<u>-: The Report Of our Project:-</u>

First, we cleaned the data using the 'janitor' package

Janitor package:-

We have functions to clean data ‘we will talk about it now’

1- we used the clean_names() function to make all set lowercase, and used "_" as a separator.

2- we used remove_empty() function which removes any column that is empty and entire row that is empty.

Dplyr Package:-

We used this package “distinct” to remove any column that is empty and entire row that is empty.

==

Now we will talk about everyone’s job

The members who did the K’MEANS are:-

1- Ali Amr Ali (G2)(20221460130)

2- Abdalla Gaber Ashour (G3)(20221460120)

And their notes on their code are:-

What does the program do?

- **The program will take from the user the number of clusters to display a table.**
  This table will display the age, customer name, total, and the clusters divided into groups depending on the number of clusters.

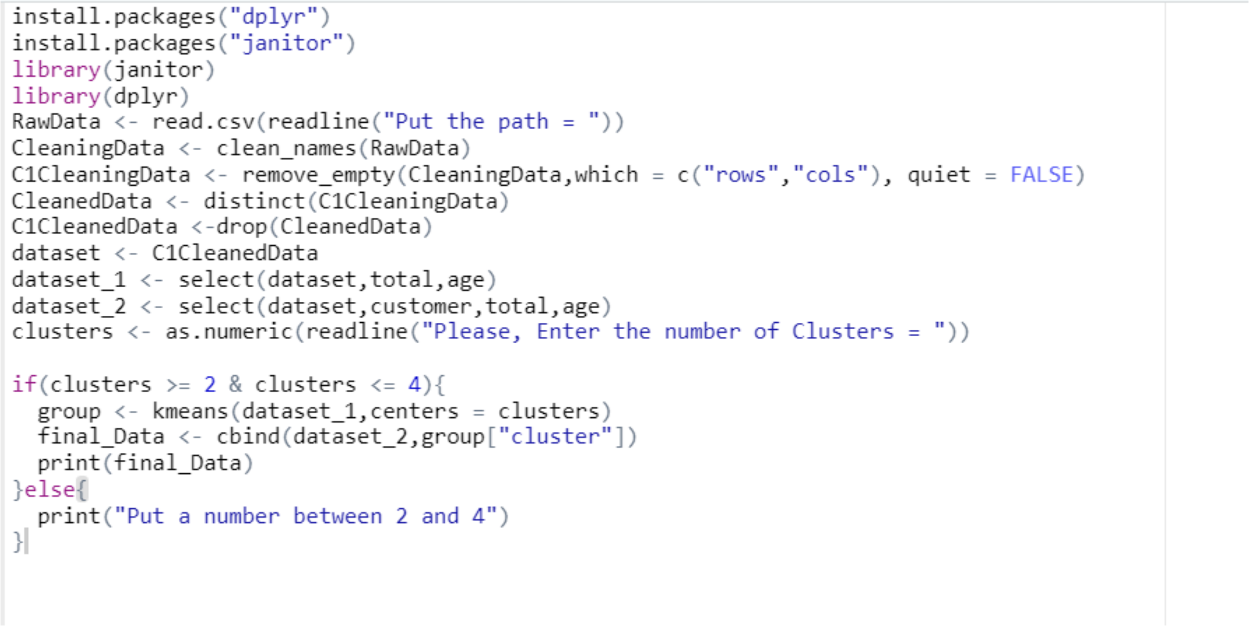
What will the input of the program be?

- **After selecting my required things, such as (total and age) using dplyr, and selected function will use Select code to take from my data_set (Total and Age), and will call it Data_Set_1.**
- **Then we will select the same data_set to print all customer names, ages, total, and the computed clusters numbers.**
- **After selecting all what I want to work on, we must make the user take care of choosing his cluster by using the if statement and telling the user * Please, Enter the number of Clusters = ***
- **If the user put (From 2 to 4), then the user can go to the next step.**
- **Now by using the built-in function *K-mean* to calculate the *Data_Set_1* we will center = the n of clusters which the user put.**
- **Finally, we will use C-Bind to bind the columns, clusters, total, and age WITH Data_Set_2(total, age, customer name)**

What will the output of the program?

