1. **Project Implementation**

**5.1: Conversion of JSON file format to CSV file format.**

*Tool used: Apache Pig.*

**Step 1:** Loading JSON file into Pig Grunt Shell.

loadJson = LOAD '*/Project/Census/Input/RawInput/sample.dat'* USING JsonLoader ('Age:INT,Education:chararray,MaritalStatus:chararray,Gender:chararray,TaxFilerStatus:chararray,Income:Double,Parents:chararray,CountryOfBirth:chararray,Citizenship:chararray,WeeksWorked:chararray');

**Step 2:** Storing Converted csv file into HDFS

STORE loadJson INTO '*/Project/Census/Input/csv'* USING PigStorage(',');

**RAW INPUT:**

{"Age": 73,"Education": " High school graduate","MaritalStatus": " Widowed","Gender": " Female","TaxFilerStatus": " Nonfiler","Income": 1700.09,"Parents": " Not in universe","CountryOfBirth": " United-States","Citizenship": " Native- Born in the United States","WeeksWorked": 0}

**SAMPLE OUTPUT:**

73, High school graduate, Widowed, Female, Nonfiler,1700.09, Not in universe, United-States, Native- Born in the United States,0

**5.2: Joining the datasets.**

*Tool used: Apache Pig.*

**Step 1:** Loading Census data file into Pig Grunt Shell. (In csv format)

census = load '*/Project/Census/Input/csv/part-m-00000’* using PigStorage(',') as (Age,Education,MartialStatus,Gender,TaxFilersStatus,Income:DOUBLE,Parents,CountryOfBirth,Citizenship,WeeksWorked:INT);

**Step 2:** Generating row keys for each row.

A = RANK census; --generating row keys

**Step 3:** Loading age group dataset into Pig Grunt Shell.

age = load '*/Project/Census/Input/RawInput/agegroup.dat'* using PigStorage('\t') as (Age, AgeWise);

**Step 4:** Joining two datasets by age column.

joining = JOIN A BY Age, age BY Age;

**Step 5:** Formatting the columns of joined datasets.

census\_data = FOREACH joining GENERATE ($0,$1,$12,$2,$3,$4,$5,$6,$7,$8,$9,$10);

**Step 6:** Storing the formatted datasets into HDFS.

STORE census\_data INTO '*/Project/Census/Input/WithoutTax’* USING PigStorage(‘,’);

**SAMPLE OUTPUT:**

1385,0,infants, Children, Never married, Female, Nonfiler,26590.6, Both parents present, United-States, Native- Born in the United States,0

1462,0,infants, Children, Never married, Female, Nonfiler,1606.46, Mother only present, United-States, Native- Born in the United States,0

1675,0,infants, Children, Never married, Male, Nonfiler,2195.99, Both parents present, United-States, Native- Born in the United States,0

**5.3: Finding Annual Income for each individual citizen.**

*Tool used: Apache Pig.*

**Step 1:** Loading the data from HDFS.

income = LOAD '*/Project/Census/Input/WithoutTax'* USING PigStorage(',') AS(ID,Age,AgeGroup,Education,MartialStatus,Gender,TaxFilersStatus,Income:DOUBLE,

Parents,CountryOfBirth,Citizenship,WeeksWorked);

**Step 2:** Finding annual income.

annual = FOREACH income GENERATE $0,$1,$2,$3,$4,$5,$6,ROUND\_TO($7\*12,2),$8,$9,$10,$11;

**Step 3:** Storing the processed data into HDFS.

STORE annual INTO '*/Project/Census/Input/annual\_income/*' USING PigStorage(',');

**SAMPLE OUTPUT:**

982,60,middle-aged, 1st 2nd 3rd or 4th grade, Separated, Male, Nonfiler,41678.88, Not in universe, United-States, Native- Born in the United States,0

972,49,middle-aged, 11th grade, Married-spouse absent, Female, Nonfiler,8965.32, Not in universe, ?, Native- Born abroad of American Parents,0

965,15,Teenager, 9th grade, Never married, Male, Nonfiler,30159.48, Both parents present, United-States, Native- Born in the United States,0

**5.4: Finding number of tax-filers & maximum income based on educational background.**

*Tool Used: Apache Pig.*

**Step 1:** Loading the Data into Pig Grunt Shell.

census = load '/home/bala/Desktop/census\_data/annual\_income/finally/FinalInput' using PigStorage(',') as (ID,Age:INT, AgeGroup, Education, MartialStatus, Gender, TaxFilersStatus, Income:Double, Parents, CountryOfBirth, Citizenship, WeeksWorked:INT);

