AMERICAN INTERNATIONAL UNIVERSITY-BANGLADESH

Faculty of Science and Technology

Project Title:	**Data Preparation and Analysis on Heart Prediction Quantum Dataset using R**
Date of Submission:	**27 April 2025**
Course Title:	**INTRODUCTION TO DATA SCIENCE**
Section:	**D**
Semester:	**Spring 2024-25 CourseTeacher: Tohedul Islam**

Declaration and Statement of Authorship:

1. I/we hold a copy of this Assignment/Case-Study, which can be produced if the original is lost/damaged.
2. This Assignment/Case-Study is my/our original work and no part of it has been copied from any other student's work or from any other source except where due acknowledgement is made.
3. No part of this Assignment/Case-Study has been written for me/us by any other person except where such collaboration has been authorized by the concerned teacher and is clearly acknowledged in the assignment.
4. I/we have not previously submitted or currently submitting this work for any other course/unit.
5. This work may be reproduced, communicated, compared and archived for the purpose of detecting plagiarism.
6. I/we give permission for a copy of my/our marked work to be retained by the faculty for review and comparison, including review by external examiners.
7. I/we understand that Plagiarism is the presentation of the work, idea or creation of another person as though it is your own. It is a formofcheatingandisaveryseriousacademicoffencethatmayleadtoexpulsionfromtheUniversity. Plagiarized material can be drawn from, and presented in, written, graphic and visual form, including electronic data, and oral presentations. Plagiarism occurs when the origin of their arterial used is not appropriately cited.
8. I/we also understand that enabling plagiarism is the act of assisting or allowing another person to plagiarize or to copy my/our work.

* *Student(s) must complete all details except the faculty use part.*

** Please submit all assignments to your course teacher or the office of the concerned teacher

Group No: 5

No	Name	ID	Program	Signature
1	**TAHMEED ALI PATWARY**	**21-44428-1**	**BSc [CSE]**	
2	**JINIA SULTANA SAMA**	**22-46301-1**	**BSc [CSE]**	
3	**KANIZ FARIA AHAMED**	**22-46429-1**	**BSc [CSE]**	
4	**AIRIN AKTHER**	**22-46744-1**	**BSc [CSE]**	

Faculty use only		
FACULTY COMMENTS	**Marks Obtained**	
	Total Marks	

Project: Data Preparation and Analysis of Heart Prediction Quantum Dataset

Introduction of the dataset:
The Heart Prediction Quantum Dataset is used to predict heart disease and combines traditional medical data with an additional feature called QuantumPatternFeature. It has important factors like Age, Gender, Blood Pressure, Cholesterol, and Heart Rate, which are commonly used to assess the risk of heart disease.The QuantumPatternFeature adds complexity by identifying detailed patterns, making the dataset useful for advanced predictive models, including both classical and quantum machine learning approaches.

The dataset contains 500 samples and 7 attributes. Below is a detailed description of these attributes:

1. **Age:** The patient's age in years, represented as an integer.
2. **Gender:** The patient's gender, commonly "Male" or "Female" (categorical).
3. **BloodPressure:** Measurement of resting blood pressure in mm Hg, given as an integer.
4. **Cholesterol:** Level of serum cholesterol in mg/dl, represented as an integer.
5. **HeartRate:** The highest heart rate reached during exercise, indicated as an integer.
6. **QuantumPatternFeature:** A feature that captures intricate, non-linear patterns for quantum modeling, expressed as a float.
7. **HeartDisease:** Indicates whether the patient has heart disease (1 = yes, 0 = no), represented as an integer.

R Program Code with Explanation:

1. This code installs the required R packages and loads them into the environment for performing data manipulation, visualization, handling missing data, and calculating modes.

```
install.packages(c("dplyr", "ggplot2"))
install.packages("naniar")
install.packages("modeest")
install.packages("readxl")
library(dplyr)       # For data cleaning and manipulation
library(ggplot2)
library(naniar)
library(modeest)
library(readxl)
```

dplyr: Performs data manipulation tasks.
ggplot2: Creates custom data visualizations.
naniar: Visualizes and handles missing data.
modeest: Calculates modes of variables.
readxl: Reads Excel files into R.