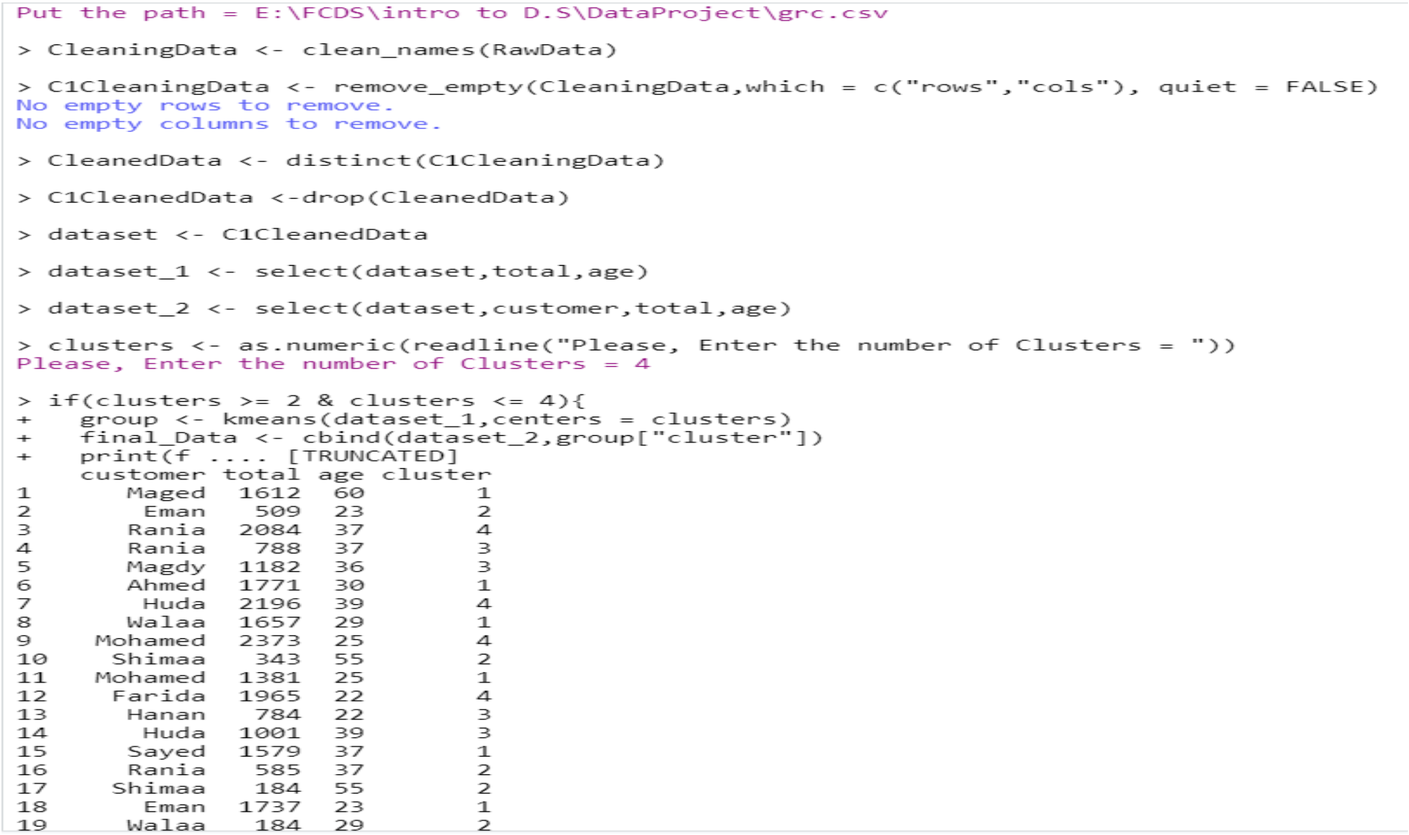
- **A table displaying each of, (Customer name), (Age), (Total), (And the clusters).**

And this is The full code of k-mean(with cleaning code data):-



```
install.packages("dplyr")
install.packages("janitor")
library(janitor)
library(dplyr)
RawData <- read.csv(readline("Put the path = "))
CleaningData <- clean_names(RawData)
C1CleaningData <- remove_empty(CleaningData,which = c("rows","cols"), quiet = FALSE)
CleanedData <- distinct(C1CleaningData)
C1CleanedData <-drop(CleanedData)
dataset <- C1CleanedData
dataset_1 <- select(dataset,total,age)
dataset_2 <- select(dataset,customer,total,age)
clusters <- as.numeric(readline("Please, Enter the number of Clusters = "))

if(clusters >= 2 & clusters <= 4){
  group <- kmeans(dataset_1,centers = clusters)
  final_Data <- cbind(dataset_2,group["cluster"])
  print(final_Data)
}else{
  print("Put a number between 2 and 4")
}
```

An example of the Output



```
Put the path = E:\FCDS\intro to D.S\DataProject\grc.csv

> CleaningData <- clean_names(RawData)

> C1CleaningData <- remove_empty(CleaningData,which = c("rows","cols"), quiet = FALSE)
No empty rows to remove.
No empty columns to remove.

> CleanedData <- distinct(C1CleaningData)

> C1CleanedData <-drop(CleanedData)

> dataset <- C1CleanedData

> dataset_1 <- select(dataset,total,age)

> dataset_2 <- select(dataset,customer,total,age)

> clusters <- as.numeric(readline("Please, Enter the number of Clusters = "))
Please, Enter the number of Clusters = 4

> if(clusters >= 2 & clusters <= 4){
+   group <- kmeans(dataset_1,centers = clusters)
+   final_Data <- cbind(dataset_2,group["cluster"])
+   print(f .... [TRUNCATED]
    customer total age cluster
1      Maged  1612  60       1
2       Eman   509  23       2
3      Rania  2084  37       4
4      Rania   788  37       3
5      Magdy  1182  36       3
6      Ahmed  1771  30       1
7       Huda  2196  39       4
8      Walaa  1657  29       1
9    Mohamed  2373  25       4
10    Shimaa   343  55       2
11   Mohamed  1381  25       1
12    Farida  1965  22       4
13     Hanan   784  22       3
14      Huda  1001  39       3
15     Sayed  1579  37       1
16     Rania   585  37       2
17    Shimaa   184  55       2
18      Eman  1737  23       1
19     Walaa   184  29       2
```

===

The members who did the Association Rule are:-

1- Mahmoud Essam Fathy (G3) (20221460231)

2- Aya Nabil (G3), ID (20221452375)

And their notes on their code are:-

What will the program do?

