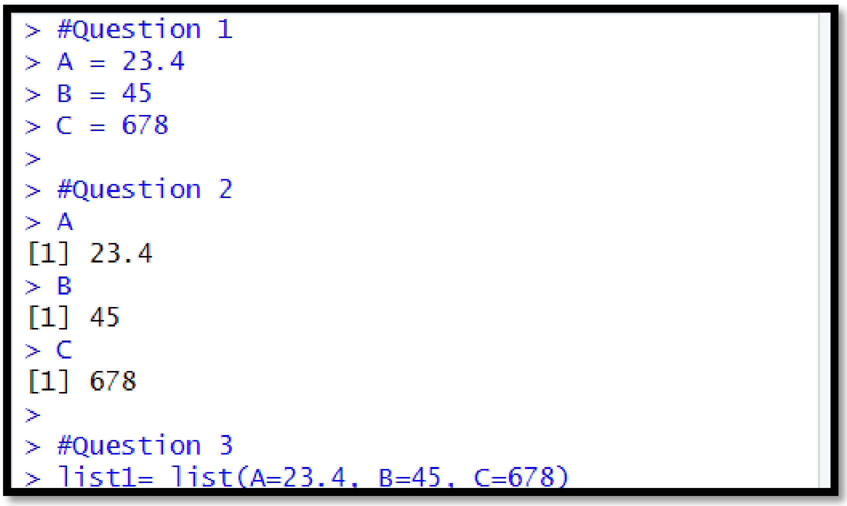
LAB ASSIGNMENT 1

Submitted by: Kamya Mehra

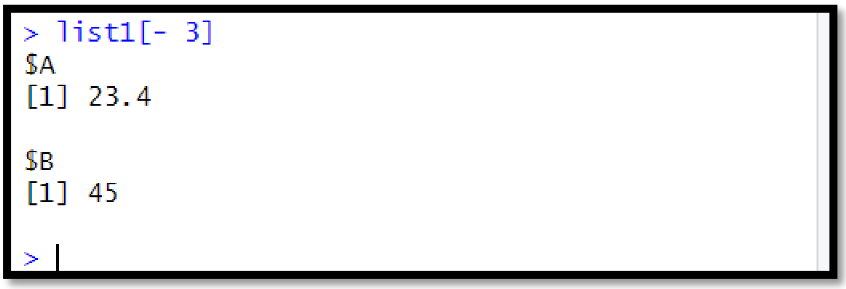
102213026

3CO35

QUESTION 1, 2, 3 :-

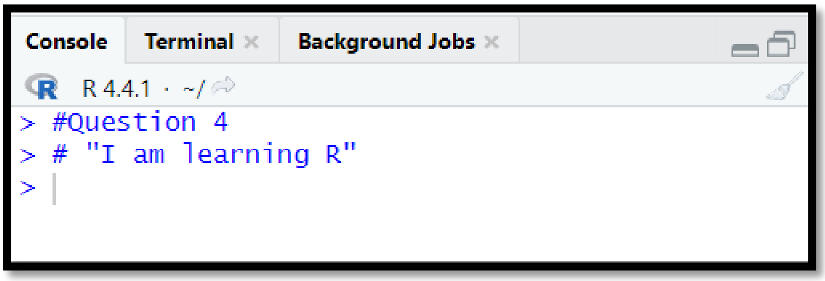


```
> #Question 1
> A = 23.4
> B = 45
> C = 678
>
> #Question 2
> A
[1] 23.4
> B
[1] 45
> C
[1] 678
>
> #Question 3
> list1= list(A=23.4, B=45, C=678)
```



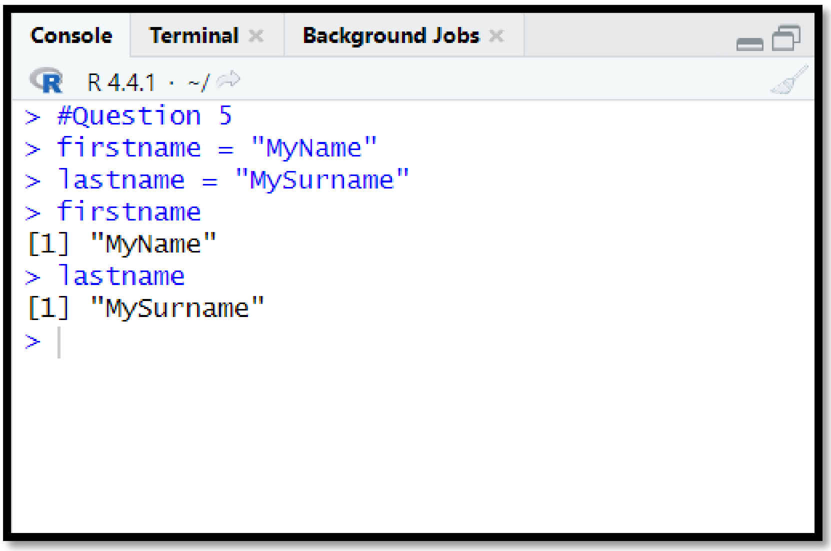
```
> list1[- 3]
$A
[1] 23.4

$B
[1] 45

>
```

QUESTION 4 :-



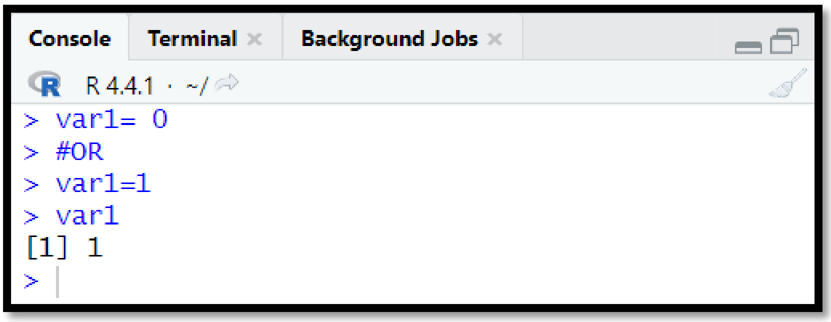
```
> #Question 4
> # "I am learning R"
>
```

QUESTION 5 :-



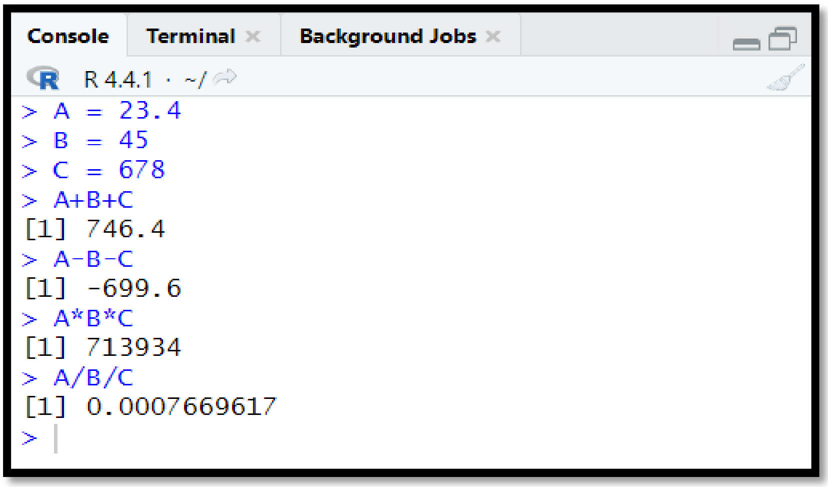
```
> #Question 5
> firstname = "MyName"
> lastname = "MySurname"
> firstname
[1] "MyName"
> lastname
[1] "MySurname"
> 
```

QUESTION 6 :-



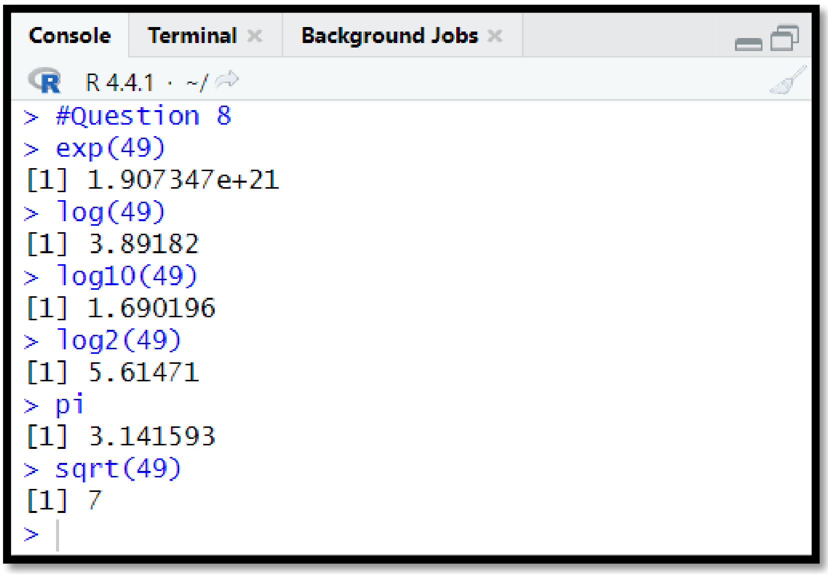
```
> var1= 0
> #OR
> var1=1
> var1
[1] 1
> 
```

QUESTION 7 :-



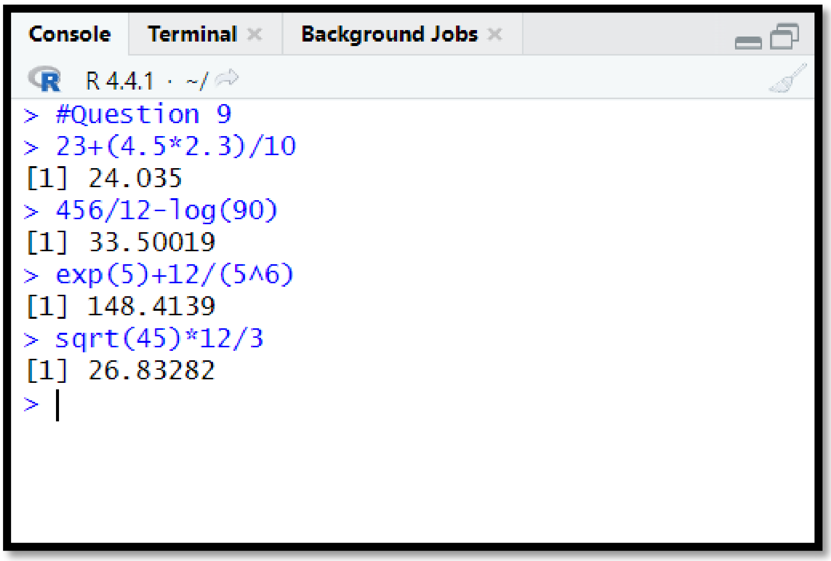
```
> A = 23.4
> B = 45
> C = 678
> A+B+C
[1] 746.4
> A-B-C
[1] -699.6
> A*B*C
[1] 713934
> A/B/C
[1] 0.0007669617
> 
```

QUESTION 8 :-



```
> #Question 8
> exp(49)
[1] 1.907347e+21
> log(49)
[1] 3.89182
> log10(49)
[1] 1.690196
> log2(49)
[1] 5.61471
> pi
[1] 3.141593
> sqrt(49)
[1] 7
> 
```

QUESTION 9 :-

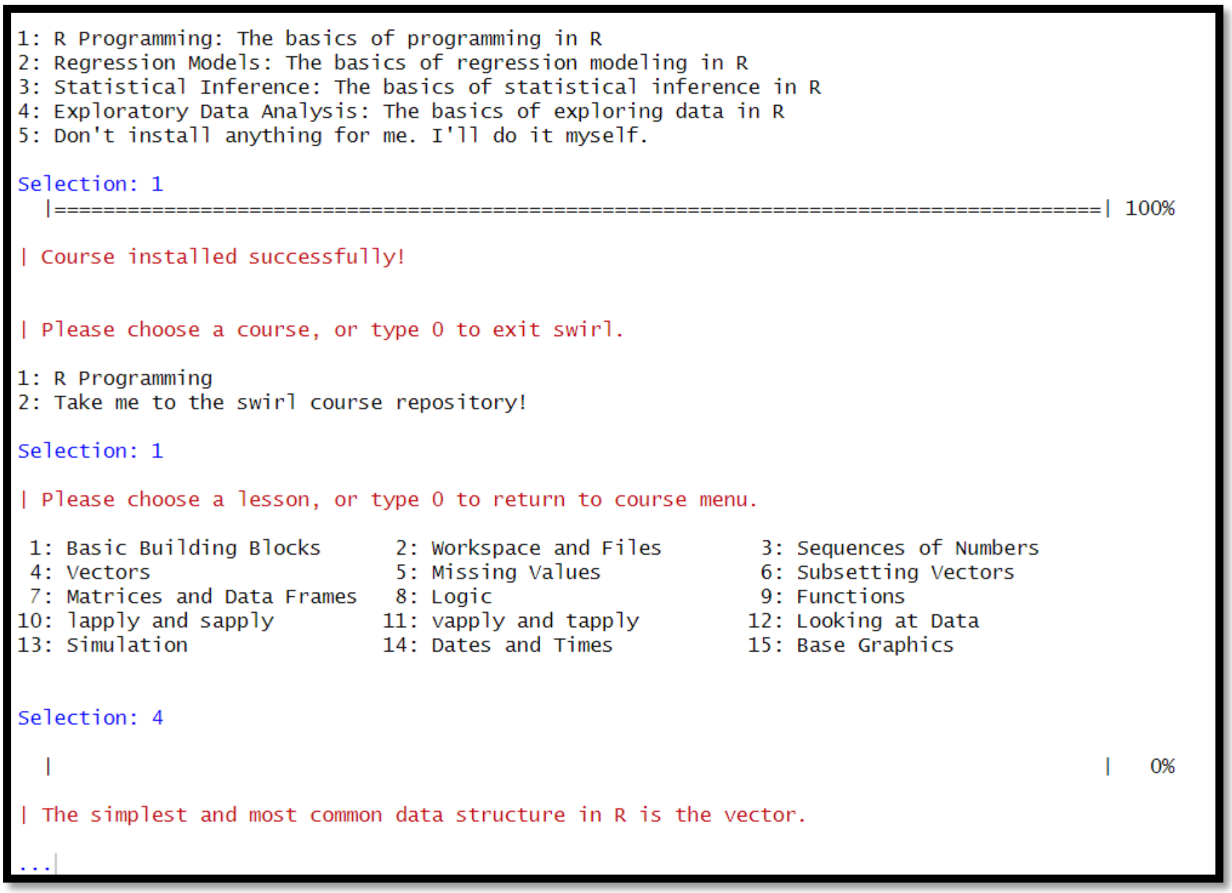


```
> #Question 9
> 23+(4.5*2.3)/10
[1] 24.035
> 456/12-log(90)
[1] 33.50019
> exp(5)+12/(5^6)
[1] 148.4139
> sqrt(45)*12/3
[1] 26.83282
> 
```

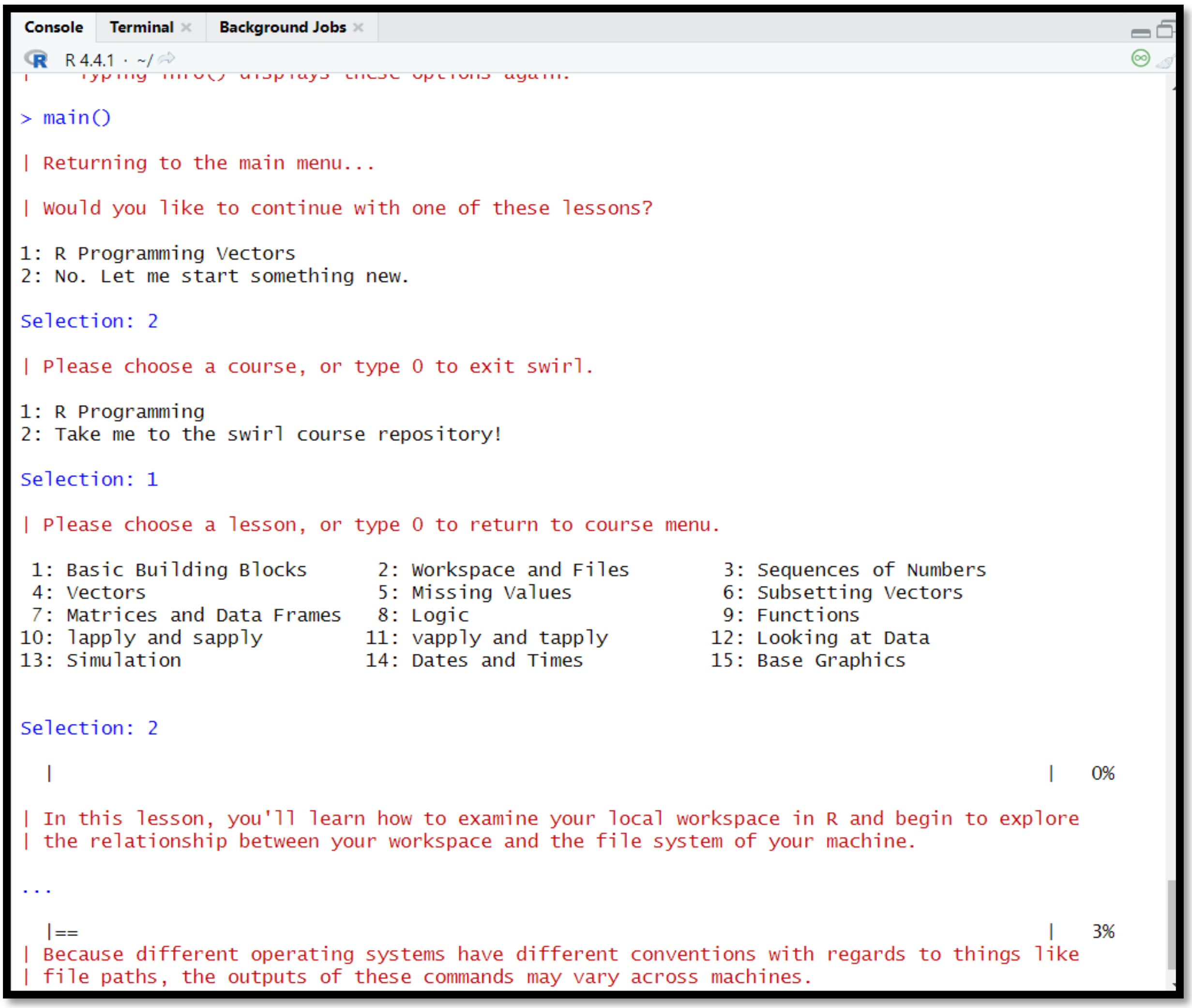
LAB ASSIGNMENT 2

Submitted By: Kamya Mehra 102213026 3CO35

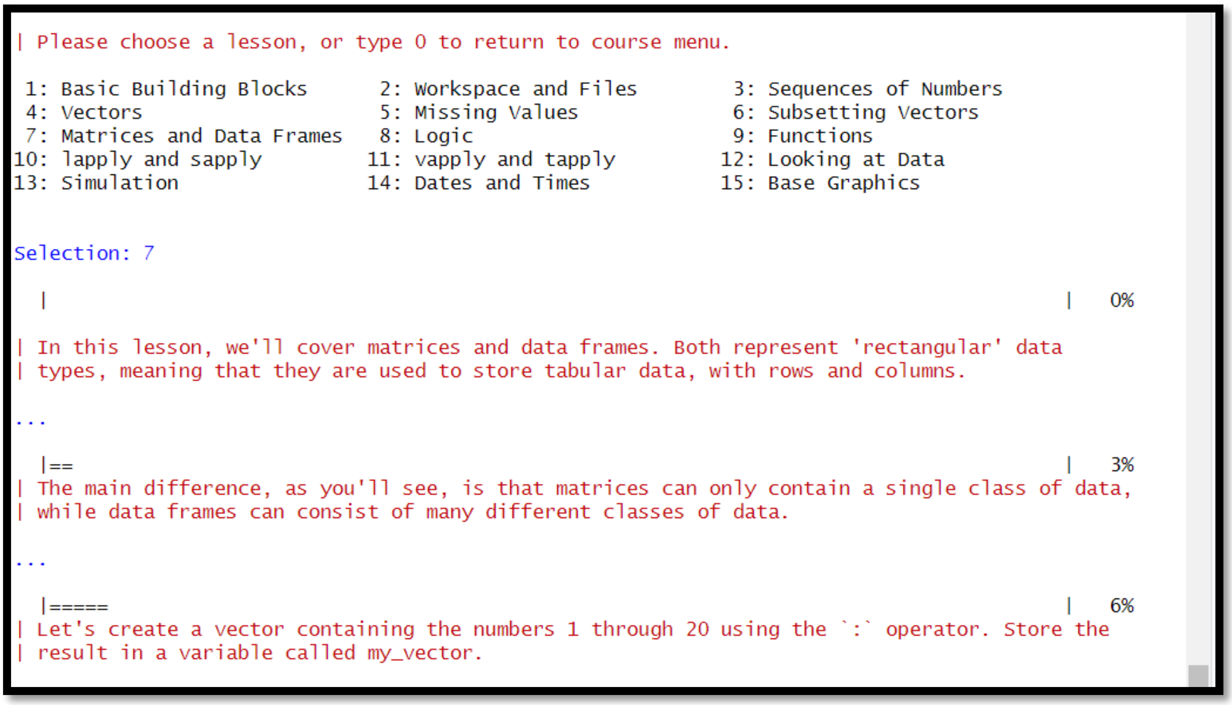
Here are some snippets of the output produced while working in the swirl library: -



```
1: R Programming: The basics of programming in R
2: Regression Models: The basics of regression modeling in R
3: Statistical Inference: The basics of statistical inference in R
4: Exploratory Data Analysis: The basics of exploring data in R
5: Don't install anything for me. I'll do it myself.

Selection: 1
  |=======================================================================================| 100%

| Course installed successfully!


| Please choose a course, or type 0 to exit swirl.

1: R Programming
2: Take me to the swirl course repository!

Selection: 1

| Please choose a lesson, or type 0 to return to course menu.

 1: Basic Building Blocks      2: Workspace and Files        3: Sequences of Numbers
 4: Vectors                    5: Missing Values             6: Subsetting Vectors
 7: Matrices and Data Frames   8: Logic                      9: Functions
10: lapply and sapply         11: vapply and tapply         12: Looking at Data
13: Simulation                14: Dates and Times           15: Base Graphics


Selection: 4

  |                                                                                       |   0%

| The simplest and most common data structure in R is the vector.

...
```



```
|   Typing info() displays these options again.

> main()

| Returning to the main menu...

| Would you like to continue with one of these lessons?

1: R Programming Vectors
2: No. Let me start something new.

Selection: 2

| Please choose a course, or type 0 to exit swirl.

1: R Programming
2: Take me to the swirl course repository!

Selection: 1

| Please choose a lesson, or type 0 to return to course menu.

 1: Basic Building Blocks      2: Workspace and Files        3: Sequences of Numbers
 4: Vectors                    5: Missing Values             6: Subsetting Vectors
 7: Matrices and Data Frames   8: Logic                      9: Functions
10: lapply and sapply         11: vapply and tapply         12: Looking at Data
13: Simulation                14: Dates and Times           15: Base Graphics


Selection: 2

  |                                                                                |   0%

| In this lesson, you'll learn how to examine your local workspace in R and begin to explore
| the relationship between your workspace and the file system of your machine.

...

  |==                                                                              |   3%
| Because different operating systems have different conventions with regards to things like
| file paths, the outputs of these commands may vary across machines.
```



```
| Please choose a lesson, or type 0 to return to course menu.

 1: Basic Building Blocks      2: Workspace and Files        3: Sequences of Numbers
 4: Vectors                    5: Missing Values             6: Subsetting Vectors
 7: Matrices and Data Frames   8: Logic                      9: Functions
10: lapply and sapply         11: vapply and tapply         12: Looking at Data
13: Simulation                14: Dates and Times           15: Base Graphics


Selection: 7

  |                                                                                |   0%

| In this lesson, we'll cover matrices and data frames. Both represent 'rectangular' data
| types, meaning that they are used to store tabular data, with rows and columns.

...

  |==                                                                              |   3%
| The main difference, as you'll see, is that matrices can only contain a single class of data,
| while data frames can consist of many different classes of data.

...

  |=====                                                                           |   6%
| Let's create a vector containing the numbers 1 through 20 using the `:` operator. Store the
| result in a variable called my_vector.
```

