**Set-1.**

data(ChickWeight)

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

summary(model)

ChickWeight$predicted\_weight <- predict(model)

plot(ChickWeight$predicted\_weight, ChickWeight$weight,

xlab = "Predicted Weight", ylab = "Actual Weight")

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

new\_data <- data.frame(Time = 10, Diet = factor(1))

predicted\_weight <- predict(model, newdata = new\_data)

predicted\_weight

mse <- mean(model$residuals^2)

print(paste("Mean Squared Error (MSE) of the model:", mse))

**2.**

data(airquality)

# Get summary statistics

summary(airquality)

# Load the reshape2 package

library(reshape2)

# Melt the airquality dataset

molten\_airquality <- melt(airquality)

head(molten\_airquality)

# Melt with Month and Day as ID variables

molten\_airquality\_id <- melt(airquality, id.vars = c("Month", "Day"))

head(molten\_airquality\_id)

# Cast the molten dataset with respect to Month and Day

cast\_airquality <- dcast(molten\_airquality\_id, Month + Day ~ variable, value.var = "value")

head(cast\_airquality)

# Compute the average of Ozone, Solar.R, Wind, and Temperature per month

average\_airquality <- aggregate(. ~ Month, data = airquality[, c("Month", "Ozone", "Solar.R", "Wind", "Temp")], FUN = mean, na.rm = TRUE)

head(average\_airquality)

**3.** library(caTools)

library(caret)

library(nnet)

data(iris)

sample\_split <- sample.split(iris$Species, SplitRatio = 0.8)

train\_data <- subset(iris, sample\_split == TRUE)

test\_data <- subset(iris, sample\_split == FALSE)

multinom\_model <- multinom(Species ~ Petal.Length + Petal.Width, data = train\_data)

test\_data$predicted\_species <- predict(multinom\_model, newdata = test\_data)

conf\_matrix <- confusionMatrix(as.factor(test\_data$predicted\_species), as.factor(test\_data$Species))

print(conf\_matrix)

values <- c(90, 50, 70, 80, 70, 60, 20, 30, 80, 90, 20)

mean\_value <- mean(values)

median\_value <- median(values)

freq\_table <- table(values)

print(freq\_table)

max\_freq <- max(freq\_table)

print(max\_freq)

modevector <- names(freq\_table[freq\_table == max\_freq])

cat("Mean of the vector ",mean\_value,"\n")

cat("Median of the vector ",median\_value,"\n")

cat("Mode of the vector ",modevector,"\n")

sorted\_values <- sort(values, decreasing = TRUE)

unique\_values <- unique(sorted\_values)

unique\_values

second\_highest <- unique\_values[2]

third\_lowest <- sort(unique\_values)[3]

second\_highest

third\_lowest

**4.** # Define the data for the array

data <- c(1:24) # Create a vector of 24 elements (4 \* 3 \* 2 = 24)

# Create an array with 4 columns, 3 rows, and 2 tables

my\_array <- array(data, dim = c(3, 4, 2))

# Display the content of the array

print("Array content:")

print(my\_array)

# Create the data frame exam\_data

**Set-2:1**.

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')

)

# Display the data frame

print(exam\_data)

**2.#** Create a vector

my\_vector <- c(1, 2, 3, 4, 5)

# Create a matrix

my\_matrix <- matrix(1:9, nrow=3, ncol=3)

# Define a function

my\_function <- function(x) {

return(x^2)

}

# Create a list containing the vector, matrix, and function

my\_list <- list(vector = my\_vector, matrix = my\_matrix, square\_function = my\_function)

# Print the content of the list

print("Content of the list:")

print(my\_list)

# Test the function within the list

print("Function output (square of 4):")

print(my\_list$square\_function(4))

**3.#** Load the dataset

data("USArrests")

# (i) Part a: Explore the summary of the dataset

print("Summary of the dataset:")

summary(USArrests)

# Number of features and their types

print("Number of features and their types:")

str(USArrests)

# Number of records for each feature

print("Number of records for each feature:")

nrow(USArrests)

