# 1. Take user input and display values, print R version

name <- readline(prompt="Enter your name: ")

age <- as.integer(readline(prompt="Enter your age: "))

cat("Name:", name, "\nAge:", age, "\n")

cat("R Version:", R.version.string, "\n")

# 2. Get details of objects in memory

print(ls())

# 3. Sequence, mean, and sum

seq\_20\_50 <- 20:50

mean\_20\_60 <- mean(20:60)

sum\_51\_91 <- sum(51:91)

cat("Sequence from 20 to 50:", seq\_20\_50, "\n")

cat("Mean of 20 to 60:", mean\_20\_60, "\n")

cat("Sum of 51 to 91:", sum\_51\_91, "\n")

# 4. Vector with 10 random integers between -50 and 50

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

print(random\_vector)

# 5. First 10 Fibonacci numbers

fibonacci <- numeric(10)

fibonacci[1:2] <- c(0,1)

for (i in 3:10) fibonacci[i] <- fibonacci[i-1] + fibonacci[i-2]

print(fibonacci)

# 6. Sieve of Eratosthenes for prime numbers up to n

sieve\_of\_eratosthenes <- function(n) {

primes <- rep(TRUE, n)

primes[1] <- FALSE

for (i in 2:sqrt(n)) {

if (primes[i]) {

primes[seq(i^2, n, i)] <- FALSE

}

}

which(primes)

}

n <- as.integer(readline("Enter a number: "))

print(sieve\_of\_eratosthenes(n))

# 7. FizzBuzz

for (i in 1:100) {

if (i %% 3 == 0 & i %% 5 == 0) print("FizzBuzz")

else if (i %% 3 == 0) print("Fizz")

else if (i %% 5 == 0) print("Buzz")

else print(i)

}

# 8. Extracting letters

letters\_lower <- letters[1:10]

letters\_upper <- LETTERS[17:26]

letters\_22\_24 <- LETTERS[22:24]

print(letters\_lower)

print(letters\_upper)

print(letters\_22\_24)

# 9. Find factors of a number

num <- as.integer(readline("Enter a number: "))

factors <- function(n) {

return(which(n %% 1:n == 0))

}

print(factors(num))

# 10. Find max and min value of a vector

vec <- c(10, 20, 30, 5, 40, 15)

cat("Max:", max(vec), " Min:", min(vec), "\n")

# 11. Unique elements of string and vector

string <- "hello world"

unique\_chars <- unique(strsplit(string, "")[[1]])

vector\_nums <- c(1, 2, 2, 3, 4, 4, 5)

unique\_numbers <- unique(vector\_nums)

print(unique\_chars)

print(unique\_numbers)

# 12. Create 3 vectors and combine into a 3x3 matrix

a <- c(1, 2, 3)

b <- c(4, 5, 6)

c <- c(7, 8, 9)

matrix\_3x3 <- cbind(a, b, c)

print(matrix\_3x3)

# 13-22 (previous tasks already included)

# 23. Create an array with four columns, three rows, and two tables

array\_4x3x2 <- array(1:24, dim = c(3, 4, 2))

print(array\_4x3x2)

# 24. Create a 5x3 array of even numbers > 50

even\_array <- matrix(seq(52, 80, by=2), nrow=5, ncol=3)

print(even\_array)

# 25. Create a data frame and perform operations

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

print(exam\_data[c(3,5), c(1,3)])

# (b) Add a new column 'country'

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

print(exam\_data)

# (c) Add new rows

ew\_rows <- data.frame(name = c('Robert', 'Sophia'), score = c(10.5, 9), attempts = c(1, 3), qualify = c('yes', 'no'))

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

print(exam\_data)

# (d) Sort by name and score

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

print(exam\_data)

# (e) Save and read the data frame from a file

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

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

print(read\_exam\_data)

# 26. Use built-in dataset 'airquality', check type, order, and modify

print(is.data.frame(airquality))

ordered\_airquality <- airquality[order(airquality[,1], airquality[,2]), ]

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

print(modified\_airquality)

# 27. Create a factor corresponding to height in the women dataset

height\_factor <- factor(women$height)

print(height\_factor)

