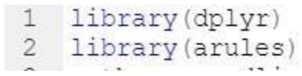
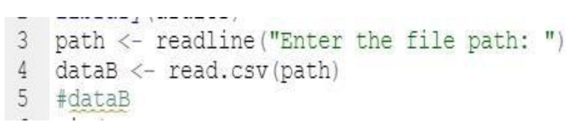
Name	**Student role in this project**	**ID**	**Group**
Bishoy Ayad Fahmy Habib Gadelrab	(Data Visualization) 1/Compare cash and credit totals. 2/Compare each age and sum of total spending. 3/Put all previous plots in one dashboard	20221374009	G1
Mohab Emad Morris Khela	(Data Visualization) 1/Show each city total spending and arrange it by total descending. 2/Display the distribution of total spending.	20221374199	G1
Mahmoud Yasser Abdelrazek Rizk Saeed	(K-mean) *Split the customers to (n) groups using one of the studied methods (n will be user input) according to the sum of total spending and their ages and print a table displaying each customer name, age, total and the computed cluster number.	20221372316	G1
Gaser Ezzat Abdelaziz Aboelazm Youssef	(K-mean) *Split the customers to (n) groups using one of the studied methods (n will be user input) according to the sum of total spending and their ages and print a table displaying each customer name, age, total and the computed cluster number.	20221442258	G2
Abdelrahman Ashraf Ragab Mohammed Hassan	(Association Rules) *Generate association rules between items with minimum support and confidence taken from the user inputs (State the algorithm used). Apriori algorithm	20221374041	G1

(Note that the R script has some comments to check and test a few things)



```
1  library(dplyr)
2  library(arules)
```

These lines are used to import libraries we will be use in this project.

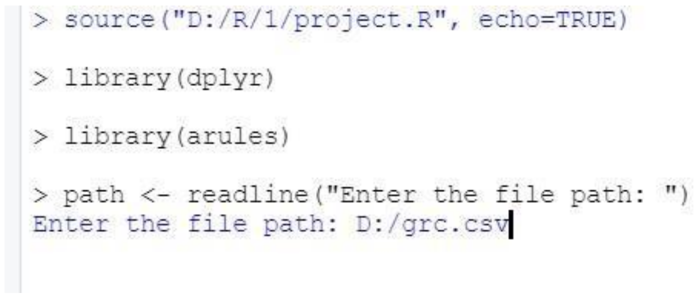


```
3  path <- readline("Enter the file path: ")
4  dataB <- read.csv(path)
5  #dataB
```

Problem is to take dataset path from user…solved by

Line 3: To take dataset path from user.

Line 4: To read this file in csv file.   (Note: Line 5 is a comment) **<u>Output for this code :</u>**

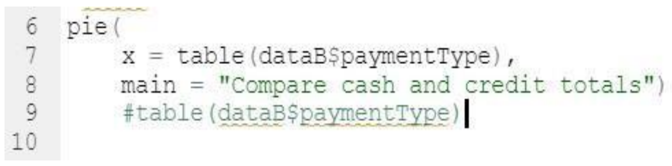


```
> source("D:/R/1/project.R", echo=TRUE)

> library(dplyr)

> library(arules)

> path <- readline("Enter the file path: ")
Enter the file path: D:/grc.csv
```



```
6  pie(
7      x = table(dataB$paymentType),
8      main = "Compare cash and credit totals")
9      #table(dataB$paymentType)
10
```

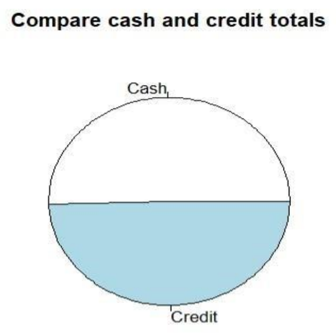
Problem is to Compare cash and credit totals…solved by

This code is to Compare cash and credit totals in pie chart.

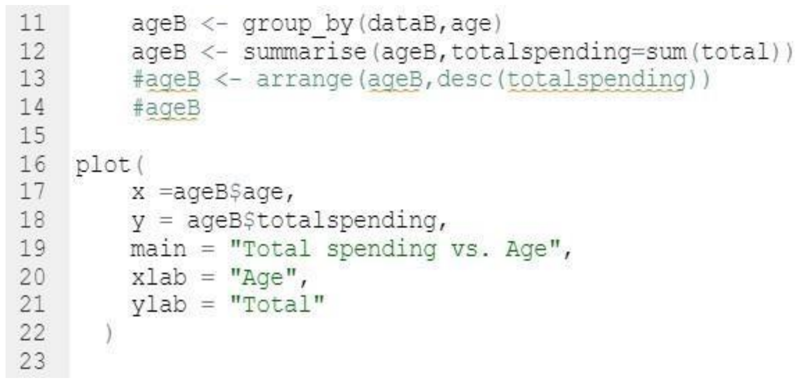
Line 7 to collect the numbers of both cash and credit and put them in a table.

Line 8 to create name for pie chart.

<u>Output for this code:</u>



by this pie chart we realize that the number of Cash > the number of Credit.



```
11     ageB <- group_by(dataB,age)
12     ageB <- summarise(ageB,totalspending=sum(total))
13     #ageB <- arrange(ageB,desc(totalspending))
14     #ageB
15
16 plot(
17     x =ageB$age,
18     y = ageB$totalspending,
19     main = "Total spending vs. Age",
20     xlab = "Age",
21     ylab = "Total"
22   )
23
```

Problem is Compare each age and sum of total spending…. solved by

This code to Compare each age and sum of total spending in Scatterplot.

*Line 11 To sum the similar ages as one element and put them in a variable. (filter dataset)

*Line 12 To Collect each age and sum of total spending. (Note Line 13&14 are comments)

*Line 16 is the start to draw Scatterplot.

