

A BEST APPROACH TO RANGE-AGGREGATE QUERIES IN BIG DATA ENVIRONMENT

Big data for better solution

*Version* *<1.0>*

*Date <29/11/2016>*

*Author: Janakiraman*

**Project Definition**

**What is BIG DATA?**

Big data is a term that describes the large volume of data – both structured and unstructured – that inundates a business on a day-to-day basis. But it’s not the amount of data that’s important. It’s what organizations do with the data that matters. Big data can be analysed for insights that lead to better decisions and strategic business moves. Big data is changing the way people within organizations work together. It is creating a culture in which business and IT leaders must join forces to realize value from all data. Insights from big data can enable all employees to make better decisions—deepening customer engagement, optimizing operations, preventing threats and fraud, and capitalizing on new sources of revenue. But escalating demand for insights requires a fundamentally new approach to architecture, tools and practices.

**Background**

Range-aggregate queries execute the aggregate function on number of columns with simultaneously in a given query range. The processing of range-aggregate queries on large amount of data takes the long time to provide the accurate result.

**Business Case**

Huge amount of data being generated by everything around us at all times. Every digital process and social media exchange produces it. Industries struggling with handle this amount of data. So we made it as a business to give accurate data solution.

**Project scope**

To increase the processing speed of range-aggregate query and to achieve scalability. The main aim of this project is handling data efficiently for the aggregate functions which are fired on one or more column on the big data.

**Desired output for project**

This project`s output comes with clear scenarios, use cases, conditions and filtration that has applied on each phases. So it should be clear vision about what we expected in the particular range.

**Tools and Techniques**

Various complex tools and mind crashing techniques we applied for this project are…

Map Reduce

HDFS

Hive

Pig

Sqoop

**Hadoop Framework**

The Apache Hadoop software library is a framework that allows for the distributed processing of large data sets across clusters of computers using simple programming models. It is designed to scale up from single servers to thousands of machines, each offering local computation and storage. Rather than rely on hardware to deliver high-availability, the library itself is designed to detect and handle failures at the application layer, so delivering a highly-available service on top of a cluster of computers, each of which may be prone to failures.

The project includes these modules:

**Hadoop Common**: The common utilities that support the other Hadoop modules.

**Hadoop Distributed File System (HDFS)**: A distributed file system that provides high-throughput access to application data.

**Hadoop YARN**: A framework for job scheduling and cluster resource management.

**Hadoop MapReduce**: A YARN-based system for parallel processing of large data sets.

**benefits**

Some of the reasons organizations use Hadoop is its’ ability to store, manage and analyse vast amounts of structured and unstructured data quickly, reliably, flexibly and at low-cost.

**Scalability and Performance** – distributed processing of data local to each node in a cluster enables Hadoop to store, manage, process and analyse data at petabyte scale.

**Reliability**– large computing clusters are prone to failure of individual nodes in the cluster. Hadoop is fundamentally resilient – when a node fails processing is re-directed to the remaining nodes in the cluster and data is automatically re-replicated in preparation for future node failures.

**Flexibility**– unlike traditional relational database management systems, you don’t have to create structured schemas before storing data. You can store data in any format, including semi-structured or unstructured formats, and then parse and apply schema to the data when read.

**Low Cost**– unlike proprietary software, Hadoop is open source and runs on low-cost commodity hardware.

**Hive**

The Apache Hive data warehouse software facilitates reading, writing, and managing large datasets residing in distributed storage using SQL. Structure can be projected onto data already in storage. A command line tool and JDBC driver are provided to connect users to Hive.

**Benefits**

**Time**-It take very less time to write Hive Query compared to Map Reduce code. For example, the word count problem which takes around 50 lines of code can be written in 5 lines in Hive. So, you save time.

**Easy**-It is very easy to write query involving joins (if there are few joins) in Hive.

**Maintenance**-It has very low maintenance and is very simple to learn & use (low learning curve).

**Pig**

**Apache Pig** is a platform for analysing large data sets that consists of a high-level language for expressing data analysis programs, coupled with infrastructure for evaluating these programs. The salient property of Pig programs is that their structure is amenable to substantial parallelization, which in turns enables them to handle very large data sets.

