External Practical Examination Programs of CA 304© Data Analytics

1. **Write a program for creating R objects Data frame and Factor.**

```
#factors:-
#1)create vector:-
flowers<-c("Rose","Lily","Lotus","Mogra")
#2)create a factor object:-
factor_names<-factor(flowers)
#3)print the factor:-
print(factor_names)
print(nlevels(factor_names))
#Data frame:-
stu_data<-data.frame(
 Roll_no=c(1:5),
 stu_name=c("Roshani","Dipti","Komal","Leena","Suvarna"),
Course=c("MCA","MBA","BBA","BCA","IMCA"),
Birth_date=as.Date(c("15-01-2001","09-09-2000","10-04-2000","25-07-2000","10-02-   2000"))
)
print(stu_data)
```

2. **Write a program to implement basic mathematical operations in R Programming.**

R Script:-

```
#Assignment
s <- 10
print(s)

#case sensitive
a <- 5
b <- 2
c <- 1
print(a+B+c)

#Arithmetic Operations
#creation of vector

a <- c(10, 20, 30, 40, 50)
b <- c(1, 2, 3, 4, 5)
print(a)
print(b)

#Addition of tow vector
Result <- a+b
print(Result)
#Subtraction of tow vector
Result <- a-b
print(Result)

#Multiplication of tow vector
Result <- a*b
print(Result)

#Division of tow vector
Result <- a/b
print(Result)

#options()
1/7#by default it will show 7 digits output.

options(digits = 3)#by using this it will show only 3 digits after decimal point
1/7

#Miscellaneous Mathematical functions
x<-20
abs(x)  #Absolute Value
sqrt(x) #square root
exp(x)  #exponential transformation
```

```
log(x)  #logarithmic transformation
cos(x)  #cosine and other trigonometric transformation
#infinite and Nan Number
y<-5
z<-6
ls()  #List all object
exists("y")   #identify R object with 'y' name
rm(y)        #remove object.
rm(y,z)      #remove multiple object.
rm(list=ls()) #remove everything on working environment.
```

OUTPUT-

```
> #Assignment
> s <- 10
> print(s)
[1] 10
>
> #case sensitive
> a <- 5
> b <- 2
> c <- 1
> print(a+B+c)
Error in print(a + B + c) : object 'B' not found
>
> #Arithmetic Operations
> #creation of vector
>
> a <- c(10, 20, 30, 40, 50)
> b <- c(1, 2, 3, 4, 5)
> print(a)
[1] 10 20 30 40 50
> print(b)
[1] 1 2 3 4 5
>
> #Addition of tow vector
> Result <- a+b
> print(Result)
[1] 11 22 33 44 55
>
> #Subtraction of tow vector
> Result <- a-b
> print(Result)
[1]  9 18 27 36 45
>
> #Multiplication of tow vector
> Result <- a*b
> print(Result)
[1]  10  40  90 160 250
>
> #Division of tow vector
> Result <- a/b
> print(Result)
[1] 10 10 10 10 10
>
> #options()
> 1/7#by default it will show 7 digits output.
[1] 0.1428571
>
> options(digits = 3)#by using this it will show only 3 digits after decimal point
> 1/7
[1] 0.143
>
> #Miscellaneous Mathematical functions
>
> x<-20
> abs(x)  #Absolute Value
[1] 20
> sqrt(x) #square root
[1] 4.47
> exp(x)  #exponential transformation
[1] 4.85e+08
> log(x)  #logarithmic transformation
[1] 3
> cos(x)  #cosine and other trigonometric transformation
```

```
[1] 0.408
>
> #infinite and Nan Number
> y<-5
> z<-6
>
> ls()  #List all object
[1] "a"      "b"      "c"      "Result" "s"      "x"      "y"      "z"
> exists("y")   #identify R object with 'y' name
[1] TRUE
> rm(y)         #remove object.
> rm(y,z)       #remove multiple object.
Warning message:
In rm(y, z) : object 'y' not found
> rm(list=ls()) #remove everything on working environment.
```

3. Write a program for creation of atomic vectors in R and access the elements on the basis of indexing.

```
#In R, there are four types of Atomic vectors:
#1)Numeric Vector:-
  d<-44.5
   d
  class(d)
   num_vector<-c(12.5,22,34.0)
   num_vector
  class(num_vector)

#2)Integer Vector:-
   a<-as.integer(10)
   a
   b<-20L
   b
   class(b)
   num<-c(2L,6L,4L,9L)
   num
   class(num)
   #Access elements basis of indexing:
   seq_vector<-seq(1,7,length.out=5)
   seq_vector
   seq_vector[2]

#3)Character Vector:-
   a<-"Roshani"
   a
   char_vec1<-c("shubham","arpita","nishka","vaishali")
   char_vec1
   x<-20
   x<-as.character(x)
   x
   class(x)
   #Access elements basis of indexing:
   name_vec<-c("Roshani"=48,"Leena"=82,"Dipti"=70,"Komal"=03)
   name_vec
   name_vec["Komal"]

#4)Logical Vector:-
   a<-as.integer(20)
   b<-as.integer(10)
   d<-as.integer(5)
   result<-b>a
   result
   x<-a>b
   x
   d
   class(x)
```

```
#Access elements basis of indexing:
z<-c(1,2,3,4,5,6)
z[c(TRUE,FALSE,TRUE,TRUE,FALSE,TRUE)]
```

