**APPENDIX 1**

**SAMPLE CODE**

**PREPROCESSING DATA**

sed 's/"//g' account.asc | sed -e '1d' |sed 's/;/,/g'> acccount.csv

sed 's/"//g' card.asc | sed -e '1d' |sed 's/;/,/g'> card.csv

sed 's/"//g' client.asc | sed -e '1d' |sed 's/;/,/g'> client.csv

sed 's/"//g' district.asc | sed -e '1d' |sed 's/;/,/g'> district.csv

sed 's/"//g' disp.asc | sed -e '1d' |sed 's/;/,/g'> disp.csv

sed 's/"//g' loan.asc | sed -e '1d' |sed 's/;/,/g'> loan.csv

sed 's/"//g' order.asc | sed -e '1d' |sed 's/;/,/g'> order.csv

sed 's/"//g' trans.asc | sed -e '1d' |sed 's/;/,/g'> trans.csv

**HADOOP CODE**

hadoop dfs -put project

loan = load '/project/loan.csv' using PigStorage(',') ;

loan\_fields = foreach loan generate $1 as ac\_id,$0 as loan\_id,$3 as amount,$6 as status;

grp\_loan\_ac\_id = group loan\_fields by $0;

grp\_loan\_ac\_id\_flatten = foreach grp\_loan\_ac\_id generate FLATTEN(loan\_fields);

filtered\_grp = filter grp\_loan\_ac\_id\_flatten by $3 != 'status';

store filtered\_grp into '/project/loan\_required\_out' using PigStorage(',');

loan = load '/project/loan.csv' using PigStorage(',') ;

loan\_fields = foreach loan generate $1 as ac\_id,$0 as loan\_id,$3 as amount,$6 as status;

grp\_loan\_ac\_id = group loan\_fields by $0;

grp\_loan\_ac\_id\_flatten = foreach grp\_loan\_ac\_id generate FLATTEN(loan\_fields);

filtered\_grp = filter grp\_loan\_ac\_id\_flatten by $3 != 'status';

store filtered\_grp into '/project/loan\_required\_out' using PigStorage(',');

REGISTER /home/training/Desktop/calc.jar

client = load '/project/client.csv' using PigStorage(',') AS (client\_id:int,birth\_number:chararray,district\_id:int) ;

client\_fields = foreach client generate $0 as client\_id,$2 as district\_id,$1 as birthday\_n\_sex;

grp\_client\_id = group client\_fields by $0;

grp\_client\_flat = FOREACH grp\_client\_id GENERATE FLATTEN(client\_fields);

B = filter grp\_client\_flat by $2 != 'birth\_number';

age = foreach B generate $0,$1, bank.calc(birthday\_n\_sex) ;

store age into '/project/age\_required\_out' using PigStorage(',');

transaction = load '/project/trans.csv' using PigStorage(',') as (trans\_id:int,ac\_id:int,date:chararray,type:chararray,operation:chararray,amount:int,bal:int,k\_sym:chararray,bank:int,account:int);

transaction\_fields = foreach transaction generate $1 as ac\_id,$2 as date\_of\_transaction,$3 as transaction\_type,$5 as amount,$6 as bal\_post\_trnsaction;

filtered\_trans = filter transaction\_fields by (int)SUBSTRING($1,0,2) > 97;

grp\_ac = group filtered\_trans by $0;

MAX\_grp\_ac = FOREACH grp\_ac GENERATE group, SUM(filtered\_trans.$3),SUM(filtered\_trans.$4);

store MAX\_grp\_ac into '/project/transaction\_left\_bal\_required\_out' using PigStorage(',');

card = load '/project/card.csv' using PigStorage(',') ;

card\_fields = foreach card generate $1 as disposition\_id,$2 as card\_type;

grp\_card\_disp\_id = group card\_fields by $0;

flatten\_card = foreach grp\_card\_disp\_id generate FLATTEN(card\_fields);

filtered\_card = filter flatten\_card by card\_type != 'type';

store filtered\_card into '/project/card\_required\_out' using PigStorage(',');