**Benefits**

**Ease of programming.** It is trivial to achieve parallel execution of simple, "embarrassingly parallel" data analysis tasks

**Optimization opportunities.** The way in which tasks are encoded permits the system to optimize their execution automatically, allowing the user to focus on semantics rather than efficiency.

**Extensibility.** Users can create their own functions to do special-purpose processing.

**Sqoop**

Sqoop is a tool designed to transfer data between Hadoop and relational database servers. It is used to import data from relational databases such as MySQL, Oracle to Hadoop HDFS, and export from Hadoop file system to relational databases.

**Database for US Citizens Details**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Age** | **Education** | **Marital Status** | **Gender** | **Tax Filer Status** | **Income** | **Parents** | **Country Of Birth** | **Citizenship** | **Weeks Worked** |
| 73 | High school graduate | Widowed | Female | Non filer | 1700.09 | Not in universe | United-States | Native- Born in the United States | 0 |

**List of use cases**

* Total count of male/female based on education.
* Total count of employed/unemployed based on education.
* Total count for people in age range of 18-25 based on education.
* Tax analysis total and gender wise
* Per Capita Income (PCI) analysis consolidated, gender wise and category wise.
* Total amount dispensed on pension in x year(s)
* Total amount dispensed on scholarship in current year
* For given age range employable female widowed and divorced count
* Voter(s) count in x year(s)
* Senior Citizen(s) count in x year(s)
* Total number of Male/Female
* Citizens and immigrants count for employed lot
* Degree wise count for employability
* Customer base analysis
* Non-US citizen(s) tax filer status
* Country of birth wise count for US citizenship

**Total count of male/female based on education:**

Consider Stanford University, they are trying to offer education in less fee in various categories, but they don`t know about who are all looking for Higher education, who are all looking for Bachelor degree, who are all looking for Master degree and who are all looking for Research in a field. This scenario will help them to filter peoples based on education and they can easily offer their courses.

**Used Technologies:** HIVE and PIG

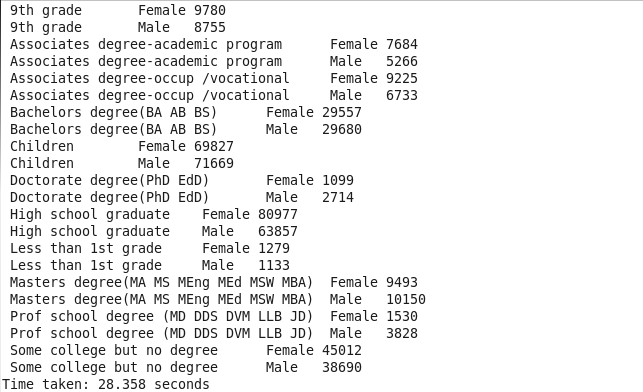
**Input:** Total US Citizens Details.

**Expected output:** Total count male and female based on their education.

**Hive**:

Query: select edu,gen, COUNT(\*) Total from final\_census1 group by edu,gen;

Output:



**PIG:**

**Script:**

step1 = load '/user/cloudera/Census\_Records.json' using JsonLoader('Age:int,Education:chararray,MartialStatus:chararray,Gender:chararray,TaxFilerStatus:chararray,Income:float,Parents:chararray,CountryOfBirth:chararray,Citizenship:chararray,WeeksWorked:chararray');

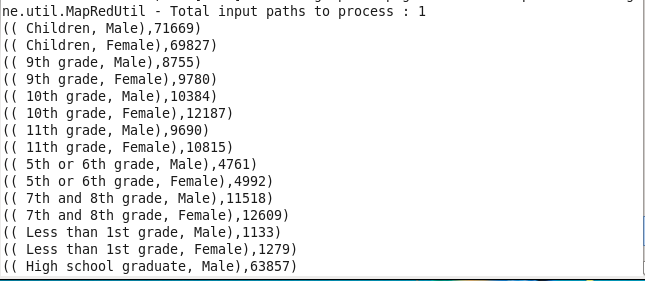
step2 = foreach step1 generate $1 as Edu,$3 as Gen;

step3 = group step2 by ($0,$1);

step4 = foreach step3 generate group,COUNT(step2.Gen);

dump step4;