# (i) Part b: Print the state which saw the largest total number of rape

max\_rape\_state <- rownames(USArrests)[which.max(USArrests$Rape)]

print(paste("State with the largest number of rapes:", max\_rape\_state))

# (i) Part c: Print the states with the max and min crime rates for murder

max\_murder\_state <- rownames(USArrests)[which.max(USArrests$Murder)]

min\_murder\_state <- rownames(USArrests)[which.min(USArrests$Murder)]

print(paste("State with the maximum murder rate:", max\_murder\_state))

print(paste("State with the minimum murder rate:", min\_murder\_state))

# (ii) Part a: Find the correlation among the features

print("Correlation matrix among the features:")

correlation\_matrix <- cor(USArrests)

print(correlation\_matrix)

# (ii) Part b: Print the states which have assault arrests more than the median of the country

median\_assault <- median(USArrests$Assault)

high\_assault\_states <- rownames(USArrests)[USArrests$Assault > median\_assault]

print("States with assault arrests more than the median:")

print(high\_assault\_states)

# (ii) Part c: Print the states that are in the bottom 25% of murder rates

murder\_quantile <- quantile(USArrests$Murder, 0.25)

low\_murder\_states <- rownames(USArrests)[USArrests$Murder <= murder\_quantile]

print("States in the bottom 25% of murder rates:")

print(low\_murder\_states)

# (iii) Part a: Create a histogram and density plot of murder arrests

library(ggplot2)

# Histogram

ggplot(USArrests, aes(x = Murder)) +

geom\_histogram(binwidth = 1, fill = "blue", color = "black") +

labs(title = "Histogram of Murder Arrests", x = "Murder Arrests", y = "Frequency")

# Density plot

ggplot(USArrests, aes(x = Murder)) +

geom\_density(fill = "blue", alpha = 0.5) +

labs(title = "Density Plot of Murder Arrests", x = "Murder Arrests", y = "Density")

# (iii) Part b: Plot relationship between murder arrest rate and urban population, with assault arrest coloring

ggplot(USArrests, aes(x = Murder, y = UrbanPop, color = Assault)) +

geom\_point(size = 3) +

scale\_color\_gradient(low = "blue", high = "red") +

labs(title = "Murder Arrest Rate vs Urban Population", x = "Murder Arrests", y = "Urban Population (%)")

# (iii) Part c: Bar graph showing murder rate for each of the 50 states

USArrests$State <- rownames(USArrests) # Add state names as a column for easy plotting

ggplot(USArrests, aes(x = reorder(State, -Murder), y = Murder)) +

geom\_bar(stat = "identity", fill = "steelblue") +

coord\_flip() +

labs(title = "Murder Rate by US State", x = "State", y = "Murder Arrests")

**4.** # Read the CSV file

data <- read.csv("file\_path.csv")

# Display the content of the file

print("Content of the CSV file:")

print(data)

**set-3:1.** # Create a numeric vector

numeric\_vector <- c(1, 2, 3.5, 4.8, 6)

# Create a character vector

character\_vector <- c("apple", "banana", "cherry")

# Create a logical vector

logical\_vector <- c(TRUE, FALSE, TRUE, FALSE)

# Display the content and type of each vector

print("Numeric Vector:")

print(numeric\_vector)

print(paste("Type of numeric\_vector:", typeof(numeric\_vector)))

print("Character Vector:")

print(character\_vector)

print(paste("Type of character\_vector:", typeof(character\_vector)))

print("Logical Vector:")

print(logical\_vector)

print(paste("Type of logical\_vector:", typeof(logical\_vector)))

**2.** # Create the initial 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) Extract 3rd and 5th rows with 1st and 3rd columns

extracted\_data <- exam\_data[c(3, 5), c(1, 3)]

print("Extracted 3rd and 5th rows, 1st and 3rd columns:")

print(extracted\_data)

# (b) Add a new column 'country' to the data frame

exam\_data$country <- c("USA", "USA", "USA", "USA", "UK", "USA", "USA", "India", "USA", "USA")

print("Data frame after adding 'country' column:")

print(exam\_data)

