

# SHETH L.U.J. AND SIR M.V. COLLEGE

## SUBJECT: DATA ANALYSIS WITH R

### PRACTICAL NO. 11

AIM: Reshaping data using pivot\_longer()/pivot\_wider() (R).

OUTPUT:

```
RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Source
Console Terminal Background Jobs
R - R432 - ~/...
[1] student_exam_scores.csv
> # R Script: Reshaping Data with pivot_longer() and pivot_wider()
> # Dataset: student_exam_scores.csv
> library(dplyr)
> library(tidyverse)
> # 1. IMPORT DATA
> df <- read.csv("D:/S095 Aashka/student_exam_scores.csv")
> head(df)
  student_id hours_studied sleep_hours attendance_percent previous_scores exam_score
1      S001           8.0           8.8             72.1         45          30.2
2      S002           1.3           8.6             60.7         55          25.0
3      S003           4.0           8.2             73.7         86          35.8
4      S004           3.5           4.8             95.1         66          34.0
5      S005           9.1           6.4             89.8         71          40.3
6      S006           8.4           5.1             58.5         75          35.7
>
> print("--- 1. Original data ---")
[1] "--- 1. Original data ---"
> print(head(df))
  student_id hours_studied sleep_hours attendance_percent previous_scores exam_score
1      S001           8.0           8.8             72.1         45          30.2
2      S002           1.3           8.6             60.7         55          25.0
3      S003           4.0           8.2             73.7         86          35.8
4      S004           3.5           4.8             95.1         66          34.0
5      S005           9.1           6.4             89.8         71          40.3
6      S006           8.4           5.1             58.5         75          35.7
>
> # 2. PIVOT_LONGER (wide -> long)
> # Purpose: Combine numeric variables into long format
> long_df <- df %>%
+   pivot_longer(
+     cols = c(hours_studied, sleep_hours, attendance_percent,
+               previous_scores, exam_score),
+     names_to = "Metric",
+     values_to = "value"
+   )
>
> print("--- 2. Long Format (pivot_longer) ---")
[1] "--- 2. Long Format (pivot_longer) ---"
> print(head(long_df, 10))
# A tibble: 10 x 3
  student_id Metric      value
  <chr>      <chr>      <dbl>
1 S001 hours_studied      8
2 S001 sleep_hours      8.8
3 S001 attendance_percent 72.1
4 S001 previous_scores    45
5 S001 exam_score        30.2
6 S002 hours_studied      1.3
7 S002 sleep_hours      8.6
8 S002 attendance_percent 60.7
9 S002 previous_scores    55
10 S002 exam_score        25
>
> # 3. PIVOT_WIDER (Long -> wide)
> # Purpose: Convert long_df BACK to original format
> wide_df <- long_df %>%
+   pivot_wider(
+     names_from = Metric,
+     values_from = value
+   )
>
> print("--- 3. Wide Format (pivot_wider) ---")
[1] "--- 3. Wide Format (pivot_wider) ---"
> print(head(wide_df))
```

NAME: AASHKA MISHRA  
ROLL NO: S095

# SHETH L.U.J. AND SIR M.V. COLLEGE

## SUBJECT: DATA ANALYSIS WITH R

The image displays two screenshots of the RStudio interface, demonstrating data manipulation using the dplyr package.

**Top Screenshot:**

```
> # =====
> # 3. PIVOT_WIDER (Long -> Wide)
> # Purpose: Convert long_df BACK to original format
> # =====
> wide_df <- long_df %>%
+   pivot_wider(
+     names_from = Metric,
+     values_from = Value
+   )
>
> print("--- 3. Wide Format (pivot_wider) ---")
> print(head(wide_df))
# A tibble: 6 x 6
  student_id hours_studied sleep_hours attendance_percent previous_scores exam_score
  <chr>      <dbl>      <dbl>      <dbl>      <dbl>      <dbl>
1 S001      8        8.8        72.1      45        30.2
2 S002      1.3      8.6        60.7      55        25
3 S003      4        8.2        73.7      86        35.8
4 S004      3.5      4.8        95.1      66        34
5 S005      9.1      6.4        89.8      71        40.3
6 S006      8.4      5.1        58.5      75        35.7
>
> # =====
> # 4. ADVANCED EXAMPLE:
> # Pivot where rows = student_id and columns = attendance_percent levels
> # (Useful for reporting or heatmaps)
> # =====
> attendance_pivot <- df %>%
+   mutate(attendance_group = cut(attendance_percent,
+                                 breaks = c(0, 50, 75, 100),
+                                 labels = c("Low", "Medium", "High"))) %>%
+   select(student_id, attendance_group, exam_score) %>%
+   pivot_wider(
+     names_from = attendance_group,
+     values_from = exam_score
+   )
>
> print("--- 4. Attendance Group Pivot ---")
> print(head(attendance_pivot))
# A tibble: 6 x 3
  student_id Medium High
  <chr>      <dbl> <dbl>
1 S001      30.2 NA
2 S002      25 NA
3 S003      35.8 NA
4 S004      NA 34
5 S005      NA 40.3
6 S006      35.7 NA
>
```

