1 a) Is the number of petitions with Data Engineer job title increasing over time?

import java.io.\*;

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

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

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

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

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

import org.apache.hadoop.mapreduce.lib.output.\*;

public class data {

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

{

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

{

try{

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

long year = Long.parseLong(str[7]);

String jobname = str[4];

if(jobname.equals("DATA ENGINEER"))

{

context.write(new LongWritable(year),new Text(jobname));

}}

catch(Exception e)

{

System.out.println(e.getMessage());

}

}

}

public static class ReduceClass extends Reducer<LongWritable,Text,LongWritable,LongWritable>

{

//private IntWritable result = new IntWritable(0);

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

long count=0;

for (Text val : values)

{

count++;

}

context.write(key, new LongWritable(count));

}

}

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

Configuration conf = new Configuration();

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

job.setJarByClass(data.class);

job.setMapperClass(MapClass.class);

//job.setCombinerClass(ReduceClass.class);

job.setReducerClass(ReduceClass.class);

//job.setNumReduceTasks(2);

job.setMapOutputKeyClass(LongWritable.class);

job.setMapOutputValueClass(Text.class);

job.setOutputKeyClass(LongWritable.class);

job.setOutputValueClass(LongWritable.class);

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

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

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

}

}

1 b) Find top 5 job titles who are having highest avg growth in applications.[ALL]

pro = load '/project/fin' using PigStorage('\t') AS (s\_no:int,case\_status:chararray, employer\_name:chararray, soc\_name:chararray, job\_title:chararray, full\_time\_position:chararray,prevailing\_wage:long,year:int, worksite:chararray, longitute:long, latitute:long);

--dump pro;

kk = filter pro by year == 2011;

--dump kk;

ff = group kk by job\_title;

--dump ff;

ss = foreach ff generate $0, (float)COUNT(kk.$1);

--dump ss;

kk12 = filter pro by year == 2012;

--dump kk12;

ff12 = group kk12 by job\_title;

--dump ff12;

ss12 = foreach ff12 generate $0, (float)COUNT(kk12.$1);

--dump ss12;

jo = join ss by $0,ss12 by $0;

--dump jo;

gn = foreach jo generate $0, $1, $3, ((($3-$1)/$1)\*100) as av;

--dump gn;

kk13 = filter pro by year == 2013;

--dump kk13;

ff13 = group kk13 by job\_title;

--dump ff13;

ss13 = foreach ff13 generate $0, (float)COUNT(kk13.$1);

--dump ss13;

jo13 = join ss12 by $0,ss13 by $0;

--dump jo13;

gn13 = foreach jo13 generate $0, $1, $3, ((($3-$1)/$1)\*100) as av1;

--dump gn13;

kk14 = filter pro by year == 2014;

--dump kk14;

ff14 = group kk14 by job\_title;

--dump ff14;

ss14 = foreach ff14 generate $0, (float)COUNT(kk14.$1);

--dump ss14;

jo14 = join ss13 by $0,ss14 by $0;

--dump jo14;

gn14 = foreach jo14 generate $0, $1, $3, ((($3-$1)/$1)\*100) as av2;

--dump gn14;

kk15 = filter pro by year == 2015;

--dump kk15;

ff15 = group kk15 by job\_title;

--dump ff15;

ss15 = foreach ff15 generate $0, (float)COUNT(kk15.$1);

--dump ss15;

jo15 = join ss14 by $0,ss15 by $0;

--dump jo15;

gn15 = foreach jo15 generate $0, $1, $3, ((($3-$1)/$1)\*100) as av3;

--dump gn15;

kk16 = filter pro by year == 2016;

--dump kk16;

ff16 = group kk16 by job\_title;

--dump ff16;

ss16 = foreach ff16 generate $0, (float)COUNT(kk16.$1);

--dump ss16;

jo16 = join ss15 by $0,ss16 by $0;

--dump jo16;

gn16 = foreach jo16 generate $0, $1, $3, ((($3-$1)/$1)\*100) as av4;

--dump gn16;

gg = join gn by $0,gn13 by $0,gn14 by $0,gn15 by $0,gn16 by $0;

--dump gg;

rr = foreach gg generate $0, ((av+av1+av2+av3+av4)/5) as cv;

--dump rr;

od = order rr by cv desc;

--dump od;

li = limit od 5;

dump li;

store li into '/home/hduser/Downloads/pro1band';

2 a) Which part of the US has the most Data Engineer jobs for each year?

