Project Report "77"

Name	ID	Role
Aya youssef Mohamed Ahmed (Leader)	23011051	K-means
Aya Khedr Zaki	23011050	Association rule
Asmaa Adbullah Moussa	23011043	Data Cleaning +Gui
Salma Farag Gaber Abbas Ibrahim	23011082	Data visualization
Shahd Basel Ibrahim Alhassan	23011981	Data visualization
Rania Ali Abouzeid Mahmoud	23010113	Report

- ❖ **This program displays a graphical user interface screen for analyzed, visualized grocery store data by common analyzing techniques such as k-means clustering and association rule**.
- ❖ **A large data set of a grocery store was the input of the program which contained many columns such as:**
- ✓ **Customer: names of each customer who bought from the store.**
- ✓ **Items: describing the items in the grocery store bought by customers.**
- ✓ **Count: number of items that customers bought.**
- ✓ **Total: price of items bought by customers.**
- ✓ **City: where customers are from.**
- ✓ **Age: age of each customer.**
- ✓ **Rnd: a special number ID for each customer.**
- ✓ Payment type: cash or credit.
- **Before the analytic process, data set have been cleaned and prepared by some methods.**
- ❖ **After the analytic process, some outputs have been extracted :**
- ❖ **Some analytic graphs based on the relationship between data set items displayed by visualization methods to help in understanding the data and extracting important information**
- ❖ **Customers have been grouped based on their common attributes according to their age and the sum of total spending**
- ❖ **The relationship between the products in the store and each other, as they were divided according to the products most purchased by customers together.**

- ❖ **The best-selling products in the store were identified and known which helps to increase the store's sales**

PROJECT STEPS:

DATA CLEANING:

1. **Loading the Data:**

```r
install.packages("reader")
library("reader")
dataa<-read.csv("D:/New folder/data.csv")#to acsses the data
dataa
```

- **The `read.csv` function from the `reader` package is used to import the data from the CSV file and store it in the `dataa` variable.**

2. **Identifying Duplicates:**

```r
sum(duplicated(dataa))#display sum of duplicateds
```

- **The `duplicated` function is used to check for duplicate rows in the data.**
- **The `sum` function calculates the total number of duplicate rows present in the data.**

3. **Removing Duplicates:**

```r
dataa<-unique(dataa)#remove the duplicateds
sum(duplicated(dataa))#check
```

- **The `unique` function is employed to eliminate duplicate rows from the `dataa` dataset.**
- **The `sum` function is again used to verify that no duplicates remain after the cleaning process.**

4. **Identifying Missing Values:**

```r
sum(is.na(dataa))#display the empty cells
```

- The `is.na` function is used to identify any missing values (represented as NA) within the `dataa` dataset.
- The `sum` function calculates the total number of missing values present.

5. Data Rearrangement:

```
dataa<-dataa[,c("items","city","customer","paymentType",
                "count","total","rnd","age")]#rearrange cells
```

• The data is rearranged to select specific columns using square brackets `[]`. The selected columns are: "items", "city", "customer", "paymentType", "count", "total", "rnd", and "age". This rearranges the data to group numeric columns together.

Exploratory Data Analysis (EDA):

6. Identifying Outliers:

```
boxplot(dataa[,5:8])
outlier<- boxplot(dataa$count)$out#display outliers in count field
```

• The `boxplot` function will make it easy to show the outliers in the numeric columns, along with the `$out` operator, which is used to extract the outlier values identified for the "count" field. The `$out` operator retrieves the specific data points classified as outliers based on the boxplot analysis.

