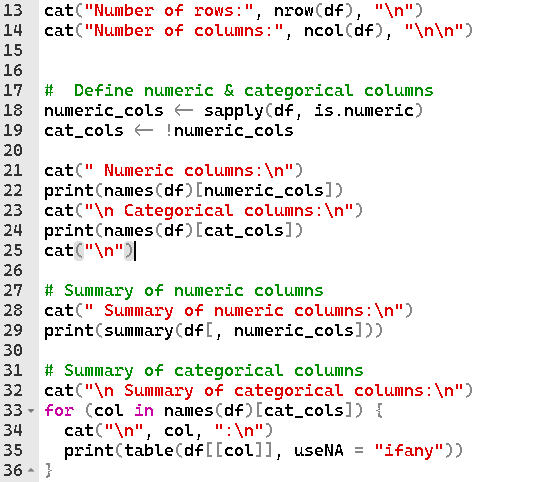
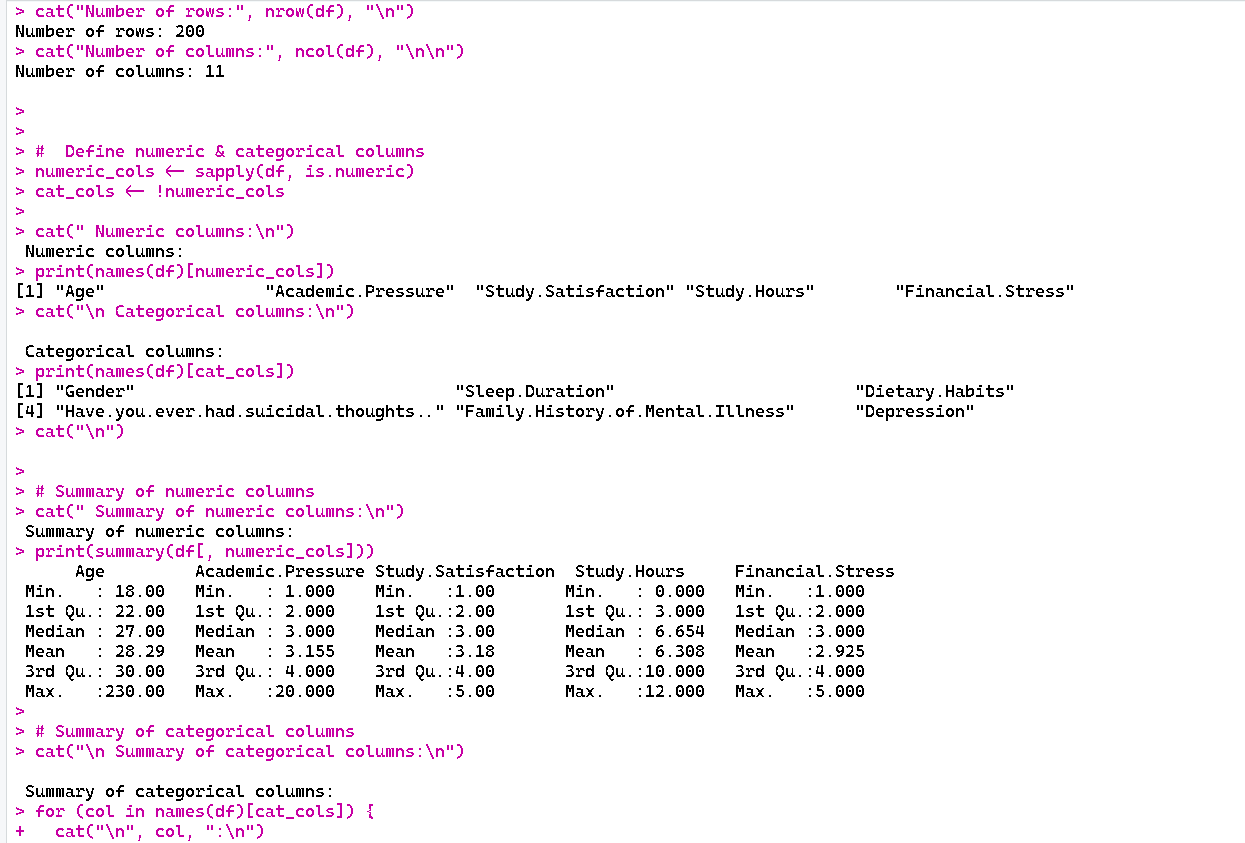
**Numeric And Categorical Column Define**

