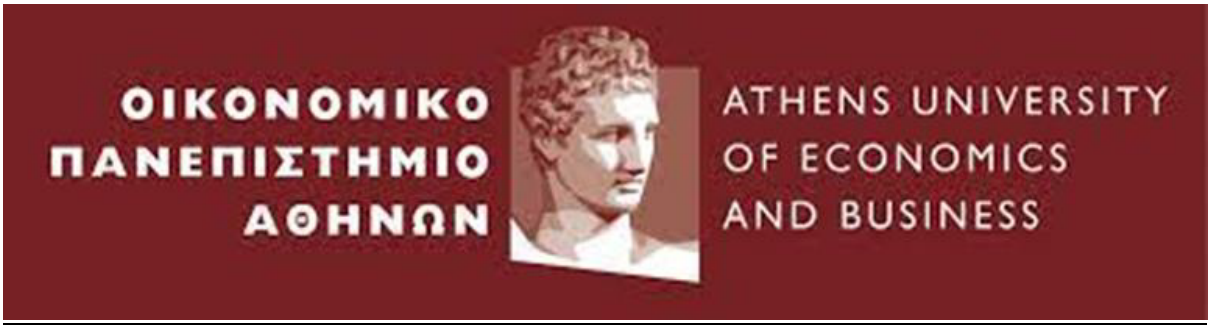



ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
DEPARTMENT OF MANAGEMENT, SCIENCE & TECHNOLOGY
MSc BUSINESS ANALYTICS

# "Redis & MongoDB Assignment 2023"

Course: Big Data Systems and Architectures - PT
Full Name: ATHANASIOS ALEXANDRIS
Register Number: p2822202

# Table of Contents

## Abstract

In this assignment we had to work as data analysts in the motorcycle market field, creating a sequence of queries, using REDIS and MongoDB, that lead us to strategic decisions. We had access to a large dataset of 30k classified ads from the used motorcycles market, and other data that contain some seller related actions that have been tracked for the period January to March.

## 1. Task 1 – REDIS

### 1.1 Introduction

In the first task of this assignment, we were given two datasets. The first one contains data related to emails that had been sent from classified providers to sellers with a number of suggestions on how they could improve their listings. More particularly from this dataset we had the following data: the email ID, the user ID (seller), month ID (January, February, March) and a column that indicates whether the email has been opened or not (1=yes, 0=no).
The second dataset contain contains data related to the changes done by all sellers that received an email. More particularly from this dataset we had the following data: the user ID (seller), month ID (January, February, March) and a column that indicates whether the seller has made a change or no(1=yes, 0=no).
Working on these datasets with REDIS we had to work on the questions of 1.2 section. At the beginning of our work, we used library "redux" in R and we created a connection with REDIS. Then we load the two datasets in R, and for both datasets we checked for missing values or nan values. Finally we cleaned our REDIS database ( FLUSHALL command).

## 1.2 Questions

### 1.2.1 How many users modified their listing on January?

The answer is 9969 users.

For this question we worked on the 'modified_listings' dataset. We created a bitmap through a for loop, that first checks whether the month is January (MonthID[i]=1) and the seller has made a change (ModifiedListing[i]=1), and then creates the bitmap (SETBIT) for the observations that apply to the above limitations. In the second argument of the SETBIT function we used UserID as we wanted our bitmap to contain the information of unique users that modified their listings on January.

```
> ##q1 An:9969
> for (i in 1:length(modified_listings$UserID)){
+   if ((modified_listings$MonthID[i]==1)&(modified_listings$ModifiedL
isting[i]==1)){
+     r$SETBIT("ModificationsJanuary",modified_listings$UserID[i],"1")
+   }
+ }
>
> r$BITCOUNT("ModificationsJanuary")
[1] 9969
```

### 1.2.2 How many users did NOT modify their listing on January?

In this question we used the NOT option of BITOP function, that modifies the bits of our bitmap, changing the 1 with 0 and the 0 with 1. The BITCOUNT function showed a number of 10031 users. So by adding this result to the previous we end up to 20000 users total. The execution of the length command for the unique UserID's showed that the actual number is 19999 users. So we have a difference, due to the usage of NOT option in BITOP function that lead us to a false result, since BITOP operations happen at byte-level increments. In our case the last userID is the 19997, which divided by 8 (bits) give us the number 2499.625, which is not an integer. So in order to BITOP to make the operation used zeros to complete the byte. So in order to answer right to the question we did exactly the same process with question 1 changing only that we want users not to have changed their listings, and the number of users that did not modify their listing on January is 10030.

```
> r$BITOP("NOT","NonModificationsJanuary","ModificationsJanuary")
[1] 2500
>
> r$BITCOUNT("NonModificationsJanuary")
[1] 10031
>
> 9969+10031
[1] 20000
> length(unique(modified_listings$UserID))
[1] 19999
>
> 19997/8
[1] 2499.625
> for (i in 1:length(modified_listings$UserID)){
+   if ((modified_listings$MonthID[i]==1)&(modified_listings$ModifiedL
isting[i]==0)){
+     r$SETBIT("NoModificationsJanuary",modified_listings$UserID
[i],"1")
+   }
+ }
> r$BITCOUNT("NoModificationsJanuary")
[1] 10030
```