In conclusion:

We removed the 2 duplicates which we found, we did not find any missing values (NA) and finally by using the boxplot we found the outliers only in "count" columns, so we did not remove it as it is a normal thing to be outliers in this field

DATA VISUALIZATION:

FIRST GRAPH

In the first graph "comparing between cash and credit totals" pie chart will be used because the pie chart is a common type of data visualization that

used to represent categorical data it displays data as slices of a circular pie, where each slice represents a category and the size of each slice corresponds to the proportion of that category in the dataset which is needed in this question .

```r
install.packages("reader")  # install reader package to deal with csv file
library("reader")  # to import the reader package
```

First the reader package will be installed and imported to deal with the csv file which is the data that will be worked on and then read the csv file.

```r
output$plotText<-renderText({
  req(input$file)
  "The Visualisations"

})
output$plot <- renderPlot({  #displaying graphs
  big_data <- data()         # loading or generating a dataset that will be used and assigning it in a variable
  if(input$vusal==1){

    p<- table(big_data$paymentType)
    percentage <- sprintf("%.2f%%", 100 * p / sum(p))  # this is to calculate the percentage of each the cash and
    #credit and display two digits after decimal place and follow it by "%"
    pie(p, labels = percentage, main = "Compare cash and credit totals", col = c("#4F9FF0", "pink")) #create a pie chart to compare the
    #totals of cash and credit payments
    legend("bottomleft", legend = c("Cash", "Credit"), fill = c("#4F9FF0", "pink")) #to add a legend to the pie chart to explain the
    # colors used:light blue for cash and pink for credit
```

- ❖ **this code is used to create a pie chart and the code contains**

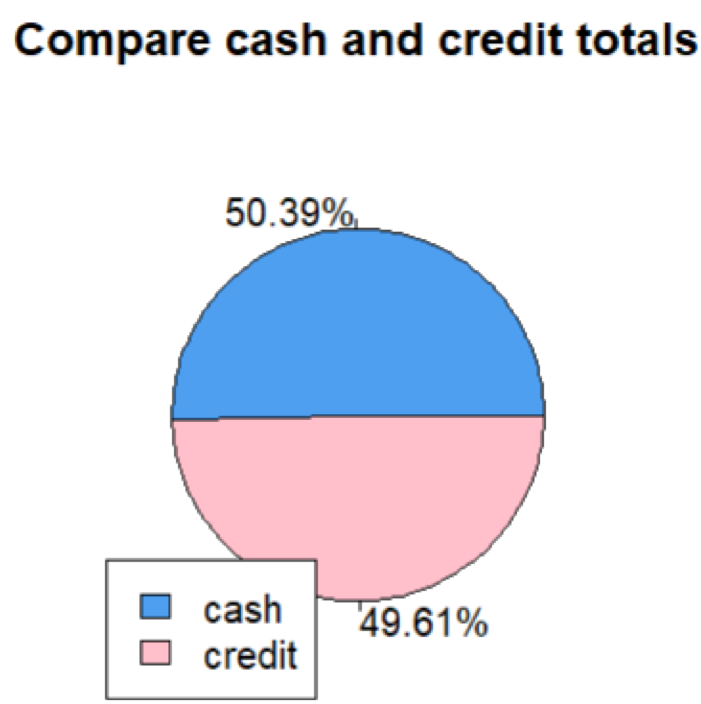
1- x: which contains the values that used in the pie chart

2- labels: which gives the description to the slices in the pie chart

3-main: which represents the title of the pie chart

4- col: which determine the color of the slices of the pie chart

Then , legend was used to put the information about the data that in the pie chart to make it easier to understand the pie chart and here is the output:



we can realize from the pie chart that the number of cash is greater than the number of credit, but with a very small difference