**Bottom Screenshot:**

```
> print("--- 3. Wide Format (pivot_wider) ---")
> print(head(wide_df))
# A tibble: 6 x 6
  student_id hours_studied sleep_hours attendance_percent previous_scores exam_score
  <chr>      <dbl>      <dbl>      <dbl>      <dbl>      <dbl>
1 S001      8        8.8        72.1      45        30.2
2 S002      1.3      8.6        60.7      55        25
3 S003      4        8.2        73.7      86        35.8
4 S004      3.5      4.8        95.1      66        34
5 S005      9.1      6.4        89.8      71        40.3
6 S006      8.4      5.1        58.5      75        35.7
>
> # =====
> # 4. ADVANCED EXAMPLE:
> # Pivot where rows = student_id and columns = attendance_percent levels
> # (Useful for reporting or heatmaps)
> # =====
> attendance_pivot <- df %>%
+   mutate(attendance_group = cut(attendance_percent,
+                                 breaks = c(0, 50, 75, 100),
+                                 labels = c("Low", "Medium", "High"))) %>%
+   select(student_id, attendance_group, exam_score) %>%
+   pivot_wider(
+     names_from = attendance_group,
+     values_from = exam_score
+   )
>
> print("--- 4. Attendance Group Pivot ---")
> print(head(attendance_pivot))
# A tibble: 6 x 3
  student_id Medium High
  <chr>      <dbl> <dbl>
1 S001      30.2 NA
2 S002      25 NA
3 S003      35.8 NA
4 S004      NA 34
5 S005      NA 40.3
6 S006      35.7 NA
>
```

NAME: AASHKA MISHRA  
ROLL NO: S095

SHEETH L.U.J. AND SIR M.V. COLLEGE  
SUBJECT: DATA ANALYSIS WITH R

PRACTICAL NO. 12

AIM: Combining datasets vertically (concatenation) using rbind() (R).

```
R - R4.5.2 - ~/
> # Script: Vertical concatenation using rbind()
> # Datasets used:
> # 1. iris.csv
> # 2. flower_dataset.csv
>
> # 1. SETUP: Import both datasets -----
> iris_df <- read.csv("D:/S095 Aashka/iris.csv")
> flower_df <- read.csv("D:/S095 Aashka/flower_dataset.csv")
>
> print("--- Original Column Names ---")
> print(names(iris_df))
[1] "id" "sepal.length.cm" "sepal.width.cm" "petal.length.cm" "petal.width.cm" "species"
> print(names(flower_df))
[1] "species" "size" "fragrance" "height.cm"
>
> # 2. DATA PREPARATION (Aligning Columns)
>
> # Your iris.csv column names:
> # "id", "sepal.length.cm", "sepal.width.cm", "petal.length.cm", "petal.width.cm", "species"
> # Your flower_dataset.csv column names:
> # "species", "height.cm"
>
> # For matching structure, we create common columns:
> # Species -> character
> # Height -> numeric
>
> # 2.1 Prepare iris data
> iris_clean <- iris_df[, c("species", "sepal.length.cm")]
> names(iris_clean) <- c("species", "height")
>
> # 2.2 Prepare flower data
> flower_clean <- flower_df[, c("species", "height.cm")]
> names(flower_clean) <- c("species", "height")
>
> # Convert heights to numeric
> iris_clean$height <- as.numeric(iris_clean$height)
> flower_clean$height <- as.numeric(flower_clean$height)
>
> # 3. VERTICAL COMBINATION WITH rbind()
>
> combined_data <- rbind(iris_clean, flower_clean)
>
> print("--- Combined Data Summary ---")
> print(paste("iris rows:", nrow(iris_clean)))
[1] "iris rows: 150"
> print(paste("flower rows:", nrow(flower_clean)))
[1] "flower rows: 10000"
> print(paste("Total rows expected:", nrow(iris_clean) + nrow(flower_clean)))
[1] "Total rows expected: 10150"
> print(paste("Total rows actual:", nrow(combined_data)))
[1] "Total rows actual: 10150"
>
> # Preview top & bottom rows
> print("--- Preview of Combined Data (Top and Bottom) ---")
> print(head(combined_data))
  species height
1 iris-setosa  5.1
2 iris-setosa  4.9
3 iris-setosa  4.7
4 iris-setosa  4.6
5 iris-setosa  5.0
6 iris-setosa  5.4
> print(tail(combined_data))
  species height
10145 rose  87.69
10146 hibiscus 109.52
10147 shoeblack plant 145.23
10148 hibiscus 126.69
10149 shoeblack plant 77.62
10150 rose  88.11
> # 2.1 Prepare iris data
> iris_clean <- iris_df[, c("species", "sepal.length.cm")]
>
> |
```