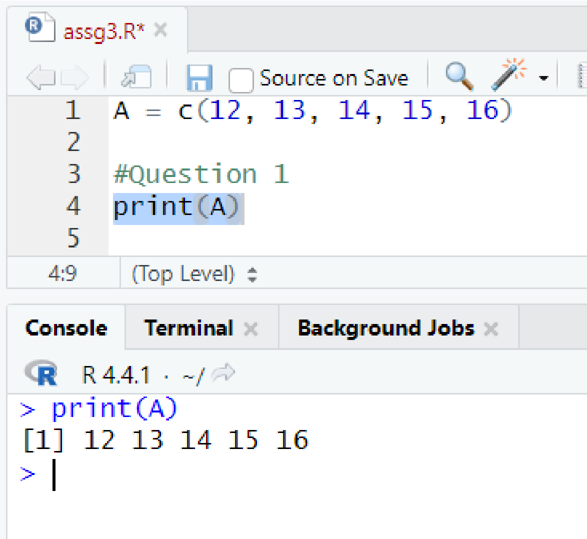
LAB ASSIGNMENT 3

Submitted By: Kamya Mehra 102213026   3CO35

A = c(12,13,14,15,16)

This line needs to be executed with each code snippet as “A” is being used

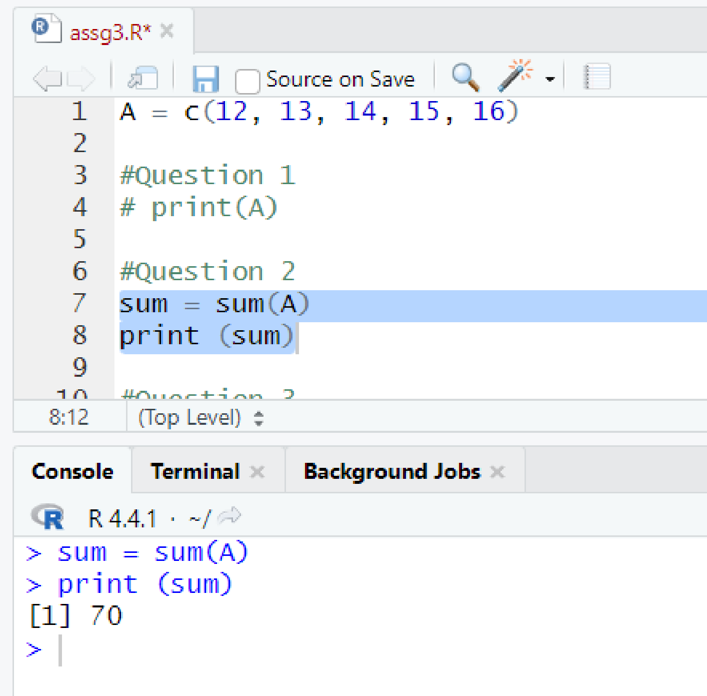
Question 1



```r
A = c(12, 13, 14, 15, 16)

#Question 1
print(A)

```

```
> print(A)
[1] 12 13 14 15 16
> 
```

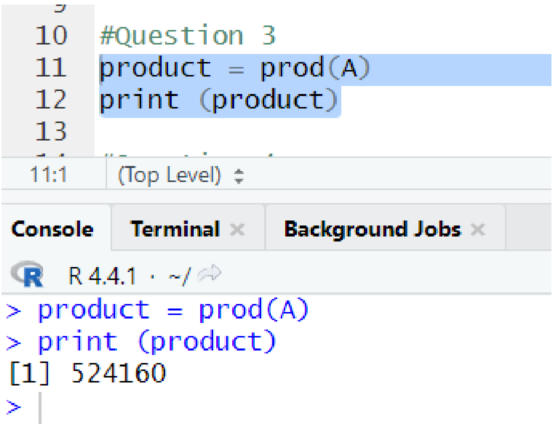
Question 2



```r
A = c(12, 13, 14, 15, 16)

#Question 1
# print(A)

#Question 2
sum = sum(A)
print (sum)

#Question 3
```

```
> sum = sum(A)
> print (sum)
[1] 70
> 
```

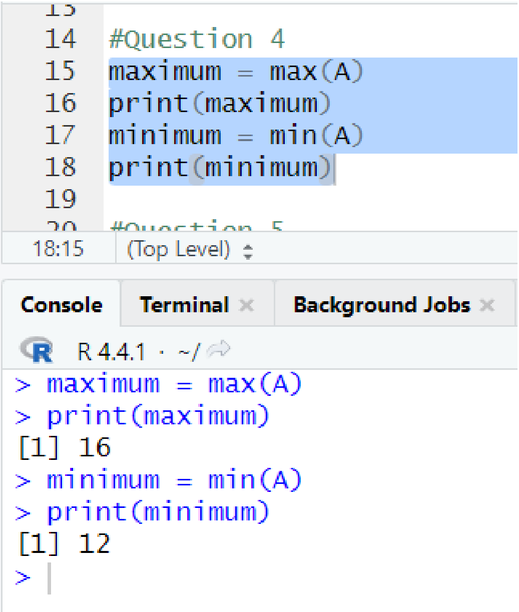
Question 3



```
#Question 3
product = prod(A)
print (product)
```

```
> product = prod(A)
> print (product)
[1] 524160
> 
```

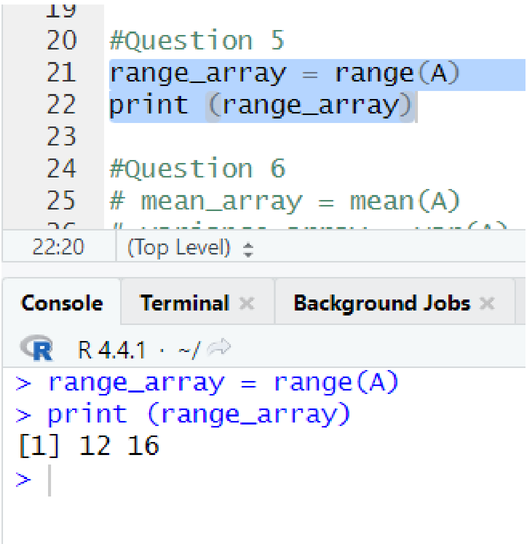
Question 4



```
#Question 4
maximum = max(A)
print(maximum)
minimum = min(A)
print(minimum)
```

```
> maximum = max(A)
> print(maximum)
[1] 16
> minimum = min(A)
> print(minimum)
[1] 12
> 
```

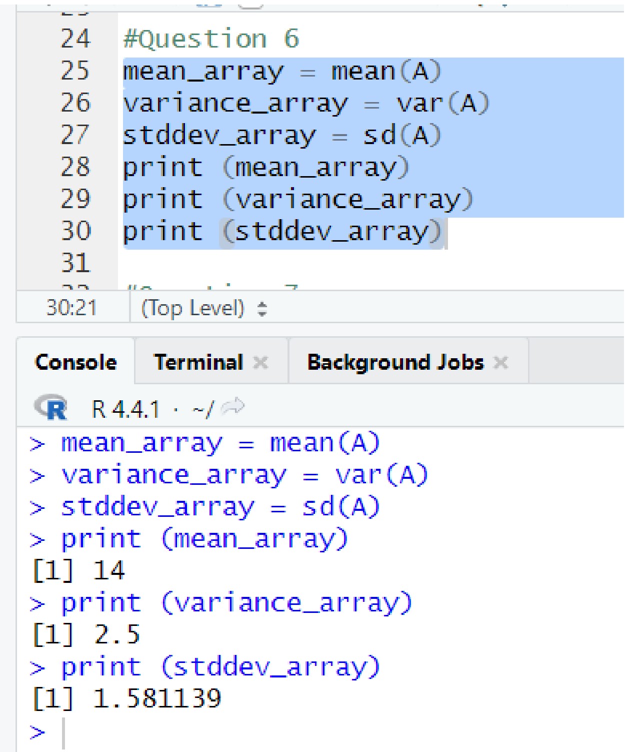
Question 5



```
#Question 5
range_array = range(A)
print (range_array)

#Question 6
# mean_array = mean(A)
```

```
> range_array = range(A)
> print (range_array)
[1] 12 16
> 
```

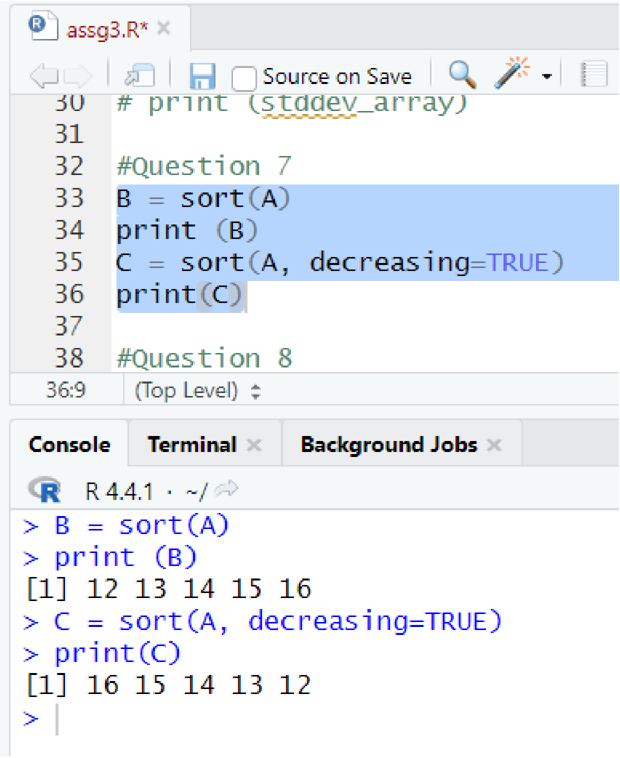
Question 6



```
#Question 6
mean_array = mean(A)
variance_array = var(A)
stddev_array = sd(A)
print (mean_array)
print (variance_array)
print (stddev_array)
```

```
> mean_array = mean(A)
> variance_array = var(A)
> stddev_array = sd(A)
> print (mean_array)
[1] 14
> print (variance_array)
[1] 2.5
> print (stddev_array)
[1] 1.581139
> 
```

Question 7

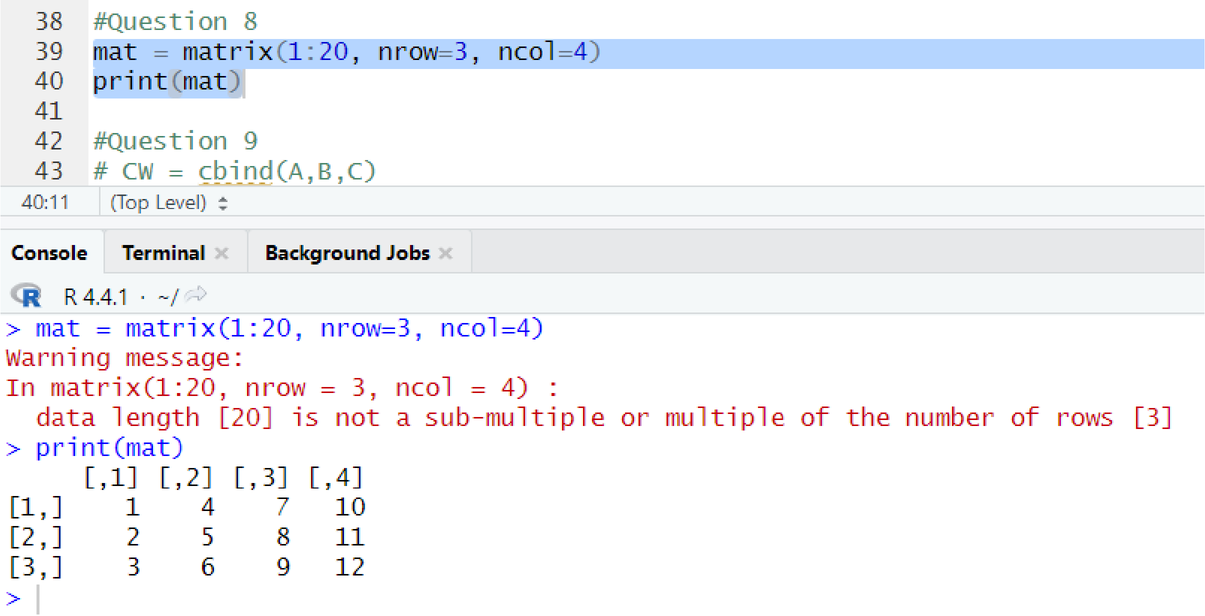


```
# print (stddev_array)

#Question 7
B = sort(A)
print (B)
C = sort(A, decreasing=TRUE)
print(C)

#Question 8
```

```
> B = sort(A)
> print (B)
[1] 12 13 14 15 16
> C = sort(A, decreasing=TRUE)
> print(C)
[1] 16 15 14 13 12
> 
```

Question 8

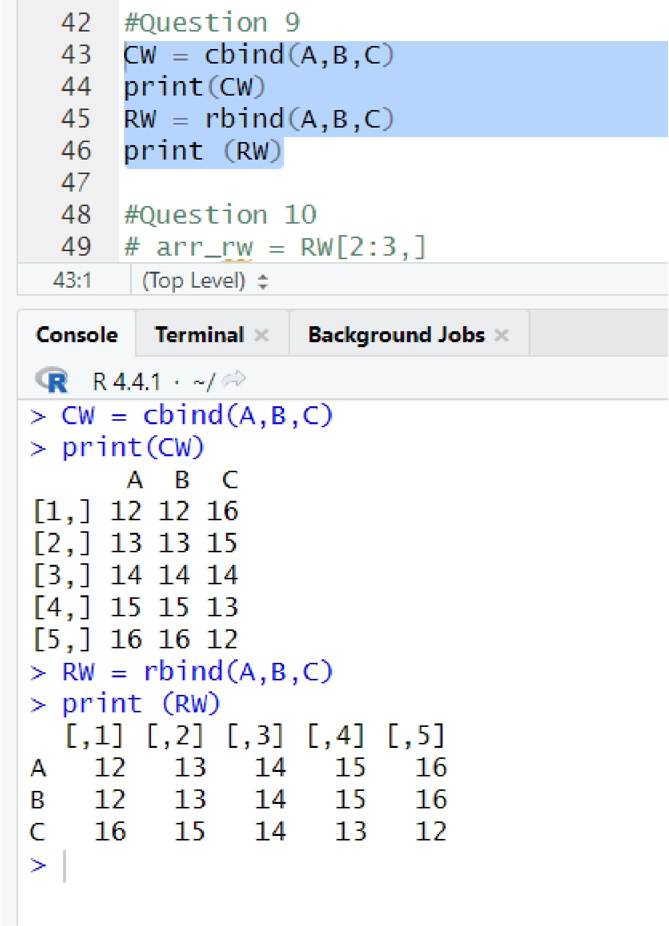
A warning message is displayed since the 1:20 range has 20 elements, but if 3 rows and 4 columns are specified, only 12 elements can be accommodated



```
#Question 8
mat = matrix(1:20, nrow=3, ncol=4)
print(mat)

#Question 9
# CW = cbind(A,B,C)
```

```
> mat = matrix(1:20, nrow=3, ncol=4)
Warning message:
In matrix(1:20, nrow = 3, ncol = 4) :
  data length [20] is not a sub-multiple or multiple of the number of rows [3]
> print(mat)
     [,1] [,2] [,3] [,4]
[1,]    1    4    7   10
[2,]    2    5    8   11
[3,]    3    6    9   12
> 
```

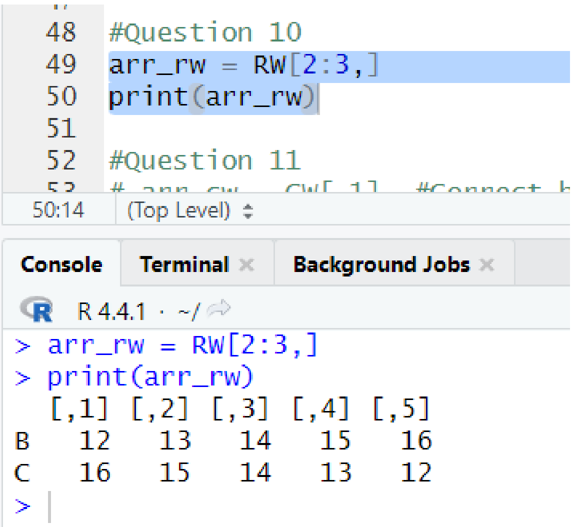
Question 9



```
#Question 9
CW = cbind(A,B,C)
print(CW)
RW = rbind(A,B,C)
print (RW)

#Question 10
# arr_rw = RW[2:3,]
```

```
> CW = cbind(A,B,C)
> print(CW)
      A  B  C
[1,] 12 12 16
[2,] 13 13 15
[3,] 14 14 14
[4,] 15 15 13
[5,] 16 16 12
> RW = rbind(A,B,C)
> print (RW)
  [,1] [,2] [,3] [,4] [,5]
A   12   13   14   15   16
B   12   13   14   15   16
C   16   15   14   13   12
> 
```

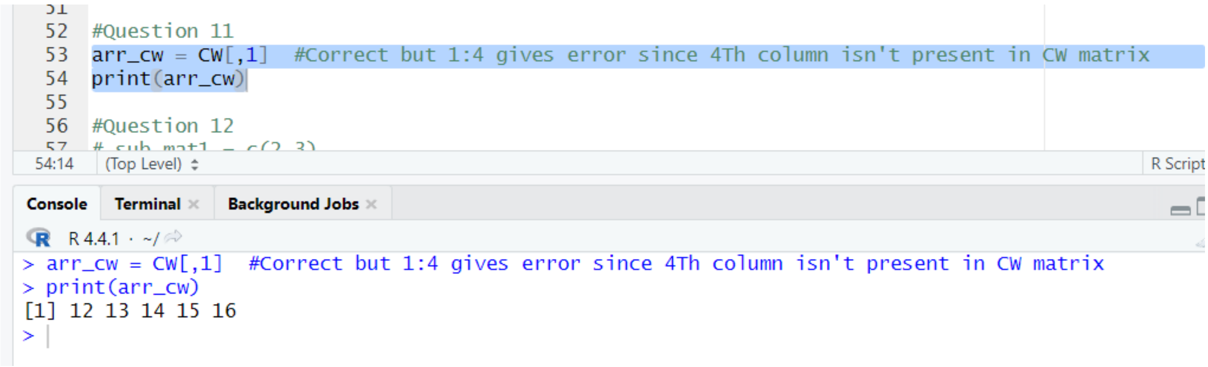
Question 10



```
#Question 10
arr_rw = RW[2:3,]
print(arr_rw)

#Question 11
# arr_cw = CW[,1]  #Correct b
```

```
> arr_rw = RW[2:3,]
> print(arr_rw)
  [,1] [,2] [,3] [,4] [,5]
B   12   13   14   15   16
C   16   15   14   13   12
> 
```

Question 11



```
#Question 11
arr_cw = CW[,1]  #Correct but 1:4 gives error since 4Th column isn't present in CW matrix
print(arr_cw)

#Question 12
# sub_mat1 = c(2,3)
```

```
> arr_cw = CW[,1]  #Correct but 1:4 gives error since 4Th column isn't present in CW matrix
> print(arr_cw)
[1] 12 13 14 15 16
> 
```

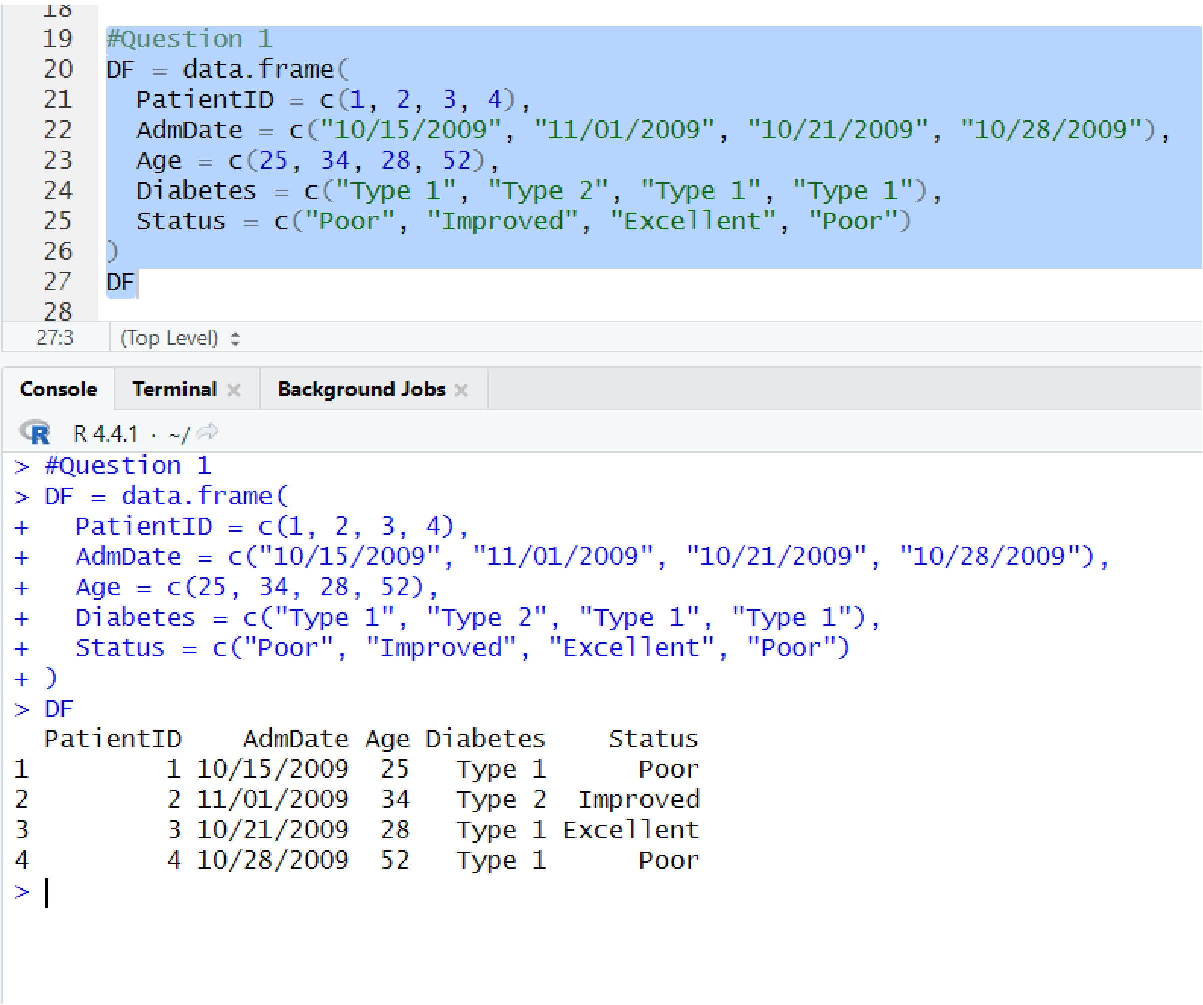
Question 12

LAB ASSIGNMENT 4.1

Submitted By: Kamya Mehra   102213026

3CO35

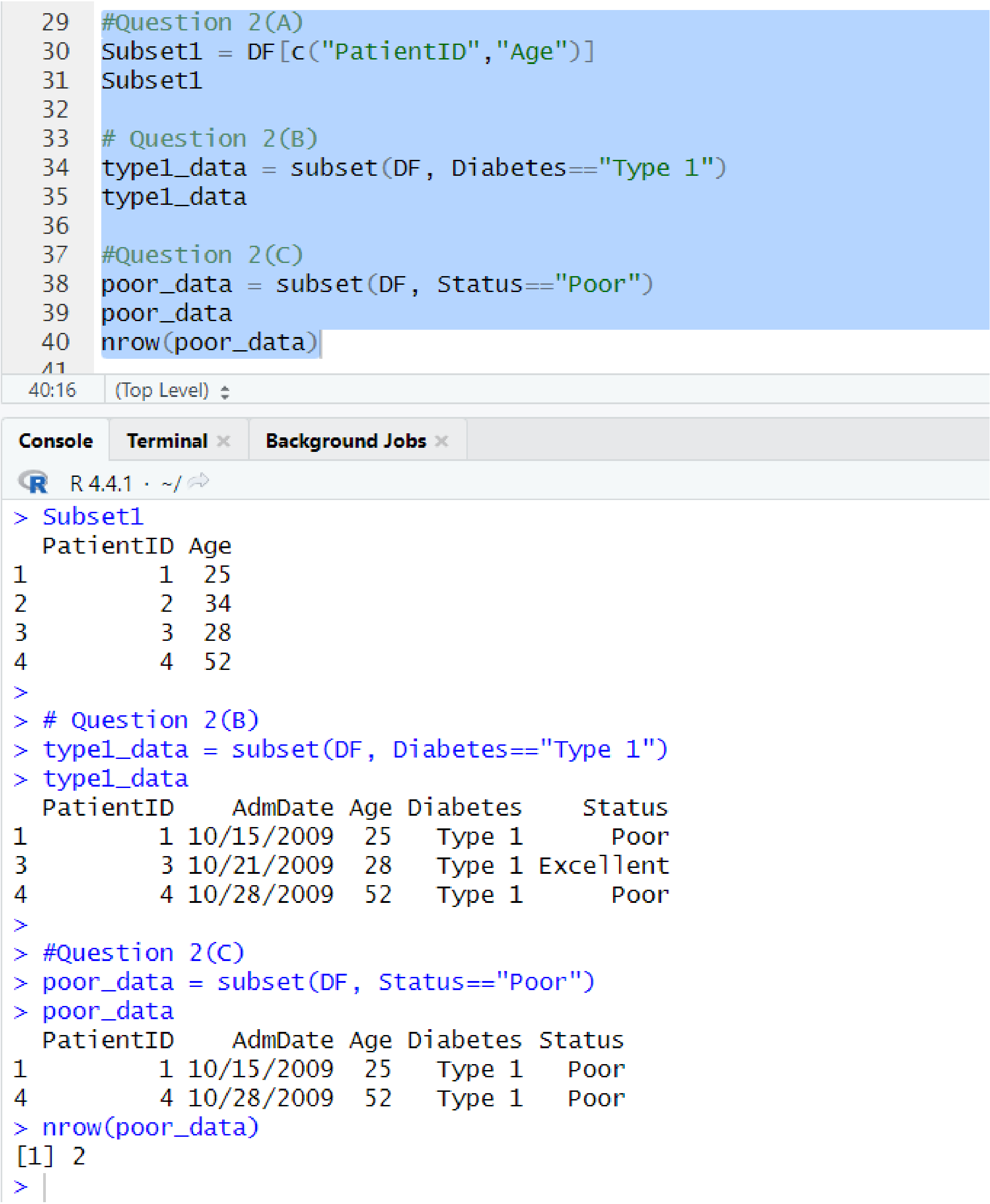
Question 1



```
#Question 1
DF = data.frame(
  PatientID = c(1, 2, 3, 4),
  AdmDate = c("10/15/2009", "11/01/2009", "10/21/2009", "10/28/2009"),
  Age = c(25, 34, 28, 52),
  Diabetes = c("Type 1", "Type 2", "Type 1", "Type 1"),
  Status = c("Poor", "Improved", "Excellent", "Poor")
)
DF
```

