Exercise : 02

DATA TYPE - R OBJECT AND ATTRIBUTES DATATYPE VECTOR AND LIST

1. Basic Data Types in R

Numeric

```
num_var <- 10.5
print(num_var)
class(num_var)
```

Integer

```
int_var <- 10L
print(int_var)
class(int_var)
```

Character

```
char_var <- "Hello R"
print(char_var)
```

Logical

```
log_var <- TRUE
print(log_var)
class(log_var)
```

Complex

```
comp_var <- 3 + 2i
print(comp_var)
class(comp_var)
```

2. R Objects and Attributes

Create a vector with attributes

```
vec <- c(1, 2, 3, 4)
attr(vec, "description") <- "This is a numeric vector"
print(vec)
attributes(vec)
```

Create a matrix with attributes

```
mat <- matrix(1:9, nrow=3)
dimnames(mat) <- list(c("Row1", "Row2", "Row3"), c("Col1", "Col2",
"Col3"))
print(mat)
attributes(mat)
```

3. Vectors in R

Numeric vector

```
num_vec <- c(1, 2, 3, 4, 5)
print(num_vec)
```

Character vector

```
char_vec <- c("apple", "banana", "cherry")
print(char_vec)
```

Logical vector

```
log_vec <- c(TRUE, FALSE, TRUE, FALSE)
print(log_vec)
```

Operations on vectors

```
vec1 <- c(1, 2, 3)
vec2 <- c(4, 5, 6)

sum_vec <- vec1 + vec2 # Element-wise addition
print(sum_vec)
```

4. Lists in R

Creating a list with different data types

```
my_list <- list(name = "Dhayanidhi", age = 20, marks = c(85, 90, 95))
print(my_list)
```

Accessing elements in a list

```
print(my_list$name) # Access by name
print(my_list[[2]]) # Access by index
```

Modifying a list

```
my_list$age <- 21
print(my_list)
```

Exercise : 03

Data Types – Data Frame, Matrices, Factors, and Functions

Data Frames

```
df <- data.frame(Name = c("Alice", "Bob", "Charlie"), Age = c(25, 30,
35), Score = c(90, 85, 88))
print(df)
```

Matrices

```
a <- matrix(1:12, nrow = 3, ncol = 4)
b <- matrix(11:22, nrow = 3, ncol = 4)
```

```
print(a+b)
print(a-b)
print(a*b)
print(a/b)
```

Factors

```
gender <- factor(c("Male", "Female", "Male", "Female"))
print(gender)
levels(gender)
```

Functions

```
square <- function(x) {
  return(x^2)
}
print(square(5))

my_function <- function(x, y) {
  return(x + y)
}
print(my_function(3, 5))
```

Exercise : 04

EXPRESSION AND LOGICAL STATEMENT IN R

Logical Expressions

```
a <- 10
b <- 20
print(a > b)
print(a == b)
print(a != b)
print(a < b)
```

Conditional Statements

```
x <- 5
if (x > 0) {
  print("Positive number")
} else {
  print("Non-positive number")
}
```

Looping with Logical Conditions

```
for (i in 1:5) {
  if (i %% 2 == 0) {
    print(paste(i, "is even"))
  } else {
    print(paste(i, "is odd"))
  }
}
```

Exercise : 05

SUBSETTING OF LIST , MATRIX AND DATA FRAME

Subsetting Lists

```
new_list <- list(animal = "Tiger", count = 5, colors = c("Orange",
"Black"))
print(new_list$animal) # By name
print(new_list[[2]]) # By index
print(new_list[1:2]) # Subsetting multiple elements
```

Subsetting Matrices

```
new_mat <- matrix(1:12, nrow=3, byrow=TRUE)
print(new_mat[1,])  # First row
print(new_mat[,2])  # Second column
print(new_mat[2,3]) # Specific element
```

Subsetting Data Frames

```
new_df <- data.frame(City = c("Chennai", "Mumbai"), Population =
c(8000000, 20000000), Area = c(426, 603))
print(new_df$City)    # Selecting a column
print(new_df[1,])     # Selecting a row
print(new_df[1:2, 2]) # Selecting specific rows and column
```

Exercise : 06

DATA FRAME FUNCTION ON INBUILT DATASET

Working with Inbuilt Dataset

```
data(iris)  # Loading iris dataset
print(head(iris))  # Display first six rows
```

Summary Statistics

```
print(summary(iris))  # Summary statistics of the dataset
```

Selecting Specific Columns

```
print(iris$Species)  # Selecting a specific column
print(iris[, c("Sepal.Length", "Sepal.Width")])  # Selecting multiple
columns
```

Filtering Data

```
filtered_data <- subset(iris, Species == "setosa")
print(head(filtered_data))
```

Aggregating Data

```
aggr_data <- aggregate(Sepal.Length ~ Species, data = iris, FUN =
mean)
print(aggr_data)
```

Adding a New Column

```
iris$Sepal.Area <- iris$Sepal.Length * iris$Sepal.Width
print(head(iris))
```

Removing a Column

```
iris <- subset(iris, select = -Sepal.Area)
print(head(iris))
```

Exercise : 07

DPLYR FUNCTION ON RETAIL DATASET

```
library(dplyr)
library(ggplot2)
```

Using the `diamonds` Dataset from ggplot2

```
data("diamonds")
diamonds_sample <- diamonds %>% select(cut, color, clarity, price,
carat)
print(head(diamonds_sample))
```

Selecting Specific Columns

```
selected_data <- select(diamonds_sample, cut, price)
print(head(selected_data))
```

Filtering Data

```
filtered_data <- filter(diamonds_sample, price > 5000)
print(head(filtered_data))
```

