USECASES IMPLEMENTED IN PIG

USECASE :

getting the total count of employment based on education status.

Factors(module) 1: education

Implementation:

s1 = load '/user/hive/warehouse/census\_voter' using PigStorage(',') as (age:int,education:chararray,maritalstatus:chararray,gender:chararray,taxfilerstatus:chararray,income:chararray,parents:chararray,countryofbirth:chararray,citizenship:chararray,weekworked:int);

s2 = foreach s1 generate education,age,weekworked;

s2a = filter s2 by weekworked>0;

s2b = filter s2 by weekworked==0;

s3a = group s2a by education;

s3b = group s2b by education;

s4a = foreach s3a generate group,'Employed',COUNT(s2a.age);

s4b = foreach s3b generate group,'Unemployed',COUNT(s2b.age);

store s4a into '/user/cloudera/Education\_wise\_employed\_count';

store s4b into '/user/cloudera/Education\_wise\_unemployed\_count';

dump s4a;

dump s4b;

Factors(module) 3: total income

Implementation :

s1 = load '/user/hive/warehouse/census\_voter' using PigStorage(',') as (age:int,education:chararray,maritalstatus:chararray,gender:chararray,taxfilerstatus:chararray,income:chararray,parents:chararray,countryofbirth:chararray,citizenship:chararray,weekworked:int);

s2 = foreach s1 generate education,age,weekworked;

s2a = filter s2 by weekworked>0;

s2b = filter s2 by weekworked==0;

s3a = group s2a by education;

s3b = group s2b by education;

s4a = foreach s3a generate group,'Employed',COUNT(s2a.age);

s4b = foreach s3b generate group,'Unemployed',COUNT(s2b.age);

store s4a into '/user/cloudera/Education\_wise\_employed\_count';

store s4b into '/user/cloudera/Education\_wise\_unemployed\_count';

dump s4a;

dump s4b;

Factor(module) 4: total orphans, widows

Implementation :

//Total no of Orphans & widows

s1 = load '/user/hive/warehouse/censusdb.db/censustable/' using PigStorage(',') as (age:int,education:chararray,maritalstatus:chararray,gender:chararray,taxfilerstatus:chararray,income:chararray,parents:chararray,countryofbirth:chararray,citizenship:chararray,weekworked:int);

s2 = foreach s1 generate age,parents,maritalstatus;

s3 = group s2 by parents;

s4 = foreach s3 generate group,COUNT(s2.age);

s5 = group s2 by maritalstatus;

s6 = foreach s5 generate group,COUNT(s2.age);

Total\_Orphans = filter s4 by ($0==' Not in universe');

Total\_Widowed = filter s6 by ($0==' Widowed');

store Total\_Orphans into '/user/cloudera/No\_of\_orphans';

store Total\_Widowed into '/user/cloudera/No\_of\_widowed';

Factors (module) 2: education wise gender count

Implementation :

s1 = load '/user/hive/warehouse/censusdb.db/censustable' using PigStorage(',') as (age:int,education:chararray,maritalstatus:chararray,gender:chararray,taxfilerstatus:chararray,income:chararray,parents:chararray,countryofbirth:chararray,citizenship:chararray,weekworked:int);

s2 = foreach s1 generate age,education,gender;

s3 = group s2 by (education,gender);

s4 = foreach s3 generate FLATTEN(group) as (e,g),COUNT(s2.age);

store s4 into '/user/cloudera/Education\_wise\_gender\_count';

dump s4;

MODULES IMPLEMENTED IN HIVE

**USECASE1: FACTOR (TAX) People born in different place and paying tax more than citizenship**

**IMPLEMENTATION**

create view out as select count (countryofbirth),sum(income),taxfilerstatus as p ,q from(select taxfilerstatus as q from censusdata where countryofbirth !='united-States' and income > 500 and citizenship='Native- Born in United States') a, censusdata where countryofbirth =='united-States' and income > 500 and citizenship='Native- Born in United States';

select p > q as out;

**USECASE 2:FACTOR(TAX)Highest tax is belong to is a citizen or a foriegner**

**IMPLEMENTATION**

select citizenship as d,sum(income) as e ,p ,q from (select citizenship as q ,sum(income )as p from censusb where citizenship !='Native- Born in United States' ) a,censusb where

citizenship ='Native- Born in United States' group by p,q;

**Factor name: education(going to be held after 12 months)**

**Usecase**

people whos's age is greater than 18 belong to same place and getting data of people btw 17-18 yrs

create view differentsum as select count(age) as q,p from (select count(age) as p from censusdata where age > 18 and citixenship='Native- Born in United States

