1.CLUSTERING ANALYSIS

```
library(ggplot2)
customer_data <- data.frame(
  CustomerID = 1:5,
  Gender = c("Male", "Male", "Female", "Female", "Female"),
  Age = c(19, 21, 20, 23, 31),
  AnnualIncome = c(15, 15, 16, 16, 17),
  SpendingScore = c(39, 81, 6, 77, 40)
)
print(customer_data)
data_for_clustering <- customer_data[, c("AnnualIncome", "SpendingScore")]
n <- nrow(data_for_clustering)
k <- min(2, n)
set.seed(123)
kmeans_result <- kmeans(data_for_clustering, centers = k, nstart = 25)
customer_data$Cluster <- as.factor(kmeans_result$cluster)
ggplot(customer_data, aes(x = AnnualIncome, y = SpendingScore, color = Cluster)) +
  geom_point(size = 4) +
  scale_color_manual(values = c("red", "blue")) +
  labs(title = "Customer Segments Based on Annual Income and Spending Score",
       x = "Annual Income (k$)",
       y = "Spending Score (1-100)") +
  theme_minimal()
```

4.PERSON TOTAL COUNT

```
people_data <- data.frame(
  Person = c("Gopu", "Babu", "Baby", "Gopal", "Krishna", "Jai", "Dev", "Malini", "Hema", "Anu"),
  Vegetarian = c("yes", "yes", "yes", "no", "yes", "no", "no", "yes", "yes", "yes"))
veg_count <- sum(people_data$Vegetarian == "yes")
non_veg_count <- sum(people_data$Vegetarian == "no")
cat("Number of Vegetarians:", veg_count, "\n")
cat("Number of Non-Vegetarians:", non_veg_count, "\n")
if (veg_count > non_veg_count) {
  cat("The number of vegetarians is greater.\n")
} else if (veg_count < non_veg_count) {
  cat("The number of non-vegetarians is greater.\n")
} else {
  cat("The number of vegetarians and non-vegetarians is equal.\n")
}
```

5. SCATTER PLOT FOR MOBILE PHONE

```
x <- c(4, 1, 5, 7, 10, 2, 50, 25, 90, 36)
y <- c(12, 5, 13, 19, 31, 7, 153, 72, 275, 110)
plot(x, y, main = "Scatter Plot of Mobile Phones Sold vs Money",
     xlab = "Number of Mobile Phones Sold", ylab = "Money",
     pch = 19, col = "blue")
abline(lm(y ~ x), col = "red", lwd = 2)
```

6. TRANSACTION FOR PURCHASED

```
if(!require(arules)) {
  install.packages("arules")
  library(arules)
}
transactions_list <- list(
  c("Bread", "Cheese", "Egg", "Juice"),
  c("Bread", "Cheese", "Juice"),
  c("Bread", "Milk", "Yogurt"),
  c("Bread", "Juice", "Milk"),
  c("Cheese", "Juice", "Milk"))
transactions <- as(transactions_list, "transactions")
inspect(transactions)
rules <- apriori(transactions, parameter = list(supp = 0.5, conf = 0.75))
inspect(rules)
```

8. HISTOGRAM FOR STUDENT MARKS

```
marks <- c(55, 60, 71, 63, 55, 65, 50, 55, 58, 59, 61, 63, 65, 67, 71, 72, 75)
if(!require(cluster)) {
  install.packages("cluster")
  library(cluster)
}
num_bins <- 3
bin_size <- ceiling(length(marks) / num_bins)
sorted_marks <- sort(marks)
equal_freq_bins <- split(sorted_marks, ceiling(seq_along(sorted_marks) / bin_size))
cat("\nEqual-Frequency Partitioning Bins:\n")
print(equal_freq_bins)
range_marks <- range(marks)
bin_width <- ceiling((range_marks[2] - range_marks[1]) / num_bins)
equal_width_bins <- cut(marks, breaks = seq(range_marks[1], range_marks[2] + bin_width, by = bin_width), include.lowest = TRUE)
cat("\nEqual-Width Partitioning Bins:\n")
print(table(equal_width_bins))
set.seed(123) # For reproducibility
kmeans_result <- kmeans(marks, centers = num_bins, nstart = 20)
clustering_bins <- kmeans_result$cluster
cat("\nClustering Results:\n")
print(clustering_bins)
pdf("plots.pdf", width = 8, height = 12)  # Adjust the size as needed
par(mfrow = c(3, 1))
hist(marks, breaks = length(marks), main = "Histogram: Equal-Frequency Partitioning",
     xlab = "Marks", col = "lightblue", border = "black")
abline(v = unlist(lapply(equal_freq_bins, max)), col = "red", lwd = 2, lty = 2)
hist(marks, breaks = seq(range_marks[1], range_marks[2] + bin_width, by = bin_width),
     main = "Histogram: Equal-Width Partitioning", xlab = "Marks", col = "lightgreen", border = "black")
abline(v = seq(range_marks[1], range_marks[2] + bin_width, by = bin_width), col = "blue", lwd = 2, lty = 2)
hist(marks, breaks = length(marks), main = "Histogram: Clustering",
     xlab = "Marks", col = "lightpink", border = "black")
abline(v = tapply(marks, clustering_bins, mean), col = "purple", lwd = 2, lty = 2)
par(mfrow = c(1, 1))
dev.off()
plot(1:10, main = "Test Plot")
```

