AIM: BASIC R COMMANDS

THEORY:

Introduction to R:

R is a language and environment for statistical computing and graphics. It is a GNU project which is similar to the S language and environment which was developed at Bell Laboratories (formerly AT&T, now Lucent Technologies) by John Chambers and colleagues. R can be considered as a different implementation of S. There are some important differences, but much code written for S runs unaltered under R.

R provides a wide variety of statistical (linear and nonlinear modelling, classical statistical tests, time-series analysis, classification, clustering, …) and graphical techniques, and is highly extensible. The S language is often the vehicle of choice for research in statistical methodology, and R provides an Open Source route to participation in that activity. One of R's strengths is the ease with which well-designed publication-quality plots can be produced, including mathematical symbols and formulae where needed. Great care has been taken over the defaults for the minor design choices in graphics, but the user retains full control.

R is available as Free Software under the terms of the Free Software Foundation's

GNU General Public License in source code form. It compiles and runs on a wide variety of UNIX platforms and similar systems (including FreeBSD and Linux), Windows and MacOS.

R environment:

R is an integrated suite of software facilities for data manipulation, calculation and graphical display. It includes

- ✓ an effective data handling and storage facility,
- ✓ a suite of operators for calculations on arrays, in particular matrices,
- ✓ a large, coherent, integrated collection of intermediate tools for data analysis,
- ✓ graphical facilities for data analysis and display either on-screen or on hardcopy, and
- ✓ a well-developed, simple and effective programming language which includes conditionals, loops, user-defined recursive functions and input and output facilities.

The term "environment" is intended to characterize it as a fully planned and coherent system, rather than an incremental accretion of very specific and inflexible tools, as is frequently the case with other data analysis software.

R, like S, is designed around a true computer language, and it allows users to add additional functionality by defining new functions. Much of the system is itself written in the R dialect of S, which makes it easy for users to follow the algorithmic choices made. For computationally intensive tasks, C, C++ and Fortran code can be linked and called at run time. Advanced users can write C code to manipulate R objects directly.

Many users think of R as a statistics system. We prefer to think of it as an environment within which statistical techniques are implemented. R can be extended (easily) via packages. There are about eight packages supplied with the R distribution and many more are available through the CRAN family of Internet sites covering a very wide range of modern statistics.

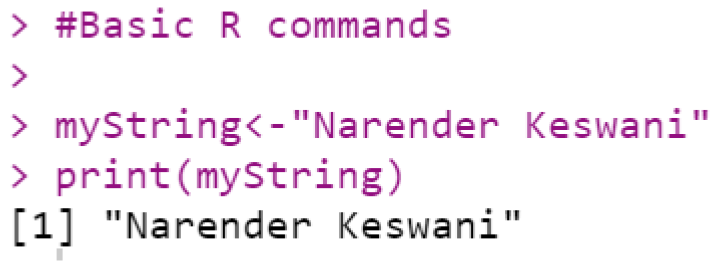
R has its own LaTeX-like documentation format, which is used to supply comprehensive documentation, both on-line in a number of formats and in hardcopy.

A) PRINTING STRING:

SOURCE CODE:

```
myString<-"Narender Keswani"
print(myString)
```

OUTPUT:



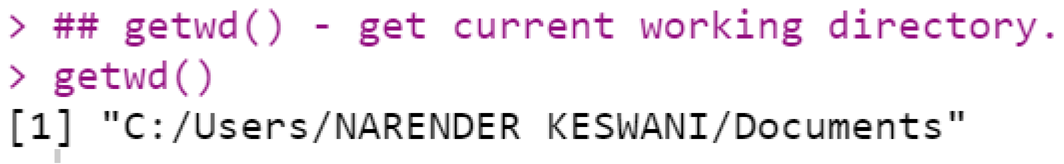
```
> #Basic R commands
>
> myString<-"Narender Keswani"
> print(myString)
[1] "Narender Keswani"
```

B) GET CURRENT WORKING DIRECTORY:

SOURCE CODE:

```
## getwd() - get current working directory.
getwd()
```

OUTPUT:



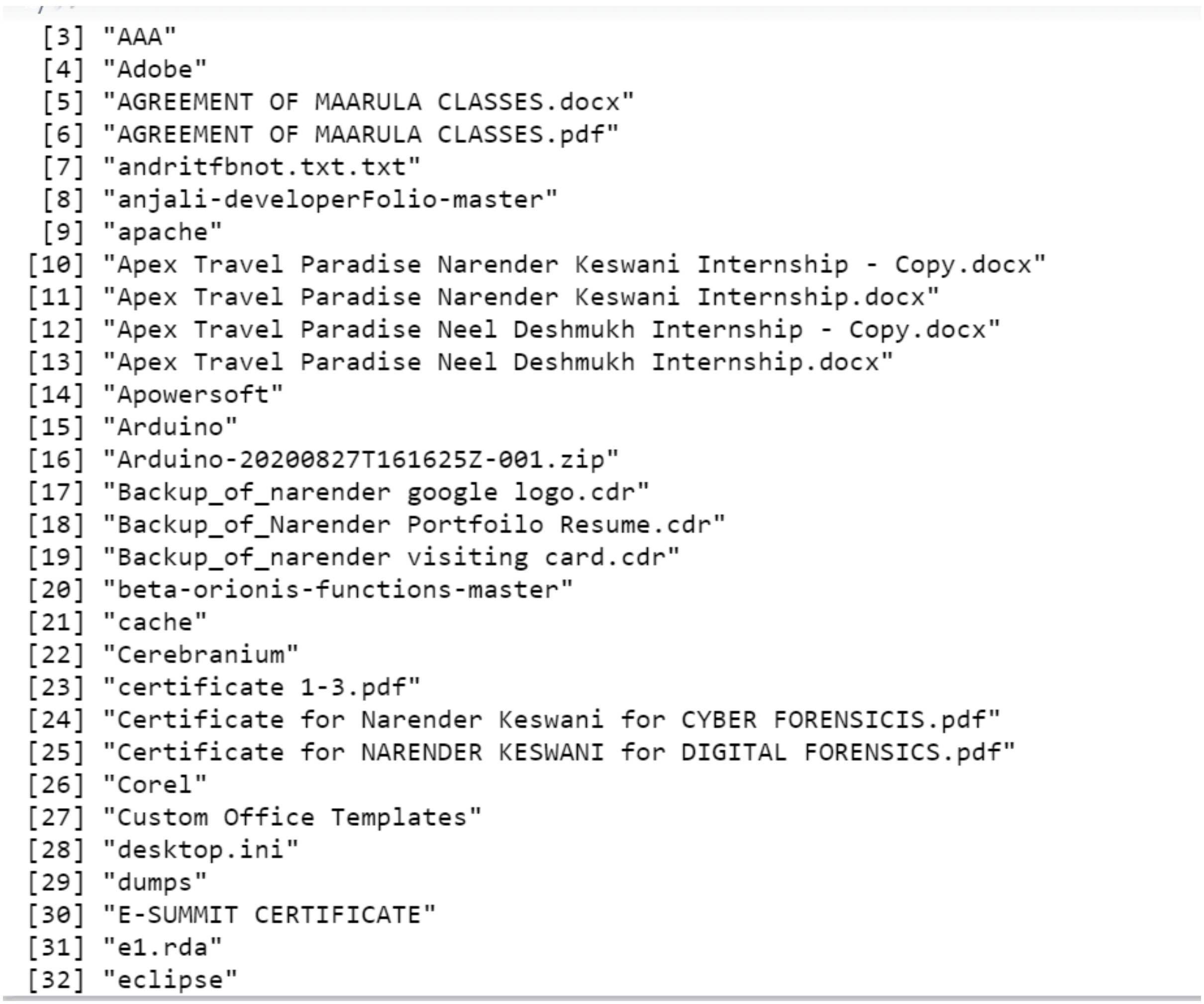
```
> ## getwd() - get current working directory.
> getwd()
[1] "C:/Users/NARENDER KESWANI/Documents"
```

C) GET LIST OF DIRECTORIES:

SOURCE CODE:

```
## dir() - lists the contents of current working directory.
dir()
```

OUTPUT:



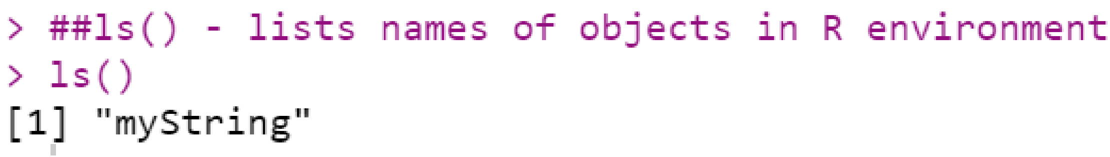
```
  [3] "AAA"
  [4] "Adobe"
  [5] "AGREEMENT OF MAARULA CLASSES.docx"
  [6] "AGREEMENT OF MAARULA CLASSES.pdf"
  [7] "andritfbnot.txt.txt"
  [8] "anjali-developerFolio-master"
  [9] "apache"
 [10] "Apex Travel Paradise Narender Keswani Internship - Copy.docx"
 [11] "Apex Travel Paradise Narender Keswani Internship.docx"
 [12] "Apex Travel Paradise Neel Deshmukh Internship - Copy.docx"
 [13] "Apex Travel Paradise Neel Deshmukh Internship.docx"
 [14] "Apowersoft"
 [15] "Arduino"
 [16] "Arduino-20200827T161625Z-001.zip"
 [17] "Backup_of_narender google logo.cdr"
 [18] "Backup_of_Narender Portfoilo Resume.cdr"
 [19] "Backup_of_narender visiting card.cdr"
 [20] "beta-orionis-functions-master"
 [21] "cache"
 [22] "Cerebranium"
 [23] "certificate 1-3.pdf"
 [24] "Certificate for Narender Keswani for CYBER FORENSICIS.pdf"
 [25] "Certificate for NARENDER KESWANI for DIGITAL FORENSICS.pdf"
 [26] "Corel"
 [27] "Custom Office Templates"
 [28] "desktop.ini"
 [29] "dumps"
 [30] "E-SUMMIT CERTIFICATE"
 [31] "e1.rda"
 [32] "eclipse"
```

