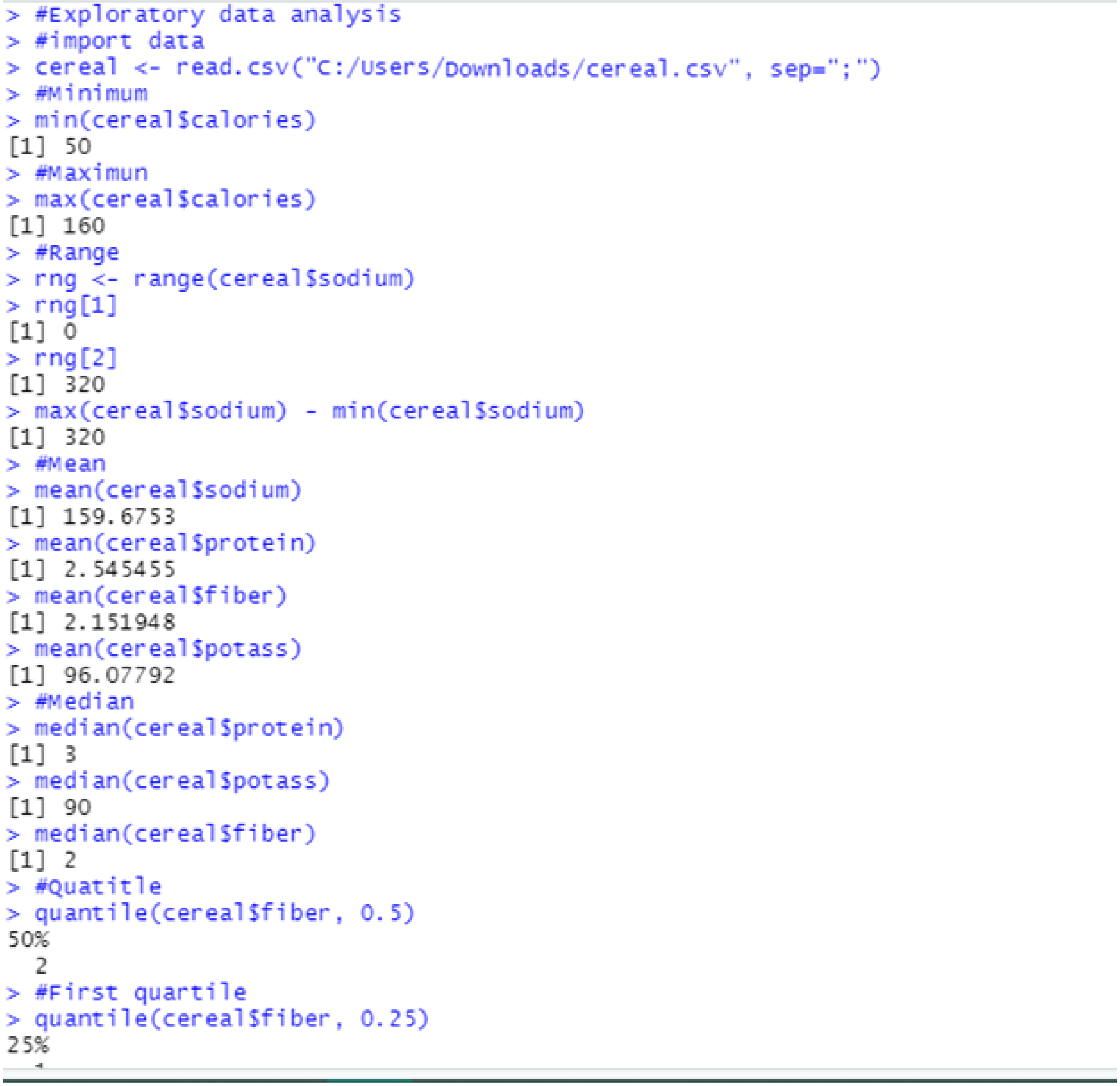
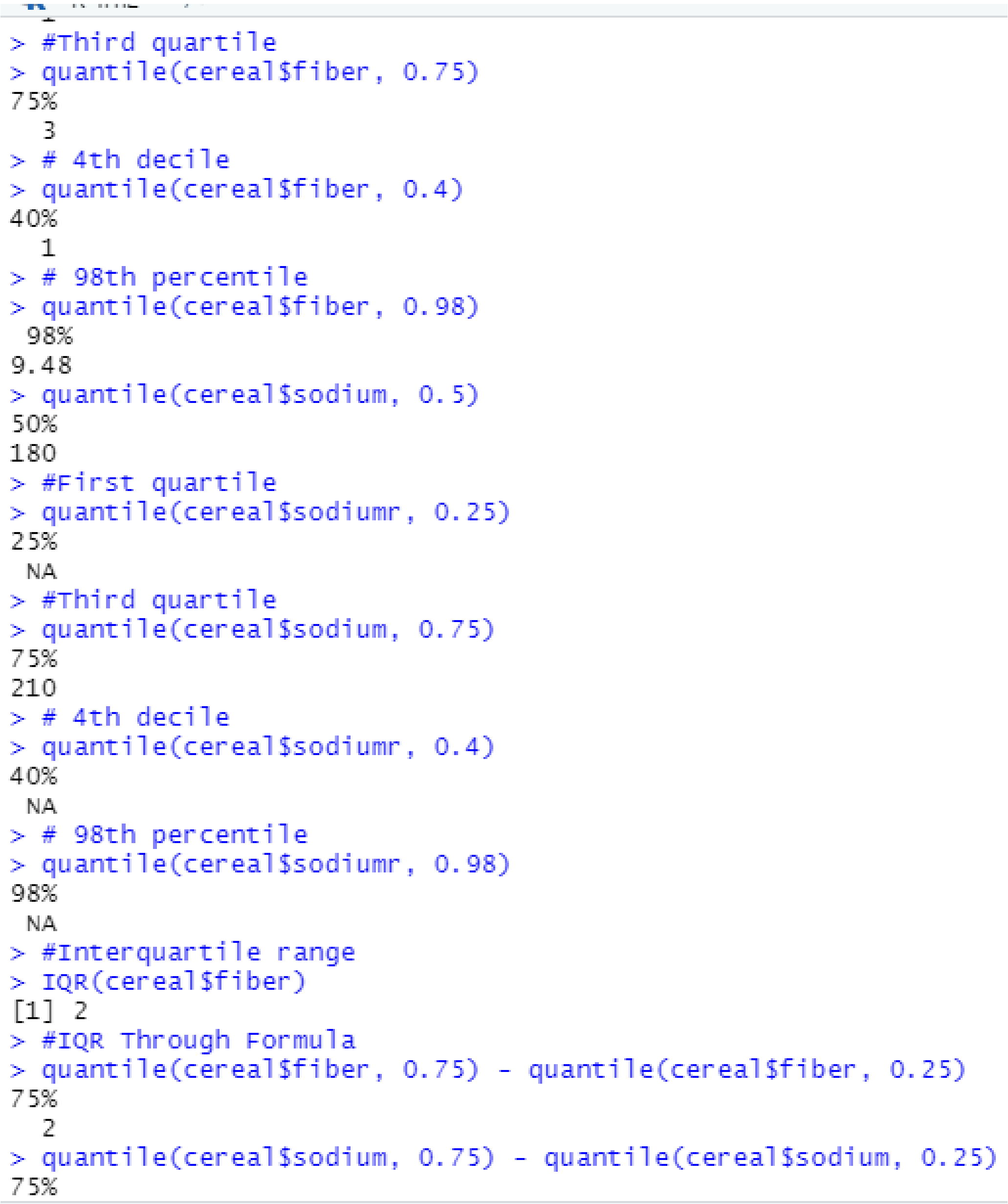
Name: Hrudyesh Salunke   Roll No: 18101B0042