4. Write a program for creation of vectors and perform operations on vectors in R.

```
names<-c("roshani","dipti","komal","leena","suvarna")
names
num<-c(1,2,3,4,5)
num
#operations on vectors:
#1)combining vectors:
data_vec<-c(names,num)
data_vec

#2)Arithmetic operations:
    a<-c(1,3,5,7)
    b<-c(2,4,6,8)
            a+b
     a-b
    a*b
  a/b

#3)Logical Index vector:
  z<-c(1,2,3,4,5,6)
  z[c(TRUE,FALSE,TRUE,TRUE,FALSE,TRUE)]

#4)Numeric Index:-
    q<-c("shubham","arpita","nishka","gunjan","vaishali","sumit")
    q[2]
    q[-4]
    q[15]

#5)Duplicate Index:-
    q<-c("shubham","arpita","nishka","gunjan","vaishali","sumit")
    q[c(2,4,4,3)]

#6)Range Indexes:-
    q<-c("shubham","arpita","nishka","gunjan","vaishali","sumit")
    b<-q[2:5]
    b

#7)out-of-order Indexes:-
    q<-c("shubham","arpita","nishka","gunjan","vaishali","sumit")
    q[c(2,1,3,4,5,6)]

#8)Named vectors members:-
    z=c("Roshani","Kawale")
    z
    names(z)=c("FirstName","LastName")
    z
    z["FirstName"]
```

5. Write a program in R to convert the vector into list and print the elements.

```
list1 <- list(10:20)
    print(list1)

    list2<-list("Neesha","Riya")
    print(list2)
    #convert list into vector:-
    v1 <- unlist(list1)
    print(v1)

    v2<-unlist(list2)
```

```
print(v2)
```

6. **Write a program for creation of lists and perform operation on list in R programming.**

#creation of List:-

```
list_1<-list("Shubham","Arpita","Vaishali")
list_1
list_data<-list("Shubham","Arpita",c(1,2,3,4,5),TRUE,FALSE,22.5,12L)
print(list_data)
```

#Operation on lists:-

1)Giving name to list:-

```
list_data <- list(c("Shubham","Nishka","Gunjan"), matrix(c(40,80,60,70,90,80), nrow = 2),
   list("BCA","MCA","B.tech"))
names(list_data) <- c("Students", "Marks", "Course")
list_data
```

2)Accessing elements using index:-

```
print(list_data[1])
```

3)Accessing elements using names:-

```
print(list_data["Students"])
print(list_data$Marks)
```

4)Merging Lists:-

```
Even_list <- list(2,4,6)
Odd_list <- list(1,3,5)
# Merging the two lists.
merged.list <- list(Even_list,Odd_list)
print(merged.list)
```

OUTPUT: -

```
#creation of List:-
> list_1<-list("Shubham","Arpita","Vaishali")
> list_1
[[1]]
[1] "Shubham"
[[2]]
[1] "Arpita"
[[3]]
[1] "Vaishali"
> list_data<-list("Shubham","Arpita",c(1,2,3,4,5),TRUE,FALSE,22.5,12L)
> print(list_data)
[[1]]
[1] "Shubham"
[[2]]
[1] "Arpita"
[[3]]
[1] 1 2 3 4 5
[[4]]
[1] TRUE
[[5]]
[1] FALSE
```

```
[[6]]
[1] 22.5
[[7]]
[1] 12
> #Operation on lists:-
> #1)Giving name to list:-
> list_data <- list(c("Shubham","Nishka","Gunjan"), matrix(c(40,80,60,70,90,80), nrow =2),
+     list("BCA","MCA","B.tech"))
> names(list_data) <- c("Students", "Marks", "Course")
> list_data
$Students
[1] "Shubham" "Nishka"  "Gunjan"
$Marks
     [,1] [,2] [,3]
[1,]   40   60   90
[2,]   80   70   80

$Course
$Course[[1]]
[1] "BCA"
$Course[[2]]
[1] "MCA"
$Course[[3]]
[1] "B.tech"
> #2)Accessing elements using index:-
> print(list_data[1])
$Students
[1] "Shubham" "Nishka"  "Gunjan"
> #3)Accessing elements using names:-
> print(list_data["Students"])
$Students
[1] "Shubham" "Nishka"  "Gunjan"

> print(list_data$Marks)
     [,1] [,2] [,3]
[1,]   40   60   90
[2,]   80   70   80
> #4)Merging Lists:-
> Even_list <- list(2,4,6)
> Odd_list <- list(1,3,5)
> # Merging the two lists.
> merged.list <- list(Even_list,Odd_list)
> print(merged.list)
[[1]]
[[1]][[1]]
[1] 2
```

