1. Write a R program to take input from the user (name and age) and display the values. Also print the version of R installation.

   Ans: #input name

```
name = readline(prompt="Input your name: ")
age =  readline(prompt="Input your age: ")
print(paste("My name is",name, "and I am",age ,"years old."))
print(R.version.string)
```

   o/p\:Input your name: John

```
Input your age: 25
"My name is John and I am 25 years old."
```

2. Write a R program to get the details of the objects in memory.

   Ans: name = "Python";

```
n1 =  10;
n2 =  0.5
nums = c(10, 20, 30, 40, 50, 60)
print(ls())
print("Details of the objects in memory:")
print(ls.str())
```

   op:

```
[1] "name" "n1"   "n2"   "nums"
[1] "Details of the objects in memory:"
name :  chr "Python"
n1 :  num 10
n2 :  num 0.5
nums :  num [1:6] 10 20 30 40 50 60
```

3. Write a R program to create a sequence of numbers from 20 to 50 and find the mean of numbers from 20 to 60 and sum of numbers from 51 to 91.

   ans :print("Sequence of numbers from 20 to 50:")

```
print(seq(20,50))
print("Mean of numbers from 20 to 60:")
print(mean(20:60))
print("Sum of numbers from 51 to 91:")
print(sum(51:91))
```

   op:

```
[1] "Sequence of numbers from 20 to 50:"
[1] 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49
50
[1] "Mean of numbers from 20 to 60:"
[1] 40
[1] "Sum of numbers from 51 to 91:"
[1] 3196
```

4. Write a R program to create a vector which contains 10 random integer values between -50 and +50.

```
Ans: v = sample(-50:50, 10, replace=TRUE)
print("Content of the vector:")
print("10 random integer values between -50 and +50:")
print(v)
op:  [1] "Content of the vector:"
[1] "10 random integer values between -50 and +50:"
[1]  12 -23   5  37  45 -15  28  -3  17  10
```

5. Write a R program to get the first 10 Fibonacci numbers.

```
Ans: Fibonacci <- numeric(10)
Fibonacci[1] <- Fibonacci[2] <- 1
for (i in 3:10) Fibonacci[i] <- Fibonacci[i - 2] + Fibonacci[i - 1]
cat("First 10 Fibonacci numbers:\n", Fibonacci)
op: First 10 Fibonacci numbers:
 1 1 2 3 5 8 13 21 34 55
```

6. Write a R program to get all prime numbers up to a given number (based on the sieve of Eratosthenes).

```
Ans: prime_numbers <- function(n) {
  if (n >= 2) {
    x = seq(2, n)
    prime_nums = c()
    for (i in seq(2, n)) {
      if (any(x == i)) {
        prime_nums = c(prime_nums, i)
        x = c(x[(x %% i) != 0], i)
      }
    }
    return(prime_nums)
  }
  else
  {
    stop("Input number should be at least 2.")
  }
}
prime_numbers(12)
op: ip prime_numbers(12)
op[1]  2  3  5  7 11
```

7. Write a R program to print the numbers from 1 to 100 and print "Fizz" for multiples of 3, print "Buzz" for multiples of 5, and print "FizzBuzz" for multiples of both.

```
Ans: for (n in 1:100) {
  if (n %% 3 == 0 & n %% 5 == 0) {print("FizzBuzz")}
  else if (n %% 3 == 0) {print("Fizz")}
  else if (n %% 5 == 0) {print("Buzz")}
  else print(n)
}
```

Op:

```
1
2
Fizz
4
Buzz
Fizz
```

**8.** Write a R program to extract first 10 english letter in lower case and last 10 letters in upper case and extract letters between 22nd to 24th letters in upper case.

Ans:

```
print("First 10 letters in lower case:")
t = (letters[1:10])
print(t)
print("Last 10 letters in upper case:")
t = tail(LETTERS, 10)
print(t)
print("Letters between 22nd to 24th letters in upper case:")
e = tail(LETTERS[22:24])
print(e)
```

op:

```
[1] "First 10 letters in lower case:"
[1] "a" "b" "c" "d" "e" "f" "g" "h" "i" "j"
[1] "Last 10 letters in upper case:"
[1] "Q" "R" "S" "T" "U" "V" "W" "X" "Y" "Z"
[1] "Letters between 22nd to 24th letters in upper case:"
[1] "X" "Y" "Z"
```

