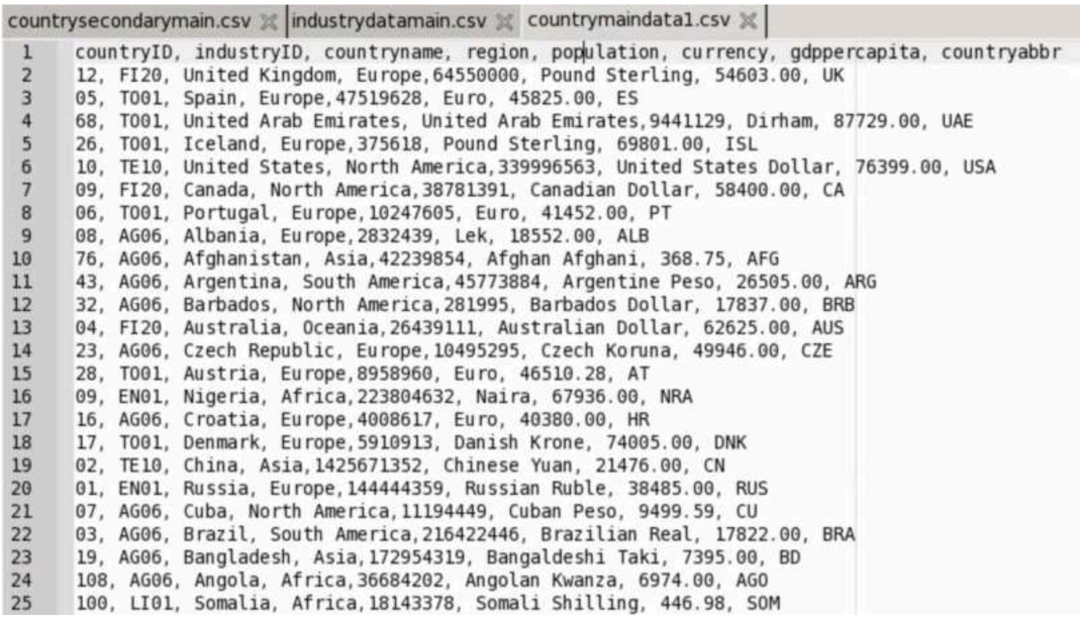
Big Data Report

Akeem Babalola

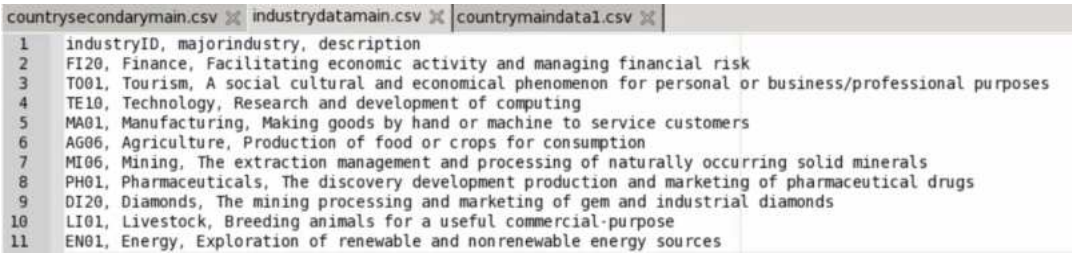
Task A – Hive Data Warehouse Design

The data warehouse in Hive consists of 50 records collectively that have been loaded into 3 tables from 3 separate CSV files. The CSV files has been separated based on the primary country data (country), data related to each available industry (industry) and secondary country data (countrysecondary). Collectively the input of these datasets into these tables allow for the execution of queries to highlight useful information. Below I will highlight a step-by-step implementation of data, queries, and the results of queries.

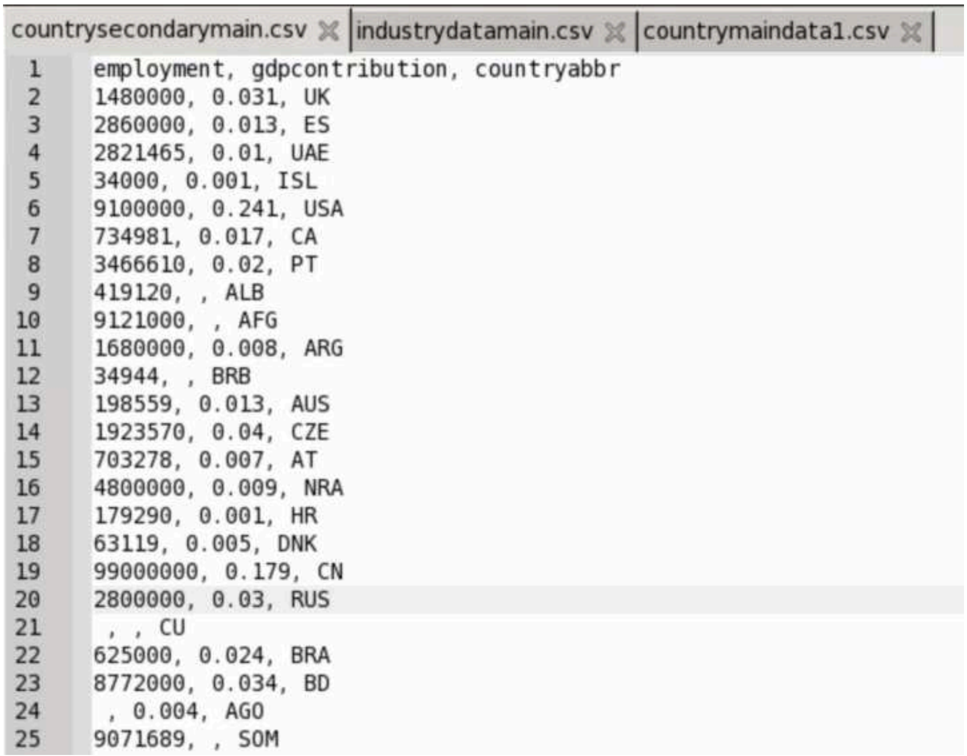
Creation of datasets stored in CSV files:



```
countryID, industryID, countryname, region, population, currency, gdppercapita, countryabbr
12, FI20, United Kingdom, Europe,64550000, Pound Sterling, 54603.00, UK
05, T001, Spain, Europe,47519628, Euro, 45825.00, ES
68, T001, United Arab Emirates, United Arab Emirates,9441129, Dirham, 87729.00, UAE
26, T001, Iceland, Europe,375618, Pound Sterling, 69801.00, ISL
10, TE10, United States, North America,339996563, United States Dollar, 76399.00, USA
09, FI20, Canada, North America,38781391, Canadian Dollar, 58400.00, CA
06, T001, Portugal, Europe,10247605, Euro, 41452.00, PT
08, AG06, Albania, Europe,2832439, Lek, 18552.00, ALB
76, AG06, Afghanistan, Asia,42239854, Afghan Afghani, 368.75, AFG
43, AG06, Argentina, South America,45773884, Argentine Peso, 26505.00, ARG
32, AG06, Barbados, North America,281995, Barbados Dollar, 17837.00, BRB
04, FI20, Australia, Oceania,26439111, Australian Dollar, 62625.00, AUS
23, AG06, Czech Republic, Europe,10495295, Czech Koruna, 49946.00, CZE
28, T001, Austria, Europe,8958960, Euro, 46510.28, AT
09, EN01, Nigeria, Africa,223804632, Naira, 67936.00, NRA
16, AG06, Croatia, Europe,4008617, Euro, 40380.00, HR
17, T001, Denmark, Europe,5910913, Danish Krone, 74005.00, DNK
02, TE10, China, Asia,1425671352, Chinese Yuan, 21476.00, CN
01, EN01, Russia, Europe,144444359, Russian Ruble, 38485.00, RUS
07, AG06, Cuba, North America,11194449, Cuban Peso, 9499.59, CU
03, AG06, Brazil, South America,216422446, Brazilian Real, 17822.00, BRA
19, AG06, Bangladesh, Asia,172954319, Bangaldeshi Taki, 7395.00, BD
108, AG06, Angola, Africa,36684202, Angolan Kwanza, 6974.00, AGO
100, LI01, Somalia, Africa,18143378, Somali Shilling, 446.98, SOM
```

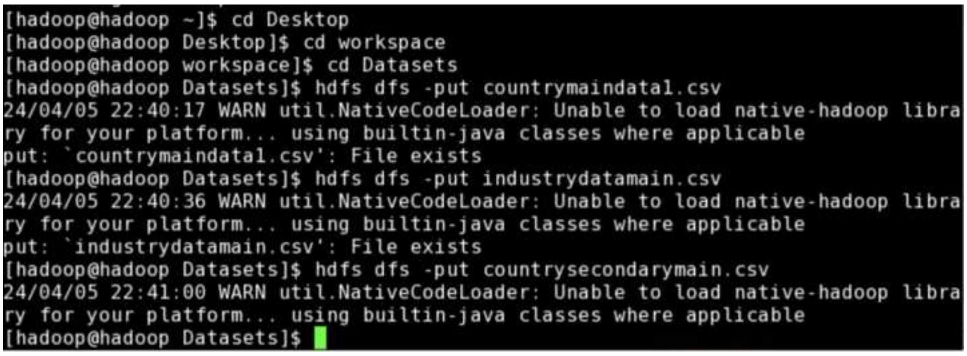


```
industryID, majorindustry, description
FI20, Finance, Facilitating economic activity and managing financial risk
T001, Tourism, A social cultural and economical phenomenon for personal or business/professional purposes
TE10, Technology, Research and development of computing
MA01, Manufacturing, Making goods by hand or machine to service customers
AG06, Agriculture, Production of food or crops for consumption
MI06, Mining, The extraction management and processing of naturally occurring solid minerals
PH01, Pharmaceuticals, The discovery development production and marketing of pharmaceutical drugs
DI20, Diamonds, The mining processing and marketing of gem and industrial diamonds
LI01, Livestock, Breeding animals for a useful commercial-purpose
EN01, Energy, Exploration of renewable and nonrenewable energy sources
```



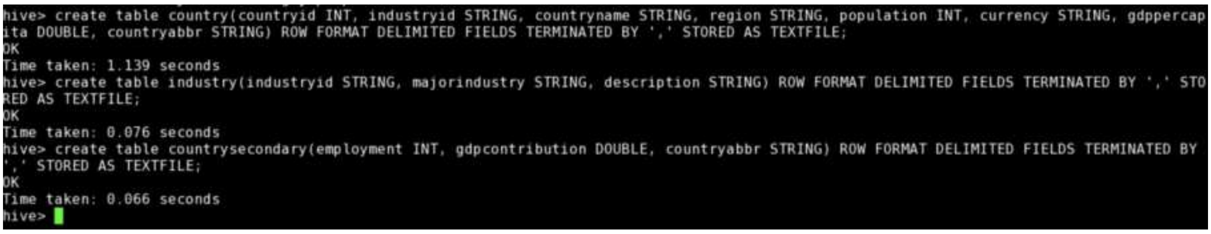
```
employment, gdpcontribution, countryabbr
1480000, 0.031, UK
2860000, 0.013, ES
2821465, 0.01, UAE
34000, 0.001, ISL
9100000, 0.241, USA
734981, 0.017, CA
3466610, 0.02, PT
419120, , ALB
9121000, , AFG
1680000, 0.008, ARG
34944, , BRB
198559, 0.013, AUS
1923570, 0.04, CZE
703278, 0.007, AT
4800000, 0.009, NRA
179290, 0.001, HR
63119, 0.005, DNK
99000000, 0.179, CN
2800000, 0.03, RUS
 , , CU
625000, 0.024, BRA
8772000, 0.034, BD
 , 0.004, AGO
9071689, , SOM
```

