#server.R Code

#  
# This is the server logic of a Shiny web application. You can run the  
# application by clicking 'Run App' above.  
#  
# Find out more about building applications with Shiny here:  
#  
# http://shiny.rstudio.com/  
#  
  
library(shiny)  
library(tidyverse)  
library(maps)  
library(scales)  
library(sf)  
library(shinycssloaders)  
lendingClubLoanData <- read.csv("data/lending\_club\_loan\_data\_final.csv")  
fullStateNames <- read.csv("data/states.csv")  
states <- st\_as\_sf(map("state", plot = FALSE, fill = TRUE))  
incomeLabels <- c('0-20K','20-40K','40-60K','60-80K','80-100K','100-120K','120-140K','140-160K','160-180K','180-200K', '200-220K', '220-240K', '240-260K', '260-280K', '280-300K')  
  
loan\_statuses <- c("Current",  
 "Fully Paid",  
 "Late (31-120 days)",  
 "In Grace Period",  
 "Charged Off",  
 "Late (16-30 days)")  
  
filteredLendingClubData <- lendingClubLoanData %>%  
 drop\_na(annual\_inc)  
  
filtereDdtiData <- lendingClubLoanData %>%  
 drop\_na(dti)%>%  
 drop\_na(dti) %>%  
 filter(dti < 100)  
  
filteredAnnualIncomeData <- lendingClubLoanData %>%  
 drop\_na(annual\_inc)%>%  
 filter(annual\_inc < 300000)  
  
  
# Define server logic required to draw a histogram  
shinyServer(function(input, output) {  
   
 loanStatusFilter <- function(loanStatusValue) {  
 toReturn <- c("Current",  
 "Fully Paid",  
 "Late (31-120 days)",  
 "In Grace Period",  
 "Charged Off",  
 "Late (16-30 days)",  
 "Default",  
 "Does not meet the credit policy. Status:Fully Paid",  
 "Does not meet the credit policy. Status:Charged Off")  
 if (loanStatusValue != "Any"){  
 toReturn <- c(loanStatusValue)  
 }  
 return(toReturn)  
 }  
   
 number\_of\_loans\_each\_year <- reactive({  
 numberOfLoansEachYear <- lendingClubLoanData %>%  
 filter (loan\_amnt >= input$loanAmountRange[1] & loan\_amnt <= input$loanAmountRange[2]) %>%  
 filter(grade %in% input$grades) %>%  
 filter(home\_ownership %in% input$homeOwnerships) %>%  
 filter(loan\_status %in% loanStatusFilter(loanStatusValue = input$loanStatus)) %>%  
 filter (dti >= input$dti[1] & dti <= input$dti[2]) %>%  
 group\_by(orig\_year) %>%  
 summarise(loanCountByYear=n())  
 })  
   
 total\_amount\_funded\_each\_year <- reactive({  
 totalFundedAMountPerYear <- lendingClubLoanData %>%   
 filter (loan\_amnt >= input$loanAmountRange[1] & loan\_amnt <= input$loanAmountRange[2]) %>%  
 filter(grade %in% input$grades) %>%  
 filter(home\_ownership %in% input$homeOwnerships) %>%  
 filter(loan\_status %in% loanStatusFilter(loanStatusValue = input$loanStatus)) %>%  
 filter (dti >= input$dti[1] & dti <= input$dti[2]) %>%  
 group\_by(orig\_year)%>%  
 summarise(totalFundedAmount= sum(as.numeric(funded\_amnt)))  
 })  
   
 loan\_amt\_term\_relation <- reactive({  
 lendingClubLoanData %>%  
 filter (loan\_amnt >= input$loanAmountRange[1] & loan\_amnt <= input$loanAmountRange[2]) %>%  
 filter(grade %in% input$grades) %>%  
 filter(home\_ownership %in% input$homeOwnerships) %>%  
 filter(loan\_status %in% loanStatusFilter(loanStatusValue = input$loanStatus)) %>%  
 filter (dti >= input$dti[1] & dti <= input$dti[2])  
 })  
   
 dti\_trend <- reactive({  
 filtereDdtiData %>%  
 filter(loan\_amnt >= input$loanAmountRange[1] & loan\_amnt <= input$loanAmountRange[2]) %>%  
 filter(grade %in% input$grades) %>%  
 filter(home\_ownership %in% input$homeOwnerships) %>%  
 filter(loan\_status %in% loanStatusFilter(loanStatusValue = input$loanStatus)) %>%  
 filter (dti >= input$dti[1] & dti <= input$dti[2])  
 })  
   
