**1.Create a new directories on HDFS and store census data .**

>hadoop fs -mkdir /project

>hadoop fs -put /home/hduser/Downloads/Census/sample.dat /project

>hadoop fs -cat /project/sample.dat

we store json data into hdfs

**2.Convert Jsondata into CSV format by using Pig**

>pig

>Jsondata = LOAD '/project/sample.dat' USING JsonLoader('Age:INT,Education:chararray,MaritalStatus:chararray,Gender:chararray,TaxFilerStatus:chararray,Income:INT,Parents:chararray,CountryOfBirth:chararray,Citizenship:chararray,WeeksWorked:INT');

>STORE Jsondata INTO '/project/pigcensus' USING PigStorage(',');

\*\*\*MYSQL\*\*\*\*

>Create a secondary table in mysql.

>create database census;

>create table census(Gender varchar(30) not null, Min\_Income int not null,Max\_Income not null, Tax int not null);

>insert into census values('Male',50,1000,5);

>insert into census values('Female',50,1000,4);.....and so on..

\*\*\*\*SQOOP\*\*\*

>Import data from mysql into hdfs.

>sqoop import --connect jdbc:mysql://quickstart.cloudera:3306/census --username root --password cloudera --m 1 --table census --target-dir /project/pigdata/sqlimport

>Jsondata1 = LOAD '/project/pigdata' USING PigStorage(',');

>STORE Jsondata INTO '/project/pigsecondarydata' USING PigStorage(',');

3.Store census data from hdfs into hive

> create database census;

>show databases;

>use census;

>create table projdb(Age INT, Education String, MaritalStatus String, Gender String, TaxFilerStatus String, Income INT, Parents String, CountryOfBirth String, Citizenship String, WeeksWorked INT)

row format delimited

fields terminated by ','

stored as textfile;

>load data inpath '/myproject/pig\_output/part-m-00000' into table projdb;

PROJECT TASKS

Education-HIVE

**1.Total count of male/female based on education.**

>I owned insaurance company LIC, i want to figure out the total number of female and male citizen based on their education who can effort my insaurance

>select education,gender,count(gender) from social group by education,gender;

>output:

**2.Total count of employed/unemployed based on education.**

>analysis of total number of employed and unemployed citizen, based on education criteria.

>select education,count(weeksworked) from social where weeksworked>=0 group by education;

Output:

>select education,count(weeksworked) from social where weeksworked=0 group by education;

Output:

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

>analysis of the people whose age lies between or who are in age group of 18-25 based on their education.

>select Education,count(Age) from projdb where Age between 18 and 25 group by Education;

Output:

Planning---PIG

**1.Voter(s) count in x year(s)**

>analysis of the teenager who r going to be an adult after three years

>>AGE

process:

>>census = load '/project/part-m-00000' using PigStorage(',') as (Age,Education,MartialStatus,Gender,TaxFilersStatus,Income,Parents,CountryOfBirth,Citizenship,WeeksWorked);

>>age = FOREACH census GENERATE Age;

>>vote = FILTER age BY Age > 15;

>>group\_vote = Group vote All;

>>count\_age = FOREACH group\_vote GENERATE COUNT(vote);

>>dump count\_age;

output:

(1515)

**2.Senior Citizen(s) count in x year(s)**

>analysis of the people who r going to be an senior citizen after three years.

process:

>>census = load '/project/pigcensus/part-m-00000' using PigStorage(',') as (Age,Education,MartialStatus,Gender:bytearray,TaxFilersStatus:bytearray,Income:DOUBLE,Parents,CountryOfBirth,Citizenship,WeeksWorked:INT);

>>senior\_age = FOREACH census GENERATE Age;

>>senior\_agecount = FILTER senior\_age BY Age > 57;

>> group\_seniorage = Group senior\_age All;

>>count\_senior = FOREACH group\_seniorage GENERATE COUNT(senior\_agecount);

>>dump count\_senior;

>>output:

(306)

**3.Total number of Male/Female**

> To find total number of male and female citizen .