NAME: AASHKA MISHRA  
ROLL NO: S095

# SHETH L.U.J. AND SIR M.V. COLLEGE

## SUBJECT: DATA ANALYSIS WITH R

### PRACTICAL NO. 13

AIM: Identifying and handling duplicates using distinct() (R).

OUTPUT:

```
RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Source
Console Terminal Background Jobs
R - R452 - ~/
> # =====
> # R Script: Identifying and Handling Duplicates using distinct()
> # Dataset: student_exam_scores.csv
> # =====
>
> library(dplyr)
> # =====
> # 1. IMPORT YOUR DATA
> # =====
df <- read.csv("D:/S095 Aashka/student_exam_scores.csv")
>
> print("--- 1. Original Dataset (Full Data) ---")
[1] "--- 1. Original Dataset (Full Data) ---"
> print(head(df))
  student_id hours_studied sleep_hours attendance_percent previous_scores exam_score
1      S001          8.0          8.8           72.1         45      30.2
2      S002          1.3          8.6           60.7         55      25.0
3      S003          4.0          8.2           73.7         86      35.8
4      S004          3.5          4.8           95.1         66      34.0
5      S005          9.1          6.4           89.8         71      40.3
6      S006          8.4          5.1           58.5         75      35.7
> print(paste("Total rows:", nrow(df)))
[1] "Total rows: 200"
> # =====
> # 2. IDENTIFYING DUPLICATES (Exact duplicates of ENTIRE ROW)
> # =====
duplicates_report <- df %>%
+ group_by(across(everything())) %>% # group by all columns
+ count() %>% # count occurrences
+ filter(n > 1) # keep only duplicates
>
> print("--- 2. Duplicate Rows Found (Exact Matches) ---")
[1] "--- 2. Duplicate Rows Found (Exact Matches) ---"
> print(duplicates_report)
# A tibble: 0 x 7
# Groups:   student_id, hours_studied, sleep_hours, attendance_percent, previous_scores, exam_score [0]
# 7 variables: student_id <chr>, hours_studied <dbl>, sleep_hours <dbl>, attendance_percent <dbl>, previous_scores <int>,
# exam_score <dbl>, n <int>
> # =====
> # 3. REMOVING EXACT DUPLICATES
```

```
RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Source
Console Terminal Background Jobs
R - R452 - ~/
> # =====
> # 3. REMOVING EXACT DUPLICATES
> # =====
df_no_dups <- df %>%
+ distinct() # removes only rows fully identical
>
> print("--- 3. Dataset After Removing Exact Duplicates ---")
[1] "--- 3. Dataset After Removing Exact Duplicates ---"
> print(head(df_no_dups))
  student_id hours_studied sleep_hours attendance_percent previous_scores exam_score
1      S001          8.0          8.8           72.1         45      30.2
2      S002          1.3          8.6           60.7         55      25.0
3      S003          4.0          8.2           73.7         86      35.8
4      S004          3.5          4.8           95.1         66      34.0
5      S005          9.1          6.4           89.8         71      40.3
6      S006          8.4          5.1           58.5         75      35.7
> print(paste("Rows after removing duplicates:", nrow(df_no_dups)))
[1] "Rows after removing duplicates: 200"
> # =====
> # 4. REMOVING DUPLICATES BASED ON SPECIFIC COLUMNS
> # =====
> # Example: Keep FIRST occurrence of each student_id
> # If student_id repeats, only the FIRST record is kept.
>
> unique_students <- df %>%
+ distinct(student_id, .keep_all = TRUE)
>
> print("--- 4. Unique Students (Based on student_id) ---")
[1] "--- 4. Unique Students (Based on student_id) ---"
> print(head(unique_students))
  student_id hours_studied sleep_hours attendance_percent previous_scores exam_score
1      S001          8.0          8.8           72.1         45      30.2
2      S002          1.3          8.6           60.7         55      25.0
3      S003          4.0          8.2           73.7         86      35.8
4      S004          3.5          4.8           95.1         66      34.0
5      S005          9.1          6.4           89.8         71      40.3
6      S006          8.4          5.1           58.5         75      35.7
> print(paste("Total unique students:", nrow(unique_students)))
[1] "Total unique students: 200"
> # Another Example: Unique combination of (student_id, exam_score)
```