# (c) Add new rows to the data frame

new\_exam\_data <- data.frame(

name = c('Robert', 'Sophia'),

score = c(10.5, 9),

attempts = c(1, 3),

qualify = c('yes', 'no')

)

# Adding new rows to the existing data frame

exam\_data <- rbind(exam\_data, new\_exam\_data)

print("Data frame after adding new rows:")

print(exam\_data)

# (d) Sort the data frame by 'name' and 'score'

sorted\_data <- exam\_data[order(exam\_data$name, exam\_data$score), ]

print("Data frame sorted by 'name' and 'score':")

print(sorted\_data)

# (e) Save the data frame to a file and display the information

write.csv(exam\_data, "exam\_data.csv", row.names = FALSE)

print("Data saved to 'exam\_data.csv'.")

# To display the saved data from the file

saved\_data <- read.csv("exam\_data.csv")

print("Data from the saved file:")

print(saved\_data)

**3.** # Load the women dataset

data("women"

# Create a factor from the heights in the women dataset

height\_factor <- factor(women$height)

# Print the factor levels for the heights

cat("Factor corresponding to heights in the women dataset:\n")

print(height\_factor)

# Print the levels of the factor

cat("\nLevels of the height factor:\n")

print(levels(height\_factor))

**4.** # 1. Create a 5x4 matrix filled by rows

matrix\_5x4 <- matrix(1:20, nrow = 5, ncol = 4, byrow = TRUE)

print("5x4 Matrix filled by rows:")

print(matrix\_5x4)

# 2. Create a 3x3 matrix with labels and fill by rows

matrix\_3x3 <- matrix(1:9, nrow = 3, ncol = 3, byrow = TRUE,

dimnames = list(c("Row1", "Row2", "Row3"), c("Col1", "Col2", "Col3")))

print("3x3 Matrix with labels, filled by rows:")

print(matrix\_3x3)

# 3. Create a 2x2 matrix with labels and fill by columns

matrix\_2x2 <- matrix(1:4, nrow = 2, ncol = 2, byrow = FALSE,

dimnames = list(c("RowA", "RowB"), c("ColX", "ColY")))

print("2x2 Matrix with labels, filled by columns:")

print(matrix\_2x2)

**set-4:1.** # Create a vector of values

values <- c(1:24)

# Define dimensions for the array (3x4x2)

dimensions <- c(3, 4, 2)

# Define names for each dimension

dimnames\_list <- list(

Row = c("Row1", "Row2", "Row3"),

Column = c("Col1", "Col2", "Col3", "Col4"),

Layer = c("Layer1", "Layer2")

)

# Create the array

my\_array <- array(values, dim = dimensions, dimnames = dimnames\_list)

# Print the array

print("Array with dimension names:")

print(my\_array)

**2.** # Load the built-in airquality dataset

data("airquality")

# Check if airquality is a data frame

is\_data\_frame <- is.data.frame(airquality)

print(paste("Is airquality a data frame?", is\_data\_frame))

# Order the entire data frame by the first (Ozone) and second (Solar.R) columns

ordered\_airquality <- airquality[order(airquality$Ozone, airquality$Solar.R), ]

print("Data frame ordered by first (Ozone) and second (Solar.R) columns:")

print(head(ordered\_airquality)

# Remove the variables 'Solar.R' and 'Wind'

modified\_airquality <- ordered\_airquality[, !colnames(ordered\_airquality) %in% c("Solar.R", "Wind")]

print("Data frame after removing 'Solar.R' and 'Wind':")

print(head(modified\_airquality))

**3.** # Load the ChickWeight dataset

data("ChickWeight")

# (i) Order the data frame by 'weight' grouped by 'Diet', and extract the last 6 records

ordered\_data <- ChickWeight[order(ChickWeight$Diet, ChickWeight$weight), ]

last\_6\_records <- tail(ordered\_data, 6)

print("Last 6 records of the ordered data frame:")

print(last\_6\_records)

