Questions on Module 6

1 Consider the following data frame given below **[DEC2022 10 marks]**

Subject	Class	Marks
1	1	56
2	2	75
3	1	48
4	2	69
5	1	84
6	2	53

i. Create a subset of subjects less than 4 by using subset() function and demonstrate the output.

```
# Create a sample data frame with the given marks values
data <- data.frame(
  subject = c(1, 2, 3, 4, 5, 6),
  class = c(1, 2, 1, 2, 1, 2),
  marks = c(56, 75, 48, 69, 84, 53)
)

# Display the original data frame
print("Original Data Frame:")
print(data)

# Create a subset where subject is less than 4
subset_data <- subset(data, subject < 4)

# Display the subset data frame
print("Subset Data Frame (subject < 4):")
print(subset_data)
```

```
[1] "Original Data Frame:"
subject class marks
1       1     1    56
2       2     2    75
3       3     1    48
4       4     2    69
5       5     1    84
6       6     2    53
[1] "Subset Data Frame (subject < 4):"
subject class marks
1       1     1    56
2       2     2    75
3       3     1    48
```

This code creates a data frame with the specified marks values and then creates a subset where the "subject" is less than 4.

ii. Create a subset where the subject column is less than 3 and the class equals to 2 by using [ ] brackets and demonstrate the output.

```
# Create a sample data frame with the given marks values
data <- data.frame(
  subject = c(1, 2, 3, 4, 5, 6),
  class = c(1, 2, 1, 2, 1, 2),
  marks = c(56, 75, 48, 69, 84, 53)
)

# Display the original data frame
print("Original Data Frame:")
print(data)

# Create a subset where subject is less than 3 and class equals to 2
subset_data <- data[data$subject < 3 & data$class == 2, ]

# Display the subset data frame
print("Subset Data Frame (subject < 3 and class == 2):")
print(subset_data)
```

```
[1] "Original Data Frame:"
subject class marks
1       1     1    56
2       2     2    75
3       3     1    48
4       4     2    69
5       5     1    84
6       6     2    53
[1] "Subset Data Frame (subject < 3 and class == 2):"
subject class marks
2       2     2    75
```

In this example, data$subject < 3 checks if the "subject" is less than 3, and data$class == 2 checks if the "class" is equal to 2. The resulting subset includes rows where both conditions are true. The subset is created using square brackets [].

2 The data analyst of Argon technology Mr. John needs to enter the salaries of 10 employees in R. The Salaries of the employees are given in the following table: **[Dec 2024, 10 marks]**

Sr. No	Name Of Employee	Salaries
1	Vivek	21000
2	Karan	55000
3	James	67000
4	Soham	50000
5	Renu	54000
6	Farah	40000
7	Hetal	30000
8	Mary	70000
9	Ganesh	20000
10	Krish2	15000

i. Which R commands will Mr. John use to enter these values? Demonstrate the output.

```
# Create a data frame with the given records
employee_data <- data.frame(
 sr_number = 1:10,
 name = c("Vivek", "Karan", "James", "Soham", "Renu", "Farah", "Hetal", "Mary",
"Ganesh", "Krish")
)

print("Employee Dataset:")
print(employee_data)

salary <- c(21000, 55000, 67000, 50000, 54000, 40000, 30000, 70000, 20000, 15000)

employee_data$salary <- salary

# Display the dataset
print("Employee Dataset:")
print(employee_data)
```

```
[1] "Employee Dataset:"
   sr_number   name salary
1          1  Vivek  21000
2          2  Karan  55000
3          3  James  67000
4          4  Soham  50000
5          5   Renu  54000
6          6  Farah  40000
7          7  Hetal  30000
8          8   Mary  70000
9          9 Ganesh  20000
10        10  Krish  15000
```

ii. Now Mr. John wants to add the salaries of 5 new employees in the existing table, which commands he will use to join datasets with new values in R. Demonstrate the output.

```
# Create a data frame with the salaries of 5 new employees
new_employees <- data.frame(
 sr_number = 11:15,
 name = c("Amit", "Neha", "Rahul", "Sara", "Rohit"),
 salary = c(60000, 45000, 58000, 52000, 48000)
)

# Join the new salaries with the existing dataset
combined_data <- rbind(employee_data, new_employees)

# Display the combined dataset
print("Combined Employee Dataset:")
print(combined_data)
```