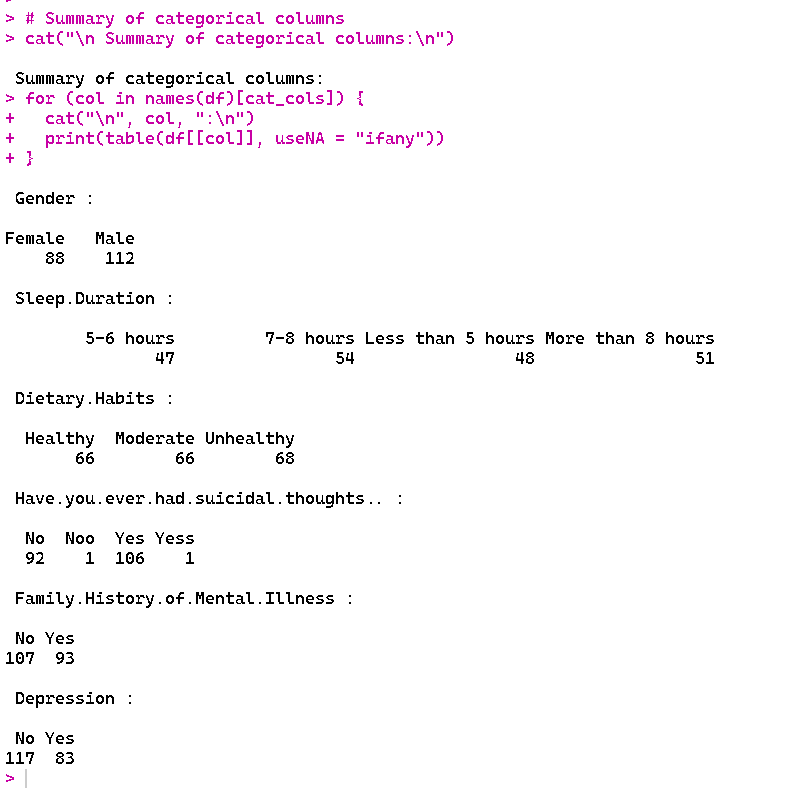
**Description:** Print the Numeric And Categorical Column and also summary of Numeric And Categorical Column.

**Code :**

****

**Output:**

****

****

**Code Description :** The given R code summarizes the structure and contents of the dataset df in order to do a first exploratory study. Using nrow(df) and ncol(df), it first outputs the total number of rows and columns. In this instance, it reveals that the dataset has 201 rows and 11 columns. The code then determines whether columns are category and numeric. It generates a logical vector numeric\_cols that indicates which columns are numeric using sapply(df, is.numeric). Categorical columns are identified by cat\_cols, the vector's complement. "Age", "Academic.Pressure", "Study.Satisfaction", "Study.Hours", and "Financial.Stress" are the numeric columns selected from the output. "Gender," "Sleep Duration," "Dietary Habits," "Have you ever had suicidal thoughts?" "Family History of Mental Illness," and "Depression" are the category columns.The code generates a summary for numeric columns using summary(df[, numeric\_cols]), which gives each column's lowest, first quartile, median, mean, third quartile, maximum, and NA counts. For instance, "Age" contains three missing values and ranges from a minimum of 18, a median of 26, a mean of 28.25, and a maximum of 230. In a similar vein, additional numerical columns like "Academic.Pressure" and "Study.Hours" display their distribution and aid in identifying anomalies or outliers (such as "Age" max = 230, which could be a mistake).When dealing with categorical columns, the code iterates over each one and uses table() to create a frequency table that includes the number of missing values (useNA = "ifany"). Distributions like these are displayed in the output:

"Gender": 110 females, 88 men, and 3 absent.

"Sleep.Duration": the majority of pupils sleep for seven to eight hours or longer.

"Dietary.Habits": a good mix of moderate, unhealthy, and healthy habits.

Some of the entries in "Have.you.ever.had.suicidal.thoughts.." are inconsistent, having "Noo" and "Yess" in between.

"Family.History.of.Mental.Illness" has 93 historical entries and 108 without.