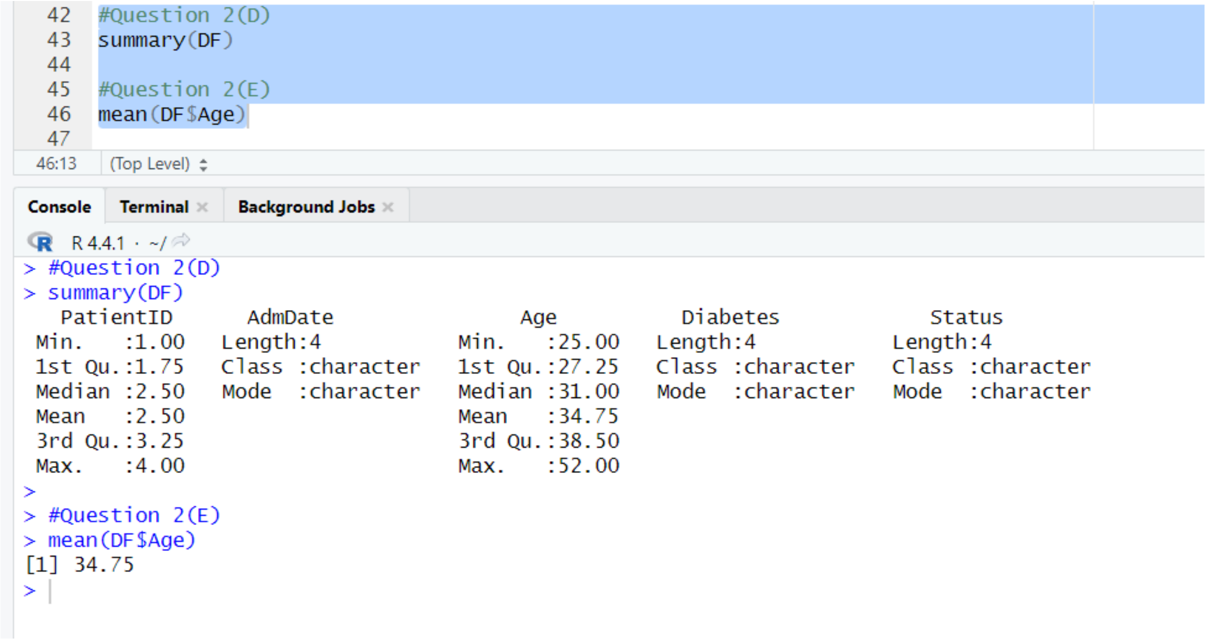
```
> #Question 1
> DF = data.frame(
+   PatientID = c(1, 2, 3, 4),
+   AdmDate = c("10/15/2009", "11/01/2009", "10/21/2009", "10/28/2009"),
+   Age = c(25, 34, 28, 52),
+   Diabetes = c("Type 1", "Type 2", "Type 1", "Type 1"),
+   Status = c("Poor", "Improved", "Excellent", "Poor")
+ )
> DF
  PatientID    AdmDate Age Diabetes    Status
1         1 10/15/2009  25   Type 1      Poor
2         2 11/01/2009  34   Type 2  Improved
3         3 10/21/2009  28   Type 1 Excellent
4         4 10/28/2009  52   Type 1      Poor
> |
```

Question 2



```
#Question 2(A)
Subset1 = DF[c("PatientID","Age")]
Subset1

# Question 2(B)
type1_data = subset(DF, Diabetes=="Type 1")
type1_data

#Question 2(C)
poor_data = subset(DF, Status=="Poor")
poor_data
nrow(poor_data)
```

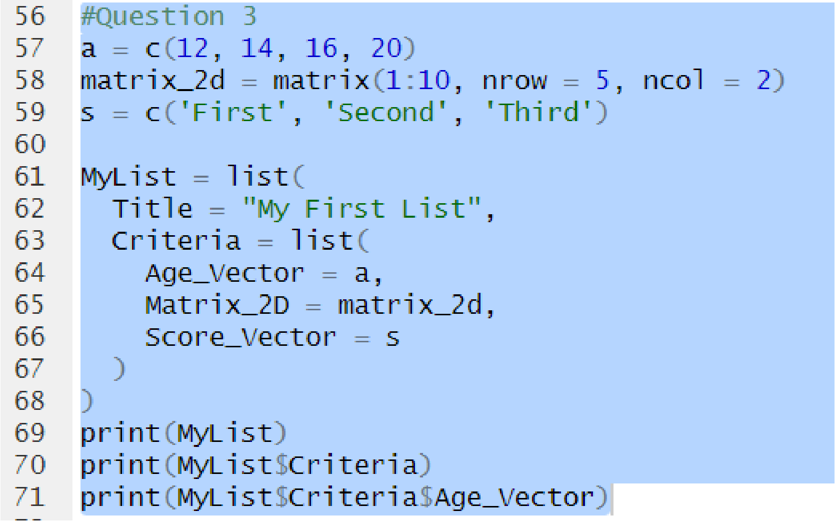
```
> Subset1
  PatientID Age
1         1  25
2         2  34
3         3  28
4         4  52
>
> # Question 2(B)
> type1_data = subset(DF, Diabetes=="Type 1")
> type1_data
  PatientID    AdmDate Age Diabetes    Status
1         1 10/15/2009  25   Type 1      Poor
3         3 10/21/2009  28   Type 1 Excellent
4         4 10/28/2009  52   Type 1      Poor
>
> #Question 2(C)
> poor_data = subset(DF, Status=="Poor")
> poor_data
  PatientID    AdmDate Age Diabetes Status
1         1 10/15/2009  25   Type 1   Poor
4         4 10/28/2009  52   Type 1   Poor
> nrow(poor_data)
[1] 2
> 
```



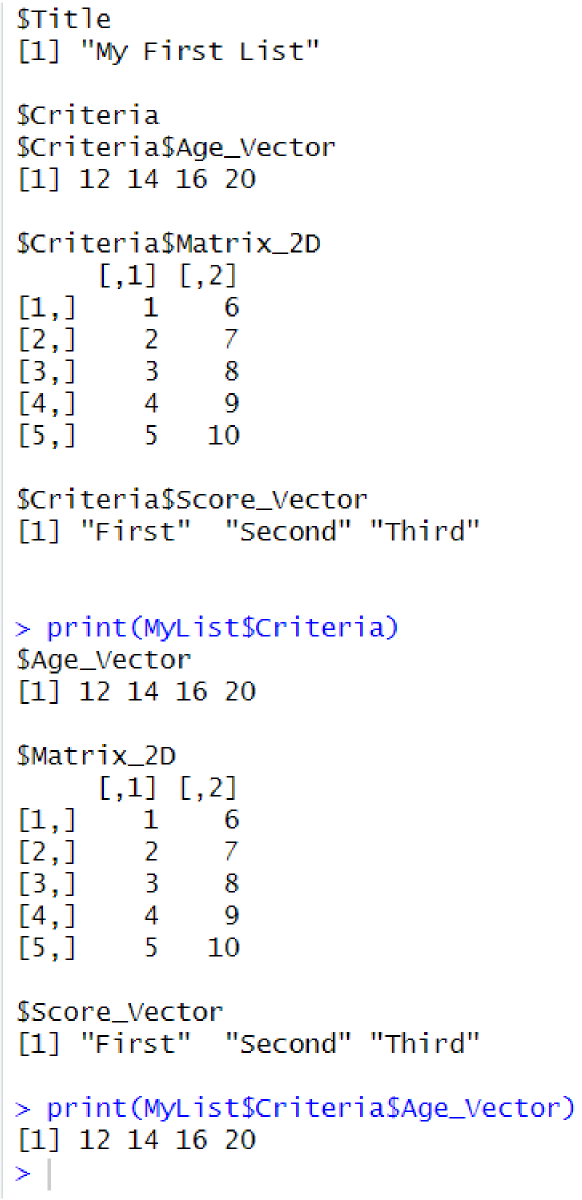
```
#Question 2(D)
summary(DF)

#Question 2(E)
mean(DF$Age)

```

```
> #Question 2(D)
> summary(DF)
   PatientID        AdmDate               Age          Diabetes             Status         
 Min.   :1.00   Length:4           Min.   :25.00   Length:4           Length:4          
 1st Qu.:1.75   Class :character   1st Qu.:27.25   Class :character   Class :character  
 Median :2.50   Mode  :character   Median :31.00   Mode  :character   Mode  :character  
 Mean   :2.50                      Mean   :34.75                                        
 3rd Qu.:3.25                      3rd Qu.:38.50                                        
 Max.   :4.00                      Max.   :52.00                                        
> 
> #Question 2(E)
> mean(DF$Age)
[1] 34.75
> 
```

Question 3



```
#Question 3
a = c(12, 14, 16, 20)
matrix_2d = matrix(1:10, nrow = 5, ncol = 2)
s = c('First', 'Second', 'Third')

MyList = list(
  Title = "My First List",
  Criteria = list(
    Age_Vector = a,
    Matrix_2D = matrix_2d,
    Score_Vector = s
  )
)
print(MyList)
print(MyList$Criteria)
print(MyList$Criteria$Age_Vector)
```



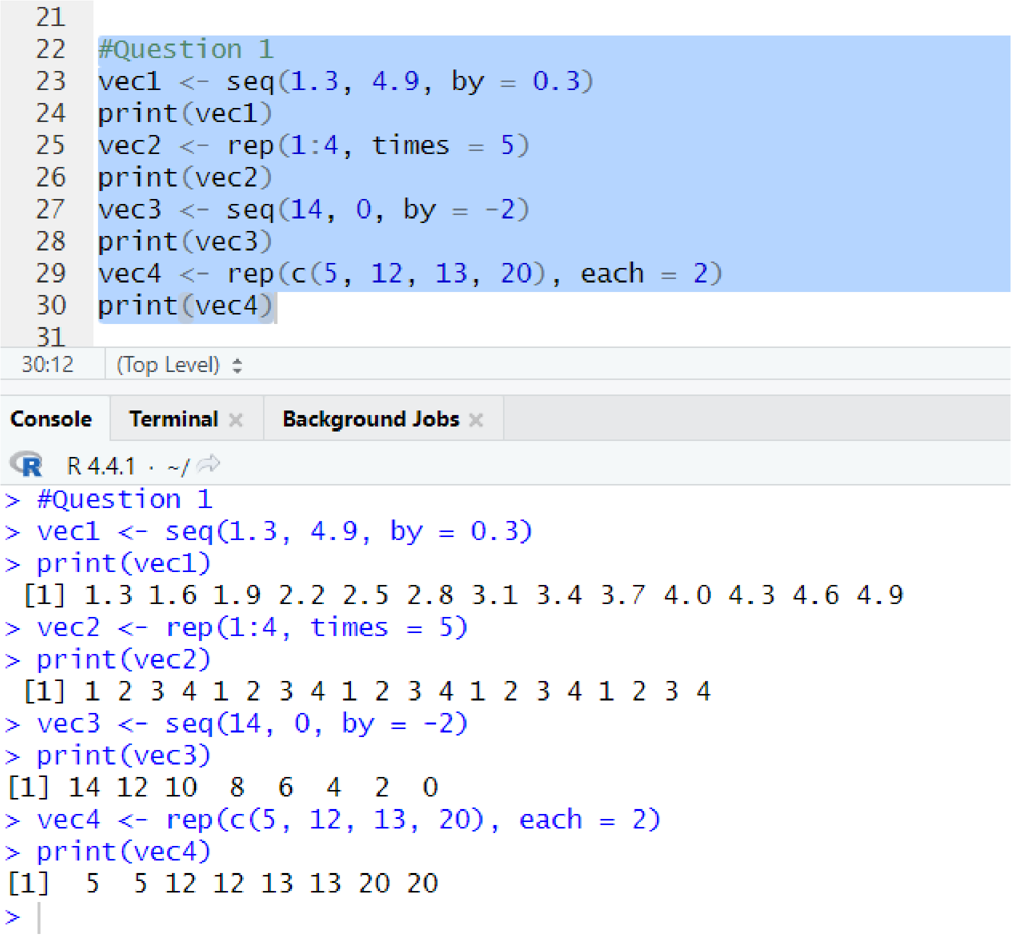
```
$Title
[1] "My First List"

$Criteria
$Criteria$Age_Vector
[1] 12 14 16 20

$Criteria$Matrix_2D
     [,1] [,2]
[1,]    1    6
[2,]    2    7
[3,]    3    8
[4,]    4    9
[5,]    5   10

$Criteria$Score_Vector
[1] "First"  "Second" "Third"


> print(MyList$Criteria)
$Age_Vector
[1] 12 14 16 20

$Matrix_2D
     [,1] [,2]
[1,]    1    6
[2,]    2    7
[3,]    3    8
[4,]    4    9
[5,]    5   10

$Score_Vector
[1] "First"  "Second" "Third"

> print(MyList$Criteria$Age_Vector)
[1] 12 14 16 20
> 
```

LAB ASSIGNMENT 4.2

Submitted By: Kamya Mehra 102213026

3CO35

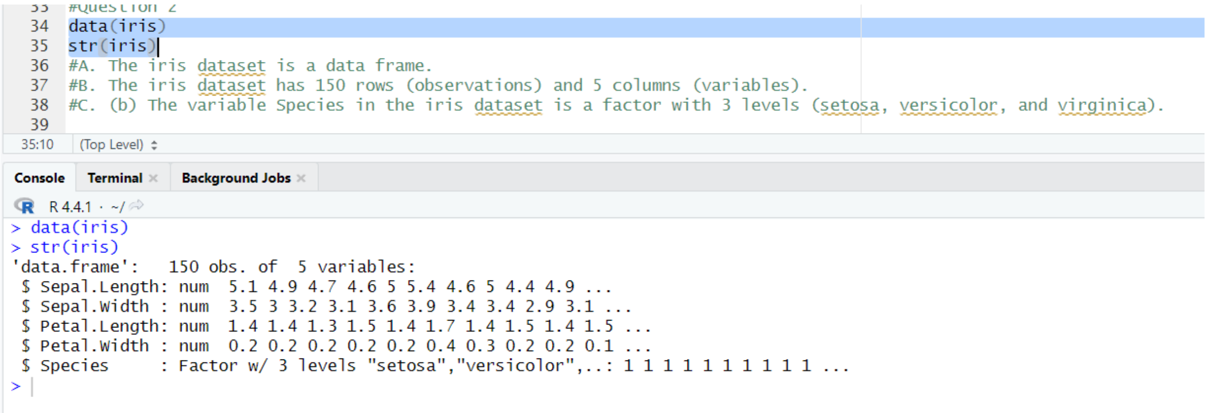
Question 1



```
#Question 1
vec1 <- seq(1.3, 4.9, by = 0.3)
print(vec1)
vec2 <- rep(1:4, times = 5)
print(vec2)
vec3 <- seq(14, 0, by = -2)
print(vec3)
vec4 <- rep(c(5, 12, 13, 20), each = 2)
print(vec4)
```

```
> #Question 1
> vec1 <- seq(1.3, 4.9, by = 0.3)
> print(vec1)
 [1] 1.3 1.6 1.9 2.2 2.5 2.8 3.1 3.4 3.7 4.0 4.3 4.6 4.9
> vec2 <- rep(1:4, times = 5)
> print(vec2)
 [1] 1 2 3 4 1 2 3 4 1 2 3 4 1 2 3 4 1 2 3 4
> vec3 <- seq(14, 0, by = -2)
> print(vec3)
[1] 14 12 10  8  6  4  2  0
> vec4 <- rep(c(5, 12, 13, 20), each = 2)
> print(vec4)
[1]  5  5 12 12 13 13 20 20
> 
```

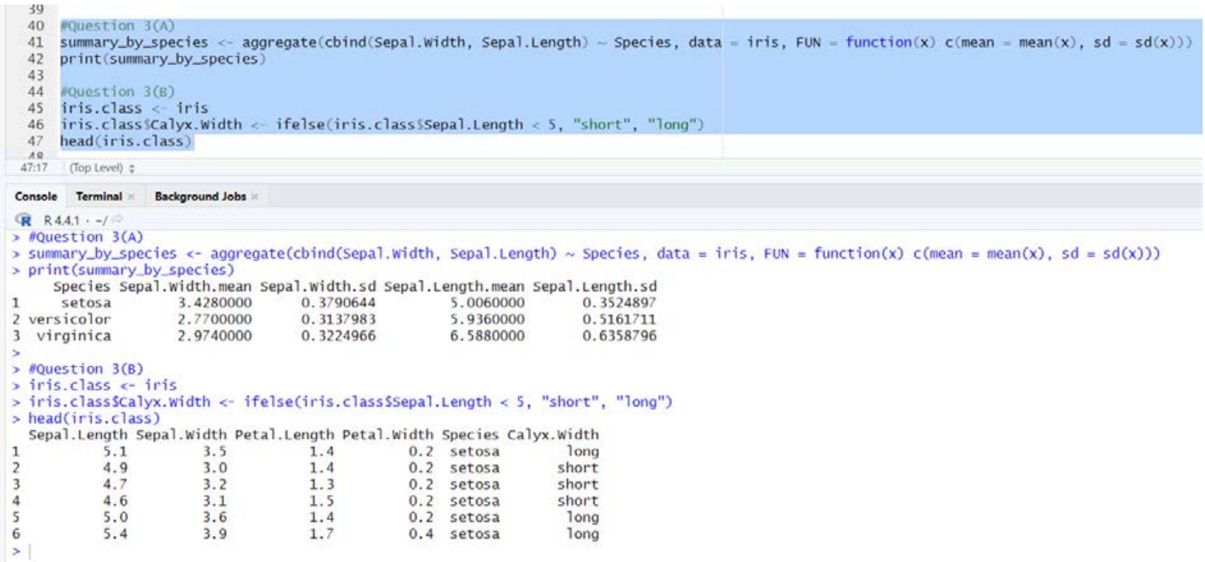
Question 2



```
#Question 2
data(iris)
str(iris)
#A. The iris dataset is a data frame.
#B. The iris dataset has 150 rows (observations) and 5 columns (variables).
#C. (b) The variable Species in the iris dataset is a factor with 3 levels (setosa, versicolor, and virginica).
```

```
> data(iris)
> str(iris)
'data.frame':	150 obs. of  5 variables:
 $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
 $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
 $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
 $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
 $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
> 
```

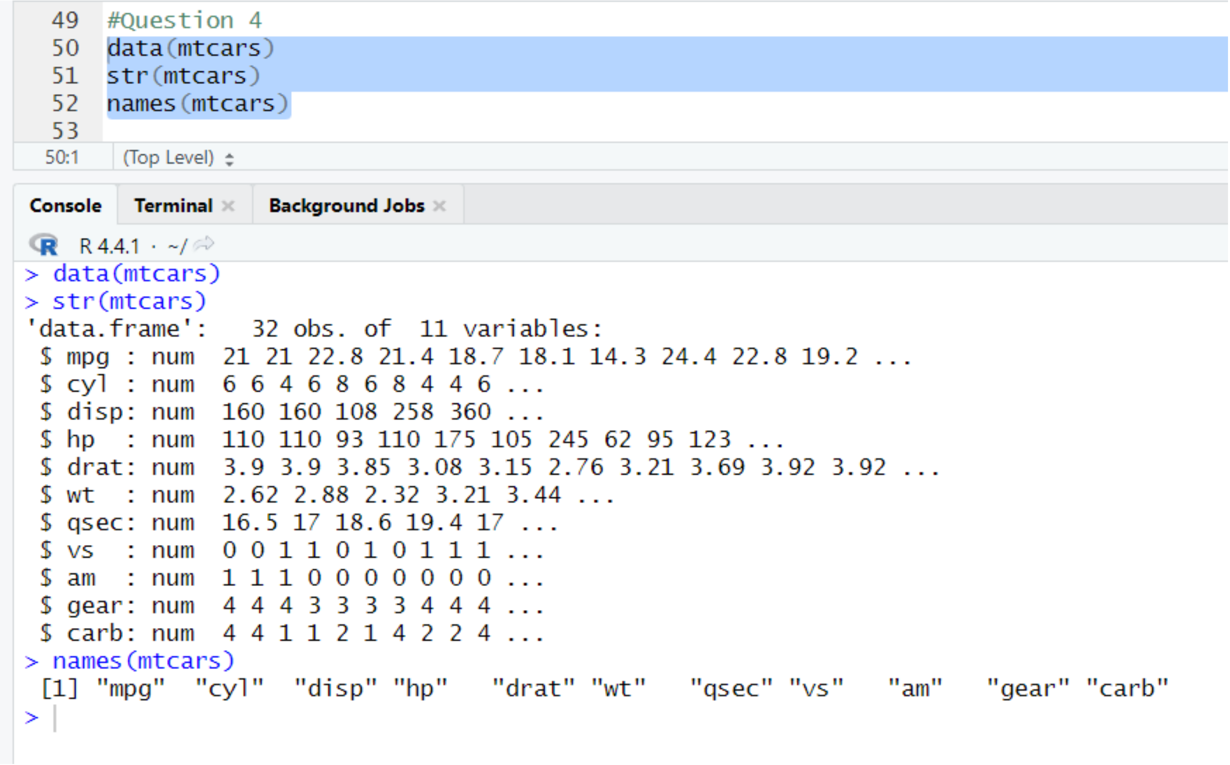
Question 3



```
#Question 3(A)
summary_by_species <- aggregate(cbind(Sepal.Width, Sepal.Length) ~ Species, data = iris, FUN = function(x) c(mean = mean(x), sd = sd(x)))
print(summary_by_species)

#Question 3(B)
iris.class <- iris
iris.class$Calyx.Width <- ifelse(iris.class$Sepal.Length < 5, "short", "long")
head(iris.class)
```