# (ii.a) Perform melting function based on "Chick", "Time", "Diet" as ID variables

library(reshape2)

melted\_data <- melt(ChickWeight, id.vars = c("Chick", "Time", "Diet"), measure.vars = "weight")

print("Melted data:")

print(head(melted\_data))

# (ii.b) Perform cast function to display the mean value of weight grouped by Diet

mean\_weight\_by\_diet <- dcast(melted\_data, Diet ~ variable, fun.aggregate = mean)

print("Mean weight grouped by Diet:")

print(mean\_weight\_by\_diet)

# (ii.c) Perform cast function to display the mode of weight grouped by Diet

# Define a custom function to calculate the mode

calculate\_mode <- function(x) {

unique\_x <- unique(x)

unique\_x[which.max(tabulate(match(x, unique\_x)))]

}

mode\_weight\_by\_diet <- dcast(melted\_data, Diet ~ variable, fun.aggregate = calculate\_mode)

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

print(mode\_weight\_by\_diet)

**4.** # Load necessary libraries

library(caTools) # For splitting data

library(caret) # For creating the confusion matrix

# Load the iris dataset

data("iris")

# (1) Randomly split the dataset into 80% training and 20% testing

set.seed(123) # Set seed for reproducibility

split <- sample.split(iris$Species, SplitRatio = 0.8)

train\_data <- subset(iris, split == TRUE)

test\_data <- subset(iris, split == FALSE

# (2) Create logistic regression model using 'Species' as target, 'Petal.Length' and 'Petal.Width' as features

# We need to convert the Species into a binary factor for binary logistic regression

# Let's focus on two classes only (e.g., versicolor and virginica)

binary\_train <- subset(train\_data, Species != "setosa")

binary\_train$Species <- factor(binary\_train$Species)

logistic\_model <- glm(Species ~ Petal.Length + Petal.Width, data = binary\_train, family = binomial)

# (3) Predict the probabilities on the test data

binary\_test <- subset(test\_data, Species != "setosa")

binary\_test$Species <- factor(binary\_test$Species)

predicted\_prob <- predict(logistic\_model, newdata = binary\_test, type = "response")

# Convert probabilities to predicted classes (0.5 threshold)

predicted\_classes <- ifelse(predicted\_prob > 0.5, "virginica", "versicolor")

# (4) Create a confusion matrix

confusion\_matrix <- confusionMatrix(factor(predicted\_classes), binary\_test$Species)

# Print the confusion matrix

print("Confusion Matrix:")

print(confusion\_matrix)

**set-5:2**

# Set seed for reproducibility

set.seed(123)

# Create a random sample of letters from the LETTERS factor

sample\_size <- 10 # Define the size of the sample

random\_sample <- sample(LETTERS, size = sample\_size, replace = TRUE)

# Convert the random sample to a factor

factor\_sample <- factor(random\_sample)

# Extract the unique levels of the factor

unique\_levels <- levels(factor\_sample)

# Extract five levels from the factor

five\_levels <- unique\_levels[1:5] # Extract the first five levels

# Print the five levels

print("Five levels of the factor from the random sample:")

print(five\_levels)

**3.** # Set seed for reproducibility

set.seed(123)

# Create a vector with 10 random integer values between -50 and +50

random\_integers <- sample(-50:50, size = 10, replace = TRUE)

# Print the vector

print("Vector of 10 random integers between -50 and +50:")

print(random\_integers)

**4.** # Loop through numbers from 1 to 100

for (i in 1:100) {

if (i %% 3 == 0 && i %% 5 == 0) {

# Print "FizzBuzz" for multiples of both 3 and 5

print("FizzBuzz")

} else if (i %% 3 == 0) {

# Print "Fizz" for multiples of 3

print("Fizz")

} else if (i %% 5 == 0) {

# Print "Buzz" for multiples of 5

print("Buzz")

} else {

# Print the number itself if it's not a multiple of 3 or 5

print(i)