> count total male and female for future refrence

>>GENDER

process:

>>census = load '/project/pigcensus/part-m-00000' using PigStorage(',') as (Age,Education,MartialStatus,Gender:bytearray,TaxFilersStatus:bytearray,Income:DOUBLE,Parents,CountryOfBirth,Citizenship,WeeksWorked:INT);

>>groupgender = GROUP census BY Gender;

>>count = FOREACH groupgender GENERATE group as Gender, COUNT(census);

>>dump count;

>>output:

( Male,939)

( Female,1061)

Or done in MapReduce also

**4.Citizens and immigrants count for employed lot.**

>To figure out the count of the employee based on the citizenship.

>>CITIZENSHIP,weeksworked

process:

>>census = load '/project/pigcensus/part-m-00000' using PigStorage(',') as (Age,Education,MartialStatus,Gender:bytearray,TaxFilersStatus:bytearray,Income:DOUBLE,Parents,CountryOfBirth,Citizenship,WeeksWorked:INT);

>>bag1 = FILTER census BY WeeksWorked != 0;

>>bag2 = GROUP bag1 BY Citizenship;

>>bag3 = FOREACH bag2 GENERATE group as Citizenship, COUNT(bag1);

>>dump bag3;

>>output;

( Native- Born in the United States,968)

( Foreign born- Not a citizen of U S ,82)

( Native- Born abroad of American Parent(s),12)

( Foreign born- U S citizen by naturalization,43)

( Native- Born in Puerto Rico or U S Outlying,6)

Miscellaneous---PIG

--------------------------------------------------------------------------

>Analysed the people with repect to their education, Marital status,citizenship for future business needs.

**1.Degree wise count for employability**

>>The count of employee based on the qualification .

>>census = load '/project/pigcensus/part-m-00000' using PigStorage(',') as (Age,Education,MartialStatus,Gender:bytearray,TaxFilersStatus:bytearray,Income:DOUBLE,Parents,CountryOfBirth,Citizenship,WeeksWorked:INT);

>>bag1 = FILTER census BY WeeksWorked != 0;

>>bag2 = GROUP bag1 BY Education;

>>bag3 = FOREACH bag2 GENERATE group as Education, COUNT(bag1);

>>dump bag3;

>>output:

**2.Customer base analysis:**

> Analysis of the people with respect to their age group that should be below 30,between 31 to 50 and above 50 with their marital status as well as income must be above 2500.

>program:

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.Partitioner;

import org.apache.hadoop.mapreduce.Reducer;

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

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

public class census {

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

{

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

{

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

context.write(new Text(str[2]), new Text(str[0]+","+str[3]+","+str[5]));

}

}

public static class myPartitioner extends Partitioner <Text, Text>

{

public int getPartition(Text key, Text value, int numReduceTasks)

{

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

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

if(age<=30)

{

return 0;

}

else if (age>31 && age<=50)

{

return 1;

}

else

{

return 2;

}

}

}

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

{

public void reduce(Text key, Iterable<Text> values, Context context) throws IOException, InterruptedException

{

double maxFemale\_income = -1;

double maxMale\_income = -1;

for(Text val : values)

{

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

String gender = str[1];

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

if(gender.contains(" Female"))

{

if(income>maxFemale\_income)

{

maxFemale\_income = income;

myvalueFemale = str[0]+","+gender+","+maxFemale\_income;

}

}

if(gender.contains(" Male"))

{

if(income>maxMale\_income)

{

maxMale\_income = income;

myvalueMale = str[0]+","+gender+","+maxMale\_income;

}

}

}

result = myvalueFemale+"\t"+myvalueMale;

}

context.write(key, new Text(result));

}

}

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

{

Configuration conf = new Configuration();

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

job.setJarByClass(census.class);

job.setMapperClass(myMapper.class);

job.setMapOutputKeyClass(Text.class);

job.setMapOutputValueClass(Text.class);

job.setPartitionerClass(myPartitioner.class);

job.setNumReduceTasks(3);

job.setReducerClass(myReducer.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(Text.class);

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

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

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

}

}

OUTPUT:

1)AGE <=30 :

2 age>31 && age<=50 :

3)ABOVE 50 :

**3.Non-US citizen(s) tax filer status**

>>To find the tax filer status who doesn’t have citizenship of US.

