#Zadania:

#1.Utworz funkcje: rankAccount <- function(dataFrame,colName,groupName,valueSort,num)

#ktora bedzie zwracala dla danej tabeli(dataFrame) n wierszy posiadajace najwieksze

#wartosci(sortowanie po kolumnie valueSort)

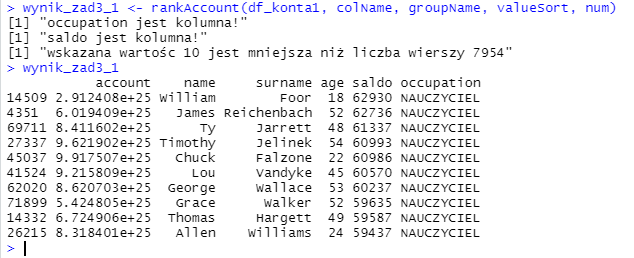
#dla wybranej grupy(konkretna wartosc komorki , np. "NAUCZYCIEL) z kolumny(colName)

#np. occupation-zawod.

# num to ile ma zwrócić jak top, limit

## Zad.1

Wyniki działania



kod

|  |
| --- |
| df\_konta1 <- read.csv(file = 'konta.csv')  #View(konta1)  class(df\_konta1)  object.size(df\_konta1)  rankAccount <- function(dataFrame,colName,groupName, valueSort,num) {  df <- dataFrame  #sprawdzanie kolumn  if (colName %in% colnames(df)) { print(paste0(colName, " jest kolumna!"))  } else { print(paste0(colName, " nie ma takiej kolumny!"))  stop("BŁĄD - nie ma takiej kolumny") }  if(valueSort %in% colnames(df)) { print(paste0(valueSort, " jest kolumna!"))  } else { print(paste0(valueSort, " nie ma takiej kolumny!"))  stop("BŁĄD - nie ma takiej kolumny") }    wynik\_filtorwania <- df[df[colName] == groupName,]  wynik\_sortowania <- wynik\_filtorwania[order(-wynik\_filtorwania[valueSort]),]    #sprawdzanie limitu, czy nie przekracza  wynik\_limitu <- NA  liczba\_wierszy <- nrow(wynik\_sortowania)  if(num < liczba\_wierszy) { wynik\_limitu <- head(wynik\_sortowania, num)  print(paste0("wskazana wartośc ", num, " jest mniejsza niż liczba wierszy ", liczba\_wierszy))  } else {  wynik\_limitu <- wynik\_sortowania  print(paste0("wskazana wartośc ", num, " przekracza liczbe wierszy ", liczba\_wierszy, "dlatego max ograniczono"))  }  wynik\_limitu  }  colName <- "occupation1"  groupName <- "NAUCZYCIEL"  valueSort = "saldo"  num<- 10  wynik\_zad3\_1 <- rankAccount(df\_konta1, colName, groupName, valueSort, num)  wynik\_zad3\_1 |

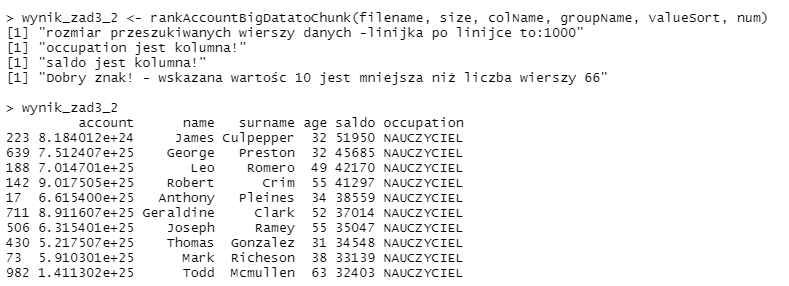
## Zad.2.

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

#2.Tak jak w 1 tylko z uzyciem datachunku.

#przyklad naglowka:

Wynik działania



Kod

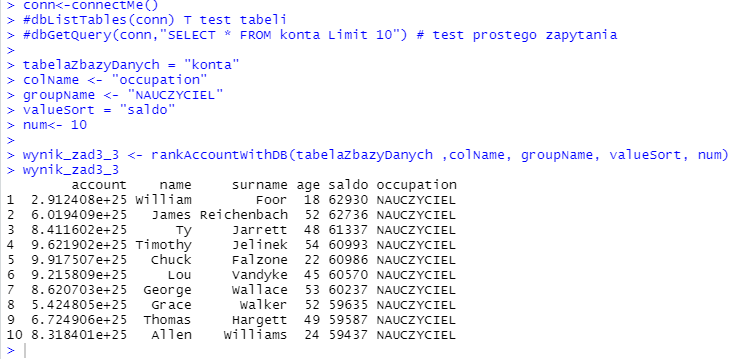
|  |
| --- |
| ##########################################  #2.Tak jak w 1 tylko z uzyciem datachunku.  #przyklad naglowka:  rankAccountBigDatatoChunk <- function(filename , size, colName, groupName, valueSort, num){  header<- TRUE  sep=","  counter<-0  nrows\_size = 1  print(paste0("rozmiar przeszukiwanych wierszy danych -linijka po linijce to:", size ))  fileConnection<- file(description = filename, open = "r")  data<-read.table(fileConnection,nrows=nrows\_size,header=header,fill=TRUE,sep=sep)  columnNames<-names(data)  repeat{  if(nrow(data)==0 | counter >= size){  close(fileConnection)  break  }  counter<-counter + nrows\_size  #print(paste0("nrow(data): ", nrow(data)))  #print(paste0("counter: ", counter))  data1<-read.table(fileConnection,nrows=nrows\_size,col.names = columnNames,fill=TRUE,sep=sep)  data <- rbind(data, data1)  }  #data  wynik <- rankAccount(data, colName, groupName, valueSort, num)  wynik  }  filename = "konta.csv"  size = 1000  colName <- "occupation"  groupName <- "NAUCZYCIEL"  valueSort = "saldo"  num<- 10  wynik\_zad3\_2 <- rankAccountBigDatatoChunk(filename, size, colName, groupName, valueSort, num)  wynik\_zad3\_2 |

## Zad.3.

#3.SPRAWIDZIC CZY DA SIE ZROBIC TO SAMO W zapytaniu SQL dla takich wartosci jak: tabelaZbazyDanych,occupation, nauczyciel, saldo

**TAK, udało się!**

Wynik działania



Kod, najpierw przygotowanie potem cześć właściwa z funkcją rankAccountWithDB

|  |
| --- |
| #3.SPRAWIDZIC CZY DA SIE ZROBIC TO SAMO W zapytaniu SQL dla takich wartosci jak: tabelaZbazyDanych,occupation, nauczyciel, saldo  # przygotowanie  # połączenie i zapis do bazy danych tabeli z csv  connectMe<-function(typ=Postgres(),dbname="ulvimbvm",host="tai.db.elephantsql.com",user="ulvimbvm", pass="NBN9Vp6JoWZMfYvu04hc36k7LFm-TeoN"){  con<-dbConnect(typ,  dbname=dbname,  host=host,  user=user,  password=pass) #askForPassword("database pass"))  }  conn<-connectMe()  dbGetInfo(conn)  #dbDisconnect(conn)  readToBaseWithDbConn<-function(filepath,dbConn,tablename,size,sep=",",header=TRUE,delete=TRUE){  ap<-!delete  ov<-delete  fileConnection<- file(description = filepath, open = "r")  data<-read.table(fileConnection,nrows=size,header=header,fill=TRUE,sep=sep)  columnNames<-names(data)  dbWriteTable(conn = dbConn,name=tablename,data,append=ap,overwrite=ov)  repeat{    if(nrow(data)==0){  close(fileConnection)  dbDisconnect(dbConn)  break  }  data<-read.table(fileConnection,nrows=size,col.names = columnNames,fill=TRUE,sep=sep)  dbWriteTable(conn = dbConn,name=tablename,data,append=TRUE,overwrite=FALSE)  }  }  conn<-connectMe()  readToBaseWithDbConn("konta.csv", conn, "konta", size = lengthOfFile("konta.csv") )  ###############  #część własciwa  #odczyt z bazy danych  rankAccountWithDB <- function(tabelaZbazyDanych ,colName, groupName, valueSort, num){  query <- paste0("SELECT \* FROM ", tabelaZbazyDanych, " where ", colName, " = '", groupName, "'")  query\_order <- paste(query, " order by ", valueSort, " desc")  query\_limit <- paste(query\_order, " limit ", num)  wynik <- dbGetQuery(conn, query\_limit)  wynik  }  conn<-connectMe()  #dbListTables(conn) T test tabeli  #dbGetQuery(conn,"SELECT \* FROM konta Limit 10") # test prostego zapytania  tabelaZbazyDanych = "konta"  colName <- "occupation"  groupName <- "NAUCZYCIEL"  valueSort = "saldo"  num<- 10  wynik\_zad3\_3 <- rankAccountWithDB(tabelaZbazyDanych ,colName, groupName, valueSort, num)  wynik\_zad3\_3 |