In this demonstration, the rbind() function is used to combine the existing dataset (employee_data) with the data frame of salaries for 5 new employees (new_employees). The result is a combined dataset (combined_data) with the salaries of all 15 employees.

```
[1] "Combined Employee Dataset:"
   sr_number   name salary
1          1  Vivek  21000
2          2  Karan  55000
3          3  James  67000
4          4  Soham  50000
5          5   Renu  54000
6          6  Farah  40000
7          7  Hetal  30000
8          8   Mary  70000
9          9 Ganesh  20000
10        10  Krish  15000
11        11   Amit  60000
12        12   Neha  45000
13        13  Rahul  58000
14        14   Sara  52000
15        15  Rohit  48000
```

3 i. Write the script to sort the values contained in the following vector in ascending order and descending order: (23, 45, 10, 34, 89, 20, 67, 99). Demonstrate the output.

ii. Name and explain the operators used to form data subsets in R. **[Dec 2022, 10 marks]**

```
# Define the vector
vector <- c(23, 45, 10, 34, 89, 20, 67, 99)

# Sort in ascending order
ascending_order <- sort(vector)

# Sort in descending order
descending_order <- sort(vector, decreasing = TRUE)

# Display the results
ascending_order
descending_order
```

```
> ascending_order
[1] 10 20 23 34 45 67 89 99

> descending_order
[1] 99 89 67 45 34 23 20 10
```

ii) Name and explain the operators used to form data subsets in R

subset() function used to filter data frames or matrices based on conditions.

Example: subset(dataframe, column_name > 10)
Above code will filter rows where the column_name is greater than 10.

V <- c(1,2,3,4,5,6)
subset(V, V<4)

Sample Output
[1] 1 2 3

[ ] brackets can also be used to create subset of the data

Name <- c("N1","N2","N3", "N4")
Marks <- c (50, 40 , 35, 20)

Df= data.frame(Name, Marks)

Df [Df$Marks < 40,]

4 Consider the following data frame given below: **[May 2023, 10 Marks] [May 2024, 10 Marks]**

Course	Id	Class	Marks
1	11	1	56
2	12	2	75
3	13	1	48
4	14	2	69
5	15	1	84
6	16	2	53

i. Create a subset of course less than 3 by using [ ] brackets and demonstrate the output.

# Creating the data frame with the given information
course_data <- data.frame(
course = c(1, 2, 3, 4, 5, 6),

```
 id = c(11, 12, 13, 14, 15, 16),
 class = c(1, 2, 1, 2, 1, 2),
 marks = c(56, 75, 48, 69, 84, 53)
)

# Displaying the data frame
print("Course Data Frame:")
print(course_data)

# Subset using []
subset_course_less_than_3 <- course_data [ course_data$course < 3, ]

# Display the subset
print("Subset of Course less than 3 using [] brackets:")
print(subset_course_less_than_3)
```

```
[1] "Course Data Frame:"
  course id class marks
1      1 11     1    56
2      2 12     2    75
3      3 13     1    48
4      4 14     2    69
5      5 15     1    84
6      6 16     2    53
[1] "Subset of Course less than 3 using [] brackets:"
  course id class marks
1      1 11     1    56
2      2 12     2    75
```

Here square brackets are [] are used to select rows where the course column is less than 3

ii. Create a subset where the course column is less than 3 or the class equals to 2 by using subset() function and demonstrate the output.

```
# Subset using subset()
subset_course_class_condition <- subset(course_data, course < 3 | class == 2)

# Display the subset
print("Subset where course < 3 or class == 2 using subset():")
print(subset_course_class_condition)
```

```
[1] "Subset where course < 3 or class == 2 using subset():"
  course id class marks
1      1 11     1    56
2      2 12     2    75
4      4 14     2    69
6      6 16     2    53
```

Used the subset() function to select rows where either the "course" column is less than 3 or the "class" column equals 2.