11.MIN-MAX,Z-SCORE

```
strike_rates <- c(100, 70, 60, 90, 90)
min_value <- 0
max_value <- 1
data_min <- min(strike_rates)
data_max <- max(strike_rates)
min_max_normalized <- (strike_rates - data_min) / (data_max - data_min) * (max_value - min_value) + min_value
cat("Min-Max Normalization:\n")
print(min_max_normalized)
mean_value <- mean(strike_rates)
std_dev <- sd(strike_rates)
z_score_normalized <- (strike_rates - mean_value) / std_dev
cat("\nZ-Score Normalization:\n")
print(z_score_normalized)
mad <- mean(abs(strike_rates - mean_value))
z_score_mad_normalized <- (strike_rates - mean_value) / mad
cat("\nZ-Score Normalization using MAD:\n")
print(z_score_mad_normalized)
j <- ceiling(log10(max(abs(strike_rates)))) # Calculate j for decimal scaling
decimal_scaled <- strike_rates / (10^j)
cat("\nDecimal Scaling Normalization:\n")
print(decimal_scaled)
```

12.SD &VARIANCE OF AVG SPEED

```
AvgSpeed <- c(78, 81, 82, 74, 83, 82, 77, 80)
TotalTime <- c(39, 37, 36, 42, 35, 36, 40, 70, 38, 46)
std_dev_avg_speed <- sd(AvgSpeed)
std_dev_total_time <- sd(TotalTime)
cat("Standard Deviation:\n")
cat("AvgSpeed:", std_dev_avg_speed, "\n")
cat("TotalTime:", std_dev_total_time, "\n")
variance_avg_speed <- var(AvgSpeed)
variance_total_time <- var(TotalTime)
cat("\nVariance:\n")
cat("AvgSpeed:", variance_avg_speed, "\n")
cat("TotalTime:", variance_total_time, "\n")
```

16.COVARIENCE & CORRELATION

```
photograph_A <- c(18, 2, 20)
photograph_B <- c(22, 28, 10)
photograph_C <- c(20, 40, 40)
preferences <- data.frame(A = photograph_A, B = photograph_B, C = photograph_C)
cov_BC <- cov(preferences$B, preferences$C)
cat("Sample Covariance between B and C:", cov_BC, "\n")
cov_matrix <- cov(preferences)
cat("Sample Covariance Matrix:\n")
print(cov_matrix)
cor_BC <- cor(preferences$B, preferences$C)
cat("Sample Correlation between B and C:", cor_BC, "\n")
cor_matrix <- cor(preferences)
cat("Sample Correlation Matrix:\n")
print(cor_matrix)
```

17.HISTOGRAM FOR FD

```
prices <- c(1, 1, 5, 5, 5, 5, 5, 8, 8, 10, 10, 10, 12, 14, 14,
            14, 15, 15, 15, 15, 15, 15, 18, 18, 18, 18, 18, 18, 18,
            18, 20, 20, 20, 20, 20, 20, 20, 21, 21, 21, 21, 25,
            25, 25, 25, 25, 28, 28, 30, 30, 30)
num_bins <- 3
bin_size <- ceiling(length(prices) / num_bins)
sorted_prices <- sort(prices)
equal_freq_bins <- split(sorted_prices, ceiling(seq_along(sorted_prices) / bin_size))
cat("Equal-Frequency Partitioning Bins:\n")
print(equal_freq_bins)
bin_means <- sapply(equal_freq_bins, mean)
bin_boundaries <- lapply(equal_freq_bins, range)
cat("\nBin Means:\n")
print(bin_means)
cat("\nBin Boundaries:\n")
print(bin_boundaries)
breaks <- seq(min(prices) - 1, max(prices) + 1, by = 5)
hist(prices, breaks = breaks,
     main = "Histogram of Prices",
     xlab = "Price",
     ylab = "Frequency",
     col = "lightblue",
     border = "black")
abline(v = bin_means, col = "red", lwd = 2, lty = 2)
```

18.MEAN,MEDIAN &BOX PLOT

```
class_A <- c(76, 35, 47, 64, 95, 66, 89, 36, 84)
class_B <- c(51, 56, 84, 60, 59, 70, 63, 66, 50)
mean_A <- mean(class_A)
median_A <- median(class_A)
range_A <- range(class_A)
mean_B <- mean(class_B)
median_B <- median(class_B)
range_B <- range(class_B)
cat("Class A:\n")
cat("Mean:", mean_A, "\n")
cat("Median:", median_A, "\n")
cat("Range:", range_A[1], "to", range_A[2], "\n")
cat("\nClass B:\n")
cat("Mean:", mean_B, "\n")
cat("Median:", median_B, "\n")
cat("Range:", range_B[1], "to", range_B[2], "\n")
boxplot(class_A, class_B,
        names = c("Class A", "Class B"),
        main = "Boxplot of Class A and Class B Exam Scores",
        ylab = "Scores",
        col = c("lightblue", "lightgreen"))
grid()
```

