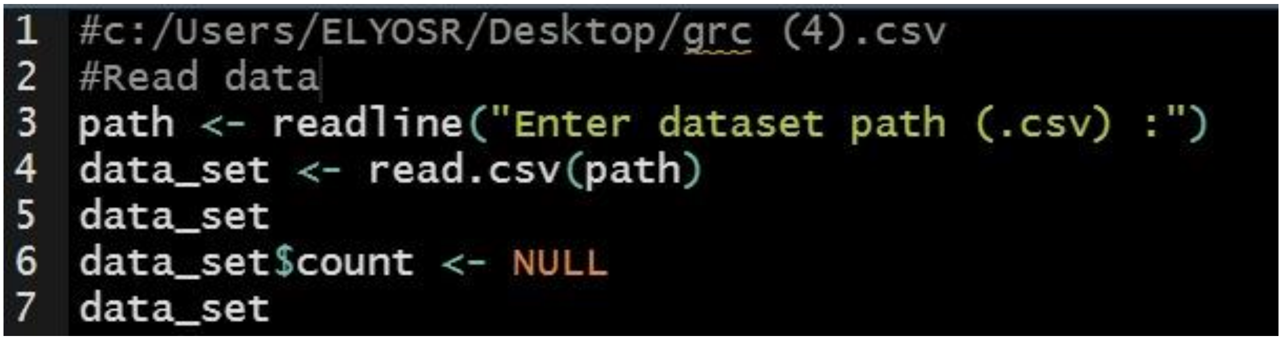
problem description:

a. Our program will read the user data set , and it display the given Grocery (GRC) dataset using different type of Data Visualization tools , it present cluster analysis for the data set using the K-mean method Association rules which discover relationships in given transaction dataset for market basket analysis .

b. the input: the data set, number of clusters in K-mean, transaction data, minimum support, and the minimum confidence for Association rules.

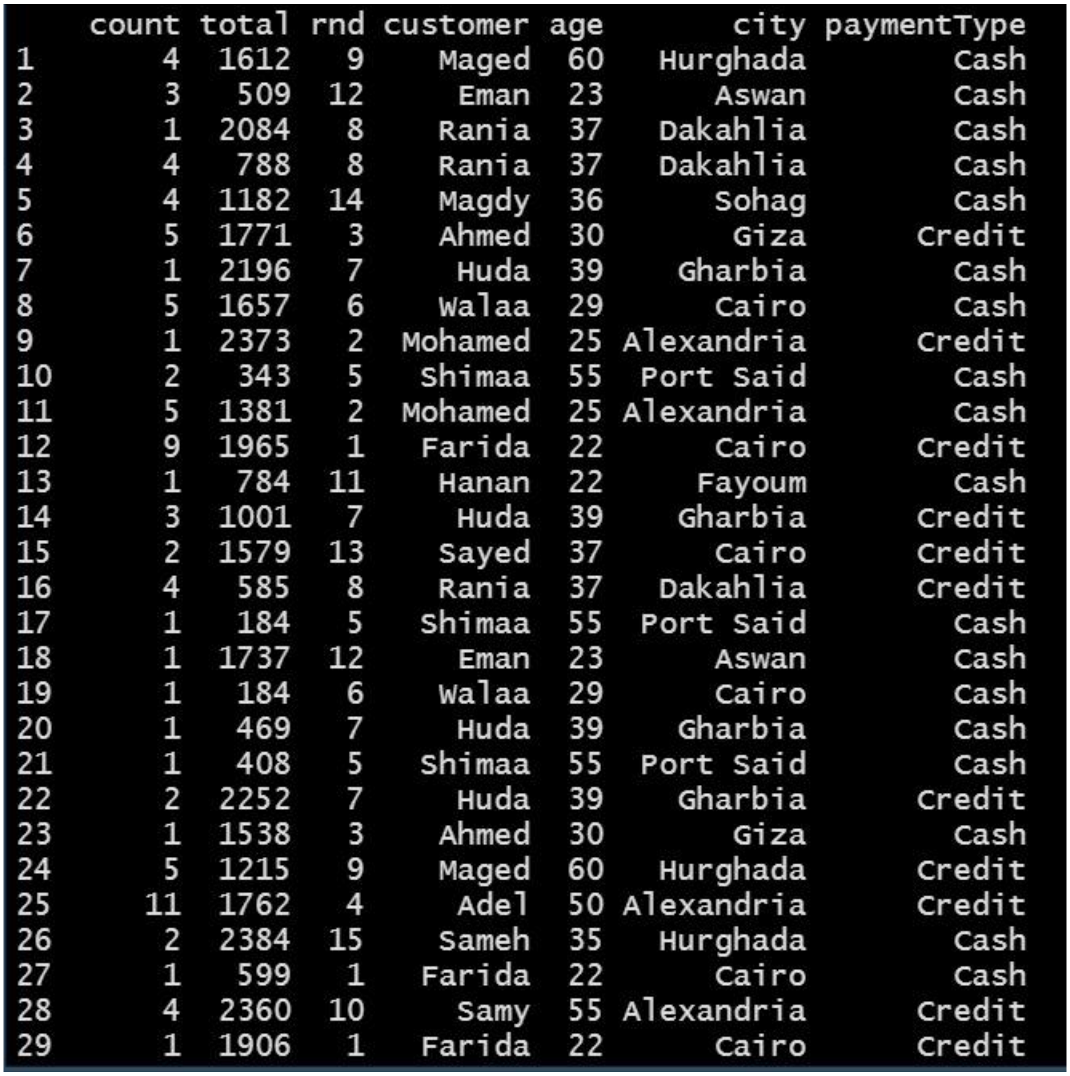
c. the output: the displayed data, k-mean final cluster centers, and The final cluster assignments of the data set, Final Association Rules.

1-Read and clean the data set :



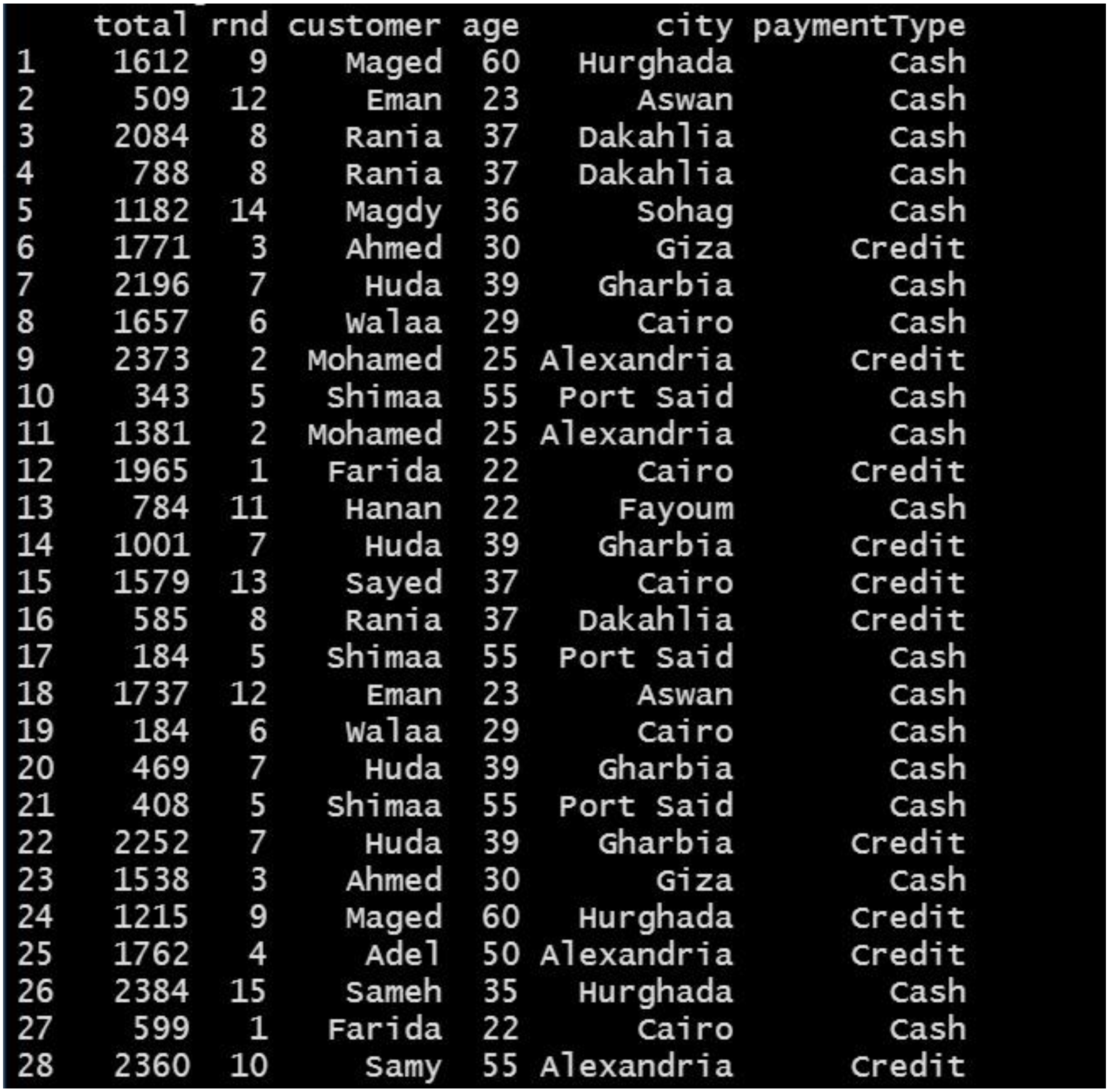
```
#c:/Users/ELYOSR/Desktop/grc (4).csv
#Read data
path <- readline("Enter dataset path (.csv) :")
data_set <- read.csv(path)
data_set
data_set$count <- NULL
data_set
```

Data set out put :



```
   count total rnd customer age       city paymentType
1      4  1612   9    Maged  60   Hurghada        Cash
2      3   509  12     Eman  23      Aswan        Cash
3      1  2084   8    Rania  37   Dakahlia        Cash
4      4   788   8    Rania  37   Dakahlia        Cash
5      4  1182  14    Magdy  36      Sohag        Cash
6      5  1771   3    Ahmed  30       Giza      Credit
7      1  2196   7     Huda  39    Gharbia        Cash
8      5  1657   6    Walaa  29      Cairo        Cash
9      1  2373   2  Mohamed  25 Alexandria      Credit
10     2   343   5   Shimaa  55  Port Said        Cash
11     5  1381   2  Mohamed  25 Alexandria        Cash
12     9  1965   1   Farida  22      Cairo      Credit
13     1   784  11    Hanan  22     Fayoum        Cash
14     3  1001   7     Huda  39    Gharbia      Credit
15     2  1579  13    Sayed  37      Cairo      Credit
16     4   585   8    Rania  37   Dakahlia      Credit
17     1   184   5   Shimaa  55  Port Said        Cash
18     1  1737  12     Eman  23      Aswan        Cash
19     1   184   6    Walaa  29      Cairo        Cash
20     1   469   7     Huda  39    Gharbia        Cash
21     1   408   5   Shimaa  55  Port Said        Cash
22     2  2252   7     Huda  39    Gharbia      Credit
23     1  1538   3    Ahmed  30       Giza        Cash
24     5  1215   9    Maged  60   Hurghada      Credit
25    11  1762   4     Adel  50 Alexandria      Credit
26     2  2384  15    Sameh  35   Hurghada        Cash
27     1   599   1   Farida  22      Cairo        Cash
28     4  2360  10     Samy  55 Alexandria      Credit
29     1  1906   1   Farida  22      Cairo      Credit
```

Cleaned data out put :



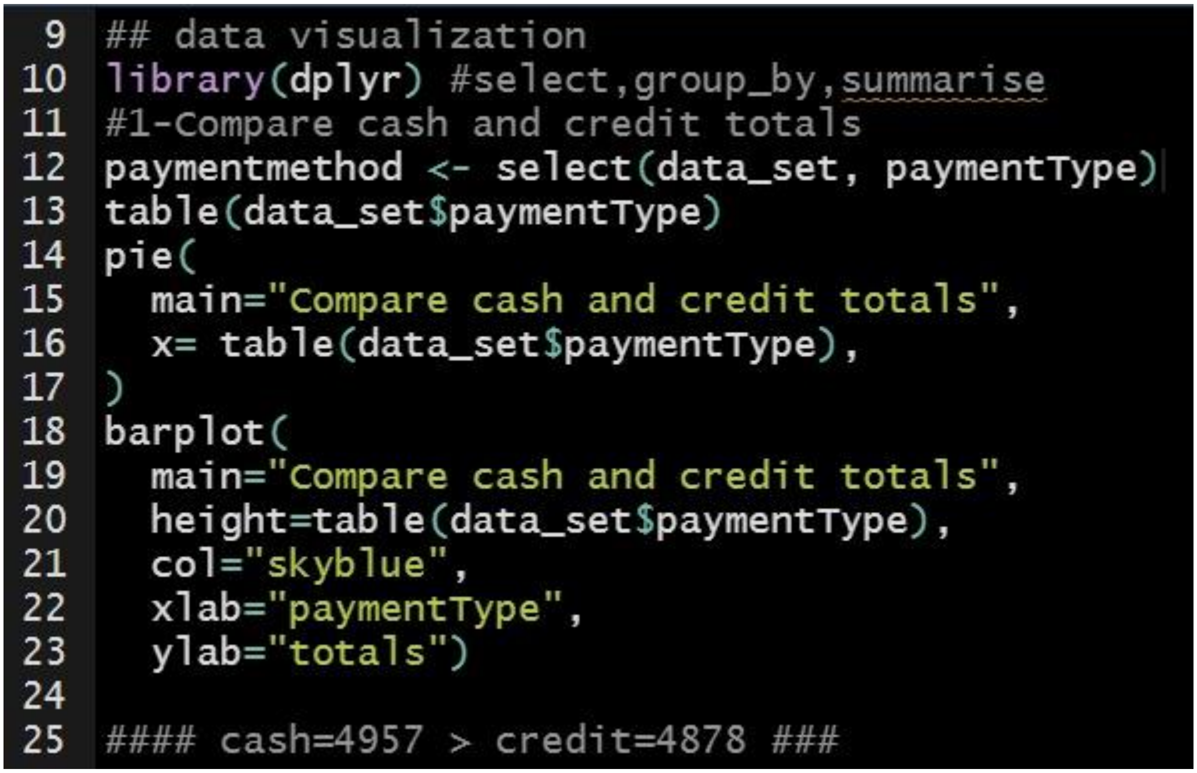
```
    total rnd customer age       city paymentType
1    1612   9    Maged  60   Hurghada        Cash
2     509  12     Eman  23      Aswan        Cash
3    2084   8    Rania  37   Dakahlia        Cash
4     788   8    Rania  37   Dakahlia        Cash
5    1182  14    Magdy  36      Sohag        Cash
6    1771   3    Ahmed  30       Giza      Credit
7    2196   7     Huda  39    Gharbia        Cash
8    1657   6    Walaa  29      Cairo        Cash
9    2373   2  Mohamed  25 Alexandria      Credit
10    343   5   Shimaa  55  Port Said        Cash
11   1381   2  Mohamed  25 Alexandria        Cash
12   1965   1   Farida  22      Cairo      Credit
13    784  11    Hanan  22     Fayoum        Cash
14   1001   7     Huda  39    Gharbia      Credit
15   1579  13    Sayed  37      Cairo      Credit
16    585   8    Rania  37   Dakahlia      Credit
17    184   5   Shimaa  55  Port Said        Cash
18   1737  12     Eman  23      Aswan        Cash
19    184   6    Walaa  29      Cairo        Cash
20    469   7     Huda  39    Gharbia        Cash
21    408   5   Shimaa  55  Port Said        Cash
22   2252   7     Huda  39    Gharbia      Credit
23   1538   3    Ahmed  30       Giza        Cash
24   1215   9    Maged  60   Hurghada      Credit
25   1762   4     Adel  50 Alexandria      Credit
26   2384  15    Sameh  35   Hurghada        Cash
27    599   1   Farida  22      Cairo        Cash
28   2360  10     Samy  55 Alexandria      Credit
```

observation:

the count is not useful for required tasks.