Que 1:

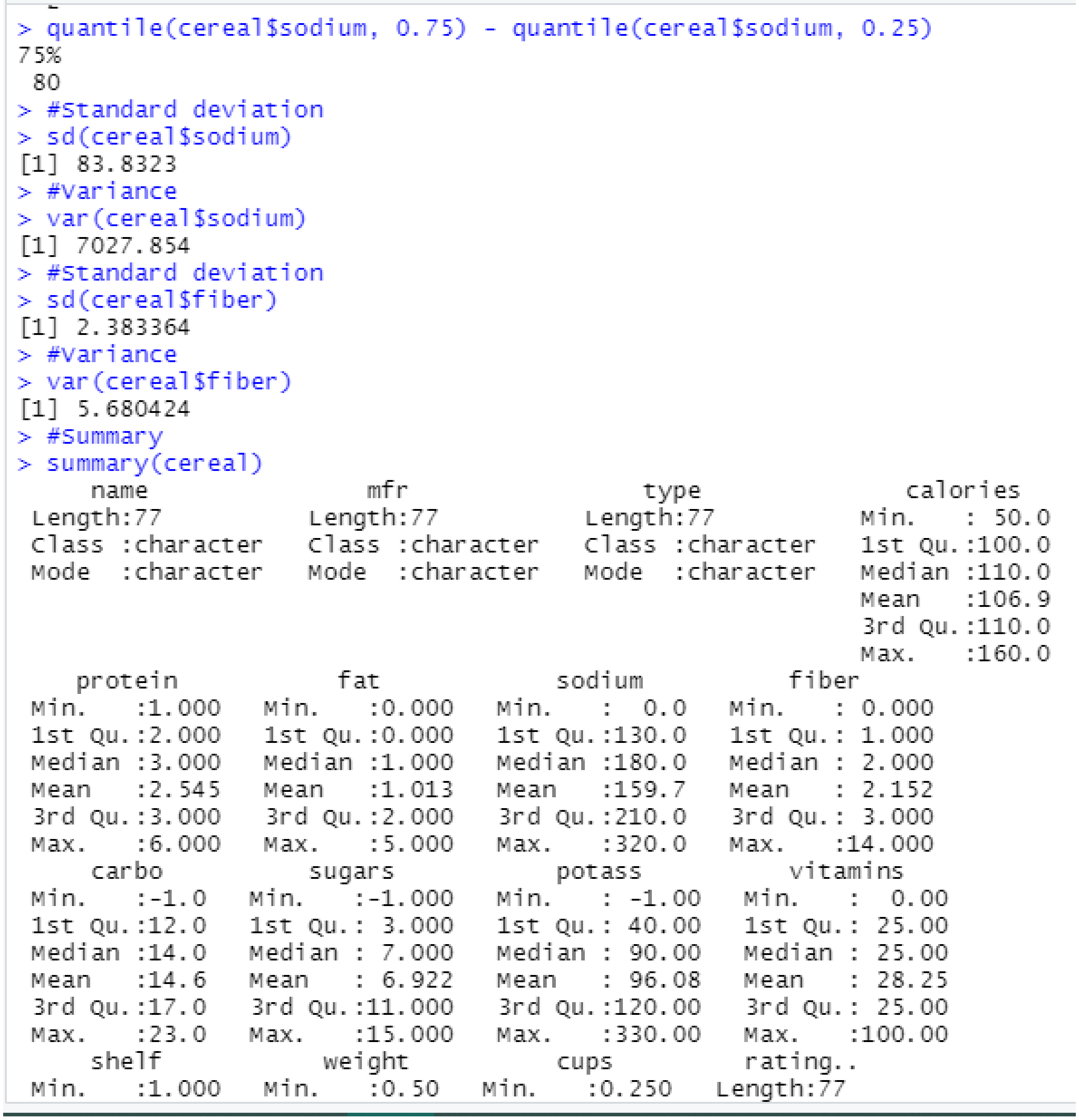
Write a R program to import a dataset (other than what you have taken for experiment submission) and perform exploratory data analysis in it.



