US Census – Immigration Related Statistical Analysis

Query - https://github.com/bashyan/Immigration_Data_Statistical_Analysis

1. Age group wise Immigration Percentage
2. Education wise Immigration Percentage
3. Population and Poverty Distribution
4. Which country sources large educated people and contributes more to USA by tax?
5. Tax Distribution and Per Capita Income between Natives and Immigrants, Is "Immigrant Ban" required?

1. **Age group wise Immigration Percentage**

- Tool : Hive
- Prerequisite : Sample data and lookup files should be loaded in Hive table.
- Sample data loaded in '**census**' table and lookup file loaded in '**age**' table of database '**immigration**'

```
hive (immigration)> select b.agegroup as Age_Group, round((count(a.Citizenship)/t.tot)*100,2)  as
                    Immigrants_Percentage
                    from
                    census a, age b,
                        (
                        select
                        (count(Citizenship)) as tot
                        from
                        census
                        where (Citizenship=" Foreign born- Not a citizen of U S " or
                        Citizenship= " Foreign          born- U S citizen by naturalization")
                        ) t
                    where a.age=b.age and (a.Citizenship=" Foreign born- Not a citizen of U S " or
                    a.Citizenship= " Foreign born- U S citizen by naturalization")
                    group by b.agegroup, t.tot;
```

Output:

Age Group	Immigrant Percentage(%)
Teenager	2.51
Adult	49.75
Elderly	2.01
Infants	5.53
Middle-aged	30.15
Senior Citizen	10.05

2. **Education Group Wise Immigrant Population**

```
hive (immigration)> select education,round((count(Citizenship)/t.tot)*100,2) as
                    Immigrants_Percentage
                    from
                    census,
                    (
                        select (count(Citizenship)) as tot
                        from
                        census
                        where (Citizenship=" Foreign born- Not a citizen of U S " or
                        Citizenship= " Foreign born- U S citizen by naturalization")
                    ) t
                    where (Citizenship=" Foreign born- Not a citizen of U S " or
                    Citizenship= " Foreign born- U S citizen by naturalization")
                    group by education, t.tot
                    order by Immigrants_Percentage desc;
```

Output:

Education	Percentage(%)
High school graduate	27.14
Some college but no degree	12.56
Bachelors degree(BA AB BS)	11.56
Children	5.53
7th and 8th grade	5.53
5th or 6th grade	5.03
9th grade	5.03
1st 2nd 3rd or 4th grade	4.02
Associates degree-academic program	4.02
10th grade	4.02
Less than 1st grade	3.52
11th grade	3.02
Associates degree-occup /vocational	3.02
Masters degree(MA MS MEng MEd MSW MBA)	2.01
Prof school degree (MD DDS DVM LLB JD)	2.01
12th grade no diploma	1.01
Doctorate degree(PhD EdD)	1.01

3. **Population and Poverty Distribution:**

- Tool : Pig
- Prerequisite : Sample data and lookup data available for Pig to load
- UDF : Apache DataFu Pig is a collection of useful user-defined functions for data analysis
- Methodology : For reusability, Pig commands are executed in BATCH MODE using .pig script file