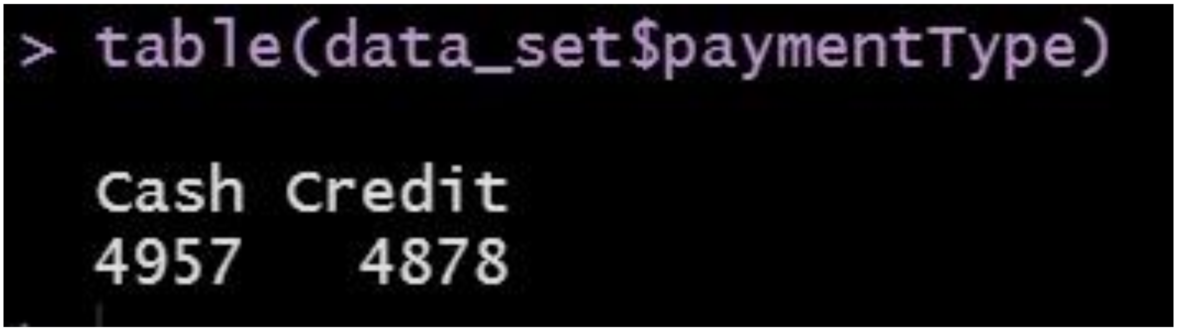
2-data visualization

Compare cash and credit totals:

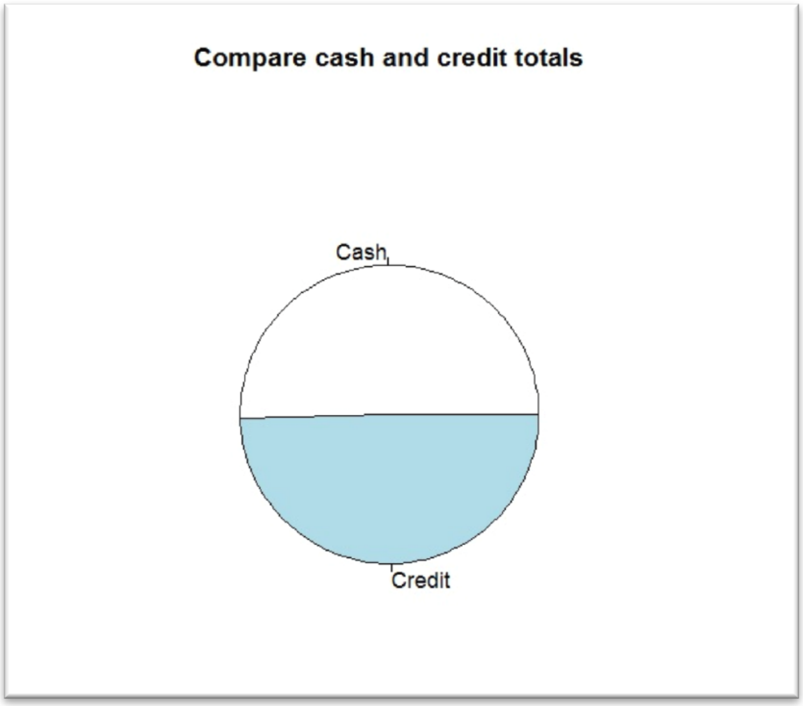


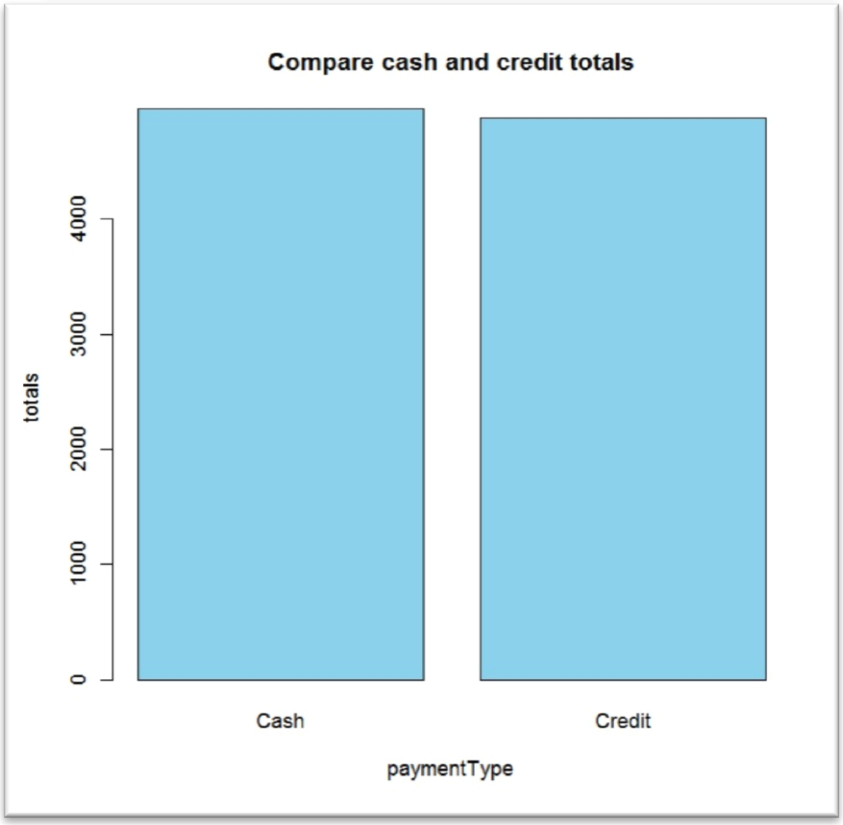
```
## data visualization
library(dplyr) #select,group_by,summarise
#1-Compare cash and credit totals
paymentmethod <- select(data_set, paymentType)
table(data_set$paymentType)
pie(
  main="Compare cash and credit totals",
  x= table(data_set$paymentType),
)
barplot(
  main="Compare cash and credit totals",
  height=table(data_set$paymentType),
  col="skyblue",
  xlab="paymentType",
  ylab="totals")

#### cash=4957 > credit=4878 ###
```

Out Put:



```
> table(data_set$paymentType)

  Cash Credit
  4957   4878
```

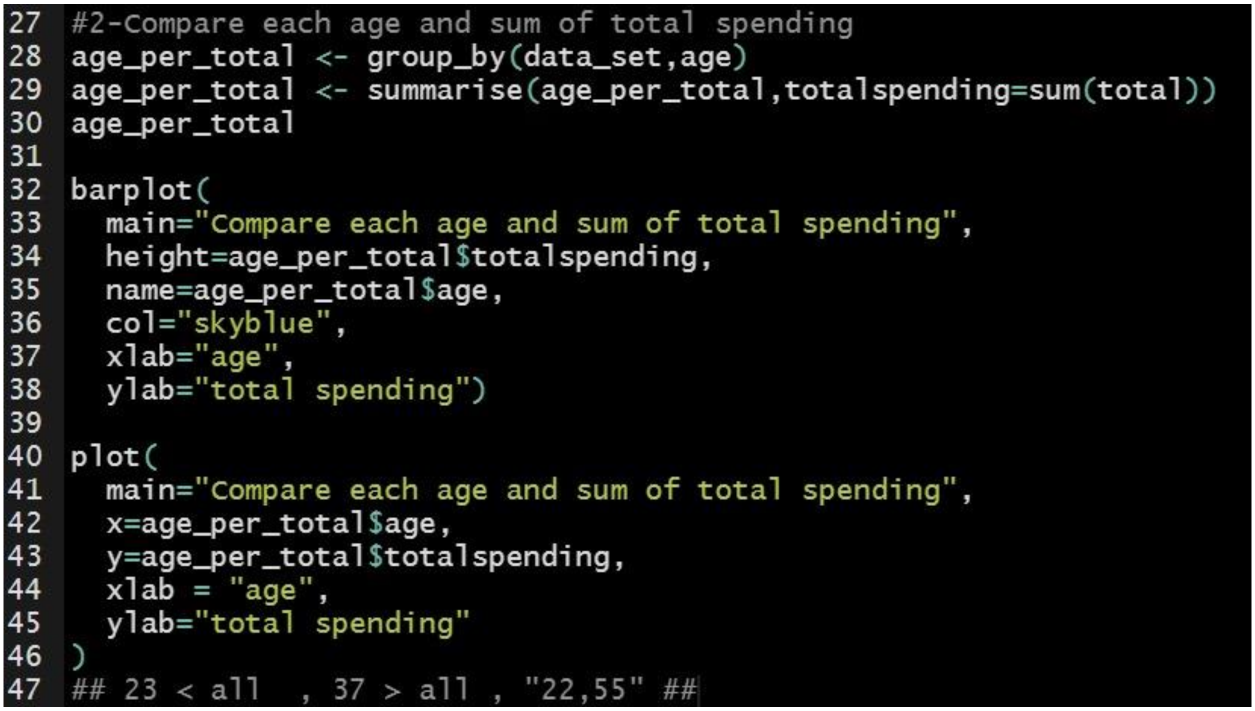




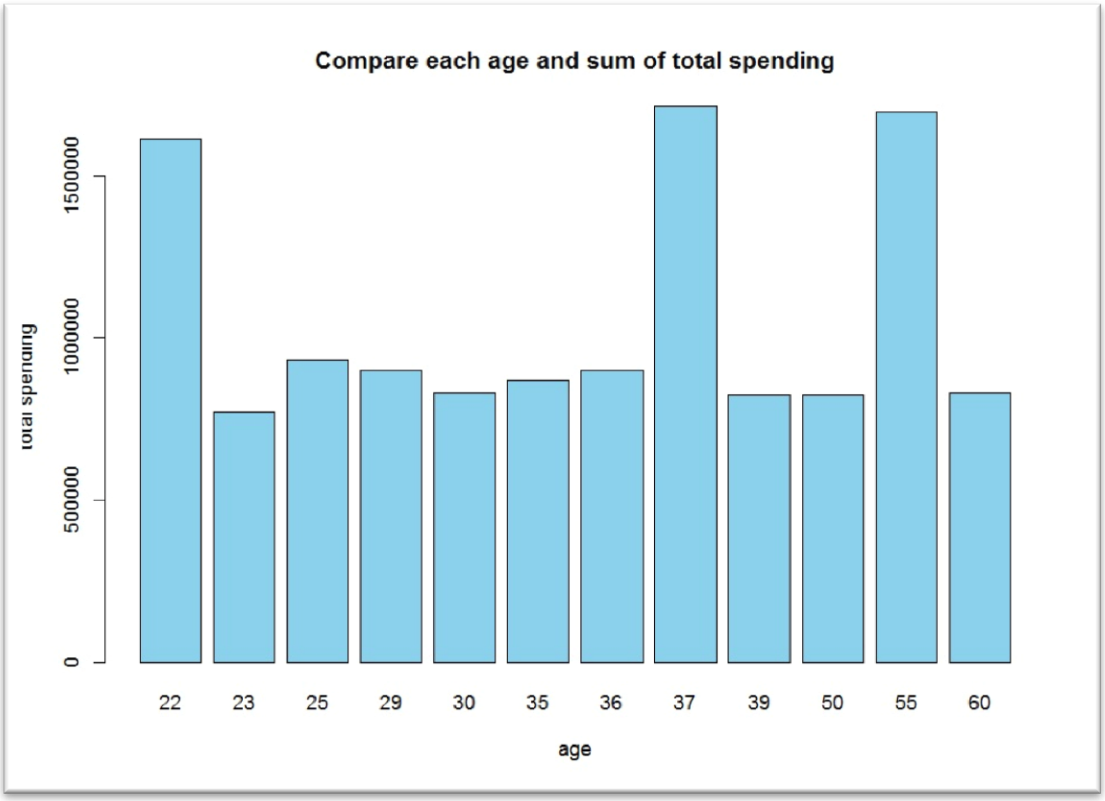
Observation:

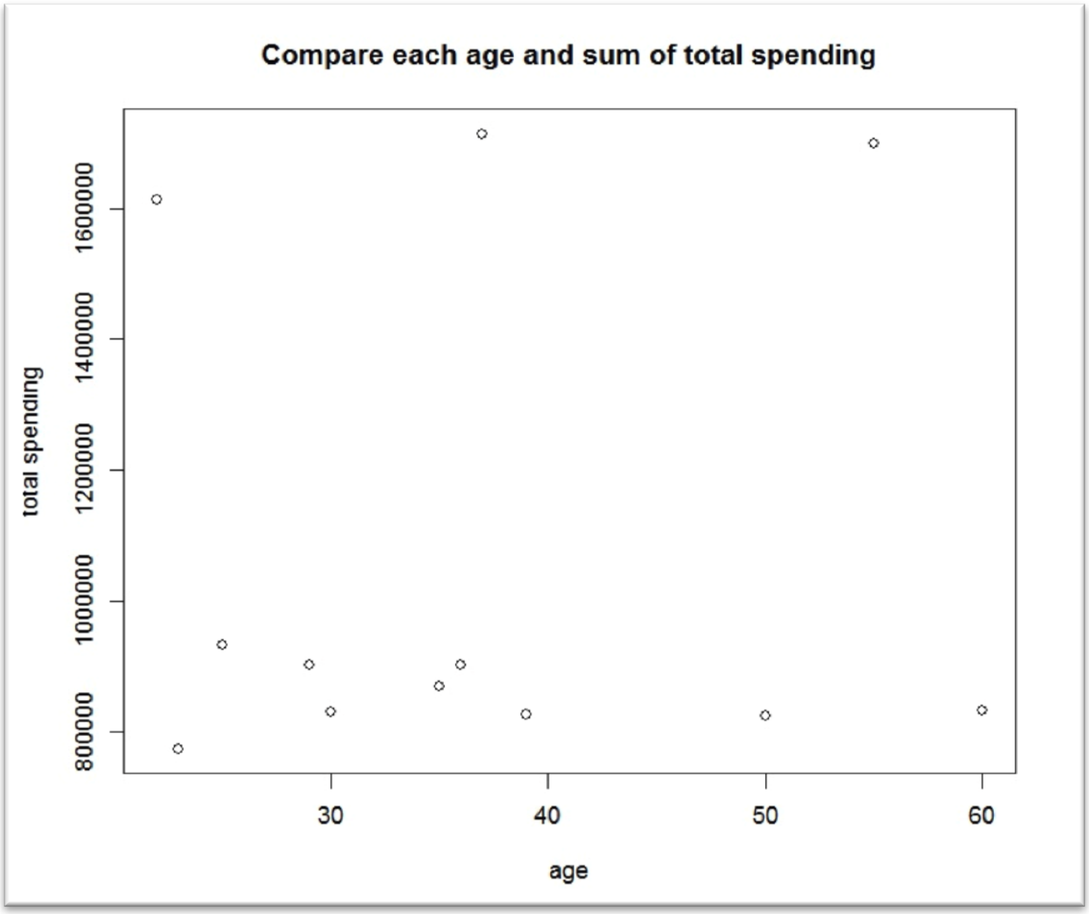
The payment method cash is used more than credit