[[1]][[2]]
[1] 4
[[1]][[3]]
[1] 6
[[2]]
[[2]][[1]]
[1] 1
[[2]][[2]]
[1] 3
[[2]][[3]]
[1] 5

7. Write a program for creation of Matrix and perform operations in R programming.

#creation of matrix:-

```
P <- matrix(c(5:16), nrow = 4, byrow = TRUE)
print(P)
Q <- matrix(c(3:14), nrow = 4, byrow = FALSE)
print(Q)
```

#operations on Matrix:-

#1)Addition:-

```
sum<-P+Q
print(sum)
```

#2)Subtraction:-

```
sub<-P-Q
print(sub)
```

#3)Multiplication(*):-

```
mult<-P*Q
print(mult)
```

#4)Multiplication(by constant):-

```
mult<-P*5
print(mult)
```

#5)Division:-

```
div<-P/Q
div
```

OUTPUT:-

#creation of matrix:-

```
>   P <- matrix(c(5:16), nrow = 4, byrow = TRUE)
>   print(P)
     [,1] [,2] [,3]
[1,]    5    6    7
[2,]    8    9   10
[3,]   11   12   13
[4,]   14   15   16
>
>   Q <- matrix(c(3:14), nrow = 4, byrow = FALSE)
```

```
>    print(Q)
     [,1] [,2] [,3]
[1,]    3    7   11
[2,]    4    8   12
[3,]    5    9   13
[4,]    6   10   14
>
> #operations on Matrix:-
> #1)Addition:-
>    sum<-P+Q
>    print(sum)
     [,1] [,2] [,3]
[1,]    8   13   18
[2,]   12   17   22
[3,]   16   21   26
[4,]   20   25   30
> #2)Subtraction:-
>    sub<-P-Q
>    print(sub)
     [,1] [,2] [,3]
[1,]    2   -1   -4
[2,]    4    1   -2
[3,]    6    3    0
[4,]    8    5    2
> #3)Multiplication(*):-
>    mult<-P*Q
>    print(mult)
     [,1] [,2] [,3]
[1,]   15   42   77
[2,]   32   72  120
[3,]   55  108  169
[4,]   84  150  224
> #4)Multiplication(by constant):-
>    mult<-P*5
>    print(mult)
     [,1] [,2] [,3]
[1,]   25   30   35
[2,]   40   45   50
[3,]   55   60   65
[4,]   70   75   80
> #5)Division:-
>    div<-P/Q
>   div
       [,1]     [,2]     [,3]
[1,] 1.666667 0.8571429 0.6363636
[2,] 2.000000 1.1250000 0.8333333
[3,] 2.200000 1.3333333 1.0000000
```

[4,] 2.333333 1.5000000 1.1428571
>

8. **Write a program for creation of Array an perform operations on array in R programming.**

#creation of Arrays:-

```
vec1 <-c(1,3,5)
vec2 <-c(10,11,12,13,14,15)
res <- array(c(vec1,vec2),dim=c(3,3,2))
print(res)

#Naming Of Arrays
col_names <- c("Col1","Col2","Col3")
row_names <- c("Row1","Row2","Row3")
matrix_names <- c("Matrix1","Matrix2")
res <-
array(c(vec1,vec2),dim=c(3,3,2),dimnames=list(row_names,col_names,matrix_names))
print(res)
```

OUTPUT: -

```
#creation of Arrays:-
> vec1 <-c(1,3,5)
> vec2 <-c(10,11,12,13,14,15)
> res <- array(c(vec1,vec2),dim=c(3,3,2))
> print(res)
, , 1

     [,1] [,2] [,3]
[1,]    1   10   13
[2,]    3   11   14
[3,]    5   12   15

, , 2

     [,1] [,2] [,3]
[1,]    1   10   13
[2,]    3   11   14
[3,]    5   12   15

>
> #Naming Of Arrays
> col_names <- c("Col1","Col2","Col3")
> row_names <- c("Row1","Row2","Row3")
> matrix_names <- c("Matrix1","Matrix2")
> res <-
array(c(vec1,vec2),dim=c(3,3,2),dimnames=list(row_names,col_names,matrix_names))
> print(res)
```