**9.** Write a R program to find the factors of a given number.

Ans:

```
print("First 10 letters in lower case:")
t = (letters[1:10])
print(t)
print("Last 10 letters in upper case:")
t = tail(LETTERS, 10)
print(t)
print("Letters between 22nd to 24th letters in upper case:")
e = tail(LETTERS[22:24])
print(e)
```

op:

```
The factors of 4 are:
1
2
4
The factors of 7 are:
1
7
```

**10.** Write a R program to find the maximum and the minimum value of a given vector.

```
Ans: nums = c(10, 20, 30, 40, 50, 60)
print('Original vector:')
print(nums)
print(paste("Maximum value of the said vector:",max(nums)))
print(paste("Minimum value of the said vector:",min(nums)))
op: [1] "Original vector:"
[1] 10 20 30 40 50 60
[1] "Maximum value of the said vector: 60"
[1] "Minimum value of the said vector: 10"
```

**11.** Write a R program to get the unique elements of a given string and unique numbers of vector.

```
Ans: str1 = "The quick brown fox jumps over the lazy dog."
print("Original vector(string)")
print(str1)
print("Unique elements of the said vector:")
print(unique(tolower(str1)))
nums = c(1, 2, 2, 3, 4, 4, 5, 6)
print("Original vector(number)")
print(nums)
print("Unique elements of the said vector:")
print(unique(nums))
op:  [1] "Original vector(string)"
[1] "The quick brown fox jumps over the lazy dog."
[1] "Unique elements of the said vector:"
[1] "the quick brown fox jumps over lazy dog."
2) [1] "Original vector(number)"
[1] 1 2 2 3 4 4 5 6
[1] "Unique elements of the said vector:"
[1] 1 2 3 4 5 6
```

**12.** Write a R program to create three vectors a,b,c with 3 integers. Combine the three vectors to become a 3×3 matrix where each column represents a vector. Print the content of the matrix.

```
Ans: A<-c(1,2,3)
B<-c(4,5,6)
C<-c(7,8,9)
mat<-cbind(A,B,C)
mat
op:    A B C
[1,] 1 4 7
[2,] 2 5 8
[3,] 3 6 9
```

**13.** Write a R program to create a list of random numbers in normal distribution and count occurrences of each value.

```
Ans: n = floor(rnorm(10, 5, 10))
```

```
print('List of random numbers in normal distribution:')
print(n)
t = table(n)
print("Count occurrences of each value:")
print(t)
op: [1] "List of random numbers in normal distribution:"
 [1]  3  0  0  4 -3  0  4  6  4  6
[1] "Count occurrences of each value:"
n
-3  0  3  4  6
 1  3  1  3  2
```

**14.** Write a R program to read the .csv file and display the content.

Ans:
```
data <- read.csv("data.csv")
print("Content of the CSV file:")
print(data)
op: [1] "Content of the CSV file:"
  Column1 Column2 Column3
1      1       A      10
2      2       B      20
3      3       C      30
4      4       D      40
5      5       E      50
```

**15.** Write a R program to create three vectors numeric data, character data and logical data. Display the content of the vectors and their type.

Ans:
```
a = c(1, 2, 5, 3, 4, 0, -1, -3)
b = c("Red", "Green", "White")
c = c(TRUE, TRUE, TRUE, FALSE, TRUE, FALSE)
print(a)
print(typeof(a))
print(b)
print(typeof(b))
print(c)
print(typeof(c))
op: [1]  1  2  5  3  4  0 -1 -3
[1] "double"
[1] "Red"   "Green" "White"
[1] "character"
[1]  TRUE  TRUE  TRUE FALSE  TRUE FALSE
[1] "logical"
```

**16.** Write a R program to create a 5 x 4 matrix , 3 x 3 matrix with labels and fill the matrix by rows and 2 × 2 matrix with labels and fill the matrix by columns.

```
Ans: m1 = matrix(1:20, nrow=5, ncol=4)
print("5 × 4 matrix:")
print(m1)
cells = c(1,3,5,7,8,9,11,12,14)
rnames = c("Row1", "Row2", "Row3")
cnames = c("Col1", "Col2", "Col3")
m2 = matrix(cells, nrow=3, ncol=3, byrow=TRUE, dimnames=list(rnames, cnames))
print("3 × 3 matrix with labels, filled by rows: ")
print(m2)
print("3 × 3 matrix with labels, filled by columns: ")
m3 = matrix(cells, nrow=3, ncol=3, byrow=FALSE, dimnames=list(rnames, cnames))
print(m3)
```

```
op: [1] "3 × 3 matrix with labels, filled by rows:"
     Col1 Col2 Col3
Row1    1    3    5
Row2    7    8    9
Row3   11   12   14
```