Compare each age and sum of total spending :



```
#2-Compare each age and sum of total spending
age_per_total <- group_by(data_set,age)
age_per_total <- summarise(age_per_total,totalspending=sum(total))
age_per_total

barplot(
  main="Compare each age and sum of total spending",
  height=age_per_total$totalspending,
  name=age_per_total$age,
  col="skyblue",
  xlab="age",
  ylab="total spending")

plot(
  main="Compare each age and sum of total spending",
  x=age_per_total$age,
  y=age_per_total$totalspending,
  xlab = "age",
  ylab="total spending"
)
## 23 < all  , 37 > all , "22,55" ##
```

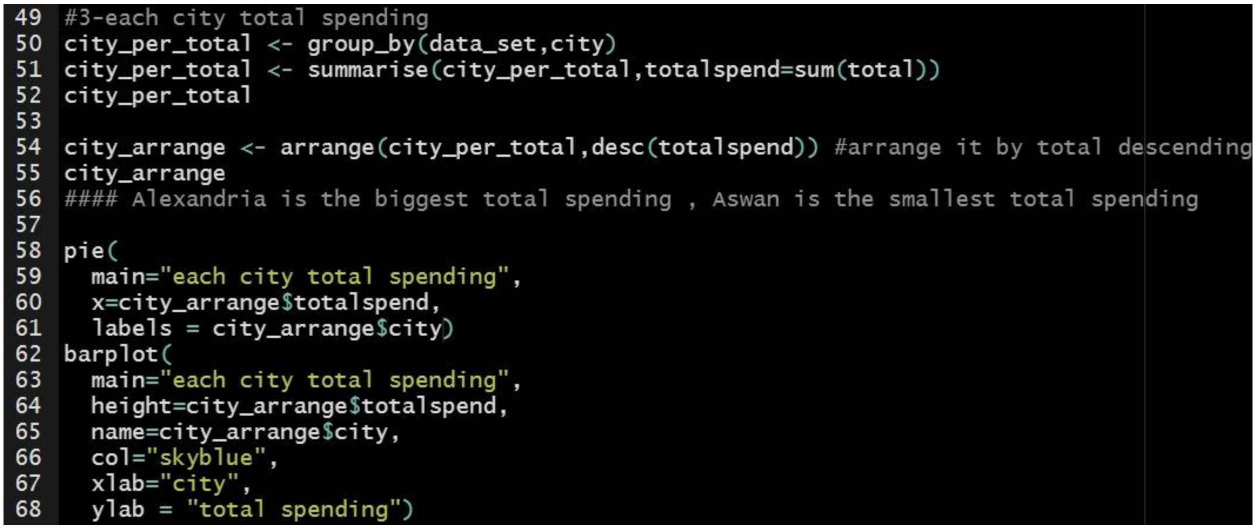




Observations:

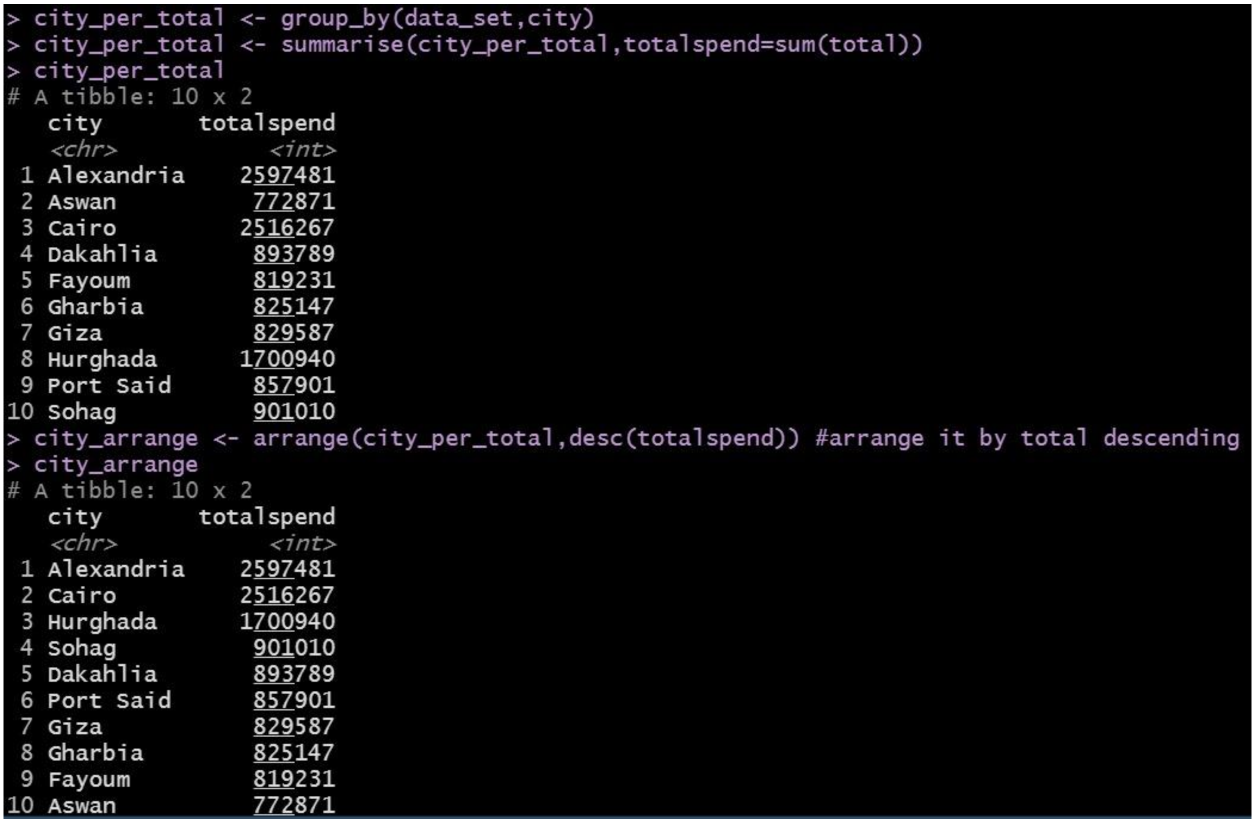
- ages 37 have the highest total spending, the lowest is ages 23.
- There is an inverse relationship between age and total spending.
- ages 22 and 55 are abnormal .

compare each city with total spending:

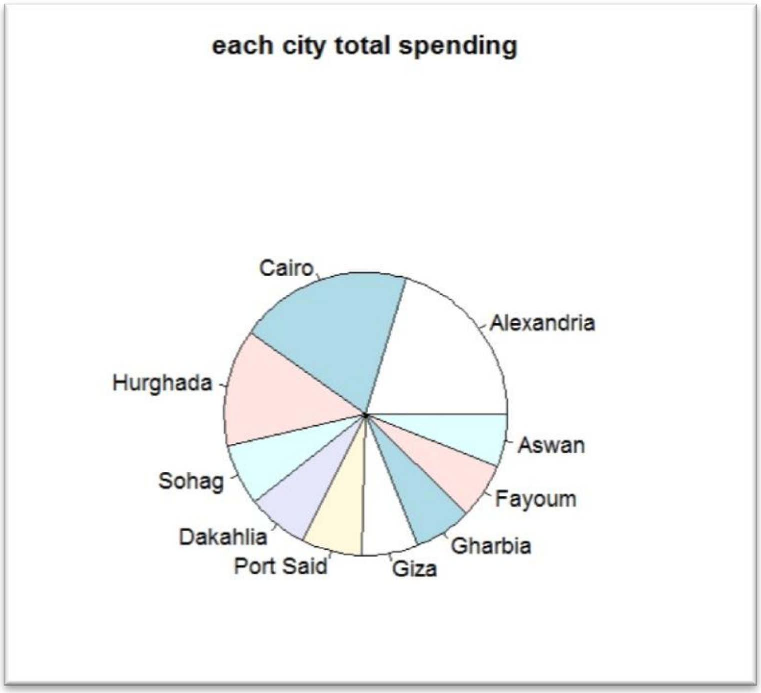


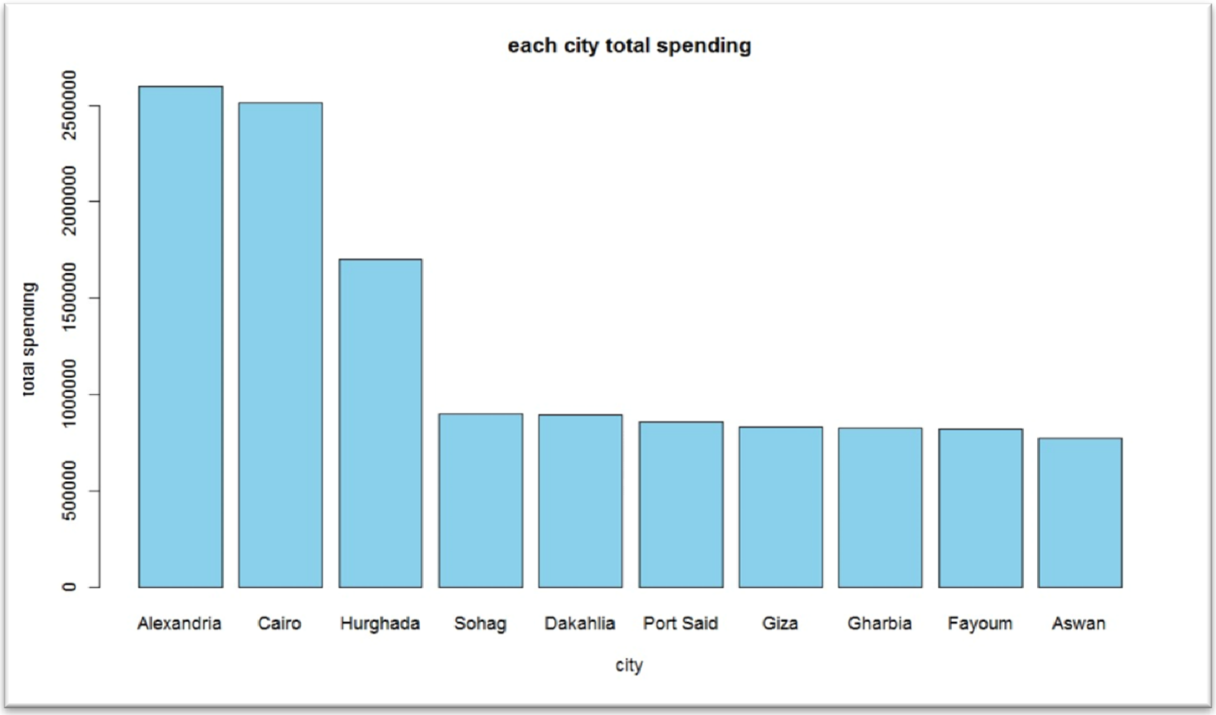
```
#3-each city total spending
city_per_total <- group_by(data_set,city)
city_per_total <- summarise(city_per_total,totalspend=sum(total))
city_per_total

city_arrange <- arrange(city_per_total,desc(totalspend)) #arrange it by total descending
city_arrange
#### Alexandria is the biggest total spending , Aswan is the smallest total spending

pie(
  main="each city total spending",
  x=city_arrange$totalspend,
  labels = city_arrange$city)
barplot(
  main="each city total spending",
  height=city_arrange$totalspend,
  name=city_arrange$city,
  col="skyblue",
  xlab="city",
  ylab = "total spending")
```

out put :



```
> city_per_total <- group_by(data_set,city)
> city_per_total <- summarise(city_per_total,totalspend=sum(total))
> city_per_total
# A tibble: 10 x 2
   city       totalspend
   <chr>           <int>
 1 Alexandria    2597481
 2 Aswan          772871
 3 Cairo         2516267
 4 Dakahlia       893789
 5 Fayoum         819231
 6 Gharbia        825147
 7 Giza           829587
 8 Hurghada      1700940
 9 Port Said      857901
10 Sohag          901010
> city_arrange <- arrange(city_per_total,desc(totalspend)) #arrange it by total descending
> city_arrange
# A tibble: 10 x 2
   city       totalspend
   <chr>           <int>
 1 Alexandria    2597481
 2 Cairo         2516267
 3 Hurghada      1700940
 4 Sohag          901010
 5 Dakahlia       893789
 6 Port Said      857901
 7 Giza           829587
 8 Gharbia        825147
 9 Fayoum         819231
10 Aswan          772871
```