') a,censusdata where age between 17 and 18 and citizenship= 'Native- Born in United States' group by p;

create view total as select count(age) as p from censusdata where age > 17 and citizenship='Native- Born in United States' ;

Factor name: education

usecase

people born in different place and got the citizenship and eligible for voting

select count(countryOfBirth) from censusdata where age > 18 countryofbirth!='United-States' and citizenship='Native- Born in United States';

USECASES IMPLEMENTED BY USING MAPREDUCE

Usecase 1 :

**Factor : tax**

**income above 1500 and tax payer**

import java.io.IOException;  
import org.apache.hadoop.conf.Configuration;  
import org.apache.hadoop.fs.Path;  
import org.apache.hadoop.io.DoubleWritable;  
import org.apache.hadoop.io.LongWritable;  
import org.apache.hadoop.io.Text;  
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;  
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;  
  
public class Taxfile1 {  
  
  
public static class Mymapper1 extends  
Mapper<LongWritable, Text, Text, Text> {  
protected void map(LongWritable key, Text value, Context context)  
throws IOException, InterruptedException {  
String[] columns = value.toString().split(",");  
if (Integer.parseInt(columns[5]) > 1500 && !columns[4].equals("Nonfiler")){  
context.write(new Text("Filer"),value);  
}  
}  
  
}  
public static class Myreducer1 extends  
Reducer<Text, Text, Text, DoubleWritable> {  
protected void reduce(Text key, Iterable<Text> values, Context context)  
throws IOException, InterruptedException {  
int count = 0;  
for (Text retreive : values) {  
count++;  
}  
context.write(new Text("finnalcounting"), new DoubleWritable(count));  
}  
  
}  
public static void main(String[] args) throws IOException,  
ClassNotFoundException, InterruptedException {  
Configuration conf = new Configuration();  
Job job = Job.getInstance(conf);  
job.setJarByClass(Taxfile1.class);  
job.setMapperClass(Mymapper1.class);  
job.setMapOutputKeyClass(Text.class);  
job.setMapOutputValueClass(Text.class);  
job.setReducerClass(Myreducer1.class);  
FileInputFormat.addInputPath(job, new Path("/dataforjava"));  
FileOutputFormat.setOutputPath(job, new Path(args[0]));  
job.waitForCompletion(true);  
}  
}

**usecase 2:top 5 average income of tax payers**

**factor name: tax**

**IMPLEMENTATION**

import java.io.IOException;  
import java.util.Iterator;  
import java.util.NavigableMap;  
import java.util.Set;  
import java.util.TreeMap;  
  
import org.apache.hadoop.conf.Configuration;  
import org.apache.hadoop.fs.Path;  
import org.apache.hadoop.io.DoubleWritable;  
import org.apache.hadoop.io.LongWritable;  
import org.apache.hadoop.io.Text;  
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;  
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;  
  
public class Taxfile3 {  
  
public static class Mymapper1 extends  
Mapper<LongWritable, Text, Text, DoubleWritable> {  
protected void map(LongWritable key, Text value, Context context)  
throws IOException, InterruptedException {  
String[] columns = value.toString().split(",");  
if (!columns[4].equals("Nonfiler")) {  
context.write(new Text("Filereducation"),new DoubleWritable(Double.parseDouble(columns[5])));  
}  
}  
  
}  
public static class Myreducer1 extends  
Reducer<Text, DoubleWritable, Text, DoubleWritable> {  
protected void reduce(Text key, Iterable<DoubleWritable> values, Context context)  
throws IOException, InterruptedException {  
TreeMap<Double,String> max=new TreeMap<Double,String>();  
Double average=0.0;  
int check=0;  
double sum=0;  
for (DoubleWritable retreive : values) {  
max.put((Double)retreive.get(),"maximumincome");   
}   
NavigableMap<Double, String> nav=max.descendingMap();  
Set<Double> set=nav.keySet();  
Iterator<Double> itr=set.iterator();  
  
while(itr.hasNext()){  
if(check>10){  
   break;   
}   
else  
{  
sum=sum+itr.next();  
}   
check++;   
}  
average=sum/(check+1);  
context.write(new Text("averageoftop5"),new DoubleWritable(average));  
context.write(new Text("highestincome"),new DoubleWritable(itr.next()));  
  
}  
}   
public static void main(String[] args) throws IOException,  
ClassNotFoundException, InterruptedException {  
Configuration conf = new Configuration();  
Job job = Job.getInstance(conf);  
job.setJarByClass(Taxfile1.class);  
job.setMapperClass(Mymapper1.class)