EXPT NO : 07	DATA FRAMES
DATE : 12.03.2025	

AIM:

To implement and perform operation using data frames.

1. Creating and Displaying a Data Frame:

• Create a data frame named student_data with the following columns and 5 rows:

o ID: Numeric student IDs (101, 102, 103, 104, 105)

o Name: Character vector (store student names)

o Age: Numeric vector (store student ages)

o Marks: Numeric vector (store marks out of 100)

o Passed: Logical vector (TRUE if marks ≥ 50, otherwise FALSE)

• Print the entire data frame.

• Display only the column names of the data frame.

• Display the structure of the data frame using str().

• Display the summary of the data frame using summary().

CODE:

```
student_data <- data.frame(
  ID = c(101, 102, 103, 104, 105),
  Name = c("Abi", "banu", "Charlie", "Danish", "jerry"),
  Age = c(20, 19, 19, 20, 18),
  Marks = c(85, 45, 76, 90, 50),
  Passed = c(TRUE, FALSE, TRUE, TRUE, TRUE))
print(student_data)
print(colnames(student_data))
str(student_data)
summary(student_data)
```

OUTPUT:

```
> print(student_data)
   ID     Name Age Marks Passed
1 101      Abi  20    85   TRUE
2 102    banu   19    45  FALSE
3 103 Charlie  19    76   TRUE
4 104   Danish  20    90   TRUE
5 105    jerry  18    50   TRUE
> print(colnames(student_data))
[1] "ID"     "Name"   "Age"    "Marks"  "Passed"
> str(student_data)
'data.frame':   5 obs. of  5 variables:
 $ ID    : num  101 102 103 104 105
 $ Name  : chr  "Abi" "banu" "Charlie" "Danish" ...
 $ Age   : num  20 19 19 20 18
 $ Marks : num  85 45 76 90 50
 $ Passed: logi  TRUE FALSE TRUE TRUE TRUE
> summary(student_data)
       ID            Name                 Age           Marks         Passed
 Min.   :101   Length:5           Min.   :18.0   Min.   :45.0   Mode :logical
 1st Qu.:102   Class :character   1st Qu.:19.0   1st Qu.:50.0   FALSE:1
 Median :103   Mode  :character   Median :19.0   Median :76.0   TRUE :4
 Mean   :103                      Mean   :19.2   Mean   :69.2
 3rd Qu.:104                      3rd Qu.:20.0   3rd Qu.:85.0
 Max.   :105                      Max.   :20.0   Max.   :90.0
> |
```

2. Accessing and Extracting Data from a Data Frame:

• Extract and display the Name and Marks columns from student_data.

• Extract and print the data of students whose marks are greater than 75.

• Extract and print the data of students who are below 20 years old.

• Extract and print the first three rows of the data frame.

• Extract and print the student name and marks of the student with ID = 103.

CODE:

```
cat("Name and Marks columns:\n")
print(student_data[, c("Name", "Marks")])
cat("\nStudents with marks > 75:\n")
print(subset(student_data, Marks > 75))
cat("\nStudents below 20 years old:\n")
print(subset(student_data, Age < 20))
cat("\nFirst three rows of the data frame:\n")
print(head(student_data, 3))
cat("\nStudent with ID = 103:\n")
print(subset(student_data, ID == 103, select = c("Name", "Marks")))
```

OUTPUT:

```
Name and Marks columns:
> print(student_data[, c("Name", "Marks")])
     Name Marks
1     Abi     85
2    banu     45
3 Charlie     76
4  Danish     90
5   jerry     50
> cat("\nStudents with marks > 75:\n")

Students with marks > 75:
> print(subset(student_data, Marks > 75))
   ID     Name Age Marks Passed
1 101      Abi  20     85   TRUE
3 103 Charlie  19     76   TRUE
4 104  Danish  20     90   TRUE
> cat("\nStudents below 20 years old:\n")

Students below 20 years old:
> print(subset(student_data, Age < 20))
   ID     Name Age Marks Passed
2 102    banu  19     45  FALSE
3 103 Charlie  19     76   TRUE
5 105   jerry  18     50   TRUE
> cat("\nFirst three rows of the data frame:\n")

First three rows of the data frame:
> print(head(student_data, 3))
   ID     Name Age Marks Passed
1 101      Abi  20     85   TRUE
2 102    banu  19     45  FALSE
3 103 Charlie  19     76   TRUE
> cat("\nStudent with ID = 103:\n")

Student with ID = 103:
> print(subset(student_data, ID == 103, select = c("Name", "Marks")))
     Name Marks
3 Charlie     76
```

3. Adding and Modifying Data in a Data Frame:

• Add a new column Grade to student_data based on Marks:

o Marks ≥ 90 → "A"

o Marks ≥ 75 & < 90 → "B"

o Marks ≥ 50 & < 75 → "C"

o Marks < 50 → "F"

• Add a new row for a student with ID = 106, Name = "David", Age = 21, Marks = 88, and compute Passed and Grade.

• Modify the marks of the student with ID = 102 to 95 and update their Grade.

• Delete the Passed column from the data frame.

• Display the updated data frame.