D) **GET LIST NAMES OF OBJECTS IN R ENVIRONMENT:**

SOURCE CODE:

```
##ls() - lists names of objects in R environment
ls()
```

OUTPUT:



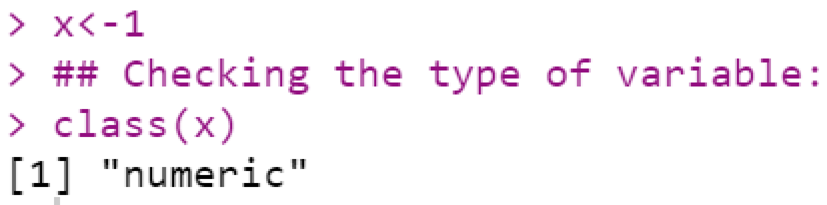
```
> ##ls() - lists names of objects in R environment
> ls()
[1] "myString"
```

E) CHECKING TYPE OF OBJECT:

SOURCE CODE:

```
x<-1
## Checking the type of variable:
class(x)
```

OUTPUT:



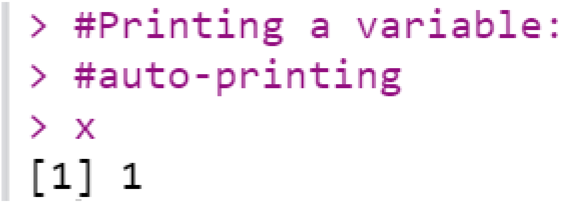
```
> x<-1
> ## Checking the type of variable:
> class(x)
[1] "numeric"
```

F) EXAMPLE OF AUTO-PRINTING:

SOURCE CODE:

```
#Printing a variable:
#auto-printing
X
```

OUTPUT:



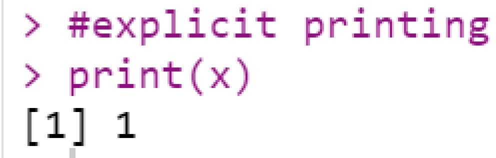
```
> #Printing a variable:
> #auto-printing
> x
[1] 1
```

G) EXAMPLE OF EXPLICIT PRINTING:

SOURCE CODE:

```
#explicit printing
print(x)
```

OUTPUT:



```
> #explicit printing
> print(x)
[1] 1
```

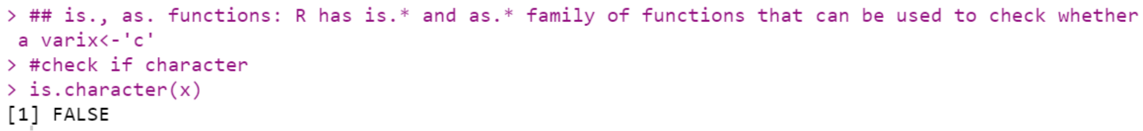
H) CHECK DATATYPE

1) CHARACTER:

SOURCE CODE:

is., as. functions: R has is.* and as.* family of functions that can be used to check whether a varix<-'c'
#check if character
is.character(x)

OUTPUT:



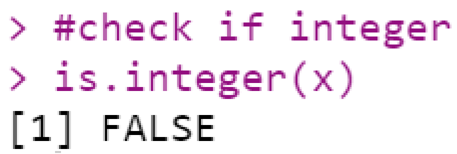
```
> ## is., as. functions: R has is.* and as.* family of functions that can be used to check whether
 a varix<-'c'
> #check if character
> is.character(x)
[1] FALSE
```

2) INTEGER:

SOURCE CODE:

#check if integer
is.integer(x)

OUTPUT:



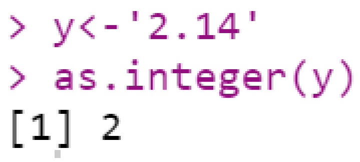
```
> #check if integer
> is.integer(x)
[1] FALSE
```

I) CONVERT TO INTEGER:

SOURCE CODE:

y<-'2.14'
as.integer(y)

OUTPUT:



```
> y<-'2.14'
> as.integer(y)
[1] 2
```

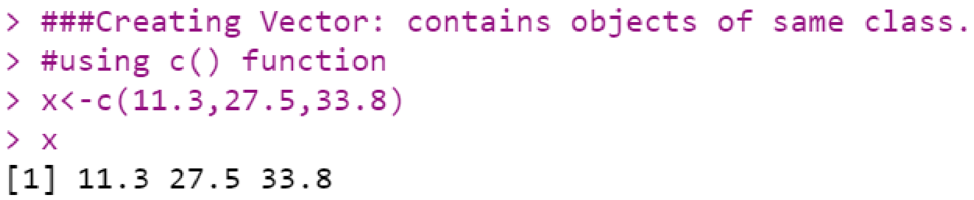
J) CREATE VECTOR:

1) USING c() FUNCTION:

SOURCE CODE:

```
###Creating Vector: contains objects of same class.
#using c() function
x<-c(11.3,27.5,33.8)
x
```

OUTPUT:



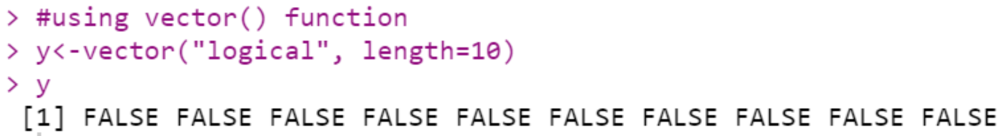
```
> ###Creating Vector: contains objects of same class.
> #using c() function
> x<-c(11.3,27.5,33.8)
> x
[1] 11.3 27.5 33.8
```

2) USING VECTOR() FUNCTION:

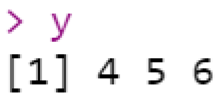
SOURCE CODE:

```
#using vector() function
y<-vector("logical", length=10)
y
y<-c(4,5,6)
y
```

OUTPUT:



```
> #using vector() function
> y<-vector("logical", length=10)
> y
 [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
```



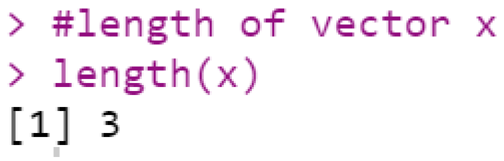
```
> y
[1] 4 5 6
```

K) FIND LENGTH OF VECTOR:

SOURCE CODE:

```
#length of vector x
length(x)
```

OUTPUT:



```
> #length of vector x
> length(x)
[1] 3
```

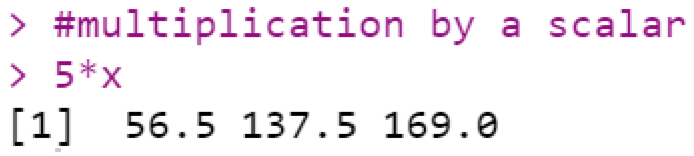
L) ARTHIMETIC OPERATIONS:

1) MULTIPLICATION OF SCALAR:

SOURCE CODE:

```
#multiplication by a scalar
5*x
```

OUTPUT:



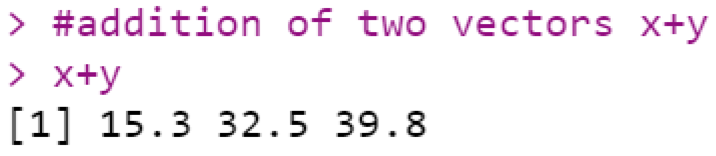
```
> #multiplication by a scalar
> 5*x
[1]  56.5 137.5 169.0
```

2) ADDITION OF VECTORS:

SOURCE CODE:

```
#addition of two vectors x+y
x+y
```

OUTPUT:



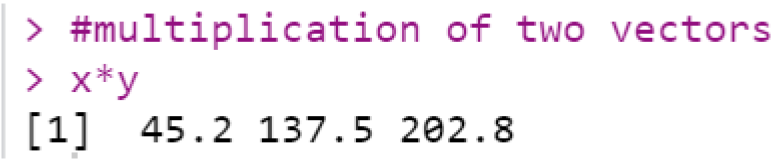
```
> #addition of two vectors x+y
> x+y
[1] 15.3 32.5 39.8
```

3) **MULTIPLICATION OF VECTORS:**

SOURCE CODE:

```
#multiplication of two vectors
x*y
```

OUTPUT:



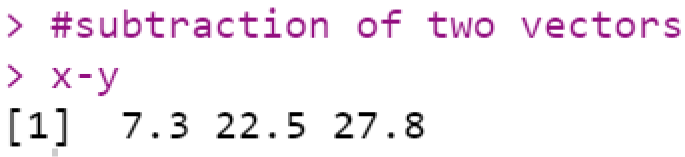
```
> #multiplication of two vectors
> x*y
[1]  45.2 137.5 202.8
```

4) **SUBTRACTION OF VECTORS:**

SOURCE CODE:

```
#subtraction of two vectors
x-y
```

OUTPUT:



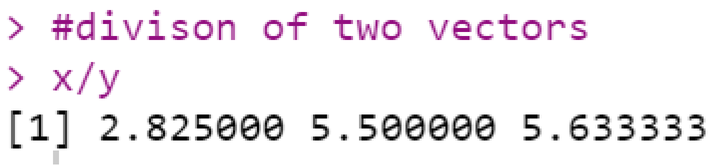
```
> #subtraction of two vectors
> x-y
[1]  7.3 22.5 27.8
```

5) **DIVISION OF VECTORS:**

SOURCE CODE:

```
#divison of two vectors
x/y
```

OUTPUT:



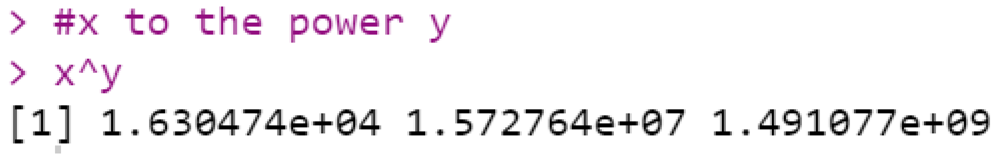
```
> #divison of two vectors
> x/y
[1] 2.825000 5.500000 5.633333
```