NAME: AASHKA MISHRA

ROLL NO: S095

# SHETH L.U.J. AND SIR M.V. COLLEGE

## SUBJECT: DATA ANALYSIS WITH R

The screenshot shows the RStudio interface with the following components:

- Source Editor:** Contains R code for data cleaning. The code includes comments and commands to remove duplicates based on specific columns (student\_id and exam\_score) using the `distinct()` function from the `dplyr` package. It also includes a `print()` statement to display the unique students.
- Console:** Shows the output of the R code, including the number of rows after removing duplicates (200) and the unique students (200).
- Environment Pane:** Displays the objects in the R environment, including `split_matrix`, `starts_with_s`, `student_clean`, `student_df`, `student_exam`, `tidy_df`, `unique_score`, and `unique_studen`.
- Files Pane:** Shows the file structure, including `askpass`, `backports`, `base64enc`, `bit`, `bit64`, `blob`, `brglm`, `BSDA`, `bslib`, `cachem`, `callr`, `cellranger`, `cli`, `clipr`, `conflicted`, `cpl11`, `crayon`, `curl`, and `data.table`.

```
R - R 4.5.2 - ~/R
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function Addins
Source
Console Terminal Background Jobs
R - R 4.5.2 - ~/R
> print(paste("Rows after removing duplicates:", nrow(df_no_dupes)))
[1] "Rows after removing duplicates: 200"
>
> # 4. REMOVING DUPLICATES BASED ON SPECIFIC COLUMNS
> # Example: keep FIRST occurrence of each student_id
> # If student_id repeats, only the FIRST record is kept.
>
> unique_students <- df %>%
+   distinct(student_id, .keep_all = TRUE)
>
> print("--- 4. Unique Students (Based on student_id) ---")
[1] "--- 4. Unique Students (Based on student_id) ---"
> print(head(unique_students))
  student_id hours_studied sleep_hours attendance_percent previous_scores exam_score
1      S001         8.0         8.8         72.1         45         30.2
2      S002         1.3         8.6         60.7         55         25.0
3      S003         4.0         8.2         73.7         86         35.8
4      S004         3.5         4.8         95.1         66         34.0
5      S005         9.1         6.4         89.8         71         40.3
6      S006         8.4         5.1         58.5         75         35.7
> print(paste("Total unique students:", nrow(unique_students)))
[1] "Total unique students: 200"
>
> # Another Example: Unique combination of (student_id, exam_score)
> unique_score_entries <- df %>%
+   distinct(student_id, exam_score, .keep_all = TRUE)
>
> print("--- 5. Unique (student_id + exam_score) combinations ---")
[1] "--- 5. Unique (student_id + exam_score) combinations ---"
> print(head(unique_score_entries))
  student_id hours_studied sleep_hours attendance_percent previous_scores exam_score
1      S001         8.0         8.8         72.1         45         30.2
2      S002         1.3         8.6         60.7         55         25.0
3      S003         4.0         8.2         73.7         86         35.8
4      S004         3.5         4.8         95.1         66         34.0
5      S005         9.1         6.4         89.8         71         40.3
6      S006         8.4         5.1         58.5         75         35.7
>
```

Environment History Connections Tutorial

R - Global Environment - 118 MB

split\_matrix chr [1:4362, 1:2] "NA" "NA" "NA" ...