1.2.3 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)?

In this question we created three bitmaps, one for each month. Each bitmap created contained 1 for the users that received at least one mail for the respective month. For the creation of each bitmap we used a for loop that checks the MonthID, and then creates through SETBIT function. As we have already mentioned in the second argument of the SETBIT we used the UserID, to avoid duplicates users. Then we used the AND option of BITOP, for the three bitmaps that we created, and the final result is 2668 users.

```
> ##Q3 An:2668
> for (i in 1:length(emails_sent$UserID)){
+   if (emails_sent$MonthID[i]==1){
+     r$SETBIT("EmailsJanuary",emails_sent$UserID[i],"1")
+   }
+ }
> for (i in 1:length(emails_sent$UserID)){
+   if (emails_sent$MonthID[i]==2){
+     r$SETBIT("EmailsFebruary",emails_sent$UserID[i],"1")
+   }
+ }
> for (i in 1:length(emails_sent$UserID)){
+   if (emails_sent$MonthID[i]==3){
+     r$SETBIT("EmailsMarch",emails_sent$UserID[i],"1")
+   }
+ }
> r$BITOP("AND","AtLeastOnePerMonth",c("EmailsJanuary","EmailsFebruar
y","EmailsMarch"))
[1] 2500
> r$BITCOUNT("AtLeastOnePerMonth")
[1] 2668
```

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

First we performed an inversion (“NoneEmailsFebruary”) of the bitmap “EmailsFebruary” that we created for the previous question, in order to have a bitmap for the users that did not receice an email on February. Next we performed a BITOP with the AND option for the three following bitmaps : “NoneEmailsFebruary”, “EmailsJanuary” and “EmailsMarch”. The result of the BITCOUNT was 2417 users.

```
> ##Q4 An:2417
>
> r$BITOP("NOT","NoneEmailsFebruary","EmailsFebruary")
[1] 2500
> r$BITOP("AND","AtLeastOneJanMArchNoFeb",c("EmailsJanuary","NoneEmail
sFebruary","EmailsMarch"))
[1] 2500
> r$BITCOUNT("AtLeastOneJanMArchNoFeb")
[1] 2417
```

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

First we created a bitmap named “EmailsReceivedNotOpenedJanuary” for the unique UserID’s that received an email on January (MonthID[i]=1) and they did not open it (EmailOpened[i]=0).

Then we performed a BITOP AND for this bitmap and the bitmap “ModificationsJanuary” that contained the UserID’s that made a change on January.

The result of BITCOUNT function of the new bitmap ModificationsAndNotReceivedJanuary, is 2807 users.

```
> ##Q5 An:2807
>
> for (i in 1:length(emails_sent$UserID)){
+   if ((emails_sent$MonthID[i]==1) & (emails_sent$EmailOpened[i]==0))
{
+     r$SETBIT("EmailsReceivedNotOpenedJanuary",emails_sent$UserID
[i],"1")
+   }
+ }
> r$BITOP("AND","ModificationsAndNotReceivedJanuary",c("ModificationsJ
anuary","EmailsReceivedNotOpenedJanuary"))
[1] 2500
> r$BITCOUNT("ModificationsAndNotReceivedJanuary")
[1] 2807
```

1.2.6 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?

First we created a bitmap for the users that received at least one email and they did not open it on February named “EmailsReceivedNotOpenedFebruary”.

Next we created another bitmap for the users that modified their listings on February named “ModificationsFebruary”.

We performed a BITOP AND for the two bitmaps mentioned above and created a bitmap named “ModificationsAndNotReceivedFebruary”.

Subsequently we performed exactly the same process for March, and we created another bitmap named “ModificationsAndNotReceivedMarch”.