2. Code:

```
heart_data<-read_excel("C:/Users/jinia/Desktop/CSE/Semester10/Data Science/Dataset_MIdterm_sectoin(D).xlsx")
heart_data
str(heart_data)
head(heart_data)
summary(heart_data)
```

Output:

```
> heart_data <- read_excel("C:/Users/jinia/Desktop/CSE/Semester 10/Data Science/Dataset_MIdterm_sectoin(D).xlsx")
> heart_data
# A tibble: 151 × 7
     Age Gender BloodPressure Cholesterol Heart_Rate QuantumPatternFeature HeartDisease
   <dbl>  <dbl> <chr>               <dbl> <chr>                      <dbl>         <dbl>
 1    68      1 105                   191 High                        8.36             1
 2    58      0 97                    249 Low                         9.25             0
 3    44      0 93                    190 Low                         7.94             1
 4    72      1 93                    183 High                        6.50             1
 5    37      0 145                   166 High                        7.65             1
 6    50      1 114                   271 Low                         8.63             0
 7    68     NA 156                   225 Low                         7.56             1
 8    NA      0 156                   236 Low                         9.15             0
 9    52      0 NA                    266 High                        9.15             0
10    40      1 121                   255 Low                         9.68             0
# ℹ 141 more rows
# ℹ Use `print(n = ...)` to see more rows
> str(heart_data)
tibble [151 × 7] (S3: tbl_df/tbl/data.frame)
 $ Age                 : num [1:151] 68 58 44 72 37 50 68 NA 52 40 ...
 $ Gender              : num [1:151] 1 0 0 1 0 1 NA 0 0 1 ...
 $ BloodPressure       : chr [1:151] "105" "97" "93" "93" ...
 $ Cholesterol         : num [1:151] 191 249 190 183 166 271 225 236 266 255 ...
 $ Heart_Rate          : chr [1:151] "High" "Low" "Low" "High" ...
 $ QuantumPatternFeature: num [1:151] 8.36 9.25 7.94 6.5 7.65 ...
 $ HeartDisease        : num [1:151] 1 0 1 1 1 0 1 0 0 0 ...
```

```
> head(heart_data)
# A tibble: 6 × 7
    Age Gender BloodPressure Cholesterol Heart_Rate QuantumPatternFeature HeartDisease
  <dbl>  <dbl> <chr>               <dbl> <chr>                      <dbl>         <dbl>
1    68      1 105                   191 High                        8.36             1
2    58      0 97                    249 Low                         9.25             0
3    44      0 93                    190 Low                         7.94             1
4    72      1 93                    183 High                        6.50             1
5    37      0 145                   166 High                        7.65             1
6    50      1 114                   271 Low                         8.63             0
> summary(heart_data)
      Age             Gender       BloodPressure       Cholesterol     Heart_Rate
 Min.   :-65.00   Min.   :0.0000   Length:151         Min.   :152.0   Length:151
 1st Qu.: 41.00   1st Qu.:0.0000   Class :character   1st Qu.:183.0   Class :character
 Median : 53.50   Median :0.0000   Mode  :character   Median :214.0   Mode  :character
```

This code reads the Excel file using the read_excel() function and stores it in the variable heart_data. It then displays the dataset by calling heart_data, shows the structure of the dataset using str(), displays the first six rows with head(), and provides a statistical summary using summary().

3. Code:

```
is.na(heart_data)
colSums(is.na(heart_data))
missing_rows<-list(Age= which(is.na(heart_data$Age)),
Gender = which(is.na(heart_data$Gender)))
print(missing_rows)
total_missing <- sum(is.na(heart_data))
cat("Total Missing: ", total_missing, "\n")
```

Output:

```
> colSums(is.na(heart_data))
                  Age                Gender         BloodPressure           Cholesterol
                    3                     3                     3                     0
           Heart_Rate QuantumPatternFeature          HeartDisease
                    3                     0                     0
> missing_rows <- list(Age = which(is.na(heart_data$Age)),
+ Gender = which(is.na(heart_data$Gender)))
> print(missing_rows)
$Age
[1]  8 35 53

$Gender
[1]  7 17 50

> total_missing <- sum(is.na(heart_data))
> cat("Total Missing: ", total_missing, "\n")
Total Missing:  12
>
```

