DEPARTMENT OF MANAGEMENT SCIENCE & TECHNOLOGY

Business Analytics

Master of Science

Course:
Big Data Systems and Architectures

REDIS & MongoDB Assignment

Konstantopoulou Marianna | p2822122
Zafeiropoulou Maria | p2822113

Athens

22 February 2022

TASK 1 | Redis

First, we create a connection to the local instance of Redis.

```
setwd("~/Desktop/MSc Business Analytics/Big Data Systems and Architectures/Redis/RECORDED_ACTIONS")
# Load the library
library("redux")
library("dplyr")

# Create a connection to the local instance of REDIS
r <- redux::hiredis(
  redux::redis_config(
    host = "127.0.0.1",
    port = "6379"))
```

1.1 How many users modified their listing on January?

ANSWER:

9.969 users modified their listing on January

```
##1##
#Read the csv file
modified_listings <- read.csv("modified_listings.csv")
View(modified_listings)
#Create a BITMAP called “ModificationsJanuary” using “SETBIT -> 1” for each user that modified their listing
for (i in 1:length(modified_listings$UserID)){
  if (modified_listings$ModifiedListing[i] ==1 & modified_listings$MonthID[i] ==1){
  r$SETBIT("ModificationsJanuary", modified_listings$UserID[i], "1")
  }
}
#Calculate the answer
r$BITCOUNT("ModificationsJanuary")
```

1.2 How many users did NOT modify their listing on January?

ANSWER:

10.031 users did not modify their listing on January. The numbers of this and the previous answer does not add up to the total of the users because for bit operations. BITOP happens at byte-level increments, the results of the BITOP are always an integer number multiple of 8 (as each byte has 8 bits).

```
##2##
#Total users
length(unique(modified_listings$UserID))
#Performing inversion on the "ModificationsJanuary" BITMAP
r$BITOP("NOT","results","ModificationsJanuary")
#Using BITCOUNT to calculate the result
r$BITCOUNT("results")
```

1.3 How many users received at least one e-mail per month (at least one e-mail in January and at least one e-mail in February and at least one e-mail in March)?

ANSWER:

2.668 users received at least one e-mail per month.

```
##3##
#Read the csv file
emails_sent <- read.csv("emails_sent.csv")
View(emails_sent)
length(unique(emails_sent$UserID))
#JANUARY
for (i in 1:length(emails_sent$UserID)){
  if (emails_sent$MonthID[i] ==1){
    r$SETBIT("EmailsJanuary", as.character(emails_sent$UserID[i]), "1")
  }
}
#Calculate the answer
r$BITCOUNT("EmailsJanuary")

#FEBRUARY
for (i in 1:length(emails_sent$UserID)){
  if (emails_sent$MonthID[i] ==2){
    r$SETBIT("EmailsFebruary", as.character(emails_sent$UserID[i]), "1")
  }
}
#Calculate the answer
r$BITCOUNT("EmailsFebruary")

#MARCH
for (i in 1:length(emails_sent$UserID)){
  if (emails_sent$MonthID[i] ==3){
    r$SETBIT("EmailsMarch", as.character(emails_sent$UserID[i]), "1")
  }
}
#Calculate the answer
r$BITCOUNT("EmailsMarch")

r$BITOP("AND","results3",c("EmailsJanuary", "EmailsFebruary", "EmailsMarch"))
#Using BITCOUNT to calculate the result
r$BITCOUNT("results3")
```

1.4 How many users received an e-mail on January and March but NOT on February?

ANSWER:

2.417 users received an e-mail on January and March but not on February.

```
##4##
#Creating a BITOP wth the users that received an email on January and March
r$BITOP("AND","results4",c("EmailsJanuary", "EmailsMarch"))
#Creating a BITOP with the users that didn't receive an email on February
r$BITOP("NOT","results5","EmailsFebruary")
#Combing the results
r$BITOP("AND", "results6", c("results5","results4"))
#Counting the results
r$BITCOUNT("results6")
```

1.5 How many users received an e-mail on January that they did not open but they updated their listing anyway?

ANSWER:

1.961 users received an e-mail on January that they did not open but they updated their listing anyway.

```
##5##
emails_opened_jan <- subset(emails_sent, emails_sent$MonthID==1 & emails_sent$EmailOpened==1)

for (i in unique(emails_opened_jan$UserID)){
    r$SETBIT("EmailsOpenedJanuary", i, "1")
}
r$BITCOUNT("EmailsOpenedJanuary")

r$BITOP("NOT","EmailsNotOpened","EmailsOpenedJanuary")
r$BITCOUNT("EmailsNotOpened")

r$BITOP("AND", "EmailsNotOpenedJanuary", c("EmailsNotOpened","EmailsJanuary"))
r$BITCOUNT("EmailsNotOpenedJanuary")

r$BITOP("AND", "finalresult", c("EmailsNotOpenedJanuary","ModificationsJanuary"))
r$BITCOUNT("finalresult")
```

1.6 How many users received an e-mail on January that they did not open but they updated their listing anyway on January OR they received an e-mail on February that they did not open but they updated their listing anyway on February OR they received an e-mail on March that they did not open but they updated their listing anyway on March?

ANSWER:

5.249 users received an e-mail on January that they did not open but they updated their listing anyway on January or they received an e-mail on February that they did not open but they updated their listing anyway on February or they received an e-mail on March that they did not open but they updated their listing anyway on March.

```
##6##

#FEBRUARY#
for (i in 1:length(modified_listings$UserID)){
  if (modified_listings$ModifiedListing[i] ==1 & modified_listings$MonthID[i] ==2){
    r$SETBIT("ModificationsFebruary", modified_listings$UserID[i], "1")
  }
}
#Calculate the answer
r$BITCOUNT("ModificationsFebruary")

emails_opened_feb <- subset(emails_sent, emails_sent$MonthID==2 & emails_sent$EmailOpened==1)

for (i in unique(emails_opened_feb$UserID)){
  r$SETBIT("EmailsOpenedFebruary", i, "1")
}
r$BITCOUNT("EmailsOpenedFebruary")

r$BITOP("NOT","EmailsNotOpened2","EmailsOpenedFebruary")
r$BITCOUNT("EmailsNotOpened2")

r$BITOP("AND", "EmailsNotOpenedFebruary", c("EmailsNotOpened2","EmailsFebruary"))
r$BITCOUNT("EmailsNotOpenedFebruary")

r$BITOP("AND", "finalresult2", c("EmailsNotOpenedFebruary","ModificationsFebruary"))
r$BITCOUNT("finalresult2")

#MARCH#
for (i in 1:length(modified_listings$UserID)){
  if (modified_listings$ModifiedListing[i] ==1 & modified_listings$MonthID[i] ==3){
    r$SETBIT("ModificationsMarch", modified_listings$UserID[i], "1")
  }
}
#Calculate the answer
r$BITCOUNT("ModificationsMarch")

emails_opened_mar <- subset(emails_sent, emails_sent$MonthID==3 & emails_sent$EmailOpened==1)

for (i in unique(emails_opened_mar$UserID)){
  r$SETBIT("EmailsOpenedMarch", i, "1")
}
r$BITCOUNT("EmailsOpenedMarch")

r$BITOP("NOT","EmailsNotOpened3","EmailsOpenedMarch")
r$BITCOUNT("EmailsNotOpened3")

r$BITOP("AND", "EmailsNotOpenedMarch", c("EmailsNotOpened3","EmailsMarch"))
r$BITCOUNT("EmailsNotOpenedMarch")

r$BITOP("AND", "finalresult3", c("EmailsNotOpenedMarch","ModificationsMarch"))
r$BITCOUNT("finalresult3")

r$BITOP("OR", "finalresult4", c("finalresult", "finalresult2", "finalresult3"))
r$BITCOUNT("finalresult4")
```

1.7 Does it make any sense to keep sending e-mails with recommendations to sellers? Does this strategy really work? How would you describe this in terms a business person would understand?