# 28. Extract five levels of a factor created from a random sample of LETTERS

letter\_factor <- factor(sample(LETTERS, 10, replace=TRUE))

print(levels(letter\_factor)[1:5])

# 29. Perform EDA on the Iris dataset

data(iris)

# (i) Dimension, structure, summary, and standard deviation

print(dim(iris))

print(str(iris))

print(summary(iris))

print(sapply(iris[,1:4], sd))

# (ii) Mean and standard deviation of features grouped by Species

print(aggregate(. ~ Species, data=iris, FUN=mean))

print(aggregate(. ~ Species, data=iris, FUN=sd))

# (iii) Quantile values of Sepal Width and Sepal Length

print(quantile(iris$Sepal.Width))

print(quantile(iris$Sepal.Length))

# (iv) Create iris1 with a new column categorizing Sepal.Length by quantile

iris1 <- iris

iris1$Sepal.Length.Cate <- cut(iris$Sepal.Length, quantile(iris$Sepal.Length), include.lowest=TRUE)

print(head(iris1))

# (v) Average numerical values by Species and Sepal.Length.Cate

print(aggregate(. ~ Species + Sepal.Length.Cate, data=iris1[,1:5], FUN=mean))

# (vi) Mean value of numerical variables by Species and Sepal.Length.Cate

print(aggregate(. ~ Species + Sepal.Length.Cate, data=iris1[,1:5], FUN=mean))

# (vii) Create a pivot table

library(tidyr)

pivot\_table <- iris1 %>%

group\_by(Species, Sepal.Length.Cate) %>%

summarise(Count = n(), .groups='drop')

print(pivot\_table)

# 30. Train and test logistic regression model on Iris dataset

set.seed(123)

sample\_index <- sample(1:nrow(iris), 0.8\*nrow(iris))

train\_data <- iris[sample\_index, ]

test\_data <- iris[-sample\_index, ]

# Train logistic regression model

model <- glm(Species ~ Petal.Width + Petal.Length, data=train\_data, family=binomial)

pred\_probs <- predict(model, test\_data, type="response")

# Convert probabilities to class predictions

pred\_labels <- ifelse(pred\_probs > 0.5, "versicolor", "setosa") # Adjust as needed

# Create confusion matrix

table(Predicted=pred\_labels, Actual=test\_data$Species)

# 31. Compute mean, median, mode, and find second highest and third lowest values

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

# (i) Mean, Median, Mode

mean\_val <- mean(values)

median\_val <- median(values)

mode\_val <- as.numeric(names(sort(table(values), decreasing=TRUE)[1])) # Mode

cat("Mean:", mean\_val, "Median:", median\_val, "Mode:", mode\_val, "\n")

# (ii) Second highest and third lowest values

sorted\_values <- sort(unique(values))

second\_highest <- sorted\_values[length(sorted\_values) - 1]

third\_lowest <- sorted\_values[3]

cat("2nd Highest:", second\_highest, "3rd Lowest:", third\_lowest, "\n")

# 32. Explore the airquality dataset

data(airquality)

# (i) Compute mean temperature (without using mean function)

sum\_temp <- sum(airquality$Temp, na.rm=TRUE)

count\_temp <- sum(!is.na(airquality$Temp))

mean\_temp <- sum\_temp / count\_temp

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

# (ii) Extract the first five rows

print(head(airquality, 5))

# (iii) Extract all columns except Temp and Wind

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

print(airquality\_subset)

# (iv) Find the coldest day

coldest\_day <- airquality[which.min(airquality$Temp), ]

print(coldest\_day)

# (v) Count days where wind speed > 17 mph

wind\_high\_days <- sum(airquality$Wind > 17, na.rm=TRUE)

cat("Days with Wind Speed > 17 mph:", wind\_high\_days, "\n")

# 33. Data Transformation using Melt and Cast

library(reshape2)

# (i) Summary statistics of airquality

print(summary(airquality))

# (ii) Melt airquality into long format

melted\_airquality <- melt(airquality)

print(head(melted\_airquality))

# (iii) Melt data specifying Month and Day as ID variables

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

print(head(melted\_airquality\_id))

# (iv) Cast molten data with respect to Month and Date

cast\_airquality <- dcast(melted\_airquality\_id, Month + Day ~ variable)

print(head(cast\_airquality))

