ASSIGNMENT ADVANCED HBASE 2

Case Study Description:

Let us take up the CUSTOMER and TRANSACTIONS table we have created in the Let's Do Together section. Let us solve the following use cases using these tables :-

SOLUTION:

EXPLANATION: SO HERE WE HAVE CREATED TWO TABLES "CUSTOMER" AND "TRANSACTIONS". SO FOR THAT WE NEED TO CREATE A DATABASE IN HIVE AND THEN CREATE THE TABLES. SO BELOW WE CREATED A DATABASE "acadgilddb" AND THEN USING THE SAME

COMMAND: create database acadgilddb;

SOLUTION REPORT:

```
hive> create database acadgilddb;
OK
Time taken: 0.67 seconds
```

COMMAND: use acadgilddb;

```
hive> use acadgilddb;
OK
Time taken: 0.055 seconds
```

COMMAND: CREATE TABLE CUSTOMER(custid INT, fname STRING, lname STRING, age INT, profession STRING)

```
         row format delimited
       fields terminated by ',';
```

EXPLANATION: CREATED A TABLE 'CUSTOMER' WITH SCHEMA FOR CUSTOMER ID, FIRST NAME, LAST NAME, AGE, PROFESSION. THEN LATER WE NEED TO LOAD THE DATASET INSIDE THE TABLE WHICH WILL HAVE ROWS DELIMITED BY '.' .

SOLUTION REPORT:

```
hive> CREATE TABLE CUSTOMER(
    > custid INT,
    > fname STRING,
    > lname STRING,
    > age INT,
    > profession STRING)
    > row format delimited fields terminated by ',';
OK
Time taken: 2.249 seconds
```

EXPLANATION: THE DATASET 'customer.txt' IS SAVED IN LOCAL DIRECTORY AT THE PATH '/home/acadgild/customer.txt' . THEN USING THE LOAD COMMAND WE WILL LOAD THE DATASET INTO THE CUSTOMER TABLE.

COMMAND:load data local inpath 'file:///home/acadgild/customer.txt' into table CUSTOMER.

SOLUTION REPORT:

```
hive> load data local inpath 'file:///home/acadgild/customer.txt' into
table CUSTOMER;
Loading data to table acadgilddb.customer
OK
Time taken: 5.711 seconds
```

EXPLANATION: CREATED A TABLE 'TRANSACTIONS' WITH SCHEMA FOR TRANSACTION NUMBER, TRANSACTION DATE, CUSTOMER ID, AMOUNT, CATEGORY, PRODUCT, CITY, STATE, SPENT BY. THEN LATER WE NEED TO LOAD THE DATASET INSIDE THE TABLE WHICH WILL HAVE ROWS DELIMITED BY '.' .

COMMAND: CREATE TABLE TRANSACTIONS(txnno INT, txndate STRING, custno INT, amount DOUBLE, category STRING, product STRING, city STRING, state STRING, spendby STRING)

```
         row format delimited
       fields terminated by ',';
```

SOLUTION:

```
hive> CREATE TABLE TRANSACTIONS (
    > txnno INT,
    > txndate STRING,
    > custno INT,
    > amount DOUBLE,
    > category STRING,
    > product STRING,
    > city STRING,
    > state STRING,
    > spendby STRING)
    > row format delimited fields terminated by ',';
OK
Time taken: 0.358 seconds
```

EXPLANATION: THE DATASET 'transaction.txt' IS SAVED IN LOCAL DIRECTORY AT THE PATH '/home/acadgild/transaction.txt' . THEN USING THE LOAD COMMAND WE WILL LOAD THE DATASET INTO THE CUSTOMER TABLE.

COMMAND:load data local inpath 'file:///home/acadgild/transaction.txt' into table TRANSACTIONS;

SOLUTION REPORT:

```
hive> load data local inpath 'file:///home/acadgild/transaction.txt' into
table TRANSACTIONS;
Loading data to table acadgilddb.transactions
OK
Time taken: 2.162 seconds
```

EXPLANATION: USING THE SELECT OPERATION WE CAN SEE THE DATASET HAS BEEN LOADED TO THE CUSTOMER TABLE AND TRANSACTIONS TABLE.