ANSWER:

The strategy of sending emails with recommendations to sellers does not seem to work well, since, on average only 28.13% of the Modified Listings came from people that opened these emails.

```
##7##

r$BITOP("AND", "Jan_opened_modified", c("EmailsOpenedJanuary","ModificationsJanuary"))
r$BITCOUNT("Jan_opened_modified")
r$BITCOUNT("ModificationsJanuary")
#=2797/9969= 28% of the modifications came from opened emails in January

r$BITOP("AND", "Feb_opened_modified", c("EmailsOpenedFebruary","ModificationsFebruary"))
r$BITCOUNT("Feb_opened_modified")
r$BITCOUNT("ModificationsFebruary")
#=2874/10007= 28.7%  of the modifications came from opened emails in February

r$BITOP("AND", "Mar_opened_modified", c("EmailsOpenedMarch","ModificationsMarch"))
r$BITCOUNT("Mar_opened_modified")
r$BITCOUNT("ModificationsMarch")
#=2783/9991= 27.8%  of the modifications came from opened emails in March
```

TASK 2 | Analytics with MongoDB

2.1 Add your data to MongoDB.

In the following seven points, we present the steps we followed to create, clean and insert the dataset in MongoDB,

1. Create a list with the paths of the files.
2. Open a Collection (named “Initial”) in MongoDB.
3. Insert the Json files in the “Initial” Collection.
4. Create a Data Frame of the files to perform the cleaning.
5. Create a function to clean the data.
   The cleaning actions are the following:
   - From the values of the variable “**Price**” we remove the € symbol, we set the “Askforprice” values and ones less than 100 € as NULL. Finally, we set the “Price” variable as numerical. It is important to notice that, we have decided to, essentially, exclude from our analysis the listings with price less than 100 €, even if it is arbitrary, because these listings include crashed or very old bikes.
   - From the variable “**Mileage**” we remove the “km” symbol and we set the variable type as numerical.
   - We set the “**Registration**” values that are less than 1910 as NULL and we add to the dataset a new variable “Age”, which indicates the registration age of the listing.
   - Finally, we add a variable named “Negotiable” in the dataset, which takes the price TRUE if there is the word negotiable in the ad.
6. Insert the cleaned data to a MongoDB collection (named “Cleaned”).
7. Finally, we are ready to perform queries.

In the next page, we can see the code of the procedure mentioned.

```r
# LIBRARIES #

library(jsonlite)
library(mongolite)
library(dplyr)
library(stringr)

######################## Read & Insert Data to MongoDB ########################


# STEP 1:
# Import a vector containing all the paths
vec_ = dir(path = "C:\\            \\BIKES_DATASET\\BIKES", pattern = "\\.json$",
           full.names = TRUE, recursive = TRUE)
# Vector as a list
vec_1 = as.list(vec_)

# STEP 2:
# Open the Mongo "Initial" Collection
m <- mongo(collection = "Initial",  db = "ERGASIAMONGO", url = "mongodb://localhost")

# STEP 3:
# Insert the Json files in the "Initial" Collection
for(i in  1:length(vec_1)){
  m$insert(fromJSON(readLines(vec_1[[i]], warn = F)))
}

# STEP 4:
# Create DataFrame
initial_df = m$find("{}")
View(initial_df)

df <- initial_df

l <- length(df$query[,1])

############################## CLEANING FUNCTION ################################

clean <- function(df1){

  for (i in 1:l){
    if(df1$ad_data$Price[i] == 'Askforprice') {
      df1$ad_data$Price[i] <- NA
    }
    else {
      df1$ad_data$Price[i] <- (gsub('\\D+','', df1$ad_data$Price[i])) #as.numeric
    }
    if (!is.na(df1$ad_data$Price[i])==TRUE){
      if (df1$ad_data$Price[i] < 100){
        df1$ad_data$Price[i] <- NA
      }
    }}
  df1$ad_data$Price <- as.numeric(df1$ad_data$Price)
  df1$ad_data$Mileage<- as.numeric(gsub("[,km]", "", df1$ad_data$Mileage))

  RegistrationYear <- as.numeric(str_sub(df1$ad_data$Registration,-4))

  for (i in 1:l){
    if (RegistrationYear[i] < 1910){
      df1$ad_data$Registration[i] <- NA
      df1$Age[i] <- NA
    }
    else{
      df1$Age[i] <- 2022-RegistrationYear[i]
    }
  }

  for (i in 1:l){
    if(grepl("Negotiable",df1$metadata$model[i]) == TRUE){
      df1$Negotiable[i] <- TRUE
    }
    else {
      df1$Negotiable[i] <- FALSE
    }
  }

  return(df1)
}


df <- clean(df)


# INSERT CLEAN DATA TO MONGODB

tnew <- mongo(collection = "Cleaned",  db = "ERGASIAMONGO", url = "mongodb://localhost")
tnew$insert(df)
```

2.2 How many bikes are there for sale?

ANSWER:

There are 29.701 bikes for sale.

```
db.getCollection('Cleaned').find({})


#Question 2.2
db.getCollection('Cleaned').count({})
```

2.3 What is the average price of a motorcycle (give a number)? What is the number of listings that were used in order to calculate this average (give a number as well)? Is the number of listings used the same as the answer in 2.2? Why?

