Exercise 1

**Problem 1: Understanding Variables and Data Types**

* Create variables to store the following information: the name of a city, its population (in millions), and whether it is a capital city (TRUE/FALSE). Display these variables.
  + *Hint 1*: Create a variable using the assignment operator <-, like this: city\_name <- "Accra".
  + *Hint 2*: To display a variable, use print(), like print(city\_name).
  + *Hint 3*: For numeric values, use population <- 2.5 and for logical values, use is\_capital <- TRUE.
* Write a function that takes the population and returns a message indicating whether the city is highly populated (over 5 million) or moderately populated.
  + *Hint 1*: Define a function using function(), like this: my\_function <- function(input) { ... }.
  + *Hint 2*: Use an if statement to check if the population is greater than 5: if (input > 5) { return("Highly populated") } else { return("Moderately populated") }.

**Problem 2: Converting Data with Functions**

* Define a function that takes a vector of temperatures (in Celsius) and converts them to Fahrenheit using the formula F = C \* 9/5 + 32.
  + *Hint 1*: Define a function as convert\_to\_fahrenheit <- function(celsius\_vector) { ... }.
  + *Hint 2*: Use the formula inside the function to convert each element, like fahrenheit\_vector <- celsius\_vector \* 9/5 + 32.
  + *Hint 3*: Use sapply() to apply your function to a vector, e.g., sapply(c(22, 25, 30), convert\_to\_fahrenheit).
* Apply the function to c(22, 25, 30, 28, 27) and display the original and converted temperatures side by side.
  + *Hint*: Store the result in a variable, like converted\_temperatures <- sapply(...), and use data.frame() to combine the original and converted vectors for display.

**Problem 3: Subsetting Vectors**

* Create a vector of student scores: c(85, 78, 92, 74, 88, 90, 67, 80).
  + *Hint*: Create a vector using scores <- c(85, 78, 92, ...).
* Extract the scores that are greater than 80 using subsetting. Store the result in a new variable called high\_scores.
  + *Hint 1*: Use square brackets for subsetting: high\_scores <- scores[scores > 80].
  + *Hint 2*: Inside the brackets, place the condition that each element should satisfy.
* Write a function that takes the high\_scores vector and returns the number of students who scored above 80.
  + *Hint 1*: Use the length() function inside your function to count the elements: length(high\_scores).
  + *Hint 2*: Define the function as count\_high\_scores <- function(scores) { ... }.

**Problem 4: Using Functions on Vectors**

* Create a vector of ages: c(23, 45, 34, 28, 31, 38, 26, 42).
  + *Hint*: Use ages <- c(23, 45, 34, ...).
* Calculate the mean and median of the ages using mean() and median().
  + *Hint*: Use mean(ages) and median(ages).
* Write a function to classify the ages into "Young" (less than 30), "Middle-aged" (30 to 40), and "Senior" (above 40). Use sapply() to apply this function to the age vector.
  + *Hint 1*: Define the function as classify\_age <- function(age) { ... }.
  + *Hint 2*: Inside the function, use if statements to return the appropriate category based on the age.
  + *Hint 3*: Apply the function using sapply(ages, classify\_age).

**Problem 5: Working with Dates Using lubridate**

* Create a vector of date strings: c("2024-01-01", "2024-04-15", "2024-07-20", "2024-10-31").
  + *Hint*: Use dates <- c("2024-01-01", "2024-04-15", ...).
* Convert these strings into date objects using the ymd() function from the lubridate package.
  + *Hint 1*: Install lubridate using install.packages("lubridate") and load it with library(lubridate).
  + *Hint 2*: Use ymd(dates) to convert the strings into date objects.
* Write a function that calculates the number of days from each date to today’s date using Sys.Date().
  + *Hint 1*: Use difftime() to calculate the difference in days: difftime(Sys.Date(), date, units = "days").
  + *Hint 2*: Inside the function, use sapply() to apply the difference calculation to each date.

**Problem 6: Manipulating Strings Using stringr**

* Create a vector of names: c("Anna Smith", "John Doe", "Emily Brown", "Michael Johnson").
  + *Hint*: Use names <- c("Anna Smith", "John Doe", ...).
* Write a function that extracts the first name from each full name using str\_split() from stringr.
  + *Hint 1*: Use str\_split() to split each name at the space: str\_split(name, " ").
  + *Hint 2*: Use sapply() to extract the first part from the split result.
* Write another function to check if a name contains the letter "o" using str\_detect() and filter out names that do not contain this letter.
  + *Hint 1*: Use str\_detect(name, "o") to check for the letter.
  + *Hint 2*: Use names[str\_detect(names, "o")] to filter the vector.

**Problem 7: Normalizing Data**

* Create a vector of income values (in thousands): c(45, 60, 55, 72, 80, 39, 91, 52).
  + *Hint*: Use incomes <- c(45, 60, 55, ...).
* Write a function to normalize these values so that they lie between 0 and 1 using the formula (x - min(x)) / (max(x) - min(x)).
  + *Hint 1*: Define the function as normalize <- function(x) { ... }.
  + *Hint 2*: Use min() and max() functions inside the formula.
  + *Hint 3*: Apply the function to the vector using sapply() or directly on the vector.
* Write a function that classifies the normalized incomes into "Low", "Medium", and "High" income categories.
  + *Hint*: Use ifelse() to categorize each value based on custom thresholds.

**Problem 8: Combining String and Date Operations**

* Create a vector of strings representing dates and times of events: c("2024-01-01 10:00", "2024-04-15 15:30", "2024-07-20 09:00", "2024-10-31 12:45").
  + *Hint*: Use event\_times <- c("2024-01-01 10:00", ...).
* Convert these strings into datetime objects using ymd\_hm() from lubridate.
  + *Hint*: Use ymd\_hm(event\_times) for conversion.
* Extract the hour and minute using hour() and minute() functions.
  + *Hint*: Use hour(datetime\_object) to get the hour.
  + *Hint*: Use sapply() to apply hour() and minute() to the vector of datetime objects.

**Problem 9: Data Analysis with Vectors**

* Assume you have a vector of daily rainfall (in mm) for a week: c(12.5, 0, 0, 7.8, 2.3, 0, 15.6).
  + *Hint*: Use rainfall <- c(12.5, 0, 0, ...).
* Write a function to calculate the total rainfall for the week.
  + *Hint*: Use sum(rainfall) inside your function.
* Write another function that counts the days with no rainfall using length() and a subset condition.
  + *Hint*: Subset days with rainfall == 0 and use length().

**Problem 10: Vector Modification and Aggregation**

* Create a vector of sales figures: c(100, 200, 150, 80, 90, 300, 50).
  + *Hint*: Use sales <- c(100, 200, ...).
* Write a function that sets a minimum sales threshold of 100 units using ifelse().
  + *Hint*: Use ifelse(sales < 100, 100, sales) to modify values.
* Calculate the total sales after modification using sum().
  + *Hint*: Use sum() on the modified sales vector.