REDIS & MongoDB Assignment 2023

Panagiotis G. Vaidomarkakis (p2822203)

Course: Big Data Systems (Part Time 2022-2024)

Tutor: Spyros Safras

27/03/2023

Table of Contents

1 Redis 2

1.1 How many users modified their listing on January? 2

1.2 How many users did NOT modified their listing on January? 3

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)? 4

1.4 How many users received an e-mail on January and March but NOT on February? 4

1.5 How many users received an e-mail on January that they did not open but they updated their listing anyway? 5

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? 5

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? 6

2 MongoDB 9

2.1 Add your data to MongoDB. 9

2.2 How many bikes are there for sale? 12

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? 12

2.4 What is the maximum and minimum price of a motorcycle currently available in the market? 13

2.5 How many listings have a price that is identified as negotiable? 13

2.6 For each Brand, what percentage of its listings is listed as negotiable? 13

2.7 What is the motorcycle brand with the highest average price? 15

2.8 What are the TOP 10 models with the highest average age? (Round age by one decimal number) 15

2.9 How many bikes have "ABS" as an extra? 16

2.10 What is the average Mileage of bikes that have "ABS" AND "Led lights" as an extra? 16

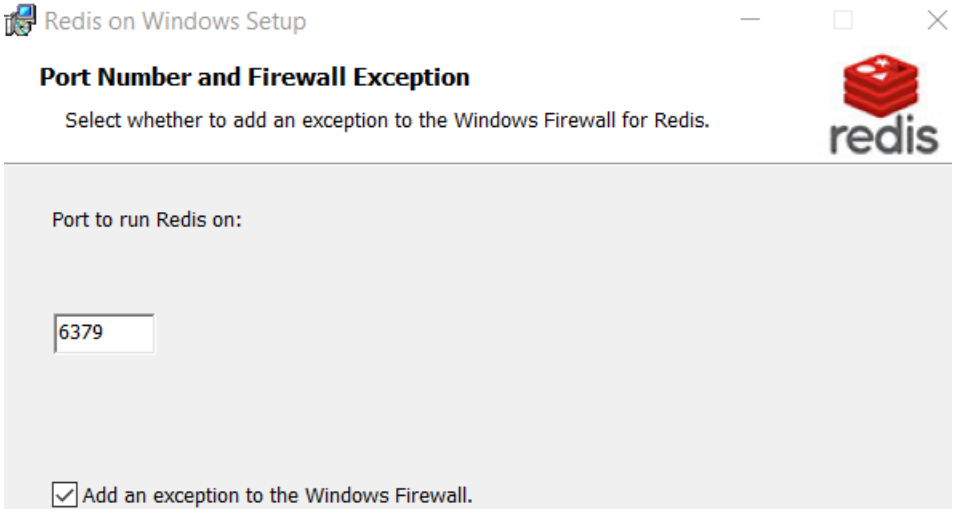
2.11 What are the TOP 3 colors per bike category? 16

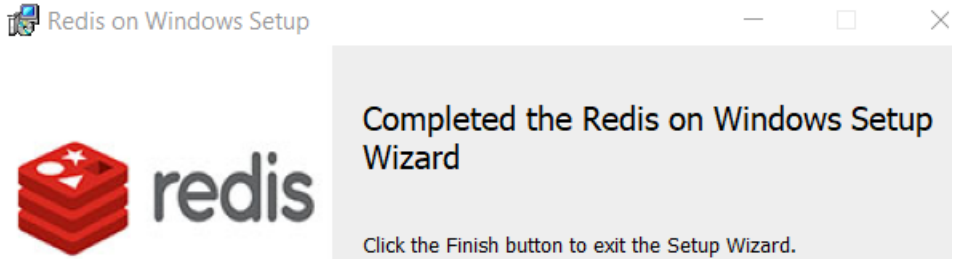
2.12 Identify a set of ads that you consider "Best Deals". 17

1 Redis

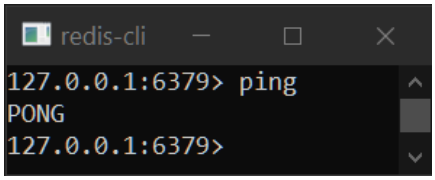
For the First Part of the assignment, we will use Redis Database. Before we proceed with any question from the assignment, we installed Redis for Windows 5.0.14.1 located in this link.