ANSWER:

The average price of a motorcycle is 3.030,62 €. The number of listings used to calculate the average price is 28.490, i.e., 1.211 less listings than the total number of motorcycles for sale. This is because 1.211 listings have stored the price as "Askforprice" or have price less than 100 €.

```
#Question 2.3
db.Cleaned.aggregate(
     [
        {
            "$match": {
            "ad_data.Price" : {
                "$exists": true
                    }
            }
        },
        {
            "$group" : {
            "_id" : null,
            "avg_price":{
                "$avg": "$ad_data.Price"
                },
              "count":{
                  "$sum":1
                  }
            }
        }
     ]
)
```

2.4 What is the maximum and minimum price of a motorcycle currently available in the market?

ANSWER:

The maximum price is 89.000 € and the minimum price is 100 €.

```
#Question 2.4

db.Cleaned.aggregate(
[ {
            "$match": {
            "ad_data.Price" : {
                "$exists": true
                    }
            }
        },
  {
    "$group" : {
      "_id" : null,
      "max_price": {
        "$max" : "$ad_data.Price"
      }
    }
  }
 ]
)

db.Cleaned.aggregate(
[ {
            "$match": {
            "ad_data.Price" : {
                "$exists": true
                    }
            }
        },
  {
    "$group" : {
      "_id" : null,
      "min_price": {
        "$min" : "$ad_data.Price"
      }
    }
  }
 ]
)
```

2.5 How many listings have a price that is identified as negotiable?

ANSWER:

The number of listings having a price that is identified as Negotiable is 1.348.

```
#Question 2.5

db.Cleaned.aggregate(
    [
        { "$match" :
                {"metadata.model":
                        {"$regex" : "Negotiable",
                         "$options" : "i"
                        }
                }
            },
                {"$group":
                   {"_id": null,
                       "count": {
                           "$sum" : 1
                        }
                }
         }
    ]
)
```

Optional

2.7 What is the motorcycle brand with the highest average price?

The motorcycle brand with the highest average price is Semog.

```
#Question 2.7 (optional)


db.Cleaned.aggregate(
     [
        {
          "$group":
              {"_id": "$metadata.brand",
              "Average_Price": {"$avg":"$ad_data.Price"},
              "count": {"$sum": 1}}},
        {
          "$sort":
              {"Average_Price": -1}
      },
            {
             "$limit": 1
        }
])
```

2.8 What are the TOP 10 models with the highest average age?

The TOP 10 are the following

Brand	Avg Age	No of models
Bsa	73,8	15
Norton	70,8	5
Horex	69,6	6
Victoria	69	1
Nsu	66,7	9
Adler	66	1
Heinkle	62,8	5
Kuberg	62	1
Dkw	61,4	16
Maico	58,5	4

```
#Question 2.8 (optional)

db.Cleaned.aggregate(
        [{
            "$group" :
             {"_id" : "$metadata.brand",
              "avg_age" : {"$avg" : {"$avg":"$Age"}},
              "count" : {"$sum": 1}}},
          {"$sort"  :
                      {"avg_age" : -1}}, {"$limit" : 10}])
```

2.9 How many bikes have “ABS” as an extra?

4.025 Bikes have “ABS” as an extra.

```
#Question 2.9 (optional)


db.Cleaned.find({'extras': 'ABS'}).count()
```

2.10 What is the average Mileage of bikes that have “ABS” AND “Led lights” as an extra?

The average Mileage is 34.392,5

```
#Question 2.10 (optional)


db.Cleaned.aggregate(
    [
        {
            "$match" : {
                "extras" : "ABS",
                "extras" : "Led lights",
                "ad_data.Mileage" : {
                    "$exists" : true
                }
            }
        },
        {

          "$group":
              {"_id": null,
              "Average_Mileage": {"$avg":"$ad_data.Mileage"},
              }
}])
```