Finally we performed a BITOP OR, for the bitmaps “ModificationsAndNotReceivedFebruary”, “ModificationsAndNotReceivedMarch”, “ModificationsAndNotReceivedJanuary” (Q5), and we created a new bitmap “ModificationsAndNotReceivedJanOrFebOrMarch”. The result of the Bitcount for this bitmap is 7221.

```
> for (i in 1:length(emails_sent$UserID)){
+   if ((emails_sent$MonthID[i]==2) & (emails_sent$EmailOpened[i]==0))
{
+     r$SETBIT("EmailsReceivedNotOpenedFebruary",emails_sent$UserID
[i],"1")
+   }
+ }
> for (i in 1:length(modified_listings$UserID)){
+   if ((modified_listings$MonthID[i]==2)&(modified_listings$ModifiedL
isting[i]==1)){
+     r$SETBIT("ModificationsFebruary",modified_listings$UserID
[i],"1")
+   }
+ }
> r$BITOP("AND","ModificationsAndNotReceivedFebruary",c("Modifications
February","EmailsReceivedNotOpenedFebruary"))
[1] 2500
```

```
> for (i in 1:length(emails_sent$UserID)){
+   if ((emails_sent$MonthID[i]==3) & (emails_sent$EmailOpened[i]==0))
{
+     r$SETBIT("EmailsReceivedNotOpenedMarch",emails_sent$UserID
[i],"1")
+   }
+ }
> for (i in 1:length(modified_listings$UserID)){
+   if ((modified_listings$MonthID[i]==3)&(modified_listings$ModifiedL
isting[i]==1)){
+     r$SETBIT("ModificationsMarch",modified_listings$UserID[i],"1")
+   }
+ }
> r$BITOP("AND","ModificationsAndNotReceivedMarch",c("ModificationsMar
ch","EmailsReceivedNotOpenedMarch"))
[1] 2500
> r$BITOP("OR","ModificationsAndNotReceivedJanOrFebOrMarch",c("Modific
ationsAndNotReceivedJanuary","ModificationsAndNotReceivedFebruary","Mo
dificationsAndNotReceivedMarch"))
[1] 2500
> r$BITCOUNT("ModificationsAndNotReceivedJanOrFebOrMarch")
[1] 7221
```

1.2.7 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?

First we calculated the amount of the unique users that they received an email. Then we created a bitmap for the users that they opened at least one email in the period of the three months. The two amounts are equal, which means that all users have opened at least one time an email from the classified providers.

```
> Amount_of_users_that_received_email
[1] 16006
> for (i in 1:length(emails_sent$UserID)){
+   if (emails_sent$EmailOpened[i]==1){
+     r$SETBIT("EmailsOpenned",emails_sent$UserID[i],"1")
+   }
+ }
>
> Emails_opened<-r$BITCOUNT("EmailsOpenned")
> ratio1<-Emails_opened/Amount_of_users_that_received_email
> ratio1
[1] 1
> Emails_opened
[1] 16006
```

Next we created a bitmap for the users that have opened an email at least one time on January named “EmailsReceivedOpenedJanuary”.

Then we performed a BITOP AND for the “EmailsReceivedOpenedJanuary” bitmap and the bitmap “ModificationsJanuary” which contains the users that modified their listing on January. So the bitmap created called “ModificationsAndOpenedJanuary” contains all the users that received an email on January which they opened it, and made a change in January.

Next we performed exactly the same process for months February and March, by creating the bitmaps “ModificationsAndOpenedFebruary” and “ModificationsAndOpenedMarch” respectively, that they contain the respective information with the bitmap “ModificationsAndOpenedJanuary”, but for months February and March.

Finally we performed a BITOP OR for the three bitmaps mentioned above “ModificationsAndOpenedJanuary”, “ModificationsAndOpenedFebruary”, “ModificationsAndOpenedMarch” and the created bitmap called “ModificationsAndOpenedJanOrFebOrMarch” indicates the UserID’s that they received at least one email, which lead them to make a change in the period January to March.

The outcome of the BITCOUNT for the “ModificationsAndOpenedJanOrFebOrMarch” bitmap is 7190 users.

So we calculated the ratio of the sellers that they received at least one email, which lead them to make a change in the period January to March, which equals to approximately 0.4989244.

```
> r$BITCOUNT("ModificationsAndOpenedJanOrFebOrMarch")
[1] 7190
```

```
> ratio_openedANDmodified
[1] 0.4989244
```

In conclusion we believe that it is important to mention that from the users that receive at least on email on the period January to March they all have at least one time opened the email. Additionally, the percentage of users that opened at least an email, and make changes in the same month, is 50%. Hence we believe that both ratios show that the strategy of the company affects users at an important level and it makes sense. However the company should work on it more, to make this strategy even more efficient.

