use "help" to get list of supported commands.
Use "exit" to quit this interactive shell.
For Reference, please visit: http://hbase.apache.org/2.0/book.html#shell
Version 2.4.15, r35310fcd6b11a1d04d75eb7db2e592dd34e4d5b6, Thu Oct 13 11:42:20 PDT 2022
Took 0.0021 seconds
hbase:001:0> get 'transaction_detail_HBase_tbl', '10'
COLUMN                                       CELL
 customer_data:transaction_city_name         timestamp=2025-12-04T17:20:04.853, value=Karachi
 customer_data:transaction_country_name      timestamp=2025-12-04T17:20:04.853, value=Pakistan
 transaction_data:transaction_amount         timestamp=2025-12-04T17:20:04.853, value=499.99
 transaction_data:transaction_card_type      timestamp=2025-12-04T17:20:04.853, value=Visa
 transaction_data:transaction_datetime       timestamp=2025-12-04T17:20:04.853, value=2019-05-25 14:30:00
 transaction_data:transaction_ecommerce_website_ timestamp=2025-12-04T17:20:04.853, value=www.amazon.com
 name
 transaction_data:transaction_product_name   timestamp=2025-12-04T17:20:04.853, value=Gaming Laptop
1 row(s)
Took 0.6702 seconds
hbase:002:0>
```

2.Retrieving data from HBase Table Using Java API Create the configuration:

Configuration conf = HBaseConfiguration.create(); Instantiate a new table reference:

HTable table = new HTable(conf, "testtable"); Create a Get with a specific row: Get get = new Get(Bytes.toBytes("row1")); Add a column to the Get:

get.addColumn(Bytes.toBytes("colfam1"), Bytes.toBytes("qual1")); Retrieve a row with selected columns from HBase:

Result result = table.get(get);

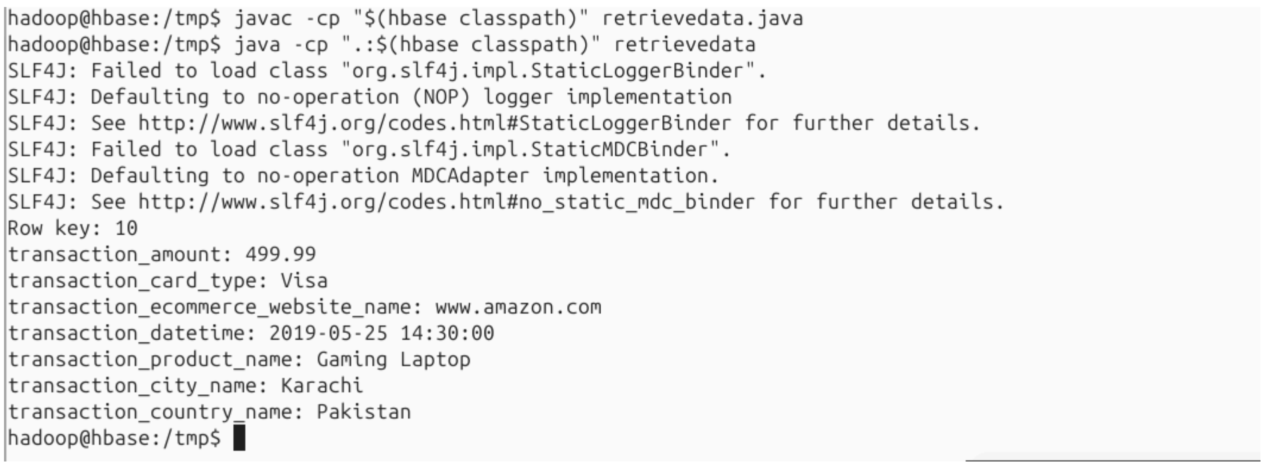
Get a specific value for the given column:

byte[] val = result.getValue(Bytes.toBytes("colfam1"), Bytes.toBytes("qual1")); Print out the value while converting it back:

System.out.println("Value: " + Bytes.toString(val));

retrievedata.java:

```
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.hbase.HBaseConfiguration;
import org.apache.hadoop.hbase.TableName;
import org.apache.hadoop.hbase.client.*;
import org.apache.hadoop.hbase.util.Bytes;
import java.io.IOException;

public class retrievedata {
    public static void main(String[] args) {
        Configuration config = null;
        Connection connection = null;
        Table table = null;

        try {
            // Create HBase configuration
            config = HBaseConfiguration.create();

            // Establish connection
            connection = ConnectionFactory.createConnection(config);

            // Get table reference
            table =
connection.getTable(TableName.valueOf("transaction_detail_HBase_tbl"));

            // Create Get object for row key '10'
            Get get = new Get(Bytes.toBytes("10"));
            // Retrieve the result
            Result result = table.get(get);