19.MIN,MAX & Z-SCORE

```
data <- c(200, 300, 400, 600, 1000)
min_val <- min(data)
max_val <- max(data)
min_max_normalized <- (data - min_val) / (max_val - min_val)
cat("Min-Max Normalized Values:\n")
print(min_max_normalized)
mean_val <- mean(data)
sd_val <- sd(data)
z_score_normalized <- (data - mean_val) / sd_val
cat("\nZ-Score Normalized Values:\n")
print(z_score_normalized)
```

20.MPG &QSEC

```
 data("AirPassengers")
data("mtcars")
hist(AirPassengers,
     breaks = seq(100, 700, by = 150),
     main = "Histogram of AirPassengers",
     xlab = "Number of Passengers",
     col = "lightblue",
     xlim = c(100, 700),
     border = "black")
plot(mtcars$mpg, type = "l", col = "blue", lwd = 2, ylim = c(10, 30),
     ylab = "Values", xlab = "Index", main = "Line Chart of mpg and qsec")
lines(mtcars$qsec, col = "red", lwd = 2)
legend("topright", legend = c("mpg", "qsec"), col = c("blue", "red"), lty = 1, lwd = 2)
```

21.MORTALITY

```
set.seed(123)
hardness <- seq(0, 100, by = 10)
mortality <- 5 + 0.4 * hardness + rnorm(length(hardness), mean = 0, sd = 2) # Adding noise
water <- data.frame(hardness, mortality)
str(water)
plot(water$hardness, water$mortality,
     main = "Mortality vs Hardness",
     xlab = "Hardness",
     ylab = "Mortality",
     col = "blue",
     pch = 19)
linear_model <- lm(mortality ~ hardness, data = water)
summary(linear_model)
hardness_value <- 88
predicted_mortality <- predict(linear_model, newdata = data.frame(hardness = hardness_value))
cat("Predicted Mortality for Hardness =", hardness_value, "is:", predicted_mortality, "\n")
```

22. MPG,CYL,MTCARS

```
library(ggplot2)
data(mtcars)
boxplot(mpg ~ as.factor(cyl),
        data = mtcars,
        main = "Boxplot of MPG by Number of Cylinders",
        xlab = "Number of Cylinders",
        ylab = "Miles per Gallon (MPG)",
        col = "lightblue",
        border = "black")
grid()
```

23.TENNIS BOX PLOT

```
scores <- c(20, 22, 23, 24, 25, 24, 26, 27, 30, 31, 40, 42, 43, 45, 50)
boxplot(scores,
        main = "Boxplot of Tennis Players' Scores",
        ylab = "Scores",
        col = "lightgreen",
        border = "black")
grid()
outliers <- boxplot.stats(scores)$out
points(rep(1, length(outliers)), outliers, col = "red", pch = 19)
```

24.SCATTER PLOT &BAR CHART

```
library(ggplot2)
library(dplyr)
set.seed(123)
n <- 100
age <- sample(20:80, n, replace = TRUE)
blood_pressure <- rnorm(n, mean = 70 + (age - 20) * 0.5, sd = 10)
diabetes_data <- data.frame(Age = age, BloodPressure = blood_pressure)
ggplot(diabetes_data, aes(x = Age, y = BloodPressure)) +
  geom_point(color = "blue", alpha = 0.6) +
  labs(title = "Blood Pressure vs Age",
       x = "Age",
       y = "Blood Pressure") +
  theme_minimal()
diabetes_data <- diabetes_data %>%
  mutate(AgeGroup = case_when(
    Age < 30 ~ "Under 30",
    Age >= 30 & Age < 40 ~ "30-39",
    Age >= 40 & Age < 50 ~ "40-49",
    Age >= 50 & Age < 60 ~ "50-59",
    Age >= 60 & Age < 70 ~ "60-69",
    Age >= 70 ~ "70 and above"
  ))
age_bp_summary <- diabetes_data %>%
  group_by(AgeGroup) %>% summarise(AverageBloodPressure = mean(BloodPressure, na.rm = TRUE))
ggplot(age_bp_summary, aes(x = AgeGroup, y = AverageBloodPressure, fill = AgeGroup)) +
  geom_bar(stat = "identity") +
  labs(title = "Average Blood Pressure by Age Group",
       x = "Age Group",
       y = "Average Blood Pressure") +
  theme_minimal() +
  theme(legend.position = "none")  # Hide the legend
```

25.

```
library(ggplot2)
data("water", package = "datasets")
str(water)
ggplot(water, aes(x = hardness, y = mortality)) +
  geom_point(color = "blue") +
  labs(title = "Scatter Plot of Mortality vs Hardness",
       x = "Hardness (mg/L)",
       y = "Mortality Rate") +
  theme_minimal()
model <- lm(mortality ~ hardness, data = water)
summary(model)
new_data <- data.frame(hardness = 88)
predicted_mortality <- predict(model, newdata = new_data)
cat("Predicted mortality for hardness = 88:", predicted_mortality, "\n")
```