---

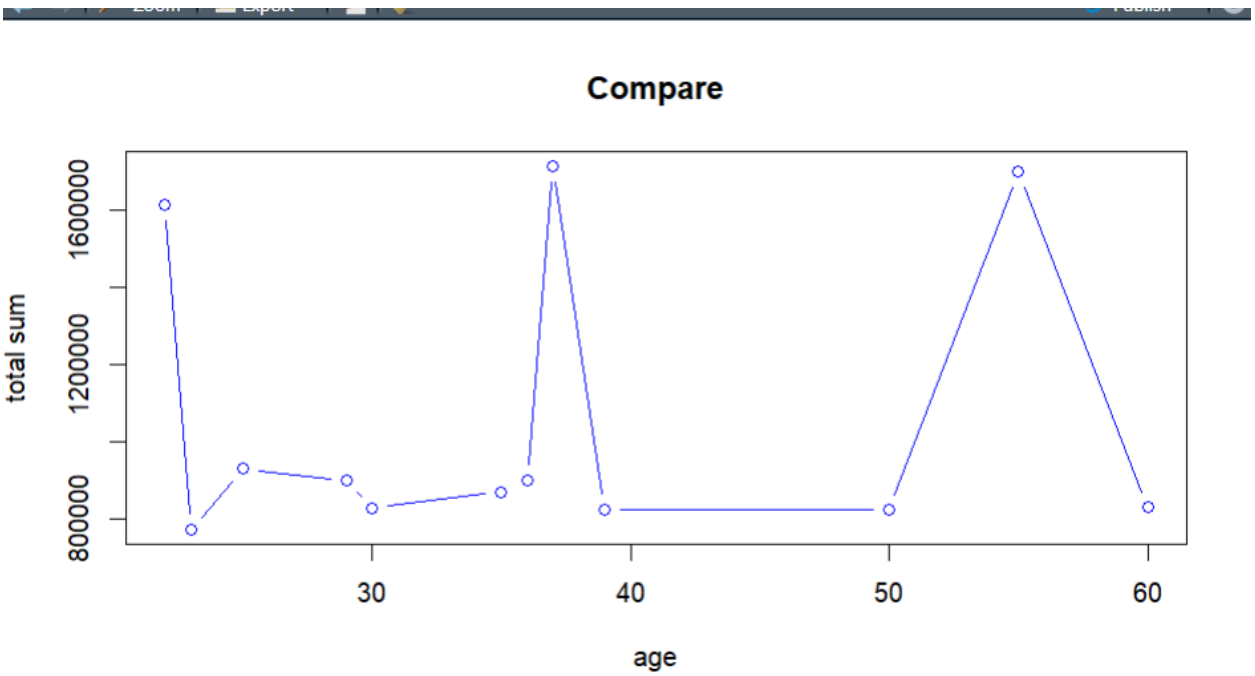
Second graph:

```
install.packages('dplyr')
library('dplyr')
```

the package is used to allow us of using the function 'group by' , "install" downloads the package with the needed functions , while "library" is a function that reads that data.

```
}else if(input$vusal==2){
   grp_tbl<-big_data %>%  #grouping the age col with sum of total spending
group_by(age)  #in here the function got the ages collected into a group of all the diffrent ages and how many times they had repeatred
   agg_tbl<-grp_tbl%>%
summarise(sum(total))  #this part collected the total sum of spendings , and associated with the previous part to show how much did each
   plot(x=agg_tbl$age , y=agg_tbl$`sum(total)`, main= "Compare",xlab="age",ylab="total sum",type='b',col="#4F9FF0") # create scatter p
```

this scatter plot is showing the distribution of the points , showing the data of each age on the level of their total spendings ,first part of it (x) is for the data that is put on the x-axix , second part (y) is the part that got the data of the totals to be placed pn the y-axis ,the (main) is the part responsibile for the name chosen which is "compare" , xlab and ylab are to put labels on the x-axis and y-axis , and the last part (type) is a part of the function that connected the dots together to give a clearer image of the data distribution