SOLUTION REPORT:

```
hive> SELECT * FROM CUSTOMER;
```

```
OK
101   Amitabh     Bacchan     65    Actor
102   Sharukh     Khan  45    Doctor
103   Akshay      Kumar 38    Dentist
104   Anubahv     kumar 58    Business
105   Pawan Trivedi     34    service
106   Aamir Null  42    scientest
107   Salman      Khan  43    Surgen
108   Ranbir      Kapoor      26    Industrialist
NULL  NULL  NULL  NULL  NULL
Time taken: 0.394 seconds, Fetched: 9 row(s)

hive> SELECT * FROM TRANSACTIONS;
OK
97834 05/02/2018  101   965.0 Entertainment    Movie Pune  Maharashtra
      Daughter
98396 12/01/2018  102   239.0 Food  Grocery    Patna Bihar Self
34908 06/01/2018  101   875.0 Travel      Air   Bangalore   Karnataka
      Spouse
70958 17/02/2018  104   439.0 Food  Restaurant  Delhi Delhi Wife
9874  21/01/2018  105   509.0 Entertainment    Park  Kolkata     West
Bengal      NULL
94585 19/01/2018  106   629.0 Rent  House Hyderabad   Telangana   Self
45509 20/01/2018  107   953.0 Travel      Rail  Chennai     Tamil Nadu
      Brother
7864  01/02/2018  108   569.0 Rent  Parking    Goa   Goa   Wife
NULL  NULL  NULL  NULL  NULL  NULL  NULL  NULL  NULL
Time taken: 0.66 seconds, Fetched: 9 row(s)
```

TASK 1: Find out the number of transaction done by each customer (These should be take up in module 8 itself)

COMMAND: SELECT CUSTID, FNAME, SUM(AMOUNT) FROM CUSTOMER C, TRANSACTIONS T WHERE C.CUSTID=T.CUSTNO GROUP BY CUSTID, FNAME;

EXPLANATION: HERE THIS OPERATION GIVES THE RESULT WITH THE NUMBER OF TRANSACTIONS DONE BY EACH CUSTOMER. SO THE SELECT OPERATION SELECTS THE CUSTOMER ID, FIRST AND THE TOTAL OF THE AMOUNT FROM THE TABLES CUSTOMER AND TRANSACTIONS WHICH ARE JOINED TOGETHER USING 'JOIN' KEYWORD WITH ALIAS NAME 'C' FOR CUSTOMER AND 'T' FOR TRANSACTIONS.

SOLUTION REPORT:

```
hive> SELECT CUSTID, FNAME, SUM(AMOUNT) FROM CUSTOMER C, TRANSACTIONS T
WHERE C.CUSTID=T.CUSTNO GROUP BY CUSTID, FNAME;
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.
Query ID = acadgild_20190109035131_6c758d49-1a24-4aac-95eb-52d919d64846
Total jobs = 1
```

```
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/acadgild/install/hive/apache-
hive-2.3.2-bin/lib/log4j-slf4j-impl-
2.6.2.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/acadgild/install/hadoop/hadoop-
2.6.5/share/hadoop/common/lib/slf4j-log4j12-
1.7.5.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an
explanation.
SLF4J: Actual binding is of type
[org.apache.logging.slf4j.Log4jLoggerFactory]
2019-01-09 03:51:52    Starting to launch local task to process map join;
      maximum memory = 518979584
2019-01-09 03:51:56    Dump the side-table for tag: 0 with group count: 8
into file: file:/tmp/acadgild/35c7bbfa-9569-48e6-9072-
77162e214a97/hive_2019-01-09_03-51-31_385_5914681181293595335-1/-local-
10005/HashTable-Stage-2/MapJoin-mapfile120--.hashtable
2019-01-09 03:51:56    Uploaded 1 File to: file:/tmp/acadgild/35c7bbfa-
9569-48e6-9072-77162e214a97/hive_2019-01-09_03-51-
31_385_5914681181293595335-1/-local-10005/HashTable-Stage-2/MapJoin-
mapfile120--.hashtable (469 bytes)
2019-01-09 03:51:56    End of local task; Time Taken: 4.835 sec.
Execution completed successfully
MapredLocal task succeeded
Launching Job 1 out of 1
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1546964532394_0014, Tracking URL =
http://localhost:8088/proxy/application_1546964532394_0014/
Kill Command = /home/acadgild/install/hadoop/hadoop-2.6.5/bin/hadoop job
-kill job_1546964532394_0014
Hadoop job information for Stage-2: number of mappers: 1; number of
reducers: 1
2019-01-09 03:52:19,191 Stage-2 map = 0%,  reduce = 0%
2019-01-09 03:52:39,964 Stage-2 map = 100%,  reduce = 0%, Cumulative CPU
6.29 sec
2019-01-09 03:53:01,987 Stage-2 map = 100%,  reduce = 67%, Cumulative CPU
10.62 sec
2019-01-09 03:53:04,568 Stage-2 map = 100%,  reduce = 100%, Cumulative
CPU 11.83 sec
MapReduce Total cumulative CPU time: 11 seconds 830 msec
Ended Job = job_1546964532394_0014
MapReduce Jobs Launched:
Stage-Stage-2: Map: 1  Reduce: 1   Cumulative CPU: 11.83 sec   HDFS Read:
13364 HDFS Write: 292 SUCCESS
Total MapReduce CPU Time Spent: 11 seconds 830 msec
OK
101   Amitabh    1840.0
102   Sharukh    239.0
```

```
104   Anubahv     439.0
105   Pawan 509.0
106   Aamir 629.0
107   Salman      953.0
108   Ranbir      569.0
Time taken: 95.689 seconds, Fetched: 7 row(s)
```