# 2. Task 2 – MongoDB

## 2.1 Introduction

At this task we had access to 28k of bikes related listings from the biggest classifieds website for vehicles and parts in Greece. Each listing was in a json format in many separate files. The first thing we did was to create a vector named files in R that contained all the paths for each json file. To do this, we first changed our directory in our terminal to the bikes folder, and then used the command "find * | grep json > files_list.txt", so a txt file was created that contained all listing paths.

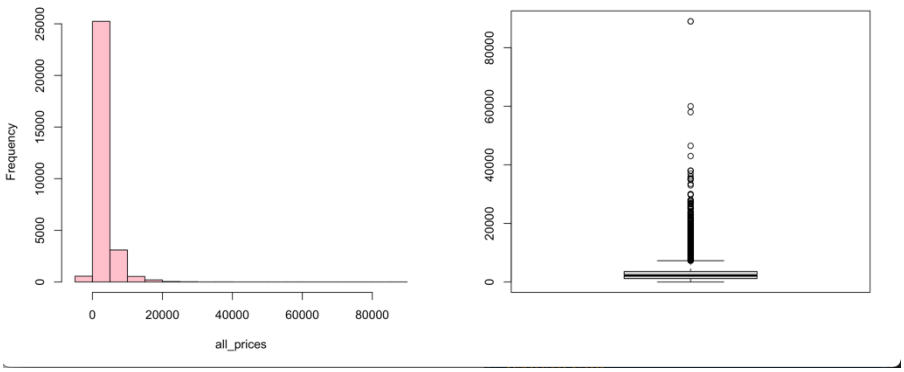
## 2.2 Cleaning

Then we had to clean our data based on the questions that we are called to answer subsequently, so we created a for loop, which first loads the data of the json, then creates separates lists for the fields that we were interested in (price, mileage, age of the vehicle, brand, model, and color).

By observing the unique observations of the listings prices, we noticed that the prices were in a format like this, € 1.000, so we had to remove the euro currency symbol and the ".", in order to transform them in numbers. We also noticed that there was a value "Askforprice", for the listings that either had to contact the seller to learn the price or for fake listings. We decided to set a value "-1" to the price of these listings. Then we noticed from the unique prices list, their summary, their histogram and their boxplot presented below , that were many listings having either very low or high prices, that usually means that the listing is fake, or the price were given only through contacting the seller. So, we decided to calculate the 5% (260) and the 95% (8000) of our prices percentiles, and use that range for our listing prices in the upcoming questions, keeping out the outliers that we discussed.

```
> all_prices_unique
  [1]   -1    1    2    3    4    5    7    8    9
 [10]   10   11   12   15   16   17   20   22   25
 [19]   30   35   40   50   51   60   70   80   99
 [28]  100  101  111  118  120  125  130  140  145
 [37]  149  150  170  175  180  190  199  200  210
 [46]  220  230  240  245  249  250  260  270  280
```

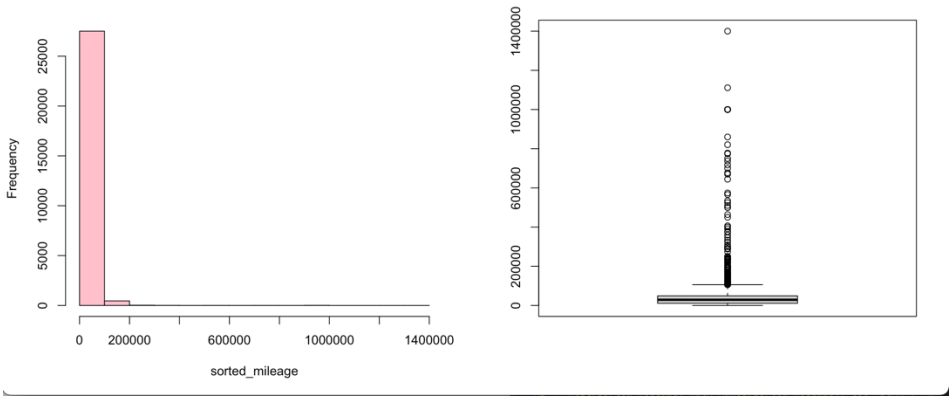
```
> summary(sorted_prices[sorted_prices>0])
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
      1    1200    2200    2963    3700   89000
```




