Experiment No.8

❖ **Problem Statement 1:-**

1. Create a Student Data Frame: Create a data frame named students with the columns: Name, Age,

   Gender, Grade, and Score. Add 5 rows with fictional data.

   - *Tasks*: Access specific columns and rows, filter students with a Score greater than 80, and select only the Name and Grade columns for students who are older than 18.

2. Employee Data Manipulation: Create an employees data frame with columns: EmployeeID, Name, Department, Salary, and YearsOfService. Populate it with at least 5 rows of data.

   - *Tasks*: Add a column Bonus where each employee receives a bonus of 10% of their Salary. Sort the data frame by Salary in descending order. Filter and display only the employees in the "HR" department.

- Code:-

```
students <- data.frame(
  Name = c("Alice", "Bob", "Charlie", "David", "Eva"),
  Age = c(17, 19, 18, 20, 21),
  Gender = c("F", "M", "M", "M", "F"),
  Grade = c("A", "B", "B", "A", "C"),
  Score = c(85, 75, 90, 60, 95)
)
specific_columns <- students[, c("Name", "Score")]  # Access Name and Score columns
specific_row <- students[3, ]  # Access the 3rd row
score_above_80 <- subset(students, Score > 80)
older_than_18 <- subset(students, Age > 18, select = c(Name, Grade))
cat("Students Data Frame:\n")
print(students)
cat("\nName and Score Columns:\n")
print(specific_columns)
cat("\nSpecific Row:\n")
print(specific_row)
```

```
cat("\nStudents with Score > 80:\n")
print(score_above_80)
cat("\nName and Grade for students older than 18:\n")
print(older_than_18)
employees <- data.frame(
  EmployeeID = c(101, 102, 103, 104, 105),
  Name = c("John", "Jane", "Jim", "Jack", "Jill"),
  Department = c("HR", "Finance", "IT", "HR", "Marketing"),
  Salary = c(50000, 60000, 55000, 52000, 48000),
  YearsOfService = c(5, 7, 4, 6, 3)
)
employees$Bonus <- employees$Salary * 0.1
sorted_employees <- employees[order(-employees$Salary), ]
hr_employees <- subset(employees, Department == "HR")
cat("\nEmployees Data Frame:\n")
print(employees)
cat("\nEmployees sorted by Salary (descending):\n")
print(sorted_employees)
cat("\nEmployees in HR Department:\n")
print(hr_employees)
```

- **Output:-**

```
Employees Data Frame:
  EmployeeID   Name   Department    Salary    YearsOfService     Bonus
1       101        John         HR             50000            5                 5000
2       102        Jane      Finance       60000            7                 6000
3       103         Jim          IT              55000            4                 5500
4       104        Jack         HR              52000            6                 5200
5       105         Jill       Marketing      48000            3                 4800
```

```
Employees sorted by Salary (descending):
  EmployeeID   Name     Department    Salary   YearsOfService   Bonus
2    102       Jane      Finance      60000         7           6000
3    103       Jim         IT          55000         4            5500
4    104       Jack        HR          52000         6            5500
1    101       John        HR          50000         5            5000
5    105        Jill     Marketing     48000         3            4800

Employees in HR Department:
  EmployeeID  Name   Department   Salary  YearsOfService     Bonus
1    101      John       HR         50000       5               5000
4    104      Jack       HR         52000       6               5200
```

Problem Set 2: Data Frame Aggregation and Summary Statistics

1. Sales Data Aggregation: Create a sales data frame with columns Salesperson, Region, Product, and Sales Amount. Populate it with data representing at least 10 sales.

- *Tasks*: Use the aggregate() function to find the total SalesAmount for each Salesperson. Also, find the average SalesAmount for each Region.

2. Course Grades Summary: Create a grades data frame with columns StudentName, Course, Score, and GradeLevel.

- *Tasks*: Calculate the average Score for each GradeLevel using the aggregate() function. Display summary statistics for the Score column using the summary() function.