**Step 2:** Finding count of Tax-filers based on their Tax-Filer-Status.

tax = GROUP census BY TaxFilersStatus;

tax1 = FOREACH tax GENERATE GROUP, COUNT(census.TaxFilersStatus);

--STORE tax1 INTO '/Project/Census/Pig/TaxFilersCount' USING PigStorage(',');

DUMP tax1;

**OUTPUT:**

( Single,385)

( Nonfiler,705)

( Joint both 65+,71)

( Head of household,88)

( Joint both under 65,721)

( Joint one under 65 & one 65+,30)

**Step 3:** Finding educational base annual income.

edu = GROUP census BY Education;

edu\_income = FOREACH edu GENERATE GROUP, SUM(census.Income);

edu\_based\_income = ORDER edu\_income BY $1 DESC;

--STORE edu\_based\_income INTO '/Project/Census/Pig/EduBasedIncome' USING PigStorage(',');

DUMP edu\_based\_income;

**OUTPUT:**

( High school graduate,887049)

( Children,743548)

( Some college but no degree,544230)

( Bachelors degree(BA AB BS),328712)

( 10th grade,140397)

( 7th and 8th grade,128254)

( 11th grade,117724)

( Masters degree(MA MS MEng MEd MSW MBA),108873)

( Associates degree-occup /vocational,99498)

( 9th grade,93312)

( Associates degree-academic program,82022)

( Prof school degree (MD DDS DVM LLB JD),41057)

( 5th or 6th grade,33328)

( Doctorate degree(PhD EdD),29304)

( 1st 2nd 3rd or 4th grade,28596)

( 12th grade no diploma,28487)

( Less than 1st grade,21132)

**5.5: Finding average income or Per-capita Income based on the citizenship & average or Per capita income based on country of born.**

*Tool Used: Apache Hive.*

**Step 1:** Creating new database in Hive.

CREATE DATABASE Project;

USE Project;

**Step2:** Creating new table in the database.

CREATE EXTERNAL TABLE census(id INT, Age INT, AgeGroup STRING, Education STRING, martial STRING, gender STRING, taxfilerstatus STRING, Income DOUBLE, parents STRING, citizenship STRING, nativeBorn STRING, WeeksWorked STRING)

ROW FORMAT DELIMITED

FIELDS TERMINATED BY ','

STORED AS TEXTFILE;

**Step 3:** Loading processed CSV file from HDFS into the created table.

LOAD DATA LOCAL INPATH '*/Project/Census/Input/annual\_income/*' OVERWRITE INTO TABLE census;

**Step 4:** Finding avg income based on country of born:

SELECT nativeBorn,ROUND(AVG(Income),2) AS AverageIncome FROM census GROUP BY nativeBorn ORDER BY AverageIncome DESC;