*Line 17 to create (x . axis = collected age).

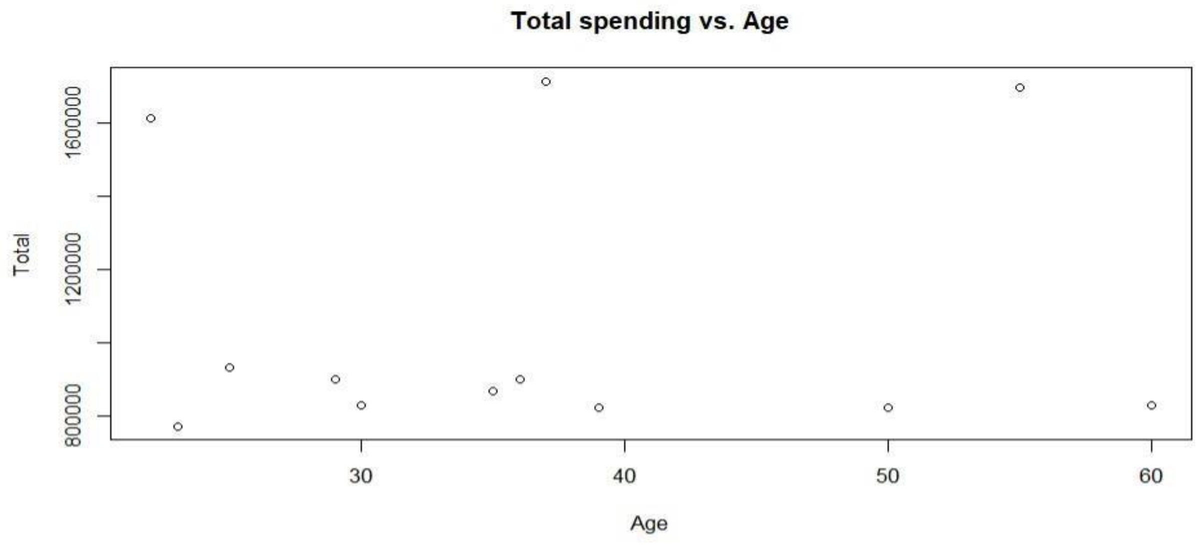
*Line 18 to create (y. axis = totalspending).

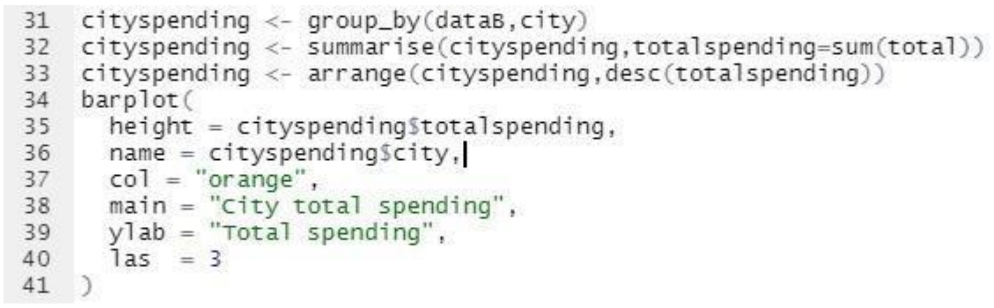
*Line 19 to create name for Scatterplot.

*Line 20 to create name for x-axis.

*Line 21 to create name for y-axis.

<u>Output for this code:</u>





```
31 cityspending <- group_by(dataB,city)
32 cityspending <- summarise(cityspending,totalspending=sum(total))
33 cityspending <- arrange(cityspending,desc(totalspending))
34 barplot(
35   height = cityspending$totalspending,
36   name = cityspending$city,
37   col = "orange",
38   main = "City total spending",
39   ylab = "Total spending",
40   las  = 3
41 )
```

Problem is show each city total spending and arrange it by total descending ….solved by

The code is Compare each age and sum of total spending in barplot

*Line 31 To make the similar cities as one element and put them in a variable.(filter dataset) *Line