select worksite,count(job\_title) as tot from h1b\_final where job\_title = "DATA ENGINEER" and year = 2011 group by worksite order by tot desc limit 1;

select worksite,count(job\_title) as tot from h1b\_final where job\_title = "DATA ENGINEER" and year = 2012 group by worksite order by tot desc limit 1;

select worksite,count(job\_title) as tot from h1b\_final where job\_title = "DATA ENGINEER" and year = 2013 group by worksite order by tot desc limit 1;

select worksite,count(job\_title) as tot from h1b\_final where job\_title = "DATA ENGINEER" and year = 2014 group by worksite order by tot desc limit 1;

select worksite,count(job\_title) as tot from h1b\_final where job\_title = "DATA ENGINEER" and year = 2015 group by worksite order by tot desc limit 1;

select worksite,count(job\_title) as tot from h1b\_final where job\_title = "DATA ENGINEER" and year = 2016 group by worksite order by tot desc limit 1;

2b) find top 5 locations in the US who have got certified visa for each year.[certified]

import java.io.IOException;

import java.util.TreeMap;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.NullWritable;

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 loc

{

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

{

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

{

String[] str = values.toString().split("\t");

con.write(new Text(str[8]), values);

}

}

public static class YearPartitioner extends Partitioner<Text, Text>

{

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

{

String[] str = values.toString().split("\t");

if(str[7].equals("2011"))

{

return 0;

}

else if(str[7].equals("2012"))

{

return 1;

}

else if(str[7].equals("2013"))

{

return 2;

}

else if(str[7].equals("2014"))

{

return 3;

}

else if(str[7].equals("2015"))

{

return 4;

}

else

{

return 5;

}

}

}

public static class ReduceClass extends Reducer<Text, Text, NullWritable, Text>

{

public TreeMap<Long, Text> tm = new TreeMap<Long, Text>();

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

{

long count=0;

String tot="";

for(Text val:values)

{

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

if(str[1].equals("CERTIFIED"))

{

count++;

tot = str[7]+"\t"+key;

}

}

String totval = tot+"\t"+count;

tm.put(new Long(count), new Text(totval));

if(tm.size()>5)

{

tm.remove(tm.firstKey());

}

}

public void cleanup(Context con) throws IOException, InterruptedException

{

for(Text t:tm.descendingMap().values())

{

con.write(NullWritable.get(), t);

}

}

}

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

{

Configuration conf = new Configuration();

Job job = Job.getInstance(conf,"Most Data Scientist");

job.setJarByClass(loc.class);

job.setMapperClass(MapClass.class);

job.setPartitionerClass(YearPartitioner.class);

job.setReducerClass(ReduceClass.class);

job.setNumReduceTasks(6);

job.setMapOutputKeyClass(Text.class);

job.setMapOutputValueClass(Text.class);

job.setOutputKeyClass(NullWritable.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);

}

}

3)Which industry(SOC\_NAME) has the most number of Data Scientist positions?

[certified]

pro = load '/project/fin' using PigStorage('\t') AS (s\_no:int,case\_status:chararray, employer\_name:chararray, soc\_name:chararray, job\_title:chararray, full\_time\_position:chararray,prevailing\_wage:long,year:int, worksite:chararray, longitute:long, latitute:long);

--dump pro;

soc = filter pro by job\_title == 'DATA SCIENTIST' and case\_status == 'CERTIFIED';

--dump soc;

grp = group soc by $3;

--dump grp;

cou = foreach grp generate $0, COUNT(soc.job\_title) as tot;

--dump cou;

od = order cou by tot desc;

--dump od;

li = limit od 1;

dump li;

store li into '/home/hduser/Downloads/pro3ans';

4)Which top 5 employers file the most petitions each year? - Case Status - ALL

import java.io.IOException;

import java.util.TreeMap;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.NullWritable;

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 prtition4

{

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

{

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

{

String[] str = values.toString().split("\t");

String epname=str[2];

context.write(new Text(epname), new Text(values));

}

}

public static class Year extends Partitioner<Text, Text>

{

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

{

String[] str = values.toString().split("\t");

long year = Long.parseLong(str[7]);

if(year==2011)

{

return 0;

}

else if(year==2012)

{

return 1;

}

else if(year==2013)

{

return 2;

}

else if(year==2014)

{

return 3;

}

else if(year==2015)

{

return 4;

}

else

{

return 5;

}

}

}

public static class ReduceClass extends Reducer<Text,Text,NullWritable,Text>

{

private TreeMap<Long, Text> tm = new TreeMap<Long, Text>();