}

}

**Set-6:1-**

# Function to find factors of a number

find\_factors <- function(n) {

factors <- c() # Initialize an empty vector to store factors

for (i in 1:n) {

if (n %% i == 0) {

factors <- c(factors, i) # Append the factor to the vector

}

}

return(factors)

}

# Input: Specify the number for which to find factors

number <- 24 # You can change this to any number you want

# Find and print the factors

factors\_of\_number <- find\_factors(number)

cat("Factors of", number, "are:", factors\_of\_number, "\n")

**2:** # Set seed for reproducibility

set.seed(123)

# Create a vector of random numbers from a normal distribution

# Mean = 0, Standard Deviation = 1, Sample Size = 100

random\_numbers <- rnorm(100, mean = 0, sd = 1)

# Print the generated random numbers

print("Generated random numbers:")

print(random\_numbers)

# Count occurrences of each value using table()

occurrences <- table(random\_numbers)

# Print the occurrences

print("Occurrences of each value:")

print(occurrences)

**3.** install.packages("titanic")

install.packages("ggplot2")

# Load necessary libraries

library(titanic)

library(ggplot2)

# Load the Titanic dataset

data("titanic")

# a. Draw a Bar chart to show details of "Survived" based on passenger Class

bar\_chart <- ggplot(titanic, aes(x = factor(Pclass), fill = factor(Survived))) +

geom\_bar(position = "dodge") +

labs(title = "Survival on Titanic by Passenger Class",

x = "Passenger Class",

y = "Count",

fill = "Survived") +

theme\_minimal(

print(bar\_chart)

# b. Modify the above plot based on gender of people who survived

bar\_chart\_gender <- ggplot(titanic, aes(x = factor(Pclass), fill = factor(Survived))) +

geom\_bar(position = "dodge") +

facet\_wrap(~ Sex) + # Facet by gender

labs(title = "Survival on Titanic by Passenger Class and Gender",

x = "Passenger Class",

y = "Count",

fill = "Survived") +

theme\_minimal()

print(bar\_chart\_gender)

# c. Draw histogram plot to show distribution of feature "Age"

age\_histogram <- ggplot(titanic, aes(x = Age)) +

geom\_histogram(binwidth = 5, fill = "blue", color = "black", alpha = 0.7) +

labs(title = "Age Distribution of Titanic Passengers",

x = "Age",

y = "Count") +

theme\_minimal()

print(age\_histogram)

**4.** # Create three sample arrays

array1 <- array(1:12, dim = c(3, 4)) # 3 rows, 4 columns

array2 <- array(13:24, dim = c(3, 4)) # 3 rows, 4 columns

array3 <- array(25:36, dim = c(3, 4)) # 3 rows, 4 columns

# Print the original arrays

cat("Array 1:\n")

print(array1)

cat("\nArray 2:\n")

print(array2)

cat("\nArray 3:\n")

print(array3)

# Combine the first rows of the three arrays

combined\_array <- rbind(array1[1, ], array2[1, ], array3[1, ])

# Print the combined array

cat("\nCombined Array (First Rows):\n")

print(combined\_array)

**set-7:1-**

# Define two vectors

vector1 <- 1:9 # First vector containing 1 to 9

vector2 <- 10:18 # Second vector containing 10 to 18

# Create two 3x3 matrices from the vectors

matrix1 <- matrix(vector1, nrow = 3, ncol = 3, byrow = TRUE)

matrix2 <- matrix(vector2, nrow = 3, ncol = 3, byrow = TRUE)

# Combine the two matrices into an array

array\_of\_matrices <- array(c(matrix1, matrix2), dim = c(3, 3, 2))

# Print the original matrices

cat("Matrix 1:\n")

print(matrix1)

cat("\nMatrix 2:\n")

print(matrix2)

# Print the second row of the second matrix

cat("\nSecond row of the second matrix:\n")

print(array\_of\_matrices[2, , 2]) # Accessing the second row of the second matrix