censusPoverty.pig

```
register 'path_of_jar/datafu-1.2.0.jar';

define Median datafu.pig.stats.Median();

define BagConcat datafu.pig.bags.BagConcat();

census = load 'path_of_samplefile/sampledata'

using PigStorage(',') as (Age: int, Income:double, Citizenship:chararray);

total = foreach (group census all) generate COUNT(census) as total;

totimmi = filter census by (Citizenship == ' Foreign born- Not a citizen of U S ') OR (Citizenship == ' Foreign born- U S citizen by naturalization');

immigrants = foreach (group totimmi all) generate COUNT(totimmi) as immigrants;

age = load 'path_of_look_up_file/agegroup' using PigStorage('\t') as (Age: int, Agegroup:chararray);

joinage = join census by Age, age by Age;

joinedbag = foreach joinage generate $0,$11 as Agegroup,$1,$2,$3,$4,$5 as Income,$6,$7,$8 as Citizenship,$9;

ordercensus = order joinedbag by Income asc;

medianincome = foreach (group ordercensus all) generate FLATTEN(Median(ordercensus.Income)) as Median;

poverty = filter ordercensus by (Income<(medianincome.Median*0.60));

totalpoverty = foreach (group poverty all) generate COUNT(poverty) as tot_poverty;

                    ---------- POVERTY IMMIGRANT----------

immipov = filter poverty by (Citizenship == ' Foreign born- Not a citizen of U S ') OR (Citizenship == ' Foreign born- U S citizen by naturalization');

immipovcount = foreach (group immipov all) generate COUNT(immipov) as immigrant_poverty;

immichild = filter immipov by (Agegroup == 'infants');

childpov = foreach (group immichild all) generate COUNT(immichild) as childpov;

immiold = filter immipov by (Agegroup == 'senior citizen') or (Agegroup == 'elderly');

oldpov = foreach (group immiold all) generate COUNT(immiold) as oldpov;
```

```
joinbags = cogroup total by total, immigrants by immigrants, totalpoverty by tot_poverty,
immipovcount by immigrant_poverty;

bagcon = foreach joinbags generate BagConcat(total,immigrants,totalpoverty,immipovcount);

inters = foreach bagcon generate total.total, immigrants.immigrants, (total.total -
immigrants.immigrants) as natives, totalpoverty.tot_poverty,  immipovcount.immigrant_poverty,
(totalpoverty.tot_poverty - immipovcount.immigrant_poverty) as native_poverty,
childpov.childpov, oldpov.oldpov;

interlimit = limit inters 1;

percentage = foreach interlimit generate total, natives,

        (((double)natives*100)/(double)total) as native_per,immigrants,

        (((double)immigrants*100)/(double)total) as immi_per, tot_poverty,

        (((double)tot_poverty*100)/(double)total) as poverty_per, native_poverty,

        ROUND_TO((((double)native_poverty*100)/(double)natives),2) as nativepov_per,

        ROUND_TO((((double)native_poverty*100)/(double)tot_poverty),2) as
nativetotpov_per, immigrant_poverty,

        ROUND_TO((((double)immigrant_poverty*100)/(double)immigrants),2) as
immipoverty_per,

        ROUND_TO((((double)immigrant_poverty*100)/(double)tot_poverty),2) as
immitotpov_per,

        ROUND_TO((((double)childpov*100)/(double)immigrants),2) as immichildpov_per,

        ROUND_TO((((double)oldpov*100)/(double)immigrants),2) as immioldpov_per;

percent = foreach percentage generate CONCAT(

        '\nTotal Sample Population \t',(chararray)$0,

        '\n\nTotal Natives Percentage\t',(chararray)$2,'% of total population',

        '\nTotal Immigrants Percentage\t',(chararray)$4,'% of total population',

        '\n\nTotal Poverty Percentage\t',(chararray)$6,'% of total population',

        '\n\nNative Poverty percentage\t',(chararray)$8,'% among natives',

        '\nNative Poverty Percentage\t',(chararray)$9,'% among total poverty',

        '\n\nImmigrant Poverty Percent.\t',(chararray)$11,'% among immigrants',

        '\nImmigrant Poverty Percent.\t',(chararray)$12,'% among total poverty',

        '\n\nInfant Poverty Percentage\t',(chararray)$13,'% among immigrants',

        '\nOldage Poverty Percentage\t',(chararray)$14,'% among immigrants\n');

store percent into 'output_path/PovertyPercentage';
```

Output:

/PovertPercentage/part-r-00000 file,

Total Sample Population	2000
Total Natives Percentage	90.05% of total population
Total Immigrants Percentage	9.95% of total population
Total Poverty Percentage	20.1% of total population
Native Poverty percentage	19.66% among natives
Native Poverty Percentage	88.06% among total poverty
Immigrant Poverty Percent.	24.12% among immigrants
Immigrant Poverty Percent.	11.94% among total poverty
Infant Poverty Percentage	2.01% among immigrants
Oldage Poverty Percentage	3.02% among immigrants