**17.** Write a R program to create an array, passing in a vector of values and a vector of dimensions. Also provide names for each dimension.

```
Ans: a =  array(
  6:30,
  dim = c(4, 3, 2),
  dimnames = list(
    c("Col1", "Col2", "Col3", "Col4"),
    c("Row1", "Row2", "Row3"),
    c("Part1", "Part2")
  )
)
print(a)
op:
```

**18.** Write a R program to create an array with three columns, three rows, and two "tables", taking two vectors as input to the array. Print the array.

```
Ans: v1 = c(1, 3, 5, 7)
v2 = c(2, 4, 6, 8, 10)
arra1 = array(c(v1, v2),dim = c(3,3,2))
print(arra1)
```

**19.** Write a R program to create a list of elements using vectors, matrices and a functions. Print the content of the list.

```
Ans: l = list(
  c(1, 2, 2, 5, 7, 12),
  month.abb,
  matrix(c(3, -8, 1, -3), nrow = 2),
  asin
)
```

```
print("Content of the list:")
print(l)
```

**20.** Write a R program to draw an empty plot and an empty plot specify the axes limits of the graphic

Ans:
```
#print("Empty plot:")
plot.new()
#print("Empty plot specify the axes limits of the graphic:")
plot(1, type="n", xlab="", ylab="", xlim=c(0, 20), ylim=c(0, 20))
```

**21.** Write a R program to create an array of two 3x3 matrices each with 3 rows and 3 columns from two given two vectors. Print the second row of the second matrix of the array and the element in the 3rd row and 3rd column of the 1st matrix.

Ans:
```
a <- c(1,2,3)
b <- c(4,5,6)
c <- c(7,8,9)
mat1 <- rbind(a,b,c)
print(mat1[3:3])
d <- c(3,2,1)
e <- c(6,4,5)
f <- c(6,9,7)
mat2 <- rbind(d,e,f)
print(mat2[2,])
```

**22.** Write a R program to combine three arrays so that the first row of the first array is followed by the first row of the second array and then first row of the third array.

Ans:
```
# Create three sample arrays
array1 <- matrix(1:3, nrow = 1)
array2 <- matrix(4:6, nrow = 1)
array3 <- matrix(7:9, nrow = 1)

combined_array <- rbind(array1, array2, array3)
print(combined_array)
```

**23.** Write a R program to create an array using four given columns, three given rows, and two given tables and display the content of the array.

Ans:
```
a = array(
  6:30,
  dim = c(4, 3, 2),
  dimnames = list(
    c("Col1", "Col2", "Col3", "Col4"),
    c("Row1", "Row2", "Row3"),
    c("Part1", "Part2")
  )
)
print(a)
```

**24.** Write a R program to create a two-dimensional 5x3 array of sequence of even integers greater than 50.

```
Ans: # Define the dimensions of the array

rows <- 5

cols <- 3


# Create an empty matrix to store the even integers

even_matrix <- matrix(nrow = rows, ncol = cols)


# Initialize a variable to generate even integers greater than 50

even_num <- 52

for (i in 1:rows) {

  for (j in 1:cols) {

    even_matrix[i, j] <- even_num

    even_num <- even_num + 2

  }

}

print(even_matrix)
```

**25.** Create below data frame exam_data = data.frame(

```
name = c('Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael', 'Matthew', 'Laura', 'Kevin', 'Jonas'),
score = c(12.5, 9, 16.5, 12, 9, 20, 14.5, 13.5, 8, 19),
attempts = c(1, 3, 2, 3, 2, 3, 1, 1, 2, 1),
qualify = c('yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes')
)
```

a. Write a R program to extract 3rd and 5th rows with 1st and 3rd columns from a given data frame

b. Write a R program to add a new column named country in a given data frame

Country<-c("USA","USA","USA","USA","UK","USA","USA","India","USA","USA")

c. Write a R program to add new row(s) to an existing data frame

new_exam_data = data.frame(name = c('Robert', 'Sophia'),score = c(10.5, 9), attempts = c(1, 3),qualify = c('yes', 'no'))