Respectively for the mileage, we observed the unique observations of the listings mileage, we noticed that the data were in a format like this, “km 1,000”, so we had to remove the km symbol and the “,”, in order to transform them in numbers. Then we noticed from the unique mileage list, their summary, their histogram and their boxplot presented below , that were many listings having either very low or high kilometers, that usually means that the listing is fake, or the seller have made a mistake and instead of 15000 km for instance he has put 15 km. In both cases we cannot involve such extreme numbers for km, so we decided to calculate the 5% (22) and the 95% (83000) percentiles of the mileage list, and use that range for our listing mileage in the upcoming questions, keeping out the outliers that we discussed.

```
> summary(sorted_mileage)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
      1   11000   29001   34125   49000 1400000
```

```
> all_mileage_unique
  [1]    1    2    3    4    5    6    7    8    9   10   11
 [12]   12   13   14   15   16   17   18   19   20   21   22
 [23]   23   24   25   26   27   28   29   30   31   32   33
 [34]   34   35   36   37   38   39   40   41   42   43   44
 [45]   45   46   47   48   49   50   51   52   53   54   55
```




Regarding the age of the model, that were given as “Registration” in the listings, in "6 / 1999" format, we isolated the month and the year, and we added them in the listings as separated fields, “Month” and “Year” respectively.

We also checked if there was any special issue with brands and colors, but we did not find any significant issues. For the model section, we observed that sellers had added additional information of the model and also have included the word “negotiable” in many occasions. However, since there is a question related the negotiable listings, we decided not to change anything in the model section.

## 2.3 Questions

### 2.3.1 Load data in MongoDB

Once we finished our cleaning process, we created a for loop that loads every json file in a variable named “A”, then executes all the cleaning commands that we described in the previous section, where they were applying, and then a new variable B was creating in a json format. Then every listing were inserted in a data frame called “dfm”.

By the end of the loop we created a connection with MongoDB and we created a database and a collection, as “MB” , that contains all the listings.

### 2.3.2 How many bikes are there for sale?

We performed the $count command, which counts, all elements of our collection, and the outcome was 29701 bike listings.

```
> MB$count()
[1] 29701
```

### 2.3.3 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?

To answer this question we performed the command “$aggregate”. In this command we also included the “$match”, so we can filter the data, in order to include only the listings that were between the 5$^{th}$ and the 95$^{th}$ percentile, as we have mentioned in the cleaning section, so the fake listings or parts listings not to be included. We used the option “$avg” of the aggregate command, to calculate the average price, and then we also used the “$sum” : 1 to count how many listings were used in order to calculate the average price. The average price is €2551.93 and the number of bikes is 26794.

```
> #Question 3
> MB$aggregate('[{"$match": {"ad_data.Price" : {"$gte": 260, "$lte":80
00}}}, {"$group": {"_id": null, "avg_price": {"$avg": "$ad_data.Pric
e"}, "count_bikes":{"$sum": 1}}}]')
  _id avg_price count_bikes
1  NA  2551.933       26794
```

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

To answer this question we performed the command “$aggregate”. In this command again we used the “$match”, in order to include only the listings that were between the 5th and the 95th percentile, as we have mentioned in the cleaning section, so the fake listings or parts listings not to be included. We used the option “$min” of the aggregate command, to calculate the minimum price, the option “$max” of the aggregate command, to calculate the maximum price. Both minimum and maximum prices are exactly on the boundaries of the range that we have set, so we have minimum price €260 and maximum price €8000.

```
> #Question 4
> MB$aggregate('[{"$match": {"ad_data.Price" : {"$gte": 260, "$lte":80
00}}}, {"$group": {"_id": null, "min_price": {"$min": "$ad_data.Pric
e"}, "max_price":{"$max": "$ad_data.Price"}}}]')
  _id min_price max_price
1  NA       260      8000
```

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

Again we used the command “$aggregate”. This time in order to filter our data we used $match function in the “matadata.model” where we had observed in our cleaning process, that the word negotiable occurs in some listings. We used “$regex” function which identifies whether a particular regular expression is included in a specific field. Then as we have filtered our data we used the $sum: 1 to count the number of the listings which is equal to 1348.

```
> #Question 5
> MB$aggregate('[{"$match": {"metadata.model" : {"$regex" : "Negotiabl
e"}}}, {"$group": {"_id": null,  "count_negotiable_listings": {"$sum":
 1}}}]')
  _id count_negotiable_listings
1  NA                      1348
```

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

First we  group the documents by their "metadata.brand" and "metadata.model" fields, and count the number of documents in each group. For each group, we check if the "metadata.model" field contains the word "Negotiable", and set the "negotiable" field to 1 if it does, and 0 otherwise.

The pipeline then groups the resulting documents by their "metadata.brand" field, and computes the total count of all documents and the count of documents where "negotiable" is 1 for each brand.