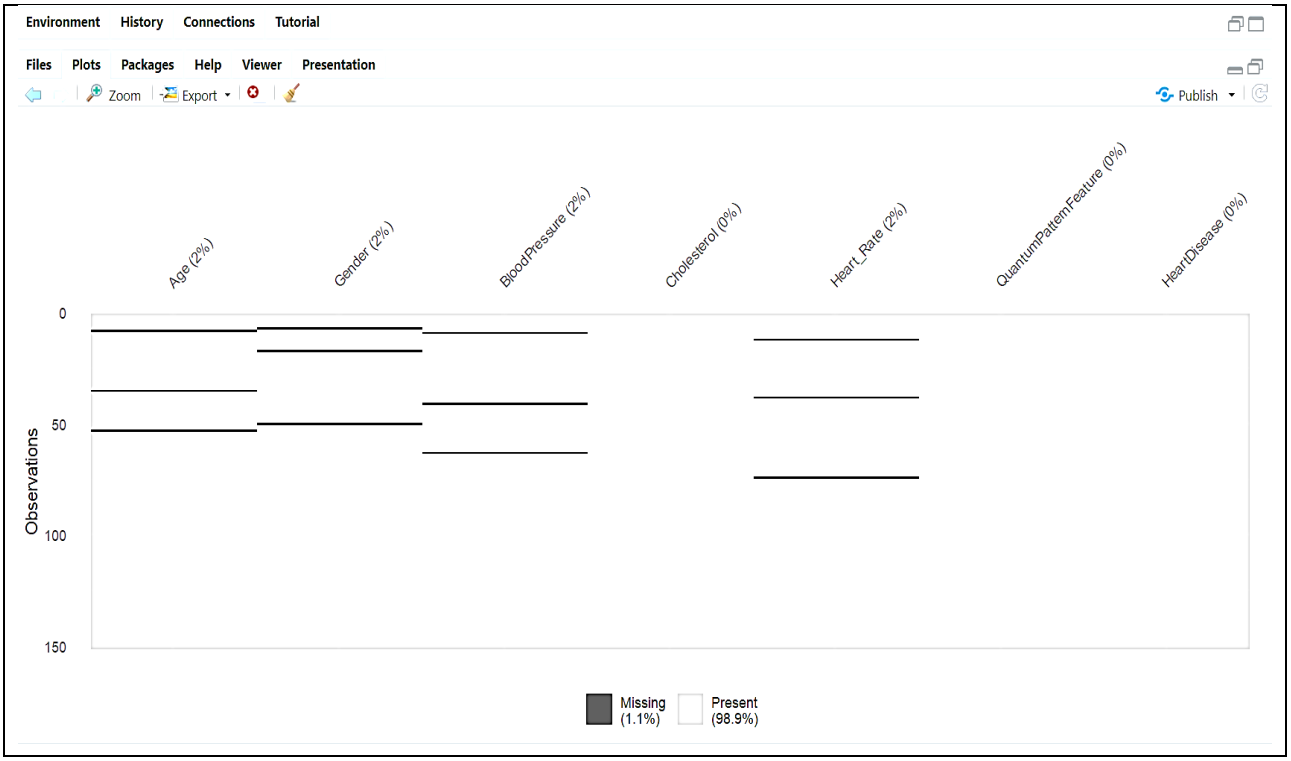
The code checks for missing values in the dataset using is.na(), calculates the total number of missing values in each column using colSums(), identifies the rows with missing values in Age and Gender columns, and prints them. It also calculates and displays the total number of missing values in the entire dataset.

4. Code:

```
gg_miss_var(heart_data)
vis_miss(heart_data)
```

Output:

```
> gg_miss_var(heart_data)
> vis_miss(heart_data)
```



The code visually shows missing data patterns in the dataset.
gg_miss_var() creates a bar plot showing the number of missing values for each variable.
vis_miss() provides a detailed visualization, displaying where missing and non-missing values occur across the dataset.