d. Write a R program to sort a given data frame by name and score

e. Write a R program to save the information of a data frame in a file and display the information of the file.

```
Ans: library(dplyr)

# Step 1
exam_data = data.frame(
  name = c('Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael', 'Matthew', 'Laura', 'Kevin',
       'Jonas'),
  score = c(12.5, 9, 16.5, 12, 9, 20, 14.5, 13.5, 8, 19),
  attempts = c(1, 3, 2, 3, 2, 3, 1, 1, 2, 1),
  qualify = c('yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes')
)
print(exam_data)

# step 2
selected_data <- exam_data[c(3, 5), c(1, 3)]
print(selected_data)

# Step 3
exam_data$country <- country<-c("USA","USA","USA","USA","UK","USA","USA","India","USA","USA")
print(exam_data)

# Step 4
new_exam_data <- data.frame(
  name = c('Robert', 'Sophia'),
  score = c(10.5, 9),
  attempts = c(1, 3),
  qualify = c('yes', 'no')
)
updated_exam_data <- bind_rows(exam_data, new_exam_data)
print(updated_exam_data)

# Step 5
sorted_exam_data <- exam_data[order(exam_data$name, exam_data$score), ]
print(sorted_exam_data)

# step 6
write.csv(exam_data, file = "exam_data.csv", row.names = FALSE)
read_exam_data <- read.csv("exam_data.csv")
print(read_exam_data)
```

26. Write a R program to call the (built-in) dataset airquality. Check whether it is a data frame or not? Order the entire data frame by the first and second column. remove the variables 'Solar.R' and 'Wind' and display the data frame.

```
Ans: data("airquality")
if (is.data.frame(airquality)) {
  print("airquality is a data frame.")
} else {
  print("airquality is not a data frame.")
}
ordered_airquality <- airquality[order(airquality$Month, airquality$Day), ]
modified_airquality <- ordered_airquality[, !(names(ordered_airquality) %in% c("Solar.R",
"Wind"))]
print(modified_airquality)
```

27. Write a R program to create a factor corresponding to height of women data set , which inbuild in R, contains height and weights for a sample of women.

Ans: data("women")

```
# print(women)

height_factor <- factor(women$height)

weight_factor <- factor(women$weight)

print(levels(height_factor))

print(levels(weight_factor))
```

28. Write a R program to extract the five of the levels of factor created from a random sample from the LETTERS (Part of the base R distribution.)

Ans:

```
set.seed(123)

random_letters <- sample(LETTERS, 26, replace = TRUE)

letter_factor <- factor(random_letters)

first_five_levels <- levels(letter_factor)[1:5]

print(first_five_levels)
```

29. **Iris** dataset is a very famous dataset in almost all data mining, machine learning courses, and it has been an R build-in dataset. The dataset consists of 50 samples from each of three species of Iris flowers (Iris setosa, Iris virginica and Iris versicolor). Four features(variables) were measured from each sample, they are the **length** and the **width** of sepal and petal, in centimetres. Perform the following EDA steps .

(i)Find dimension, Structure, Summary statistics, Standard Deviation of all features.
(ii)Find mean  and standard deviation of features groped by three species of Iris flowers (Iris setosa, Iris virginica and Iris versicolor)
(iii)Find quantile value of sepal width and length
(iV)create new data frame named iris1 which have  a new column name **Sepal.Length.Cate** that categorizes "Sepal.Length" by quantile

