DAY 1

Question 1:

```r
#Reg No : 192224215
age <- c(5,15,20,50,80,110)
frequency <- c(200,450,300,1500,700,44)
median(age)
median(frequency)
```

Output:

```
> #Reg No : 192224215
> age <- c(5,15,20,50,80,110)
> frequency <- c(200,450,300,1500,700,44)
> median(age)
[1] 35
> median(frequency)
[1] 375
>
```

Question 2:

```r
#Reg No : 192224215
age <- c(13,15,16,16,19,20,20,21,22,22,25,25,25,25,30,33,33,35,35,35,35,36,40,45,46,52,70)
mean(age)
median(age)
mode_age <- names(table(age))[table(age)==max(table(age))]
mode_age
range(age)
quantile(age,.25)
quantile(age,.75)
```

Output:

```
> age <- c(13,15,16,16,19,20,20,21,22,22,25,25,25,25,30,33,33,35,35,35,35,36,40,45,46,52,70)
> mean(age)
[1] 29.96296
> median(age)
[1] 25
> mode_age <- names(table(age))[table(age)==max(table(age))]
> mode_age
[1] "25" "35"
> range(age)
[1] 13 70
> quantile(age,.25)
 25%
20.5
> quantile(age,.75)
75%
 35
```

Question 3:

```r
#Reg No : 192224215
data <- c(200, 300, 400, 600, 1000)

min_max_norm <- function(x) {
  (x - min(x)) / (max(x) - min(x))
}
min_max_normalized <- min_max_norm(data)

z_score_norm <- function(x) {
  (x - mean(x)) / sd(x)
}
z_score_normalized <- z_score_norm(data)

cat("Original data:", data, "\n")
cat("Min-max normalized data:", min_max_normalized, "\n")
cat("Z-score normalized data:", z_score_normalized, "\n")
```

Output:

```
> data <- c(200, 300, 400, 600, 1000)
>
> min_max_norm <- function(x) {
+   (x - min(x)) / (max(x) - min(x))
+ }
> min_max_normalized <- min_max_norm(data)
>
> z_score_norm <- function(x) {
+   (x - mean(x)) / sd(x)
+ }
> z_score_normalized <- z_score_norm(data)
>
> cat("Original data:", data, "\n")
Original data: 200 300 400 600 1000
> cat("Min-max normalized data:", min_max_normalized, "\n")
Min-max normalized data: 0 0.125 0.25 0.5 1
> cat("Z-score normalized data:", z_score_normalized, "\n")
Z-score normalized data: -0.9486833 -0.6324555 -0.3162278 0.3162278 1.581139
```

Question 4:

```r
#Reg No : 192224215
data <- c(11,13,13,15,15,16,19,20,20,20,21,21,22,23,24,30,40,45,45,45,71,72,73,75)
bins <- 5
bin_indices <- cut(data, bins)
mean_smooth <- tapply(data, bin_indices, mean)
print(mean_smooth)
median_smooth <- tapply(data, bin_indices, median)
median_smooth
min_max_smooth <- tapply(data, bin_indices, function(x) c(min(x), max(x)))
print(min_max_smooth)
```

Output:

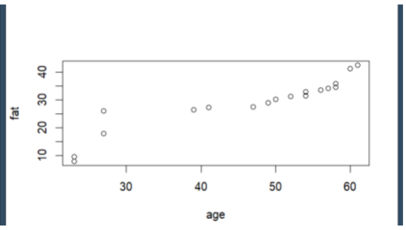
```
> #Reg No : 192224215
> data <- c(11,13,13,15,15,16,19,20,20,20,21,21,22,23,24,30,40,45,45,45,71,72,73,75)
> bins <- 5
> bin_indices <- cut(data, bins)
> mean_smooth <- tapply(data, bin_indices, mean)
> print(mean_smooth)
(10.9,23.8] (23.8,36.6] (36.6,49.4] (49.4,62.2] (62.2,75.1]
   17.78571    27.00000    43.75000          NA    72.75000
> median_smooth <- tapply(data, bin_indices, median)
> median_smooth
(10.9,23.8] (23.8,36.6] (36.6,49.4] (49.4,62.2] (62.2,75.1]
       19.5        27.0        45.0          NA        72.5
> min_max_smooth <- tapply(data, bin_indices, function(x) c(min(x), max(x)))
> print(min_max_smooth)
$`(10.9,23.8]`
[1] 11 23

$`(23.8,36.6]`
[1] 24 30

$`(36.6,49.4]`
[1] 40 45

$`(49.4,62.2]`
NULL

$`(62.2,75.1]`
[1] 71 75
```