            // Print each column value
            byte[] transactionAmount =
result.getValue(Bytes.toBytes("transaction_data"),
Bytes.toBytes("transaction_amount"));
            byte[] cardType =
result.getValue(Bytes.toBytes("transaction_data"),
Bytes.toBytes("transaction_card_type"));
            byte[] website =
result.getValue(Bytes.toBytes("transaction_data"),
Bytes.toBytes("transaction_ecommerce_website_name"));
            byte[] datetime =
result.getValue(Bytes.toBytes("transaction_data"),
Bytes.toBytes("transaction_datetime"));
            byte[] productName =
result.getValue(Bytes.toBytes("transaction_data"),
Bytes.toBytes("transaction_product_name"));

            byte[] city = result.getValue(Bytes.toBytes("customer_data"),
Bytes.toBytes("transaction_city_name"));
            byte[] country = result.getValue(Bytes.toBytes("customer_data"),
Bytes.toBytes("transaction_country_name"));

            System.out.println("Row key: 10");
            System.out.println("transaction_amount: " +
Bytes.toString(transactionAmount));
            System.out.println("transaction_card_type: " +
Bytes.toString(cardType));
            System.out.println("transaction_ecommerce_website_name: " +
Bytes.toString(website));
            System.out.println("transaction_datetime: " +
Bytes.toString(datetime));
            System.out.println("transaction_product_name: " +
Bytes.toString(productName));
            System.out.println("transaction_city_name: " +
Bytes.toString(city));
            System.out.println("transaction_country_name: " +
Bytes.toString(country));

        } catch (IOException e) {
            e.printStackTrace();
        } finally {
            // Close resources
            try {
                if (table != null) table.close();
                if (connection != null) connection.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }
}
```



```
hadoop@hbase:/tmp$ javac -cp "$(hbase classpath)" retrievedata.java
hadoop@hbase:/tmp$ java -cp ".:$(hbase classpath)" retrievedata
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
SLF4J: Failed to load class "org.slf4j.impl.StaticMDCBinder".
SLF4J: Defaulting to no-operation MDCAdapter implementation.
SLF4J: See http://www.slf4j.org/codes.html#no_static_mdc_binder for further details.
Row key: 10
transaction_amount: 499.99
transaction_card_type: Visa
transaction_ecommerce_website_name: www.amazon.com
transaction_datetime: 2019-05-25 14:30:00
transaction_product_name: Gaming Laptop
transaction_city_name: Karachi
transaction_country_name: Pakistan
hadoop@hbase:/tmp$
```

3. Deleting data from HBase Table Using Java API

Create a Delete with a specific row: Delete delete = new Delete(Bytes.toBytes("row1"));

Set a timestamp for row deletes: delete.setTimestamp(1); Delete a specific version in one column: delete.deleteColumn(Bytes.toBytes("colfam1"), Bytes.toBytes("qual1"), 1); Delete all versions in one column: delete.deleteColumns(Bytes.toBytes("colfam2"), Bytes.toBytes("qual1")); Delete the given and all older versions in one column:

delete.deleteColumns(Bytes.toBytes("colfam2"), Bytes.toBytes("qual3"), 15);

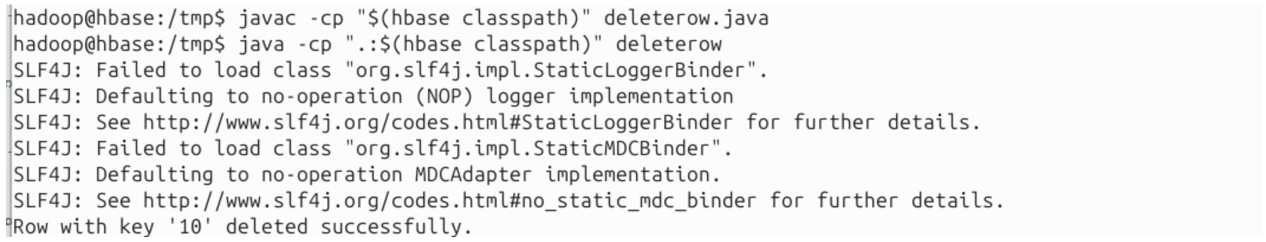
Delete the entire family, all columns and versions:
delete.deleteFamily(Bytes.toBytes("colfam3"));

Delete the given and all older versions in the entire column family, that is, from all columns therein:

delete.deleteFamily(Bytes.toBytes("colfam3"), 3); Delete the data from the HBase table: table.delete(delete); table.close();

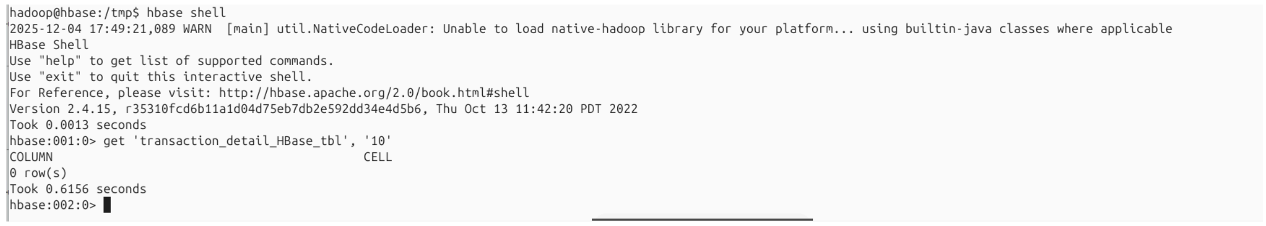
Deleterow.java:

```
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.hbase.HBaseConfiguration;
import org.apache.hadoop.hbase.TableName;
import org.apache.hadoop.hbase.client.*;
import org.apache.hadoop.hbase.util.Bytes;
import java.io.IOException;

public class deleterow {
    public static void main(String[] args) {
        Configuration config = HBaseConfiguration.create();
        Connection connection = null;
        Table table = null;

        try {
            // Connect to HBase
            connection = ConnectionFactory.createConnection(config);
            table =
connection.getTable(TableName.valueOf("transaction_detail_HBase_tbl"));

            // Delete the entire row with key "10"
            Delete deleteRow = new Delete(Bytes.toBytes("10"));
            table.delete(deleteRow);

            System.out.println("Row with key '10' deleted successfully.");

        } catch (IOException e) {
            e.printStackTrace();
        } finally {
            try {
                if (table != null) table.close();
                if (connection != null) connection.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }
}
```



```
hadoop@hbase:/tmp$ javac -cp "$(hbase classpath)" deleterow.java
hadoop@hbase:/tmp$ java -cp ".:$(hbase classpath)" deleterow
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
SLF4J: Failed to load class "org.slf4j.impl.StaticMDCBinder".
SLF4J: Defaulting to no-operation MDCAdapter implementation.
SLF4J: See http://www.slf4j.org/codes.html#no_static_mdc_binder for further details.
Row with key '10' deleted successfully.
```

Checking if row has been deleted from Hbase shell:



```
hadoop@hbase:/tmp$ hbase shell
2025-12-04 17:49:21,089 WARN  [main] util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
HBase Shell
Use "help" to get list of supported commands.
Use "exit" to quit this interactive shell.
For Reference, please visit: http://hbase.apache.org/2.0/book.html#shell
Version 2.4.15, r35310fcd6b11a1d04d75eb7db2e592dd34e4d5b6, Thu Oct 13 11:42:20 PDT 2022
Took 0.0013 seconds
hbase:001:0> get 'transaction_detail_HBase_tbl', '10'
COLUMN                                             CELL
0 row(s)
Took 0.6156 seconds
hbase:002:0>
```

4. Scan The HBase Table Using Java API

The getScanner() method of the HTable class can be used to scan the entire data of Hbase table. The getScanner() method expects an instance of the Scan class. To summarize, steps to scan the entire data of HBase table are:

Instantiate the Configuration Class:

Configuration conf = HBaseConfiguration.create(); Instantiate the HTable Class:

HTable table = new HTable(conf, "testtable");

Instantiate the Scan Class: Scan scan = new Scan();

// scan the columns scan.addColumn(Bytes.toBytes("colfam1"), Bytes.toBytes("qual1")); scan.addColumn(Bytes.toBytes("colfam1"), Bytes.toBytes("val1"));

Get the ResultScanner instances by calling HTable's getScanner() method:

ResultScanner scanner = table.getScanner(scan); for (Result result = scanner.next(); result != null; result=scanner.next())

System.out.println("Found row : " + result);

Read values from ResultScanner and close it: scanner.close();

scandata.java:

```
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.hbase.HBaseConfiguration;
import org.apache.hadoop.hbase.TableName;
import org.apache.hadoop.hbase.client.*;
import org.apache.hadoop.hbase.util.Bytes;
import java.io.IOException;

public class scandata {
    public static void main(String[] args) {
        Configuration conf = HBaseConfiguration.create();
        Connection connection = null;
        Table table = null;
        ResultScanner scanner = null;

        int limit = 5; // Limit the number of rows
        int count = 0;

        try {
            connection = ConnectionFactory.createConnection(conf);
            table =
connection.getTable(TableName.valueOf("transaction_detail_HBase_tbl"));

            Scan scan = new Scan();
            // Optional: scan specific columns
            // scan.addColumn(Bytes.toBytes("transaction_data"),
Bytes.toBytes("transaction_amount"));
            // scan.addColumn(Bytes.toBytes("customer_data"),
Bytes.toBytes("transaction_city_name"));

            scanner = table.getScanner(scan);

            for (Result result = scanner.next(); result != null; result =
scanner.next()) {
                if (count >= limit) break; // Stop after reaching the limit
                count++;