5 i. The following table shows the number of units of different products sold on different days: **[May 2023, 10 Marks]**

Product	Monday	Tuesday	Wednesday	Thursday	Friday
Bread	12	3	5	11	9
Milk	21	27	18	20	15
Cola Cans	10	1	33	6	12
Chocolate Bars	6	7	4	13	12
Detergent	5	8	12	20	23

Create five sample numeric vectors from this data.

```
# Create a data frame for the given sales data
sales_data <- data.frame(
  product = c("bread", "milk", "cola cans", "chocolate bars", "detergent"),
  monday = c(12, 21, 10, 6, 5),
  tuesday = c(3, 27, 1, 7, 8),
  wednesday = c(5, 18, 33, 4, 12),
  thursday = c(11, 20, 6, 13, 20),
  friday = c(9, 15, 12, 12, 23)
)

# Display the sales data table
print("Sales Data Table:")
print(sales_data)

# Create five sample numeric vectors
sample_vector1 <- sales_data[sample(1:nrow(sales_data)), "monday"]
sample_vector2 <- sales_data[sample(1:nrow(sales_data)), "tuesday"]
sample_vector3 <- sales_data[sample(1:nrow(sales_data)), "wednesday"]
sample_vector4 <- sales_data[sample(1:nrow(sales_data)), "thursday"]

sample_vector5 <- sales_data[sample(1:nrow(sales_data)), "friday"]
```

```
# Display the sample vectors
print("Sample Numeric Vectors:")
print(sample_vector1)
print(sample_vector2)
print(sample_vector3)
print(sample_vector4)
print(sample_vector5)
```

```
[1]"Sales Data Table:"
product monday tuesday wednesdaythursday friday
1          bread     12       3         5       11       9
2           milk     21      27        18       20      15
3      cola cans     10       1        33        6      12
4 chocolate bars      6       7         4       13      12
5      detergent      5       8        12       20      23
[1] "Sample Numeric Vectors:"
[1]  5 10 12 21  6
[1]  8  1 27  3  7
[1] 12 33  5 18  4
[1] 13  620 20 11
[1] 15  9 12 23 12
```

In the given sales data table, each row represents a different product, and each column from Monday to Friday represents the number of units sold for that product on each respective day. The sample numeric vectors are randomly selected columns from this table, representing the sales data for a particular day across all products. These vectors can be used for further analysis, such as calculating daily averages or comparing the sales performance of different products on a specific day.

ii. Name and explain the operators used to form data subsets in R

In R, the primary operators and functions for forming data subsets include:

1. Square Brackets [] Operator:

- Syntax: data[rows, columns]
- Explanation: Used to subset data frames or matrices based on specific row and column indices.

2. Logical Operators (&, |, !):

- Syntax: data[logical_condition,]
- Explanation: Logical operators are employed to create conditions for subsetting data based on specific criteria.

3. Subset() Function:

- Syntax: subset(data, logical_condition, select = c(columns))
- Explanation: The subset() function is designed for concise subsetting of data frames using logical conditions.

4. %in% Operator:

- Syntax: data[data$column %in% c(values),]
- Explanation: Checks if elements in a column are present in a specified set of values, commonly used for categorical variables.

5. Double Square Brackets [[]] and $ Operator for Extracting Columns:

- Syntax: data[[column_name]] or data$column_name
- Explanation: Used to extract a specific column from a data frame.

These operators and functions offer flexibility in manipulating and analyzing data, allowing users to efficiently subset datasets based on various conditions or criteria.

6 **i. Create a data frame from the following 4 vectors and demonstrate the output:**
emp_id = c(1:5)
emp_name = c("Rick", "Dan", "Michelle", "Ryan", "Gary")
start_date = c("2012-01-01", "2013-09-23", "2014-11-15", "2014-05-11", "2015-03-27")
salary = c(60000, 45000,75000, 84000, 20000)

ii. Display structure and summary of the above data frame
iii. Extract the emp_name and salary columns from the above data frame.
iv. Extract the employee details whose salary is less than or equal to 60000.

