American International University-Bangladesh (AIUB)

Department of Computer Science

Faculty of Science & Technology (FST)

Introduction to Data Science

SEC: B

Supervised By

TOHEDUL ISLAM

A Introduction to Data Science Project Submitted
By

Semester: Summer_23_24		**Section: B**
SN	Student Name	Student ID
1	MAHMUD AL ALVI	22-46128-1
2	MD MAHADI HASAN NAYEEM	22-46137-1

Dataset Description:

This project focuses on the comprehensive preparation and analysis of a dataset comprising records of 105 passengers from the Titanic disaster. These records contain 10 features per passenger, including demographic and survival information. The primary objective of this project is to properly moderate the dataset by addressing issues such as missing, noisy, and invalid values. In addition, for advanced analysis, continuous or numeric attributes may be converted into categorical attributes or vice versa, filtering, mismatched value correction and normalization methods can be applied.

The project begins with a thorough investigation of visualization techniques to identify patterns and develop informed approaches for handling missing data. Methods for identifying and correcting invalid and incorrect data are also explored to enhance the dataset's quality and reliability. Using various visualization techniques, such as box plots and bar charts, essential data points can be analyzed and visualized more efficiently, leading to more intelligent analysis.

By establishing a solid foundation for dataset preparation and univariate data exploration, the project aims to facilitate sophisticated analyses that yield valuable insights into the factors influencing survival on the Titanic. These insights have the potential to inform safety measures and enhance our understanding of historical events.

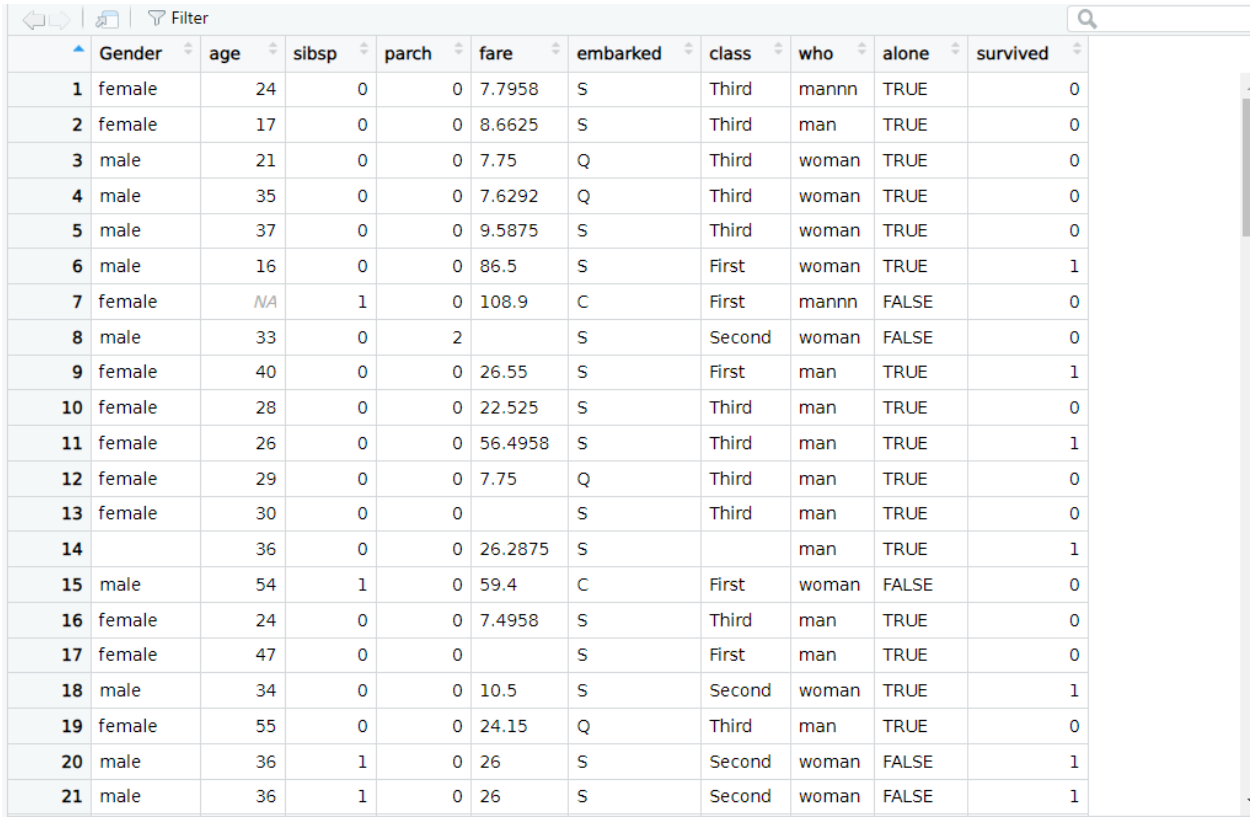
This report outlines the methodologies utilized, challenges encountered, and insights gained throughout the project, providing a comprehensive overview of the data preparation and analysis processes undertaken.

1. Importing Dataset into Rstudio

Code:

```
mydata<-read.csv("E:/8th Sem/MIDTERM/DATA SCIENCE/project/Midterm Project 2/Midterm Project/Mid_Dataset.csv",header=TRUE,sep=",")

mydata
```

Output:



Filter

	Gender	age	sibsp	parch	fare	embarked	class	who	alone	survived
1	female	24	0	0	7.7958	S	Third	mannn	TRUE	0
2	female	17	0	0	8.6625	S	Third	man	TRUE	0
3	male	21	0	0	7.75	Q	Third	woman	TRUE	0
4	male	35	0	0	7.6292	Q	Third	woman	TRUE	0
5	male	37	0	0	9.5875	S	Third	woman	TRUE	0
6	male	16	0	0	86.5	S	First	woman	TRUE	1
7	female	NA	1	0	108.9	C	First	mannn	FALSE	0
8	male	33	0	2		S	Second	woman	FALSE	0
9	female	40	0	0	26.55	S	First	man	TRUE	1
10	female	28	0	0	22.525	S	Third	man	TRUE	0
11	female	26	0	0	56.4958	S	Third	man	TRUE	1
12	female	29	0	0	7.75	Q	Third	man	TRUE	0
13	female	30	0	0		S	Third	man	TRUE	0
14		36	0	0	26.2875	S		man	TRUE	1
15	male	54	1	0	59.4	C	First	woman	FALSE	0
16	female	24	0	0	7.4958	S	Third	man	TRUE	0
17	female	47	0	0		S	First	man	TRUE	0
18	male	34	0	0	10.5	S	Second	woman	TRUE	1
19	female	55	0	0	24.15	Q	Third	man	TRUE	0
20	male	36	1	0	26	S	Second	woman	FALSE	1
21	male	36	1	0	26	S	Second	woman	FALSE	1

Description:

Here is the code of import the dataset as csv file. It is the output of the dataset which is imported in RStudio.

2. Handling Missing Values in Dataset

Code:

```
mydata$Gender <- ifelse(mydata$Gender == "", NA, mydata$Gender)

mydata$fare <- ifelse(mydata$fare == "", NA, mydata$fare)

mydata$embarked <- ifelse(mydata$embarked == "", NA, mydata$embarked)

mydata$class <- ifelse(mydata$class == "", NA, mydata$class)

mydata$who <- ifelse(mydata$who == "", NA, mydata$who)

mydata
```

Output:

```
 [ reached 'max' / getOption("max.print") -- omitted 5 rows ]
> mydata$Gender <- ifelse(mydata$Gender == "", NA, mydata$Gender)
> mydata$fare <- ifelse(mydata$fare == "", NA, mydata$fare)
> mydata$embarked <- ifelse(mydata$embarked == "", NA, mydata$embarked)
> mydata$class <- ifelse(mydata$class == "", NA, mydata$class)
> mydata$who <- ifelse(mydata$who == "", NA, mydata$who)
> mydata$Gender <- ifelse(mydata$Gender == "", NA, mydata$Gender)
> mydata
    Gender age sibsp parch     fare embarked  class   who alone survived
1   female  24     0     0   7.7958        S  Third mannn  TRUE        0
2   female  17     0     0   8.6625        S  Third   man  TRUE        0
3     male  21     0     0     7.75        Q  Third woman  TRUE        0
4     male  35     0     0   7.6292        Q  Third woman  TRUE        0
5     male  37     0     0   9.5875        S  Third woman  TRUE        0
6     male  16     0     0     86.5        S  First woman  TRUE        1
7   female  NA     1     0    108.9        C  First mannn FALSE        0
8     male  33     0     2     <NA>        S Second woman FALSE        0
9   female  40     0     0    26.55        S  First   man  TRUE        1
10  female  28     0     0   22.525        S  Third   man  TRUE        0
11  female  26     0     0  56.4958        S  Third   man  TRUE        1
12  female  29     0     0     7.75        Q  Third   man  TRUE        0
13  female  30     0     0     <NA>        S  Third   man  TRUE        0
14    <NA>  36     0     0  26.2875        S   <NA>   man  TRUE        1
15    male  54     1     0     59.4        C  First woman FALSE        0
16  female  24     0     0   7.4958        S  Third   man  TRUE        0
17  female  47     0     0     <NA>        S  First   man  TRUE        0
18    male  34     0     0     10.5        S Second woman  TRUE        1
19  female  55     0     0    24.15        Q  Third   man  TRUE        0
20    male  36     1     0       26        S Second woman FALSE        1
21    male  36     1     0       26        S Second woman FALSE        1
22  female  NA     0     0   7.8958        S  Third   man  TRUE        0
23    male  30     0     0     93.5        S  First woman  TRUE        1
24    <NA>  22     0     0   7.8958        S  Third   man  TRUE        0
25  female  40     0     0    7.225        C  Third   man  TRUE        0
```

Description:

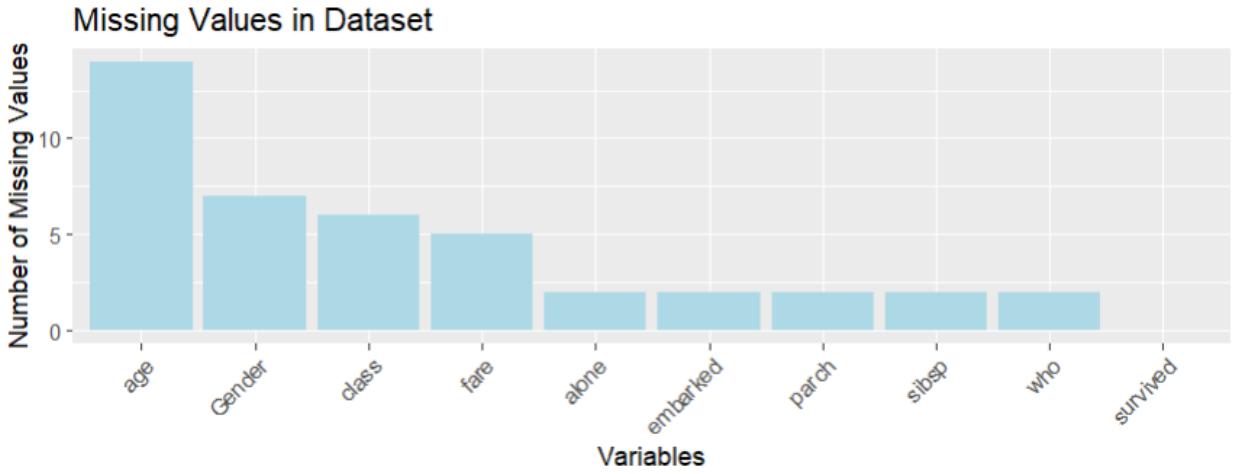
This code replaces empty string entries in the Gender, fare, embarked, class, and who columns of mydata with NA to mark them as missing values.

3. Graph of Missing Values of the Dataset

Code:

```
missing_values_plot <- ggplot(missing_values_df, aes(x = reorder(Variable, -Missing), y = Missing)) +
  geom_bar(stat = "identity", fill = "lightblue") +
  labs(title = "Missing Values in Dataset",
       x = "Variables",
       y = "Number of Missing Values") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
print(missing_values_plot)
```

Output:



Description:

This code creates a bar plot (missing_values_plot) using ggplot2 to visualize the number of missing values for each variable in the dataset .The plot displays variables reordered by the count of missing values on the x-axis and the corresponding number of missing values on the y-axis. Axes labels and plot title are specified to enhance clarity, with x-axis text rotated for better readability.

4. Summarizing Central Tendency and Spread

Code:

summary(mydata)

Output:

```
> summary(mydata)
    Gender               age              sibsp            parch            fare
 Length:105         Min.   :  2.00   Min.   :0.0000   Min.   :0.0000   Length:105
 Class :character   1st Qu.: 23.50   1st Qu.:0.0000   1st Qu.:0.0000   Class :character
 Mode  :character   Median : 33.00   Median :0.0000   Median :0.0000   Mode  :character
                    Mean   : 36.25   Mean   :0.3495   Mean   :0.3398
                    3rd Qu.: 40.50   3rd Qu.:1.0000   3rd Qu.:0.0000
                    Max.   :152.00   Max.   :4.0000   Max.   :4.0000
                    NA's   :14       NA's   :2        NA's   :2
   embarked            class               who              alone           survived
 Length:105         Length:105         Length:105         Mode :logical   Min.   :0.0000
 Class :character   Class :character   Class :character   FALSE:37        1st Qu.:0.0000
 Mode  :character   Mode  :character   Mode  :character   TRUE :66        Median :0.0000
                                                          NA's :2         Mean   :0.3619
                                                                          3rd Qu.:1.0000
                                                                          Max.   :1.0000
```

Description:

This code provides a summary of mydata, displaying key measures of central tendency (mean, median) and spread (min, max, quartiles) for each column.