115 "Yes" and 83 "No" responses for "Depression"; 3 are missing.

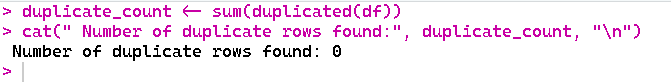
**Check for duplicate**

**Description:**Counts and displays the number of duplicate rows.

**Code :**

****

**Output:**

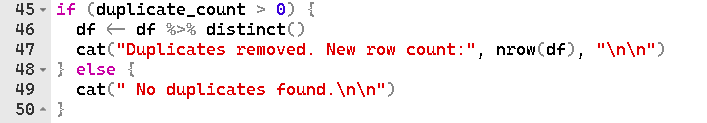
****

**Code Description:** This R function is intended to identify and tally duplicate rows in the df data frame. It makes use of the duplicated(df) function, which looks at every row and gives back a logical vector that indicates if a row is a copy of one that came before it. Duplicate rows are indicated by TRUE, whereas unique rows are indicated by FALSE. Since TRUE is considered as 1 and FALSE as 0 in arithmetic operations, the code determines the total number of duplicate rows by applying sum(duplicated(df)). The cat() function is then used to report the outcome, printing the duplicate count and a descriptive message. There was just one duplicate row in the output displayed. Finding duplicates is an essential step in data cleaning that helps preserve dataset accuracy and prevent biased modeling or analysis errors.

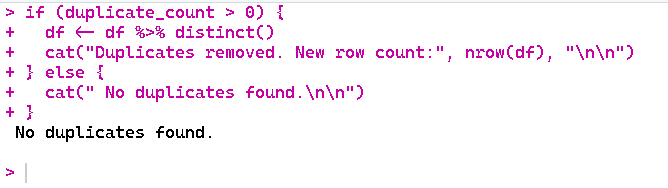
**Handle duplicate**

**Description:** remove duplicate row

**Code:**

****

**Output:**

****

**Code Description:**

The purpose of this R code snippet is to identify and eliminate duplicate rows from the data frame `df`. The variable `duplicate\_count`, which should hold the number of duplicate rows, is first checked to see if it is larger than zero. If duplicates are present, the code removes all duplicate rows using the `distinct()` function from the `dplyr` package, leaving only unique entries. Following the removal of duplicates, it uses `nrow(df)` to display the updated number of rows in the data frame and produces a message stating that duplicates have been eliminated. The `else` block runs and prints a message stating that no duplicates were identified if `duplicate\_count` is 0. Prior to additional analysis, this guarantees that the data is clear and devoid of duplicate entries. The result indicates that 200 rows are left.

**Missing values**

**Description:** Convert missing values to NA

**Code:**

****

**Output:**

****

**Code Description:** He supplied In order to manage and detect blank cells in a dataset, R code transforms them into NA, which stands for missing or null values in R. df[df == "" | df == " "], the first line \- NA searches the full data frame df using logical indexing to find any cells that are either entirely empty ("") or have a single space (" "). Upon identifying such cells, they are substituted with NA, therefore designating them as missing values. This enables R functions and packages to appropriately identify and manage them throughout the analytic process. The second line, cat(" Blank cells converted to NA (null values in R)"), gives users unambiguous feedback on the data cleaning stage by indicating that the conversion has been completed. By doing this, mistakes brought on by inconsistent or blank entries are prevented and the dataset is guaranteed to be normalized for further analysis.