, , Matrix1

	Col1	Col2	Col3
Row1	1	10	13
Row2	3	11	14
Row3	5	12	15

, , Matrix2

	Col1	Col2	Col3
Row1	1	10	13
Row2	3	11	14
Row3	5	12	15

9. **Write a program in R for creating a factor of eight people with attributes first name, last name, gender and month of birth including print the levels of gender.**
10. **Write a program to create a list and giving the name to list elements.**

```
list_data <- list(c("Shubham","Nishka","Gunjan"), matrix(c(40,80,60,70,90,80),
nrow = 2),list("BCA","MCA","B.tech"))

 # Giving names to the elements in the list.
     names(list_data) <- c("Students", "Marks", "Course")
     print(list_data)
```

11. **Write a program in R to access the elements of list using names.**

```
list_data <- list(c("Shubham","Nishka","Gunjan"), matrix(c(40,80,60,70,90,80), nrow = 2),list("BCA","MCA","B.tech"))

 # Giving names to the elements in the list.
 names(list_data) <- c("Students", "Marks", "Course")

 #Accessing hte elements of list:-
     print(list_data["Students"])
     print(list_data$Marks)
     print(list_data["Course"])
```

12. **Write a program in R for creation of vector by using colon operator and sequence function.**

```
a<-4:-10
     a
     #Sequence function:
     seq_vec<-seq(1,4,by=0.5)
     seq_vec
     class(seq_vec)
```

13. **Write a program to demonstrate Importing and exporting of data in R programming.**

```
#IMPORT
getwd()
#Importing csv file.

path<-"C:/Users/Leena/OneDrive/Documents/candidate-elimination.csv"
content<-read.csv(path)
print(content)

#Importing Text file.

x<-read.table("C:/Users/Leena/OneDrive/Documents/file.txt",header=FALSE)
print(x)

#Importing CSV file using csv2.
```

```
x<-read.csv2("C:/Users/Leena/OneDrive/Documents/candidate-elimination.csv")
      print(x)
```

OUTPUT

```
   SKY TEMP  HUMID   WIND WATER FOREST OUTPUT
1 sunny warm normal strong  warm   same    yes
2 sunny warm   high strong  warm   same    yes
3 rainy cold   high strong  warm change     no
4 sunny warm   high strong  cool change    yes
```

```
SKY.TEMP.HUMID.WIND.WATER.FOREST.OUTPUT
1  sunny,warm,normal,strong,warm,same,yes
2   sunny,warm,high,strong,warm,same,yes
3   rainy,cold,high,strong,warm,change,no
4  sunny,warm,high,strong,cool,change,yes
```

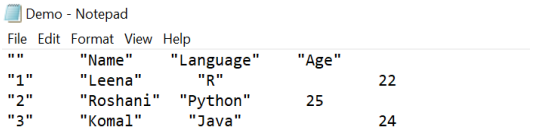
```
 SKY.TEMP.HUMID.WIND.WATER.FOREST.OUTPUT

1  sunny,warm,normal,strong,warm,same,yes
2   sunny,warm,high,strong,warm,same,yes
3   rainy,cold,high,strong,warm,change,no
4  sunny,warm,high,strong,cool,change,yes
```

EXPORT

1).Export a data frame to a text file using write.table().

```
      df=data.frame(
  "Name"=c("Leena","Roshani","Komal"),
  "Language"=c("R","Python","Java"),
  "Age"=c(22,25,24)
)
write.table(df,
        file="Demo.txt",
        sep = "\t",
        row.names = TRUE,
        col.names = NA)
```

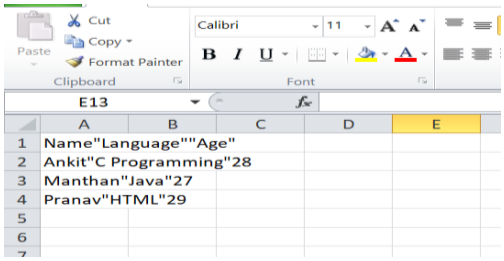
OUTPUT :



2).**Exporting Data to a csv file.**

```
df=data.frame(
 "Name"=c("Ankit","Manthan","Pranav"),
 "Language"=c("C Programming","Java","HTML"),
 "Age"=c(28,27,29)
)
write.table(df,
        file="myFile.csv",
        sep = "\t",
              row.names = FALSE)
```