32 To Collect each city spending and sum of total spending.

*line 33 to arrange the total spending by descending

*line 34 is the start to draw barplot

*line 35 to creat the height of barplot which is total spending

*line 36 to creat names which is the names of cities

*line 37 to creat color for the barplot *line

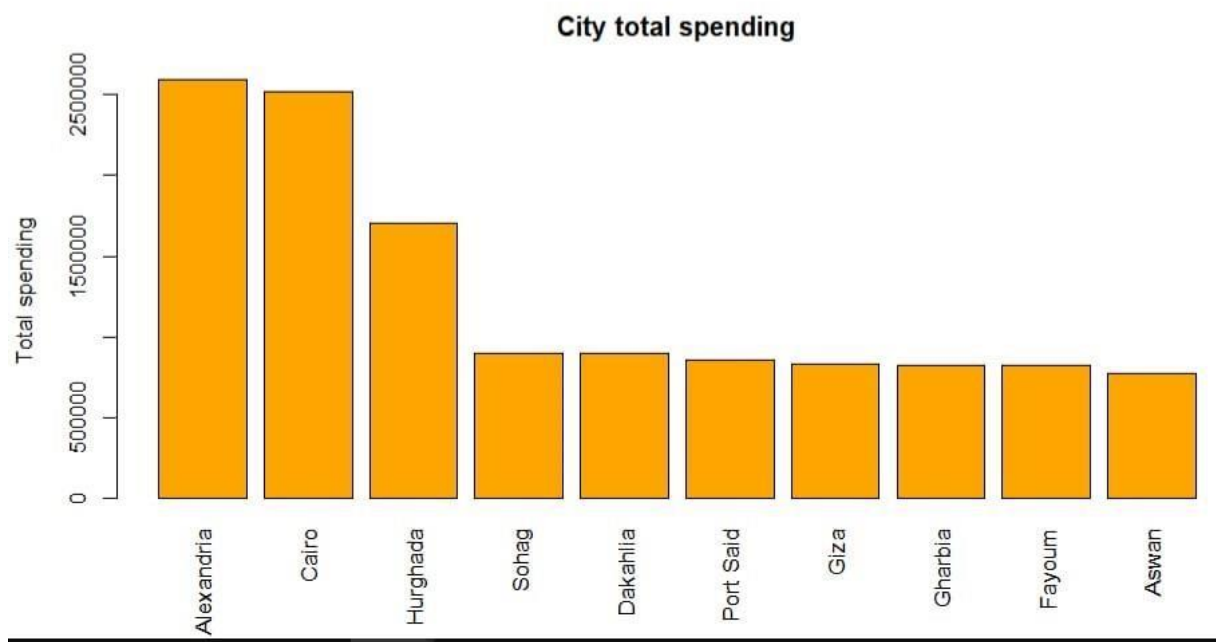
38 to creat the main of this barplot

*line 39 to creat label for y.axis.

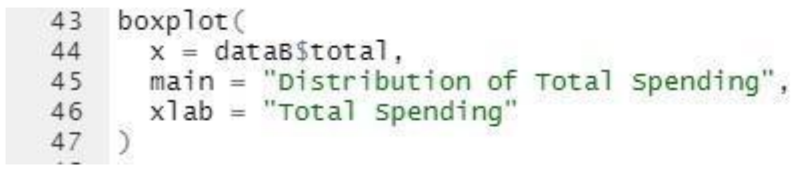
*line 40 to make space between label and total spending

*line 41 to close the barplot

<u>Output for this code :</u>



By this barplot we realize that the total spending is maximum in Alexandria and minimum in Aswan



```
43 boxplot(
44   x = dataB$total,
45   main = "Distribution of Total Spending",
46   xlab = "Total Spending"
47 )
```

Problem is display the distribution of total spending…..solved by

The code is display the distribution of total spending in boxplot

*line 43 is start to draw boxplot

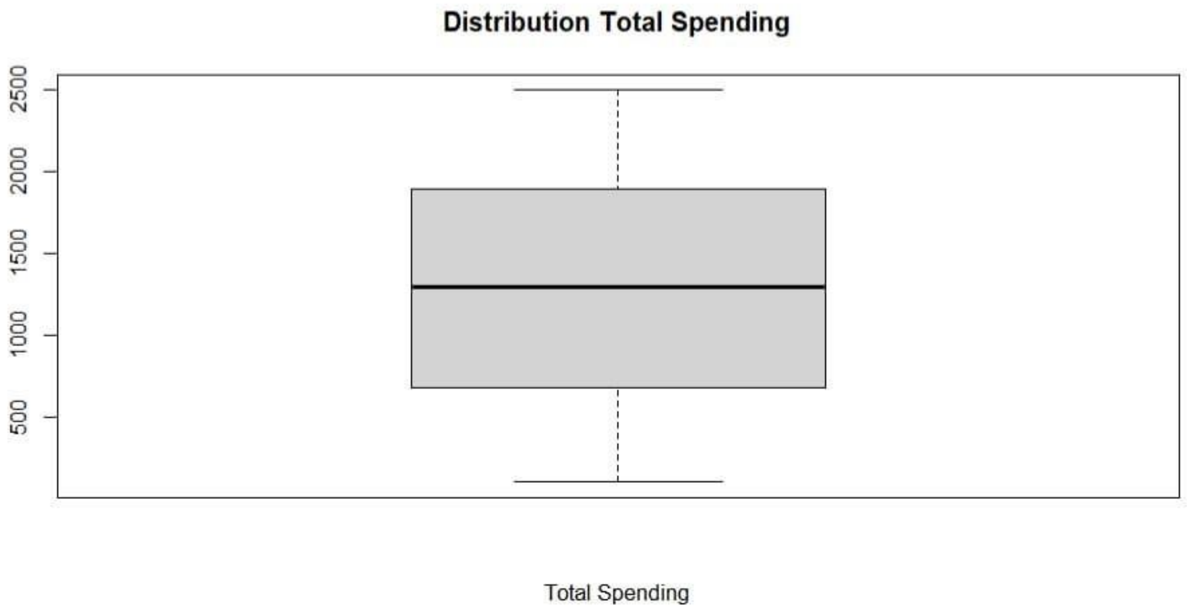
*line 44 to creat a variable which is contain the total of dataB