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

{

long count=0;

String year="";

for(Text val:values)

{

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

year = str[7];

count++;

}

String myValue = year+"\t"+key+"\t"+count;

tm.put(new Long(count), new Text(myValue));

if(tm.size()>5)

{

tm.remove(tm.firstKey());

}

}

protected void cleanup(Context context) throws IOException, InterruptedException

{

for(Text t:tm.descendingMap().values())

{

context.write(NullWritable.get(), t);

}

}

}

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

{

Configuration conf = new Configuration();

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

job.setJarByClass(prtition4.class);

job.setMapperClass(MapClass.class);

job.setPartitionerClass(Year.class);

job.setReducerClass(ReduceClass.class);

job.setNumReduceTasks(6);

job.setMapOutputKeyClass(Text.class);

job.setMapOutputValueClass(Text.class);

job.setOutputKeyClass(NullWritable.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);

}

}

5) Find the most popular top 10 job positions for H1B visa applications for each year?

a) for all the applications.

select job\_title,count(job\_title) as tot from h1b\_final where year = 2011 group by job\_title order by tot desc limit 10;

select job\_title,count(job\_title) as tot from h1b\_final where year = 2012 group by job\_title order by tot desc limit 10;

select job\_title,count(job\_title) as tot from h1b\_final where year = 2013 group by job\_title order by tot desc limit 10;

select job\_title,count(job\_title) as tot from h1b\_final where year = 2014 group by job\_title order by tot desc limit 10;

select job\_title,count(job\_title) as tot from h1b\_final where year = 2015 group by job\_title order by tot desc limit 10;

select job\_title,count(job\_title) as tot from h1b\_final where year = 2016 group by job\_title order by tot desc limit 10;

5b) for only certified applications.

select job\_title,count(job\_title) as tot from h1b\_final where year = 2011 and case\_status = 'CERTIFIED' group by job\_title order by tot desc limit 10;

select job\_title,count(job\_title) as tot from h1b\_final where year = 2012 and case\_status = 'CERTIFIED' group by job\_title order by tot desc limit 10;

select job\_title,count(job\_title) as tot from h1b\_final where year = 2013 and case\_status = 'CERTIFIED' group by job\_title order by tot desc limit 10;

select job\_title,count(job\_title) as tot from h1b\_final where year = 2014 and case\_status = 'CERTIFIED' group by job\_title order by tot desc limit 10;

select job\_title,count(job\_title) as tot from h1b\_final where year = 2015 and case\_status = 'CERTIFIED' group by job\_title order by tot desc limit 10;

select job\_title,count(job\_title) as tot from h1b\_final where year = 2016 and case\_status = 'CERTIFIED' group by job\_title order by tot desc limit 10;

6) Find the percentage and the count of each case status on total applications for each year. Create a line graph depicting the pattern of All the cases over the period of time.

pro = load '/project/fin' using PigStorage('\t') AS (s\_no:int,case\_status:chararray, employer\_name:chararray, soc\_name:chararray, job\_title:chararray, full\_time\_position:chararray,prevailing\_wage:long,year:int, worksite:chararray, longitute:long, latitute:long);

--dump pro;

grp1 = group pro by ($1,$7);

--dump grp1;

fla = foreach grp1 generate flatten(group), (float)COUNT(pro.case\_status);

--dump fla;

kk = group pro by year;

--dump kk;

fd = foreach kk generate $0, (float)COUNT(pro.$1);

--dump fd;

de = join fla by $1, fd by $0;

--dump de;

fr = foreach de generate $0,$1,$2,$4;

--dump fr;

fi = foreach fr generate $0,$1,$2, ROUND\_TO((($2/$3)\*100),2);

--dump fi;

om = order fi by $0 asc,$1 asc;

dump om;

store om into '/home/hduser/Downloads/6ans';

7) Create a bar graph to depict the number of applications for each year [All]

import java.io.\*;

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

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

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

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

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

import org.apache.hadoop.mapreduce.lib.output.\*;

public class coun {

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

{

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

{

try{

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

long year = Long.parseLong(str[7]);

long sno = Long.parseLong(str[0]);

context.write(new LongWritable(year),new LongWritable(sno));

}

catch(Exception e)

{

System.out.println(e.getMessage());

}

}

}

public static class ReduceClass extends Reducer<LongWritable,LongWritable,LongWritable,LongWritable>

{

//private IntWritable result = new IntWritable(0);

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

long count=0;

for (LongWritable val : values)

{

count++;

