**Code for HMDA Data Challenge**

setwd('C:\\Users\\dalalshobhit\\Desktop\\CapitalOneData')

# Provide paths for loan data and institution data (Data location on local computer)

loanDataPath <- "C:\\Users\\dalalshobhit\\Desktop\\CapitalOneData\\2012\_to\_2014\_loans\_data.csv"

institutionDataPath <- "C:\\Users\\dalalshobhit\\Desktop\\CapitalOneData\\2012\_to\_2014\_institutions\_data.csv"

# hmda\_init() implements a method to merge and get expanded data

hmda\_init <- function() {

loanData <- read.csv(loanDataPath, header = TRUE)

typeof(loanData)

institutionData <- read.csv(institutionDataPath, header = TRUE)

mergedData <- merge(loanData, institutionData, by=c("Agency\_Code","Respondent\_ID", "As\_of\_Year"), all.x = TRUE, na.rm = TRUE)

return(mergedData)

}

# call hmda\_init() and create a data frame mergedExpandedData

mergedExpandedData <- hmda\_init()

##################################################################################

library(jsonlite) # import jsonlite library

# hmda\_to\_json(data, states, conventional\_conforming) exports filtered data in json to disk

hmda\_to\_json <- function(data, state, conventional\_conforming) {

# if else loops for filtering data

if(missing(state) ) {

if(missing(conventional\_conforming) ){ # if state is missing & not conventional\_conforming

filteredData <- data

}

else { # if state is missing & conventional\_conforming

filteredData <- subset(data, Conventional\_Conforming\_Flag == conventional\_conforming)

}

}

else {

if(missing(conventional\_conforming)) { # if state is given & not conventional\_conforming

filteredData <- subset(data, State == state)

}

else { # if state is given & conventional\_conforming

filteredData <- subset(data, State == state & Conventional\_Conforming\_Flag == conventional\_conforming)

}

}

# Convert to JSON data

sink("jsonData.json")

cat(toJSON(filteredData))

sink()

# export the expanded dataset to disk (Give a path on your local machine where you want to write this file)

write.table(filteredData, "C:\\Users\\dalalshobhit\\Desktop\\CapitalOneData\\jsonData.json", sep=":")

return(filteredData)

}

# call hmda\_to\_json function and store it in showFilteredData

showFilteredData <- hmda\_to\_json(mergedExpandedData,,"N")

######################################################################################

library(maps)

library(mapproj)

library(sqldf)

# Create SQL queries for each type of loan grouped by year

conventionalLoanSQL <- "SELECT As\_of\_Year,count(\*) AS Conventional\_Loans FROM mergedExpandedData WHERE Loan\_Type\_Description LIKE 'Conventional' GROUP BY As\_of\_Year"

VALoanSQL <- "SELECT As\_of\_Year,count(\*) AS VA\_Loans FROM mergedExpandedData WHERE Loan\_Type\_Description LIKE 'VA guaranteed' GROUP BY As\_of\_Year"

FSA\_RHS\_LoanSQL <- "SELECT As\_of\_Year,count(\*) AS FSA\_RHS\_Loans FROM mergedExpandedData WHERE Loan\_Type\_Description LIKE 'FSA/RHS guaranteed' GROUP BY As\_of\_Year"

FHALoanSQL <- "SELECT As\_of\_Year,count(\*) AS FHA\_Loans FROM mergedExpandedData WHERE Loan\_Type\_Description LIKE 'FHA insured' GROUP BY As\_of\_Year"

# Create tables for each type of loans

conventionalLoans <- sqldf(conventionalLoanSQL)

VALoans <- sqldf(VALoanSQL)

FSA\_RHSLoans <- sqldf(FSA\_RHS\_LoanSQL)

FHA\_Loans <- sqldf(FHALoanSQL)

# Query to merge all tables

all\_LoansSQL <- "SELECT c.As\_of\_Year, c.Conventional\_Loans, v.VA\_Loans, f.FSA\_RHS\_Loans, fh.FHA\_Loans