Finally, the pipeline calculates the percentage of negotiable products for each brand, by dividing the count of negotiable products by the total count, and multiplying by 100. The outcome is presented below.

```
> #Question 6
> MB$aggregate('[[{"$group": {"_id": {"brand": "$metadata.brand","negotiable": { "$cond": [{ "$regexMatch": { "input": "$metadata.model", "regex": "Negotiable" } }, 1, 0] }},
+       "count": { "$sum": 1 }}},{
+     "$group": {
+       "_id": "$_id.brand",
+       "count_negotiable": { "$sum": { "$cond": [{ "$eq": ["$_id.negotiable", 1] }, "$count", 0] } },
+       "count_all": { "$sum": "$count" }}},{
+     "$project": {"_id": 0,"brand": "$_id","percentage_negotiable": {
  "$multiply": [{ "$divide": ["$count_negotiable", "$count_all"] }, 100] }}}]')
```

| | brand | percentage_negotiable |
|---|---|---|
| 1 | Adly | 0.000000 |
| 2 | Subaru | 0.000000 |
| 3 | Puch | 22.222222 |
| 4 | Derbi | 5.813953 |
| 5 | Beeline | 0.000000 |
| 6 | Victory | 100.000000 |
| 7 | Linhai | 4.166667 |
| 8 | HighPer | 100.000000 |
| 9 | Amstrong | 100.000000 |
| 10 | Shandong Liangzi | 0.000000 |
| 11 | Buell | 0.000000 |
| 12 | Quadro | 40.000000 |
| 13 | CAN-AM | 9.090909 |
| 14 | Peugeot | 1.960784 |
| 15 | Nipponia | 11.111111 |
| 16 | Tgb | 14.285714 |
| 17 | Adiva | 80.000000 |
| 18 | Polini | 0.000000 |
| 19 | Lambretta | 71.428571 |
| 20 | AGM motors | 20.000000 |
| 21 | Beta | 27.500000 |
| 22 | SMC | 33.333333 |
| 23 | Lingben | 0.000000 |
| 24 | Dayang | 16.666667 |
| 25 | Enfield | 0.000000 |
| 26 | Daelim | 1.449275 |
| 27 | Asus | 20.000000 |
| 28 | Nsu | 0.000000 |
| 29 | Dinli | 0.000000 |
| 30 | Mikilon | 0.000000 |
| 31 | Sachs | 30.645161 |
| 32 | Gemini | 66.666667 |
| 33 | Euromotors | 0.000000 |
| 34 | Jmstar | 100.000000 |
| 35 | Benelli | 7.317073 |
| 36 | Ducati | 1.608579 |
| 37 | Lifan | 24.038462 |
| 38 | Xingyue | 37.500000 |
| 39 | Adler | 0.000000 |
| 40 | Moto Guzzi | 2.000000 |
| 41 | Vor | 16.666667 |
| 42 | Maico | 25.000000 |
| 43 | Polaris | 17.142857 |
| 44 | Unili | 0.000000 |
| 45 | Laverda | 0.000000 |
| 46 | TCB | 9.090909 |
| 47 | G-force | 0.000000 |
| 48 | Lem | 25.000000 |
| 49 | Arctic Cat | 25.000000 |
| 50 | CPI | 10.000000 |
| 51 | Hyosung | 11.627907 |
| 52 | Shan Yang | 0.000000 |
| 53 | Husaberg | 3.333333 |
| 54 | Piaggio | 1.675978 |
| 55 | Access | 33.333333 |
| 56 | Kinroad | 25.000000 |
| 57 | Joyner | 100.000000 |
| 58 | Montesa | 33.333333 |
| 59 | Vedim | 0.000000 |
| 60 | Xgjao | 87.500000 |
| 61 | Cectek | 0.000000 |
| 62 | Hsun | 28.571429 |
| 63 | Kymco | 1.829268 |
| 64 | Gas-Gas | 8.571429 |
| 65 | Norton | 20.000000 |
| 66 | Aie | 0.000000 |
| 67 | Shineray | 10.714286 |
| 68 | Pgo | 10.000000 |
| 69 | Daytona | 13.231552 |
| 70 | Hartford | 0.000000 |
| 71 | KL | 0.000000 |
| 72 | Horex | 16.666667 |
| 73 | Access Motor | 28.571429 |
| 74 | Goes | 0.000000 |
| 75 | Genata | 0.000000 |
| 76 | Loncin | 12.000000 |
| 77 | Sherco | 100.000000 |
| 78 | Buggy Motors | 100.000000 |
| 79 | Bajaj | 33.333333 |
| 80 | Gilera | 3.050847 |
| 81 | Vespa | 5.109489 |
| 82 | Harlow | 0.000000 |
| 83 | E-ATV | 100.000000 |
| 84 | Nomik | 50.000000 |
| 85 | Morini | 0.000000 |
| 86 | Jawa | 43.750000 |