Below, you will see some pictures of the procedure which was pretty straightforward.





Now, a last check need to be done in order to proceed to R-Studio for loading the data and answering the questions.



1.1 How many users modified their listing on January?

After loading the libraries and the datasets, we use *SETBIT* and after that *BITCOUNT* to take all the Listing Modification in January.

```
# Load the library
if (!requireNamespace("redux", quietly = TRUE)) {
  install.packages("redux")}
library("redux")
```

```
# Local Connection
redis_con <- redux::hiredis(
  redux::redis_config(
    host = "127.0.0.1",
    port = "6379"))

# Load data sets
setwd(use your path for your working space here)
listings <- read.csv("modified_listings.csv", header = TRUE, sep=",", stringsAsFactors = FALSE)
emails <- read.csv("emails_sent.csv", header = TRUE, sep=",", stringsAsFactors = FALSE)

for(i in 1:nrow(listings))
{
  if ((listings$ModifiedListing[i] ==1) & (listings$MonthID[i]==1))
  {
    redis_con$SETBIT("ModificationsJanuary",listings$UserID[i],"1")
  }
}

q1_modified_jan_count <- redis_con$BITCOUNT("ModificationsJanuary")
```

The result of the code was 9969 users modified their listing in January.

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

We used BITOP NOT in order to get the result.

```
invisible(redis_con$BITOP("NOT","NoModificationsJanuary","ModificationsJanuary"))
q2_not_modified_count <- redis_con$BITCOUNT("NoModificationsJanuary")
```

The result of the code was 10031 users didn’t modified their listing on January.

Our initial users were 19999, but through Redis, our users are 20000. Redis stores binary strings using whole bytes, so even if only a few bits are needed to store a value, Redis will use a whole byte. With 19999 users, Redis would need 19999 bits, which is equivalent to 2499,875 bytes. Since Redis uses whole bytes, it would round up to 2500 bytes to store the data, so that is why we have 20000 users(2500*8).

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)?

We create 3 BITMAPS for emails in January, February and March. Then we fill them using SETBIT and after that, we make a BITOP AND following BITCOUNT in order to retrieve them.

```
for(i in 1:nrow(emails))
{
 if (emails$MonthID[i]==1)
 {
  redis_con$SETBIT("EmailsJanuary",emails$UserID[i],"1")
 }
 else if(emails$MonthID[i]==2)
 {
  redis_con$SETBIT("EmailsFebruary",emails$UserID[i],"1")
 }
 else if (emails$MonthID[i]==3)
 {
  redis_con$SETBIT("EmailsMarch",emails$UserID[i],"1")
 }
}

invisible(redis_con$BITCOUNT("EmailsJanuary"))
invisible(redis_con$BITCOUNT("EmailsFebruary"))
invisible(redis_con$BITCOUNT("EmailsMarch"))
invisible(redis_con$BITOP("AND","ReceivedAtLeast1EmailPerMonth",c("EmailsJanuary","EmailsFebruary" , "EmailsMarch")))

q3_users_with_at_least_one_email <- redis_con$BITCOUNT("ReceivedAtLeast1EmailPerMonth")
```

The result of the code was 2668 users received at least one email in January, at least one email in February and at least one email in March.

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

We BITOP AND EmailsJanuary and EmailsMarch, we invert EmailsFebruary and the we BITOP AND both the previous one and BITCOUNT.

```
invisible(redis_con$BITOP("AND","ReceivedAtLeast1EmailJanMar",c("EmailsJanuary","EmailsMarch")))
invisible(redis_con$BITCOUNT("ReceivedAtLeast1EmailJanMar"))
```

```
invisible(redis_con$BITOP("NOT", "NoEmailsFebruary", "EmailsFebruary"))

invisible(redis_con$BITCOUNT("NoEmailsFebruary"))


invisible(redis_con$BITOP("AND","ReceivedJanMarButNoFeb",c("ReceivedAtLeast1EmailJanMar","NoEmailsFebruary")))


q4_users_with_at_received_email_JanMar_but_not_Feb <- redis_con$BITCOUNT("ReceivedJanMarButNoFeb")
```

The result of the code was 2417 received at least one email in January and at least one email in March but no email in February.