**OUTPUT:**

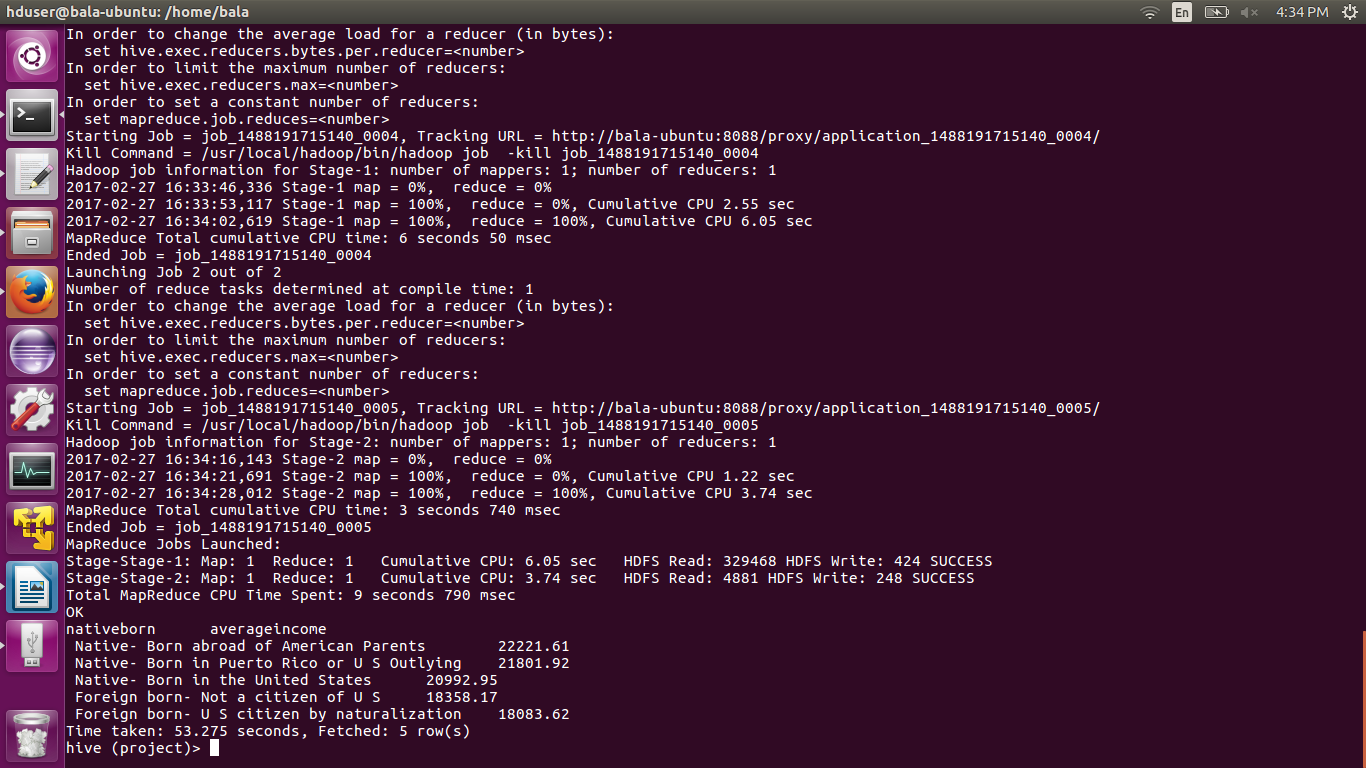
Native- Born abroad of American Parents 22221.61

Native- Born in Puerto Rico or U S Outlying 21801.92

Native- Born in the United States 20992.95

Foreign born- Not a citizen of U S 18358.17

Foreign born- U S citizen by naturalization 18083.62



**Step 4: Finding Average income based on citizenship**

SELECT citizenship,ROUND(AVG(Income),2) AS AverageIncome FROM census GROUP BY citizenship ORDER BY AverageIncome DESC;

**OUTPUT:**

France 51161.16

Portugal 43908.36

Cambodia 37148.4

Japan 37023.36

Scotland 25408.56

Peru 24216.72

? 23939.75

Canada 23916.48

Guatemala 23139.6

England 22264.06

Outlying-U S Guam USVI etc 22144.62

Germany 21923.28

Puerto-Rico 21733.38

Honduras 21262.56

United-States 20992.95

Trinadad&Tobago 20935.5

Hungary 20852.4

Vietnam 20828.12

Poland 20285.56

Haiti 18711.24

Jamaica 18301.37

Philippines 17929.94

Mexico 17666.12

Hong Kong 17026.44

Ireland 16986.3

Nicaragua 16986.09

Italy 16974.79

India 16460.0

El-Salvador 16449.0

Cuba 15771.34

China 15743.86

Taiwan 14183.04

Iran 13710.0

South Korea 11127.54

Columbia 10795.76

Dominican-Republic 9248.47

Ecuador 7730.46

**5.6: Finding the Non-filers whose age is below 14 and generating tax as 0.**

*Tool used: Apache Hive.*

**Step 1:** Writing Hive query to find the Nonfilers whose age below 14 and generating income tax as 0 and storing it int0 HDFS as CSV file.

INSERT OVERWRITE DIRECTORY '/Project/hive/NONFILERS\_AGE\_BELOW14'

ROW FORMAT DELIMITED

FIELDS TERMINATED BY ','

SELECT \*, CASE WHEN taxfilerstatus = " Nonfiler" AND age <=14 THEN 0 END as tax from census WHERE taxfilerstatus = " Nonfiler" AND age <=14;

**SAMPLE OUTPUT:**

1385,0,infants, Children, Never married, Female, Nonfiler,31908.72, Both parents present, United-States, Native- Born in the United States,0,0

1462,0,infants, Children, Never married, Female, Nonfiler,19277.52, Mother only present, United-States, Native- Born in the United States,0,0

1675,0,infants, Children, Never married, Male, Nonfiler,26351.88, Both parents present, United-States, Native- Born in the United States,0,0

**5.7: Finding Income-Tax for each individual user based on their tax filer status and age by following US Taxation rates.**