}

context.write(key, new LongWritable(count));

}

}

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

Configuration conf = new Configuration();

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

job.setJarByClass(coun.class);

job.setMapperClass(MapClass.class);

//job.setCombinerClass(ReduceClass.class);

job.setReducerClass(ReduceClass.class);

//job.setNumReduceTasks(2);

job.setMapOutputKeyClass(LongWritable.class);

job.setMapOutputValueClass(LongWritable.class);

job.setOutputKeyClass(LongWritable.class);

job.setOutputValueClass(LongWritable.class);

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

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

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

}

}

8) Find the average Prevailing Wage for each Job for each Year (take part time and full time separate). Arrange the output in descending order - [Certified and Certified Withdrawn.]

FOR FULL TIME JOB:

pro = load '/project/fin' using PigStorage('\t') AS (s\_no:int,case\_status:chararray, employer\_name:chararray, soc\_name:chararray, job\_title:chararray, full\_time\_position:chararray,prevailing\_wage:long,year:chararray, worksite:chararray, longitute:long, latitute:long);

--dump pro;

kk = filter pro by case\_status == 'CERTIFIED' or case\_status == 'CERTIFIED-WITHDRAWN';

--dump kk;

pr = filter kk by $5 == 'Y';

--dump pr;

kl = group pr by ($7,$4);

--dump kl;

ad = foreach kl generate flatten(group), AVG(pr.$6);

--dump ad;

hy = filter ad by $0 == '2011';

--dump hy;

uy = order hy by $2 desc;

--dump uy;

hi = filter ad by $0 == '2012';

--dump hi;

uy12 = order hi by $2 desc;

--dump uy12;

hyu = filter ad by $0 == '2013';

--dump hyu;

uy13 = order hyu by $2 desc;

--dump uy13;.

hy14 = filter ad by $0 == '2014';

--dump hy14;

uy14 = order hy14 by $2 desc;

--dump uy14;

hy15 = filter ad by $0 == '2015';

--dump hy15;

uy15 = order hy15 by $2 desc;

--dump uy15;

hy16 = filter ad by $0 == '2016';

--dump hy16;

uy16 = order hy16 by $2 desc;

--dump uy16;

uni = UNION uy,uy12,uy13,uy14,uy15,uy16;

--dump uni;

de = order uni by $0 asc;

dump de;

store de into '/home/hduser/Downloads/8yans';

FOR PART TIME:

pro = load '/project/fin' using PigStorage('\t') AS (s\_no:int,case\_status:chararray, employer\_name:chararray, soc\_name:chararray, job\_title:chararray, full\_time\_position:chararray,prevailing\_wage:long,year:chararray, worksite:chararray, longitute:long, latitute:long);

--dump pro;

kk = filter pro by case\_status == 'CERTIFIED' or case\_status == 'CERTIFIED-WITHDRAWN';

--dump kk;

pr = filter kk by $5 == 'N';

--dump pr;

kl = group pr by ($7,$4);

--dump kl;

ad = foreach kl generate flatten(group), AVG(pr.$6);

--dump ad;

hy = filter ad by $0 == '2011';

--dump hy;

uy = order hy by $2 desc;

--dump uy;

hi = filter ad by $0 == '2012';

--dump hi;

uy12 = order hi by $2 desc;

--dump uy12;

hyu = filter ad by $0 == '2013';

--dump hyu;

uy13 = order hyu by $2 desc;

--dump uy13;.

hy14 = filter ad by $0 == '2014';

--dump hy14;

uy14 = order hy14 by $2 desc;

--dump uy14;

hy15 = filter ad by $0 == '2015';

--dump hy15;

uy15 = order hy15 by $2 desc;

--dump uy15;

hy16 = filter ad by $0 == '2016';

--dump hy16;

uy16 = order hy16 by $2 desc;

--dump uy16;

uni = UNION uy,uy12,uy13,uy14,uy15,uy16;

--dump uni;

de = order uni by $0 asc;

dump de;

store de into '/home/hduser/Downloads/8nans';

9) Which are the employers along with the number of petitions who have the success rate more than 70% in petitions. (total petitions filed 1000 OR more than 1000) ?.

pro = load '/project/fin' using PigStorage('\t') AS (s\_no:int,case\_status:chararray, employer\_name:chararray, soc\_name:chararray, job\_title:chararray, full\_time\_position:chararray,prevailing\_wage:long,year:int, worksite:chararray, longitute:long, latitute:long);

--dump pro;

kk = group pro by employer\_name;

--dump kk;

pp = foreach kk generate $0, (float)COUNT(pro.employer\_name);

--dump pp;

aa = filter pro by case\_status == 'CERTIFIED' or case\_status == 'CERTIFIED-WITHDRAWN';

--dump aa;

ff = group aa by $2;

ss = foreach ff generate $0, (float)COUNT(aa.$1);

--dump ss;

--aa1 = filter pro by case\_status == 'CERTIFIED-WITHDRAWN';

--dump aa1;

--ss1 = foreach ff generate $0, (float)COUNT(aa.$1);

--dump ss1;

dd = join ss by $0, pp by $0;

--dump dd;

gh = foreach dd generate $0,$1,$3;

--dump gh;

vv = foreach gh generate $0,$2,($1/$2)\*100;

--dump vv;

fg = filter vv by $1>=1000;;

--dump fg;

re = filter fg by $2>70.0;

--dump re;

fr = order re by $2 desc;

dump fr;

store fr into '/home/hduser/Downloads/9ans';

10) Which are the job positions along with the number of petitions which have the success rate more than 70% in petitions (total petitions filed 1000 OR more than 1000)?

pro = load '/project/fin' using PigStorage('\t') AS (s\_no:int,case\_status:chararray, employer\_name:chararray, soc\_name:chararray, job\_title:chararray, full\_time\_position:chararray,prevailing\_wage:long,year:int, worksite:chararray, longitute:long, latitute:long);

--dump pro;

kk = group pro by job\_title;

--dump kk;

pp = foreach kk generate $0, (float)COUNT(pro.job\_title);

--dump pp;

aa = filter pro by case\_status == 'CERTIFIED' or case\_status == 'CERTIFIED-WITHDRAWN';

--dump aa;

ff = group aa by $4;

ss = foreach ff generate $0, (float)COUNT(aa.$1);

--dump ss;

--aa1 = filter pro by case\_status == 'CERTIFIED-WITHDRAWN';

--dump aa1;

--ss1 = foreach ff generate $0, (float)COUNT(aa.$1);

--dump ss1;

dd = join ss by $0, pp by $0;

--dump dd;

gh = foreach dd generate $0,$1,$3;

--dump gh;

vv = foreach gh generate $0,$2,($1/$2)\*100;

--dump vv;

fg = filter vv by $1>=1000;;

--dump fg;

re = filter fg by $2>70.0;

--dump re;

fr = order re by $2 desc;

dump fr;

store fr into '/user/hive/warehouse/niit.db/h1b\_final/out';

11) Export result for question no 10 to MySql database.

sqoop export --connect jdbc:mysql://localhost/proj10 --username root --password 'secureword' --table sqoo --update-mode allowinsert --update-key job\_title --export-dir /user/warehouse/hive/niit.db/h1b\_final/out10 --input-fields-terminated-by '\t' ;

**Shell script**

#!/bin/bash

show\_menu()

{

NORMAL=`echo "\033[m"`

MENU=`echo "\033[36m"` #Blue

NUMBER=`echo "\033[33m"` #yellow

FGRED=`echo "\033[41m"`

RED\_TEXT=`echo "\033[31m"`

ENTER\_LINE=`echo "\033[33m"`

echo -e "${MENU}\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*APP MENU\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*${NORMAL}"

echo -e "${MENU}\*\*${NUMBER} 1)${MENU} 1a ${NORMAL}"

echo -e "${MENU}\*\*${NUMBER} 2)${MENU} 1b ${NORMAL}"

echo -e "${MENU}\*\*${NUMBER} 3)${MENU} 2a ${NORMAL}"

echo -e "${MENU}\*\*${NUMBER} 4)${MENU} 2b ${NORMAL}"

echo -e "${MENU}\*\*${NUMBER} 5)${MENU} 3 ${NORMAL}"

echo -e "${MENU}\*\*${NUMBER} 6)${MENU} 4 ${NORMAL}"

echo -e "${MENU}\*\*${NUMBER} 7)${MENU} 5a ${NORMAL}"

echo -e "${MENU}\*\*${NUMBER} 8)${MENU} 5b ${NORMAL}"

echo -e "${MENU}\*\*${NUMBER} 9)${MENU} 6 ${NORMAL}"

echo -e "${MENU}\*\*${NUMBER} 10)${MENU} 7 ${NORMAL}"

echo -e "${MENU}\*\*${NUMBER} 11)${MENU} 8 ${NORMAL}"

echo -e "${MENU}\*\*${NUMBER} 12)${MENU} 9 ${NORMAL}"

echo -e "${MENU}\*\*${NUMBER} 13)${MENU} 10 ${NORMAL}"

echo -e "${MENU}\*\*${NUMBER} 14)${MENU} 11 ${NORMAL}"

echo -e "${MENU}\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*${NORMAL}"

echo -e "${ENTER\_LINE}Please enter a menu option and enter or ${RED\_TEXT}enter to exit. ${NORMAL}"

read opt

}

function option\_picked()