                String rowKey = Bytes.toString(result.getRow());
                System.out.println("Found row: " + rowKey);

                result.listCells().forEach(cell -> {
                    String family = Bytes.toString(cell.getFamilyArray(),
cell.getFamilyOffset(), cell.getFamilyLength());
                    String qualifier =
Bytes.toString(cell.getQualifierArray(), cell.getQualifierOffset(),
cell.getQualifierLength());
                    String value = Bytes.toString(cell.getValueArray(),
cell.getValueOffset(), cell.getValueLength());
                    System.out.println("  " + family + ":" + qualifier + " = "
+ value);
                });
                System.out.println();
            }
```

```java
        } catch (IOException e) {
            e.printStackTrace();
        } finally {
            try {
                if (scanner != null) scanner.close();
                if (table != null) table.close();
                if (connection != null) connection.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }
}
```

```
hadoop@hbase:/tmp$ java -cp ".:$(hbase classpath)" scandata
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
SLF4J: Failed to load class "org.slf4j.impl.StaticMDCBinder".
SLF4J: Defaulting to no-operation MDCAdapter implementation.
SLF4J: See http://www.slf4j.org/codes.html#no_static_mdc_binder for further details.
Found row: 1
  customer_data:transaction_city_name = Mumbai
  customer_data:transaction_country_name = India
  transaction_data:transaction_amount = 50.85
  transaction_data:transaction_card_type = MasterCard
  transaction_data:transaction_datetime = 2019-05-14 15:24:12
  transaction_data:transaction_ecommerce_website_name = www.ebay.com
  transaction_data:transaction_product_name = Laptop

Found row: 10
  customer_data:transaction_city_name = Karachi
  customer_data:transaction_country_name = Pakistan
  transaction_data:transaction_amount = 499.99
  transaction_data:transaction_card_type = Visa
  transaction_data:transaction_datetime = 2019-05-25 14:30:00
  transaction_data:transaction_ecommerce_website_name = www.amazon.com
  transaction_data:transaction_product_name = Gaming Laptop
```

```
Found row: 10
  customer_data:transaction_city_name = Karachi
  customer_data:transaction_country_name = Pakistan
  transaction_data:transaction_amount = 499.99
  transaction_data:transaction_card_type = Visa
  transaction_data:transaction_datetime = 2019-05-25 14:30:00
  transaction_data:transaction_ecommerce_website_name = www.amazon.com
  transaction_data:transaction_product_name = Gaming Laptop

Found row: 100
  customer_data:transaction_city_name = Karachi
  customer_data:transaction_country_name = Pakistan
  transaction_data:transaction_amount = 499.99
  transaction_data:transaction_card_type = Visa
  transaction_data:transaction_datetime = 2019-05-25 14:30:00
  transaction_data:transaction_ecommerce_website_name = www.amazon.com
  transaction_data:transaction_product_name = Gaming Laptop

Found row: 2
  customer_data:transaction_city_name = Pune
  customer_data:transaction_country_name = India
  transaction_data:transaction_amount = 259.12
  transaction_data:transaction_card_type = MasterCard
  transaction_data:transaction_datetime = 2019-05-14 15:24:13
  transaction_data:transaction_ecommerce_website_name = www.amazon.com
  transaction_data:transaction_product_name = Wrist Band
```

```
Found row: 10
  customer_data:transaction_city_name = Karachi
  customer_data:transaction_country_name = Pakistan
  transaction_data:transaction_amount = 499.99
  transaction_data:transaction_card_type = Visa
  transaction_data:transaction_datetime = 2019-05-25 14:30:00
  transaction_data:transaction_ecommerce_website_name = www.amazon.com
  transaction_data:transaction_product_name = Gaming Laptop

Found row: 100
  customer_data:transaction_city_name = Karachi
  customer_data:transaction_country_name = Pakistan
  transaction_data:transaction_amount = 499.99
  transaction_data:transaction_card_type = Visa
  transaction_data:transaction_datetime = 2019-05-25 14:30:00
  transaction_data:transaction_ecommerce_website_name = www.amazon.com
  transaction_data:transaction_product_name = Gaming Laptop

Found row: 2
  customer_data:transaction_city_name = Pune
  customer_data:transaction_country_name = India
  transaction_data:transaction_amount = 259.12
  transaction_data:transaction_card_type = MasterCard
  transaction_data:transaction_datetime = 2019-05-14 15:24:13
  transaction_data:transaction_ecommerce_website_name = www.amazon.com
  transaction_data:transaction_product_name = Wrist Band
```