- **Code:-**

```
sales <- data.frame(
  Salesperson = c("Alice", "Bob", "Charlie", "Alice", "David",
                  "Eva", "Charlie", "Eva", "Bob", "David"),
  Region = c("North", "South", "North", "East", "South",
             "East", "North", "East", "South", "East"),
  Product = c("A", "B", "A", "C", "B",
             "C", "A", "B", "C", "A"),
  SalesAmount = c(500, 700, 300, 450, 650, 800, 400, 750, 600, 900)
)
total_sales_by_salesperson <- aggregate(SalesAmount ~ Salesperson, data = sales, sum)
average_sales_by_region <- aggregate(SalesAmount ~ Region, data = sales, mean)
cat("Sales Data Frame:\n")
print(sales)
cat("\nTotal SalesAmount by Salesperson:\n")
print(total_sales_by_salesperson)
cat("\nAverage SalesAmount by Region:\n")
print(average_sales_by_region)
grades <- data.frame(
  StudentName = c("Alice", "Bob", "Charlie", "David", "Eva",
                 "Frank", "Grace", "Hannah", "Ivy", "Jack"),
  Course = c("Math", "Science", "Math", "Science", "Math",
            "Science", "Math", "Science", "Math", "Science"),
  Score = c(85, 78, 92, 88, 76, 90, 95, 80, 89, 83),
  GradeLevel = c("Grade 10", "Grade 10", "Grade 11", "Grade 11", "Grade 10",
                "Grade 12", "Grade 11", "Grade 12", "Grade 10", "Grade 12")
)
average_score_by_gradelevel <- aggregate(Score ~ GradeLevel, data = grades, mean)
score_summary <- summary(grades$Score)
cat("\nGrades Data Frame:\n")
print(grades)
cat("\nAverage Score by GradeLevel:\n")
print(average_score_by_gradelevel)
cat("\nSummary Statistics for Scores:\n")
print(score_summary)
```

- **Output:-**

```
Sales Data Frame:
   Salesperson   Region   Product    SalesAmount
1        Alice       North       A            500
2          Bob       South       B            700
3     Charlie      North       A            300
4        Alice        East        C           450
5        David      South       B           650
6          Eva        East        C           800
7     Charlie      North       A           400
8          Eva        East        B           750
9          Bob       South       C           600
10       David      East        A          900

Total SalesAmount by Salesperson:
      Salesperson    SalesAmount
1       Alice             950
2         Bob            1300
3    Charlie           700
4       David           1550
5         Eva           1550

Average SalesAmount by Region:
    Region   SalesAmount
1   East         725
2  North         400
3  South         650

Grades Data Frame:
    StudentName     Course   Score     GradeLevel
1        Alice            Math       85        Grade 10
2          Bob           Science     78        Grade 10
3     Charlie           Math       92        Grade 11
4        David          Science     88        Grade 11
5          Eva            Math       76         Grade 10
6        Frank          Science     90        Grade 12
7        Grace           Math       95        Grade 11
8      Hannah         Science     80        Grade 12
9          Ivy             Math       89         Grade 10
10        Jack          Science     83         Grade 12

Average Score by GradeLevel:
  GradeLevel         Score
1   Grade 10       82.00000
2   Grade 11       91.66667
```

```
3   Grade 12       84.33333

Summary Statistics for Scores:-
   Min.   1st Qu.  Median    Mean   3rd Qu.    Max.
  76.00    80.75    86.50    85.60    89.75    95.00
```

Problem Set 3: Advanced Data Frame Operations

1. **Merging Customer and Order Data**: Create two data frames, customers and orders. The customers data frame should have columns CustomerID, Name, and City. The orders data frame should have columns OrderID, CustomerID, OrderAmount, and OrderDate.

- *Tasks*: Merge these two data frames on CustomerID to get a complete order history for each customer. Filter the resulting data frame to show orders placed in the last 6 months.

2. **Employee Salary Increase Analysis**: Create a company data frame with columns EmployeeID, Name, Department, CurrentSalary, and YearsOfExperience.

- *Tasks*: Add a new column NewSalary, where employees with more than 5 years of experience receive a 15% increase in salary. Calculate the average NewSalary for each department and sort departments by the average salary in descending order.