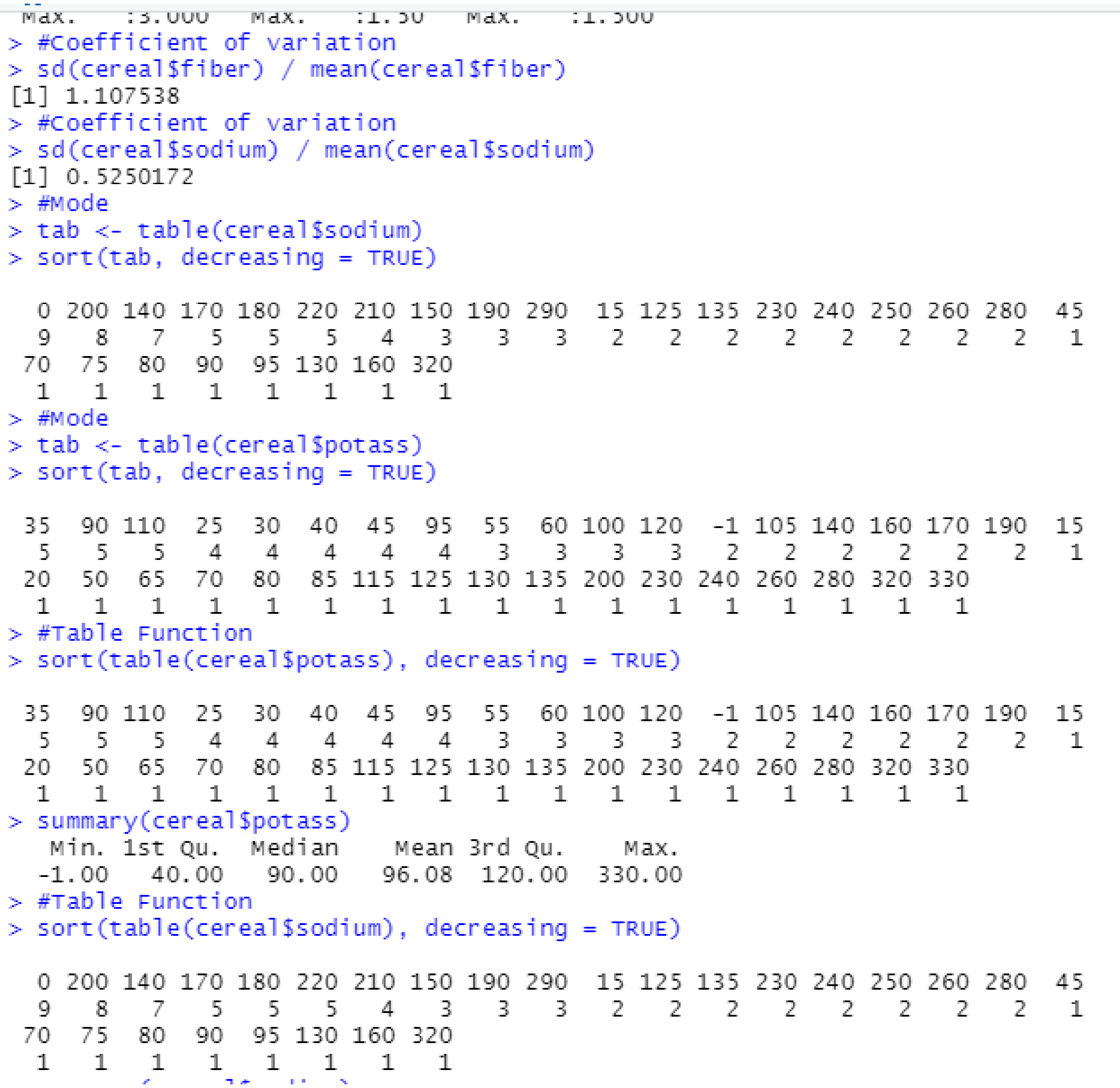
```
> #Exploratory data analysis
> #import data
> cereal <- read.csv("C:/Users/Downloads/cereal.csv", sep=";")
> #Minimum
> min(cereal$calories)
[1] 50
> #Maximun
> max(cereal$calories)
[1] 160
> #Range
> rng <- range(cereal$sodium)
> rng[1]
[1] 0
> rng[2]
[1] 320
> max(cereal$sodium) - min(cereal$sodium)
[1] 320
> #Mean
> mean(cereal$sodium)
[1] 159.6753
> mean(cereal$protein)
[1] 2.545455
> mean(cereal$fiber)
[1] 2.151948
> mean(cereal$potass)
[1] 96.07792
> #Median
> median(cereal$protein)
[1] 3
> median(cereal$potass)
[1] 90
> median(cereal$fiber)
[1] 2
> #Quatitle
> quantile(cereal$fiber, 0.5)
50% 
  2 
> #First quartile
> quantile(cereal$fiber, 0.25)
25% 
  1 
```