Locating the directory that holds the relevant csv files and uploading them to Hadoop:



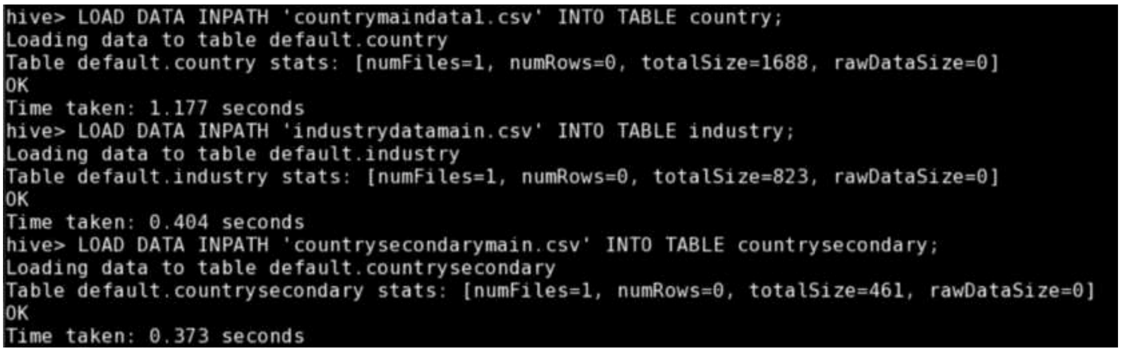
```
[hadoop@hadoop ~]$ cd Desktop
[hadoop@hadoop Desktop]$ cd workspace
[hadoop@hadoop workspace]$ cd Datasets
[hadoop@hadoop Datasets]$ hdfs dfs -put countrymaindata1.csv
24/04/05 22:40:17 WARN util.NativeCodeLoader: Unable to load native-hadoop libra
ry for your platform... using builtin-java classes where applicable
put: `countrymaindata1.csv': File exists
[hadoop@hadoop Datasets]$ hdfs dfs -put industrydatamain.csv
24/04/05 22:40:36 WARN util.NativeCodeLoader: Unable to load native-hadoop libra
ry for your platform... using builtin-java classes where applicable
put: `industrydatamain.csv': File exists
[hadoop@hadoop Datasets]$ hdfs dfs -put countrysecondarymain.csv
24/04/05 22:41:00 WARN util.NativeCodeLoader: Unable to load native-hadoop libra
ry for your platform... using builtin-java classes where applicable
[hadoop@hadoop Datasets]$
```

After this, **Hive** was accessed by input, allowing for the creation of three tables (country, industry, countrysecondary). The fields specified in each table correspond with the first row (headings) of each csv file and the data type of these values have been specified to encourage consistency:



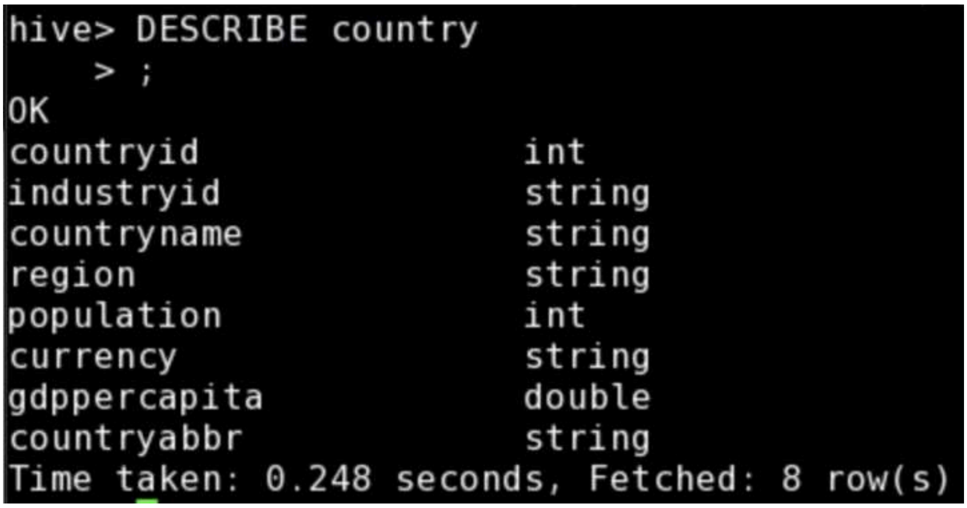
```
hive> create table country(countryid INT, industryid STRING, countryname STRING, region STRING, population INT, currency STRING, gdppercap
ita DOUBLE, countryabbr STRING) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' STORED AS TEXTFILE;
OK
Time taken: 1.139 seconds
hive> create table industry(industryid STRING, majorindustry STRING, description STRING) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' STO
RED AS TEXTFILE;
OK
Time taken: 0.076 seconds
hive> create table countrysecondary(employment INT, gdpcontribution DOUBLE, countryabbr STRING) ROW FORMAT DELIMITED FIELDS TERMINATED BY
',' STORED AS TEXTFILE;
OK
Time taken: 0.066 seconds
hive>
```

As the files have been csv files uploaded onto Hadoop, the commands below load this data into the tables on Hive:

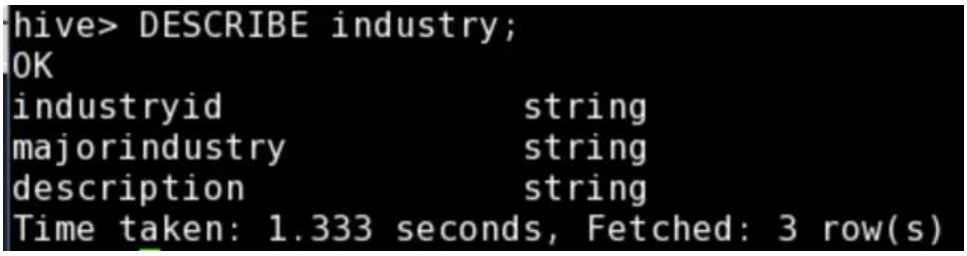


```
hive> LOAD DATA INPATH 'countrymaindata1.csv' INTO TABLE country;
Loading data to table default.country
Table default.country stats: [numFiles=1, numRows=0, totalSize=1688, rawDataSize=0]
OK
Time taken: 1.177 seconds
hive> LOAD DATA INPATH 'industrydatamain.csv' INTO TABLE industry;
Loading data to table default.industry
Table default.industry stats: [numFiles=1, numRows=0, totalSize=823, rawDataSize=0]
OK
Time taken: 0.404 seconds
hive> LOAD DATA INPATH 'countrysecondarymain.csv' INTO TABLE countrysecondary;
Loading data to table default.countrysecondary
Table default.countrysecondary stats: [numFiles=1, numRows=0, totalSize=461, rawDataSize=0]
OK
Time taken: 0.373 seconds
```

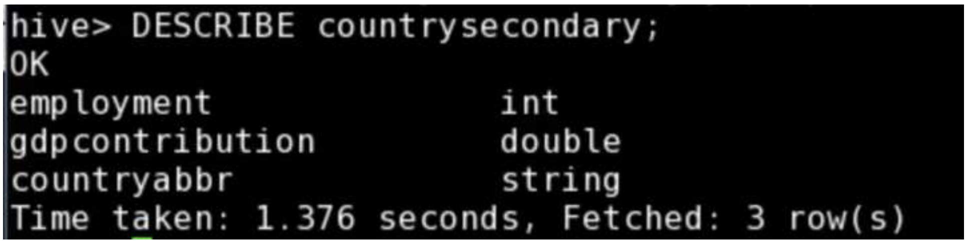
Next, I will DESCRIBE each table ahead of running queries to confirm that the fields have been inputted:



```
hive> DESCRIBE country
    > ;
OK
countryid               int
industryid              string
countryname             string
region                  string
population              int
currency                string
gdppercapita            double
countryabbr             string
Time taken: 0.248 seconds, Fetched: 8 row(s)
```



```
hive> DESCRIBE industry;
OK
industryid              string
majorindustry           string
description             string
Time taken: 1.333 seconds, Fetched: 3 row(s)
```

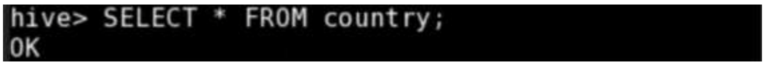


```
hive> DESCRIBE countrysecondary;
OK
employment              int
gdpcontribution         double
countryabbr             string
Time taken: 1.376 seconds, Fetched: 3 row(s)
```

As the csv’s have been loaded into their respective tables, the implementation of data has been completed in preparation of query commands which will be detailed below alongside the results.

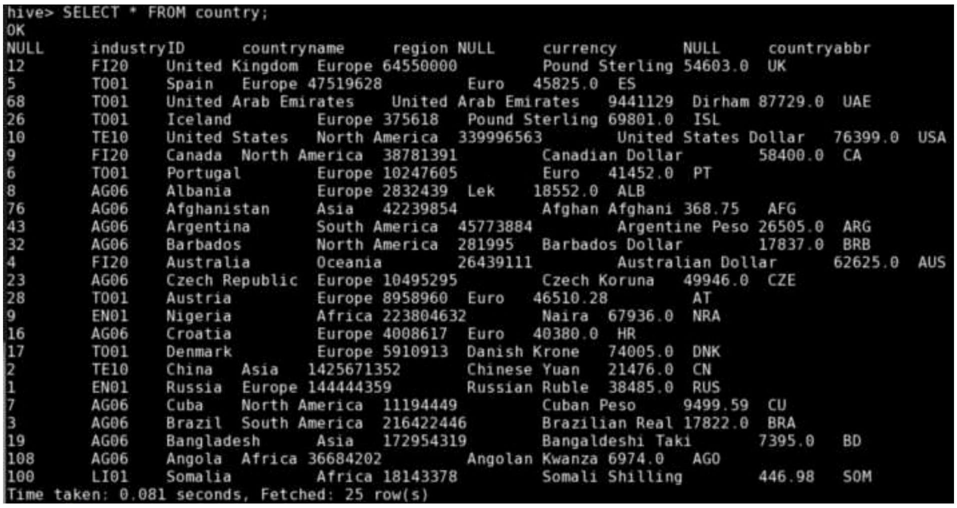
The first three queries are essential for data exploration in understanding the content of the table and the detailed structure. *Also, note that the column names of each table have been considered as a row.*

Query 1 - This query selects all the values within the table ‘country’:



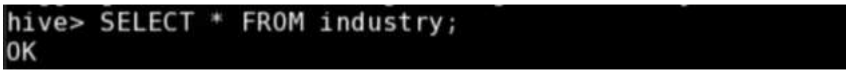
```
hive> SELECT * FROM country;
OK
```

Results of Query 1:



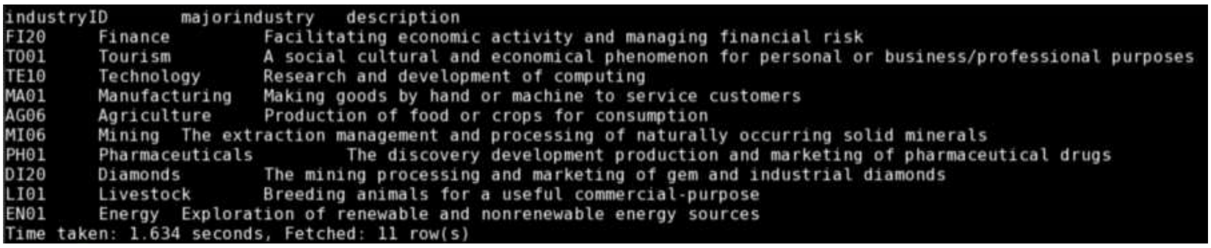
```
hive> SELECT * FROM country;
OK
NULL    industryID      countryname     region NULL     currency        NULL    countryabbr
12      FI20    United Kingdom  Europe 64550000         Pound Sterling 54603.0  UK
5       T001    Spain   Europe 47519628         Euro    45825.0  ES
68      T001    United Arab Emirates    United Arab Emirates    9441129  Dirham 87729.0  UAE
26      T001    Iceland         Europe 375618   Pound Sterling 69801.0  ISL
10      TE10    United States   North America  339996563        United States Dollar    76399.0  USA
9       FI20    Canada  North America  38781391         Canadian Dollar         58400.0  CA
6       T001    Portugal        Europe 10247605         Euro    41452.0  PT
8       AG06    Albania         Europe 2832439  Lek     18552.0  ALB
76      AG06    Afghanistan     Asia   42239854         Afghan Afghani 368.75   AFG
43      AG06    Argentina       South America  45773884         Argentine Peso 26505.0  ARG
32      AG06    Barbados        North America  281995   Barbados Dollar         17837.0  BRB
4       FI20    Australia       Oceania        26439111         Australian Dollar       62625.0  AUS
23      AG06    Czech Republic  Europe 10495295         Czech Koruna    49946.0  CZE
28      T001    Austria         Europe 8958960  Euro    46510.28        AT
9       EN01    Nigeria         Africa 223804632        Naira  67936.0  NRA
16      AG06    Croatia         Europe 4008617  Euro    40380.0  HR
17      T001    Denmark         Europe 5910913  Danish Krone    74005.0  DNK
2       TE10    China   Asia   1425671352       Chinese Yuan    21476.0  CN
1       EN01    Russia  Europe 144444359        Russian Ruble  38485.0  RUS
7       AG06    Cuba    North America  11194449         Cuban Peso      9499.59  CU
3       AG06    Brazil  South America  216422446        Brazilian Real 17822.0  BRA
19      AG06    Bangladesh      Asia   172954319        Bangaldeshi Taki        7395.0   BD
108     AG06    Angola  Africa 36684202         Angolan Kwanza 6974.0   AGO
100     LI01    Somalia         Africa 18143378         Somali Shilling         446.98   SOM
Time taken: 0.081 seconds, Fetched: 25 row(s)
```

Query 2 - This query selects all the values within the table ‘industry’:



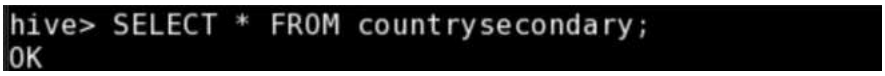
```
hive> SELECT * FROM industry;
OK
```

Results of Query 2:



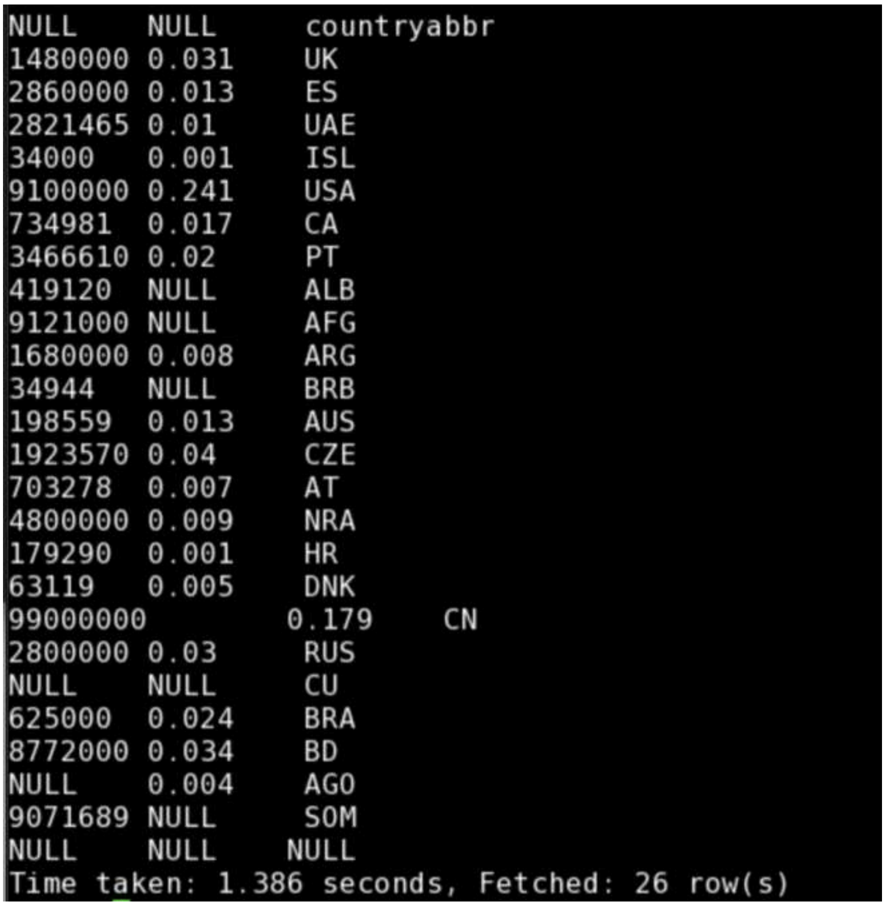
```
industryID       majorindustry   description
FI20     Finance         Facilitating economic activity and managing financial risk
T001     Tourism         A social cultural and economical phenomenon for personal or business/professional purposes
TE10     Technology      Research and development of computing
MA01     Manufacturing   Making goods by hand or machine to service customers
AG06     Agriculture     Production of food or crops for consumption
MI06     Mining  The extraction management and processing of naturally occurring solid minerals
PH01     Pharmaceuticals         The discovery development production and marketing of pharmaceutical drugs
DI20     Diamonds        The mining processing and marketing of gem and industrial diamonds
LI01     Livestock       Breeding animals for a useful commercial-purpose
EN01     Energy  Exploration of renewable and nonrenewable energy sources
Time taken: 1.634 seconds, Fetched: 11 row(s)
```

Query 3 - This query selects all the values within the table ‘countrysecondary:



```
hive> SELECT * FROM countrysecondary;
OK
```

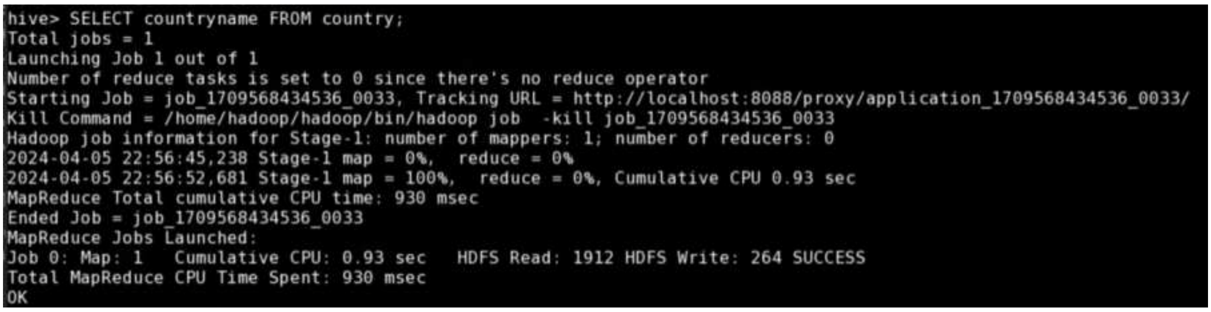
Results of Query 3:



```
NULL    NULL     countryabbr
1480000 0.031    UK
2860000 0.013    ES
2821465 0.01     UAE
34000   0.001    ISL
9100000 0.241    USA
734981  0.017    CA
3466610 0.02     PT
419120  NULL     ALB
9121000 NULL     AFG
1680000 0.008    ARG
34944   NULL     BRB
198559  0.013    AUS
1923570 0.04     CZE
703278  0.007    AT
4800000 0.009    NRA
179290  0.001    HR
63119   0.005    DNK
99000000        0.179    CN
2800000 0.03     RUS
NULL    NULL     CU
625000  0.024    BRA
8772000 0.034    BD
NULL    0.004    AGO
9071689 NULL     SOM
NULL    NULL    NULL
Time taken: 1.386 seconds, Fetched: 26 row(s)
```

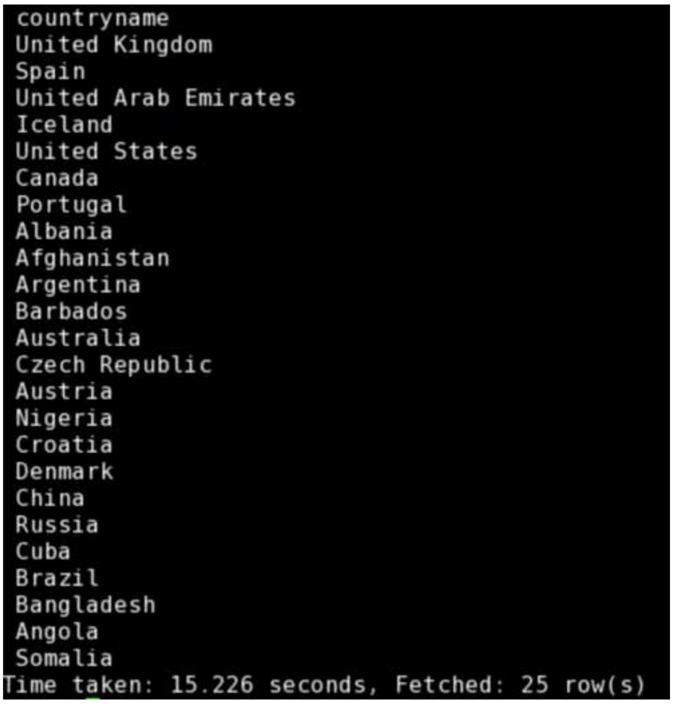
From the total fetched rows across 3 tables, there are over 50 rows which suggests the collection meet the criteria.

Query 4 – This query selects all values of the ‘countryname’ within the table ‘country’.