starts\_with\_s 200 obs. of 2 variables

student\_clean 200 obs. of 6 variables

student\_df 200 obs. of 6 variables

student\_exam 200 obs. of 6 variables

tidy\_df 4362 obs. of 13 variables

unique\_score 200 obs. of 6 variables

unique\_studen 200 obs. of 6 variables

Files Plots Packages Help Viewer Presentation

Package Install Update Source Version

User Library

askpass CRAN 1.2.1

backports CRAN 1.5.0

base64enc CRAN 0.1-3

bit CRAN 4.6.0

bit64 CRAN 4.6.0-1

blob CRAN 1.2.4

brglm CRAN 10.10

BSDA CRAN 1.2.2

bslib CRAN 0.9.0

cachem CRAN 1.1.0

callr CRAN 3.7.6

cellranger CRAN 1.1.0

cli CRAN 3.6.5

clipr CRAN 0.8.0

conflicted CRAN 1.2.0

cpl11 CRAN 0.5.2

crayon CRAN 1.5.3

curl CRAN 7.0.0

data.table CRAN 1.17.8

23°C Mostly sunny

Search

ENG IN 11:08 08-12-2025

NAME: AASHKA MISHRA  
ROLL NO: S095

SHEETH L.U.J. AND SIR M.V. COLLEGE  
SUBJECT: DATA ANALYSIS WITH R

PRACTICAL NO. 14

AIM: Extracting date components using lubridate:: functions (R).

OUTPUT:

```
R - R 4.5.2 - ~/R/
> # -----
> # Extracting Date Components using lubridate (student dataset)
> # -----
> library(dplyr)

Attaching package: 'dplyr'

The following objects are masked from 'package:stats':
  filter, lag

The following objects are masked from 'package:base':
  intersect, setdiff, setequal, union

> library(lubridate)

Attaching package: 'lubridate'

The following objects are masked from 'package:base':
  date, intersect, setdiff, union

>
> # 1. Load your dataset
> df <- read.csv("D:/S095 Aashka/student_exam_scores.csv")
>
> # -----
> # 2. Add a sample exam date column (because your dataset has none)
> # -----
> # Example: Assign random exam dates in 2024
> set.seed(123) # for reproducibility
>
> df <- df %>%
+   mutate(
+     exam_date = sample(seq(ymd("2024-01-01"), ymd("2024-12-31")), by = "day"),
+                       size = n(), replace = TRUE)
+ )
>
```

```
> # -----
> # 3. Extract date components
> # -----
> df_processed <- df %>%
+   mutate(
+     Year = year(exam_date),
+     Month_number = month(exam_date),
+     Month_Name = month(exam_date, label = TRUE),
+     Day = day(exam_date),
+     weekday_number = wday(exam_date),
+     weekday_Name = wday(exam_date, label = TRUE, abbr = FALSE),
+     Quarter = quarter(exam_date),
+     Day_of_Year = yday(exam_date)
+   )
>
> # -----
> # 4. Print Results
> # -----
> print("---- Data with Extracted Date Components ----")
> print(df_processed)
[1] "---- Data with Extracted Date Components ----"
student_id hours_studied sleep_hours attendance_percent previous_scores exam_score exam_date Year Month_number Month_Name
1 S001 8.0 8.8 72.1 45 30.2 2024-06-27 2024 6 Jun
2 S002 1.3 8.6 60.7 55 25.0 2024-01-14 2024 1 Jan
3 S003 4.0 8.2 73.7 86 35.8 2024-07-13 2024 7 Jul
4 S004 3.5 4.8 95.1 66 34.0 2024-11-01 2024 11 Nov
5 S005 9.1 6.4 89.8 71 40.3 2024-04-27 2024 4 Apr
6 S006 8.4 5.1 58.5 75 35.7 2024-10-25 2024 10 Oct
7 S007 10.8 6.0 54.2 88 37.9 2024-08-16 2024 8 Aug
8 S008 2.0 4.3 75.8 55 18.3 2024-08-31 2024 8 Aug
9 S009 5.6 5.9 81.6 84 34.7 2024-01-14 2024 1 Jan
10 S010 1.3 8.9 66.8 70 24.7 2024-06-01 2024 6 Jun
11 S011 3.4 5.3 90.9 81 29.3 2024-03-30 2024 3 Mar
12 S012 6.6 7.9 87.6 85 35.1 2024-03-31 2024 3 Mar
13 S013 1.3 6.3 83.6 71 31.2 2024-09-12 2024 9 Sep
14 S014 3.2 6.1 61.2 68 30.2 2024-07-15 2024 7 Jul
15 S015 8.1 8.8 60.0 90 41.1 2024-03-31 2024 3 Mar
16 S016 7.0 9.0 51.2 41 34.1 2024-12-13 2024 12 Dec
17 S017 3.4 6.8 62.2 45 28.9 2024-05-16 2024 5 May
18 S018 7.5 7.6 73.8 58 36.3 2024-12-20 2024 12 Dec
19 S019 9.9 4.8 92.5 54 35.6 2024-11-23 2024 11 Nov
```