5. Code:
heart_data$Age[is.na(heart_data$Age)] <- median(heart_data$Age, na.rm = TRUE) heart_data$Gender[is.na(heart_data$Gender)] <- mfv(heart_data$Gender, na_rm = TRUE) colSums(is.na(heart_data)) heart_data$BloodPressure <- as.numeric(heart_data$BloodPressure) str(heart_data$BloodPressure)
Output:

```
> gg_miss_var(heart_data)
> vis_miss(heart_data)
> heart_data$Age[is.na(heart_data$Age)] <- median(heart_data$Age, na.rm = TRUE)
>
> heart_data$Gender[is.na(heart_data$Gender)] <-  mfv(heart_data$Gender, na_rm = TRUE)
> colSums(is.na(heart_data))
                 Age               Gender         BloodPressure           Cholesterol
                   0                    0                     3                     0
          Heart_Rate QuantumPatternFeature          HeartDisease
                   3                    0                     0
>
> heart_data$BloodPressure <- as.numeric(heart_data$BloodPressure)
Warning message:
NAs introduced by coercion
> str(heart_data$BloodPressure)
 num [1:151] 105 97 93 93 145 114 156 156 NA 121 ...
>
```

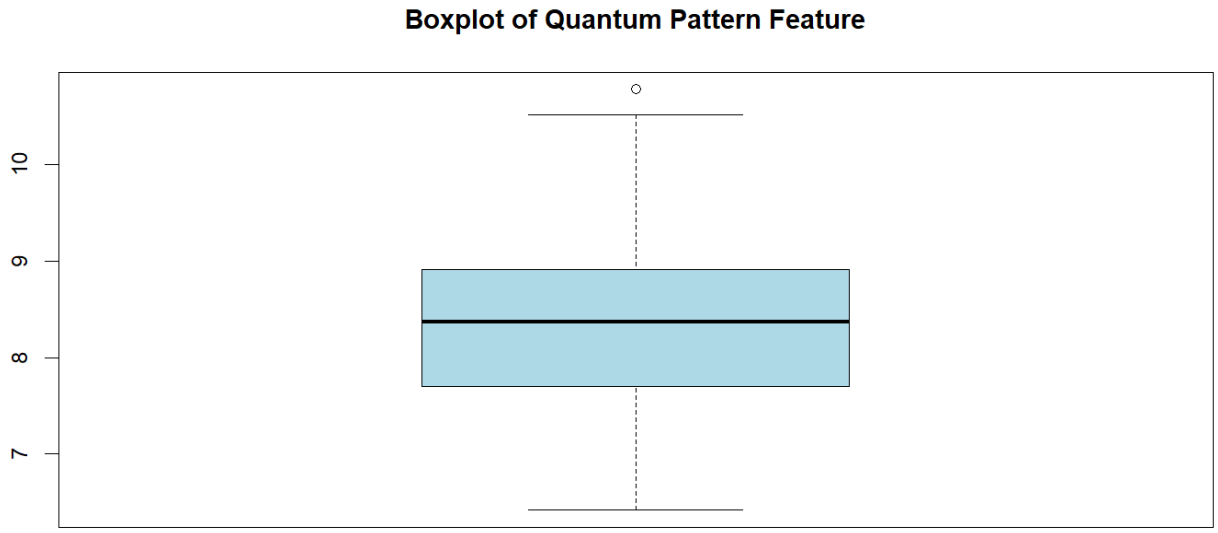
This code fills out the missing values and adjusts data types for proper analysis.

- The missing values in Age are replaced with the median of the Age column.
- The missing values in Gender are replaced with the mode (most frequent value) of the Gender column.
- colSums(is.na()) checks if any missing values remain.

- The BloodPressure column is converted to numeric type to ensure correct data format.
- str() is used to confirm the data type of the BloodPressure column.

6. Code:
boxplot(heart_data$Age, main = "Boxplot of Age", col = "lightblue") boxplot(heart_data$Cholesterol, main = "Boxplot of Cholesterol", col = "lightblue") boxplot(heart_data$QuantumPatternFeature, main = "Boxplot of Quantum Pattern Feature", col = "lightblue")
Output:

```
> boxplot(heart_data$Age, main = "Boxplot of Age", col = "lightblue")
>
> boxplot(heart_data$Cholesterol, main = "Boxplot of Cholesterol", col = "lightblue")
> boxplot(heart_data$QuantumPatternFeature, main = "Boxplot of Quantum Pattern Feature", col = "lightblue")
>
```



This code creates boxplots to show the distribution and find outliers for Age, Cholesterol, and QuantumPatternFeature. It helps quickly spot the spread and any unusual values in each variable.