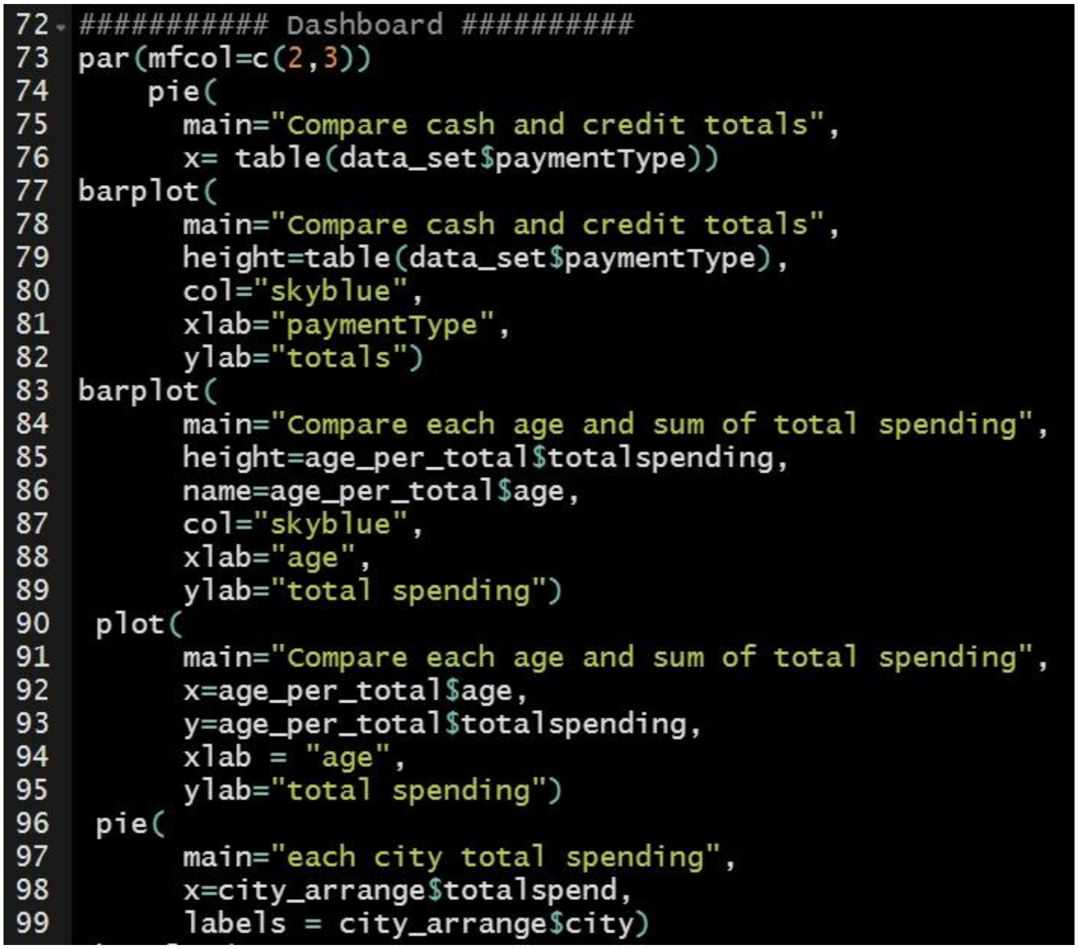




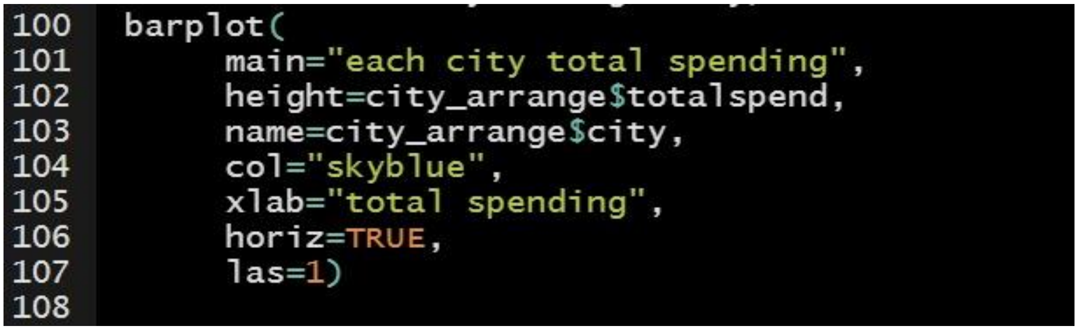
observation:

Alexandria is the highest in total spending, Aswan is the least.

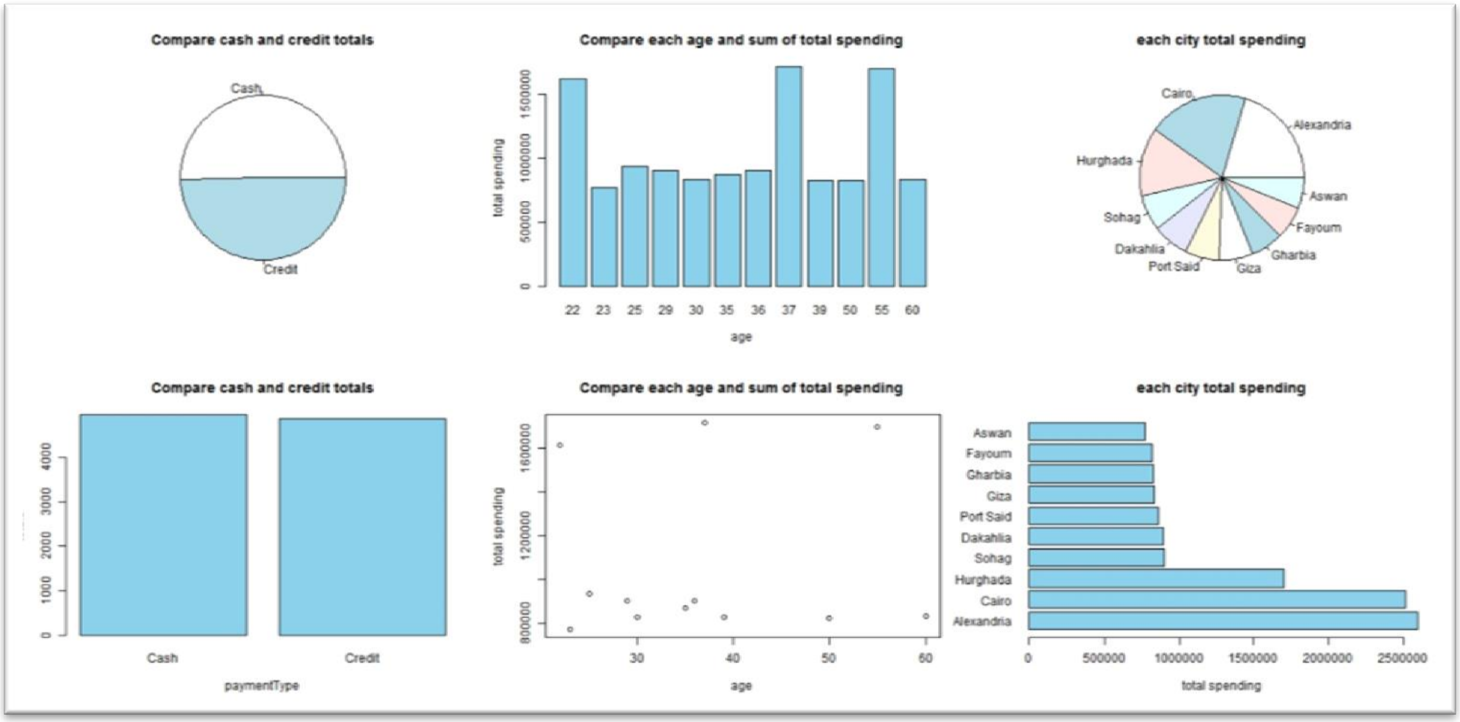
3- Dashboard:



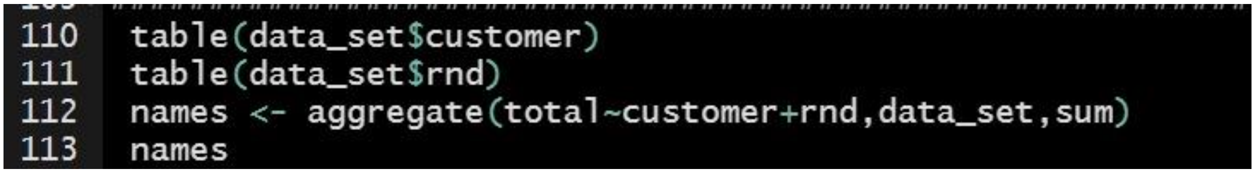
```r
########### Dashboard ##########
par(mfcol=c(2,3))
    pie(
      main="Compare cash and credit totals",
      x= table(data_set$paymentType))
barplot(
      main="Compare cash and credit totals",
      height=table(data_set$paymentType),
      col="skyblue",
      xlab="paymentType",
      ylab="totals")
barplot(
      main="Compare each age and sum of total spending",
      height=age_per_total$totalspending,
      name=age_per_total$age,
      col="skyblue",
      xlab="age",
      ylab="total spending")
 plot(
      main="Compare each age and sum of total spending",
      x=age_per_total$age,
      y=age_per_total$totalspending,
      xlab = "age",
      ylab="total spending")
 pie(
      main="each city total spending",
      x=city_arrange$totalspend,
      labels = city_arrange$city)
```



```r
 barplot(
      main="each city total spending",
      height=city_arrange$totalspend,
      name=city_arrange$city,
      col="skyblue",
      xlab="total spending",
      horiz=TRUE,
      las=1)

```

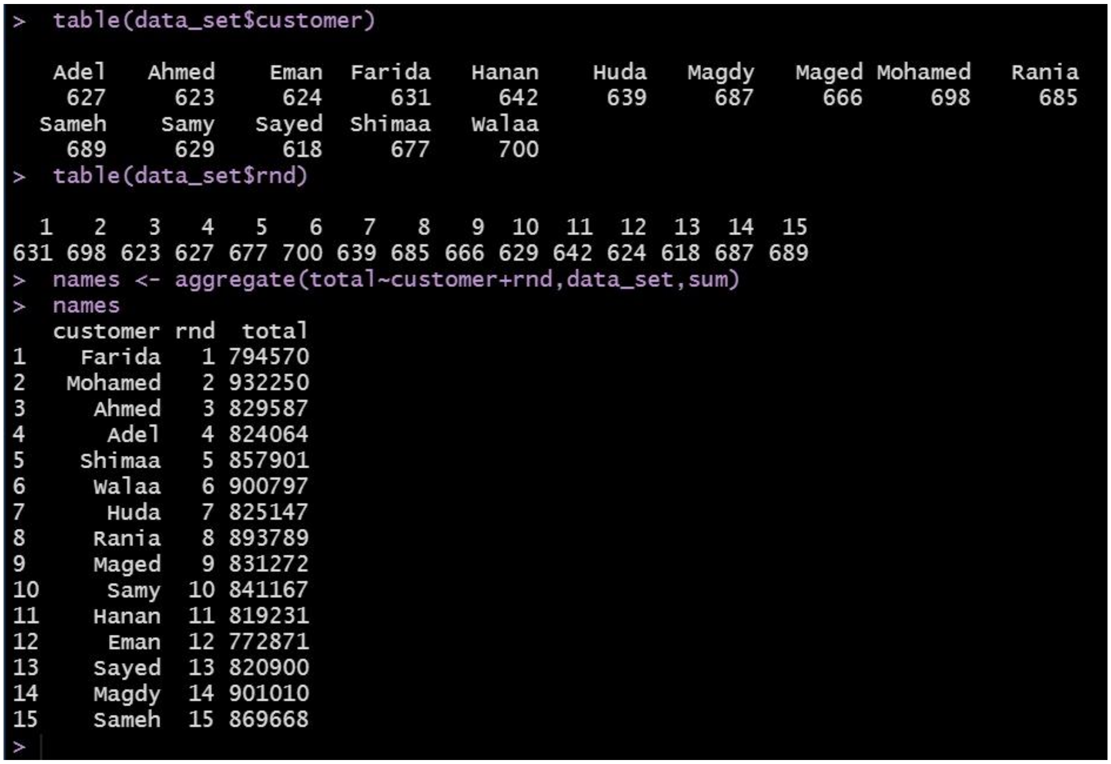

Compare cash and credit totals
Cash
Credit
Compare each age and sum of total spending
total spending
0
500000
1000000
1500000
22
23
25
29
30
35
36
37
39
50
55
60
age
each city total spending
Cairo
Alexandria
Hurghada
Aswan
Sohag
Fayoum
Dakahlia
Gharbia
Port Said
Giza
Compare cash and credit totals
0
1000
2000
3000
4000
Cash
Credit
paymentType
Compare each age and sum of total spending
total spending
800000
1200000
1600000
30
40
50
60
age
each city total spending
Aswan
Fayoum
Gharbia
Giza
Port Said
Dakahlia
Sohag
Hurghada
Cairo
Alexandria
0
500000
1000000
1500000
2000000
2500000
total spending

3- k-Mean:



```
table(data_set$customer)
table(data_set$rnd)
names <- aggregate(total~customer+rnd,data_set,sum)
names
```

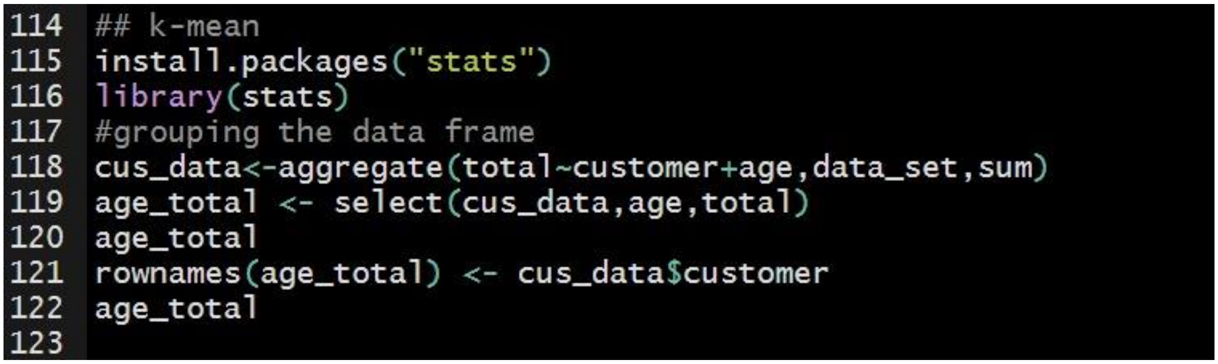
Out Put :



```
>  table(data_set$customer)

   Adel   Ahmed    Eman  Farida   Hanan    Huda   Magdy   Maged Mohamed   Rania 
    627     623     624     631     642     639     687     666     698     685 
  Sameh    Samy   Sayed  Shimaa   Walaa 
    689     629     618     677     700 
>  table(data_set$rnd)

  1   2   3   4   5   6   7   8   9  10  11  12  13  14  15 
631 698 623 627 677 700 639 685 666 629 642 624 618 687 689 
>  names <- aggregate(total~customer+rnd,data_set,sum)
>  names
   customer rnd  total
1    Farida   1 794570
2   Mohamed   2 932250
3     Ahmed   3 829587
4      Adel   4 824064
5    Shimaa   5 857901
6     Walaa   6 900797
7      Huda   7 825147
8     Rania   8 893789
9     Maged   9 831272
10     Samy  10 841167
11    Hanan  11 819231
12     Eman  12 772871
13    Sayed  13 820900
14    Magdy  14 901010
15    Sameh  15 869668
> 
```

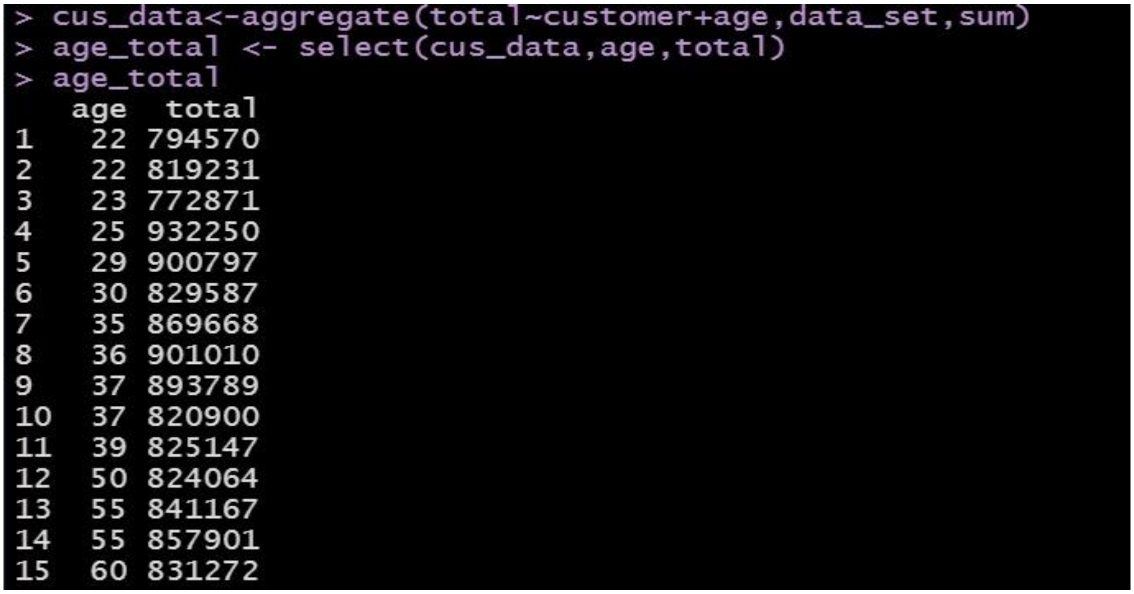
Observation:

- after observation the dataset we found out that the rnd is a unique number of the customer.
- testing that: we create A frequency table to represent the number of every customer and another one to represent the number of every rnd .
- conclusion: we have 15 customers each of them has a unique rnd.

