Instructions

For each answer, please include your answer as text, and any screenshot(s) which demonstrate your answer was executed. Most importantly, make sure to include evidence your answer is correct. This will most likely be a screenshot. If you had issues, problems, or had to make assumptions include them in your answer.

Your Answers:

1. From Impala, use the two external tables **weblogs** created from **clickstream/logs_noheader** and **iplookup** created from **clickstream/iplookup_noheader** you created in the previous assignment to complete this question. Use the impala shell to answer the following questions, making sure to include the SELECT query you used to answer it.

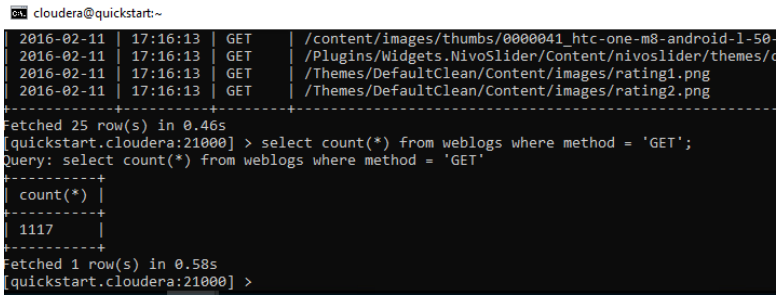
    a. How many GET and POST requests are there in the weblogs?

    > use clickstream;

    > show tables;

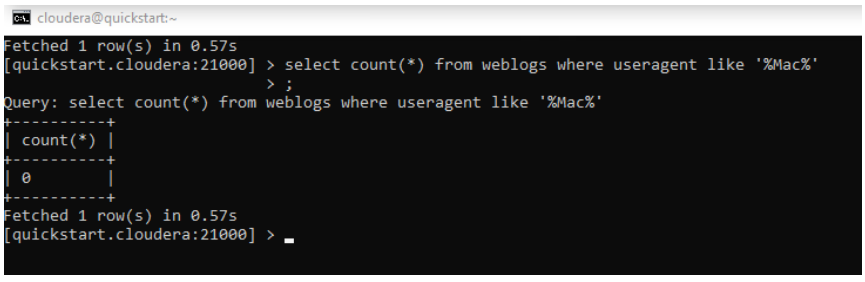
    > select * from weblogs limit 10;

    > select count(*) from weblogs where method = 'GET';



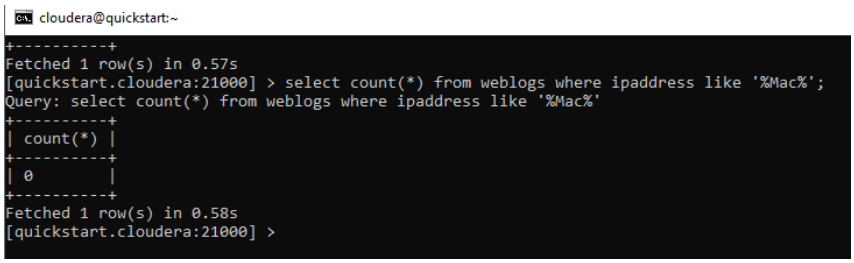
    b. How many requests have Mac in the user agent?

    > select count(*) from weblogs where useragent like '%Mac%'



    c. How many hosts (ip addresses) have Mac in the user agent?

    > select count(*) from weblogs where useragent like '%Mac%'

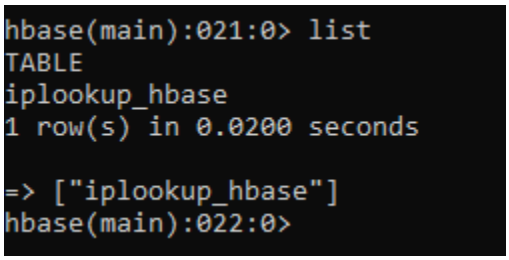


2. From the HBase shell, include the commands required to complete the following.
    a. Create a table named **computers** with column family **info**.

    disable the table because it already exists
    > disable ‘computers’