REGISTER /home/training/Desktop/unemp.jar

district = load '/project/district.csv' using PigStorage(',') AS (dist\_id:int,dist\_name:chararray,region:chararray,no\_inhabs:long,mun\_499:int,mun\_1999:int,mun\_10k:int,mun\_more:int,no\_of\_cities:int,no\_of\_urban\_inhabs:double

,avg\_sal:int,unemp\_95:double,unemp\_96:double,entre\_ratio:int);

district\_fields = foreach district generate $0 as district\_id,$1 as district\_name,$2 as region,$10 as avg\_salary,$11 as unemp\_rate\_95,$12 as unemp\_rate\_96,$13 as entrepreneur\_per\_1000;

grp\_dist\_id = group district\_fields by $0;

MAX\_grp\_dist = FOREACH grp\_dist\_id GENERATE group,FLATTEN(district\_fields);

B = filter MAX\_grp\_dist by unemp\_rate\_95 > 0.0 AND unemp\_rate\_96 > 0.0;

unem\_percentage = foreach B generate $1, district\_name,avg\_salary,bank.unemp(unemp\_rate\_95,unemp\_rate\_96),entrepreneur\_per\_1000 ;

store unem\_percentage into '/project/district\_required\_out' using PigStorage(',');

disposition = load '/project/disp.csv' using PigStorage(',') ;

disposition\_fields = foreach disposition generate $2 as ac\_id,$0 as disposition\_id,$3 as disposition\_type,$1 as client\_id;

grp\_disposition\_disp\_id = group disposition\_fields by $1;

flatten\_disposition\_disp\_id = foreach grp\_disposition\_disp\_id generate FLATTEN(disposition\_fields);

filtered\_disposition\_disp\_id = filter flatten\_disposition\_disp\_id by disposition\_type != 'type';

Client\_age = load '/project/age\_required\_out' using PigStorage(',') AS (client\_id:int,dist\_id:int,age:double,sex:chararray);

card\_type = load '/project/card\_required\_out' using PigStorage(',') AS (disp\_id:int,type:chararray);

transaction\_sum = load '/project/transaction\_left\_bal\_required\_out' using PigStorage(',') AS (ac\_id:int,trans\_sum:long,bal\_sum:long);

loan\_status = load '/project/loan\_required\_out' using PigStorage(',') AS (ac\_id:int,loan\_id:int,amount:int,status:chararray);

district\_info = load '/project/district\_required\_out' using PigStorage(',') AS (district\_id:int,dist\_name:chararray,avg\_sal:int,unemprate:float,entrepreneur:int);

join\_disp\_client = join filtered\_disposition\_disp\_id by $3,Client\_age by $0;

join\_disp\_client\_card = join join\_disp\_client by $1,card\_type by $0;

join\_disp\_client\_card\_district = join join\_disp\_client\_card by $5,district\_info by $0;

join\_disp\_client\_card\_district\_trans\_loan = join join\_disp\_client\_card\_district by $0,transaction\_sum by $0,loan\_status by $0;

pick\_fields = foreach join\_disp\_client\_card\_district\_trans\_loan generate $0 as ac\_id,$2 as disp\_type,$9 as card\_type,$11 as dist\_name,$12 as avg\_sal,$13 as unemp\_rate,$14 as no\_of\_entre,$16 as transaction\_sum,$20 as loan\_amount,$21 as loan\_status;

store pick\_fields into '/project/combined\_out' using PigStorage(',');

Good = filter pick\_fields by transaction\_sum > 1000000 AND avg\_sal > 10000 AND loan\_status == 'A' ;

store Good into '/project/VIP\_customer' using PigStorage(',');

Normal = filter pick\_fields by transaction\_sum < 1000000 AND transaction\_sum >150000 AND avg\_sal > 6000 AND (loan\_status=='A' OR loan\_status=='C') AND no\_of\_entre < 0.80;

store Normal into '/project/good\_customer' using PigStorage(',');

Risky = filter pick\_fields by avg\_sal > 6000 AND (loan\_status == 'B' OR loan\_status == 'D') AND no\_of\_entre > 100;

store Risky into '/project/risky\_customer' using PigStorage(',');

Now, we have three categories of customers.

hadoop dfs -copyToLocal /project/combined\_out /home/training/Desktop

**JAR FILES**

**AGE JAR FILE**

package bank;

import java.io.IOException;

import java.text.ParseException;

import java.text.SimpleDateFormat;

import java.util.Date;

import org.apache.pig.EvalFunc;

import org.apache.pig.data.Tuple;

import org.joda.time.DateTime;

import org.joda.time.Period;

import org.joda.time.ReadableInstant;

@SuppressWarnings("unused")

public class age extends EvalFunc<String>{

@Override

public String exec(Tuple input) throws IOException {

if (input == null || input.size() == 0)

return null;

try{

String s = (String)input.get(0);

String month = s.substring(2,4);

if(Integer.*parseInt*(month)>50)

{

int x = Integer.*parseInt*(month)-50;

if(x<10)

month = "0"+String.*valueOf*(x);

else

month = String.*valueOf*(x);

}

String str = s.substring(0,2).concat(month).concat(s.substring(4,6));

Date date = new Date();

DateTime today = new DateTime(date);

Date dob = null;

try {

dob = new SimpleDateFormat("yyMMdd").parse(str);

}

catch (ParseException e) {

e.printStackTrace();

}

DateTime dt\_dob = new DateTime(dob);

Period period = new Period(dt\_dob,today);

int age\_res = period.getYears();

return String.*valueOf*(age\_res);

}

catch(Exception e){

throw new IOException("exception caught"+e.getMessage(),e);

}

}

}

**UNEMPLOYMENT CALCULATION JAR**

package bank;

import java.io.IOException;

import org.apache.pig.EvalFunc;

import org.apache.pig.data.Tuple;

public class unemp extends EvalFunc<Double>{

public Double exec(Tuple input) throws IOException {

if (input == null || input.size() == 0)

return null;

Double emp\_1 = (Double)input.get(0);

Double emp\_2 = (Double)input.get(1);

if(emp\_2 > emp\_1)

return emp\_2-emp\_1;

else

return emp\_1-emp\_2;

}

}

**R CODE FOR CLUSTERING**

**CLUSTERING OF TRANSACTIONLESS DATA**

library(readxl)

vipcsv <- read.csv(file.choose(),header=T)

summary(vipcsv)

colnames(vipcsv) <- c("ac\_id","disposal\_type","card\_type","dist","avg\_sal","unemp\_rate","entrepreneur\_no","trans\_sum","loan\_amount","loan\_status")

vipcsv[!complete.cases(vipcsv),]

vipcsv[,!complete.cases(vipcsv)]

## If any missing values are there omit them

vipcsv <- na.omit(vipcsv,na.action=TRUE)

##selecting only numerical data and removing ac\_id column

mydata <- vipcsv[,c(5,9,6,7)]

mydata

boxplot(mydata)

boxplot(mydata[,c(2)])

plot(mydata[,c(2)])

mydata <- scale(mydata[,1:4])

mydata

wss <- (nrow(mydata)-1)\*sum(apply(mydata,2,var))

wss

for(i in 2:15)wss[i]<- sum(fit=kmeans(mydata,centers=i,15)$withinss)

plot(1:15,wss,type="b",main="15 clusters",xlab="no. of cluster",ylab="with clsuter sum of squares")

plot(1:15,wss,type="b",main="15 clusters",xlab="no. of cluster",ylab="with clsuter sum of squares")

fit <- kmeans(mydata,3)

fit

fit$withinss

fit$betweenss

fit$cluster

plot(mydata,col=fit$cluster,pch=15)

points(fit$centers,col=1:7,pch=3)

library(cluster)

#install.packages("fpc")

library(fpc)

plotcluster(mydata,fit$cluster)

points(fit$centers,col=1:8,pch=16)

plotcluster(mydata,fit$cluster)

points(fit$centers,col=1:8,pch=16)

clusplot(mydata, fit$cluster, color=TRUE, shade=TRUE, labels=2, lines=0)

mydata <- vipcsv[,c(5:9)]

mydata <- data.frame(mydata,fit$cluster)

cluster\_mean <- aggregate(mydata[,1:6],by = list(fit$cluster),FUN = mean)

cluster\_mean

clusplot(mydata, fit$cluster, color=TRUE, shade=TRUE, labels=2, lines=0)

plot(mydata,col=fit$cluster,pch=15)

points(fit$centers,col=1:8,pch=3)

loan <- c(16,161,5)

loan\_percent=round((100\*loan)/sum(loan), 1)

pie(loan,labels=loan\_percent,main= "without transaction" , col=rainbow(3))

legend("topright",c("VIP","GOOD","RISKY"),cex=0.8,fill=rainbow(3))

trans <- c(8,43,5)

trans\_percent=round((100\*loan)/sum(trans), 1)

pie(trans,labels=trans\_percent,main= "transaction" , col=rainbow(5))

legend("topright",c("VIP","GOOD","RISKY"),cex=0.8,fill=rainbow(5))

dev.off()

plot(rnorm(50),rnorm(50))

**CLUSTERING CODE FOR TRANSACTIONAL DATA**

library(readxl)

vipcsv <- read.csv(file.choose(),header=T)

summary(vipcsv)

colnames(vipcsv) <- c("ac\_id","disposal\_type","card\_type","dist","avg\_sal","unemp\_rate","entrepreneur\_no","trans\_sum","loan\_amount","loan\_status")

vipcsv[!complete.cases(vipcsv),]

vipcsv[,!complete.cases(vipcsv)]

## If any missing values are there omit them

vipcsv <- na.omit(vipcsv,na.action=TRUE)

##selecting only numerical data and removing ac\_id column

mydata <- vipcsv[,c(5,8,6,7,9)]

mydata

boxplot(mydata)

boxplot(mydata[,c(5)])

plot(mydata[,c(5)])

mydata <- scale(mydata[,1:5])

mydata

wss <- (nrow(mydata)-1)\*sum(apply(mydata,2,var))

wss

for(i in 2:15)wss[i]<- sum(fit=kmeans(mydata,centers=i,15)$withinss)

plot(1:15,wss,type="b",main="15 clusters",xlab="no. of cluster",ylab="with clsuter sum of squares")

plot(1:15,wss,type="b",main="15 clusters",xlab="no. of cluster",ylab="with clsuter sum of squares")

fit <- kmeans(mydata,3)

fit

fit$withinss

fit$betweenss

fit$cluster

plot(mydata,col=fit$cluster,pch=15)

points(fit$centers,col=1:8,pch=3)

library(cluster)

#install.packages("fpc")

library(fpc)

plotcluster(mydata,fit$cluster)

points(fit$centers,col=1:8,pch=16)

plotcluster(mydata,fit$cluster)

points(fit$centers,col=1:8,pch=16)

clusplot(mydata, fit$cluster, color=TRUE, shade=TRUE, labels=2, lines=0)

mydata <- vipcsv[,c(5:10)]

mydata <- data.frame(mydata,fit$cluster)

cluster\_mean <- aggregate(mydata[,1:6],by = list(fit$cluster),FUN = mean)

cluster\_mean

clusplot(mydata, fit$cluster, color=TRUE, shade=TRUE, labels=2, lines=0)

plot(mydata,col=fit$cluster,pch=15)

points(fit$centers,col=1:8,pch=3)

**APPENDIX 2**

**PROJECT CREDENTIALS**