6) POWER:

SOURCE CODE:

```
#x to the power y
x^y
```

OUTPUT:



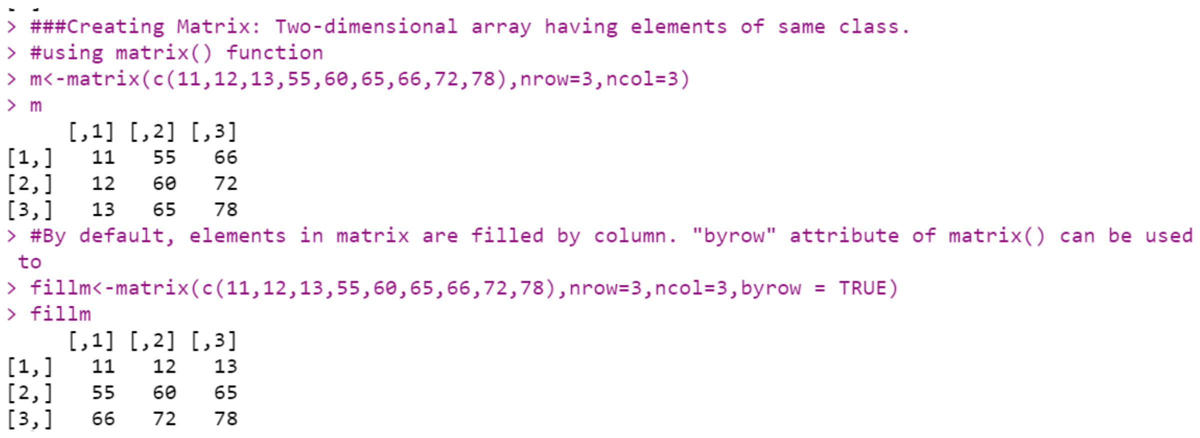
```
> #x to the power y
> x^y
[1] 1.630474e+04 1.572764e+07 1.491077e+09
```

M) CREATION OF MATRIX:

SOURCE CODE:

```
###Creating Matrix: Two-dimensional array having elements of same class.
#using matrix() function
m<-matrix(c(11,12,13,55,60,65,66,72,78),nrow=3,ncol=3)
m

#By default, elements in matrix are filled by column. "byrow" attribute of matrix() can be
used to
fillm<-matrix(c(11,12,13,55,60,65,66,72,78),nrow=3,ncol=3,byrow = TRUE)
fillm
```

OUTPUT:



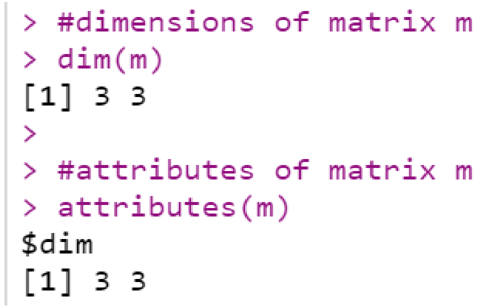
```
> ###Creating Matrix: Two-dimensional array having elements of same class.
> #using matrix() function
> m<-matrix(c(11,12,13,55,60,65,66,72,78),nrow=3,ncol=3)
> m
     [,1] [,2] [,3]
[1,]   11   55   66
[2,]   12   60   72
[3,]   13   65   78
> #By default, elements in matrix are filled by column. "byrow" attribute of matrix() can be used
 to
> fillm<-matrix(c(11,12,13,55,60,65,66,72,78),nrow=3,ncol=3,byrow = TRUE)
> fillm
     [,1] [,2] [,3]
[1,]   11   12   13
[2,]   55   60   65
[3,]   66   72   78
```

N) FIND DIMENSION & ATTRIBUTE OF A MATRIX:

SOURCE CODE:

```
#dimensions of matrix m
dim(m)

#attributes of matrix m
attributes(m)
```

OUTPUT:



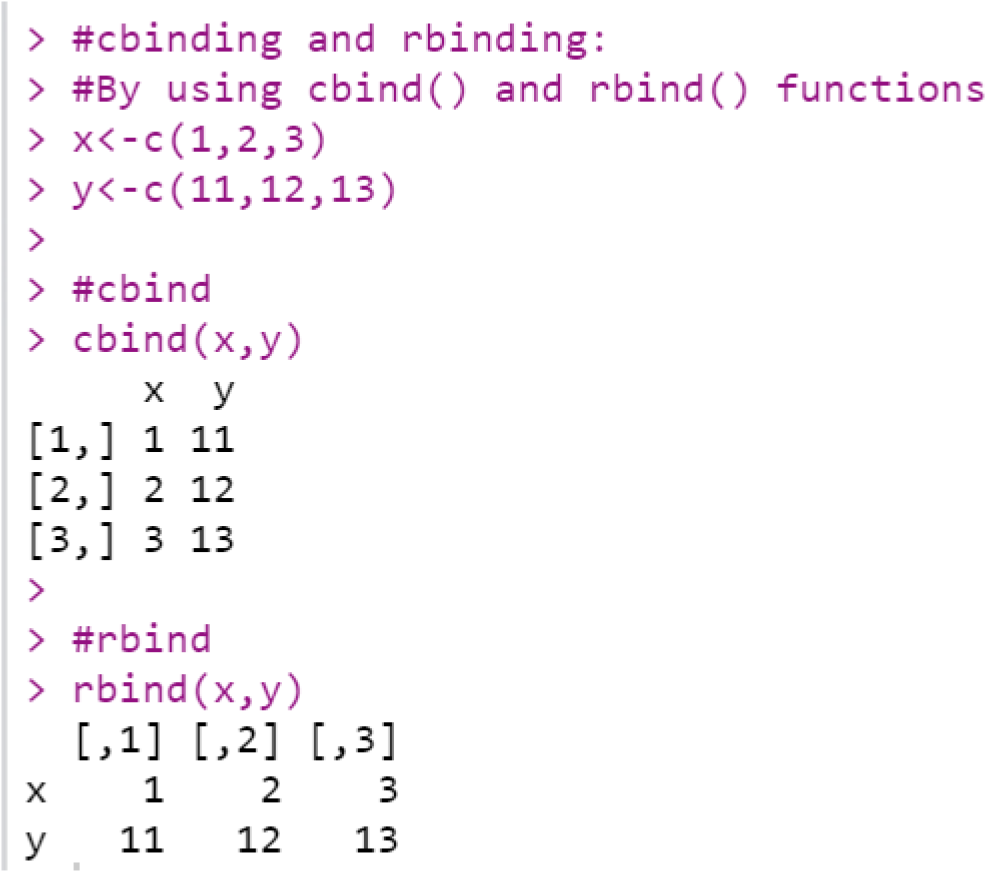
```
> #dimensions of matrix m
> dim(m)
[1] 3 3
>
> #attributes of matrix m
> attributes(m)
$dim
[1] 3 3
```

O) CBIND() & RBIND():

SOURCE CODE:

```
#cbinding and rbinding:
#By using cbind() and rbind() functions
x<-c(1,2,3)
y<-c(11,12,13)

#cbind
cbind(x,y)

#rbind
rbind(x,y)
```

OUTPUT:



```
> #cbinding and rbinding:
> #By using cbind() and rbind() functions
> x<-c(1,2,3)
> y<-c(11,12,13)
>
> #cbind
> cbind(x,y)
     x  y
[1,] 1 11
[2,] 2 12
[3,] 3 13
>
> #rbind
> rbind(x,y)
  [,1] [,2] [,3]
x    1    2    3
y   11   12   13
```

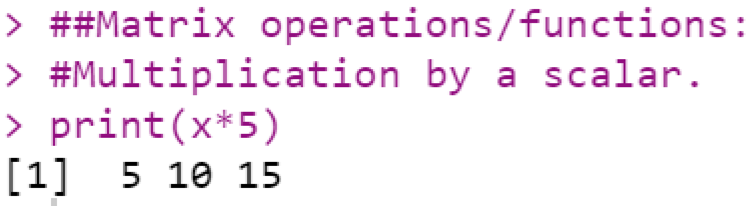
P) OPERATIONS ON MATRIX:

1) MULTIPLICATION BY A SCALAR:

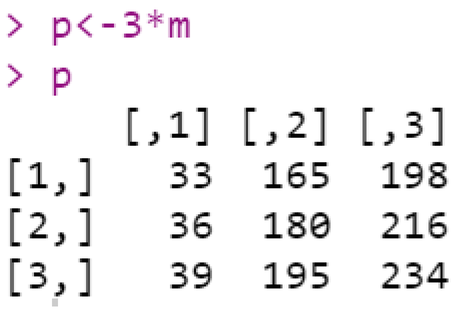
SOURCE CODE:

```
##Matrix operations/functions:
#Multiplication by a scalar.
print(x*5)

p<-3*m
p
```

OUTPUT:



```
> ##Matrix operations/functions:
> #Multiplication by a scalar.
> print(x*5)
[1]  5 10 15
```



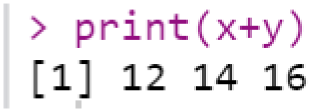
```
> p<-3*m
> p
     [,1] [,2] [,3]
[1,]   33  165  198
[2,]   36  180  216
[3,]   39  195  234
```