[Dec 2023, 10 Marks]

 # Step i: Create a data frame from the given vectors

```
emp_id <- c(1:5)
emp_name <- c("Rick", "Dan", "Michelle", "Ryan", "Gary")
start_date <- as.Date(c("2012-01-01", "2013-09-23", "2014-11-15", "2014-05-11", "2015-03-27"))
salary <- c(60000, 45000, 75000, 84000, 20000)

# Create the data frame
employee_data <- data.frame(emp_id, emp_name, start_date, salary)
```

Step ii: Display structure and summary of the data frame

```
str(employee_data)
summary(employee_data)
```

```
> # Step ii: Display structure and summary of the data frame
> str(employee_data)
'data.frame':   5 obs. of  4 variables:
 $ emp_id    : int  1 2 3 4 5
 $ emp_name  : chr  "Rick" "Dan" "Michelle" "Ryan" ...
 $ start_date: Date, format: "2012-01-01" "2013-09-23" ...
 $ salary    : num  60000 45000 75000 84000 20000
```

 # Step iii: Extract the emp_name and salary columns

```
employee_data[ , c("emp_name", "salary")]
```

```
> # Step iii: Extract the emp_name and salary colum
> employee_data[, c("emp_name", "salary")]
  emp_name salary
1     Rick  60000
2      Dan  45000
3 Michelle  75000
4     Ryan  84000
5     Gary  20000
```

v. Extract the employee details whose salary is less than or equal to 60000.

Step iv: Extract employee details whose salary is less than or equal to 60000

employee_data[employee_data$salary <= 60000,]

```
> # Step iv: Extract employee details whose salary is less than or equal to 60000
> employee_data[employee_data$salary <= 60000, ]
  emp_id emp_name start_date salary
1      1     Rick 2012-01-01  60000
2      2      Dan 2013-09-23  45000
5      5     Gary 2015-03-27  20000
```

7 List and explain various functions that allow users to handle data in R workspace with appropriate examples. **[Dec 2023, 10 Marks]**

Solution

R provides a range of functions that allow users to handle data effectively in the workspace. Here's an explanation of some of these functions with appropriate examples:

1. Vectors

A vector is the most basic data type in R and represents a sequence of data elements of the same type (numeric, character, or logical). You can create and manipulate vectors using various functions**.**

```
# Creating a numeric vector
num_vec <- c(1, 2, 3, 4, 5)
print(num_vec)

# Creating a character vector
char_vec <- c("apple", "banana", "cherry")
print(char_vec)

# Accessing elements of a vector
num_vec[2]  # Returns the second element of num_vec
```

A data frame is a two-dimensional data structure in R where each column can contain data of different types (e.g., numeric, character, logical). It's one of the most commonly used data structures in R for handling datasets.

```
# Creating a data frame
df <- data.frame(
  Name = c("Alice", "Bob", "Charlie"),
  Age = c(25, 30, 35),
  Salary = c(50000, 60000, 70000)
)
print(df)

# Accessing specific elements in a data frame
df$Age  # Accesses the Age column
df[1, 2]  # Accesses the element at row 1, column 2
```

The subset() function in R allows you to select specific rows and columns from a data frame or matrix based on conditions.

```
# Using subset to select rows where Age is greater than 28
 subset(df, Age > 28)
```

The sort() function is used to sort a vector or data frame in ascending or descending order.

```
# Sorting a numeric vector
sorted_vec <- sort(c(5, 2, 8, 1, 3))
print(sorted_vec)

# Sorting in descending order
sorted_vec_desc <- sort(c(5, 2, 8, 1, 3), decreasing = TRUE)
print(sorted_vec_desc)

# Sorting a column in a data frame
sorted_df <- df[order(df$Salary), ]  # Sorts the data frame by Salary in ascending order
print(sorted_df)
```

merge() function is used to merge the data contained in different data frames on the basis of columns and rows.
It combines the data of two frames on the basis of the existence of a common column between the two. The merge() function takes the following arguments.

x – specifies a data frame
y – specifies a data frame
by, by.x, by.y specifies the names of the columns common in both x and y

all, all.x, all.y - Specify logical values for the type of merge. The default value is all FALSE.

Example

```
course_data <- data.frame (
 course = c(1, 2, 3, 4, 5, 6),
 id = c(11, 12, 13, 14, 15, 16),
 class = c(1, 2, 1, 2, 1, 2),
)

marks_data <- data.frame (

 id = c(11, 12, 13, 14, 15, 16),
 marks = c(56, 75, 48, 69, 84, 53)   )

merge(course_data, marks_data, "id")
```

cbind() function is used to add columns of datasets having an equal set and identical order of rows.