*Tool used: Apache Map-Reduce.*

**US-Tax rates on 2016.**

**Tax-Filer-Status = 'Single' and age <65**

|  |  |
| --- | --- |
| **Taxable Income** | **Tax Rate** |
| $0—$9,275 | 10% |
| $9,276—$37,650 | $927.50 plus 15% of the amount over $9,275 |
| $37,651—$91,150 | $5,183.75 plus 25% of the amount over $37,650 |
| $91,151—$190,150 | $18,558.75 plus 28% of the amount over $91,150 |
| $190,151—$ 413,350 | $46,278.75 plus 33% of the amount over $190,150 |
| $413,351—$415,050 | $119,934.75 plus 35% of the amount over $413,350 |

**Tax-Filer-Status = 'Single' and age >65**

|  |  |
| --- | --- |
| **Taxable Income** | **Tax Rates** |
| $0—$11,900 | 10% of amount |
| $11,901—$37650 | $927.50 plus 15% of the amount over $9,275 |
| $37,651—$91,150 | $5,183.75 plus 25% of the amount over $37,650 |
| $91,151—$190,150 | $18,558.75 plus 28% of the amount over $91,150 |
| $190,151—$ 413,350 | $46,278.75 plus 33% of the amount over $190,150 |
| $413,351—$415,050 | $119,934.75 plus 35% of the amount over $413,350 |
| $415,051 or more | $120,529.75 plus 39.6% of the amount over $415,050 |

**Tax-Filer-Status = 'Head of Household' and age <65**

|  |  |
| --- | --- |
| Taxable Income | Tax Rates |
| $0—$13,250 | 10% of amount |
| $13,251—$50,400 | $1,325 plus 15% of the amount over $13,250 |
| $50,401—$130,150 | $6,897.50 plus 25% of the amount over $50,400 |
| $130,151—$210,800 | $26,835 plus 28% of the amount over $130,150 |
| $210,801—$413,350 | $49,417 plus 33% of the amount over $210,800 |
| $413,351—$441,000 | $116,258.50 plus 35% of the amount over $413,350 |
| $441,001 or more | $125,936 plus 39.6% of the amount over $441,000 |

**Tax-Filer-Status = 'Head of Household' and age >65**

|  |  |
| --- | --- |
| **Taxable Income** | **Tax Rates** |
| $0—$14,900 | 10% of amount. |
| $14,901—$50,400 | $1,325 plus 15% of the amount over $14,900 |
| $50,401—$130,150 | $6,897.50 plus 25% of the amount over $50,400 |
| $130,151—$210,800 | $26,835 plus 28% of the amount over $130,150 |
| $210,801—$413,350 | $49,417 plus 33% of the amount over $210,800 |
| $413,351—$441,000 | $116,258.50 plus 35% of the amount over $413,350 |
| $441,001 or more | $125,936 plus 39.6% of the amount over $441,000 |

**Tax-Filer-Status = 'Joint both under 65'**

|  |  |
| --- | --- |
| **Taxable Income** | **Tax Rates** |
| $0—$18,550 | 10% of amount. |
| $18,551—$75,300 | $1,855 plus 15% of the amount over $18,550 |
| $75,301—$151,900 | $10,367.50 plus 25% of the amount over $75,301 |
| $15,1901—$23,1450 | $29,517 plus 28% of the amount over $151,901 |
| $23,1451—$413,350 | $51,791.50 plus 33% of the amount over $23,1451 |
| $413,351—$466,950 | $111,818.50 plus 35% of the amount over $413,351 |
| $466,951 or more | $130,578.50 plus 39.6% of the amount over $466,951 |

**Tax-Filer-Status = 'Joint both above 65'**

|  |  |
| --- | --- |
| **Taxable Income** | **Tax Rates** |
| $0—$23,200 | 10% of amount. |
| $23,201—$75,300 | $1,855 plus 15% of the amount over $18,550 |
| $75,301—$15,1900 | $10,367.50 plus 25% of the amount over $75,301 |
| $15,1901—$23,1450 | $29,517 plus 28% of the amount over $151,901 |
| $231,451—$413,350 | $51,791.50 plus 33% of the amount over $23,1451 |
| $413,351—$466,950 | $111,818.50 plus 35% of the amount over $413,351 |
| $466,951 or more | $130,578.50 plus 39.6% of the amount over $466,951 |

**Tax-Filer-Status = 'Joint one under 65 & one above 65'**

|  |  |
| --- | --- |
| Taxable Income | Tax Rates |
| $0—$21,950 | 10% of amount. |
| $21,951—$37,650 | $1,855 plus 15% of the amount over $18,550 |
| $37,651—$91,150 | $10,367.50 plus 25% of the amount over $75,301 |
| $91,151—$190,150 | $29,517 plus 28% of the amount over $151,901 |
| $190,151—$413,350 | $51,791.50 plus 33% of the amount over $23,1451 |
| $413,351—$415,050 | $111,818.50 plus 35% of the amount over $413,351 |
| $415,050 or more | $130,578.50 plus 39.6% of the amount over $466,951 |

**Step 1:** Writing Map-Reduce java program to generate income tax those are citizens and age above 14 by following US tax rates.

import java.io.IOException;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Job;

import org.apache.hadoop.mapreduce.Mapper;

import org.apache.hadoop.mapreduce.Reducer;

import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;

import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class Taxation

{

public static class MyMapper extends Mapper<LongWritable,Text,Text,Text>

{

private Text ID = new Text();//getting ID

private Text taxFiler = new Text();//getting age,TaxFilerStatus,Income

public void map(LongWritable key, Text value, Context con) throws IOException, InterruptedException

{

String[] row = value.toString().split(",");

int age = Integer.parseInt(row[1].trim());

if (age>=14 && (row[6].equals(" Single"))

|| (row[6].equals(" Head of household"))

|| (row[6].equals(" Joint both under 65"))

|| (row[6].equals(" Joint both 65+"))

|| (row[6].equals(" Joint one under 65 & one 65+")))

{

ID.set(row[0]+","+row[1]+","+row[2]+","+row[3]+","+row[4]+","+row[5]+","+row[6]+","+row[7]+","+row[8]+","+row[9]+","+row[10]+","+row[11]);

String str = (row[1]+","+row[6]+","+row[7]);

taxFiler.set(str);

con.write(ID, taxFiler);

}

}

}

public static class MyReducer extends Reducer<Text,Text,Text,Text>

{

public void reduce(Text key, Iterable <Text> value, Context con) throws IOException, InterruptedException

{

for (Text val : value)

{

String[] str = val.toString().split(",");

int age = Integer.parseInt(str[0]);

double income = Double.parseDouble(str[2]);

double tax = 0;

if (age<65 && income < 9275 && str[1].equals(" Single"))

{

tax = income\*0.1;

}

if (age<65 && (income >= 9276 && income <=37650) && str[1].equals(" Single"))

{

tax = ((income-9275)\*0.15)+927.5;

}

if (age<65 &&(income >= 37651 && income <= 91150) && str[1].equals(" Single"))

{

tax = ((income-37650)\*0.25)+5183.75;

}

if (age<65 &&(income >= 91151 && income <= 190150) && str[1].equals(" Single"))

{

tax = ((income-91150)\*0.28)+18558.75;

}

if (age>=65 && income < 11900 && str[1].equals(" Single"))

{

tax = income\*0.1;

}

if (age>=65 &&(income >= 9276 && income <= 37650) && str[1].equals(" Single"))

{

tax = ((income-9275)\*0.15)+927.5;

}

if (age>=65 &&(income >= 37651 && income <= 91150) && str[1].equals(" Single"))

{

tax = ((income-37650)\*0.25)-5183.75;

}

if (age>=65 &&(income >= 91151 && income <= 190150) && str[1].equals(" Single"))

{

tax = ((income-91150)\*0.28)-18558.75;

}

if (age<65 && income < 13250 && str[1].equals(" Head of household"))

{

tax = income\*0.1;

}

if (age<65 &&(income >= 13251 && income <= 50400) && str[1].equals(" Head of household"))

{

tax = ((income-13250)\*0.15)+1325;

}

if (age<65 &&(income >= 50401 && income <= 130150) && str[1].equals(" Head of household"))

{

tax = ((income-50400)\*0.25)+5183.75;

}

if (age<65 &&(income >= 130151 && income <= 210800) && str[1].equals(" Head of household"))

{

tax = ((income -130150)\*0.28)+18558.75;

}

if (age>=65 && income < 14900 && str[1].equals(" Head of household"))

{

tax = income\*0.1;

}

if (age>=65 &&(income >= 14901 && income <= 50400) && str[1].equals(" Head of household"))

{

tax = ((income-14900)\*0.15)+1325;

}

if (age>=65 &&(income >= 50401 && income <= 130150) && str[1].equals(" Head of household"))

{

tax = ((income-50400)\*0.25)+6897.5;

}

if (age>=65 &&(income >= 130151 && income <= 210800) && str[1].equals(" Head of household"))

{

tax = ((income -130150)\*0.28)+26835;

}

if (age<65 && income < 18550 && str[1].equals(" Joint both under 65"))

{

tax = income\*0.1;

}

if (age<65 &&(income >= 18551 && income <= 75300) && str[1].equals(" Joint both under 65"))

{

tax = ((income-18550)\*0.15)+1855;

}

if (age<65 &&(income >= 75301 && income <= 151900) && str[1].equals(" Joint both under 65"))

{

tax = ((income-75301)\*0.25)+10367.5;

}

if (age<65 &&(income >= 151901 && income <= 231450) && str[1].equals(" Joint both under 65"))

{

tax = ((income -151901)\*0.28)+29517;

}

if (age>=65 && income < 23200 && str[1].equals(" Joint both 65+"))

{

tax = income\*0.1;

}

if (age>=65 &&(income >= 23201 && income <= 75300) && str[1].equals(" Joint both 65+"))

{

tax = ((income-18550)\*0.15)+1855;

}

if (age>=65 &&(income >= 75301 && income <= 151900) && str[1].equals(" Joint both 65+"))

{

tax = ((income-75301)\*0.25)+10367.5;

}

if (age>=65 &&(income >= 151901 && income <= 231450) && str[1].equals(" Joint both 65+"))

{

tax = ((income -151901)\*0.28)+29517;

}

if (income<21950 && str[1].equals(" Joint one under 65 & one 65+"))

{

tax = income\*0.1;

}

if ((income >= 21951 && income <= 37650) && str[1].equals(" Joint one under 65 & one 65+"))

{

tax = ((income-18550)\*0.15)+1855;

}

if ((income >= 37651 && income <= 91150) && str[1].equals(" Joint one under 65 & one 65+"))

{

tax = ((income-75301)\*0.25)+10367.5;

}

if ((income >= 91151 && income <= 190150) && str[1].equals(" Joint one under 65 & one 65+"))

{

tax = ((income -151901)\*0.28)+29517;

}

String income\_tax = String.format("%.2f", tax);

con.write(key, new Text(income\_tax));

}

}

}

public static void main(String[] args) throws Exception

{

Configuration conf = new Configuration();

conf.set("mapred.textoutputformat.separator",",");

Job job = Job.getInstance(conf, " ");

job.setJarByClass(Taxation.class);

job.setMapperClass(MyMapper.class);

//job.setCombinerClass(IntSumReducer.class);

job.setReducerClass(MyReducer.class);

job.setMapOutputKeyClass(Text.class);

job.setMapOutputValueClass(Text.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(Text.class);

//job.setNumReduceTasks(0);

FileInputFormat.addInputPath(job, new Path(args[0]));

FileOutputFormat.setOutputPath(job, new Path(args[1]));

System.exit(job.waitForCompletion(true) ? 0 : 1);

}

}

**Step 2:** Running the Map-Reduce program (jar file) in Hadoop cluster and storing the results into HDFS.

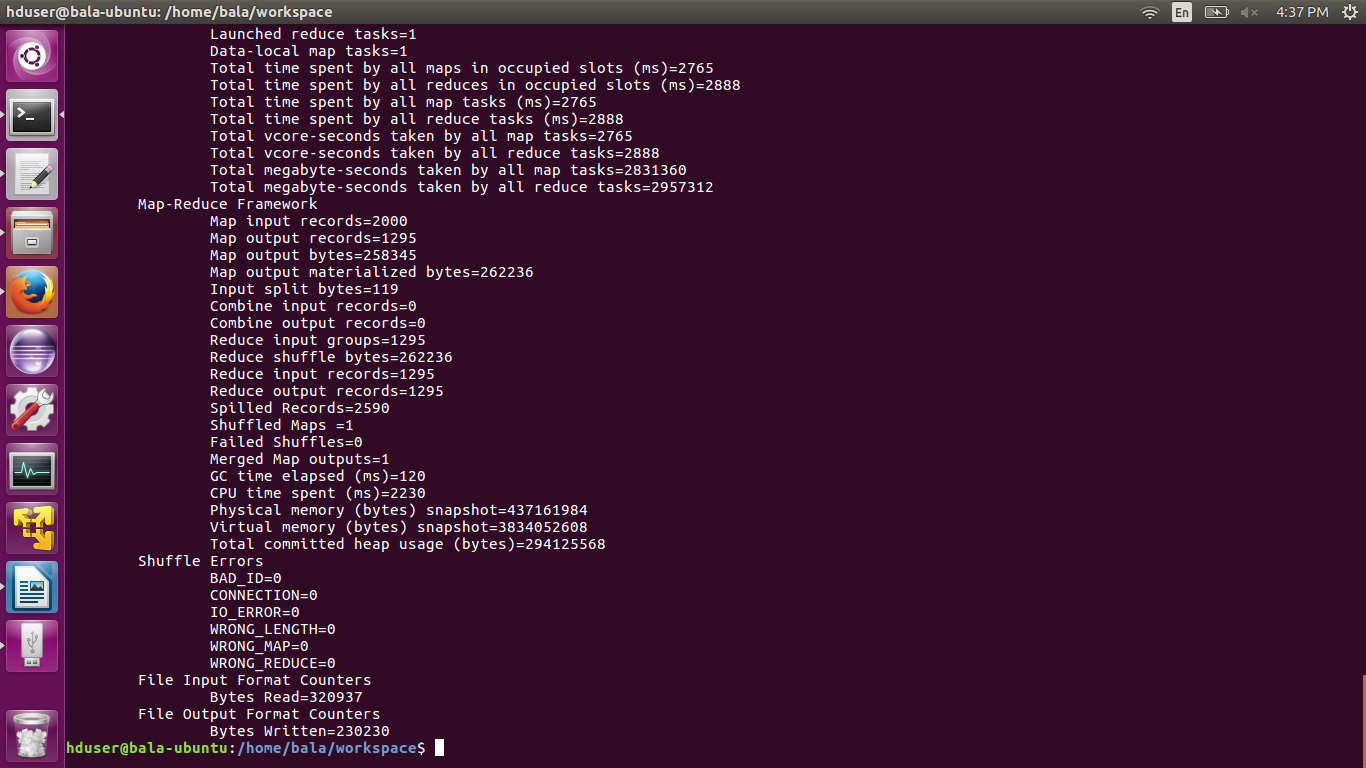
hadoop jar ProjectTaxation.jar Taxation /Project/Census/Input/WithoutTax /Project/Census/TaxationOut

**SAMPLE OUPUT:**

982,60,middle-aged, 1st 2nd 3rd or 4th grade, Separated, Male, Nonfiler,41678.88, Not in universe, United-States, Native- Born in the United States,0,4167.89

972,49,middle-aged, 11th grade, Married-spouse absent, Female, Nonfiler,8965.32, Not in universe, ?, Native- Born abroad of American Parents,0,896.53

965,15,Teenager, 9th grade, Never married, Male, Nonfiler,30159.48, Both parents present, United-States, Native- Born in the United States,0,3015.95



**5.8: Finding the Non-filers of age above 14 and generating them income tax of 10% of tax based on annual income.**

*Tool used: Map-Reduce*

**Step 1:** Writing Map-Reduce java program to generate income tax those are Nonfilers and age above 14.

import java.io.IOException;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Job;

import org.apache.hadoop.mapreduce.Mapper;

import org.apache.hadoop.mapreduce.Reducer;

import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;

import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class NonFilers

{

public static class MyMapper extends Mapper<LongWritable,Text,Text,Text>

{

private Text ID = new Text();//getting ID

private Text taxFiler = new Text();//getting age,TaxFilerStatus,Income

public void map(LongWritable key, Text value, Context con) throws IOException, InterruptedException

{

String[] row = value.toString().split(",");

int age = Integer.parseInt(row[1].trim());

if (age>14 && row[6].equals(" Nonfiler"))

{

ID.set(row[0]+","+row[1]+","+row[2]+","+row[3]+","+row[4]+","+row[5]+","+row[6]+","+row[7]+","+row[8]+","+row[9]+","+row[10]+","+row[11]);

String str = (row[7]);

taxFiler.set(str);

con.write(ID, taxFiler);//extracting key: ID, Value: age above 14, income

}

}

}

public static class MyReducer extends Reducer<Text,Text,Text,Text>

{

public void reduce(Text key, Iterable <Text> value, Context con) throws IOException, InterruptedException

{

for (Text val : value)

{

String[] str = val.toString();

double income = Double.parseDouble(str[0]);

double tax = 0;

tax = income\*0.1; //10% of tax

String income\_tax = String.format("%.2f", tax);

con.write(key, new Text(income\_tax));

}

}

}

public static void main(String[] args) throws Exception

{

Configuration conf = new Configuration();

conf.set("mapred.textoutputformat.separator",",");