2) **ADDITION OF MATRICES:**

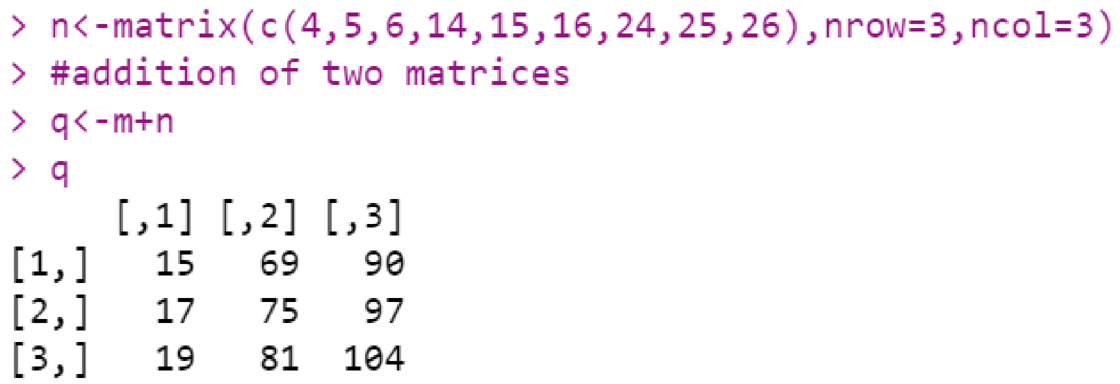
SOURCE CODE:

```
print(x+y)

n<-matrix(c(4,5,6,14,15,16,24,25,26),nrow=3,ncol=3)
q<-m+n
q
```

OUTPUT:



```
> print(x+y)
[1] 12 14 16
```



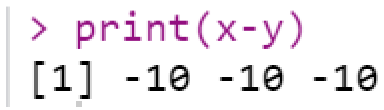
```
> n<-matrix(c(4,5,6,14,15,16,24,25,26),nrow=3,ncol=3)
> #addition of two matrices
> q<-m+n
> q
     [,1] [,2] [,3]
[1,]   15   69   90
[2,]   17   75   97
[3,]   19   81  104
```

3) **SUBTRACTION OF MATRICES:**

SOURCE CODE:

```
print(x-y)
```

OUTPUT:



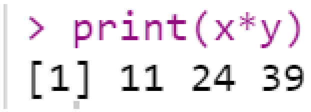
```
> print(x-y)
[1] -10 -10 -10
```

4) MULTIPLICATION OF MATRICES:

SOURCE CODE:

```
print(x*y)
```

OUTPUT:



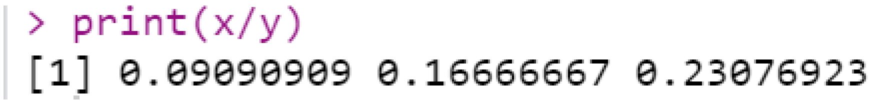
```
> print(x*y)
[1] 11 24 39
```

5) DIVISION OF MATRICES:

SOURCE CODE:

```
print(x/y)
```

OUTPUT:



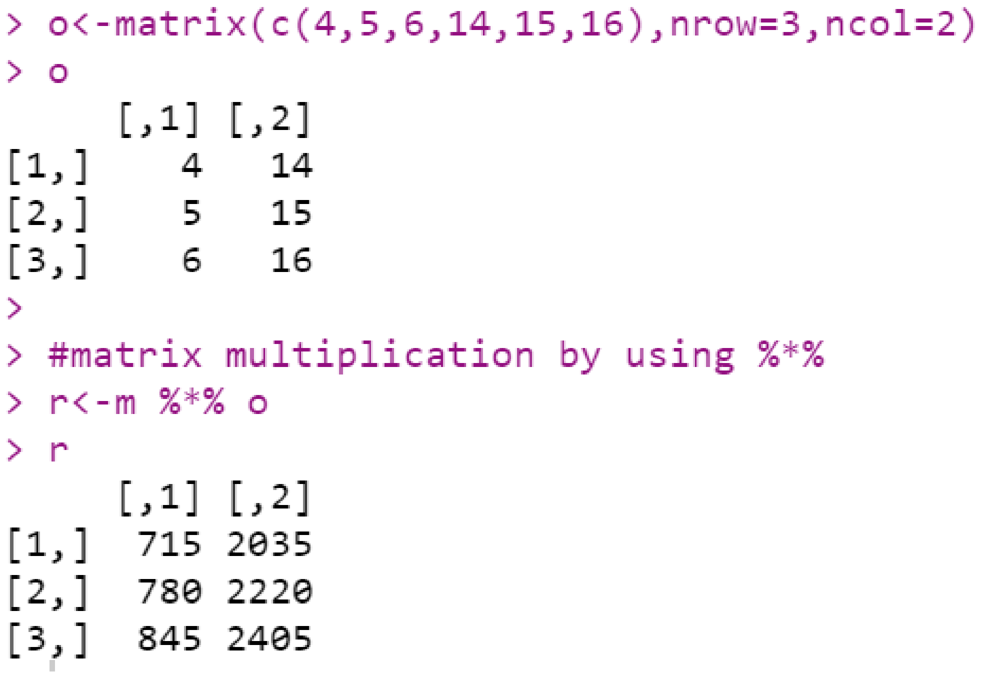
```
> print(x/y)
[1] 0.09090909 0.16666667 0.23076923
```

6) MATRIX MULTIPLICATION BY USING %*%

SOURCE CODE:

```
o<-matrix(c(4,5,6,14,15,16),nrow=3,ncol=2)
o

#matrix multiplication by using %*%
r<-m %*% o
r
```

OUTPUT:



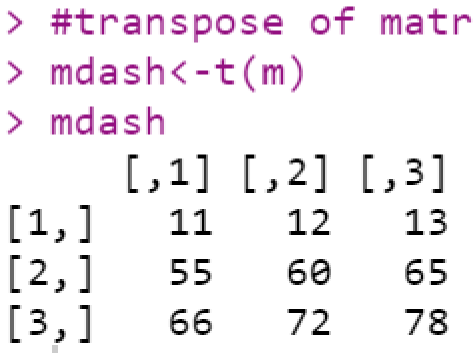
```
> o<-matrix(c(4,5,6,14,15,16),nrow=3,ncol=2)
> o
     [,1] [,2]
[1,]    4   14
[2,]    5   15
[3,]    6   16
>
> #matrix multiplication by using %*%
> r<-m %*% o
> r
     [,1] [,2]
[1,]  715 2035
[2,]  780 2220
[3,]  845 2405
```

7) **TRANSPOSE OF MATRIX:**

SOURCE CODE:

```
#transpose of matrix
mdash<-t(m)
mdash
```

OUTPUT:



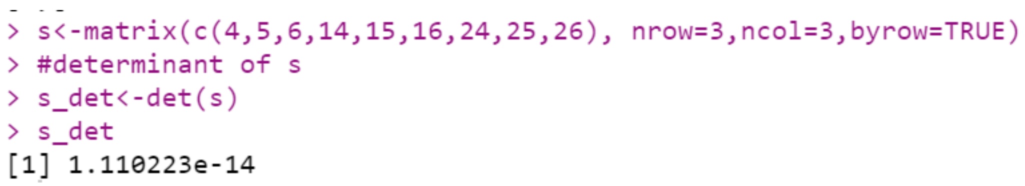
```
> #transpose of matrix
> mdash<-t(m)
> mdash
     [,1] [,2] [,3]
[1,]   11   12   13
[2,]   55   60   65
[3,]   66   72   78
```

8) **FIND DETERMINANT FROM MATRIX:**

SOURCE CODE:

```
s<-matrix(c(4,5,6,14,15,16,24,25,26), nrow=3,ncol=3,byrow=TRUE)
#determinant of s
s_det<-det(s)
s_det
```

OUTPUT:



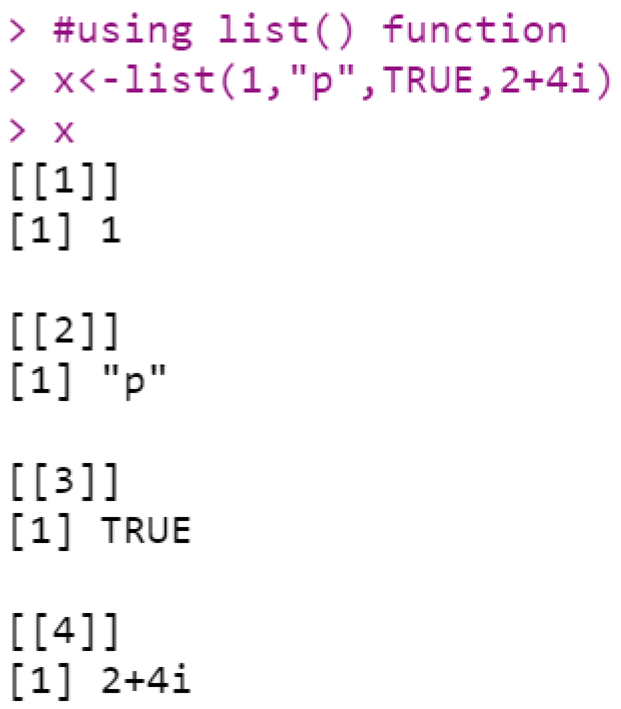
```
> s<-matrix(c(4,5,6,14,15,16,24,25,26), nrow=3,ncol=3,byrow=TRUE)
> #determinant of s
> s_det<-det(s)
> s_det
[1] 1.110223e-14
```

Q) EXAMPLE OF LIST:

SOURCE CODE:

```
#using list() function
x<-list(1,"p",TRUE,2+4i)
x
```

OUTPUT:



```
> #using list() function
> x<-list(1,"p",TRUE,2+4i)
> x
[[1]]
[1] 1

[[2]]
[1] "p"

[[3]]
[1] TRUE

[[4]]
[1] 2+4i
```

R) EXAMPLE OF FACTOR & LEVELS:

SOURCE CODE:

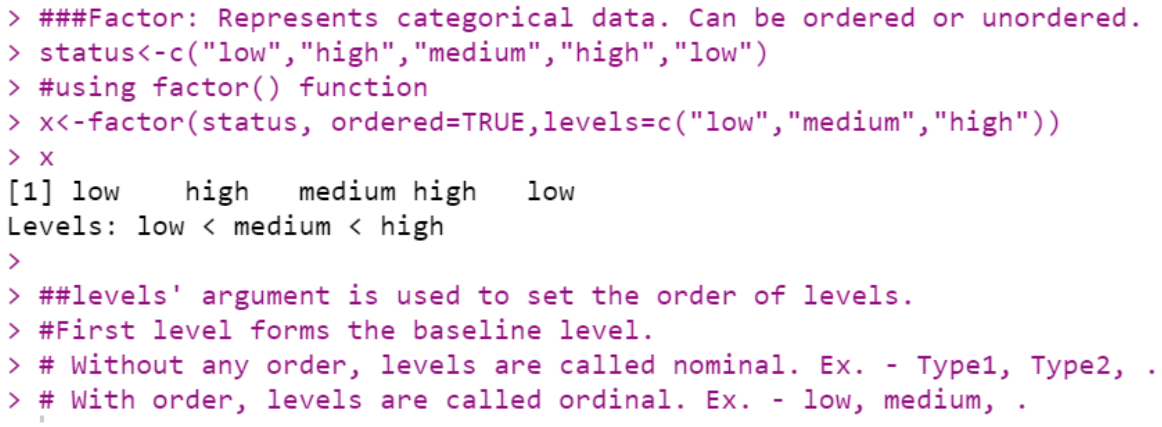
```
###Factor: Represents categorical data. Can be ordered or unordered.
status<-c("low","high","medium","high","low")
#using factor() function
x<-factor(status, ordered=TRUE,levels=c("low","medium","high"))
x

##levels' argument is used to set the order of levels.
```

#First level forms the baseline level.