7. Code:
Q1 <- quantile(heart_data$Age, 0.25) Q3 <- quantile(heart_data$Age, 0.75) IQR <- Q3 - Q1
Output:

```
> Q1 <- quantile(heart_data$Age, 0.25)
> Q3 <- quantile(heart_data$Age, 0.75)
> IQR <- Q3 - Q1
>
```

This code calculates the first quartile (Q1), third quartile (Q3), and interquartile range (IQR) for the Age variable. These values are used to understand the spread and detect outliers.

8. Code:

```
Q1 <- quantile(heart_data$Age, 0.25)
Q3 <- quantile(heart_data$Age, 0.75)
IQR <- Q3 - Q1
lower <- Q1 - 1.5 * IQR
upper <- Q3 + 1.5 * IQR
outliers_age <- which(heart_data$Age < lower | heart_data$Age > upper)
heart_data$Age[outliers_age] <- median(heart_data$Age)
heart_data[outliers_age, "Age"]
heart_data[outliers_age, ]
summary(heart_data["Age"])
```

Output:

```
> Q1 <- quantile(heart_data$Age, 0.25)
> Q3 <- quantile(heart_data$Age, 0.75)
> IQR <- Q3 - Q1
>
> lower <- Q1 - 1.5 * IQR
> upper <- Q3 + 1.5 * IQR
>
> outliers_age <- which(heart_data$Age < lower | heart_data$Age > upper)
> heart_data$Age[outliers_age] <- median(heart_data$Age)
>
> heart_data[outliers_age, "Age"]
# A tibble: 0 × 1
# ℹ 1 variable: Age <dbl>
> heart_data[outliers_age, ]
# A tibble: 0 × 7
# ℹ 7 variables: Age <dbl>, Gender <dbl>, BloodPressure <dbl>, Cholesterol <dbl>, Heart_Rate <chr>,
#   QuantumPatternFeature <dbl>, HeartDisease <dbl>
>
> summary(heart_data["Age"])
      Age
 Min.   :30.00
 1st Qu.:43.00
 Median :53.50
 Mean   :53.92
 3rd Qu.:66.50
 Max.   :79.00
```

This code identifies and replaces outliers in the Age variable. It calculates the lower and upper bounds based on the interquartile range (IQR) and replaces any values outside these bounds with the median value of Age. It then displays the outliers and provides a summary of the modified Age data.

9. Code:

```
heart_data$Gender <- factor(heart_data$Gender,
levels = c(0, 1),
labels = c("Female", "Male"))
heart_data$Gender[is.na(heart_data$Gender)] <-  mfv(heart_data$Gender, na_rm = TRUE)
unique(heart_data$Gender)
```

Output:

```
> heart_data$Gender <- factor(heart_data$Gender,
+                             levels = c(0, 1),
+                             labels = c("Female", "Male"))
> heart_data$Gender[is.na(heart_data$Gender)] <-  mfv(heart_data$Gender, na_rm = TRUE)
> unique(heart_data$Gender)
[1] <NA>
Levels: Female Male
>
```

This code modifies the Gender column by converting its values from numeric (0 and 1) to categorical labels ("Female" and "Male"). It then fills missing values (NA) in the Gender column using the mfv() function, which finds the most frequent value (mode). Finally, it displays the unique values in the Gender column with the unique() function.

10. Code

```
summary(heart_data$Cholesterol)
heart_data$Cholesterol_level <- cut(
  heart_data$Cholesterol,
  breaks = c(0, 199, 239, max(heart_data$Cholesterol, na.rm = TRUE)),
  labels = c("Low", "Borderline", "High"),
  right = TRUE
)
summary(heart_data$Cholesterol_level)
```

Output:

```
> summary(heart_data$Cholesterol)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  152.0   183.0   214.0   220.8   258.0   299.0
>
> heart_data$Cholesterol_level <- cut(
+   heart_data$Cholesterol,
+   breaks = c(0, 199, 239, max(heart_data$Cholesterol, na.rm = TRUE)),
+   labels = c("Low", "Borderline", "High"),
+   right = TRUE
+ )
> summary(heart_data$Cholesterol_level)
       Low Borderline       High
        60         34         57
```

This code provides a summary of the Cholesterol column and then categorizes the cholesterol levels into three groups: Low, Borderline, and High. It uses the cut() function to create these categories based on specified cholesterol ranges. After that, the code displays a summary of the newly created Cholesterol_level column.