Question 5:

```
#Reg No : 192224215
age <- c(23,23,27,27,39,41,47,49,50,52,54,54,56,57,58,58,60,61)
fat <- c(9.5,26.5,7.8,17.8,31.4,25.9,27.4,27.2,31.2,34.6,42.5,28.8,33.4,30.2,34.1,32.9,41.2,35.7)
mean(age)
median(age)
sd(age)
mean(fat)
median(fat)
sd(fat)
boxplot(age,fat)
scatter.smooth(age,fat)
qqplot(age,fat)
```

Output:

```
> #Reg No : 192224215
> age <- c(23,23,27,27,39,41,47,49,50,52,54,54,56,57,58,58,60,61)
> fat <- c(9.5,26.5,7.8,17.8,31.4,25.9,27.4,27.2,31.2,34.6,42.5,28.8,33.4,30.2,34.1,32.9,41.2,35.7)
> mean(age)
[1] 46.44444
> median(age)
[1] 51
> sd(age)
[1] 13.21862
> mean(fat)
[1] 28.78333
> median(fat)
[1] 30.7
> sd(fat)
[1] 9.254395
> boxplot(age,fat)
> scatter.smooth(age,fat)
> qqplot(age,fat)
```



Question 6:

```r
#Reg No : 192224215
v <- c(23,23,27,27,39,41,47,49,50,52,54,54,56,57,58,58,60,61)
min <- 0
max <- 1
min_max <- ((35-min(v))/(max(v)-min(v)))
print(min_max)
m <- mean(v)
s <- 12.94
z_score <- (35-m)/s
print(z_score)
m <- 35
j <- max(m)<1
decimal_scaling <- m/10^j
print(decimal_scaling)
```

Output:

```
> #Reg No : 192224215
> v <- c(23,23,27,27,39,41,47,49,50,52,54,54,56,57,58,58,60,61)
> min <- 0
> max <- 1
> min_max <- ((35-min(v))/(max(v)-min(v)))
> print(min_max)
[1] 0.3157895
> m <- mean(v)
> s <- 12.94
> z_score <- (35-m)/s
> print(z_score)
[1] -0.8844238
> m <- 35
> j <- max(m)<1
> decimal_scaling <- m/10^j
> print(decimal_scaling)
[1] 35
>
```

Question 7:

```r
#Reg No : 192224215
pencils <- c(9,25,23,12,11,6,7,8,9,10)
mean(pencils)
median(pencils)
mode <- names(table(pencils))[table(pencils)==max(table(pencils))]
mode
```

Output:

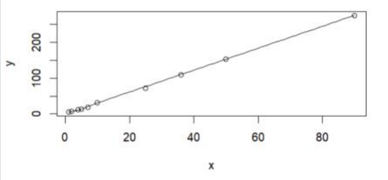
```
> #Reg No : 192224215
> pencils <- c(9,25,23,12,11,6,7,8,9,10)
> mean(pencils)
[1] 12
> median(pencils)
[1] 9.5
> mode <- names(table(pencils))[table(pencils)==max(table(pencils))]
> mode
[1] "9"
```

Question 8:

```r
#Reg No : 192224215
x <- c(4,1,5,7,10,2,50,25,90,36)
y <- c(12,5,13,19,31,7,153,72,275,110)
scatter.smooth(x,y)
```

Output:

```
> #Reg No : 192224215
> x <- c(4,1,5,7,10,2,50,25,90,36)
> y <- c(12,5,13,19,31,7,153,72,275,110)
> scatter.smooth(x,y)
> |
```