TASK 2: Create a new table called TRANSACTIONS_COUNT. This table should have 3 fields - custid, fname and count. (Again to be done in module 8)

COMMAND:

```
create table TRANSACTION_COUNT(custid INT, fname STRING, count
INT)
         row format delimited
         fields terminated by ',';
```

EXPLANATION: CREATED TABLE TRANSACTIONS_COUNT AND PROVIDED SCHEMA FOR THE FIELDS CUSTOMER ID, FIRST NAME AND COUNT.

SOLUTION REPORT:

```
hive> create table TRANSACTION_COUNT(custid INT, fname STRING, count INT)
    > row format delimited
    > fields terminated by ',';
OK
Time taken: 0.219 seconds
```

TASK 3: Now write a hive query in such a way that the query populates the data obtained in Step 1 above and populate the table in step 2 above. (This has to be done in module 9).

COMMAND:

```
INSERT INTO TABLE TRANSACTION_COUNT SELECT CUSTID, FNAME,
COUNT(DISTINCT CUSTID) FROM CUSTOMER A JOIN TRANSACTIONS B WHERE
A.CUSTID=B.CUSTNO GROUP BY CUSTID,FNAME;
```

EXPLANATION: USING THE INSERT COMMAND WE ARE LOADING THE TABLE WITH DATA PRODUCED IN THE OUTPUT OF TASK 1. SO HERE WE USING A SELECT OPERATION TO SAVE CUSTOMER ID, FIRST NAME AND COUNT FROM THE COMBINED TABLE(CUSTOMER AND TRANSACTIONS) IN KEYWORD "COUNT(DISTICT CUSTID)" WILL GIVE THE NUMBER OF CUSTOMER ID'S AND THEN INSERT KEYWORD WILL ENTER THE FOLLOWING DATA INTO THE TABLE UNDER FIELDS CUSTID, FNAME AND COUNT.

SOLUTION REPORT:

```
hive> INSERT INTO TABLE TRANSACTION_COUNT SELECT CUSTID, FNAME,
COUNT(DISTINCT CUSTID) FROM CUSTOMER A JOIN TRANSACTIONS B WHERE
A.CUSTID=B.CUSTNO GROUP BY CUSTID,FNAME;
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.
Query ID = acadgild_20190109021312_acc018a2-685e-4d27-99e6-ea66495b5ed0
Total jobs = 1
```

```
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/acadgild/install/hive/apache-
hive-2.3.2-bin/lib/log4j-slf4j-impl-
2.6.2.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/acadgild/install/hadoop/hadoop-
2.6.5/share/hadoop/common/lib/slf4j-log4j12-
1.7.5.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an
explanation.
SLF4J: Actual binding is of type
[org.apache.logging.slf4j.Log4jLoggerFactory]
2019-01-09 02:13:23    Starting to launch local task to process map join;
      maximum memory = 518979584
2019-01-09 02:13:26    Dump the side-table for tag: 0 with group count: 8
into file: file:/tmp/acadgild/35c7bbfa-9569-48e6-9072-
77162e214a97/hive_2019-01-09_02-13-12_128_8238481662940432256-1/-local-
10003/HashTable-Stage-2/MapJoin-mapfile90--.hashtable
2019-01-09 02:13:26    Uploaded 1 File to: file:/tmp/acadgild/35c7bbfa-
9569-48e6-9072-77162e214a97/hive_2019-01-09_02-13-
12_128_8238481662940432256-1/-local-10003/HashTable-Stage-2/MapJoin-
mapfile90--.hashtable (469 bytes)
2019-01-09 02:13:26    End of local task; Time Taken: 2.88 sec.
Execution completed successfully
MapredLocal task succeeded
Launching Job 1 out of 1
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1546964532394_0011, Tracking URL =
http://localhost:8088/proxy/application_1546964532394_0011/
Kill Command = /home/acadgild/install/hadoop/hadoop-2.6.5/bin/hadoop job
-kill job_1546964532394_0011
Hadoop job information for Stage-2: number of mappers: 1; number of
reducers: 1
2019-01-09 02:13:40,210 Stage-2 map = 0%,  reduce = 0%
2019-01-09 02:13:51,337 Stage-2 map = 100%,  reduce = 0%, Cumulative CPU
2.95 sec
2019-01-09 02:14:05,726 Stage-2 map = 100%,  reduce = 100%, Cumulative
CPU 6.69 sec
MapReduce Total cumulative CPU time: 6 seconds 690 msec
Ended Job = job_1546964532394_0011
Loading data to table acadgilddb.transaction_count
MapReduce Jobs Launched:
Stage-Stage-2: Map: 1  Reduce: 1   Cumulative CPU: 6.69 sec   HDFS Read:
13962 HDFS Write: 176 SUCCESS
Total MapReduce CPU Time Spent: 6 seconds 690 msec
OK
Time taken: 55.538 seconds
```