CODE:

```
student_data$Grade <- ifelse(student_data$Marks >= 90, "A", ifelse(student_data$Marks >= 75, "B",
ifelse(student_data$Marks >= 50, "C", "F")))
print("Original Data Frame:")
print(student_data)
new_student <- data.frame(ID = 106, Name = "David", Age = 21, Marks = 88, Passed = TRUE,
Grade = ifelse(88 >= 90, "A", ifelse(88 >= 75, "B", ifelse(88 >= 50, "C", "F"))))
student_data <- rbind(student_data, new_student)
student_data$Marks[student_data$ID == 102] <- 95
```

```
student_data$Grade[student_data$ID == 102] <- "A"

student_data$Passed <- NULL

print("Updated Data Frame:")

print(student_data)
```

OUTPUT:

```
> print("Original Data Frame:")
[1] "Original Data Frame:"
> print(student_data)
   ID     Name Age Marks Passed Grade
1 101      Abi  20    85   TRUE     B
2 102     Banu  19    45  FALSE     F
3 103  Charlie  19    76   TRUE     B
4 104   Danish  20    90   TRUE     A
5 105    Jerry  18    50   TRUE     C
> new_student <- data.frame(
+   ID = 106, Name = "David", Age = 21, Marks = 88, Passed = TRUE,
+   Grade = ifelse(88 >= 90, "A", ifelse(88 >= 75, "B", ifelse(88 >= 50, "C", "F")))
+ )
> student_data <- rbind(student_data, new_student)
> student_data$Marks[student_data$ID == 102] <- 95
> student_data$Grade[student_data$ID == 102] <- "A"
> student_data$Passed <- NULL
> print("Updated Data Frame:")
[1] "Updated Data Frame:"
> print(student_data)
   ID     Name Age Marks Grade
1 101      Abi  20    85     B
2 102     Banu  19    95     A
3 103  Charlie  19    76     B
4 104   Danish  20    90     A
5 105    Jerry  18    50     C
6 106    David  21    88     B
>
```

4. Sorting, Ordering, and Aggregating Data:

• Sort the student_data based on Marks in descending order and display the sorted data frame.

• Sort the student_data based on Name in alphabetical order and display the result.

• Calculate and print the average marks of all students.

• Calculate and print the highest and lowest marks.

• Count and print the number of students in each Grade category.

CODE:

```
sorted_by_marks <- student_data[order(-student_data$Marks), ]

print("Sorted by Marks (Descending):")

print(sorted_by_marks)
```

```
sorted_by_name <- student_data[order(student_data$Name), ]
print("Sorted by Name (Alphabetical Order):")
print(sorted_by_name)
average_marks <- mean(student_data$Marks)
print(paste("Average Marks:", average_marks))
highest_marks <- max(student_data$Marks)
lowest_marks <- min(student_data$Marks)
print(paste("Highest Marks:", highest_marks))
print(paste("Lowest Marks:", lowest_marks))
grade_counts <- table(student_data$Grade)
print("Number of Students in Each Grade Category:")
print(grade_counts)
```

OUTPUT:

```
[1] "Sorted by Marks (Descending):"
> print(sorted_by_marks)
   ID     Name Age Marks Grade
2 102     Banu  19     95     A
4 104   Danish  20     90     A
6 106    David  21     88     B
1 101      Abi  20     85     B
3 103 Charlie  19     76     B
5 105    Jerry  18     50     C
> sorted_by_name <- student_data[order(student_data$Name), ]
> print("Sorted by Name (Alphabetical Order):")
[1] "Sorted by Name (Alphabetical Order):"
> print(sorted_by_name)
   ID     Name Age Marks Grade
1 101      Abi  20     85     B
2 102     Banu  19     95     A
3 103 Charlie  19     76     B
4 104   Danish  20     90     A
6 106    David  21     88     B
5 105    Jerry  18     50     C
> average_marks <- mean(student_data$Marks)
> print(paste("Average Marks:", average_marks))
[1] "Average Marks: 80.6666666666667"
> highest_marks <- max(student_data$Marks)
> lowest_marks <- min(student_data$Marks)
> print(paste("Highest Marks:", highest_marks))
[1] "Highest Marks: 95"
> print(paste("Lowest Marks:", lowest_marks))
[1] "Lowest Marks: 50"
> grade_counts <- table(student_data$Grade)
> print("Number of Students in Each Grade Category:")
[1] "Number of Students in Each Grade Category:"
> print(grade_counts)

A B C
2 3 1
```

5. Exporting and Importing Data Frames:

• Save the student_data data frame as a CSV file named "students.csv".

• Save the student_data data frame as an Excel file (if necessary, use openxlsx package).

• Read the data from the CSV file into a new data frame new_data and print the first 3 rows.

• Check and print whether new_data is identical to student_data.

• Display the column names of new_data after importing.

CODE:

```
install.packages("openxlsx")
library(openxlsx)
write.csv(student_data, "students.csv", row.names = FALSE)
write.xlsx(student_data, "students.xlsx")
new_data <- read.csv("students.csv")
print(head(new_data, 3))
identical_check <- identical(new_data, student_data)
print(paste("Are the data frames identical?", identical_check))
print(colnames(new_data))
```

OUTPUT:

```
> print(head(new_data, 3))
   ID     Name Age Marks Passed
1 101      Abi  20    85   TRUE
2 102     banu  19    45  FALSE
3 103 Charlie  19    76   TRUE
>
> identical_check <- identical(new_data, student_data)
> print(paste("Are the data frames identical?", identical_check))
[1] "Are the data frames identical? FALSE"
>
> print(colnames(new_data))
[1] "ID"     "Name"   "Age"    "Marks"  "Passed"
>
```

RESULT:

Thus the programs using vectors, lists, matrices, arrays, factors are implemented and executed successfully.