It is used to bind the column names of two datasets. It helps in restricting the number of columns to be included in new datasets

Example

```
course_data <- data.frame (
 course = c(1, 2, 3, 4, 5, 6),
class = c(1, 2, 1, 2, 1, 2),
)

marks_data <- data.frame (
 id = c(11, 12, 13, 14, 15, 16),
 marks = c(56, 75, 48, 69, 84, 53)   )

cbind(course_data[ ,c(1,2,3)], marks_data[,c(1))
```

rbind() function is used to add rows in datasets having an equal number of columns

```
data1 <- data.frame( id = c(1,2,3,4,5),
                     name = c("Ann", "John", "Ben", "jim","Tim")   )
data2 <- data.frame (id = c(6,7,8) ,
                     name = c("May", "Jenny", "carry") )
rbind(data1, data2)
```

8 i. What are the advantages of using functions over scripts?
ii. Suppose you have two datasets A and B,
Dataset A has the following data 6 7 8 9
Dataset B has the following data 1 2 4 5
Which function is used to combine the data from both datasets into dataset C.
Demonstrate the function with the input values and write the output.
[Dec 2023, 10 Marks]

Solution;

In R, both functions and scripts can be used to execute code. However, functions offer several advantages over scripts, especially when it comes to reusability, readability, and maintainability.
Functions: Once a function is defined, it can be reused multiple times throughout the script or in other scripts by simply calling it with appropriate arguments. This prevents redundancy and avoids having to write the same code repeatedly.
Scripts: Scripts are typically a collection of commands that run sequentially, but if you need the same block of code again, you'd have to rewrite or copy-paste it.

Using Functions in R provide following two main advantages:

1. They can work with variable inputs ie with different values
2. They return the result as an object, this result can further be used as an input for another function.

ii.

```
# Dataset A
A <- c(6, 7, 8, 9)

# Dataset B
B <- c(1, 2, 4, 5)

# Combine A and B
C <- c(A, B)
print(C)
```

Output:

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

To combine two datasets (vectors, matrices, or data frames) into one in R, you can use functions like `c()` for vectors or `rbind()` and `cbind()` for matrices and data frames.

9 **Describe applications of data visualization. [May 2024, 10 Marks]**

Visualization is a pictorial or visual representation technique. Anything that is represented in pictorial or graphical form, with the help of diagrams, charts, pictures.
Data visualization is a pictorial or visual representation of data with the help of visual aids such as graphs, bars, histograms, tables, pie charts, mind maps etc.
Visualization is an excellent medium to analyse, comprehend and share information for following reasons.

1. Visual images transmit huge information to human brain at a glance.
2. Visual images help in establishing relationships and patterns
3. Allows data exploration from different angles.
4. It helps in understanding problems and outliers.

Applications of visualization

These tools are used in various applications

1. Education: Can be used for simulation modelling of any object or process
2. Information: Can be applied to transform abstract data into visual forms for easy interpretations.
3. Production: Various applications are used to create 3D models of products for better viewing and manipulation. Real estate, communication, and automobile industry extensively use 3D advertisement to provide a better loo and feel to their products.
4. Science: Every field of science including fluid dynamics, astrophysics and medicine use visual representation of information.
5. System Visualization – System visualization is a relatively new concept that integrates visual techniques to better describe complex systems.
6. Visual Communication – Multimedia and entertainment industry use visual to communicate their ideas and information.
7. Visual analytics – it refers to the science of analytical reasoning supported by the interactive visual interface. The data generated by social media interaction is interpreted using visual analytics technique.

10 **Consider the following data frame given below**: **[May 2024, 10 Marks]**

Course	Id	Class	Marks
1	11	1	56
2	12	2	75
3	13	1	48
4	14	2	69
5	15	1	84
6	16	2	53

i. Create a subset of course less than 5 by using [ ] brackets and demonstrate the output.
# Creating the data frame with the given information
course_data <- data.frame(

```
 course = c(1, 2, 3, 4, 5, 6),
 id = c(11, 12, 13, 14, 15, 16),
 class = c(1, 2, 1, 2, 1, 2),
 marks = c(56, 75, 48, 69, 84, 53)
)

 # Displaying the data frame

print("Course Data Frame:")
print(course_data)

# Subset using []
subset_course_less_than_5 <- course_data[course_data$course < 5, ]

# Display the subset
print("Subset of Course less than 5 using [] brackets:")
print(subset_course_less_than_5)
```

```
> print("Subset of Course less than 5 using [] bracke
[1] "Subset of Course less than 5 using [] brackets:"
> print(subset_course_less_than_5)
  course id class marks
1      1 11     1    56
2      2 12     2    75
3      3 13     1    48
4      4 14     2    69
```