Without any order, levels are called nominal. Ex. - Type1, Type2, .

With order, levels are called ordinal. Ex. - low, medium, .

OUTPUT:



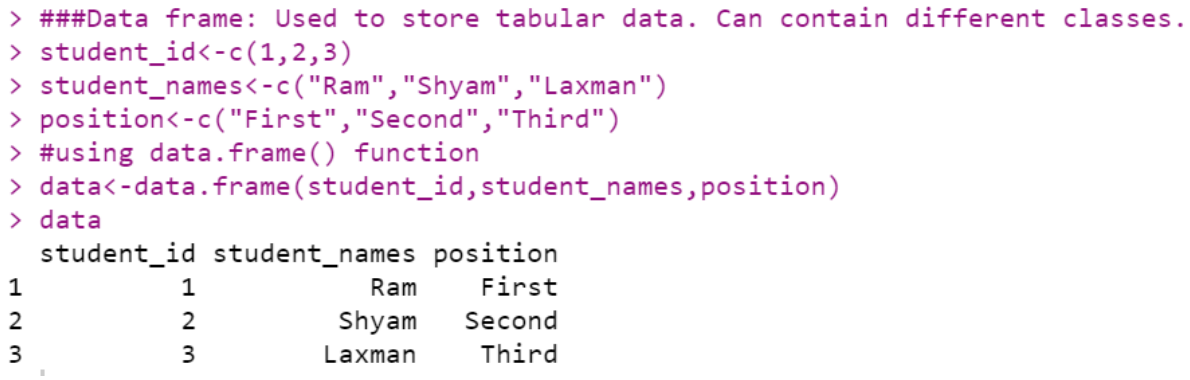
```
> ###Factor: Represents categorical data. Can be ordered or unordered.
> status<-c("low","high","medium","high","low")
> #using factor() function
> x<-factor(status, ordered=TRUE,levels=c("low","medium","high"))
> x
[1] low    high   medium high   low
Levels: low < medium < high
>
> ##levels' argument is used to set the order of levels.
> #First level forms the baseline level.
> # Without any order, levels are called nominal. Ex. - Type1, Type2, .
> # With order, levels are called ordinal. Ex. - low, medium, .
```

S) EXAMPLE OF DATAFRAME:

SOURCE CODE:

```
###Data frame: Used to store tabular data. Can contain different classes.
student_id<-c(1,2,3)
student_names<-c("Ram","Shyam","Laxman")
position<-c("First","Second","Third")
#using data.frame() function
data<-data.frame(student_id,student_names,position)
data
```

OUTPUT:



```
> ###Data frame: Used to store tabular data. Can contain different classes.
> student_id<-c(1,2,3)
> student_names<-c("Ram","Shyam","Laxman")
> position<-c("First","Second","Third")
> #using data.frame() function
> data<-data.frame(student_id,student_names,position)
> data
  student_id student_names position
1          1           Ram    First
2          2         Shyam   Second
3          3        Laxman    Third
```

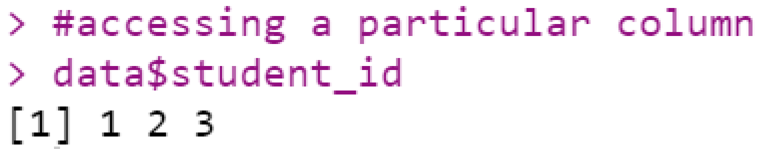
T) FUNCTIONS OF DATAFRAME:

1) ACCESSING A PARTICULAR COLUMN:

SOURCE CODE:

```
#accessing a particular column
data$student_id
```

OUTPUT:



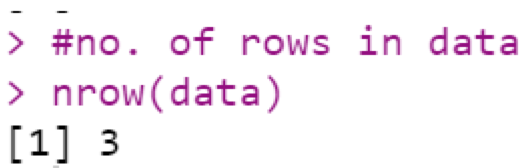
```
> #accessing a particular column
> data$student_id
[1] 1 2 3
```

2) **NUMBER OF ROWS IN DATAFRAME:**

SOURCE CODE:

```
#no. of rows in data
nrow(data)
```

OUTPUT:



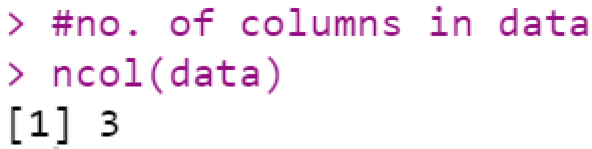
```
> #no. of rows in data
> nrow(data)
[1] 3
```

3) **NUMBER OF COLUMNS IN DATAFRAME:**

SOURCE CODE:

```
#no. of columns in data
ncol(data)
```

OUTPUT:



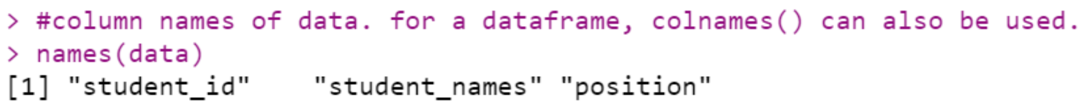
```
> #no. of columns in data
> ncol(data)
[1] 3
```

4) **GET COLUMN NAMES OF A DATAFRAME:**

SOURCE CODE:

```
#column names of data. for a dataframe, colnames() can also be used.
names(data)
```

OUTPUT:



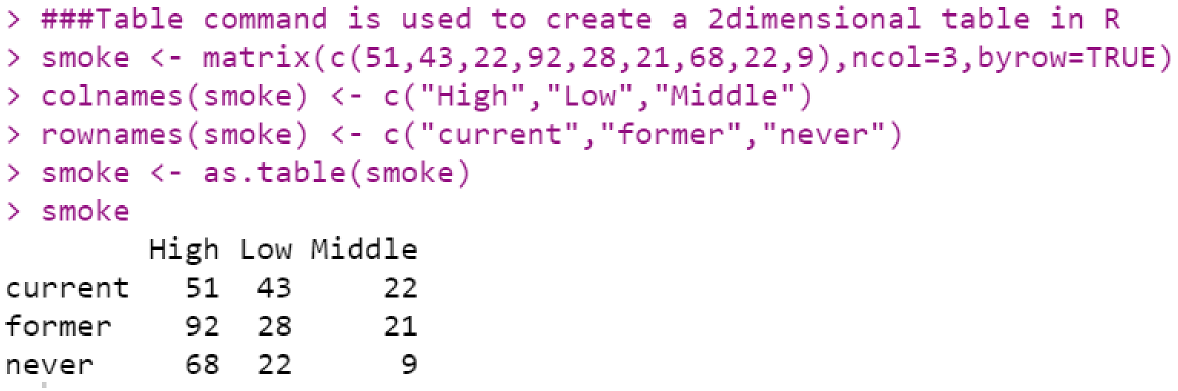
```
> #column names of data. for a dataframe, colnames() can also be used.
> names(data)
[1] "student_id"    "student_names" "position"
```

U) CREATE 2-DIMENSIONAL TABLE:

SOURCE CODE:

```
###Table command is used to create a 2dimensional table in R
smoke <- matrix(c(51,43,22,92,28,21,68,22,9),ncol=3,byrow=TRUE)
colnames(smoke) <- c("High","Low","Middle")
rownames(smoke) <- c("current","former","never")
smoke <- as.table(smoke)
smoke
```

OUTPUT:



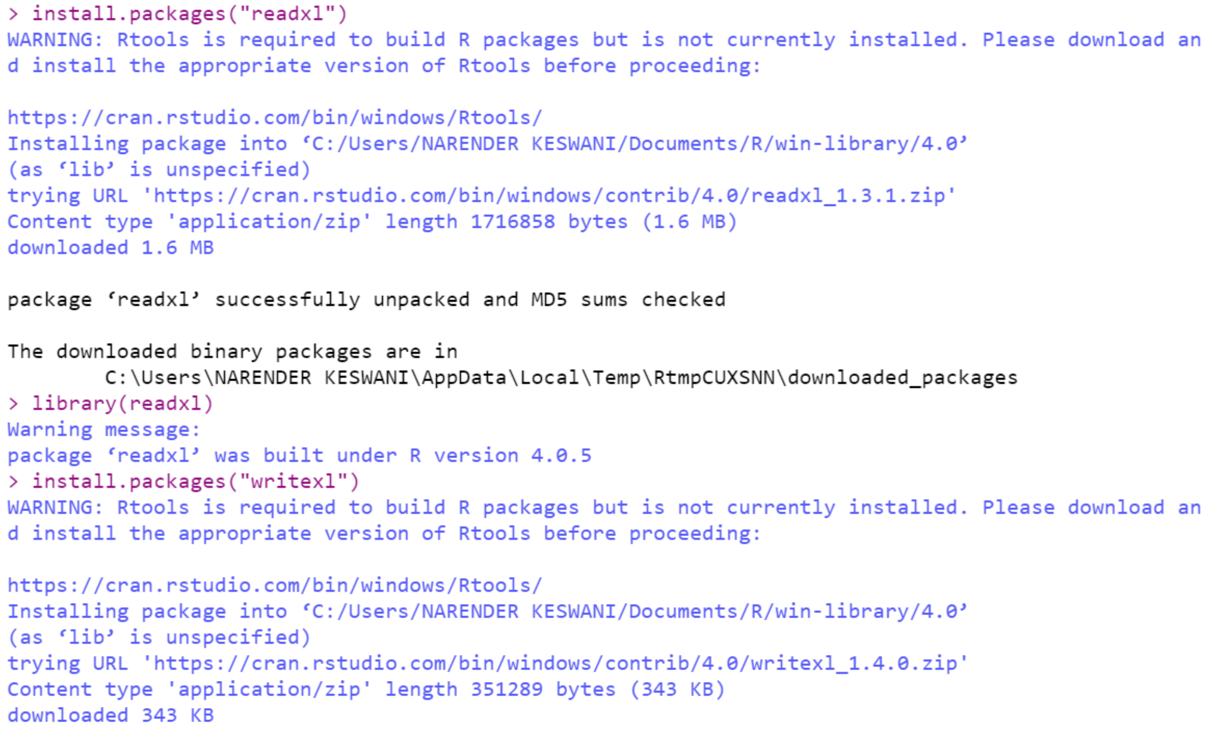
```
> ###Table command is used to create a 2dimensional table in R
> smoke <- matrix(c(51,43,22,92,28,21,68,22,9),ncol=3,byrow=TRUE)
> colnames(smoke) <- c("High","Low","Middle")
> rownames(smoke) <- c("current","former","never")
> smoke <- as.table(smoke)
> smoke
        High Low Middle
current   51  43     22
former    92  28     21
never     68  22      9
```

V) INSTALL LIBARIES:

SOURCE CODE:

```
#install.packages("package_name")
library(XLConnect)
install.packages("readxl")
library(readxl)
install.packages("writexl")
library(writexl)
```

OUTPUT:



```
> install.packages("readxl")
WARNING: Rtools is required to build R packages but is not currently installed. Please download an
d install the appropriate version of Rtools before proceeding:

https://cran.rstudio.com/bin/windows/Rtools/
Installing package into ‘C:/Users/NARENDER KESWANI/Documents/R/win-library/4.0’
(as ‘lib’ is unspecified)
trying URL 'https://cran.rstudio.com/bin/windows/contrib/4.0/readxl_1.3.1.zip'
Content type 'application/zip' length 1716858 bytes (1.6 MB)
downloaded 1.6 MB

package ‘readxl’ successfully unpacked and MD5 sums checked

The downloaded binary packages are in
        C:\Users\NARENDER KESWANI\AppData\Local\Temp\RtmpCUXSNN\downloaded_packages
> library(readxl)
Warning message:
package ‘readxl’ was built under R version 4.0.5
> install.packages("writexl")
WARNING: Rtools is required to build R packages but is not currently installed. Please download an
d install the appropriate version of Rtools before proceeding:

https://cran.rstudio.com/bin/windows/Rtools/
Installing package into ‘C:/Users/NARENDER KESWANI/Documents/R/win-library/4.0’
(as ‘lib’ is unspecified)
trying URL 'https://cran.rstudio.com/bin/windows/contrib/4.0/writexl_1.4.0.zip'
Content type 'application/zip' length 351289 bytes (343 KB)
downloaded 343 KB
```

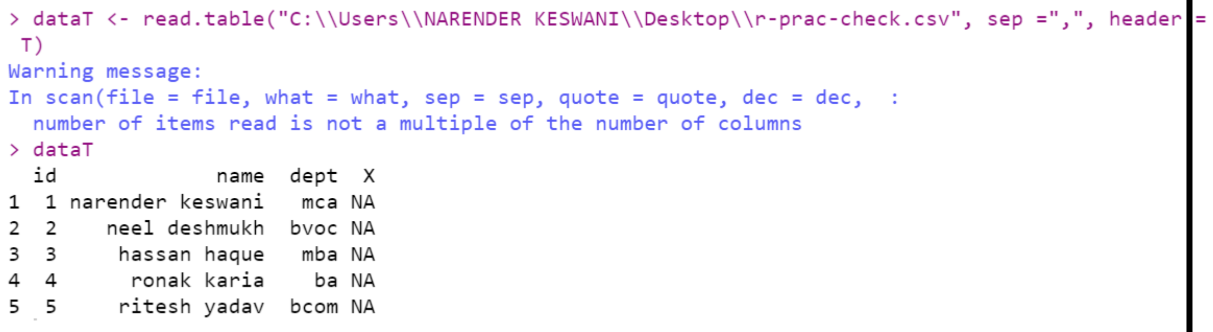
W) EXAMPLES OF CSV:

1) READING DATA FROM CSV:

SOURCE CODE:

```
dataT <- read.table("C:\\Users\\NARENDER KESWANI\\Desktop\\r-prac-check.csv", sep
=",", header = T)
dataT
```

OUTPUT:



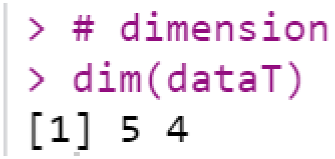
```
> dataT <- read.table("C:\\Users\\NARENDER KESWANI\\Desktop\\r-prac-check.csv", sep =",", header =
 T)
Warning message:
In scan(file = file, what = what, sep = sep, quote = quote, dec = dec,  :
  number of items read is not a multiple of the number of columns
> dataT
  id             name  dept  X
1  1 narender keswani   mca NA
2  2     neel deshmukh  bvoc NA
3  3      hassan haque   mba NA
4  4       ronak karia    ba NA
5  5      ritesh yadav  bcom NA
```

2) GET DIMENSIONS OF CSV FILE:

SOURCE CODE:

```
# dimension
dim(dataT)
```

OUTPUT:



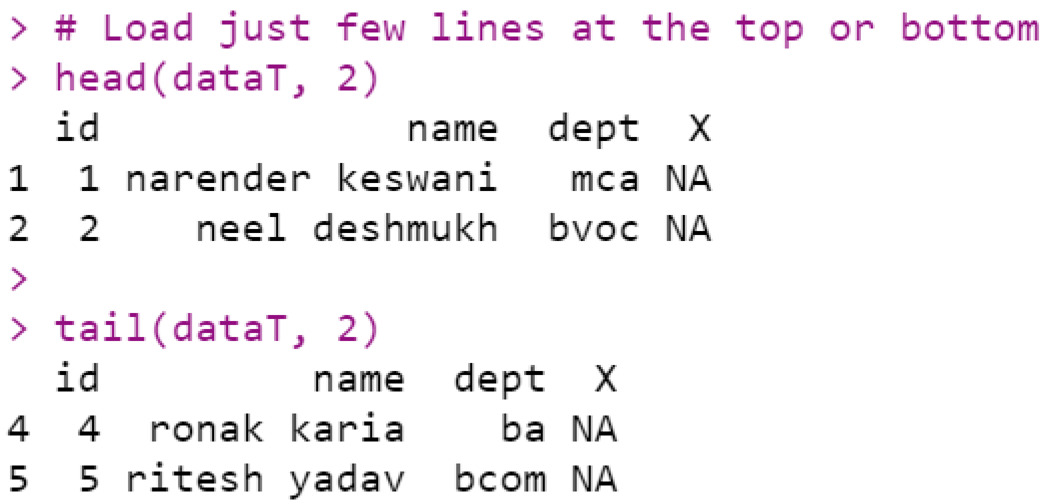
```
> # dimension
> dim(dataT)
[1] 5 4
```

3) **HEAD & TAIL:**

SOURCE CODE:

```
# Load just few lines at the top or bottom
head(dataT, 2)

tail(dataT, 2)
```

OUTPUT:



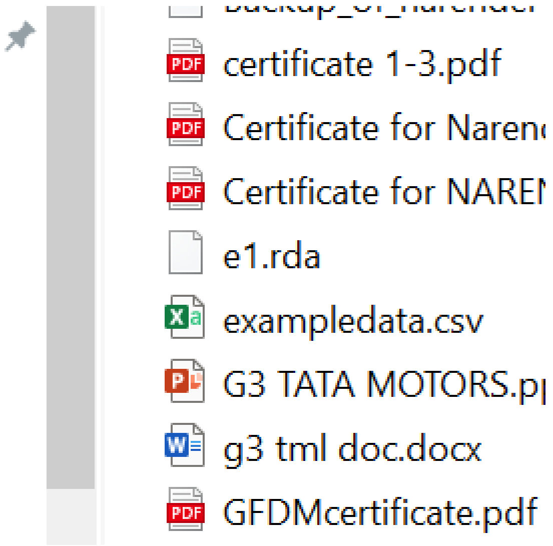
```
> # Load just few lines at the top or bottom
> head(dataT, 2)
  id             name  dept  X
1  1 narender keswani   mca NA
2  2    neel deshmukh  bvoc NA
> 
> tail(dataT, 2)
  id         name  dept  X
4  4  ronak karia    ba NA
5  5 ritesh yadav  bcom NA
```

4) **WRITING DATA TO CSV FILE:**

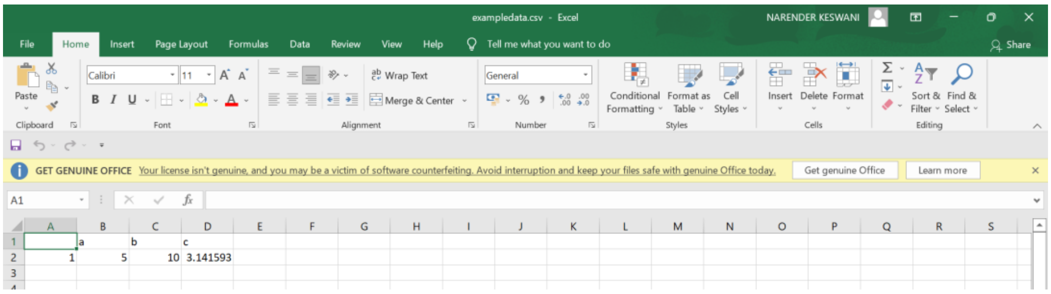
SOURCE CODE:

```
z <- data.frame(a = 5, b = 10, c = pi)
write.csv(z,file="exampledata.csv")
```

OUTPUT:



certificate 1-3.pdf
Certificate for Naren
Certificate for NAREI
e1.rda
exampledata.csv
G3 TATA MOTORS.p
g3 tml doc.docx
GFDMcertificate.pdf



exampledata.csv - Excel

	A	B	C	D
1		a	b	c
2	1	5	10	3.141593

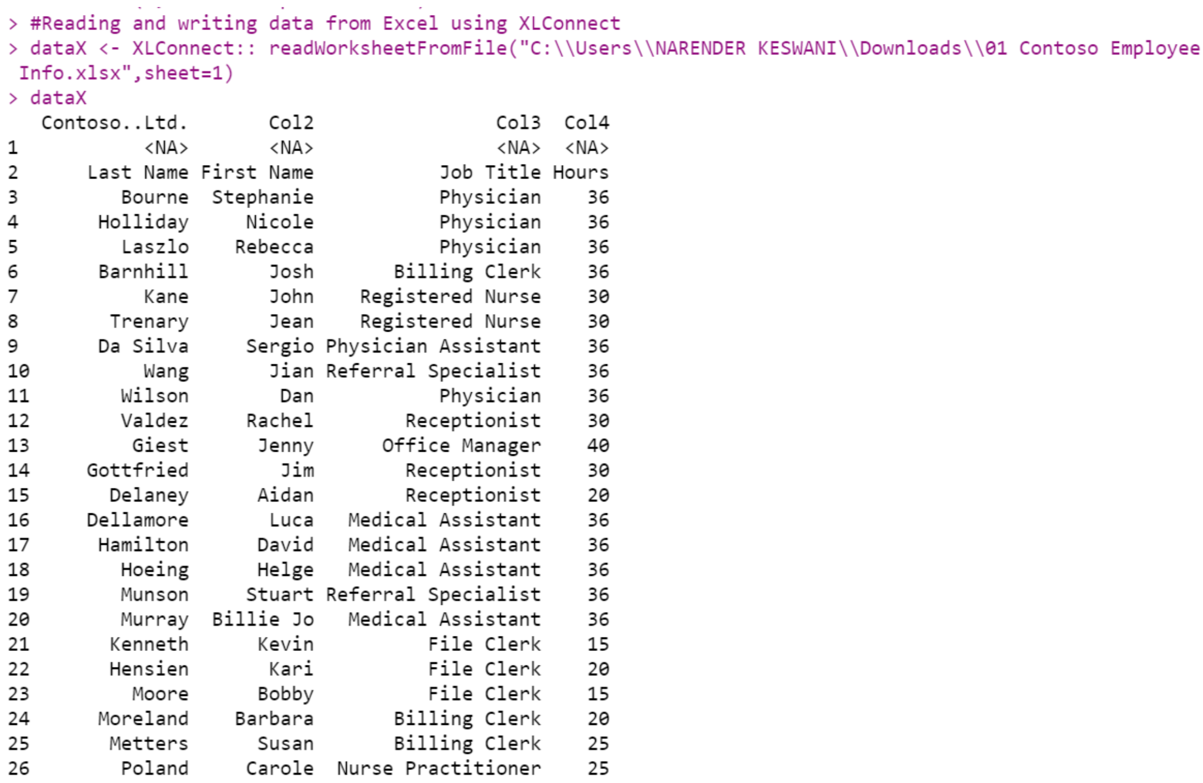
X) READING AND WRITING DATA FROM EXCEL USING XLCONNECT:

SOURCE CODE:

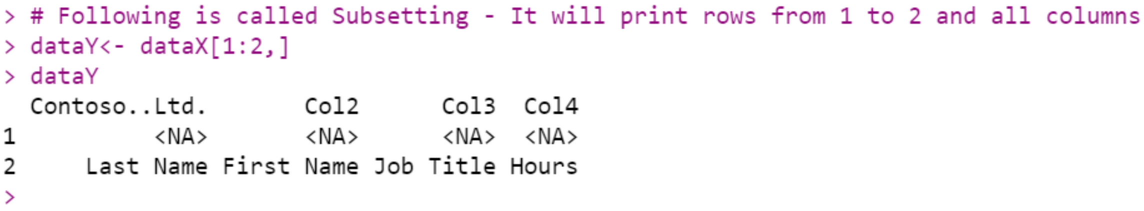
```
install.packages('Rcpp')
library(Rcpp)

#Reading and writing data from Excel using XLConnect
dataX <- XLConnect:: readWorksheetFromFile("C:\\Users\\NARENDER
KESWANI\\Downloads\\01 Contoso Employee Info.xlsx",sheet=1)
dataX

# Following is called Subsetting - It will print rows from 1 to 2 and all columns
dataY<- dataX[1:2,]
dataY

#Reading and writing data from Excel using readXL and writeXL
data2 <- read_excel("C:\\Users\\NARENDER KESWANI\\Downloads\\01 Contoso Employee
Info.xlsx", sheet = "1")
data2
data3<- data2[1:5,]
write_xlsx(data3, "e2.xlsx")

# create an empty data frame
```

```
data <- data.frame(Name=character(), Age=numeric())
# load interface and assign edited values to data back - uncomment following
data <- edit(data)
#print those values
data
```

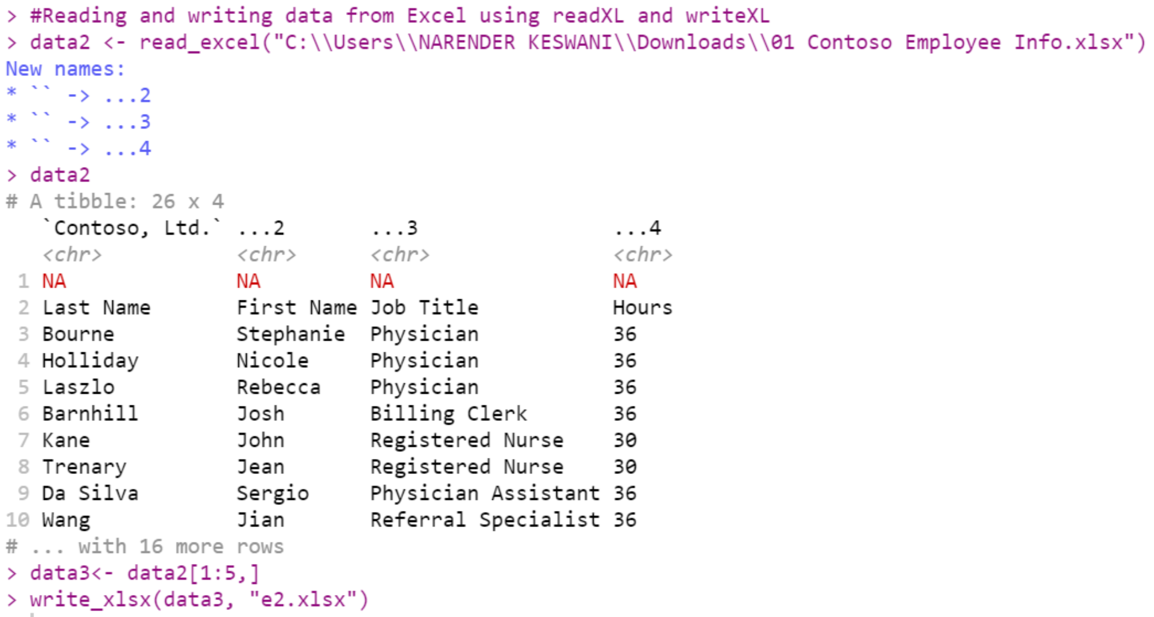
OUTPUT:



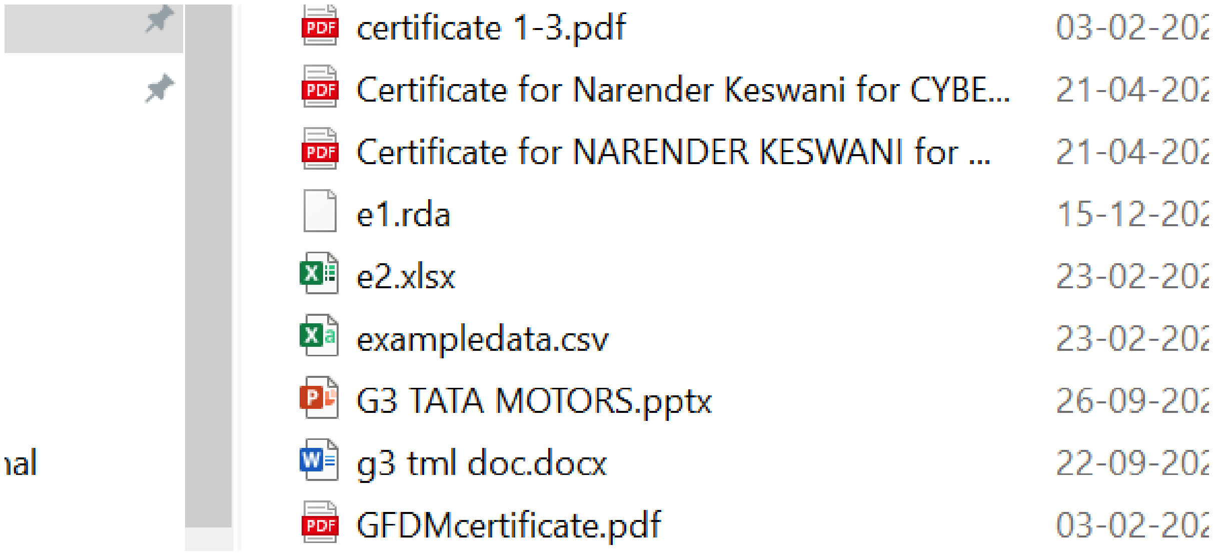
```
> #Reading and writing data from Excel using XLConnect
> dataX <- XLConnect:: readWorksheetFromFile("C:\\Users\\NARENDER KESWANI\\Downloads\\01 Contoso Employee
Info.xlsx",sheet=1)
> dataX
   Contoso..Ltd.       Col2                Col3  Col4
1           <NA>       <NA>                <NA>  <NA>
2      Last Name First Name           Job Title Hours
3         Bourne  Stephanie           Physician    36
4       Holliday     Nicole           Physician    36
5         Laszlo    Rebecca           Physician    36
6       Barnhill       Josh       Billing Clerk    36
7           Kane       John    Registered Nurse    30
8        Trenary       Jean    Registered Nurse    30
9       Da Silva     Sergio Physician Assistant    36
10          Wang       Jian Referral Specialist    36
11        Wilson        Dan           Physician    36
12        Valdez     Rachel        Receptionist    30
13         Giest      Jenny      Office Manager    40
14     Gottfried        Jim        Receptionist    30
15       Delaney      Aidan        Receptionist    20
16     Dellamore       Luca   Medical Assistant    36
17      Hamilton      David   Medical Assistant    36
18        Hoeing      Helge   Medical Assistant    36
19        Munson     Stuart Referral Specialist    36
20        Murray  Billie Jo   Medical Assistant    36
21       Kenneth      Kevin          File Clerk    15
22       Hensien       Kari          File Clerk    20
23         Moore      Bobby          File Clerk    15
24      Moreland    Barbara       Billing Clerk    20
25       Metters      Susan       Billing Clerk    25
26        Poland     Carole  Nurse Practitioner    25
```



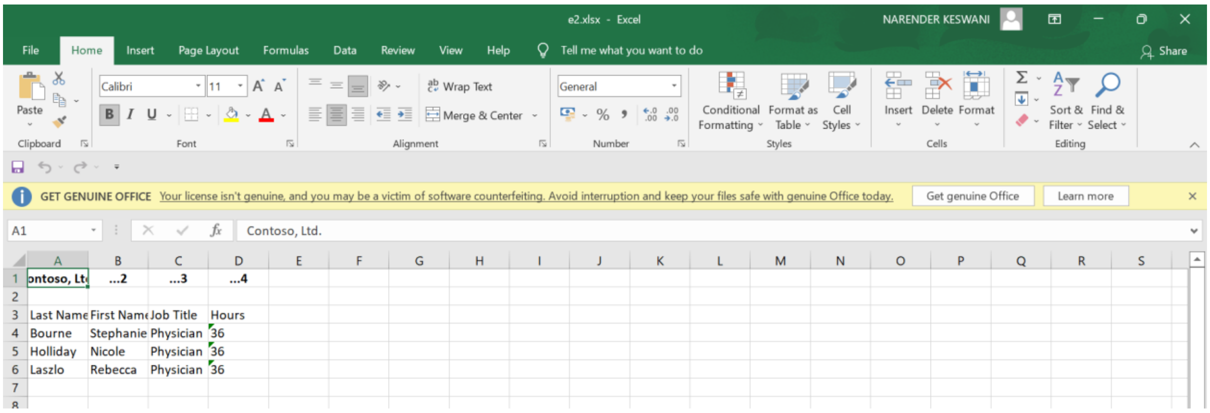
```
> # Following is called Subsetting - It will print rows from 1 to 2 and all columns
> dataY<- dataX[1:2,]
> dataY
  Contoso..Ltd.       Col2      Col3  Col4
1          <NA>       <NA>      <NA>  <NA>
2     Last Name First Name Job Title Hours
>
```



```
> #Reading and writing data from Excel using readXL and writeXL
> data2 <- read_excel("C:\\Users\\NARENDER KESWANI\\Downloads\\01 Contoso Employee Info.xlsx")
New names:
* `` -> ...2
* `` -> ...3
* `` -> ...4
> data2
# A tibble: 26 x 4
   `Contoso, Ltd.` ...2       ...3                ...4 
   <chr>           <chr>      <chr>               <chr>
 1 NA              NA         NA                  NA   
 2 Last Name       First Name Job Title           Hours
 3 Bourne          Stephanie  Physician           36   
 4 Holliday        Nicole     Physician           36   
 5 Laszlo          Rebecca    Physician           36   
 6 Barnhill        Josh       Billing Clerk       36   
 7 Kane            John       Registered Nurse    30   
 8 Trenary         Jean       Registered Nurse    30   
 9 Da Silva        Sergio     Physician Assistant 36   
10 Wang            Jian       Referral Specialist 36   
# ... with 16 more rows
> data3<- data2[1:5,]
> write_xlsx(data3, "e2.xlsx")
```

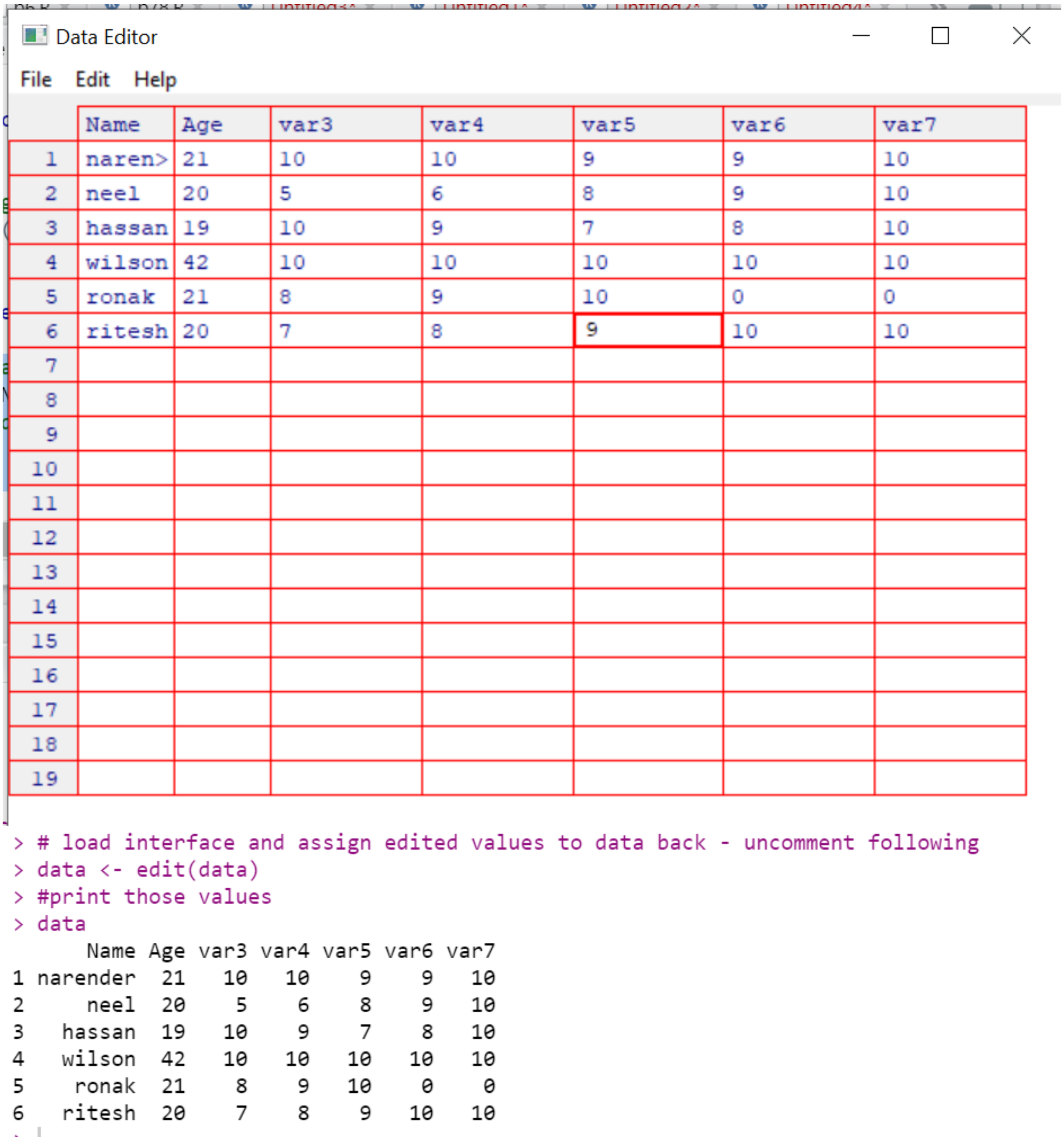


certificate 1-3.pdf	03-02-202
Certificate for Narender Keswani for CYBE...	21-04-202
Certificate for NARENDER KESWANI for ...	21-04-202
e1.rda	15-12-202
e2.xlsx	23-02-202
exampledata.csv	23-02-202
G3 TATA MOTORS.pptx	26-09-202
g3 tml doc.docx	22-09-202
GFDMcertificate.pdf	03-02-202



Contoso, Ltd.

	A	B	C	D
1	ontoso, Ltd	...2	...3	...4
2				
3	Last Name	First Name	Job Title	Hours
4	Bourne	Stephanie	Physician	36
5	Holliday	Nicole	Physician	36
6	Laszlo	Rebecca	Physician	36
7				



	Name	Age	var3	var4	var5	var6	var7
1	naren>	21	10	10	9	9	10
2	neel	20	5	6	8	9	10
3	hassan	19	10	9	7	8	10
4	wilson	42	10	10	10	10	10
5	ronak	21	8	9	10	0	0
6	ritesh	20	7	8	9	10	10
7							
8							
9							
10							
11							
12							
13							
14							
15							
16							
17							
18							
19							

```
> # load interface and assign edited values to data back - uncomment following
> data <- edit(data)
> #print those values
> data
      Name Age var3 var4 var5 var6 var7
1 narender  21   10   10    9    9   10
2     neel  20    5    6    8    9   10
3   hassan  19   10    9    7    8   10
4   wilson  42   10   10   10   10   10
5    ronak  21    8    9   10    0    0
6   ritesh  20    7    8    9   10   10
```

CONCLUSION:

From this practical, I have learned the basics of R programming.