5. Exploring Data Structure

Code:

str(mydata)

Output:

```
> #5.to know the structure of the data set
> str(mydata)
'data.frame':   105 obs. of  12 variables:
 $ Gender        : chr  "female" "female" "male" "male" ...
 $ age           : int  24 17 21 35 37 16 NA 33 40 28 ...
 $ sibsp         : int  0 0 0 0 0 0 1 0 0 0 ...
 $ parch         : int  0 0 0 0 0 0 0 2 0 0 ...
 $ fare          : chr  "7.7958" "8.6625" "7.75" "7.6292" ...
 $ embarked      : chr  "S" "S" "Q" "Q" ...
 $ class         : chr  "Third" "Third" "Third" "Third" ...
 $ who           : chr  "woman" "woman" "man" "man" ...
 $ alone         : logi  TRUE TRUE TRUE TRUE TRUE TRUE ...
 $ survived      : int  0 0 0 0 0 1 0 0 1 0 ...
 $ age_group     : Factor w/ 4 levels "1-18","19-30",..: 2 1 2 3 3 1 NA 3 3 2 ...
 $ survived_group: Factor w/ 2 levels "0","1": 1 1 1 1 1 2 1 1 2 1 ...
>
```

Description:

This code displays the structure of mydata, summarizing its dimensions and variable types for quick reference.

6. Detecting Null Values

Code:

mydata[mydata == ""] <- NA

is.na(mydata)

Output:

```
> mydata[mydata == ""] <- NA
> is.na(mydata)
      Gender   age sibsp parch  fare embarked class   who alone survived
 [1,]  FALSE FALSE FALSE FALSE FALSE    FALSE FALSE FALSE FALSE    FALSE
 [2,]  FALSE FALSE FALSE FALSE FALSE    FALSE FALSE FALSE FALSE    FALSE
 [3,]  FALSE FALSE FALSE FALSE FALSE    FALSE FALSE FALSE FALSE    FALSE
 [4,]  FALSE FALSE FALSE FALSE FALSE    FALSE FALSE FALSE FALSE    FALSE
 [5,]  FALSE FALSE FALSE FALSE FALSE    FALSE FALSE FALSE FALSE    FALSE
 [6,]  FALSE FALSE FALSE FALSE FALSE    FALSE FALSE FALSE FALSE    FALSE
 [7,]  FALSE  TRUE FALSE FALSE FALSE    FALSE FALSE FALSE FALSE    FALSE
 [8,]  FALSE FALSE FALSE FALSE  TRUE    FALSE FALSE FALSE FALSE    FALSE
 [9,]  FALSE FALSE FALSE FALSE FALSE    FALSE FALSE FALSE FALSE    FALSE
[10,]  FALSE FALSE FALSE FALSE FALSE    FALSE FALSE FALSE FALSE    FALSE
[11,]  FALSE FALSE FALSE FALSE FALSE    FALSE FALSE FALSE FALSE    FALSE
[12,]  FALSE FALSE FALSE FALSE FALSE    FALSE FALSE FALSE FALSE    FALSE
[13,]  FALSE FALSE FALSE FALSE  TRUE    FALSE FALSE FALSE FALSE    FALSE
[14,]   TRUE FALSE FALSE FALSE FALSE    FALSE  TRUE FALSE FALSE    FALSE
[15,]  FALSE FALSE FALSE FALSE FALSE    FALSE FALSE FALSE FALSE    FALSE
[16,]  FALSE FALSE FALSE FALSE FALSE    FALSE FALSE FALSE FALSE    FALSE
[17,]  FALSE FALSE FALSE FALSE  TRUE    FALSE FALSE FALSE FALSE    FALSE
[18,]  FALSE FALSE FALSE FALSE FALSE    FALSE FALSE FALSE FALSE    FALSE
[19,]  FALSE FALSE FALSE FALSE FALSE    FALSE FALSE FALSE FALSE    FALSE
[20,]  FALSE FALSE FALSE FALSE FALSE    FALSE FALSE FALSE FALSE    FALSE
[21,]  FALSE FALSE FALSE FALSE FALSE    FALSE FALSE FALSE FALSE    FALSE
```

Description:

This code replaces all empty string entries in mydata with NA to mark them as null values. It then checks for the presence of NA values in the dataset, outputting TRUE for null values and FALSE otherwise, identifying the location of missing data points.

7. Counting Null Values in Each Column

Code:

colSums(is.na(mydata))

Output:

```
> colSums(is.na(mydata))
  Gender      age    sibsp    parch     fare embarked    class      who    alone survived
       7       14        2        2        5        2        6        2        2        0
```

Description:

This code counts the number of NA (null) values in each column of mydata and returns the result. This helps to understand the extent of missing data in each column.

8. Identifying Rows of Null Values

Code:

which(is.na(mydata$Gender))

which(is.na(mydata$age))

which(is.na(mydata$sibsp))

which(is.na(mydata$parch))

which(is.na(mydata$fare))

which(is.na(mydata$embarked))

which(is.na(mydata$class))

which(is.na(mydata$who))

which(is.na(mydata$alone))

Output:

```
> which(is.na(mydata$Gender))
[1]  14  24  35  46  48 104 105
> which(is.na(mydata$age))
 [1]   7  22  40  61  73  78  83  89  94  98 101 103 104 105
> which(is.na(mydata$sibsp))
[1] 104 105
> which(is.na(mydata$parch))
[1] 104 105
> which(is.na(mydata$fare))
[1]   8  13  17 104 105
> which(is.na(mydata$embarked))
[1] 104 105
> which(is.na(mydata$class))
[1]  14  38  67 102 104 105
> which(is.na(mydata$who))
[1] 104 105
> which(is.na(mydata$alone))
[1] 104 105
>
```

Description:

This code identifies the specific rows in mydata that contain NA (null) values for each specified column (Gender, age, sibsp, parch, fare, embarked, class, who, and alone). It returns the indices of these rows, helping to locate where the missing data points are in the dataset.

9. Removing Rows with Missing Values (Discard Instance)

Code:

remove<-na.omit(mydata)

remove

Output:

```
> remove<-na.omit(mydata)
> remove
    Gender age sibsp parch     fare embarked  class   who alone survived
1   female  24     0     0   7.7958        S  Third mannn  TRUE        0
2   female  17     0     0   8.6625        S  Third   man  TRUE        0
3     male  21     0     0     7.75        Q  Third woman  TRUE        0
4     male  35     0     0   7.6292        Q  Third woman  TRUE        0
5     male  37     0     0   9.5875        S  Third woman  TRUE        0
6     male  16     0     0     86.5        S  First woman  TRUE        1
9   female  40     0     0    26.55        S  First   man  TRUE        1
10  female  28     0     0   22.525        S  Third   man  TRUE        0
11  female  26     0     0  56.4958        S  Third   man  TRUE        1
12  female  29     0     0     7.75        Q  Third   man  TRUE        0
15    male  54     1     0     59.4        C  First woman FALSE        0
```

Description:

This code removes all rows with NA values from mydata, resulting in a new dataset called remove that contains only complete cases.

10. Calculating Mean Age

Code:

mean_age <- as.integer(mean(mydata$age, na.rm = TRUE))

print(mean_age)

Output:

```
> mean_age <- as.integer(mean(mydata$age, na.rm = TRUE))
> print(mean_age)
[1] 36
>
```

Description:

This code calculates the mean age from the age column in mydata, ignoring NA values (na.rm = TRUE). The result, stored in mean_age, is then printed as an integer value.

11. Imputing Missing Values with Mean(Age)

Code:

mydata <- mydata %>%

mutate(age = ifelse(is.na(age), mean_age, age))

mydata

Output:

```
> mydata <- mydata %>%
+   mutate(age = ifelse(is.na(age), mean_age, age))
> mydata
    Gender age sibsp parch     fare embarked  class   who alone survived
1   female  24     0     0   7.7958        S  Third mannn  TRUE        0
2   female  17     0     0   8.6625        S  Third   man  TRUE        0
3     male  21     0     0     7.75        Q  Third woman  TRUE        0
4     male  35     0     0   7.6292        Q  Third woman  TRUE        0
5     male  37     0     0   9.5875        S  Third woman  TRUE        0
6     male  16     0     0     86.5        S  First woman  TRUE        1
7   female  36     1     0    108.9        C  First mannn FALSE        0
8     male  33     0     2     <NA>        S Second woman FALSE        0
9   female  40     0     0    26.55        S  First   man  TRUE        1
10  female  28     0     0   22.525        S  Third   man  TRUE        0
11  female  26     0     0 56.4958        S  Third   man  TRUE        1
12  female  29     0     0     7.75        Q  Third   man  TRUE        0
13  female  30     0     0     <NA>        S  Third   man  TRUE        0
14    <NA>  36     0     0 26.2875        S   <NA>   man  TRUE        1
15    male  54     1     0     59.4        C  First woman FALSE        0
16  female  24     0     0   7.4958        S  Third   man  TRUE        0
17  female  47     0     0     <NA>        S  First   man  TRUE        0
18    male  34     0     0     10.5        S Second woman  TRUE        1
```

Description:

This code replaces missing values in the age column of mydata with the previously calculated mean age (mean_age). It updates mydata with these imputed values, ensuring more complete data for further analysis.

12. Calculating Median Age

Code:

```
median_age <- as.integer(median(mydata$age, na.rm = TRUE))

print(median_age)
```

Output:

```
> median_age <- as.integer(median(mydata$age, na.rm = TRUE))
> print(median_age)
[1] 33
>
```

Description:

This code calculates the median age from the age column in mydata, ignoring NA values (na.rm = TRUE). The result, stored in median_age, is then printed as an integer value.

13. Imputing Missing Values with Median(Age)

Code:

```
mydata <- mydata %>%

  mutate(age = ifelse(is.na(age), median_age, age))

mydata
```

Output:

	Gender	age	sibsp	parch	fare	embarked	class	who	alone	survived
1	female	24	0	0	7.7958	S	Third	mannn	TRUE	0
2	female	17	0	0	8.6625	S	Third	man	TRUE	0
3	male	21	0	0	7.75	Q	Third	woman	TRUE	0
4	male	35	0	0	7.6292	Q	Third	woman	TRUE	0
5	male	37	0	0	9.5875	S	Third	woman	TRUE	0
6	male	16	0	0	86.5	S	First	woman	TRUE	1
7	female	33	1	0	108.9	C	First	mannn	FALSE	0
8	male	33	0	2	<NA>	S	Second	woman	FALSE	0
9	female	40	0	0	26.55	S	First	man	TRUE	1
10	female	28	0	0	22.525	S	Third	man	TRUE	0
11	female	26	0	0	56.4958	S	Third	man	TRUE	1
12	female	29	0	0	7.75	Q	Third	man	TRUE	0
13	female	30	0	0	<NA>	S	Third	man	TRUE	0
14	<NA>	36	0	0	26.2875	S	<NA>	man	TRUE	1
15	male	54	1	0	59.4	C	First	woman	FALSE	0
16	female	24	0	0	7.4958	S	Third	man	TRUE	0

Description:

This code replaces missing values in mydata$age with the median age (median_age), ensuring all data points are accounted for before analysis.

14. Imputing Missing Values with Mode(Sibsp)

Code:

```
column_name <- "sibsp"

mode_value <- as.numeric(names(which.max(table(mydata[[column_name]]))))

print(mode_value)
```

Output:

```
> print(mode_value)
[1] 0
```

Code:

```
column_name <- "sibsp"

replacement_value <- mode_value

mydata[[column_name]][is.na(mydata[[column_name]])] <- replacement_value

which(is.na(mydata$sibsp ))

mydata
```

Output:

```
> column_name <- "sibsp"
> replacement_value <- mode_value
> mydata[[column_name]][is.na(mydata[[column_name]])] <- replacement_value
> which(is.na(mydata$sibsp ))
integer(0)
> mydata
    Gender age sibsp parch     fare embarked  class   who alone survived
1   female  24     0     0   7.7958        S  Third mannn  TRUE        0
2   female  17     0     0   8.6625        S  Third   man  TRUE        0
3     male  21     0     0     7.75        Q  Third woman  TRUE        0
4     male  35     0     0   7.6292        Q  Third woman  TRUE        0
5     male  37     0     0   9.5875        S  Third woman  TRUE        0
6     male  16     0     0     86.5        S  First woman  TRUE        1
7   female  29     1     0    108.9        C  First mannn FALSE        0
8     male  33     0     2     <NA>        S Second woman FALSE        0
9   female  40     0     0    26.55        S  First   man  TRUE        1
10  female  28     0     0   22.525        S  Third   man  TRUE        0
11  female  26     0     0  56.4958        S  Third   man  TRUE        1
12  female  29     0     0     7.75        Q  Third   man  TRUE        0
13  female  30     0     0     <NA>        S  Third   man  TRUE        0
14    <NA>  36     0     0  26.2875        S   <NA>   man  TRUE        1
15    male  54     1     0     59.4        C  First woman FALSE        0
```

	Gender	age	sibsp	parch	fare	embarked	class	who	alone	survived
92	female	60	1	1	79.2	C	First	man	FALSE	1
93	female	22	0	0	8.05	S	Third	man	TRUE	0
94	female	*NA*	0	0	8.05	S	Third	man	TRUE	0
95	female	35	0	0	7.125	S	Third	man	TRUE	0
96	male	152	1	0	78.2667	C	First	woman	FALSE	1
97	female	47	0	0	7.25	S	Third	man	TRUE	0
98	male	*NA*	0	2	7.75	Q	Third	woman	FALSE	0
99	female	37	1	0	26	S	Second	man	FALSE	0
100	female	36	1	1	24.15	S	Third	man	FALSE	0
101	male	*NA*	0	0	33	S	Second	woman	TRUE	1
102	female	149	0	0	0	S	*NA*	man	TRUE	0
103	female	*NA*	0	0	7.225	C	Third	man	TRUE	0
104	*NA*	*NA*	0	*NA*	*NA*	*NA*	*NA*	*NA*	*NA*	0
105	*NA*	*NA*	0	*NA*	*NA*	*NA*	*NA*	*NA*	*NA*	0

Description:

This code calculates the mode value for the sibsp column in mydata. It then replaces any NA values in the sibsp column with this mode value, ensuring all missing data is filled with the most frequent value in that column.