>>bag1 = FILTER census By Citizenship MATCHES '.\*Foreign born- Not a citizen of U S.\*';

>>bag2 = GROUP bag1 BY TaxFilerStatus;

>>bag3 = FOREACH bag2 GENERATE group as TaxFilersStatus, COUNT(bag1);

>>dump bag3;

>>output:

( Single,24)

( Nonfiler,36)

( Joint both 65+,1)

( Head of household,6)

( Joint both under 65,68)

**4.Country of birth wise count for US citizenship by naturalisation**

>>based on the their nationality analysis of the people who have citizenship by naturalisation..

>>bag1 = FILTER census By Citizenship MATCHES '.\*Foreign born- U S citizen by naturalization.\*';

>>bag2 = GROUP bag1 BY CountryOfBirth;

>>bag3 = FOREACH bag2 GENERATE group as CountryOfBirth, COUNT(bag1);

>>dump bag3;

>>output:

Social---Mapside join,Hive

---------------------------------------------------------------------------

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

>>analysis of the total amount for the pension condering after three years

>>program:

program:

**package** mapside1;

**import** java.io.BufferedReader;

**import** java.io.FileReader;

**import** java.io.IOException;

**import** java.net.URI;

**import** java.util.HashMap;

**import** java.util.Map;

**import** org.apache.hadoop.conf.Configuration;

**import** org.apache.hadoop.fs.Path;

**import** org.apache.hadoop.io.LongWritable;

**import** org.apache.hadoop.io.DoubleWritable;

**import** org.apache.hadoop.io.Text;

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

**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;

**public** **class** pension

{

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

{

**private** Map<Integer, String> abMap = **new** HashMap<Integer, String>();

**protected** **void** setup(Context context) **throws** java.io.IOException, InterruptedException

{

**super**.setup(context);

URI[] files = context.getCacheFiles(); // getCacheFiles returns null

Path p = **new** Path(files[0]);

**if** (p.getName().equals("scho.txt"))

{

BufferedReader reader = **new** BufferedReader(**new** FileReader(p.toString()));

String line = reader.readLine();

**while**(line != **null**)

{

String[] tokens = line.split(",");

**int** P\_id = Integer.*parseInt*(tokens[0]);

String Pensionper = tokens[3];

abMap.put(P\_id, Pensionper);

line = reader.readLine();

}

}

**if** (abMap.isEmpty()) {

**throw** **new** IOException("MyError:Unable to load data.");

}

}

**protected** **void** map(LongWritable key, Text value, Context context)

**throws** java.io.IOException, InterruptedException {

**int** P\_id=0;

String row = value.toString();

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

**double** income = Double.*parseDouble*(tokens[5]);

**if** (income >50 && income <=2000)

{

P\_id = 1;

}

**else** **if** (income >2000 && income <=4000)

{

P\_id = 2;

}

**else** **if** (income >4000 && income <=6000)

{

P\_id = 3;

}

**else** **if** (income >6000 && income <=8000)

{

P\_id = 4;

}

**else** **if** (income >8000 && income <=9000)

{

P\_id = 5;

}

String id = Integer.*toString*(P\_id);

String pension = abMap.get(P\_id);

**double** age = Double.*parseDouble*(tokens[0]);

**double** pensamt = income \* Double.*parseDouble*(pension)/100;

String age1 = String.*valueOf*(age);

String pensamt1 = String.*valueOf*(pensamt);

String output = age1 + "," + pensamt1;

context.write(**new** Text(tokens[3]), **new** Text(output));

}

}

**public** **static** **class** TotalReducer **extends** Reducer<Text,Text,Text,DoubleWritable>

{

**public** **void** reduce(Text key, Iterable<Text>values,Context context) **throws** IOException, InterruptedException {

**double** total = 0;

**for** (Text val : values){

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

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

**double** pensamt = Double.*parseDouble*(str[1]);

**if** (age==57)

{

total += pensamt;