 funded\_amt\_term\_interest\_relation <- reactive({  
 filteredLendingClubData <- filteredLendingClubData %>%  
 filter(annual\_inc <= 300000) %>%  
 filter(loan\_amnt >= input$loanAmountRange[1] & loan\_amnt <= input$loanAmountRange[2]) %>%  
 filter(grade %in% input$grades) %>%  
 filter(home\_ownership %in% input$homeOwnerships) %>%  
 filter(loan\_status %in% loanStatusFilter(loanStatusValue = input$loanStatus)) %>%  
 filter (dti >= input$dti[1] & dti <= input$dti[2])  
   
 groupedData <- filteredLendingClubData %>%   
 group\_by(incomeGroup = cut(annual\_inc, breaks= seq(0, 300000, by = 20000), right = TRUE, include.lowest = TRUE, labels = incomeLabels) ) %>%   
 summarise(averageInterest= mean(int\_rate), averageLoanLoanFundedAmount = mean(funded\_amnt))   
   
 })  
   
 income\_trend <- reactive({  
 filteredAnnualIncomeData %>%  
 filter (loan\_amnt >= input$loanAmountRange[1] & loan\_amnt <= input$loanAmountRange[2]) %>%  
 filter(grade %in% input$grades) %>%  
 filter(home\_ownership %in% input$homeOwnerships) %>%  
 filter(loan\_status %in% loanStatusFilter(loanStatusValue = input$loanStatus)) %>%  
 filter (dti >= input$dti[1] & dti <= input$dti[2])  
 })  
   
 loan\_funded\_amt\_by\_state <- reactive({  
 fundedAmountByState <- lendingClubLoanData %>%   
 filter (loan\_amnt >= input$loanAmountRange[1] & loan\_amnt <= input$loanAmountRange[2]) %>%  
 filter(grade %in% input$grades) %>%  
 filter(home\_ownership %in% input$homeOwnerships) %>%  
 filter(loan\_status %in% loanStatusFilter(loanStatusValue = input$loanStatus)) %>%  
 filter (dti >= input$dti[1] & dti <= input$dti[2]) %>%  
 group\_by(addr\_state)%>%  
 summarise(totalFundedAmount= sum(as.numeric(funded\_amnt)))  
   
 fundedAmountByState <- fundedAmountByState %>%  
 inner\_join(fullStateNames, by = c("addr\_state" = "abbreviation"))  
   
 states2 <- states %>% left\_join(fundedAmountByState, by = c("ID" = "state" ))  
 })  
   
 loans\_by\_status <- reactive({  
 numberOfLoansByLoanStatus <- lendingClubLoanData %>%   
 filter(loan\_status %in% loan\_statuses) %>%  
 filter(loan\_amnt >= input$loanAmountRange[1] & loan\_amnt <= input$loanAmountRange[2]) %>%  
 filter(grade %in% input$grades) %>%  
 filter(home\_ownership %in% input$homeOwnerships) %>%  
 filter(loan\_status %in% loanStatusFilter(loanStatusValue = input$loanStatus)) %>%  
 filter (dti >= input$dti[1] & dti <= input$dti[2]) %>%  
 group\_by(loan\_status)%>%  
 summarise(numberOfLoans = n())  
 })  
   
 loans\_by\_purpose <- reactive({  
 numberOfLoansByPurpose <- lendingClubLoanData %>%   
 filter(loan\_amnt >= input$loanAmountRange[1] & loan\_amnt <= input$loanAmountRange[2]) %>%  
 filter(grade %in% input$grades) %>%  
 filter(home\_ownership %in% input$homeOwnerships) %>%  
 filter(loan\_status %in% loanStatusFilter(loanStatusValue = input$loanStatus)) %>%  
 filter (dti >= input$dti[1] & dti <= input$dti[2]) %>%  
 group\_by(purpose)%>%  
 summarise(numberOfLoans = n())  
 })  
   