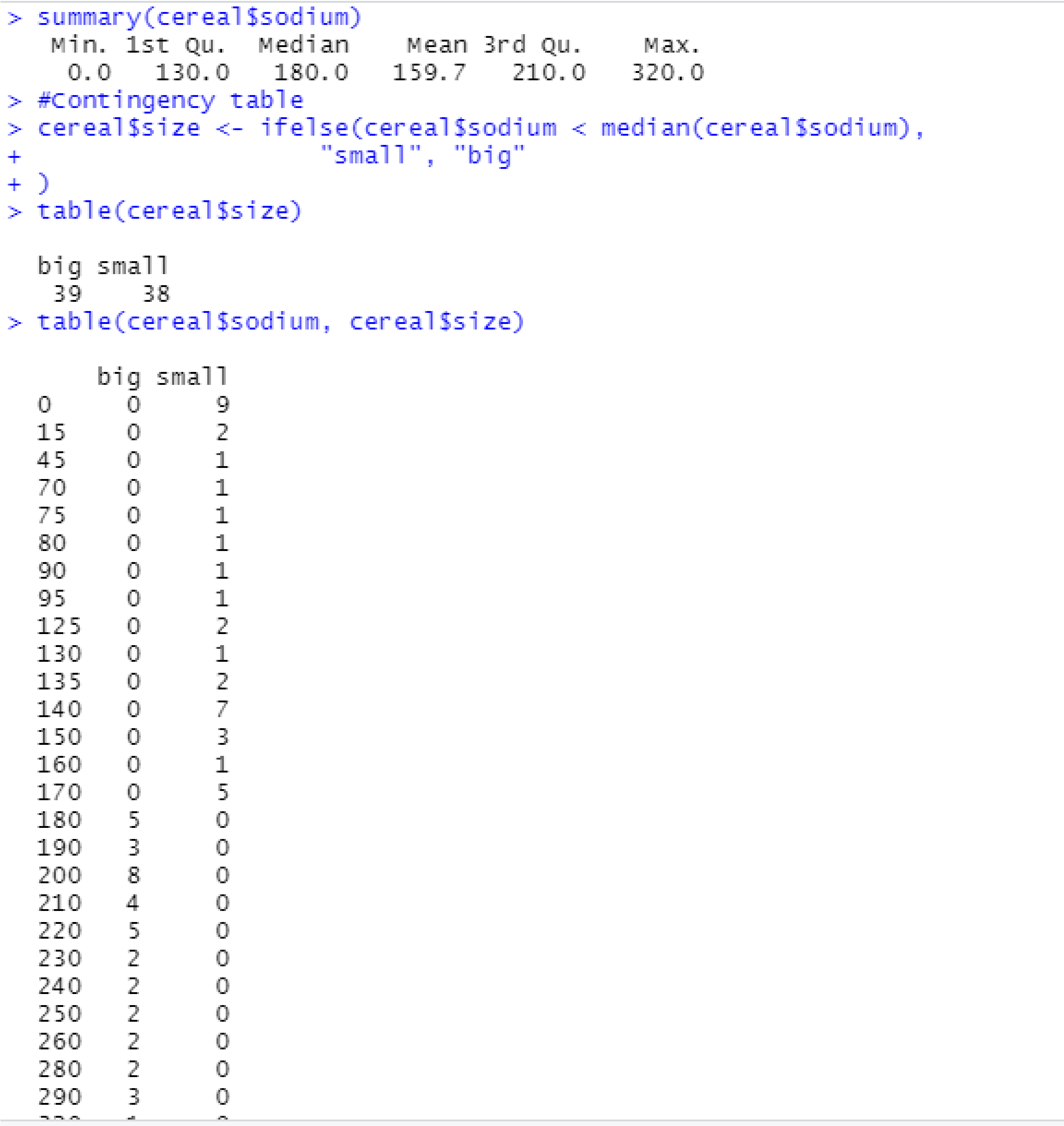
```
> #Third quartile
> quantile(cereal$fiber, 0.75)
75% 
  3 
> # 4th decile
> quantile(cereal$fiber, 0.4)
40% 
  1 
> # 98th percentile
> quantile(cereal$fiber, 0.98)
 98% 
9.48 
> quantile(cereal$sodium, 0.5)
50% 
180 
> #First quartile
> quantile(cereal$sodiumr, 0.25)
25% 
 NA 
> #Third quartile
> quantile(cereal$sodium, 0.75)
75% 
210 
> # 4th decile
> quantile(cereal$sodiumr, 0.4)
40% 
 NA 
> # 98th percentile
> quantile(cereal$sodiumr, 0.98)
98% 
 NA 
> #Interquartile range
> IQR(cereal$fiber)
[1] 2
> #IQR Through Formula
> quantile(cereal$fiber, 0.75) - quantile(cereal$fiber, 0.25)
75% 
  2 
> quantile(cereal$sodium, 0.75) - quantile(cereal$sodium, 0.25)
75% 
```



```
> quantile(cereal$sodium, 0.75) - quantile(cereal$sodium, 0.25)
75%
 80
> #Standard deviation
> sd(cereal$sodium)
[1] 83.8323
> #Variance
> var(cereal$sodium)
[1] 7027.854
> #Standard deviation
> sd(cereal$fiber)
[1] 2.383364
> #Variance
> var(cereal$fiber)
[1] 5.680424
> #Summary
> summary(cereal)
     name                mfr                type              calories    
 Length:77          Length:77          Length:77          Min.   : 50.0  
 Class :character   Class :character   Class :character   1st Qu.:100.0  
 Mode  :character   Mode  :character   Mode  :character   Median :110.0  
                                                          Mean   :106.9  
                                                          3rd Qu.:110.0  
                                                          Max.   :160.0  
    protein           fat            sodium          fiber       
 Min.   :1.000   Min.   :0.000   Min.   :  0.0   Min.   : 0.000  
 1st Qu.:2.000   1st Qu.:0.000   1st Qu.:130.0   1st Qu.: 1.000  
 Median :3.000   Median :1.000   Median :180.0   Median : 2.000  
 Mean   :2.545   Mean   :1.013   Mean   :159.7   Mean   : 2.152  
 3rd Qu.:3.000   3rd Qu.:2.000   3rd Qu.:210.0   3rd Qu.: 3.000  
 Max.   :6.000   Max.   :5.000   Max.   :320.0   Max.   :14.000  
     carbo          sugars           potass          vitamins     
 Min.   :-1.0   Min.   :-1.000   Min.   : -1.00   Min.   :  0.00  
 1st Qu.:12.0   1st Qu.: 3.000   1st Qu.: 40.00   1st Qu.: 25.00  
 Median :14.0   Median : 7.000   Median : 90.00   Median : 25.00  
 Mean   :14.6   Mean   : 6.922   Mean   : 96.08   Mean   : 28.25  
 3rd Qu.:17.0   3rd Qu.:11.000   3rd Qu.:120.00   3rd Qu.: 25.00  
 Max.   :23.0   Max.   :15.000   Max.   :330.00   Max.   :100.00  
     shelf           weight          cups          rating..        
 Min.   :1.000   Min.   :0.50   Min.   :0.250   Length:77          
```



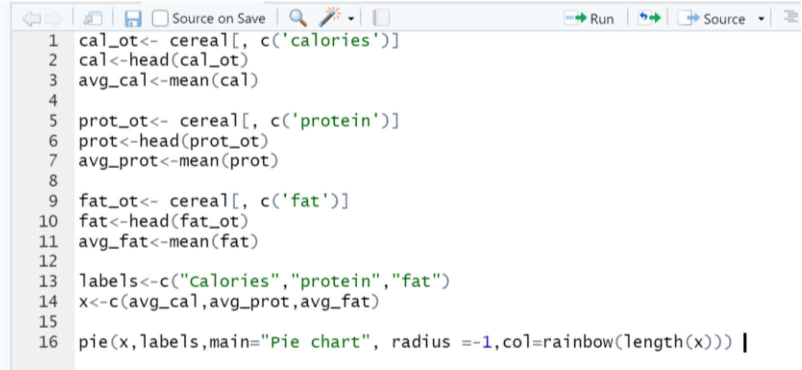
```
 Max.   :3.000   Max.   :1.50   Max.   :1.500  
> #Coefficient of variation
> sd(cereal$fiber) / mean(cereal$fiber)
[1] 1.107538
> #Coefficient of variation
> sd(cereal$sodium) / mean(cereal$sodium)
[1] 0.5250172
> #Mode
> tab <- table(cereal$sodium)
> sort(tab, decreasing = TRUE)

  0 200 140 170 180 220 210 150 190 290  15 125 135 230 240 250 260 280  45 
  9   8   7   5   5   5   4   3   3   3   2   2   2   2   2   2   2   2   1 
 70  75  80  90  95 130 160 320 
  1   1   1   1   1   1   1   1 
> #Mode
> tab <- table(cereal$potass)
> sort(tab, decreasing = TRUE)

 35  90 110  25  30  40  45  95  55  60 100 120  -1 105 140 160 170 190  15 
  5   5   5   4   4   4   4   4   3   3   3   3   2   2   2   2   2   2   1 
 20  50  65  70  80  85 115 125 130 135 200 230 240 260 280 320 330 
  1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1 
> #Table Function
> sort(table(cereal$potass), decreasing = TRUE)

 35  90 110  25  30  40  45  95  55  60 100 120  -1 105 140 160 170 190  15 
  5   5   5   4   4   4   4   4   3   3   3   3   2   2   2   2   2   2   1 
 20  50  65  70  80  85 115 125 130 135 200 230 240 260 280 320 330 
  1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1 
> summary(cereal$potass)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  -1.00   40.00   90.00   96.08  120.00  330.00 
> #Table Function
> sort(table(cereal$sodium), decreasing = TRUE)

  0 200 140 170 180 220 210 150 190 290  15 125 135 230 240 250 260 280  45 
  9   8   7   5   5   5   4   3   3   3   2   2   2   2   2   2   2   2   1 
 70  75  80  90  95 130 160 320 
  1   1   1   1   1   1   1   1 
```