{

COLOR='\033[01;31m' # bold red

RESET='\033[00;00m' # normal white

MESSAGE="$1" #modified to post the correct option selected

echo -e "${COLOR}${MESSAGE}${RESET}"

}

function getpinCodeBank(){

echo "in getPinCodebank"

echo $1

echo $2

#hive -e "Select \* from AppData where PinCode = $1 AND Bank = '$2'"

}

clear

show\_menu

while [ opt != '' ]

do

if [[ $opt = "" ]]; then

exit;

else

case $opt in

1) clear;

option\_picked "1a) Is the number of petitions with Data Engineer job title increasing over time?";

bash /home/hduser/Desktop/1a.sh

show\_menu;

;;

2) clear;

option\_picked "1b) Find top 5 job titles who are having highest avg growth in applications ";

pig /home/hduser/1b.pig

show\_menu;

;;

3) clear;

option\_picked "2a) Which part of the US has the most Data Engineer jobs for each year";

bash /home/hduser/Desktop/3.sh

show\_menu;

;;

4) clear;

option\_picked "2b) find top 5 locations in the US who have got certified visa for each year";

bash /home/hduser/Desktop/2b.sh

show\_menu;

;;

5) clear;

option\_picked "3)Which industry(SOC\_NAME) has the most number of Data Scientist positions?";

pig /home/hduser/3.pig

show\_menu;

;;

6) clear;

option\_picked "4)Which top 5 employers file the most petitions each year?";

bash /home/hduser/Desktop/4.sh

show\_menu;

;;

7) clear;

option\_picked "5) Find the most popular top 10 job positions for H1B visa applications for each year?a) for all the applications";

bash /home/hduser/Desktop/5a.sh

show\_menu;

;;

8) clear;

option\_picked "5) Find the most popular top 10 job positions for H1B visa applications for each year?b) for only certified applications.";

bash /home/hduser/Desktop/5b.sh

show\_menu;

;;

9) clear;

option\_picked "6) Find the percentage and the count of each case status on total applications for each year. Create a line graph depicting the pattern of All the cases over the period of time ";

pig /home/hduser/6.pig

show\_menu;

;;

10) clear;

option\_picked "7) Create a bar graph to depict the number of applications for each year";

bash /home/hduser/Desktop/7.sh

show\_menu;

;;

11) clear;

option\_picked "8) Find the average Prevailing Wage for each Job for each Year (take part time and full time separate). Arrange the output in descending order";

echo -e "${MENU}Select Full Time Job or Part Time Job ${NORMAL}"

echo -e "${MENU}\*\*${NUMBER} 1)${MENU} Full Time Job ${NORMAL}"

echo -e "${MENU}\*\*${NUMBER} 2)${MENU} Part Time Job ${NORMAL}"

read job

case $job in

1) echo "FULL TIME JOB SELECTED"

pig /home/hduser/8y.pig

;;

2) echo "PART TIME JOB SELECTED"

pig /home/hduser/8n.pig

;;

\*) echo "Please Select one among the option[1-2]";;

esac

show\_menu;

;;

12) clear;

option\_picked "9) Which are the employers along with the number of petitions who have the success rate more than 70% in petitions. (total petitions filed 1000 OR more than 1000) ?";

pig /home/hduser/9.pig

show\_menu;

;;

13) clear;

option\_picked "10) Which are the job positions along with the number of petitions which have the success rate more than 70% in petitions (total petitions filed 1000 OR more than 1000)?";

pig /home/hduser/10.pig

show\_menu;

;;

14) clear;

option\_picked "11) Export result for question no 10 to MySql database.";

sqoop export --connect jdbc:mysql://localhost/proj10 --username root --P --table sqoo --update-mode allowinsert --update-key job\_title --export-dir /user/hive/warehouse/niit.db/h1b\_final/out --input-fields-terminated-by '\t' ;

show\_menu;

;;

\n) exit;

;;

\*) clear;

option\_picked "Pick an option from the menu";

show\_menu;

;;

esac

fi

done.