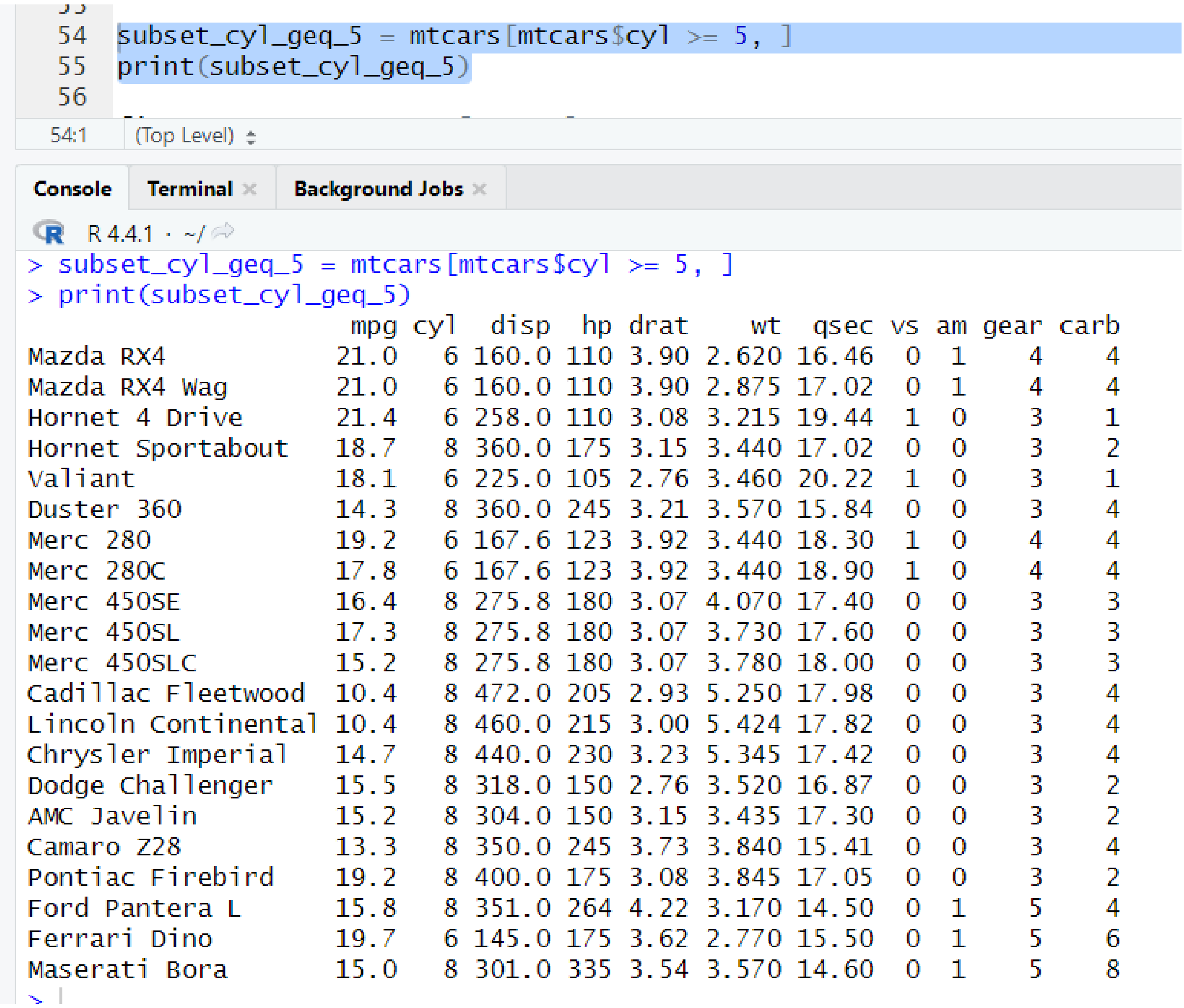
```
> #Question 3(A)
> summary_by_species <- aggregate(cbind(Sepal.Width, Sepal.Length) ~ Species, data = iris, FUN = function(x) c(mean = mean(x), sd = sd(x)))
> print(summary_by_species)
     Species Sepal.Width.mean Sepal.Width.sd Sepal.Length.mean Sepal.Length.sd
1     setosa        3.4280000      0.3790644         5.0060000       0.3524897
2 versicolor        2.7700000      0.3137983         5.9360000       0.5161711
3  virginica        2.9740000      0.3224966         6.5880000       0.6358796
>
> #Question 3(B)
> iris.class <- iris
> iris.class$Calyx.Width <- ifelse(iris.class$Sepal.Length < 5, "short", "long")
> head(iris.class)
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species Calyx.Width
1          5.1         3.5          1.4         0.2  setosa        long
2          4.9         3.0          1.4         0.2  setosa       short
3          4.7         3.2          1.3         0.2  setosa       short
4          4.6         3.1          1.5         0.2  setosa       short
5          5.0         3.6          1.4         0.2  setosa        long
6          5.4         3.9          1.7         0.4  setosa        long
> |
```

Question 4



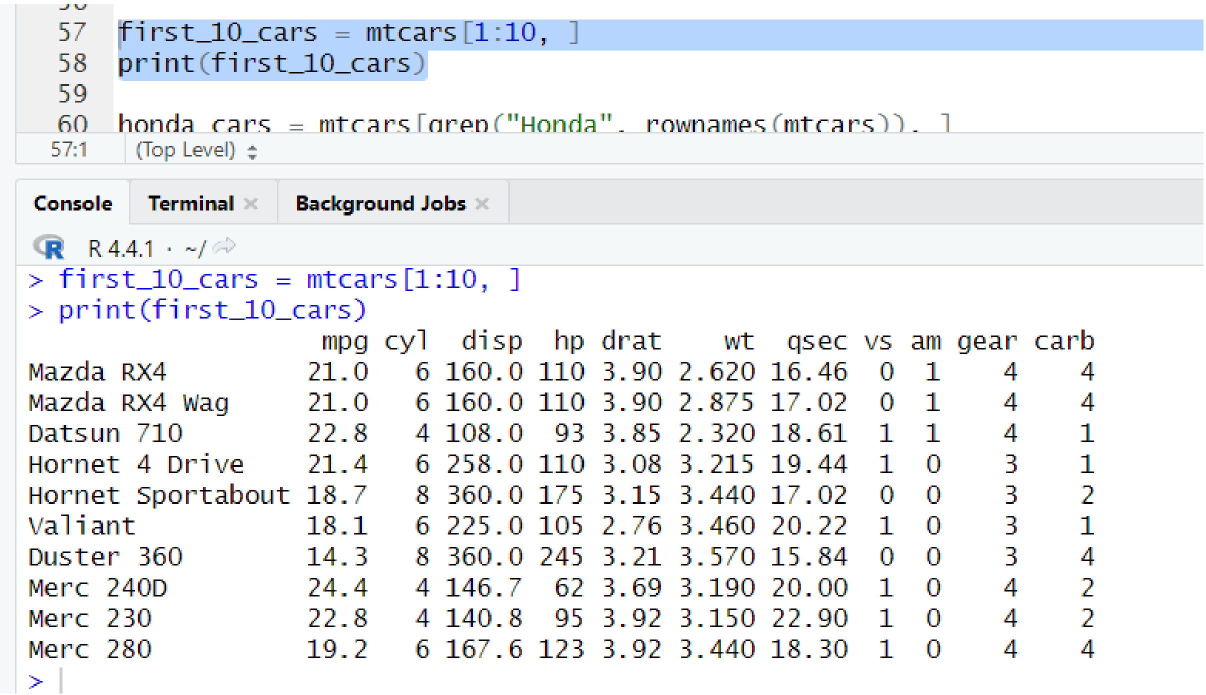
```
#Question 4
data(mtcars)
str(mtcars)
names(mtcars)
```

```
> data(mtcars)
> str(mtcars)
'data.frame':	32 obs. of  11 variables:
 $ mpg : num  21 21 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 ...
 $ cyl : num  6 6 4 6 8 6 8 4 4 6 ...
 $ disp: num  160 160 108 258 360 ...
 $ hp  : num  110 110 93 110 175 105 245 62 95 123 ...
 $ drat: num  3.9 3.9 3.85 3.08 3.15 2.76 3.21 3.69 3.92 3.92 ...
 $ wt  : num  2.62 2.88 2.32 3.21 3.44 ...
 $ qsec: num  16.5 17 18.6 19.4 17 ...
 $ vs  : num  0 0 1 1 0 1 0 1 1 1 ...
 $ am  : num  1 1 1 0 0 0 0 0 0 0 ...
 $ gear: num  4 4 4 3 3 3 3 4 4 4 ...
 $ carb: num  4 4 1 1 2 1 4 2 2 4 ...
> names(mtcars)
 [1] "mpg"  "cyl"  "disp" "hp"   "drat" "wt"   "qsec" "vs"   "am"   "gear" "carb"
> |
```



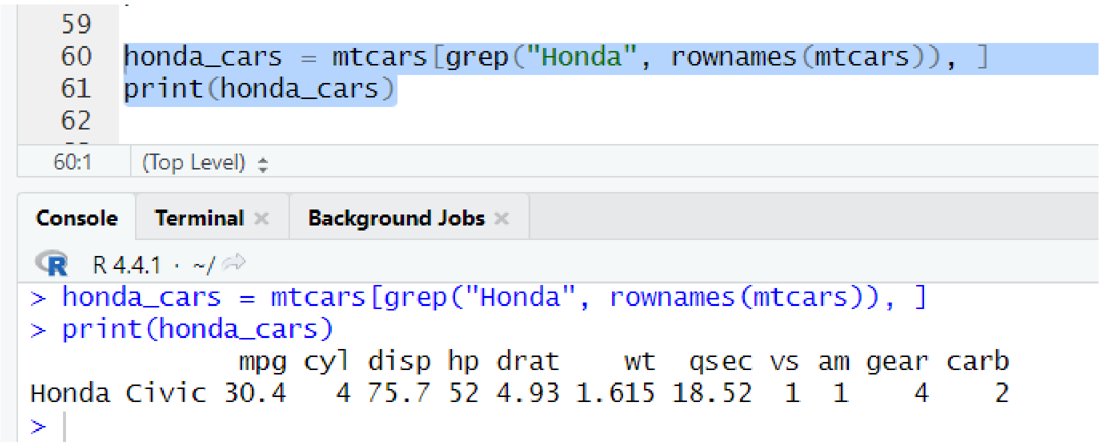
```r
subset_cyl_geq_5 = mtcars[mtcars$cyl >= 5, ]
print(subset_cyl_geq_5)
```

```
> subset_cyl_geq_5 = mtcars[mtcars$cyl >= 5, ]
> print(subset_cyl_geq_5)
                     mpg cyl  disp  hp drat    wt  qsec vs am gear carb
Mazda RX4           21.0   6 160.0 110 3.90 2.620 16.46  0  1    4    4
Mazda RX4 Wag       21.0   6 160.0 110 3.90 2.875 17.02  0  1    4    4
Hornet 4 Drive      21.4   6 258.0 110 3.08 3.215 19.44  1  0    3    1
Hornet Sportabout   18.7   8 360.0 175 3.15 3.440 17.02  0  0    3    2
Valiant             18.1   6 225.0 105 2.76 3.460 20.22  1  0    3    1
Duster 360          14.3   8 360.0 245 3.21 3.570 15.84  0  0    3    4
Merc 280            19.2   6 167.6 123 3.92 3.440 18.30  1  0    4    4
Merc 280C           17.8   6 167.6 123 3.92 3.440 18.90  1  0    4    4
Merc 450SE          16.4   8 275.8 180 3.07 4.070 17.40  0  0    3    3
Merc 450SL          17.3   8 275.8 180 3.07 3.730 17.60  0  0    3    3
Merc 450SLC         15.2   8 275.8 180 3.07 3.780 18.00  0  0    3    3
Cadillac Fleetwood  10.4   8 472.0 205 2.93 5.250 17.98  0  0    3    4
Lincoln Continental 10.4   8 460.0 215 3.00 5.424 17.82  0  0    3    4
Chrysler Imperial   14.7   8 440.0 230 3.23 5.345 17.42  0  0    3    4
Dodge Challenger    15.5   8 318.0 150 2.76 3.520 16.87  0  0    3    2
AMC Javelin         15.2   8 304.0 150 3.15 3.435 17.30  0  0    3    2
Camaro Z28          13.3   8 350.0 245 3.73 3.840 15.41  0  0    3    4
Pontiac Firebird    19.2   8 400.0 175 3.08 3.845 17.05  0  0    3    2
Ford Pantera L      15.8   8 351.0 264 4.22 3.170 14.50  0  1    5    4
Ferrari Dino        19.7   6 145.0 175 3.62 2.770 15.50  0  1    5    6
Maserati Bora       15.0   8 301.0 335 3.54 3.570 14.60  0  1    5    8
```



```r
first_10_cars = mtcars[1:10, ]
print(first_10_cars)

honda_cars = mtcars[grep("Honda", rownames(mtcars)), ]
```

```
> first_10_cars = mtcars[1:10, ]
> print(first_10_cars)
                   mpg cyl  disp  hp drat    wt  qsec vs am gear carb
Mazda RX4         21.0   6 160.0 110 3.90 2.620 16.46  0  1    4    4
Mazda RX4 Wag     21.0   6 160.0 110 3.90 2.875 17.02  0  1    4    4
Datsun 710        22.8   4 108.0  93 3.85 2.320 18.61  1  1    4    1
Hornet 4 Drive    21.4   6 258.0 110 3.08 3.215 19.44  1  0    3    1
Hornet Sportabout 18.7   8 360.0 175 3.15 3.440 17.02  0  0    3    2
Valiant           18.1   6 225.0 105 2.76 3.460 20.22  1  0    3    1
Duster 360        14.3   8 360.0 245 3.21 3.570 15.84  0  0    3    4
Merc 240D         24.4   4 146.7  62 3.69 3.190 20.00  1  0    4    2
Merc 230          22.8   4 140.8  95 3.92 3.150 22.90  1  0    4    2
Merc 280          19.2   6 167.6 123 3.92 3.440 18.30  1  0    4    4
>
```



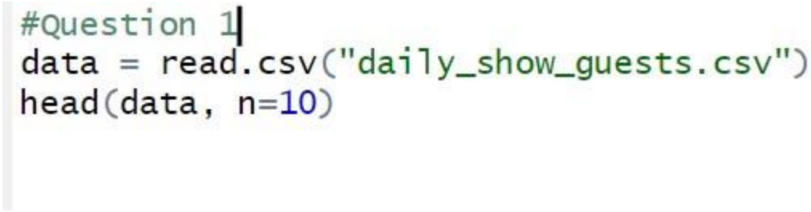
```
honda_cars = mtcars[grep("Honda", rownames(mtcars)), ]
print(honda_cars)
```

```
> honda_cars = mtcars[grep("Honda", rownames(mtcars)), ]
> print(honda_cars)
             mpg cyl disp hp drat    wt  qsec vs am gear carb
Honda Civic 30.4   4 75.7 52 4.93 1.615 18.52  1  1    4    2
>
```

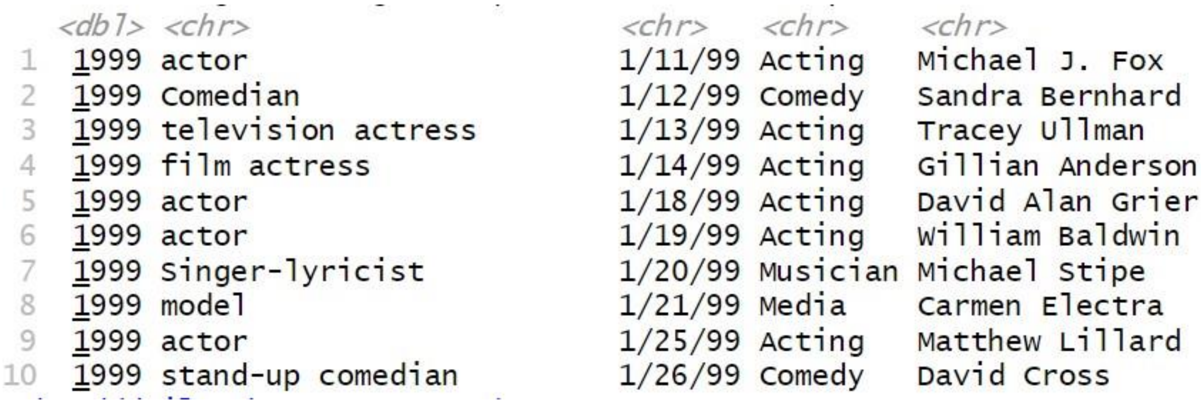
LAB ASSIGNMENT 5

Submitted By: Kamya Mehra 102213026 3CO35

Question 1

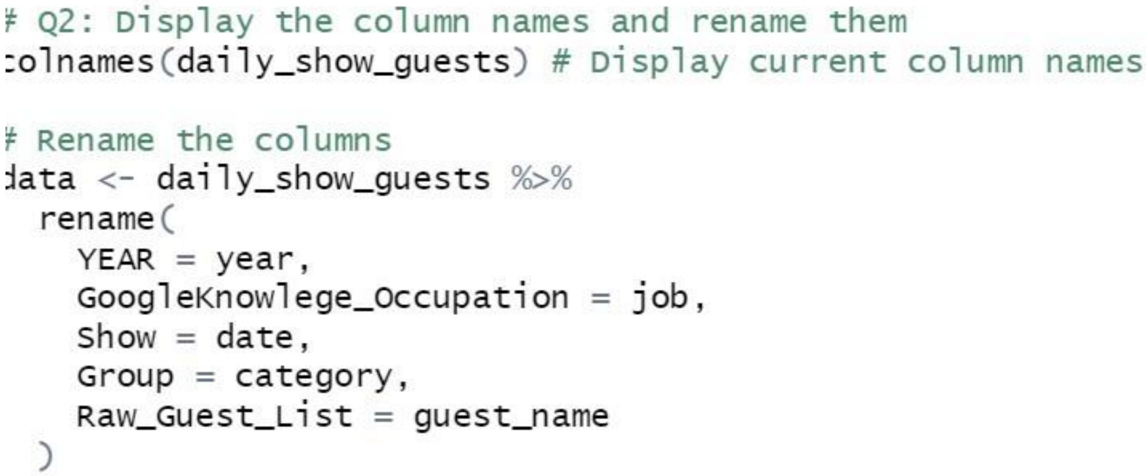


```
#Question 1
data = read.csv("daily_show_guests.csv")
head(data, n=10)
```



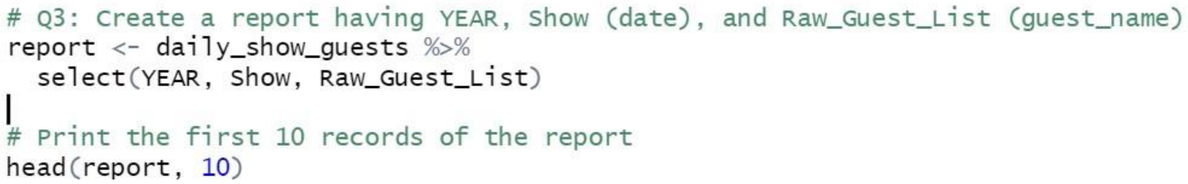
```
   <dbl> <chr>                       <chr>   <chr>    <chr>
 1  1999 actor                       1/11/99 Acting   Michael J. Fox
 2  1999 Comedian                    1/12/99 Comedy   Sandra Bernhard
 3  1999 television actress          1/13/99 Acting   Tracey Ullman
 4  1999 film actress                1/14/99 Acting   Gillian Anderson
 5  1999 actor                       1/18/99 Acting   David Alan Grier
 6  1999 actor                       1/19/99 Acting   William Baldwin
 7  1999 Singer-lyricist             1/20/99 Musician Michael Stipe
 8  1999 model                       1/21/99 Media    Carmen Electra
 9  1999 actor                       1/25/99 Acting   Matthew Lillard
10  1999 stand-up comedian           1/26/99 Comedy   David Cross
```

Question 2

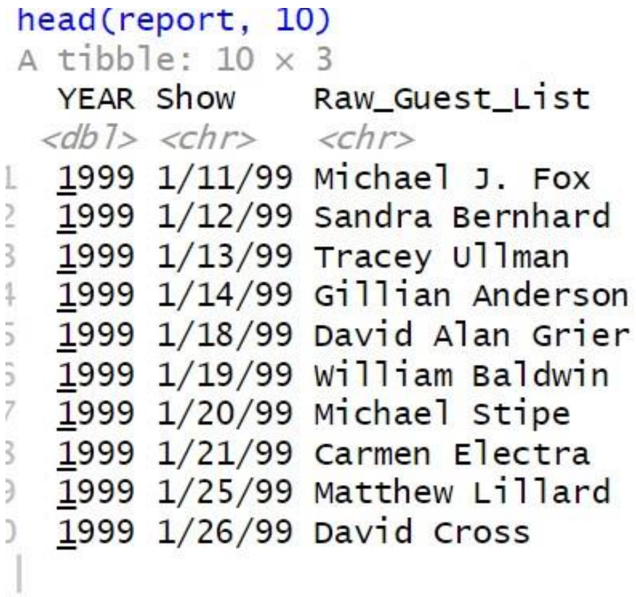


```
# Q2: Display the column names and rename them
colnames(daily_show_guests) # Display current column names

# Rename the columns
data <- daily_show_guests %>%
  rename(
    YEAR = year,
    GoogleKnowlege_Occupation = job,
    Show = date,
    Group = category,
    Raw_Guest_List = guest_name
  )
```

Question 3

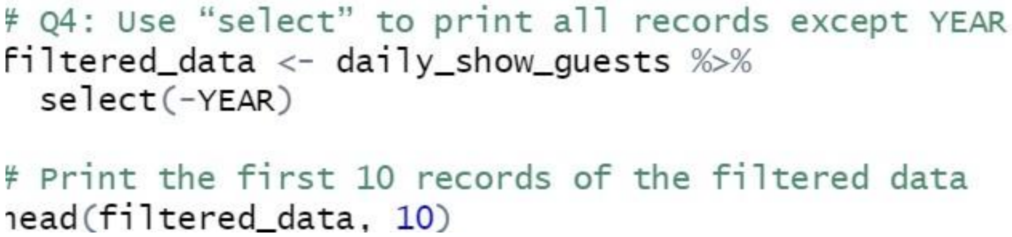


```
# Q3: Create a report having YEAR, Show (date), and Raw_Guest_List (guest_name)
report <- daily_show_guests %>%
  select(YEAR, Show, Raw_Guest_List)

# Print the first 10 records of the report
head(report, 10)
```

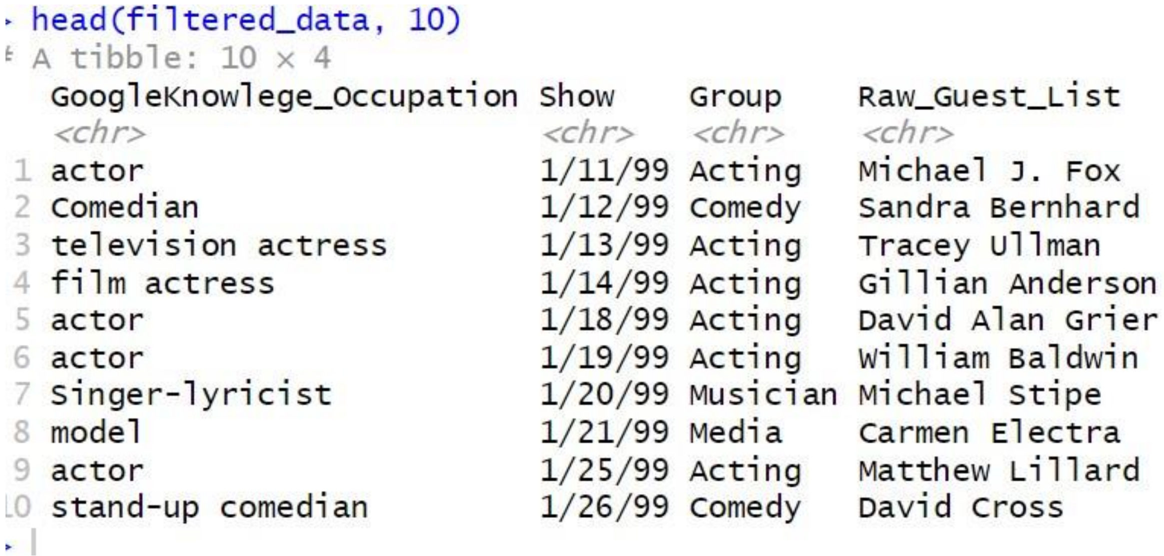


```
head(report, 10)
A tibble: 10 × 3
   YEAR Show    Raw_Guest_List
  <dbl> <chr>   <chr>
L  1999 1/11/99 Michael J. Fox
2  1999 1/12/99 Sandra Bernhard
3  1999 1/13/99 Tracey Ullman
4  1999 1/14/99 Gillian Anderson
5  1999 1/18/99 David Alan Grier
5  1999 1/19/99 William Baldwin
7  1999 1/20/99 Michael Stipe
3  1999 1/21/99 Carmen Electra
)  1999 1/25/99 Matthew Lillard
)  1999 1/26/99 David Cross
```