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

In order to answer that, we created a new BITMAP NotOpenJanuary and then we BITOP AND the result with the result from the 1.1(ModificationsJanuary).

```
for(i in 1:nrow(emails))
{
  if ((emails$MonthID[i]==1) & (emails$EmailOpened[i]==0))
  {
    redis_con$SETBIT("NotOpenJanuary",emails$UserID[i],"1")
  }
}


invisible(redis_con$BITOP("AND","ReceivedNoOpenButModifiedJanuary",c("ModificationsJanuary","NotOpenJanuary")))


q5_users_who_received_didnt_open_and_modified_jan <- redis_con$BITCOUNT("ReceivedNoOpenButModifiedJanuary")
```

The result of the code was 2807 users received at least one email in January, didn’t open it and they updated their listing.

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?

We created four other BITMAPs ModificationsFebruary, ModificationsMarch, NotOpenFebruary and NotOpenMarch. We then BITOP AND the BITMAPs with the same month and in the end, we BITOP OR the ReceivedNoOpenButModified of each month and BITCOUNT that.

```
for(i in 1:nrow(listings))
{
 if ((listings$ModifiedListing[i] ==1) & (listings$MonthID[i]==2))
 {
  redis_con$SETBIT("ModificationsFebruary",listings$UserID[i],"1")

 }
 else if ((listings$ModifiedListing[i] ==1) & (listings$MonthID[i]==3)) {
  redis_con$SETBIT("ModificationsMarch",listings$UserID[i],"1")
 }
}

for(i in 1:nrow(emails))
{
 if ((emails$MonthID[i]==2) & (emails$EmailOpened[i]==0))
 {
  redis_con$SETBIT("NotOpenFebruary",emails$UserID[i],"1")
 }
 else if ((emails$MonthID[i]==3) & (emails$EmailOpened[i]==0))
 {
  redis_con$SETBIT("NotOpenMarch",emails$UserID[i],"1")
 }
}

invisible(redis_con$BITOP("AND","ReceivedNoOpenButModifiedFebruary",c("ModificationsFebruary","NotOpenFebruary")))
invisible(redis_con$BITOP("AND","ReceivedNoOpenButModifiedMarch",c("ModificationsMarch","NotOpenMarch")))

invisible(redis_con$BITOP("OR","ReceivedNoOpenButModified",c("ReceivedNoOpenButModifiedJanuary","ReceivedNoOpenButModified
February","ReceivedNoOpenButModifiedMarch")))
q6_users_who_received_didnt_open_and_modified <- redis_con$BITCOUNT("ReceivedNoOpenButModified")
```

The result of this code was 7221 users received an email, didn't open it but they modified their listing.

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?

In order to answer that, we need to see the percentage of users who open the email and modified their listing in comparison to the users who open the email and didn't modified. We will make six BITMAPs with EmailsOpenedModifiedJanuary, EmailsOpenedModifiedFebruary, EmailsOpenedModifiedMarch, EmailsOpenedNotModifiedJanuary, EmailsOpenedNotModifiedFebruary and

EmailsOpenedNotModifiedMarch. We will then see if there is any improvement in the percentage of each month.

```
# Inner join of 2 csv files
merged<-merge(x=emails, y=listings, by=c("UserID","MonthID"))

# E-mails opened per month with the listing to be modified
for(i in 1:nrow(merged))
{
  if ((merged$EmailOpened[i]==1) & (merged$ModifiedListing[i] ==1)) {
    if (merged$MonthID[i]==1) {
      redis_con$SETBIT("EmailsOpenedModifiedJanuary",merged$UserID[i],"1")
    } else if (merged$MonthID[i]==2) {
      redis_con$SETBIT("EmailsOpenedModifiedFebruary",merged$UserID[i],"1")
    } else {
      redis_con$SETBIT("EmailsOpenedModifiedMarch",merged$UserID[i],"1")
    }
  }
}

# E-mails opened per month without the listing to be modified
for(i in 1:nrow(merged))
{
  if ((merged$EmailOpened[i]==1) & (merged$ModifiedListing[i] ==0)) {
    if (merged$MonthID[i]==1) {
      redis_con$SETBIT("EmailsOpenedNotModifiedJanuary",merged$UserID[i],"1")
    } else if (merged$MonthID[i]==2) {
      redis_con$SETBIT("EmailsOpenedNotModifiedFebruary",merged$UserID[i],"1")
    } else {
      redis_con$SETBIT("EmailsOpenedNotModifiedMarch",merged$UserID[i],"1")
    }
  }
}

invisible(redis_con$BITCOUNT("EmailsOpenedModifiedJanuary"))
invisible(redis_con$BITCOUNT("EmailsOpenedNotModifiedJanuary"))
# Percentage of modified listings for January)
JanuaryPerc <- round(redis_con$BITCOUNT("EmailsOpenedModifiedJanuary")
/(redis_con$BITCOUNT("EmailsOpenedModifiedJanuary")+redis_con$BITCOUNT("EmailsOpenedNotModifiedJanuary")) * 100,3)
```