- **The program should take from the user minimum support and minimum confidence then the association rule will use them to generate it.**

What will the input to the program be?

- **After using cleaned data, the program should use package "Dplyr" to make the following requirement:**
  - **select the items from GRC.csv**
- **After we used "select" to choose our columns from the excel to work on, then we used Write Table to create a new table from selected columns in the excel sheet, then read it again as transactions.**
- **Then we will install the package called ("Arules"), as this package will do the association rule as required.**

We will use 2 things in "arules":-

1- read. transactions -> which reads the transactions of items -we have done - from the excel sheet.

2- Apriori -> the function which will make the algorithm

- **we started to ask the user to put 2 things:**

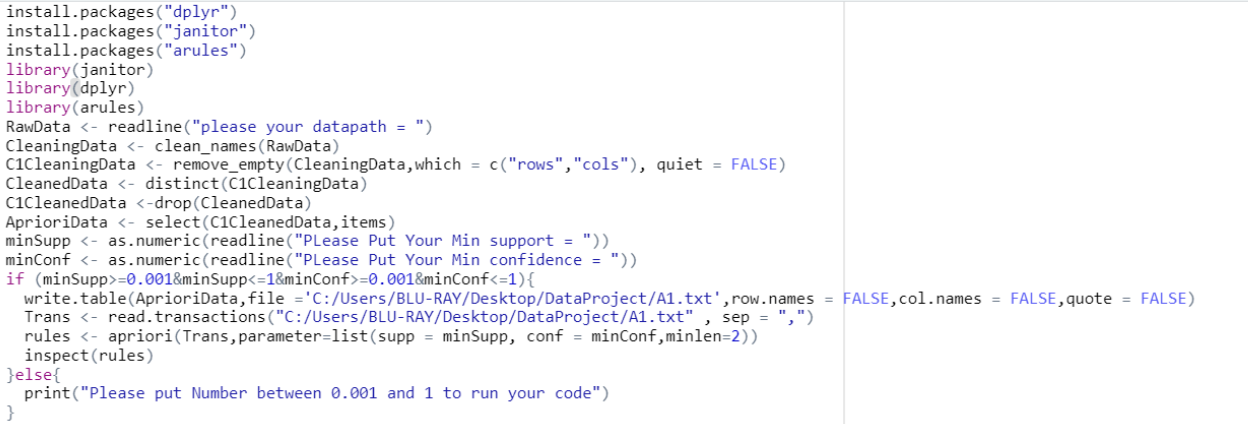
1- choose the minimum support (from 0.001 to 1)

2- choose the minimum confidence (also from 0.001 to 1)

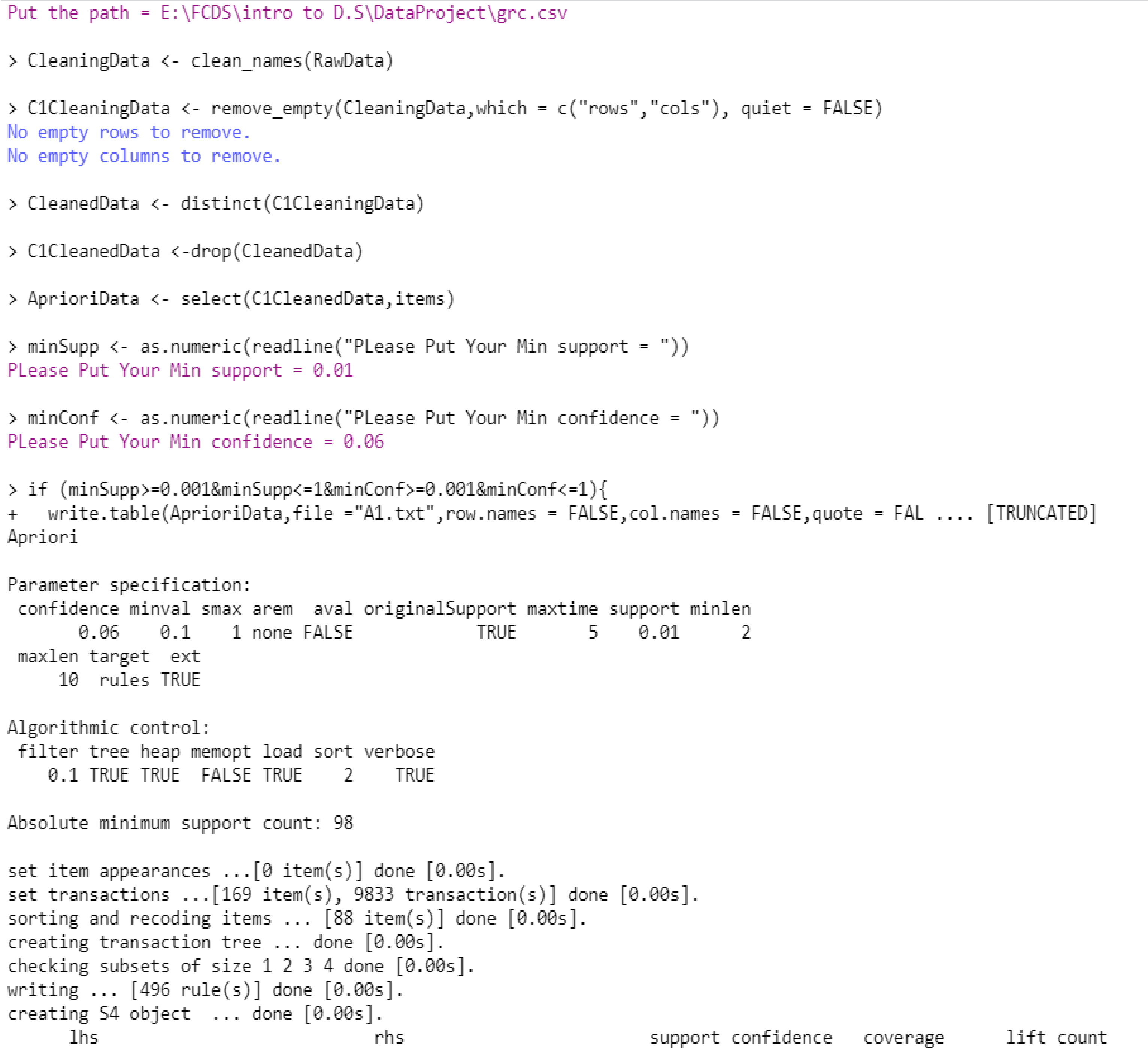
What will the output from the program be?