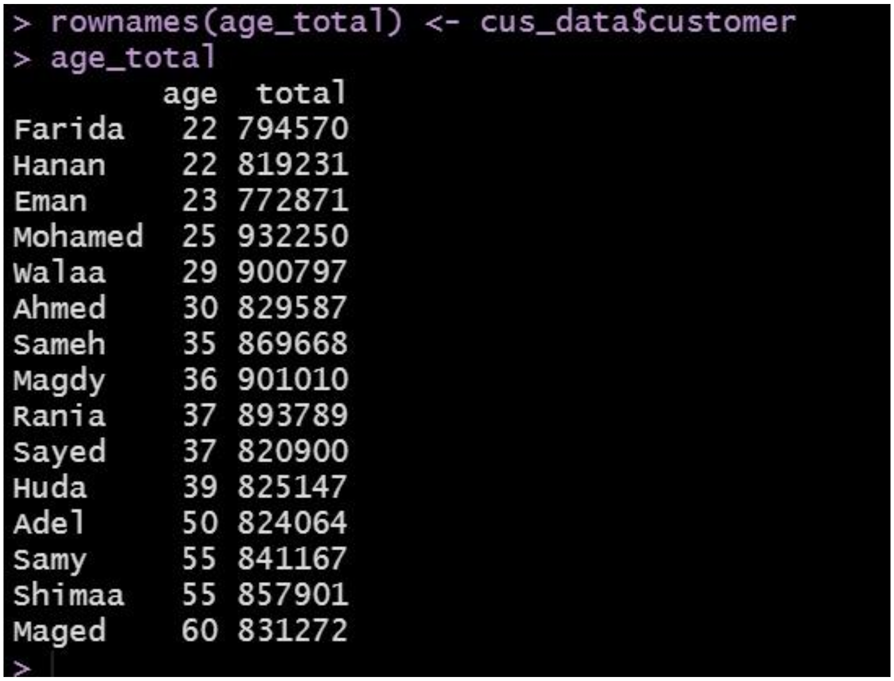


```
## k-mean
install.packages("stats")
library(stats)
#grouping the data frame
cus_data<-aggregate(total~customer+age,data_set,sum)
age_total <- select(cus_data,age,total)
age_total
rownames(age_total) <- cus_data$customer
age_total

```

Out put: each customer age and their total spending

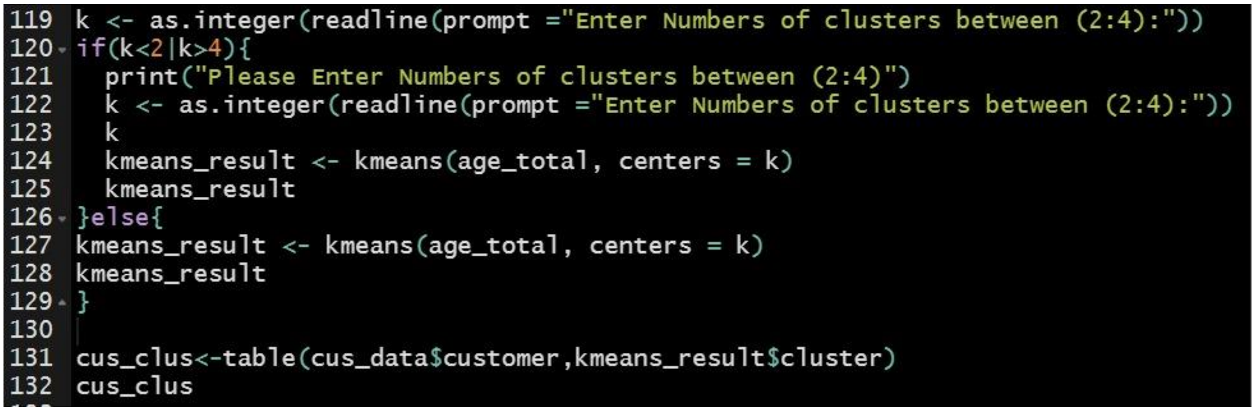


```
> cus_data<-aggregate(total~customer+age,data_set,sum)
> age_total <- select(cus_data,age,total)
> age_total
   age  total
1   22 794570
2   22 819231
3   23 772871
4   25 932250
5   29 900797
6   30 829587
7   35 869668
8   36 901010
9   37 893789
10  37 820900
11  39 825147
12  50 824064
13  55 841167
14  55 857901
15  60 831272
```



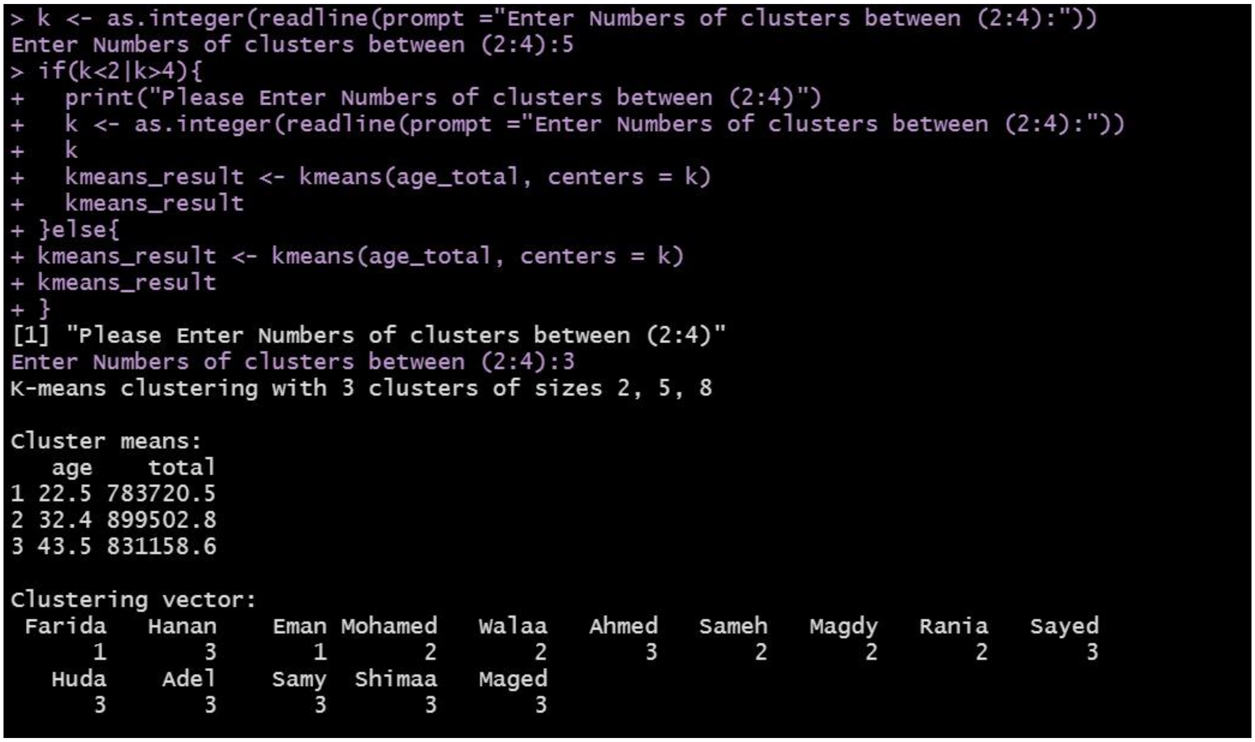
```
> rownames(age_total) <- cus_data$customer
> age_total
        age  total
Farida   22 794570
Hanan    22 819231
Eman     23 772871
Mohamed  25 932250
Walaa    29 900797
Ahmed    30 829587
Sameh    35 869668
Magdy    36 901010
Rania    37 893789
Sayed    37 820900
Huda     39 825147
Adel     50 824064
Samy     55 841167
Shimaa   55 857901
Maged    60 831272
>
```

K_Mean function& Read The Number of clusters :



```
k <- as.integer(readline(prompt ="Enter Numbers of clusters between (2:4):"))
if(k<2|k>4){
  print("Please Enter Numbers of clusters between (2:4)")
  k <- as.integer(readline(prompt ="Enter Numbers of clusters between (2:4):"))
  k
  kmeans_result <- kmeans(age_total, centers = k)
  kmeans_result
}else{
kmeans_result <- kmeans(age_total, centers = k)
kmeans_result
}

cus_clus<-table(cus_data$customer,kmeans_result$cluster)
cus_clus
```

Out Put : only accept k value 2:4

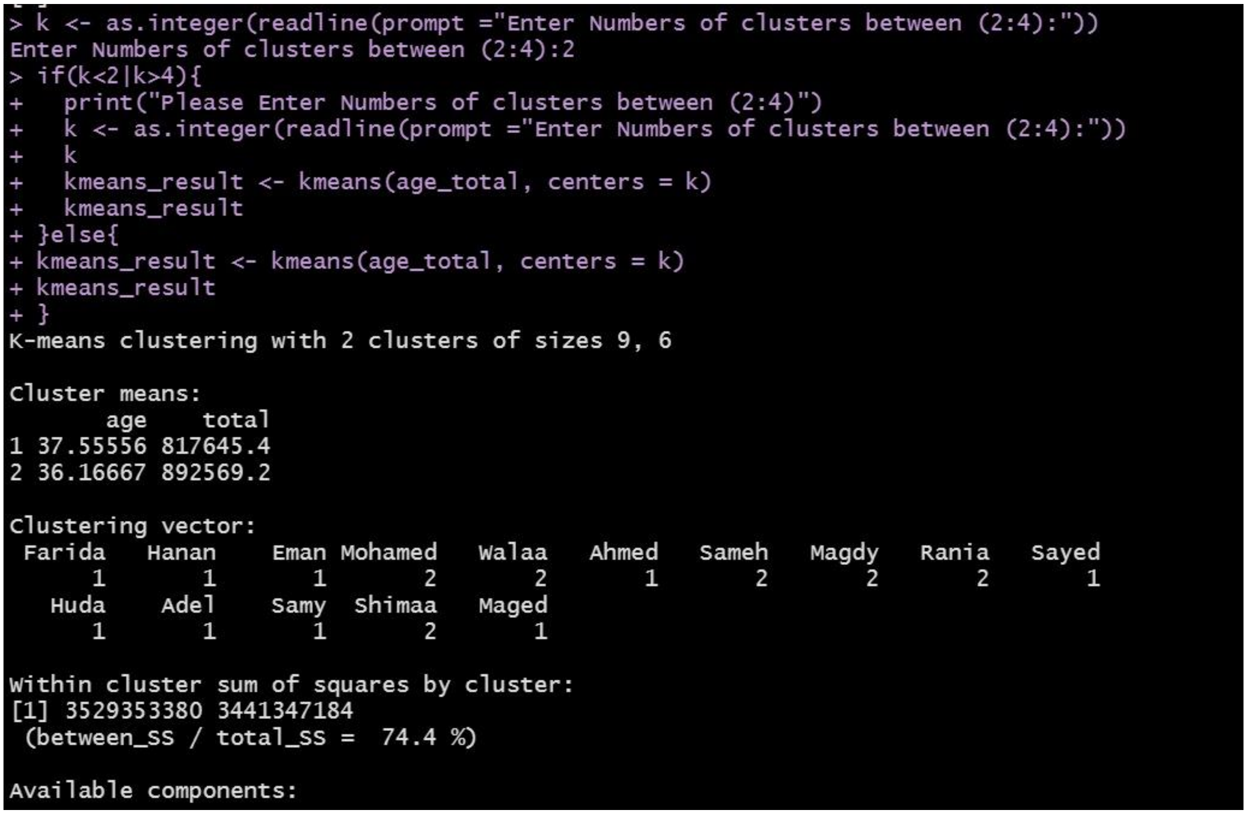


```
> k <- as.integer(readline(prompt ="Enter Numbers of clusters between (2:4):"))
Enter Numbers of clusters between (2:4):5
> if(k<2|k>4){
+   print("Please Enter Numbers of clusters between (2:4)")
+   k <- as.integer(readline(prompt ="Enter Numbers of clusters between (2:4):"))
+   k
+   kmeans_result <- kmeans(age_total, centers = k)
+   kmeans_result
+ }else{
+ kmeans_result <- kmeans(age_total, centers = k)
+ kmeans_result
+ }
[1] "Please Enter Numbers of clusters between (2:4)"
Enter Numbers of clusters between (2:4):3
K-means clustering with 3 clusters of sizes 2, 5, 8

Cluster means:
   age    total
1 22.5 783720.5
2 32.4 899502.8
3 43.5 831158.6

Clustering vector:
 Farida    Hanan    Eman Mohamed   Walaa    Ahmed    Sameh    Magdy    Rania    Sayed 
      1        3       1       2       2        3        2        2        2        3 
   Huda     Adel    Samy  Shimaa   Maged 
      3        3       3       3       3 
```