EXPLANATION: USING THE SELECT OPERATION WE CAN SEE THE TRANSACTION_COUNT HAS BEEN LOADED WITH THE REQUIRED DATA.

SOLUTION REPORT:

```
hive> SELECT * FROM TRANSACTION_COUNT;
OK
101   Amitabh     1
102   Sharukh     1
104   Anubahv     1
105   Pawan 1
106   Aamir 1
107   Salman      1
108   Ranbir      1
Time taken: 0.201 seconds, Fetched: 7 row(s)
```

TASK 4:Now lets make the TRANSACTIONS_COUNT table Hbase complaint. In the sence, use Ser Des And Storate handler features of hive to change the TRANSACTIONS_COUNT table to be able to create a TRANSACTIONS table
in Hbase. (This has to be done in module 10)

EXPLANATION: SO FOR DOING THE ABOVE OPERATION, FIRST LOGIN INTO HBASE SHELL AND THEN CREATE A TRANSACTIONS TABLE IN HBASE WITH A COLUMN FAMILY "DETAILS".

COMMAND: create 'TRANSACTIONS','DETAILS'

SOLUTION REPORT:

```
hbase(main):003:0> create 'TRANSACTIONS','DETAILS'
0 row(s) in 2.8030 seconds

=> Hbase::Table - TRANSACTIONS
```

EXPLANATION: NOW WE CREATED A HIVE TABLE POINTING TO HBASE TABLE. HERE CREATED A EXTERNAL TABLE "TRANSACTIONS_hbase" WITH THE SCHEMA FOR THE FIELDS CUSTOMER ID, FIRST NAME AND COUNT. AS WE CREATED A NON-NATIVE HIVE tABLE USING STORAGE HANDLER SO USED THE STORED BY clause. AS FOR THE INTEGRATING HBASE WITH HIVE STORAGE HANDLERS IN HIVE ARE USED. THEY ARE THE COMBINATION OF INPUT AND OUTPUT FORMAT AS SERDE A SPECFIC CODE THAT HIVE USES TO IDENTIFY AN EXTERNAL ENTITY AS A HIVE TABLE. HERE "hbase.columns.mapping" IS USED TO MAP THE HIVE COLUMNS WITH THE HBASE COLUMNS . THE FIRST COLUMN MUST BE THE KEY COLUMN WHICH WOULD ALSO BE SAME AS HBASE'S ROW KEY COLUMN AND PROVIDED THE HBASE TABLE "TRANSACTIONS" SO THAT THE HIVE TABLE CONNECTS WITH THE HBASE TABLE.