- **Generation of Rules with perfect confidence and support to achieve the required goal**

And this is the full apriori code(with cleaning code data)



```
install.packages("dplyr")
install.packages("janitor")
install.packages("arules")
library(janitor)
library(dplyr)
library(arules)
RawData <- readline("please your datapath = ")
CleaningData <- clean_names(RawData)
C1CleaningData <- remove_empty(CleaningData,which = c("rows","cols"), quiet = FALSE)
CleanedData <- distinct(C1CleaningData)
C1CleanedData <-drop(CleanedData)
AprioriData <- select(C1CleanedData,items)
minSupp <- as.numeric(readline("PLease Put Your Min support = "))
minConf <- as.numeric(readline("PLease Put Your Min confidence = "))
if (minSupp>=0.001&minSupp<=1&minConf>=0.001&minConf<=1){
  write.table(AprioriData,file ='C:/Users/BLU-RAY/Desktop/DataProject/A1.txt',row.names = FALSE,col.names = FALSE,quote = FALSE)
  Trans <- read.transactions("C:/Users/BLU-RAY/Desktop/DataProject/A1.txt" , sep = ",")
  rules <- apriori(Trans,parameter=list(supp = minSupp, conf = minConf,minlen=2))
  inspect(rules)
}else{
  print("Please put Number between 0.001 and 1 to run your code")
}
```



```
Put the path = E:\FCDS\intro to D.S\DataProject\grc.csv

> CleaningData <- clean_names(RawData)

> C1CleaningData <- remove_empty(CleaningData,which = c("rows","cols"), quiet = FALSE)
No empty rows to remove.
No empty columns to remove.

> CleanedData <- distinct(C1CleaningData)

> C1CleanedData <-drop(CleanedData)

> AprioriData <- select(C1CleanedData,items)

> minSupp <- as.numeric(readline("PLease Put Your Min support = "))
PLease Put Your Min support = 0.01

> minConf <- as.numeric(readline("PLease Put Your Min confidence = "))
PLease Put Your Min confidence = 0.06

> if (minSupp>=0.001&minSupp<=1&minConf>=0.001&minConf<=1){
+   write.table(AprioriData,file ="A1.txt",row.names = FALSE,col.names = FALSE,quote = FAL .... [TRUNCATED] 
Apriori

Parameter specification:
 confidence minval smax arem  aval originalSupport maxtime support minlen
       0.06    0.1    1 none FALSE            TRUE       5    0.01      2
 maxlen target  ext
     10  rules TRUE

Algorithmic control:
 filter tree heap memopt load sort verbose
    0.1 TRUE TRUE  FALSE TRUE    2    TRUE

Absolute minimum support count: 98 

set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[169 item(s), 9833 transaction(s)] done [0.00s].
sorting and recoding items ... [88 item(s)] done [0.00s].
creating transaction tree ... done [0.00s].
checking subsets of size 1 2 3 4 done [0.00s].
writing ... [496 rule(s)] done [0.00s].
creating S4 object  ... done [0.00s].
      lhs                            rhs                          support confidence    coverage       lift count
```

The member who did the visualization code is:-

1- Ahmed Hesham (G3)(20221453234)

And his note on his code is:-

What does the program do?

1 – Compares total spending by cash or by credit in the pie chart.

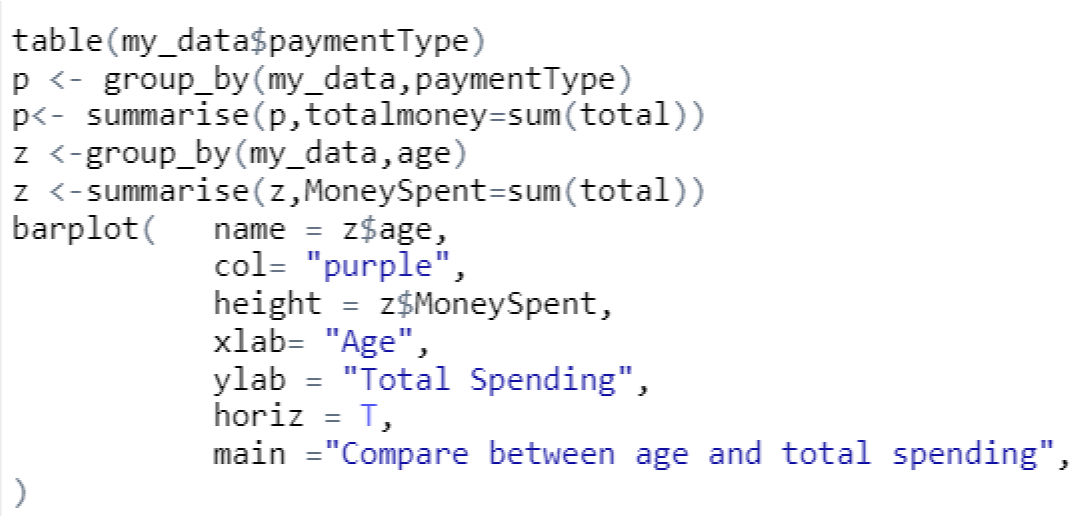
2 – Compares each age and the sum of total spending in the horizontal bar-plot chart.