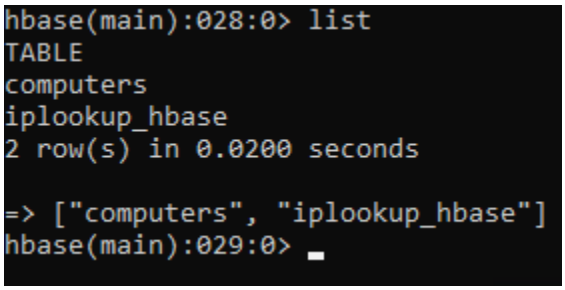
    drop the table after disabling it
    > drop ‘computers’

    check to make sure that It dropped
    > list



    create the computers table
    > create ‘computers’, ‘info’

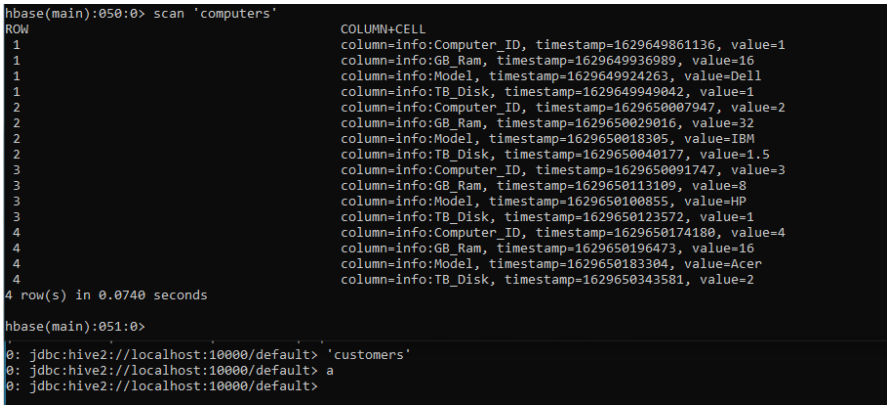
    show that the table is there
    > list



    b. Issue HBase commands to write the following data to the table in the column family:

Computer ID	Model	GB_Ram	TB_Disk
1	Dell	16	1
2	IBM	32	1.5
3	HP	8	1
4	Acer	16	2

Write the pieces of data to the table
> put ‘computers’, 1, ‘info:Computer_ID’, 1
> put ’computers’, 1, ‘info:Model’, ‘Dell’
> put ‘computers’, 1, ‘info:GB_Ram’, 16
> put ‘computers’, 1, ‘info:TB_Disk’, 1
> put ‘computers’, 2, ‘info:Computer_ID’, 2
> put ’computers’, 2, ‘info:Model’, ‘IBM’
> put ‘computers’, 2, ‘info:GB_Ram’, 32
> put ‘computers’, 2, ‘info:TB_Disk’, 1.5
> put ‘computers’, 3, ‘info:Computer_ID’, 3
> put ’computers’, 3, ‘info:Model’, ‘HP’
> put ‘computers’, 3, ‘info:GB_Ram’, 8
> put ‘computers’, 3, ‘info:TB_Disk’, 1
> put ‘computers’, 4, ‘info:Computer_ID’, 4
> put ’computers’, 4, ‘info:Model’, ‘Acer’
> put ‘computers’, 4, ‘info:GB_Ram’, 16
> put ‘computers’, 4, ‘info:TB_Disk’, 2



3. From the Hive shell, write an HQL statement to create an external Hive table from the HBase **computers** table. Then write a hive query to add up the total ram and disk across all computers. Your answer should include all HQL statements.

   create an external hive table from the hbase computers table
   > create external table computers_hbase(key int, Computer_ID int, Model string, GB_Ram float, TB_Disk float) stored by ‘org.apache.hadoop.hive.hbase.HBaseStorageHandler’ WITH

