1 Hadoop distributed file system (HDFS)

On commodity hardware, the distributed file system HDFS manages enormous data sets. A single Apache Hadoop cluster can be scaled up to hundreds or even thousands of nodes using HDFS. One of Apache Hadoop's key components, along with Map Reduce and YARN, is HDFS.

The dataset on which the analysis is being done in this assignment are:

1) MedianHouseholdIncome2015.csv

2) PercentagePeopleBelowPovertyLevel.csv

3) PercentOver25CompletedHighSchool.csv.

4) PoliceKillingsUS.csv

5) ShareRaceByCity.csv

These datasets are first uploaded in Local storage. Then these files are copied from Local storage to HDFS.

hadoop fs −put data/MedianHouseholdIncome2015.csv /user/$USER/

hdfs dfs −cat /user/anandba065877/MedianHouseholdIncome2015.csv

hadoop fs −put data/PercentagePeopleBelowPovertyLevel.csv /user/$USER/

hdfs dfs −cat /user/anandba065877/PercentagePeopleBelowPovertyLevel.csv

hadoop fs −put data/PercentOver25CompletedHighSchool.csv /user/$USER/

hdfs dfs −cat /user/anandba065877/PercentOver25CompletedHighSchool.csv

hadoop fs −put data/PoliceKillingsUS.csv /user/$USER/

hdfs dfs −cat /user/anandba065877/PoliceKillingsUS.csv

hadoop fs −put data/ShareRaceByCity.csv /user/$USER/

hdfs dfs −cat /user/anandba065877/ShareRaceByCity.csv

Other commands to perform different other operation on HDFS which were used here are:

hadoop fs −ls /user/$USER/

hdfs dfs −rm −r /user/anandba065877/SentimentFiles

hdfs dfs −rm −r /user/anandba065877/data

hdfs dfs −rm −r /user/anandba065877/.Trash

hdfs dfs −count −q −h /user/$USER/

hdfs dfs −du −h /user/$USER/

hdfs dfs −du /user/$USER/

hdfs dfs −df −h /

hdfs dfs −df

2 Implementation using Hive, MAPREDUCE, MySQL and Sqoop interface in CloudxLab

2.1 Loading file MedianHouseholdIncome2015.csv file on Hive Interface:

Here, we are using sg Database on Hive and creating table income test.

Then we are loading MedianHouseholdIncome2015.csv file into income test table.

```
use sg;

DROP TABLE IF EXISTS Income_test;

create table if not exists Income_test (Geographic_Area String,

City String, Median_Income int)

COMMENT 'Income2015_details'

ROW FORMAT DELIMITED

FIELDS TERMINATED BY ','

STORED AS TEXTFILE

tblproperties("skip.header.line.count" = "1");

LOAD DATA LOCAL INPATH 'data/MedianHouseholdIncome2015.csv' INTO TABLE Income_test;
```

```
hive> use sg
    > DROP TABLE IF EXISTS Income_test;
FAILED: ParseException line 2:0 missing EOF at 'DROP' near 'sg'
hive> create table if not exists Income_test (Geographic_Area String,
    > City String, Median_Income int)
    > COMMENT 'Income2015_details'
    > ROW FORMAT DELIMITED
    > FIELDS TERMINATED BY ','
    > STORED AS TEXTFILE
    > tblproperties("skip.header.line.count"="1");
OK
Time taken: 1.396 seconds
hive> LOAD DATA LOCAL INPATH 'data/MedianHouseholdIncome2015.csv' INTO TABLE Income_test;
Loading data to table default.income_test
Table default.income_test stats: [numFiles=1, numRows=0, totalSize=738306, rawDataSize=0]
OK
Time taken: 0.97 seconds
```

Here, we are loading top ten entries from income test table:

```
SELECT * FROM Income_test limit 10;
```

```
hive> SELECT *FROM Income_test limit 10;
OK
AL      Abanda CDP      11207
AL      Abbeville city  25615
AL      Adamsville city 42575
AL      Addison town    37083
AL      Akron town      21667
AL      Alabaster city  71816
AL      Albertville city        32911
AL      Alexander City city     29874
AL      Alexandria CDP  56058
AL      Aliceville city 21131
Time taken: 0.764 seconds, Fetched: 10 row(s)
```

Here, we are overwriting directory '/apps/hive/warehouse/sg.db/income test' with the contents from income test table with the help of MAP REDUCE program running in the background

insert overwrite directory '/apps/hive/warehouse/sg.db
/income_test' row format delimited fields terminated by ',' stored as textfile

select Geographic_Area, City, nvl(Median_Income, 10000) as Median_Income from Income_test limit 200;

```
2023-05-22 06:15:03,138 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 5.74 sec
2023-05-22 06:15:10,445 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 10.41 sec
MapReduce Total cumulative CPU time: 10 seconds 410 msec
Ended Job = job_1648130833540_31221
Moving data to directory /apps/hive/warehouse/sg.db/income_test
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 10.41 sec   HDFS Read: 139058 HDFS Write: 4618 SUCCESS
Total MapReduce CPU Time Spent: 10 seconds 410 msec
OK
Time taken: 25.748 seconds
```

2.2 Loading file PercentagePeopleBelowPovertyLevel.csv file on Hive Interface:

Here, we are using sg Database on Hive and creating table BelowPovertyLevel.

Then we are loading PercentagePeopleBelowPovertyLevel.csv file into BelowPovertyLevel table.

DROP TABLE IF EXISTS BelowPovertyLevel;

create table if not exists BelowPovertyLevel (PovertyArea String,

City String, poverty_rate Float)

ROW FORMAT DELIMITED

FIELDS TERMINATED BY ','

STORED AS TEXTFILE

tblproperties("skip.header.line.count" = "1");

LOAD DATA LOCAL INPATH 'data
/PercentagePeopleBelowPovertyLevel.csv' INTO TABLE BelowPovertyLevel;

```
hive> DROP TABLE IF EXISTS BelowPovertyLevel;
OK
Time taken: 1.816 seconds
hive> create table if not exists BelowPovertyLevel (PovertyArea String,
    > City String, poverty_rate Float)
    > ROW FORMAT DELIMITED
    > FIELDS TERMINATED BY ','
    > STORED AS TEXTFILE
    > tblproperties("skip.header.line.count"="1");
OK
Time taken: 0.209 seconds
hive> LOAD DATA LOCAL INPATH 'data/PercentagePeopleBelowPovertyLevel.csv' INTO TABLE BelowPovertyLevel;
Loading data to table default.belowpovertylevel
Table default.belowpovertylevel stats: [numFiles=1, numRows=0, totalSize=693751, rawDataSize=0]
OK
Time taken: 0.923 seconds
```

Here, we are loading top ten entries from BelowPovertyLevel table:

*SELECT * FROM BelowPovertyLevel limit* 10;

```
hive> SELECT *FROM BelowPovertyLevel limit 10;
OK
AL      Abanda CDP      78.8
AL      Abbeville city  29.1
AL      Adamsville city 25.5
AL      Addison town    30.7
AL      Akron town      42.0
AL      Alabaster city  11.2
AL      Albertville city        26.7
AL      Alexander City city     30.4
AL      Alexandria CDP  9.7
AL      Aliceville city 41.3
Time taken: 0.423 seconds, Fetched: 10 row(s)
```

Here, we are overwriting directory ' /apps/hive/warehouse/sg.db/BelowPovertyLevel' with the contents from BelowPovertyLevel table with the help of MAP REDUCE program running in the background.

insert overwrite directory '/apps/hive/warehouse/sg.db
/income_test' row format delimited fields terminated by ',' stored as textfile

select PovertyArea, PovertyCity, nvl(poverty_rate, 0.0*) as poverty_rate from BelowPovertyLevel limit* 200;

```
2023-05-22 06:10:55,919 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 4.18 sec
2023-05-22 06:11:06,278 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 6.61 sec
MapReduce Total cumulative CPU time: 6 seconds 610 msec
Ended Job = job_1648130833540_31220
Moving data to directory /apps/hive/warehouse/sg.db/BelowPovertyLevel
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 6.61 sec   HDFS Read: 139315 HDFS Write: 6526 SUCCESS
Total MapReduce CPU Time Spent: 6 seconds 610 msec
OK
Time taken: 32.895 seconds
```

2.3 Loading file PercentOver25CompletedHighSchool.csv file on Hive Interface:

Here, we are using sg Database on Hive and creating table completed_hs.

Then we are loading PercentOver25CompletedHighSchool.csv file into completed_hs table.

```
DROP TABLE IF EXISTS completed_hs;

create table if not exists completed_hs (hsArea String,

City String,percent_completed_hs Float)

COMMENT 'Income2015_details'

ROW FORMAT DELIMITED

FIELDS TERMINATED BY ','

STORED AS TEXTFILE

tblproperties("skip.header.line.count" = "1");

LOAD DATA LOCAL INPATH 'data
                /PercentOver25CompletedHighSchool.csv' INTO TABLE completed_hs;
```

```
hive> DROP TABLE IF EXISTS completed_hs;
OK
Time taken: 0.025 seconds
hive> create table if not exists completed_hs (hsArea String,
    > City String, percent_completed_hs Float)
    > COMMENT 'Income2015_details'
    > ROW FORMAT DELIMITED
    > FIELDS TERMINATED BY ','
    > STORED AS TEXTFILE
    > tblproperties("skip.header.line.count"="1");
OK
Time taken: 0.267 seconds
hive>
    > LOAD DATA LOCAL INPATH 'data/PercentOver25CompletedHighSchool.csv' INTO TABLE completed_hs;
Loading data to table default.completed_hs
Table default.completed_hs stats: [numFiles=1, numRows=0, totalSize=705655, rawDataSize=0]
OK
```

Here, we are loading top ten entries from completed_hs table:

```
SELECT * FROM completed_hs limit 10;
```

```
hive> SELECT * FROM completed_hs limit 10;
OK
AL      Abanda CDP      21.2
AL      Abbeville city  69.1
AL      Adamsville city 78.9
AL      Addison town    81.4
AL      Akron town      68.6
AL      Alabaster city  89.3
AL      Albertville city        72.7
AL      Alexander City city     78.1
AL      Alexandria CDP  88.8
AL      Aliceville city 74.3
Time taken: 0.05 seconds, Fetched: 10 row(s)
```

Here, we are overwriting directory ' /apps/hive/warehouse/sg.db/completed_hs 'with the contents from completed_hs with the help of MAP REDUCE program running in the background

insert overwrite directory '/apps/hive/warehouse/sg.db
/completed_hs' row format delimited fields terminated by ',' stored as textfile

select hsArea, hsCity, nvl(percent_completed_hs, 0.0) as percent_completed_hs from completed_hs limit 200;

```
2023-05-22 06:31:07,849 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 4.0 sec
2023-05-22 06:31:14,084 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 7.52 sec
MapReduce Total cumulative CPU time: 7 seconds 520 msec
Ended Job = job_1648130833540_31222
Moving data to directory /apps/hive/warehouse/sg.db/completed_hs
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 7.52 sec   HDFS Read: 139298 HDFS Write: 6420 SUCCESS
Total MapReduce CPU Time Spent: 7 seconds 520 msec
OK
Time taken: 26.507 seconds
```

2.4 Loading file MedianHouseholdIncome2015.csv file on MYSQL Interface:

Here, we are launching MySQL in CloudxLab

mysql −h cxln2.c.thelab−240901.internal −u sqoopuser −pNHkkP876rp

```
-bash-4.2$ mysql -h cxln2.c.thelab-240901.internal -u sqoopuser -pNHkkP876rp
Warning: Using a password on the command line interface can be insecure.
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 194629
Server version: 5.6.44 MySQL Community Server (GPL)

Copyright (c) 2000, 2019, Oracle and/or its affiliates. All rights reserved.
```

Here we are using sqoopex database and creating Income2015 table in MySQL

```
use sqoopex;

create table if not exists Income2015 (

Geographic_Area varchar(30) not null default 'New',

City varchar(30) not null default 'New',

Median_Income int not null);
```

```
mysql> use sqoopex;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> DROP TABLE IF EXISTS Income2015;
Query OK, 0 rows affected (0.03 sec)

mysql> create table if not exists Income2015 (
    -> Geographic_Area varchar(30) not null default 'New',
    -> City varchar(30) not null default 'New',
    -> Median_Income int not null);
Query OK, 0 rows affected (0.04 sec)
```

Here we are performing Sqoop Export - Hive to MySQL with the help of MAP REDUCE program running in the background:

```
sqoop export --connect jdbc:mysql://cxln2.c.thelab-240901.internal/sqoopex -m 1
             --table Income2015 --export
             --dir /apps/hive/warehouse/sg.db/income_test/000000_0 --input-fields
             -terminated-by ',' --username sqoopuser --password NHkkP876rp;
```

```
        Map-Reduce Framework
                Map input records=200
                Map output records=200
                Input split bytes=170
                Spilled Records=0
                Failed Shuffles=0
                Merged Map outputs=0
                GC time elapsed (ms)=81
                CPU time spent (ms)=770
                Physical memory (bytes) snapshot=223567872
                Virtual memory (bytes) snapshot=4568371200
                Total committed heap usage (bytes)=150994944
        File Input Format Counters
                Bytes Read=0
        File Output Format Counters
                Bytes Written=0
23/05/22 06:39:59 INFO mapreduce.ExportJobBase: Transferred 4.6787 KB in 18.6824 seconds (256.444 bytes/sec)
23/05/22 06:39:59 INFO mapreduce.ExportJobBase: Exported 200 records.
```

Here, we are loading top ten entries from Income2015table in MySQL interface in CloudxLab:

*SELECT * FROM Income2015 limit 10;*

```
mysql> SELECT *FROM Income2015 limit 10;
+-----------------+-----------------+---------------+
| Geographic_Area | City            | Median_Income |
+-----------------+-----------------+---------------+
| AL              | Garden City town |         31591 |
| AL              | Gantt town      |         47000 |
| AL              | Gallant CDP     |         45859 |
| AL              | Gainesville town |         14444 |
| AL              | Gadsden city    |         29136 |
| AL              | Fyffe town      |         42356 |
| AL              | Fultondale city |         51587 |
| AL              | Fulton town     |         32083 |
| AL              | Fruithurst town |         35625 |
| AL              | Fruitdale CDP   |         63125 |
+-----------------+-----------------+---------------+
10 rows in set (0.00 sec)
```

