

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

PRACTICAL NO:11

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

CODE:

```
library(dplyr)
```

```
library(tidyr)
```

```
#
```

```
=====
```

```
=====
```

```
# 1. SETUP: Read Your Dataset
```

```
df <- read.csv("C:/Users/info/Downloads/Retail Product_ - Retail Product.csv",
```

```
             na.strings = c("", "NA")) %>%
```

```
mutate(ProductID = row_number()) %>%
```

```
select(ProductID, Category, Price, Discount)
```

```
print("--- 1. Original Wide Data ---")
```

```
print(head(df))
```

```
#
```

```
=====
```

```
=====
```

```
# 2. PIVOT_LONGER (Wide → Long)
```

```
#
```

```
=====
```

```
=====
```

```
long_df <- df %>%
```

```
  pivot_longer(
```

```
    cols = c(Price, Discount),
```

```
    names_to = "Metric",
```

```
    values_to = "Value"
```

```
)
```

```
print("--- 2. Long Format ---")
```

```
print(head(long_df, 6))
```

```
#
```

```
=====
```

```
=====
```

```
# 3. PIVOT_WIDER (Long → Wide)
```

RIYA RANE

S107 SYCS

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

```
#
=====

wide_df <- long_df %>%
  pivot_wider(
    names_from = Metric,
    values_from = Value
  )

print("--- 3. Wide Format (Back to Original) ---")
print(head(wide_df))

#
=====

# 4. ADVANCED EXAMPLE (Category Pivot)
#
=====

df_clean <- df %>%
  mutate(Category = ifelse(is.na(Category), "Unknown", Category))

category_pivot <- df_clean %>%
  select(ProductID, Category, Price) %>%
  pivot_wider(
    names_from = Category,
    values_from = Price
  )

print("--- 4. Category Pivot (Spreading Categories) ---")
print(head(category_pivot))
```

OUTPUT:

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

The screenshot displays an R Studio interface with the following components:

- Script Editor:** Contains R code for data manipulation. The code includes comments and function calls like `read.csv`, `pivot_longer`, `pivot_wider`, and `mutate`.
- Console:** Shows the execution output of the R code, including data frames and their dimensions.
- Environment Pane:** Lists the objects in the R environment, including data frames like `long_df`, `new_products`, and various datasets.
- Files Pane:** Shows the file explorer with a list of files and folders.

**Code Snippets:**

```
# 1. SETUP: Read Your Dataset
df <- read.csv("C:/Users/info/Downloads/Retail Product_ - Retail Product.csv",
+ na.strings = c("", "NA")) %>%
+ mutate(ProductID = row_number()) %>%
+ select(ProductID, Category, Price, Discount)

# 2. PIVOT_LONGER (Wide -> Long)
long_df <- df %>%
+ pivot_longer(
+ cols = c(Price, Discount),
+ names_to = "Metric",
+ values_to = "value"
+ )

# 3. PIVOT_WIDER (Long -> Wide)
wide_df <- long_df %>%
+ pivot_wider(
+ names_from = Metric,
+ values_from = value
+ )

# 4. ADVANCED EXAMPLE (Category Pivot)
df_clean <- df %>%
+ mutate(Category = ifelse(is.na(Category), "Unknown", Category))
category_pivot <- df_clean %>%
+ select(ProductID, Category, Price) %>%
+ pivot_wider(
+ names_from = Category,
+ values_from = Price
+ )
```

**Environment Pane Data:**

Object	Dimensions
long_df	8724 obs. of 4 variables
new_products	2 obs. of 4 variables
dropped_range	4362 obs. of 2 variables
employee_salary_data	50 obs. of 9 variables
final_dataset	4364 obs. of 9 variables
flower_dataset	10000 obs. of 4 variables
housing	4362 obs. of 5 variables
long_df	8724 obs. of 4 variables
new_products	2 obs. of 4 variables

**Console Output:**

```
[1] "---- 1. Original wide Data ----"
[1] "---- 2. Long Format ----"
# A tibble: 6 x 4
  ProductID Category Price Discount
  <int> <chr> <int> <int>
1 1 NA 5548 0
2 2 NA 3045 38
3 3 NA 4004 0
4 4 NA 4808 33
5 5 NA 1817 23
6 6 NA 3522 NA

[1] "---- 3. Wide Format (Back to original) ----"
# A tibble: 6 x 4
  ProductID Category Price Discount
  <int> <chr> <int> <int>
1 1 NA 5548 0
2 2 NA 3045 38
3 3 NA 4004 0
4 4 NA 4808 33
5 5 NA 1817 23
6 6 NA 3522 NA

[1] "---- 4. Category Pivot (Spreading Categories) ----"
# A tibble: 6 x 6
  ProductID Unknown C A B D
  <int> <chr> <int> <int> <int> <int>
1 1 5548 NA NA NA NA
2 2 3045 NA NA NA NA
3 3 4004 NA NA NA NA
4 4 4808 NA NA NA NA
5 5 1817 NA NA NA NA
6 6 3522 NA NA NA NA
```