4. Which country sources large educated people and contributes more to US by tax?

- Tool : Pig
- Prerequisite : Sample data available for Pig to load, no lookup file needed
- The Sample data contains Tax field which is executed in MapReduce, and will be discussed later, the MapReduce Java file is available in https://github.com/bashyan/Immigration_Data_Statistical_Analysis/blob/master/tax.java

country.pig

```
loaddata = load 'path_of_input_file/TaxMapper' using PigStorage(',') as (education:chararray, taxfilerstatus:chararray, tax:double, country:chararray, citizenship:chararray);

immigrants = filter loaddata by (taxfilerstatus != ' Nonfiler') and ((citizenship == ' Foreign born- Not a citizen of U S ') OR (citizenship == ' Foreign born- U S citizen by naturalization')) and (country != ' ?');

taxcal = foreach (group immigrants by country) generate group, ROUND_TO(SUM(immigrants.tax),2) as taxes ;

listcountry = order taxcal by taxes desc;

topcountry = limit listcountry 1;

countrytaxper = foreach (group listcountry all) generate CONCAT('Tax Contribution : ', (chararray)topcountry.$0, CONCAT(' contributed more in income tax and accounted for about ', (chararray)ROUND_TO((topcountry.$1*100)/SUM(listcountry.$1),2)), '% of total tax revenue from immigrants');

educated = filter loaddata by ((education == ' Bachelors degree(BA AB BS)') or (education == ' Masters degree(MA MS MEng MEd MSW MBA)') or (education == ' Prof school degree (MD DDS DVM LLB JD)') or (education == ' Associates degree-academic program') or (education == ' Doctorate degree(PhD EdD)') or (education == ' Associates degree-occup /vocational') or (education == ' High school graduate') or (education == ' Some college but no degree')) and
```

((citizenship == ' Foreign born- Not a citizen of U S ') OR (citizenship == ' Foreign born- U S citizen by naturalization')) and (country != ' ?');

educal = foreach (group educated by country) generate group, COUNT(educated);

listedu = order educal by $1 desc;

topedu = limit listedu 1;

topeduper = foreach (group listedu all) generate CONCAT('Educated Immigrants : ', (chararray)topedu.$0, CONCAT(' sources more educated immigrants and accountes ',(chararray)((topedu.$1*100)/SUM(listedu.$1))), '% of total educated immigrant population');

store countrytaxper into '/output_path_1/CountryTaxPaid';

store topeduper into '/output_path_2/CountryEducated';

Output 1: /CountryTaxPaid/part-r-00000 file,

Tax Contribution : **Mexico** contributed more in income tax and accounted for about 31.77% of total tax revenue from immigrants

Output 2: /CountryEducated/part-r-00000 file,

Educated Immigrants : **Mexico** sources more educated immigrants and accountes 18% of total educated immigrant population