NAME: AASHKA MISHRA  
ROLL NO: S095



# SHETH L.U.J. AND SIR M.V. COLLEGE

## SUBJECT: DATA ANALYSIS WITH R

The image displays two screenshots of the RStudio interface, showing data analysis work. The top screenshot shows a data frame with columns: student\_id, hours\_studied, sleep\_hours, attendance\_percent, previous\_scores, exam\_score, exam\_date, Year, Month\_Number, and Month\_Name. The bottom screenshot shows a data frame with columns: day, weekday\_number, weekday\_name, quarter, and day\_of\_year. Both screenshots show the RStudio environment with the console, source editor, and package manager.

**Top Screenshot Data (Partial):**

student_id	hours_studied	sleep_hours	attendance_percent	previous_scores	exam_score	exam_date	Year	Month_Number	Month_Name
S001	8.0	8.8	72.1	45	30.2	2024-06-27	2024	6	Jun
S002	1.3	8.6	60.7	55	25.0	2024-01-14	2024	1	Jan
S003	4.0	8.2	73.7	86	35.8	2024-07-13	2024	7	Jul
S004	3.5	4.8	95.1	66	34.0	2024-11-01	2024	11	Nov
S005	9.1	6.4	89.8	71	40.3	2024-04-27	2024	4	Apr

**Bottom Screenshot Data (Partial):**

day	weekday_number	weekday_name	quarter	day_of_year
1	5	Thursday	2	179
2	14	Sunday	1	14
3	13	Saturday	3	195
4	1	Friday	4	306
5	27	Saturday	2	118

NAME: AASHKA MISHRA  
ROLL NO: S095

# SHETH L.U.J. AND SIR M.V. COLLEGE

## SUBJECT: DATA ANALYSIS WITH R

The image displays two screenshots of the RStudio interface, showing the execution of R code and the resulting environment.

**Top Screenshot:**

- Source:** The console shows the execution of a loop that generates data for each day of the week (Monday to Sunday) across 14 iterations. The output is a table with 14 rows and 5 columns.
- Environment:** The environment pane shows the following objects:
  - `df_appended`: 202 obs. of 6 variables
  - `df_high_atten...`: 101 obs. of 6 variables
  - `df_low_atten...`: 99 obs. of 6 variables
  - `df_no_dupes`: 200 obs. of 6 variables
  - `df_part1`: 200 obs. of 3 variables
  - `df_part2`: 200 obs. of 4 variables
  - `df_processed`: 200 obs. of 15 variables
  - `drop_multiple`: 200 obs. of 4 variables
  - `drop_one`: 200 obs. of 5 variables

**Bottom Screenshot:**

- Source:** The console shows the execution of a loop that generates data for each day of the week (Monday to Sunday) across 14 iterations. The output is a table with 14 rows and 5 columns.
- Environment:** The environment pane shows the following objects:
  - `df_appended`: 202 obs. of 6 variables
  - `df_high_atten...`: 101 obs. of 6 variables
  - `df_low_atten...`: 99 obs. of 6 variables
  - `df_no_dupes`: 200 obs. of 6 variables
  - `df_part1`: 200 obs. of 3 variables
  - `df_part2`: 200 obs. of 4 variables
  - `df_processed`: 200 obs. of 15 variables
  - `drop_multiple`: 200 obs. of 4 variables
  - `drop_one`: 200 obs. of 5 variables

NAME: AASHKA MISHRA  
ROLL NO: S095



# SHETH L.U.J. AND SIR M.V. COLLEGE

## SUBJECT: DATA ANALYSIS WITH R

### PRACTICAL NO. 15

AIM: Generating basic summaries using str() or summary() (R).