Output**:**



**Total count of employed/unemployed based on education:**

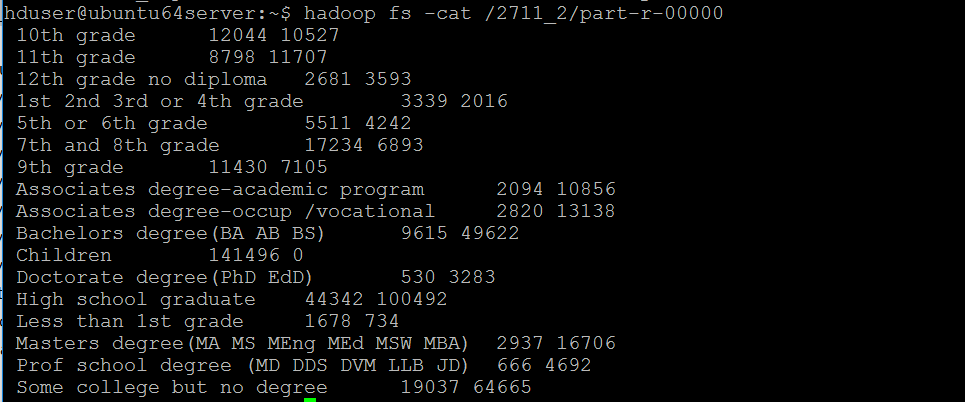
Consider, Microsoft corporation need employees for different categories like security, office staff, and software engineer as fresher and software engineer in experienced. But they don`t know about ho w many peoples are employed and unemployed. So this scenario will help them to filter peoples based on employability, and based on their education they can provide related jobs.

**Used Technologies:** HIVE and PIG Advance MapReduce

**Input:** Total US Citizens Details.

**Expected output:** Total count employed and unemployed based on their education.

Advance MapReduce:



PIG:

**Employed Counts:**

**Script:**

step1 = load '/user/cloudera/Census\_Records.json' using JsonLoader('Age:int,Education:chararray,MartialStatus:chararray,Gender:chararray,TaxFilerStatus:chararray,Income:float,Parents:chararray,CountryOfBirth:chararray,Citizenship:chararray,WeeksWorked:int');

step2 = foreach step1 generate $1 as Edu,$9 as ww;

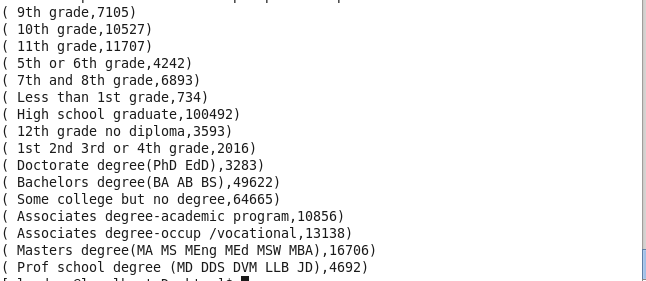
step3 = filter step2 by $1>0;

step4 = group step3 by $0;

step5 = foreach step4 generate group,COUNT($1);

dump step5;

Output:



**Unemployed Counts:**

**Script:**

step1 = load '/user/cloudera/Census\_Records.json' using JsonLoader('Age:int,Education:chararray,MartialStatus:chararray,Gender:chararray,TaxFilerStatus:chararray,Income:float,Parents:chararray,CountryOfBirth:chararray,Citizenship:chararray,WeeksWorked:int');

step2 = foreach step1 generate $1 as Edu,$9 as ww;

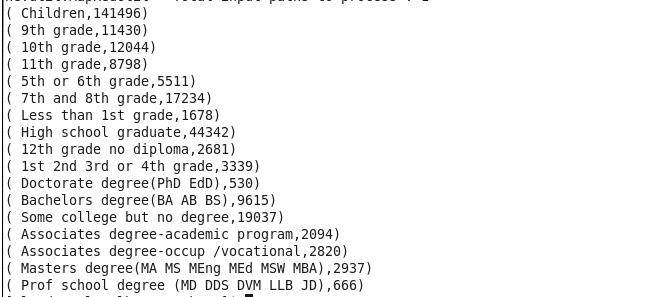
step3 = filter step2 by $1==0;

step4 = group step3 by $0;

step5 = foreach step4 generate group,COUNT($1);

dump step5;