Question 4

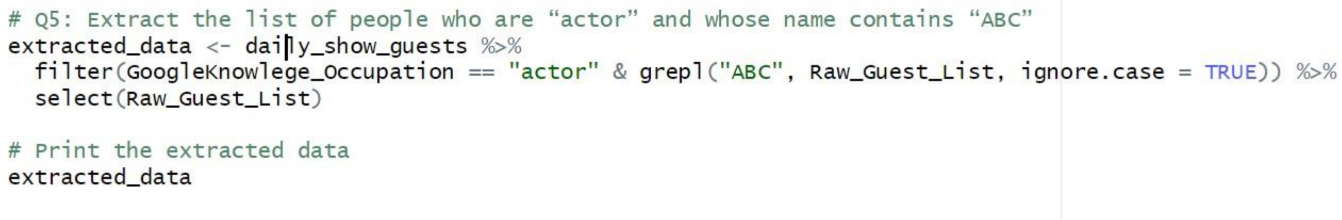


```
# Q4: Use "select" to print all records except YEAR
filtered_data <- daily_show_guests %>%
  select(-YEAR)

# Print the first 10 records of the filtered data
ıead(filtered_data, 10)
```



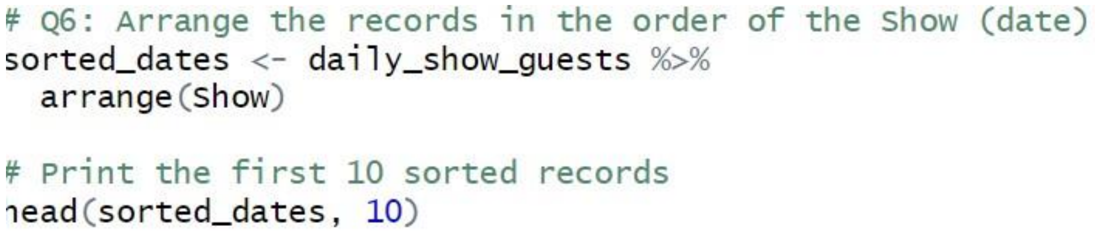
```
› head(filtered_data, 10)
# A tibble: 10 × 4
   GoogleKnowlege_Occupation Show    Group    Raw_Guest_List
   <chr>                     <chr>   <chr>    <chr>
 1 actor                     1/11/99 Acting   Michael J. Fox
 2 Comedian                  1/12/99 Comedy   Sandra Bernhard
 3 television actress        1/13/99 Acting   Tracey Ullman
 4 film actress              1/14/99 Acting   Gillian Anderson
 5 actor                     1/18/99 Acting   David Alan Grier
 6 actor                     1/19/99 Acting   William Baldwin
 7 Singer-lyricist           1/20/99 Musician Michael Stipe
 8 model                     1/21/99 Media    Carmen Electra
 9 actor                     1/25/99 Acting   Matthew Lillard
10 stand-up comedian         1/26/99 Comedy   David Cross
›
```

Question 5

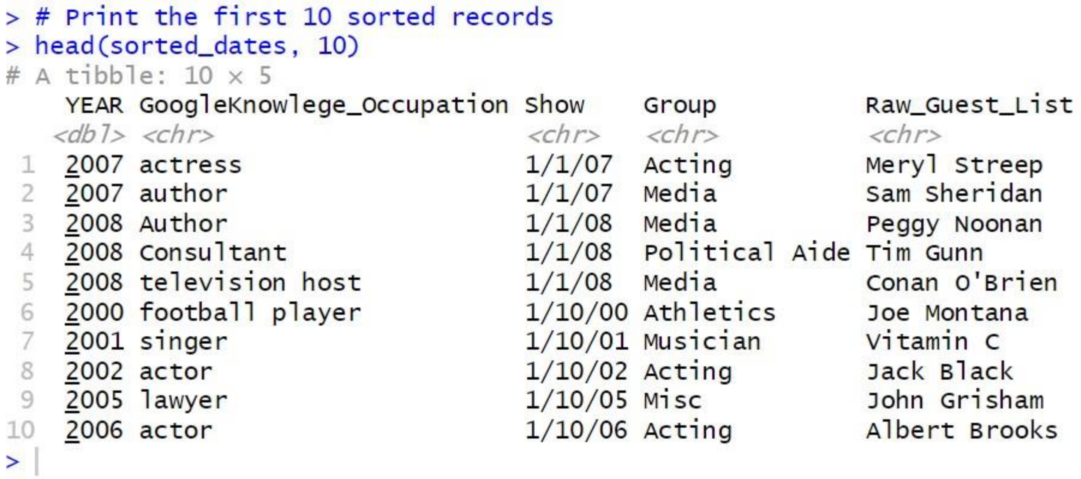


```
# Q5: Extract the list of people who are “actor” and whose name contains “ABC”
extracted_data <- daily_show_guests %>%
  filter(GoogleKnowlege_Occupation == "actor" & grepl("ABC", Raw_Guest_List, ignore.case = TRUE)) %>%
  select(Raw_Guest_List)

# Print the extracted data
extracted_data
```

Question 6

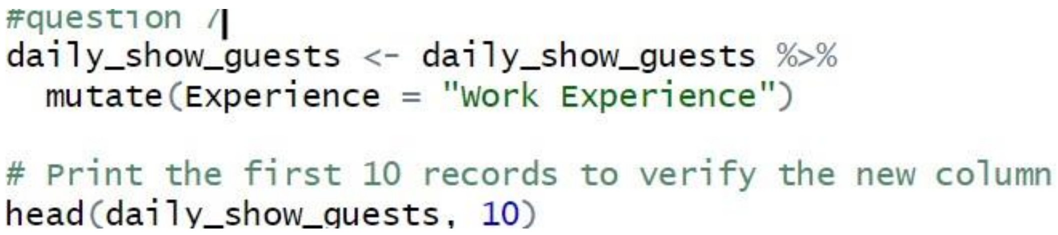


```
# Q6: Arrange the records in the order of the Show (date)
sorted_dates <- daily_show_guests %>%
  arrange(Show)

# Print the first 10 sorted records
head(sorted_dates, 10)
```

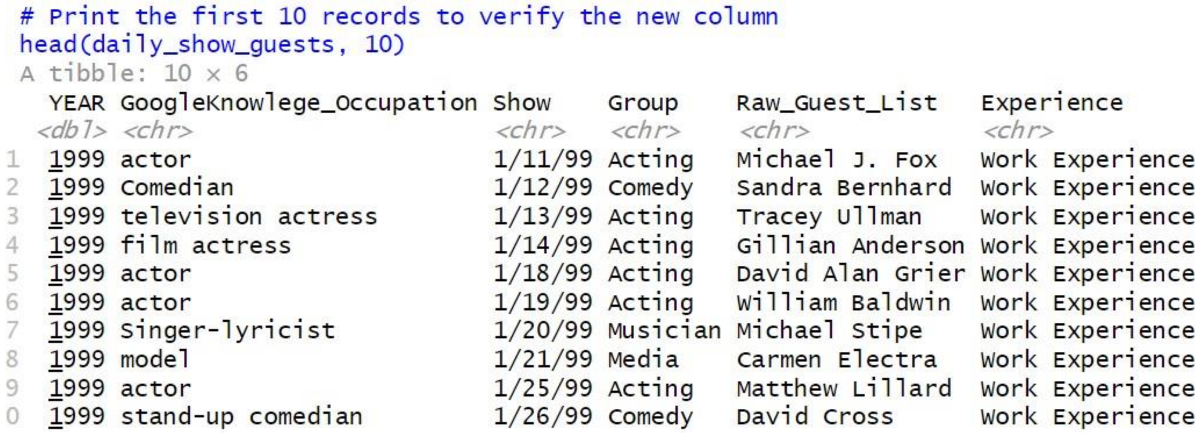


```
> # Print the first 10 sorted records
> head(sorted_dates, 10)
# A tibble: 10 × 5
    YEAR GoogleKnowlege_Occupation Show    Group          Raw_Guest_List
   <dbl> <chr>                     <chr>   <chr>          <chr>         
 1  2007 actress                   1/1/07  Acting         Meryl Streep  
 2  2007 author                    1/1/07  Media          Sam Sheridan  
 3  2008 Author                    1/1/08  Media          Peggy Noonan  
 4  2008 Consultant                1/1/08  Political Aide Tim Gunn      
 5  2008 television host           1/1/08  Media          Conan O'Brien 
 6  2000 football player           1/10/00 Athletics      Joe Montana   
 7  2001 singer                    1/10/01 Musician       Vitamin C     
 8  2002 actor                     1/10/02 Acting         Jack Black    
 9  2005 lawyer                    1/10/05 Misc           John Grisham  
10  2006 actor                     1/10/06 Acting         Albert Brooks 
> 
```

Question 7



```
#question 7
daily_show_guests <- daily_show_guests %>%
  mutate(Experience = "Work Experience")

# Print the first 10 records to verify the new column
head(daily_show_guests, 10)
```

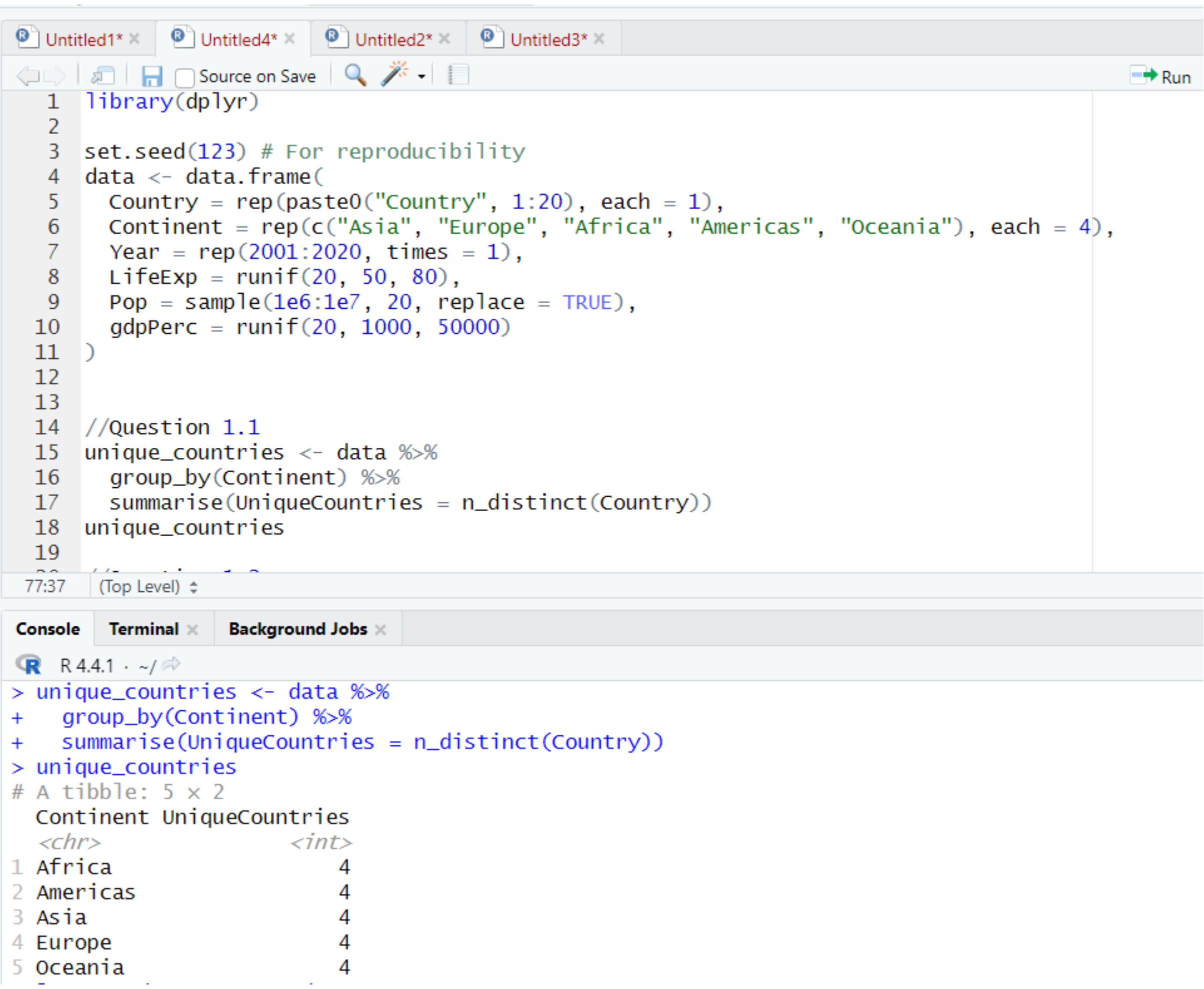


```
# Print the first 10 records to verify the new column
head(daily_show_guests, 10)
A tibble: 10 × 6
   YEAR GoogleKnowlege_Occupation Show    Group    Raw_Guest_List   Experience
  <dbl> <chr>                     <chr>   <chr>    <chr>            <chr>
1  1999 actor                     1/11/99 Acting   Michael J. Fox   Work Experience
2  1999 Comedian                  1/12/99 Comedy   Sandra Bernhard  Work Experience
3  1999 television actress        1/13/99 Acting   Tracey Ullman    Work Experience
4  1999 film actress              1/14/99 Acting   Gillian Anderson Work Experience
5  1999 actor                     1/18/99 Acting   David Alan Grier Work Experience
6  1999 actor                     1/19/99 Acting   William Baldwin  Work Experience
7  1999 Singer-lyricist           1/20/99 Musician Michael Stipe    Work Experience
8  1999 model                     1/21/99 Media    Carmen Electra   Work Experience
9  1999 actor                     1/25/99 Acting   Matthew Lillard  Work Experience
0  1999 stand-up comedian         1/26/99 Comedy   David Cross      Work Experience
```

LAB ASSIGNMENT 6

Submitted By: Kamya Mehra 102213026 3CO35

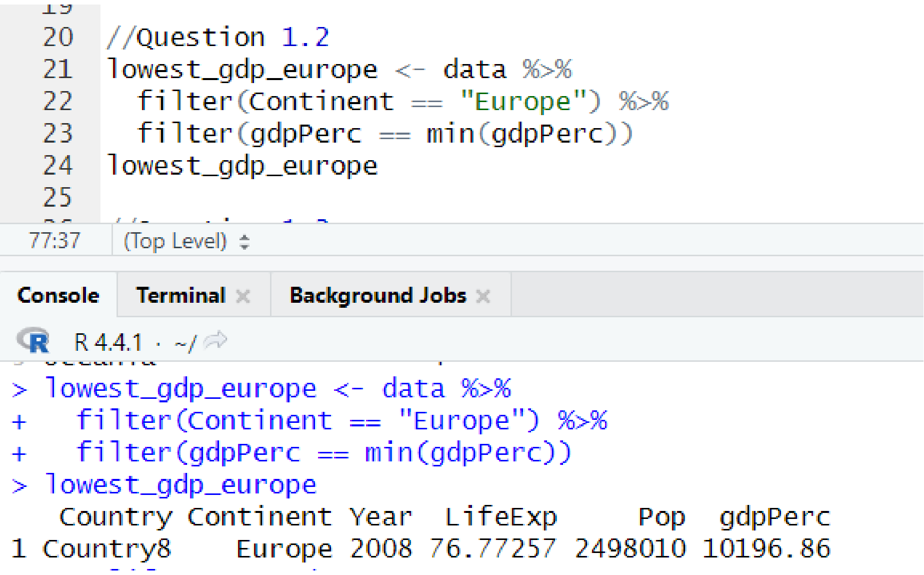
Question 1.1



```
library(dplyr)

set.seed(123) # For reproducibility
data <- data.frame(
  Country = rep(paste0("Country", 1:20), each = 1),
  Continent = rep(c("Asia", "Europe", "Africa", "Americas", "Oceania"), each = 4),
  Year = rep(2001:2020, times = 1),
  LifeExp = runif(20, 50, 80),
  Pop = sample(1e6:1e7, 20, replace = TRUE),
  gdpPerc = runif(20, 1000, 50000)
)


//Question 1.1
unique_countries <- data %>%
  group_by(Continent) %>%
  summarise(UniqueCountries = n_distinct(Country))
unique_countries

```

```
> unique_countries <- data %>%
+   group_by(Continent) %>%
+   summarise(UniqueCountries = n_distinct(Country))
> unique_countries
# A tibble: 5 × 2
  Continent UniqueCountries
  <chr>               <int>
1 Africa                  4
2 Americas                4
3 Asia                    4
4 Europe                  4
5 Oceania                 4
```

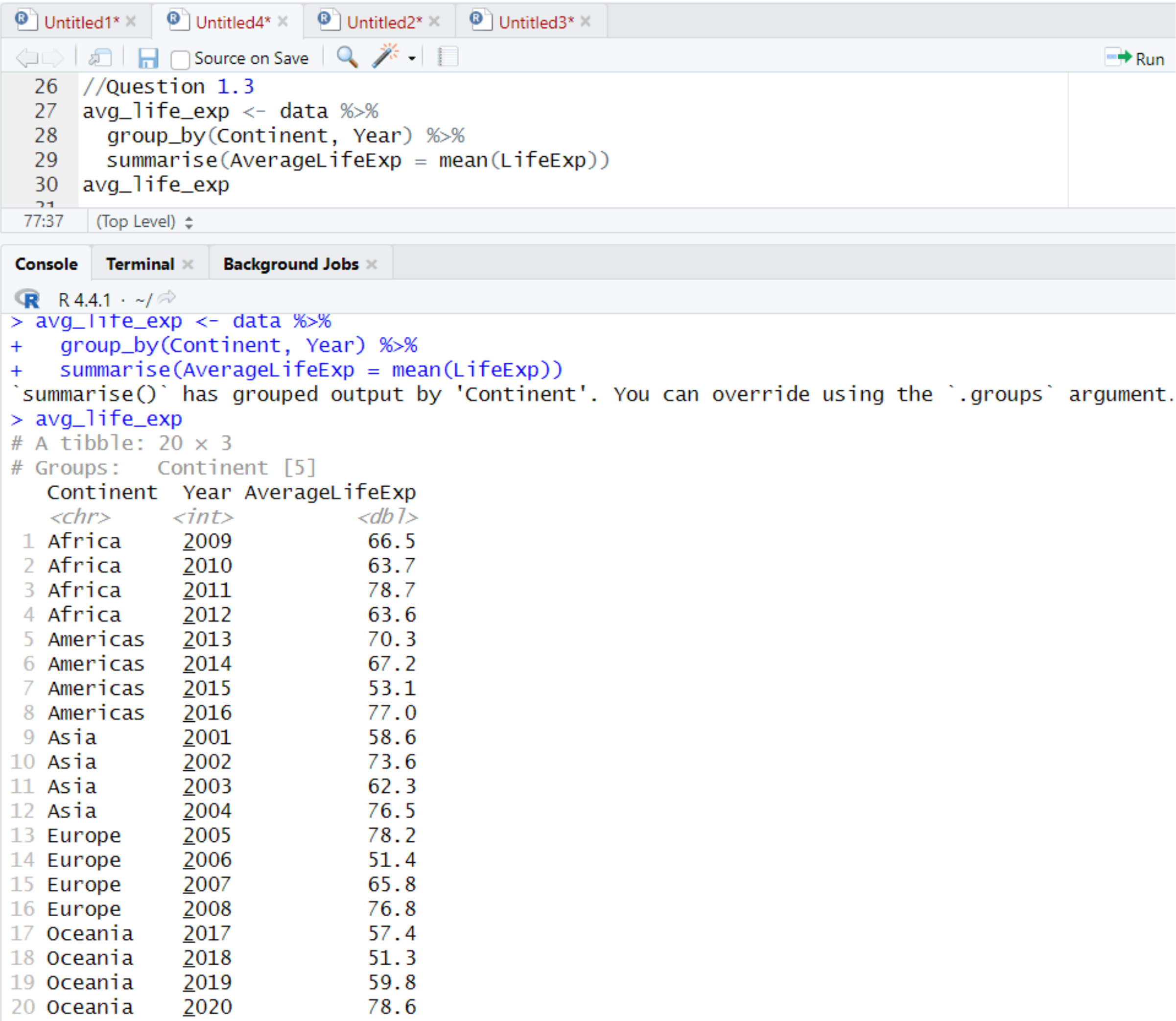
Question 1.2



```
//Question 1.2
lowest_gdp_europe <- data %>%
  filter(Continent == "Europe") %>%
  filter(gdpPerc == min(gdpPerc))
lowest_gdp_europe

```

```
> lowest_gdp_europe <- data %>%
+   filter(Continent == "Europe") %>%
+   filter(gdpPerc == min(gdpPerc))
> lowest_gdp_europe
   Country Continent Year  LifeExp     Pop  gdpPerc
1 Country8    Europe 2008 76.77257 2498010 10196.86
```

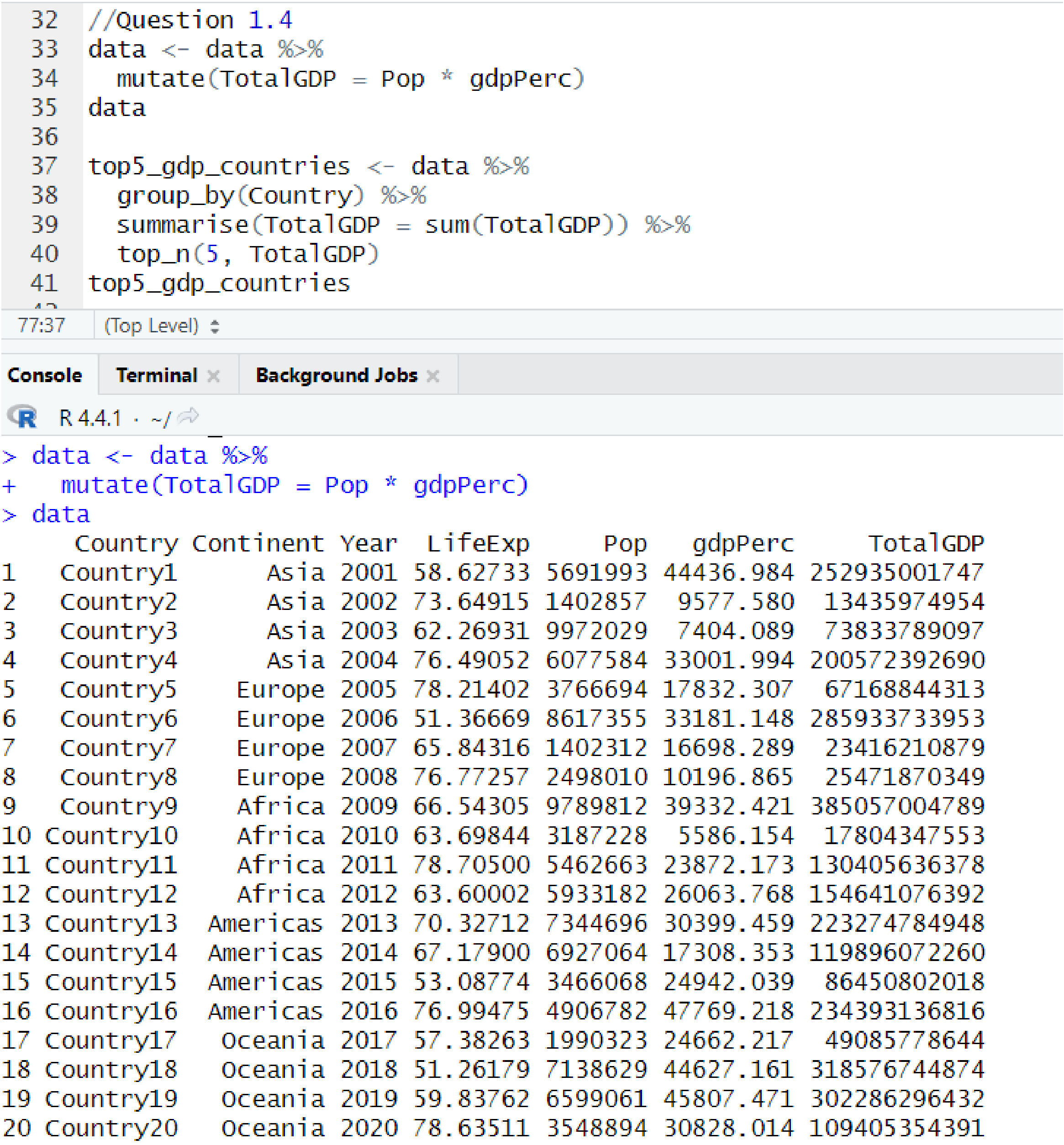
Question 1.3



```
//Question 1.3
avg_life_exp <- data %>%
  group_by(Continent, Year) %>%
  summarise(AverageLifeExp = mean(LifeExp))
avg_life_exp
```