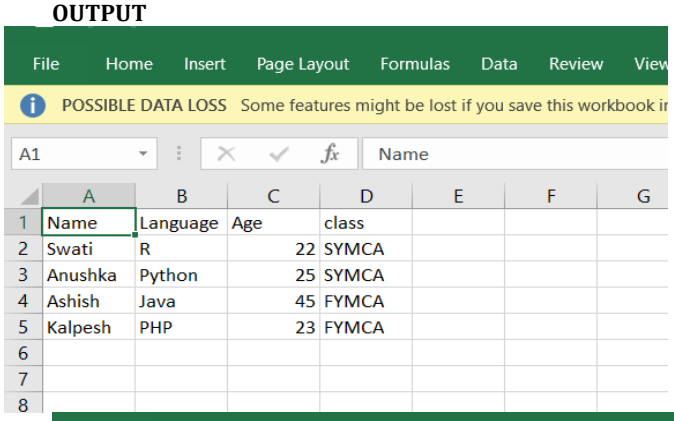
OUTPUT :

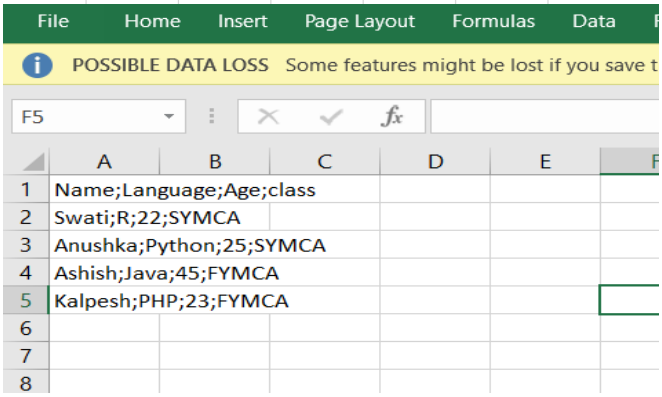


3)Exporting data to a csv2 file

```
library(readr)
df2=data.frame(
  "Name"=c("Swati","Anushka","Ashish","Kalpesh"),
        "Language"=c("R","Python","Java","PHP"),
        "Age"=c(22,25,45,23),
        "class"=c("SYMCA","SYMCA","FYMCA","FYMCA")
)
       write_csv(df2,path="Demo2.csv")
       write_csv2(df2,path="Demo3.csv")
```

OUTPUT



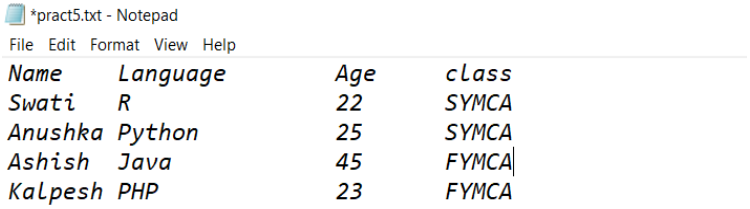


4)Exporting data using write_tsv()function

```
getwd()
#library(readr)
df2=data.frame(
  "Name"=c("Swati","Anushka","Ashish","Kalpesh"),
  "Language"=c("R","Python","Java","PHP"),
  "Age"=c(22,25,45,23),
  "class"=c("SYMCA","SYMCA","FYMCA","FYMCA")
  )
```

write_tsv(df2,path="pract5.txt")

OUTPUT



14. Write a program for Validating the data in R programming.

#Validating data:- data(cars)
head(cars, 3)

```
library(validate) rules <- validator(speed
>= 0,
              dist >= 0,                speed/dist <=
1.5,              cor(speed, dist)>=0.2)
out <- confront(cars, rules)
summary(out)
```

Output: -

```
data(cars) > head(cars, 3)
speed dist
1    4   2
2    4  10
3    7   4
>
>
> library(validate)
> rules <- validator(speed >= 0,
+               dist >= 0,
+               speed/dist <= 1.5,
+               cor(speed, dist)>=0.2)
> out <- confront(cars, rules) > summary(out)
  name items passes fails nNA error warning            expression
1   V1    50     50     0   0 FALSE   FALSE     speed - 0 >= -1e-08
2   V2    50     50     0   0 FALSE   FALSE      dist - 0 >= -1e-08
3   V3    50     48     2   0 FALSE   FALSE       speed/dist <= 1.5
4   V4     1      1     0   0 FALSE   FALSE cor(speed, dist) >= 0.2
```

15. Write a program for Exploring Data Manipulations (Summarizing, Sorting, Sub setting, Merging, Joining).

1) Summarizing:-

```
#create a data frame
data1<-data.frame(player=c('A','B','c','D','E'),

      runs=c(100,200,105,50,90),

          wickets=c(15,20,8,5,8)
          )
data1
```

```
#summarize method
summarize(data1,sum(runs),mean(runs),mode(wickets))
//summarize(data1)
```