RIYA RANE  
S107 SYCS

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

PRACTICAL NO:12

12 Combining datasets vertically (concatenation) using rbind() (R).

CODE:

```
#
=====
=====
# R Script: Vertical Concatenation using rbind()
#
=====
=====

# 1.1 Load the built-in iris dataset
data(iris)

# 1.2 Load your custom CSV files (FIXED PATHS)
iris_file <- read.csv("C:/Users/info/Downloads/Iris_ - Iris.csv")
flower_df <- read.csv("C:/Users/info/Downloads/flower_dataset__ - flower_dataset.csv")

print("--- Data Structure Before Transformation ---")
print(names(iris))
print(names(flower_df))

#
=====
=====
# 2. DATA PREPARATION (Aligning Columns)
#
=====
=====

# Iris preparation
iris_clean <- iris[, c("Species", "Sepal.Length")]
names(iris_clean) <- c("Species", "Height")

# Flower dataset preparation
flower_clean <- flower_df[, c("species", "height_cm")]
names(flower_clean) <- c("Species", "Height")

# Convert to numeric
iris_clean$Height <- as.numeric(iris_clean$Height)
flower_clean$Height <- as.numeric(flower_clean$Height)
```

RIYA RANE  
S107 SYCS

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

#

=====

=====

# 3. VERTICAL COMBINATION

#

=====

=====

```
combined_data <- rbind(iris_clean, flower_clean)
```

```
print("--- Combined Data Summary ---")
print(paste("Iris rows:", nrow(iris_clean)))
print(paste("Flower rows:", nrow(flower_clean)))
print(paste("Total rows:", nrow(combined_data)))
print("--- Preview ---")
print(head(combined_data))
print(tail(combined_data))
```

OUTPUT:

The screenshot displays the RStudio interface. The console on the left shows the execution of R code that loads two CSV files, cleans them, and combines them vertically using `rbind()`. The environment pane on the right lists the objects created, including `iris_clean` (150 obs. of 2 variables) and `flower_clean` (150 obs. of 6 variables). The file explorer at the bottom shows the project directory structure.

```
> # 1.2 Load your custom CSV files (FIXED PATHS)
> iris_file <- read.csv("C:/Users/info/Downloads/Iris_ - Iris.csv")
> # 1.2 Load your custom CSV files (FIXED PATHS)
> iris_file <- read.csv("C:/Users/info/Downloads/Iris_ - Iris.csv")
> flower_df <- read.csv("C:/Users/info/Downloads/flower_dataset_ - flower_dataset.csv")
>
> print("--- Data Structure Before Transformation ---")
[1] "---- Data Structure Before Transformation ----"
> print(names(iris))
[1] "Sepal.Length" "Sepal.width" "Petal.Length" "Petal.width" "Species"
> print(names(flower_df))
[1] "species" "size" "fragrance" "height_cm"
> # =====
> # 2. DATA PREPARATION (Aligning columns)
> # =====
> # Iris preparation
> iris_clean <- iris[, c("Species", "Sepal.Length")]
> names(iris_clean) <- c("Species", "Height")
> # =====
> # R Script: Vertical concatenation using rbind()
> # =====
> # 1.1 Load the built-in iris dataset
> data(iris)
>
> # 1.2 Load your custom CSV files (FIXED PATHS)
> iris_file <- read.csv("C:/Users/info/Downloads/Iris_ - Iris.csv")
> flower_df <- read.csv("C:/Users/info/Downloads/flower_dataset_ - flower_dataset.csv")
>
> print("--- Data Structure Before Transformation ---")
[1] "---- Data Structure Before Transformation ----"
> print(names(iris))
[1] "Sepal.Length" "Sepal.width" "Petal.Length" "Petal.width" "Species"
> print(names(flower_df))
[1] "species" "size" "fragrance" "height_cm"
```

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

The screenshot shows the RStudio interface with the following code in the console:

```
R > R452 ~ />
> print(names(flower_df))
[1] "species" "size" "fragrance" "height_cm"
> # =====
> # 2. DATA PREPARATION (Aligning Columns)
> # =====
> # Iris preparation
> iris_clean <- iris[, c("Species", "Sepal.Length")]
> names(iris_clean) <- c("Species", "Height")
> # Flower dataset preparation
> flower_clean <- flower_df[, c("species", "height_cm")]
> names(flower_clean) <- c("Species", "Height")
> # Convert to numeric
> iris_clean$Height <- as.numeric(iris_clean$Height)
> flower_clean$Height <- as.numeric(flower_clean$Height)
> # =====
> # 3. VERTICAL COMBINATION
> # =====
> combined_data <- rbind(iris_clean, flower_clean)
> print("---- Combined Data Summary ----")
[1] "---- 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:", nrow(combined_data)))
[1] "Total rows: 10150"
> print("---- Preview ----")
[1] "---- Preview ----"
> print(head(combined_data))
  Species Height
1 setosa    5.1
2 setosa    4.9
3 setosa    4.7
```

The Environment pane on the right shows the following objects:

Object	Size
iris...Iris	150 obs. of 6 variables
iris_clean	150 obs. of 2 variables
iris_file	150 obs. of 6 variables
long_df	8724 obs. of 4 variables
new_products	2 obs. of 4 variables
range_cols	4362 obs. of 3 variables
retail_df	4362 obs. of 5 variables
retail.Product....Ret...	4362 obs. of 5 variables
retail.Product.107....	4362 obs. of 5 variables
selected_cols	4362 obs. of 3 variables

The screenshot shows the RStudio interface with the following code in the console:

```
R > R452 ~ />
> names(flower_clean) <- c("Species", "Height")
> # convert to numeric
> iris_clean$Height <- as.numeric(iris_clean$Height)
> flower_clean$Height <- as.numeric(flower_clean$Height)
> # =====
> # 3. VERTICAL COMBINATION
> # =====
> combined_data <- rbind(iris_clean, flower_clean)
> print("---- Combined Data Summary ----")
[1] "---- 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:", nrow(combined_data)))
[1] "Total rows: 10150"
> print("---- Preview ----")
[1] "---- Preview ----"
> print(head(combined_data))
  Species Height
1 setosa    5.1
2 setosa    4.9
3 setosa    4.7
4 setosa    4.6
5 setosa    5.0
6 setosa    5.4
> print(tail(combined_data))
  Species Height
10145  rose    87.69
10146 hhibiscus 109.52
10147 shoeblack plant 145.23
10148 hhibiscus 126.69
10149 shoeblack plant 77.62
10150  rose    88.11
> <
```

The Environment pane on the right shows the following objects:

Object	Size
iris...Iris	150 obs. of 6 variables
iris_clean	150 obs. of 2 variables
iris_file	150 obs. of 6 variables
long_df	8724 obs. of 4 variables
new_products	2 obs. of 4 variables
range_cols	4362 obs. of 3 variables
retail_df	4362 obs. of 5 variables
retail.Product....Ret...	4362 obs. of 5 variables
retail.Product.107....	4362 obs. of 5 variables
selected_cols	4362 obs. of 3 variables

## PRACTICAL NO:13

13. Identifying and handling duplicates using distinct() (R studio ).

CODE:

RIYA RANE  
S107 SYCS

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

```
#
=====
=====
# R Script: Identifying and Handling Duplicates
# Function: distinct() from the dplyr package
#
=====
=====

# Load library
library(dplyr)

#
=====
=====
# 1. SETUP: Create a Dataset with Intentional Duplicates

orders_df <- data.frame(
  OrderID = c(101, 102, 102, 103, 104, 101, 104),
  Customer = c("NISHITA", "YUKTA", "SIYA", "NANDINI", "SIMRAN", "PURVI", "GAZALA"),
  Product = c("Laptop", "Phone", "Phone", "Tablet", "Monitor", "Laptop", "Mouse")
)

print("--- 1. Original Dataset (Note 7 rows) ---")
print(orders_df)

#
=====
=====
# 2. IDENTIFYING DUPLICATES
#
=====
=====

duplicates_report <- orders_df %>%
  group_by(OrderID, Customer, Product) %>%
  count() %>%      # Count occurrences
  filter(n > 1)     # Keep only rows that appear more than once

print("--- 2. Identification Report (Rows that are duplicated) ---")
print(duplicates_report)
```

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

```
#
=====
=====
# 3. HANDLING EXACT DUPLICATES
#
=====
=====

clean_exact <- orders_df %>%
  distinct()      # Removes only exact row matches

print("--- 3. Removed Exact Duplicates (distinct) ---")
print(clean_exact)

#
=====
=====
# 4. HANDLING DUPLICATES BASED ON SPECIFIC COLUMN
#
=====
=====

unique_customers <- orders_df %>%
  distinct(Customer, .keep_all = TRUE) # Keeps first appearance of each customer

print("--- 4. Unique Customers Only (Partial duplicates removed) ---")
print(unique_customers)
```

OUTPUT:



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

```
RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function Addins

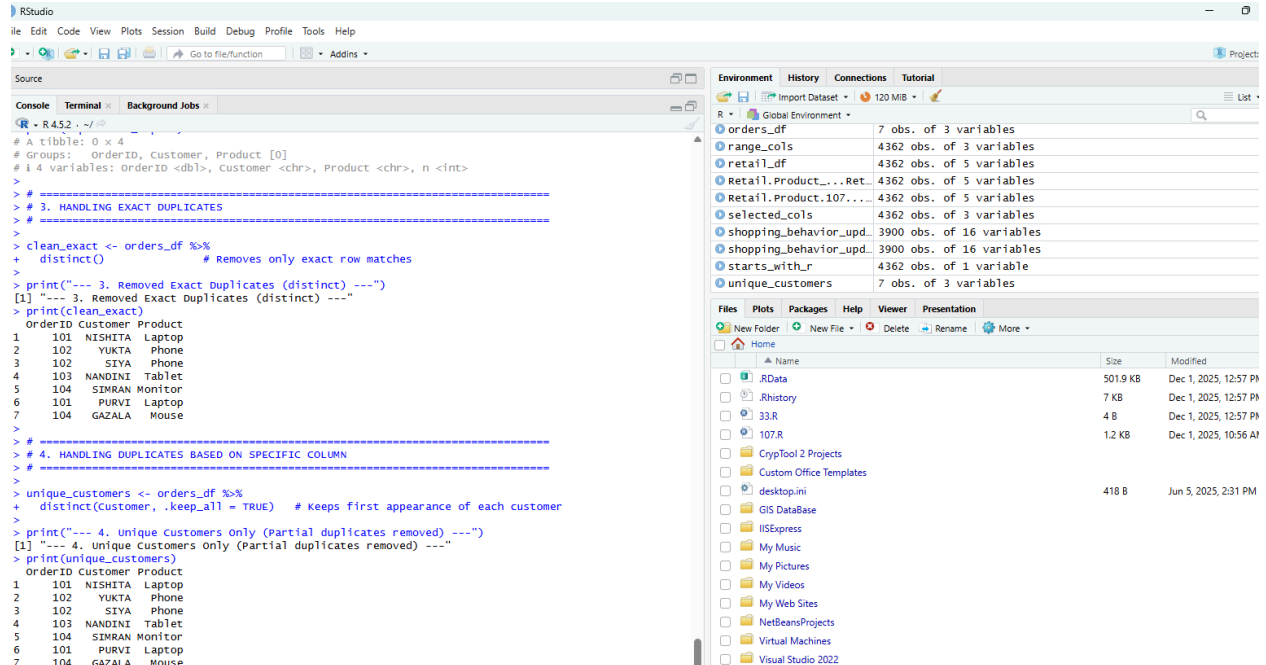
Source
Console Terminal Background Jobs
R - R 4.5.2 - ~/RStudio
1 # R script: Identifying and Handling Duplicates
2 # Function: distinct() from the dplyr package
3 #
4 # Load library
5 library(dplyr)
6
7 # 1. SETUP: create a dataset with Intentional Duplicates
8
9 orders_df <- data.frame(
10   orderID = c(101, 102, 102, 103, 104, 101, 104),
11   customer = c("NISHITA", "YUKTA", "SIYA", "NANDINI", "SIMRAN", "PURVI", "GAZALA"),
12   product = c("Laptop", "Phone", "Phone", "Tablet", "Monitor", "Laptop", "Mouse")
13 )
14
15 print("--- 1. Original dataset (Note 7 rows) ---")
16 [1] "--- 1. Original Dataset (Note 7 rows) ---"
17 > print(orders_df)
18   orderID customer product
19 1     101  NISHITA  Laptop
20 2     102   YUKTA   Phone
21 3     102   SIYA   Phone
22 4     103  NANDINI Tablet
23 5     104  SIMRAN Monitor
24 6     101   PURVI  Laptop
25 7     104   GAZALA  Mouse
26
27 # 2. IDENTIFYING DUPLICATES
28
29 duplicates_report <- orders_df %>%
30   group_by(orderID, customer, product) %>%
31   count() %>%
32   filter(n > 1)
33
34 print("--- 2. Identification Report (Rows that are duplicated) ---")
35 [1] "--- 2. Identification Report (Rows that are duplicated) ---"
36 > print(duplicates_report)
37 # A tibble: 0 x 4
38 # Groups:   orderID, customer, product [0]
39 #   orderID customer product n
40   <dbl>   <chr>   <chr> <int>
41
42 # 3. HANDLING EXACT DUPLICATES
43
44 clean_exact <- orders_df %>%
45   distinct()
46
47 print("--- 3. Removed Exact Duplicates (distinct) ---")
48 [1] "--- 3. Removed Exact Duplicates (distinct) ---"
49 > print(clean_exact)
50   orderID customer product
51 1     101  NISHITA  Laptop
52 2     102   YUKTA   Phone
53 3     102   SIYA   Phone
54 4     103  NANDINI Tablet
55 5     104  SIMRAN Monitor
56 6     101   PURVI  Laptop
57 7     104   GAZALA  Mouse
58
59 # 4. HANDLING DUPLICATES BASED ON SPECIFIC COLUMN
60
61 unique_customers <- orders_df %>%
62   distinct(customer)
63
64 print("--- 4. Unique Customers ---")
65 [1] "--- 4. Unique Customers ---"
66 > print(unique_customers)
67 # A tibble: 7 x 1
68 #   customer
69   <chr>
70 1 NISHITA
71 2 YUKTA
72 3 SIYA
73 4 NANDINI
74 5 SIMRAN
75 6 PURVI
76 7 GAZALA
```

```
RStudio
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 - ~/RStudio
1 # R script: Identifying and Handling Duplicates
2 # Function: distinct() from the dplyr package
3 #
4 # Load library
5 library(dplyr)
6
7 # 1. SETUP: create a dataset with Intentional Duplicates
8
9 orders_df <- data.frame(
10   orderID = c(101, 102, 102, 103, 104, 101, 104),
11   customer = c("NISHITA", "YUKTA", "SIYA", "NANDINI", "SIMRAN", "PURVI", "GAZALA"),
12   product = c("Laptop", "Phone", "Phone", "Tablet", "Monitor", "Laptop", "Mouse")
13 )
14
15 print("--- 1. Original dataset (Note 7 rows) ---")
16 [1] "--- 1. Original Dataset (Note 7 rows) ---"
17 > print(orders_df)
18   orderID customer product
19 1     101  NISHITA  Laptop
20 2     102   YUKTA   Phone
21 3     102   SIYA   Phone
22 4     103  NANDINI Tablet
23 5     104  SIMRAN Monitor
24 6     101   PURVI  Laptop
25 7     104   GAZALA  Mouse
26
27 # 2. IDENTIFYING DUPLICATES
28
29 duplicates_report <- orders_df %>%
30   group_by(orderID, customer, product) %>%
31   count() %>%
32   filter(n > 1)
33
34 print("--- 2. Identification Report (Rows that are duplicated) ---")
35 [1] "--- 2. Identification Report (Rows that are duplicated) ---"
36 > print(duplicates_report)
37 # A tibble: 0 x 4
38 # Groups:   orderID, customer, product [0]
39 #   orderID customer product n
40   <dbl>   <chr>   <chr> <int>
41
42 # 3. HANDLING EXACT DUPLICATES
43
44 clean_exact <- orders_df %>%
45   distinct()
46
47 print("--- 3. Removed Exact Duplicates (distinct) ---")
48 [1] "--- 3. Removed Exact Duplicates (distinct) ---"
49 > print(clean_exact)
50   orderID customer product
51 1     101  NISHITA  Laptop
52 2     102   YUKTA   Phone
53 3     102   SIYA   Phone
54 4     103  NANDINI Tablet
55 5     104  SIMRAN Monitor
56 6     101   PURVI  Laptop
57 7     104   GAZALA  Mouse
58
59 # 4. HANDLING DUPLICATES BASED ON SPECIFIC COLUMN
60
61 unique_customers <- orders_df %>%
62   distinct(customer)
63
64 print("--- 4. Unique Customers ---")
65 [1] "--- 4. Unique Customers ---"
66 > print(unique_customers)
67 # A tibble: 7 x 1
68 #   customer
69   <chr>
70 1 NISHITA
71 2 YUKTA
72 3 SIYA
73 4 NANDINI
74 5 SIMRAN
75 6 PURVI
76 7 GAZALA
```

RIYA RANE  
S107 SYCS

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



The screenshot shows the RStudio interface. The console on the left displays the following R code and its output:

```
# A tibble: 0 x 4
# Groups:   orderID, customer, Product [0]
# i 4 variables: orderID <dbl>, customer <chr>, Product <chr>, n <int>
>
> # =====
> # 3. HANDLING EXACT DUPLICATES
> # =====
> clean_exact <- orders_df %>%
+   distinct()           # Removes only exact row matches
>
> print("--- 3. Removed Exact Duplicates (distinct) ---")
[1] "--- 3. Removed Exact Duplicates (distinct) ---"
> print(clean_exact)
  orderID Customer Product
1     101  NISHITA  Laptop
2     102   YUKTA   Phone
3     102   SIYA    Phone
4     103  NANDINI  Tablet
5     104  SIMRAN   Monitor
6     101   PURVI   Laptop
7     104   GAZALA   Mouse
>
> # =====
> # 4. HANDLING DUPLICATES BASED ON SPECIFIC COLUMN
> # =====
> unique_customers <- orders_df %>%
+   distinct(customer, .keep_all = TRUE) # Keeps first appearance of each customer
>
> print("--- 4. Unique Customers only (Partial duplicates removed) ---")
[1] "--- 4. Unique Customers only (Partial duplicates removed) ---"
> print(unique_customers)
  orderID Customer Product
1     101  NISHITA  Laptop
2     102   YUKTA   Phone
3     102   SIYA    Phone
4     103  NANDINI  Tablet
5     104  SIMRAN   Monitor
6     101   PURVI   Laptop
7     104   GAZALA   Mouse
```

The Environment pane on the right shows the following objects:

Object	Size
orders_df	7 obs. of 3 variables
range_cols	4362 obs. of 3 variables
retail_df	4362 obs. of 5 variables
Retail.Product...Ret...	4362 obs. of 5 variables
Retail.Product.107...	4362 obs. of 5 variables
selected_cols	4362 obs. of 3 variables
shopping_behavior_upd...	3900 obs. of 16 variables
shopping_behavior_upd...	3900 obs. of 16 variables
starts_with_r	4362 obs. of 1 variable
unique_customers	7 obs. of 3 variables

## PRACTICAL NO:6

#

# R Script: Extracting Date Components using lubridate

#

# Install and load necessary libraries

install.packages("lubridate")

library(lubridate)

library(dplyr)

#

# 1. SETUP: Create Sample Data

#

```
dates_df <- data.frame(
  Event_ID = 1:4,
  Date_String = c("2023-01-15", "2023-10-31", "2024-02-29", "2024-12-25")
)
```

RIYA RANE

S107 SYCS

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

```
)

#
=====
=====
# 2. PARSE AND EXTRACT
#
=====
=====

processed_data <- dates_df %>%
  mutate(
    # A. Parsing: Convert text to actual Date
    Actual_Date = ymd(Date_String),

    # B. Extraction Components
    Year_Num   = year(Actual_Date),      # Year (e.g., 2023)
    Month_Num  = month(Actual_Date),     # Month number (1–12)
    Month_Name = month(Actual_Date, label = TRUE), # Month abbreviation
    Day_Num    = day(Actual_Date),       # Day (1–31)
    Weekday_Num = wday(Actual_Date),     # Day of week (1=Sun)
    Weekday_Name = wday(Actual_Date, label = TRUE, abbr = FALSE), # Full Name
    Quarter    = quarter(Actual_Date),  # Quarter (1–4)
    Day_of_Year = yday(Actual_Date)     # Day of the year (1–366)
  )

print("--- Data with Extracted Date Components ---")
print(processed_data)

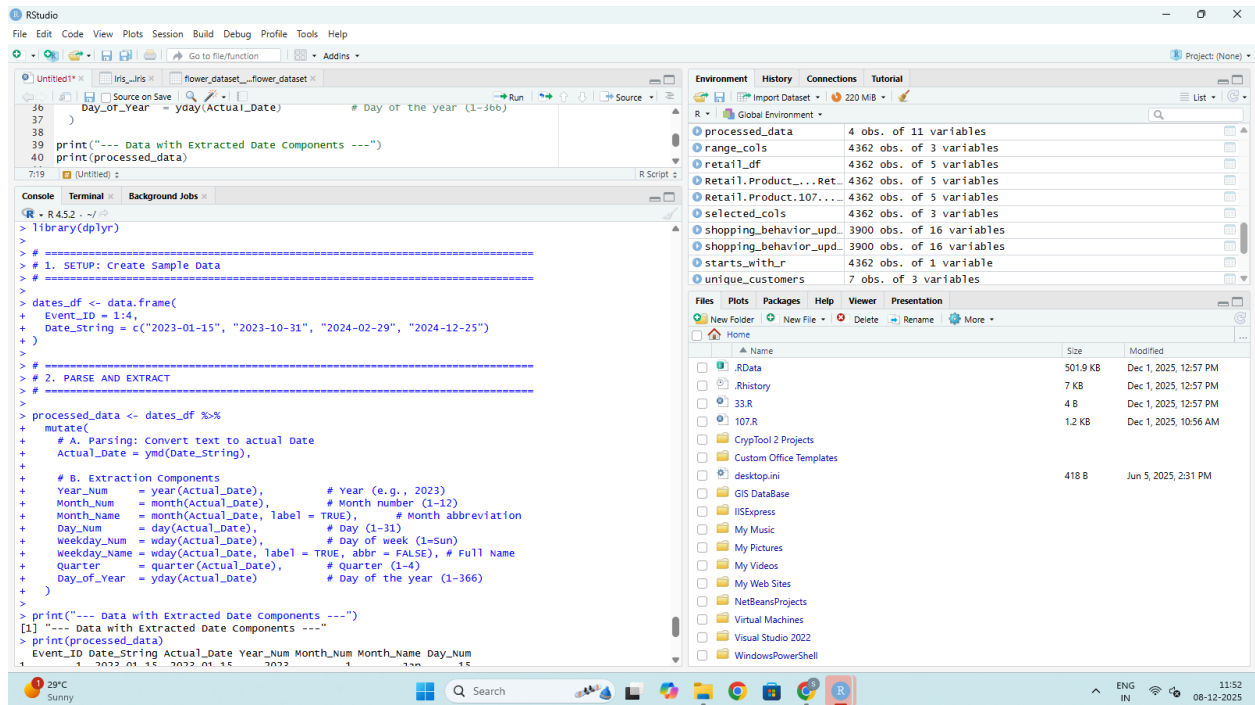
#
=====
=====
# 3. SYSTEM DATE & TIME (now)
#
=====
=====

current_time <- now()

print("--- Current Time Extraction ---")
print(paste("Current Year:", year(current_time)))
print(paste("Current Hour:", hour(current_time)))
print(paste("Current Minute:", minute(current_time)))
```

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

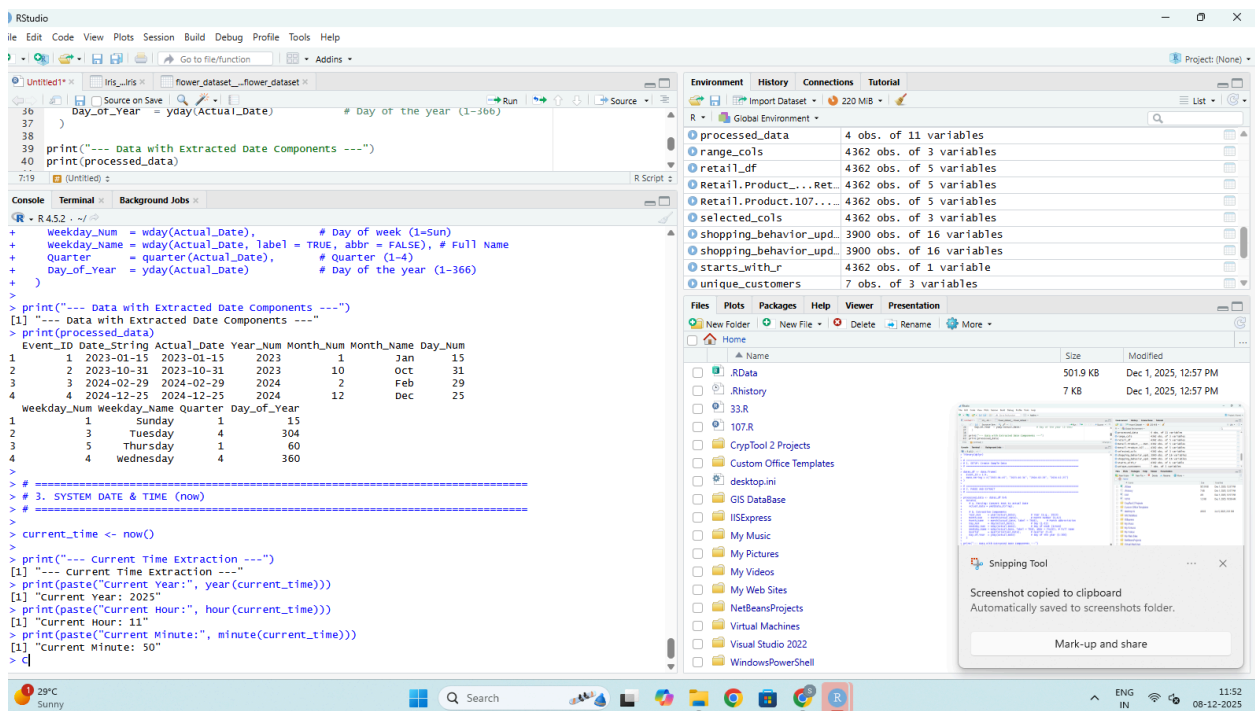
OUTPUT:



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

# 1. SETUP: Create Sample Data
# 2. PARSE AND EXTRACT

> library(dplyr)
> # 1. SETUP: Create Sample Data
> # 2. PARSE AND EXTRACT
> dates_df <- data.frame(
+   Event_ID = 1:4,
+   Date_String = c("2023-01-15", "2023-10-31", "2024-02-29", "2024-12-25")
+ )
> # 2. PARSE AND EXTRACT
> processed_data <- dates_df %>%
+   mutate(
+     # A. Parsing: Convert text to actual date
+     Actual_Date = ymd(Date_String),
+     # B. Extraction Components
+     Year_Num = year(Actual_Date), # Year (e.g., 2023)
+     Month_Num = month(Actual_Date), # Month number (1-12)
+     Month_Name = month(Actual_Date, label = TRUE), # Month abbreviation
+     Day_Num = day(Actual_Date), # Day (1-31)
+     weekday_Num = wday(Actual_Date), # Day of week (1=Sun)
+     weekday_Name = wday(Actual_Date, label = TRUE, abbr = FALSE), # Full Name
+     Quarter = quarter(Actual_Date), # Quarter (1-4)
+     Day_of_Year = yday(Actual_Date), # Day of the year (1-366)
+   )
> print("--- Data with Extracted Date Components ---")
[1] "--- Data with Extracted Date Components ---"
> print(processed_data)
  Event_ID Date_String Actual_Date Year_Num Month_Num Month_Name Day_Num
1         1 2023-01-15 2023-01-15 2023         1         Jan        15
2         2 2023-10-31 2023-10-31 2023        10         Oct        31
3         3 2024-02-29 2024-02-29 2024         2         Feb        29
4         4 2024-12-25 2024-12-25 2024        12         Dec        25
  weekday_Num weekday_Name Quarter Day_of_Year
1           1      Sunday         1          15
2           3    Tuesday         4         304
3           5   Thursday         1          60
4           4   Wednesday         4         360
```



```
# 3. SYSTEM DATE & TIME (now)
> current_time <- now()
> print("--- Current Time Extraction ---")
[1] "--- Current Time Extraction ---"
> print(paste("Current Year:", year(current_time)))
[1] "Current Year: 2025"
> print(paste("Current Hour:", hour(current_time)))
[1] "Current Hour: 11"
> print(paste("Current Minute:", minute(current_time)))
[1] "Current Minute: 50"
```

PRACTICAL NO:15

CODE:

RIYA RANE  
S107 SYCS

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

```
#
=====

=====
# R Script: Generating Basic Summaries
# Functions: str() and summary()
# Dataset: Retail Product Data (Simulated)
#
=====

=====

#
=====

=====
# 1. SETUP: Create Sample Data
#
=====

=====

retail_df <- data.frame(
  ID = 1:6,
  Category = c("Electronics", "Home", "Electronics", "Clothing", "Home", "Clothing"),
  Price = c(500.50, 45.00, 900.00, NA, 300.00, 25.00), # Note NA
  In_Stock = c(TRUE, TRUE, FALSE, TRUE, FALSE, TRUE),
  Rating = c(4.5, 3.8, 4.9, 4.0, 3.5, 4.2)
)

print("--- Data Loaded ---")
print(retail_df)

#
=====

=====
# 2. USING str() — Structure of the Data
#
=====

=====

print("--- OUTPUT OF str() ---")
str(retail_df)

#
=====

=====
# 3. USING summary() — Statistical Summary
```

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

```
#
=====

=====

print("--- OUTPUT OF summary() [Before Factor Conversion] ---")
summary(retail_df)

#
=====

=====

# 4. IMPROVING summary() WITH FACTORS
#
=====

=====

# Convert Category from character → factor for meaningful summary counts
retail_df$Category <- as.factor(retail_df$Category)

print("--- OUTPUT OF summary() [After Factor Conversion] ---")
summary(retail_df)

#
=====

=====

# 5. Accessing Specific Summaries
#
=====

=====

avg_rating <- mean(retail_df$Rating)          # Average rating
max_price <- max(retail_df$Price, na.rm = TRUE)  # Ignore NA

print(paste("Average Rating:", avg_rating))
print(paste("Highest Price:", max_price))
```

OUTPUT:

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

```
RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function Addins

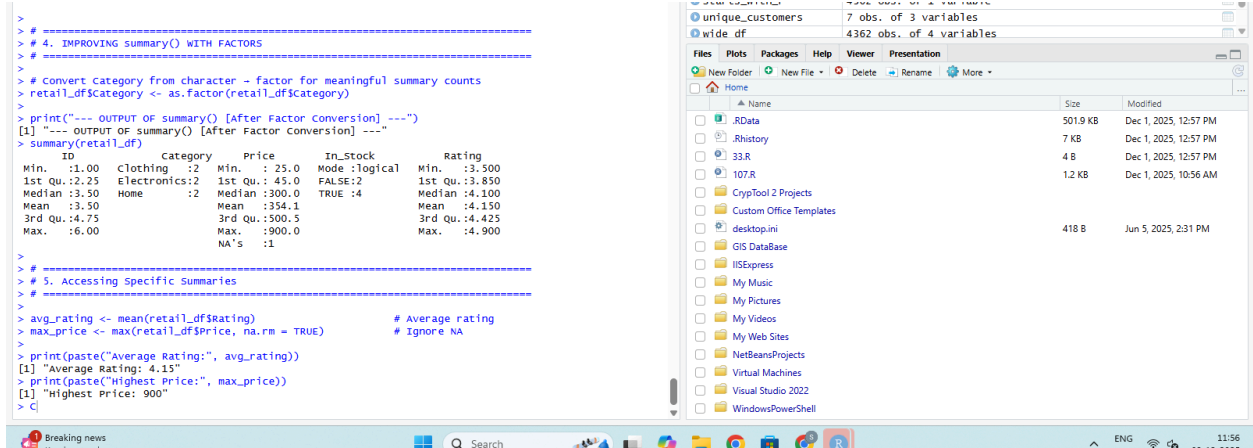
Source
Console Terminal Background Jobs
R - R 4.5.2 - ~/
> # 1. SETUP: Create Sample Data
> # =====
> retail_df <- data.frame(
+   ID = 1:6,
+   Category = c("Electronics", "Home", "Electronics", "Clothing", "Home", "Clothing"),
+   Price = c(500.50, 45.00, 900.00, NA, 300.00, 25.00), # Note NA
+   In_Stock = c(TRUE, TRUE, FALSE, TRUE, FALSE, TRUE),
+   Rating = c(4.5, 3.8, 4.9, 4.0, 3.5, 4.2)
+ )
>
> print("--- Data Loaded ---")
[1] "--- Data Loaded ---"
> print(retail_df)
  ID Category Price In_Stock Rating
1 1 Electronics 500.5    TRUE   4.5
2 2      Home   45.0    TRUE   3.8
3 3 Electronics 900.0   FALSE   4.9
4 4 Clothing    NA    TRUE   4.0
5 5      Home  300.0   FALSE   3.5
6 6 Clothing   25.0    TRUE   4.2
>
> # =====
> # 2. USING str() - Structure of the Data
> # =====
>
> print("--- OUTPUT OF str() ---")
[1] "--- OUTPUT OF str() ---"
> str(retail_df)
'data.frame':   6 obs. of  5 variables:
 $ ID      : int  1 2 3 4 5 6
 $ Category: chr  "Electronics" "Home" "Electronics" "Clothing" ...
 $ Price   : num  500 45 900 NA 300 ...
 $ In_Stock: logi  TRUE TRUE FALSE TRUE FALSE TRUE
 $ Rating  : num  4.5 3.8 4.9 4.0 3.5 4.2
>
> # =====
> # 3. USING summary() - Statistical Summary
> # =====
>
> print("--- OUTPUT OF summary() [Before Factor Conversion] ---")
[1] "--- OUTPUT OF summary() [Before Factor Conversion] ---"
> summary(retail_df)
```

```
RStudio
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 - ~/
> # 3. USING summary() - Statistical Summary
> # =====
>
> print("--- OUTPUT OF summary() [Before Factor Conversion] ---")
[1] "--- OUTPUT OF summary() [Before Factor Conversion] ---"
> summary(retail_df)
  ID          Category      Price      In_Stock      Rating
Min. :1.00   Length:6      Min. : 25.0      Mode :logical Min. :3.500
1st Qu.:2.25  Class :character 1st Qu.: 45.0      FALSE:2      1st Qu.:3.850
Median :3.50  Mode :character   Median :300.0     TRUE :4       Median :4.100
Mean :3.50                                Mean :354.1      Mean :4.150
3rd Qu.:4.75                                3rd Qu.:500.5    3rd Qu.:4.425
Max. :6.00                                Max. :900.0      Max. :4.900
NA's :1
>
> # =====
> # 4. IMPROVING summary() WITH FACTORS
> # =====
>
> # Convert Category from character + factor for meaningful summary counts
> retail_df$Category <- as.factor(retail_df$Category)
>
> print("--- OUTPUT OF summary() [After Factor Conversion] ---")
[1] "--- OUTPUT OF summary() [After Factor Conversion] ---"
> summary(retail_df)
  ID          Category      Price      In_Stock      Rating
Min. :1.00   clothing :2      Min. : 25.0      Mode :logical Min. :3.500
1st Qu.:2.25  Electronics:2      1st Qu.: 45.0      FALSE:2      1st Qu.:3.850
Median :3.50      Home :2       Median :300.0     TRUE :4       Median :4.100
Mean :3.50                                Mean :354.1      Mean :4.150
3rd Qu.:4.75                                3rd Qu.:500.5    3rd Qu.:4.425
Max. :6.00                                Max. :900.0      Max. :4.900
NA's :1
>
> # =====
```

RIYA RANE  
S107 SYCS

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



The screenshot displays the RStudio interface. The left pane shows R code being executed, and the right pane shows the output of the code and a file explorer window.

**R Code:**

```
> # =====  
> # 4. IMPROVING summary() WITH FACTORS  
> # =====  
> # Convert Category from character + factor for meaningful summary counts  
> retail_df$category <- as.factor(retail_df$category)  
> # =====  
> print("---- OUTPUT OF summary() [After Factor Conversion] ----")  
[1] "---- OUTPUT OF summary() [After Factor Conversion] ----"  
> summary(retail_df)
```

**Output of summary(retail\_df):**

ID	Category	Price	In_Stock	Rating	
Min.	:1.00	clothing :2	Min. : 25.0	Mode :logical	Min. :3.500
1st Qu.	:2.25	Electronics:2	1st Qu.: 45.0	FALSE:2	1st Qu.:3.850
Median	:3.50	Home :2	Median :300.0	TRUE :4	Median :4.100
Mean	:3.50		Mean :354.1		Mean :4.150
3rd Qu.	:4.75		3rd Qu.:500.5		3rd Qu.:4.425
Max.	:6.00		Max. :900.0		Max. :4.900
			NA's :1		

**Additional R Code and Output:**

```
> # =====  
> # 5. Accessing Specific Summaries  
> # =====  
> avg_rating <- mean(retail_df$rating) # Average rating  
> max_price <- max(retail_df$price, na.rm = TRUE) # Ignore NA  
> # =====  
> print(paste("Average Rating:", avg_rating))  
[1] "Average Rating: 4.15"  
> print(paste("Highest Price:", max_price))  
[1] "Highest Price: 900"  
> C
```

**File Explorer Window:**

Name	Size	Modified
.RData	501.9 KB	Dec 1, 2025, 12:57 PM
.Rhistory	7 KB	Dec 1, 2025, 12:57 PM
33.R	4 B	Dec 1, 2025, 12:57 PM
107.R	1.2 KB	Dec 1, 2025, 10:56 AM
Cryptool 2 Projects		
Custom Office Templates		
desktop.ini	418 B	Jun 5, 2025, 2:31 PM
GIS DataBase		
IISExpress		
My Music		
My Pictures		
My Videos		
My Web Sites		
NetBeansProjects		
Virtual Machines		
Visual Studio 2022		
WindowsPowerShell		