15. Imputing Missing Values with Mode(Parch)

Code:

```
column_name <- "parch"

mode_value <- as.numeric(names(which.max(table(mydata[[column_name]]))))

print(mode_value)
```

Output:

```
> #8.2 calculate mode of parse
> column_name <- "parch"
> mode_value <- as.numeric(names(which.max(table(mydata[[column_name]]))))
> print(mode_value)
[1] 0
>
```

Code:

```
column_name <- "parch"

replacement_value <- mode_value

mydata[[column_name]][is.na(mydata[[column_name]])] <- replacement_value
```

which(is.na(mydata$parch))

mydata

Output:

```
> column_name <- "parch"
> replacement_value <- mode_value
> mydata[[column_name]][is.na(mydata[[column_name]])] <- replacement_value
> which(is.na(mydata$parch ))
integer(0)
> mydata
    Gender age sibsp parch     fare embarked  class   who alone survived
1   female  24     0     0   7.7958        S  Third mannn  TRUE        0
2   female  17     0     0   8.6625        S  Third   man  TRUE        0
3     male  21     0     0     7.75        Q  Third woman  TRUE        0
4     male  35     0     0   7.6292        Q  Third woman  TRUE        0
5     male  37     0     0   9.5875        S  Third woman  TRUE        0
6     male  16     0     0     86.5        S  First woman  TRUE        1
7   female  29     1     0    108.9        C  First mannn FALSE        0
8     male  33     0     2     <NA>        S Second woman FALSE        0
9   female  40     0     0    26.55        S  First   man  TRUE        1
10  female  28     0     0   22.525        S  Third   man  TRUE        0
11  female  26     0     0 56.4958        S  Third   man  TRUE        1
12  female  29     0     0     7.75        Q  Third   man  TRUE        0
13  female  30     0     0     <NA>        S  Third   man  TRUE        0
14    <NA>  36     0     0 26.2875        S   <NA>   man  TRUE        1
15    male  54     1     0     59.4        C  First woman FALSE        0
```

	Gender	age	sibsp	parch	fare	embarked	class	who	alone	survived
92	female	60	1	1	79.2	C	First	man	FALSE	1
93	female	22	0	0	8.05	S	Third	man	TRUE	0
94	female	*NA*	0	0	8.05	S	Third	man	TRUE	0
95	female	35	0	0	7.125	S	Third	man	TRUE	0
96	male	152	1	0	78.2667	C	First	woman	FALSE	1
97	female	47	0	0	7.25	S	Third	man	TRUE	0
98	male	*NA*	0	2	7.75	Q	Third	woman	FALSE	0
99	female	37	1	0	26	S	Second	man	FALSE	0
100	female	36	1	1	24.15	S	Third	man	FALSE	0
101	male	*NA*	0	0	33	S	Second	woman	TRUE	1
102	female	149	0	0	0	S	*NA*	man	TRUE	0
103	female	*NA*	0	0	7.225	C	Third	man	TRUE	0
104	*NA*	*NA*	0	0	*NA*	*NA*	*NA*	*NA*	*NA*	0
105	*NA*	*NA*	0	0	*NA*	*NA*	*NA*	*NA*	*NA*	0

Description:

This code calculates the mode value for the parch column in mydata. It then replaces any NA values in the parch column with this mode value, ensuring all missing data is filled with the most frequent value in that column.

16. Imputing Missing Values with Mode(Alone)

Code:

```
column_name <- "alone"

mode_value <- as.logical(names(which.max(table(mydata[[column_name]]))))

print(mode_value)
```

Output:

```
> column_name <- "alone"
> mode_value <- as.logical(names(which.max(table(mydata[[column_name]]))))
> print(mode_value)
[1] TRUE
>
```

Code:

```
column_name <- "alone"

replacement_value <- mode_value

mydata[[column_name]][is.na(mydata[[column_name]])] <- replacement_value

which(is.na(mydata$alone ))

mydata
```

Output:

```
> column_name <- "alone"
> replacement_value <- mode_value
> mydata[[column_name]][is.na(mydata[[column_name]])] <- replacement_value
> which(is.na(mydata$alone ))
integer(0)
> mydata
    Gender age sibsp parch     fare embarked  class   who alone survived
1   female  24     0     0   7.7958        S  Third mannn  TRUE        0
2   female  17     0     0   8.6625        S  Third   man  TRUE        0
3     male  21     0     0     7.75        Q  Third woman  TRUE        0
4     male  35     0     0   7.6292        Q  Third woman  TRUE        0
5     male  37     0     0   9.5875        S  Third woman  TRUE        0
6     male  16     0     0     86.5        S  First woman  TRUE        1
7   female  29     1     0    108.9        C  First mannn FALSE        0
8     male  33     0     2     <NA>        S Second woman FALSE        0
9   female  40     0     0    26.55        S  First   man  TRUE        1
10  female  28     0     0   22.525        S  Third   man  TRUE        0
11  female  26     0     0  56.4958        S  Third   man  TRUE        1
12  female  29     0     0     7.75        Q  Third   man  TRUE        0
13  female  30     0     0     <NA>        S  Third   man  TRUE        0
14    <NA>  36     0     0  26.2875        S   <NA>   man  TRUE        1
15    male  54     1     0     59.4        C  First woman FALSE        0
```

	Gender	age	sibsp	parch	fare	embarked	class	who	alone	survived
92	female	60	1	1	79.2	C	First	man	FALSE	1
93	female	22	0	0	8.05	S	Third	man	TRUE	0
94	female	*NA*	0	0	8.05	S	Third	man	TRUE	0
95	female	35	0	0	7.125	S	Third	man	TRUE	0
96	male	152	1	0	78.2667	C	First	woman	FALSE	1
97	female	47	0	0	7.25	S	Third	man	TRUE	0
98	male	*NA*	0	2	7.75	Q	Third	woman	FALSE	0
99	female	37	1	0	26	S	Second	man	FALSE	0
100	female	36	1	1	24.15	S	Third	man	FALSE	0
101	male	*NA*	0	0	33	S	Second	woman	TRUE	1
102	female	149	0	0	0	S	*NA*	man	TRUE	0
103	female	*NA*	0	0	7.225	C	Third	man	TRUE	0
104	*NA*	*NA*	0	0	*NA*	*NA*	*NA*	*NA*	TRUE	0
105	*NA*	*NA*	0	0	*NA*	*NA*	*NA*	*NA*	TRUE	0

Description:

This code calculates the mode value for the alone column in mydata. It then replaces any NA values in the alone column with this mode value, ensuring all missing data is filled with the most frequent value in that column.

17. Imputing Missing Values with Mode(Gender)

Code:

column_name <- "Gender"

mode_value <- as.character(names(which.max(table(mydata[[column_name]], useNA = "no"))))

print(mode_value)

Output:

```
> column_name <- "Gender"
> mode_value <- as.character(names(which.max(table(mydata[[column_name]], useNA = "no"))))
> print(mode_value)
[1] "female"
> 
```

Code:

column_name <- "Gender"

replacement_value <- mode_value

mydata[[column_name]][is.na(mydata[[column_name]])] <- replacement_value

which(is.na(mydata$Gender))

mydata

Output:

```
> column_name <- "Gender"
> replacement_value <- mode_value
> mydata[[column_name]][is.na(mydata[[column_name]])] <- replacement_value
> which(is.na(mydata$Gender ))
integer(0)
> mydata
    Gender age sibsp parch     fare embarked  class   who alone survived
1   female  24     0     0   7.7958        S  Third mannn  TRUE        0
2   female  17     0     0   8.6625        S  Third   man  TRUE        0
3     male  21     0     0     7.75        Q  Third woman  TRUE        0
4     male  35     0     0   7.6292        Q  Third woman  TRUE        0
5     male  37     0     0   9.5875        S  Third woman  TRUE        0
6     male  16     0     0     86.5        S  First woman  TRUE        1
7   female  NA     1     0    108.9        C  First mannn FALSE        0
8     male  33     0     2     <NA>        S Second woman FALSE        0
9   female  40     0     0    26.55        S  First   man  TRUE        1
10  female  28     0     0   22.525        S  Third   man  TRUE        0
11  female  26     0     0 56.4958        S  Third   man  TRUE        1
12  female  29     0     0     7.75        Q  Third   man  TRUE        0
13  female  30     0     0     <NA>        S  Third   man  TRUE        0
14  female  36     0     0 26.2875        S   <NA>   man  TRUE        1
15    male  54     1     0     59.4        C  First woman FALSE        0
16  female  24     0     0   7.4958        S  Third   man  TRUE        0
17  female  47     0     0     <NA>        S  First   man  TRUE        0
18    male  34     0     0     10.5        S Second woman  TRUE        1
19  female  55     0     0    24.15        Q  Third   man  TRUE        0
20    male  36     1     0       26        S Second woman FALSE        1
21    male  36     1     0       26        S Second woman FALSE        1
22  female  NA     0     0   7.8958        S  Third   man  TRUE        0
23    male  30     0     0     93.5        S  First woman  TRUE        1
24  female  22     0     0   7.8958        S  Third   man  TRUE        0
25  female  40     0     0    7.225        C  Third   man  TRUE        0
26    male  44     0     1 57.9792        C  First woman FALSE        1
```

Description:

This code calculates the mode value for the Gender column in mydata, excluding NA values. It then replaces any NA values in the Gender column with this mode value, ensuring all missing data is filled with the most frequent value in that column.

18.Imputing Missing Values with Mode(Class)

Code:

column_name <- "class"

mode_value <- as.character(names(which.max(table(mydata[[column_name]], useNA = "no"))))

print(mode_value)

Output:

```
> column_name <- "class"
> mode_value <- as.character(names(which.max(table(mydata[[column_name]], useNA = "no"))))
> print(mode_value)
[1] "Third"
>
```

Code:

column_name <- "class"

replacement_value <- mode_value

mydata[[column_name]][is.na(mydata[[column_name]])] <- replacement_value

which(is.na(mydata$class))

mydata

Output:

```
> column_name <- "class"
> replacement_value <- mode_value
> mydata[[column_name]][is.na(mydata[[column_name]])] <- replacement_value
> which(is.na(mydata$class ))
integer(0)
> mydata
    Gender age sibsp parch     fare embarked  class   who alone survived
1   female  24     0     0   7.7958        S  Third mannn  TRUE        0
2   female  17     0     0   8.6625        S  Third   man  TRUE        0
3     male  21     0     0     7.75        Q  Third woman  TRUE        0
4     male  35     0     0   7.6292        Q  Third woman  TRUE        0
5     male  37     0     0   9.5875        S  Third woman  TRUE        0
6     male  16     0     0     86.5        S  First woman  TRUE        1
7   female  29     1     0    108.9        C  First mannn FALSE        0
8     male  33     0     2     <NA>        S Second woman FALSE        0
9   female  40     0     0    26.55        S  First   man  TRUE        1
10  female  28     0     0   22.525        S  Third   man  TRUE        0
11  female  26     0     0 56.4958        S  Third   man  TRUE        1
12  female  29     0     0     7.75        Q  Third   man  TRUE        0
13  female  30     0     0     <NA>        S  Third   man  TRUE        0
14  female  36     0     0 26.2875        S  Third   man  TRUE        1
15    male  54     1     0     59.4        C  First woman FALSE        0
```

Description:

This code calculates the mode value for the class column in mydata, excluding NA values. It then replaces any NA values in the class column with this mode value, ensuring all missing data is filled with the most frequent value in that column.

19.Imputing Missing Values with Mode (Embarked):

Code:

column_name <- "embarked"

mode_value <- as.character(names(which.max(table(mydata[[column_name]], useNA = "no"))))

print(mode_value)

Output:

```
> column_name <- "embarked"
> mode_value <- as.character(names(which.max(table(mydata[[column_name]], useNA = "no"))))
> print(mode_value)
[1] "S"
>
```

Code:

column_name <- "embarked"

replacement_value <- mode_value

mydata[[column_name]][is.na(mydata[[column_name]])] <- replacement_value

which(is.na(mydata$embarked))

mydata

Output:

```
> column_name <- "embarked"
> replacement_value <- mode_value
> mydata[[column_name]][is.na(mydata[[column_name]])] <- replacement_value
> which(is.na(mydata$embarked ))
integer(0)
> mydata
    Gender age sibsp parch     fare embarked  class   who alone survived
1   female  24     0     0   7.7958        S  Third mannn  TRUE        0
2   female  17     0     0   8.6625        S  Third   man  TRUE        0
3     male  21     0     0     7.75        Q  Third woman  TRUE        0
4     male  35     0     0   7.6292        Q  Third woman  TRUE        0
5     male  37     0     0   9.5875        S  Third woman  TRUE        0
6     male  16     0     0     86.5        S  First woman  TRUE        1
7   female  29     1     0    108.9        C  First mannn FALSE        0
8     male  33     0     2     <NA>        S Second woman FALSE        0
9   female  40     0     0    26.55        S  First   man  TRUE        1
10  female  28     0     0   22.525        S  Third   man  TRUE        0
11  female  26     0     0 56.4958        S  Third   man  TRUE        1
12  female  29     0     0     7.75        Q  Third   man  TRUE        0
13  female  30     0     0     <NA>        S  Third   man  TRUE        0
14  female  36     0     0 26.2875        S  Third   man  TRUE        1
15    male  54     1     0     59.4        C  First woman FALSE        0
```