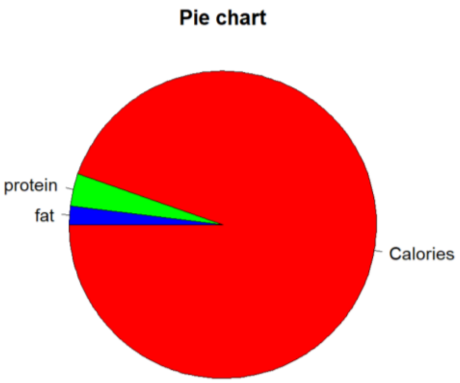
```
> summary(cereal$sodium)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
    0.0   130.0   180.0   159.7   210.0   320.0 
> #Contingency table
> cereal$size <- ifelse(cereal$sodium < median(cereal$sodium),
+                       "small", "big"
+ )
> table(cereal$size)

  big small 
   39    38 
> table(cereal$sodium, cereal$size)
     
      big small
  0     0     9
  15    0     2
  45    0     1
  70    0     1
  75    0     1
  80    0     1
  90    0     1
  95    0     1
  125   0     2
  130   0     1
  135   0     2
  140   0     7
  150   0     3
  160   0     1
  170   0     5
  180   5     0
  190   3     0
  200   8     0
  210   4     0
  220   5     0
  230   2     0
  240   2     0
  250   2     0
  260   2     0
  280   2     0
  290   3     0
```

Que 2: Write a R program to make various plots (data set same as in Que 1).

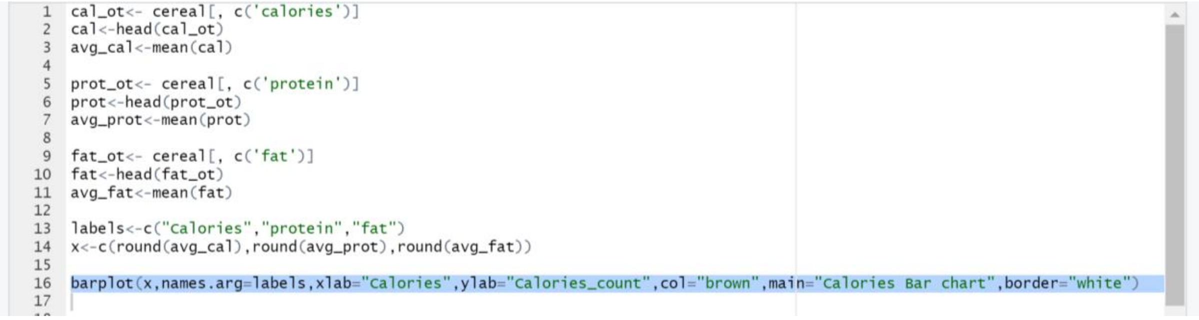
Pie Chart:



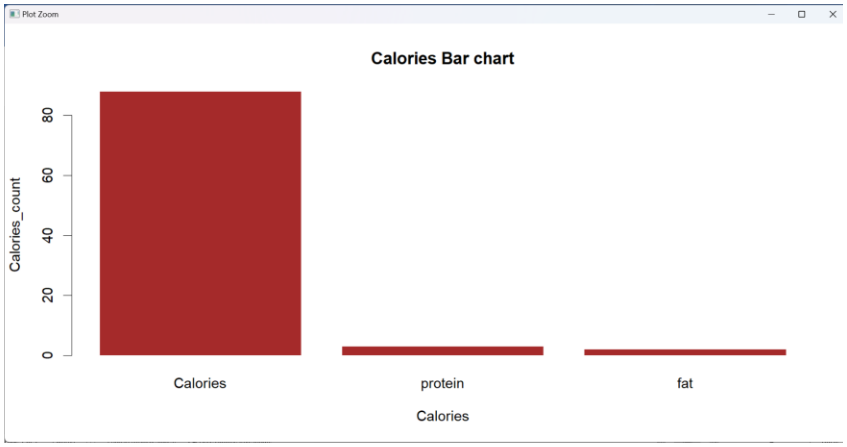
```
cal_ot<- cereal[, c('calories')]
cal<-head(cal_ot)
avg_cal<-mean(cal)

prot_ot<- cereal[, c('protein')]
prot<-head(prot_ot)
avg_prot<-mean(prot)

fat_ot<- cereal[, c('fat')]
fat<-head(fat_ot)
avg_fat<-mean(fat)

labels<-c("Calories","protein","fat")
x<-c(avg_cal,avg_prot,avg_fat)

pie(x,labels,main="Pie chart", radius =-1,col=rainbow(length(x)))
```



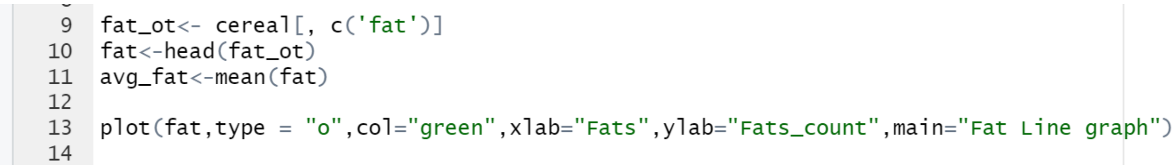
Barplot:



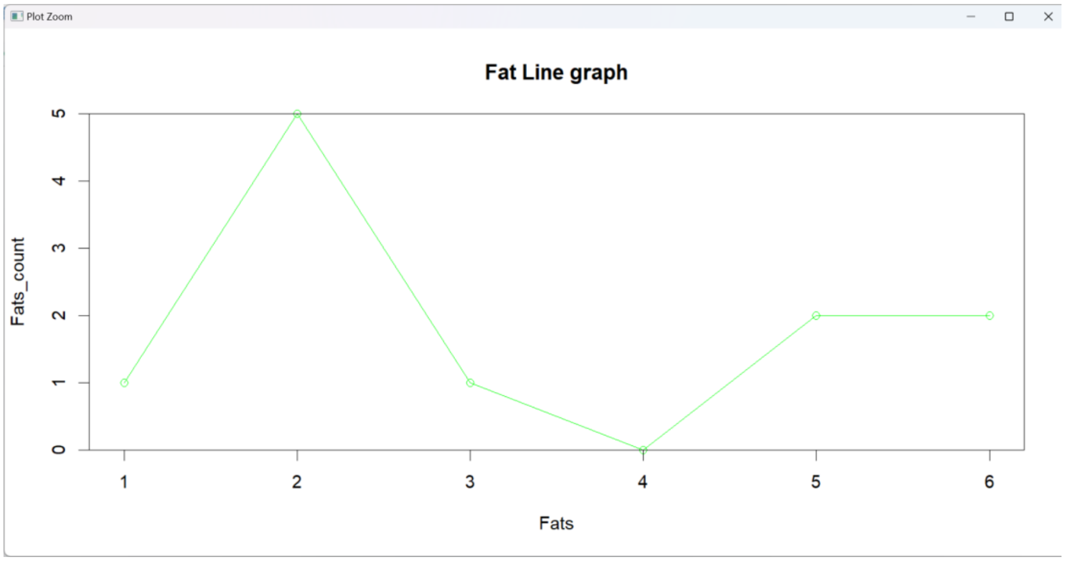
```
cal_ot<- cereal[, c('calories')]
cal<-head(cal_ot)
avg_cal<-mean(cal)

prot_ot<- cereal[, c('protein')]
prot<-head(prot_ot)
avg_prot<-mean(prot)

fat_ot<- cereal[, c('fat')]
fat<-head(fat_ot)
avg_fat<-mean(fat)

labels<-c("Calories","protein","fat")
x<-c(round(avg_cal),round(avg_prot),round(avg_fat))

barplot(x,names.arg=labels,xlab="Calories",ylab="Calories_count",col="brown",main="Calories Bar chart",border="white")
```



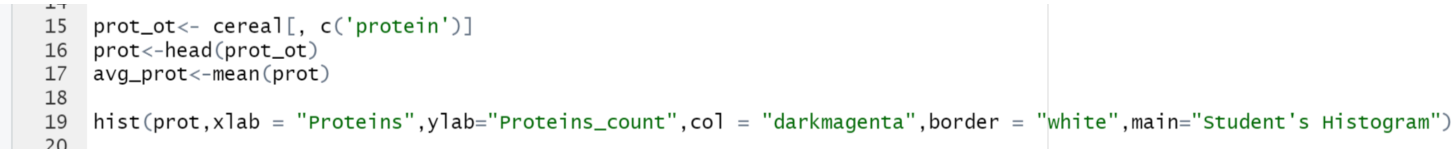
Line Graph:

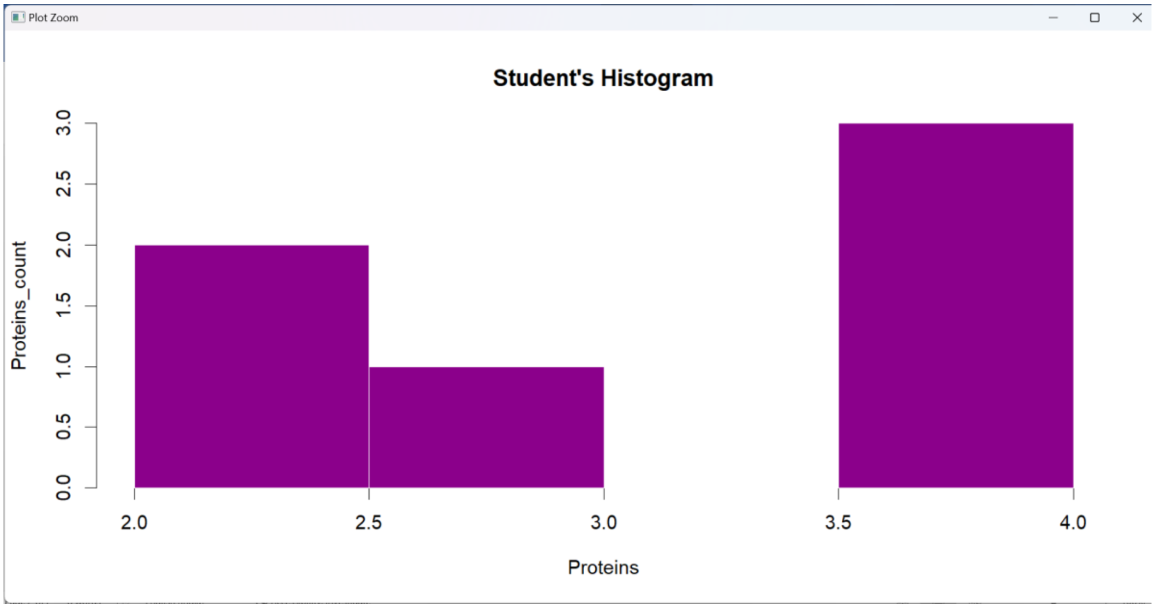


```
fat_ot<- cereal[, c('fat')]
fat<-head(fat_ot)
avg_fat<-mean(fat)

plot(fat,type = "o",col="green",xlab="Fats",ylab="Fats_count",main="Fat Line graph")
```