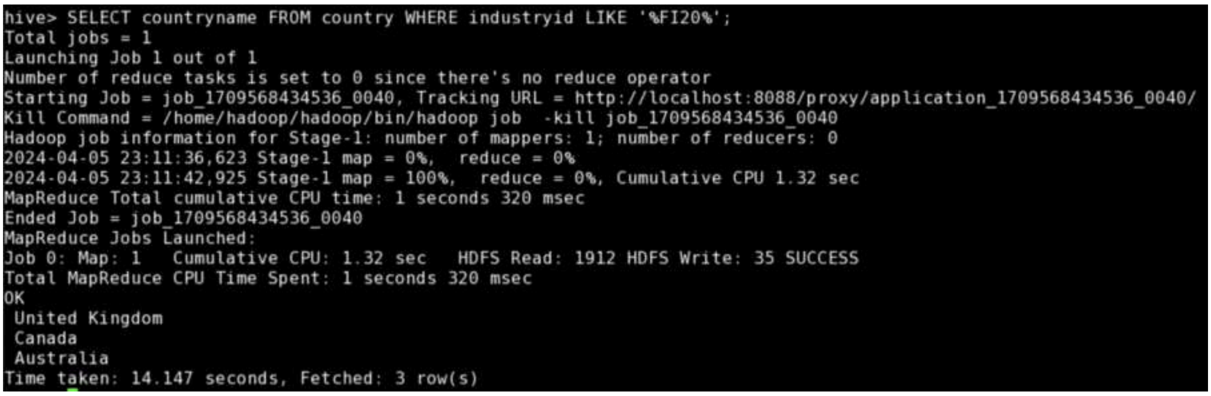
```
hive> SELECT countryname FROM country;
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks is set to 0 since there's no reduce operator
Starting Job = job_1709568434536_0033, Tracking URL = http://localhost:8088/proxy/application_1709568434536_0033/
Kill Command = /home/hadoop/hadoop/bin/hadoop job  -kill job_1709568434536_0033
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 0
2024-04-05 22:56:45,238 Stage-1 map = 0%,  reduce = 0%
2024-04-05 22:56:52,681 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 0.93 sec
MapReduce Total cumulative CPU time: 930 msec
Ended Job = job_1709568434536_0033
MapReduce Jobs Launched:
Job 0: Map: 1   Cumulative CPU: 0.93 sec   HDFS Read: 1912 HDFS Write: 264 SUCCESS
Total MapReduce CPU Time Spent: 930 msec
OK
```

Results of Query 4:



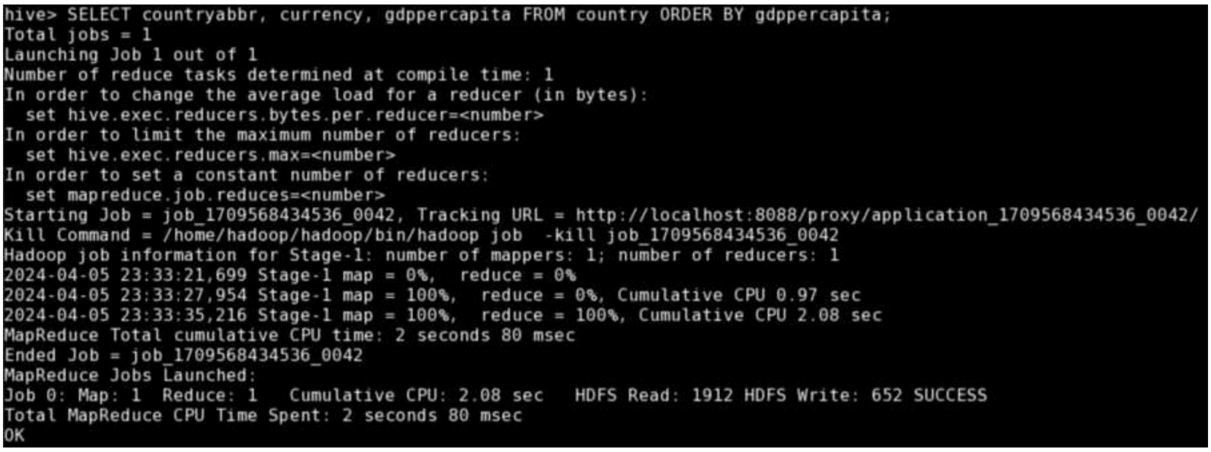
```
 countryname
 United Kingdom
 Spain
 United Arab Emirates
 Iceland
 United States
 Canada
 Portugal
 Albania
 Afghanistan
 Argentina
 Barbados
 Australia
 Czech Republic
 Austria
 Nigeria
 Croatia
 Denmark
 China
 Russia
 Cuba
 Brazil
 Bangladesh
 Angola
 Somalia
Time taken: 15.226 seconds, Fetched: 25 row(s)
```

Query 5 *including results* – This query selects values of the column ‘countryname’ from the ‘country’ table under the condition that they have an ‘industryid’ of ‘FI20’ (which is linked with countries that consider Finance as their major industry). The results of this are United Kingdom, Canada, and Australia, as seen below:



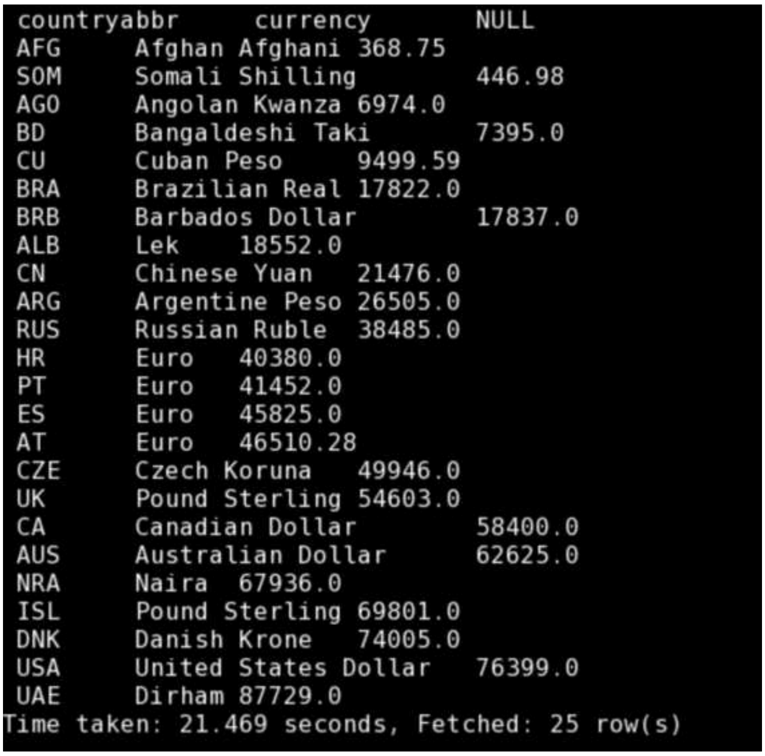
```
hive> SELECT countryname FROM country WHERE industryid LIKE '%FI20%';
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks is set to 0 since there's no reduce operator
Starting Job = job_1709568434536_0040, Tracking URL = http://localhost:8088/proxy/application_1709568434536_0040/
Kill Command = /home/hadoop/hadoop/bin/hadoop job  -kill job_1709568434536_0040
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 0
2024-04-05 23:11:36,623 Stage-1 map = 0%,  reduce = 0%
2024-04-05 23:11:42,925 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 1.32 sec
MapReduce Total cumulative CPU time: 1 seconds 320 msec
Ended Job = job_1709568434536_0040
MapReduce Jobs Launched:
Job 0: Map: 1   Cumulative CPU: 1.32 sec   HDFS Read: 1912 HDFS Write: 35 SUCCESS
Total MapReduce CPU Time Spent: 1 seconds 320 msec
OK
 United Kingdom
 Canada
 Australia
Time taken: 14.147 seconds, Fetched: 3 row(s)
```

Query 6 – This query selects values of the columns ‘countryabbr’, ‘currency’ and ‘gdppercapita’ from the table ‘country’ and the results will be ranked in ascending order from the lowest GDP per capita to the highest:



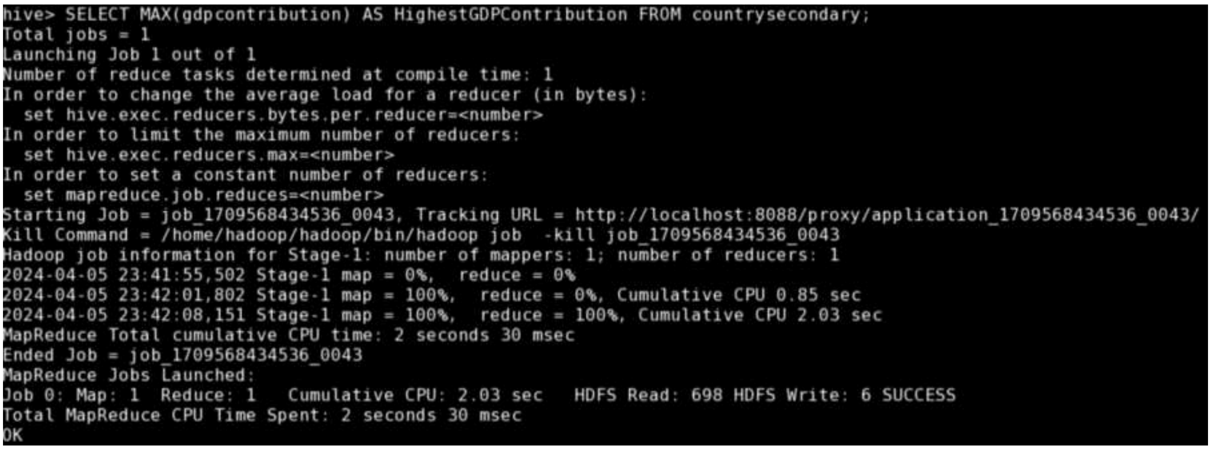
```
hive> SELECT countryabbr, currency, gdppercapita FROM country ORDER BY gdppercapita;
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order 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_1709568434536_0042, Tracking URL = http://localhost:8088/proxy/application_1709568434536_0042/
Kill Command = /home/hadoop/hadoop/bin/hadoop job  -kill job_1709568434536_0042
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2024-04-05 23:33:21,699 Stage-1 map = 0%,  reduce = 0%
2024-04-05 23:33:27,954 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 0.97 sec
2024-04-05 23:33:35,216 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 2.08 sec
MapReduce Total cumulative CPU time: 2 seconds 80 msec
Ended Job = job_1709568434536_0042
MapReduce Jobs Launched:
Job 0: Map: 1  Reduce: 1   Cumulative CPU: 2.08 sec   HDFS Read: 1912 HDFS Write: 652 SUCCESS
Total MapReduce CPU Time Spent: 2 seconds 80 msec
OK
```

Results of Query 6 – this could be used to ***infer*** countries standard of living from the lowest to the highest:



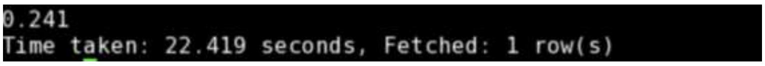
```
 countryabbr     currency       NULL
 AFG     Afghan Afghani 368.75
 SOM     Somali Shilling        446.98
 AGO     Angolan Kwanza 6974.0
 BD      Bangaldeshi Taki       7395.0
 CU      Cuban Peso     9499.59
 BRA     Brazilian Real 17822.0
 BRB     Barbados Dollar        17837.0
 ALB     Lek    18552.0
 CN      Chinese Yuan   21476.0
 ARG     Argentine Peso 26505.0
 RUS     Russian Ruble  38485.0
 HR      Euro   40380.0
 PT      Euro   41452.0
 ES      Euro   45825.0
 AT      Euro   46510.28
 CZE     Czech Koruna   49946.0
 UK      Pound Sterling 54603.0
 CA      Canadian Dollar        58400.0
 AUS     Australian Dollar      62625.0
 NRA     Naira  67936.0
 ISL     Pound Sterling 69801.0
 DNK     Danish Krone   74005.0
 USA     United States Dollar   76399.0
 UAE     Dirham 87729.0
Time taken: 21.469 seconds, Fetched: 25 row(s)
```

Query 7 – This query selects the maximum value of GDP contribution from the table countrysecondary and highlights ‘HighestGDPContribution’ as label which is useful for finding the highest GDP of an industry to an economy within a country:



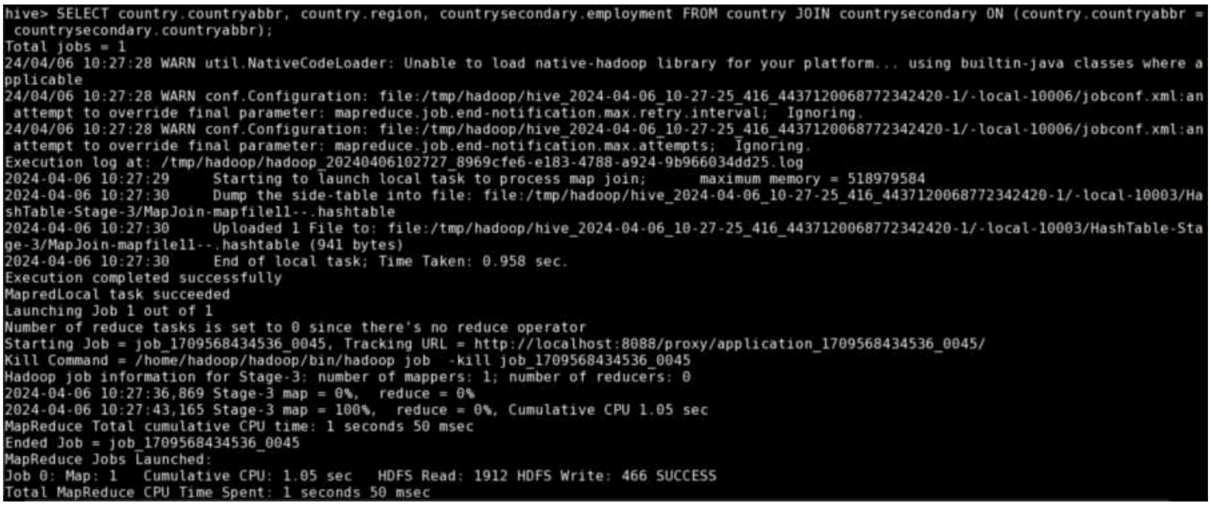
```
hive> SELECT MAX(gdpcontribution) AS HighestGDPContribution FROM countrysecondary;
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order 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_1709568434536_0043, Tracking URL = http://localhost:8088/proxy/application_1709568434536_0043/
Kill Command = /home/hadoop/hadoop/bin/hadoop job  -kill job_1709568434536_0043
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2024-04-05 23:41:55,502 Stage-1 map = 0%,  reduce = 0%
2024-04-05 23:42:01,802 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 0.85 sec
2024-04-05 23:42:08,151 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 2.03 sec
MapReduce Total cumulative CPU time: 2 seconds 30 msec
Ended Job = job_1709568434536_0043
MapReduce Jobs Launched:
Job 0: Map: 1  Reduce: 1   Cumulative CPU: 2.03 sec   HDFS Read: 698 HDFS Write: 6 SUCCESS
Total MapReduce CPU Time Spent: 2 seconds 30 msec
OK
```

Results of Query 7 – Equivalent to 24.1%:



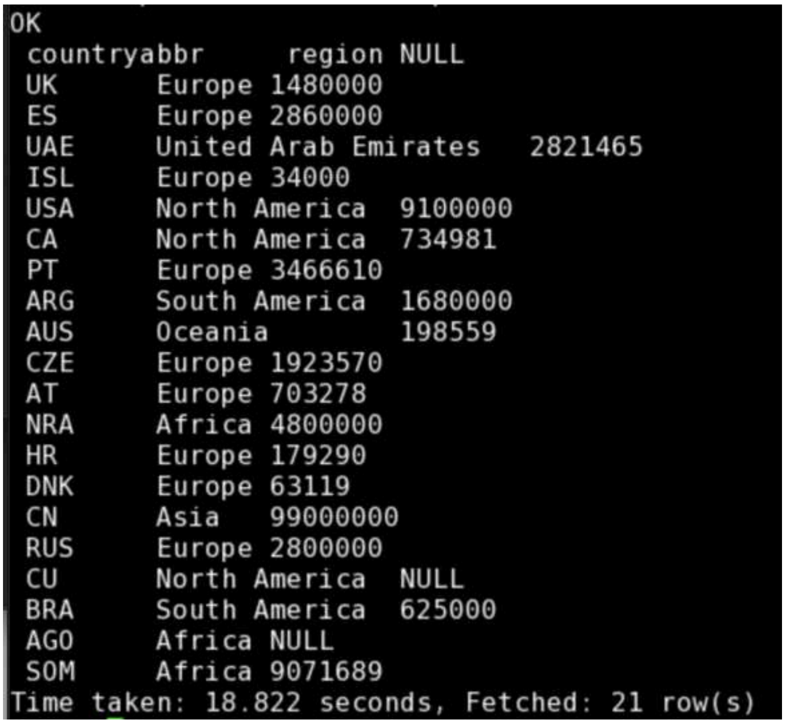
```
0.241
Time taken: 22.419 seconds, Fetched: 1 row(s)
```

Query 8 – The following query selects values from multiple tables and joins these values together to create a new table display. In this case, country abbreviations and region (continent) from the ‘country’ table have been selected as well as the employment column from the countrysecondary table to join them together:



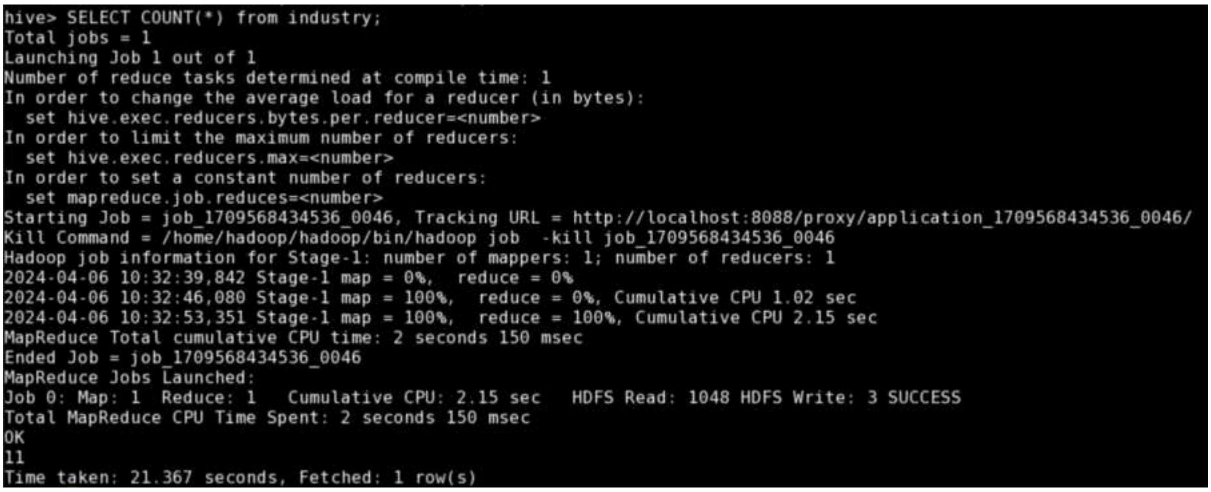
```
hive> SELECT country.countryabbr, country.region, countrysecondary.employment FROM country JOIN countrysecondary ON (country.countryabbr =
 countrysecondary.countryabbr);
Total jobs = 1
24/04/06 10:27:28 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where a
pplicable
24/04/06 10:27:28 WARN conf.Configuration: file:/tmp/hadoop/hive_2024-04-06_10-27-25_416_4437120068772342420-1/-local-10006/jobconf.xml:an
 attempt to override final parameter: mapreduce.job.end-notification.max.retry.interval;  Ignoring.
24/04/06 10:27:28 WARN conf.Configuration: file:/tmp/hadoop/hive_2024-04-06_10-27-25_416_4437120068772342420-1/-local-10006/jobconf.xml:an
 attempt to override final parameter: mapreduce.job.end-notification.max.attempts;  Ignoring.
Execution log at: /tmp/hadoop/hadoop_20240406102727_8969cfe6-e183-4788-a924-9b966034dd25.log
2024-04-06 10:27:29     Starting to launch local task to process map join;      maximum memory = 518979584
2024-04-06 10:27:30     Dump the side-table into file: file:/tmp/hadoop/hive_2024-04-06_10-27-25_416_4437120068772342420-1/-local-10003/Ha
shTable-Stage-3/MapJoin-mapfile11--.hashtable
2024-04-06 10:27:30     Uploaded 1 File to: file:/tmp/hadoop/hive_2024-04-06_10-27-25_416_4437120068772342420-1/-local-10003/HashTable-Sta
ge-3/MapJoin-mapfile11--.hashtable (941 bytes)
2024-04-06 10:27:30     End of local task; Time Taken: 0.958 sec.
Execution completed successfully
MapredLocal task succeeded
Launching Job 1 out of 1
Number of reduce tasks is set to 0 since there's no reduce operator
Starting Job = job_1709568434536_0045, Tracking URL = http://localhost:8088/proxy/application_1709568434536_0045/
Kill Command = /home/hadoop/hadoop/bin/hadoop job  -kill job_1709568434536_0045
Hadoop job information for Stage-3: number of mappers: 1; number of reducers: 0
2024-04-06 10:27:36,869 Stage-3 map = 0%,  reduce = 0%
2024-04-06 10:27:43,165 Stage-3 map = 100%,  reduce = 0%, Cumulative CPU 1.05 sec
MapReduce Total cumulative CPU time: 1 seconds 50 msec
Ended Job = job_1709568434536_0045
MapReduce Jobs Launched:
Job 0: Map: 1   Cumulative CPU: 1.05 sec   HDFS Read: 1912 HDFS Write: 466 SUCCESS
Total MapReduce CPU Time Spent: 1 seconds 50 msec
```

Results of Query 8 – the is useful for detailing the number of people in employment based the country by abbreviation and the region:



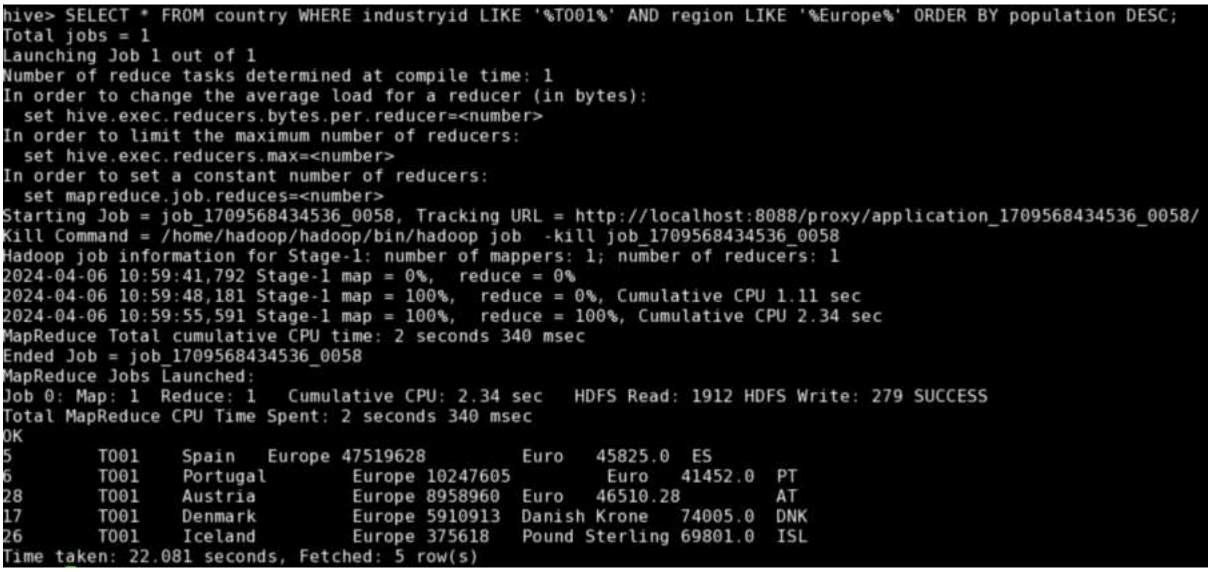
```
OK
 countryabbr     region NULL
 UK      Europe 1480000
 ES      Europe 2860000
 UAE     United Arab Emirates   2821465
 ISL     Europe 34000
 USA     North America  9100000
 CA      North America  734981
 PT      Europe 3466610
 ARG     South America  1680000
 AUS     Oceania        198559
 CZE     Europe 1923570
 AT      Europe 703278
 NRA     Africa 4800000
 HR      Europe 179290
 DNK     Europe 63119
 CN      Asia   99000000
 RUS     Europe 2800000
 CU      North America  NULL
 BRA     South America  625000
 AGO     Africa NULL
 SOM     Africa 9071689
Time taken: 18.822 seconds, Fetched: 21 row(s)
```

Query 9 *including results* – this query counts the number of rows within the industry table which can be useful for understanding how many types of industries exists relative to the dataset:



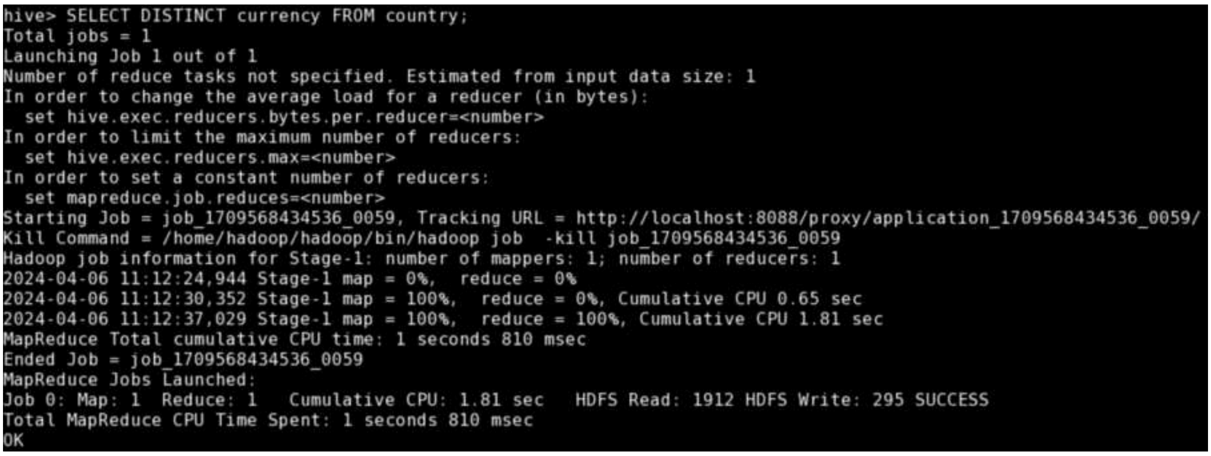
```
hive> SELECT COUNT(*) from industry;
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1709568434536_0046, Tracking URL = http://localhost:8088/proxy/application_1709568434536_0046/
Kill Command = /home/hadoop/hadoop/bin/hadoop job  -kill job_1709568434536_0046
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2024-04-06 10:32:39,842 Stage-1 map = 0%,  reduce = 0%
2024-04-06 10:32:46,080 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 1.02 sec
2024-04-06 10:32:53,351 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 2.15 sec
MapReduce Total cumulative CPU time: 2 seconds 150 msec
Ended Job = job_1709568434536_0046
MapReduce Jobs Launched:
Job 0: Map: 1  Reduce: 1   Cumulative CPU: 2.15 sec   HDFS Read: 1048 HDFS Write: 3 SUCCESS
Total MapReduce CPU Time Spent: 2 seconds 150 msec
OK
11
Time taken: 21.367 seconds, Fetched: 1 row(s)
```

Query 10 *including results* – this query selects countries and ranks them from highest to lowest based on their population, ‘industryid’ being ‘TO01’ (Tourism) and their region being in Europe. This is useful to identify European countries that make majority of their income from Tourism and as this is in descending order by region, it suggests the countries that people likely to visit within Europe (in order):



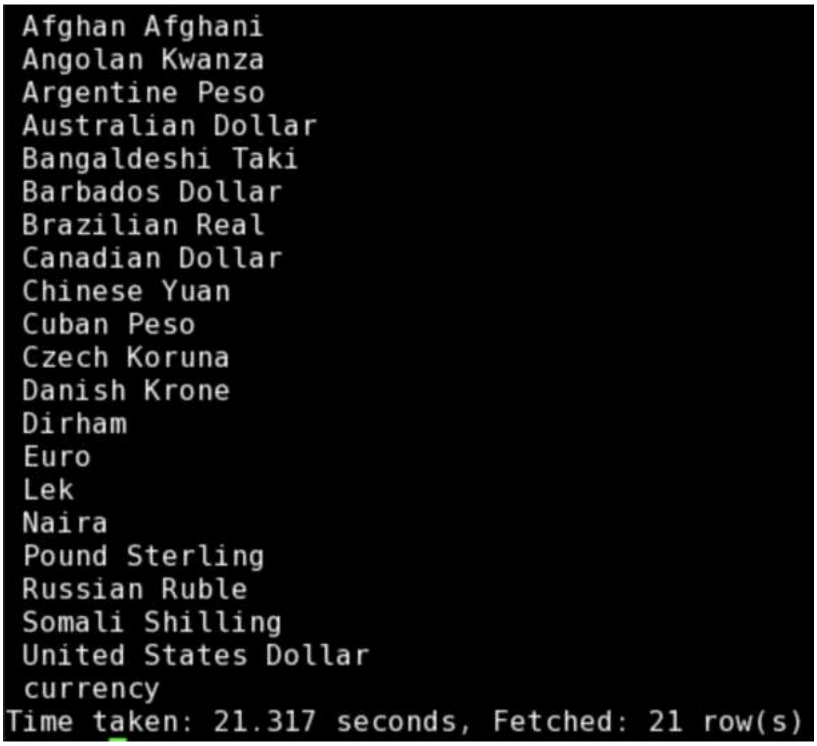
```
hive> SELECT * FROM country WHERE industryid LIKE '%T001%' AND region LIKE '%Europe%' ORDER BY population DESC;
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<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_1709568434536_0058, Tracking URL = http://localhost:8088/proxy/application_1709568434536_0058/
Kill Command = /home/hadoop/hadoop/bin/hadoop job  -kill job_1709568434536_0058
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2024-04-06 10:59:41,792 Stage-1 map = 0%,  reduce = 0%
2024-04-06 10:59:48,181 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 1.11 sec
2024-04-06 10:59:55,591 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 2.34 sec
MapReduce Total cumulative CPU time: 2 seconds 340 msec
Ended Job = job_1709568434536_0058
MapReduce Jobs Launched:
Job 0: Map: 1  Reduce: 1   Cumulative CPU: 2.34 sec   HDFS Read: 1912 HDFS Write: 279 SUCCESS
Total MapReduce CPU Time Spent: 2 seconds 340 msec
OK
5       T001    Spain   Europe 47519628         Euro   45825.0  ES
6       T001    Portugal        Europe 10247605         Euro   41452.0  PT
28      T001    Austria         Europe 8958960  Euro   46510.28         AT
17      T001    Denmark         Europe 5910913  Danish Krone   74005.0  DNK
26      T001    Iceland         Europe 375618   Pound Sterling 69801.0  ISL
Time taken: 22.081 seconds, Fetched: 5 row(s)
```

Query 11 – This query selects the unique currencies from the table ‘country’ which can be useful for identifying the unique currencies that exists within the data and to avoiding the display of duplicates:



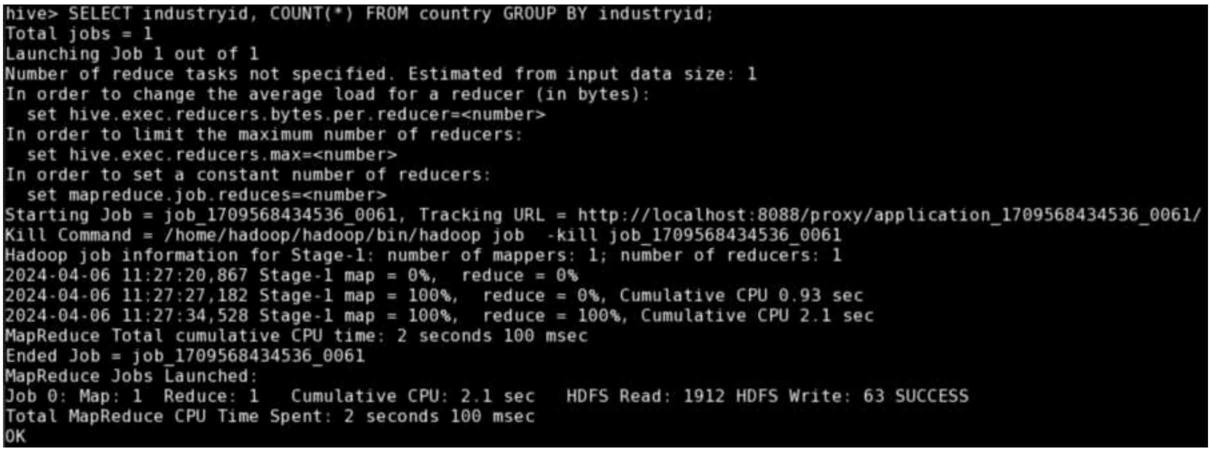
```
hive> SELECT DISTINCT currency FROM country;
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum 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_1709568434536_0059, Tracking URL = http://localhost:8088/proxy/application_1709568434536_0059/
Kill Command = /home/hadoop/hadoop/bin/hadoop job  -kill job_1709568434536_0059
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2024-04-06 11:12:24,944 Stage-1 map = 0%,  reduce = 0%
2024-04-06 11:12:30,352 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 0.65 sec
2024-04-06 11:12:37,029 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 1.81 sec
MapReduce Total cumulative CPU time: 1 seconds 810 msec
Ended Job = job_1709568434536_0059
MapReduce Jobs Launched:
Job 0: Map: 1  Reduce: 1   Cumulative CPU: 1.81 sec   HDFS Read: 1912 HDFS Write: 295 SUCCESS
Total MapReduce CPU Time Spent: 1 seconds 810 msec
OK
```

Results of Query 11:



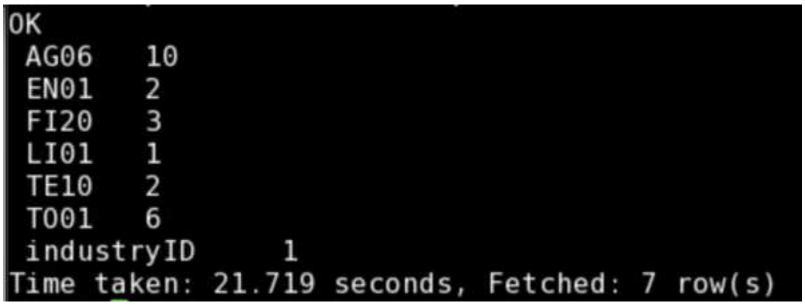
```
 Afghan Afghani
 Angolan Kwanza
 Argentine Peso
 Australian Dollar
 Bangaldeshi Taki
 Barbados Dollar
 Brazilian Real
 Canadian Dollar
 Chinese Yuan
 Cuban Peso
 Czech Koruna
 Danish Krone
 Dirham
 Euro
 Lek
 Naira
 Pound Sterling
 Russian Ruble
 Somali Shilling
 United States Dollar
 currency
Time taken: 21.317 seconds, Fetched: 21 row(s)
```

Query 12 - this query groups and counts ‘industryid’ from the ‘country’ table which can be useful for identifying the most popular industries across countries within the data:



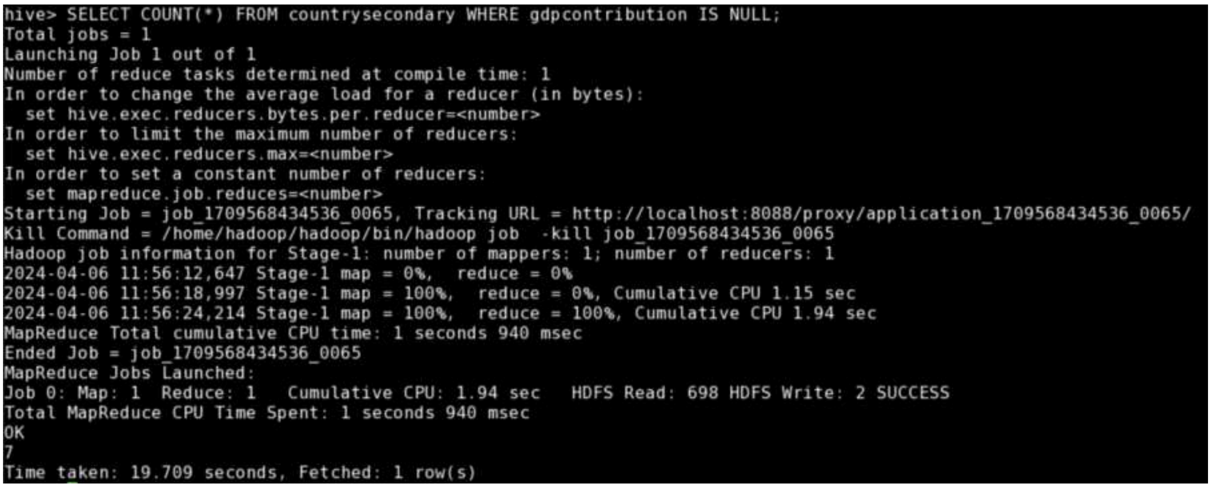
```
hive> SELECT industryid, COUNT(*) FROM country GROUP BY industryid;
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In 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_1709568434536_0061, Tracking URL = http://localhost:8088/proxy/application_1709568434536_0061/
Kill Command = /home/hadoop/hadoop/bin/hadoop job  -kill job_1709568434536_0061
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2024-04-06 11:27:20,867 Stage-1 map = 0%,  reduce = 0%
2024-04-06 11:27:27,182 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 0.93 sec
2024-04-06 11:27:34,528 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 2.1 sec
MapReduce Total cumulative CPU time: 2 seconds 100 msec
Ended Job = job_1709568434536_0061
MapReduce Jobs Launched:
Job 0: Map: 1  Reduce: 1   Cumulative CPU: 2.1 sec   HDFS Read: 1912 HDFS Write: 63 SUCCESS
Total MapReduce CPU Time Spent: 2 seconds 100 msec
OK
```

Results of Query 12:



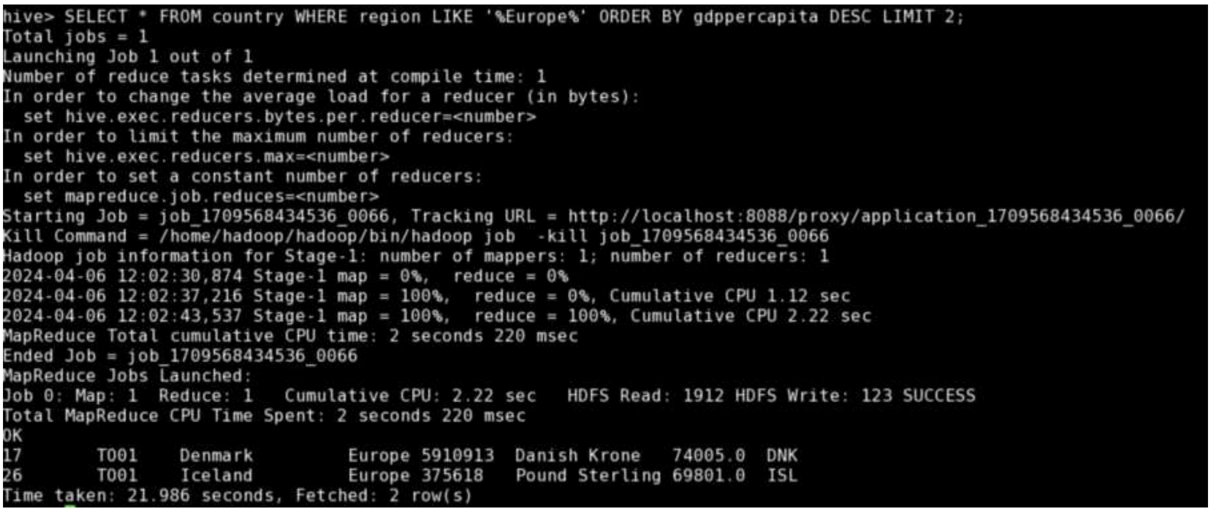
```
OK
 AG06   10
 EN01   2
 FI20   3
 LI01   1
 TE10   2
 TO01   6
 industryID     1
Time taken: 21.719 seconds, Fetched: 7 row(s)
```

Query 13 *including results* – This query displays a count for the number of null values that exists for gdpcontribution from the 'countrysecondary' table which suggests that the data includes 7 countries that are still early in their development stage or have a limited economic power:



```
hive> SELECT COUNT(*) FROM countrysecondary WHERE gdpcontribution IS NULL;
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 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_1709568434536_0065, Tracking URL = http://localhost:8088/proxy/application_1709568434536_0065/
Kill Command = /home/hadoop/hadoop/bin/hadoop job  -kill job_1709568434536_0065
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2024-04-06 11:56:12,647 Stage-1 map = 0%,  reduce = 0%
2024-04-06 11:56:18,997 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 1.15 sec
2024-04-06 11:56:24,214 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 1.94 sec
MapReduce Total cumulative CPU time: 1 seconds 940 msec
Ended Job = job_1709568434536_0065
MapReduce Jobs Launched:
Job 0: Map: 1  Reduce: 1   Cumulative CPU: 1.94 sec   HDFS Read: 698 HDFS Write: 2 SUCCESS
Total MapReduce CPU Time Spent: 1 seconds 940 msec
OK
7
Time taken: 19.709 seconds, Fetched: 1 row(s)
```

Query 14 *including results* – this query selects and ranks the rows of European countries from lowest to highest based on their gdppercapita and limits the output to 2 which is effective for finding the countries with the two lowest GDP per capita in Europe:



```
hive> SELECT * FROM country WHERE region LIKE '%Europe%' ORDER BY gdppercapita DESC LIMIT 2;
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 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_1709568434536_0066, Tracking URL = http://localhost:8088/proxy/application_1709568434536_0066/
Kill Command = /home/hadoop/hadoop/bin/hadoop job  -kill job_1709568434536_0066
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2024-04-06 12:02:30,874 Stage-1 map = 0%,  reduce = 0%
2024-04-06 12:02:37,216 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 1.12 sec
2024-04-06 12:02:43,537 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 2.22 sec
MapReduce Total cumulative CPU time: 2 seconds 220 msec
Ended Job = job_1709568434536_0066
MapReduce Jobs Launched:
Job 0: Map: 1  Reduce: 1   Cumulative CPU: 2.22 sec   HDFS Read: 1912 HDFS Write: 123 SUCCESS
Total MapReduce CPU Time Spent: 2 seconds 220 msec
OK
17      TO01    Denmark         Europe 5910913  Danish Krone   74005.0  DNK
26      TO01    Iceland         Europe 375618   Pound Sterling 69801.0  ISL
Time taken: 21.986 seconds, Fetched: 2 row(s)
```

Task B – Big Data Project Analysis

Task B.1

The attributes of data lakes align with the requirements of the ABC Investment Bank Ltd to achieve an application of large storage, high availability, scalability, and accessibility in addition to confidentiality. The purpose of building this solution will be provided in detail including the implementation approach, to provide insights into the concept and potential of a data lake that make it appropriate for the firm's needs.

'A data lake is defined by a large-scale storage system architecture for storing various raw data of the enterprise and retaining the content. Data in the data lake can be accessed, processed, analysed, and transmitted, and massive data analysis and processing is also supported' (Zhang et al., 2023). In addition, data lakes can be considered flexible as they store data of all forms (structured, semi-structured and unstructured) which caters to the range of example data provided by ABC and as the range of stored data can remain in its original state, it allows for a greater flow of data through the system for analysis at a later stage. The flexibility of data lakes also suggests that ABC can utilise horizontal and traditional scaling to increase storage capacity as they expect a large volume of data (approximately 200 petabytes).

'Horizontal scaling obtains more computing power by adding more virtual machines, and traditional scaling obtains more computing power by adding resources in the virtual machine' (Liu et al., 2014). The process of driving capacity has gradually moved from traditional to horizontal means as the potential for companies to reach desired capacity is limited by the ability of singular hardware and costs substantial amounts to maintain. On the other hand, *scaling out* allows organisations to spread workload across multiple machines which facilitates the processing of large data volumes and encourages fault tolerance as the system can continuously run despite a few machine failures because of the network being synchronised. Collectively, both forms of scaling are still utilised for optimal results but nowadays, horizontal tends to be prioritised as it is more cost effective and enables high availability for increased performance.

'A data warehouse is a database optimised to analyse relational data coming from transactional systems and business applications' (Huang, 2024). As ABC has an internal enterprise system, this indicates they are storing historical company data which can be easily filtered through as it is stored on a relational database. Languages such as SQL can be used to query data which can facilitate the extraction of data. This can be effective collectively with other data forms to identify trends that allow for the identification of trends and insights that can influence trading decisions. Though data warehousing can process large volumes of data, this is tailored to structured data and requires more processing power to cater to other data types and as the firm plan to utilise all forms of data, data lakes would prove more efficient in processing a diverse range of large volume data and offers flexibility as it isn't confined to the schemas of structured data and can equally cater to processing semi-structured and unstructured data with ease is ideal for identifying 'hidden' data patterns.

There are four steps to the implementation of the data lake that will prove beneficial to ABC's system requirements: landing zone, data catalog, data ingestion and data archiving. 'A landing zone in building a data lake is the central place where data will land. It contains all the raw data from all different source systems available' (Huang, 2024). This adds to the flexibility and scalability of the system as it handles various forms of data and can store large volumes. The use of cloud storage solutions such as 'Amazon S3' align with horizontal scaling which is encouraged throughout the construction of the system to ensure consistency and relative ease in data management. Additionally, the bank plans to maintain confidentiality of trading decisions and therefore, this should be considered throughout the process of handling data; cloud services allow for the enforcement of policies to enable administrators to assign designated users which is beneficial for strengthening the security of data resources.

The next stage consists of 'building a data catalog which tracks all the newly loaded data and versions that were created by data transformation, data processing, and analytics to provide a query-able interface of all data stored in the data lake' (Huang, 2024). The data catalog acts as a centralised library of metadata to provide users with necessary accessibility to develop a deeper comprehension of data for assessment. This is advantageous coupled with search functionalities as it allows users to efficiently filter through data to derive insights suitable for task requirements and can encourage collaboration as users within an organisation tend to have shared access to material that can build transparency across the bank's departments and offices worldwide. Azure Data Catalog is an example of a cloud data calalog that can be implemented into the system to achieve effective transformation, processing and analytics as well as maintain consistency.

'Data ingestion is the process of collecting data from various sources and moving it to your data warehouse or lake for processing and analysis' (Monte Carlo, 2023). This is also a crucial element of the process the company specifies the need to collate data from social media and online news feeds which can be considered customer sentiment and tends to be delivered in real time. Simultaneously, ABC also makes use of market and corporate data which collectively totals to large volumes which require batch processing. As a result, this stage will utilise a cloud provider such as Azure as this has built in functions that cater to batch and streaming data and can streamline the process of handling data of diverse types to reduce complexity. Additionally, the bank's internal enterprise system is built on relational databases (including MS SQL Server which is also a Microsoft service) so, the use of a cloud-based data ingestion provider will automate the transfer of data into the data lake which in turn, limits the expenditure of IT team resources.

'Lastly, data archiving is a method of archiving enterprise data that is rarely used, to improve efficiency. This process classifies data as hot, warm, or cold; hot data is used often, warm data is used from time to time; and cold data is rarely used' (Huang, 2024). The benefit of data archiving is optimised processing time as the system does not waste time on processing cold data as this is irrelevant and this data can still be accessed as it has not been deleted but stored separately to the primary use data.

The four stages mentioned are crucial to the retrieval and processing of data in preparation of further analysis that can lead to effective insights that influence trading decisions. The prioritisation of the horizontal scaling (cloud computing) in implementation of this system is essential to meeting the bank's requirements of high availability, scalability, accessibility, and data security whilst keeping costs low. Nevertheless, an important security factor to consider is the shared responsibility between the cloud provider and the bank. Though there are many esteemed cloud providers, growing technology poses potential risk to the data security so it is key that to keep up to date with changes in the cloud realm.

Task B.2

ABC Investment Bank wants to utilise real time processing to filter and retrieve discussions on social media centred around financial products. 'Real-time processing is a fast and prompt data processing technology that combines data capturing, data processing and data exportation together. The main purpose of Big Data real-time processing is to realise an entire system that can process such mesh data in a short time' (Yang et al., 2013). As social media platforms allow users to interact in real time to facilitate a speedy spread of information, real time processing can be a useful tool for collating this data in a timely manner which in turn, increased the speed of decision making as analytics can be drawn much faster and can also influence insights are they are inferred directly from users of financial models.

The use of a parallel distrusted cluster suggests that the bank has utilised horizontal scaling to achieve scalability and faster processing which created a great environment for real time performance. In addition, the bank is considering MapReduce as a complementary framework over other technologies. 'MapReduce is a programming model specifically developed for the management and processing of "Big Data" – extremely large amounts of data that expects a high level of analysing capabilities' (Goyal and Bharti, 2015, p.16). Though MapReduce can process large volumes of data, which is beneficial for improving scalability, this attribute is not applicable to the requirements of this task as real time data prioritises the processing speed over large volume, allowing for quicker insights. As information on social media flows in streams, batch processing may struggle to pick up relevant data points as it schedules and processes historical data in batches.

Alternatively, the use of a streaming framework such as Apache Spark can be beneficial to achieving a greater real time performance. 'Apache Spark is an open source, distributed processing system used for big data workloads, utilising memory caching, optimised query execution for fast analytic queries against data of any size' (Huang, 2024). An advantage of Spark over the MapReduce framework is it's in-memory capability which allows data to be stored on a machine's RAM, reducing the need for disk storage required by MapReduce and to enable faster processing of real time data in micro batches; on the other hand, MapReduce reads and writes all data to and from disk storage which increases processing time. As a result, this aspect of Spark offers increased efficiency and reduces the resources spent on computing. Apache Spark also offers further capabilities such 'interactive queries, real-time analytics, machine learning, and graph processing' (AWS, 2024) which are useful tools that can support ABC beyond the collation and processing of real time data.

'In May 2021 alone, Twitter saw 694,000 conversations about finance in the UK, close to doubling from 369,000 in January 2019' (X Marketing, 2021). This suggests that conversation of finance on media platforms is increasing over time as people in correlation with the UK look to discuss methods of increasing or maintaining wealth. However, as seen in Figure 1 (Statista, 2024), the number of users on the X (Twitter) platform amounted to 362.4 million which implies that approximately 0.001% of discussion on Twitter were centred around finance in the UK. This fraction is significantly small 'considering historical data' and suggests that the topic is still relatively niche. Whilst there are frameworks that are more focused on real-time data such as Apache Flink as opposed to 'near real-time', Spark provides a great balance between interactive processing and batch processing which is suitable to a finance industry given the growing interest online.

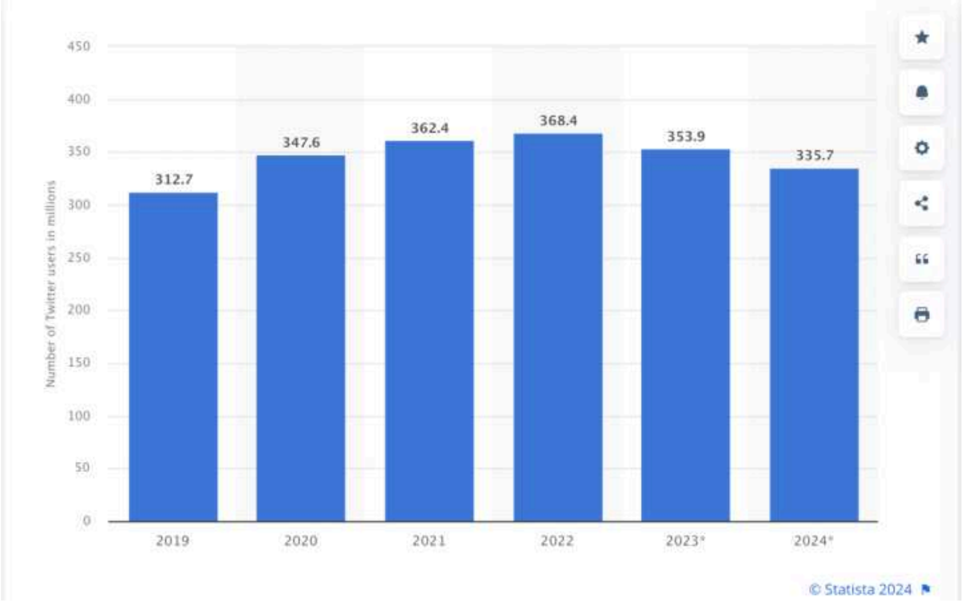


Figure 1: Number of X (formerly Twitter) users worldwide from 2019 to 2024

Considering the responsibility of investment banks and their duty to clients, processing time is significant within a framework and must be heavily considered to deliver efficient solutions, therefore I would recommend the use of Apache Spark as an alternative.

Task B.3

From the specification provided by the client, the requirements of the Big Data project entail a reasonable amount of storage space suitable for approximately 200 petabytes, high availability, scalability, accessibility from worldwide offices, and data confidentiality. All these attributes collectively influence the hosting strategy for this project as it shapes the environment in which resources are utilised.

‘Cloud computing is the on-demand access of computing resources—physical servers or virtual servers, data storage, networking capabilities, application development tools, software, AI-powered analytic tools and more—over the internet with pay-per-use pricing’ (IBM, 2024). As cloud services typically offer the full package, this makes it a cost-effective choice for hosting the system as companies spend less on acquiring hardware and software that is ready made through the cloud. Besides, the resources are readily available for developers which increases the speed and workflow of the organisation.

There are three types of hosting strategies that make up a cloud deployment model: ‘a public cloud which is owned by an outsourced cloud provider and is accessible to many businesses through the internet on a pay-per-use model, private cloud which offers a more controlled environment in which IT resources are more centralised within the business and hybrid cloud which is for businesses seeking the benefits of both private and public cloud deployment models’ (Huang, 2024).

The most suitable strategy for this project is the hybrid cloud as it provides a great balance between public and private attributes that serve as complementary factors. For instance, security is relatively strong as cloud platforms can allow ABC to assign policies to users to control the use of resources. As a result of this, offices worldwide can gain access to data under conditions such as being a part of the ‘IT and technology’ department, which supports data governance and maintaining confidentiality. Though this differs from the private model which prioritises maximum security, this misaligns with the criteria of providing access to company data across offices worldwide.

As company data is held typically at data centres in a particular location(s), spreading these centres globally can help to increase the availability of the system as it becomes more fault tolerant as data is distributed across the network to reduce workflow and decrease the chances of system failure as backups are available at each region.

Nevertheless, the hybrid approach also offers flexibility and scalability as tools are ready for use upon subscribing to the model which speeds up the process of developing and implementing the project as ABC do not need to acquire additional hardware or install software and can immediately focus on innovation. Similarly, the company is dealing with data lakes and streams, and this can vary in flow of traffic, so the hybrid model allows for the alternation of resources to meet business needs which differs from the private model which demands upfront costs that may not be consistent with the use of resources.

The hybrid model can be considered cost-effective as it inherits the ‘pay-per-use pricing’ model from which means that the organisation will only expense the resources they use as opposed to paying an upfront fee which allows the ABC to pay the equivalent of approximately 200 petabytes for storage and limits excess in expenditure rather than a reserved instance model that requires the ties companies in for a specified period of time.

Overall, whilst the recommended strategy caters to the requirement of the Big Data project, it is important to compare providers to understand the different components on offer that cater most to the desired criteria.

References

T. Zhang, H. Liu, C. Yu, and P. Wang (2023) ‘Corddl: An Efficient and Extensible Connector between Relational Databases and Data Lakes’, *2023 5th International Conference on Machine Learning, Big Data and Business Intelligence (MLBDBI), pp.* 173, doi: 10.1109/MLBDBI60823.2023.10481929.

C. -Y. Liu, M. -R. Shie, Y. -F. Lee, Y. -C. Lin and K. -C. Lai (2014) ‘Vertical/Horizontal Resource Scaling Mechanism for Federated Clouds’, 2014 *International Conference on Information Science & Applications (ICISA)*, pp. 1-4, doi: 10.1109/ICISA.2014.6847479.

Huang, H. (2024). 'Apache Pig and Data Lake' [PowerPoint presentation]. *COMP1702: Big Data.* Available at: https://moodlecurrent.gre.ac.uk/course/view.php?id=92361 (Accessed: 6th April 2024).

Monte Carlo (2023) ‘Data Ingestion: 7 Challenges and 4 Best Practices’, *Data Platforms,* Available at: https://www.montecarlodata.com/blog-data-ingestion/#:~:text=CTO%20and%20Co%2Dfounder%2C%20Monte,wizard%20and%20lover%20of%20cats.&text=Data%20ingestion%20is%20the%20process,in%20modern%20data%20management%20workflows (Accessed: 6th April 2024).

W. Yang, X. Liu, L. Zhang, and L. T. Yang. (2023) ‘Big Data Real-Time Processing Based on Storm,’ *2013 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications*, pp. 1784, doi: 10.1109/TrustCom.2013.247.

A. Goyal and Bharti. (2015) ‘Study on emerging implementations of MapReduce’, International *Conference on Computing, Communication & Automation*, pp. 16, doi: 10.1109/CCAA.2015.7148364.

Huang, H. (2024). 'Big Data - Spark' [PowerPoint presentation]. *COMP1702: Big Data.* Available at: https://moodlecurrent.gre.ac.uk/course/view.php?id=92361 (Accessed: 6th April 2024).

AWS (2024). ‘What is Apache Spark,’ Available at: https://aws.amazon.com/what-is/apache-spark/#:~:text=Apache%20Spark%20comes%20with%20the,can%20combine%20multiple%20workloads%20seamlessly (Accessed: 6th April 2024).

X Marketing (2021). ‘How Twitter connects those interested in finance and investment,’ *Research,* Available at: https://marketing.x.com/en_gb/insights/a-wealth-of-conversation--how-twitter-connects-those-interested- (Accessed: 6th April 2024).

Statista (2024) ‘Number of X (formerly Twitter) users worldwide from 2019 to 2024 (in millions)’ *Internet,* Available at: https://www.statista.com/statistics/303681/twitter-users-worldwide/ (Accessed: 6th April 2024).

IBM (2024) ‘What is Cloud Computing’, *Think: tech news, education and events,* Available at: https://www.ibm.com/topics/cloud-computing (Accessed: 6th April 2024).