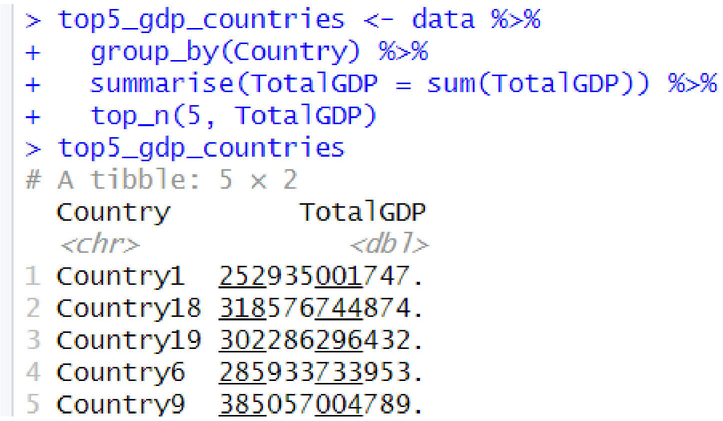
```
> avg_life_exp <- data %>%
+   group_by(Continent, Year) %>%
+   summarise(AverageLifeExp = mean(LifeExp))
`summarise()` has grouped output by 'Continent'. You can override using the `.groups` argument.
> avg_life_exp
# A tibble: 20 × 3
# Groups:   Continent [5]
   Continent  Year AverageLifeExp
   <chr>     <int>          <dbl>
 1 Africa     2009           66.5
 2 Africa     2010           63.7
 3 Africa     2011           78.7
 4 Africa     2012           63.6
 5 Americas   2013           70.3
 6 Americas   2014           67.2
 7 Americas   2015           53.1
 8 Americas   2016           77.0
 9 Asia       2001           58.6
10 Asia       2002           73.6
11 Asia       2003           62.3
12 Asia       2004           76.5
13 Europe     2005           78.2
14 Europe     2006           51.4
15 Europe     2007           65.8
16 Europe     2008           76.8
17 Oceania    2017           57.4
18 Oceania    2018           51.3
19 Oceania    2019           59.8
20 Oceania    2020           78.6
```

Question 1.4



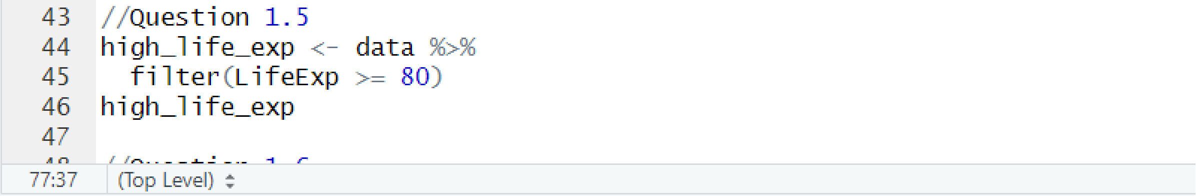
```
//Question 1.4
data <- data %>%
  mutate(TotalGDP = Pop * gdpPerc)
data

top5_gdp_countries <- data %>%
  group_by(Country) %>%
  summarise(TotalGDP = sum(TotalGDP)) %>%
  top_n(5, TotalGDP)
top5_gdp_countries
```

```
> data <- data %>%
+   mutate(TotalGDP = Pop * gdpPerc)
> data
     Country Continent Year  LifeExp     Pop   gdpPerc     TotalGDP
1   Country1      Asia 2001 58.62733 5691993 44436.984 252935001747
2   Country2      Asia 2002 73.64915 1402857  9577.580  13435974954
3   Country3      Asia 2003 62.26931 9972029  7404.089  73833789097
4   Country4      Asia 2004 76.49052 6077584 33001.994 200572392690
5   Country5    Europe 2005 78.21402 3766694 17832.307  67168844313
6   Country6    Europe 2006 51.36669 8617355 33181.148 285933733953
7   Country7    Europe 2007 65.84316 1402312 16698.289  23416210879
8   Country8    Europe 2008 76.77257 2498010 10196.865  25471870349
9   Country9    Africa 2009 66.54305 9789812 39332.421 385057004789
10 Country10    Africa 2010 63.69844 3187228  5586.154  17804347553
11 Country11    Africa 2011 78.70500 5462663 23872.173 130405636378
12 Country12    Africa 2012 63.60002 5933182 26063.768 154641076392
13 Country13  Americas 2013 70.32712 7344696 30399.459 223274784948
14 Country14  Americas 2014 67.17900 6927064 17308.353 119896072260
15 Country15  Americas 2015 53.08774 3466068 24942.039  86450802018
16 Country16  Americas 2016 76.99475 4906782 47769.218 234393136816
17 Country17   Oceania 2017 57.38263 1990323 24662.217  49085778644
18 Country18   Oceania 2018 51.26179 7138629 44627.161 318576744874
19 Country19   Oceania 2019 59.83762 6599061 45807.471 302286296432
20 Country20   Oceania 2020 78.63511 3548894 30828.014 109405354391
```

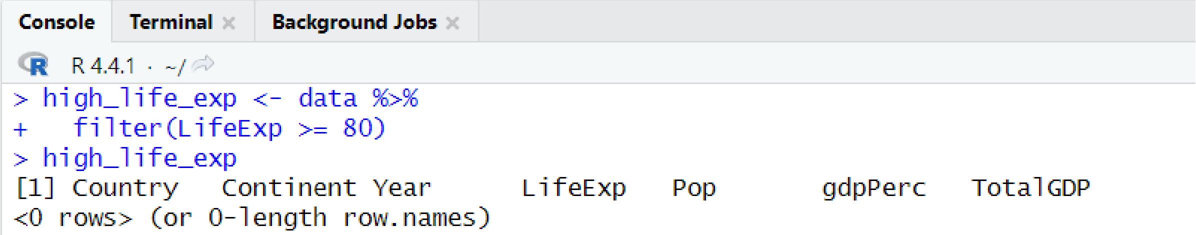


```
> top5_gdp_countries <- data %>%
+   group_by(Country) %>%
+   summarise(TotalGDP = sum(TotalGDP)) %>%
+   top_n(5, TotalGDP)
> top5_gdp_countries
# A tibble: 5 × 2
  Country        TotalGDP
  <chr>             <dbl>
1 Country1  252935001747.
2 Country18 318576744874.
3 Country19 302286296432.
4 Country6  285933733953.
5 Country9  385057004789.
```

Question 1.5

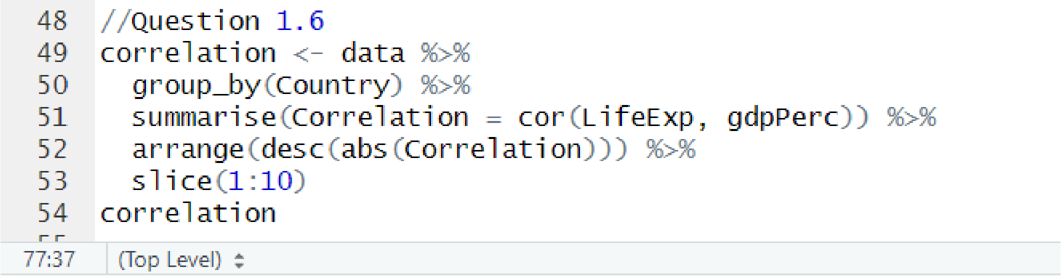


```
//Question 1.5
high_life_exp <- data %>%
  filter(LifeExp >= 80)
high_life_exp

```

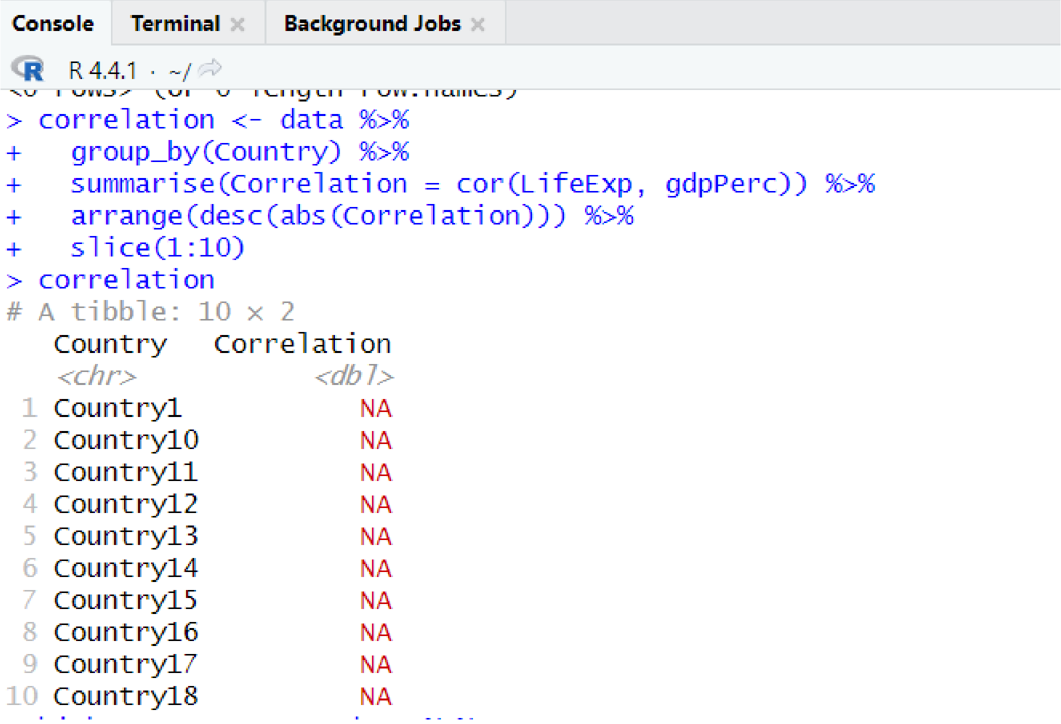


```
> high_life_exp <- data %>%
+   filter(LifeExp >= 80)
> high_life_exp
[1] Country   Continent Year      LifeExp   Pop       gdpPerc   TotalGDP 
<0 rows> (or 0-length row.names)
```

Question 1.6

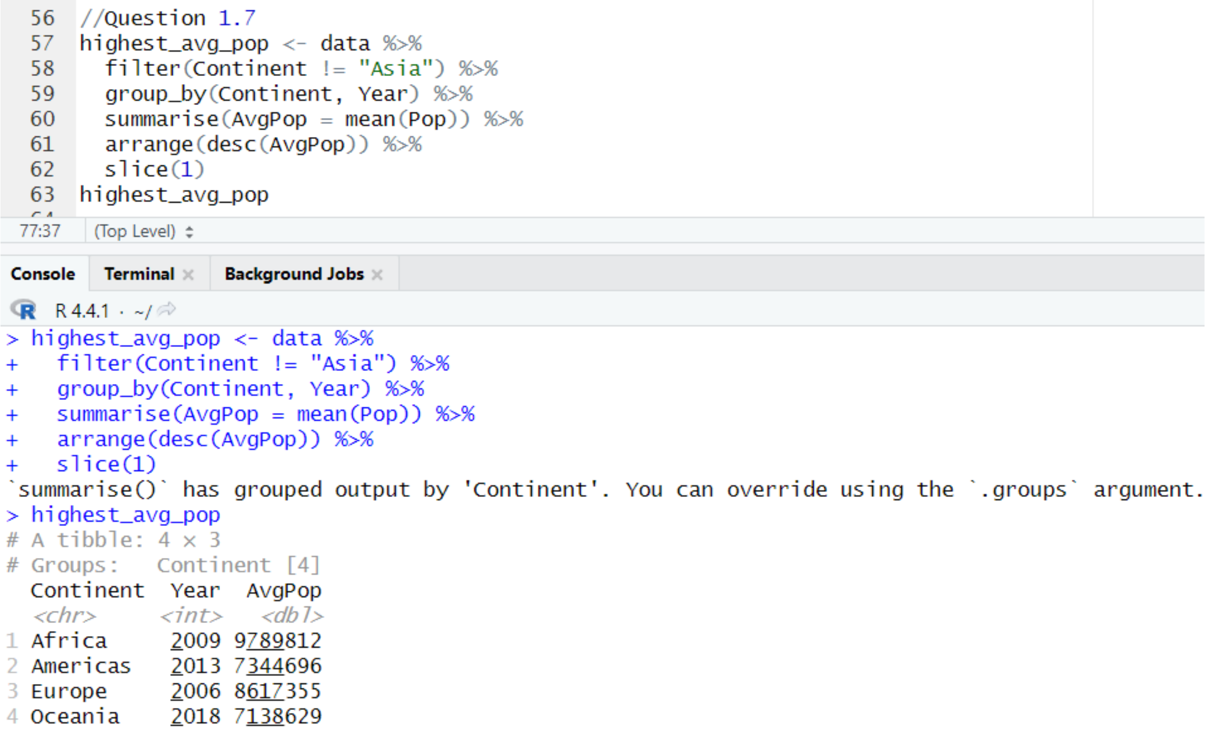


```
//Question 1.6
correlation <- data %>%
  group_by(Country) %>%
  summarise(Correlation = cor(LifeExp, gdpPerc)) %>%
  arrange(desc(abs(Correlation))) %>%
  slice(1:10)
correlation
```



```
> correlation <- data %>%
+   group_by(Country) %>%
+   summarise(Correlation = cor(LifeExp, gdpPerc)) %>%
+   arrange(desc(abs(Correlation))) %>%
+   slice(1:10)
> correlation
# A tibble: 10 × 2
   Country   Correlation
   <chr>           <dbl>
 1 Country1           NA
 2 Country10          NA
 3 Country11          NA
 4 Country12          NA
 5 Country13          NA
 6 Country14          NA
 7 Country15          NA
 8 Country16          NA
 9 Country17          NA
10 Country18          NA
```

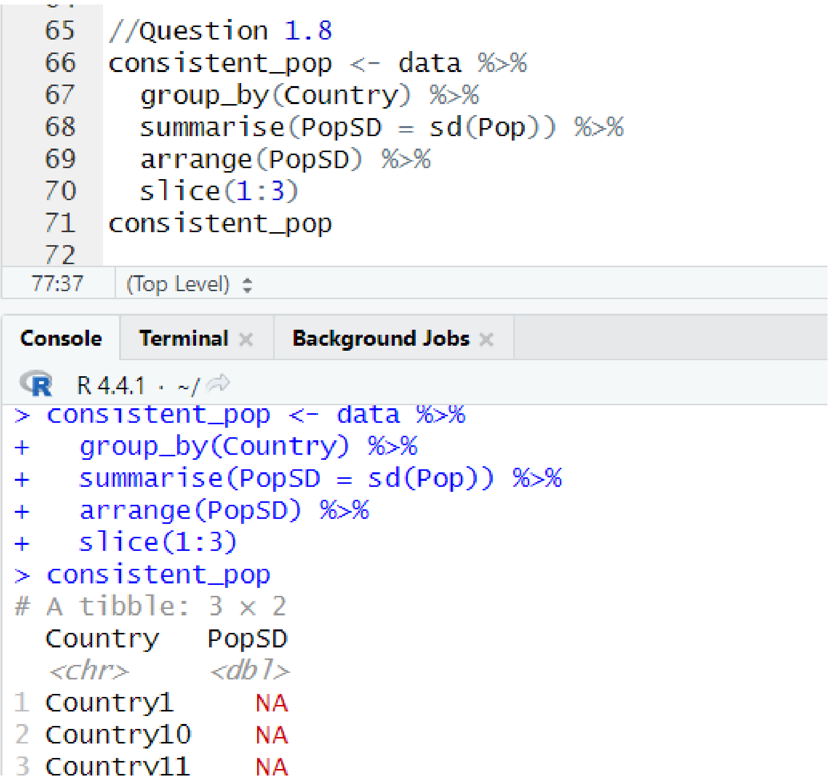
Question 1.7



```
//Question 1.7
highest_avg_pop <- data %>%
  filter(Continent != "Asia") %>%
  group_by(Continent, Year) %>%
  summarise(AvgPop = mean(Pop)) %>%
  arrange(desc(AvgPop)) %>%
  slice(1)
highest_avg_pop
```

```
> highest_avg_pop <- data %>%
+   filter(Continent != "Asia") %>%
+   group_by(Continent, Year) %>%
+   summarise(AvgPop = mean(Pop)) %>%
+   arrange(desc(AvgPop)) %>%
+   slice(1)
`summarise()` has grouped output by 'Continent'. You can override using the `.groups` argument.
> highest_avg_pop
# A tibble: 4 × 3
# Groups:   Continent [4]
  Continent  Year  AvgPop
  <chr>     <int>   <dbl>
1 Africa     2009 9789812
2 Americas   2013 7344696
3 Europe     2006 8617355
4 Oceania    2018 7138629
```

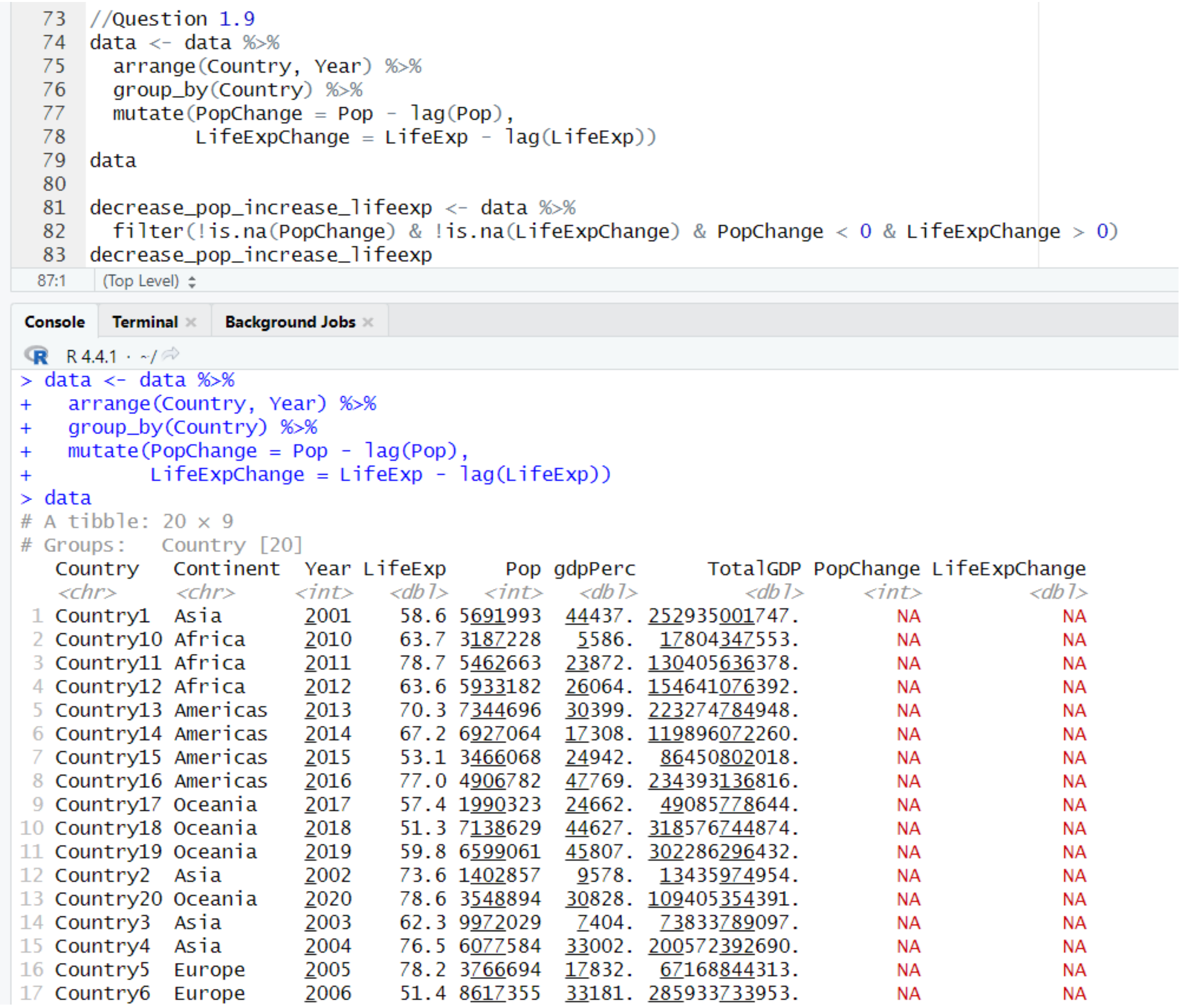
Question 1.8



```
//Question 1.8
consistent_pop <- data %>%
  group_by(Country) %>%
  summarise(PopSD = sd(Pop)) %>%
  arrange(PopSD) %>%
  slice(1:3)
consistent_pop
```

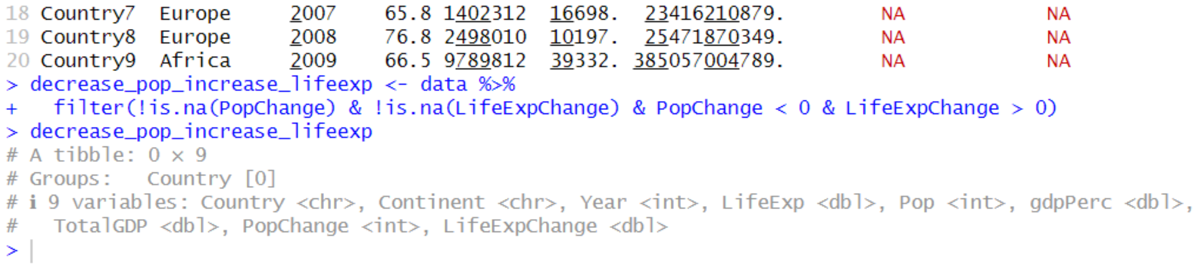
```
> consistent_pop <- data %>%
+   group_by(Country) %>%
+   summarise(PopSD = sd(Pop)) %>%
+   arrange(PopSD) %>%
+   slice(1:3)
> consistent_pop
# A tibble: 3 × 2
  Country   PopSD
  <chr>     <dbl>
1 Country1     NA
2 Country10    NA
3 Country11    NA
```

Question 1.9



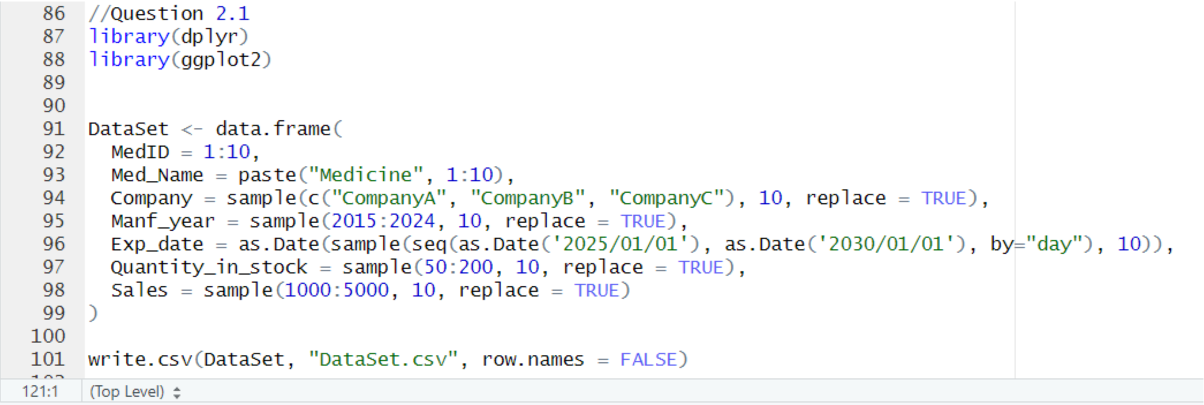
```
//Question 1.9
data <- data %>%
  arrange(Country, Year) %>%
  group_by(Country) %>%
  mutate(PopChange = Pop - lag(Pop),
         LifeExpChange = LifeExp - lag(LifeExp))
data

decrease_pop_increase_lifeexp <- data %>%
  filter(!is.na(PopChange) & !is.na(LifeExpChange) & PopChange < 0 & LifeExpChange > 0)
decrease_pop_increase_lifeexp
```

```
> data <- data %>%
+   arrange(Country, Year) %>%
+   group_by(Country) %>%
+   mutate(PopChange = Pop - lag(Pop),
+          LifeExpChange = LifeExp - lag(LifeExp))
> data
# A tibble: 20 × 9
# Groups:   Country [20]
   Country   Continent  Year LifeExp     Pop gdpPerc      TotalGDP PopChange LifeExpChange
   <chr>     <chr>     <int>   <dbl>   <int>   <dbl>         <dbl>     <int>         <dbl>
 1 Country1  Asia       2001    58.6 5691993  44437. 252935001747.        NA            NA
 2 Country10 Africa     2010    63.7 3187228   5586.  17804347553.        NA            NA
 3 Country11 Africa     2011    78.7 5462663  23872. 130405636378.        NA            NA
 4 Country12 Africa     2012    63.6 5933182  26064. 154641076392.        NA            NA
 5 Country13 Americas   2013    70.3 7344696  30399. 223274784948.        NA            NA
 6 Country14 Americas   2014    67.2 6927064  17308. 119896072260.        NA            NA
 7 Country15 Americas   2015    53.1 3466068  24942.  86450802018.        NA            NA
 8 Country16 Americas   2016    77.0 4906782  47769. 234393136816.        NA            NA
 9 Country17 Oceania    2017    57.4 1990323  24662.  49085778644.        NA            NA
10 Country18 Oceania    2018    51.3 7138629  44627. 318576744874.        NA            NA
11 Country19 Oceania    2019    59.8 6599061  45807. 302286296432.        NA            NA
12 Country2  Asia       2002    73.6 1402857   9578.  13435974954.        NA            NA
13 Country20 Oceania    2020    78.6 3548894  30828. 109405354391.        NA            NA
14 Country3  Asia       2003    62.3 9972029   7404.  73833789097.        NA            NA
15 Country4  Asia       2004    76.5 6077584  33002. 200572392690.        NA            NA
16 Country5  Europe     2005    78.2 3766694  17832.  67168844313.        NA            NA
17 Country6  Europe     2006    51.4 8617355  33181. 285933733953.        NA            NA
```