*line 45 to creat the main of this boxplot

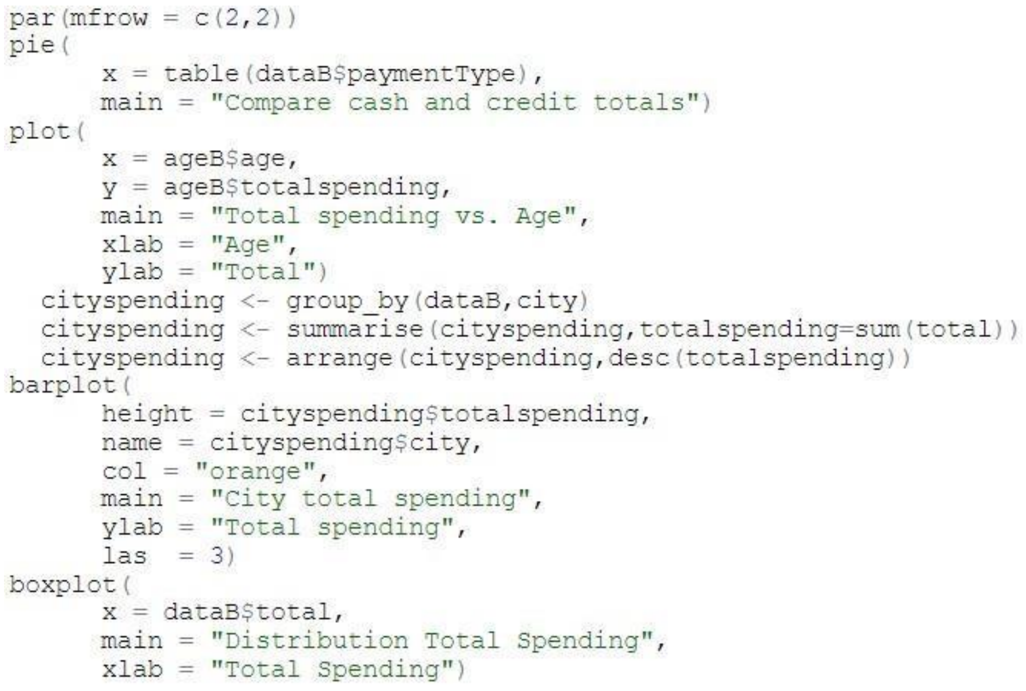
*line 46 to creat label for x.axis

*line 41 to close the boxplot

<u>Output for this code :</u>

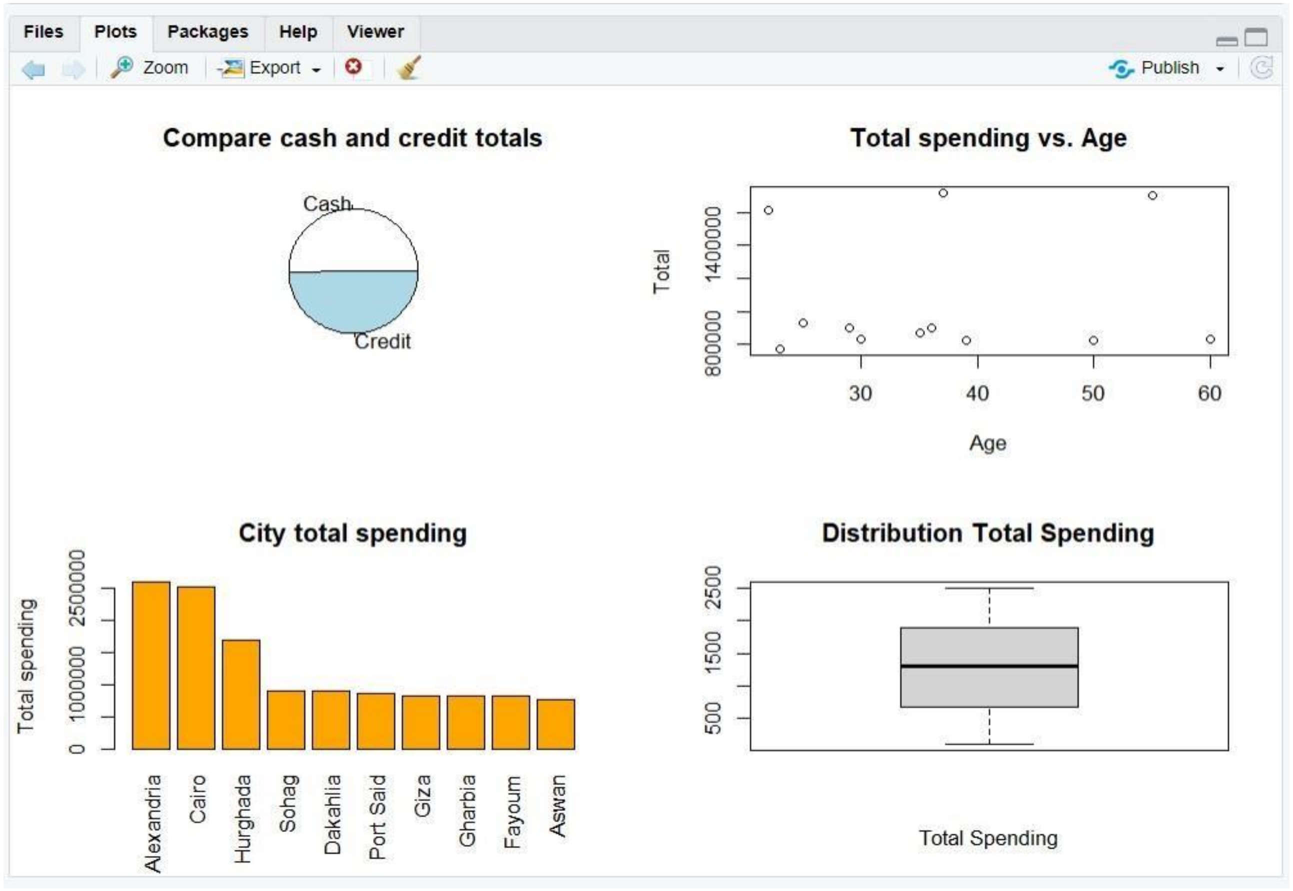


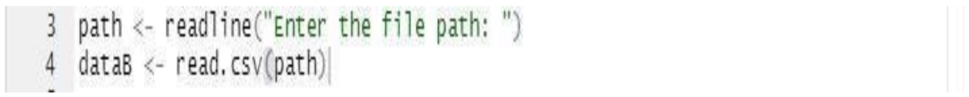
Problem is to Put all previous plots in one dashboard….solved by



```
par(mfrow = c(2,2))
pie(
      x = table(dataB$paymentType),
      main = "Compare cash and credit totals")
plot(
      x = ageB$age,
      y = ageB$totalspending,
      main = "Total spending vs. Age",
      xlab = "Age",
      ylab = "Total")
  cityspending <- group_by(dataB,city)
  cityspending <- summarise(cityspending,totalspending=sum(total))
  cityspending <- arrange(cityspending,desc(totalspending))
barplot(
      height = cityspending$totalspending,
      name = cityspending$city,
      col = "orange",
      main = "City total spending",
      ylab = "Total spending",
      las  = 3)
boxplot(
      x = dataB$total,
      main = "Distribution Total Spending",
      xlab = "Total Spending")
```

This code Put all previous plots in one dashboard.