# Print the element in the 3rd row and 3rd column of the first matrix

cat("\nElement in the 3rd row and 3rd column of the first matrix:\n")

print(array\_of\_matrices[3, 3, 1]) # Accessing the 3rd row and 3rd column of the first matrix

**2:** # Define two vectors

vector1 <- c(1, 2, 3, 4, 5, 6, 7, 8, 9) # First vector with 9 elements

vector2 <- c(10, 11, 12, 13, 14, 15, 16, 17, 18) # Second vector with 9 elements

# Create a 3D array using the two vectors

array\_3d <- array(c(vector1, vector2), dim = c(3, 3, 2))

# Print the 3D array

cat("3D Array:\n")

print(array\_3d)

**3.** # Load the airquality dataset

data("airquality")

# i. Compute the mean temperature without using built-in functions

# Sum of the temperature

temp\_sum <- 0

n <- nrow(airquality) # Total number of row

for (i in 1:n) {

temp\_sum <- temp\_sum + airquality$Temp[i]

]

mean\_temp <- temp\_sum / n # Mean temperature

cat("Mean Temperature:", mean\_temp, "\n")

# ii. Extract the first five rows from airquality

first\_five\_rows <- head(airquality, 5)

cat("First five rows of airquality:\n")

print(first\_five\_rows)

# iii. Extract all columns except Temp and Wind

airquality\_no\_temp\_wind <- airquality[, !(names(airquality) %in% c("Temp", "Wind"))]

cat("Airquality dataset without Temp and Wind columns:\n")

print(airquality\_no\_temp\_wind)

# iv. Find the coldest day during the period

coldest\_day\_index <- which.min(airquality$Temp)

coldest\_day <- airquality[coldest\_day\_index, ]

cat("Coldest day:\n")

print(coldest\_day)

# v. Count the number of days with wind speed greater than 17 mph

high\_wind\_days\_count <- sum(airquality$Wind > 17)

cat("Number of days with wind speed greater than 17 mph:", high\_wind\_days\_count, "\n")

**4.** # Create a sample 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)

)

# Print the original data frame

cat("Original Data Frame:\n")

print(exam\_data)

# Sort the data frame by 'name' and 'score'

sorted\_data <- exam\_data[order(exam\_data$name, exam\_data$score), ]

# Print the sorted data frame

cat("\nSorted Data Frame by Name and Score:\n")

print(sorted\_data)

**set-8-1:**

# Set the graphical parameters for an empty plot

plot(1, type = "n", xlab = "X-axis", ylab = "Y-axis",

xlim = c(-10, 10), ylim = c(-10, 10),

main = "Empty Plot with Specified Axes Limits")

# Add grid lines for better visibility (optional)

grid()

**2.** Define a vector of values

values <- 1:12 # A vector containing values from 1 to 12

# Define a vector of dimensions

dimensions <- c(3, 2, 2) # 3 rows, 2 columns, and 2 tables

# Create the array using the values and dimensions

my\_array <- array(values, dim = dimensions)

# Set dimension names

dimnames(my\_array) <- list(

Row = c("Row1", "Row2", "Row3"), # Names for rows

Column = c("Col1", "Col2"), # Names for columns

Table = c("Table1", "Table2") # Names for tables

)

# Print the created array

cat("Created Array:\n")

print(my\_array)

**3.** # Load the airquality dataset

data("airquality")

# (i) Get the Summary Statistics of air quality dataset

summary\_statistics <- summary(airquality)

cat("Summary Statistics of airquality dataset:\n")

print(summary\_statistics)

# (iii) Melt airquality data set and display as a long-format data

library(reshape2)

melted\_airquality <- melt(airquality)

cat("\nMelted airquality dataset (long format):\n")

print(head(melted\_airquality))

# (iv) Melt airquality data and specify month and day as ID variables

melted\_airquality\_id <- melt(airquality, id.vars = c("Month", "Day"))

cat("\nMelted airquality dataset with Month and Day as ID variables:\n")

print(head(melted\_airquality\_id))