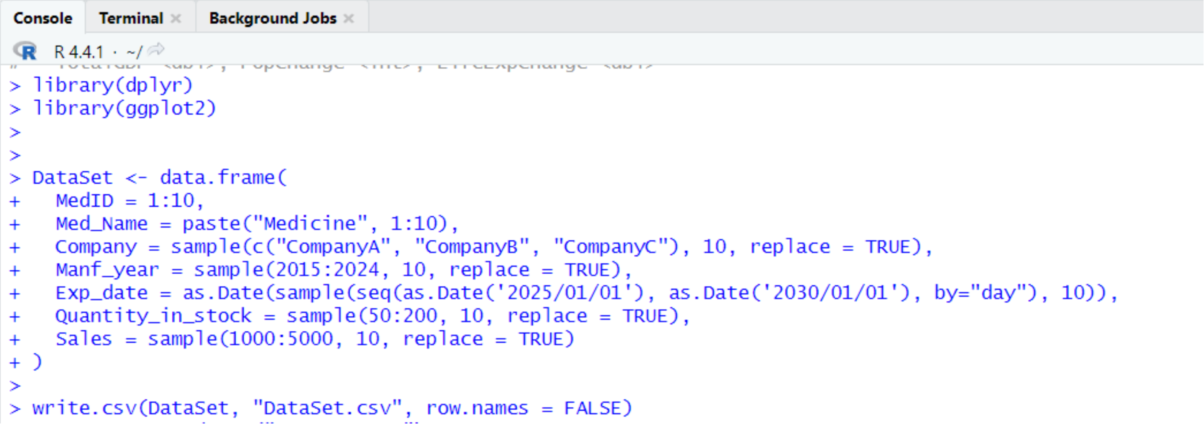
```
18 Country7  Europe     2007    65.8 1402312  16698.  23416210879.        NA            NA
19 Country8  Europe     2008    76.8 2498010  10197.  25471870349.        NA            NA
20 Country9  Africa     2009    66.5 9789812  39332. 385057004789.        NA            NA
> decrease_pop_increase_lifeexp <- data %>%
+   filter(!is.na(PopChange) & !is.na(LifeExpChange) & PopChange < 0 & LifeExpChange > 0)
> decrease_pop_increase_lifeexp
# A tibble: 0 × 9
# Groups:   Country [0]
# ℹ 9 variables: Country <chr>, Continent <chr>, Year <int>, LifeExp <dbl>, Pop <int>, gdpPerc <dbl>,
#   TotalGDP <dbl>, PopChange <int>, LifeExpChange <dbl>
> 
```

Question 2.1



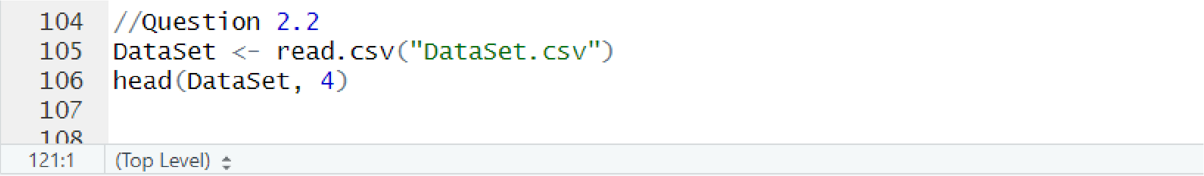
```
//Question 2.1
library(dplyr)
library(ggplot2)


DataSet <- data.frame(
  MedID = 1:10,
  Med_Name = paste("Medicine", 1:10),
  Company = sample(c("CompanyA", "CompanyB", "CompanyC"), 10, replace = TRUE),
  Manf_year = sample(2015:2024, 10, replace = TRUE),
  Exp_date = as.Date(sample(seq(as.Date('2025/01/01'), as.Date('2030/01/01'), by="day"), 10)),
  Quantity_in_stock = sample(50:200, 10, replace = TRUE),
  Sales = sample(1000:5000, 10, replace = TRUE)
)

write.csv(DataSet, "DataSet.csv", row.names = FALSE)
```

121:1 (Top Level) ÷



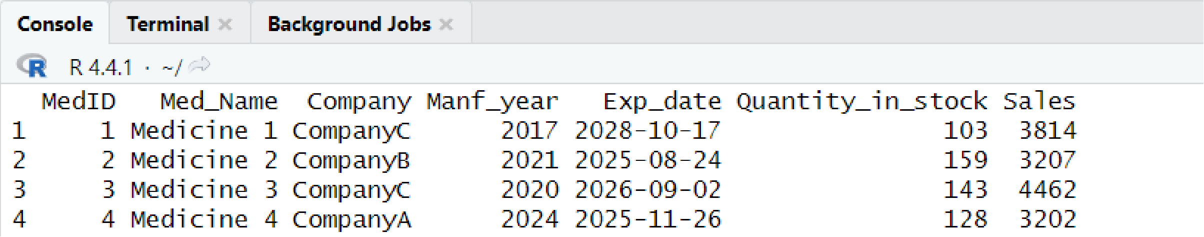
```
> library(dplyr)
> library(ggplot2)
>
>
> DataSet <- data.frame(
+   MedID = 1:10,
+   Med_Name = paste("Medicine", 1:10),
+   Company = sample(c("CompanyA", "CompanyB", "CompanyC"), 10, replace = TRUE),
+   Manf_year = sample(2015:2024, 10, replace = TRUE),
+   Exp_date = as.Date(sample(seq(as.Date('2025/01/01'), as.Date('2030/01/01'), by="day"), 10)),
+   Quantity_in_stock = sample(50:200, 10, replace = TRUE),
+   Sales = sample(1000:5000, 10, replace = TRUE)
+ )
>
> write.csv(DataSet, "DataSet.csv", row.names = FALSE)
```

Question 2.2



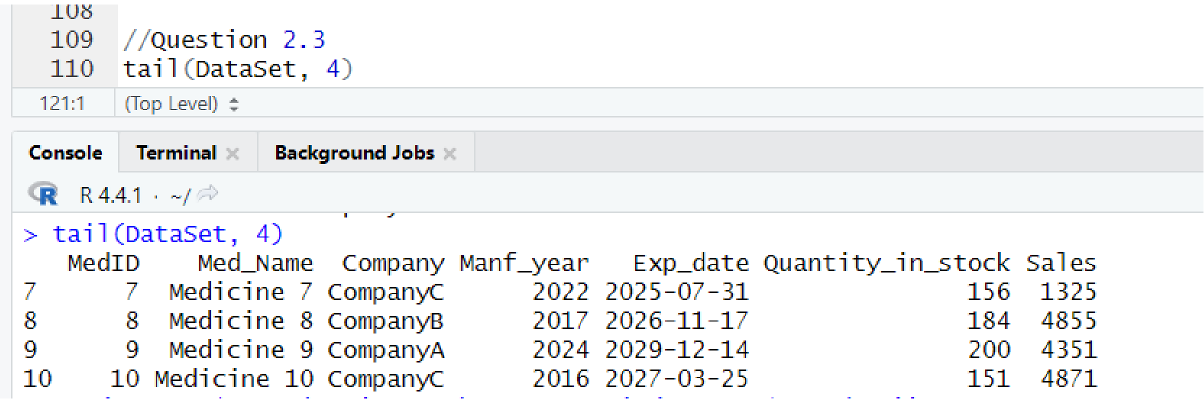
```
//Question 2.2
DataSet <- read.csv("DataSet.csv")
head(DataSet, 4)

```

121:1 (Top Level) ÷



```
  MedID   Med_Name  Company Manf_year   Exp_date Quantity_in_stock Sales
1     1 Medicine 1 CompanyC      2017 2028-10-17               103  3814
2     2 Medicine 2 CompanyB      2021 2025-08-24               159  3207
3     3 Medicine 3 CompanyC      2020 2026-09-02               143  4462
4     4 Medicine 4 CompanyA      2024 2025-11-26               128  3202
```

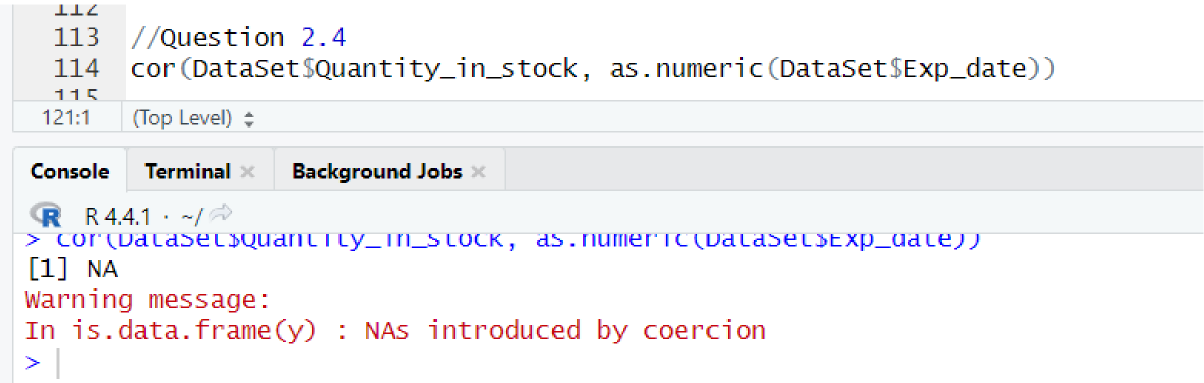
Question 2.3



```
//Question 2.3
tail(DataSet, 4)
```

```
> tail(DataSet, 4)
   MedID    Med_Name  Company Manf_year   Exp_date Quantity_in_stock Sales
7      7  Medicine 7 CompanyC      2022 2025-07-31               156  1325
8      8  Medicine 8 CompanyB      2017 2026-11-17               184  4855
9      9  Medicine 9 CompanyA      2024 2029-12-14               200  4351
10    10 Medicine 10 CompanyC      2016 2027-03-25               151  4871
```

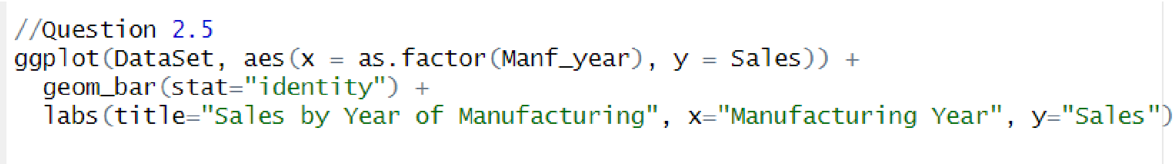
Question 2.4



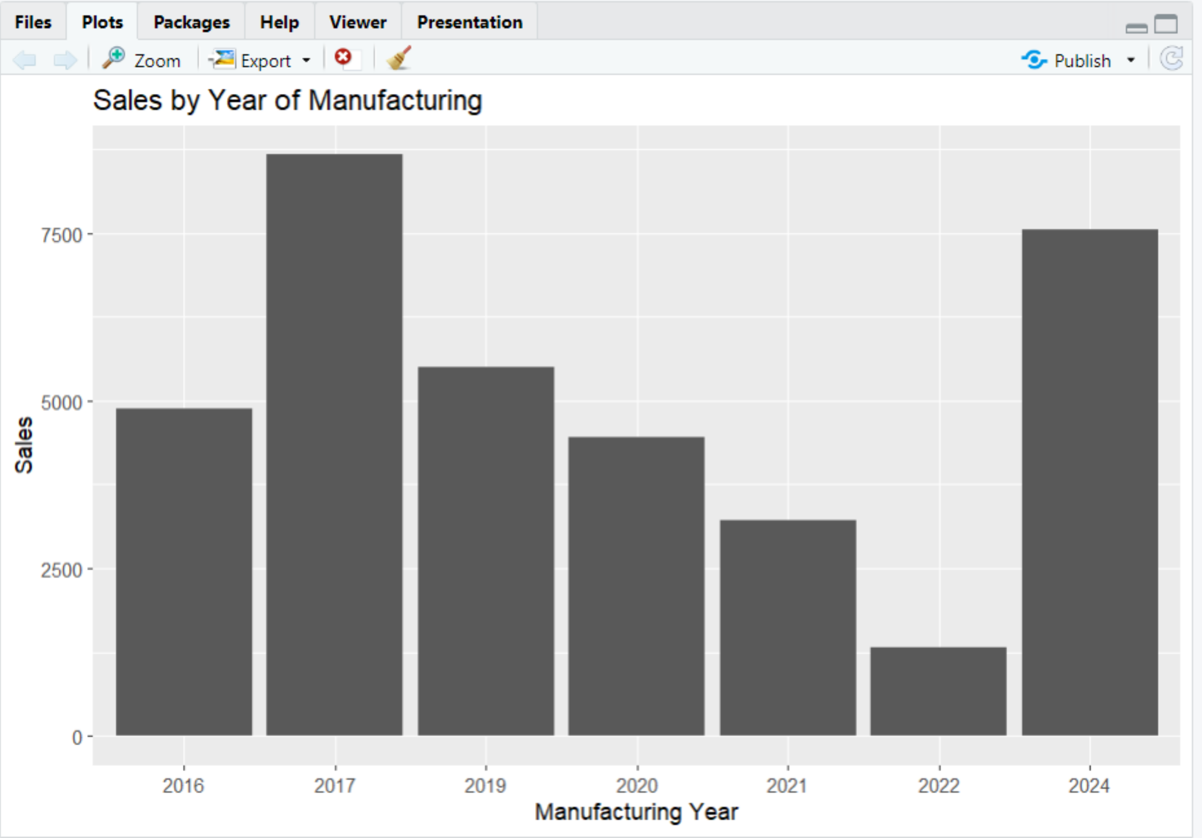
```
//Question 2.4
cor(DataSet$Quantity_in_stock, as.numeric(DataSet$Exp_date))
```

```
> cor(DataSet$Quantity_in_stock, as.numeric(DataSet$Exp_date))
[1] NA
Warning message:
In is.data.frame(y) : NAs introduced by coercion
> 
```

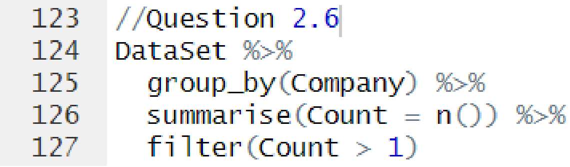
Question 2.5



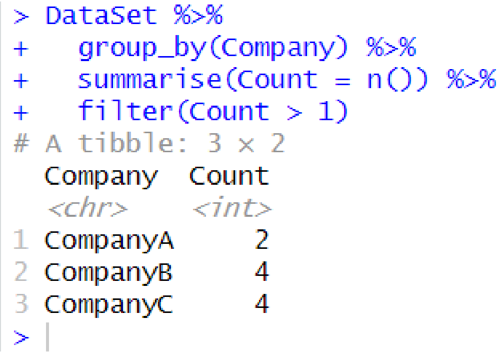
```
//Question 2.5
ggplot(DataSet, aes(x = as.factor(Manf_year), y = Sales)) +
  geom_bar(stat="identity") +
  labs(title="Sales by Year of Manufacturing", x="Manufacturing Year", y="Sales")
```



Question 2.6

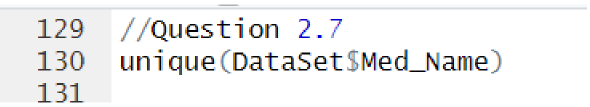


```
//Question 2.6
DataSet %>%
  group_by(Company) %>%
  summarise(Count = n()) %>%
  filter(Count > 1)
```

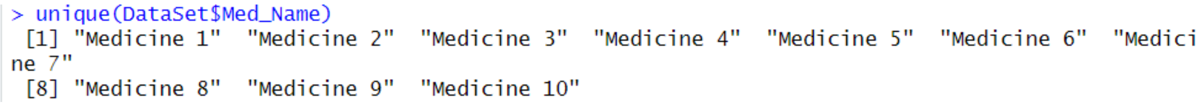


```
> DataSet %>%
+   group_by(Company) %>%
+   summarise(Count = n()) %>%
+   filter(Count > 1)
# A tibble: 3 × 2
  Company  Count
  <chr>    <int>
1 CompanyA     2
2 CompanyB     4
3 CompanyC     4
> 
```

Question 2.7

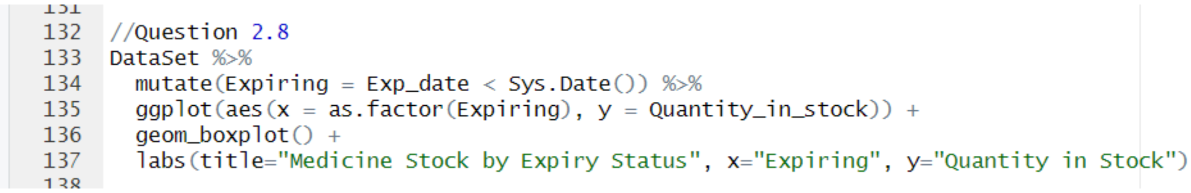


```
//Question 2.7
unique(DataSet$Med_Name)

```

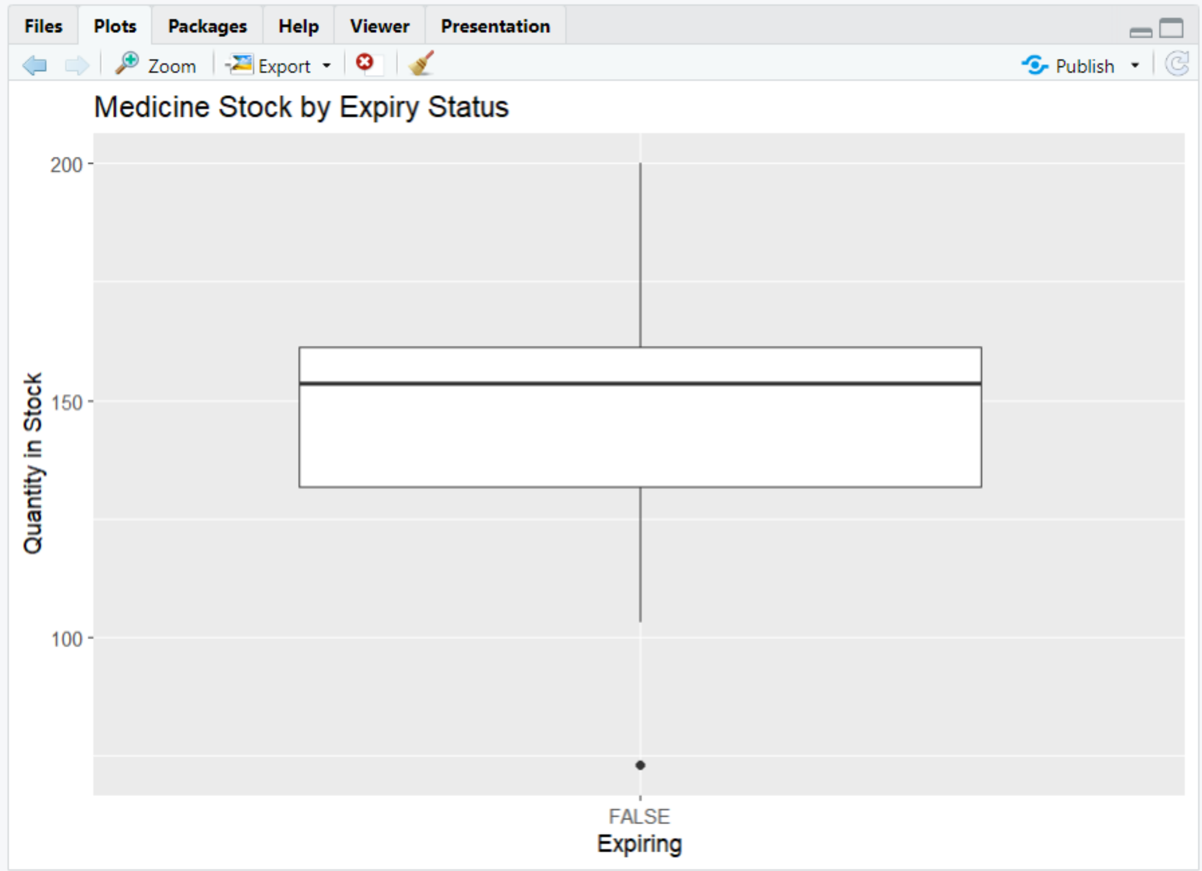


```
> unique(DataSet$Med_Name)
 [1] "Medicine 1"  "Medicine 2"  "Medicine 3"  "Medicine 4"  "Medicine 5"  "Medicine 6"  "Medici
ne 7"
 [8] "Medicine 8"  "Medicine 9"  "Medicine 10"
```

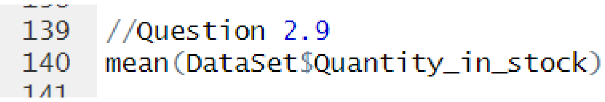
Question 2.8



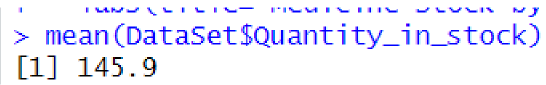
```
//Question 2.8
DataSet %>%
  mutate(Expiring = Exp_date < Sys.Date()) %>%
  ggplot(aes(x = as.factor(Expiring), y = Quantity_in_stock)) +
  geom_boxplot() +
  labs(title="Medicine Stock by Expiry Status", x="Expiring", y="Quantity in Stock")
```



Question 2.9

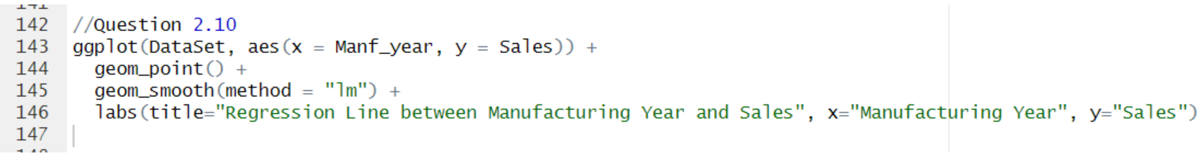


```
//Question 2.9
mean(DataSet$Quantity_in_stock)
```

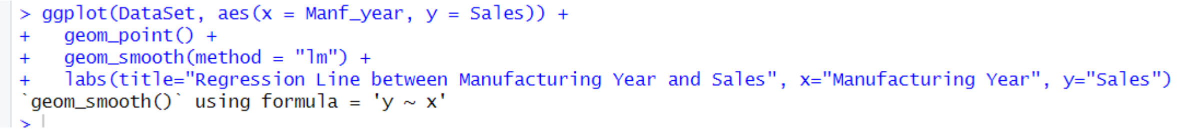


```
> mean(DataSet$Quantity_in_stock)
[1] 145.9
```