| 87 | Seckam | 0.000000 |
| 88 | AJP | 0.000000 |
| 89 | Bmw | 2.008608 |
| 90 | Haojin | 70.000000 |
| 91 | Dkw | 56.250000 |
| 92 | Nitro Motors | 100.000000 |
| 93 | Aeon | 3.030303 |
| 94 | Victoria | 0.000000 |
| 95 | Kreidler | 20.481928 |
| 96 | Xinling | 0.000000 |
| 97 | Boom-Trikes | 100.000000 |
| 98 | Emw | 50.000000 |
| 99 | Bultaco | 50.000000 |
| 100 | Imr | 40.000000 |
| 101 | Regal-Raptor | 100.000000 |
| 102 | Super Moto | 50.000000 |
| 103 | Fever | 100.000000 |
| 104 | Zongshen | 19.230769 |
| 105 | Yamaha | 2.821487 |
| 106 | AMS | 0.000000 |
| 107 | Solex | 26.666667 |
| 108 | Motivas | 33.333333 |
| 109 | Malaguti | 3.636364 |
| 110 | Semog | 0.000000 |
| 111 | New Force Motor | 0.000000 |
| 112 | Generic | 16.666667 |
| 113 | FB Mondial | 50.000000 |
| 114 | Keeway | 8.737864 |
| 115 | Αλλο | 35.693215 |
| 116 | Fym | 0.000000 |
| 117 | Honda | 3.990307 |
| 118 | Modenas | 3.831418 |
| 119 | Royal Enfield | 38.461538 |
| 120 | Gamax | 0.000000 |
| 121 | Dias | 100.000000 |
| 122 | Skyjet | 50.000000 |
| 123 | Brixton | 0.000000 |
| 124 | Bimota | 0.000000 |
| 125 | Sokudo | 0.000000 |
| 126 | Mtg | 100.000000 |
| 127 | Jonway | 60.000000 |
| 128 | Zundapp | 32.258065 |
| 129 | Jincheng | 28.571429 |
| 130 | JetMoto | 50.000000 |
| 131 | Niu | 100.000000 |
| 132 | Vee Raad | 100.000000 |
| 133 | LML | 22.222222 |
| 134 | AB | 61.538462 |
| 135 | Suzuki | 2.198732 |
| 136 | Odess | 100.000000 |
| 137 | Garelli | 12.500000 |
| 138 | Mobster | 0.000000 |
| 139 | Jianshe | 0.000000 |
| 140 | Nova | 0.000000 |
| 141 | Cheetah | 35.294118 |
| 142 | Kawasaki | 3.328213 |
| 143 | KTM | 2.173913 |
| 144 | Rewaco | 33.333333 |
| 145 | Simson | 27.272727 |
| 146 | X-Motors | 0.000000 |
| 147 | Triumph | 1.791045 |
| 148 | Kaisar | 75.000000 |
| 149 | Wsk | 100.000000 |
| 150 | Fuxin | 25.000000 |
| 151 | Skyteam | 2.272727 |
| 152 | Dirt Motos | 40.000000 |
| 153 | Moto Morini | 25.000000 |
| 154 | Motobi | 100.000000 |
| 155 | Jinlun | 100.000000 |
| 156 | Znen | 66.666667 |
| 157 | Italjet | 20.000000 |
| 158 | E-Ton | 50.000000 |
| 159 | Bombardier | 100.000000 |
| 160 | Mv Agusta | 6.250000 |
| 161 | Barossa | 0.000000 |
| 162 | Heinkel | 40.000000 |
| 163 | Lintex | 33.333333 |
| 164 | Qingqi | 100.000000 |
| 165 | Indian | 66.666667 |
| 166 | Bashan | 52.380952 |
| 167 | Vmoto | 16.666667 |
| 168 | Aprilia | 1.793722 |
| 169 | Ural | 20.000000 |
| 170 | Harley Davidson | 3.883495 |
| 171 | Geely | 0.000000 |
| 172 | Chang Jiang | 0.000000 |
| 173 | Zuendapp | 42.857143 |
| 174 | Swm | 33.333333 |
| 175 | Sym | 1.834862 |
| 176 | Cagiva | 9.375000 |
| 177 | Bsa | 33.333333 |
| 178 | Ccm | 0.000000 |
| 179 | Masai | 0.000000 |
| 180 | CFmoto | 14.285714 |
| 181 | Emb | 0.000000 |
| 182 | Siamoto | 20.000000 |
| 183 | MBK | 40.000000 |
| 184 | Apokotos | 100.000000 |
| 185 | Baotian | 57.142857 |
| 186 | Husqvarna | 4.137931 |
| 187 | Ariel | 0.000000 |
| 188 | Boatian | 33.333333 |
| 189 | Kuberg | 100.000000 |
| 190 | Jialing | 5.000000 |
| 191 | Eagle | 0.000000 |
| 192 | Hercules | 33.333333 |
| 193 | TM | 18.181818 |
| 194 | Kxd | 14.285714 |
| 195 | Ymc | 22.222222 |
| 196 | MZ | 27.272727 |
| 197 | ZhongYu | 100.000000 |
| 198 | Evomoto | 0.000000 |