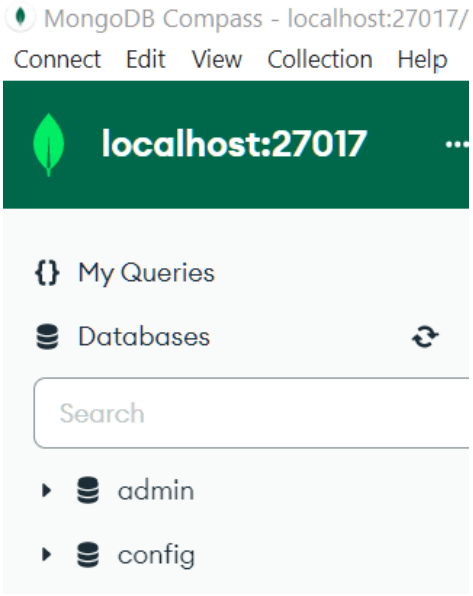
```
print(paste('Percentage of January users who who opened and modified divided by all users who opened in
January:',round(JanuaryPerc,2),'%'), quote = FALSE)


invisible(redis_con$BITCOUNT("EmailsOpenedModifiedFebruary"))

invisible(redis_con$BITCOUNT("EmailsOpenedNotModifiedFebruary"))

# Percentage of modified listings for February

FebruaryPerc <- round((redis_con$BITCOUNT("EmailsOpenedModifiedFebruary") /
(redis_con$BITCOUNT("EmailsOpenedModifiedFebruary") + redis_con$BITCOUNT("EmailsOpenedNotModifiedFebruary"))) * 100,3)

print(paste('Percentage of February users who who opened and modified divided by all users who opened in
February:',round(FebruaryPerc,2),'%'), quote = FALSE)


invisible(redis_con$BITCOUNT("EmailsOpenedModifiedMarch"))

invisible(redis_con$BITCOUNT("EmailsOpenedNotModifiedMarch"))

# Percentage of modified listings for March

MarchPerc <- round((redis_con$BITCOUNT("EmailsOpenedModifiedMarch") / (redis_con$BITCOUNT("EmailsOpenedModifiedMarch")
+ redis_con$BITCOUNT("EmailsOpenedNotModifiedMarch"))) * 100,3)

print(paste('Percentage of March users who who opened and modified divided by all users who opened in
March:',round(MarchPerc,2),'%'), quote = FALSE)
```

In January, we have 49,55% who modified their listing. In February, we have 50,24% who modified their listing. In March, we have 49,95% who modified their listing. As we can see, almost 50% modified their listings. In that sense, we need to keep sending these emails because 50% from those who open emails tend to modify after reading that. Moreover, 4768 people modify without opening their email and 6246 appear to modify their listings after the opening, so, unless it is very costly, we should continue sending emails.

2 MongoDB

For the Second Part of the assignment, we will use MongoDB, a non-relational database. Before we proceed with any question from the assignment, we installed MongoDB Community Server for Windows 6.0.5 located in this link.

Below, you will see a picture of MongoDB Compass, a GUI for the database. The procedure was pretty straightforward.



We will use R and RStudio in order to manipulate the data and import them in MongoDB. After the import, we will be able to see our database with our collections as a new database above.

2.1 Add your data to MongoDB.

Below is the code for data cleaning before we import it in MongoDB.