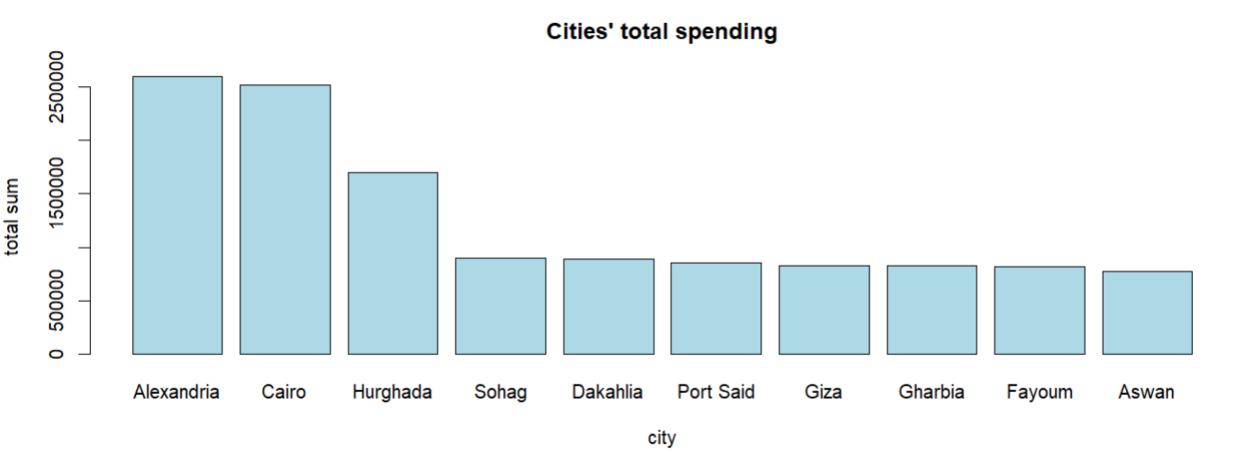
- **the reason for choosing a scatter plot is to show the relation between two numerical variables , to show their distribution and ranges**

❖ **from the graph: the most ages that have the largest total spending are: 55,37,22.**

Third graph

```
  }else if(input$vusal==3){
    grp_tbl<-big_data %>% group_by(city)
    grp_tbl    #this part grouped the different cities there were in the data and collected their amounts
    agg_tbl<-grp_tbl%>% summarise(sum(total))
  agg_tbl  #in here , the sum total of spending was collected , got associated with the different cities from the previous part
  #to show each city's spending
    agg_tbl_dec<-agg_tbl[order(-agg_tbl$`sum(total)`),]    #descending order
barplot(agg_tbl_dec$`sum(total)`,names.arg = agg_tbl_dec$city, main = "Cities' total spending" , xlab="city",ylab ="total sum", col='lig
    # create a box plot of the 'total' column of the 'data' data frame to show the distribution of the total spending
```

the barplot function turns the entered data into a visual graph using bars to show its amounts of occurrences , the reason behind choosing and using a barplot was to show the distribution of the data points , and because it was the best to compare the differences between different groups , the first part taking the sum total and applying the data into the y-axis , second part taking the cities and putting them onto the x-axis , the main giving the graph the desired name , xlab and ylab giving labels to the x-axis and y-axis , and 'col' giving colour to the bars in the graph .



- and we can see by the results of that function , the graph helps us understand more of the data , and see more into it and comprehend the groups more easily , and we can notice that Alexandria had the highest spending total out of all the cities with over 2500000 spent , while Aswan had the least spending total, and we can easily see their order and which country spent more than the other countries because of the previous that made them reordered going from highest to lowest making them in a descending order .