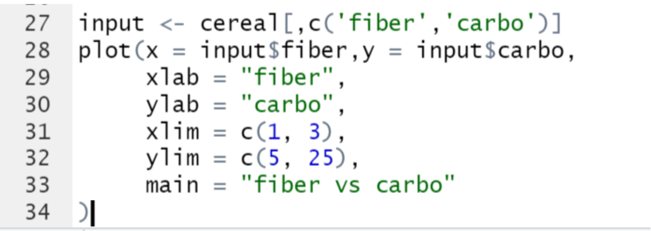


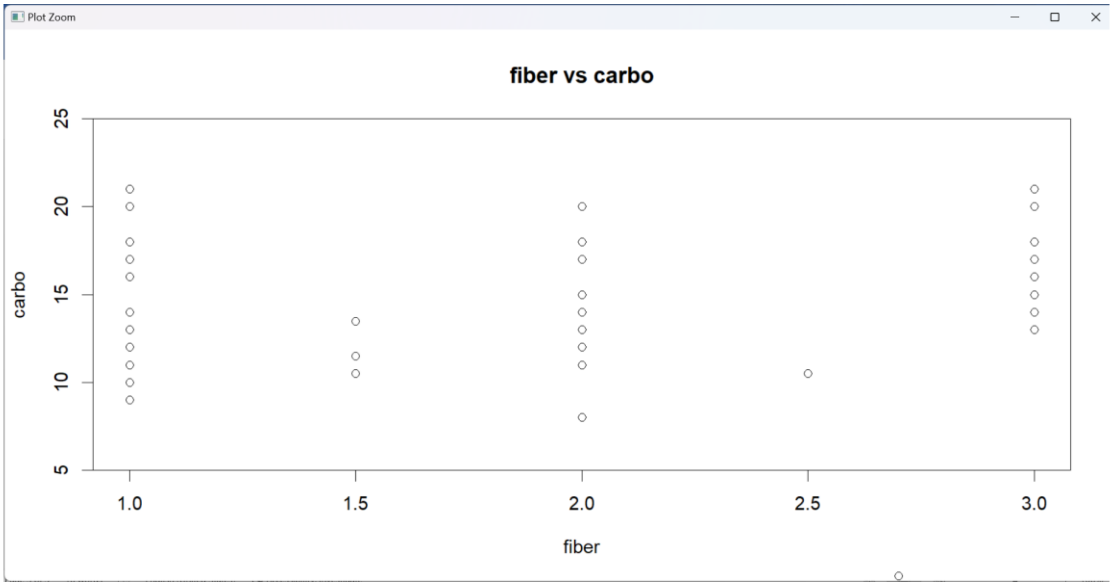
Histogram:

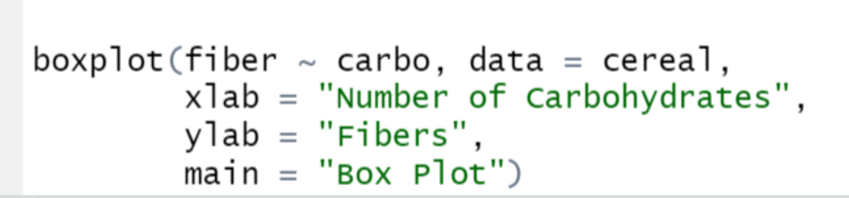

```
prot_ot<- cereal[, c('protein')]
prot<-head(prot_ot)
avg_prot<-mean(prot)

hist(prot,xlab = "Proteins",ylab="Proteins_count",col = "darkmagenta",border = "white",main="Student's Histogram")
```



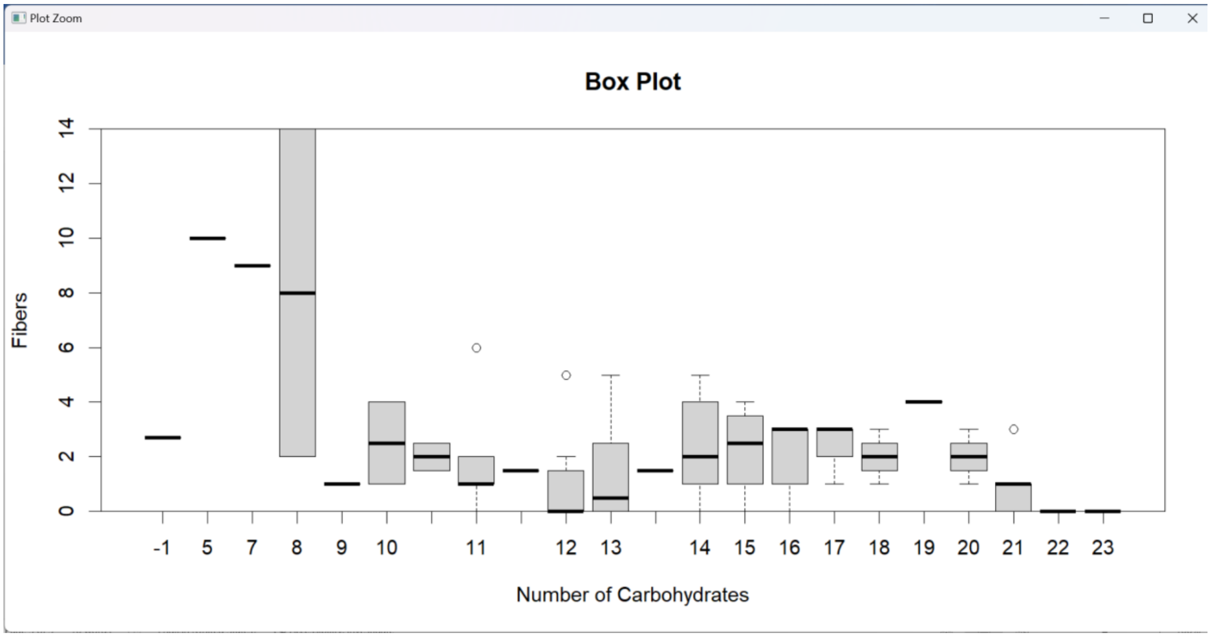
Scatter Plot:


```
input <- cereal[,c('fiber','carbo')]
plot(x = input$fiber,y = input$carbo,
      xlab = "fiber",
      ylab = "carbo",
      xlim = c(1, 3),
      ylim = c(5, 25),
      main = "fiber vs carbo"
)
```