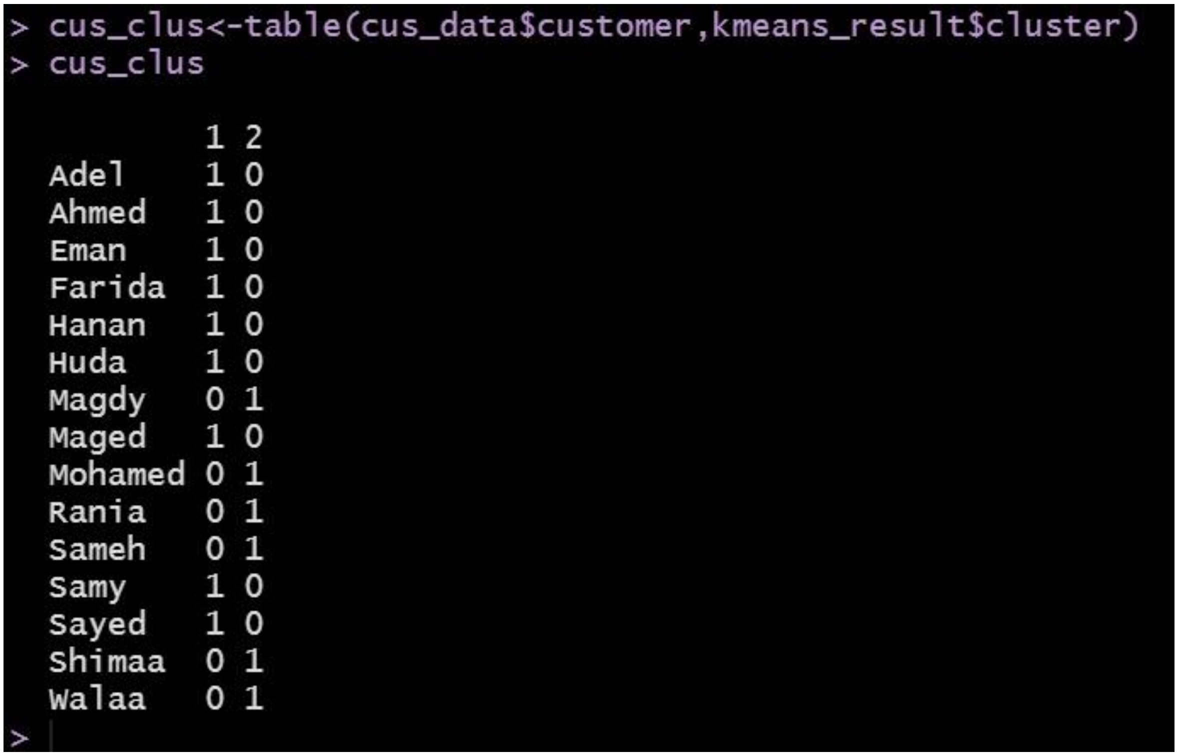
* let k = 3

cluster(1) has few data points -> decreasing the value of K

*Then let k=2

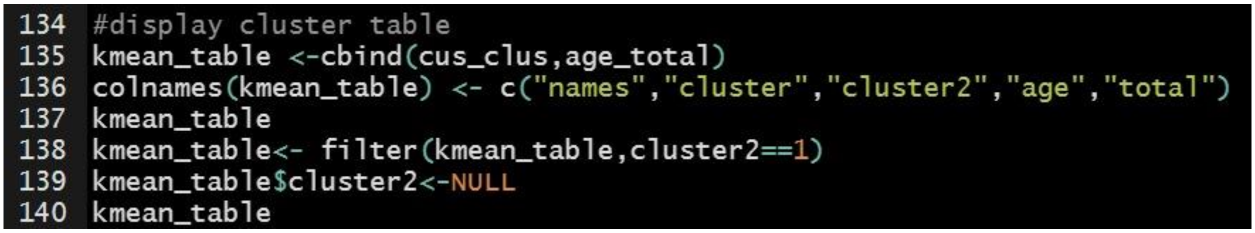


```
> k <- as.integer(readline(prompt ="Enter Numbers of clusters between (2:4):"))
Enter Numbers of clusters between (2:4):2
> if(k<2|k>4){
+   print("Please Enter Numbers of clusters between (2:4)")
+   k <- as.integer(readline(prompt ="Enter Numbers of clusters between (2:4):"))
+   k
+   kmeans_result <- kmeans(age_total, centers = k)
+   kmeans_result
+ }else{
+ kmeans_result <- kmeans(age_total, centers = k)
+ kmeans_result
+ }
K-means clustering with 2 clusters of sizes 9, 6

Cluster means:
       age    total
1 37.55556 817645.4
2 36.16667 892569.2

Clustering vector:
 Farida   Hanan    Eman Mohamed   Walaa   Ahmed   Sameh   Magdy   Rania   Sayed 
      1       1       1       2       2       1       2       2       2       1 
   Huda    Adel    Samy  Shimaa   Maged 
      1       1       1       2       1 

Within cluster sum of squares by cluster:
[1] 3529353380 3441347184
 (between_SS / total_SS =  74.4 %)

Available components:
```



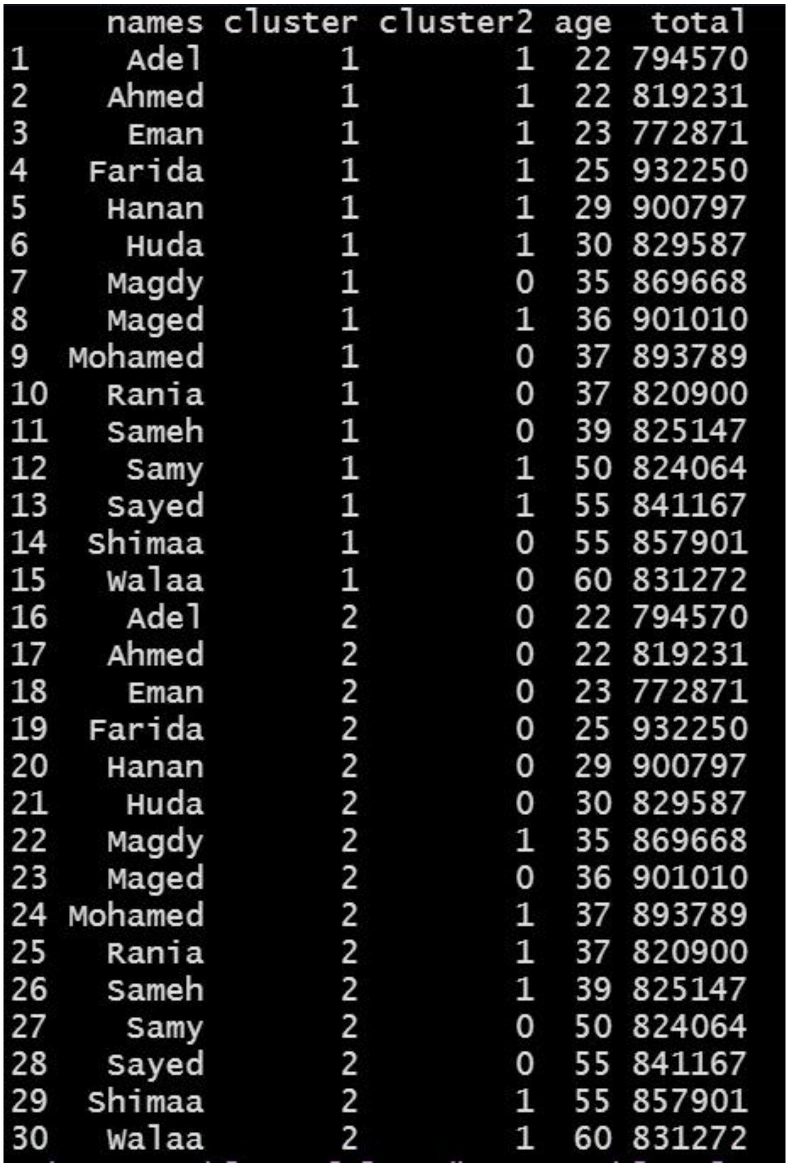
```
> cus_clus<-table(cus_data$customer,kmeans_result$cluster)
> cus_clus

          1 2
  Adel    1 0
  Ahmed   1 0
  Eman    1 0
  Farida  1 0
  Hanan   1 0
  Huda    1 0
  Magdy   0 1
  Maged   1 0
  Mohamed 0 1
  Rania   0 1
  Sameh   0 1
  Samy    1 0
  Sayed   1 0
  Shimaa  0 1
  Walaa   0 1
> 
```

Display Cluster Table :



```
#display cluster table
kmean_table <-cbind(cus_clus,age_total)
colnames(kmean_table) <- c("names","cluster","cluster2","age","total")
kmean_table
kmean_table<- filter(kmean_table,cluster2==1)
kmean_table$cluster2<-NULL
kmean_table
```

Out Put :

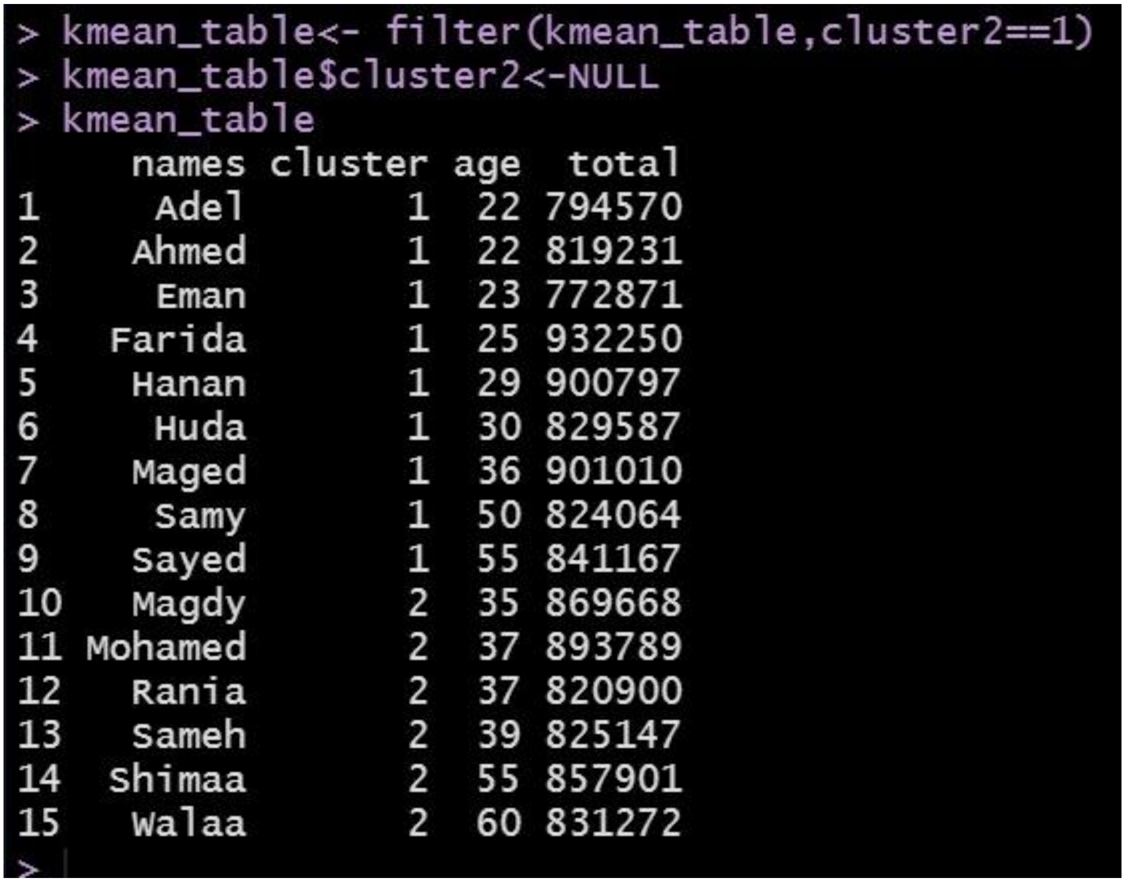


```
     names cluster cluster2 age  total
1     Adel       1        1  22 794570
2    Ahmed       1        1  22 819231
3     Eman       1        1  23 772871
4   Farida       1        1  25 932250
5    Hanan       1        1  29 900797
6     Huda       1        1  30 829587
7    Magdy       1        0  35 869668
8    Maged       1        1  36 901010
9  Mohamed       1        0  37 893789
10   Rania       1        0  37 820900
11   Sameh       1        0  39 825147
12    Samy       1        1  50 824064
13   Sayed       1        1  55 841167
14  Shimaa       1        0  55 857901
15   Walaa       1        0  60 831272
16    Adel       2        0  22 794570
17   Ahmed       2        0  22 819231
18    Eman       2        0  23 772871
19  Farida       2        0  25 932250
20   Hanan       2        0  29 900797
21    Huda       2        0  30 829587
22   Magdy       2        1  35 869668
23   Maged       2        0  36 901010
24 Mohamed       2        1  37 893789
25   Rania       2        1  37 820900
26   Sameh       2        1  39 825147
27    Samy       2        0  50 824064
28   Sayed       2        0  55 841167
29  Shimaa       2        1  55 857901
30   Walaa       2        1  60 831272
```

K-means preparation:

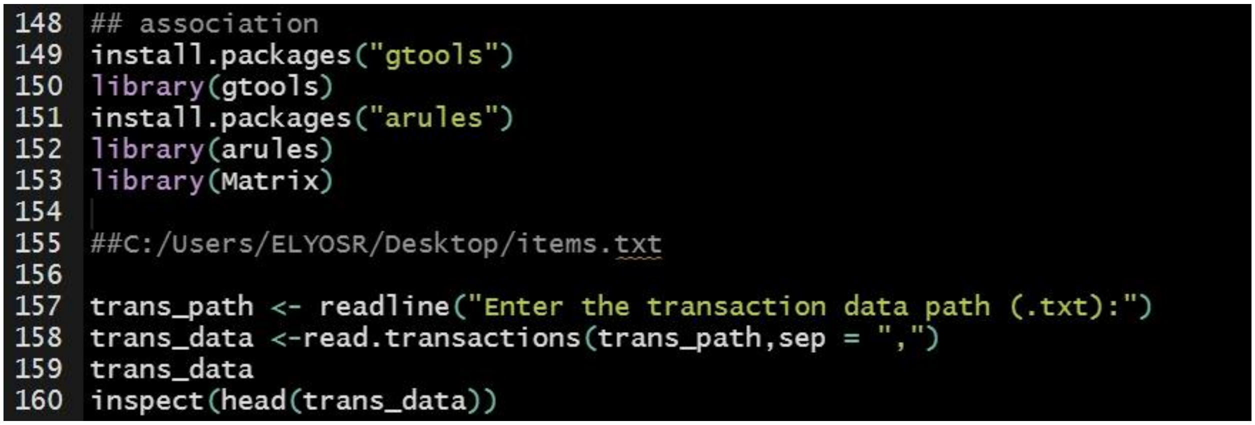
- Grouping cus_clus & age_total
- Filtering cluster groups
- Clean kmean_table