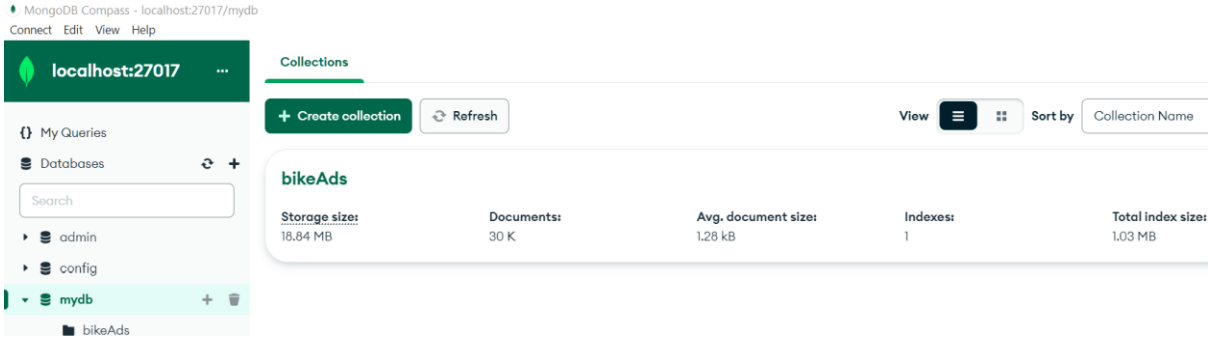
```
# List of libraries to check
libraries <- c("mongolite", "lubridate", "httpuv", "jsonlite", "rmutil", "stringr", "dplyr", "rmarkdown")

# Check if each library is installed and install it if necessary
for (lib in libraries) {
  if (!requireNamespace(lib, quietly = TRUE)) {
    install.packages(lib)}
  library(lib, character.only = TRUE)
}

setwd(use your path for your working space here)
```

```
# Read file with paths of JSON files to a list
my_data <- readLines("BIKES/files_list.txt", skipNul = TRUE)
my_data <- my_data[-1][nzchar(my_data[-1])]

# Create a mongo connection object and create an empty collection
m <- mongo(collection = "bikeAds",  db = "mydb", url = "mongodb://localhost")

ageScore <- 0
mileageScore <- 0
priceScore <- 0
cc_bhpScore <- 0

lower1age <- 0
lower2age <- 6
lower3age <- 11
upper1age <- 5
upper2age <- 10

lower1mlg <- 0
lower2mlg <- 20001
lower3mlg <- 50001
upper1mlg <- 20000
upper2mlg <- 50000

lower1price <- 0
lower2price <- 2001
lower3price <- 5001
upper1price <- 2000
upper2price <- 5000

group1Factor <- 50
group2Factor <- 30
group3Factor <- 20

lower1cc_bhp <- 0
lower2cc_bhp <- 0.051
lower3cc_bhp <- 0.11
upper1cc_bhp <- 0.05
upper2cc_bhp <- 0.1
```

```
for (row in 1:length(my_data)) {

  data <- fromJSON(readLines(paste('BIKES/',my_data[row],sep =''), encoding = 'UTF-8'))

  data$ad_data$Negotiable <- any(grepl("Negotiable", unlist(data), ignore.case = TRUE),grepl("συζητήσιμη", unlist(data), ignore.case = TRUE),grepl("συζητησιμη", unlist(data), ignore.case = TRUE))

  data$ad_data$Mileage <- as.numeric(gsub("[^0-9.]|\\.", "", data$ad_data$Mileage))

  data$ad_data$Price <- as.numeric(gsub("[^0-9.]|\\.", "", data$ad_data$Price))

  data$metadata$model <- gsub(" -.*", "", data$metadata$model)

  data$ad_data$`Cubic capacity` <- as.numeric(gsub("[^0-9.]|\\.", "", data$ad_data$`Cubic capacity`))

  data$ad_data$Power <- as.numeric(gsub("[^0-9.]|\\.", "", data$ad_data$Power))

  data$ad_data$Price[which(data$ad_data$Price == 'Askforprice')] <- 0

  data$ad_data$Registration <- as.Date(paste0("01/",gsub(" ", "", data$ad_data$Registration)), format = '%d/%m/%Y')

  data$ad_data$Age <- round(as.numeric(difftime(as.Date(paste0("01/", month(Sys.Date()), "/", year(Sys.Date())), format = "%d/%m/%Y"), as.Date(data$ad_data$Registration), units = "days")) / 365.25,0)


  ageScore <- ifelse(data$ad_data$Age >= lower1age & data$ad_data$Age <= upper1age, data$ad_data$Age * group1Factor,

                ifelse(data$ad_data$Age >= lower2age & data$ad_data$Age <= upper2age, data$ad_data$Age * group2Factor,

                    ifelse(data$ad_data$Age >= lower3age, data$ad_data$Age * group3Factor, 0)))


  mileageScore <- ifelse(data$ad_data$Mileage >= lower1mlg & data$ad_data$Mileage <= upper1mlg, data$ad_data$Mileage * group1Factor,

                    ifelse(data$ad_data$Mileage >= lower2mlg & data$ad_data$Mileage <= upper2mlg, data$ad_data$Mileage * group2Factor,

                        ifelse(data$ad_data$Mileage >= lower3mlg, data$ad_data$Mileage * group3Factor, 0)))


  priceScore <- ifelse(data$ad_data$Price >= lower1price & data$ad_data$Price <= upper1price, data$ad_data$Price * group1Factor,

                ifelse(data$ad_data$Price >= lower2price & data$ad_data$Price <= upper2price, data$ad_data$Price * group2Factor,

                    ifelse(data$ad_data$Price >= lower3price, data$ad_data$Price * group3Factor, 0)))


  cc_bhpScore <- if (length(data$ad_data$`Cubic capacity`) > 0) {

              ifelse((data$ad_data$Power / data$ad_data$`Cubic capacity`) >= lower1cc_bhp & (data$ad_data$Power / data$ad_data$`Cubic capacity`) <= upper1cc_bhp, (data$ad_data$Power / data$ad_data$`Cubic capacity`) * group3Factor,

                  ifelse((data$ad_data$Power / data$ad_data$`Cubic capacity`) >= lower2cc_bhp & (data$ad_data$Power / data$ad_data$`Cubic capacity`) <= upper2cc_bhp, (data$ad_data$Power / data$ad_data$`Cubic capacity`) * group2Factor,

                      ifelse((data$ad_data$Power / data$ad_data$`Cubic capacity`) >= lower3cc_bhp, (data$ad_data$Power / data$ad_data$`Cubic capacity`) * group1Factor, 0))))}

          else {0}


  data$ad_data$Score <-ageScore + mileageScore + priceScore + cc_bhpScore


  data <- toJSON(data, auto_unbox = TRUE)

  m$insert(data)
```