OUTPUT:

```
RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help

Source
Console Terminal Background Jobs

> # R Script: Generating Basic Summaries using str() and summary()
> # Dataset: student_exam_scores.csv
>
> # 1. SETUP: Load Dataset
> df <- read.csv("D:/S095 Aashka/student_exam_scores.csv")
>
> print("--- Data Loaded ---")
[1] "--- Data Loaded ---"
> print(head(df)) # Preview first rows
student_id hours_studied sleep_hours attendance_percent previous_scores exam_score
1 S001 8.0 8.8 72.1 45 30.2
2 S002 1.3 8.6 60.7 55 25.0
3 S003 4.0 8.2 73.7 86 35.8
4 S004 3.5 4.8 95.1 66 34.0
5 S005 9.1 6.4 89.8 71 40.3
6 S006 8.4 5.1 58.5 75 35.7
>
> # 2. USING str() - Structure of Dataset
> #
> print("--- OUTPUT OF str() ---")
[1] "--- OUTPUT OF str() ---"
> str(df)
'data.frame': 200 obs. of 6 variables:
 $ student_id : chr "S001" "S002" "S003" "S004" ...
 $ hours_studied : num 8.1 3.4 3.5 9.1 8.4 10.8 2.5 6.1 3.3 ...
 $ sleep_hours : num 8.8 8.6 8.2 4.8 6.4 5.1 6.4 3.5 9.8 ...
 $ attendance_percent : num 72.1 60.7 73.7 95.1 89.8 58.5 54.2 75.8 81.6 ...
 $ previous_scores : int 45 55 86 66 71 75 88 55 84 ...
 $ exam_score : num 30.2 25 35.8 34 40.3 35.7 37.9 18.3 34.7 ...
>
> # 3. USING summary() - Summary Statistics
> #
> print("--- OUTPUT OF summary() ---")
[1] "--- OUTPUT OF summary() ---"
> summary(df)
student_id      hours_studied      sleep_hours      attendance_percent      previous_scores      exam_score
Length:200      Min. : 1.000      Min. :4.000      Min. : 50.30      Min. :40.0      Min. :17.10
Class :character 1st Qu.: 3.500      1st Qu.:5.300      1st Qu.: 62.20      1st Qu.:54.0      1st Qu.:29.50
Mode :character  Median : 6.150      Median :6.700      Median : 75.25      Median :67.5      Median :34.05
>
23°C Mostly sunny 11:30 08-12-2025
```

```
RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help

Source
Console Terminal Background Jobs

>
> # 3. USING summary() - Summary Statistics
> #
> print("--- OUTPUT OF summary() ---")
[1] "--- OUTPUT OF summary() ---"
> summary(df)
student_id      hours_studied      sleep_hours      attendance_percent      previous_scores      exam_score
Length:200      Min. : 1.000      Min. :4.000      Min. : 50.30      Min. :40.0      Min. :17.10
Class :character 1st Qu.: 3.500      1st Qu.:5.300      1st Qu.: 62.20      1st Qu.:54.0      1st Qu.:29.50
Mode :character  Median : 6.150      Median :6.700      Median : 75.25      Median :67.5      Median :34.05
>
> # 4. IMPROVING summary() BY CONVERTING ID TO FACTOR (optional)
> # Reason: student_id is categorical, not numeric
> df$student_id <- as.factor(df$student_id)
>
> print("--- OUTPUT OF summary() After Converting student_id to Factor ---")
[1] "--- OUTPUT OF summary() After Converting student_id to Factor ---"
> summary(df)
student_id      hours_studied      sleep_hours      attendance_percent      previous_scores      exam_score
S001 : 1 Min. : 1.000      Min. :4.000      Min. : 50.30      Min. :40.0      Min. :17.10
S002 : 1 1st Qu.: 3.500      1st Qu.:5.300      1st Qu.: 62.20      1st Qu.:54.0      1st Qu.:29.50
S003 : 1 Median : 6.150      Median :6.700      Median : 75.25      Median :67.5      Median :34.05
S004 : 1 Mean : 6.325      Mean :6.622      Mean : 74.83      Mean :66.8      Mean :33.95
S005 : 1 3rd Qu.: 9.000      3rd Qu.:8.025      3rd Qu.: 87.42      3rd Qu.:80.0      3rd Qu.:38.75
S006 : 1 Max. :12.000      Max. :9.000      Max. :100.00      Max. :95.0      Max. :51.30
(Other):194
>
> # 5. Accessing Specific Summaries
> #
> avg_exam_score <- mean(df$exam_score)
> max_study_hours <- max(df$hours_studied)
>
> print(paste("Average Exam Score:", avg_exam_score))
[1] "Average Exam Score: 33.955"
> print(paste("Maximum Hours Studied:", max_study_hours))
[1] "Maximum Hours Studied: 12"
>
23°C Mostly sunny 11:31 08-12-2025
```