11. Code:

```
colnames(heart_data)
View(heart_data)
head(heart_data)
```

Output

```
> colnames(heart_data)
[1] "Age"                  "Gender"               "BloodPressure"
[4] "Cholesterol"          "Heart_Rate"           "QuantumPatternFeature"
[7] "HeartDisease"         "Cholesterol_level"
> View(heart_data) # Opens the spreadsheet-like viewer in RStudio
> head(heart_data)  # Shows first 6 rows of all columns
# A tibble: 6 × 8
    Age Gender BloodPressure Cholesterol Heart_Rate QuantumPatternFeature HeartDisease
  <dbl> <fct>          <dbl>       <dbl> <chr>                      <dbl>        <dbl>
1    68 NA               105         191 High                        8.36            1
2    58 NA                97         249 Low                         9.25            0
3    44 NA                93         190 Low                         7.94            1
4    72 NA                93         183 High                        6.50            1
5    37 NA               145         166 High                        7.65            1
6    50 NA               114         271 Low                         8.63            0
# i 1 more variable: Cholesterol_level <fct>
```

This code retrieves the column names of the heart_data dataset using colnames(). It also opens the data in a spreadsheet-like format within RStudio using View(), and displays the first six rows of the dataset with head() to give a quick overview of the data.

12. Code	Output
table(heart_data$HeartDisease)	> table(heart_data$HeartDisease) No Yes 90 90

It displays the count of patients with and without heart disease in the dataset.

13. Code	Output
majority_class <- heart_data %>% filter(HeartDisease == names(which.max(table(HeartDisease)))) minority_class <- heart_data %>% filter(HeartDisease == names(which.min(table(HeartDisease))))	> majority_class <- heart_data %>% filter(HeartDisease == names(which.max(table(HeartDisease)))) > minority_class <- heart_data %>% filter(HeartDisease == names(which.min(table(HeartDisease)))) >

These lines separate the dataset into majority and minority classes based on heart disease presence for balancing purposes.

14. Code	Output
majority_class_undersampled <- majority_class %>% sample_n(nrow(minority_class)) balanced_data_under <- bind_rows(minority_class, majority_class_undersampled) balanced_data_under <- balanced_data_under %>% sample_frac(1) table(balanced_data_under$HeartDisease)	> majority_class_undersampled <- majority_class %>% sample_n(nrow(minority_class)) > balanced_data_under <- bind_rows(minority_class, majority_class_undersampled) > balanced_data_under <- balanced_data_under %>% sample_frac(1) > table(balanced_data_under$HeartDisease) No Yes 180 0 >

This code performs undersampling to balance the dataset by randomly reducing the majority class to match the size of the minority class.

15. Code	Output
minority_class_oversampled <- minority_class %>% sample_n(nrow(majority_class), replace = TRUE) balanced_data_over <- bind_rows(majority_class, minority_class_oversampled) balanced_data_over <- balanced_data_over %>% sample_frac(1) table(balanced_data_over$HeartDisease)	> minority_class_oversampled <- minority_class %>% + sample_n(nrow(majority_class), replace = TRUE) > balanced_data_over <- bind_rows(majority_class, minority_class_oversampled) > balanced_data_over <- balanced_data_over %>% sample_frac(1) > table(balanced_data_over$HeartDisease) No Yes 180 0 >

This code performs oversampling by duplicating the minority class to match the majority class size, creating a balanced dataset.

16. Code	Output
age_gender_summary <- heart_data %>% group_by(Gender) %>% summarise(Mean_Age = mean(Age, na.rm = TRUE), Median_Age = median(Age, na.rm = TRUE), Mode_Age = mfv(Age, na_rm = TRUE)[1]) age_gender_summary	> age_gender_summary # A tibble: 2 × 4 Gender Mean_Age Median_Age Mode_Age <fct> <dbl> <dbl> <dbl> 1 Female 0.454 0.449 0.265 2 Male 0.394 0.429 0.204 >

This code calculates the mean, median, and mode of age separately for each gender group.