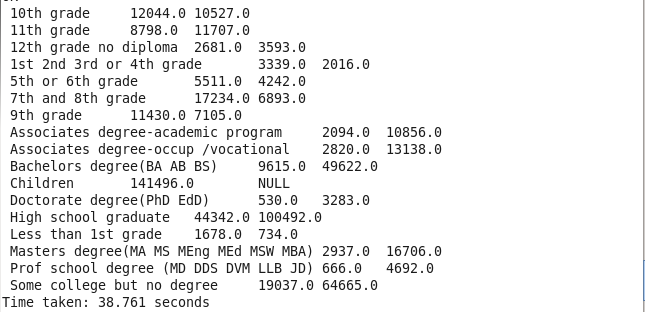
Output**:**



HIVE:

Query**:** select edu, SUM(CASE when ww <=0 then '1' else null END) as Employed , SUM(CASE when ww >0 then '1' else null END) as Unemployed from final\_census1 group by edu;

Output:



**Total count for people in age range of 18-25 based on education:**

Consider, US government need 5000 peoples for their military defence and those peoples must in 18-25 age range. So in these cases this scenario will help US government to restrict peoples who are all between 18-25 ages.

**Used Technologies:** HIVE and PIG

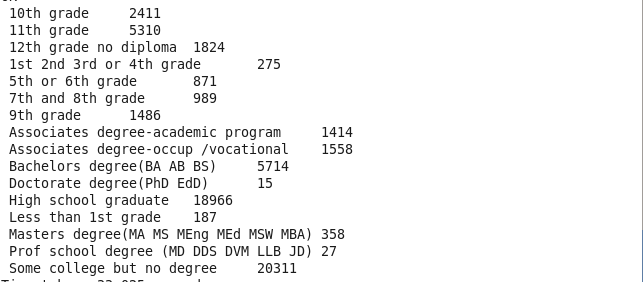
**Input:** Total US Citizens Details.

**Expected output:** Total number of peoples based on their age.

**HIVE:**

Query: select edu,count(\*) as total\_peoples from the final\_census where age between 18 and 25 group by edu;

Output:



**PIG:**

Script**:**

a = load '/user/cloudera/Census\_Records.json' using JsonLoader('age:int,edu:chararray,mar:chararray,gen:chararray,tax:chararray,income:chararray,parent:chararray,country:chararray,citizen:chararray,ww:int');

b = foreach a generate age,edu;

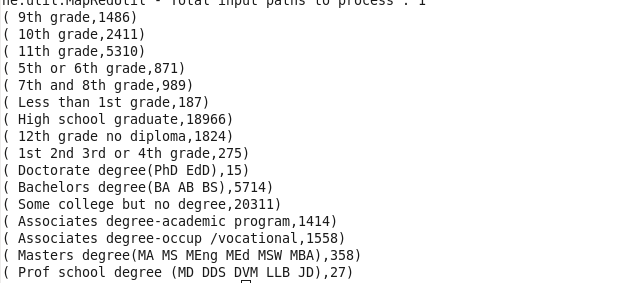
c = filter b by age>17 and age<26;

j = group c by edu;

d = foreach j generate group,COUNT(c.age);

dump d;

Output:



**Tax analysis total and gender wise:**

Consider, Income Tax Department want to know total tax filers and gender wise tax filers, then this scenario will help them to filter total tax filers and gender wise tax filers.

**Used Technologies:** HIVE

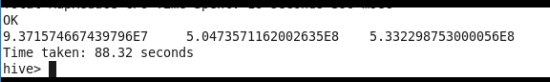
**Input:** Total US Citizens Details.

