BVV Sangha, Bagalkot

AMRUTA INSTITUTE OF ENGINEERING & MANAGEMENT SCIENCES

Approved by AICTE, New Delhi

Recognized by Government of Karnataka & Affiliated to VTU, Belagavi

AIEMS BENGALURU

DEPARTMENT OF ARTIFICIAL INTELLIGENCE & MACHINE LEARNING

III SEMESTER DATA ANALYTICS WITH R (BDS306C)

LAB MANUAL

Vision & Mission statements of AIEMS

VISION OF AIEMS

- To be an Institute of repute, imparting sustainable quality education, research and innovation in technical and Management fields to meet the socio-economic challenges

MISSION of AIEMS

- To bestow Outcome-based Quality Education to meet the industrial , societal and environmental challenges
- To establish Center of Excellence, collaborations with industries & academia to promote research culture among students and staff
- To inculcate innovation and entrepreneurial attitude among the students & faculty by providing incubation and consulting facilities.

VISION OF AI-ML DEPARTMENT

To create the young minds into an excellent AI and ML professionals with intense practical, research & managerial skills to meet technological and social needs of the Society.

MISSION OF AI-ML DEPARTMENT

1. To impart strong theoretical and practical exposure to meet global standard quality education
2. To provide state of art Infrastructure for young thinking minds with active research and teaching-learning environment.
3. To develop socially responsible human beings with professional ethics and high leadership qualities.

4. PROGRAM EDUCATIONAL OBJECTIVES (PEO'S)

PEO 1	Graduates will be able to poses a successful Software Engineering career.
PEO 2	Graduates will be active into research and able to pursue higher education
PEO 3	Graduates will be professionals with ethics, who acts as assets to the IT Industries with good human values and serve as social responsible citizens
PEO 4	Graduates shall have proficient Technical, managerial and communication skills to become global competent professionals.

PROGRAM SPECIFIC OUTCOMES (PSO'S)

PSO 1	Understand fundamentals of Artificial Intelligence and Machine language concepts with proficiency in programming Skills to develop efficient AI based software system.
PSO 2	Apply Artificial Intelligence and machine language concepts to design, develop and solve real world problems as per user requirements.
PSO 3	Integrate the Engineering skills and serve the industry and society equally well at the global standards

Edition: 2025- 26

Document Owner

The primary contact for questions regarding this document is:

Author(s): Dr. Sanjana Prasad

Department:

Artificial Intelligence and Machine learning

Contact email ids :

sanjana@aiems.edu.in

Experiment 1

Demonstrate the steps for installation of R and R Studio. Perform the following:

a) Assign different type of values to variables and display the type of variable. Assign different types such as Double, Integer, Logical, Complex and Character and understand the difference between each data type.

b) Demonstrate Arithmetic and Logical Operations with simple examples.

c) Demonstrate generation of sequences and creation of vectors.

d) Demonstrate Creation of Matrices.

e) Demonstrate the Creation of Matrices from Vectors using Binding Function.

f) Demonstrate element extraction from vectors, matrices and arrays.

CODE:

Steps for Installation of R and RStudio

Step 1: Install R

1. Go to **https://cran.r-project.org**
2. Select your operating system (Windows / Mac / Linux)
3. Download the latest version of **R**
4. Run the installer and complete installation

Step 2: Install RStudio

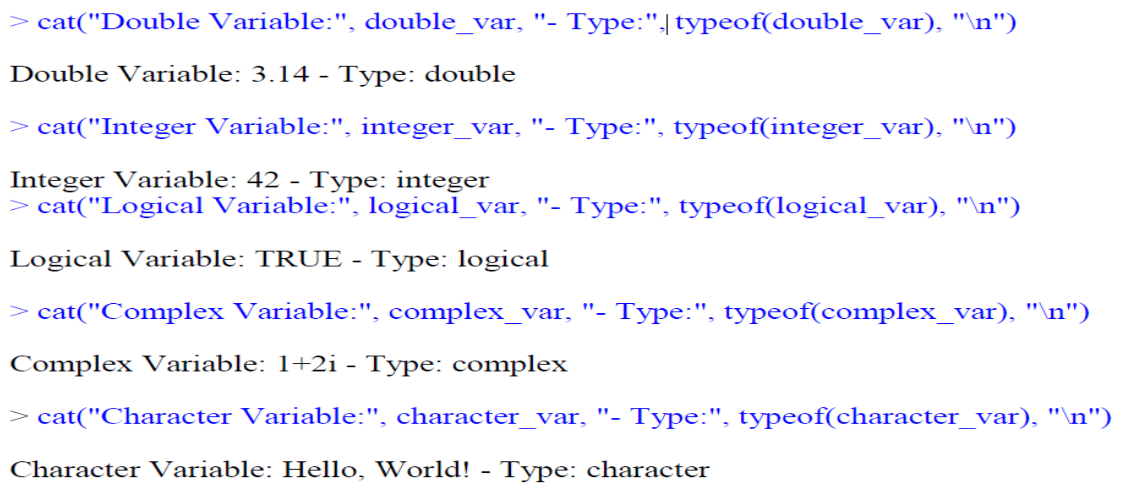
1. Go to **https://posit.co/download/rstudio/**
2. Download **RStudio Desktop (Free Version)**
3. Install RStudio after R is installed
4. Open RStudio → R Console will be ready

a) **Assign different type of values to variables and display the type of variable. Assign different types such as Double, Integer, Logical, Complex and Character and understand the difference between each data type.**

```
# Assigning different types of values to variables

double_var <- 3.14

integer_var <- 42L

logical_var <- TRUE
```

```
complex_var <- 1 + 2i

character_var <- "Hello, World!"

# Displaying the values and types of variables

cat("Double Variable:", double_var, "- Type:", typeof(double_var), "\n")

cat("Integer Variable:", integer_var, "- Type:", typeof(integer_var), "\n")

cat("Logical Variable:", logical_var, "- Type:", typeof(logical_var), "\n")

cat("Complex Variable:", complex_var, "- Type:", typeof(complex_var), "\n")

cat("Character Variable:", character_var, "- Type:", typeof(character_var), "\n")
```

Output:



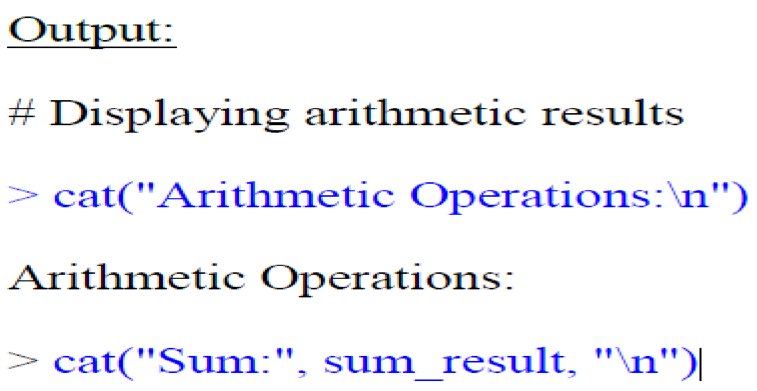
```
> cat("Double Variable:", double_var, "- Type:", typeof(double_var), "\n")
Double Variable: 3.14 - Type: double
> cat("Integer Variable:", integer_var, "- Type:", typeof(integer_var), "\n")
Integer Variable: 42 - Type: integer
> cat("Logical Variable:", logical_var, "- Type:", typeof(logical_var), "\n")
Logical Variable: TRUE - Type: logical
> cat("Complex Variable:", complex_var, "- Type:", typeof(complex_var), "\n")
Complex Variable: 1+2i - Type: complex
> cat("Character Variable:", character_var, "- Type:", typeof(character_var), "\n")
Character Variable: Hello, World! - Type: character
```

b) Demonstrate Arithmetic and Logical Operations with simple examples.

```
sum_result <- num1 + num2

difference_result <- num1 - num2

product_result <- num1 * num2

quotient_result <- num1 / num2

remainder_result <- num1 %% num2
```

# Displaying arithmetic results

```
cat("Arithmetic Operations:\n")
cat("Sum:", sum_result, "\n")
cat("Difference:", difference_result, "\n")
cat("Product:", product_result, "\n")
cat("Quotient:", quotient_result, "\n")
cat("Remainder:", remainder_result, "\n\n")
# Logical operations
logical_var1 <- TRUE
logical_var2 <- FALSE
and_result <- logical_var1 & logical_var2
or_result <- logical_var1 | logical_var2
not_result <- !logical_var1
# Displaying logical results
cat("Logical Operations:\n")
cat("AND:", and_result, "\n")
cat("OR:", or_result, "\n")
cat("NOT:", not_result, "\n")
```



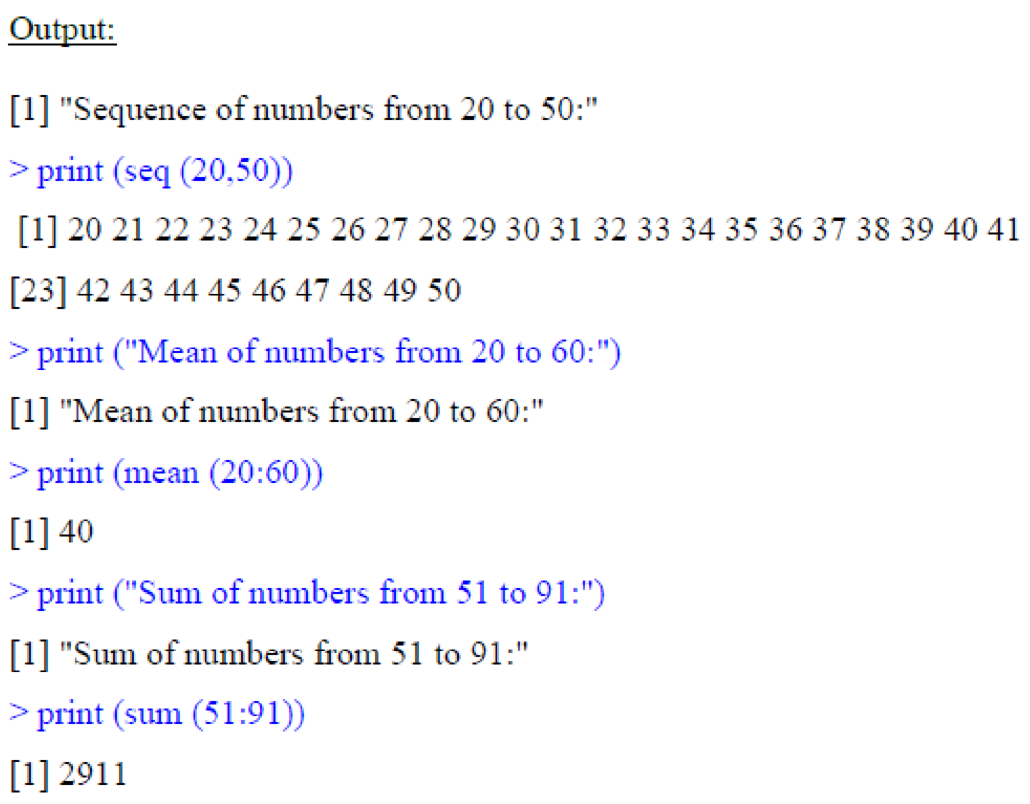
Output:

```
# Displaying arithmetic results
> cat("Arithmetic Operations:\n")
Arithmetic Operations:
> cat("Sum:", sum_result, "\n")
```

```
Sum: 15
> cat("Difference:", difference_result, "\n")
Difference: 5
> cat("Product:", product_result, "\n")
Product: 50
> cat("Quotient:", quotient_result, "\n")
Quotient: 2
> cat("Remainder:", remainder_result, "\n\n")
Remainder: 0

# Displaying logical results
> cat("Logical Operations:\n")
Logical Operations:
> cat("AND:", and_result, "\n")
AND: FALSE
> cat("OR:", or_result, "\n")
OR: TRUE
> cat("NOT:", not_result, "\n")
NOT: FALSE
```

c)Demonstrate generation of sequences and creation of vectors.

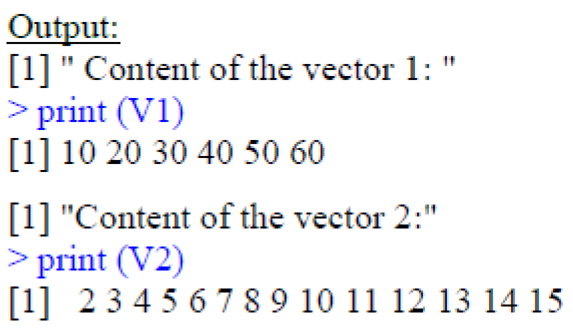
```
print ("Sequence of numbers from 20 to 50:")
print (seq (20,50))
print ("Mean of numbers from 20 to 60:")
print (mean (20:60))
print ("Sum of numbers from 51 to 91:")
print (sum (51:91))
```



Output:

```
[1] "Sequence of numbers from 20 to 50:"
> print (seq (20,50))
 [1] 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41
[23] 42 43 44 45 46 47 48 49 50
> print ("Mean of numbers from 20 to 60:")
[1] "Mean of numbers from 20 to 60:"
> print (mean (20:60))
[1] 40
> print ("Sum of numbers from 51 to 91:")
[1] "Sum of numbers from 51 to 91:"
> print (sum (51:91))
[1] 2911
```

```
#Example 1
V1 = c(10, 20, 30, 40, 50, 60)
print('Content of the vector 1:')
print(V1)
#Example 2
V2 = seq(2,15)
print ("Content of the vector 2:")
print (V2)
#Example 3
V3= sample (-50:50, 10, replace=TRUE)
print ("Content of the vector 3:")
print (V3)
```



```
Output:
[1] " Content of the vector 1: "
> print (V1)
[1] 10 20 30 40 50 60
[1] "Content of the vector 2:"
> print (V2)
[1]   2 3 4 5 6 7 8 9 10 11 12 13 14 15
```

d) Demonstrate Creation of Matrices.

#Example 1

```
m1 = matrix(1:20, nrow=5, ncol=4)

print("5 × 4 matrix:")

print(m1)
```

#Example 2

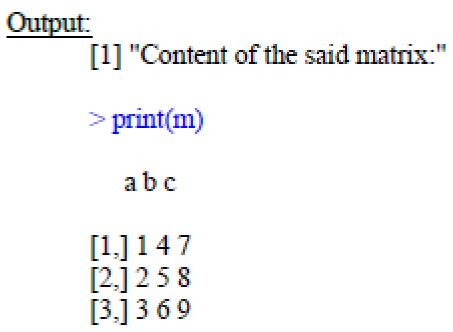
```
cells = c(1,3,5,7,8,9,11,12,14)

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)
```

```
Output:
[1] "5 × 4 matrix:"

> print(m1)

     [,1] [,2] [,3] [,4]

[1,]    1    6   11   16
[2,]    2    7   12   17
[3,]    3    8   13   18
[4,]    4    9   14   19
[5,]    5   10   15   20


> print("3 × 3 matrix with labels, filled by rows: ")

[1] "3 × 3 matrix with labels, filled by rows: "

> print(m2)

     Col1 Col2 Col3

Row1    1    3    5
Row2    7    8    9
Row3   11   12   14

> print("3 × 3 matrix with labels, filled by columns: ")

[1] "3 × 3 matrix with labels, filled by columns:"
> print(m3)

     Col1 Col2 Col3

Row1    1    7   11
Row2    3    8   12



Row3    5    9   14
```

e) Demonstrate the Creation of Matrices from Vectors using Binding Function.

```
a<-c(1,2,3)

b<-c(4,5,6)

c<-c(7,8,9)

m<-cbind(a,b,c)
```

```
print("Content of the matrix:")

print(m)
```

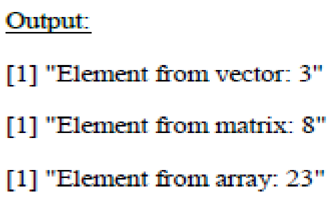


Output:

```
[1] "Content of the said matrix:"

> print(m)

    a b c

[1,] 1 4 7
[2,] 2 5 8
[3,] 3 6 9
```

f) Demonstrate element extraction from vectors, matrices and arrays.

```
# Creating a vector
my_vector <- c(1, 2, 3, 4, 5)
# Extracting the third element from the vector
element_from_vector <- my_vector[3]
print(paste("Element from vector:", element_from_vector))

# Creating a matrix
my_matrix <- matrix(1:9, nrow = 3)
# Extracting the element in the second row and third column of the matrix
element_from_matrix <- my_matrix[2, 3]
print(paste("Element from matrix:", element_from_matrix))
# Creating a 3D array
my_array <- array(1:27, dim = c(3, 3, 3))
# Extracting the element in the second row, second column, and third depth of the array
element_from_array <- my_array[2, 2, 3]
print(paste("Element from array:", element_from_array))
```



Output:

```
[1] "Element from vector: 3"

[1] "Element from matrix: 8"

[1] "Element from array: 23"
```

Program 2

AIM: Assess the Financial Statement of an Organization being supplied with 2 vectors of data: Monthly Revenue and Monthly Expenses for the Financial Year. You can create your own sample data vector for this experiment) Calculate the following financial metrics:

a. Profit for each month.
b. Profit after tax for each month (Tax Rate is 30%).
c. Profit margin for each month equals to profit after tax divided by revenue.
d. Good Months – where the profit after tax was greater than the mean for the year.
e. Bad Months – where the profit after tax was less than the mean for the year.
f. The best month – where the profit after tax was max for the year.
g. The worst month – where the profit after tax was min for the year.

Note:

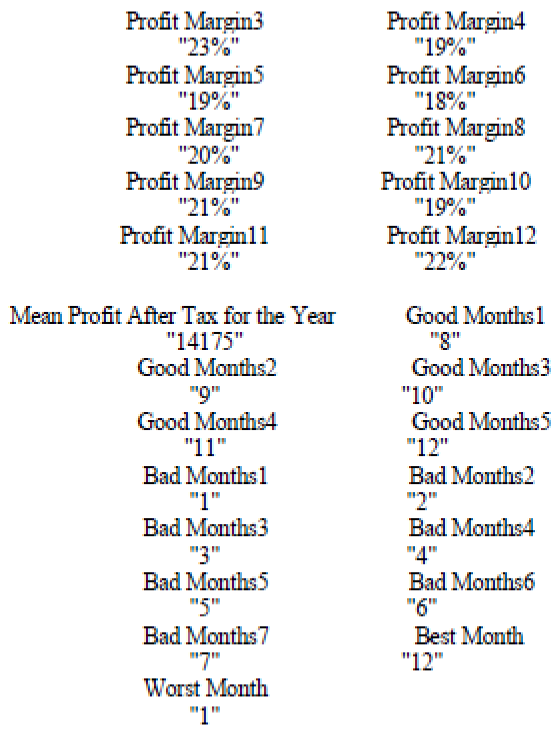
a. All Results need to be presented as vectors
b. Results for Dollar values need to be calculated with $0.01 precision, but need to be presented in Units of $1000 (i.e 1k) with no decimal points
c. Results for the profit margin ratio need to be presented in units of % with no decimal point.
d. It is okay for tax to be negative for any given month (deferred tax asset)
e. Generate CSV file for the data.

```
# Sample data for Monthly Revenue and Monthly Expenses

monthly_revenue <- c(50000, 55000, 60000, 58000, 62000, 65000, 70000, 75000, 78000,
80000, 85000, 90000)

monthly_expenses <- c(35000, 38000, 40000, 42000, 45000, 48000, 50000, 52000, 55000,
58000, 60000, 62000)

# Calculate Profit for each month

profit <- monthly_revenue - monthly_expenses

# Tax Rate

tax_rate <- 0.3

# Calculate Profit After Tax for each month

profit_after_tax <- profit * (1 - tax_rate)

# Calculate Profit Margin for each month

profit_margin <- round(profit_after_tax / monthly_revenue * 100, 0)

# Calculate Mean Profit After Tax for the year
```

```
mean_profit_after_tax <- round(mean(profit_after_tax), 2)
# Identify Good Months, Bad Months, Best Month, and Worst Month
good_months <- which(profit_after_tax > mean_profit_after_tax)
bad_months <- which(profit_after_tax < mean_profit_after_tax)
best_month <- which.max(profit_after_tax)
worst_month <- which.min(profit_after_tax)
# Displaying the results as vectors
results_vector <- c("Monthly Revenue" = monthly_revenue / 1000,
"Monthly Expenses" = monthly_expenses / 1000,
"Profit" = round(profit / 1000, 2),
"Profit After Tax" = round(profit_after_tax / 1000, 2),
"Profit Margin" = paste0(profit_margin, "%"),
"Mean Profit After Tax for the Year" = mean_profit_after_tax,
"Good Months" = good_months,
"Bad Months" = bad_months,
"Best Month" = best_month,
"Worst Month" = worst_month)
# Displaying the results vector
print(results_vector)
# Generate CSV file
results_data <- data.frame(Month = 1:12,
Revenue = monthly_revenue,
Expenses = monthly_expenses,
Profit = profit,
Profit_After_Tax = profit_after_tax,
Profit_Margin = profit_margin)
write.csv(results_data, "financial_results.csv", row.names = FALSE)
```

Output:

```
               Monthly Revenue1           Monthly Revenue2
                    "50"                    "55"
               Monthly Revenue3           Monthly Revenue4
                    "60"                    "58"
               Monthly Revenue5           Monthly Revenue6
                    "62"                    "65"
               Monthly Revenue7           Monthly Revenue8


         "70"                 "75"
   Monthly Revenue9      Monthly Revenue10
         "78"                 "80"
   Monthly Revenue11     Monthly Revenue12
         "85"                 "90"
   Monthly Expenses1     Monthly Expenses2
         "35"                 "38"
   Monthly Expenses3     Monthly Expenses4
         "40"                 "42"
   Monthly Expenses5     Monthly Expenses6
         "45"                 "48"
   Monthly Expenses7     Monthly Expenses8
         "50"                 "52"
   Monthly Expenses9     Monthly Expenses10
         "55"                 "58"
   Monthly Expenses11    Monthly Expenses12
         "60"                 "62"

        Profit1              Profit2
         "15"                 "17"
        Profit3              Profit4
         "20"                 "16"
        Profit5              Profit6
         "17"                 "17"
        Profit7              Profit8
         "20"                 "23"
        Profit9              Profit10
         "23"                 "22"
        Profit11             Profit12
         "25"                 "28"

   Profit After Tax1     Profit After Tax2
        "10.5"               "11.9"
   Profit After Tax3     Profit After Tax4
         "14"                "11.2"
   Profit After Tax5     Profit After Tax6
        "11.9"               "11.9"
   Profit After Tax7     Profit After Tax8
         "14"                "16.1"
   Profit After Tax9     Profit After Tax10
        "16.1"               "15.4"
   Profit After Tax11    Profit After Tax12
        "17.5"               "19.6"

     Profit Margin1        Profit Margin2
         "21%"                "22%"
```



```
     Profit Margin3          Profit Margin4
         "23%"                   "19%"
     Profit Margin5          Profit Margin6
         "19%"                   "18%"
     Profit Margin7          Profit Margin8
         "20%"                   "21%"
     Profit Margin9          Profit Margin10
         "21%"                   "19%"
     Profit Margin11         Profit Margin12
         "21%"                   "22%"

Mean Profit After Tax for the Year    Good Months1
           "14175"                        "8"
      Good Months2                    Good Months3
          "9"                        "10"
      Good Months4                    Good Months5
          "11"                        "12"
      Bad Months1                     Bad Months2
          "1"                         "2"
      Bad Months3                     Bad Months4
          "3"                         "4"
      Bad Months5                     Bad Months6
          "5"                         "6"
      Bad Months7                      Best Month
          "7"                        "12"
      Worst Month
          "1"
```

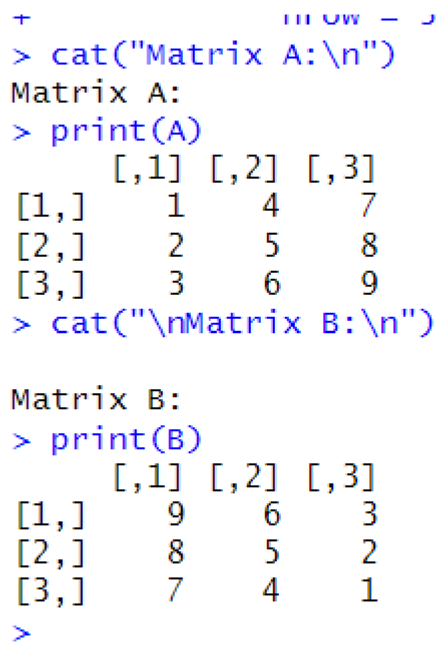
Program 3

Develop a program to create two 3 X 3 matrices A and B and perform the following operations: a) Transpose of the matrix b) Addition c) Subtraction d) Multiplication

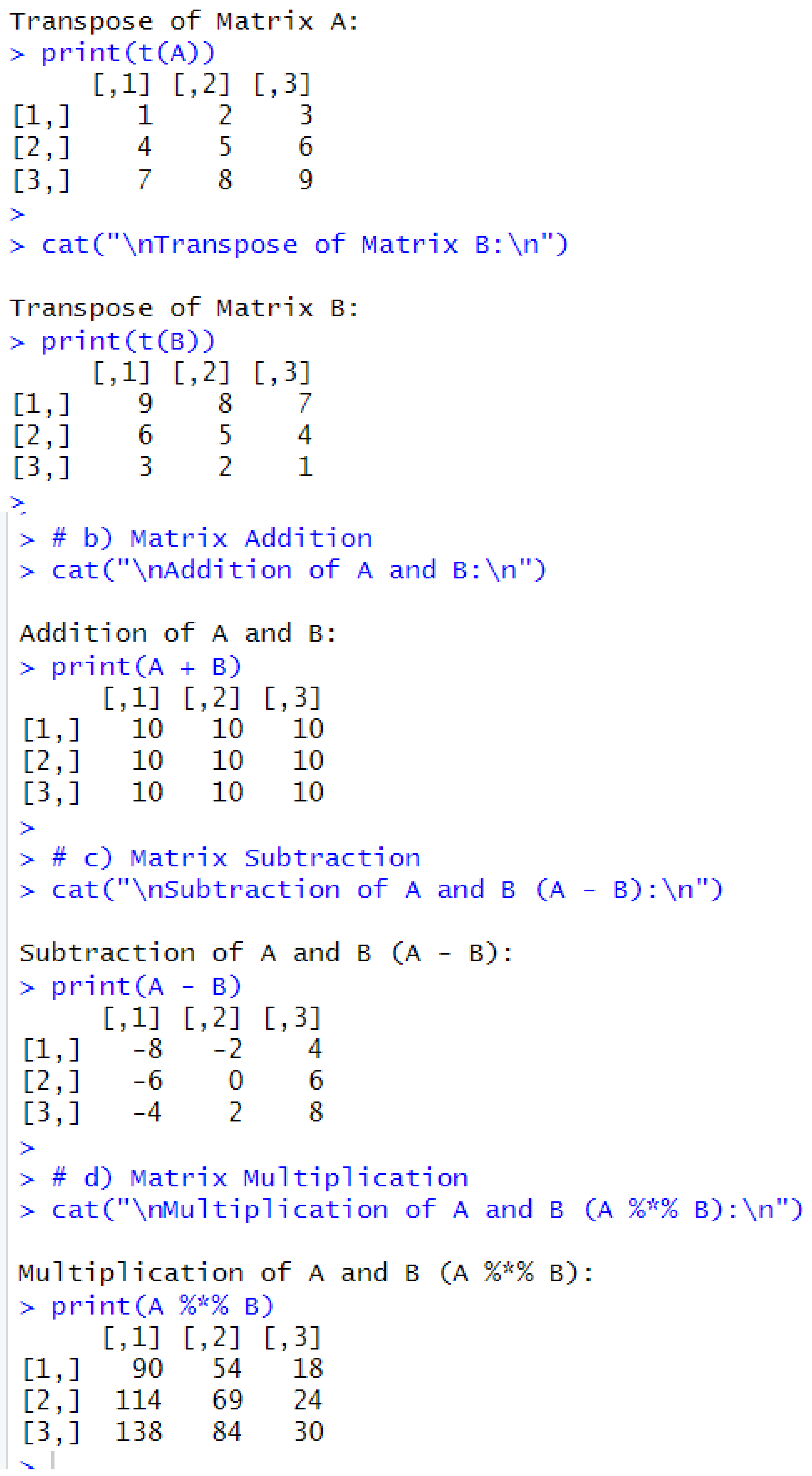
CODE

```
# Creation of two 3 X 3 matrices A and B
A <- matrix(c(1, 2, 3,
              4, 5, 6,
              7, 8, 9),
            nrow = 3, ncol = 3)
B <- matrix(c(9, 8, 7,
              6, 5, 4,
              3, 2, 1),
            nrow = 3, ncol = 3)
cat("Matrix A:\n")
print(A)
cat("\nMatrix B:\n")
print(B)

#a) Transpose of matrices
cat("\nTranspose of Matrix A:\n")
print(t(A))
```

```
cat("\nTranspose of Matrix B:\n")
print(t(B))

# b) Matrix Addition
cat("\nAddition of A and B:\n")
print(A + B)

# c) Matrix Subtraction
cat("\nSubtraction of A and B (A - B):\n")
print(A - B)

# d) Matrix Multiplication
cat("\nMultiplication of A and B (A %*% B):\n")
print(A %*% B)
```



```
+              nrow = 3, ncol =
> cat("Matrix A:\n")
Matrix A:
> print(A)
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
> cat("\nMatrix B:\n")

Matrix B:
> print(B)
     [,1] [,2] [,3]
[1,]    9    6    3
[2,]    8    5    2
[3,]    7    4    1
> 
```

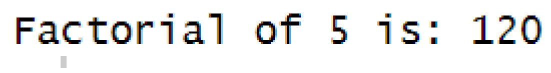


```
Transpose of Matrix A:
> print(t(A))
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
[3,]    7    8    9
>
> cat("\nTranspose of Matrix B:\n")

Transpose of Matrix B:
> print(t(B))
     [,1] [,2] [,3]
[1,]    9    8    7
[2,]    6    5    4
[3,]    3    2    1
>
> # b) Matrix Addition
> cat("\nAddition of A and B:\n")

Addition of A and B:
> print(A + B)
     [,1] [,2] [,3]
[1,]   10   10   10
[2,]   10   10   10
[3,]   10   10   10
>
> # c) Matrix Subtraction
> cat("\nSubtraction of A and B (A - B):\n")

Subtraction of A and B (A - B):
> print(A - B)
     [,1] [,2] [,3]
[1,]   -8   -2    4
[2,]   -6    0    6
[3,]   -4    2    8
>
> # d) Matrix Multiplication
> cat("\nMultiplication of A and B (A %*% B):\n")

Multiplication of A and B (A %*% B):
> print(A %*% B)
     [,1] [,2] [,3]
[1,]   90   54   18
[2,]  114   69   24
[3,]  138   84   30
>
```

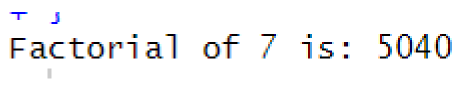
Experiment 4 : Develop a program to find the factorial of given number using recursive function calls.

```
# --------------------------------------------------
# Factorial using recursion with proper input validation
# --------------------------------------------------

factorial_recursive <- function(n) {
  if (n == 0 || n == 1) {
    return(1)
  } else {
    return(n * factorial_recursive(n - 1))
  }
}

# Read user input
num <- as.integer(readline(prompt = "Enter a number: "))

# Input validation
if (is.na(num)) {
  cat("Invalid input. Please enter a valid integer.")
} else if (num < 0) {
  cat("Factorial is not defined for negative numbers.")
} else {
  result <- factorial_recursive(num)
  cat("Factorial of", num, "is:", result)
}
```

Output :



```
Factorial of 5 is: 120
```

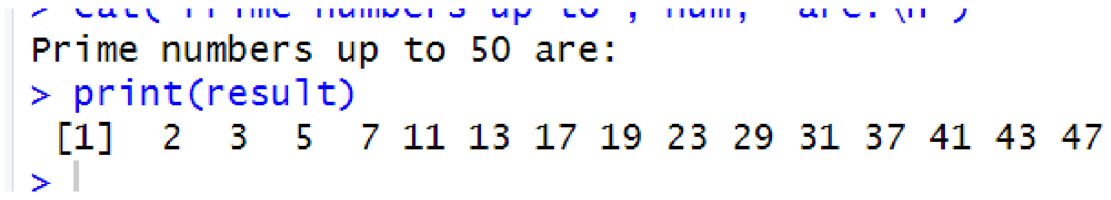


```
Factorial of 7 is: 5040
```

Program 5
Develop an R Program using functions to find all the prime numbers up to a specified number by the method of Sieve of Eratosthenes.

```
# Function to find prime numbers using Sieve of Eratosthenes

sieve_eratosthenes <- function(n) {

  # Check for valid input
  if (n < 2) {
    return("No prime numbers exist below 2")
  }

  # Create a logical vector of TRUE values
  prime <- rep(TRUE, n)

  # 1 is not a prime number
  prime[1] <- FALSE

  # Apply Sieve of Eratosthenes
  for (i in 2:floor(sqrt(n))) {
    if (prime[i]) {
      for (j in seq(i * i, n, by = i)) {
        prime[j] <- FALSE
      }
    }
  }

  # Return prime numbers
  return(which(prime))
}

# Specify the number
num <- 50

# Call the function
result <- sieve_eratosthenes(num)

# Display output
cat("Prime numbers up to", num, "are:\n")
print(result)
```

OUTPUT:



```
Prime numbers up to 50 are:
> print(result)
 [1]  2  3  5  7 11 13 17 19 23 29 31 37 41 43 47
> 
```

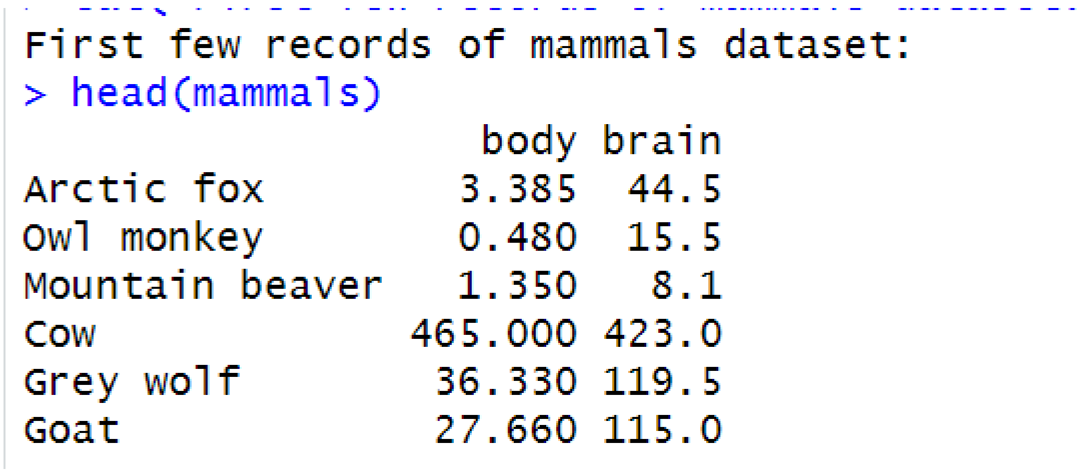
Program 6
The built-in data set mammals containing data on body weight versus brain weight.
Develop R commands to:
a) Find the Pearson and Spearman correlation coefficients. Are they similar?
b) Plot the data using the plot command.
c) Plot the logarithm (log) of each variable and see if that makes a difference.

```
# Load the required package and dataset

library(MASS)       # mammals dataset is available in MASS
data(mammals)

# Display first few rows
cat("First few records of mammals dataset:\n")
head(mammals)

# a) Pearson and Spearman Correlation Coefficients

# Pearson correlation
pearson_corr <- cor(mammals$body, mammals$brain, method = "pearson")

# Spearman correlation
spearman_corr <- cor(mammals$body, mammals$brain, method = "spearman")

cat("\nPearson Correlation Coefficient:", pearson_corr, "\n")
cat("Spearman Correlation Coefficient:", spearman_corr, "\n")

# b) Plot body weight vs brain weight# ------------------------------------------------

plot(mammals$body, mammals$brain,
     main = "Body Weight vs Brain Weight",
     xlab = "Body Weight (kg)",
     ylab = "Brain Weight (g)",
     pch = 19,
     col = "blue")
```

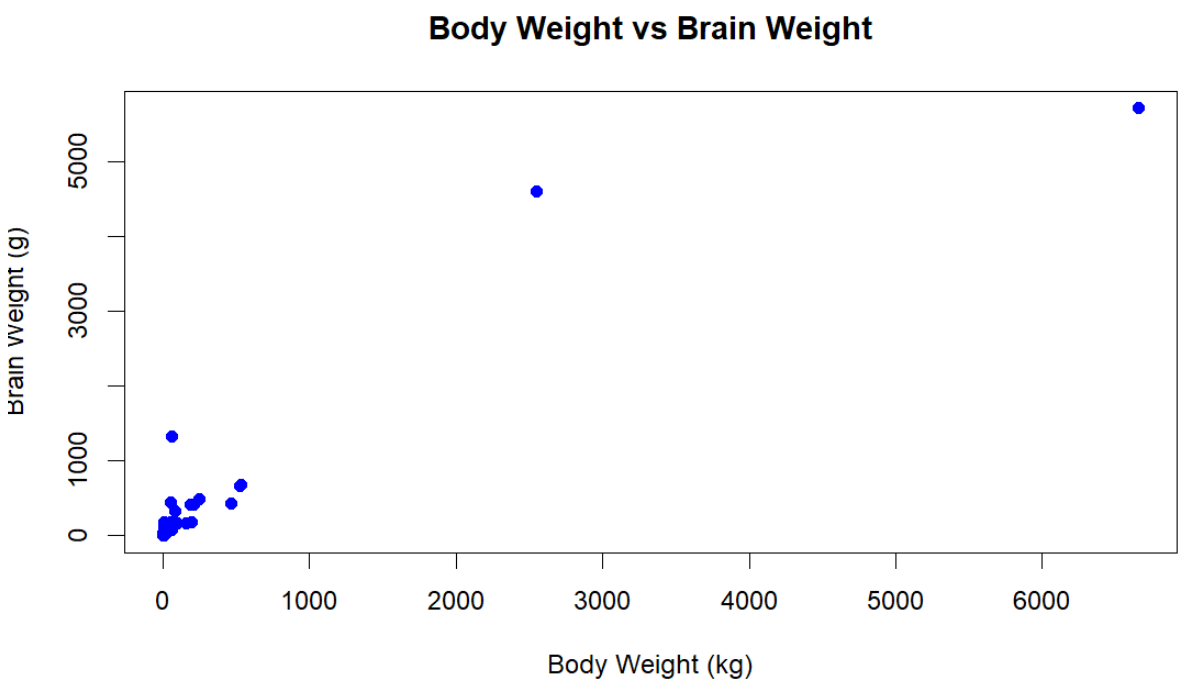
# c) Plot log-transformed variables

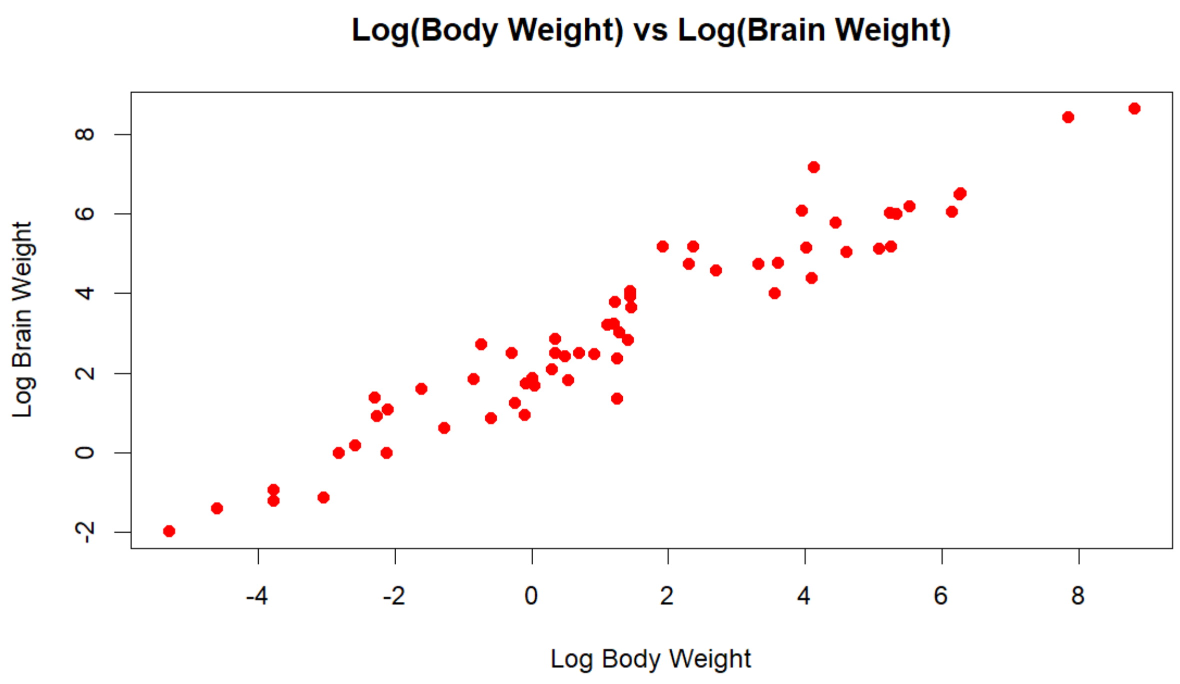
```
plot(log(mammals$body), log(mammals$brain),
    main = "Log(Body Weight) vs Log(Brain Weight)",
    xlab = "Log Body Weight",
    ylab = "Log Brain Weight",
    pch = 19,
    col = "red")
```

OUTPUT :



```
First few records of mammals dataset:
> head(mammals)
                     body brain
Arctic fox          3.385  44.5
Owl monkey          0.480  15.5
Mountain beaver     1.350   8.1
Cow               465.000 423.0
Grey wolf          36.330 119.5
Goat               27.660 115.0
```





Program 7
Develop R program to create a Data Frame with following details and do the following operations.

itemCode	itemCategory	itemPrice
1001	Electronics	700
1002	Desktop Supplies	300
1003	Office Supplies	350
1004	USB	400
1005	CD Drive	800

a) Subset the Data frame and display the details of only those items whose price is greater than or equal to 350.
b) Subset the Data frame and display only the items where the category is either “Office Supplies” or “Desktop Supplies”.
c) Create another Data Frame called “item-details” with three different fields itemCode, ItemQtyonHand and ItemReorderLvl and merge the two frames.

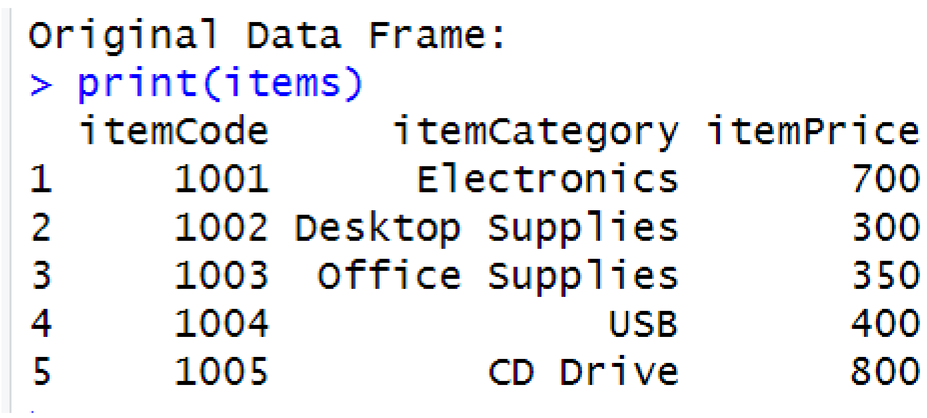
CODE :

```
# Create the main Data Frame

items <- data.frame(
  itemCode = c(1001, 1002, 1003, 1004, 1005),
  itemCategory = c("Electronics", "Desktop Supplies",
                   "Office Supplies", "USB", "CD Drive"),
  itemPrice = c(700, 300, 350, 400, 800)
)

cat("Original Data Frame:\n")
print(items)

# a) Subset items with price >= 350

price_subset <- subset(items, itemPrice >= 350)

cat("\nItems with price greater than or equal to 350:\n")
print(price_subset)

# b) Subset items by category
#    Office Supplies or Desktop Supplies

category_subset <- subset(items,
                          itemCategory == "Office Supplies" |
                            itemCategory == "Desktop Supplies")

cat("\nItems with category Office Supplies or Desktop Supplies:\n")
print(category_subset)

# c) Create another Data Frame and merge

item_details <- data.frame(
  itemCode = c(1001, 1002, 1003, 1004, 1005),
  ItemQtyonHand = c(50, 40, 60, 30, 20),
  ItemReorderLvl = c(10, 15, 20, 10, 5)
)

cat("\nItem Details Data Frame:\n")
print(item_details)
```

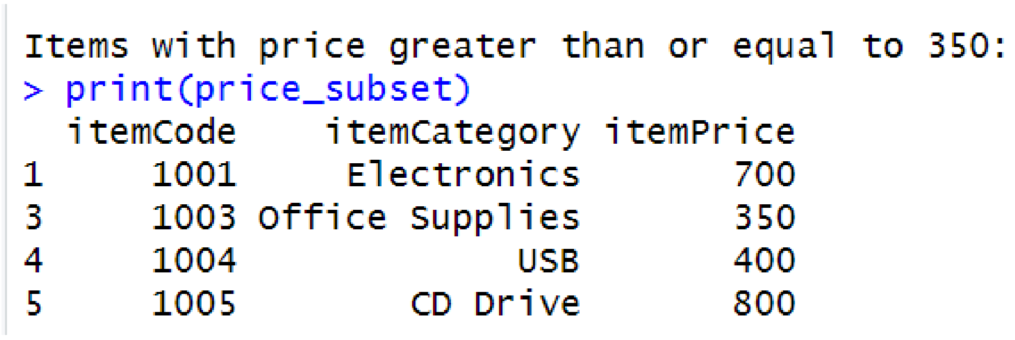
# Merge the two data frames
merged_data <- merge(items, item_details, by = "itemCode")

cat("\nMerged Data Frame:\n")
print(merged_data)

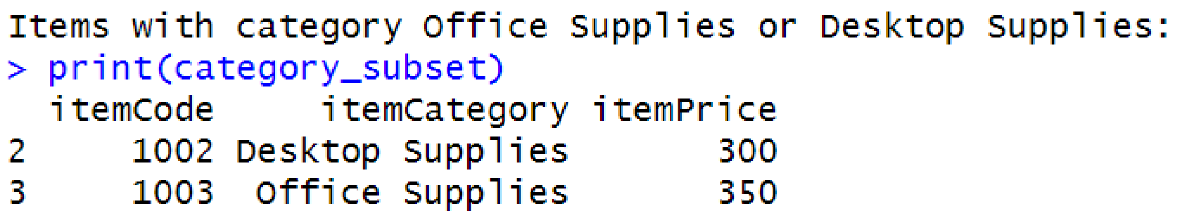
OUTPUT:



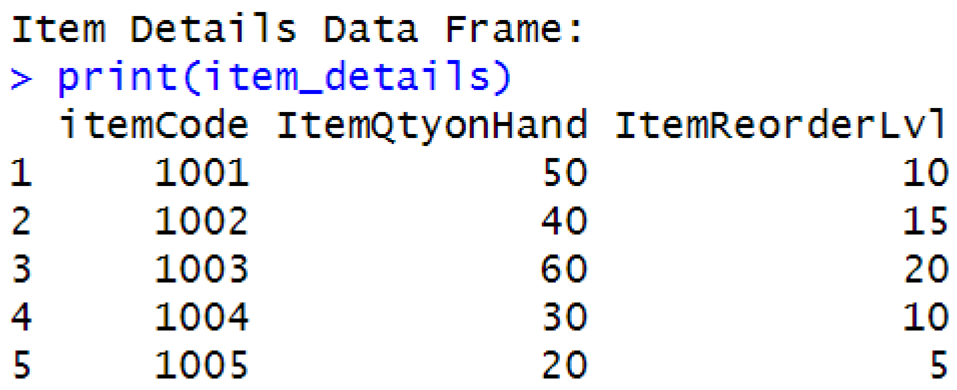
```
Original Data Frame:
> print(items)
  itemCode     itemCategory itemPrice
1     1001      Electronics       700
2     1002 Desktop Supplies       300
3     1003  Office Supplies       350
4     1004              USB       400
5     1005         CD Drive       800
```



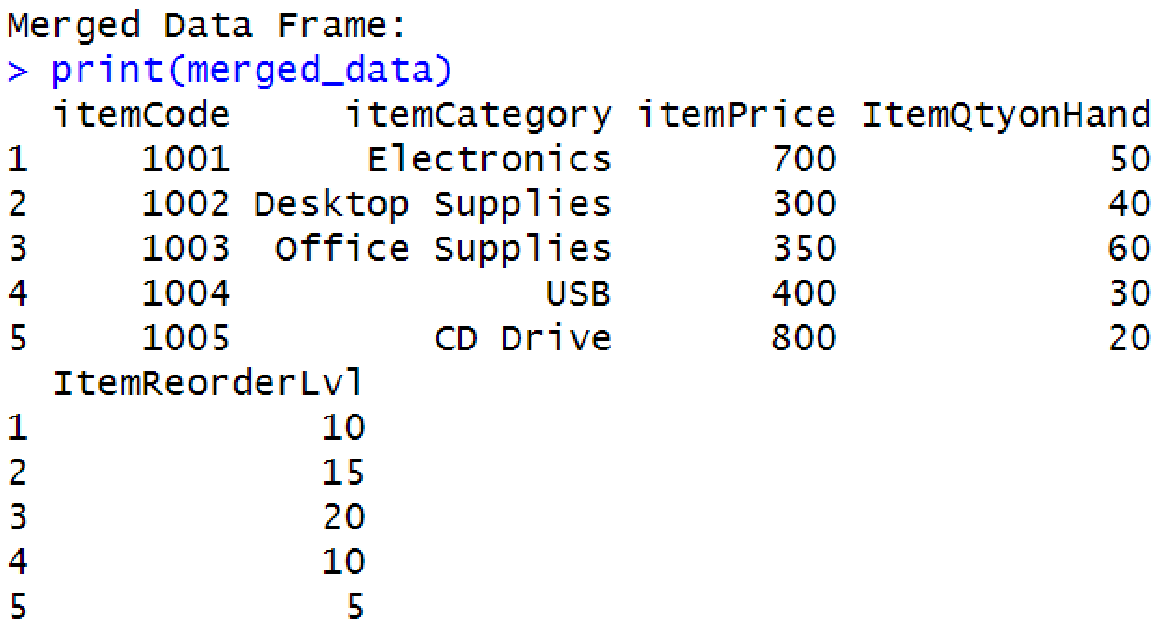
```
Items with price greater than or equal to 350:
> print(price_subset)
  itemCode    itemCategory itemPrice
1     1001     Electronics       700
3     1003 Office Supplies       350
4     1004             USB       400
5     1005        CD Drive       800
```



```
Items with category Office Supplies or Desktop Supplies:
> print(category_subset)
  itemCode     itemCategory itemPrice
2     1002 Desktop Supplies       300
3     1003  Office Supplies       350
```



```
Item Details Data Frame:
> print(item_details)
  itemCode ItemQtyonHand ItemReorderLvl
1     1001            50             10
2     1002            40             15
3     1003            60             20
4     1004            30             10
5     1005            20              5
```



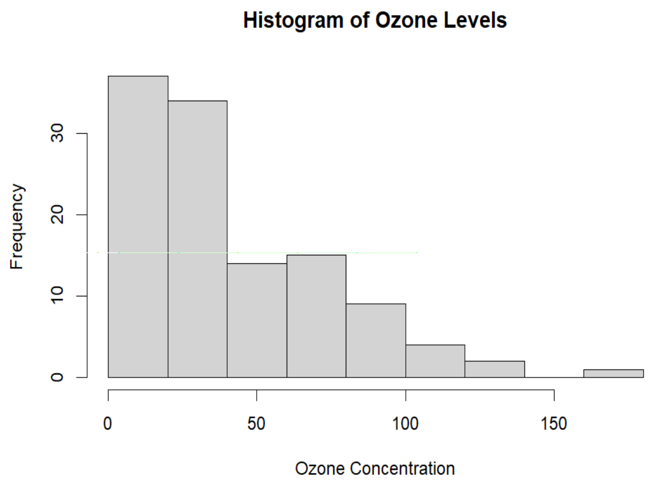
```
Merged Data Frame:
> print(merged_data)
  itemCode     itemCategory itemPrice ItemQtyonHand
1     1001      Electronics       700            50
2     1002 Desktop Supplies       300            40
3     1003  Office Supplies       350            60
4     1004              USB       400            30
5     1005         CD Drive       800            20
  ItemReorderLvl
1             10
2             15
3             20
4             10
5              5
```

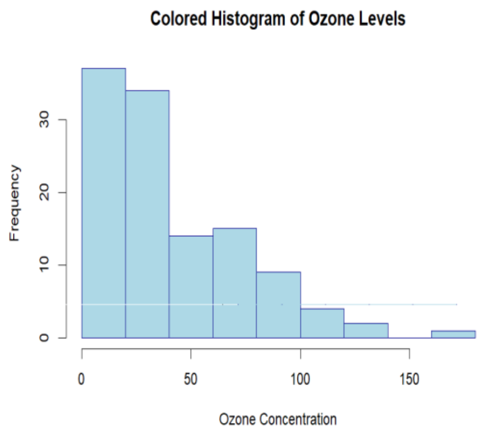
Program 8
Let us use the built-in dataset air quality which has Daily air quality measurements in New York, May to September 1973. Develop R program to generate histogram by using appropriate arguments for the following statements.
a) Assigning names, using the air quality data set.
b) Change colors of the Histogram
c) Remove Axis and Add labels to Histogram
d) Change Axis limits of a Histogram
e) Add Density curve to the histogram

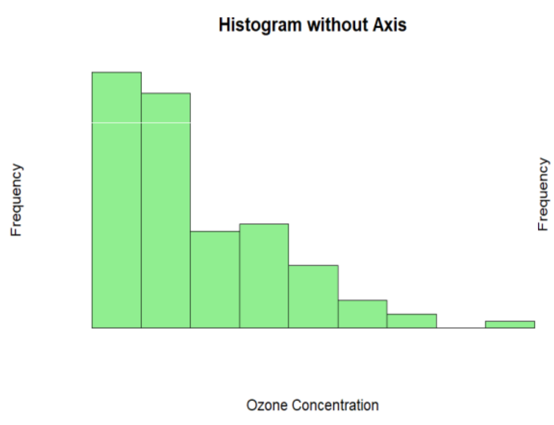
Code :

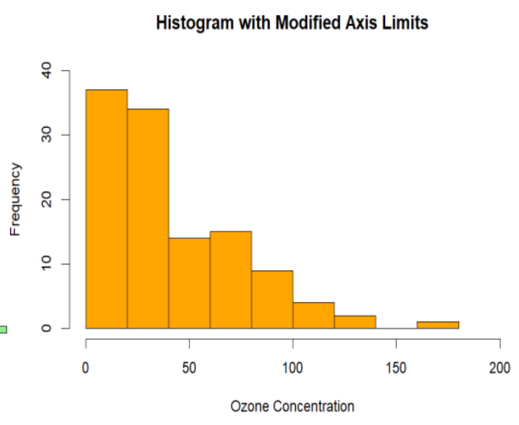
```
# Load the built-in airquality dataset
data(airquality)

# Display first few rows
head(airquality)

# Ozone variable is used for histogram
# Remove NA values
ozone_data <- na.omit(airquality$Ozone)

# a) Histogram with title and labels
hist(ozone_data,
     main = "Histogram of Ozone Levels",
     xlab = "Ozone Concentration",
     ylab = "Frequency")

# b) Change colors of the Histogram

hist(ozone_data,
     main = "Colored Histogram of Ozone Levels",
     xlab = "Ozone Concentration",
     ylab = "Frequency",
     col = "lightblue",
     border = "darkblue")

# c) Remove Axis and Add Labels
hist(ozone_data,
     main = "Histogram without Axis",
     xlab = "Ozone Concentration",
     ylab = "Frequency",
     axes = FALSE,
     col = "lightgreen")

# Add custom axis
axis(1)
axis(2)
box()
```

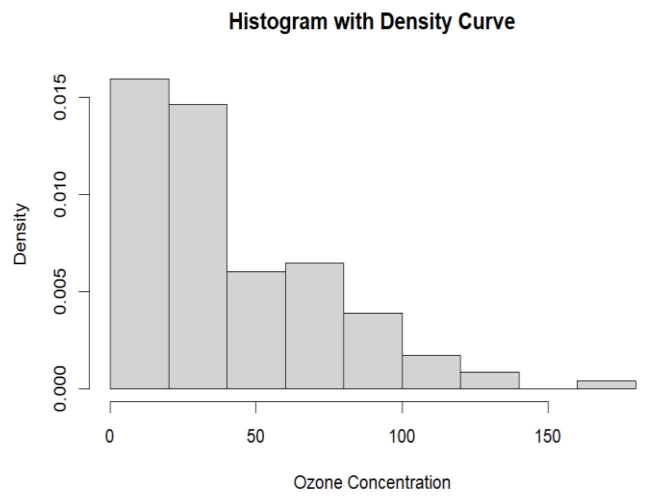
```
# d) Change Axis Limits of Histogram
hist(ozone_data,
     main = "Histogram with Modified Axis Limits",
     xlab = "Ozone Concentration",
     ylab = "Frequency",
     xlim = c(0, 200),
     ylim = c(0, 40),
     col = "orange")

# e) Add Density Curve to the Histogram
hist(ozone_data,
     probability = TRUE,
     main = "Histogram with Density Curve",
     xlab = "Ozone Concentration",
     col = "lightgray",
     border = "black")

# Add density curve
lines(density(ozone_data),
      col = "red",
      lwd = 2)
```

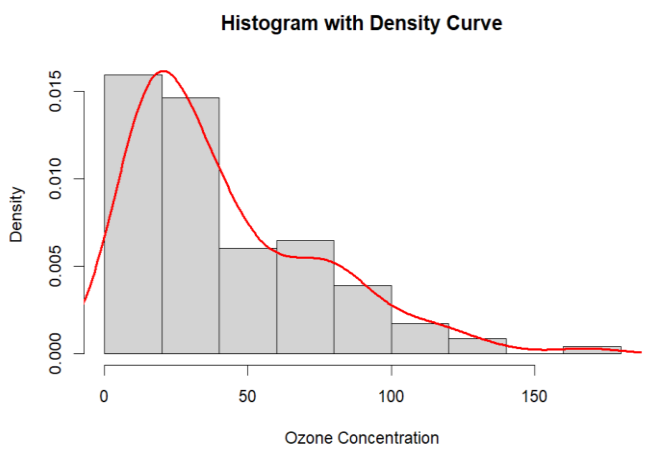












Program 9

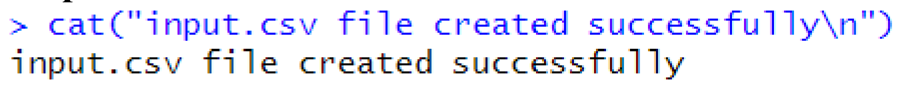
Design a data frame in R for storing about 20 employee details.
Create a CSV file named input.csv containing employee information such as id, name, salary, start_date, dept.
Import the file into R and perform the following operations:
a) Find total number of rows & columns
b) Find the maximum salary
c) Retrieve details of employee with maximum salary
d) Retrieve all employees working in IT department
e) Retrieve IT employees whose salary > 20000 and write to output.csv

CODE :

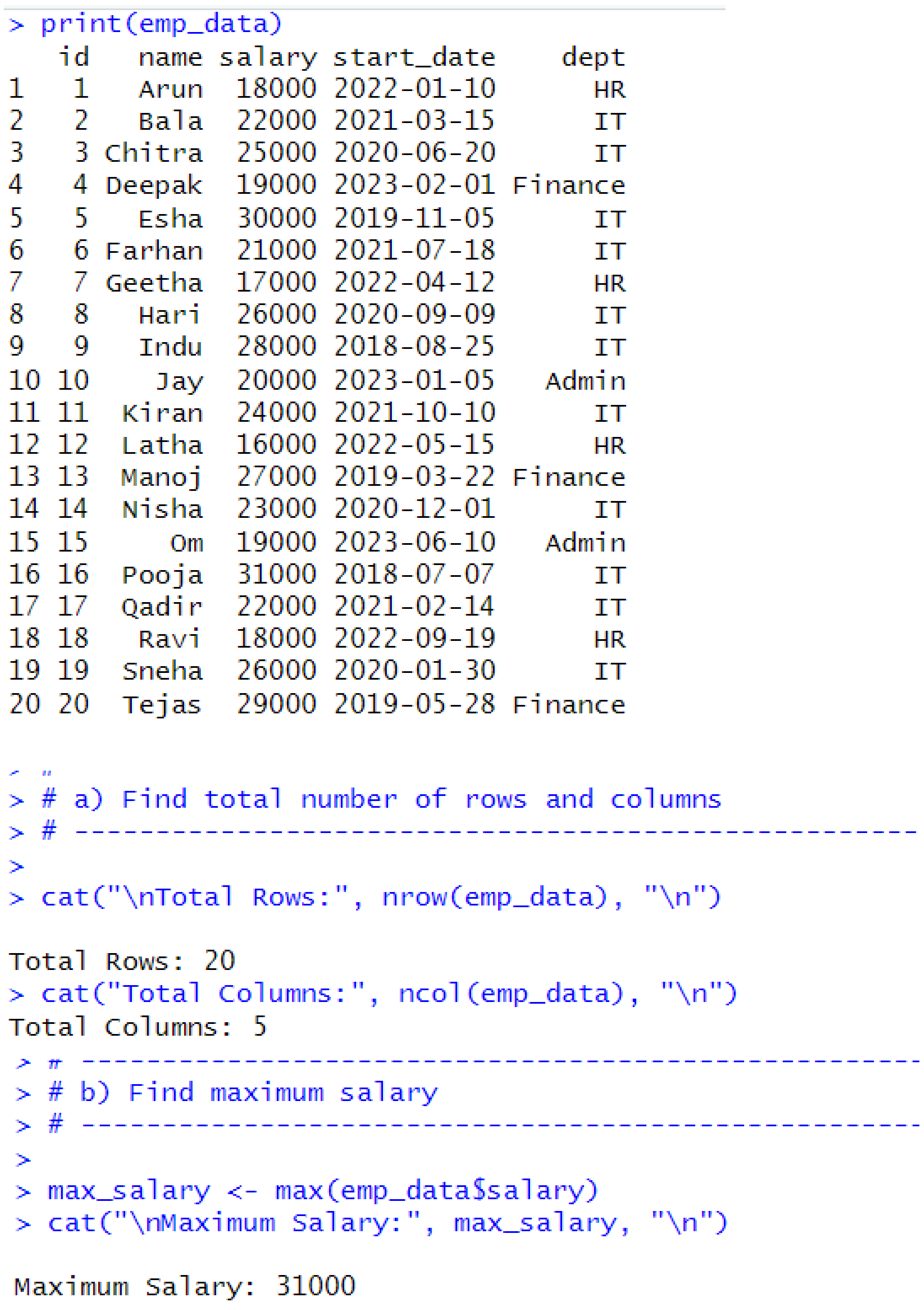
```
# Step 1: Create Employee Data Frame

employee <- data.frame(
  id = 1:20,
  name = c("Arun","Bala","Chitra","Deepak","Esha",
           "Farhan","Geetha","Hari","Indu","Jay",
           "Kiran","Latha","Manoj","Nisha","Om",
           "Pooja","Qadir","Ravi","Sneha","Tejas"),
  salary = c(18000,22000,25000,19000,30000,
             21000,17000,26000,28000,20000,
             24000,16000,27000,23000,19000,
             31000,22000,18000,26000,29000),
  start_date = as.Date(c("2022-01-10","2021-03-15","2020-06-20","2023-02-01",
                         "2019-11-05","2021-07-18","2022-04-12","2020-09-09",
                         "2018-08-25","2023-01-05","2021-10-10","2022-05-15",
                         "2019-03-22","2020-12-01","2023-06-10","2018-07-07",
                         "2021-02-14","2022-09-19","2020-01-30","2019-05-28")),
  dept = c("HR","IT","IT","Finance","IT",
           "IT","HR","IT","IT","Admin",
           "IT","HR","Finance","IT","Admin",
           "IT","IT","HR","IT","Finance")
)

# Write data frame to CSV
write.csv(employee, "input.csv", row.names = FALSE)

cat("input.csv file created successfully\n")

# Step 2: Import CSV file into R

emp_data <- read.csv("input.csv")

cat("\nEmployee Data:\n")
print(emp_data)

# a) Find total number of rows and columns

cat("\nTotal Rows:", nrow(emp_data), "\n")
cat("Total Columns:", ncol(emp_data), "\n")

# b) Find maximum salary
```

```
max_salary <- max(emp_data$salary)
cat("\nMaximum Salary:", max_salary, "\n")

# c) Employee details with maximum salary

max_salary_emp <- emp_data[emp_data$salary == max_salary, ]
cat("\nEmployee with Maximum Salary:\n")
print(max_salary_emp)

# d) Retrieve all employees working in IT department

it_employees <- subset(emp_data, dept == "IT")
cat("\nEmployees working in IT Department:\n")
print(it_employees)

# e) IT employees with salary > 20000 and write to output.csv

it_high_salary <- subset(emp_data, dept == "IT" & salary > 20000)
write.csv(it_high_salary, "output.csv", row.names = FALSE)
cat("\nIT Employees with salary > 20000 written to output.csv\n")
```

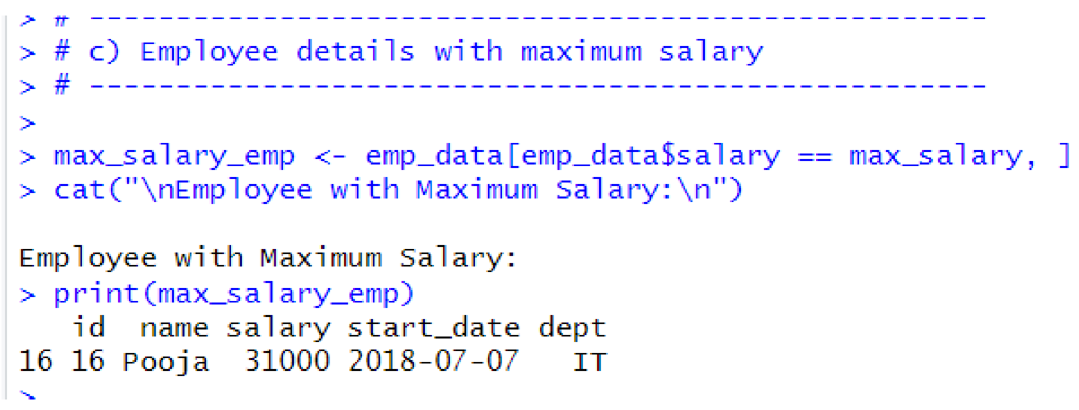
Output:



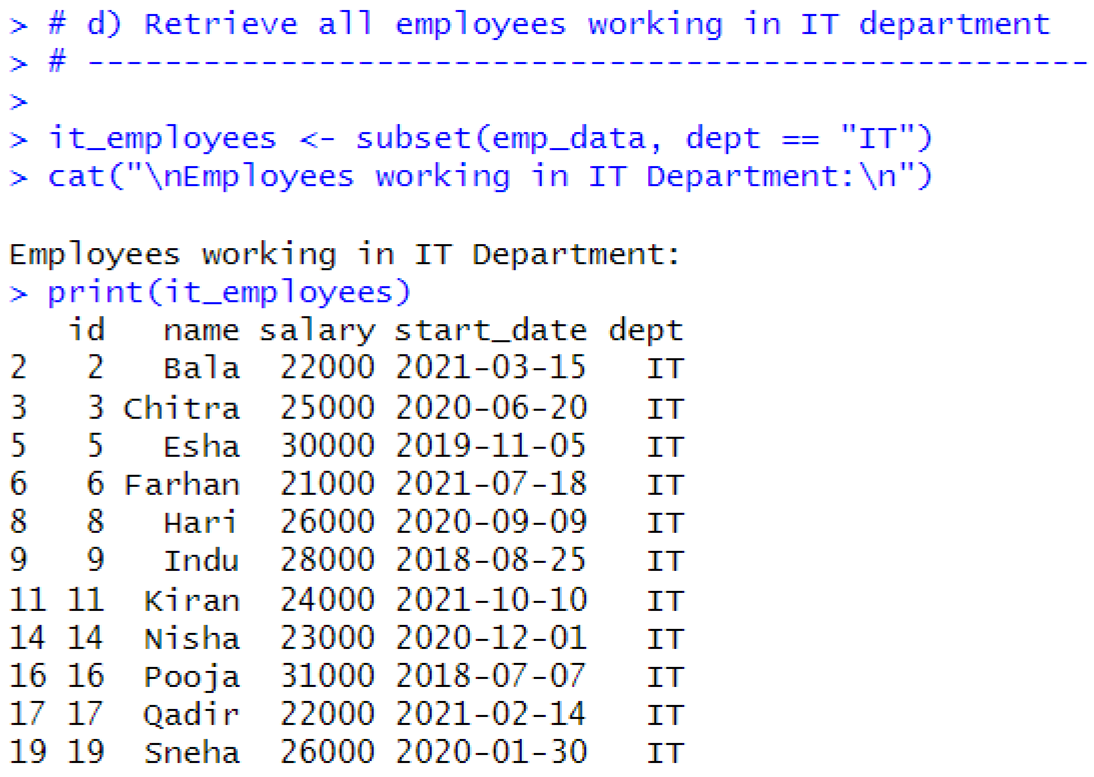
```
> cat("input.csv file created successfully\n")
input.csv file created successfully
```



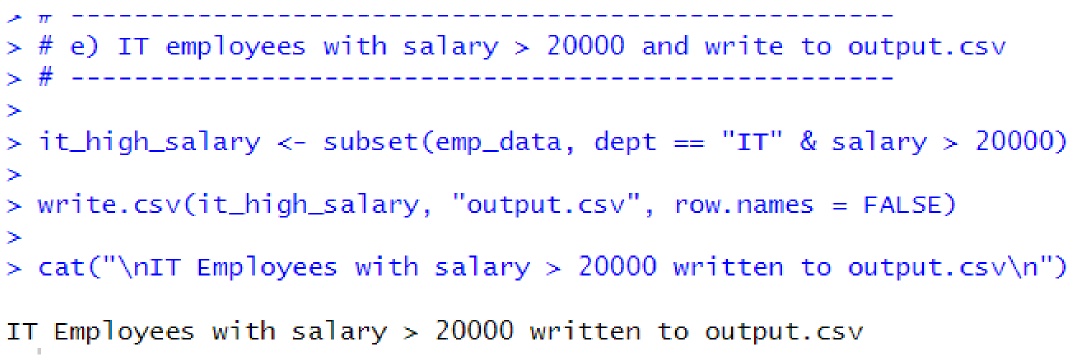
```
> print(emp_data)
   id   name salary start_date    dept
1   1   Arun  18000 2022-01-10      HR
2   2   Bala  22000 2021-03-15      IT
3   3 Chitra  25000 2020-06-20      IT
4   4 Deepak  19000 2023-02-01 Finance
5   5   Esha  30000 2019-11-05      IT
6   6 Farhan  21000 2021-07-18      IT
7   7 Geetha  17000 2022-04-12      HR
8   8   Hari  26000 2020-09-09      IT
9   9   Indu  28000 2018-08-25      IT
10 10    Jay  20000 2023-01-05   Admin
11 11  Kiran  24000 2021-10-10      IT
12 12  Latha  16000 2022-05-15      HR
13 13  Manoj  27000 2019-03-22 Finance
14 14  Nisha  23000 2020-12-01      IT
15 15     Om  19000 2023-06-10   Admin
16 16  Pooja  31000 2018-07-07      IT
17 17  Qadir  22000 2021-02-14      IT
18 18   Ravi  18000 2022-09-19      HR
19 19  Sneha  26000 2020-01-30      IT
20 20  Tejas  29000 2019-05-28 Finance

> # a) Find total number of rows and columns
> # ---------------------------------------------------
>
> cat("\nTotal Rows:", nrow(emp_data), "\n")

Total Rows: 20
> cat("Total Columns:", ncol(emp_data), "\n")
Total Columns: 5
> # ---------------------------------------------------
> # b) Find maximum salary
> # ---------------------------------------------------
>
> max_salary <- max(emp_data$salary)
> cat("\nMaximum Salary:", max_salary, "\n")

Maximum Salary: 31000
```



```
> # ----------------------------------------------------
> # c) Employee details with maximum salary
> # ----------------------------------------------------
>
> max_salary_emp <- emp_data[emp_data$salary == max_salary, ]
> cat("\nEmployee with Maximum Salary:\n")

Employee with Maximum Salary:
> print(max_salary_emp)
   id  name salary start_date dept
16 16 Pooja  31000 2018-07-07   IT
>
```

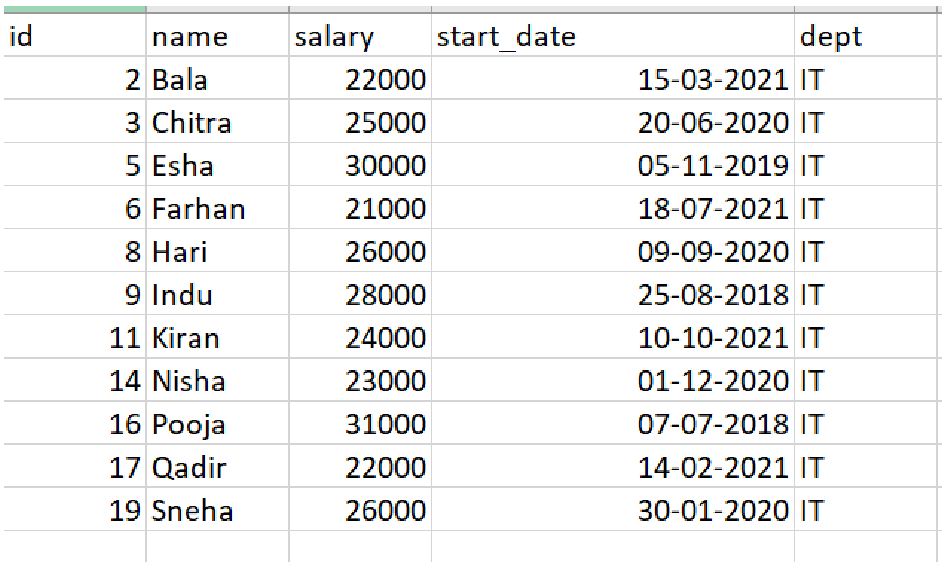


```
> # d) Retrieve all employees working in IT department
> # ----------------------------------------------------
>
> it_employees <- subset(emp_data, dept == "IT")
> cat("\nEmployees working in IT Department:\n")

Employees working in IT Department:
> print(it_employees)
   id   name salary start_date dept
2   2   Bala  22000 2021-03-15   IT
3   3 Chitra  25000 2020-06-20   IT
5   5   Esha  30000 2019-11-05   IT
6   6 Farhan  21000 2021-07-18   IT
8   8   Hari  26000 2020-09-09   IT
9   9   Indu  28000 2018-08-25   IT
11 11  Kiran  24000 2021-10-10   IT
14 14  Nisha  23000 2020-12-01   IT
16 16  Pooja  31000 2018-07-07   IT
17 17  Qadir  22000 2021-02-14   IT
19 19  Sneha  26000 2020-01-30   IT
```



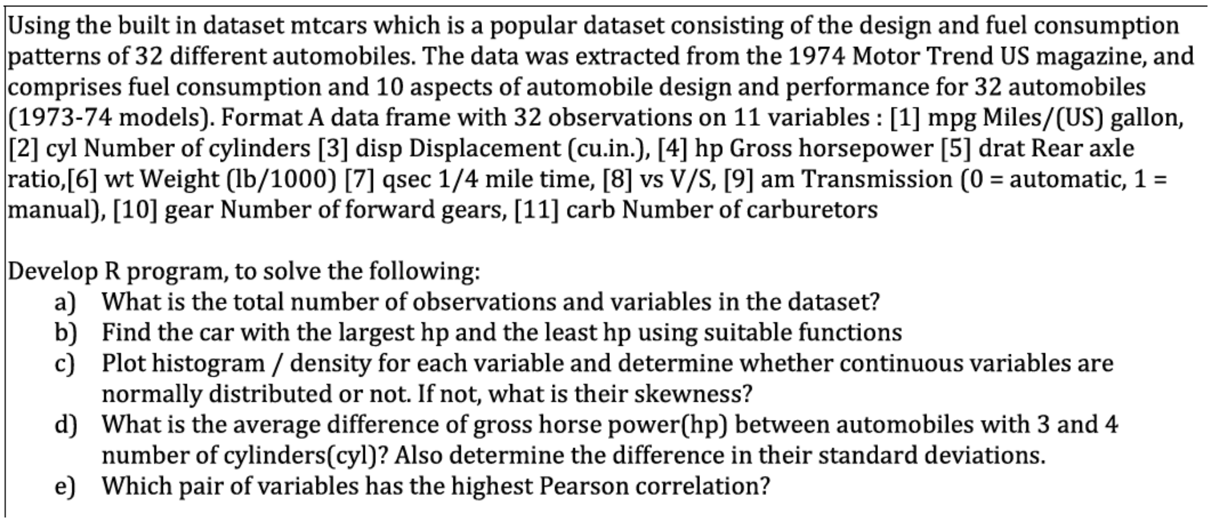
```
> # ----------------------------------------------------
> # e) IT employees with salary > 20000 and write to output.csv
> # ----------------------------------------------------
>
> it_high_salary <- subset(emp_data, dept == "IT" & salary > 20000)
>
> write.csv(it_high_salary, "output.csv", row.names = FALSE)
>
> cat("\nIT Employees with salary > 20000 written to output.csv\n")

IT Employees with salary > 20000 written to output.csv
```

Output.csv



id	name	salary	start_date	dept
2	Bala	22000	15-03-2021	IT
3	Chitra	25000	20-06-2020	IT
5	Esha	30000	05-11-2019	IT
6	Farhan	21000	18-07-2021	IT
8	Hari	26000	09-09-2020	IT
9	Indu	28000	25-08-2018	IT
11	Kiran	24000	10-10-2021	IT
14	Nisha	23000	01-12-2020	IT
16	Pooja	31000	07-07-2018	IT
17	Qadir	22000	14-02-2021	IT
19	Sneha	26000	30-01-2020	IT

Experiment 10 :



Using the built in dataset mtcars which is a popular dataset consisting of the design and fuel consumption patterns of 32 different automobiles. The data was extracted from the 1974 Motor Trend US magazine, and comprises fuel consumption and 10 aspects of automobile design and performance for 32 automobiles (1973-74 models). Format A data frame with 32 observations on 11 variables : [1] mpg Miles/(US) gallon, [2] cyl Number of cylinders [3] disp Displacement (cu.in.), [4] hp Gross horsepower [5] drat Rear axle ratio,[6] wt Weight (lb/1000) [7] qsec 1/4 mile time, [8] vs V/S, [9] am Transmission (0 = automatic, 1 = manual), [10] gear Number of forward gears, [11] carb Number of carburetors

Develop R program, to solve the following:

a) What is the total number of observations and variables in the dataset?
b) Find the car with the largest hp and the least hp using suitable functions
c) Plot histogram / density for each variable and determine whether continuous variables are normally distributed or not. If not, what is their skewness?
d) What is the average difference of gross horse power(hp) between automobiles with 3 and 4 number of cylinders(cyl)? Also determine the difference in their standard deviations.
e) Which pair of variables has the highest Pearson correlation?

Code:

```
# Load the mtcars dataset

data(mtcars)
cat("First few records of mtcars dataset:\n")
head(mtcars)

# a) Total number of observations and variables

cat("\nTotal Observations (Rows):", nrow(mtcars), "\n")
cat("Total Variables (Columns):", ncol(mtcars), "\n")

# b) Car with largest and least horsepower

max_hp <- max(mtcars$hp)
```

```r
min_hp <- min(mtcars$hp)

cat("\nMaximum Horsepower:", max_hp, "\n")
cat("Minimum Horsepower:", min_hp, "\n")

cat("\nCar(s) with Maximum Horsepower:\n")
print(mtcars[mtcars$hp == max_hp, ])

cat("\nCar(s) with Minimum Horsepower:\n")
print(mtcars[mtcars$hp == min_hp, ])

# c) Histogram and Density plots & Skewness

# Function to compute skewness manually
skewness <- function(x) {
  mean((x - mean(x))^3) / sd(x)^3
}

# Continuous variables
cont_vars <- c("mpg","disp","hp","drat","wt","qsec")

for (var in cont_vars) {
  hist(mtcars[[var]],
       probability = TRUE,
       main = paste("Histogram & Density of", var),
       xlab = var,
       col = "lightblue")
  lines(density(mtcars[[var]]), col = "red", lwd = 2)

  cat("\nSkewness of", var, ":", skewness(mtcars[[var]]), "\n")
}

# d) Average hp difference and SD difference
#    between 3-cylinder and 4-cylinder cars

hp_3 <- mtcars$hp[mtcars$cyl == 3]
hp_4 <- mtcars$hp[mtcars$cyl == 4]

mean_diff <- mean(hp_3) - mean(hp_4)
sd_diff <- sd(hp_3) - sd(hp_4)

cat("\nAverage HP of 3-cylinder cars:", mean(hp_3), "\n")
cat("Average HP of 4-cylinder cars:", mean(hp_4), "\n")
```

cat("Difference in Average HP (3 - 4):", mean_diff, "\n")

cat("\nSD of HP (3-cylinder):", sd(hp_3), "\n")
cat("SD of HP (4-cylinder):", sd(hp_4), "\n")
cat("Difference in SD (3 - 4):", sd_diff, "\n")

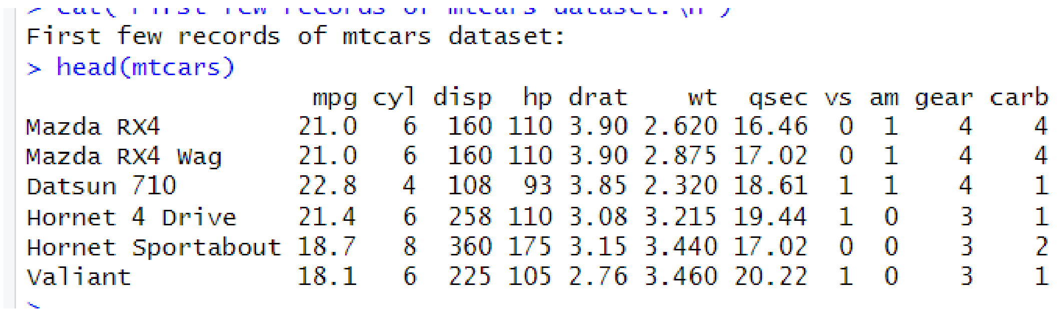
# e) Pair of variables with highest Pearson correlation

corr_matrix <- cor(mtcars)

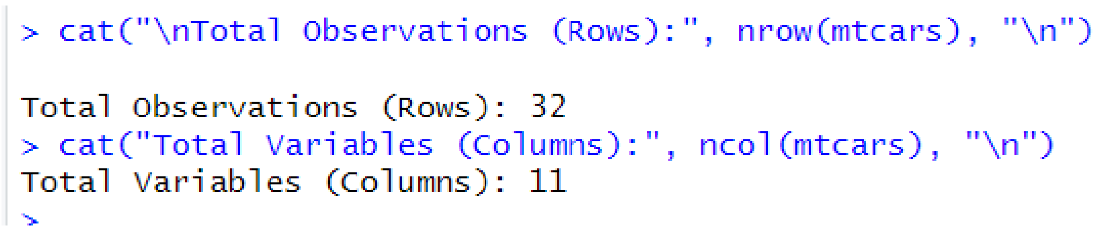
# Ignore self-correlations
corr_matrix[lower.tri(corr_matrix, diag = TRUE)] <- NA

max_corr <- max(abs(corr_matrix), na.rm = TRUE)

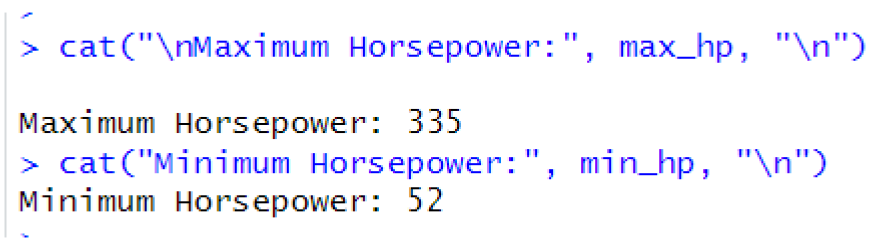
cat("\nHighest Pearson Correlation Value:", max_corr, "\n")
cat("Variable Pair with Highest Correlation:\n")
print(which(abs(corr_matrix) == max_corr, arr.ind = TRUE))



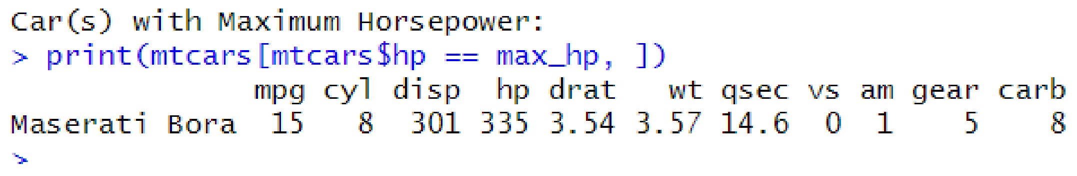
```
> cat("First few records of mtcars dataset:\n")
First few records of mtcars dataset:
> head(mtcars)
                   mpg cyl disp  hp drat    wt  qsec vs am gear carb
Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
Valiant           18.1   6  225 105 2.76 3.460 20.22  1  0    3    1
>
```



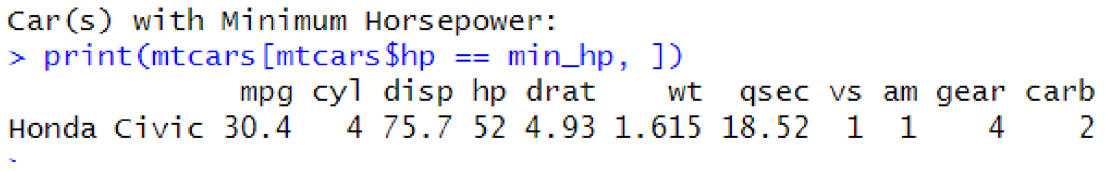
```
> cat("\nTotal Observations (Rows):", nrow(mtcars), "\n")

Total Observations (Rows): 32
> cat("Total Variables (Columns):", ncol(mtcars), "\n")
Total Variables (Columns): 11
>
```



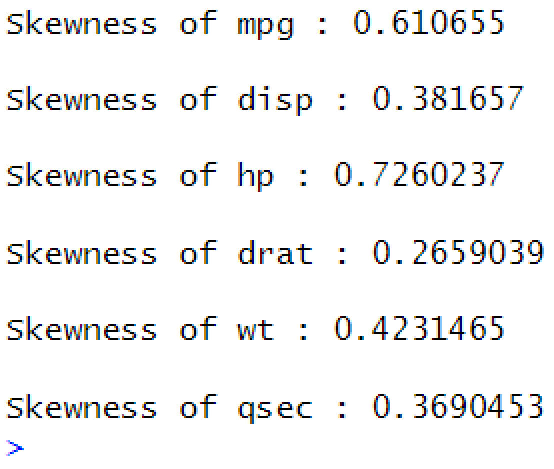
```
> cat("\nMaximum Horsepower:", max_hp, "\n")

Maximum Horsepower: 335
> cat("Minimum Horsepower:", min_hp, "\n")
Minimum Horsepower: 52
```



```
Car(s) with Maximum Horsepower:
> print(mtcars[mtcars$hp == max_hp, ])
              mpg cyl disp  hp drat   wt qsec vs am gear carb
Maserati Bora  15   8  301 335 3.54 3.57 14.6  0  1    5    8
>
```



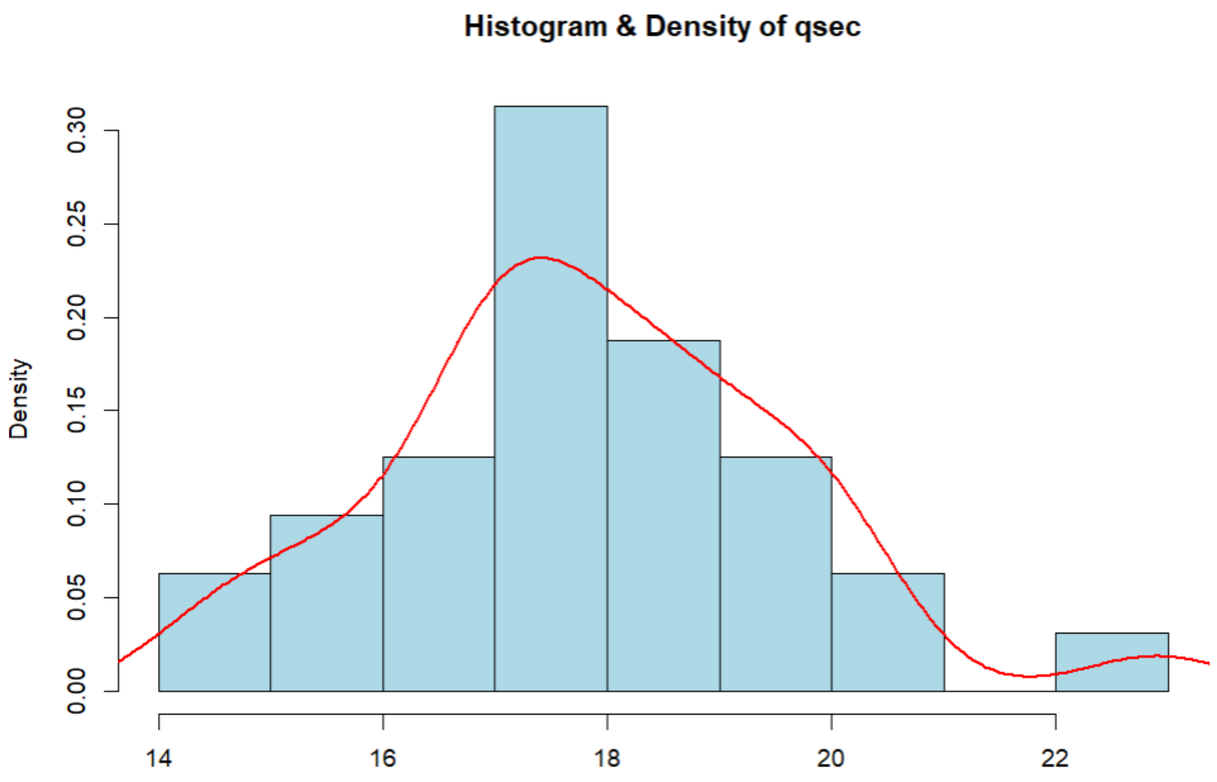
```
Car(s) with Minimum Horsepower:
> print(mtcars[mtcars$hp == min_hp, ])
             mpg cyl disp hp drat    wt  qsec vs am gear carb
Honda Civic 30.4   4 75.7 52 4.93 1.615 18.52  1  1    4    2
```



```
Skewness of mpg : 0.610655

Skewness of disp : 0.381657

Skewness of hp : 0.7260237

Skewness of drat : 0.2659039

Skewness of wt : 0.4231465

Skewness of qsec : 0.3690453
>
```

```
>
> cat("\nAverage HP of 3-cylinder cars:", mean(hp_3), "\n")

Average HP of 3-cylinder cars: NaN
> cat("Average HP of 4-cylinder cars:", mean(hp_4), "\n")
Average HP of 4-cylinder cars: 82.63636
> cat("Difference in Average HP (3 - 4):", mean_diff, "\n")
Difference in Average HP (3 - 4): NaN
>
> cat("\nSD of HP (3-cylinder):", sd(hp_3), "\n")

SD of HP (3-cylinder): NA
> cat("SD of HP (4-cylinder):", sd(hp_4), "\n")
SD of HP (4-cylinder): 20.93453
> cat("Difference in SD (3 - 4):", sd_diff, "\n")
Difference in SD (3 - 4): NA
```

```
Highest Pearson Correlation Value: 0.9020329
> cat("Variable Pair with Highest Correlation:\n")
Variable Pair with Highest Correlation:
> print(which(abs(corr_matrix) == max_corr, arr.ind = TRUE))
    row col
cyl   2   3
```



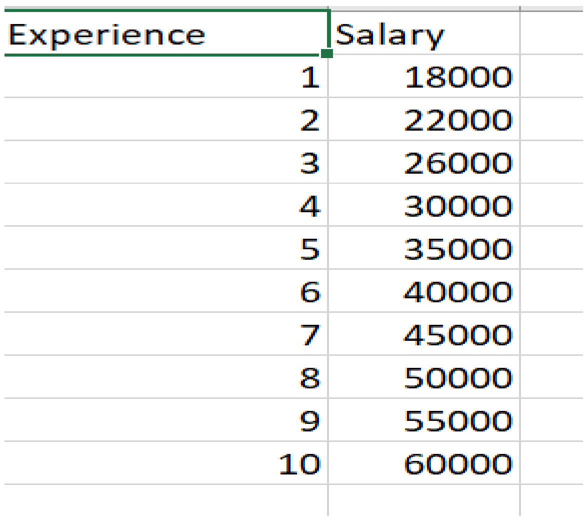
Experiment 11

Demonstrate the progression of salary with years of experience using a suitable data set (You can create your own dataset). Plot the graph visualizing the best fit line on the plot of the given data points. Plot a curve of Actual Values vs. Predicted values to show their correlation and performance of the model. Interpret the meaning of the slope and y-intercept of the line with respect to the given data. Implement using lm function. Save the graphs and coefficients in files. Attach the predicted values of salaries as a new column to the original data set and save the data as a new CSV file.

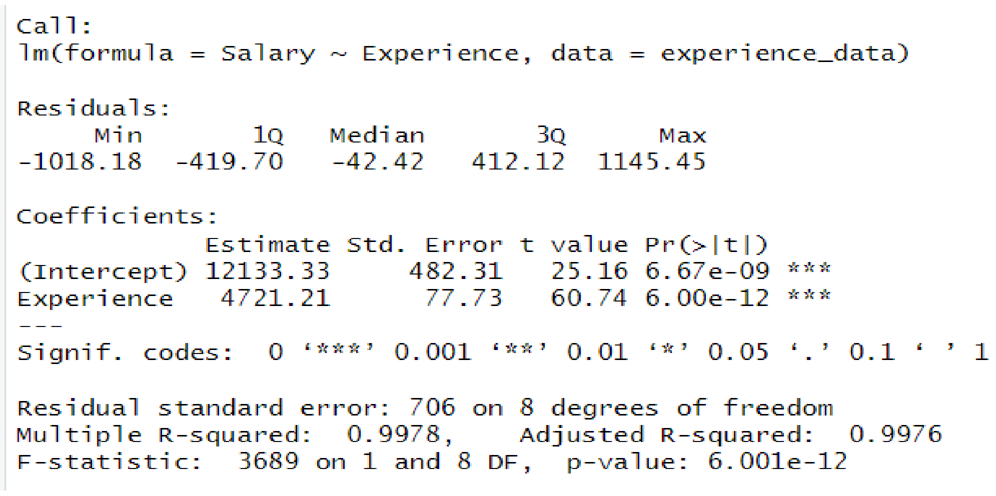
CODE:

```
# Step 1: Create dataset (Years of Experience vs Salary)

experience_data <- data.frame(
  Experience = c(1,2,3,4,5,6,7,8,9,10),
  Salary = c(18000,22000,26000,30000,35000,
             40000,45000,50000,55000,60000)
)

# Save original dataset
write.csv(experience_data, "salary_experience_input.csv", row.names = FALSE)

# Step 2: Apply Linear Regression using lm()

model <- lm(Salary ~ Experience, data = experience_data)

# Display model summary
summary(model)

# Step 3: Plot Best Fit Line

png("salary_vs_experience.png")

plot(experience_data$Experience, experience_data$Salary,
     main = "Salary vs Years of Experience",
     xlab = "Years of Experience",
     ylab = "Salary",
     pch = 19,
     col = "blue")

abline(model, col = "red", lwd = 2)

dev.off()
```

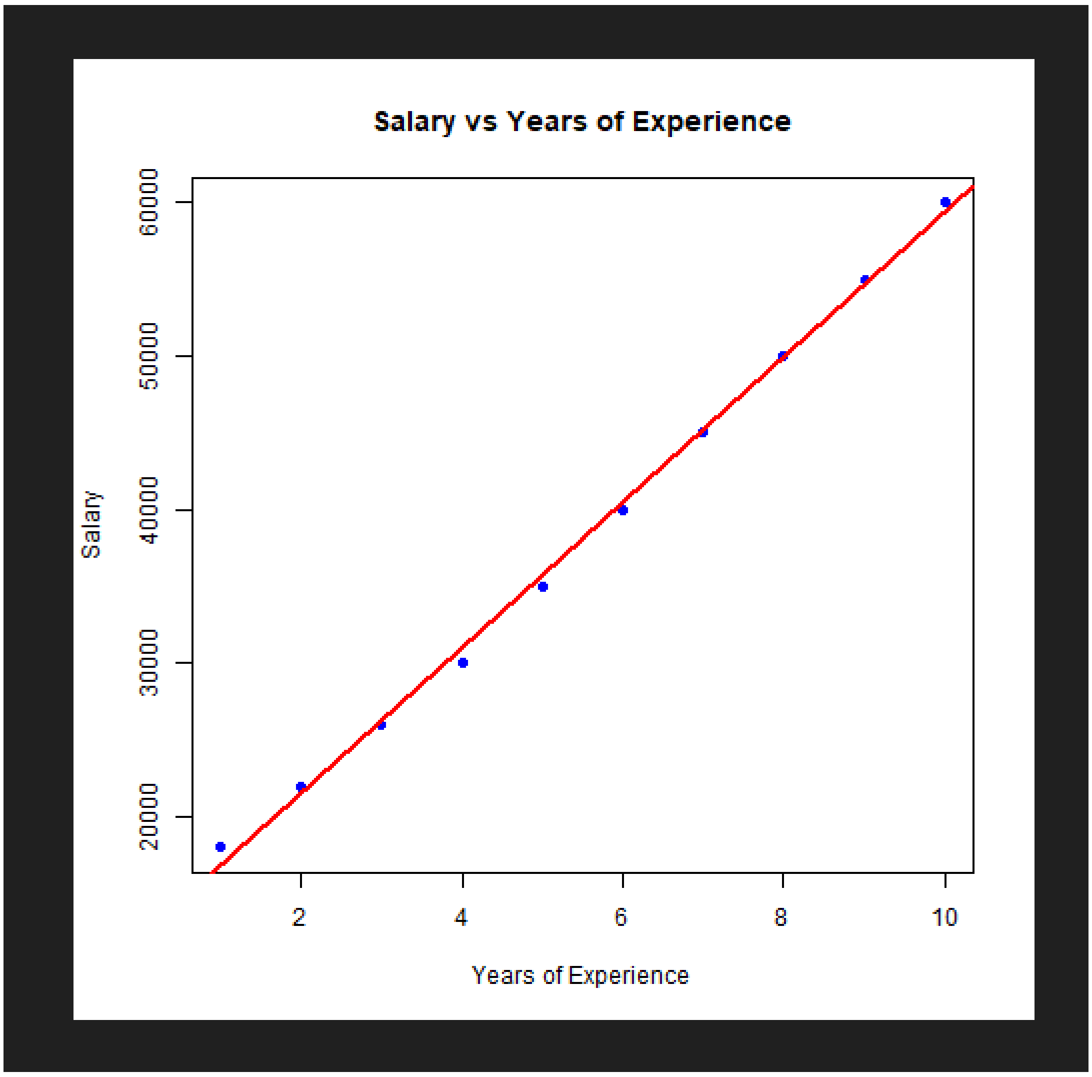
```
# Step 4: Predicted Values

experience_data$Predicted_Salary <- predict(model)

# Step 5: Actual vs Predicted Plot

png("actual_vs_predicted.png")

plot(experience_data$Salary, experience_data$Predicted_Salary,
     main = "Actual vs Predicted Salary",
     xlab = "Actual Salary",
     ylab = "Predicted Salary",
     pch = 19,
     col = "green")

abline(0,1,col="red",lwd=2)

dev.off()

# Step 6: Save Coefficients to File

coefficients <- coef(model)
write.csv(coefficients, "model_coefficients.csv")

# Step 7: Save Updated Dataset
write.csv(experience_data, "salary_experience_output.csv", row.names = FALSE)

cat("Program executed successfully.\n")
```

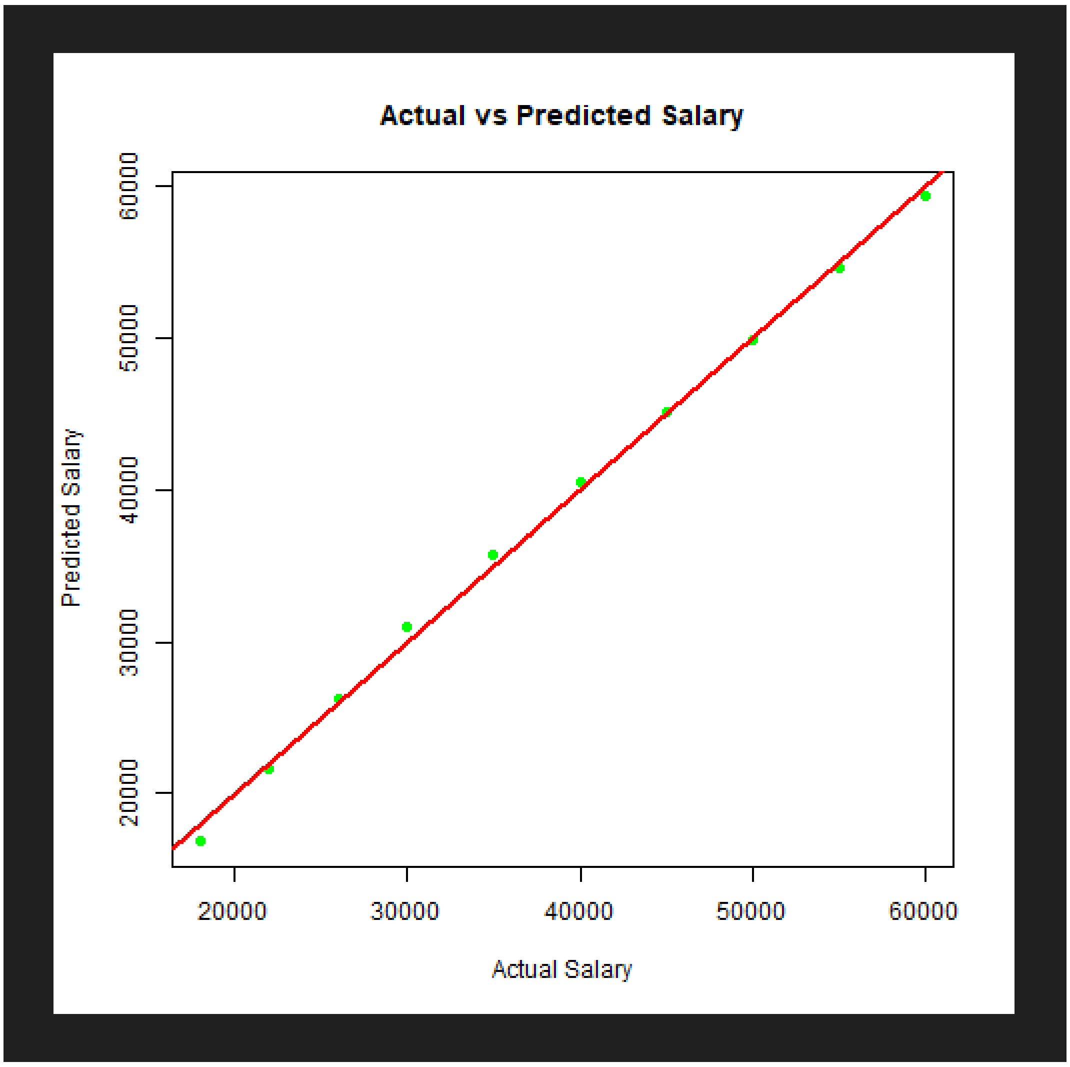


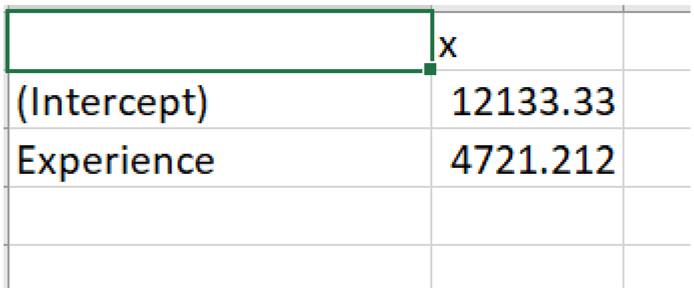
Experience	Salary
1	18000
2	22000
3	26000
4	30000
5	35000
6	40000
7	45000
8	50000
9	55000
10	60000



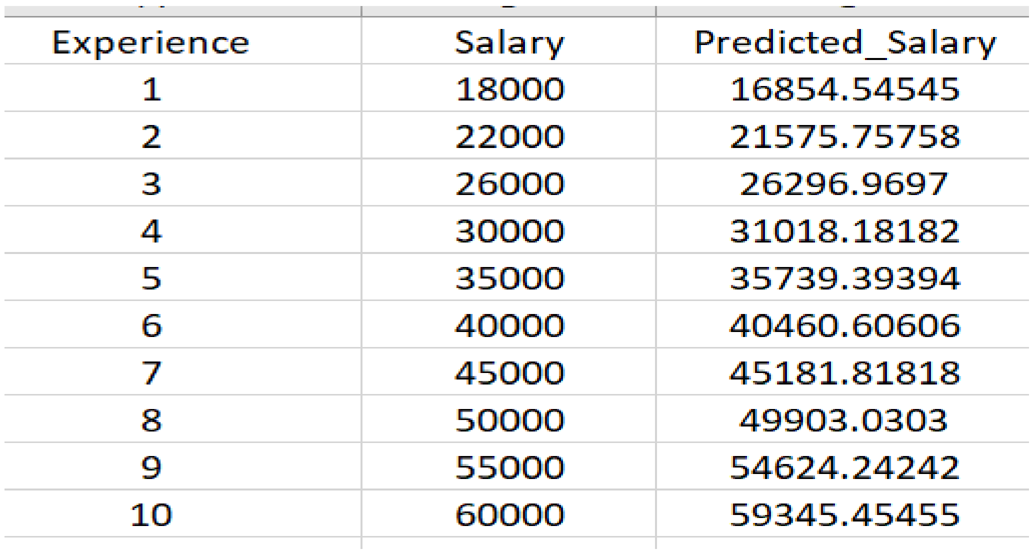
```
Call:
lm(formula = Salary ~ Experience, data = experience_data)

Residuals:
     Min       1Q   Median       3Q      Max 
-1018.18  -419.70   -42.42   412.12  1145.45 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) 12133.33     482.31   25.16 6.67e-09 ***
Experience   4721.21      77.73   60.74 6.00e-12 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 706 on 8 degrees of freedom
Multiple R-squared:  0.9978,	Adjusted R-squared:  0.9976 
F-statistic:  3689 on 1 and 8 DF,  p-value: 6.001e-12
```


Salary vs Years of Experience
Salary
20000
30000
40000
50000
60000
2
4
6
8
10
Years of Experience





	x
(Intercept)	12133.33
Experience	4721.212



Experience	Salary	Predicted_Salary
1	18000	16854.54545
2	22000	21575.75758
3	26000	26296.9697
4	30000	31018.18182
5	35000	35739.39394
6	40000	40460.60606
7	45000	45181.81818
8	50000	49903.0303
9	55000	54624.24242
10	60000	59345.45455

VIVA QUESTIONS & ANSWERS

A. R Basics & Environment

1. What is R?
Ans: R is a programming language and environment used for statistical computing, data analysis, and visualization.

2. What is RStudio?
Ans: RStudio is an Integrated Development Environment (IDE) for writing and running R programs.

3. What is a data frame in R?
Ans: A data frame is a table-like structure where each column can have a different data type.

4. How do you load a built-in dataset in R?
Ans: Using data(dataset_name).

5. What is the use of head()?
Ans: It displays the first six rows of a dataset.

6. What is NA in R?
Ans: NA represents missing or undefined values.

B. Data Types & Variables

7. What are basic data types in R?
Ans: Numeric (double), Integer, Logical, Character, Complex.

8. How do you check the data type of a variable?
Ans: Using class().

9. What is the difference between integer and numeric?
Ans: Integer stores whole numbers; numeric stores decimal values.

10. What is a logical data type?
Ans: It stores TRUE or FALSE values.

11. How do you convert character input to integer?
Ans: Using as.integer().

12. Why does readline() return character data?
Ans: Because all user input is read as text.

C. Control Structures & Recursion

13. What is a recursive function?
Ans: A function that calls itself.

14. Why is a base case important in recursion?
Ans: It stops infinite recursive calls.

15. What is the base condition for factorial?
Ans: When n = 0 or n = 1.

16. Why does factorial not work for negative numbers?
Ans: Factorial is mathematically undefined for negative numbers.

17. What is the role of is.na()?
Ans: It checks whether a value is NA.

18. What happens if base condition is missing?
Ans: The program results in infinite recursion and crashes.

D. Vectors, Matrices & Arrays
19. How do you create a vector in R?
Ans: Using c().

20. How do you create a matrix?
Ans: Using matrix().

21. What is rbind()?
Ans: It binds vectors by rows.

22. What is cbind()?
Ans: It binds vectors by columns.

23. How do you extract an element from a matrix?
Ans: Using [row, column].

24. How do you create an array?
Ans: Using array().

E. Data Frames, CSV & Merging

25. How do you write a data frame to CSV?
Ans: Using write.csv().

26. How do you read a CSV file?
Ans: Using read.csv().

27. What is subsetting?
Ans: Extracting specific rows or columns based on conditions.

28. What is subset() used for?
Ans: To filter data using conditions.

29. What is the use of merge()?
Ans: To combine two data frames using a common key.

30. What is a primary key in merging?
Ans: A common column used to join two data frames.

F. Correlation & Statistical Analysis

31. What does correlation measure?
Ans: Strength and direction of relationship between variables.

32. What is Pearson correlation?
Ans: Measures linear relationship between two variables.

33. What is Spearman correlation?
Ans: Measures rank-based relationship.

34. Why are Pearson and Spearman values different?
Ans: Due to non-linearity or outliers.

35. How do you calculate correlation in R?
Ans: Using cor().

36. What is skewness?
Ans: Measure of asymmetry in data distribution.

G. Histograms & Visualization

37. What is a histogram?

Ans: A graphical representation of data distribution.

38. What does col do in hist()?

Ans: Changes bar color.

39. What does xlim control?

Ans: X-axis limits.

40. Why use probability = TRUE?

Ans: To plot density instead of frequency.

41. How do you add a density curve?

Ans: Using lines(density(data)).

H. Linear Regression & Prediction

42. What is lm()?

Ans: Function to fit linear regression models.

43. What does predict() do?

Ans: Generates predicted values from a model.

44. Meaning of data$Predicted <- predict(model)?

Ans: Adds predicted values as a new column.

45. What does the slope represent?

Ans: Change in dependent variable per unit change in independent variable.