5. Tax Distribution and Per Capita Income between Natives and Immigrants, Is "Immigrant Ban" required?

- Tool : MapReduce, Eclipse
- Prerequisite : Sample data files should be in HDFS
- Methodology: Reduce Side Join is used in this process. One Mapper calculate **TAX** and other Mapper is meant for cleansing data for **Per Capita Income** calculation. The reducer calculate the tax distribution and per capita income (mean income).

```
import java.io.IOException;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.mapreduce.lib.input.MultipleInputs;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

// Reduce Side Join

public class taxpayer
{
```

```
public static class taxMapper extends
            Mapper<LongWritable, Text, LongWritable, Text>
//calculate tax for all record
{

      public void map(LongWritable key, Text value, Context context)throws
      IOException, InterruptedException
      {
            String[] line = value.toString().split(",");
            int age = Integer.parseInt(line[0]);
            double income = (Double.parseDouble(line[5]))*12;
            double tax = 0.0;
            String filer = line[4];
            String citizen = line[8];

            if(age > 14)
            {
                  if((filer.contains(" Single")) ||
                                          (filer.contains(" Nonfiler")))
                        // impose non-filer as Single and calculate tax
                  {

                        if(income < 9275)
                        {
                              tax = income * 0.10;
                        }
                        else if(income > 9276 && income < 37650)
                        {
                              tax = ((income - 9275)*0.15) + 927.50;
                        }
                        else if(income > 37651 && income < 91150)
                        {
                              tax = ((income - 37650)*0.25) + 5183.75;
                        }
                        else if(income > 91151 && income < 190150)
                        {
                              tax = ((income - 91150)*0.28) + 18558.75;
                        }
                        else if(income > 190151 && income < 413350)
                        {
                              tax = ((income - 190150)*0.33) + 46278.75;
                        }
                        else if(income > 413351 && income < 415050)
                        {
                              tax = ((income - 413350)*0.35) + 119934.75;
                        }
                        else if (income > 415051)
                        {
                              tax = ((income - 415050)*0.396) + 120529.75;
                        }
                  }

                  else if(filer.contains(" Joint"))
                  {
                        if(income < 18550)
                        {
                              tax = income * 0.10;
                        }
                        else if(income > 18551 && income < 75300)
                        {
                              tax = ((income - 18550)*0.15) + 1855;
                        }
                        else if(income > 75301 && income < 151900)
```

```
		{
			tax = ((income - 75300)*0.25) + 10367.50;
		}
		else if(income > 151901 && income < 231450)
		{
			tax = ((income - 151900)*0.28) + 29517.50;
		}
		else if(income > 231451 && income < 413350)
		{
			tax = ((income - 231450)*0.33) + 51791.50;
		}
		else if(income > 413351 && income < 466950)
		{
			tax = ((income - 413350)*0.35) + 111818.50;
		}
		else if (income > 466951)
		{
			tax = ((income - 466950)*0.396) + 130578.50;
		}
	}

	else if (filer.contains(" Head"))
	{
		if(income < 13250)
		{
			tax = income * 0.10;
		}
		else if(income > 13251 && income < 50400)
		{
			tax = ((income - 13250)*0.15) + 1325;
		}
		else if(income > 50401 && income < 130150)
		{
			tax = ((income - 50401)*0.25) + 6897.50;
		}
		else if(income > 130151 && income < 210800)
		{
			tax = ((income - 130151)*0.28) + 26835;
		}
		else if(income > 210801 && income < 413350)
		{
			tax = ((income - 210800)*0.33) + 49417;
		}
		else if(income > 413351 && income < 441000)
		{
			tax = ((income - 413350)*0.35) + 116258.50;
		}
		else if (income > 441001)
		{
			tax = ((income - 441000)*0.396) + 125936;
		}
	}
	String taxcal = String.format("%.2f",tax);


	context.write(new LongWritable(0),
			new Text("TaxAll\t"+taxcal));
	//write tax for all record

	if(citizen.contains(" Foreign"))
	{
		context.write(new LongWritable(1),
			new Text("ForeignTax\t"+taxcal+","+filer));
```

```
                        //write tax and filerstatus only for immigrant
                }
                if(citizen.contains(" Native"))
                {
                        context.write(new LongWritable(2),
                                new Text("NativeTax\t"+taxcal+","+filer));
                        //write tax and filerstatus only for native
                }
        }
    }
}

public static class incomeMapper extends
            Mapper<LongWritable, Text, LongWritable, Text>
//extract income
{


    public void map(LongWritable key, Text value, Context context)throws
    IOException, InterruptedException
    {
            String[] line = value.toString().split(",");
            double income = (Double.parseDouble(line[5]))*12;
            String citizen = line[8];
            String incval = String.format("%.2f", income);
            context.write(new LongWritable(3),
                            new Text("IncomeAll\t"+incval));
            //write income of all record