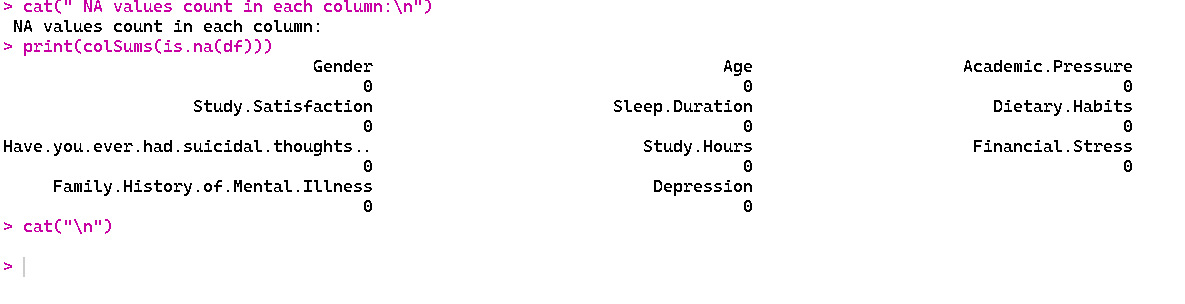
**Null Value**

**Description:** Count NA values

**Code:**

****

**Output:**

****

**Code Description:**

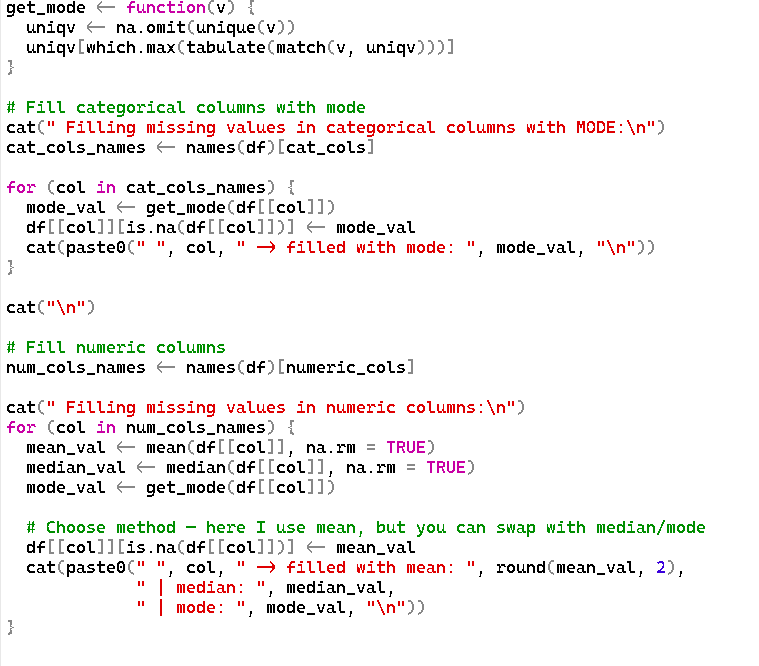
The purpose of the offered R code is to detect and manage blank or empty cells in order to clean a dataset. R's standard representation of missing or null values, `NA`, is used in the first line, `df[df == "" | df == " "] <- NA`, to search the full data frame `df` for cells that are either totally empty (`""`) or contain a single space (`" "`). This prevents blank strings from being mistakenly interpreted as genuine entries in later studies, ensuring that missing data is appropriately recognized. After that, a message stating that the blank cells have been changed to `NA` is printed using the `cat()` method. Using `colSums(is.na(df))`, the output further provides a summary of missing values in each column, displaying the precise number of `NA`s in each column. Preprocessing data and evaluating its quality depend on this stage.