COMMAND:

```
create external table TRANSACTIONS_hbase(custid INT, fname
string, count INT)
         STORED BY 'org.apache.hadoop.hive.hbase.HBaseStorageHandler'
         with serdeproperties
("hbase.columns.mapping"=":key,DETAILS:fname,DETAILS:count")
         tblproperties("hbase.table.name"="TRANSACTIONS");
```

SOLUTION REPORT:

```
hive> create external table TRANSACTIONS_hbase(custid INT, fname string,
count INT)
    > STORED BY 'org.apache.hadoop.hive.hbase.HBaseStorageHandler'
    >  with serdeproperties
("hbase.columns.mapping"=":key,DETAILS:fname,DETAILS:count")
    >  tblproperties("hbase.table.name"="TRANSACTIONS");
OK
Time taken: 1.947 seconds
```

TASK 5: Now insert the data in TRANSACTIONS_COUNT table using the query in step 3 again, this should populate the Hbase TRANSACTIONS table automatically (This has to be done in module 10)

EXPLANATION: HERE USED THE SAME QUERY WHICH WAS USED IN TASK 3, TO INSERT THE DATA INTO THE TABLE "TRANSACTIONS_hbase" WHICH WAS CREATED ABOVE AS A EXTERNAL TABLE SO THAT THE DATA IS INSERTED INTO HIVE TABLE AND LATER WHEN THE QUERY IN EXECUTED THE DATA IS AUTOMATICALLY LOADED INTO HBASE TABLE "TRANSACTIONS" AS WELL.

COMMAND: INSERT INTO TABLE TRANSACTIONS_hbase SELECT CUSTID, FNAME, COUNT(DISTINCT CUSTID) FROM CUSTOMER A JOIN TRANSACTIONS B WHERE A.CUSTID=B.CUSTNO GROUP BY CUSTID,FNAME;

SOLUTION REPORT:

```
hive> INSERT INTO TABLE TRANSACTIONS_hbase SELECT CUSTID, FNAME,
COUNT(DISTINCT CUSTID) FROM CUSTOMER A JOIN TRANSACTIONS B WHERE
A.CUSTID=B.CUSTNO GROUP BY CUSTID,FNAME;
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.
Query ID = acadgild_20190109025722_20bbbd3a-64cc-4704-867d-bc3aa7fe458f
Total jobs = 1
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/acadgild/install/hive/apache-
hive-2.3.2-bin/lib/log4j-slf4j-impl-
2.6.2.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/acadgild/install/hadoop/hadoop-
2.6.5/share/hadoop/common/lib/slf4j-log4j12-
1.7.5.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an
explanation.
SLF4J: Actual binding is of type
[org.apache.logging.slf4j.Log4jLoggerFactory]
2019-01-09 02:57:43    Starting to launch local task to process map join;
      maximum memory = 518979584
2019-01-09 02:57:49    Dump the side-table for tag: 0 with group count: 8
into file: file:/tmp/acadgild/35c7bbfa-9569-48e6-9072-
77162e214a97/hive_2019-01-09_02-57-22_059_2728555604714371678-1/-local-
10002/HashTable-Stage-4/MapJoin-mapfile110--.hashtable
2019-01-09 02:57:49    Uploaded 1 File to: file:/tmp/acadgild/35c7bbfa-
9569-48e6-9072-77162e214a97/hive_2019-01-09_02-57-
22_059_2728555604714371678-1/-local-10002/HashTable-Stage-4/MapJoin-
mapfile110--.hashtable (469 bytes)
2019-01-09 02:57:49    End of local task; Time Taken: 5.698 sec.
```

```
Execution completed successfully
MapredLocal task succeeded
Launching Job 1 out of 1
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1546964532394_0013, Tracking URL =
http://localhost:8088/proxy/application_1546964532394_0013/
Kill Command = /home/acadgild/install/hadoop/hadoop-2.6.5/bin/hadoop job
-kill job_1546964532394_0013
Hadoop job information for Stage-4: number of mappers: 1; number of
reducers: 1
2019-01-09 02:58:22,299 Stage-4 map = 0%,  reduce = 0%
2019-01-09 02:58:42,899 Stage-4 map = 100%,  reduce = 0%, Cumulative CPU
5.99 sec
2019-01-09 02:59:06,766 Stage-4 map = 100%,  reduce = 67%, Cumulative CPU
10.62 sec
2019-01-09 02:59:11,512 Stage-4 map = 100%,  reduce = 100%, Cumulative
CPU 14.47 sec
MapReduce Total cumulative CPU time: 14 seconds 470 msec
Ended Job = job_1546964532394_0013
MapReduce Jobs Launched:
Stage-Stage-4: Map: 1  Reduce: 1   Cumulative CPU: 14.47 sec   HDFS Read:
20550 HDFS Write: 0 SUCCESS
Total MapReduce CPU Time Spent: 14 seconds 470 msec
OK
Time taken: 111.756 seconds
```

EXPLANATION: HERE USING SELECT OPERATION WE CAN SEE THE DATA HAS BEEN INSERTED INTO THE HIVE TABLE "TRANSACTIONS_hbase".