ii. Create a subset where the course column is less than 4 or the class equals to 1 by using subset() function and demonstrate the output.

```
 # Subset using subset()
 subset_course_class_condition <- subset(course_data, course < 4 | class == 1)

# Display the subset
print("Subset where course < 4 or class == 1 using subset():")
print(subset_course_class_condition)
```

```
> # Subset using subset()
> subset_course_class_condition <- subset(course_data, course < 4 | class == 1)
>
> # Display the subset
> print("Subset where course < 4 or class == 1 using subset():")
[1] "Subset where course < 4 or class == 1 using subset():"
> print(subset_course_class_condition)
  course id class marks
1      1 11     1    56
2      2 12     2    75
3      3 13     1    48
5      5 15     1    84
```

11 i. Write a script to create a dataset named data1 in R containing following text.

Text: 2, 3, 4, 5, 6.7, 7, 8.1, 9

ii. Explain the various functions provided by R to combine different sets of data.

[May 2024, 10 Marks]

Solution

Script to create dataset named data1

```
# Create the dataset as text
data1 <- c("2", "3", "4", "5", "6.7", "7", "8.1", "9")

# View the dataset
print(data1)
```

output

```
[1] "2"   "3"   "4"   "5"   "6.7" "7"   "8.1" "9"
```

If numeric data is passed, the output will look like

```
data1 = c(2, 3, 4, 5, 6.7, 7, 8.1, 9)
data1
```

```
[1] 2.0 3.0 4.0 5.0 6.7 7.0 8.1 9.0
```

Explain the various functions provided by R to combine different sets of data.

Solution:
Ways to combine different datasets to be merged together are:

merge() function
cbind() function

rbind() function

merge() function is used to merge the data contained in different data frames on the basis of columns and rows.
It combines the data of two frames on the basis of the existence of a common column between the two. The merge() function takes the following arguments.

x – specifies a data frame
y – specifies a data frame
by, by.x, by.y specifies the names of the columns common in both x and y
all, all.x, all.y - Specify logical values for the type of merge. The default value is all FALSE.

Example

```
course_data <- data.frame (
 course = c(1, 2, 3, 4, 5, 6),
 id = c(11, 12, 13, 14, 15, 16),
 class = c(1, 2, 1, 2, 1, 2),
)

marks_data <- data.frame (

 id = c(11, 12, 13, 14, 15, 16),
 marks = c(56, 75, 48, 69, 84, 53)   )

merge(course_data, marks_data, “id”)
```

```
> merge(course_data, marks_data, "id")
  id course class marks
1 11      1     1    56
2 12      2     2    75
3 13      3     1    48
4 14      4     2    69
5 15      5     1    84
6 16      6     2    53
```

cbind() function is used to add columns of datasets having an equal set and identical order of rows.

It is used to bind the column names of two datasets. It helps in restricting the number of columns to be included in new datasets
Example

```
course_data <- data.frame (
 course = c(1, 2, 3, 4, 5, 6),
 id = c(11, 12, 13, 14, 15, 16),
 class = c(1, 2, 1, 2, 1, 2),
)

marks_data <- data.frame (
```

```
id = c(11, 12, 13, 14, 15, 16),
marks = c(56, 75, 48, 69, 84, 53)   )

cbind(course_data[ ,c(1,2,3)], marks_data[,c(1))
```

```
  course id class marks_data[, c(1)]
1      1 11     1                  11
2      2 12     2                  12
3      3 13     1                  13
4      4 14     2                  14
5      5 15     1                  15
6      6 16     2                  16
>
```

rbind() function is used to add rows in datasets having an equal number of columns

```
data1 <- data.frame( id = c(1,2,3,4,5),
                     name = c("Ann", "John", "Ben", "jim","Tim")   )
data2 <- data.frame (id = c(6,7,8) ,
                     name = c("May", "Jenny", "carry") )

rbind(data1, data2)
```

```
1  1   Ann
2  2  John
3  3   Ben
4  4   jim
5  5   Tim
6  6   May
7  7 Jenny
8  8 carry
```