3 – Shows each city's total spending and arrange it by total descending in a vertical bar plot.

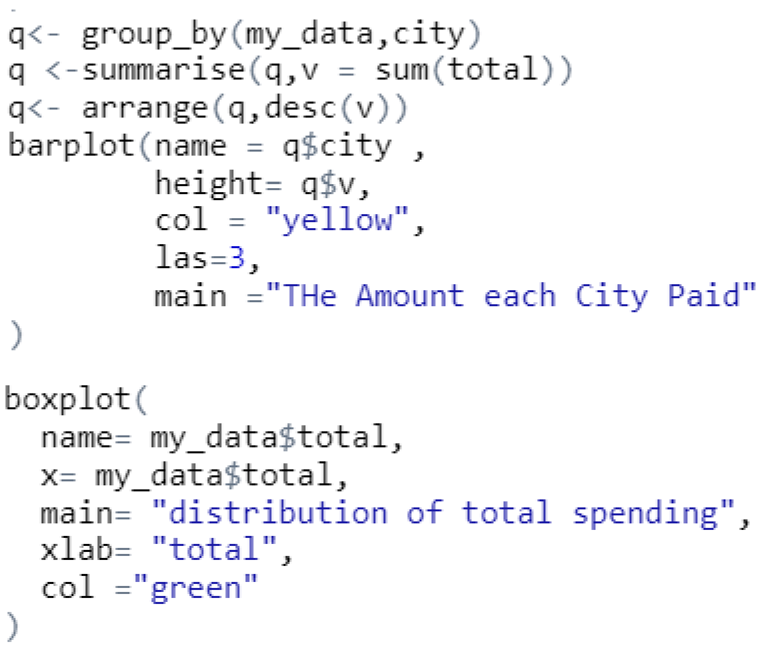
4 – Displays the distribution of total spending in a box plot chart.

5 – puts all the comparison charts in one dashboard.

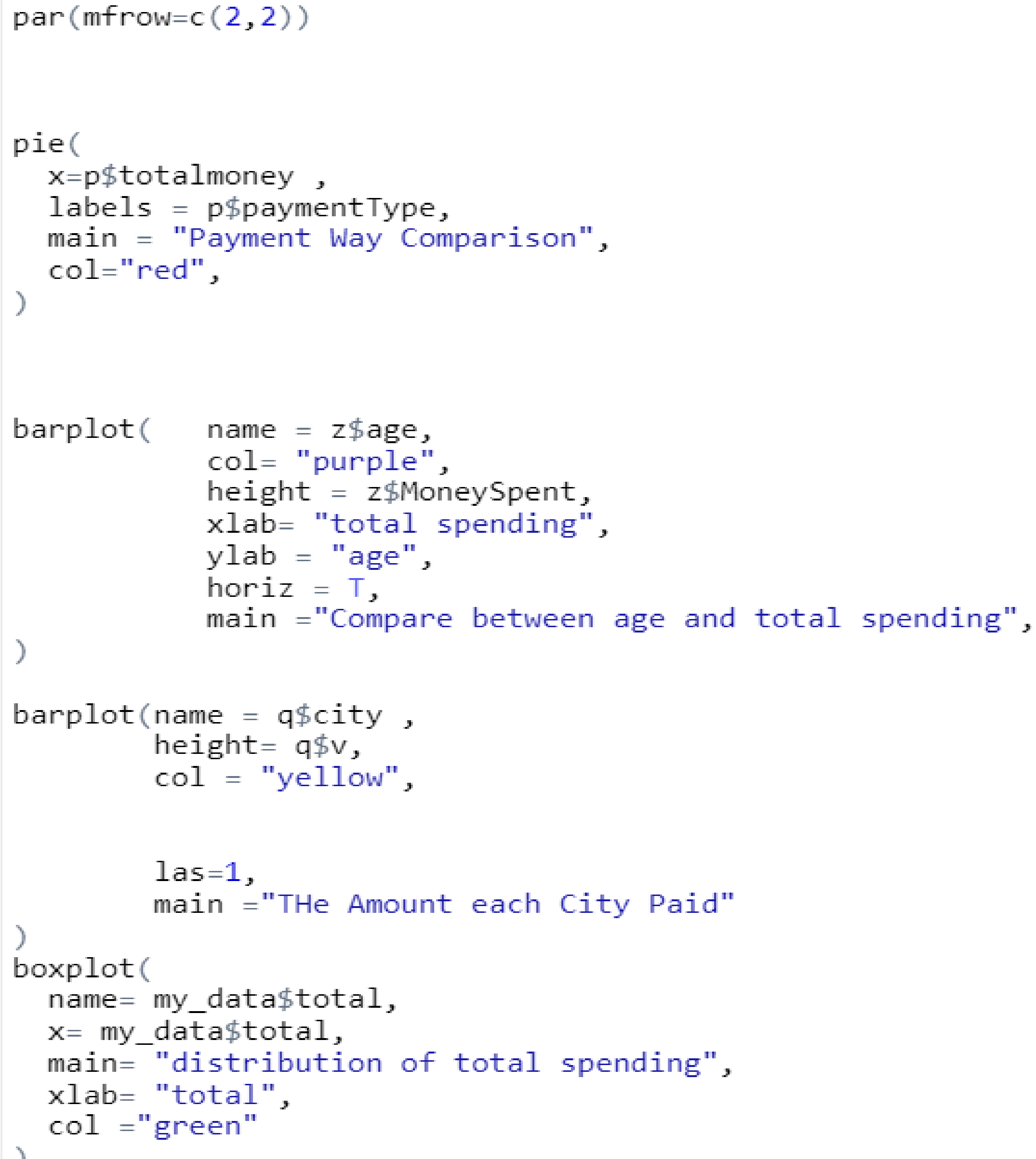
What will the input to the program be?



```
table(my_data$paymentType)
p <- group_by(my_data,paymentType)
p<- summarise(p,totalmoney=sum(total))
z <-group_by(my_data,age)
z <-summarise(z,MoneySpent=sum(total))
barplot(    name = z$age,
            col= "purple",
            height = z$MoneySpent,
            xlab= "Age",
            ylab = "Total Spending",
            horiz = T,
            main ="Compare between age and total spending",
)
```