	Gender	age	sibsp	parch	fare	embarked	class	who	alone	survived
92	female	60	1	1	79.2	C	First	man	FALSE	1
93	female	22	0	0	8.05	S	Third	man	TRUE	0
94	female	*NA*	0	0	8.05	S	Third	man	TRUE	0
95	female	35	0	0	7.125	S	Third	man	TRUE	0
96	male	152	1	0	78.2667	C	First	woman	FALSE	1
97	female	47	0	0	7.25	S	Third	man	TRUE	0
98	male	*NA*	0	2	7.75	Q	Third	woman	FALSE	0
99	female	37	1	0	26	S	Second	man	FALSE	0
100	female	36	1	1	24.15	S	Third	man	FALSE	0
101	male	*NA*	0	0	33	S	Second	woman	TRUE	1
102	female	149	0	0	0	S	Third	man	TRUE	0
103	female	*NA*	0	0	7.225	C	Third	man	TRUE	0
104	female	*NA*	0	0	*NA*	S	Third	*NA*	TRUE	0
105	female	*NA*	0	0	*NA*	S	Third	*NA*	TRUE	0

Description:

This code calculates the mode value for the embarked column in mydata, excluding NA values. It then replaces any NA values in the embarked column with this mode value, ensuring all missing data is filled with the most frequent value in that column. The updated mydata is displayed after imputation

20.Imputing Missing Values with Mode(Who):

Code:

```
column_name <- "who"

mode_value <- as.character(names(which.max(table(mydata[[column_name]], useNA = "no"))))

print(mode_value)
```

Output:

```
> column_name <- "who"
> mode_value <- as.character(names(which.max(table(mydata[[column_name]], useNA = "no"))))
> print(mode_value)
[1] "man"
>
```

Code:

```
column_name <- "who"

replacement_value <- mode_value

mydata[[column_name]][is.na(mydata[[column_name]])] <- replacement_value

which(is.na(mydata$who ))

mydata
```

Output:

```
> column_name <- "who"
> replacement_value <- mode_value
> mydata[[column_name]][is.na(mydata[[column_name]])] <- replacement_value
> which(is.na(mydata$who ))
integer(0)
> mydata
    Gender age sibsp parch     fare embarked  class   who alone survived
1   female  24     0     0   7.7958        S  Third mannn  TRUE        0
2   female  17     0     0   8.6625        S  Third   man  TRUE        0
3     male  21     0     0     7.75        Q  Third woman  TRUE        0
4     male  35     0     0   7.6292        Q  Third woman  TRUE        0
5     male  37     0     0   9.5875        S  Third woman  TRUE        0
6     male  16     0     0     86.5        S  First woman  TRUE        1
7   female  29     1     0    108.9        C  First mannn FALSE        0
8     male  33     0     2     <NA>        S Second woman FALSE        0
9   female  40     0     0    26.55        S  First   man  TRUE        1
10  female  28     0     0   22.525        S  Third   man  TRUE        0
11  female  26     0     0  56.4958        S  Third   man  TRUE        1
12  female  29     0     0     7.75        Q  Third   man  TRUE        0
13  female  30     0     0     <NA>        S  Third   man  TRUE        0
14  female  36     0     0  26.2875        S  Third   man  TRUE        1
15    male  54     1     0     59.4        C  First woman FALSE        0
16  female  24     0     0   7.4958        S  Third   man  TRUE        0
17  female  47     0     0     <NA>        S  First   man  TRUE        0
18    male  34     0     0     10.5        S Second woman  TRUE        1
19  female  55     0     0    24.15        Q  Third   man  TRUE        0
20    male  36     1     0       26        S Second woman FALSE        1
```

	Gender	age	sibsp	parch	fare	embarked	class	who	alone	survived
92	female	60	1	1	79.2	C	First	man	FALSE	1
93	female	22	0	0	8.05	S	Third	man	TRUE	0
94	female	*NA*	0	0	8.05	S	Third	man	TRUE	0
95	female	35	0	0	7.125	S	Third	man	TRUE	0
96	male	152	1	0	78.2667	C	First	woman	FALSE	1
97	female	47	0	0	7.25	S	Third	man	TRUE	0
98	male	*NA*	0	2	7.75	Q	Third	woman	FALSE	0
99	female	37	1	0	26	S	Second	man	FALSE	0
100	female	36	1	1	24.15	S	Third	man	FALSE	0
101	male	*NA*	0	0	33	S	Second	woman	TRUE	1
102	female	149	0	0	0	S	Third	man	TRUE	0
103	female	*NA*	0	0	7.225	C	Third	man	TRUE	0
104	female	*NA*	0	0	*NA*	S	Third	man	TRUE	0
105	female	*NA*	0	0	*NA*	S	Third	man	TRUE	0

Description:

This code calculates the mode value for the who column in mydata, excluding NA values. It then replaces any NA values in the who column with this mode value, ensuring all missing data is filled with the most frequent value in that column.

21. Outlier Detection and Removal (Age):

Code:

```
Q1 <- quantile(mydata$age[mydata$age >= 40 & mydata$age <= 95], 0.25)

Q3 <- quantile(mydata$age[mydata$age >= 40 & mydata$age <= 95], 0.75)

IQR_value <- Q3 - Q1

threshold <- 1.5

outlier_condition <- (mydata$age < (Q1 - threshold * IQR_value)) | (mydata$age > (Q3 + threshold * IQR_value))

mydata <- mydata[!outlier_condition, ]

mydata
```

Output:

84	female	32	0	0	7.925	S	Third	man	TRUE	0
85	male	25	1	1	30	S	Second	woman	FALSE	0
87	female	54	0	0	26	S	Second	man	TRUE	0
88	female	36	0	0	40.125	C	First	man	TRUE	0
89	female	36	0	0	8.7125	C	Third	man	TRUE	0
91	female	47	0	0	15	S	Second	man	TRUE	0
92	female	60	1	1	79.2	C	First	man	FALSE	1
93	female	22	0	0	8.05	S	Third	man	TRUE	0
94	female	36	0	0	8.05	S	Third	man	TRUE	0
95	female	35	0	0	7.125	S	Third	man	TRUE	0
97	female	47	0	0	7.25	S	Third	man	TRUE	0
98	male	36	0	2	7.75	Q	Third	woman	FALSE	0
99	female	37	1	0	26	S	Second	man	FALSE	0
100	female	36	1	1	24.15	S	Third	man	FALSE	0
101	male	36	0	0	33	S	Second	woman	TRUE	1
103	female	36	0	0	7.225	C	Third	man	TRUE	0
104	<NA>	26	NA	NA	<NA>	<NA>	<NA>	<NA>	NA	0

Description:

This code identifies and removes outliers from the `age` variable in `mydata` for ages between 40 and 95 using the Interquartile Range (IQR) method. It calculates the IQR, defines an outlier condition based on a threshold, and then filters out the rows where the age values are considered outliers.

22.Invalid Value Detection (Fare)

Code:

```
mydata <- mydata %>%
  mutate(fare = as.numeric(as.character(fare))) %>%
  filter(!is.na(fare))
mydata
```

Output:

61	male	36	1	0	17.4000	S	Third	woman	FALSE	1
62	female	34	0	0	7.7500	Q	Third	man	TRUE	0
63	female	40	0	0	7.8958	S	Third	man	TRUE	0
64	female	28	0	0	13.5000	S	<NA>	man	TRUE	0
65	female	30	0	0	8.0500	S	Third	man	TRUE	0
66	male	40	0	0	8.0500	S	Third	woman	TRUE	0
67	female	19	0	0	7.8958	S	Third	man	TRUE	0
68	male	29	0	4	21.0750	S	Third	woman	FALSE	0
69	female	NA	0	0	7.2292	C	Third	man	TRUE	0
70	female	32	0	0	7.8542	S	Third	man	TRUE	1
71	female	62	0	0	10.5000	S	Second	man	TRUE	1
72	male	53	2	0	51.4792	S	First	woman	FALSE	1
73	female	36	0	0	26.3875	S	First	man	TRUE	1
74	female	16	0	0	8.0500	S	Third	man	TRUE	0
75	female	19	0	0	14.5000	S	Third	man	TRUE	0
76	male	34	0	0	13.0000	S	Second	woman	TRUE	1
77	male	39	1	0	55.9000	S	First	woman	FALSE	1
78	male	NA	1	0	14.4583	C	Third	woman	FALSE	0
79	female	32	0	0	7.9250	S	Third	man	TRUE	0
80	male	25	1	1	30.0000	S	Second	woman	FALSE	0
81	male	139	1	1	110.8833	C	First	woman	FALSE	1

Description:

This code converts the fare column in mydata to numeric format, ensuring any invalid values are converted to NA. It then filters out rows where fare is NA, effectively removing invalid entries from the dataset and retaining only numeric fare values for further analysis.

23. Invalid Value Detection(Who):

Code:

```
mydata <- mydata %>%
  mutate(row_number = row_number()) %>%
  filter(who %in% c("man", "woman", "child"))
mydata
```

Output:

```
> mydata <- mydata %>%
+   mutate(row_number = row_number()) %>%
+   filter(who %in% c("man", "woman", "child"))
> mydata
   Gender age sibsp parch     fare embarked  class   who alone survived row_number
1  female  17     0     0   8.6625        S  Third   man  TRUE        0          2
2    male  21     0     0   7.7500        Q  Third woman  TRUE        0          3
3    male  35     0     0   7.6292        Q  Third woman  TRUE        0          4
4    male  37     0     0   9.5875        S  Third woman  TRUE        0          5
5    male  16     0     0  86.5000        S  First woman  TRUE        1          6
6  female  40     0     0  26.5500        S  First   man  TRUE        1          8
7  female  28     0     0  22.5250        S  Third   man  TRUE        0          9
8  female  26     0     0  56.4958        S  Third   man  TRUE        1         10
9  female  29     0     0   7.7500        Q  Third   man  TRUE        0         11
10   <NA>  36     0     0  26.2875        S   <NA>   man  TRUE        1         12
11   male  54     1     0  59.4000        C  First woman FALSE        0         13
12 female  24     0     0   7.4958        S  Third   man  TRUE        0         14
13   male  34     0     0  10.5000        S Second woman  TRUE        1         15
14 female  55     0     0  24.1500        Q  Third   man  TRUE        0         16
15   male  36     1     0  26.0000        S Second woman FALSE        1         17
16   male  36     1     0  26.0000        S Second woman FALSE        1         18
17 female  NA     0     0   7.8958        S  Third   man  TRUE        0         19
18   male  30     0     0  93.5000        S  First woman  TRUE        1         20
19   <NA>  22     0     0   7.8958        S  Third   man  TRUE        0         21
20 female  40     0     0   7.2250        C  Third   man  TRUE        0         22
21   male  44     0     1  57.9792        C  First woman FALSE        1         23
22 female  28     0     0   7.2292        C  Third   man  TRUE        0         24
23 female  41     0     0   7.7500        Q  Third   man  TRUE        0         25
24 female  41     0     0   7.7500        Q  Third   man  TRUE        0         26
25 female  41     0     0   7.7500        Q  Third   man  TRUE        0         27
```

Description:

This code identifies and removes rows from mydata where the who column contains invalid values. It retains only those rows where who is either "man", "woman", or "child", ensuring the dataset is cleansed of any rows with incorrect 'who' values.

24.Converting Categorical Variables to Numeric Factors:

Code:

```
mydata <- mydata %>%
  mutate(
    Gender = factor(Gender, levels = c("male", "female"), labels = c(1, 2)),
    embarked   = factor(embarked, levels = c("C","Q","S"), labels = c(1,2,3)),
    class    = factor(class , levels = c("First", "Second", "Third"), labels = c(1, 2, 3)),
    who = factor(who, levels = c("man", "woman","child"), labels = c(1, 2,3)),
    alone = factor(alone, levels = c("TRUE", "FALSE"), labels = c(1, 2))
  )
mydata
```

Output:

```
> mydata <- mydata %>%
+   mutate(
+     Gender = factor(Gender, levels = c("male", "female"), labels = c(1, 2)),
+     embarked   = factor(embarked, levels = c("C","Q","S"), labels = c(1,2,3)),
+     class    = factor(class , levels = c("First", "Second", "Third"), labels = c(1, 2, 3)),
+     who = factor(who, levels = c("man", "woman","child"), labels = c(1, 2,3)),
+     alone = factor(alone, levels = c("TRUE", "FALSE"), labels = c(1, 2))
+   )
> mydata
    Gender age sibsp parch     fare embarked class  who alone survived
1        2  24     0     0   7.7958        3     3 <NA>     1        0
2        2  17     0     0   8.6625        3     3    1     1        0
3        1  21     0     0     7.75        2     3    2     1        0
4        1  35     0     0   7.6292        2     3    2     1        0
5        1  37     0     0   9.5875        3     3    2     1        0
6        1  16     0     0     86.5        3     1    2     1        1
7        2  NA     1     0    108.9        1     1 <NA>     2        0
8        1  33     0     2                 3     2    2     2        0
9        2  40     0     0    26.55        3     1    1     1        1
10       2  28     0     0   22.525        3     3    1     1        0
11       2  26     0     0 56.4958        3     3    1     1        1
12       2  29     0     0     7.75        2     3    1     1        0
13       2  30     0     0                 3     3    1     1        0
14    <NA>  36     0     0 26.2875        3  <NA>    1     1        1
15       1  54     1     0     59.4        1     1    2     2        0
```

Description:

This code converts categorical variables (Gender, embarked, class, who, alone) in mydata into numeric factors for further analysis. Each categorical variable is transformed using the factor() function with specified levels and labels to map categorical values to corresponding numeric codes.