}

}

context.write(key,**new** DoubleWritable(total));

}

}

**public** **static** **void** main(String[] args)

**throws** IOException, InterruptedException, ClassNotFoundException

{

Configuration conf = **new** Configuration();

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

Job job = Job.*getInstance*(conf);

//Job job = new Job(conf,"Taxanalysis");

job.setJarByClass(pension.**class**);

job.setJobName("Map Side Join");

job.setMapperClass(MyMapper.**class**);

job.setReducerClass(TotalReducer.**class**);

job.addCacheFile(**new** Path("scho.txt").toUri());

job.setNumReduceTasks(1);

job.setMapOutputKeyClass(Text.**class**);

job.setMapOutputValueClass(Text.**class**);

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

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

job.waitForCompletion(**true**);

}

}

output:

Female,8616.85

Male,8473.099999999999

**2.Total amount dispensed on scholarship in current year**

>>analysis of scholarship’s to be given to the student who are pursuing their further education.

>insert overwrite local directory '/home/cloudera/Downloads/hive' row format delimited fields terminated by ',' stored as textfile select a.age,a.parents,b.s\_id,b.subsidy from social a join scolarship b on a.parents = b.parents order by age asc;

output:

0, Both parents present,2,0

0, Mother only present,3,25000

0, Both parents present,2,0

0, Both parents present,2,0

0, Mother only present,3,25000

0, Both parents present,2,0

>create table scolarship1(age int,parents String,s\_id int,subsidy int) row format delimited fields terminated by ',' stored as textfile;

>load data local inpath '/home/cloudera/Downloads/hive/hiveop.csv' into table scolarship1;

>select parents,count(subsidy) from scolarship1 group by parents;

> Both parents present 360

Father only present 17

Mother only present 133

Neither parent present 18

Not in universe 1472

>select parents,sum(subsidy) from scolarship1 group by parents;

>Both parents present 0

Father only present 255000

Mother only present 3325000

Neither parent present 540000

Not in universe 44160000

>select sum(subsidy) from scolarship1;

>Total MapReduce CPU Time Spent: 15 seconds 290 msec

OK

48280000

**3.For given age range employable female widowed and divorced count**

>analysis for the female emplyees whose age is between 30 to 60 followed by their marital status(i.e widowed and divorced)

>>AGE,GENDER,MARITAL STATUS,WeeksWorked;

>>process:

0>>create database social;

>>use social;

>>create table social(Age INT, Education String, MaritalStatus String, Gender String, TaxFilerStatus String, Income INT, Parents String, CountryOfBirth String, Citizenship String, WeeksWorked INT)

ROW FORMAT DELIMITED

FIELDS TERMINATED BY ','

STORED AS TEXTFILE;

>>load data local inpath '/home/cloudera/Downloads/part-m-00000' into table social;

>>desc social;

>>select \* from social;

>>select maritalstatus, COUNT(maritalstatus) from social where maritalstatus =' Divorced' and gender = ' Female' or maritalstatus = ' Widowed' and age>30 and age<60 group by maritalstatus;

>>Process:

( Divorced 70)

( Widowed 13)

Finance-Mapreduce,Pig

---------------------------------------------------------------------------

**1.Tax analysis total and gender wise**

>analysis of the total amount of the tax to be paid by male and female

>>program:

**package** mapsidejoin;

**import** java.io.BufferedReader;

**import** java.io.FileReader;

**import** java.io.IOException;

**import** java.net.URI;

**import** java.util.HashMap;

**import** java.util.Map;

**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.lib.input.FileInputFormat;

**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.Mapper.Context;

**import** org.apache.hadoop.mapreduce.Reducer;

**public** **class** findtax

{

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

{

**private** Map<Integer, String> abMap = **new** HashMap<Integer, String>();

**protected** **void** setup(Context context) **throws** java.io.IOException, InterruptedException

{

**super**.setup(context);

URI[] files = context.getCacheFiles(); // getCacheFiles returns null

Path p = **new** Path(files[0]);

**if** (p.getName().equals("tax.txt"))

{

BufferedReader reader = **new** BufferedReader(**new** FileReader(p.toString()));