Graph Number 4

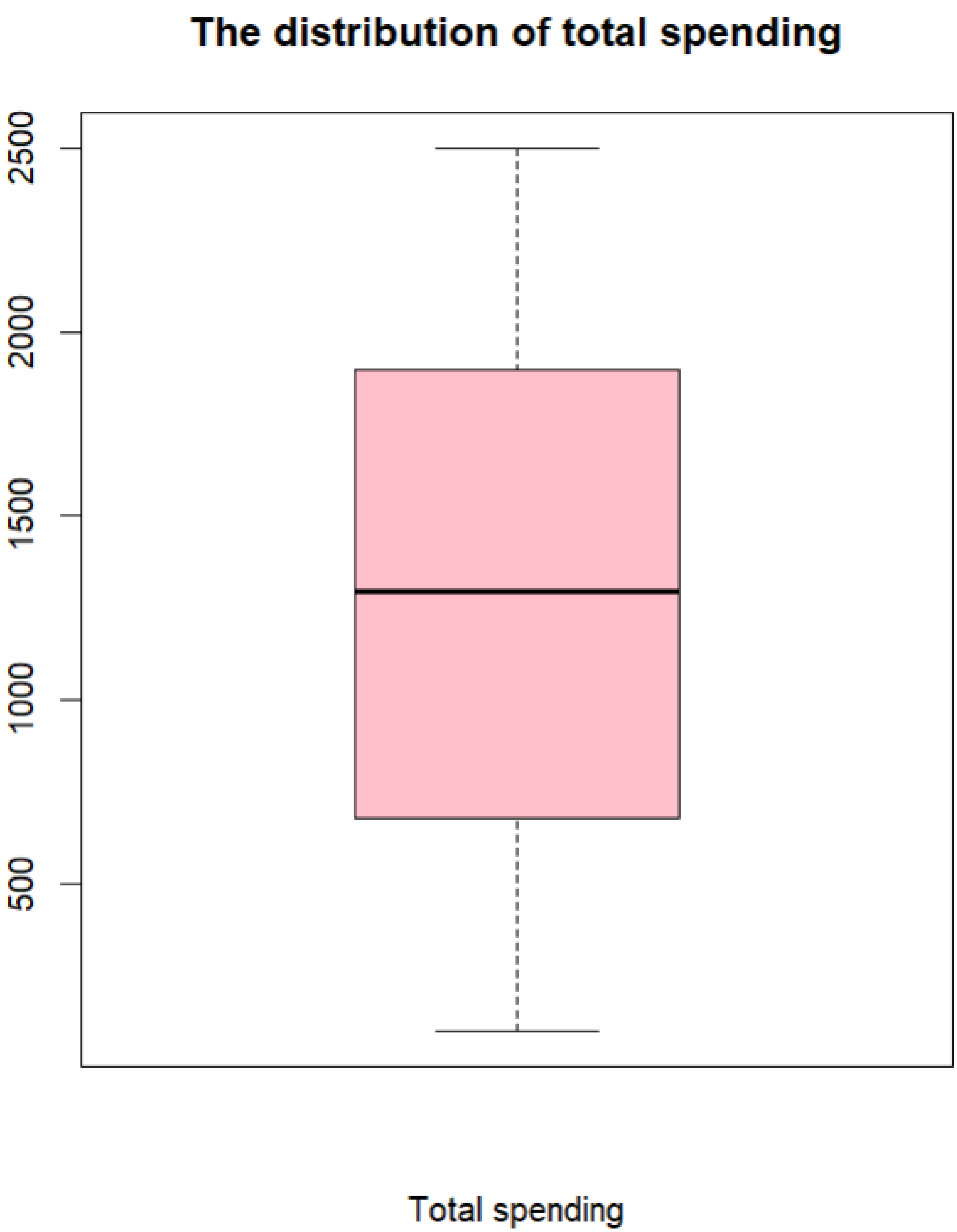
box plot will be used because the box plot shows how the data distributed, displays the five-number summary of a set of data: minimum, first quartile, median, third quartile, maximum and we can use it to identify outliers, Whish is needed in this question

```r
}else if(input$vusal==4){
  boxplot(x=big_data$total, main="The distribution of total spending", xlab="Total spending",col="pink")
  # create a box plot of the 'total' column of the 'data' data frame to show the distribution of the total spending
```

Here , box plot was created and the code contains

1. **the 'x' specifies the data for the box plot which is the 'total' column of the 'data' data frame and**
2. **the 'main' to put the title of the box plot**
3. **and the 'xlab' is the label that given to the x-axis**

```r
summary(data$total)   # to show the summary of the total spending
############################################################
```



From the box plot and the summary :

- ✓ **the minimum of total spending= 100**
- ✓ **the first quartile of total spending= 679**

- ✓ **the median of total spending= 1297**
- ✓ **the third quartile of total spending= 1897**
- ✓ **the maximum of total spending=2500**