**Expected output:** Total count of tax filers and gender wise tax filers.

**HIVE**:

Query: select SUM(income\*tax\_pct) as total,SUM(CASE f.gender when ‘ Male’ then income END) as taxmale,SUM(CASE f.gender when ‘ Female’ then income END) as taxfemale from final\_census f join genwisetax t on (f.gender=t.gender) where f.income between t.minamount and t,maxamount;

Output:

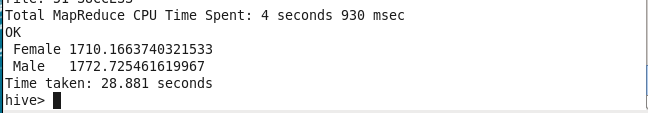


**Per Capita Income (PCI) analysis consolidated, gender wise and category wise:**

**HIVE:**

Query: select gen,sum(income)/count(gen) from final\_census group by gen;

Output:



**PIG:**

Script**:**

a = load '/user/cloudera/Census\_Records.json' using JsonLoader('age:int,edu:chararray,mar:chararray,gen:chararray,tax:chararray,income:float,parent:chararray,country:chararray,citizen:chararray,ww:int');

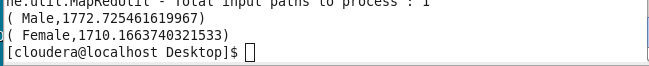
b = foreach a generate gen,income;

c = group b by gen;

d = foreach c generate group,SUM(b.income)/COUNT(b.gen);

dump d;

Output:



**Social Welfare:**

Consider, Magicbususa is the top most Non-Government Organization in US. Magicbususa ready to offer pension for senior citizens in US and scholarship for students who are all don`t have their both parents and who are all have mother only and who are all have father only. And Magicbususa takes more care on woman who are all employable and who are all widowed and who are all divorced. Magicbususa also want to know how much amount dispensed in pension, scholarship, widowed, divorced, and unemployable categories. So this scenario will definitely help them to filter peoples in several categories.

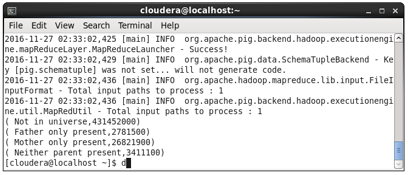
**Used Technologies:** Advance MapReduce, PIG and HIVE

**Input:** Total US Citizens Details.

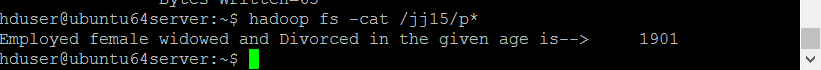
**Total amount dispensed on pension in x year(s):**



**Total amount dispensed on scholarship in current year:**

****

**For given age range employable female widowed and divorced count:**



**Process for the future:**

Consider, US government try to take a survey about voters and senior citizens after 5 years, this scenario is very suitable for take a survey.

**Used Technologies:** HIVE

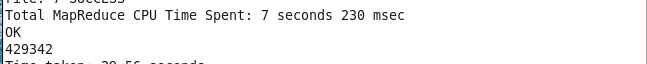
**Input:** Total US Citizens Details.

**Plan for voter(s):**

**Query:**

select COUNT(\*) as Total\_Voters from final\_census where age+(${hiveconf:year}-YEAR(from\_unixtime(unix\_timestamp())))>=18;

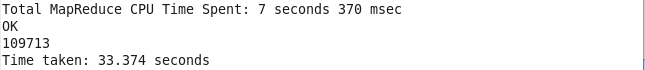
Output:



**Senior Citizen(s) count in x year(s):**

Query: select COUNT(\*) as Total\_Senior\_Citizen from final\_census where age+(${hiveconf:year}-YEAR(from\_unixtime(unix\_timestamp())))>=60;

Output:



**Total number of Male/Female:**

**Input:** Total US Citizens Details.

**Expected output:** Total count of male and female.

Query: select gender, COUNT(\*) as Total from final\_census group by gender;

Output:



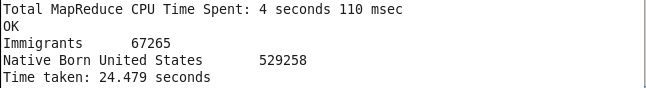
**Citizens and immigrants count for employed lot:**

**Input:** Total US Citizens Details.

**Expected output:** Total count citizen and immigrants.

Query: select citizenship, COUNT(\*) from ( select CASE citizenship when ' Native- Born in the United States' then 'Native Born United States' else 'Immigrants' END citizenship from final\_census) a group by citizenship;

Output:



**Degree wise count for employability:**

Consider, Google corporation need employees for different categories like security, office staff, and software engineer as fresher and software engineer in experienced. But they don`t know about how many peoples are unemployed. So this scenario will help them to filter peoples based on unemployed as per education, and based on their education they can provide related jobs.