FROM ((conventionalLoans c INNER JOIN VALoans v

ON c.As\_of\_Year = v.As\_of\_Year) INNER JOIN FSA\_RHSLoans f

ON c.As\_of\_Year = f.As\_of\_Year) INNER JOIN FHA\_Loans fh

ON c.As\_of\_Year = fh.As\_of\_Year"

allLoans <- sqldf(all\_LoansSQL)

library(ggplot2)

library(reshape)

dfm <- melt(allLoans[,c('As\_of\_Year','Conventional\_Loans','VA\_Loans', 'FSA\_RHS\_Loans', 'FHA\_Loans')],id.vars = 1)

# Bar plot each type of loans grouped by year

options(scipen=10000)

ggplot(dfm, aes(x=As\_of\_Year,y=value)) +

geom\_bar(stat='identity', aes(fill=variable), position='dodge')

###################################################################################

# Create SQL queries for conventional\_conforming loans vs. not conventional\_conforming loans

# grouped by state

ConventionalConformingLoanSQL <- "SELECT State,count(\*) AS ConventionalConforming\_Loans FROM mergedExpandedData WHERE Conventional\_Conforming\_Flag LIKE 'Y' GROUP BY State"

ConventionalConformingLoans <- sqldf(ConventionalConformingLoanSQL)

NonConventionalConformingLoanSQL <- "SELECT State,count(\*) AS NonConventionalConforming\_Loans FROM mergedExpandedData WHERE Conventional\_Conforming\_Flag LIKE 'N' GROUP BY State"

NonConventionalConformingLoans <- sqldf(NonConventionalConformingLoanSQL)

totalLoansSQL <- "SELECT c.State, c.ConventionalConforming\_Loans, n.NonConventionalConforming\_Loans

FROM ConventionalConformingLoans c INNER JOIN NonConventionalConformingLoans n

ON c.State = n.State

ORDER BY 1"

totalLoans <- sqldf(totalLoansSQL)

totLoans <- melt(totalLoans[,c('State','ConventionalConforming\_Loans','NonConventionalConforming\_Loans')],id.vars = 1)

# Bar plot each type of loans grouped by year

options(scipen=10000)

ggplot(totLoans, aes(x=State,y=value)) +

geom\_bar(stat='identity', aes(fill=variable), position='dodge')

######################################################################################

# Pie chart of total loans grouped by State (no segregation of loans)

total\_loansSQL <- "SELECT State, COUNT(\*) AS Tot\_loans FROM mergedExpandedData GROUP BY State"

total\_loans <- sqldf(total\_loansSQL)

Loan\_Count <- as.vector(total\_loans['Tot\_loans'])

State <- as.vector(total\_loans['State'])

pct <- round(Loan\_Count/sum(Loan\_Count)\*100)

pct <- as.integer(unlist(pct))

State <- unlist(State)

lbls <- paste(State, pct) # add percents to labels

lbls <- paste(lbls,"%",sep="") # ad % to labels

pie(pct,labels = lbls, col=c("purple", "violetred1", "green3",

"cornsilk", "cyan", "white"))

##################################################################################################

mergedExpandedData <- hmda\_init()

# Create a function cleaningData() for Quality Assssment of Applicant\_Income\_000

cleaningData <- function(mergedExpandedData) {

mergedExpandedData$Applicant\_Income\_000[mergedExpandedData$Applicant\_Income\_000=='NA '] = '0'

mergedExpandedData$Applicant\_Income\_000=as.numeric(as.character(mergedExpandedData$Applicant\_Income\_000))

class(mergedExpandedData$Applicant\_Income\_000)

# Cleaning the data

meanIncome <- mean(mergedExpandedData$Applicant\_Income\_000)

mergedExpandedData$Applicant\_Income\_000[mergedExpandedData$Applicant\_Income\_000==0] = meanIncome

return(mergedExpandedData)