25.Converting Numerical Variables to Categorical Factors:

Code:

```
mydata <- mydata %>%
  mutate(
      survived = factor(survived, levels = c(0, 1), labels = c("Dead", "Alive")),


   )
  mydata
```

Output:

```
> mydata <- mydata %>%
+   mutate(
+     survived = factor(survived, levels = c(0, 1), labels = c("Dead", "Alive")),
+
+   )
> mydata
    Gender age sibsp parch     fare embarked  class   who alone survived
1   female  24     0     0   7.7958        S  Third mannn  TRUE     Dead
2   female  17     0     0   8.6625        S  Third   man  TRUE     Dead
3     male  21     0     0     7.75        Q  Third woman  TRUE     Dead
4     male  35     0     0   7.6292        Q  Third woman  TRUE     Dead
5     male  37     0     0   9.5875        S  Third woman  TRUE     Dead
6     male  16     0     0     86.5        S  First woman  TRUE    Alive
7   female  NA     1     0    108.9        C  First mannn FALSE     Dead
8     male  33     0     2                  S Second woman FALSE     Dead
```

Description:

This code converts the survived column in mydata from a numeric representation (0 for dead, 1 for alive) to a factor with descriptive labels ("Dead" and "Alive"). This transformation helps in interpreting and analyzing survival data more intuitively.

26.Categorizing Age into Sets:

Code:

```
mydata <- mydata %>%
    mutate(age_category = case_when(
      age < 18 ~ "child",
      age >= 18 & age <= 30 ~ "young",
      age > 30 & age <= 50 ~ "middle_aged",
      age > 50 ~ "old"
```

))

mydata

Output:

```
> mydata <- mydata %>%
+   mutate(age_category = case_when(
+     age < 18 ~ "child",
+     age >= 18 & age <= 30 ~ "young",
+     age > 30 & age <= 50 ~ "middle_aged",
+     age > 50 ~ "old"
+   ))
> mydata
   Gender age sibsp parch     fare embarked  class   who alone survived age_category
1  female  24     0     0   7.7958        S  Third mannn  TRUE        0       young
2  female  17     0     0   8.6625        S  Third   man  TRUE        0       child
3    male  21     0     0     7.75        Q  Third woman  TRUE        0       young
4    male  35     0     0   7.6292        Q  Third woman  TRUE        0 middle_aged
5    male  37     0     0   9.5875        S  Third woman  TRUE        0 middle_aged
6    male  16     0     0     86.5        S  First woman  TRUE        1       child
7  female  NA     1     0    108.9        C  First mannn FALSE        0        <NA>
8    male  33     0     2     <NA>        S Second woman FALSE        0 middle_aged
9  female  40     0     0    26.55        S  First   man  TRUE        1 middle_aged
10 female  28     0     0   22.525        S  Third   man  TRUE        0       young
11 female  26     0     0 56.4958        S  Third   man  TRUE        1       young
12 female  29     0     0     7.75        Q  Third   man  TRUE        0       young
13 female  30     0     0     <NA>        S  Third   man  TRUE        0       young
14   <NA>  36     0     0 26.2875        S   <NA>   man  TRUE        1 middle_aged
15   male  54     1     0     59.4        C  First woman FALSE        0         old
```

Description:

This code categorizes the age attribute in mydata into different age groups (age_category). Age values are classified as "child" for ages under 18, "young" for ages 18 to 30, "middle_aged" for ages 31 to 50, and "old" for ages over 50. The resulting dataset mydata includes the new age_category column for further analysis based on age groups.

27.Normalizing the Age Attribute:

Code:

column_name <- "age"

column <- mydata[[column_name]]

min_value <- min(column)

max_value <- max(column)

normalized_column <- (column - min_value) / (max_value - min_value)

mydata[[column_name]] <- normalized_column

mydata$age

mydata

Output:

```
> mydata
    Gender        age sibsp parch     fare embarked  class   who alone survived
1   female 0.14666667     0     0   7.7958        S  Third mannn  TRUE        0
2   female 0.10000000     0     0   8.6625        S  Third   man  TRUE        0
3     male 0.12666667     0     0     7.75        Q  Third woman  TRUE        0
4     male 0.22000000     0     0   7.6292        Q  Third woman  TRUE        0
5     male 0.23333333     0     0   9.5875        S  Third woman  TRUE        0
6     male 0.09333333     0     0     86.5        S  First woman  TRUE        1
7   female 0.22666667     1     0    108.9        C  First mannn FALSE        0
```

Description:

This code normalizes the age attribute in mydata to a range between 0 and 1. It calculates the minimum and maximum values of age, then performs min-max normalization on the column. The normalized values replace the original age column in mydata, ensuring that age values are scaled uniformly for analysis or modeling purposes.

42.Resolving 'who' Attribute Mismatches:

Code:

```
mydata <- mydata %>%
  mutate(who = case_when(
    Gender == "male" & who != "child" ~ "man",
    Gender == "female" & who != "child" ~ "woman",
    TRUE ~ who
  ))
```

Mydata

Output:

```
+   ))
> mydata
    Gender age sibsp parch     fare embarked  class   who alone survived
1   female  24     0     0   7.7958        S  Third woman  TRUE        0
2   female  17     0     0   8.6625        S  Third woman  TRUE        0
3     male  21     0     0     7.75        Q  Third   man  TRUE        0
4     male  35     0     0   7.6292        Q  Third   man  TRUE        0
5     male  37     0     0   9.5875        S  Third   man  TRUE        0
6     male  16     0     0     86.5        S  First   man  TRUE        1
7   female  NA     1     0    108.9        C  First woman FALSE        0
8     male  33     0     2                  S Second   man FALSE        0
9   female  40     0     0    26.55        S  First woman  TRUE        1
10  female  28     0     0   22.525        S  Third woman  TRUE        0
11  female  26     0     0  56.4958        S  Third woman  TRUE        1
12  female  29     0     0     7.75        Q  Third woman  TRUE        0
13  female  30     0     0                  S  Third woman  TRUE        0
14          36     0     0  26.2875        S          man  TRUE        1
15    male  54     1     0     59.4        C  First   man FALSE        0
```

Description:

- Updates who to "man" if Gender is "male" and who is not "child".
- Updates who to "woman" if Gender is "female" and who is not "child".
- Leaves who unchanged (TRUE ~ who) if no conditions are met.

29.Finding And Removing Duplicate Value(Age)

Code:

```
duplicates_age<-distinct(mydata,age,.keep_all = TRUE)

duplicates_age
```

Output:

```
15   male  54     1     0     59.4        C  First woman FALSE        0
16 female  47     0     0     <NA>        S  First   man  TRUE        0
17   male  34     0     0     10.5        S Second woman  TRUE        1
18 female  55     0     0    24.15        Q  Third   man  TRUE        0
19   <NA>  22     0     0   7.8958        S  Third   man  TRUE        0
20   male  44     0     1  57.9792        C  First woman FALSE        1
21 female  41     0     0     7.75        Q  Third   man  TRUE        0
22   male  50     0     0     10.5        S Second woman  TRUE        1
23 female  45     0     0 221.7792        S  First   man  TRUE        0
24 female  48     0     0    7.925        S  Third   man  TRUE        0
25 female  23     2     1     11.5        S Second   man FALSE        0
26   <NA>   2     1     1       26        S Second child FALSE        1
27 female  10     0     0   7.2292        C  Third   man  TRUE        0
28   male  20     0     2  22.3583        C   <NA> woman FALSE        1
29 female  32     0     0     14.5        S  Third   man  TRUE        0
30   <NA>   9     4     2   31.275        S  Third child FALSE        0
31   male  11     4     2   31.275        S  Third child FALSE        0
32 female  64     0     0       26        S  First   man  TRUE        0
33   male  19     1     0       26        S Second woman FALSE        1
34 female   8     1     1    36.75        S Second child FALSE        1
35 female  27     0     0       26        S Second   man  TRUE        0
36 female  25     0     0   7.8292        Q  Third   man  TRUE        0
37 female  62     0     0    26.55        S  First   man  TRUE        0
38   male  39     1     1    79.65        S  First woman FALSE        1
39   male  53     2     0  51.4792        S  First woman FALSE        1
40   male 139     1     1 110.8833        C  First woman FALSE        1
41   male  18     0     2    79.65        S  First woman FALSE        1
42 female  60     1     1     79.2        C  First   man FALSE        1
43   male 152     1     0  78.2667        C  First woman FALSE        1
44 female 149     0     0        0        S   <NA>   man  TRUE        0
>
```

Description:

This code uses the distinct() function to find unique rows in mydata based on the age column. The result, stored in duplicates_age, displays each unique row where the age value is distinct, preserving all columns (keep_all = TRUE).

30.Finding And Removing Duplicate Value(Fare)

Code:

```
duplicates_fare<-distinct(mydata,fare,.keep_all = TRUE)

duplicates_fare
```

Output:

35	female	8	1	1	36.75	S	Second	child	FALSE	1
36	female	17	0	2	110.8833	C	First	man	FALSE	1
37	female	25	0	0	7.8292	Q	Third	man	TRUE	0
38	male	22	0	0	7.775	S	Third	woman	TRUE	1
39	male	NA	1	0	39.6	C	First	woman	FALSE	1
40	female	24	0	0	227.525	C	First	man	TRUE	0
41	male	39	1	1	79.65	S	First	woman	FALSE	1
42	male	36	1	0	17.4	S	Third	woman	FALSE	1
43	female	28	0	0	13.5	S	<NA>	man	TRUE	0
44	female	30	0	0	8.05	S	Third	man	TRUE	0
45	female	24	2	0	24.15x	S	Third	man	FALSE	0
46	male	29	0	4	21.075	S	Third	woman	FALSE	0
47	female	32	0	0	7.8542	S	Third	man	TRUE	1
48	male	53	2	0	51.4792	S	First	woman	FALSE	1
49	female	36	0	0	26.3875	S	First	man	TRUE	1
50	male	NA	0	0	7.75y	Q	Third	woman	TRUE	1
51	male	34	0	0	13	S	Second	woman	TRUE	1
52	male	39	1	0	55.9	S	First	woman	FALSE	1
53	male	NA	1	0	14.4583	C	Third	woman	FALSE	0
54	male	25	1	1	30	S	Second	woman	FALSE	0
55	female	36	0	0	40.125	C	First	man	TRUE	0
56	female	NA	0	0	8.7125	C	Third	man	TRUE	0
57	female	47	0	0	15	S	Second	man	TRUE	0
58	female	60	1	1	79.2	C	First	man	FALSE	1
59	female	35	0	0	7.125	S	Third	man	TRUE	0
60	male	152	1	0	78.2667	C	First	woman	FALSE	1
61	female	47	0	0	7.25	S	Third	man	TRUE	0
62	male	NA	0	0	33	S	Second	woman	TRUE	1
63	female	149	0	0	0	S	<NA>	man	TRUE	0

Description:

This code uses the distinct() function to find unique rows in mydata based on the fare column. The result, stored in duplicates_fare, displays each unique row where the fare value is distinct, preserving all columns (keep_all = TRUE). This helps in identifying unique fare values in the dataset.

31.Filtering Data by Criteria

Code:

```
data<-filter(mydata,Gender=="female")

data

data<-filter(mydata,age > 15 & age < 45)

data

data<-filter(mydata,sibsp >1 )

data

data<-filter(mydata,parch >1 )

data

data<-filter(mydata,embarked=="Q")

data
```

data<-filter(mydata,class=="Third")

data

Output:

```
   Gender age sibsp parch     fare embarked  class   who alone survived
1  female  24     0     0   7.7958        S  Third woman  TRUE        0
2  female  17     0     0   8.6625        S  Third woman  TRUE        0
3  female  36     1     0    108.9        C  First woman FALSE        0
4  female  40     0     0    26.55        S  First woman  TRUE        1
5  female  28     0     0   22.525        S  Third woman  TRUE        0
6  female  26     0     0  56.4958        S  Third woman  TRUE        1
7  female  29     0     0     7.75        Q  Third woman  TRUE        0
8  female  30     0     0     <NA>        S  Third woman  TRUE        0
9  female  36     0     0  26.2875        S  Third woman  TRUE        1
10 female  24     0     0   7.4958        S  Third woman  TRUE        0
```

```
> data<-filter(mydata,age > 15 & age < 45)
> data
   Gender age sibsp parch     fare embarked  class   who alone survived
1  female  24     0     0   7.7958        S  Third woman  TRUE        0
2  female  17     0     0   8.6625        S  Third woman  TRUE        0
3    male  21     0     0     7.75        Q  Third   man  TRUE        0
4    male  35     0     0   7.6292        Q  Third   man  TRUE        0
5    male  37     0     0   9.5875        S  Third   man  TRUE        0
6    male  16     0     0     86.5        S  First   man  TRUE        1
7  female  36     1     0    108.9        C  First woman FALSE        0
```

```
> data<-filter(mydata,sibsp >1 )
> data
  Gender age sibsp parch    fare embarked  class   who alone survived
1 female  23     2     1    11.5        S Second woman FALSE        0
2 female   9     4     2  31.275        S  Third woman FALSE        0
3   male  11     4     2  31.275        S  Third   man FALSE        0
4 female  24     2     0  24.15x        S  Third woman FALSE        0
5   male  53     2     0 51.4792        S  First   man FALSE        1
>
```

```
> data<-filter(mydata,parch >1 )
> data
   Gender age sibsp parch     fare embarked  class   who alone survived
1    male  33     0     2     <NA>        S Second   man FALSE        0
2    male  20     0     2  22.3583        C  Third   man FALSE        1
3    male  36     0     2    26.25        S Second   man FALSE        1
4    male  22     0     2     49.5        C  First   man FALSE        1
5    male  36     0     2       71        S  First   man FALSE        1
6  female   9     4     2   31.275        S  Third woman FALSE        0
7    male  11     4     2   31.275        S  Third   man FALSE        0
8  female  17     0     2 110.8833        C  First woman FALSE        1
9    male  29     0     4   21.075        S  Third   man FALSE        0
10   male  18     0     2    79.65        S  First   man FALSE        1
11   male  36     0     2     7.75        Q  Third   man FALSE        0
>
```

```
> data<-filter(mydata,embarked=="Q")
> data
   Gender age sibsp parch   fare embarked class   who alone survived
1    male  21     0     0   7.75        Q Third   man  TRUE        0
2    male  35     0     0 7.6292        Q Third   man  TRUE        0
3  female  29     0     0   7.75        Q Third woman  TRUE        0
4  female  55     0     0  24.15        Q Third woman  TRUE        0
5  female  41     0     0   7.75        Q Third woman  TRUE        0
6  female  41     0     0   7.75        Q Third woman  TRUE        0
7  female  41     0     0   7.75        Q Third woman  TRUE        0
8  female  25     0     0 7.8292        Q Third woman  TRUE        0
9  female  34     0     0   7.75        Q Third woman  TRUE        0
10   male  36     0     0  7.75y        Q Third   man  TRUE        1
11   male  36     0     2   7.75        Q Third   man FALSE        0
>
```

```
> data<-filter(mydata,class=="Third")
> data
   Gender age sibsp parch    fare embarked class   who alone survived
1  female  24     0     0  7.7958        S Third woman  TRUE        0
2  female  17     0     0  8.6625        S Third woman  TRUE        0
3    male  21     0     0    7.75        Q Third   man  TRUE        0
4    male  35     0     0  7.6292        Q Third   man  TRUE        0
5    male  37     0     0  9.5875        S Third   man  TRUE        0
6  female  28     0     0  22.525        S Third woman  TRUE        0
7  female  26     0     0 56.4958        S Third woman  TRUE        1
8  female  29     0     0    7.75        Q Third woman  TRUE        0
9  female  30     0     0    <NA>        S Third woman  TRUE        0
10 female  36     0     0 26.2875        S Third woman  TRUE        1
```