SOLUTION REPORT:

```
hive> select * from TRANSACTIONS_hbase;
OK
101   Amitabh     1
102   Sharukh     1
104   Anubahv     1
105   Pawan 1
106   Aamir 1
107   Salman      1
108   Ranbir      1
Time taken: 0.921 seconds, Fetched: 7 row(s)
```

TASK 6: Now from the Hbase level, write the Hbase java API code to access and scan the TRANSACTIONS table data from java level.

COMMAND: scan 'TRANSACTIONS'

EXPLANATION: TO CHECK WHETHER THE DATA FROM THE HIVE TABLE "TRANSACTIONS_hbase" HAS BEEN LOADED INTO THE HBASE TABLE "TRANSACTIONS".SO USED THE BELOW COMMAND.

SOLUTION REPORT:

```
hbase(main):004:0> scan 'TRANSACTIONS'
ROW                   COLUMN+CELL
 101                  column=DETAILS:count, timestamp=1546982950134,
value=1
 101                  column=DETAILS:fname, timestamp=1546982950134,
value=Amita
                      bh
 102                  column=DETAILS:count, timestamp=1546982950134,
value=1
 102                  column=DETAILS:fname, timestamp=1546982950134,
value=Sharu
                      kh
 104                  column=DETAILS:count, timestamp=1546982950134,
value=1
 104                  column=DETAILS:fname, timestamp=1546982950134,
value=Anuba
                      hv
 105                  column=DETAILS:count, timestamp=1546982950134,
value=1
 105                  column=DETAILS:fname, timestamp=1546982950134,
value=Pawan
 106                  column=DETAILS:count, timestamp=1546982950134,
value=1
 106                  column=DETAILS:fname, timestamp=1546982950134,
value=Aamir
 107                  column=DETAILS:count, timestamp=1546982950134,
value=1
 107                  column=DETAILS:fname, timestamp=1546982950134,
value=Salma
                      n
 108                  column=DETAILS:count, timestamp=1546982950134,
value=1
 108                  column=DETAILS:fname, timestamp=1546982950134,
value=Ranbi
                      r
7 row(s) in 2.0580 seconds
```

OUTPUT

TASK 1:

```
101     Amitabh 1840.0
102     Sharukh 239.0
104     Anubahv 439.0
105     Pawan   509.0
106     Aamir   629.0
107     Salman  953.0
108     Ranbir  569.0
Time taken: 95.689 seconds, Fetched: 7 row(s)
```

TASK 3:

```
hive> SELECT * FROM TRANSACTION_COUNT;
OK
101     Amitabh 1
102     Sharukh 1
104     Anubahv 1
105     Pawan   1
106     Aamir   1
107     Salman  1
108     Ranbir  1
Time taken: 0.201 seconds, Fetched: 7 row(s)
```

TASK 5:

```
hive> select * from TRANSACTIONS_hbase;
OK
101     Amitabh 1
102     Sharukh 1
104     Anubahv 1
105     Pawan   1
106     Aamir   1
107     Salman  1
108     Ranbir  1
Time taken: 3.151 seconds, Fetched: 7 row(s)
```

TASK 6:

```
hbase(main):004:0> scan 'TRANSACTIONS'
ROW                   COLUMN+CELL
 101                  column=DETAILS:count, timestamp=1546982950134, value=1
 101                  column=DETAILS:fname, timestamp=1546982950134, value=Amita
                      bh
 102                  column=DETAILS:count, timestamp=1546982950134, value=1
 102                  column=DETAILS:fname, timestamp=1546982950134, value=Sharu
                      kh
 104                  column=DETAILS:count, timestamp=1546982950134, value=1
 104                  column=DETAILS:fname, timestamp=1546982950134, value=Anuba
                      hv
 105                  column=DETAILS:count, timestamp=1546982950134, value=1
 105                  column=DETAILS:fname, timestamp=1546982950134, value=Pawan
 106                  column=DETAILS:count, timestamp=1546982950134, value=1
 106                  column=DETAILS:fname, timestamp=1546982950134, value=Aamir
 107                  column=DETAILS:count, timestamp=1546982950134, value=1
 107                  column=DETAILS:fname, timestamp=1546982950134, value=Salma
                      n
 108                  column=DETAILS:count, timestamp=1546982950134, value=1
 108                  column=DETAILS:fname, timestamp=1546982950134, value=Ranbi
                      r
7 row(s) in 2.0580 seconds
```