(V) Average value of numerical varialbes by two categorical variables: Species and Sepal.Length.Cate:
(vi) Average mean value of numerical varialbes by Species and Sepal.Length.Cate
(vii)Create Pivot Table based on Species and Sepal.Length.Cate.
Ans: data("iris")

```
# Step 1
print(dim(iris))
print(str(iris))
print(summary(iris))
print(apply(iris[, 1:4], 2, sd))

# Step 2
mean_by_species <- aggregate(. ~ Species, data = iris, FUN = mean)
print("Mean of features grouped by species:")
print(mean_by_species)

std_dev_by_species <- aggregate(. ~ Species, data = iris, FUN = sd)
print("Standard Deviation of features grouped by species:")
print(std_dev_by_species)

# Step 3
quantiles_sepal_width <- quantile(iris$Sepal.Width, probs = c(0.25, 0.5, 0.75))
quantiles_sepal_length <- quantile(iris$Sepal.Length, probs = c(0.25, 0.5, 0.75))

print(quantiles_sepal_width)
print(quantiles_sepal_length)

# Step 4
quantiles <- quantile(iris$Sepal.Length, probs = c(0, 0.25, 0.5, 0.75, 1))

categorize_sepal_length <- function(sepal_length) {
  if (sepal_length <= quantiles[2]) {
    return("Q1")
  } else if (sepal_length <= quantiles[3]) {
    return("Q2")
  } else if (sepal_length <= quantiles[4]) {
    return("Q3")
  } else {
    return("Q4")
  }
}
iris1 <- iris
iris1$Sepal.Length.Cate <- sapply(iris$Sepal.Length, categorize_sepal_length)
head(iris1)
```

```
# step 5
iris$Sepal.Length.Cate <- sapply(iris$Sepal.Length, categorize_sepal_length)

avg_by_categories <- aggregate(. ~ Species + Sepal.Length.Cate, data = iris, FUN =
mean)

print("Average value of numerical variables by Species and Sepal.Length.Cate:")
print(avg_by_categories)

# Step 6
avg_means_by_categories <- aggregate(. ~ Species + Sepal.Length.Cate, data = iris,
                                     FUN = function(x) mean(x, na.rm = TRUE))

# Print the mean of average values
print("Average mean value of numerical variables by Species and Sepal.Length.Cate:")
print(avg_means_by_categories)

# step 7
pivot_table <- iris %>% group_by(Species, Sepal.Length.Cate) %>%
summarise(mean_Sepal.Width = mean(Sepal.Width, na.rm = TRUE),
          mean_Petal.Length = mean(Petal.Length, na.rm = TRUE),
          mean_Petal.Width = mean(Petal.Width, na.rm = TRUE))

# Print the pivot table
print("Pivot Table based on Species and Sepal.Length.Cate:")
print(pivot_table)
```

30. Randomly Sample the iris dataset such as 80% data for training and 20% for test and create Logistics regression with train data, use species as target and petals width and length as feature variables , Predict the probability of the model using test data, Create Confusion matrix for above test model

```
Ans: library(caret)
library(nnet)
data("iris")
set.seed(123)
# Split data (80% train, 20% test)
splitIndex <- createDataPartition(iris$Species, p = 0.8, list = FALSE)
train_data <- iris[splitIndex,]
test_data <- iris[-splitIndex,]
# Train a multinomial logistic regression model and predict
model <- train(Species ~ Petal.Width + Petal.Length, data = train_data, method = "multinom")
predictions <- predict(model, newdata = test_data)
# Create and print confusion matrix
confusion_matrix <- confusionMatrix(predictions, test_data$Species)
print("Confusion Matrix:")
```

print(confusion_matrix)

31. (i)Write suitable R code to compute the mean, median ,mode of the following values c(90, 50, 70, 80, 70, 60, 20, 30, 80, 90, 20)

(ii) Write R code to find 2nd highest and 3rd Lowest value of above problem

Ans: values <- c(90, 50, 70, 80, 70, 60, 20, 30, 80, 90, 20)
mean_value <- mean(values)
cat("Mean:", mean_value, "\n")
median_value <- median(values)
cat("Median:", median_value, "\n")
mode_value <- as.numeric(names(sort(table(values), decreasing = TRUE)[1]))
cat("Mode:", mode_value, "\n")
second_highest <- unique(sort(values, decreasing = TRUE))[2]
cat("2nd Highest Value:", second_highest, "\n")
# Find the 3rd lowest value
third_lowest <- unique(sort(values))[3]
cat("3rd Lowest Value:", third_lowest, "\n").

32. Explore the airquality dataset. It contains daily air quality measurements from New York during a period of five months:

• Ozone: mean ozone concentration (ppb), • Solar.R: solar radiation (Langley),

• Wind: average wind speed (mph), • Temp: maximum daily temperature in degrees Fahrenheit,

• Month: numeric month (May=5, June=6, and so on),• Day: numeric day of the month (1-31).

i. Compute the mean temperature(don't use build in function) ii.Extract the first five rows from airquality.

iii.Extract all columns from airquality except Temp and Wind iv.Which was the coldest day during the period?

v.How many days was the wind speed greater than 17 mph?