NAME: AASHKA MISHRA  
ROLL NO: S095

# SHETH L.U.J. AND SIR M.V. COLLEGE

## SUBJECT: DATA ANALYSIS WITH R

The screenshot displays the RStudio interface with the following components:

- Source Editor:** Contains R code for data analysis, including comments and function calls like `summary()`, `as.factor()`, and `mean()`.
- Console:** Shows the output of the R code, including summary statistics for the `df` dataset and the results of specific calculations.
- Environment:** Lists the objects in the environment, including `unique_student_id` and `wide_df`.
- Files:** Shows the file explorer with a list of files and folders.
- Plots:** Shows the plot explorer with a list of plots.
- Packages:** Shows the package manager with a list of installed and available packages.

**Console Output:**

```
> # 3. USING summary() - Summary Statistics
> # =====
> print("--- OUTPUT OF summary() ---")
[1] "--- OUTPUT OF summary() ---"
> summary(df)
  student_id      hours_studied      sleep_hours      attendance_percent      previous_scores      exam_score
Length:200      Min.   : 1.000      Min.   :4.000      Min.   : 50.30      Min.   :40.0      Min.   :17.10
Class:character      1st Qu.: 3.500      1st Qu.: 5.300      1st Qu.: 62.20      1st Qu.: 54.0      1st Qu.: 29.50
Mode :character      Median : 6.150      Median : 6.700      Median : 75.25      Median : 67.5      Median : 34.05
      Mean   : 6.325      Mean   : 6.622      Mean   : 74.83      Mean   : 66.8      Mean   : 33.95
      3rd Qu.: 9.000      3rd Qu.: 8.025      3rd Qu.: 87.42      3rd Qu.: 80.0      3rd Qu.: 38.75
      Max.   :12.000      Max.   : 9.000      Max.   :100.00      Max.   : 95.0      Max.   : 51.30

> # =====
> # 4. IMPROVING summary() BY CONVERTING ID TO FACTOR (optional)
> # Reason: student_id is categorical, not numeric
> # =====
> df$student_id <- as.factor(df$student_id)
>
> print("--- OUTPUT OF summary() After converting student_id to Factor ---")
[1] "--- OUTPUT OF summary() After converting student_id to Factor ---"
> summary(df)
  student_id      hours_studied      sleep_hours      attendance_percent      previous_scores      exam_score
S001 : 1      Min.   : 1.000      Min.   :4.000      Min.   : 50.30      Min.   :40.0      Min.   :17.10
S002 : 1      1st Qu.: 3.500      1st Qu.: 5.300      1st Qu.: 62.20      1st Qu.: 54.0      1st Qu.: 29.50
S003 : 1      Median : 6.150      Median : 6.700      Median : 75.25      Median : 67.5      Median : 34.05
S004 : 1      Mean   : 6.325      Mean   : 6.622      Mean   : 74.83      Mean   : 66.8      Mean   : 33.95
S005 : 1      3rd Qu.: 9.000      3rd Qu.: 8.025      3rd Qu.: 87.42      3rd Qu.: 80.0      3rd Qu.: 38.75
S006 : 1      Max.   :12.000      Max.   : 9.000      Max.   :100.00      Max.   : 95.0      Max.   : 51.30
(Other):194

> # =====
> # 5. Accessing Specific Summaries
> # =====
> avg_exam_score <- mean(df$exam_score)
> max_study_hours <- max(df$hours_studied)
>
> print(paste("Average Exam Score:", avg_exam_score))
[1] "Average Exam Score: 33.955"
> print(paste("Maximum Hours Studied:", max_study_hours))
[1] "Maximum Hours Studied: 12"
>
```

NAME: AASHKA MISHRA  
ROLL NO: S095