*Displayed k_mean table



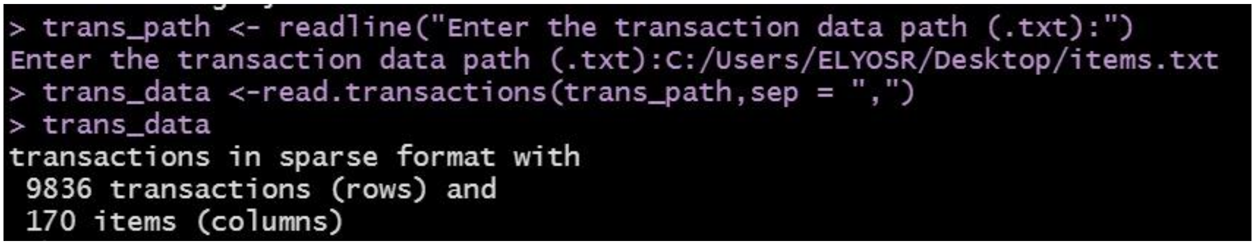
```
> kmean_table<- filter(kmean_table,cluster2==1)
> kmean_table$cluster2<-NULL
> kmean_table
     names cluster age  total
1     Adel       1  22 794570
2    Ahmed       1  22 819231
3     Eman       1  23 772871
4   Farida       1  25 932250
5    Hanan       1  29 900797
6     Huda       1  30 829587
7    Maged       1  36 901010
8     Samy       1  50 824064
9    Sayed       1  55 841167
10   Magdy       2  35 869668
11 Mohamed       2  37 893789
12   Rania       2  37 820900
13   Sameh       2  39 825147
14  Shimaa       2  55 857901
15   Walaa       2  60 831272
> 
```

4-Association & Read transaction data :

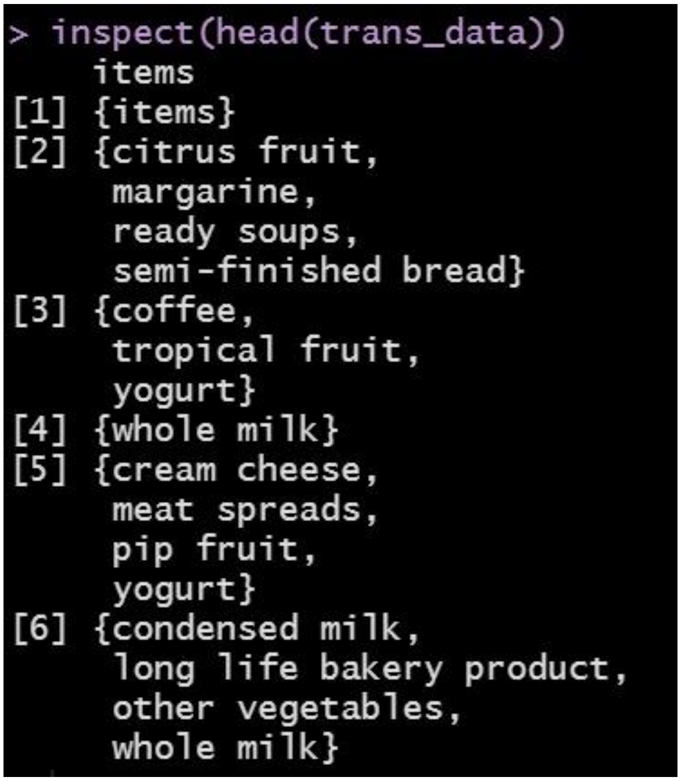


```
## association
install.packages("gtools")
library(gtools)
install.packages("arules")
library(arules)
library(Matrix)

##C:/Users/ELYOSR/Desktop/items.txt

trans_path <- readline("Enter the transaction data path (.txt):")
trans_data <-read.transactions(trans_path,sep = ",")
trans_data
inspect(head(trans_data))
```

Out Put :

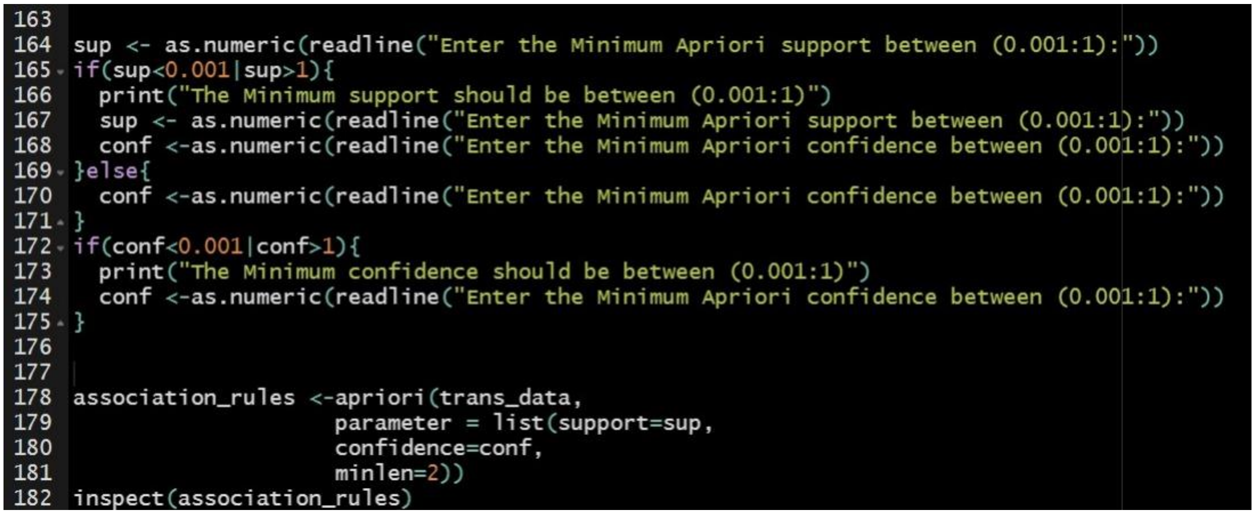


```
> trans_path <- readline("Enter the transaction data path (.txt):")
Enter the transaction data path (.txt):C:/Users/ELYOSR/Desktop/items.txt
> trans_data <-read.transactions(trans_path,sep = ",")
> trans_data
transactions in sparse format with
 9836 transactions (rows) and
 170 items (columns)
```



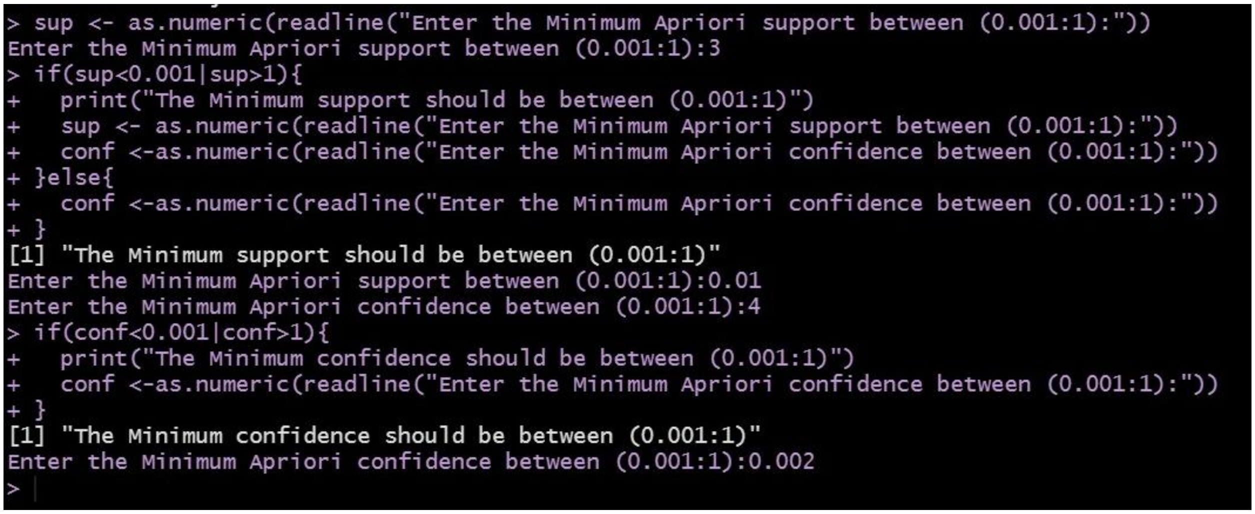
```
> inspect(head(trans_data))
    items
[1] {items}
[2] {citrus fruit,
     margarine,
     ready soups,
     semi-finished bread}
[3] {coffee,
     tropical fruit,
     yogurt}
[4] {whole milk}
[5] {cream cheese,
     meat spreads,
     pip fruit,
     yogurt}
[6] {condensed milk,
     long life bakery product,
     other vegetables,
     whole milk}
```

Apply the association and Read Min_Support and Min_Confidence :

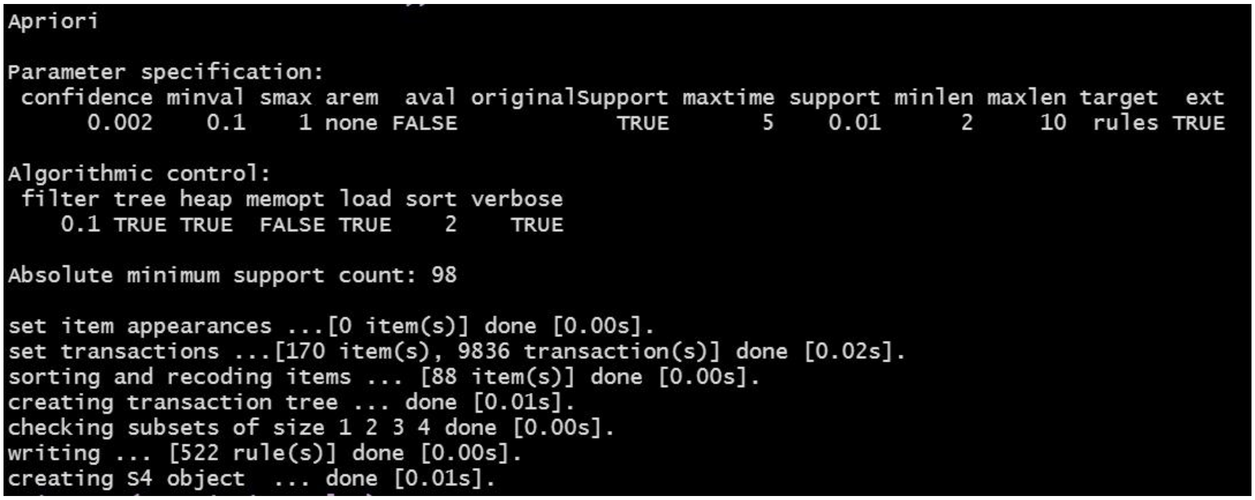


```r

sup <- as.numeric(readline("Enter the Minimum Apriori support between (0.001:1):"))
if(sup<0.001|sup>1){
  print("The Minimum support should be between (0.001:1)")
  sup <- as.numeric(readline("Enter the Minimum Apriori support between (0.001:1):"))
  conf <-as.numeric(readline("Enter the Minimum Apriori confidence between (0.001:1):"))
}else{
  conf <-as.numeric(readline("Enter the Minimum Apriori confidence between (0.001:1):"))
}
if(conf<0.001|conf>1){
  print("The Minimum confidence should be between (0.001:1)")
  conf <-as.numeric(readline("Enter the Minimum Apriori confidence between (0.001:1):"))
}


association_rules <-apriori(trans_data,
                    parameter = list(support=sup,
                    confidence=conf,
                    minlen=2))
inspect(association_rules)
```

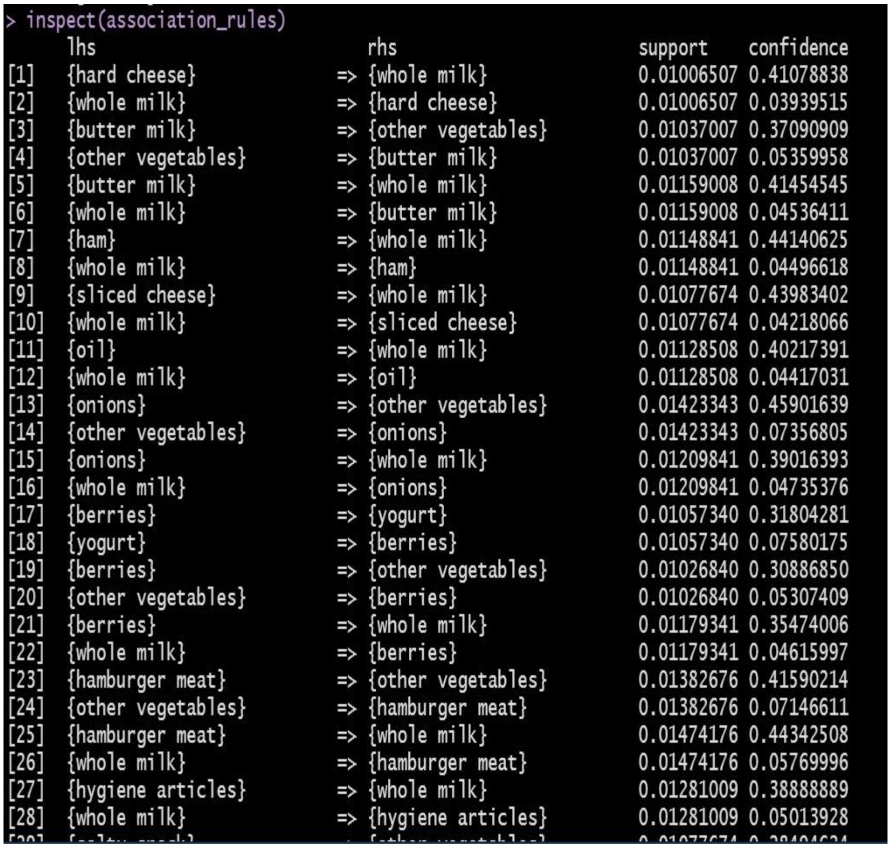
Out Put :



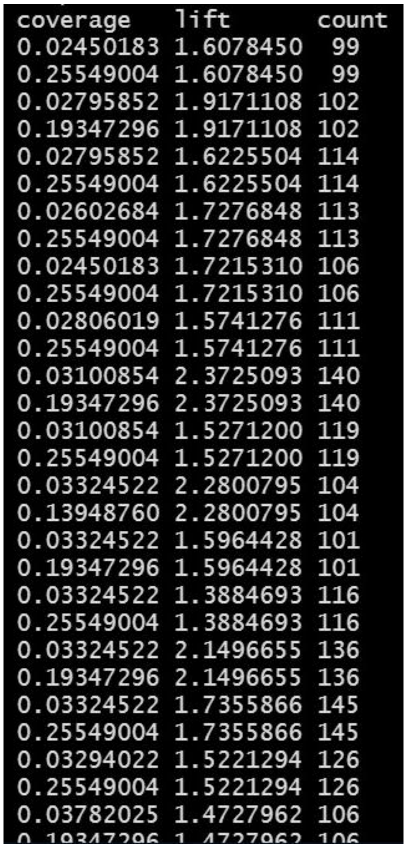
```
> sup <- as.numeric(readline("Enter the Minimum Apriori support between (0.001:1):"))
Enter the Minimum Apriori support between (0.001:1):3
> if(sup<0.001|sup>1){
+   print("The Minimum support should be between (0.001:1)")
+   sup <- as.numeric(readline("Enter the Minimum Apriori support between (0.001:1):"))
+   conf <-as.numeric(readline("Enter the Minimum Apriori confidence between (0.001:1):"))
+ }else{
+   conf <-as.numeric(readline("Enter the Minimum Apriori confidence between (0.001:1):"))
+ }
[1] "The Minimum support should be between (0.001:1)"
Enter the Minimum Apriori support between (0.001:1):0.01
Enter the Minimum Apriori confidence between (0.001:1):4
> if(conf<0.001|conf>1){
+   print("The Minimum confidence should be between (0.001:1)")
+   conf <-as.numeric(readline("Enter the Minimum Apriori confidence between (0.001:1):"))
+ }
[1] "The Minimum confidence should be between (0.001:1)"
Enter the Minimum Apriori confidence between (0.001:1):0.002
> 
```