# (v) Cast the molten airquality data set with respect to Month and Day features

# (vi) Compute the average of Ozone, Solar.R, Wind, and Temperature per month

cast\_airquality <- dcast(melted\_airquality\_id, Month ~ variable, mean, value.var = "value")

cat("\nAverage of Ozone, Solar.R, Wind, and Temperature per Month:\n")

print(cast\_airquality)

**4.** # Create the data frame based on the given table

df <- 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 the data frame

print(df)

# Create a linear regression model: Sales ~ Spends

model <- lm(Sales ~ Spends, data = df)

# Summarize the model

summary(model)

# Predict the sales for Spend = 13500

new\_spend <- data.frame(Spends = 13500)

predicted\_sales <- predict(model, new\_spend)

# Print the predicted sales

cat("Predicted Sales for Spend = 13500:", predicted\_sales, "\n")

**set-9-1:**

# Load the built-in women dataset

data("women")

# Display the first few rows of the women dataset

cat("Women dataset:\n")

print(head(women))

# Create a factor from the height column

# Let's categorize heights into three levels: Short, Medium, Tall

height\_factor <- cut(women$height,

breaks = c(-Inf, 60, 65, Inf),

labels = c("Short", "Medium", "Tall"))

# Add this factor to the dataset

women$height\_factor <- height\_factor

# Display the updated dataset with the height factor

cat("\nWomen dataset with height factor:\n")

print(women)

# Display the factor

cat("\nHeight factor levels:\n")

print(table(height\_factor))

**2.** # Set a seed for reproducibility

set.seed(123)

# Create a random sample of letters from LETTERS

random\_sample <- sample(LETTERS, size = 20, replace = TRUE)

# Convert the sample into a factor

factor\_sample <- factor(random\_sample)

# Print the factor levels

cat("Factor Levels:\n")

print(levels(factor\_sample))

# Extract the first five levels of the factor

five\_levels <- levels(factor\_sample)[1:5]

cat("\nFive Levels of the Factor:\n")

print(five\_levels)

**3.** # Define two vectors

vector1 <- 1:9 # Vector containing values from 1 to 9

vector2 <- 10:18 # Vector containing values from 10 to 18

# Create two 3x3 matrices from the vectors

matrix1 <- matrix(vector1, nrow = 3, ncol = 3, byrow = TRUE)

matrix2 <- matrix(vector2, nrow = 3, ncol = 3, byrow = TRUE)

# Create an array with the two matrices

my\_array <- array(c(matrix1, matrix2), dim = c(3, 3, 2))

# Print the second row of the second matrix (2nd layer of the array)

cat("Second row of the second matrix:\n")

print(my\_array[2, , 2])

# Print the element in the 3rd row and 3rd column of the first matrix (1st layer of the array)

cat("\nElement in the 3rd row and 3rd column of the first matrix:\n")

print(my\_array[3, 3, 1])

**4.** # Load necessary libraries

library(ggplot2)

library(dplyr)

# Load the Titanic dataset

data("Titanic")

# Convert Titanic data to a data frame

titanic\_data <- as.data.frame(Titanic)

# (a) Draw a bar chart to show details of “Survived” based on passenger Class

ggplot(titanic\_data, aes(x = Class, fill = Survived)) +

geom\_bar(position = "dodge") +

labs(title = "Survival on Titanic by Passenger Class",

x = "Passenger Class",

y = "Count",

fill = "Survived") +

theme\_minimal()

# (b) Modify the above plot based on gender of people who survived

ggplot(titanic\_data, aes(x = Class, fill = interaction(Survived, Sex))) +

geom\_bar(position = "dodge") +

labs(title = "Survival on Titanic by Passenger Class and Gender",

x = "Passenger Class",

y = "Count",

fill = "Survived and Gender") +

theme\_minimal()

# (c) Draw a histogram plot to show distribution of feature “Age”

ggplot(titanic\_data, aes(x = Age)) +

geom\_histogram(binwidth = 5, fill = "skyblue", color = "black", na.rm = TRUE) +

labs(title = "Distribution of Age on Titanic",

x = "Age",

y = "Count") +

theme\_minimal()