Ans: data("airquality")
mean_tem <- sum(airquality$Temp)/length(airquality$Temp)
cat("\nMean of Temperature : ", mean_tem, "\n")
cat("\nFirst five rows of airquality dataset: \n")
print(head(airquality, n = 5))
selected_columns <- airquality[, !(names(airquality) %in% c("Temp", "Wind"))]
cat("\nColumns from airquality dataset except Temp and Wind: \n")
print(selected_columns)
coldest_day_row <- airquality[which.min(airquality$Temp), ]
cat("\nInformation about the coldest day: \n")
print(coldest_day_row)
days_greater_than_17mph <- sum(airquality$Wind > 17)
cat("\nNumber of days with wind speed greater than 17 mph:", days_greater_than_17mph, "\n")

33. (i)Get the Summary Statistics of air quality dataset

(ii)Melt airquality data set and display as a long – format data?

(iii)Melt airquality data and specify month and day to be "ID variables"?

(iv)Cast the molten airquality data set with respect to month and date features

(v) Use cast function appropriately and compute the average of Ozone, Solar.R , Wind and temperature per month?

Ans:
```
library("dplyr")
library("reshape2")
data("airquality")
print(summary(airquality))
melted_data <- melt(airquality, id.vars = c("Month", "Day"), variable.name = "Measurement")
cat("Melted airquality dataset: \n")
print(head(melted_data))
melted_data <- melt(airquality, id.vars = c("Month", "Day"))
cat("Melted airquality dataset with Month and Day as ID variables: \n")
print(head(melted_data))
casted_data <- dcast(melted_data, Month + Day ~ variable)
cat("Casted airquality data with Month and Day as features:\n")
print(casted_data)
melted_data <- melt(airquality, id.vars = c("Month"), measure.vars = c("Ozone", "Solar.R", "Wind", "Temp"))
averages_per_month <- dcast(melted_data, Month ~ variable, mean)
cat("Average values of Ozone, Solar.R, Wind, and Temp per month:\n")
print(averages_per_month)
```

34.(i) Find any missing values(na) in features and drop the missing values if its less than 10% else replace that with mean of that feature.

(ii) Apply a linear regression algorithm using Least Squares Method on "Ozone" and "Solar.R"

(iii)Plot Scatter plot between Ozone and Solar and add regression line created by above model

Ans:
```
library("dplyr")
library("reshape2")
data("airquality")
print(summary(airquality))
melted_data <- melt(airquality, id.vars = c("Month", "Day"), variable.name =
"Measurement")
cat("Melted airquality dataset: \n")
print(head(melted_data))
melted_data <- melt(airquality, id.vars = c("Month", "Day"))
cat("Melted airquality dataset with Month and Day as ID variables: \n")
print(head(melted_data))
casted_data <- dcast(melted_data, Month + Day ~ variable)
cat("Casted airquality data with Month and Day as features:\n")
print(casted_data)
melted_data <- melt(airquality, id.vars = c("Month"), measure.vars = c("Ozone", "Solar.R",
"Wind", "Temp"))
averages_per_month <- dcast(melted_data, Month ~ variable, mean)
cat("Average values of Ozone, Solar.R, Wind, and Temp per month:\n")
print(averages_per_month)
```

35. Load dataset named ChickWeight,

( i).Order the data frame, in ascending order by feature name "weight" grouped by feature "diet" and Extract the last 6 records from order data frame.

(ii).a Perform melting function based on “Chick", "Time", "Diet"   features as ID variables
b. Perform cast function to display the mean value of weight grouped by Diet
c. Perform cast function to display the mode of weight grouped by Diet

Ans:
```
library("reshape2")

library("dplyr")

library("modeest")

data("ChickWeight")

ordered_data <- ChickWeight[order(ChickWeight$weight), ]

last_6_records <- tail(ordered_data, 6)

cat("Last 6 records from the ordered data frame:\n")

print(last_6_records)

melted_data <- melt(ChickWeight, id.vars = c("Chick", "Time", "Diet"))

cat("Melted ChickWeight data:\n")

print(head(melted_data))

mean_weight_by_diet <- dcast(ChickWeight, Diet ~ ., value.var = "weight", fun.aggregate = mean)

cat("Mean weight values grouped by Diet:\n")

print(mean_weight_by_diet)

mode_func <- function(x) {

 tbl <- table(x)

 mode_value <- as.numeric(names(tbl[tbl == max(tbl)]))

 return(mode_value)

}

# Calculate the mode of weight grouped by Diet using dplyr and pivot_wider

mode_weight_by_diet <- ChickWeight %>%

 group_by(Diet) %>%

 summarize(ModeWeight = mode_func(weight)) %>%

 pivot_wider(names_from = Diet, values_from = ModeWeight)

# Print the mode weight values grouped by Diet

print("Mode weight values grouped by Diet:")

print(mode_weight_by_diet)
```

36. a. Create Box plot for “weight” grouped by “Diet”
b. Create a Histogram for “weight” features belong to Diet- 1 category
c. Create Scatter plot for “ weight” vs “Time” grouped by Diet

Ans:
```
data("ChickWeight")
# Create a Box plot for weight grouped by Diet
boxplot(weight ~ Diet, data = ChickWeight,
        main = "Box Plot of Weight Grouped by Diet",
        xlab = "Diet", ylab = "Weight")
hist(ChickWeight$weight[ChickWeight$Diet == 1],
     main = "Histogram of Weight in Diet-1",
     xlab = "Weight", ylab = "Frequency")
plot(ChickWeight$Time, ChickWeight$weight,
     col = as.numeric(ChickWeight$Diet),
     xlab = "Time", ylab = "Weight",
     main = "Scatter Plot of Weight vs Time",
     pch = as.numeric(ChickWeight$Diet))
legend("topright", legend = levels(ChickWeight$Diet),
       col = unique(as.numeric(ChickWeight$Diet)),
       pch = unique(as.numeric(ChickWeight$Diet)),
       title = "Diet")
```

37. a. Create multi regression model to find a weight of the chicken , by “Time” and “Diet” as as predictor variables
    b. Predict weight for Time=10 and Diet=1
    c. Find the error in model for same

Ans: data("ChickWeight")

ChickWeight$Diet <- as.factor(ChickWeight$Diet)

model <- lm(weight ~ Time + Diet, data = ChickWeight)

print(summary(model))

new_data <- data.frame(Time = 10, Diet = factor(1))

predicted_weight <- predict(model, newdata = new_data)

cat("\nPredicted weight for Time=10 and Diet=1 : ", predicted_weight)

actual_weight <- ChickWeight$weight[ChickWeight$Time == 10 & ChickWeight$Diet == 1]

prediction_error <- actual_weight - predicted_weight

cat("\nPrediction error for Time=10 and Diet=1: \n")

print(prediction_error)

op:
```
## Predicted weight for Time=10 and Diet=1: 258.3276
## Prediction error for Time=10 and Diet=1:[1] 9.472365
```

38. .For this exercise, use the (built-in) dataset Titanic.
    a. Draw a Bar chart to show details of “Survived” on the Titanic based on passenger Class
    b. Modify the above plot based on gender of people who survived
    c. Draw histogram plot to show distribution of feature “Age”

Ans: data("Titanic")

survival_counts <- apply(Titanic, c("Class", "Survived"), sum)

barplot(survival_counts, beside = TRUE,

        col = c("lightblue", "lightgreen"),

        legend.text = TRUE,

        args.legend = list(x = "topright"),

```
        main = "Survived on Titanic by Passenger Class",

        xlab = "Passenger Class", ylab = "Count")

survival_counts <- apply(Titanic, c(2, 4, 1), sum)

survival_matrix <- as.matrix(survival_counts)

barplot(survival_matrix, beside = TRUE, col = c("lightblue", "lightgreen"),

        main = "Survived on Titanic by Passenger Class and Gender",

        xlab = "Passenger Class", ylab = "Count",

        legend.text = TRUE, args.legend = list(x = "topright"),

        legend = colnames(survival_matrix))

titanic_df <- as.data.frame(Titanic)

age_data <- as.numeric(na.omit(titanic_df$Age))

hist(age_data, col = "lightblue", breaks = 20,

     main = "Distribution of Age in Titanic Dataset",

     xlab = "Age", ylab = "Frequency")
```

39. Explore the USArrests dataset, contains the number of arrests for murder, assault, and rape for each of the 50 states in 1973. It also contains the percentage of people in the state who live in an urban area.
(i) a. Explore the summary of Data set, like number of Features and its type. Find the number of records for each feature. Print the statistical feature of data b. Print the state which saw the largest total number of rape
c. Print the states with the max & min crime rates for murder
(ii).a. Find the correlation among the features
b. Print the states which have assault arrests more than median of the country
c. Print the states are in the bottom 25% of murder
(iii). a. Create a histogram and density plot of murder arrests by US stat
b. Create the plot that shows the relationship between murder arrest rate and proportion of the population that is urbanised by state. Then enrich the chart by adding assault arrest rates (by colouring the points from blue (low) to red (high)).
c. Draw a bar graph to show the murder rate for each of the 50 states .

Ans:
```
data("USArrests")
print(summary(USArrests))
num_features <- ncol(USArrests)
feature_types <- sapply(USArrests, class)
num_records <- nrow(USArrests)
cat("\nNumber of Records:", num_records, "\n\n")
stat_features <- sapply(USArrests, function(x) {
  c(Mean = mean(x), Median = median(x), SD = sd(x))
})
print(stat_features)
# Find the state with the largest total number of rape
max_rape_state <- rownames(USArrests)[which.max(USArrests$Rape)]
cat("\nState with Largest Total Rape:", max_rape_state, "\n")
# Find states with the max & min crime rates for murder
max_murder_state <- rownames(USArrests)[which.max(USArrests$Murder)]
min_murder_state <- rownames(USArrests)[which.min(USArrests$Murder)]
cat("\nState with Max Murder Rate:", max_murder_state, "\n")
cat("\nState with Min Murder Rate:", min_murder_state, "\n\n")
# Calculate the correlation matrix among the features
correlation_matrix <- cor(USArrests)
print(correlation_matrix)
# Calculate the median of assault arrests for the country
median_assault <- median(USArrests$Assault)
# Find states with assault arrests more than the median
states_above_median <- rownames(USArrests)[USArrests$Assault > median_assault]
cat("\n\nStates with Assault Arrests > Median: \n", states_above_median, "\n")
# Calculate the 25th percentile of murder arrests
percentile_25 <- quantile(USArrests$Murder, 0.25)
# Find states in the bottom 25% of murder arrests
states_bottom_25 <- rownames(USArrests)[USArrests$Murder < percentile_25]
cat("\n\nStates in Bottom 25% of Murder Arrests:\n", states_bottom_25, "\n")
hist(USArrests$Murder, col = "lightblue",
    main = "Histogram of Murder Arrests",
    xlab = "Murder Arrests", ylab = "Frequency")
plot(density(USArrests$Murder),
    main = "Density Plot of Murder Arrests",
    xlab = "Murder Arrests", ylab = "Density", col = "blue")
murder_arrests <- USArrests$Murder
urban_population <- USArrests$UrbanPop
assault_arrests <- USArrests$Assault
plot(urban_population, murder_arrests, main = "Murder Arrest Rate vs. Urban Population",
    xlab = "Urban Population (%)", ylab = "Murder Arrest Rate",
    col = heat.colors(20)[cut(assault_arrests, breaks = 20)],
    pch = 19)
legend("topright", legend = "Assault Arrest Rate",
      fill = heat.colors(20), title = "Assault Rate")
barplot(USArrests$Murder, names.arg = rownames(USArrests),
       col = "lightblue", main = "Murder Rates by State",
       xlab = "State", ylab = "Murder Rate")
```

40. 4. a. Create a data frame based on below table.

| Month | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Spends | 1000 | 4000 | 5000 | 4500 | 3000 | 4000 | 9000 | 11000 | 15000 | 12000 | 7000 | 3000 |
| Sales | 9914 | 40487 | 54324 | 50044 | 34719 | 42551 | 94871 | 118914 | 158484 | 131348 | 78504 | 36284 |

b. Create a regression model for that data frame table to show the amount of sales(Sales) based on the how much the company spends (Spends) in advertising c. Predict the Sales if Spend=13500

Ans:
```
data_table <- data.frame(
  Month = 1:12,
  Spends = c(1000, 4000, 5000, 4500, 3000, 4000, 9000, 11000, 15000, 12000, 7000, 3000),
  Sales = c(9914, 40487, 54324, 50044, 34719, 42551, 94871, 118914, 158484, 131348, 78504,
36284))
print(data_table)
model <- lm(Sales ~ Spends, data = data_table)
summary(model)
new_data <- data.frame(Spends = 13500)
```

```
predicted_sales <- predict(model, newdata = new_data)
print(predicted_sales)
```

```
op: Call: lm(formula = Sales ~ Spends, data = data_table)

Residuals: Min 1Q Median 3Q Max -15058.02 -4165.75 80.60 4026.43 10763.38

Coefficients: Estimate Std. Error t value Pr(>|t|)
(Intercept) 1225.7883 4033.3948 0.304 0.770
Spends 9.2288 0.3653 25.275 7.17e-09
```