   
   
 output$loanProcessesedEachYear <- renderPlot({  
 ggplot(data = number\_of\_loans\_each\_year())+  
 geom\_line(color="steelblue", size=1.2, aes(x=orig\_year,y=loanCountByYear))+  
 geom\_point(color="steelblue", size=2.5, aes(x=orig\_year,y=loanCountByYear))+  
 scale\_x\_continuous(breaks = number\_of\_loans\_each\_year()$orig\_year) +  
 scale\_y\_continuous(labels = scales::comma\_format()) +  
 labs(x="Year",y="# Number of loans",title="Loans processed year on year (2007-2018)")+  
 theme\_minimal()   
 })  
   
 output$totalFundedLoanAmountEachYear <- renderPlot({  
 ggplot(data = total\_amount\_funded\_each\_year(), aes(x=orig\_year, y=totalFundedAmount)) +  
 geom\_bar(stat="identity", width=0.5, fill = "steelblue") +  
 scale\_x\_continuous(breaks = total\_amount\_funded\_each\_year()$orig\_year) +  
 scale\_y\_continuous(labels = scales::dollar) +  
 labs(x = "Year", y = "Total funded loan amount",title="Total loan funded amount year on year (2007-2018)") +  
 theme\_minimal()   
 })  
   
 output$loanAmtTermRelation <- renderPlot({  
 ggplot(data = loan\_amt\_term\_relation()) +  
 geom\_boxplot(aes(x=term, y=funded\_amnt, color=term)) +   
 scale\_y\_continuous(labels = scales::dollar) +  
 labs(x = "Term", y = "Loan funded amount", title="Loan funded amount and term relation") +  
 theme\_minimal()   
 })  
   
 output$fundedAmtIncomeAndInterestRelation <- renderPlot({  
 ggplot(data =funded\_amt\_term\_interest\_relation(), aes(x=incomeGroup, y=averageLoanLoanFundedAmount)) +  
 geom\_point(colour="steelblue", shape=16, aes(size=averageInterest)) +  
 geom\_smooth(aes(incomeGroup, averageLoanLoanFundedAmount, group = 1), method = "lm") +  
 scale\_y\_continuous(labels = scales::dollar) +  
 labs(x="Annual Income ($)",y="Average loan funded amount",title="Relation between funded loan Amount, income and interest rate")+  
 guides(size=guide\_legend("Average \ninterest rate (%)")) +  
 theme\_minimal() +  
 theme(axis.text.x = element\_text(angle =50, hjust=0.75))+  
 theme(legend.background = element\_rect())  
 })  
  
 output$incomeTrend <- renderPlot({  
 ggplot(data = income\_trend(), aes(x = annual\_inc)) +  
 geom\_density(fill="steelblue", color="steelblue", alpha=0.8) +  
 geom\_vline(aes(xintercept=median(annual\_inc)),color="green", linetype="dashed", size=1) +  
 scale\_x\_continuous(labels = scales::dollar) +  
 scale\_y\_continuous(labels = function(x) format(x, scientific = FALSE)) +  
 labs(x="Annual income",y="Density of loans",title="Annual income distribution") +  
 theme\_minimal()  
 })  
   
 output$loanFundedAmtByState <- renderPlot({  
 ggplot(data = loan\_funded\_amt\_by\_state()) +   
 geom\_sf(aes(fill = totalFundedAmount)) +  
 scale\_fill\_viridis\_c("Loan funded amount", labels = scales::dollar) +  
 labs(title = "Total loan funded amount by state") +  
 theme\_minimal()  
 })  
   
 output$dtiTrend <- renderPlot({  
 ggplot(data = dti\_trend(), aes(x = dti)) +  
 geom\_density(fill="steelblue", color="steelblue", alpha=0.8) +  
 geom\_vline(aes(xintercept=median(dti)),color="green", linetype="dashed", size=1) +  
 labs(x="Debt to Income Ratio (DTI) %",y="Density of loans",title="Loan distribution across DTI (Excluded > 100)") +  
 theme\_minimal()  
 })  
   
 output$loansByStatus <- renderPlot({  
 ggplot(loans\_by\_status(), aes(x=loan\_status, y=numberOfLoans)) +  
 geom\_bar(stat="identity", width=0.5, fill = "steelblue") +  
 geom\_text(aes(label=numberOfLoans), vjust=-0.3, size=3.5) +  
 scale\_y\_continuous(labels = scales::comma\_format()) +  
 labs(x = "Loan Status", y = "Number of Loans (#)",title="# Loans by status") +  
 theme\_minimal() +  
 theme(axis.text.x = element\_text(angle =50, hjust=0.75))  
 })  
   
 output$loansByPurpose <- renderPlot({  
 ggplot(loans\_by\_purpose(), aes(x=purpose, y=numberOfLoans)) +  
 geom\_bar(stat="identity", width=0.5, fill = "steelblue") +  
 scale\_y\_continuous(labels = scales::comma\_format()) +  
 labs(x = "Loan Purpose", y = "Number of Loans (#)",title="# Loans by purpose") +  
 theme\_minimal() +  
 theme(axis.text.x = element\_text(angle =50, hjust=0.75))  
 })  
   