---

2) Sorting:-

```
#creating data frame
dataBook=data.frame(Customers=c("Ruhi","James","Heera","Shubham","Joe","Priya),
                    Products=c("ProdA","ProdB","ProdC","ProdD","ProdE","prodF"),
                    Salary=c(500,600,450,700,300,400))
dataBook
#sorting the data frame in ascending order
arrange(dataBook,Salary)
#sorting the data frame in descending order
dataBook%>%arrange(desc(Salary))
```

---

3)Sub setting:-

```
#Subsetting in R using []operator:
#create vector
x<-1:15
cat("Original vector:",x,"\n")
#subsetting vector:
cat("First 5 values of vector:",x[1:5],"\n")
cat("Without values present at index 1,2and 3",x[-c(1,2,3),"\n"])
#Subsetting in R using [[]]operator:
#create list:
ls<-list(a=1,b=2,c=10,d=20)
cat("Original List:\n")
print(ls)
#select first element of list:
cat("Element of list:",ls[[3]],"\n")
#Subsetting using c() function:
ls2<-list(a=list(x=1,y="students"),b=1:10)
ls2
cat("Using c() function:\n")
//print(ls2[[c(1,2)]])
//print(ls2[[1]][[2]])
#Subsetting Using $ operator:
ls3<-list(a="Roshani",b=1,c="Hello")
ls3
cat("Using $ operator:\n")
print(ls3$a)
```

---

4) Merging: -

```
#Merge DataFrames by Row Names:-
data_frame1<-data.frame(No=c(1:5),
                        Name=letters[1:5],
                        Salary=c(200,200,300,NA,300)
                        )
data_frame1

data_frame2<-data.frame(No=c(6:8),
                        Name=letters[8:10],
                        Salary=c(400,350,NA)
                        )
data_frame2

data_frame_merge<-merge(data_frame1,data_frame2,by='row.names',all=TRUE)
```

```
print("Merged DataFrame")
print(data_frame_merge)
```

---

5) Joining:-

```
#Using Inner join:-
data1<-data.frame(ID=c(1:5))
data2<-data.frame(ID=c(4:8))
inner_join(data1,data2,by="ID")

#Using Left join:-
data1<-data.frame(ID=c(1:5),
                  Name=c("Rutuja","Lokesh","Ram","Purvi","Nita"))
data2<-data.frame(ID=c(4:8),
                  Marks=c(70,85,80,90,75))
left_join(data1,data2,by="ID")
```