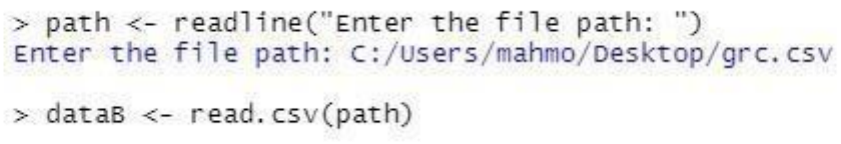
```
3 path <- readline("Enter the file path: ")
4 dataB <- read.csv(path)
```

Problem is to take dataset path from user…solved by

Line 3: To take dataset path from user.

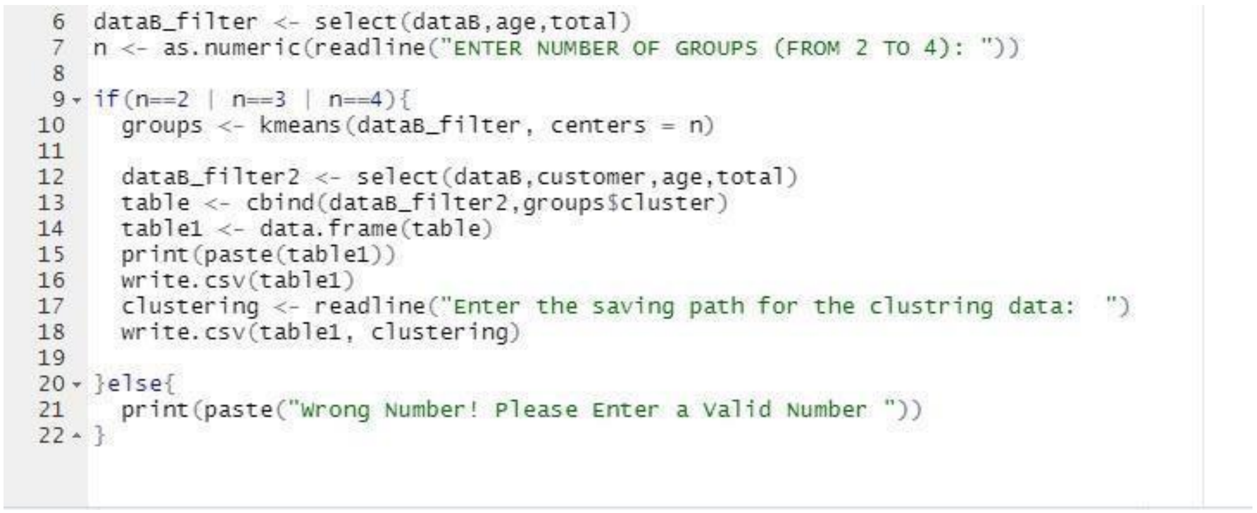
Line 4: To read this file in csv file.

<u>Output for this code :</u>



```
> path <- readline("Enter the file path: ")
Enter the file path: C:/Users/mahmo/Desktop/grc.csv

> dataB <- read.csv(path)
```



```
 6  dataB_filter <- select(dataB,age,total)
 7  n <- as.numeric(readline("ENTER NUMBER OF GROUPS (FROM 2 TO 4): "))
 8
 9  if(n==2 | n==3 | n==4){
10    groups <- kmeans(dataB_filter, centers = n)
11
12    dataB_filter2 <- select(dataB,customer,age,total)
13    table <- cbind(dataB_filter2,groups$cluster)
14    table1 <- data.frame(table)
15    print(paste(table1))
16    write.csv(table1)
17    clustering <- readline("Enter the saving path for the clustring data:  ")
18    write.csv(table1, clustering)
19
20  }else{
21    print(paste("Wrong Number! Please Enter a Valid Number "))
22  }
```

The problem in question (d) : is to Split the customers to (n) groups using one of the studied methods

As number of groups or clusters (n) should be input from the user and (n) should be an integer number from 2 to 4 also the clustering should be on the age and the total spending only. then we should print a table of the (customer) column, (age) column,(total spending) column and finally (cluster) column.

Solved by:

*Line 6: we used (select) function from (dplyr) library to select the (age) & (total) columns only from the csv file to use our method.

*Line 7: we made the variable (n) which represents number of groups and set in it the (readline) function to take the number of groups or clusters from the user input as well as using (as.numeric) function to take this input in numeric for not character form.

*Line 9: we used (if,else statement) with condition that (n) taken from user input should be equals to (2

or 3 or 4) only if that condition becomes (TRUE) then we continue by our folloeing lines.

*Line 10: we used (kmeans) function to split customers into clusters according to only their (age & total spending) columns in csv file.

*Line 12: we used (select) function again to select the (customer) column in csv file besides (age & total spending) columns to prepare for printing our table includes (customer) column too.

*Line 13: we made a variable called (table) carrying (cbind) function to combine our previous variable containing (customer,age,total) columns with (cluster) column which we obtained by using (group$cluster) (group) is the variable contains the (kmeans) function and by using (group$cluster) we obtained the (clusters) in a vector.