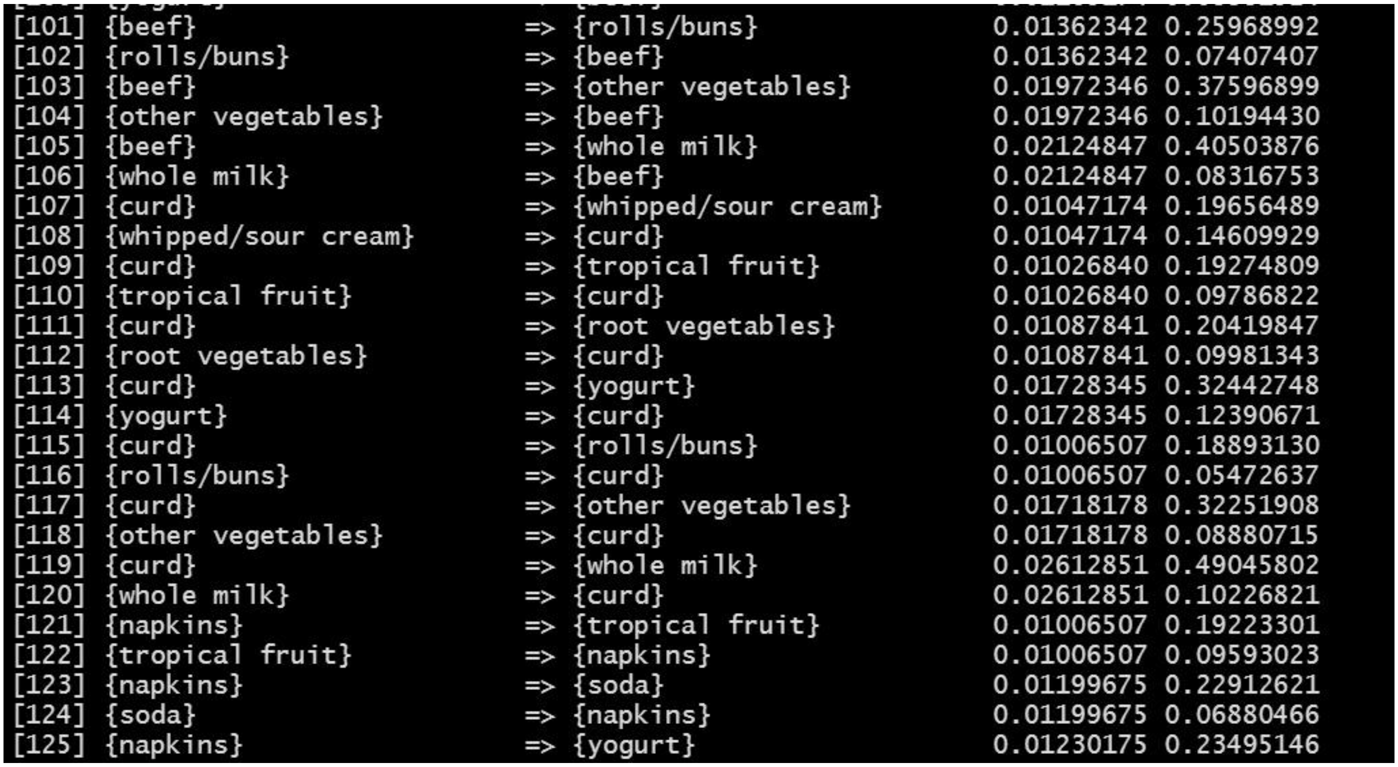
```
Apriori

Parameter specification:
 confidence minval smax arem  aval originalSupport maxtime support minlen maxlen target  ext
      0.002    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 ...[170 item(s), 9836 transaction(s)] done [0.02s].
sorting and recoding items ... [88 item(s)] done [0.00s].
creating transaction tree ... done [0.01s].
checking subsets of size 1 2 3 4 done [0.00s].
writing ... [522 rule(s)] done [0.00s].
creating S4 object  ... done [0.01s].
```



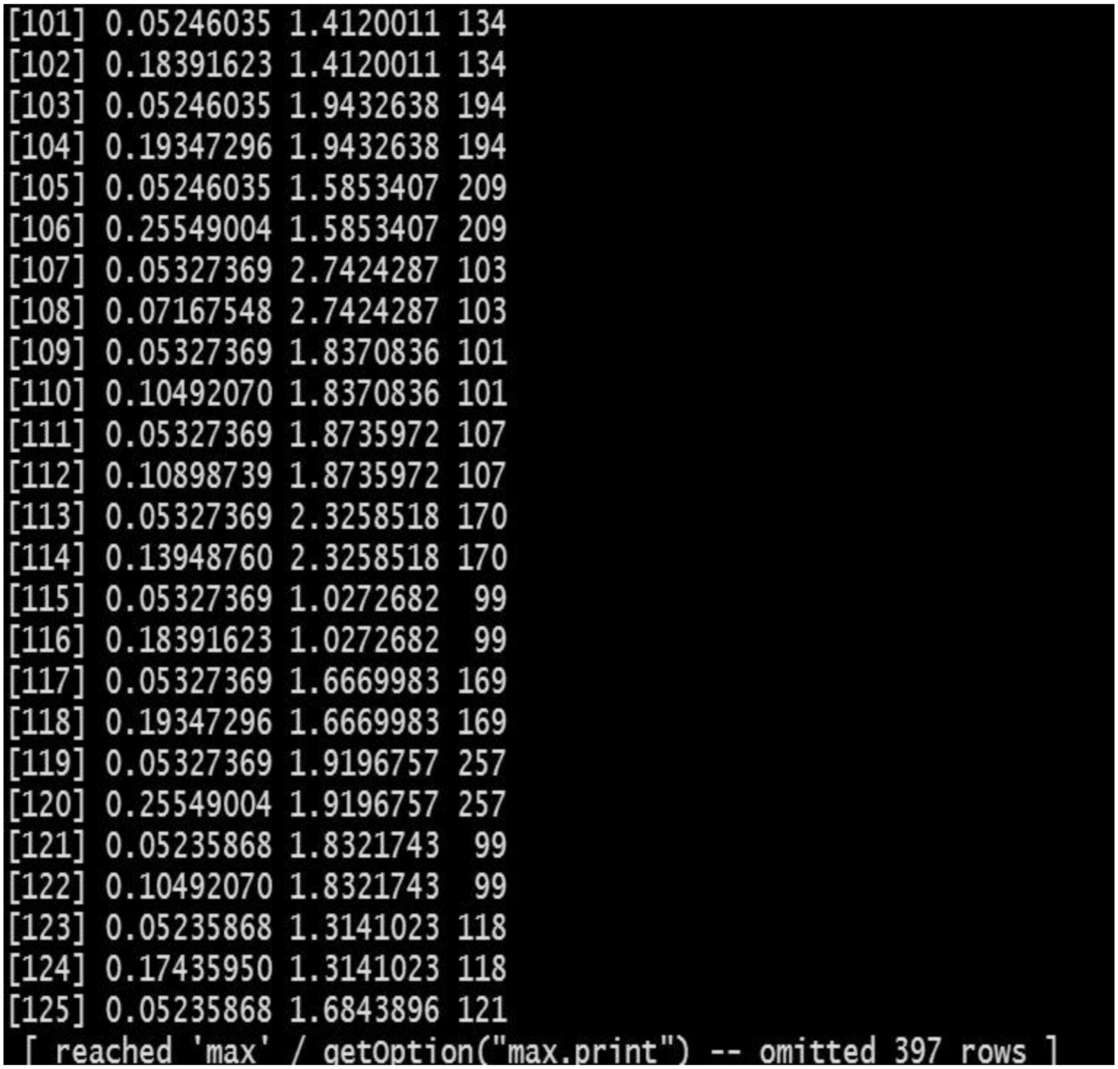
```
> inspect(association_rules)
     lhs                      rhs                  support    confidence
[1]  {hard cheese}         => {whole milk}         0.01006507 0.41078838
[2]  {whole milk}          => {hard cheese}        0.01006507 0.03939515
[3]  {butter milk}         => {other vegetables}   0.01037007 0.37090909
[4]  {other vegetables}    => {butter milk}        0.01037007 0.05359958
[5]  {butter milk}         => {whole milk}         0.01159008 0.41454545
[6]  {whole milk}          => {butter milk}        0.01159008 0.04536411
[7]  {ham}                 => {whole milk}         0.01148841 0.44140625
[8]  {whole milk}          => {ham}                0.01148841 0.04496618
[9]  {sliced cheese}       => {whole milk}         0.01077674 0.43983402
[10] {whole milk}          => {sliced cheese}      0.01077674 0.04218066
[11] {oil}                 => {whole milk}         0.01128508 0.40217391
[12] {whole milk}          => {oil}                0.01128508 0.04417031
[13] {onions}              => {other vegetables}   0.01423343 0.45901639
[14] {other vegetables}    => {onions}             0.01423343 0.07356805
[15] {onions}              => {whole milk}         0.01209841 0.39016393
[16] {whole milk}          => {onions}             0.01209841 0.04735376
[17] {berries}             => {yogurt}             0.01057340 0.31804281
[18] {yogurt}              => {berries}            0.01057340 0.07580175
[19] {berries}             => {other vegetables}   0.01026840 0.30886850
[20] {other vegetables}    => {berries}            0.01026840 0.05307409
[21] {berries}             => {whole milk}         0.01179341 0.35474006
[22] {whole milk}          => {berries}            0.01179341 0.04615997
[23] {hamburger meat}      => {other vegetables}   0.01382676 0.41590214
[24] {other vegetables}    => {hamburger meat}     0.01382676 0.07146611
[25] {hamburger meat}      => {whole milk}         0.01474176 0.44342508
[26] {whole milk}          => {hamburger meat}     0.01474176 0.05769996
[27] {hygiene articles}    => {whole milk}         0.01281009 0.38888889
[28] {whole milk}          => {hygiene articles}   0.01281009 0.05013928
```



```
coverage   lift      count
0.02450183 1.6078450  99
0.25549004 1.6078450  99
0.02795852 1.9171108 102
0.19347296 1.9171108 102
0.02795852 1.6225504 114
0.25549004 1.6225504 114
0.02602684 1.7276848 113
0.25549004 1.7276848 113
0.02450183 1.7215310 106
0.25549004 1.7215310 106
0.02806019 1.5741276 111
0.25549004 1.5741276 111
0.03100854 2.3725093 140
0.19347296 2.3725093 140
0.03100854 1.5271200 119
0.25549004 1.5271200 119
0.03324522 2.2800795 104
0.13948760 2.2800795 104
0.03324522 1.5964428 101
0.19347296 1.5964428 101
0.03324522 1.3884693 116
0.25549004 1.3884693 116
0.03324522 2.1496655 136
0.19347296 2.1496655 136
0.03324522 1.7355866 145
0.25549004 1.7355866 145
0.03294022 1.5221294 126
0.25549004 1.5221294 126
0.03782025 1.4727962 106
0.19347296 1.4727962 106
```



```
[101] {beef}                 => {rolls/buns}           0.01362342 0.25968992
[102] {rolls/buns}           => {beef}                 0.01362342 0.07407407
[103] {beef}                 => {other vegetables}     0.01972346 0.37596899
[104] {other vegetables}     => {beef}                 0.01972346 0.10194430
[105] {beef}                 => {whole milk}           0.02124847 0.40503876
[106] {whole milk}           => {beef}                 0.02124847 0.08316753
[107] {curd}                 => {whipped/sour cream}   0.01047174 0.19656489
[108] {whipped/sour cream}   => {curd}                 0.01047174 0.14609929
[109] {curd}                 => {tropical fruit}       0.01026840 0.19274809
[110] {tropical fruit}       => {curd}                 0.01026840 0.09786822
[111] {curd}                 => {root vegetables}      0.01087841 0.20419847
[112] {root vegetables}      => {curd}                 0.01087841 0.09981343
[113] {curd}                 => {yogurt}               0.01728345 0.32442748
[114] {yogurt}               => {curd}                 0.01728345 0.12390671
[115] {curd}                 => {rolls/buns}           0.01006507 0.18893130
[116] {rolls/buns}           => {curd}                 0.01006507 0.05472637
[117] {curd}                 => {other vegetables}     0.01718178 0.32251908
[118] {other vegetables}     => {curd}                 0.01718178 0.08880715
[119] {curd}                 => {whole milk}           0.02612851 0.49045802
[120] {whole milk}           => {curd}                 0.02612851 0.10226821
[121] {napkins}              => {tropical fruit}       0.01006507 0.19223301
[122] {tropical fruit}       => {napkins}              0.01006507 0.09593023
[123] {napkins}              => {soda}                 0.01199675 0.22912621
[124] {soda}                 => {napkins}              0.01199675 0.06880466
[125] {napkins}              => {yogurt}               0.01230175 0.23495146
```



```
[101] 0.05246035 1.4120011 134
[102] 0.18391623 1.4120011 134
[103] 0.05246035 1.9432638 194
[104] 0.19347296 1.9432638 194
[105] 0.05246035 1.5853407 209
[106] 0.25549004 1.5853407 209
[107] 0.05327369 2.7424287 103
[108] 0.07167548 2.7424287 103
[109] 0.05327369 1.8370836 101
[110] 0.10492070 1.8370836 101
[111] 0.05327369 1.8735972 107
[112] 0.10898739 1.8735972 107
[113] 0.05327369 2.3258518 170
[114] 0.13948760 2.3258518 170
[115] 0.05327369 1.0272682  99
[116] 0.18391623 1.0272682  99
[117] 0.05327369 1.6669983 169
[118] 0.19347296 1.6669983 169
[119] 0.05327369 1.9196757 257
[120] 0.25549004 1.9196757 257
[121] 0.05235868 1.8321743  99
[122] 0.10492070 1.8321743  99
[123] 0.05235868 1.3141023 118
[124] 0.17435950 1.3141023 118
[125] 0.05235868 1.6843896 121
 [ reached 'max' / getOption("max.print") -- omitted 397 rows ]
```