OUTPUT: -

```
#1)Summarizing:-
> #create a data frame
> data1<-data.frame(player=c('A','B','c','D','E'),
+                   runs=c(100,200,105,50,90),
+                   wickets=c(15,20,8,5,8)
+                   )
> data1
  player runs wickets
1      A  100      15
2      B  200      20
3      c  105       8
4      D   50       5
5      E   90       8

> #summarize method
> summarize(data1,sum(runs),mean(runs),mode(wickets))
  sum(runs) mean(runs) mode(wickets)
1       545       109       numeric
> #-------------------------------------------------------------------------------
> #2)Sorting:-
> #creating data frame
> dataBook=data.frame(Customers=c("Ruhi","James","Heera","Shubham","Joe","Priya"),
+                     Products=c("ProdA","ProdB","ProdC","ProdD","ProdE","prodF"),
+                     Salary=c(500,600,450,700,300,400))
> dataBook
  Customers Products Salary
1      Ruhi    ProdA    500
2     James    ProdB    600
3     Heera    ProdC    450
4   Shubham    ProdD    700
5       Joe    ProdE    300
6     Priya    prodF    400
> #sorting the data frame in ascending order
> arrange(dataBook,Salary)
  Customers Products Salary
1       Joe    ProdE    300
2     Priya    prodF    400
3     Heera    ProdC    450
4      Ruhi    ProdA    500
```

```
5    James    ProdB    600
6   Shubham    ProdD    700
> #sorting the data frame in descending order
> dataBook%>%arrange(desc(Salary))
  Customers Products Salary
1   Shubham    ProdD    700
2    James    ProdB    600
3     Ruhi    ProdA    500
4    Heera    ProdC    450
5    Priya    prodF    400
6     Joe    ProdE    300
> #-------------------------------------------------------------------------------
> #3)Subsetting:-
> #Subsetting in R using []operator:
> #create vector
> x<-1:15
> cat("Original vector:",x,"\n")
Original vector: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
> #subsetting vector:
> cat("First 5 values of vector:",x[1:5],"\n")
First 5 values of vector: 1 2 3 4 5
> cat("Without values present at index 1,2and 3:",x[-c(1,2,3)])
Without values present at index 1,2and 3: 4 5 6 7 8 9 10 11 12 13 14 15> #Subsetting in R using [[]]operator:
> #create list:
> ls<-list(a=1,b=2,c=10,d=20)
> cat("Original List:\n")
Original List:
> print(ls)
$a
[1] 1

$b
[1] 2

$c
[1] 10

$d
[1] 20

> #select first element of list:
> cat("Element of list:",ls[[3]],"\n")
Element of list: 10
> #Subsetting using c() function:
> ls2<-list(a=list(x=1,y="students"),b=1:10)
> ls2
$a
$a$x
[1] 1
$a$y
[1] "students"
$b
 [1]  1  2  3  4  5  6  7  8  9 10

> cat("Using c() function:\n")
Using c() function:
> print(ls2[[c(1,2)]])
```

```
[1] "students"
> print(ls2[[1]][[2]])
[1] "students"
> #Subsetting Using $ operator:
> ls3<-list(a="Roshani",b=1,c="Hello")
> ls3
$a
[1] "Roshani"

$b
[1] 1

$c
[1] "Hello"

> cat("Using $ operator:\n")
Using $ operator:
> print(ls3$a)
[1] "Roshani"
> #-------------------------------------------------------------------------------
> #4)Merging:-
> #Merge DataFrames by Row Names:-
> data_frame1<-data.frame(No=c(1:5),
+                         Name=letters[1:5],
+                         Salary=c(200,200,300,NA,300)
+                         )
> data_frame1
  No Name Salary
1  1    a    200
2  2    b    200
3  3    c    300
4  4    d     NA
5  5    e    300
>

> data_frame2<-data.frame(No=c(6:8),

+                         Name=letters[8:10],
+                         Salary=c(400,350,NA)
+                         )
> data_frame2
  No Name Salary
1  6    h    400
2  7    i    350
3  8    j     NA
>
 data_frame_merge<merge(data_frame1,data_frame2,by='row.names',all=TRUE)
> print("Merged DataFrame")
[1] "Merged DataFrame"
> print(data_frame_merge)
  Row.names No.x Name.x Salary.x No.y Name.y Salary.y
1         1    1      a      200    6      h      400
2         2    2      b      200    7      i      350
3         3    3      c      300    8      j       NA
4         4    4      d       NA   NA   <NA>       NA
5         5    5      e      300   NA   <NA>       NA
> #-------------------------------------------------------------------------------
```

```
> #5)Joining:-
> #Using Inner join:-
> data1<-data.frame(ID=c(1:5))
> data2<-data.frame(ID=c(4:8))
> inner_join(data1,data2,by="ID")
  ID
1  4
2  5
>
> #Using Left join:-
> data1<-data.frame(ID=c(1:5),
+                   Name=c("Rutuja","Lokesh","Ram","Purvi","Nita"))
> data2<-data.frame(ID=c(4:8),

+                   Marks=c(70,85,80,90,75))

> left_join(data1,data2,by="ID")
  ID   Name Marks
1  1 Rutuja    NA
2  2 Lokesh    NA
3  3    Ram    NA
4  4  Purvi    70
5  5   Nita    85
```

16. **Write a program to implement the T-test analysis techniques using R programming.**

```
x <- sample(c(1:100),size=20,replace=TRUE)
y <- sample(c(1:100),size=20,replace=TRUE)
t.test(x,y)
```

17. **Write a program to implement the Chi-square test analysis techniques using R programming.**

Chi-Square Test:-

```
library(MASS)
#create DataFrame:
print(str(survey))
# Create a data frame from the main data set.
stu_data = data.frame(survey$Smoke,survey$Exer)

# Create a contingency table with the needed variables.
stu_data = table(survey$Smoke,survey$Exer)
print(stu_data)
```

OUTPUT:-

```
> #3)Chi-Square Test:-
> library(MASS)
> #create DataFrame:
> print(str(survey))
'data.frame':          237 obs. of  12 variables:
 $ Sex   : Factor w/ 2 levels "Female","Male": 1 2 2 2 2 1 2 1 2 2 ...
 $ Wr.Hnd: num  18.5 19.5 18 18.8 20 18 17.7 17 20 18.5 ...
 $ NW.Hnd: num  18 20.5 13.3 18.9 20 17.7 17.7 17.3 19.5 18.5 ...
 $ W.Hnd : Factor w/ 2 levels "Left","Right": 2 1 2 2 2 2 2 2 2 2 ...
 $ Fold  : Factor w/ 3 levels "L on R","Neither",..: 3 3 1 3 2 1 1 3 3 3 ...
 $ Pulse : int  92 104 87 NA 35 64 83 74 72 90 ...
 $ Clap  : Factor w/ 3 levels "Left","Neither",..: 1 1 2 2 3 3 3 3 3 3 ...
 $ Exer  : Factor w/ 3 levels "Freq","None",..: 3 2 2 2 3 3 1 1 3 3 ...
 $ Smoke : Factor w/ 4 levels "Heavy","Never",..: 2 4 3 2 2 2 2 2 2 2 ...
 $ Height: num  173 178 NA 160 165 ...
 $ M.I   : Factor w/ 2 levels "Imperial","Metric": 2 1 NA 2 2 1 1 2 2 2 ...
 $ Age   : num  18.2 17.6 16.9 20.3 23.7 ...
NULL
> # Create a data frame from the main data set.
> stu_data = data.frame(survey$Smoke,survey$Exer)
```

```
>
> # Create a contingency table with the needed variables.
> stu_data = table(survey$Smoke,survey$Exer)
>
> print(stu_data)

  Freq None Some
  Heavy   7    1    3
  Never  87   18   84
  Occas  12    3    4
  Regul   9    1    7
```