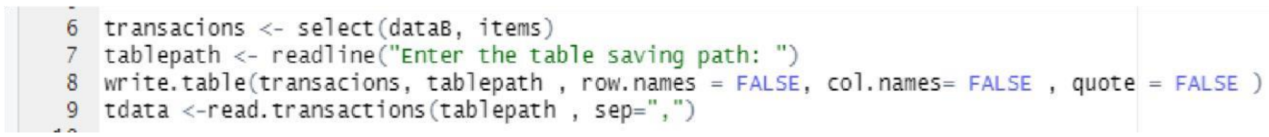
*Line 14: we made a data frame called (table1) using (data.frame) function to put our table as data frame to be easy to handle.

*Line 15 & *Line 16 : we used (print) function to print our table & (write .csv) function to print our table but in more arranged style as data frame should be printed.

*Line 17 & *Line 18: we made a variable contains (readline) function asks the user for the path that we should save the (new csv file with cluster column) in and then using (write.csv) function we made our previously printed table becomes in (csv) extension and save it in the path the user gave us.

*Line 20 & *Line 21: is the complete of the (if,else statement) by using (else) that the user entered any number not equal to (2 or 3 or 4) we show him that it is a wrong number so he can substitute it.

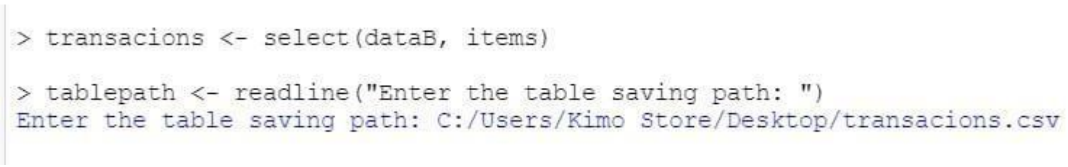
Line 6 is used to put the items only in a dataframe.
Line 7 is used to take the path to save the items table for the apriori algorithm.
Line 8 is to save the items in a table and row.names = false and col.names = false and quote = false is to make sure that the table is clean from any unwanted columns, rows and empty cells. Line



```
6  transacions <- select(dataB, items)
7  tablepath <- readline("Enter the table saving path: ")
8  write.table(transacions, tablepath , row.names = FALSE, col.names= FALSE , quote = FALSE )
9  tdata <-read.transactions(tablepath , sep=",")
```

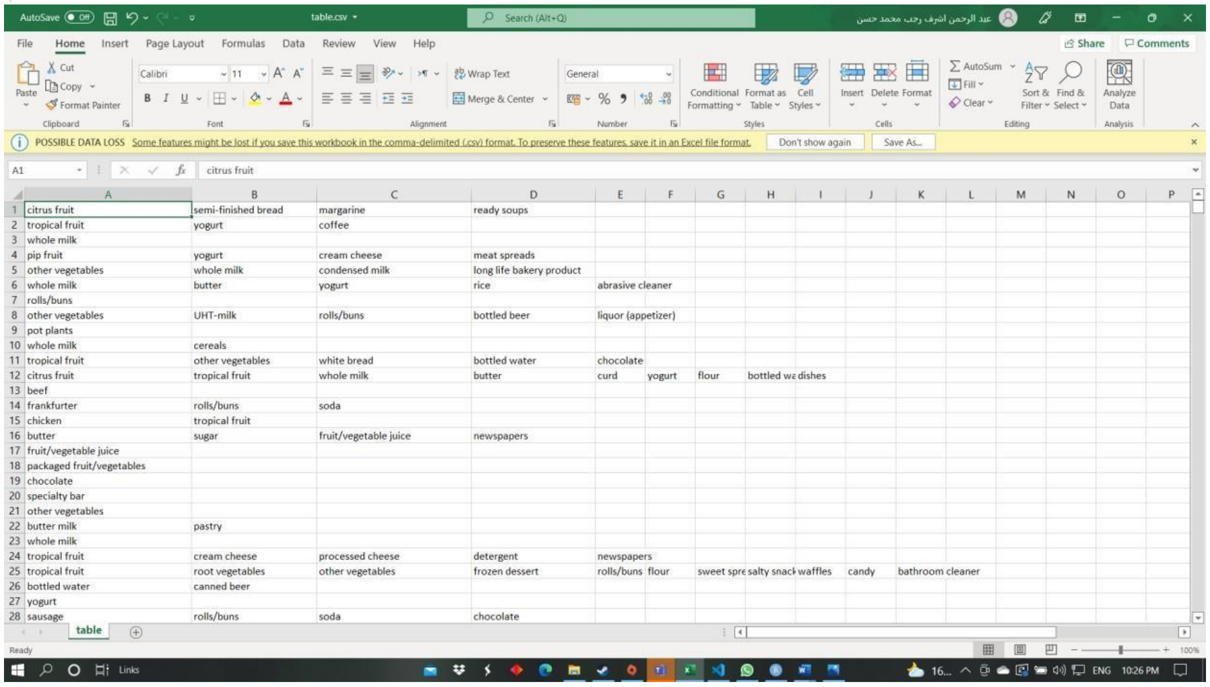
9 is used to read the table that we saved in the previous step as a transaction-type variable with "," separator to use in the apriori algorithm.

<u>The output of this code is</u>:-

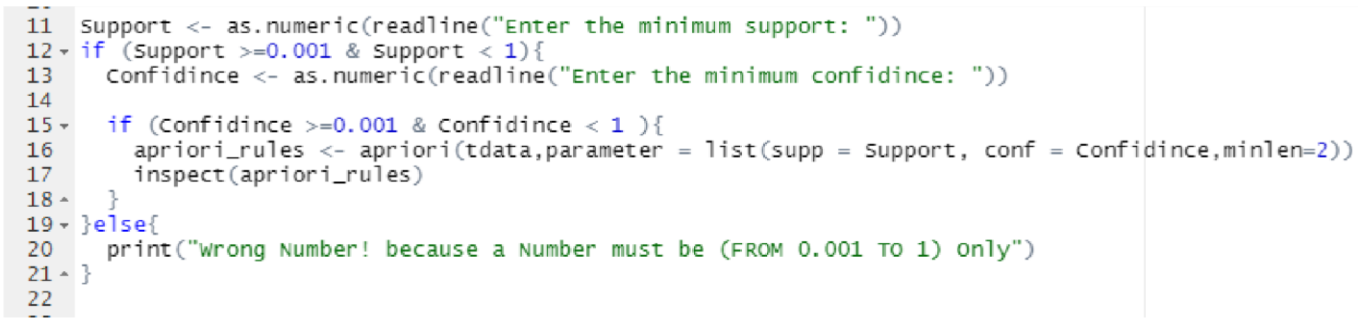


```
> transacions <- select(dataB, items)

> tablepath <- readline("Enter the table saving path: ")
Enter the table saving path: C:/Users/Kimo Store/Desktop/transacions.csv
```



	A	B	C	D	E	F	G	H	I	J	K
1	citrus fruit	semi-finished bread	margarine	ready soups							
2	tropical fruit	yogurt	coffee								
3	whole milk										
4	pip fruit	yogurt	cream cheese	meat spreads							
5	other vegetables	whole milk	condensed milk	long life bakery product							
6	whole milk	butter	yogurt	rice	abrasive cleaner						
7	rolls/buns										
8	other vegetables	UHT-milk	rolls/buns	bottled beer	liquor (appetizer)						
9	pot plants										
10	whole milk	cereals									
11	tropical fruit	other vegetables	white bread	bottled water	chocolate						
12	citrus fruit	tropical fruit	whole milk	butter	curd	yogurt	flour	bottled wa	dishes		
13	beef										
14	frankfurter	rolls/buns	soda								
15	chicken	tropical fruit									
16	butter	sugar	fruit/vegetable juice	newspapers							
17	fruit/vegetable juice										
18	packaged fruit/vegetables										
19	chocolate										
20	specialty bar										
21	other vegetables										
22	butter milk	pastry									
23	whole milk										
24	tropical fruit	cream cheese	processed cheese	detergent	newspapers						
25	tropical fruit	root vegetables	other vegetables	frozen dessert	rolls/buns	flour	sweet spre	salty snack	waffles	candy	bathroom cleaner
26	bottled water	canned beer									
27	yogurt										
28	sausage	rolls/buns	soda	chocolate							



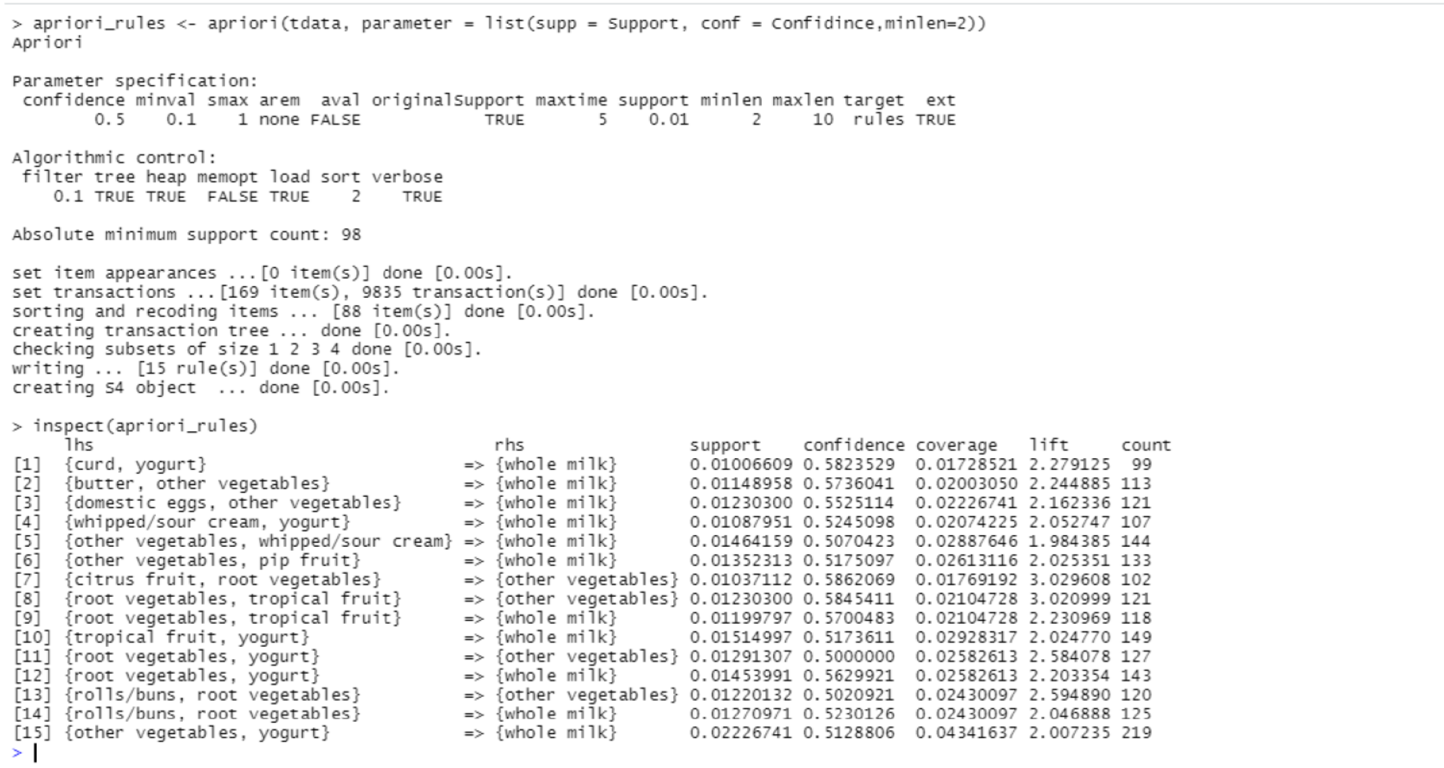
```
11  Support <- as.numeric(readline("Enter the minimum support: "))
12  if (Support >=0.001 & Support < 1){
13    Confidince <- as.numeric(readline("Enter the minimum confidince: "))
14
15    if (Confidince >=0.001 & Confidince < 1 ){
16      apriori_rules <- apriori(tdata,parameter = list(supp = Support, conf = Confidince,minlen=2))
17      inspect(apriori_rules)
18    }
19  }else{
20    print("Wrong Number! because a Number must be (FROM 0.001 TO 1) Only")
21  }
22
```

Line 11 is used to take the minimum support from the user.
Line 12 is used to make sure that the minimum support Is between 0.001 and 1.

Line 13 is used to take the minimum confidence from the user.
line 15 is used to make sure that the confidence is between 0.001 and 1.
line 16 is the apriori algorithm with is used to generate the association rules between the items from the transactions list with the minimum length of 2. line 17 is used to print the result from the algorithm to the user on the console.
line 19 and 20 to send the user a message to put right values to the minimum support and confidence.

<u>The output of this code is</u>:- (support=0.01 & confidence=0.5)

```
[1]  {curd, yogurt}                  => {whole milk}      0.01006609 0.5823529  0.01728521 2.279125  99
[2]  {butter, other vegetables}           => {whole milk}      0.01148958 0.5736041  0.02003050 2.244885
     113
[3]  {domestic eggs, other vegetables}     => {whole milk}      0.01230300 0.5525114  0.02226741
     2.162336 121
[4]  {whipped/sour cream, yogurt}         => {whole milk}      0.01087951 0.5245098  0.02074225
     2.052747 107
[5]  {other vegetables, whipped/sour cream} => {whole milk}      0.01464159 0.5070423  0.02887646
     1.984385 144
[6]  {other vegetables, pip fruit}        => {whole milk}      0.01352313 0.5175097  0.02613116 2.025351
     133
[7]  {citrus fruit, root vegetables}       => {other vegetables} 0.01037112 0.5862069  0.01769192
     3.029608 102
[8]  {root vegetables, tropical fruit}     => {other vegetables} 0.01230300 0.5845411  0.02104728
     3.020999 121
[9]  {root vegetables, tropical fruit}     => {whole milk}      0.01199797 0.5700483  0.02104728 2.230969
     118
[10] {tropical fruit, yogurt}            => {whole milk}      0.01514997 0.5173611  0.02928317 2.024770 149
[11] {root vegetables, yogurt}           => {other vegetables} 0.01291307 0.5000000  0.02582613
     2.584078 127
[12] {root vegetables, yogurt}           => {whole milk}      0.01453991 0.5629921  0.02582613 2.203354
     143
[13] {rolls/buns, root vegetables}        => {other vegetables} 0.01220132 0.5020921  0.02430097
     2.594890 120
[14] {rolls/buns, root vegetables}        => {whole milk}      0.01270971 0.5230126  0.02430097 2.046888
     125
[15] {other vegetables, yogurt}          => {whole milk}      0.02226741 0.5128806  0.04341637 2.007235
     219
```



```
> apriori_rules <- apriori(tdata, parameter = list(supp = Support, conf = Confidince,minlen=2))
Apriori

Parameter specification:
 confidence minval smax arem  aval originalSupport maxtime support minlen maxlen target  ext
        0.5    0.1    1 none FALSE            TRUE       5    0.01      2     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), 9835 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 ... [15 rule(s)] done [0.00s].
creating S4 object  ... done [0.00s].

> inspect(apriori_rules)
     lhs                                     rhs                support    confidence coverage   lift     count
[1]  {curd, yogurt}                       => {whole milk}       0.01006609 0.5823529  0.01728521 2.279125  99
[2]  {butter, other vegetables}           => {whole milk}       0.01148958 0.5736041  0.02003050 2.244885 113
[3]  {domestic eggs, other vegetables}    => {whole milk}       0.01230300 0.5525114  0.02226741 2.162336 121
[4]  {whipped/sour cream, yogurt}         => {whole milk}       0.01087951 0.5245098  0.02074225 2.052747 107
[5]  {other vegetables, whipped/sour cream} => {whole milk}     0.01464159 0.5070423  0.02887646 1.984385 144
[6]  {other vegetables, pip fruit}        => {whole milk}       0.01352313 0.5175097  0.02613116 2.025351 133
[7]  {citrus fruit, root vegetables}      => {other vegetables} 0.01037112 0.5862069  0.01769192 3.029608 102
[8]  {root vegetables, tropical fruit}    => {other vegetables} 0.01230300 0.5845411  0.02104728 3.020999 121
[9]  {root vegetables, tropical fruit}    => {whole milk}       0.01199797 0.5700483  0.02104728 2.230969 118
[10] {tropical fruit, yogurt}             => {whole milk}       0.01514997 0.5173611  0.02928317 2.024770 149
[11] {root vegetables, yogurt}            => {other vegetables} 0.01291307 0.5000000  0.02582613 2.584078 127
[12] {root vegetables, yogurt}            => {whole milk}       0.01453991 0.5629921  0.02582613 2.203354 143
[13] {rolls/buns, root vegetables}        => {other vegetables} 0.01220132 0.5020921  0.02430097 2.594890 120
[14] {rolls/buns, root vegetables}        => {whole milk}       0.01270971 0.5230126  0.02430097 2.046888 125
[15] {other vegetables, yogurt}           => {whole milk}       0.02226741 0.5128806  0.04341637 2.007235 219
> 
```