Description:

This code demonstrates filtering operations on mydata using different criteria:

- Gender == "female": Filters rows where Gender is "female".
- age > 15 & age < 45: Filters rows where age is between 15 and 45.
- sibsp > 1: Filters rows where sibsp (number of siblings/spouses aboard) is greater than 1.
- parch > 1: Filters rows where parch (number of parents/children aboard) is greater than 1.
- embarked == "Q": Filters rows where embarked (port of embarkation) is "Q" (Queenstown).
- class == "Third": Filters rows where class is "Third".

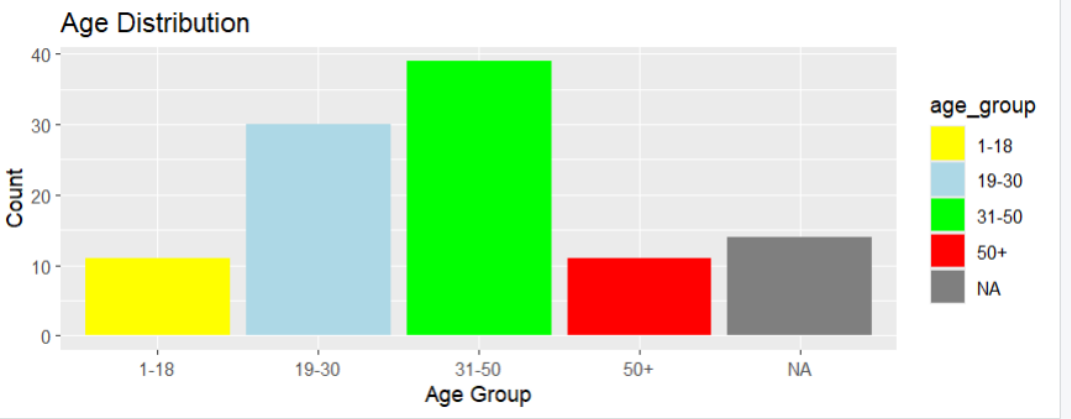
Bar Plot:

32.Age Distribution:

Code:

```
mydata$age_group <- cut(mydata$age, breaks = c(0, 18, 30,50, Inf),
                        labels = c("1-18", "19-30","31-50", "50+"), include.lowest = TRUE)
age_plot <- ggplot(mydata, aes(x = age_group, fill = age_group)) +
  geom_bar() +
  labs(title = "Age Distribution",
       x = "Age Group",
       y = "Count") +
  scale_fill_manual(values = c("1-18" = "yellow", "19-30" = "lightblue","31-50" = "green", "50+" = "red")) +
  scale_x_discrete(labels = c("1-18" = "1-18", "19-30" = "19-30","31-50" = "31-50", "50+"= "50+"))
print(age_plot)
```

Output:



Description:

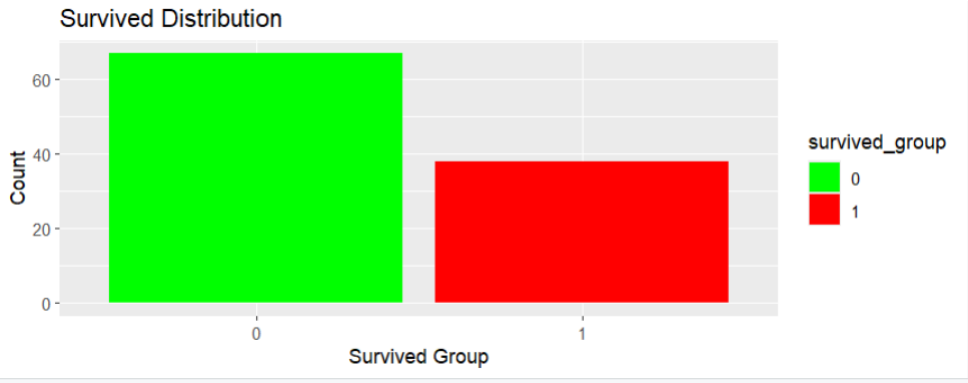
This code uses ggplot2 to create a bar plot (age_plot) displaying the distribution of passenger ages categorized into groups (age_group) in mydata, with manual color assignments for each age group.

33.Survival Distribution

Code:

```
mydata$survived_group <- cut(mydata$survived, breaks = c(-Inf, 0.5, Inf),
                             labels = c("0", "1"))
survived_plot <- ggplot(mydata, aes(x = survived_group, fill = survived_group)) +
  geom_bar() +
  labs(title = "Survived Distribution",
       x = "Survived Group",
       y = "Count") +
  scale_fill_manual(values = c("0" = "green", "1" = "red")) +
  scale_x_discrete(labels = c("0" = "0", "1" = "1"))
print(survived_plot)
```

Output:



Description:

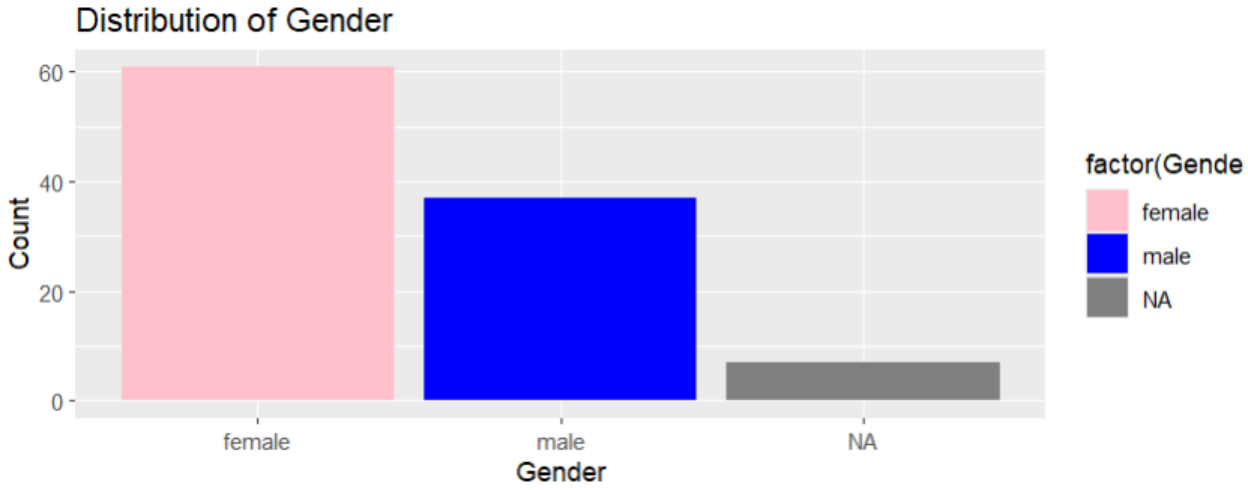
This code creates a bar plot (survived_plot) using ggplot2 to visualize the distribution of survival status in mydata, categorized into groups ("0" for not survived, "1" for survived). Colors are manually assigned to each group, with labels and titles for clarity.

34.Gender Distribution:

Code:

```
sex_plot <- ggplot(mydata, aes(x = factor(Gender), fill = factor(Gender))) +
  geom_bar() +
  labs(title = "Distribution of Gender",
       x = "Gender",
       y = "Count") +
  scale_fill_manual(values = c("male" = "blue", "female" = "pink")) +
  scale_x_discrete(labels = c("male" = "male", "female" = "female"))
print(sex_plot)
```

Output:



Description:

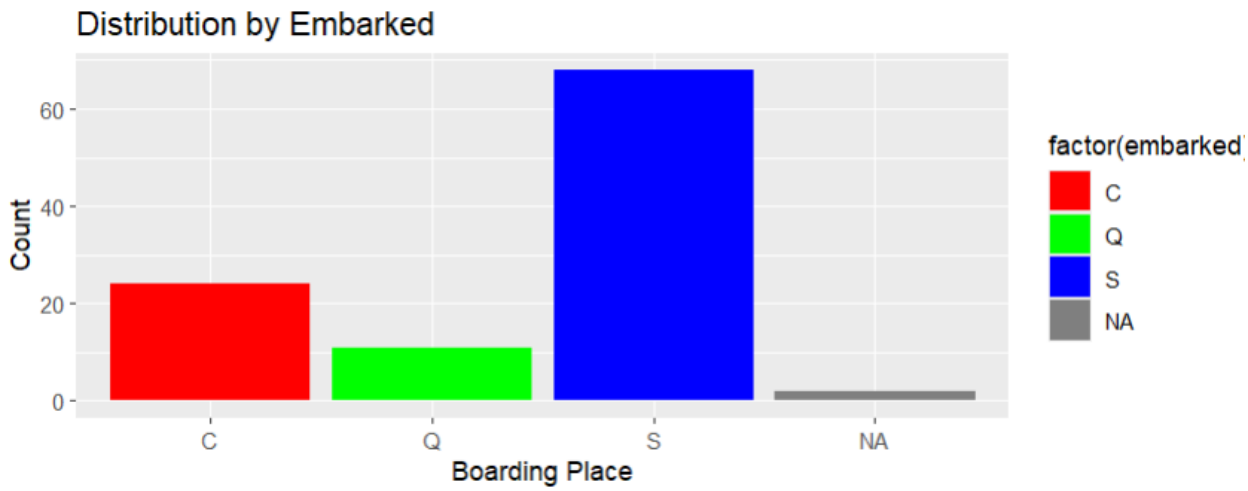
This code utilizes ggplot2 to create a bar plot (sex_plot) illustrating the distribution of gender (Gender) in mydata. Each bar represents the count of passengers categorized as "male" or "female", with manual color assignments and axis labels (x for Gender and y for Count) for enhanced visualization.

35.Embarked Distribution:

Code:

```
embarked_plot <- ggplot(mydata, aes(x = factor(embarked), fill = factor(embarked))) +
  geom_bar() +
  labs(title = "Distribution by Embarked",
       x = "Boarding Place",
       y = "Count") +
  scale_fill_manual(values = c("C" = "red", "Q" = "green","S" = "blue")) +
  scale_x_discrete(labels = c("C" = "C", "Q" = "Q","S" = "S"))
print(embarked_plot)
```

Output:



Description:

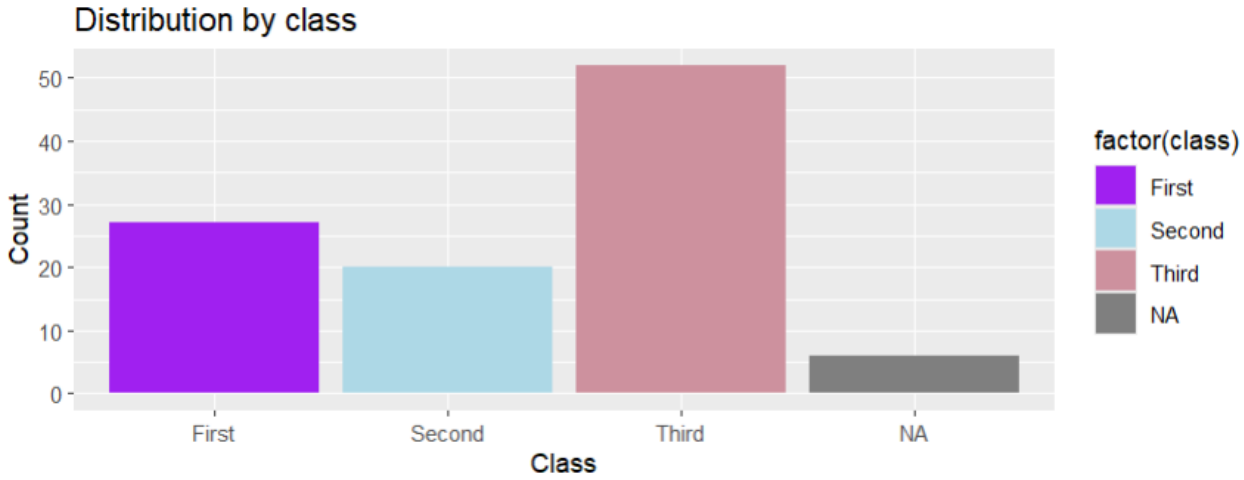
This code creates a bar plot (embarked_plot) using ggplot2 to visualize the distribution of passengers based on their boarding place (embarked) in mydata. Each bar represents the count of passengers who embarked from "C" (Cherbourg), "Q" (Queenstown), and "S" (Southampton), with manually assigned colors and axis labels (x for Boarding Place and y for Count) for clarity and visual appeal.