}

In a few words, in the beginning, I import all the necessary libraries, then I read my_data which contains bikes with various information. Then, I initialize some parameters for the last question in order to give a (weight) score based on age of the bike, the mileage, the price and the diviation of power and cubic capacity. Then, I have group1Factor = 50, group2Factor = 30, group3Factor = 20 in order to have weight on our score. Then, in the for loop, I read each document and I clean every data that I will need. I find «Negotiable», «συζητήσιμη» and «συζητησιμη» in each word of the document. Then, I remove any symbol from the mileage, price, cubic capacity and power. Then, we remove any special character from model of the bike, the Askforprice turns into 0 and last but not least, we find the age of the bike. Then, we calculate each of the four scores and in the end, we import the collections in our database. In the end, we can see our database of collections here:



2.2 How many bikes are there for sale?

bikesForSale <- m$count('{}')

print(bikesForSale)

The output of our code is 29701 bikes.

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?

```
bikesAvgPrice <- m$aggregate(
  '[
  {"$match": {"ad_data.Price": { "$gte": 100 }}},
  {"$group":{"_id": null, "average":{"$avg":"$ad_data.Price"}}}
  ]'
)


bikesUsedForAverage <-  nrow(m$aggregate('[
  {"$match": {"ad_data.Price": { "$gte": 100 }}}]'))


print(bikesAvgPrice$average)
print(bikesUsedForAverage)
```

The output of the previous queries is 3030,62 average price with 28490 bikes for the calculation. Obviously, we cannot use all the bikes for this calculation because some are sold for 1€ and 10€ which doesn't make sense. We included only bikes starting from 100€ in order to have a better feeling about the average price.

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

```
maxPrice <- m$aggregate(
  '[
  {"$match": {"ad_data.Price": { "$gt": 0 }}},
  {"$group":{"_id": null, "max":{"$max":"$ad_data.Price"}}}
  ]'
)
print(maxPrice$max)
minPrice <- m$aggregate(
  '[
  {"$match": {"ad_data.Price": { "$gt": 100 }}},
  {"$group":{"_id": null, "min":{"$min":"$ad_data.Price"}}}
  ]'
)
print(minPrice$min)
```

The minimum price is 101€ because that 100€ was our bottom limit and the maximum price is 89000€.