**Used Technologies:** HIVE, PIG and Advance MapReduce

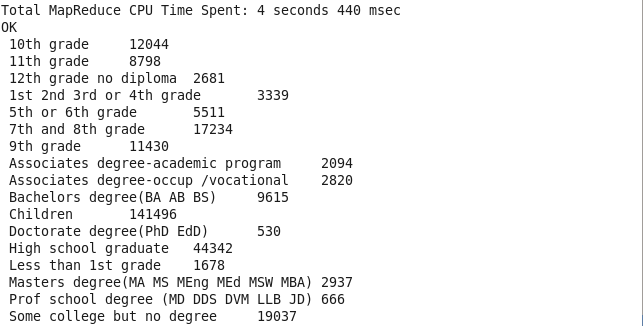
**Input:** Total US Citizens Details.

**Expected output:** Degree wise count for employability.

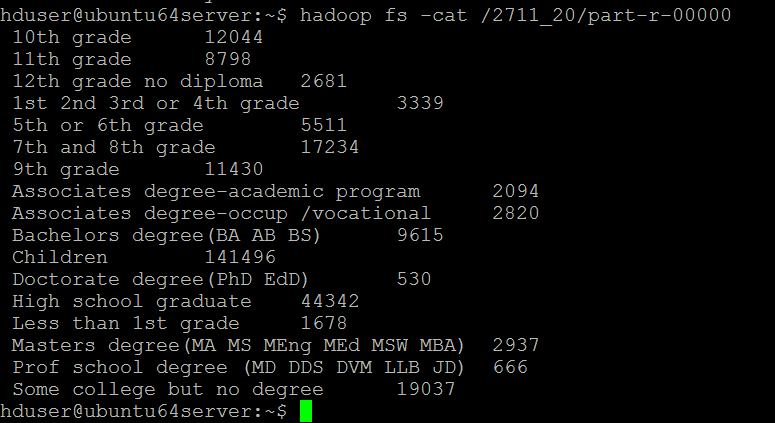
**HIVE**:

Query: select edu,COUNT(\*) from final\_census where ww=0 group by edu;

Output:



**Advance MapReduce**:



**PIG**:

a = load '/user/cloudera/Census\_Records.json' using JsonLoader('age:int,edu:chararray,mar:chararray,gen:chararray,tax:chararray,income:float,parent:chararray,country:chararray,citizen:chararray,ww:int');

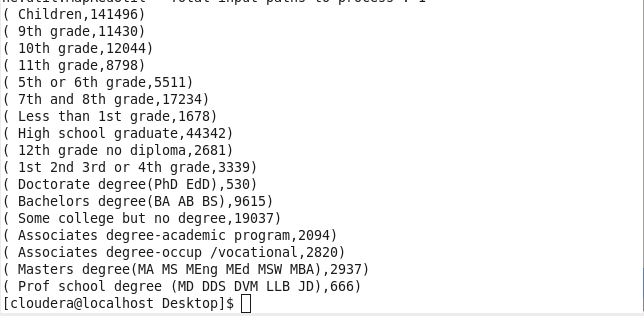
b = foreach a generate $1,$9;

c = filter b by ww==0;

d = group c by $0;

e = foreach d generate group,COUNT(c.$0);

dump e;



**Customer base analysis:**

Consider, Amazon Company made a hair gel a product, they try to sell this. This product mostly focused on adults and who have their income more than $1500. So based on US citizenship Amazon want to know how many adults are there and their incomes. This scenario will help Amazon to filter peoples based on age, income and gender wise.