36.Class Distribution:

Code:

```
class_plot <- ggplot(mydata, aes(x = factor(class), fill = factor(class))) +
geom_bar() +
labs(title = "Distribution by class",
     x = "Class",
     y = "Count") +
scale_fill_manual(values = c("First" = "purple", "Second" = "lightblue","Third" = "pink3")) +
scale_x_discrete(labels = c("First" = "First", "Second" = "Second","Third" = "Third"))
print(class_plot)
```

Output:



Description:

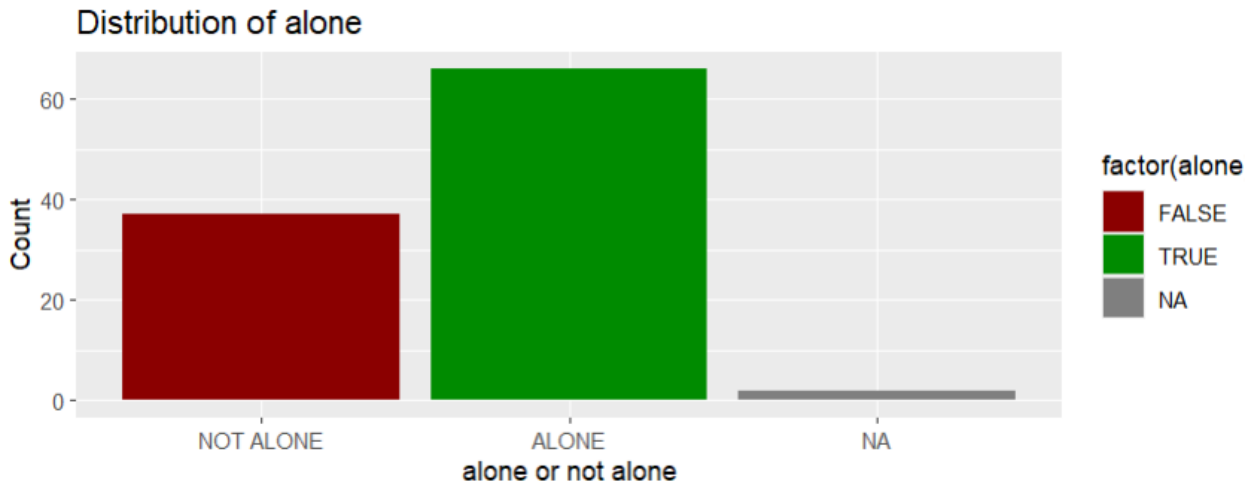
This code creates a bar plot (class_plot) using ggplot2 to visualize the distribution of passengers based on their class (class) in mydata. Each bar represents the count of passengers in "First", "Second", and "Third" classes, with manually assigned colors and axis labels (x for Class and y for Count) for clarity and visual appeal.

37.Alone Status Distribution:

Code:

```
alone_plot <- ggplot(mydata, aes(x = factor(alone), fill = factor(alone))) +
  geom_bar() +
  labs(title = "Distribution of alone",
       x = "alone or not alone",
       y = "Count") +
  scale_fill_manual(values = c("TRUE" = "green4", "FALSE" = "red4")) +
  scale_x_discrete(labels = c("TRUE" = "ALONE", "FALSE" = "NOT ALONE"))
print(alone_plot)
```

Output:



Description:

This code utilizes ggplot2 to create a bar plot (alone_plot) illustrating the distribution of passengers based on whether they were traveling alone (alone) in mydata. Each bar represents the count of passengers categorized as "ALONE" (True) or "NOT ALONE" (False), with manual color assignments and axis labels (x for Alone or Not Alone and y for Count) for clarity and visual appeal.

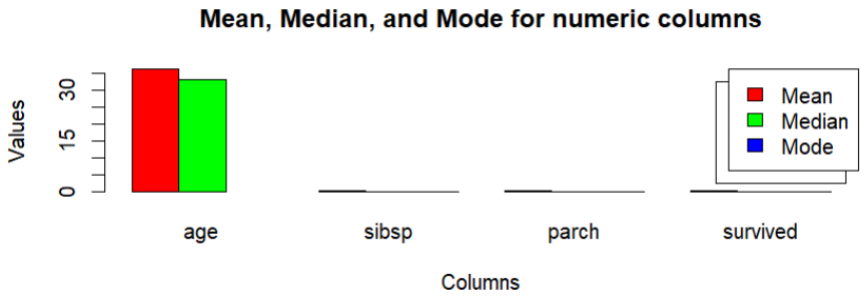
38.Statistical Summary Visualization

Code:

```
 selected_columns <- c("age", "sibsp", "parch", "survived")

selected_data <- mydata[selected_columns]

means <- sapply(selected_data, mean, na.rm = TRUE)

medians <- sapply(selected_data, median, na.rm = TRUE)

mode_func <- function(x) {

  ux <- unique(x)

  ux[which.max(tabulate(match(x, ux)))]

}

modes <- sapply(selected_data, mode_func)

summary_data <- data.frame(Columns = names(selected_data),Mean = means,Median = medians,Mode = modes)

print(summary_data)

barplot(t(summary_data[, -1]), beside = TRUE, main = "Mean, Median, and Mode for numeric columns",xlab = "Columns", ylab = "Values",col = c("red", "green", "blue"),legend.text = TRUE)

legend("topright", legend = colnames(summary_data)[-1], fill = c("red", "green", "blue"))
```

Output:

```
> print(summary_data)
          Columns       Mean Median Mode
age           age 36.2527473     33   NA
sibsp       sibsp  0.3495146      0    0
parch       parch  0.3398058      0    0
survived survived  0.3619048      0    0
```



Description:

This code computes and visualizes the mean, median, and mode for numeric columns ("age", "sibsp", "parch", "survived") in mydata. It first calculates these measures using sapply() and custom functions for mode calculation (mode_func). The summary data (summary_data) is then displayed, showing columns along with their respective mean, median, and mode values. Finally, a bar plot visualizes these statistics side by side, providing a comparative view across columns with legends for clarity.

39.Processing Fare Attribute:

Code:

```
str(mydata$fare)

mydata$fare <- as.numeric(mydata$fare)

mean_fare <- as.integer(mean(mydata$fare, na.rm = TRUE))

print(mean_fare)
```

Output:

```
> str(mydata$fare)
 num [1:105] 7.8 8.66 7.75 7.63 9.59 ...
>
```

```
> print(mean_fare)
[1] 31
```

Description:

This code snippet first inspects the structure of the fare attribute in mydata using str() to understand its current type and structure. Next, it converts fare to numeric format using as.numeric() to facilitate numerical operations. Lastly, it computes the mean fare (mean_fare) from mydata, ignoring missing values (na.rm = TRUE), and prints the result as an integer.

40.Replace Missing Values with Mean(Fare)

Code:

```
mydata <- mydata %>%

mutate(fare = ifelse(is.na(fare), mean_fare, fare))

mydata
```

Output:

```
> mydata <- mydata %>%
+   mutate(fare = ifelse(is.na(fare), mean_fare, fare))
> mydata
    Gender age sibsp parch     fare embarked  class   who alone survived
1   female  24     0     0   7.7958        S  Third mannn  TRUE        0
2   female  17     0     0   8.6625        S  Third   man  TRUE        0
3     male  21     0     0   7.7500        Q  Third woman  TRUE        0
4     male  35     0     0   7.6292        Q  Third woman  TRUE        0
5     male  37     0     0   9.5875        S  Third woman  TRUE        0
6     male  16     0     0  86.5000        S  First woman  TRUE        1
7   female  NA     1     0 108.9000        C  First mannn FALSE        0
8     male  33     0     2  31.0000        S Second woman FALSE        0
9   female  40     0     0  26.5500        S  First   man  TRUE        1
10  female  28     0     0  22.5250        S  Third   man  TRUE        0
11  female  26     0     0  56.4958        S  Third   man  TRUE        1
12  female  29     0     0   7.7500        Q  Third   man  TRUE        0
13  female  30     0     0  31.0000        S  Third   man  TRUE        0
14    <NA>  36     0     0  26.2875        S   <NA>   man  TRUE        1
15    male  54     1     0  59.4000        C  First woman FALSE        0
16  female  24     0     0   7.4958        S  Third   man  TRUE        0
17  female  47     0     0  31.0000        S  First   man  TRUE        0
18    male  34     0     0  10.5000        S Second woman  TRUE        1
19  female  55     0     0  24.1500        Q  Third   man  TRUE        0
20    male  36     1     0  26.0000        S Second woman FALSE        1
21    male  36     1     0  26.0000        S Second woman FALSE        1
22  female  NA     0     0   7.8958        S  Third   man  TRUE        0
```

Description:

This code snippet utilizes the mutate() function from dplyr to replace missing values in the fare column of mydata with the previously computed mean fare (mean_fare). The updated dataset (mydata) is then displayed to show the changes made.

41.Rounding Operations on Fare Data

Code:

round(mydata$fare)

ceiling(mydata$fare)

floor(mydata$fare)

Output:

```
> round(mydata$fare)
  [1]   8   9   8   8  10  86 109  31  27  23  56   8  31  26  59   7  31  10  24  26  26   8  94   8   7  58   7
 [28]   8   8   8  10 222   8  12  26   7   7  22   9  26  27 106  14  50  71  31  31  26 106  26  26  14  21  37
 [55] 111  26   8   7   8  27  40 228  80  17   8   8  14   8   8  31   8  21   7   8  10  51  26  31   8  14  13
 [82]  56  14   8  30 111  26  40   9  80  15  79   8   8   7  78   7   8  26  24  33   0   7  31  31
> ceiling(mydata$fare)
  [1]   8   9   8   8  10  87 109  31  27  23  57   8  31  27  60   8  31  11  25  26  26   8  94   8   8  58   8
 [28]   8   8   8  11 222   8  12  26   8   8  23   9  27  27 107  15  50  71  32  32  26 107  26  26  14  21  37
 [55] 111  26   8   8   8  27  40 228  80  18   8   8  14   9   9  31   8  22   8   8  11  52  27  31   9  15  13
 [82]  56  15   8  30 111  26  41   9  80  15  80   9   9   8  79   8   8  26  25  33   0   8  31  31
> floor(mydata$fare)
  [1]   7   8   7   7   9  86 108  31  26  22  56   7  31  26  59   7  31  10  24  26  26   7  93   7   7  57   7
 [28]   7   7   7  10 221   7  11  26   7   7  22   8  26  26 106  14  49  71  31  31  26 106  26  26  13  20  36
 [55] 110  26   7   7   7  26  39 227  79  17   7   7  13   8   8  31   7  21   7   7  10  51  26  31   8  14  13
 [82]  55  14   7  30 110  26  40   8  79  15  79   8   8   7  78   7   7  26  24  33   0   7  31  31
>
```

Description:

This code snippet demonstrates the application of rounding operations (round(), ceiling(), and floor()) on the fare column of mydata.

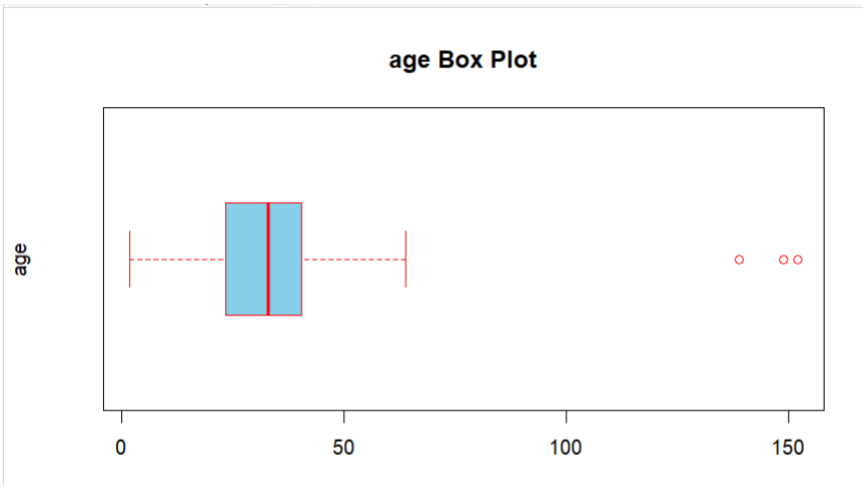
- round(mydata$fare) rounds each fare value to the nearest integer.
- ceiling(mydata$fare) rounds each fare value up to the nearest integer.
- floor(mydata$fare) rounds each fare value down to the nearest integer.