String line = reader.readLine();

**while**(line != **null**)

{

String[] tokens = line.split(",");

**int** T\_id = Integer.*parseInt*(tokens[0]);

String Tax = tokens[3];

abMap.put(T\_id, Tax);

line = reader.readLine();

}

}

**if** (abMap.isEmpty()) {

**throw** **new** IOException("MyError:Unable to load data.");

}

}

**protected** **void** map(LongWritable key, Text value, Context context)

**throws** java.io.IOException, InterruptedException {

**int** T\_id=0;

String row = value.toString();

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

**double** income = Double.*parseDouble*(tokens[5]);

**if** (income >50 && income <=2000)

{

T\_id = 1;

}

**else** **if** (income >2000 && income <=4000)

{

T\_id = 2;

}

**else** **if** (income >4000 && income <=6000)

{

T\_id = 3;

}

**else** **if** (income >6000 && income <=8000)

{

T\_id = 4;

}

**else** **if** (income >8000 && income <=9000)

{

T\_id = 5;

}

String id = Integer.*toString*(T\_id);

String tax = abMap.get(T\_id);

**double** taxamt = income \* Double.*parseDouble*(tax)/100;

context.write(**new** Text(tokens[3]), **new** Text(Double.*toString*(taxamt)));

}

}

**public** **static** **class** TotalReducer **extends** Reducer<Text,Text,Text,Text>

{

**public** **void** reduce(Text key, Iterable<Text> taxamt,Context context) **throws** IOException, InterruptedException {

**double** total= 0.0 ;

**for** (Text val : taxamt){

total += Double.*parseDouble*(val.toString());

}

context.write(key,**new** Text(Double.*toString*(total)));

}

}

**public** **static** **void** main(String[] args)

**throws** IOException, InterruptedException, ClassNotFoundException

{

Configuration conf = **new** Configuration();

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

Job job = Job.*getInstance*(conf);

//Job job = new Job(conf,"Taxanalysis");

job.setJarByClass(findtax.**class**);

job.setJobName("Map Side Join");

job.setMapperClass(MyMapper.**class**);

job.setReducerClass(TotalReducer.**class**);

job.addCacheFile(**new** Path("tax.txt").toUri());

job.setNumReduceTasks(1);

job.setMapOutputKeyClass(Text.**class**);

job.setMapOutputValueClass(Text.**class**);

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

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

job.waitForCompletion(**true**);

}

}

output:

Female,98288.71000000005

Male,95097.59999999998

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

>To analysis of the per capital income followed by criteria like age,gender,agegroup.

>>census = load '/project/part-m-00000' using PigStorage(',') as (Age,Education,MartialStatus,Gender:bytearray,TaxFilersStatus:bytearray,Income:DOUBLE,Parents,CountryOfBirth,Citizenship,WeeksWorked:INT);

>bag1 = foreach census generate Income;

>bag2 = group bag1 all;

>bag3 = foreach bag2 generate SUM(bag1.Income)/COUNT(bag1.Income);

>dump bag3;

>output:

(1727.7615)

> agegroup = load '/project/part-m-00000' using PigStorage(',') as (age:INT, category:bytearray );

> census = load '/project/part-m-00000' using PigStorage(',') as (Age,Education,MartialStatus,Gender:bytearray,TaxFilersStatus:bytearray,Income:DOUBLE,Parents,CountryOfBirth,Citizenship,WeeksWorked:INT);

>bag1 = foreach census generate Income,age;

>bag2 = foreach agegroup generate age,category;

> bag3 = join bag1 by age,bag2 by age;

> bag4 = foreach bag3 generate category,Income;

> bag5 = group bag4 by category;

> bag6 = foreach bag5 generate group,SUM(bag4.Income)/COUNT(bag4.Income);

>dump bag6;

>output:

(adult,1828.3665283540802)

(elderly,1821.6470588235295)

(infants,1631.7329974811082)

(Teenager,1758.4202898550725)

(middle-aged,1671.4414225941423)

(senior citizen,1662.0391304347827)