Job job = Job.getInstance(conf, " ");

job.setJarByClass(NonFilers.class);

job.setMapperClass(MyMapper.class);

//job.setCombinerClass(IntSumReducer.class);

job.setReducerClass(MyReducer.class);

job.setMapOutputKeyClass(Text.class);

job.setMapOutputValueClass(Text.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(Text.class);

//job.setNumReduceTasks(0);

FileInputFormat.addInputPath(job, new Path(args[0]));

FileOutputFormat.setOutputPath(job, new Path(args[1]));

System.exit(job.waitForCompletion(true) ? 0 : 1);

}

}

**Step 2:** Running the Map-Reduce program (jar file) in Hadoop cluster and storing the results into HDFS.

hadoop jar ProjectTaxation.jar NonFilers '*/Project/hive/Taxation'* '*/Project/Census/NonFilersAboveAge14'*

**SAMPLE OUTPUT:**

1,73,senior citizen, High school graduate, Widowed, Female, Nonfiler,20401.08, Not in universe, United-States, Native- Born in the United States,0,2040.11

1000,76,senior citizen, High school graduate, Married-civilian spouse present, Female, Nonfiler,11688.6, Not in universe, United-States, Native- Born in the United States,0,1168.86

1014,73,senior citizen, High school graduate, Married-civilian spouse present, Female, Nonfiler,19031.64, Not in universe, Poland, Foreign born- U S citizen by naturalization,0,1903.16

**5.9: Joining all processed data into a single file.**

*Tool used: Apache Pig.*

**Step 1:** Loading all three (Non-filers age below 14, Non-filers age above 14, Rest of the tax-filers) into Pig Grunt Shell from HDFS.

set default\_parallel 1

nonfilerbelow14 = load '*/Project/hive/NONFILERS\_AGE\_BELOW14'* using PigStorage(',') as (ID:INT,Age:INT,AgeGroup,Education,MartialStatus,Gender,TaxFilersStatus,Income:Double,Parents,CountryOfBirth,Citizenship,WeeksWorked:INT,Tax:DOUBLE);

nonfilerabove14 = load '*/Project/Census/NonFilersAboveAge14'* using PigStorage(',') as (ID:INT,Age:INT,AgeGroup,Education,MartialStatus,Gender,TaxFilersStatus,Income:Double,Parents,CountryOfBirth,Citizenship,WeeksWorked:INT,Tax:DOUBLE);

taxation = load '*/Project/hive/Taxation*' using PigStorage(',') as (ID:INT,Age:INT,AgeGroup,Education,MartialStatus,Gender,TaxFilersStatus,Income:Double,Parents,CountryOfBirth,Citizenship,WeeksWorked:INT,Tax:DOUBLE);

**Step 2:** Joining all three datasets using UNION function.

joining = union nonfilerbelow14,nonfilerabove14,taxation;

**Step 3:** Storing processed data into HDFS as single file.

record = GROUP joining BY 1; -- group ALL of the records together

final = FOREACH record GENERATE FLATTEN(joining);

STORE final INTO '*/Project/Census/WithTax'* USING PigStorage(',');

**SAMPLE OUTPUT:**

126,15,Teenager, 10th grade, Never married, Male, Single,27462.0, Both parents present, United-States, Native- Born in the United States,6,3655.55

302,15,Teenager, 7th and 8th grade, Never married, Male, Single,4418.76, Both parents present, United-States, Native- Born in the United States,16,441.88

1610,15,Teenager, 9th grade, Never married, Male, Single,14740.92, Both parents present, United-States, Native- Born in the United States,3,1747.39

**5.10: Exporting all processed data from HDFS to MySQL database.**

*Tool used: Apache Sqoop.*

**Step 1: Creating new database and new table in MySQL.**

CREATE DATABASE Project;

USE Project;

CREATE TABLE census (ID INT, Age INT, AgeGroup VARCHAR(20), Education VARCHAR(60), MartialStatus VARCHAR(80), Gender VARCHAR(10), TaxFilersStatus VARCHAR(80), Income DOUBLE(7,2), Parents VARCHAR(80), CountryOfBirth VARCHAR(80), Citizenship VARCHAR(80), WeeksWorked INT, Tax DOUBLE);

**Step 2: Exporting HDFS data to MySQL(RDBMS).**

sqoop export --connect jdbc:mysql://localhost/project --username root --password 4858 --table census --export-dir /Project/Census/TaxWithPercentage/part-r-00000