SERDEPROPERTIES(“hbase.columns.mapping”=”info:Computer_ID, info:Model, info:GB_Ram, info:TB_Disk”) TBLPROPERTIES(“hbase.table.name”=”computers”);

```
0: jdbc:hive2://localhost:10000/default> show tables;
+-------------------+--+
|     tab_name      |
+-------------------+--+
| computers         |
| computers_hbase   |
| iplookup          |
| iplookup_hbase    |
| weblogs           |
+-------------------+--+
0: jdbc:hive2://localhost:10000/default>
```

show the data in the hive table
> select * from computers_hbase;

```
0: jdbc:hive2://localhost:10000/default> select * from computers_hbase
0: jdbc:hive2://localhost:10000/default> ;
+----------------------+-----------------------------+------------------------+-------------------------+--------------------------+--+
| computers_hbase.key  | computers_hbase.computer_id | computers_hbase.model  | computers_hbase.gb_ram  | computers_hbase.tb_disk  |
+----------------------+-----------------------------+------------------------+-------------------------+--------------------------+--+
| 1                    | 1                           | Dell                   | 16.0                    | 1.0                      |
| 2                    | 2                           | IBM                    | 32.0                    | 1.5                      |
| 3                    | 3                           | HP                     | 8.0                     | 1.0                      |
| 4                    | 4                           | Acer                   | 16.0                    | 2.0                      |
+----------------------+-----------------------------+------------------------+-------------------------+--------------------------+--+
0: jdbc:hive2://localhost:10000/default>
```

write hive query to add up total ram
> select sum(computers_hbase.gb_ram) from computers_hbase

```
0: jdbc:hive2://localhost:10000/default> select sum(computers_hbase.gb_ram) from computers_hbase
0: jdbc:hive2://localhost:10000/default> ;
+-------+--+
|  _c0  |
+-------+--+
| 72.0  |
+-------+--+
```

write hive query to add up total disk
> select sum(computers_hbase.tb_disk) from computers_hbase

```
0: jdbc:hive2://localhost:10000/default> select sum(computers_hbase.tb_disk) from computers_hbase;
+------+--+
| _c0  |
+------+--+
| 5.5  |
+------+--+
0: jdbc:hive2://localhost:10000/default>
```

4. Use Hive to load the **iplookup** table you created from **clickstream/iplookup_noheader** into and HBase table, with IP address as key. Include the HQL Queries you wrote to make the table and load the data as the answer to your question.

disable the table because it already exists
> disable “iplookup_hbase”

drop the table after disabling it
> drop “iplookup_hbase”

check to make sure that It dropped
> list

```
hbase(main):054:0> list
TABLE
computers
1 row(s) in 0.0160 seconds

=> ["computers"]
hbase(main):055:0>
```

use hive to load the table into hbase
> create table iplookup_hbase3 (key string, ip string, country string, state string, city string, approxlat float, approxlng float) STORED BY ‘org.apache.hadoop.hive.hbase.HBaseStoragehandler’ WITH SERDEPROPERTIES (“hbase.columns.mapping” = “ip:key, g:country, g:state, g:city, g:approxlat, g:approxlng”) TBLPROPERTIES(“hbase.tab.ename” = “iplookup2”);

Was having some issues but got Hbase to work. Name in hive is iplookup_hbase3 and name in hbase is iplookup2

Check to make sure that it loaded in hbase
> list

```
base(main):066:0> list
ABLE
omputers
plookup
plookup2
 row(s) in 0.0140 seconds

> ["computers", "iplookup", "iplookup2"]
base(main):067:0>
```

But it doesn’t have any data right now
> scan ‘iplookup2’

```
hbase(main):067:0> scan 'iplookup2'
ROW                                                    COLUMN+CELL
 row(s) in 0.0320 seconds

hbase(main):068:0>
```

Put the data in via hive
> insert into iplookup_hbase3 select ip, country, state, city, approxlat, approxlng from iplookup;