})

#ui.R code

#  
# This is the user-interface definition of a Shiny web application. You can  
# run the application by clicking 'Run App' above.  
#  
# Find out more about building applications with Shiny here:  
#  
# http://shiny.rstudio.com/  
#  
  
library(shiny)  
library(shinythemes)  
library(shinycssloaders)  
# Define UI for application that draws a histogram  
shinyUI(fluidPage(  
 theme = shinytheme("slate"),  
 # Application title  
 titlePanel(""),  
 titlePanel(title=div(style="display:inline-block;width:100%;",  
 img(src="homelogo.png", style="height:100px;"), "Data analysis (2007-2018)"),   
 windowTitle = "Lending club data analysis"),  
   
 # Sidebar with a slider input for number of bins  
 sidebarLayout(  
 sidebarPanel(  
 # Price Range filter  
 sliderInput("loanAmountRange",  
 "Loan Amount:",  
 pre = "$",  
 min = 500,  
 max = 40000,  
 value = c(500, 40000)),  
 # Horizontal line ----  
 tags$hr(),  
   
 # House condition filter  
 checkboxGroupInput(inputId = "grades",  
 label ="Grade:",  
 choiceNames = c("A", "B", "C", "D", "E", "F", "G"),   
 choiceValues = c("A", "B", "C", "D", "E", "F", "G"),  
 selected = c("A", "B", "C", "D", "E", "F", "G"),  
 inline = TRUE),  
 # Horizontal line ----  
 tags$hr(),  
   
 # House condition filter  
 checkboxGroupInput(inputId = "homeOwnerships",  
 label ="Home Ownership:",  
 choiceNames = c("ANY", "RENT", "MORTGAGE", "OWN"),   
 choiceValues = c("ANY", "RENT", "MORTGAGE", "OWN"),  
 selected = c("ANY", "RENT", "MORTGAGE", "OWN"),  
 inline = TRUE),  
 # Horizontal line ----  
 tags$hr(),  
   
   
 # Number of bedrooms filter  
 selectInput("loanStatus", "Loan Status:",  
 c(  
 "Any" = "Any",  
 "Current" = "Current",  
 "Fully Paid" = "Fully Paid",  
 "Late (31-120 days)" = "Late (31-120 days)",  
 "In Grace Period" = "In Grace Period",  
 "Charged Off" = "Charged Off",  
 "Late (16-30 days)" = "Late (16-30 days)",  
 "Default" = "Default",  
 "Does not meet the credit policy. Status:Fully Paid" = "Does not meet the credit policy. Status:Fully Paid",  
 "Does not meet the credit policy. Status:Charged Off" = "Does not meet the credit policy. Status:Charged Off"  
 )),  
 # Horizontal line ----  
 tags$hr(),  
   
 # Price Range filter  
 sliderInput("dti",  
 "Debt to Income Ratio:",  
 pre = "%",  
 min = -1,  
 max = 999,  
 value = c(0, 100)),  
 # Horizontal line ----  
 tags$hr(),  
 tags$div(style="display:inline-block;width:100%;color:#e48806",   
 "\*\* Due to large dataset size, visualizations and filters may run slow \*\*"),  
 ),  
   
 # Show a plot of the generated distribution  
 mainPanel(  
 tabsetPanel(  
 tabPanel("Year wise loans trend", shinycssloaders::withSpinner(plotOutput("loanProcessesedEachYear")),   
 shinycssloaders::withSpinner(plotOutput("totalFundedLoanAmountEachYear"))),  
 tabPanel("Loan Amount, Term Relation",   
 shinycssloaders::withSpinner(plotOutput("loanAmtTermRelation", height="700px"))),  
 tabPanel("DTI Trend",   
 shinycssloaders::withSpinner(plotOutput("dtiTrend", height="700px"))),  
 tabPanel("Loan Amount, Income, Interest Relation",   
 shinycssloaders::withSpinner(plotOutput("fundedAmtIncomeAndInterestRelation")),  
 shinycssloaders::withSpinner(plotOutput("incomeTrend"))),  
 tabPanel("Loan Amount by state",   
 shinycssloaders::withSpinner(plotOutput("loanFundedAmtByState", height="700px"))),  
 tabPanel("Loans status and purpose",   
 shinycssloaders::withSpinner(plotOutput("loansByStatus")),  
 shinycssloaders::withSpinner(plotOutput("loansByPurpose")))  
 )  
 ),  
   
 )  
))