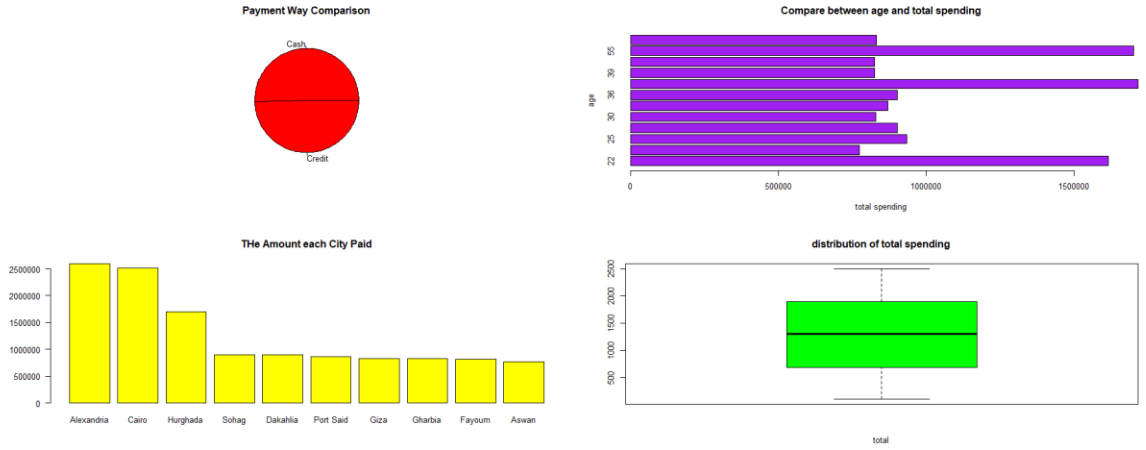
```
q<- group_by(my_data,city)
q <-summarise(q,v = sum(total))
q<- arrange(q,desc(v))
barplot(name = q$city ,
        height= q$v,
        col = "yellow",
        las=3,
        main ="THe Amount each City Paid"
)

boxplot(
  name= my_data$total,
  x= my_data$total,
  main= "distribution of total spending",
  xlab= "total",
  col ="green"
)
```



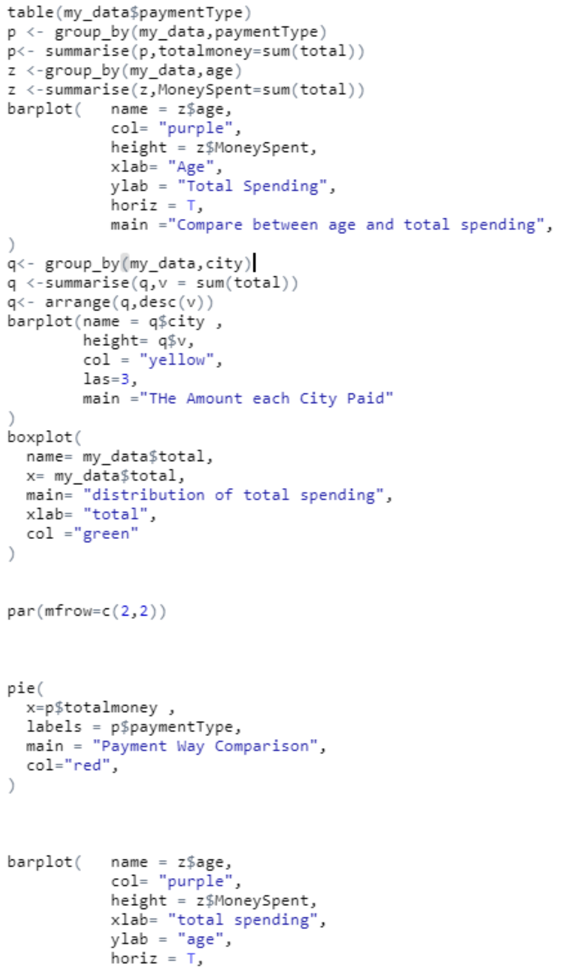
```
par(mfrow=c(2,2))



pie(
  x=p$totalmoney ,
  labels = p$paymentType,
  main = "Payment Way Comparison",
  col="red",
)



barplot(    name = z$age,
            col= "purple",
            height = z$MoneySpent,
            xlab= "total spending",
            ylab = "age",
            horiz = T,
            main ="Compare between age and total spending",
)

barplot(name = q$city ,
        height= q$v,
        col = "yellow",


        las=1,
        main ="THe Amount each City Paid"
)
boxplot(
  name= my_data$total,
  x= my_data$total,
  main= "distribution of total spending",
  xlab= "total",
  col ="green"
)
```