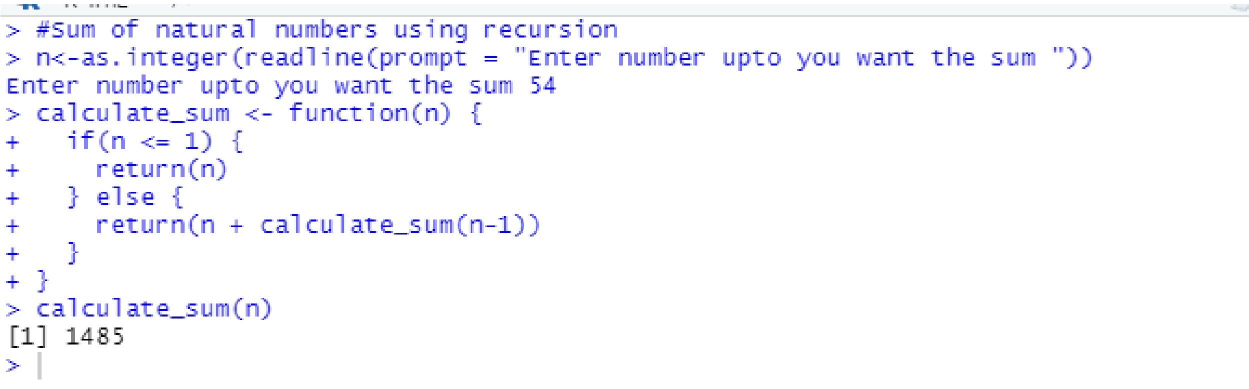


```
boxplot(fiber ~ carbo, data = cereal,
        xlab = "Number of Carbohydrates",
        ylab = "Fibers",
        main = "Box Plot")
```



Que 3: Write a R program to find the sum of natural numbers using recursion.

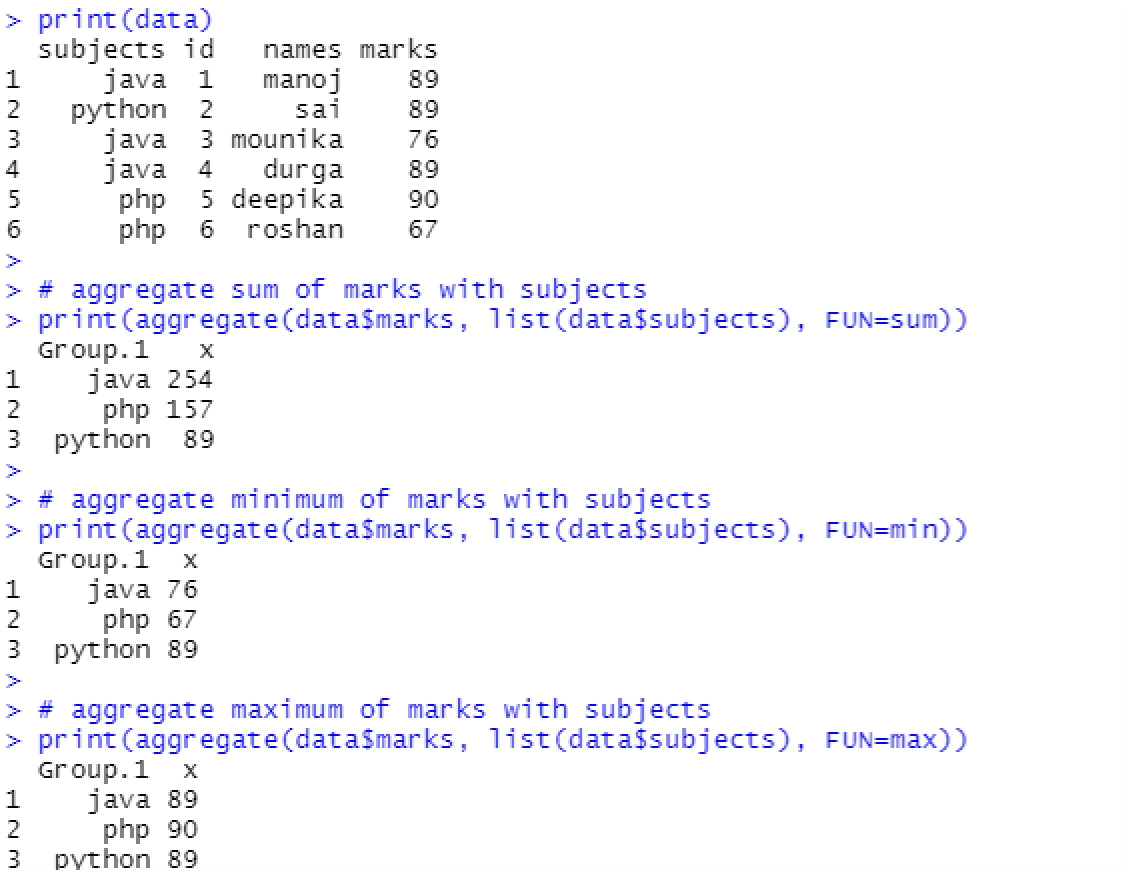
```
#Sum of natural numbers using recursion
n<-as.integer(readline(prompt = "Enter number upto you want the sum "))
calculate_sum <- function(n) {
 if(n <= 1) {
   return(n)
 } else {
   return(n + calculate_sum(n-1))
 }
}
calculate_sum(n)
```



```
> #Sum of natural numbers using recursion
> n<-as.integer(readline(prompt = "Enter number upto you want the sum "))
Enter number upto you want the sum 54
> calculate_sum <- function(n) {
+   if(n <= 1) {
+     return(n)
+   } else {
+     return(n + calculate_sum(n-1))
+   }
+ }
> calculate_sum(n)
[1] 1485
> 
```

Que 4:Write a R program demonstrating the use of aggregate function in R.

```
#aggregate function
# create a dataframe with 4 columns
data = data.frame(subjects=c("java", "python", "java",
                           "java", "php", "php"),
               id=c(1, 2, 3, 4, 5, 6),
               names=c("manoj", "sai", "mounika",
                        "durga", "deepika", "roshan"),
               marks=c(89, 89, 76, 89, 90, 67))

# display
print(data)

# aggregate sum of marks with subjects
print(aggregate(data$marks, list(data$subjects), FUN=sum))

# aggregate minimum of marks with subjects
print(aggregate(data$marks, list(data$subjects), FUN=min))

# aggregate maximum of marks with subjects
print(aggregate(data$marks, list(data$subjects), FUN=max))
```



```
> print(data)
  subjects id   names marks
1     java  1   manoj    89
2   python  2     sai    89
3     java  3 mounika    76
4     java  4   durga    89
5      php  5 deepika    90
6      php  6  roshan    67
>
> # aggregate sum of marks with subjects
> print(aggregate(data$marks, list(data$subjects), FUN=sum))
  Group.1   x
1    java 254
2     php 157
3  python  89
>
> # aggregate minimum of marks with subjects
> print(aggregate(data$marks, list(data$subjects), FUN=min))
  Group.1  x
1    java 76
2     php 67
3  python 89
>
> # aggregate maximum of marks with subjects
> print(aggregate(data$marks, list(data$subjects), FUN=max))
  Group.1  x
1    java 89
2     php 90
3  python 89
```