## 2.3.7 What is the motorcycle brand with the highest average price?

We used the aggregate function to calculate the average. Also as we have already done in previous questions we filtered our data regarding their price. Then using the $group we group the bikes by their brand and their average price. Then we sort them in descending order based their average price, and we put limit 1 to find the brand with the biggest average price. Then we used project to transform our outcome to present the brand, and its average price. The brand is Morini and its average price €6500.

```
> #Question 7
> MB$aggregate('[{"$match": {"ad_data.Price" : {"$gte": 260, "$lte":80
00}}},{
+     "$group": {"_id": "$metadata.brand","avg_price": {"$avg": "$ad_d
ata.Price"}}},
+   { "$sort": { "avg_price": -1 } },
+   { "$limit": 1 },
+   { "$project": { "_id": "$_id", "max_brand_average": "$avg_price" }
 }]')
     _id max_brand_average
1 Morini              6500
```

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

We used the aggregate function. We filtered our data as previously. We group our data by the model of the bike and then we calculated the average age. To calculate the age we add 2023 with minus the $Year that we have inserted to the listings through R. As previously we sort the data in descending order and we put as limit the first 10. Finally we use project to transform our outcome as follows.

```
> #Question 8
> MB$aggregate('[{"$match": {"ad_data.Price": {"$gte": 260,"$lte": 8000}}},{
+     "$group": {"_id": "$metadata.model","avg_age": {"$avg": {"$add": [2023,{"$multiply": [-
1,"$ad_data.Year"]}]} }}},{ "$sort": { "avg_age": -1 } },{ "$limit": 10 },
+   { "$project": { "_id": "$_id", "top10_models_age_average": "$avg_age" } }]')
                                          _id top10_models_age_average
1  Αλλο  henderson indian replica '31 - € 1.500 EUR                  92
2          Αλλο  MATCHLESS G3 350 '35 - € 5.500 EUR                  88
3                  Αλλο  Matsoules '38 - € 700 EUR                   85
4              Bsa    '39 - € 5.500 EUR (Negotiable)                 84
5             Bsa  M20 ARMY MOTO! '39 - € 3.000 EUR                  84
6           Αλλο  Matchless G3/L '39 - € 4.000 EUR                   84
7               Αλλο  NEW HUDSON '39 - € 3.800 EUR                   84
8                       Bsa  M20 '39 - € 5.500 EUR                   84
9                          Bsa   '39 - € 5.500 EUR                   84
10                       Simson   '40 - € 300 EUR                    83
```

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

We use aggregate function, and we filter our data to include only the bikes with Price between the range that we have discussed. Also we use “$regex “ to extras to include only the bikes that contain ABS. Finally we count them, and we found 3285 bikes.

```
> MB$aggregate('[{"$match": {"extras": {"$regex": "ABS"},"ad_data.Price":
 {"$gte": 260,"$lte": 8000}}},
+             {"$group": {"_id": null,"count_bikes_with_ABS": {"$sum":
 1}}}]')
  _id count_bikes_with_ABS
1  NA                 3285
```

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

In this questions we performed exactly the same operation as the previously with the following changes. In the $match field we added also the bike to have in extras the regular expression “Led Lights”, and the mileage of the listing to be in the range discussed in the cleaning section (22-83000 km). Then we calculated the average mileage for the filtered data that is 28790.01 km.

```
> #Question 10
> MB$aggregate('[{"$match": {"extras": {"$regex": "ABS"},"extras": {"$rege
x": "Led lights"},"ad_data.Price": {"$gte": 260,"$lte": 8000},"ad_data.Mil
eage": {"$gte": 22,"$lte": 83000}}},{"$group": {"_id": null,"average_milea
ge": {"$avg": "$ad_data.Mileage"}}}]')
  _id average_mileage
1  NA        28790.01
```