            if(citizen.contains(" Foreign"))
            {
                    context.write(new LongWritable(4),
                            new Text("ForeignIncome\t"+incval));
                    //write income of immigrant
            }
            if(citizen.contains(" Native"))
            {
                    context.write(new LongWritable(5),
                            new Text("NativeIncome\t"+incval));
                    //write income of native
            }
    }
}


public static class taxandmeanReducer extends
                Reducer<LongWritable, Text, LongWritable, Text>
// find tax, mean income
{
  public void reduce(LongWritable key, Iterable<Text> values, Context
  context)throws IOException, InterruptedException
    {
            double taxall = 0, foreigntax = 0, nativetax = 0;
            double taxnonpayforeign = 0, taxnonpaynative = 0;
            double tempmeanall = 0, tempmeanfor = 0, tempmeannat = 0;
            int countall = 0, countfor = 0, countnat = 0;
            for (Text entry : values)
            {
                    String parts[] = entry.toString().split("\t");
                    if (parts[0].equals("TaxAll"))
                    {
                            taxall += Double.parseDouble(parts[1]);
                            //calculate total tax
                    }
```

```
		else if (parts[0].equals("ForeignTax"))
		{
			String rows[] = parts[1].split(",");
			foreigntax += Double.parseDouble(rows[0]);
			//calculate immigrant paid tax
			if(rows[1].contains(" Nonfiler"))
			{
				taxnonpayforeign +=
				Double.parseDouble(rows[0]);
					//calculate immigrant non-paid tax
					[assumed as "Single"]
			}
		}
		else if (parts[0].equals("NativeTax"))
		{
			String rows[] = parts[1].split(",");
			nativetax += Double.parseDouble(rows[0]);
			//calculate native paid tax
			if(rows[1].contains(" Nonfiler"))
			{
				taxnonpaynative += Double.parseDouble(rows[0]);

				//calculate native non-paid tax [assumed as
				"Single"]
			}
		}
		else if (parts[0].equals("IncomeAll"))
		{
			countall++;
			tempmeanall += Double.parseDouble(parts[1]);
			//calculate overall mean income
		}
		else if (parts[0].equals("ForeignIncome"))
		{
			countfor++;
			tempmeanfor += Double.parseDouble(parts[1]);
			//calculate immigrant's mean income
		}
		else if (parts[0].equals("NativeIncome"))
		{
			countnat++;
			tempmeannat += Double.parseDouble(parts[1]);
			//calculate native's mean income
		}
	}
	double meanall = 0, meanfor = 0, meannat = 0;
	meanall = tempmeanall/countall;
	meanfor = tempmeanfor/countfor;
	meannat = tempmeannat/countnat;

	String taxsall = String.format("%.2f",taxall);
	String taxsfor = String.format("%.2f",foreigntax);
	String nonsfor = String.format("%.2f",taxnonpayforeign);
	String taxsnat = String.format("%.2f",nativetax);
	String nonsnat = String.format("%.2f",taxnonpaynative);
	String meansall = String.format("%.2f",meanall);
	String meansfor = String.format("%.2f",meanfor);
	String meansnat = String.format("%.2f",meannat);

	if(taxall != 0)
	{
		context.write(new LongWritable(1),
			new Text(" USA Total Tax:\t $ "+taxsall));
```

```
                }
                if(foreigntax != 0)
                {
                        context.write(new LongWritable(2),
                                new Text(" Total Immigrant Tax:\t  $ "+taxsfor));
                }
                if(taxnonpayforeign != 0)
                {
                        context.write(new LongWritable(3),
                                new Text(" Immigrant Nonfiled Tax: $ "+nonsfor));
                }
                if(nativetax != 0)
                {
                        context.write(new LongWritable(4),
                                new Text(" Total Native Tax:\t  $ "+taxsnat));
                }
                if(taxnonpaynative != 0)
                {
                        context.write(new LongWritable(5),
                                new Text(" Native Nonfiled Tax:\t  $ "+nonsnat));
                }
                if(!Double.isNaN(meanall))
                {
                        context.write(new LongWritable(6),
                                new Text(" US Mean Income:\t  $ "+meansall));
                }
                if(!Double.isNaN(meanfor))
                {
                        context.write(new LongWritable(7),
                                new Text(" Immigrant's Mean Income:$ "+meansfor));
                }
                if(!Double.isNaN(meannat))
                {
                        context.write(new LongWritable(8),
                                new Text(" Native's Mean Income:\t  $ "+meansnat));
                }

        }
    }


    public static void main(String[] args) throws Exception
    {
        Configuration conf = new Configuration();
        conf.set("mapred.textoutputformat.separator", ",");
        Job job = Job.getInstance(conf);
        job.setJarByClass(taxpayer.class);
        job.setJobName("Tax and Per Capita");
        job.setReducerClass(taxandmeanReducer.class);
        job.setOutputKeyClass(LongWritable.class);
        job.setOutputValueClass(Text.class);

        MultipleInputs.addInputPath(job, new Path(args[0]),
                                TextInputFormat.class, taxMapper.class);
        MultipleInputs.addInputPath(job, new Path(args[1]),
                                TextInputFormat.class, incomeMapper.class);

        FileOutputFormat.setOutputPath(job, new Path(args[2]));
        System.exit(job.waitForCompletion(true) ? 0 : 1);
    }
}
```