What are the outputs of the program?

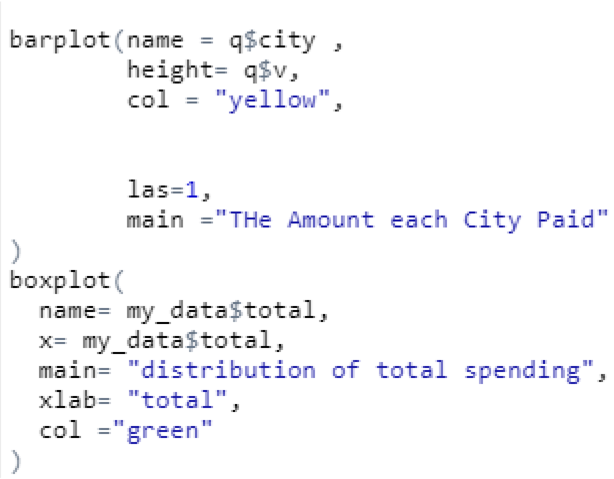
The program compares between a lot of data that everyone can understand by just looking at it



And this is The full code of DATA visualization

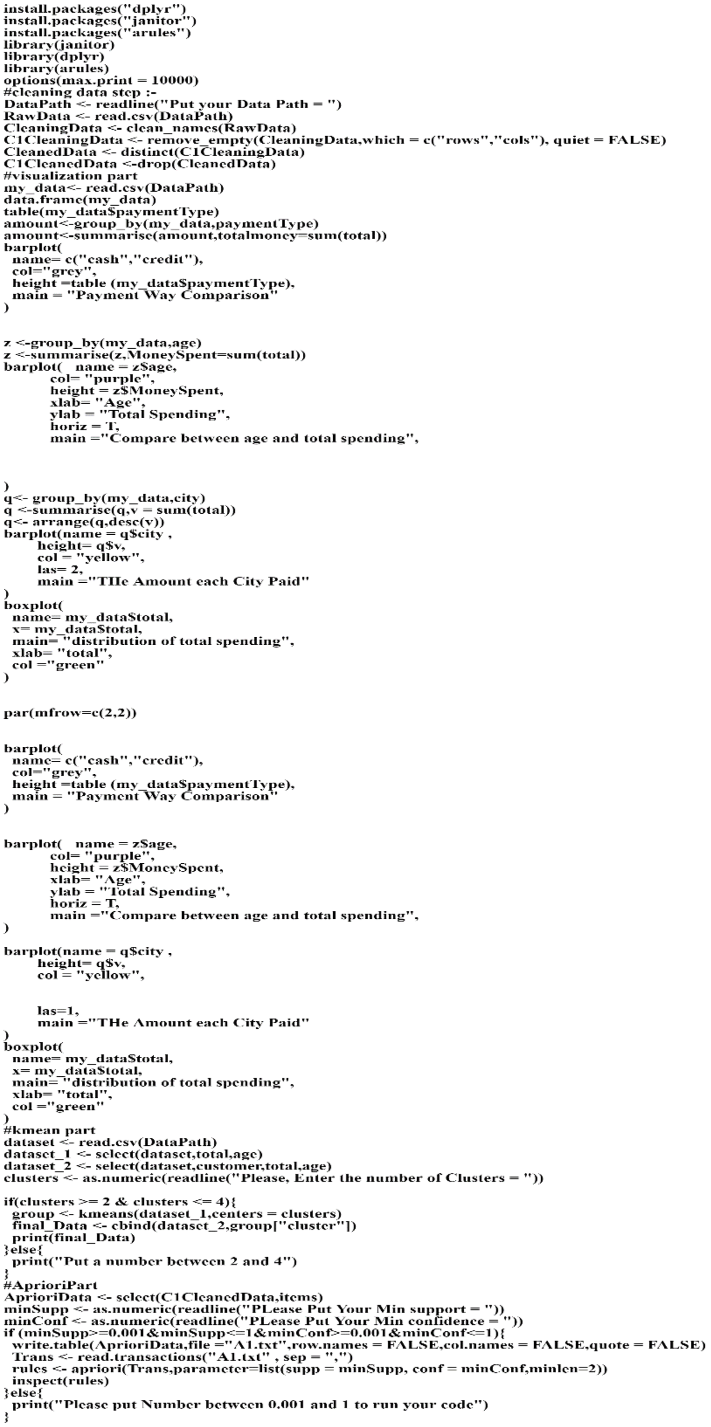


```
table(my_data$paymentType)
p <- group_by(my_data,paymentType)
p<- summarise(p,totalmoney=sum(total))
z <-group_by(my_data,age)
z <-summarise(z,MoneySpent=sum(total))
barplot(    name = z$age,
            col= "purple",
            height = z$MoneySpent,
            xlab= "Age",
            ylab = "Total Spending",
            horiz = T,
            main ="Compare between age and total spending",
)
q<- group_by(my_data,city)
q <-summarise(q,v = sum(total))
q<- arrange(q,desc(v))
barplot(name = q$city ,
         height= q$v,
         col = "yellow",
         las=3,
         main ="THe Amount each City Paid"
)
boxplot(
  name= my_data$total,
  x= my_data$total,
  main= "distribution of total spending",
  xlab= "total",
  col ="green"
)


par(mfrow=c(2,2))



pie(
  x=p$totalmoney ,
  labels = p$paymentType,
  main = "Payment Way Comparison",
  col="red",
)



barplot(    name = z$age,
            col= "purple",
            height = z$MoneySpent,
            xlab= "total spending",
            ylab = "age",
            horiz = T,
```



```
barplot(name = q$city ,
        height= q$v,
        col = "yellow",


        las=1,
        main ="THe Amount each City Paid"
)
boxplot(
  name= my_data$total,
  x= my_data$total,
  main= "distribution of total spending",
  xlab= "total",
  col ="green"
)
```