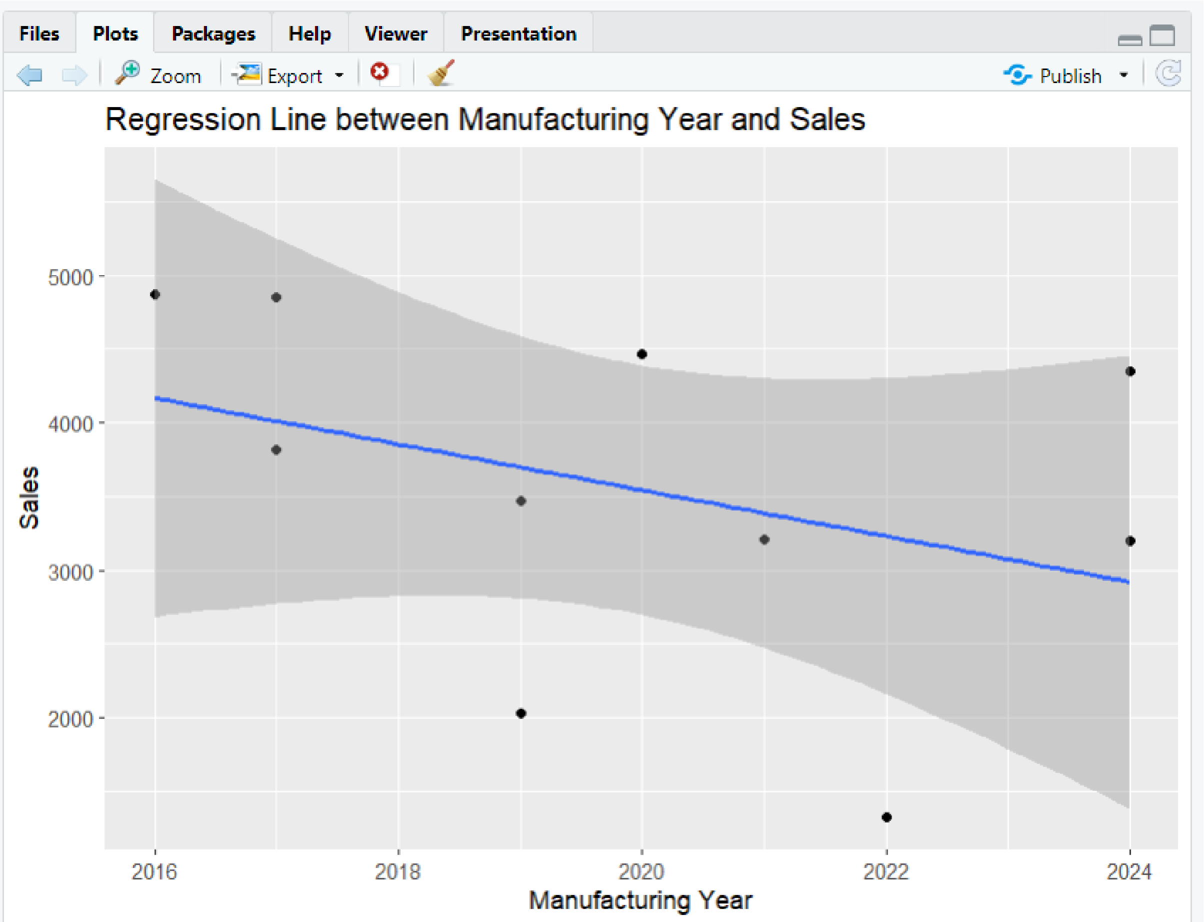
Question 2.10



```
//Question 2.10
ggplot(DataSet, aes(x = Manf_year, y = Sales)) +
  geom_point() +
  geom_smooth(method = "lm") +
  labs(title="Regression Line between Manufacturing Year and Sales", x="Manufacturing Year", y="Sales")
```



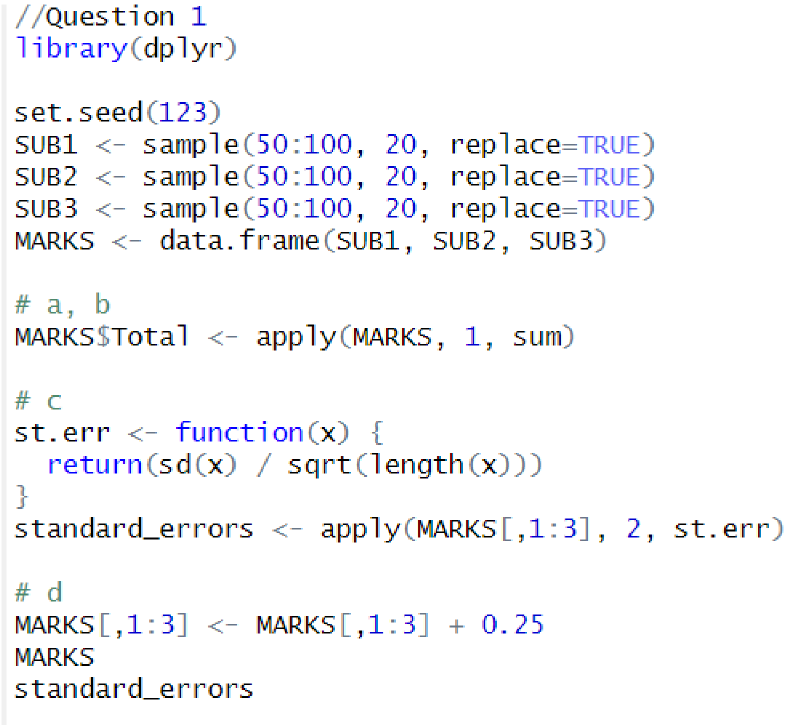
```
> ggplot(DataSet, aes(x = Manf_year, y = Sales)) +
+   geom_point() +
+   geom_smooth(method = "lm") +
+   labs(title="Regression Line between Manufacturing Year and Sales", x="Manufacturing Year", y="Sales")
`geom_smooth()` using formula = 'y ~ x'
>
```


Files
Plots
Packages
Help
Viewer
Presentation
Zoom
Export
Publish
Regression Line between Manufacturing Year and Sales
5000
4000
3000
2000
Sales
2016
2018
2020
2022
2024
Manufacturing Year

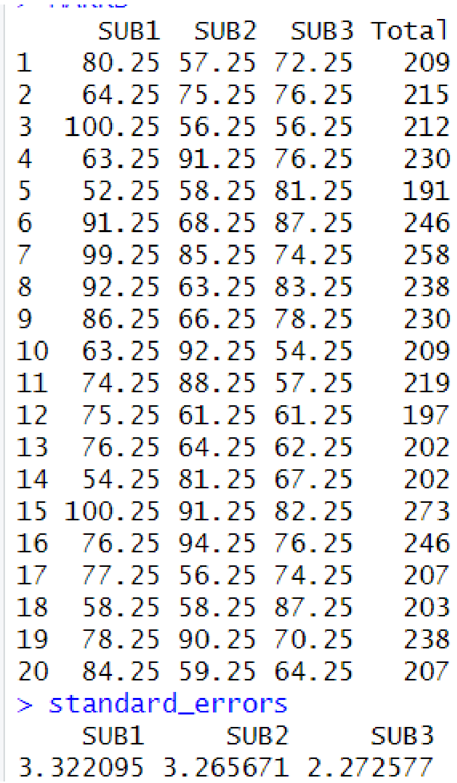
LAB ASSIGNMENT 7

Submitted By: Kamya Mehra 102213026 3CO35

Question 1

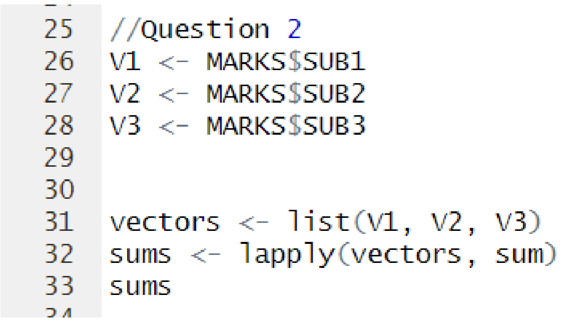


```
//Question 1
library(dplyr)

set.seed(123)
SUB1 <- sample(50:100, 20, replace=TRUE)
SUB2 <- sample(50:100, 20, replace=TRUE)
SUB3 <- sample(50:100, 20, replace=TRUE)
MARKS <- data.frame(SUB1, SUB2, SUB3)

# a, b
MARKS$Total <- apply(MARKS, 1, sum)

# c
st.err <- function(x) {
  return(sd(x) / sqrt(length(x)))
}
standard_errors <- apply(MARKS[,1:3], 2, st.err)

# d
MARKS[,1:3] <- MARKS[,1:3] + 0.25
MARKS
standard_errors
```

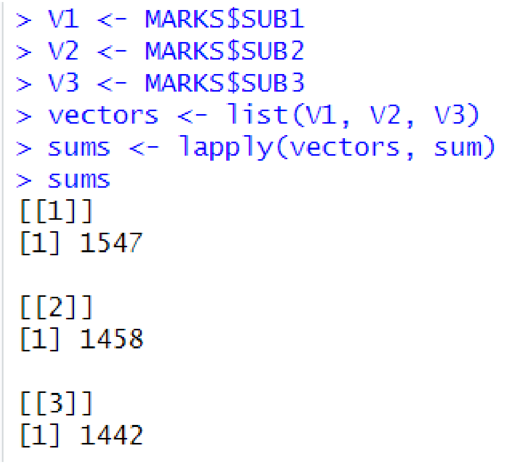


```
> MARKS
     SUB1  SUB2  SUB3 Total
1   80.25 57.25 72.25   209
2   64.25 75.25 76.25   215
3  100.25 56.25 56.25   212
4   63.25 91.25 76.25   230
5   52.25 58.25 81.25   191
6   91.25 68.25 87.25   246
7   99.25 85.25 74.25   258
8   92.25 63.25 83.25   238
9   86.25 66.25 78.25   230
10  63.25 92.25 54.25   209
11  74.25 88.25 57.25   219
12  75.25 61.25 61.25   197
13  76.25 64.25 62.25   202
14  54.25 81.25 67.25   202
15 100.25 91.25 82.25   273
16  76.25 94.25 76.25   246
17  77.25 56.25 74.25   207
18  58.25 58.25 87.25   203
19  78.25 90.25 70.25   238
20  84.25 59.25 64.25   207
> standard_errors
    SUB1     SUB2     SUB3 
3.322095 3.265671 2.272577 
```

Question 2

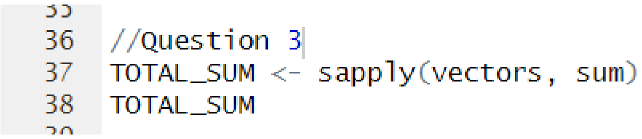


```
//Question 2
V1 <- MARKS$SUB1
V2 <- MARKS$SUB2
V3 <- MARKS$SUB3


vectors <- list(V1, V2, V3)
sums <- lapply(vectors, sum)
sums
```

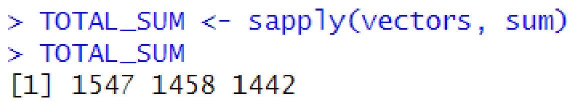


```
> V1 <- MARKS$SUB1
> V2 <- MARKS$SUB2
> V3 <- MARKS$SUB3
> vectors <- list(V1, V2, V3)
> sums <- lapply(vectors, sum)
> sums
[[1]]
[1] 1547

[[2]]
[1] 1458

[[3]]
[1] 1442
```

Question 3

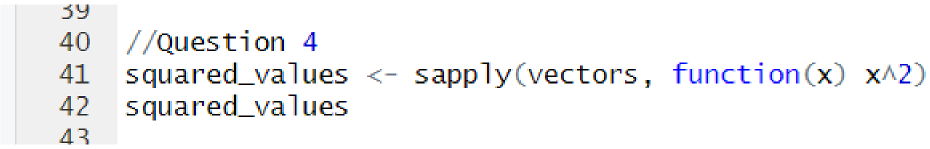


```
//Question 3
TOTAL_SUM <- sapply(vectors, sum)
TOTAL_SUM
```

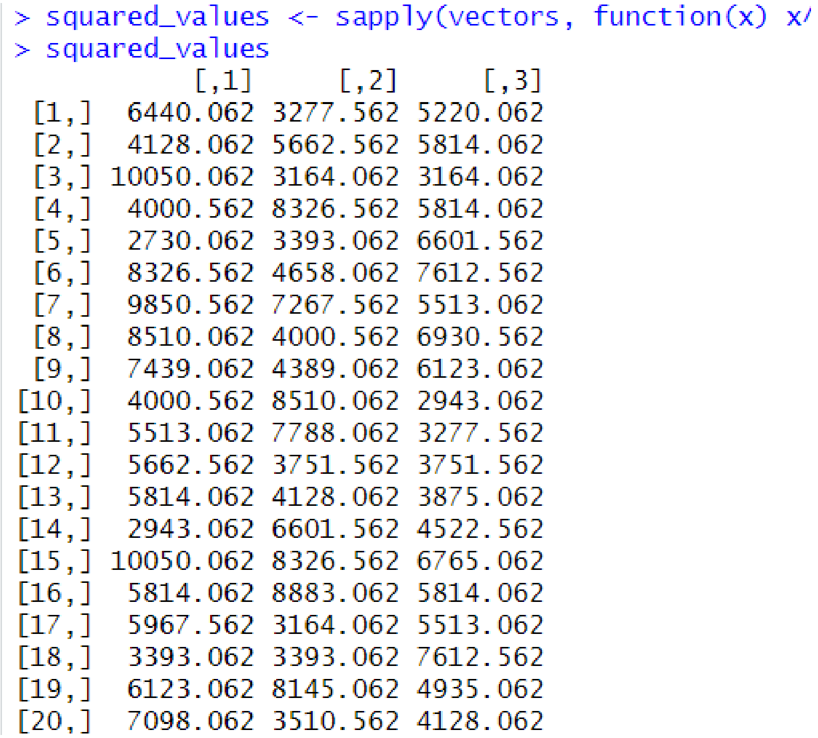


```
> TOTAL_SUM <- sapply(vectors, sum)
> TOTAL_SUM
[1] 1547 1458 1442
```

Question 4

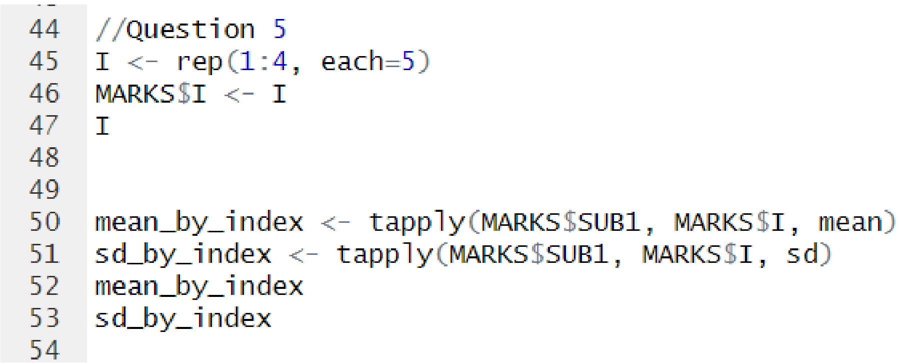


```
//Question 4
squared_values <- sapply(vectors, function(x) x^2)
squared_values
```

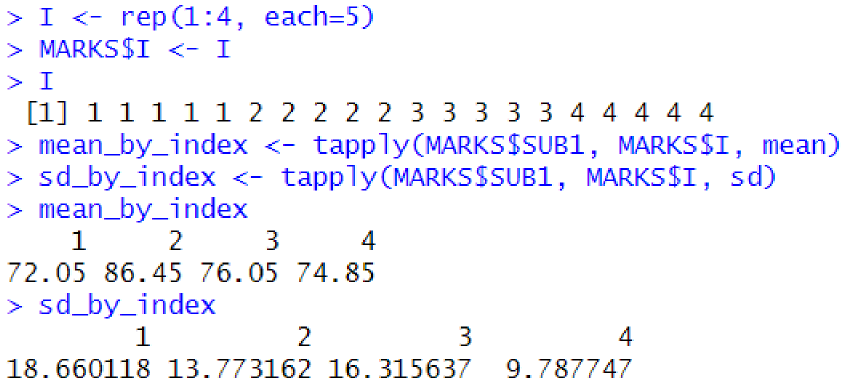


```
> squared_values <- sapply(vectors, function(x) x/
> squared_values
          [,1]     [,2]     [,3]
 [1,]  6440.062 3277.562 5220.062
 [2,]  4128.062 5662.562 5814.062
 [3,] 10050.062 3164.062 3164.062
 [4,]  4000.562 8326.562 5814.062
 [5,]  2730.062 3393.062 6601.562
 [6,]  8326.562 4658.062 7612.562
 [7,]  9850.562 7267.562 5513.062
 [8,]  8510.062 4000.562 6930.562
 [9,]  7439.062 4389.062 6123.062
[10,]  4000.562 8510.062 2943.062
[11,]  5513.062 7788.062 3277.562
[12,]  5662.562 3751.562 3751.562
[13,]  5814.062 4128.062 3875.062
[14,]  2943.062 6601.562 4522.562
[15,] 10050.062 8326.562 6765.062
[16,]  5814.062 8883.062 5814.062
[17,]  5967.562 3164.062 5513.062
[18,]  3393.062 3393.062 7612.562
[19,]  6123.062 8145.062 4935.062
[20,]  7098.062 3510.562 4128.062
```

Question 5

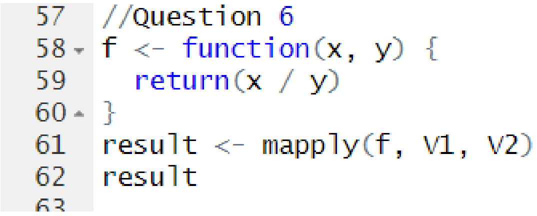


```
//Question 5
I <- rep(1:4, each=5)
MARKS$I <- I
I


mean_by_index <- tapply(MARKS$SUB1, MARKS$I, mean)
sd_by_index <- tapply(MARKS$SUB1, MARKS$I, sd)
mean_by_index
sd_by_index

```

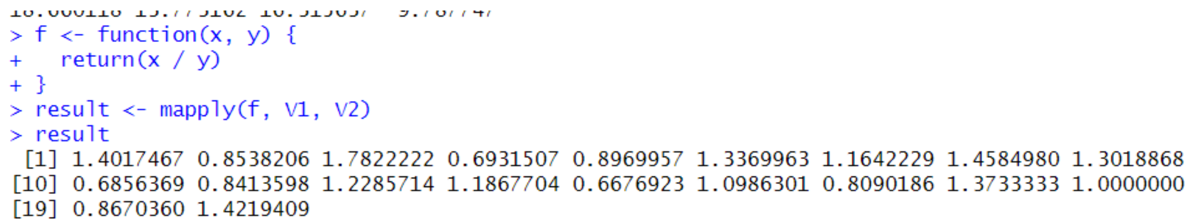


```
> I <- rep(1:4, each=5)
> MARKS$I <- I
> I
 [1] 1 1 1 1 1 2 2 2 2 2 3 3 3 3 3 4 4 4 4 4
> mean_by_index <- tapply(MARKS$SUB1, MARKS$I, mean)
> sd_by_index <- tapply(MARKS$SUB1, MARKS$I, sd)
> mean_by_index
    1     2     3     4 
72.05 86.45 76.05 74.85 
> sd_by_index
        1         2         3         4 
18.660118 13.773162 16.315637  9.787747 
```

Question 6

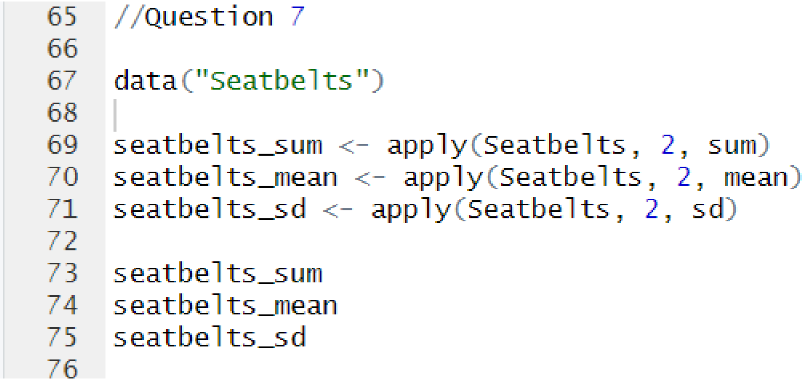


```
//Question 6
f <- function(x, y) {
  return(x / y)
}
result <- mapply(f, V1, V2)
result
```

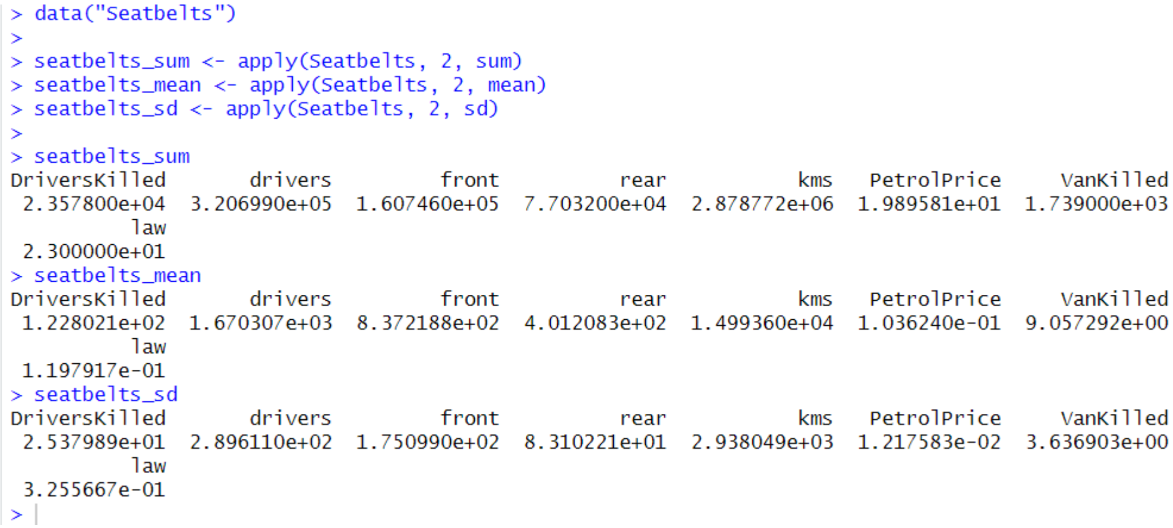


```
18.000118 15.775102 16.515057  9.787747
> f <- function(x, y) {
+   return(x / y)
+ }
> result <- mapply(f, V1, V2)
> result
 [1] 1.4017467 0.8538206 1.7822222 0.6931507 0.8969957 1.3369963 1.1642229 1.4584980 1.3018868
[10] 0.6856369 0.8413598 1.2285714 1.1867704 0.6676923 1.0986301 0.8090186 1.3733333 1.0000000
[19] 0.8670360 1.4219409
```

Question 7



```
//Question 7

data("Seatbelts")

seatbelts_sum <- apply(Seatbelts, 2, sum)
seatbelts_mean <- apply(Seatbelts, 2, mean)
seatbelts_sd <- apply(Seatbelts, 2, sd)

seatbelts_sum
seatbelts_mean
seatbelts_sd

```



```
> data("Seatbelts")
> 
> seatbelts_sum <- apply(Seatbelts, 2, sum)
> seatbelts_mean <- apply(Seatbelts, 2, mean)
> seatbelts_sd <- apply(Seatbelts, 2, sd)
> 
> seatbelts_sum
DriversKilled       drivers         front          rear           kms   PetrolPrice     VanKilled 
 2.357800e+04  3.206990e+05  1.607460e+05  7.703200e+04  2.878772e+06  1.989581e+01  1.739000e+03 
          law 
 2.300000e+01 
> seatbelts_mean
DriversKilled       drivers         front          rear           kms   PetrolPrice     VanKilled 
 1.228021e+02  1.670307e+03  8.372188e+02  4.012083e+02  1.499360e+04  1.036240e-01  9.057292e+00 
          law 
 1.197917e-01 
> seatbelts_sd
DriversKilled       drivers         front          rear           kms   PetrolPrice     VanKilled 
 2.537989e+01  2.896110e+02  1.750990e+02  8.310221e+01  2.938049e+03  1.217583e-02  3.636903e+00 
          law 
 3.255667e-01 
> 
```