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

```
negotiableBikes <- m$aggregate(
  '[
  {"$match": {"ad_data.Negotiable": true}},
  {"$group": { "_id": null, "negCount": { "$sum": 1 }}}
              ]'
)
print(negotiableBikes$negCount)
```

The output of the previous code was 1815 bikes.

2.6 For each Brand, what percentage of its listings is listed as negotiable?

```
negPercentage <- m$aggregate(
  '[
```

```
 { "$group": {
   "_id": "$metadata.brand",
   "totalCount": { "$sum": 1 },
   "negotiableCount": {
    "$sum": {
     "$cond": {
      "if":{ "$eq": [ "$ad_data.Negotiable", true ] },
      "then": 1,
      "else": 0
     }
    }
   }
 }},
 { "$addFields": {
   "negotiablePercentage": {
    "$cond": {
     "if": { "$ne": [ "$negotiableCount", 0 ] },
     "then": {
      "$multiply": [
       { "$divide": [ "$negotiableCount", "$totalCount" ] },
       100
       ]
     },
     "else": 0
    }
   }
 }},
 { "$sort": { "negotiablePercentage": -1 } }
 ]'
)
print(negPercentage)
```

The output of this code is here for the first 20 of the 198:

```
> print(negPercentage)
                _id totalCount negotiableCount negotiablePercentage
1        Boom-Trikes          1               1           100.000000
2            Victory          1               1           100.000000
3       Regal-Raptor          2               2           100.000000
4           Vee Road          1               1           100.000000
5                Mtg          4               4           100.000000
6           Apokotos          2               2           100.000000
7       Buggy Motors          3               3           100.000000
8       Nitro Motors          1               1           100.000000
9         Bombardier          1               1           100.000000
10               Niu          1               1           100.000000
11             Odess          2               2           100.000000
12               Wsk          1               1           100.000000
13          Amstrong          3               3           100.000000
14            Kuberg          1               1           100.000000
15            Motobi          1               1           100.000000
16            Qingqi          2               2           100.000000
17           HighPer          2               2           100.000000
18            Jinlun          1               1           100.000000
19           ZhongYu          1               1           100.000000
20             E-ATV          2               2           100.000000
```

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

```
bikesAvgHighestPrice <- m$aggregate(
 '[
 {"$match": {"ad_data.Price": { "$gte": 100 }}},
 {"$group":{"_id": "$metadata.brand", "average":{"$avg":"$ad_data.Price"}}},
 { "$sort": { "average": -1}},
 {"$limit": 1}
 ]'
)
print(bikesAvgHighestPrice)
```

The output of the code is Semog with average price of 15600.

2.8 What are the TOP 10 models with the highest average age? **(Round age by one decimal number)**

```
top10highest <- m$aggregate(
 '[
        {"$match": {"ad_data.Mileage": { "$gt": 0 }}},
                                {"$group":{"_id": "$metadata.model","avgMileage": {"$avg":"$ad_data.Mileage"},
"avgAGE":{"$avg":"$ad_data.Age"}}},
        {"$sort": {"avgAGE":-1, "avgMileage":1}},
        {"$limit": 10}
                ]'
)
top10highest <- select(top10highest, 1, 3)
print(top10highest)
```

You can see the output of this code below:

```
> print(top10highest)
                                  _id avgAGE
1  Αλλο  henderson indian replica '31     92
2                               R 12     89
3                          Norton   '35     88
4           Αλλο  MATCHLESS G3 350 '35     88
5                     Norton  H16 '36     87
6                   Αλλο  Matsoules '38     85
7             Αλλο  Matchless G3/L '39     84
8                 Αλλο  NEW HUDSON '39     84
9              Bsa  M20 ARMY MOTO! '39     84
10                            Bsa   '39     84
```

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

```
abs <- m$aggregate('[
 {"$match": {"extras": "ABS"}},
 {"$group": {"_id": null, "ABSCount": {"$sum": 1}}}
]')
print(abs$ABSCount)
```

4025 have ABS as an extra.

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

```
absLed <- m$aggregate(
 '[
 {"$match": { "$and": [ {"extras": "ABS"}, {"extras":"Led lights"}]}},
 {"$group":{"_id": null, "absLedAvg":{"$avg": "$ad_data.Mileage"}}}
            ]'
)
print(absLed$absLedAvg)
```

30125,7 mileage is the average for bikes that have ABS and Led lights.