3. Monthly Expenses Tracker: Create a expenses data frame with columns Month, Category (e.g., "Rent", "Food", "Utilities"), Amount, and Description.
Tasks: Use the aggregate() function to find the total amount spent on each category over the year. Display the top 3 categories with the highest expenses.

- **Code:-**

```
customers <- data.frame(
  CustomerID = c(1, 2, 3, 4, 5),
  Name = c("Alice", "Bob", "Charlie", "David", "Eva"),
  City = c("New York", "Los Angeles", "Chicago", "Houston", "Phoenix")
)

orders <- data.frame(
  OrderID = c(101, 102, 103, 104, 105),
  CustomerID = c(1, 2, 3, 1, 4),
  OrderAmount = c(250, 300, 150, 400, 500),
  OrderDate = as.Date(c("2024-05-15", "2024-06-10", "2024-04-25", "2024-11-01",
"2024-07-20"))
)
merged_data <- merge(customers, orders, by = "CustomerID")
recent_orders <- subset(merged_data, OrderDate >= as.Date(Sys.Date() - 180))
cat("Merged Data Frame:\n")
print(merged_data)
cat("\nRecent Orders (Last 6 Months):\n")
print(recent_orders)

company <- data.frame(
  EmployeeID = c(1, 2, 3, 4, 5),
  Name = c("John", "Jane", "Jim", "Jack", "Jill"),
  Department = c("HR", "IT", "Finance", "HR", "IT"),
  CurrentSalary = c(50000, 60000, 55000, 52000, 48000),
  YearsOfExperience = c(6, 4, 7, 3, 8)
)

company$NewSalary <- ifelse(company$YearsOfExperience > 5,
                            company$CurrentSalary * 1.15,
                            company$CurrentSalary)

average_salary_by_department <- aggregate(NewSalary ~ Department, data =
company, mean)

average_salary_by_department <- average_salary_by_department[order(-
average_salary_by_department$NewSalary), ]

cat("\nCompany Data Frame:\n")
print(company)
cat("\nAverage New Salary by Department (Descending):\n")
print(average_salary_by_department)

expenses <- data.frame(
  Month = c("January", "February", "March", "April", "May", "June"),
  Category = c("Rent", "Food", "Utilities", "Rent", "Food", "Utilities"),
  Amount = c(1200, 300, 150, 1250, 350, 175),
  Description = c("Apartment Rent", "Groceries", "Electricity", "Apartment Rent",
"Groceries", "Electricity")
)

cat("\nMonthly Expenses Data Frame:\n")
print(expenses)
```

- **Output:-**

```
Merged Data Frame:
   CustomerID  Name     City      OrderID  OrderAmount  OrderDate
1      1       Alice   New York     101        250       2024-05-15
2      1       Alice   New York     104        400       2024-11-01
3      2        Bob   Los Angeles   102        300       2024-06-10
4      3      Charlie  Chicago      103        150       2024-04-25
5      4       David   Houston      105        500       2024-07-20
```

```
Recent Orders (Last 6 Months):
  CustomerID   Name     City        OrderID   OrderAmount   OrderDate
2     1        Alice    New York      104         400       2024-11-01
3     2        Bob      Los Angeles   102         300       2024-06-10
5     4        David    Houston       105         500       2024-07-20

Company Data Frame:
  EmployeeID  Name Department CurrentSalary YearsOfExperience NewSalary
1     1       John     HR        50000            6            57500
2     2       Jane     IT        60000            4            60000
3     3       Jim    Finance     55000            7            63250
4     4       Jack     HR        52000            3            52000
5     5       Jill     IT        48000            8            55200

Average New Salary by Department (Descending):
  Department  NewSalary
1  Finance      63250
3    IT         57600
2    HR         54750

Monthly Expenses Data Frame:

   Month   Category  Amount   Description
1 January    Rent     1200    Apartment Rent
2 February   Food     300       Groceries
3  March   Utilities  150       Electricity
4  April     Rent     1250    Apartment Rent
5   May      Food     350       Groceries
6   June   Utilities  175       Electricity
```