18. Write a program to implement the Correlation analysis techniques using R programming.

Correlation Test:-

```
cor.test(matcars$mpg,matcars$hp)
```

Output:-

```
> #2)Correlation Test:-
> cor.test(mtcars$mpg,mtcars$hp)

        Pearson's product-moment correlation

data:  mtcars$mpg and mtcars$hp
t = -6.7424, df = 30, p-value = 1.788e-07
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 -0.8852686 -0.5860994
sample estimates:
      cor
-0.7761684
```

19. Write a program to implement the Regression analysis techniques using R programming.

```
# R program to illustrate
# Linear Regression
# Height vector
x <- c(153, 169, 140, 186, 128,
     136, 178, 163, 152, 133)

# Weight vector
y <- c(64, 81, 58, 91, 47, 57,
     75, 72, 62, 49)

# Create a linear regression model
model <- lm(y~x)

# Print regression model
print(model)

# Find the weight of a person With height 182
df <- data.frame(x = 182)
res <-  predict(model, df)
cat("\nPredicted value of a person
          with height = 182")
print(res)

# Output to be present as PNG file
png(file = "linearRegGFG.png")

# Plot
plot(x, y, main = "Height vs Weight Regression model")
abline(lm(y~x))

# Save the file.
```

```
dev.off()
```

Output:

```
Call:
lm(formula = y ~ x)
Coefficients:
(Intercept)           x
  -39.7137      0.6847
Predicted value of a person with height = 182
     1
84.9098
```

```
# Multiple Linear Regression
# Using airquality dataset
input <- airquality[1:50,c("Ozone", "Wind", "Temp")]

# Create regression model
model <- lm(Ozone~Wind + Temp,data = input)

# Print the regression model
cat("Regression model:\n")
print(model)

# Output to be present as PNG file
png(file = "multipleRegGFG.png")

# Plot
plot(model)

# Save the file.
dev.off()
```

Output:

```
Regression model:

Call:

lm(formula = Ozone ~ Wind + Temp, data = input)

Coefficients:

(Intercept)        Wind        Temp

   -58.239      -0.739       1.329
```

```
# Logistic Regression
# Using mtcars dataset
# To create the logistic model
model <- glm(formula = vs ~ wt,family = binomial,data = mtcars)

# Creating a range of wt values
x <- seq(min(mtcars$wt),max(mtcars$wt),0.01)

# Predict using weight
y <- predict(model, list(wt = x), type = "response")

# Print model
print(model)

# Output to be present as PNG file
png(file = "LogRegGFG.png")

# Plot
plot(mtcars$wt, mtcars$vs, pch = 16, xlab = "Weight", ylab = "VS")
```

```
lines(x, y)

# Saving the file
dev.off()
```

Output:

```
Call:  glm(formula = vs ~ wt, family = binomial, data = mtcars)

Coefficients:

(Intercept)         wt

    5.715      -1.911

Degrees of Freedom: 31 Total (i.e. Null);  30 Residual

Null Deviance:      43.86

Residual Deviance: 31.37        AIC: 35.37
```

20. **Write a program to implement the Analysis of Variance analysis techniques using R programming**
21. **Write a program to implement the ANNOVA analysis techniques using R programming.**

```
# Installing the package
install.packages("dplyr")

# Loading the package
library(dplyr)

# Variance in mean within group and between group
boxplot(mtcars$disp~factor(mtcars$gear) xlab = "gear", ylab = "disp")

# Step 1: Setup Null Hypothesis and Alternate Hypothesis
# H0 = mu = mu01 = mu02(There is no difference
# between average displacement for different gear)
# H1 = Not all means are equal
# Step 2: Calculate test statistics using aov function
mtcars_aov <- aov(mtcars$disp~factor(mtcars$gear))
summary(mtcars_aov)

# Step 3: Calculate F-Critical Value
# For 0.05 Significant value, critical value = alpha = 0.05
# Step 4: Compare test statistics with F-Critical value
# and conclude test p < alpha, Reject Null Hypothesis
```

Output: