Name of Student: Pushkar Sane		
Roll Number: 45	**Lab Assignment Number: 3**	
Title of Lab Assignment: **Write commands for Implementation of Data Preprocessing Techniques like:** **a. Naming and renaming variables,** **b. Adding a new variable,** **c. Dealing with missing data,** **d. Dealing with categorical data,** **e. Data reduction using subsetting**		
DOP: 17-09-2023	**DOS: 28-09-2023**	
CO Mapped: **CO2**	**PO Mapped:** **PO1, PO2, PO3, PO4, PO5, PO7, PSO1, PSO2**	**Signature:**

Practical No. 3

Aim: Write commands for Implementation of Data Preprocessing Techniques like:

a. Naming and renaming variables,
b. Adding a new variable,
c. Dealing with missing data,
d. Dealing with categorical data,
e. Data reduction using subsetting.

Theory:

Data preprocessing is an important stage in the data analysis pipeline that involves cleaning, organizing, and transforming raw data into an analysis and modeling-ready state. It is crucial in assuring the accuracy and effectiveness of data-driven processes. In this section, we'll go through several data preprocessing techniques in R, such as variable naming and renaming, variable addition, dealing with missing data, handling categorical data, and data reduction using subsetting.

1. **Naming and Renaming Variables:**
   a. **Naming Variables:** Data naming entails giving variables meaningful names that effectively reflect their purpose and substance. Descriptive variable names aid in understanding the context of the data.
   b. **Renaming Variables:** To offer more descriptive and interpretable names, you can rename variables in R using functions such as colnames() or names().
   c. **Example:**
      # Renaming variables in R
      colnames(dataframe) <- c("new_name1", "new_name2", ...)

2. **Adding a New Variable:**
   a. Adding a new variable to your dataset is a critical step in data preparation and feature engineering. This technique enables you to add new information or change current data in order to improve the quality and usefulness of your dataset for analysis or modeling.
   b. Example: # Creating a new variable based on existing ones
      dataframe$new_variable <- dataframe$column1 + dataframe$column2

3. **Dealing with Missing Data:**
   a. **Identification of Missing Data:** Start by identifying missing values in your dataset using functions like is.na() or complete.cases() in R.
   Example:
   ```
   # Identify missing values in a column
   missing_values <- is.na(dataframe$column_with_missing)
   ```

   b. **Handling Missing Data:**
      1) <u>Removal:</u> You can remove rows with missing values using complete.cases()
      Example:
      ```
      dataframe <- dataframe[complete.cases(dataframe), ]
      ```
      2) <u>Imputation:</u> Fill in missing values with appropriate substitutes. Common imputation methods include mean, median, mode imputation, or more advanced techniques like K-nearest neighbors (KNN) imputation or regression imputation.
      Example:
      ```
      # Impute missing values with the mean of the column
      dataframe$column_with_missing <- ifelse
      (is.na(dataframe$column_with_missing),
      mean(dataframe$column_with_missing, na.rm = TRUE),
      dataframe$column_with_missing)
      ```

4. **Dealing with Categorical Data:**
   1) **Categorical Data Encoding:** Categorical data needs to be converted into numerical format for most machine learning algorithms. Common methods include:
      a) One-Hot Encoding: Create binary columns for each category.
      Example:
      ```
      library(dummies)
      dataframe <- dummy.data.frame(dataframe, names =
      c("categorical_column"))
      ```

2) **Label Encoding:** Assign unique integers to each category.
   Example:
   ```
   dataframe$categorical_column <- as.factor(dataframe$categorical_column)
   dataframe$categorical_column <- as.integer(dataframe$categorical_column)
   ```

5. Data Reduction Using Subsetting in R:

1) **Data Subsetting:** Data reduction involves selecting a subset of relevant data for analysis. In R, you can use subsetting to filter rows and columns based on specific conditions.
   Example:
   ```
   # Selecting rows where a condition is met
   subset_data <- dataframe[dataframe$column > 5, ]
   # Selecting specific columns
   subset_data <- dataframe[, c("column1", "column2")]
   ```

2) **Advanced Subsetting Techniques:** You can combine multiple conditions, use logical operators (AND, OR), and even create complex filtering conditions.
   Example:
   ```
   subset_data <- dataframe[dataframe$column1 > 5 & dataframe$column2 ==
   "CategoryA", ]
   ```

Code:

1. **Naming and Renaming Variables**

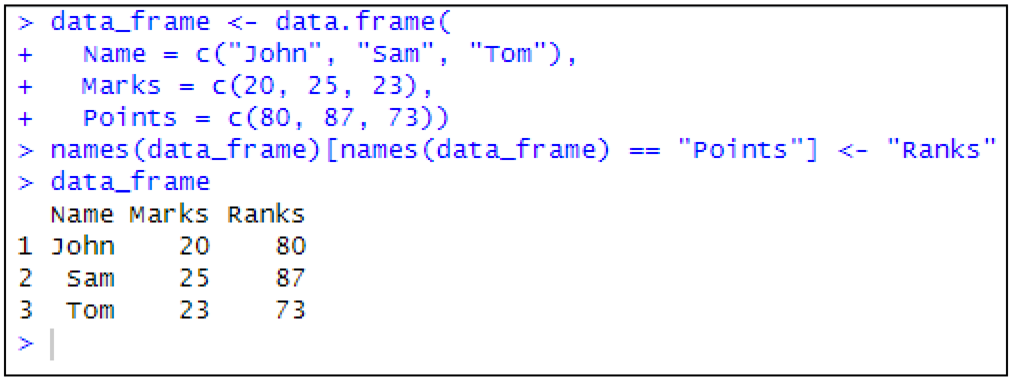
To Rename Variables:

```
# Create a sample data frame
data_frame <- data.frame(
  Name = c("John", "Sam", "Tom"),
  Marks = c(20, 25, 23),
  Points = c(80, 87, 73)
)
```

a. **Rename a Single Variable:**

```
names(data_frame)[names(data_frame) == "Points"] <- "Ranks"
data_frame
```

Output:

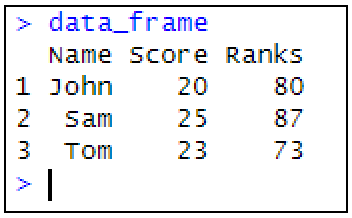


```
> data_frame <- data.frame(
+   Name = c("John", "Sam", "Tom"),
+   Marks = c(20, 25, 23),
+   Points = c(80, 87, 73))
> names(data_frame)[names(data_frame) == "Points"] <- "Ranks"
> data_frame
  Name Marks Ranks
1 John    20    80
2  Sam    25    87
3  Tom    23    73
> 
```

b. **Rename multiple variables:**

```
data_frame <- data_frame %>%
  rename(Score = Marks,
       rank = Points)
data_frame
```

Output:



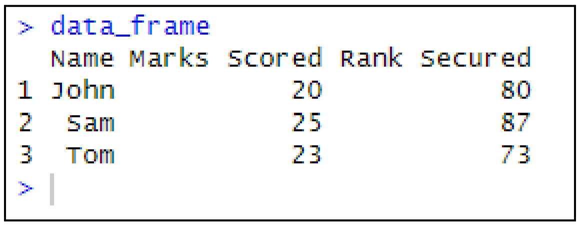
```
> data_frame
  Name Score Ranks
1 John    20    80
2  Sam    25    87
3  Tom    23    73
> 
```

c. **To Change Variable Names (without altering the data):**

names(data_frame) <- c("Name", "Marks Scored", "Rank Secured")

data_frame

Output:



```
> data_frame
  Name Marks Scored Rank Secured
1 John           20           80
2  Sam           25           87
3  Tom           23           73
> 
```

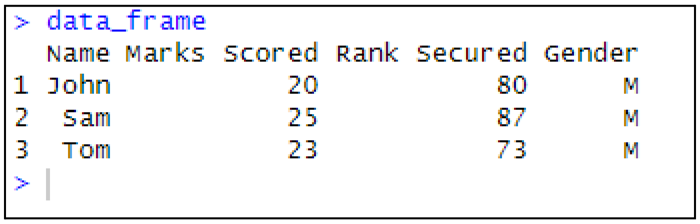
2. **Adding a New Variable**

a. **Using $ operator**

data_frame$Gender <- c("M","M","M")

data_frame

Output:



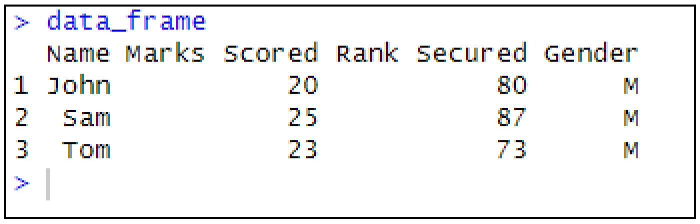
```
> data_frame
  Name Marks Scored Rank Secured Gender
1 John           20           80      M
2  Sam           25           87      M
3  Tom           23           73      M
> 
```

b. **Using [ ] notation:**

data_frame['Gender'] <- c("M","M","M")

data_frame

Output:



```
> data_frame
  Name Marks Scored Rank Secured Gender
1 John           20           80      M
2  Sam           25           87      M
3  Tom           23           73      M
> 
```

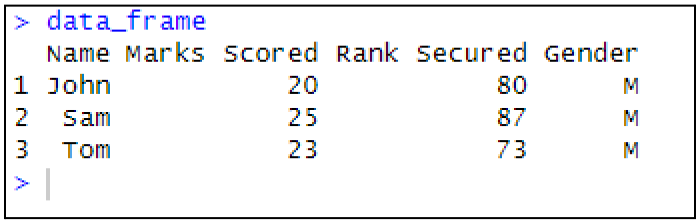
c. **Using cbind() Function:**

Gender <- c ("M","M","M")

data_frame <- cbind(data_frame, Gender)

data_frame

Output:



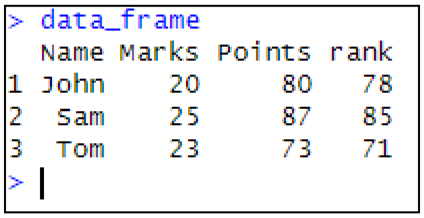
```
> data_frame
  Name Marks Scored Rank Secured Gender
1 John           20           80      M
2  Sam           25           87      M
3  Tom           23           73      M
> 
```

d. **Add a New Column from the Existing:**

data_frame$rank <- data_frame$Points-2

data_frame

Output:



```
> data_frame
  Name Marks Points rank
1 John    20     80   78
2  Sam    25     87   85
3  Tom    23     73   71
> 
```

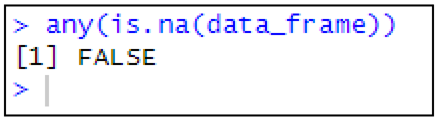
3. **Dealing with missing data**

a. **Checking for Missing Data:**

1) **Checking with not missing any data**

```
data_frame <- data.frame(
  Name = c("John", "Sam", "Tom"),
  Marks = c(20, 25, 23),
  Points = c(80, 87, 73)
)
any(is.na(data_frame))
```

Output:

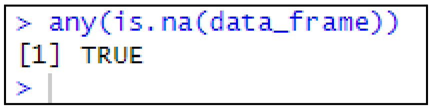


```
> any(is.na(data_frame))
[1] FALSE
> 
```

2) **Checking with missing data**

```
data_frame <- data.frame(
  Name = c("John", "Sam", "Tom"),
  Marks = c(20, 25, 23),
  Points = c(80, 87, NA)
)
any(is.na(data_frame))
```

Output:



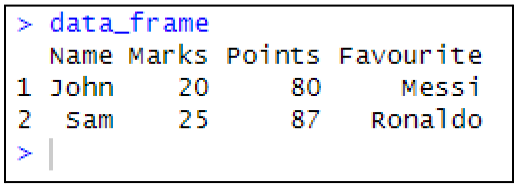
```
> any(is.na(data_frame))
[1] TRUE
> 
```

b. Handling Missing Data:

1) **Removing Row With missing Values**

```
data_frame <- data.frame(
  Name = c("John", "Sam", "Tom"),
  Marks = c(20, 25, 23),
  Points = c(80, 87, NA),
  Favourite = c("Messi", "Ronaldo", "Messi")
)
# Remove rows with missing values
data_frame <- na.omit(data_frame)
data_frame
```

Output:

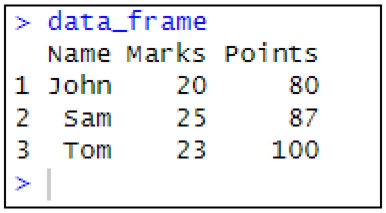


```
> data_frame
  Name Marks Points Favourite
1 John    20     80     Messi
2  Sam    25     87   Ronaldo
> 
```

2) **Fill missing values with a specific value**

```
data_frame <- data.frame(
  Name = c("John", "Sam", "Tom"),
  Marks = c(20, 25, 23),
  Points = c(80, 87, NA),
)
# Fill missing values with a specific value
data_frame$Points[is.na(data_frame$Points)] <- 100
data_frame
```

Output:



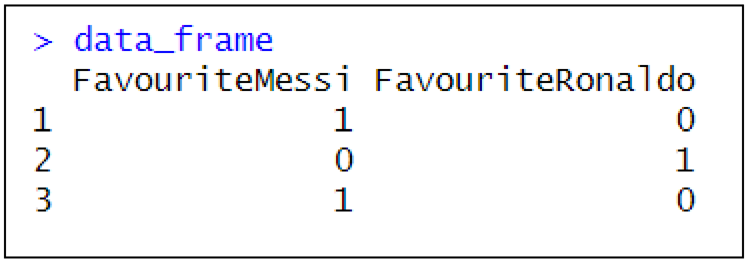
```
> data_frame
  Name Marks Points
1 John    20     80
2  Sam    25     87
3  Tom    23    100
> 
```

4. **Dealing with Categorical Data**

   a. **Converting Categorical to Dummy Variables (One-Hot Encoding):**

```
data_frame <- data.frame(
  Name = c("John", "Sam", "Tom"),
  Marks = c(20, 25, 23),
  Points = c(80, 87, NA),
  Favourite = c("Messi", "Ronaldo", "Messi")
)
# Perform one-hot encoding (dummy variable creation) for the categorical variable
dummyData <- dummyVars(~ Favourite, data = data_frame)
data_frame <- data.frame(predict(dummyData, newdata = data_frame))
data_frame
```

Output:

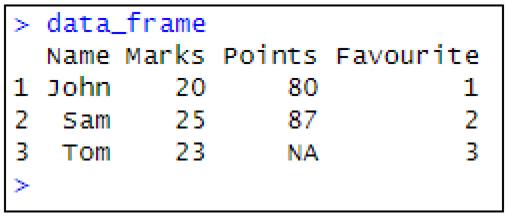


```
> data_frame
  FavouriteMessi FavouriteRonaldo
1              1                0
2              0                1
3              1                0
```

b. **Converting Categorical to Numeric Using Factorization:**

```
data_frame <- data.frame(
  Name = c("John", "Sam", "Tom"),
  Marks = c(20, 25, 23),
  Points = c(80, 87, NA),
  Favourite = c(1, 2, 3)
)
# Perform one-hot encoding (dummy variable creation) for the categorical variable
data_frame$Favourite <- as.numeric(factor(data_frame$Favourite))
data_frame
```

Output:



```
> data_frame
  Name Marks Points Favourite
1 John    20     80         1
2  Sam    25     87         2
3  Tom    23     NA         3
>
```

5. **Data Reduction Using Subsetting:**

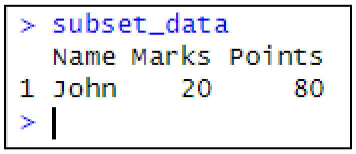
a. **Subsetting Rows Based on a Condition:**

```
data_frame <- data.frame(
  Name = c("John", "Sam", "Tom"),
  Marks = c(20, 25, 23),
  Points = c(80, 87, NA)
)
```

```
# Define the condition and desired value
variable_condition <- "Name"  # Change this to your desired condition
desired_value <- "John"      # Change this to your desired value

# Subset the data frame based on the condition
subset_data <- data_frame[data_frame[, variable_condition] == desired_value, ]
subset_data
```

Output:



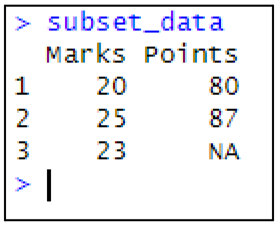
```
> subset_data
  Name Marks Points
1 John    20     80
> 
```

b. **Subsetting Columns:**

```
data_frame <- data.frame(
  Name = c("John", "Sam", "Tom"),
  Marks = c(20, 25, 23),
  Points = c(80, 87, NA)
)

# Subset data_frame to include only "Age" and "Score"
subset_data <- data_frame[, c("Marks", "Points")]
subset_data
```

Output:



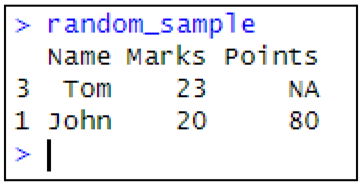
```
> subset_data
  Marks Points
1    20     80
2    25     87
3    23     NA
> 
```

c. Random Sampling:

```
data_frame <- data.frame(
  Name = c("John", "Sam", "Tom"),
  Marks = c(20, 25, 23),
  Points = c(80, 87, NA)
)

# Specify the desired sample size (e.g., 2 for a small sample)
sample_size <- 2
random_sample <- data_frame[sample(nrow(data_frame), sample_size), ]
random_sample
```

Output:



```
> random_sample
  Name Marks Points
3  Tom    23     NA
1 John    20     80
> 
```

Conclusion:

Data preprocessing is a multidimensional process that includes a variety of approaches and procedures for cleaning, enhancing, and preparing data for analysis or modeling. Handling variable names correctly, introducing new variables, addressing missing data, encoding categorical data, and decreasing data via subsetting are all key phases in the data preprocessing pipeline, ensuring the data is in an optimal shape for meaningful analysis or machine learning activities.