Arranging Data

```
sorted_data <- arrange(diamonds_sample, desc(price))
print(head(sorted_data))
```

Mutating Data (Adding a New Column)

```
diamonds_sample <- mutate(diamonds_sample, price_per_carat = price /
carat)
print(head(diamonds_sample))
```

Summarizing Data

```
sales_summary <- diamonds_sample %>% summarise(Average_Price =
mean(price))
print(sales_summary)
```

Grouping and Summarizing Data

```
grouped_summary <- diamonds_sample %>% group_by(cut) %>%
summarise(Total_Price = sum(price))
print(grouped_summary)
```

Exercise : 08

DPLYR FUNCTION ON BANKING DATASET

Loading Required Library

```
library(dplyr)
```

Creating a Sample Banking Dataset

```
banking_data <- data.frame(
  Customer_ID = 1:6,
  Account_Type = c("Savings", "Checking", "Savings", "Loan",
"Checking", "Loan"),
  Balance = c(5000, 2000, 7000, -1000, 1500, -500),
  Transactions = c(10, 25, 12, 5, 20, 8)
)
print(head(banking_data))
```

Selecting Specific Columns

```
selected_data <- select(banking_data, Customer_ID, Balance)
print(head(selected_data))
```

Filtering Data

```
filtered_data <- filter(banking_data, Balance > 1000)
print(head(filtered_data))
```

Arranging Data

```
sorted_data <- arrange(banking_data, desc(Balance))
print(head(sorted_data))
```

Mutating Data (Adding a New Column)

```
banking_data <- mutate(banking_data, Avg_Transaction = Balance /
Transactions)
print(head(banking_data))
```

Summarizing Data

```
sales_summary <- banking_data %>% summarise(Average_Balance =
mean(Balance))
print(sales_summary)
```

Grouping and Summarizing Data

```
grouped_summary <- banking_data %>% group_by(Account_Type) %>%
summarise(Total_Balance = sum(Balance))
print(grouped_summary)
```

Exercise : 09

BASIC PLOTTING WITH R

```
### Loading Required Library
library(ggplot2)

### Loading the mtcars Dataset
data("mtcars")
print(head(mtcars))

# 1. Histogram of MPG (Miles Per Gallon)
hist_plot <- ggplot(mtcars, aes(x = mpg)) +
  geom_histogram(binwidth = 2, fill = "blue", color = "black") +
  labs(title = "Histogram of MPG", x = "Miles Per Gallon (mpg)", y =
"Count")

# 2. Scatter Plot of Horsepower vs MPG
scatter_plot <- ggplot(mtcars, aes(x = hp, y = mpg)) +
  geom_point(color = "red") +
  labs(title = "Horsepower vs MPG", x = "Horsepower (hp)", y = "Miles
Per Gallon (mpg)")

# 3. Bar Chart of Cylinders Count
bar_chart <- ggplot(mtcars, aes(x = factor(cyl))) +
  geom_bar(fill = "green") +
  labs(title = "Cylinder Count Distribution", x = "Number of
Cylinders", y = "Count")

# 4. Box Plot of MPG by Number of Cylinders
box_plot <- ggplot(mtcars, aes(x = factor(cyl), y = mpg)) +
  geom_boxplot(fill = "purple") +
  labs(title = "MPG Distribution by Cylinders", x = "Number of
Cylinders", y = "Miles Per Gallon (mpg)")

# 5. Line Plot of Weight vs MPG (Trend)
line_plot <- ggplot(mtcars, aes(x = wt, y = mpg)) +
  geom_line(color = "orange") +
  labs(title = "Weight vs MPG Trend", x = "Weight (wt)", y = "Miles
Per Gallon (mpg)")

# Printing the Plots
print(hist_plot)
print(scatter_plot)
print(bar_chart)
print(box_plot)
print(line_plot)
```

Exercise : 10

GGPLOTS WITH R

```
### Loading Required Libraries
library(ggplot2)
library(palmerpenguins)

### Loading the Penguins Dataset
data("penguins")
print(head(penguins))

# 1. Histogram of Flipper Length
hist_plot <- ggplot(penguins, aes(x = flipper_length_mm)) +
  geom_histogram(binwidth = 5, fill = "blue", color = "black") +
  labs(title = "Histogram of Flipper Length", x = "Flipper Length
(mm)", y = "Count")

# 2. Scatter Plot of Bill Length vs Bill Depth
scatter_plot <- ggplot(penguins, aes(x = bill_length_mm, y =
bill_depth_mm, color = species)) +
  geom_point() +
  labs(title = "Bill Length vs Bill Depth", x = "Bill Length (mm)", y
= "Bill Depth (mm)")

# 3. Bar Chart of Species Count
bar_chart <- ggplot(penguins, aes(x = species, fill = species)) +
  geom_bar() +
  labs(title = "Species Count", x = "Penguin Species", y = "Count")

# 4. Box Plot of Body Mass by Species
box_plot <- ggplot(penguins, aes(x = species, y = body_mass_g, fill =
species)) +
  geom_boxplot() +
  labs(title = "Body Mass Distribution by Species", x = "Penguin
Species", y = "Body Mass (g)")

# 5. Line Plot of Average Flipper Length by Species
line_plot <- ggplot(penguins, aes(x = species, y = flipper_length_mm,
color = species, group = 1)) +
  stat_summary(fun = mean, geom = "point", size = 4) +  # Add points
for mean values
  stat_summary(fun = mean, geom = "line", size = 1) +   # Connect
points with a line
  labs(title = "Average Flipper Length by Species", x = "Penguin
Species", y = "Flipper Length (mm)")

# Printing the Plots
print(hist_plot)
print(scatter_plot)
print(bar_chart)
print(box_plot)
print(line_plot)
```