# (v) Compute average of Ozone, Solar.R, Wind, and Temp per month

average\_per\_month <- dcast(melted\_airquality\_id, Month ~ variable, mean, na.rm=TRUE)

print(average\_per\_month)

# 34. Handle Missing Values and Apply Linear Regression

missing\_values <- colSums(is.na(airquality))

cat("Missing Values:\n", missing\_values, "\n")

# (i) Drop columns if missing values < 10%, else replace with mean

for (col in names(airquality)) {

na\_count <- sum(is.na(airquality[[col]]))

total\_count <- nrow(airquality)

if (na\_count / total\_count < 0.1) {

airquality <- airquality[!is.na(airquality[[col]]), ]

} else {

airquality[[col]][is.na(airquality[[col]])] <- mean(airquality[[col]], na.rm=TRUE)

}

}

# (ii) Apply Linear Regression (Ozone ~ Solar

# 37. Multi Regression Model on ChickWeight

data(ChickWeight)

# (a) Create a multiple regression model with weight ~ Time + Diet

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

summary(model\_chick)

# (b) Predict weight for Time=10 and Diet=1

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

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

cat("Predicted weight for Time=10 and Diet=1:", predicted\_weight, "\n")

# (c) Find the error in model (Residual Standard Error)

cat("Model Residual Error:", summary(model\_chick)$sigma, "\n")

# 38. Titanic Dataset Analysis

data(Titanic)

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

library(ggplot2)

# (a) Bar chart of survivors based on passenger class

ggplot(titanic\_df, aes(x=Class, y=Freq, fill=Survived)) +

geom\_bar(stat="identity", position="dodge") +

labs(title="Survival Count by Passenger Class", x="Class", y="Count")

# (b) Modify the above bar chart based on gender

ggplot(titanic\_df, aes(x=Class, y=Freq, fill=Survived)) +

geom\_bar(stat="identity", position="dodge") +

facet\_wrap(~Sex) +

labs(title="Survival by Class and Gender", x="Class", y="Count")

# (c) Histogram of Age distribution (Titanic dataset does not include Age, so using an alternative dataset)

# If you use a Titanic dataset that includes age, replace "your\_titanic\_data$Age"

# ggplot(your\_titanic\_data, aes(x=Age)) + geom\_histogram(binwidth=5, fill="blue", color="black") + labs(title="Age Distribution", x="Age", y="Frequency")

# 39. Explore USArrests Dataset

data(USArrests)

# (i-a) Summary, feature count, record count, and statistical details

print(summary(USArrests))

cat("Number of Features:", ncol(USArrests), "\n")

cat("Number of Records:", nrow(USArrests), "\n")

# (i-b) State with the highest number of Rape arrests

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

cat("State with highest Rape arrests:", max\_rape\_state, "\n")

# (i-c) States with max & min murder rates

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

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

cat("State with Max Murder Rate:", max\_murder\_state, "\n")

cat("State with Min Murder Rate:", min\_murder\_state, "\n")

# (ii-a) Find correlation between features

print(cor(USArrests))

# (ii-b) States with assault arrests greater than median

above\_median\_assault <- rownames(USArrests[USArrests$Assault > median(USArrests$Assault), ])

print(above\_median\_assault)

# (ii-c) States in the bottom 25% of murder

bottom\_25\_murder <- rownames(USArrests[USArrests$Murder <= quantile(USArrests$Murder, 0.25), ])

print(bottom\_25\_murder)

# (iii-a) Histogram and density plot of murder arrests

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

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

geom\_density(aes(y=..count..), color="red") +

labs(title="Histogram and Density of Murder Arrests", x="Murder Rate", y="Count")

# (iii-b) Scatter plot of murder rate vs urban population, with assault rate color

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

geom\_point(size=3) +

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

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

# (iii-c) Bar graph for murder rate by state

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

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

coord\_flip() +

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

# 40. Sales and Advertising Spend Data

sales\_data <- 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(sales\_data)

# (b) Regression Model for Sales ~ Spends

sales\_model <- lm(Sales ~ Spends, data=sales\_data)

summary(sales\_model)

# (c) Predict Sales if Spend=13500

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

predicted\_sales <- predict(sales\_model, new\_spend)

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