-The association rules

```
output$aprioriText<-renderText({
  req(input$file)      # Check if a file is uploaded
  "The Apriori Graph"   # Display the message "The Apriori Graph"
})
# Render the plot and table for Apriori results
output$plot2<- renderPlot({
  if(input$display_apriori){   # Check if the checkbox for displaying Apriori results is checked
    req(input$file)             # Read the data from the uploaded file and convert it into transactions
    big_data<- data()           # Get the data (Note: You should define 'data()' somewhere
    i=big_data$items           # Extract the items from the dat
    t=read.transactions(textConnection(i),sep=",")    # Convert items into transactions
    inspect(head(t))                                  # Show the first few transactions for inspection
    # Apply the Apriori algorithm to find association rules:
    tdata<-apriori(t,parameter = list(supp= input$minSupport,conf= input$minConfidence,minlen=2))
    # Generate a plot showing the frequency of items:
    itemFrequencyPlot(t ,topN=6,type="absolute",col='pink')  # Plot the top 6 most frequent item

  output$apriori_table <- renderTable({      # Render the Apriori results in a table
     big_data<-data()
     # Extract items and convert them into transactions
     x<-big_data $items  # to access only at the items column
    y <- read.transactions(textConnection(x), sep = ",")  # Convert items into transactions

    # Apply the Apriori algorithm to find association rules:
    tdata <- apriori(y, parameter = list(supp= input$minSupport,conf= input$minConfidence,minlen=2))

    # Convert the results into a data frame to display:
    rules<-as(tdata,"data.frame")
    print(head(rules))     })
```

-Code explaining

1- The program checks if a file is uploaded using req(input$file)   to ensure that there is some data to work with .After that ,displays the message "The Apriori Graph" in the Shiny app interface.

2- Checks if the checkbox for displaying Apriori results is checked (input$display_apriori). If it is checked, the code proceeds to execute.

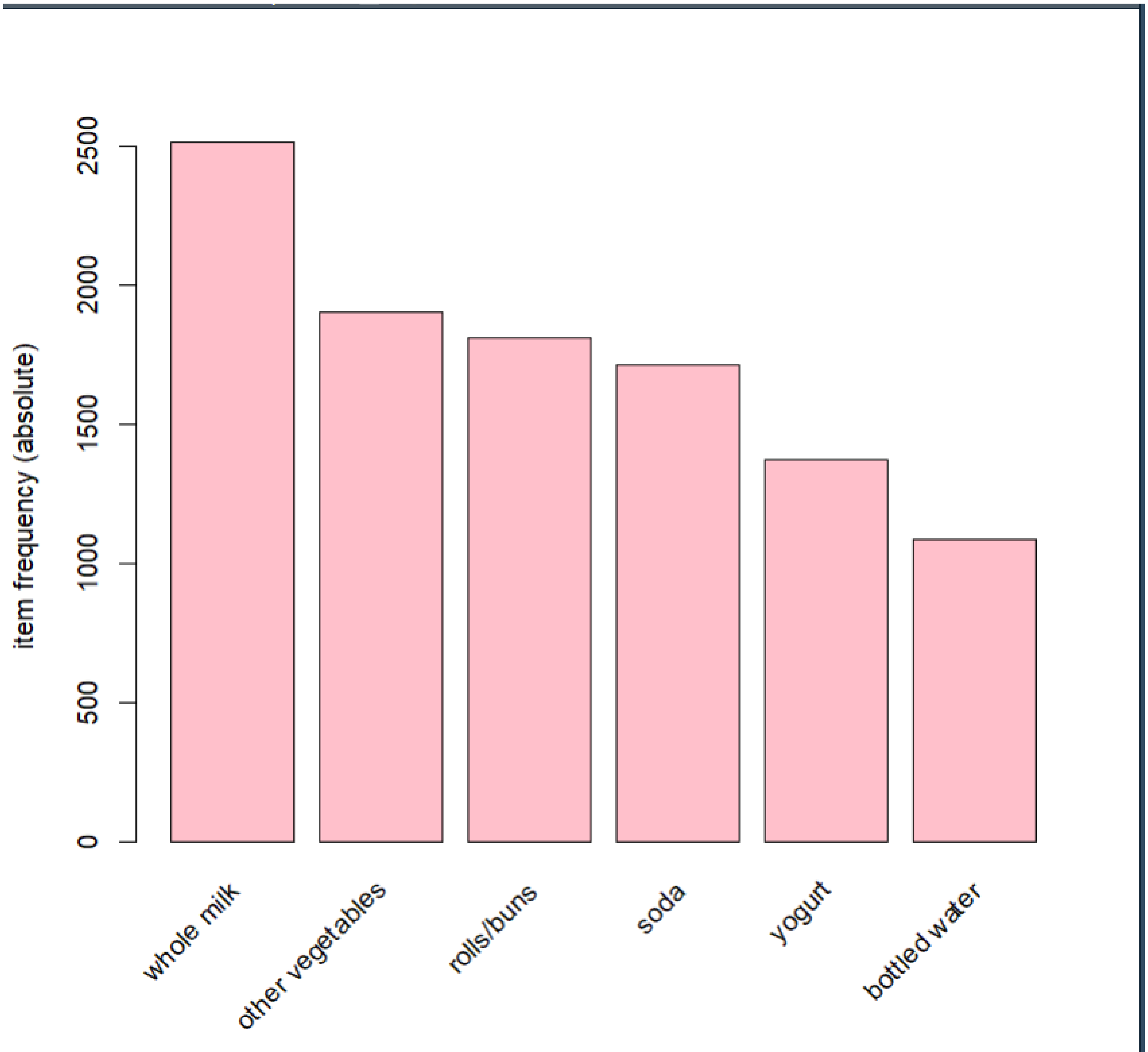
3- Extracts the items from the data using big_data$items and convert it to text format using textConnection() method .