2.11 What are the TOP 3 colors per bike category?

```
top3colors <- m$aggregate(
 '[
      {"$group":
        {"_id": {"category": "$ad_data.Category","color": "$ad_data.Color"},"count": {"$sum": 1}}},
      {"$sort": {"_id.category": -1,"count": -1}},
      {"$group":
        {"_id":"$_id.category","topColors":{"$push":"$_id.color"}}},
```

```
    {"$project": { "category": "$_id.category", "top3Colors": { "$slice": [ "$topColors", 3 ] } } }
        ]'
)

print(top3colors)
```

```
> print(top3colors)
                      _id                                top3Colors
1  Bike - UTV Side by Side                          Black, Blue, White
2     Bike - Sport Touring   Black (Metallic), Black, Silver (Metallic)
3    Bike - Four Wheel-ATV                          Red, Black, White
4             Bike - Moped                           Red, Black, Blue
5       Bike - Three Wheel                          Black, White, Red
6             Bike - Other                Black, Black (Metallic), Red
7      Bike - Super Motard              Black, Orange, Black (Metallic)
8           Bike - Chopper Black, Black (Metallic), Bordeaux (Metallic)
9       Bike - Super Sport                Black, Black (Metallic), Red
10           Bike - Enduro                          White, Orange, Red
11            Bike - Trial                  White, Red, Red (Metallic)
12            Bike - Naked    Black, Black (Metallic), White (Metallic)
13       Bike - Mini..Moto                          Red, Black, White
14        Bike - Underbone               Black, Blue, Black (Metallic)
15            Bike - Buggy                           Black, Red, Blue
16      Bike - Street Bike                Black, Black (Metallic), Red
17        Bike - Motocross                          Red, Orange, Green
18   Bike - Roller/Scooter              Black, Black (Metallic), White
19 Bike - Mobility scooter                Red, White, Black (Metallic)
20           Bike - Custom                Black, Black (Metallic), Red
21           Bike - On/Off              Black, Black (Metallic), White
22       Bike - Cafe Racer                Black, Black (Metallic), Red
```

2.12 Identify a set of ads that you consider “Best Deals”.

```
bestDeals <- m$aggregate(
  '[
  {"$match": { "$and": [ {"ad_data.Price":  { "$gt": 0 }}, {"ad_data.Price":  { "$lte": 10000 }}]}},
  {"$sort": {"ad_data.Score": -1}},
  {"$limit": 100}
    ]'
)
View(bestDeals)
```

From the score that we have measure, we select the top 100 in order to have the best deals. These 100 bikes have the lowest price, the lowest mileage, the lowest age and the best power divided by cubic capacity. They will win in at least one category and because we sum them, they have to be good in all categories in order to be in the best deals. Here are the first 15:

ad_data$Make/Model	ad_data$Classified number	ad_data$Price	ad_data$Category	ad_data$Registration	ad_data$Mileage
Honda GL 1500 Goldwing '91	19956636	5000	Bike - Sport Touring	1991-01-01	1400000
Kreidler Florett FLORETT 50 '72	19977956	1611	Bike - Moped	1972-01-01	1111111
Bsa '39	14791215	5500	Bike - Other	1939-01-01	1000000
Kawasaki Versys 650 '10	21118387	3200	Bike - On/Off	2010-01-01	1000000
Suzuki RM-Z 250 '04	20873064	1900	Bike - Motocross	2004-01-01	999999
Honda CB 400SF '98	21038101	1800	Bike - Custom	1998-01-01	1000000
Triumph 3 H '40	10977525	2500	Bike - On/Off	1940-01-01	1000000
Αλλο MOTOSTOP ΑΝΤΑΛΛΑΚΤΙΚΑ-ΑΓΟΡΕΣ-Π '01	11836319	800	Bike - Other	2001-01-01	999999
Honda Camino '76	10120235	300	Bike - Moped	1976-05-01	1000000
Haojin '19	21364743	150	Bike - Roller/Scooter	2019-03-01	1000000
Haojin '19	21364636	150	Bike - Roller/Scooter	2019-03-01	1000000
Haojin '19	21364859	150	Bike - Roller/Scooter	2019-03-01	1000000
Haojin '19	21364878	150	Bike - Roller/Scooter	2019-03-01	1000000
Lambretta LI 150 S2 '60	21391921	1	Bike - Roller/Scooter	1960-01-01	1000000
Honda CBF 250 2004 '04	18852616	1450	Bike - Street Bike	2004-01-01	860000