Experiment No 8

Title: Implementation of Data frame and its corresponding operators and functions

Objective:

The objective of this practical is to learn how to create, access, and manipulate data within Data Frames in R, using various functions and operators. This will help in organizing and preparing data for analysis or further processing.

Theory :

Data Frames in R are fundamental data structures that organize data into a 2-dimensional table format, where each column represents a variable, and each row represents an observation. They are particularly suited for handling and analyzing structured data, similar to tables in databases or spreadsheets. Here's a deep dive into the key aspects of working with Data Frames:

1. **Creation of Data Frames**

- **Manual Creation**: Data frames can be created directly in R using the `data.frame()` function, which allows you to define columns and their values.
- **Importing Data**: Data frames are often generated by importing data from external sources, such as CSV files, Excel spreadsheets, or databases, using functions like `read.csv()` and `read.table()`.

2. **Data Frame Structure and Access**

- **Accessing Elements**: You can access data within data frames by specifying row and column indices (e.g., `data[row, column]`) or by using column names.
- **Subsetting**: The `subset()` function allows you to filter data based on specific conditions, helping in selecting only the required data for analysis.
- **Column Manipulation**: Columns can be added, removed, or modified within data frames. For example, creating a new column based on existing columns can be useful for derived metrics.

3. **Functions on Data Frames**

- **Summarization**: The `summary()` function provides a quick overview of each column, such as minimum, maximum, mean, and quartiles, depending on the column type.
- **Inspection**: Functions like `str()`, `head()`, `tail()`, and `dim()` help in understanding the data's structure, previewing contents, and assessing dimensions.
- **Applying Operations Across Rows or Columns**: The `apply()` family of functions, including `apply()`, `sapply()`, and `lapply()`, enables the application of functions across rows or columns, useful for tasks like normalization or scaling.

4. **Data Transformation**

- **Filtering and Sorting**: Data frames can be filtered using logical conditions and sorted with functions like `order()`. This is essential for cleaning and ordering data prior to analysis.

- **Merging and Joining**: Combining data from multiple sources or frames is achievable using `merge()` or functions from the `dplyr` package (e.g., `left_join()`, `right_join()`), facilitating data integration for comprehensive analysis.
- **Reshaping**: Functions such as `reshape()`, `melt()`, and `cast()` (available in `reshape2` and `tidyverse`) allow you to change the data layout from wide to long formats or vice versa, a critical step in data preparation.

5. **Using Operators with Data Frames**

- **Logical Operators**: Logical operators (e.g., ==, `!=`, >, <) can filter data to meet certain conditions, allowing for selective analysis.
- **Arithmetic Operations**: Arithmetic operations can be applied across columns or rows, enabling transformations and calculations (e.g., unit conversions, percentage changes).
- **Relational Operators**: These are used to compare data across columns or within a column, often forming the basis for conditional analyses.

Practical Approach:

Implementation of Data frame and its corresponding operators and functions

Note: The students should perform for following statements using functions and recursive functions in R.

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.

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.

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.

Outcome:

Students will learn how to

- Create and manipulate data frames with various columns.
- Filter rows based on specific conditions.
- Calculate totals or averages for specific columns.

Online Reference Websites:

- **https://www.w3resource.com/r-programming-exercises/dataframe/index.php**
- **https://rpubs.com/onurhan/r-exercises-5-2**
- **http://r-tutorials.com/r-exercises-31-40-data-frame-manipulations/**

Viva Questions:

1.	What is a Data Frame in R, and how is it different from other data structures like matrices?
2.	How do you create a Data Frame in R, and what functions can you use to import external data into a Data Frame?
3.	How would you access the first few rows of a Data Frame? Which function can you use for this?
4.	Can you explain how to subset a Data Frame based on specific conditions?

5.	What are some functions you would use to get a summary of each column in a Data Frame?
6.	How can you add a new column to an existing Data Frame?
7.	What is the purpose of using functions like `apply()`, `lapply()`, and `sapply()` with Data Frames?
8.	How do logical operators work in filtering rows of a Data Frame? Can you give an example?
9.	Explain the process of merging two Data Frames. What functions are commonly used for this?

Conclusion:

practicals demonstrate how to create and work with data frames in R. By practicing basic operations like adding, filtering, and summarizing data, students gain foundational skills in managing and analyzing structured data efficiently. This knowledge is essential for handling real-world data in various applications.