check that the data is there now in hbase
> scan ‘iplookup2’

```
hbase(main):070:0> scan 'iplookup2'
ROW                                    COLUMN+CELL
 128.122.140.238                       column=g:approxlat, timestamp=1629654300566, value=40.712784
 128.122.140.238                       column=g:approxlng, timestamp=1629654300566, value=-74.00594
 128.122.140.238                       column=g:city, timestamp=1629654300566, value=New York
 128.122.140.238                       column=g:country, timestamp=1629654300566, value=USA
 128.122.140.238                       column=g:state, timestamp=1629654300566, value=NY
 128.230.122.180                       column=g:approxlat, timestamp=1629654300566, value=43.048122
 128.230.122.180                       column=g:approxlng, timestamp=1629654300566, value=-76.14742
 128.230.122.180                       column=g:city, timestamp=1629654300566, value=Syracuse
 128.230.122.180                       column=g:country, timestamp=1629654300566, value=USA
 128.230.122.180                       column=g:state, timestamp=1629654300566, value=NY
 155.100.169.152                       column=g:approxlat, timestamp=1629654300566, value=40.76078
 155.100.169.152                       column=g:approxlng, timestamp=1629654300566, value=-111.891045
 155.100.169.152                       column=g:city, timestamp=1629654300566, value=Salt Lake City
 155.100.169.152                       column=g:country, timestamp=1629654300566, value=USA
 155.100.169.152                       column=g:state, timestamp=1629654300566, value=UT
 172.189.252.8                         column=g:approxlat, timestamp=1629654300566, value=38.955856
 172.189.252.8                         column=g:approxlng, timestamp=1629654300566, value=-77.44782
 172.189.252.8                         column=g:city, timestamp=1629654300566, value=Dulles
 172.189.252.8                         column=g:country, timestamp=1629654300566, value=USA
 172.189.252.8                         column=g:state, timestamp=1629654300566, value=VA
 215.82.23.2                           column=g:approxlat, timestamp=1629654300566, value=39.961178
 215.82.23.2                           column=g:approxlng, timestamp=1629654300566, value=-82.998795
 215.82.23.2                           column=g:city, timestamp=1629654300566, value=Columbus
 215.82.23.2                           column=g:country, timestamp=1629654300566, value=USA
 215.82.23.2                           column=g:state, timestamp=1629654300566, value=OH
 38.68.15.223                          column=g:approxlat, timestamp=1629654300566, value=32.776665
 38.68.15.223                          column=g:approxlng, timestamp=1629654300566, value=-96.79699
 38.68.15.223                          column=g:city, timestamp=1629654300566, value=Dallas
 38.68.15.223                          column=g:country, timestamp=1629654300566, value=USA
 38.68.15.223                          column=g:state, timestamp=1629654300566, value=TX
```

5. From the HBase shell, write an HBase query to retrieve the city and state columns for all rows in the **iplookup** table.

query the city and state columns for all rows in hbase
> scan ‘iplookup2’, {COLUMNS => [‘g:city’, ‘g:state’]}

```
=> ["computers", "iplookup", "iplookup2"]
hbase(main):002:0> scan 'iplookup2', {COLUMNS => ['g:city', 'g:state']}
ROW                                   COLUMN+CELL
 128.122.140.238                      column=g:city, timestamp=1629654300566, value=New York
 128.122.140.238                      column=g:state, timestamp=1629654300566, value=NY
 128.230.122.180                      column=g:city, timestamp=1629654300566, value=Syracuse
 128.230.122.180                      column=g:state, timestamp=1629654300566, value=NY
 155.100.169.152                      column=g:city, timestamp=1629654300566, value=Salt Lake City
 155.100.169.152                      column=g:state, timestamp=1629654300566, value=UT
 172.189.252.8                        column=g:city, timestamp=1629654300566, value=Dulles
 172.189.252.8                        column=g:state, timestamp=1629654300566, value=VA
 215.82.23.2                          column=g:city, timestamp=1629654300566, value=Columbus
 215.82.23.2                          column=g:state, timestamp=1629654300566, value=OH
 38.68.15.223                         column=g:city, timestamp=1629654300566, value=Dallas
 38.68.15.223                         column=g:state, timestamp=1629654300566, value=TX
 54.114.107.209                       column=g:city, timestamp=1629654300566, value=Jersey City
 54.114.107.209                       column=g:state, timestamp=1629654300566, value=NJ
 56.216.127.219                       column=g:city, timestamp=1629654300566, value=Raleigh
 56.216.127.219                       column=g:state, timestamp=1629654300566, value=NC
```