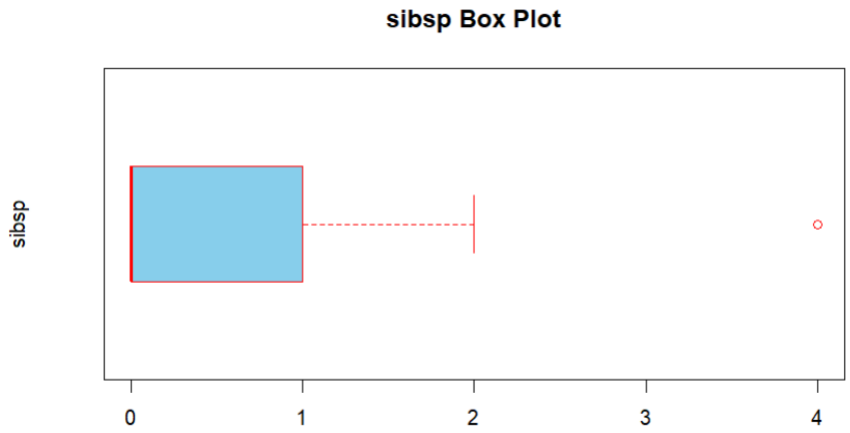
42.Generating boxplot for numerical column

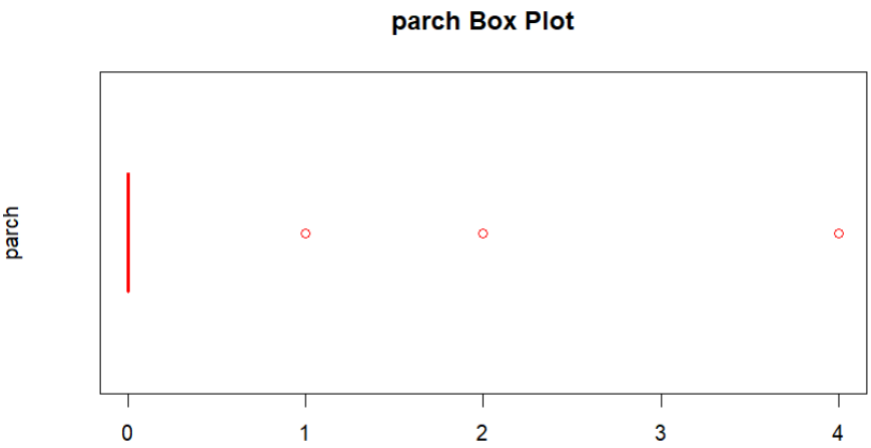
Code:

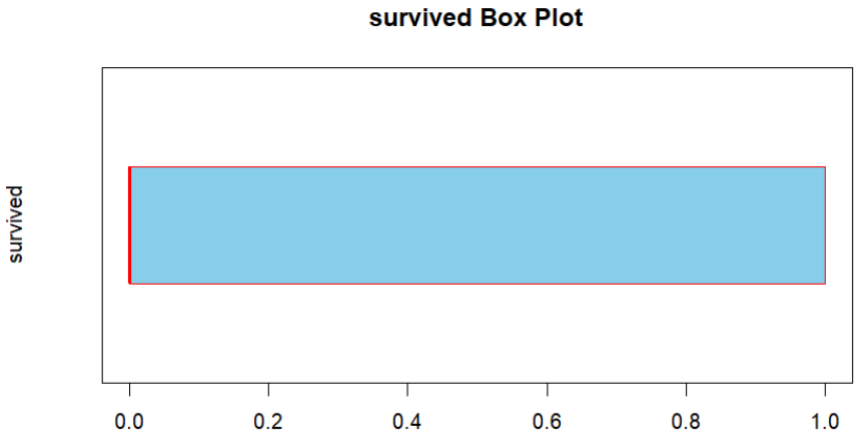
```
numerical_columns <- sapply(mydata, is.numeric)

for (col in names(mydata)[numerical_columns]) {

  boxplot(mydata[[col]], main = paste(col, "Box Plot"), ylab = col, col = "skyblue",

          border = "red", notch = FALSE, horizontal = TRUE)

}
```

Output:


age Box Plot
age
0
50
100
150


sibsp Box Plot
sibsp
0
1
2
3
4


parch Box Plot
parch
0
1
2
3
4



Description:

This code generates a horizontal boxplot for each numeric column in the mydata data frame. It first identifies which columns are numeric using sapply. Then, it iterates over these columns, creating a boxplot with specific styling options such as sky blue color and red borders. Each boxplot is titled with the column name followed by "Box Plot".

43.Balancing the “Survived ” column in the dataset

Code:

```
survived_distribution <- mydata %>% count(survived)

print(survived_distribution)


majority_survived_label <- survived_distribution %>% filter(n == max(n)) %>% pull(survived)

minority_survived_label <- survived_distribution %>% filter(n == min(n)) %>% pull(survived)


print(paste("Majority class:", majority_survived_label))

print(paste("Minority class:", minority_survived_label))


majority_survived <- mydata %>% filter(survived == majority_survived_label)

minority_survived <- mydata %>% filter(survived == minority_survived_label)
```

```
set.seed(123)

undersampled_majority <- majority_survived %>% sample_n(nrow(minority_survived))


balanced_dataset <- bind_rows(undersampled_majority, minority_survived)

balanced_dataset


balanced_distribution <- balanced_dataset %>%

  count(survived) %>%

  mutate(percentage = n / sum(n) * 100)


print(balanced_distribution)
```

Output:

```
> balanced_dataset <- bind_rows(undersampled_majority, minority_survived)
> balanced_dataset
    Gender age sibsp parch     fare embarked  class   who alone survived
1     male  11     4     2   31.275        S  Third child FALSE        0
2   female  32     0     0    7.925        S  Third   man  TRUE        0
3   female  55     0     0    24.15        Q  Third   man  TRUE        0
4     male  21     0     0     7.75        Q  Third woman  TRUE        0
5   female  30     0     0     8.05        S  Third   man  TRUE        0
6     male  NA     1     0  14.4583        C  Third woman FALSE        0
7   female  36     0     0   40.125        C  First   man  TRUE        0
8     male  40     0     0     8.05        S  Third woman  TRUE        0
9   female  62     0     0    26.55        S  First   man  TRUE        0
10    male  25     1     1       30        S Second woman FALSE        0
11  female  NA     0     0    7.225        C  Third   man  TRUE        0
12    male  NA     0     2     7.75        Q  Third woman FALSE        0
13  female  10     0     0   7.2292        C  Third   man  TRUE        0
14  female  17     1     1   7.2292        C  Third   man FALSE        0
15    male  30     0     0   8.6625        S  Third woman  TRUE        0
16    male  37     0     0   9.5875        S  Third woman  TRUE        0
17    <NA>  NA    NA    NA     <NA>     <NA>   <NA>  <NA>    NA        0
18  female  54     0     0       26        S Second   man  TRUE        0
19  female  45     0     0    26.55        S  First   man  TRUE        0
20  female  29     0     0     7.75        Q  Third   man  TRUE        0
21  female  32     0     0     14.5        S  Third   man  TRUE        0
22  female  27     0     0       26        S Second   man  TRUE        0
23  female  28     0     0   22.525        S  Third   man  TRUE        0
24  female  47     0     0       15        S Second   man  TRUE        0
25    male  33     0     2     <NA>        S Second woman FALSE        0
```

```
> balanced_distribution <- balanced_dataset %>%
+   count(survived) %>%
+   mutate(percentage = n / sum(n) * 100)
>
> print(balanced_distribution)
  survived  n percentage
1        0 38         50
2        1 38         50
>
```

Description:

This code balances the 'survived' column in a dataset by undersampling the majority class. It first determines the distribution of the 'survived' column and identifies the majority and minority classes. Then, it undersamples the majority class to match the size of the minority class and combines them to create a balanced dataset. Finally, it checks and prints the new distribution to confirm the balance.

44.Balancing the “Gender ” column in the dataset

Code:

```
Gender_distribution <- mydata %>% count(Gender)

print(Gender_distribution)


majority_Gender_label <- Gender_distribution %>% filter(n == max(n)) %>% pull(Gender)

minority_Gender_label <- Gender_distribution %>% filter(n == min(n)) %>% pull(Gender)


print(paste("Majority class:", majority_Gender_label))

print(paste("Minority class:", minority_Gender_label))


majority_Gender <- mydata %>% filter(Gender == majority_Gender_label)

minority_Gender <- mydata %>% filter(Gender == minority_Gender_label)
```

```
set.seed(123)

undersampled_majority <- majority_Gender %>% sample_n(nrow(minority_Gender))


balanced_dataset <- bind_rows(undersampled_majority, minority_Gender)

balanced_dataset


balanced_distribution <- balanced_dataset %>%

  count(Gender) %>%

  mutate(percentage = n / sum(n) * 100)


print(balanced_distribution)
```

Output:

```
> balanced_dataset <- bind_rows(undersampled_majority, minority_Gender)
> balanced_dataset
   Gender age sibsp parch     fare embarked  class   who alone survived
1  female  64     0     0       26        S  First   man  TRUE        0
2  female  16     0     0     8.05        S  Third   man  TRUE        0
3  female  22     0     0   7.8958        S  Third   man  TRUE        0
4  female  40     0     0   7.8958        S  Third   man  TRUE        0
5  female  36     0     0  26.3875        S  First   man  TRUE        1
6  female  54     0     0       26        S Second   man  TRUE        0
7  female  28     0     0     13.5        S   <NA>   man  TRUE        0
8  female  25     0     0   7.8292        Q  Third   man  TRUE        0
9  female  19     0     0     14.5        S  Third   man  TRUE        0
10 female  NA     0     0    7.225        C  Third   man  TRUE        0
11 female  37     1     0       26        S Second   man FALSE        0
12 female  17     1     1   7.2292        C  Third   man FALSE        0
13 female  45     0     0    26.55        S  First   man  TRUE        0
14 female  32     0     0     14.5        S  Third   man  TRUE        0
15 female  28     0     0   22.525        S  Third   man  TRUE        0
16 female  NA    NA    NA     <NA>     <NA>   <NA>  <NA>    NA        0
17 female  36     0     0   40.125        C  First   man  TRUE        0
18 female   9     4     2   31.275        S  Third child FALSE        0
19 female  36     0     0  26.2875        S   <NA>   man  TRUE        1
20 female  32     1     0       26        S Second   man FALSE        1
21 female  17     0     2 110.8833        C  First   man FALSE        1
22 female  30     0     0     <NA>        S  Third   man  TRUE        0
23 female  NA     0     0   8.7125        C  Third   man  TRUE        0
24 female  29     0     0     7.75        Q  Third   man  TRUE        0
25 female 149     0     0        0        S   <NA>   man  TRUE        0
26 female  36     1     1    24.15        S  Third   man FALSE        0
```

```
> balanced_distribution <- balanced_dataset %>%
+   count(Gender) %>%
+   mutate(percentage = n / sum(n) * 100)
>
> print(balanced_distribution)
  Gender  n percentage
1 female 37         50
2   male 37         50
>
```

Description:

This code balances the 'Gender' column in a dataset by undersampling the majority gender class. It identifies the majority and minority gender classes and then undersamples the majority class to match the size of the minority class. The undersampled majority class is combined with the minority class to create a balanced dataset. Finally, it checks and prints the new distribution to ensure balance.

45.Balancing the “Alone ” column in the dataset

Code:

```
alone_distribution <- mydata %>% count(alone)

print(alone_distribution)



majority_alone_label <- alone_distribution %>% filter(n == max(n)) %>% pull(alone)

minority_alone_label <- alone_distribution %>% filter(n == min(n)) %>% pull(alone)



print(paste("Majority class:", majority_alone_label))

print(paste("Minority class:", minority_alone_label))



majority_alone <- mydata %>% filter(alone == majority_alone_label)

minority_alone <- mydata %>% filter(alone == minority_alone_label)



set.seed(123)

undersampled_majority <- majority_alone %>% sample_n(nrow(minority_alone))
```

```
balanced_dataset <- bind_rows(undersampled_majority, minority_alone)

balanced_dataset


balanced_distribution <- balanced_dataset %>%

  count(alone) %>%

  mutate(percentage = n / sum(n) * 100)


print(balanced_distribution)
```

Output:

```
> balanced_dataset <- bind_rows(undersampled_majority, minority_alone)
> balanced_dataset
   Gender age sibsp parch     fare embarked  class   who alone survived
1    male  30     0     0  106.425        C  First woman  TRUE        1
2    male  NA     0     0    7.75y        Q  Third woman  TRUE        1
3  female  47     0     0     <NA>        S  First   man  TRUE        0
4  female  40     0     0   7.8958        S  Third   man  TRUE        0
5  female  36     0     0  26.3875        S  First   man  TRUE        1
6    male  34     0     0       13        S Second woman  TRUE        1
7  female  28     0     0     13.5        S   <NA>   man  TRUE        0
8  female  22     0     0    7.225        C  Third   man  TRUE        1
9  female  16     0     0     8.05        S  Third   man  TRUE        0
10 female  NA     0     0    7.225        C  Third   man  TRUE        0
11 female  47     0     0     7.25        S  Third   man  TRUE        0
12   male  50     0     0     10.5        S Second woman  TRUE        1
13 female  45     0     0 221.7792        S  First   man  TRUE        0
14 female  48     0     0    7.925        S  Third   man  TRUE        0
15   male  37     0     0   9.5875        S  Third woman  TRUE        0
16   <NA>  NA    NA    NA     <NA>     <NA>   <NA>  <NA>  TRUE        0
17 female  32     0     0    7.925        S  Third   man  TRUE        0
18 female  10     0     0   7.2292        C  Third   man  TRUE        0
19 female  26     0     0  56.4958        S  Third   man  TRUE        1
20   male  30     0     0   8.6625        S  Third woman  TRUE        0
21 female  27     0     0       26        S Second   man  TRUE        0
22 female  28     0     0   22.525        S  Third   man  TRUE        0
23 female  54     0     0       26        S Second   man  TRUE        0
24 female  40     0     0    26.55        S  First   man  TRUE        1
25 female 149     0     0        0        S   <NA>   man  TRUE        0
26   male  NA     0     0       33        S Second woman  TRUE        1
27   <NA>  22     0     0   7.8958        S  Third   man  TRUE        0
28 female  25     0     0   7.8292        Q  Third   man  TRUE        0
29 female  47     0     0       15        S Second   man  TRUE        0
30 female  NA     0     0   7.8958        S  Third   man  TRUE        0
```

```
> balanced_distribution <- balanced_dataset %>%
+   count(alone) %>%
+   mutate(percentage = n / sum(n) * 100)
>
> print(balanced_distribution)
  alone  n percentage
1 FALSE 37         50
2  TRUE 37         50
>
```

Description:

This code balances the 'alone' column in a dataset by undersampling the majority class. It first determines the distribution of the 'alone' column and identifies the majority and minority classes. The majority class is then undersampled to match the size of the minority class, and both classes are combined to create a balanced dataset. Finally, it checks and prints the new distribution to confirm the balance.