**Used Technologies:** PIG

**Input:** Total US Citizens Details.

**Expected output:** Gender wise adults and income wise greater than $1500.

PIG:

Script:

a = load '/user/cloudera/Census.json' using JsonLoader('age:int,edu:chararray,mar:chararray,gen:chararray,tax:chararray,income:long,parent:chararray,country:chararray,citizen:chararray,ww:int');

b = foreach a generate age,gen,income;

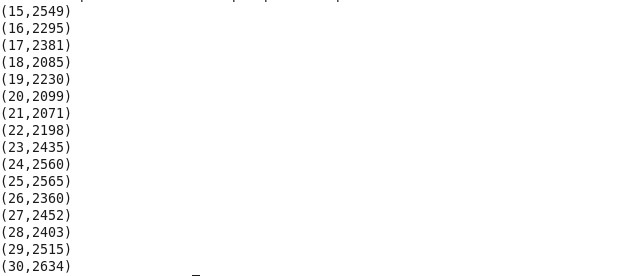
d = filter b by ((gen==' Male' and income>1500) and (age>14 and age<31)) ;

j = group d by age;

k = foreach j generate group,COUNT(d.age);

dump k;

Output:



**Non-US citizen(s) tax filer status:**

Consider, US government want to know who all Non-US citizens are paying tax in US. This scenario will help government to filter Non-US tax filers.

**Used Technologies:** HIVE

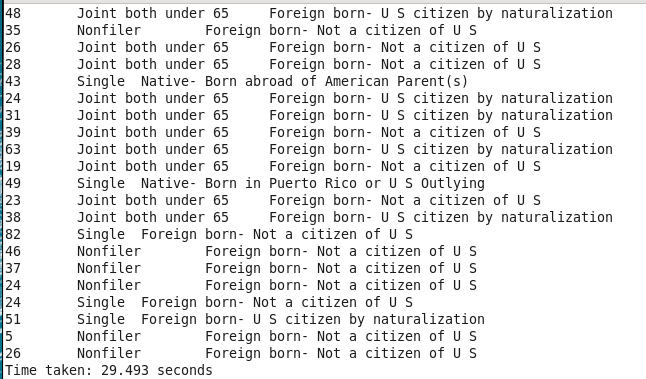
**Input:** Total US Citizens Details.

**Expected output:** Tax filers of Non-US citizens.

**HIVE**:

Query: select age,tax,citizen from final\_census where citizen not in(‘ Native- Born in the Unites States’);

Output:



**Country of birth wise count for US citizenship:**

Consider, Indian government offer Rs.50,000 for their native peoples who are all struggling in United States. If the Indian government don`t know any idea about how many peoples are settled in United States. So in this situation this scenario will help them to figure out. And this scenario will also help to United States government to keep track on birth wise other country citizens.

**Used Technologies:** HIVE

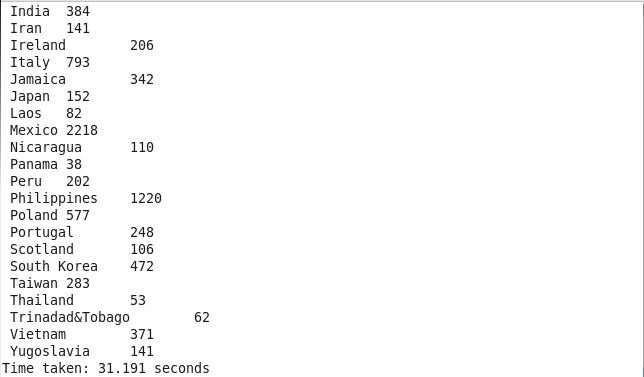
**Input:** Total US Citizens Details.

**Expected output:** Country of birth wise count for US citizenship

**HIVE:**

Query: select cntry,count(citizen) from final\_census where citizen=’ Foreign born- U S citizen by naturalization’ group by cntry;

Output:



**Software and Hardware requirement**

* **Operating System :** Windows 7,8,10 and Mac.
* **Supporting software’s:** Ubuntu,putty, Oracle VM VirtualBox,WinSCP.
* **RAM :** Minimum 4GB.

**Conclusion**

With these different scenarios I can find accurate solution with a huge dataset in different technologies. From this project I have ability to handle tools and techniques from Hadoop.