4- Convert items into transactions using read.transactions() method.

5- Applies the Apriori algorithm using **apriori()** method to find association rules based on the transactions as the data became in the suitable form . It uses parameters specified by the user for support and confidence levels.

6- Generates a plot **itemFrequencyPlot()** method showing the frequency of items. It plots the top 6 most frequent items.

7- Renders the Apriori results in a table (**apriori_table**) for display in the Shiny app interface. It converts the results into a table format using **renderTable()** function and displays the first few rules for inspection.



Conclusion

-milk ,vegetables ,rolls ,buns ,soda ,yogurt and bottled water are the best seller products

- the milk in the first place by 2500 sales.

- there is 169 items ,9833 transactions.

* if we supposed that min support = min confidence =0.01 , the rules will equal 522 **rule.**

- {hard cheese} => {whole milk}is the most common rule as it frequented 99 and that's mean that most people who buy chess buy milk too

K-mean

```
output$kmeanText<-renderText({  #displaying a text as a title
  req(input$file)               # check if the file is uploaded
  "Kmean Table"
})
output$data_table_output <- renderDT({  #displaying a table for k-mean
  if (input$display_data_table) {
    data<- read.csv(input$file$datapath)
    # Grouping and summarizing data
    data_grouped<-data %>%           #this operator to perform grouped operations on the data set
      group_by(age,customer)        #grouping the data for k-means: customers names and ages
    data_totalspending<-data_grouped %>%      #assigning the grouped data: names and ages in a new variable
      summarize(total_spending=sum(total))    #to calculate the sum of each customers total spending and assign it as anew column

    data_for_kmean<-data_totalspending[,-c(2)]   #preparing the data for the kmean operation by removing the character column
    if (input$nCluster < 2 || input$nCluster > 4) {   #checking for ncluster numbers
      showModal(modalDialog(                         # it displays a modal dialog box with the title "Invalid Input" and the message "N
        title = "Invalid Input",
        "Number of clusters must be between 2 and 4.",
        easyClose = TRUE                        # specifies that the modal dialog box can be closed by clicking outside of it or by pres
      ))
      return(NULL)
    }
    kmeanss<-kmeans(data_for_kmean,centers = input$nCluster) #performing the k-mean operation
    kmeanss
    data_table<-data.frame(data_totalspending$customer,   #creating a data frame and adding the cluster vector of each customer to it
                           data_totalspending$age,
                           data_totalspending$total_spending,
                           kmeanss$cluster)
```

In this part dataset has been grouped by age, customers and the sum of their total spending and performed k-mean clustering on it

There is a check condition if the user enters a number of clusters bigger than 4 or less than 2,then a message appears "invalid input".

The result was that customers divided into groups depending on their age and total spending

A table consists of customers names, ages, total spending and cluster vector was created to understand the relationship between them easily .