**1.Data frame**

**Write a program to create a data frame for the following data.**

| Name | Age | Gender | GPA |
|---|---|---|---|
| Saanvi | 27 | F | 3.26 |
| Aarav | 55 | M | 3.75 |
| Arjun | 34 | M | 2.98 |
| Anika | 42 | F | 3.40 |
| Divya | 20 | F | 2.75 |
| Aditya | 27 | M | 3.32 |
| Krishna | 34 | M | 3.68 |
| Meera | 42 | F | 3.97 |

Print,

a) list of names

b) Average age and gpa

c) Separate data gender wise

d) Frequency table of Age and Gender

```
# Data Frame

Name <- c("Saanvi","Aarav","Arjun","Anika","Divya","Aditya","Krishna","Meera")
Age <- c(27, 55, 34, 42, 20, 27, 34, 42)
Gender <- c("F", "M", "M", "F", "F", "M", "M", "F")
GPA <- c(3.26, 3.75, 2.98, 3.40, 2.75, 3.32, 3.68, 3.97)
df <- data.frame(Name, Age, Gender, GPA)
names <- df$Name
print(names)
averages <- sapply(df[c('Age', 'GPA')], mean)
print(averages)
```

males <- subset(df, Gender=='M')

print(males)

females <- subset(df, Gender=='F')

print(females)

table(df$Age, df$Gender)

**2.List**

**Create a list and access the 3rd element**

# Creating a list containing a vector, a matrix and a list.

list_data <- list(c("Shubham","Arpita","Nishka"), matrix(c(40,80,60,70,90,80), nrow = 2),

 list("BCA","MCA","B.tech"))

# Accessing the first element of the list.

print(list_data[1])

# Accessing the third element. The third element is also a list, so all its elements will be printed.

print(list_data[3])

**Special apply family function**

**3.Create a sample and use apply function to find mean and sum**

```
# create sample data
sample_matrix <- matrix(C<-(1:10),nrow=3, ncol=10)

print( "sample matrix:")
sample_matrix

# Use apply() function across row to find sum
print("sum across rows:")
apply( sample_matrix, 1, sum)

# use apply() function across column to find mean
print("mean across columns:")
apply( sample_matrix, 2, mean)
```

**4th Module**

**Function**

**4. Write a program and call a function with arguments**

```
# Create a function with arguments.

new.function <- function(a,b,c) {

  result <- a * b + c

  print(result)

}

# Call the function by position of arguments.

new.function(5,3,11)

# Call the function by names of the arguments.

new.function(a = 11, b = 5, c = 3)
```

**5. Write a program and call a function with default arguments 3 & 6 and call with some new values**

```
# Create a function with arguments.
new.function <- function(a = 3, b = 6) {
   result <- a * b
   print(result)
}

# Call the function without giving any argument.
new.function()

# Call the function with giving new values of the argument.
new.function(9,5)
```

**5th module**

**6. create a pointer and change the value**

```
myP=newPointer(7)

print(myP$value)  # returns '7'

newP=copy(myP)
```

```
copyP=myP

updatePointerValue(myP,9)

print(copyP$value)  # returns '9'

print(newP$value)  # returns '7'
```

## 7. Write R program illustrating error handling

```
#Applying tryCatch
tryCatch(

  # Specifying expression
  expr = {
    1 + 1
    print("Everything was fine.")
  },
  # Specifying error message
  error = function(e){
    print("There was an error message.")
  },

  warning = function(w){
    print("There was a warning message.")
  },

  finally = {
    print("finally Executed")
  }
)
```