2.5 Loading file PercentagePeopleBelowPovertyLevel.csv file on MYSQL Interface:

Here we are using sqoopex database and creating BelowPovertyLevel_sql table in MySQL

DROP TABLE IF EXISTS BelowPovertyLevel_sql;

create table if not exists BelowPovertyLevel_sql (PovertyArea varchar(30) not null default 'New',

City varchar(30) not null default 'New',

poverty_rate Float not null);

```
mysql> create table if not exists BelowPovertyLevel_sql (PovertyArea varchar(30) not null default 'New',
    -> City varchar(30) not null default 'New',
    -> poverty_rate Float not null);
Query OK, 0 rows affected (0.05 sec)
```

Here we are performing Sqoop Export - Hive to MySQL with the help of MAP REDUCE program running in the background:

sqoop export --connect jdbc:mysql://cxln2.c.thelab-240901.internal/sqoopex -m 1
--table BelowPovertyLevel_sql --export
--dir /apps/hive/warehouse/sg.db/BelowPovertyLevel/000000_0 --input-fields
-terminated-by ',' --username sqoopuser --password NHkkP876rp;

```
        Map-Reduce Framework
                Map input records=200
                Map output records=200
                Input split bytes=176
                Spilled Records=0
                Failed Shuffles=0
                Merged Map outputs=0
                GC time elapsed (ms)=70
                CPU time spent (ms)=1050
                Physical memory (bytes) snapshot=218431488
                Virtual memory (bytes) snapshot=4588646400
                Total committed heap usage (bytes)=166723584
        File Input Format Counters
                Bytes Read=0
        File Output Format Counters
                Bytes Written=0
23/05/22 06:48:20 INFO mapreduce.ExportJobBase: Transferred 6.5479 KB in 18.7472 seconds (357.6542 bytes/sec)
23/05/22 06:48:20 INFO mapreduce.ExportJobBase: Exported 200 records.
```

Here, we are loading top ten entries from BelowPovertyLevel_sql in MySQL interface in CloudxLab:

*SELECT * FROM BelowPovertyLevel_sql limit* 10;

```
mysql> SELECT *FROM BelowPovertyLevel_sql limit 10;
+-------------+------------------+--------------+
| PovertyArea | City             | poverty_rate |
+-------------+------------------+--------------+
| AL          | Garden City town |         19.3 |
| AL          | Gantt town       |          8.3 |
| AL          | Gallant CDP      |          3.9 |
| AL          | Gainesville town |         40.9 |
| AL          | Gadsden city     |         29.4 |
| AL          | Fyffe town       |         12.8 |
| AL          | Fultondale city  |           14 |
| AL          | Fulton town      |         19.9 |
| AL          | Fruithurst town  |         13.5 |
| AL          | Fruitdale CDP    |           33 |
+-------------+------------------+--------------+
10 rows in set (0.00 sec)
```

2.6 Loading file PercentOver25CompletedHighSchool.csv file on MYSQL Interface:

Here we are using sqoopex database and creating completed_hs_sql table in MySQL

```
DROP TABLE IF EXISTS completed_hs_sql;

create table if not exists completed_hs_sql (hsArea varchar(30) not null default 'New',

City varchar(30) not null default 'New',

percent_completed_hs Float not null);
```

```
mysql> DROP TABLE IF EXISTS completed_hs_sql;
Query OK, 0 rows affected (0.04 sec)

mysql> create table if not exists completed_hs_sql (hsArea varchar(30) not null default 'New',
    -> City varchar(30) not null default 'New',
    -> percent_completed_hs Float not null);
Query OK, 0 rows affected (0.03 sec)
```

Here we are performing Sqoop Export - Hive to MySQL with the help of MAP REDUCE program running in the background:

```
sqoop export --connect jdbc:mysql://cxln2.c.thelab-240901.internal/sqoopex -m 1
             --table completed_hs_sql --export
             --dir /apps/hive/warehouse/sg.db/completed_hs/000000_0 --input-fields
             -terminated-by ',' --username sqoopuser --password NHkkP876rp;
```

```
        Map-Reduce Framework
                Map input records=200
                Map output records=200
                Input split bytes=171
                Spilled Records=0
                Failed Shuffles=0
                Merged Map outputs=0
                GC time elapsed (ms)=85
                CPU time spent (ms)=840
                Physical memory (bytes) snapshot=223604736
                Virtual memory (bytes) snapshot=4568698880
                Total committed heap usage (bytes)=155189248
        File Input Format Counters
                Bytes Read=0
        File Output Format Counters
                Bytes Written=0
23/05/22 06:56:02 INFO mapreduce.ExportJobBase: Transferred 6.4395 KB in 18.8134 seconds (350.4946 bytes/sec)
23/05/22 06:56:02 INFO mapreduce.ExportJobBase: Exported 200 records.
```

Here, we are loading top ten entries from completed_hs_sql in MySQL interface in CloudxLab:

*SELECT * FROM completed_hs_sql limit* 10;

```
mysql> SELECT *FROM completed_hs_sql limit 10;
+--------+------------------+----------------------+
| hsArea | City             | percent_completed_hs |
+--------+------------------+----------------------+
| AL     | Garden City town |                 79.8 |
| AL     | Gantt town       |                 85.6 |
| AL     | Gallant CDP      |                 83.7 |
| AL     | Gainesville town |                 76.7 |
| AL     | Gadsden city     |                 78.3 |
| AL     | Fyffe town       |                 83.7 |
| AL     | Fultondale city  |                 85.1 |
| AL     | Fulton town      |                   76 |
| AL     | Fruithurst town  |                 75.8 |
| AL     | Fruitdale CDP    |                 93.3 |
+--------+------------------+----------------------+
10 rows in set (0.00 sec)
```

2.7 The insights drawn out of the analysis on MySQL interface in CloudxLab:

2.7.1The dataset on which the analysis is being done:

1) MedianHouseholdIncome2015.csv

2) PercentagePeopleBelowPovertyLevel.csv

3) PercentOver25CompletedHighSchool.csv.

Here we are using sqoopex database and creating income scan table by left outer join of Income2015 and BelowPovertyLevel_sql table in MySQL

DROP TABLE IF EXISTS income_scan;

CREATE TABLE income_scan select Income2015.Geographic_Area,Income2015.City,Income2015.Median_Income,Below

from Income2015 left outer join BelowPovertyLevel_sql

on Income2015.City = BelowPovertyLevel_sql.City

order by Income2015.Median_Income ;

```
mysql> DROP TABLE IF EXISTS income_scan;
Query OK, 0 rows affected (0.06 sec)

mysql> CREATE TABLE income_scan select Income2015.Geographic_Area,Income2015.City,Income2015.Median_Income,BelowPovertyLevel_sql.poverty_rate
    -> from Income2015 left outer join  BelowPovertyLevel_sql
    -> on Income2015.City = BelowPovertyLevel_sql.City
    -> order by Income2015.Median_Income ;
Query OK, 200 rows affected (0.05 sec)
Records: 200  Duplicates: 0  Warnings: 0
```

Here we are using sqoopex database and creating BelowPoverty_scan table by right outer join of Income2015 and BelowPovertyLevel_sql table in MySQL

```
DROP TABLE IF EXISTS BelowPoverty_scan;

CREATE TABLE BelowPoverty_scan select BelowPovertyLevel_sql.PovertyArea, BelowPovertyLevel_sql.City, Income201

from Income2015 right outer join  BelowPovertyLevel_sql

on Income2015.City  =  BelowPovertyLevel_sql.City

order by Income2015.Median_Income;
```

```
mysql> CREATE TABLE BelowPoverty_scan select BelowPovertyLevel_sql.PovertyArea,BelowPovertyLevel_sql.City,Income2015.Median_Income,BelowPovertyLevel_sql.poverty_ra
te
    -> from Income2015 right outer join  BelowPovertyLevel_sql
    -> on Income2015.City = BelowPovertyLevel_sql.City
    -> order by Income2015.Median_Income;
Query OK, 200 rows affected (0.06 sec)
Records: 200  Duplicates: 0  Warnings: 0
```


Here we are using sqoopex database and creating completed_hs_scan table by left outer join of completed_hs_sql and BelowPovertyLevel_sql table in MySQL

```
DROP TABLE IF EXISTS completed_hs_scan;

CREATE TABLE completed_hs_scan select completed_hs_sql.hsArea, completed_hs_sql.City, completed_hs_sql.percent_co

from completed_hs_sql left outer join  BelowPovertyLevel_sql

on completed_hs_sql.City  =  BelowPovertyLevel_sql.City

order by completed_hs_sql.percent_completed_hs;
```

```
mysql> CREATE TABLE completed_hs_scan select completed_hs_sql.hsArea,completed_hs_sql.City,completed_hs_sql.percent_completed_hs,BelowPovertyLevel_sql.poverty

    -> from completed_hs_sql left outer join  BelowPovertyLevel_sql
    -> on completed_hs_sql.City = BelowPovertyLevel_sql.City
    -> order by completed_hs_sql.percent_completed_hs;
Query OK, 200 rows affected (0.05 sec)
Records: 200  Duplicates: 0  Warnings: 0
```


2.7.2 Analyzing income scan table:

Here, we are querying income scan table to compare and analyze poverty rate and Median Income for people living in cities as shown below:

```
SELECT MIN(Median_Income)FROM income_scan;

SELECT MAX(Median_Income)FROM income_scan;

SELECT AVG(Median_Income)FROM income_scan;
```

```
mysql> SELECT MIN(Median_Income)FROM income_scan;
+--------------------+
| MIN(Median_Income) |
+--------------------+
|              10000 |
+--------------------+
1 row in set (0.00 sec)

mysql> SELECT MAX(Median_Income)FROM income_scan;
+--------------------+
| MAX(Median_Income) |
+--------------------+
|              96333 |
+--------------------+
1 row in set (0.00 sec)

mysql> SELECT AVG(Median_Income)FROM income_scan;
+--------------------+
| AVG(Median_Income) |
+--------------------+
|         35402.4950 |
+--------------------+
```

Here, we are querying to extract increasing ten median incomes listings from income scan Table to compare and analyze poverty rate and Median Income for people living in cities as shown below:

SELECT City, Median_Income, poverty_rate FROM income_scan

GROUP BY City HAVING Median_Income $>$ 10000 *and Median_Income* $<$ 35402

ORDER BY Median_Income

LIMIT 10;

```
mysql> SELECT City,Median_Income,poverty_rate FROM income_scan
    -> GROUP BY City HAVING Median_Income > 10000 and Median_Income < 35402
    -> ORDER BY Median_Income
    -> LIMIT 10;
+------------------+---------------+--------------+
| City             | Median_Income | poverty_rate |
+------------------+---------------+--------------+
| Abanda CDP       |         11207 |         78.8 |
| Boligee town     |         12173 |         60.3 |
| Gainesville town |         14444 |         40.9 |
| Frisco City town |         17311 |         31.3 |
| Beatrice town    |         18417 |         44.9 |
| Evergreen city   |         18661 |         52.6 |
| Forkland town    |         19063 |         48.1 |
| Cordova city     |         19139 |         32.8 |
| Clio city        |         19539 |         31.4 |
| Coffeeville town |         19583 |         41.3 |
+------------------+---------------+--------------+
10 rows in set (0.01 sec)
```

Insights drawn: We can see that the query returned ten records with growing Median Income, demonstrating that poverty rates frequently decrease as median income rises. Another conclusion is that based on the general income ranges of the cities, the poverty rate estimate vary for each city. Additionally, the bottom 10 cities with the least median income are Abanda CDP, Boligee Town, Gainesville Town, Frisco City Town, Beatrice Town, Evergreen Town, Forkland Town, Cordova Town, Clio Town, and Coffeeville Town.

Here, we are querying to extract descending ten median incomes listings from income scan Table to compare and analyze poverty rate and Median Income for people living in cities as shown below:

```
SELECT City,Median_Income,poverty_rate FROM income_scan

GROUP BY City

ORDER BY Median_Income Desc

LIMIT 10;
```

```
mysql> SELECT City,Median_Income,poverty_rate FROM income_sca
    -> GROUP BY City
    -> ORDER BY Median_Income Desc
    -> LIMIT 10;
+---------------------+---------------+--------------+
| City                | Median_Income | poverty_rate |
+---------------------+---------------+--------------+
| Blue Ridge CDP      |         96333 |          0.5 |
| Emerald Mountain CDP |         91054 |          8.2 |
| Chelsea city        |         85757 |          4.9 |
| Deatsville town     |         78125 |          5.1 |
| Alabaster city      |         71816 |         11.2 |
| Clay city           |         68717 |          6.6 |
| Fredonia CDP        |         66591 |         18.4 |
| Dauphin Island town |         63594 |          3.6 |
| Fruitdale CDP       |         63125 |           33 |
| Calera city         |         62893 |          5.8 |
+---------------------+---------------+--------------+
10 rows in set (0.00 sec)
```

Insights drawn: As we can see, the query returned the top 10 results for Median Income in descending order, showing that cities with higher median incomes had lower poverty rates. Usually, when the median income falls, the poverty rate rises. Another conclusion is that the calculation of the poverty rate differs for each city depending on the general income ranges of the cities. Apart from that, the top ten cities with declining median income are Blue Ridge CDP, Emerald Mountain CDP, Chelsea city, Deatsville town, Alabaster city, Clay city, Fredonia CDP, Duphin Island town, Fruitdale CDP, and Calera city.

SELECT COUNT(*) FROM income_scan

where Median_Income > 10000 and Median_Income < 35402;

SELECT COUNT(*) FROM income_scan

where Median_Income > 35402 and Median_Income <= 96333;

```
mysql> SELECT COUNT(*) FROM income_scan
    -> where Median_Income > 10000 and Median_Income <  35402;
+----------+
| COUNT(*) |
+----------+
|       96 |
+----------+
1 row in set (0.00 sec)

mysql> SELECT COUNT(*) FROM income_scan
    -> where Median_Income >  35402 and Median_Income <= 96333;
+----------+
| COUNT(*) |
+----------+
|       89 |
+----------+
1 row in set (0.01 sec)
```

2.7.3 Analyzing BelowPoverty_scan table:

Here, we are querying BelowPoverty_scan table to compare and analyze poverty rate and Median Income for people living in cities as shown below:

SELECT MIN(poverty_rate)FROM BelowPoverty_scan;

SELECT MAX(poverty_rate)FROM BelowPoverty_scan;

SELECT AVG(poverty_rate)FROM BelowPoverty_scan;

```
mysql> SELECT MIN(poverty_rate)FROM BelowPoverty_scan;
+-------------------+
| MIN(poverty_rate) |
+-------------------+
|                 0 |
+-------------------+
1 row in set (0.01 sec)

mysql> SELECT MAX(poverty_rate)FROM BelowPoverty_scan;
+-------------------+
| MAX(poverty_rate) |
+-------------------+
|  79.4000015258789 |
+-------------------+
1 row in set (0.00 sec)

mysql> SELECT AVG(poverty_rate)FROM BelowPoverty_scan;
+--------------------+
| AVG(poverty_rate)  |
+--------------------+
| 22.827999968528747 |
+--------------------+
1 row in set (0.00 sec)
```

Here, we are querying to extract increasing ten poverty rate listings from BelowPoverty_scan Table to compare and analyze poverty rate and Median Income for people living in cities as shown below:

SELECT City, Median_Income, poverty_rate FROM BelowPoverty_scan

GROUP BY City HAVING poverty_rate $>$ 0.0 *and poverty_rate* $<$ 22.82 *and Median_Income*! = 10000

ORDER BY poverty_rate

LIMIT 10;

```
mysql> SELECT City,Median_Income,poverty_rate FROM BelowPoverty_scan
    -> GROUP BY City HAVING poverty_rate > 0.0 and poverty_rate < 22.82 and Median_Income!=10000
    -> ORDER BY poverty_rate
    -> LIMIT 10;
+--------------------+---------------+--------------+
| City               | Median_Income | poverty_rate |
+--------------------+---------------+--------------+
| Blue Ridge CDP     |         96333 |          0.5 |
| Dauphin Island town |         63594 |          3.6 |
| Gallant CDP        |         45859 |          3.9 |
| Blue Springs town  |         45313 |            4 |
| Egypt CDP          |         56693 |          4.5 |
| Chelsea city       |         85757 |          4.9 |
| Edwardsville town  |         38438 |          5.1 |
| Deatsville town    |         78125 |          5.1 |
| Calera city        |         62893 |          5.8 |
| Franklin town      |         60417 |          6.6 |
+--------------------+---------------+--------------+
10 rows in set (0.00 sec)
```

Insights drawn: As we can see, the search produced 10 records with entries for the poverty rate, which should have been rising. However, it is evident from the statistics that based on the overall income ranges of the cities, the calculation of poverty rates varies for each city. Also, the top 10 cities with the lowest rates of poverty are Blue Ridge CDP, Dauphin Island town, Gallant CDP, Blue Springs town, Egypt CDP, Chelsea city, Edwardsville town, Deatsville town, Calera city, and Franklin town.

Here, we are querying to extract descending ten poverty_rate listings from BelowPoverty_scan Table to compare and analyze poverty rate and Median Income for people living in cities as shown below:

SELECT City, Median_Income, poverty_rate FROM BelowPoverty_scan

GROUP BY City HAVING Median_Income! = 10000

ORDER BY poverty_rate Desc

LIMIT 10;

```
mysql> SELECT City,Median_Income,poverty_rate FROM BelowPoverty_scan
    -> GROUP BY City HAVING Median_Income!=10000
    -> ORDER BY poverty_rate Desc
    -> LIMIT 10;
+------------------+---------------+--------------+
| City             | Median_Income | poverty_rate |
+------------------+---------------+--------------+
| Abanda CDP       |         11207 |         78.8 |
| Boligee town     |         12173 |         60.3 |
| Evergreen city   |         18661 |         52.6 |
| Forkland town    |         19063 |         48.1 |
| Bon Air town     |         23750 |         47.6 |
| Beatrice town    |         18417 |         44.9 |
| Bellamy CDP      |         22132 |         42.9 |
| Akron town       |         21667 |           42 |
| Aliceville city  |         21131 |         41.3 |
| Coffeeville town |         19583 |         41.3 |
+------------------+---------------+--------------+
10 rows in set (0.00 sec)
```

Insights drawn: Typically, the poverty rate increases as the median income declines. However, based on the general income ranges of the cities, the calculation of the poverty rate varies for each one. In addition, the following places are home to the bottom 10 cities with the highest rates of poverty: Abanda CDP, Boligee town, Evergreen city, Forkland town, Bon Air town, Beatrice town, Bellamy CDP, Akron town, Aliceville city, and Coffeeville town

$SELECT\ COUNT(*)\ FROM\ BelowPoverty\_scan$

$where\ poverty\_rate > 0.0\ and\ poverty\_rate < 22.82\ and\ Median\_Income! = 10000;$

$SELECT\ COUNT(*)\ FROM\ BelowPoverty\_scan$

$where\ poverty\_rate > 22.82\ and\ poverty\_rate <= 80\ and\ Median\_Income! = 10000;$

```
mysql> SELECT COUNT(*) FROM BelowPoverty_scan
    -> where poverty_rate > 0.0  and poverty_rate < 22.82 and Median_Income!=10000;
+----------+
| COUNT(*) |
+----------+
|       88 |
+----------+
1 row in set (0.00 sec)

mysql> SELECT COUNT(*) FROM BelowPoverty_scan
    -> where poverty_rate > 22.82 and poverty_rate <= 80 and Median_Income!=10000;
+----------+
| COUNT(*) |
+----------+
|       93 |
+----------+
1 row in set (0.00 sec)
```

2.7.4 Analyzing completed_hs_scan table:

Here, we are querying completed_hs_scan table to compare and analyze poverty rate and Median Income for people living in cities as shown below:

SELECT MIN(percent_completed_hs)FROM completed_hs_scan;

SELECT MAX(percent_completed_hs)FROM completed_hs_scan;

SELECT AVG(percent_completed_hs)FROM completed_hs_scan;

```
mysql> SELECT MIN(percent_completed_hs)FROM completed_hs_scan;
+---------------------------+
| MIN(percent_completed_hs) |
+---------------------------+
|        21.200000762939453 |
+---------------------------+
1 row in set (0.00 sec)

mysql> SELECT MAX(percent_completed_hs)FROM completed_hs_scan;
+---------------------------+
| MAX(percent_completed_hs) |
+---------------------------+
|                       100 |
+---------------------------+
1 row in set (0.00 sec)

mysql> SELECT AVG(percent_completed_hs)FROM completed_hs_scan;
+---------------------------+
| AVG(percent_completed_hs) |
+---------------------------+
|         80.00199993133545 |
+---------------------------+
1 row in set (0.00 sec)
```

Here, we are querying to extract increasing ten percent_completed_hs listings from completed_hs_scan Table to compare and analyze poverty rate and percent_completed_hs for people living in cities as shown below:

SELECT City, percent_completed_hs, poverty_rate FROM completed_hs_scan

GROUP BY City HAVING percent_completed_hs $>$ 21 *and percent_completed_hs* $<$ 80 *and poverty_rate*! $=$ 0

ORDER BY percent_completed_hs

LIMIT 10;

```
mysql> SELECT City,percent_completed_hs,poverty_rate FROM completed_hs_scan
    -> GROUP BY City HAVING percent_completed_hs > 21 and percent_completed_hs < 80 and poverty_rate!=0
    -> ORDER BY percent_completed_hs
    -> LIMIT 10;
+---------------------+----------------------+--------------+
| City                | percent_completed_hs | poverty_rate |
+---------------------+----------------------+--------------+
| Abanda CDP          |                 21.2 |         78.8 |
| Calvert CDP         |                 22.8 |         79.4 |
| Allgood town        |                 48.1 |         27.7 |
| Collinsville town   |                   52 |         32.1 |
| Boykin CDP          |                 53.7 |         53.7 |
| Clayton town        |                 57.8 |         29.3 |
| Bayou La Batre city |                 58.5 |         30.1 |
| Fitzpatrick CDP     |                 60.7 |         15.5 |
| Five Points town    |                 61.9 |         20.4 |
| Clio city           |                 63.1 |         31.4 |
+---------------------+----------------------+--------------+
10 rows in set (0.00 sec)
```

Insights drawn: Based on the findings stated above, we can say that cities with even the lowest rates of poverty have the lowest proportion of residents who have finished their higher secondary education, however occasionally the results are the exact opposite. The bottom 10 cities, which are shown below, are those with the lowest percentages of citizens who have completed a higher education: Abanda CDP, Calvert CDP, Allgood town, Collinsville town, Boykin CDP, Clayton town, Bayou La Batre city, Fitzpatrick CDP, Five Points town, and Clio city.

Here, we are querying to extract descending ten percent_completed_hs listings from completed_hs_scan Table to compare and analyze poverty rate and percent_completed_hs for people living in cities as shown below:

SELECT City,percent_completed_hs,poverty_rate FROM completed_hs_scan

GROUP BY City HAVING poverty_rate! = 0

ORDER BY percent_completed_hs Desc

LIMIT 10;

```
mysql> SELECT City,percent_completed_hs,poverty_rate FROM completed_hs_scan
    -> GROUP BY City HAVING poverty_rate!=0
    -> ORDER BY percent_completed_hs Desc
    -> LIMIT 10;
+--------------------+----------------------+--------------+
| City               | percent_completed_hs | poverty_rate |
+--------------------+----------------------+--------------+
| Dayton town        |                  100 |           48 |
| Fort Rucker CDP    |                 98.8 |            8 |
| Blue Ridge CDP     |                 97.6 |          0.5 |
| Deatsville town    |                 96.3 |          5.1 |
| Brook Highland CDP |                 96.2 |         11.5 |
| Dauphin Island town|                 95.8 |          3.6 |
| Daphne city        |                 95.8 |         11.2 |
| Blue Springs town  |                 95.6 |            4 |
| Fairhope city      |                 95.4 |         11.2 |
| Auburn city        |                 94.3 |         31.8 |
+--------------------+----------------------+--------------+
10 rows in set (0.00 sec)
```

Insights drawn: Cities with larger percentages of residents who have completed their higher secondary education typically don't appear to be associated to the poverty rate graph. Additionally, the following are home to the bottom 10 cities with the highest percentage of residents finishing higher secondary education: Dayton town, Fort Rucker CDP, Blue Ridge CDP, Deatsville town, Brook Highland CDP, Dauphin Island town, Daphne city, Blue Springs town, Fairhope city, Auburn city.

$SELECT\ COUNT(*)\ FROM\ completed\_hs\_scan$

$where\ percent\_completed\_hs\ >\ 21\ and\ percent\_completed\_hs\ <\ 80\ and\ poverty\_rate! = 0;$

$SELECT\ COUNT(*)\ FROM\ completed\_hs\_scan$

$where\ percent\_completed\_hs\ >\ 80\ and\ percent\_completed\_hs\ <= 100\ and\ poverty\_rate! = 0;$

```
mysql> SELECT COUNT(*) FROM completed_hs_scan
    -> where percent_completed_hs > 21 and percent_completed_hs <  80 and poverty_rate!=0;
+----------+
| COUNT(*) |
+----------+
|       97 |
+----------+
1 row in set (0.00 sec)

mysql> SELECT COUNT(*) FROM completed_hs_scan
    -> where percent_completed_hs >  80 and percent_completed_hs <= 100 and poverty_rate!=0;
+----------+
| COUNT(*) |
+----------+
|       94 |
+----------+
1 row in set (0.00 sec)
```

3 Implementation using Hive and MAPREDUCE interface in CloudxLab

3.1 Loading file PoliceKillingsUS.csv file on Hive Interface:

Here, we are using sg Database on Hive and creating table PoliceKillingsUS.

Then we are loading PoliceKillingsUS.csv file into PoliceKillingsUS table.

hive

use sg;

DROP TABLE IF EXISTS PoliceKillingsUS;

create table if not exists PoliceKillingsUS (id int,

name String, date1 String, manner_of_death String, armed String, age int,

gender String, race String, city String, state String, signs_of_mental_illness String,

threat_level String, flee String, body_camera String)

ROW FORMAT DELIMITED

FIELDS TERMINATED BY ','

STORED AS TEXTFILE

tblproperties("skip.header.line.count" = "1");

```
hive> use sg;
OK
Time taken: 1.95 seconds
hive>
    > DROP TABLE IF EXISTS PoliceKillingsUS;
OK
Time taken: 0.072 seconds
hive> create table if not exists PoliceKillingsUS (id int,
    > name String,date1 String,manner_of_death String,armed String, age int,
    > gender String,race String,city String,state String,signs_of_mental_illness String,
    > threat_level String,flee String,body_camera String)
    > ROW FORMAT DELIMITED
    > FIELDS TERMINATED BY ','
    > STORED AS TEXTFILE
    > tblproperties("skip.header.line.count"="1");
OK
Time taken: 0.249 seconds
```

Then we are loading PoliceKillingsUS.csv from HDFS into our PoliceKillingsUS Table in SG database on Hive interface in CloudxLab.

hadoop fs – put data/PoliceKillingsUS.csv /user/$USER/

load data inpath '/user/anandba065877
/PoliceKillingsUS.csv' overwrite into table sg.PoliceKillingsUS;

```
hive> load data inpath '/user/anandba065877/PoliceKillingsUS.csv' overwrite into table sg.PoliceKillingsUS;
Loading data to table sg.policekillingsus
Table sg.policekillingsus stats: [numFiles=1, numRows=0, totalSize=227653, rawDataSize=0]
OK
Time taken: 0.892 seconds
hive>
```

Here, we are loading top ten entries from PoliceKillingsUS table:

*SELECT * FROM PoliceKillingsUS limit* 10;

```
hive> SELECT *FROM PoliceKillingsUS limit 10;
OK
3       Tim Elliot      02/01/15        shot    gun     53      M       A       Shelton WA      TRUE    attack  Not fleeing     FALSE
4       Lewis Lee Lembke        02/01/15        shot    gun     47      M       W       Aloha   OR      FALSE   attack  Not fleeing     FALSE
5       John Paul Quintero      03/01/15        shot and Tasered        unarmed 23      M       H       Wichita KS      FALSE   other   Not fleeing     FALSE
8       Matthew Hoffman 04/01/15        shot    toy weapon      32      M       W       San Francisco   CA      TRUE    attack  Not fleeing     FALSE
9       Michael Rodriguez       04/01/15        shot    nail gun        39      M       H       Evans   CO      FALSE   attack  Not fleeing     FALSE
11      Kenneth Joe Brown       04/01/15        shot    gun     18      M       W       Guthrie OK      FALSE   attack  Not fleeing     FALSE
13      Kenneth Arnold Buck     05/01/15        shot    gun     22      M       H       Chandler        AZ      FALSE   attack  Car     FALSE
15      Brock Nichols   06/01/15        shot    gun     35      M       W       Assaria KS      FALSE   attack  Not fleeing     FALSE
16      Autumn Steele   06/01/15        shot    unarmed 34      F       W       Burlington      IA      FALSE   other   Not fleeing     TRUE
17      Leslie Sapp III 06/01/15        shot    toy weapon      47      M       B       Knoxville       PA      FALSE   attack  Not fleeing     FALSE
Time taken: 0.334 seconds, Fetched: 10 row(s)
```


3.2 Loading file ShareRaceByCity.csv file on Hive Interface:

Here, we are using sg Database on Hive and creating table ShareRaceByCity.

DROP TABLE IF EXISTS ShareRaceByCity;

create table if not exists ShareRaceByCity (GeographicArea String,

City String,share_white Float,share_black Float,share_native_american Float,share_asian Float,share_hispanic Float)

ROW FORMAT DELIMITED

FIELDS TERMINATED BY ','

STORED AS TEXTFILE

tblproperties("skip.header.line.count" = "1");

```
hive> DROP TABLE IF EXISTS ShareRaceByCity;
OK
Time taken: 0.53 seconds
hive> create table if not exists ShareRaceByCity (GeographicArea String,
    > City String,share_white Float,share_black Float,share_native_american Float,share_asian Float,share_hispanic Float)
    > ROW FORMAT DELIMITED
    > FIELDS TERMINATED BY ','
    > STORED AS TEXTFILE
    > tblproperties("skip.header.line.count"="1");
OK
Time taken: 0.258 seconds
```

Then we are loading ShareRaceByCity.csv file from HDFS into ShareRaceByCity table in SG database on Hive interface in CloudxLab.

hadoop fs – put data/ShareRaceByCity.csv /user/$USER/

*load data inpath '/user/anandba*065877
/ShareRaceByCity.csv' overwrite into table sg.ShareRaceByCity;

```
hive> load data inpath '/user/anandba065877/ShareRaceByCity.csv' overwrite into table sg.ShareRaceByCity;
Loading data to table sg.shareracebycity
Table sg.shareracebycity stats: [numFiles=1, numRows=0, totalSize=1094378, rawDataSize=0]
OK
Time taken: 0.792 seconds
hive>
```

Here, we are loading top ten entries from ShareRaceByCity table:

*SELECT * FROM ShareRaceByCity limit* 10;

```
hive> SELECT * FROM ShareRaceByCity limit 10;
OK
AL      Abanda CDP      67.2    30.2    0.0     0.0     1.6
AL      Abbeville city  54.4    41.4    0.1     1.0     3.1
AL      Adamsville city 52.3    44.9    0.5     0.3     2.3
AL      Addison town    99.1    0.1     0.0     0.1     0.4
AL      Akron town      13.2    86.5    0.0     0.0     0.3
AL      Alabaster city  79.4    13.5    0.4     0.9     9.0
AL      Albertville city        75.9    1.9     0.8     0.5     27.9
AL      Alexander City city     62.2    32.0    0.2     0.9     4.8
AL      Alexandria CDP  87.4    10.2    0.3     0.5     0.9
AL      Aliceville city 22.6    74.9    0.1     0.0     1.2
```

3.3 The insights drawn out of the analysis on Hive interface in CloudxLab:

3.3.1 The dataset on which the analysis is being done:

1) PoliceKillingsUS.csv

2) ShareRaceByCity.csv.

3.3.2 Analyzing ShareRaceByCity table:

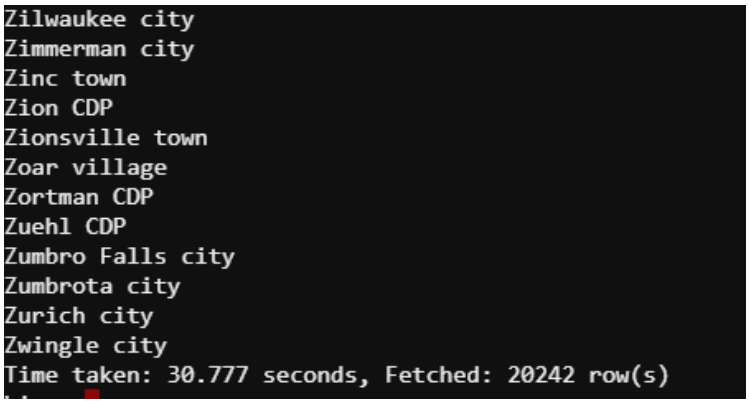
Here, we are querying to find all the cities where share of white Race is more than 70%.

SELECT City FROM ShareRaceByCity

where share_white > 70.0

GROUP BY City

limit 50000;



Insights drawn: As we can see that the query has fetched 20242 rows which indicate the count of that many cities where white Race inhabitants are more than 70%.

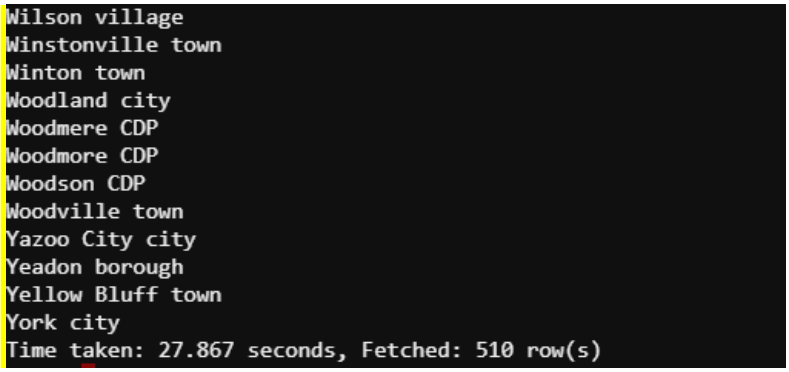
Here, we are querying to find all the cities where share of Black Race is more than 70%.

SELECT City FROM ShareRaceByCity

where share_black > 70.0

GROUP BY City

limit 50000;



Insights drawn: As we can see that the query has fetched 510 rows which indicate the count of that many cities where Black Race inhabitants are more than 70%.

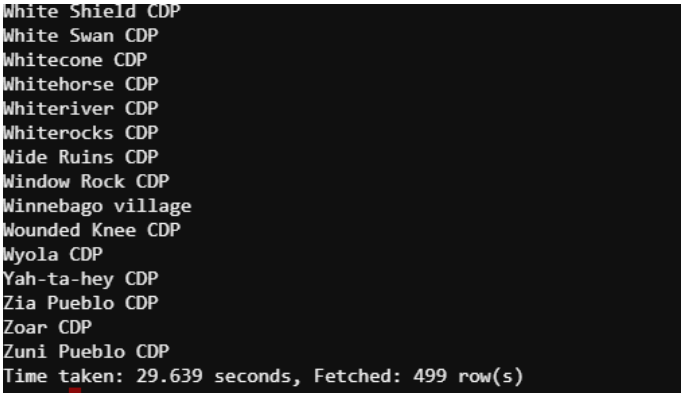
Here, we are querying to find all the cities where share of Native American Race is more than 70%.

SELECT City FROM ShareRaceByCity

where share_native_american > 70.0

GROUP BY City

limit 50000;



Insights drawn: As we can see that the query has fetched 499 rows which indicate the count of that many cities where Native American Race inhabitants are more than 70%.

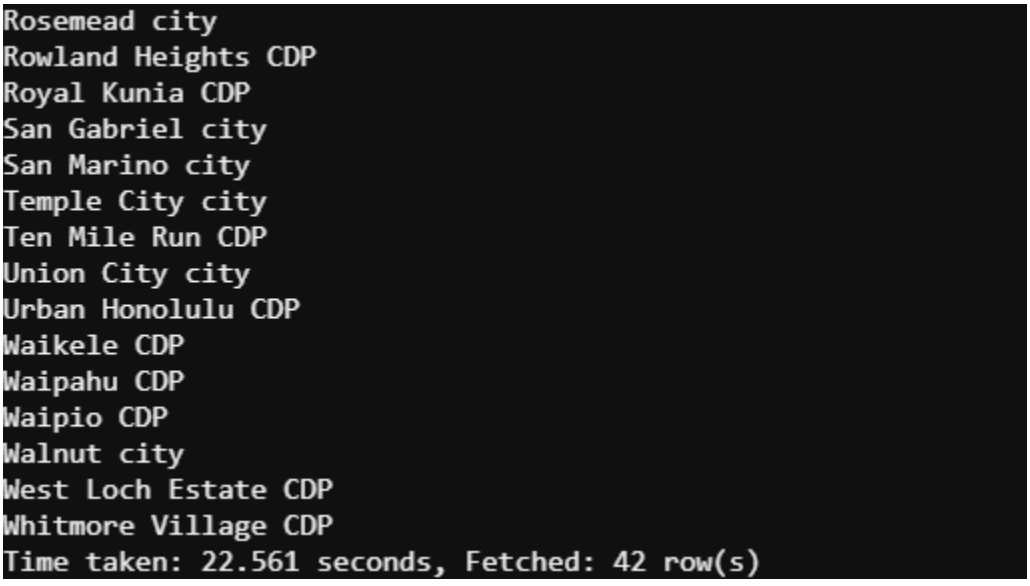
Here, we are querying to find all the cities where share of Asian Race is more than 50%.

SELECT City FROM ShareRaceByCity

where share_asian > 50.0

GROUP BY City

limit 500;



Insights drawn: As we can see that the query has fetched 42 rows which indicate the count of that many cities where Asian Race inhabitants are more than 50%.

Now we are trying to find those geographical areas where Asian are more than 50% and what are the other Race which lives more in numbers at places inhabited by Asians.

```
SELECT GeographicArea, City, share_white, share_black, share_native_american, share_asian FROM ShareRaceByCity

where share_asian > 50.0

order by share_asian desc, City

LIMIT 50;
```

CA	Rowland Heights CDP	23.5	1.6	0.4	59.8	
CA	Rosemead city	21.1	0.5	0.7	60.7	
CA	San Gabriel city	25.4	1.0	0.6	60.7	
CA	Cerritos city	23.1	6.9	0.3	61.9	
CA	Milpitas city	20.5	2.9	0.5	62.2	
CA	Cupertino city	31.3	0.6	0.2	63.3	
CA	Walnut city	23.7	2.8	0.2	63.6	
CA	Monterey Park city	19.4	0.4	0.4	66.9	
HI	Mililani Mauka CDP	17.4	2.3	0.2	50.3	
HI	Ewa Beach CDP	8.4	0.7	0.1	50.6	
HI	Halawa CDP	12.3	1.5	0.1	51.6	
HI	Keaau CDP	12.4	0.2	0.2	52.0	
HI	Kahului CDP	9.9	0.4	0.3	53.1	
HI	Pearl City CDP	16.0	2.9	0.3	53.2	
HI	Waikele CDP	16.0	2.7	0.2	54.1	
HI	Puhi CDP	18.3	0.2	0.1	54.6	
HI	Urban Honolulu CDP	17.9	1.5	0.2	54.8	
HI	West Loch Estate CDP	13.7	1.7	0.2	55.0	
HI	Lanai City CDP	14.0	0.2	0.1	56.0	
HI	Waipio CDP	11.6	1.5	0.2	56.4	
HI	Hanamaulu CDP	9.1	0.4	0.1	57.0	
HI	Aiea CDP	15.0	0.7	0.1	57.7	
HI	Eleele CDP	12.1	0.3	0.2	58.2	
HI	Royal Kunia CDP	12.0	1.8	0.2	58.4	
HI	Ewa Villages CDP	4.7	0.6	0.0	59.5	
HI	Whitmore Village CDP	4.5	0.8	0.1	64.4	
HI	Kaumakani CDP	4.5	0.0	0.1	66.9	
HI	Waipahu CDP	3.4	0.8	0.1	67.1	
NJ	Ten Mile Run CDP	31.9	11.9	0.1	50.5	
NJ	Palisades Park borough	28.9	2.0	0.3	57.8	
PA	Millbourne borough	13.7	20.1	0.6	56.3	
VA	Loudoun Valley Estates CDP	29.3	4.2	0.3	61.0	

Time taken: 21.424 seconds, Fetched: 42 row(s)

Insights drawn: we can see the following Geographical regions namely CA and HI are the regions where Asians stay more frequently with more than 50% of the population. Also White RACE population is the next higher population of race of peoples staying more frequently with Asians.

Now we are trying to find those geographical areas where Blacks are more than 85% and what is the other Race which lives more in numbers at places inhabited by Blacks.

```
SELECT GeographicArea, City, share_white, share_native_american, share_asian, share_black FROM ShareRaceByCity

where share_black > 85.0

order by GeographicArea, share_black

LIMIT 2000;
```

SC	Wilkinson Heights CDP	3.0	0.1	0.2	92.5
SC	Eastover town	5.0	0.4	0.0	93.4
SC	Gifford town	4.2	0.0	0.0	94.1
SC	Promised Land CDP	4.1	0.0	0.4	94.1
SC	Gadsden CDP	4.3	0.2	0.1	94.4
SC	Brookdale CDP	0.8	0.1	0.2	98.1
SC	Jenkinsville town	0.0	0.0	0.0	100.0
TN	"Lynchburg	NULL	2.3	0.3	95.4
TX	Prairie View city	4.9	0.2	0.4	88.6
TX	Goodlow city	8.5	0.0	0.0	90.5
VA	East Highland Park CDP	10.8	0.4	0.7	85.1
VA	Southampton Meadows CDP	8.6	0.2	0.0	85.6
VA	Cats Bridge CDP	11.8	0.0	0.0	86.9
VA	Makemie Park CDP	10.3	0.0	0.0	88.4
VA	Boston CDP	9.7	0.0	0.0	88.9
VA	Savage Town CDP	0.0	0.0	0.0	94.9
VA	Thynedale CDP	4.6	0.0	0.5	94.9
VA	Bayside CDP	4.2	0.0	0.0	95.0

Time taken: 25.819 seconds, Fetched: 225 row(s)

Insights drawn: we can see the following Geographical regions namely AL, AR, FL, GA, IL, LA, MD, MO,MS,NC,SC, and VA are the regions where Blacks stay more frequently with more than 85% of the population. Also White and Asian populations are race of peoples staying more frequently with Blacks.

Now we are trying to find those geographical areas where Native Americans are more than 90% and what is the other Race which lives more in numbers at places inhabited by Blacks.

```
SELECT GeographicArea, City, share_white, share_black, share_asian, share_native_american FROM ShareRaceByCity

where share_native_american > 90.0

order by GeographicArea, share_native_american

LIMIT 2000;
```

```
UT      Aneth CDP               1.0     0.0     0.0     97.6
UT      Tselakai Dezza CDP      0.0     0.0     0.0     98.2
UT      Halchita CDP    0.8     0.0     0.0     98.9
WA      Queets CDP      2.9     0.6     0.0     91.4
WA      Nespelem Community CDP  4.7     0.0     0.0     93.7
WI      New Odanah CDP  4.2     0.2     0.0     91.1
WI      Diaperville CDP 1.4     0.0     0.0     92.9
WI      Birch Hill CDP  3.8     0.0     0.0     93.2
WI      Keshena CDP     2.8     0.1     0.0     95.2
WI      Neopit CDP      2.0     0.0     0.0     96.7
WI      Middle Village CDP      1.1     0.0     0.0     98.9
WI      Zoar CDP        1.0     0.0     0.0     99.0
WI      Odanah CDP      0.0     0.0     0.0     100.0
WY      Fort Washakie CDP       5.6     0.0     0.0     92.5
WY      Ethete CDP      4.5     0.1     0.1     93.7
Time taken: 29.958 seconds, Fetched: 349 row(s)
```

Insights drawn: we can see the following Geographical regions namely AK, AZ, MT, NM, and SD are the regions where Native Americans stay more frequently with more than 90% of the population.

3.3.3 Analyzing PoliceKillingsUS table:

Here, we are querying to find all those who died in police killings and were less than 16 years old.

SELECT gender,count(*) FROM PoliceKillingsUS

where age <= 16

GROUP BY gender;

```
Starting Job = job_1648130833540_31329, Tracking URL = http://cxln2.c.thelab-240901.internal:8088/prox
Kill Command = /usr/hdp/2.6.2.0-205/hadoop/bin/hadoop job  -kill job_1648130833540_31329
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2023-05-23 05:00:10,142 Stage-1 map = 0%,  reduce = 0%
2023-05-23 05:00:18,435 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 4.49 sec
2023-05-23 05:00:25,655 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 8.41 sec
MapReduce Total cumulative CPU time: 8 seconds 410 msec
Ended Job = job_1648130833540_31329
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 8.41 sec   HDFS Read: 238626 HDFS Write: 9 SUCCESS
Total MapReduce CPU Time Spent: 8 seconds 410 msec
OK
F       2
M       26
Time taken: 24.644 seconds, Fetched: 2 row(s)
```

Insights drawn: we can see only 2 females and 26 males who died in police killings and were less than 16 years old.

Here, we are querying to find all those who died in police killings and were more than 16 years old.

SELECT gender,count(*) FROM PoliceKillingsUS

where age > 16

GROUP BY gender;

```
Starting Job = job_1648130833540_31330, Tracking URL = http://cxln2.c.thelab-240901.internal:8088/proxy
Kill Command = /usr/hdp/2.6.2.0-205/hadoop/bin/hadoop job  -kill job_1648130833540_31330
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2023-05-23 05:03:27,230 Stage-1 map = 0%,  reduce = 0%
2023-05-23 05:03:38,375 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 6.57 sec
2023-05-23 05:03:45,608 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 10.2 sec
MapReduce Total cumulative CPU time: 10 seconds 200 msec
Ended Job = job_1648130833540_31330
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 10.2 sec   HDFS Read: 238618 HDFS Write: 13 SUCCESS
Total MapReduce CPU Time Spent: 10 seconds 200 msec
OK
F       102
M       2327
Time taken: 30.536 seconds, Fetched: 2 row(s)
```

Insights drawn: we can see higher number of males and females having more than 16 years age dying in police killings compared to those who were less than 16 years old.

Here, we are trying to find the categories of race who were males and greater than 16 years age dying in police killings.

SELECT race,count(*)as cnt FROM PoliceKillingsUS

where age > 16 and gender = "M"

GROUP BY race

order by cnt;

```
MapReduce Total cumulative CPU time: 6 seconds 510 msec
Ended Job = job_1648130833540_31332
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 8.9 sec   HDFS Read: 238278 HDFS Write: 242 SUCCESS
Stage-Stage-2: Map: 1  Reduce: 1   Cumulative CPU: 6.51 sec   HDFS Read: 5287 HDFS Write: 39 SUCCESS
Total MapReduce CPU Time Spent: 15 seconds 410 msec
OK
N       25
O       26
A       36
        139
H       400
B       572
W       1129
Time taken: 43.758 seconds, Fetched: 7 row(s)
hive>
```

Insights drawn: we can see race W, B, H among males with more than 16 years dying more frequently in police killings compared to other races.

Here, we are trying to find the categories of race who were females and greater than 16 years age dying in police killings.

SELECT race, count(*) as cnt FROM PoliceKillingsUS

where age > 16 and gender = "F"

GROUP BY race

order by cnt;

```
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 9.18 sec   HDFS Read: 238279 HDFS Write: 235 SUCCESS
Stage-Stage-2: Map: 1  Reduce: 1   Cumulative CPU: 5.08 sec   HDFS Read: 5289 HDFS Write: 29 SUCCESS
Total MapReduce CPU Time Spent: 14 seconds 260 msec
OK
A       1
O       2
H       4
N       5
        7
B       26
W       57
Time taken: 48.053 seconds, Fetched: 7 row(s)
hive>
```

Insights drawn: we can see race B, H among Females with more than 16 years dying more frequently in police killings compared to other races.

Here, we are trying to find all those ways in which the dead tried to flee and died in police killings.

SELECT flee, count(*) as cnt FROM PoliceKillingsUS

where age > 16

GROUP BY flee

order by cnt;

```
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 6.44 sec   HDFS Read: 237653 HDFS Write: 220 SUCCESS
Stage-Stage-2: Map: 1  Reduce: 1   Cumulative CPU: 5.99 sec   HDFS Read: 5274 HDFS Write: 47 SUCCESS
Total MapReduce CPU Time Spent: 12 seconds 430 msec
OK
        60
Other   91
Foot    283
Car     367
Not fleeing     1628
Time taken: 46.748 seconds, Fetched: 5 row(s)
hive>
```

Insights drawn: we can see those who were not fleeing died the most in police killings followed by those who were driving cars and were on foot.

Here, we are trying to find the weapons carried by those who were not fleeing and were greater than 16 years and had died in police killings.

SELECT armed, count(*) as cnt FROM PoliceKillingsUS

where age > 16 and flee = 'Not fleeing'

GROUP BY armed

order by cnt desc

limit 10;

```
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 8.27 sec   HDFS Read: 238300 HDFS Write: 1958 SUCCESS
Stage-Stage-2: Map: 1  Reduce: 1   Cumulative CPU: 6.36 sec   HDFS Read: 7139 HDFS Write: 119 SUCCESS
Total MapReduce CPU Time Spent: 14 seconds 630 msec
OK
gun     897
knife   300
unarmed 95
toy weapon      77
undetermined    58
vehicle 41
machete 15
unknown weapon  13
baseball bat    8
ax      8
Time taken: 45.586 seconds, Fetched: 10 row(s)
hive>
```

Insights drawn: we can see those who were not fleeing and carried guns and knife more probably to attack died the most in police killings.

Here, we are trying to find the weapons carried by those who were driving cars and were greater than 16 years and had died in police killings.

SELECT armed, count(*) as cnt FROM PoliceKillingsUS

where age > 16 and flee = 'Car'

GROUP BY armed order by cnt desc

limit 10;

```
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 8.78 sec   HDFS Read: 238290 HDFS Write: 490 SUCCESS
Stage-Stage-2: Map: 1  Reduce: 1   Cumulative CPU: 6.58 sec   HDFS Read: 5671 HDFS Write: 121 SUCCESS
Total MapReduce CPU Time Spent: 15 seconds 360 msec
OK
gun     182
vehicle 113
unarmed 26
undetermined    18
knife   10
toy weapon      8
machete 2
hatchet 1
blunt object    1
unknown weapon  1
Time taken: 46.07 seconds, Fetched: 10 row(s)
hive>
```

Insights drawn: we can see those who were driving cars and carried guns more probably to attack died the most in police killings.

Here, we are trying to find the cities where maximum peoples had died in police killings and who were also more than 16 years old.

SELECT city, count(*) as cnt FROM PoliceKillingsUS

where age > 16 GROUP BY city

order by cnt desc

limit 10;

```
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 7.2 sec   HDFS Read: 237653 HDFS Write: 38959 SUCCESS
Stage-Stage-2: Map: 1  Reduce: 1   Cumulative CPU: 7.22 sec   HDFS Read: 44136 HDFS Write: 120 SUCCESS
Total MapReduce CPU Time Spent: 14 seconds 420 msec
OK
Los Angeles     34
Phoenix 30
Houston 25
Chicago 24
Las Vegas       19
Austin  18
Columbus        17
San Antonio     17
Miami   16
St. Louis       14
Time taken: 47.291 seconds, Fetched: 10 row(s)
hive>
```

Insights drawn: Los Angeles, Phoenix, Houston, Chicago and Las Vegas were the top 5 cities from where Maximum people died in in police killings.

4 Implementation using PIG and MAPREDUCE interface in CloudxLab

4.1 Loading file MedianHouseholdIncome2015.csv file on PIG Interface:

Here, we are loading contents of MedianHouseholdIncome2015.csv file into the object Income2015.

pig –x mapreduce

register /usr/hdp/current/pig-client/piggybank.jar;

Income2015
= load '/user/anandba065877
/MedianHouseholdIncome2015.csv' using org.apache.pig.piggybank.storage.CSVExcelStorage(',',

'NO_MULTILINE','NOCHANGE','SKIP_INPUT_HEADER') as (GeographicArea: chararray, City: chararray, Median_Income: int);

```
-bash-4.2$ pig -x mapreduce
23/05/23 06:18:14 INFO pig.ExecTypeProvider: Trying ExecType : LOCAL
23/05/23 06:18:14 INFO pig.ExecTypeProvider: Trying ExecType : MAPREDUCE
23/05/23 06:18:14 INFO pig.ExecTypeProvider: Picked MAPREDUCE as the ExecType
2023-05-23 06:18:14,366 [main] INFO  org.apache.pig.Main - Apache Pig version 0.16.0.2.6.2.0-205 (rUnversioned directory) compiled Aug 26 2017, 09:34:39
2023-05-23 06:18:14,366 [main] INFO  org.apache.pig.Main - Logging error messages to: /home/anandba065877/pig_1684822694363.log
2023-05-23 06:18:14,388 [main] INFO  org.apache.pig.impl.util.Utils - Default bootup file /home/anandba065877/.pigbootup not found
2023-05-23 06:18:14,965 [main] INFO  org.apache.pig.backend.hadoop.executionengine.HExecutionEngine - Connecting to hadoop file system at: hdfs://cxln1.c.thelab-24
0901.internal:8020
2023-05-23 06:18:15,548 [main] INFO  org.apache.pig.PigServer - Pig Script ID for the session: PIG-default-f03631c2-6766-4be2-9bc4-7a9e3c8b8314
2023-05-23 06:18:15,955 [main] INFO  org.apache.hadoop.yarn.client.api.impl.TimelineClientImpl - Timeline service address: http://cxln2.c.thelab-240901.internal:81
88/ws/v1/timeline/
2023-05-23 06:18:16,122 [main] INFO  org.apache.pig.backend.hadoop.PigATSClient - Created ATS Hook
grunt> register /usr/hdp/current/pig-client/piggybank.jar;
grunt> Income2015 = load '/user/anandba065877/MedianHouseholdIncome2015.csv' using org.apache.pig.piggybank.storage.CSVExcelStorage(',', 'NO_MULTILINE', 'NOCHANGE'
, 'SKIP_INPUT_HEADER') as (GeographicArea:chararray,City:chararray,Median_Income:int);
grunt>
```

Here, we are loading top five entries from the object Income2015 and determining the data types of the columns being used:

describe Income2015;

b = limit Income2015 5;

dump b

```
.thelab-240901.internal:8020/tmp/temp-1015061370/tmp-1804689260/_temporary/0/task__0001_m_000001
2023-05-23 06:31:50,910 [main] WARN  org.apache.pig.data.SchemaTupleBackend - SchemaTupleBackend h
2023-05-23 06:31:50,916 [main] INFO  org.apache.hadoop.mapreduce.lib.input.FileInputFormat - Total
2023-05-23 06:31:50,916 [main] INFO  org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil
(AL,Abanda CDP,11207)
(AL,Abbeville city,25615)
(AL,Adamsville city,42575)
(AL,Addison town,37083)
(AL,Akron town,21667)
grunt>
```

Here, we are filtering null values from the column Median_Income in the object Income2015 and then follow steps so as load top 5 Median incomes:

Income2015 = FILTER Income2015 BY Median_Income IS NOT NULL;

A = FOREACH Income2015 GENERATE GeographicArea,City,Median_Income;

B = ORDER A BY Median_Income DESC;

C = LIMIT B 5;

```
grunt> Income2015 = FILTER Income2015 BY Median_Income IS NOT NULL;
grunt> A = FOREACH Income2015 GENERATE GeographicArea,City,Median_Income;
grunt> B = ORDER A BY Median_Income DESC;
grunt> C = LIMIT B 5;
grunt>
```

Here, we are loading top five Median incomes and other: entries from the object Income2015:

dump C;

```
2023-05-23 11:01:37,986 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - Success!
2023-05-23 11:01:37,988 [main] INFO  org.apache.pig.data.SchemaTupleBackend - Key [pig.schematuple] was not set... will not generate code.
2023-05-23 11:01:37,992 [main] INFO  org.apache.hadoop.mapreduce.lib.input.FileInputFormat - Total input paths to process : 1
2023-05-23 11:01:37,992 [main] INFO  org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input paths to process : 1
(CO,Crisman CDP,244083)
(NY,Scarsdale village,242782)
(MD,Chevy Chase Section Three village,242500)
(CA,Hidden Hills city,241667)
(MD,Chevy Chase View town,238125)
grunt>
```

4.2 Loading file PercentagePeopleBelowPovertyLevel.csv file on PIG Interface:

Here, we are loading contents of PercentagePeopleBelowPovertyLevel.csv file into the object BelowPovertyLevel.

```
BelowPovertyLevel
= load '/user/anandba065877
/PercentagePeopleBelowPovertyLevel.csv' using org.apache.pig.piggybank.storage.CSVExcelStorage

(',','NO_MULTILINE','NOCHANGE','SKIP_INPUT_HEADER')

as (PovertyArea:chararray,City:chararray,poverty_rate:float);
```

```
grunt> BelowPovertyLevel = load '/user/anandba065877/PercentagePeopleBelowPovertyLevel.csv' using org.apache.pig.piggybank.storage.CSVExcelStorage
>> (',', 'NO_MULTILINE', 'NOCHANGE', 'SKIP_INPUT_HEADER')
>> as (PovertyArea:chararray,City:chararray,poverty_rate:float);
grunt>
```

Here, we are loading top five entries from the object BelowPovertyLevel and determining the data types of the columns being used:

```
describe BelowPovertyLevel;

b = limit BelowPovertyLevel 5;
```

```
grunt> describe BelowPovertyLevel;
BelowPovertyLevel: {PovertyArea: chararray,City: chararray,poverty_rate: float}
grunt> b = limit BelowPovertyLevel 5;
grunt>
```

```
dump b
```

```
2023-05-23 11:20:28,534 [main] INFO  org.apache.hadoop.mapreduce.lib.output.FileOutputCommitter - Saved output of task 'attempt__0001_m_00
.thelab-240901.internal:8020/tmp/temp1527963852/tmp2081189981/_temporary/0/task__0001_m_000001
2023-05-23 11:20:28,546 [main] WARN  org.apache.pig.data.SchemaTupleBackend - SchemaTupleBackend has already been initialized
2023-05-23 11:20:28,550 [main] INFO  org.apache.hadoop.mapreduce.lib.input.FileInputFormat - Total input paths to process : 1
2023-05-23 11:20:28,550 [main] INFO  org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input paths to process : 1
(AL,Abanda CDP,78.8)
(AL,Abbeville city,29.1)
(AL,Adamsville city,25.5)
(AL,Addison town,30.7)
(AL,Akron town,42.0)
grunt>
```

Here, we are filtering null values from the column poverty_rate in the object BelowPovertyLevel and then follow steps so as load top 5 poverty_rate:

BelowPovertyLevel = FILTER BelowPovertyLevel BY poverty_rate IS NOT NULL;

A = FOREACH BelowPovertyLevel GENERATE PovertyArea,City,poverty_rate;

B = ORDER A BY poverty_rate DESC;

C = LIMIT B 5;

```
grunt> BelowPovertyLevel = FILTER BelowPovertyLevel BY poverty_rate IS NOT NULL;
grunt> A = FOREACH BelowPovertyLevel GENERATE PovertyArea,City,poverty_rate;
grunt> B = ORDER A BY poverty_rate DESC;
grunt> C = LIMIT B 5;
grunt>
```

Here, we are loading top five poverty_rate and other: entries from the object BelowPovertyLevel:

dump C;

```
2023-05-23 11:33:15,700 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - Success!
2023-05-23 11:33:15,701 [main] INFO  org.apache.pig.data.SchemaTupleBackend - Key [pig.schematuple] was not set... will not generate code.
2023-05-23 11:33:15,705 [main] INFO  org.apache.hadoop.mapreduce.lib.input.FileInputFormat - Total input paths to process : 1
2023-05-23 11:33:15,705 [main] INFO  org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input paths to process : 1
(CO,Lynn CDP,100.0)
(AK,Birch Creek CDP,100.0)
(CA,Tobin CDP,100.0)
(ND,Selz CDP,100.0)
(AZ,Vaiva Vo CDP,100.0)
grunt>
```

4.3 Loading file PercentOver25CompletedHighSchool.csv file on PIG Interface:

Here, we are loading contents of PercentOver25CompletedHighSchool.csv file into the object completed_hs.

completed_hs
= load '/user/anandba065877
/PercentOver25CompletedHighSchool.csv' using org.apache.pig.piggybank.storage.CSVExcelStorage

(',','NO_MULTILINE','NOCHANGE','SKIP_INPUT_HEADER')

as (hsArea:chararray,City:chararray,percent_completed_hs:float);

```
grunt> completed_hs = load '/user/anandba065877/PercentOver25CompletedHighSchool.csv' using org.apache.pig.piggybank.storage.CSVExcelStorage
>> (',', 'NO_MULTILINE', 'NOCHANGE', 'SKIP_INPUT_HEADER')
>> as (hsArea:chararray,City:chararray,percent_completed_hs:float);
grunt>
```

Here, we are loading top five entries from the object completed_hs and determining the data types of the columns being used:

describe completed_hs;

b = limit completed_hs 5;

```
grunt> describe completed_hs;
completed_hs: {hsArea: chararray,City: chararray,percent_completed_hs: float}
grunt> b = limit completed_hs 5;
grunt>
```

dump b

```
2023-05-23 11:42:55,705 [main] INFO  org.apache.hadoop.mapreduce.lib.output.FileOutputCommitter - Saved output of task 'attempt__0001_m_000001_1' t
.thelab-240901.internal:8020/tmp/temp102922736/tmp-1458933852/_temporary/0/task__0001_m_000001
2023-05-23 11:42:55,718 [main] WARN  org.apache.pig.data.SchemaTupleBackend - SchemaTupleBackend has already been initialized
2023-05-23 11:42:55,721 [main] INFO  org.apache.hadoop.mapreduce.lib.input.FileInputFormat - Total input paths to process : 1
2023-05-23 11:42:55,721 [main] INFO  org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input paths to process : 1
(AL,Abanda CDP,21.2)
(AL,Abbeville city,69.1)
(AL,Adamsville city,78.9)
(AL,Addison town,81.4)
(AL,Akron town,68.6)
grunt>
```


Here, we are filtering null values from the column percent_completed_hs in the object completed_hs and then follow steps so as load top 5 percent_completed_hs:

completed_hs = FILTER completed_hs BY percent_completed_hs IS NOT NULL;

A = FOREACH completed_hs GENERATE hsArea,City,percent_completed_hs;

B = ORDER A BY percent_completed_hs DESC;

C = LIMIT B 5;

```
grunt> completed_hs = FILTER completed_hs BY percent_completed_hs IS NOT NULL;
grunt> A = FOREACH completed_hs GENERATE hsArea,City,percent_completed_hs;
grunt> B = ORDER A BY percent_completed_hs DESC;
grunt> C = LIMIT B 5;
grunt>
```

Here, we are loading top five percent_completed_hs and other: entries from the object completed_hs:

dump C;

```
2023-05-23 11:48:53,042 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - Success!
2023-05-23 11:48:53,043 [main] INFO  org.apache.pig.data.SchemaTupleBackend - Key [pig.schematuple] was not set... will not generate code
2023-05-23 11:48:53,046 [main] INFO  org.apache.hadoop.mapreduce.lib.input.FileInputFormat - Total input paths to process : 1
2023-05-23 11:48:53,046 [main] INFO  org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input paths to process : 1
(NC,Bowmore CDP,100.0)
(ME,Littlejohn Island CDP,100.0)
(MO,South Fork CDP,100.0)
(MO,Plevna CDP,100.0)
(NV,Paradise Valley CDP,100.0)
grunt>
```

4.3.1The dataset on which the analysis is being done:

1) MedianHouseholdIncome2015.csv

2) PercentagePeopleBelowPovertyLevel.csv

3) PercentOver25CompletedHighSchool.csv.

4.3.2 Analyzing Income_scan table:

Here we are creating Income_scan object by left outer join of Income2015 and BelowPovertyLevel objects in PIG interface in CloudxLab. To prevent duplication of columns we are only selecting column 0, column1, column2 and column 5 of Income_scan object and reassigning the same back to Income_scan object

Income_scan = JOIN Income2015 BY City LEFT OUTER, BelowPovertyLevel BY City;

Income_scan = foreach Income_scan generate $0, $1, $2, $5;

A = limit income_scan 5;

```
grunt> Income_scan =JOIN Income2015 BY City LEFT OUTER, BelowPovertyLevel BY City;
grunt> Income_scan = foreach Income_scan generate $0,$1,$2,$5;
grunt> A = limit Income_scan 5;
```

Here, we are loading top five entries from the object Income_scan:

dump A;

```
2023-05-24 03:42:37,530 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - Success!
2023-05-24 03:42:37,533 [main] INFO  org.apache.pig.data.SchemaTupleBackend - Key [pig.schematuple] was not set... will not generate c
2023-05-24 03:42:37,536 [main] INFO  org.apache.hadoop.mapreduce.lib.input.FileInputFormat - Total input paths to process : 1
2023-05-24 03:42:37,536 [main] INFO  org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input paths to process : 1
(AZ,Ajo CDP,32964,33.6)
(HI,Hawaii,69515,)
(KS,Ada CDP,39423,0.0)
(LA,Ama CDP,62689,7.7)
(OK,Bee CDP,23092,19.7)
grunt>
```

Here, we are querying to extract descending ten median incomes listings from income scan object to compare and analyze poverty rate and Median Income for people living in cities as shown below:

B = ORDER Income_scan BY Median_Income DESC;

C = LIMIT B 10;

```
grunt> B = ORDER Income_scan BY Median_Income DESC;
grunt> C = LIMIT B 10;
```

Here, we are loading descending ten Median Income and other entries from the object income scan:

dump C;

```
2023-05-24 03:55:13,374 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - Success!
2023-05-24 03:55:13,375 [main] INFO  org.apache.pig.data.SchemaTupleBackend - Key [pig.schematuple] was not set... will not generate
2023-05-24 03:55:13,398 [main] INFO  org.apache.hadoop.mapreduce.lib.input.FileInputFormat - Total input paths to process : 1
2023-05-24 03:55:13,398 [main] INFO  org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input paths to process : 1
(CO,Crisman CDP,244083,0.0)
(NY,Scarsdale village,242782,2.3)
(MD,Chevy Chase Section Three village,242500,1.8)
(CA,Hidden Hills city,241667,4.9)
(MD,Chevy Chase View town,238125,0.5)
(CO,Cherry Hills Village city,237569,2.1)
(TX,Bunker Hill Village city,236250,2.1)
(VA,Great Falls CDP,234091,1.9)
(KY,Glenview city,233036,4.8)
(NY,Muttontown village,230179,3.4)
grunt>
```

Insights drawn: As we can see, the query returned the descending ten Median Income entries, indicating that poverty rates are lower in cities with greater median incomes. The poverty rate typically increases when median income declines. Another conclusion is that depending on the general income ranges of the cities, the calculation of the poverty rate varies for each city. Also, the top ten cities with decreasing Median Income come from following geographical regions namely CO, NY, MD, CA, MD, CO, TX, VA, KY, and NY.

Here, we are querying to extract increasing ten median incomes listings from income scan object to compare and analyze poverty rate and Median Income for people living in cities as shown below:

B = ORDER Income_scan BY Median_Income ;

C = LIMIT B 10;

```
grunt> B = ORDER Income_scan BY Median_Income ;
grunt> C = LIMIT B 10;
grunt>
```

Here, we are loading increasing ten Median Income and other entries from the income scan object:

dump C;

```
2023-05-24 04:13:19,935 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - Success!
2023-05-24 04:13:19,936 [main] INFO  org.apache.pig.data.SchemaTupleBackend - Key [pig.schematuple] was not set... will not gen
2023-05-24 04:13:19,938 [main] INFO  org.apache.hadoop.mapreduce.lib.input.FileInputFormat - Total input paths to process : 1
2023-05-24 04:13:19,938 [main] INFO  org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input paths to proce
(AZ,Stanfield CDP,4511,68.5)
(CA,Delft Colony CDP,6917,100.0)
(ND,Conway city,7083,18.4)
(ND,Conway city,7083,44.4)
(ND,Conway city,7083,34.2)
(ND,Conway city,7083,32.0)
(ND,Conway city,7083,0.0)
(AZ,Lower Santan Village CDP,7175,76.4)
(VA,Union Level CDP,8015,52.5)
(OK,Badger Lee CDP,8229,93.8)
grunt>
```

Insights drawn: As we can see, the query produced ten records with increasing Median Income, showing that poverty rates often decline as median income increases. Another conclusion is that the calculation of the poverty rate differs for each city depending on the general income ranges of the cities. Also, the bottom ten cities with least Median Income come from following geographical regions namely AZ, CA, ND, ND, ND, ND, ND, AZ, VA, and OK.

Here, we are storing income scan object into HDFS storage and then loading the HDFS file back into our local storage as well.

```
STORE Income_scan INTO '/user/anandba065877/Income_scan' USING PigStorage (',');
```

```
hdfs dfs -get "/user/anandba065877/Income_scan" "data/"
```

4.3.3 Analyzing BelowPoverty_scan table:

Here we are creating BelowPoverty_scan object by right outer join of Income2015 and BelowPovertyLevel objects in PIG interface in CloudxLab. To prevent duplication of columns we are only selecting column 0, column1, column2 and column 5 of BelowPoverty_scan object and reassigning the same back to BelowPoverty_scan.

```
BelowPoverty_scan = JOIN Income2015 BY City RIGHT OUTER, BelowPovertyLevel BY City;
```

```
BelowPoverty_scan = foreach BelowPoverty_scan generate $0, $1, $2, $5;
```

```
A = limit BelowPoverty_scan 5;
```

```
grunt> BelowPoverty_scan =JOIN Income2015 BY City RIGHT OUTER, BelowPovertyLevel BY City;
grunt> BelowPoverty_scan = foreach BelowPoverty_scan generate $0,$1,$2,$5;
grunt> A = limit BelowPoverty_scan 5;
grunt>
```

Here, we are loading top five entries from the BelowPoverty_scan object:

```
dump A;
```

```
2023-05-24 04:56:58,124 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - Success!
2023-05-24 04:56:58,125 [main] INFO  org.apache.pig.data.SchemaTupleBackend - Key [pig.schematuple] was not set... will not gene
2023-05-24 04:56:58,127 [main] INFO  org.apache.hadoop.mapreduce.lib.input.FileInputFormat - Total input paths to process : 1
2023-05-24 04:56:58,127 [main] INFO  org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input paths to proces
(AZ,Ajo CDP,32964,33.6)
(KS,Ada CDP,39423,0.0)
(LA,Ama CDP,62689,7.7)
(OK,Bee CDP,23092,19.7)
(OK,Box CDP,57778,8.0)
grunt>
```

Here, we are querying to extract descending ten poverty_rate listings from BelowPoverty_scan object to compare and analyze poverty rate and Median Income for people living in cities as shown below:

```
B = ORDER BelowPoverty_scan BY poverty_rate DESC;
```

```
C = LIMIT B 10;
```

```
grunt> B = ORDER BelowPoverty_scan BY poverty_rate DESC;
grunt> C = LIMIT B 10;
grunt>
```

Here, we are loading descending ten poverty_rate listings and other entries from BelowPoverty_scan object:

dump C;

```
2023-05-24 05:04:40,541 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - Success!
2023-05-24 05:04:40,543 [main] INFO  org.apache.pig.data.SchemaTupleBackend - Key [pig.schematuple] was not set... will not ge
2023-05-24 05:04:40,545 [main] INFO  org.apache.hadoop.mapreduce.lib.input.FileInputFormat - Total input paths to process : 1
2023-05-24 05:04:40,545 [main] INFO  org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input paths to proce
(RI,Kingston CDP,71786,100.0)
(WA,Kingston CDP,47153,100.0)
(NV,Kingston CDP,43452,100.0)
(AL,Oak Grove town,40972,100.0)
(FL,Middleburg CDP,48430,100.0)
(TX,Oak Grove town,85078,100.0)
(LA,Oak Grove town,28938,100.0)
(AR,Oak Grove town,36625,100.0)
(WA,Addy CDP,9167,100.0)
(,,,100.0)
```

Insights drawn: Usually, when the median income falls, the poverty rate rises. The determination of the poverty rate, however, differs for each city depending on the general income ranges of the cities. Additionally, the bottom 10 cities with the highest poverty rates are located in the following regions: RI, WA, NV, AL, FL, TX, LA, AR, and WA.

Here, we are querying to extract increasing ten poverty rate listings from BelowPoverty_scan object to compare and analyze poverty rate and Median Income for people living in cities as shown below:

B = ORDER BelowPoverty_scan BY poverty_rate ;

C = LIMIT B 10;

```
grunt> B = ORDER BelowPoverty_scan BY poverty_rate ;
grunt> C = LIMIT B 10;
grunt>
```

Here, we are loading increasing ten poverty rate and other entries from the BelowPoverty_scan object:

dump C;

```
2023-05-24 05:17:58,778 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - Success!
2023-05-24 05:17:58,779 [main] INFO  org.apache.pig.data.SchemaTupleBackend - Key [pig.schematuple] was not set... will not gen
2023-05-24 05:17:58,782 [main] INFO  org.apache.hadoop.mapreduce.lib.input.FileInputFormat - Total input paths to process : 1
2023-05-24 05:17:58,782 [main] INFO  org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input paths to proce
(WY,Taylor CDP,77917,0.0)
(,,,0.0)
(,,,0.0)
(NJ,Hainesburg CDP,90156,0.0)
(OK,Tiawah CDP,90313,0.0)
(PA,Grier City CDP,75078,0.0)
(NC,Jefferson town,27174,0.0)
(,,,0.0)
(CA,Tipton CDP,31445,0.0)
(SC,Jefferson town,30750,0.0)
```

Insights drawn: As we can see, the search returned 10 records with poverty rate listings that ought to have been increasing However; it is clear from the data that the calculation of poverty rates differs for each city depending on the general income ranges of the cities. WY, NJ, OK, PA, NC, CA, and SC are among the states where the top 10 cities with the lowest rates of poverty are situated.

Here, we are storing BelowPoverty_scan object into HDFS storage and then loading the HDFS file back into our local storage as well.

```
STORE BelowPoverty_scan INTO '/user/anandba065877/BelowPoverty_scan' USING PigStorage (',');
```

```
hdfs dfs -get "/user/anandba065877/BelowPoverty_scan" "data/"
```

4.3.4 Analyzing completed_hs_scan table:

Here we are creating completed_hs_scan object by left outer join of completed_hs and BelowPovertyLevel objects in PIG interface in CloudxLab. To prevent duplication of columns we are only selecting column 0, column1, column2 and column 5 of completed_hs_scan object and reassigning the same back to completed_hs_scan.

```
completed_hs_scan = JOIN completed_hs BY City LEFT OUTER, BelowPovertyLevel BY City;
```

```
completed_hs_scan = foreach completed_hs_scan generate $0,$1,$2,$5;
```

```
A = limit completed_hs_scan 5;
```

```
grunt> completed_hs_scan =JOIN completed_hs BY City LEFT OUTER, BelowPovertyLevel BY City;
grunt> completed_hs_scan = foreach completed_hs_scan generate $0,$1,$2,$5;
grunt> A = limit completed_hs_scan 5;
grunt>
```

Here, we are loading top five entries from the completed_hs_scan object:

```
dump A;
```

```
2023-05-24 05:34:20,874 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - Success!
2023-05-24 05:34:20,875 [main] INFO  org.apache.pig.data.SchemaTupleBackend - Key [pig.schematuple] was not set... will not genera
2023-05-24 05:34:20,878 [main] INFO  org.apache.hadoop.mapreduce.lib.input.FileInputFormat - Total input paths to process : 1
2023-05-24 05:34:20,878 [main] INFO  org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input paths to process
(AZ,Ajo CDP,80.0,33.6)
(KS,Ada CDP,71.7,0.0)
(LA,Ama CDP,87.0,7.7)
(OK,Bee CDP,67.8,19.7)
(OK,Box CDP,85.6,8.0)
```

Here, we are querying to extract descending ten percent_completed_hs listings from completed_hs_scan object to compare and analyze poverty rate and percent_completed_hs for people living in cities as shown below:

```
grunt> B = ORDER completed_hs_scan BY percent_completed_hs DESC;
grunt> C = LIMIT B 10;
grunt>
```

Here, we are loading descending ten percent_completed_hs listings and other entries from completed_hs_scan object:

```
2023-05-24 05:40:18,548 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - Success!
2023-05-24 05:40:18,549 [main] INFO  org.apache.pig.data.SchemaTupleBackend - Key [pig.schematuple] was not set... will not gene
2023-05-24 05:40:18,551 [main] INFO  org.apache.hadoop.mapreduce.lib.input.FileInputFormat - Total input paths to process : 1
2023-05-24 05:40:18,551 [main] INFO  org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input paths to proces
(SD,Roswell CDP,100.0,42.9)
(CA,Sattley CDP,100.0,0.0)
(IA,Rossie city,100.0,11.1)
(MD,Whitehaven CDP,100.0,19.6)
(PA,Clinton CDP,100.0,5.4)
(CO,Eldorado Springs CDP,100.0,4.7)
(MT,Rosebud CDP,100.0,43.7)
(MT,Rosebud CDP,100.0,7.6)
(FL,Morriston CDP,100.0,0.0)
(PA,Clinton CDP,100.0,8.7)
```

Insights drawn: Cities with larger percentages of residents who have completed their higher secondary education typically don't appear to be associated to the poverty rate graph. Additionally, the following regions are home to the bottom 10 cities with the highest percentage of residents finishing higher secondary education: IA, MD, PA, CO, MT, MT, FL, and PA

Here, we are querying to extract increasing ten percent_completed_hs from completed_hs_scan object to compare and analyze poverty rate and percent_completed_hs for people living in cities as shown below:

B = ORDER completed_hs_scan BY percent_completed_hs;

C = LIMIT B 10;

```
grunt> B = ORDER completed_hs_scan BY percent_completed_hs;
grunt> C = LIMIT B 10;
grunt>
```

Here, we are loading increasing ten percent_completed_hs and other entries from the completed_hs_scan object:

dump C;

```
2023-05-24 05:55:46,825 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - Success!
2023-05-24 05:55:46,826 [main] INFO  org.apache.pig.data.SchemaTupleBackend - Key [pig.schematuple] was not set... will not ge
2023-05-24 05:55:46,828 [main] INFO  org.apache.hadoop.mapreduce.lib.input.FileInputFormat - Total input paths to process : 1
2023-05-24 05:55:46,828 [main] INFO  org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input paths to proc
(KY,Rosine CDP,0.0,0.0)
(TX,La Esperanza CDP,0.0,0.0)
(NM,Kingston CDP,0.0,0.0)
(NM,Kingston CDP,0.0,100.0)
(TX,Tierra Dorada CDP,0.0,100.0)
(TX,Casa Blanca CDP,0.0,60.3)
(TX,Chaparrito CDP,0.0,100.0)
(TX,Casa Blanca CDP,0.0,0.0)
(KS,Harris CDP,0.0,0.0)
(TX,Evergreen CDP,0.0,100.0)
```

Insights drawn: We can conclude from the observations listed above that cities with even the lowest rates of poverty have the lowest percentage of citizens who have completed their higher secondary school, but sometimes the results are exactly the contrary. Additionally, The bottom 10 cities with the lowest percentage of residents completing higher secondary education are located in the following regions : KY,TX,NM,NM,TX,TX,TX,TX,KS, and TX.

Here, we are storing completed_hs_scan object into HDFS storage and then loading the HDFS file back into our local storage as well.

```
STORE completed_hs_scan INTO '/user/anandba065877/completed_hs_scan' USING PigStorage (',');

hdfs dfs -get "/user/anandba065877/completed_hs_scan" "data/"
```

5 SQOOP

It is an Open source tool to efficiently transferring bulk data between Hadoop components (HDFS, Hive, H base) and structured data stores such as MySQL, Oracle, and PostgreSQL.

5.1 Sqoop Export - Hive to MySQL

Here, we are using sg Database on Hive and creating table PoliceKillingsUS_HBASE.

```
hive

use sg;

DROP TABLE IF EXISTS PoliceKillingsUS_HBASE;

create table if not exists PoliceKillingsUS_HBASE (id int,

name String)

ROW FORMAT DELIMITED

FIELDS TERMINATED BY ','

STORED AS TEXTFILE

tblproperties("skip.header.line.count" = "1");
```

```
hive> use sg;
OK
Time taken: 1.82 seconds
hive>
    > DROP TABLE IF EXISTS PoliceKillingsUS_HBASE;
OK
Time taken: 0.077 seconds
hive> create table if not exists PoliceKillingsUS_HBASE (id int,
    > name String)
    > ROW FORMAT DELIMITED
    > FIELDS TERMINATED BY ','
    > STORED AS TEXTFILE
    > tblproperties("skip.header.line.count"="1");
OK
Time taken: 0.257 seconds
```

Then we are loading PoliceKillingsUS_HBASE.csv file into PoliceKillingsUS_HBASE table.

LOAD DATA LOCAL INPATH 'data/PoliceKillingsUS_HBASE.csv' INTO TABLE PoliceKillingsUS_HBASE;

```
hive> LOAD DATA LOCAL INPATH 'data/PoliceKillingsUS_HBASE.csv' INTO TABLE PoliceKillingsUS_HBASE;
Loading data to table sg.policekillingsus_hbase
Table sg.policekillingsus_hbase stats: [numFiles=1, numRows=0, totalSize=56031, rawDataSize=0]
OK
Time taken: 0.979 seconds
```

Here, we are loading top ten entries from PoliceKillingsUS_HBASE table:

*SELECT * FROM PoliceKillingsUS_HBASE limit 10;*

SELECT id,name FROM PoliceKillingsUS_HBASE

where id = 216 or id = 292 or id = 290 or id = 213 or id = 212 or id = 210 or id = 209 or id = 208 or id = 287 or id = 207;

```
hive> SELECT id,name FROM PoliceKillingsUS_HBASE
    > where id=216 or  id=292 or id=290 or id=213 or id=212
    > or id=210 or id=209 or id=208 or id=287 or id=207;
OK
207     James Richard Jimenez
287     Richard Castilleja
208     Clifton Reintzel
209     Aaron Siler
210     Troy Ray Boyd
212     Justin Tolkinen
213     William Dean Poole
290     Eugene Smith
292     Roberto Leon
216     Andrew Charles Shipley
Time taken: 0.173 seconds, Fetched: 10 row(s)
```

Here, we are overwriting directory ' /apps/hive/warehouse/sg.db/PoliceKillingsUS_HBASE' with the contents from PoliceKillingsUS_HBASE table with the help of MAP REDUCE program running in the background

```
insert overwrite directory '/apps/hive/warehouse/sg.db
                /PoliceKillingsUS_HBASE' row format delimited fields terminated by ','
stored as textfile select id,name from PoliceKillingsUS_HBASE limit 200;
```

```
Moving data to directory /apps/hive/warehouse/sg.db/PoliceKillingsUS_HBASE
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 6.19 sec   HDFS Read: 62916 HDFS Write: 4057 SUCCESS
Total MapReduce CPU Time Spent: 6 seconds 190 msec
OK
Time taken: 22.731 seconds
hive>
```

Here, we are launching MySQL in CloudxLab

```
mysql -h cxln2.c.thelab-240901.internal -u sqoopuser -pNHkkP876rp
```

```
-bash-4.2$ mysql -h cxln2.c.thelab-240901.internal -u sqoopuser -pNHkkP876rp
Warning: Using a password on the command line interface can be insecure.
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 203668
Server version: 5.6.44 MySQL Community Server (GPL)

Copyright (c) 2000, 2019, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

Here we are using sqoopex database and creating PoliceKillingsUS_HBASE table in MySQL

```
use sqoopex;

DROP TABLE IF EXISTS PoliceKillingsUS_HBASE;

create table if not exists PoliceKillingsUS_HBASE (

id int not null,

name varchar(40) not null default 'New'

);
```

```
mysql> use sqoopex;

Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> DROP TABLE IF EXISTS PoliceKillingsUS_HBASE;
Query OK, 0 rows affected (0.01 sec)

mysql> create table if not exists PoliceKillingsUS_HBASE (
    -> id int not null,
    -> name varchar(40) not null default 'New'
    -> );
Query OK, 0 rows affected (0.07 sec)

mysql>
```

Here we are performing Sqoop Export - Hive to MySQL with the help of MAP REDUCE program running in the background:

```
        Map-Reduce Framework
                Map input records=200
                Map output records=200
                Input split bytes=181
                Spilled Records=0
                Failed Shuffles=0
                Merged Map outputs=0
                GC time elapsed (ms)=97
                CPU time spent (ms)=1070
                Physical memory (bytes) snapshot=223731712
                Virtual memory (bytes) snapshot=4589690880
                Total committed heap usage (bytes)=221773824
        File Input Format Counters
                Bytes Read=0
        File Output Format Counters
                Bytes Written=0
23/05/24 13:19:28 INFO mapreduce.ExportJobBase: Transferred 4.1416 KB in 20.1675 seconds (210.2893 bytes/sec)
23/05/24 13:19:28 INFO mapreduce.ExportJobBase: Exported 200 records.
-bash-4.2$
```

Here, we are loading top ten entries from PoliceKillingsUS_HBASE table in MySQL interface in CloudxLab:

*SELECT * FROM PoliceKillingsUS_HBASE limit* 10;

```
mysql> SELECT *FROM PoliceKillingsUS_HBASE limit 10;
+-----+------------------------+
| id  | name                   |
+-----+------------------------+
| 216 | Andrew Charles Shipley |
| 292 | Roberto Leon           |
| 290 | Eugene Smith           |
| 213 | William Dean Poole     |
| 212 | Justin Tolkinen        |
| 210 | Troy Ray Boyd          |
| 209 | Aaron Siler            |
| 208 | Clifton Reintzel       |
| 287 | Richard Castilleja     |
| 207 | James Richard Jimenez  |
+-----+------------------------+
10 rows in set (0.00 sec)

mysql>
```

5.2 Sqoop Import - MySQL to HDFS

Importing income_scan Table from sqoopex database in MYSQL to HDFS at the location /user/anandba065877/income_scan

$$sqoop\ import\ --connect\ jdbc:mysql://10.142.1.2/sqoopex\ --table\ income\_scan\ -m\ 1\ --username\ sqoopuser\ --password\ NHkkP876rp\ --target\ -dir\ /user/anandba065877/income\_scan$$

```
        Map-Reduce Framework
                Map input records=200
                Map output records=200
                Input split bytes=87
                Spilled Records=0
                Failed Shuffles=0
                Merged Map outputs=0
                GC time elapsed (ms)=54
                CPU time spent (ms)=1490
                Physical memory (bytes) snapshot=244633600
                Virtual memory (bytes) snapshot=4597121024
                Total committed heap usage (bytes)=204996608
        File Input Format Counters
                Bytes Read=0
        File Output Format Counters
                Bytes Written=5585
23/05/24 09:47:45 INFO mapreduce.ImportJobBase: Transferred 5.4541 KB in 19.1317 seconds (291.9242 bytes/sec)
23/05/24 09:47:45 INFO mapreduce.ImportJobBase: Retrieved 200 records.
```

Displaying top 10 entries of the income_scan Table imported from sqoopex database in MYSQL to HDFS at the location /user/anandba065877/income_scan

$$hdfs\ dfs\ -ls\ /user/anandba065877/income\_scan/$$

$$hdfs\ dfs\ -cat\ /user/anandba065877/income\_scan/part-m-00000|head\ -n\ 10$$

```
-bash-4.2$ hdfs dfs -ls /user/anandba065877/income_scan/

Found 2 items
-rw-r--r--   3 anandba065877 anandba065877          0 2023-05-24 09:47 /user/anandba065877/income_scan/_SUCCESS
-rw-r--r--   3 anandba065877 anandba065877       5585 2023-05-24 09:47 /user/anandba065877/income_scan/part-m-00000
-bash-4.2$ hdfs dfs -cat /user/anandba065877/income_scan/part-m-00000|head -n 10
AL,Bristow Cove CDP,10000,39.7
AL,Boykin CDP,10000,53.7
AL,Delta CDP,10000,0.0
AL,Bucks CDP,10000,0.0
AL,Cardiff town,10000,30.0
AL,Cullomburg CDP,10000,8.5
AL,Dayton town,10000,48.0
AL,Calvert CDP,10000,79.4
AL,Bon Secour CDP,10000,18.2
AL,Chunchula CDP,10000,0.0
```

5.3 Sqoop Import - MySQL to HIVE

Importing income_scan Table from sqoopex database in MYSQL to HIVE in sg database:

hdfs dfs −rm −r /user/anandba065877/income_scan/

sqoop import −−connect jdbc:mysql://cxln2.c.thelab−240901.internal/sqoopex −m 1
−−table income_scan −−hive−import −−username sqoopuser
−−password NHkkP876rp −−hive−database sg

```
23/05/24 10:19:22 INFO hive.HiveImport: Loading uploaded data into Hive

Logging initialized using configuration in jar:file:/usr/hdp/2.6.2.0-205/hive/lib/hive-
OK
Time taken: 2.428 seconds
Loading data to table sg.income_scan
Table sg.income_scan stats: [numFiles=1, numRows=0, totalSize=5585, rawDataSize=0]
OK
Time taken: 1.161 seconds
```

Displaying top 10 entries of the income_scan Table imported from sqoopex database in MYSQL to HIVE in sg Database:

hive

use sg;

*SELECT * FROM income_scan limit 10;*

DROP TABLE IF EXISTS income_scan;

```
hive> SELECT *FROM income_scan limit 10;
OK
AL      Bristow Cove CDP        10000   39.7
AL      Boykin CDP      10000   53.7
AL      Delta CDP       10000   0.0
AL      Bucks CDP       10000   0.0
AL      Cardiff town    10000   30.0
AL      Cullomburg CDP  10000   8.5
AL      Dayton town     10000   48.0
AL      Calvert CDP     10000   79.4
AL      Bon Secour CDP  10000   18.2
AL      Chunchula CDP   10000   0.0
Time taken: 0.529 seconds, Fetched: 10 row(s)
```

5.4 Sqoop Import - MySQL to HBase

Importing PoliceKillingsUS_HBASE Table from sqoopex database in MYSQL to HBase

```
sqoop import --connect jdbc:mysql://cxln2.c.thelab-240901.internal/sqoopex
--table PoliceKillingsUS_HBASE --hbase-table 'PoliceKillingsUS_HBASE'
--column-family KLD_ID_NM --username sqoopuser --hbase-create-table
--columns id,name --hbase-row-key id -m 1 --password NHkkP876rp
```

```
        Map-Reduce Framework
                Map input records=200
                Map output records=200
                Input split bytes=87
                Spilled Records=0
                Failed Shuffles=0
                Merged Map outputs=0
                GC time elapsed (ms)=110
                CPU time spent (ms)=2840
                Physical memory (bytes) snapshot=292888576
                Virtual memory (bytes) snapshot=4602724352
                Total committed heap usage (bytes)=268959744
        File Input Format Counters
                Bytes Read=0
        File Output Format Counters
                Bytes Written=0
23/05/24 13:29:05 INFO mapreduce.ImportJobBase: Transferred 0 bytes in 19.1708 seconds (0 bytes/sec)
23/05/24 13:29:05 INFO mapreduce.ImportJobBase: Retrieved 200 records.
-bash-4.2$
```

Displaying top 10 entries of the PoliceKillingsUS_HBASE Table imported from sqoopex database in MYSQL to HBase:

```
hbase shell

enable 'PoliceKillingsUS_HBASE'

scan 'PoliceKillingsUS_HBASE',{'LIMIT' => 10}

disable 'PoliceKillingsUS_HBASE'

drop 'PoliceKillingsUS_HBASE'
```

```
-bash-4.2$ hbase shell

HBase Shell; enter 'help<RETURN>' for list of supported commands.
Type "exit<RETURN>" to leave the HBase Shell
Version 1.1.2.2.6.2.0-205, r5210d2ed88d7e241646beab51e9ac147a973bdcc, Sat Aug 26 09:33:50 UTC 2017

hbase(main):001:0> enable 'PoliceKillingsUS_HBASE'
0 row(s) in 0.3100 seconds

hbase(main):002:0> scan 'PoliceKillingsUS_HBASE', {'LIMIT' => 10}
ROW                                   COLUMN+CELL
 100                                  column=KLD_ID_NM:name, timestamp=1684934943670, value=Kristiana Coignard
 101                                  column=KLD_ID_NM:name, timestamp=1684934943670, value=Demaris Turner
 102                                  column=KLD_ID_NM:name, timestamp=1684934943670, value=Jose Antonio Espinoza Ruiz
 105                                  column=KLD_ID_NM:name, timestamp=1684934943670, value=Daryl Myler
 107                                  column=KLD_ID_NM:name, timestamp=1684934943670, value=Darin Hutchins
 108                                  column=KLD_ID_NM:name, timestamp=1684934943670, value=Orlando Jude Lopez
 11                                   column=KLD_ID_NM:name, timestamp=1684934943670, value=Kenneth Joe Brown
 110                                  column=KLD_ID_NM:name, timestamp=1684934943670, value=William Campbell
 111                                  column=KLD_ID_NM:name, timestamp=1684934943670, value=Tiffany Terry
 112                                  column=KLD_ID_NM:name, timestamp=1684934943670, value=Alan James
10 row(s) in 0.0630 seconds

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

6 HBASE

It is a column family-oriented data store. It is great for storing data having 100s of millions of records or more. It is based on Google's paper on Big Table. Hbase runs on top of Hadoop meaning it stores data files in HDFS and it can process the data using Map Reduce.

Commands to perform different other operation on HDFS which were used here are:

//

hbase shell

status

table_help

create 'empANAND','personal data','professional data'

list

disable 'empANAND'

scan 'empANAND'

is_disabled 'empANAND'

enable 'empANAND'

is_enabled 'empANAND'

scan 'empANAND'

//

describe 'empANAND'

alter 'empANAND', NAME ⇒ 'personal data', VERSIONS ⇒ 5

alter 'empANAND', 'delete' ⇒ 'professional'

exists 'empANAND'

disable 'empANAND'

drop 'empANAND'

create 'empANAND', 'personal data', 'professional data'

put 'empANAND', '1', 'personal data: name', 'raju'

put 'empANAND', '1', 'personal data: city', 'hyderabad'

put 'empANAND', '1', 'professional data: designation', 'manager'

put 'empANAND', '1', 'professional data: salary', '50000'

scan 'empANAND'

//

put 'empANAND', '1', 'personal data: city', 'Delhi'

scan 'empANAND'

get 'empANAND', '1'

delete 'empANAND', '1', 'personal data: city'

scan 'empANAND'

deleteall 'empANAND', '1'

scan 'empANAND'

7 Flume

Flume is a simple, robust and extensible tool for data ingestion from various data sources into Hadoop. It is used for collecting, aggregating and transporting a large amount of streaming data such as events and logs from various sources to a centralized data store such as HDFS.

#getting a copy of sample flume conf from common data

hadoop fs − copyToLocal /data/flume/conf

Change the port to 44440 and location to hdfs://10.142.1.1/user/anandba065877/flume_webdata in HDFS

vim conf/flume.properties

#Launch the flume agent

$flume-ng\ agent\ --conf\ conf\ --conf-file\ conf/flume.properties$
$--name\ a1\ Dflume.root.logger=INFO,console$

```
:r1,state:IDLE} }} sinkRunners:{hdfs-Cluster1-sink=SinkRunner: { policy:org.apache.flume.sink.DefaultSinkProcessor@661c13f3 counterGroup:{ name:null counters:{} }
}} channels:{c1=org.apache.flume.channel.MemoryChannel{name: c1}} }
23/05/24 13:54:32 INFO node.Application: Starting Channel c1
23/05/24 13:54:32 INFO instrumentation.MonitoredCounterGroup: Monitored counter group for type: CHANNEL, name: c1: Successfully registered new MBean.
23/05/24 13:54:32 INFO instrumentation.MonitoredCounterGroup: Component type: CHANNEL, name: c1 started
23/05/24 13:54:32 INFO node.Application: Starting Sink hdfs-Cluster1-sink
23/05/24 13:54:32 INFO node.Application: Starting Source r1
23/05/24 13:54:32 INFO source.NetcatSource: Source starting
23/05/24 13:54:32 INFO instrumentation.MonitoredCounterGroup: Monitored counter group for type: SINK, name: hdfs-Cluster1-sink: Successfully registered new MBean.
23/05/24 13:54:32 INFO instrumentation.MonitoredCounterGroup: Component type: SINK, name: hdfs-Cluster1-sink started
23/05/24 13:54:32 INFO source.NetcatSource: Created serverSocket:sun.nio.ch.ServerSocketChannelImpl[/127.0.0.1:44440]
```


Open a new console and Connect to the same port that you defined in config

$nc\ localhost\ 44440$

Generate some data .Type something in the console.

```
-bash-4.2$ nc localhost 44440
Hello this is Flume interface on Hadoop in cloudxlab
OK
```

```
23/05/24 13:54:32 INFO instrumentation.MonitoredCounterGroup: Component type: CHANNEL, name: c1 started
23/05/24 13:54:32 INFO node.Application: Starting Sink hdfs-Cluster1-sink
23/05/24 13:54:32 INFO node.Application: Starting Source r1
23/05/24 13:54:32 INFO source.NetcatSource: Source starting
23/05/24 13:54:32 INFO instrumentation.MonitoredCounterGroup: Monitored counter group for type: SINK, na
23/05/24 13:54:32 INFO instrumentation.MonitoredCounterGroup: Component type: SINK, name: hdfs-Cluster1-
23/05/24 13:54:32 INFO source.NetcatSource: Created serverSocket:sun.nio.ch.ServerSocketChannelImpl[/127
23/05/24 13:58:46 INFO hdfs.HDFSSequenceFile: writeFormat = Writable, UseRawLocalFileSystem = false
23/05/24 13:58:46 INFO hdfs.BucketWriter: Creating hdfs://10.142.1.1/user/anandba065877/flume_webdata/Fl
23/05/24 13:59:17 INFO hdfs.BucketWriter: Closing hdfs://10.142.1.1/user/anandba065877/flume_webdata/Flu
23/05/24 13:59:17 INFO hdfs.BucketWriter: Renaming hdfs://10.142.1.1/user/anandba065877/flume_webdata/Fl
065877/flume_webdata/FlumeData.1684936726394
23/05/24 13:59:17 INFO hdfs.HDFSEventSink: Writer callback called.
```

#Open a new console and Check in HDFS using the following commands:

$hadoop\ fs\ -ls\ '/user/anandba065877/flume\_webdata/'$

$hadoop\ fs\ -cat\ '/user/anandba065877/flume\_webdata/FlumeData.1684937136897'$

```
-bash-4.2$ hadoop fs -ls '/user/anandba065877/flume_webdata/'
Found 1 items
-rw-r--r--   3 anandba065877 anandba065877        167 2023-05-24 14:06 /user/anandba065877/flume_webdata/FlumeData.1684937136897
-bash-4.2$ hadoop fs -cat '/user/anandba065877/flume_webdata/FlumeData.1684937136897'
SEQ!org.apache.hadoop.io.LongWritable"org.apache.hadoop.io.BytesWritable       e
                                                                                  "P|(NHello this is Flume interface on Hadoop in cloudxlab-
-bash-4.2$
```