}

clean\_data <- cleaningData(mergedExpandedData)

##################################################################################################

# Create a function cleaning\_LoanAmount() for Quality Assssment of Loan\_Amount\_000

cleaning\_LoanAmount <- function(clean\_data) {

clean\_data$Loan\_Amount\_000=as.numeric(as.character(clean\_data$Loan\_Amount\_000))

class(clean\_data$Loan\_Amount\_000)

# clean\_data$Loan\_Amount\_000[ which(clean\_data$Loan\_Amount\_000 < 999)]

clean\_data$Loan\_Amount\_000[clean\_data$Loan\_Amount\_000>999 & clean\_data$Loan\_Amount\_000<99999] =

clean\_data$Loan\_Amount\_000/1000

return(clean\_data)

}

# Call cleaning\_LoanAmount(data) function for Quality assessment of Loan\_Amount\_000

clean\_data <- cleaning\_LoanAmount(clean\_data)

cleanedData <- clean\_data$Loan\_Amount\_000[ which(clean\_data$Loan\_Amount\_000 > 999)]

length(clean\_data$Loan\_Amount\_000[ which(clean\_data$Loan\_Amount\_000 > 999)])

clean\_data$Loan\_Amount\_000[clean\_data$Loan\_Amount\_000 < 999]

##################################################################################################

library(plyr)

# Mean applicant income grouped by state and county

mean\_Data <- aggregate(clean\_data$Applicant\_Income\_000,by=list(x=clean\_data$State,y=clean\_data$County\_Name),mean,na.rm=TRUE)

class(mainData)

mean(clean\_data$Applicant\_Income\_000[clean\_data$County\_Name=='RUSSELL' & clean\_data$State=='VA'])

##################################################################################################

# Mean applicant income grouped by state

mean\_income\_state <- aggregate(clean\_data$Applicant\_Income\_000,by=list(State=clean\_data$State),mean,na.rm=TRUE)

ggplot(mean\_income\_state, aes(x=reorder(State,-x),y=x)) +

geom\_bar(stat='identity', aes(fill=x)) +

labs(x="State", y="Avg. applicant income (000's)")

##################################################################################################

# Create SQL queries for analyzing data for Maryland

Maryland\_dataSQL <- "SELECT County\_Name,count(\*) AS no\_of\_loans,Conforming\_Limit\_000 FROM clean\_data WHERE State LIKE 'MD' GROUP BY County\_Name"

Maryland\_data <- sqldf(Maryland\_dataSQL)

Maryland\_data <- Maryland\_data[order(-Maryland\_data$no\_of\_loans),]

##################################################################################################

# Create SQL queries for analyzing data for Virginia

Virginia\_dataSQL <- "SELECT County\_Name,count(\*) AS no\_of\_loans,Conforming\_Limit\_000 FROM clean\_data WHERE State LIKE 'VA' GROUP BY County\_Name"

Virginia\_data <- sqldf(Virginia\_dataSQL)

Virginia\_data <- Virginia\_data[order(-Virginia\_data$no\_of\_loans),]

##################################################################################################

DC\_dataSQL <- "SELECT County\_Name,count(\*) AS no\_of\_loans,Conforming\_Limit\_000 FROM clean\_data WHERE State LIKE 'DC' GROUP BY County\_Name"

DC\_data <- sqldf(DC\_dataSQL)

DC\_data <- DC\_data[order(-DC\_data$no\_of\_loans),]

##################################################################################################

# Average loan amount grouped by state

mean\_loanAmt\_state <- aggregate(clean\_data$Loan\_Amount\_000,by=list(State=clean\_data$State),mean,na.rm=TRUE)

mean\_loanAmt\_state <- mean\_loanAmt\_state[order(-mean\_loanAmt\_state$x),]

ggplot(mean\_loanAmt\_state, aes(x=reorder(State,-x),y=x)) +

geom\_bar(stat='identity', aes(fill=x)) +

labs(x="State", y="Avg. loan amount (000's)")