17. Code	Output
age_hr_summary <- heart_data %>% group_by(Heart_Rate) %>% summarise(Mean_Age = mean(Age, na.rm = TRUE), Median_Age = median(Age, na.rm = TRUE), Mode_Age = mfv(Age, na_rm = TRUE)[1]) age_hr_summary	> age_hr_summary # A tibble: 2 × 4 Heart_Rate Mean_Age Median_Age Mode_Age <fct> <dbl> <dbl> <dbl> 1 High 0.412 0.429 0.429 2 Low 0.445 0.469 0.510 >

This code summarizes the mean, median, and mode of age for each heart rate category.

18. Code	Output
age_gender_spread <- heart_data %>% group_by(Gender) %>% summarise(Min_Age = min(Age), Max_Age = max(Age), Range_Age = paste0(min(Age), "–", max(Age)), IQR_Age = IQR(Age), Variance_Age = var(Age), SD_Age = sd(Age)) age_gender_spread	> age_gender_spread # A tibble: 2 x 7 Gender Min_Age Max_Age Range_Age IQR_Age Variance_Age SD_Age <fct> <dbl> <dbl> <chr> <dbl> <dbl> <dbl> 1 Female 0.122 0.959 0.122448979591837-0.959183673469… 0.255 0.0585 0.242 2 Male 0.0204 0.653 0.0204081632653061-0.65306122448… 0.163 0.0201 0.142 >

This code computes age spread statistics (min, max, range, IQR, variance, and standard deviation) for each gender group.

19.Code:

min_max <- function(x) {
(x - min(x)) / (max(x) - min(x))
}

Output:
> min_max <- function(x) { + (x - min(x)) / (max(x) - min(x)) + } >

This code defines a function min_max() that scales a numeric variable to a 0–1 range using min-max normalization.

20.Code:
heart_data$Age <- min_max(heart_data$Age) heart_data$Cholesterol <- min_max(heart_data$Cholesterol) heart_data$BloodPressure <- min_max(heart_data$BloodPressure) heart_data$QuantumPatternFeature <- min_max(heart_data$QuantumPatternFeature)
Output:
heart_data$Age <- min_max(heart_data$Age) heart_data$Cholesterol <- min_max(heart_data$Cholesterol) heart_data$BloodPressure <- min_max(heart_data$BloodPressure) heart_data$QuantumPatternFeature <- min_max(heart_data$QuantumPatternFeature)

This code applies min_max() to rescale Age, Cholesterol, BloodPressure, and QuantumPatternFeature columns between 0 and 1.

21.Code:
summary(heart_data)
Output:

```
> summary(heart_data)
      Age            Gender   BloodPressure     Cholesterol     Heart_Rate QuantumPatternFeature
 Min.   :0.0000   Female:92   Min.   :0.0000   Min.   :0.0000   High: 66   Min.   :0.0000
 1st Qu.:0.2551   Male  :88   1st Qu.:0.2444   1st Qu.:0.2449   Low :114   1st Qu.:0.3556
 Median :0.4694               Median :0.5169   Median :0.5340              Median :0.5336
 Mean   :0.4747               Mean   :0.4992   Mean   :0.5074              Mean   :0.5040
 3rd Qu.:0.7194               3rd Qu.:0.7528   3rd Qu.:0.7347              3rd Qu.:0.6377
 Max.   :1.0000               Max.   :1.0000   Max.   :1.0000              Max.   :1.0000
 HeartDisease  Cholesterol_level
 No :90        Low       :59
 Yes:90        Borderline:38
               High      :83
```

This code uses summary(heart_data) to quickly inspect the basic statistics of the dataset after normalization (like Min, Max, Mean).

22. Code:
train_index <- sample(1:nrow(heart_data), size = 0.8 * nrow(heart_data)) train_data <- heart_data[train_index,] test_data <- heart_data[-train_index,]
Output:

```
> train_index <- sample(1:nrow(heart_data), size = 0.8 * nrow(heart_data))
> train_data <- heart_data[train_index, ]
> test_data <- heart_data[-train_index, ]
>
```

This code randomly selects 80% of the rows for train_data and the remaining 20% for test_data to prepare for model training and evaluation.

23.Code:
dim(train_data) dim(test_data)
Output:

```
> dim(train_data)
[1] 144   8
> dim(test_data)
[1] 36  8
>
```

This displays the number of rows and columns in the training and testing datasets to confirm the split was done correctly.