The last form of the “FullProject.R”



```
install.packages("dplyr")
install.packages("janitor")
install.packages("arules")
library(janitor)
library(dplyr)
library(arules)
options(max.print = 10000)
#cleaning data step :-
DataPath <- readline("Put your Data Path = ")
RawData <- read.csv(DataPath)
CleaningData <- clean_names(RawData)
C1CleaningData <- remove_empty(CleaningData,which = c("rows","cols"), quiet = FALSE)
CleanedData <- distinct(C1CleaningData)
C1CleanedData <-drop(CleanedData)
#visualization part
my_data<- read.csv(DataPath)
data.frame(my_data)
table(my_data$paymentType)
amount<-group_by(my_data,paymentType)
amount<-summarise(amount,totalmoney=sum(total))
barplot(
  name= c("cash","credit"),
  col="grey",
  height =table (my_data$paymentType),
  main = "Payment Way Comparison"
)


z <-group_by(my_data,age)
z <-summarise(z,MoneySpent=sum(total))
barplot(   name = z$age,
           col= "purple",
           height = z$MoneySpent,
           xlab= "Age",
           ylab = "Total Spending",
           horiz = T,
           main ="Compare between age and total spending",



)
q<- group_by(my_data,city)
q <-summarise(q,v = sum(total))
q<- arrange(q,desc(v))
barplot(name = q$city ,
        height= q$v,
        col = "yellow",
        las= 2,
        main ="THe Amount each City Paid"
)
boxplot(
  name= my_data$total,
  x= my_data$total,
  main= "distribution of total spending",
  xlab= "total",
  col ="green"
)


par(mfrow=c(2,2))


barplot(
  name= c("cash","credit"),
  col="grey",
  height =table (my_data$paymentType),
  main = "Payment Way Comparison"
)


barplot(   name = z$age,
           col= "purple",
           height = z$MoneySpent,
           xlab= "Age",
           ylab = "Total Spending",
           horiz = T,
           main ="Compare between age and total spending",
)

barplot(name = q$city ,
        height= q$v,
        col = "yellow",


        las=1,
        main ="THe Amount each City Paid"
)
boxplot(
  name= my_data$total,
  x= my_data$total,
  main= "distribution of total spending",
  xlab= "total",
  col ="green"
)
#kmean part
dataset <- read.csv(DataPath)
dataset_1 <- select(dataset,total,age)
dataset_2 <- select(dataset,customer,total,age)
clusters <- as.numeric(readline("Please, Enter the number of Clusters = "))

if(clusters >= 2 & clusters <= 4){
  group <- kmeans(dataset_1,centers = clusters)
  final_Data <- cbind(dataset_2,group["cluster"])
  print(final_Data)
}else{
  print("Put a number between 2 and 4")
}
#AprioriPart
AprioriData <- select(C1CleanedData,items)
minSupp <- as.numeric(readline("PLease Put Your Min support = "))
minConf <- as.numeric(readline("PLease Put Your Min confidence = "))
if (minSupp>=0.001&minSupp<=1&minConf>=0.001&minConf<=1){
  write.table(AprioriData,file ="A1.txt",row.names = FALSE,col.names = FALSE,quote = FALSE)
  Trans <- read.transactions("A1.txt" , sep = ",")
  rules <- apriori(Trans,parameter=list(supp = minSupp, conf = minConf,minlen=2))
  inspect(rules)
}else{
  print("Please put Number between 0.001 and 1 to run your code")
}
```

this project

was carried out

under the supervision of

DR. Magda Madbouly

and by her students:-

Ali Amr Ali (G2)(20221460130)

Abdalla Gaber Ashour (G3)(20221460120)

Mahmoud Essam Fathy (G3) (20221460231)

Aya Nabil (G3), ID (20221452375)

Ahmed Hesham (G3)(20221453234)