**Handle missing values**

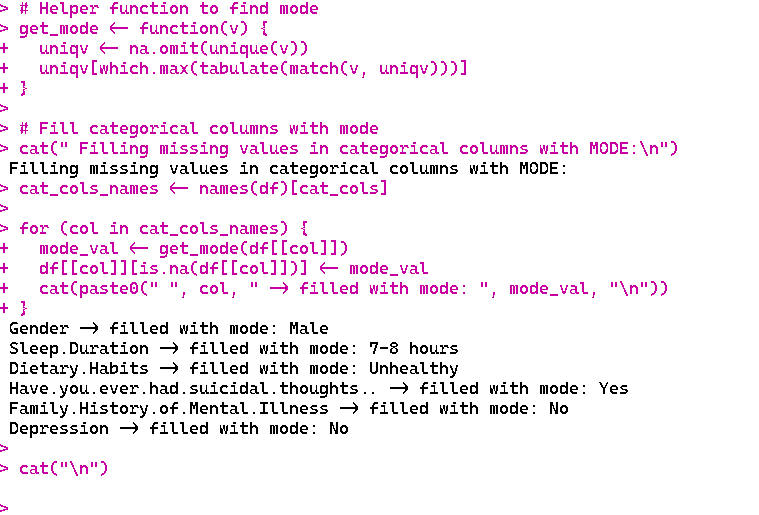
**Description:**

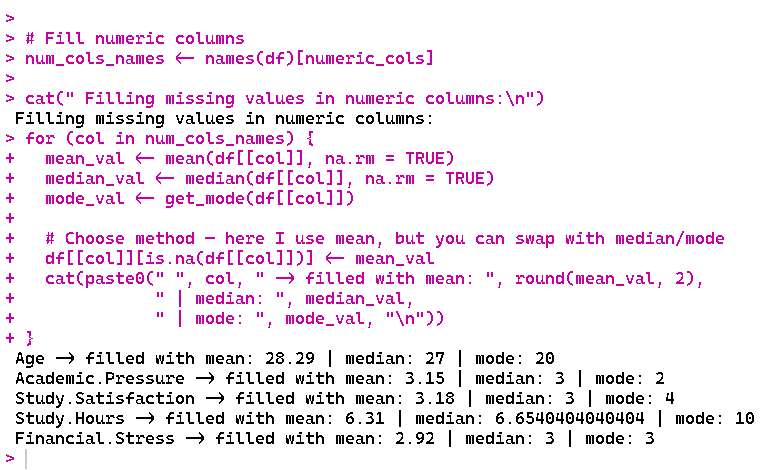
Iterating over all columns and reporting the changes, the code fills in missing values by substituting the mean (or optionally the median/mode) in numeric columns and the mode in categorical columns for `NA`s.

**Code:**

****

**Output:**

****

****

**Code Description:**

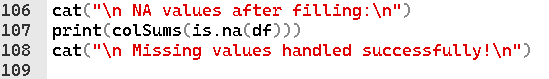
This R function fills up the numeric and category columns with the relevant summary statistics to deal with missing values in a dataset. To determine a vector's mode while disregarding `NA` values, a utility method called `get\_mode()` is developed. The most common category (mode) is used to replace missing values in categorical columns. This process iterates through all categorical column names and reports the replacements. The mean of the column is used to fill in missing values for numeric columns, and the median and mode are also computed and shown for reference. The code gives you the option of impute using the mean, median, or mode. To make sure there are no more `NA`s in the dataset, `cat()` is used throughout to produce explicit console output that indicates which columns were modified and the values utilized.

**Check Missing Value**

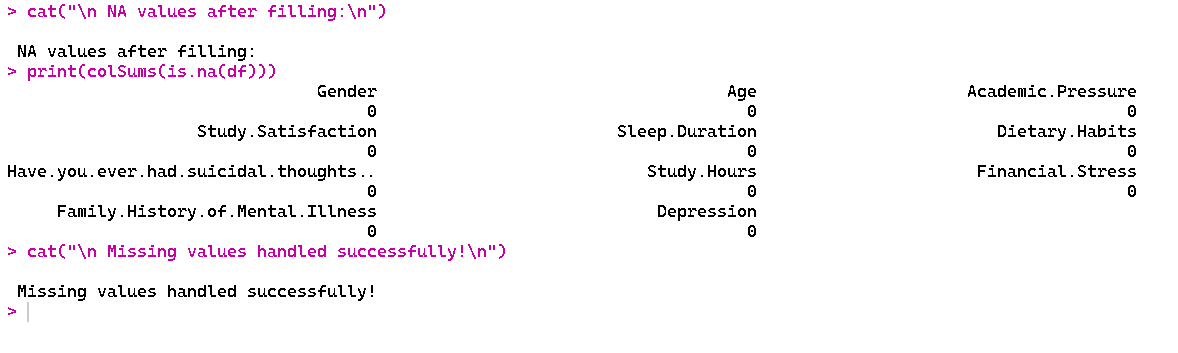
**Description:**

Final Null value check

**Code:**

****

**Output:**

****

**Code Description:**

The purpose of this R code snippet is to validate that handling missing values in a dataset was successful. The code initially prints the unambiguous message, "NA values after filling," to signal the beginning of the verification procedure following imputation or replacement of NA values in earlier phases. The total number of remaining NA values in each column of the data frame df is then determined using the colSums(is.na(df)) function, giving a column-by-column summary of any missing items.

This data, which indicates that all previously missing values in columns like Gender, Age, Academic.Pressure, and others have been properly filled, is produced in a legible manner using the print() method. The user is reassured that the dataset is now complete and prepared for additional analysis when the confirmation message, "Missing values handled successfully," is finally shown.