Output part-r-00000 file,

1, USA Total Tax:	$ 4001038.81
2, Total Immigrant Tax:	$ 405905.40
3, Immigrant Nonfiled Tax:	$ 87723.19
4, Total Native Tax:	$ 3595133.41
5, Native Nonfiled Tax:	$ 657909.09
6, US Mean Income:	$ 20739.14
7, Immigrant's Mean Income:	$ 18269.87
8, Native's Mean Income:	$ 21011.98

It shows that on an average, Immigrant pay tax 2.1% more than Native people and the per capita income of Immigrant is 13.05% lesser than Native. It signifies that Immigrant's contribution to US wealth is highly assessable and rising ban on Immigrant will decline the nation's revenue.

Data Export to Relational Database

Steps for export:

1. Create database in MySQL
2. Create table with exact schema of HDFS file
3. Using Sqoop export command the data can be exported,

    - ~$ sqoop export --connect jdbc:mysql://localhost/census --username root --password '------' --table census_data --export-dir /immigrantProjet/Censusdata/tax/part-m-00000

4. In MySQL, the table can be used for further analysis

    - mysql> select Education, round(sum(tax),2) as Tax_Paid from census_data group by education order by tax_paid desc limit 3;

      Output:

Education	Tax_Paid	
High school graduate	$	1,314,928.20
Some college but no degree	$	814,841.71
Bachelors degree(BA AB BS)	$	469,355.37

Analysis Report: from the sample census data

- In overall population foreign born accounts for 9.95% and natives accounts for 90.05%
- Adult age group with high school graduated are the large numbers of foreign born living in USA.
- Among entire population nearly 20.1% of people are living under poverty and natives are the most hit people of poverty and they account for about 88.06% of total poverty.
- Among immigrants, children and elderly age people have poor standard and are under poverty, they composite nearly 2.01% and 3.02% respectively.
- Mexico sources more immigrants and their population has the higher literacy rate, they comprise nearly 18 % of total educated immigrants and also they pay more income tax which generates tax revenue of about 31.77% of total tax collected from foreign born.
- The per capita income of immigrants is lesser than that of natives, as it shows that immigrants are paid lesser.
- Analysing the tax filer details, more number of immigrants doesn't file their tax.
- On analysing the average tax paid, it shows that Immigrant pay tax 2.1% more than Native people and the per capita income of Immigrant is 13.05% lesser than Native. It signifies that Immigrant's contribution to US wealth is highly assessable and rising ban on Immigrant will decline the nation's revenue.

Suggestions from the report:

- Native people standard should be improved and more jobs to be created for natives.
- Though children and old age immigrants have lesser poverty level, healthcare and education for them to be improved. They should be given higher social security.
- The per capita income of immigrants is 14% lesser than that of natives, Civil Rights Act to be revised and equal employment opportunity and salary should be ensured.
- More immigrants come from Mexico, and the diplomatic relation to be improved and community education opportunity can be provided for Mexicans.
- On analysing the sample census data **without** considering **illegal** migrants, the foreign born contribute more to US and it can be concluded that 'Immigration Ban' is not necessary for United States of America.