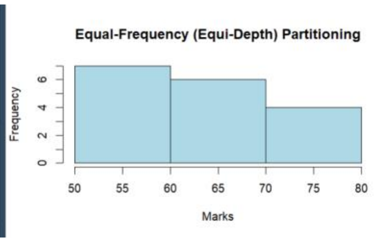


Question 9:

```r
#Reg No : 192224215
marks <- c(55, 60, 71, 63, 55, 65, 50, 55, 58, 59, 61, 63, 65, 67, 71, 72, 75)
num_bins <- 3
bins_eq_frequency <- cut(marks, breaks = num_bins, labels = FALSE)
hist(marks, breaks = num_bins, col = "lightblue", xlab = "Marks", main = "Equal-Frequency (Equi-Depth) Partitioni

bin_mean <- tapply(marks, cut(marks, num_bins), mean)
smoothed_data_by_mean <- unname(bin_mean[as.character(cut(marks, num_bins))])
bin_median <- tapply(marks, cut(marks, num_bins), median)
smoothed_data_by_median <- unname(bin_median[as.character(cut(marks, num_bins))])
bin_boundaries <- tapply(marks, cut(marks, num_bins), function(x) c(min(x), max(x)))
smoothed_data_by_boundaries <- unlist(bin_boundaries[as.character(cut(marks, num_bins))])

print("Original data:")
print(marks)
print("Smoothed data by bin mean:")
print(smoothed_data_by_mean)
print("Smoothed data by bin median:")
print(smoothed_data_by_median)
print("Smoothed data by bin boundaries:")
print(smoothed_data_by_boundaries)
```

Output:

```
> print(smoothed_data_by_mean)
 [1] 54.60000 62.28571 71.20000 62.28571 54.60000 62.28571 54.60000 54.60000 54.60000 62.28571 62.28571 62.28571
[13] 62.28571 71.20000 71.20000 71.20000 71.20000
> print("Smoothed data by bin median:")
[1] "Smoothed data by bin median:"
> print(smoothed_data_by_median)
 [1] 55 63 71 63 55 63 55 55 55 63 63 63 63 71 71 71 71
> print("Smoothed data by bin boundaries:")
[1] "Smoothed data by bin boundaries:"
> print(smoothed_data_by_boundaries)
  (50,58.3]1   (50,58.3]2 (58.3,66.7]1 (58.3,66.7]2   (66.7,75]1   (66.7,75]2 (58.3,66.7]1 (58.3,66.7]2
          50           58           59           65           67           75           59           65
  (50,58.3]1   (50,58.3]2 (58.3,66.7]1 (58.3,66.7]2   (50,58.3]1   (50,58.3]2   (50,58.3]1   (50,58.3]2
          50           58           59           65           50           58           50           58
  (50,58.3]1   (50,58.3]2 (58.3,66.7]1 (58.3,66.7]2 (58.3,66.7]1 (58.3,66.7]2 (58.3,66.7]1 (58.3,66.7]2
          50           58           59           65           59           65           59           65
(58.3,66.7]1 (58.3,66.7]2   (66.7,75]1   (66.7,75]2   (66.7,75]1   (66.7,75]2   (66.7,75]1   (66.7,75]2
          59           65           67           75           67           75           67           75
  (66.7,75]1   (66.7,75]2
          67           75
```



Question 10:

```r
#Reg No : 192224215
v <- c(78.3,81.8,82,74.2,83.4,84.5,82.9,77.5,80.9,70.6)
IQR(v)
sd(v)
```

Output:

```
> #Reg No : 192224215
> v <- c(78.3,81.8,82,74.2,83.4,84.5,82.9,77.5,80.9,70.6)
> IQR(v)
[1] 4.975
> sd(v)
[1] 4.445835
```

Question 11:

```
#Reg No : 192224215
age <- c(13,15,16,16,19,20,20,21,22,22,25,25,25,25,30,33,33,35,35,35,35,36,40,45,46,52,70)
quantile(age,.25)
quantile(age,.75)
```

Output:

```
> #Reg No : 192224215
> age <- c(13,15,16,16,19,20,20,21,22,22,25,25,25,25,30,33,33,35,35,35,35,36,40,45,46,52,70)
> quantile(age,.25)
 25%
20.5
> quantile(age,.75)
75%
 35
```

DAY 2

Question 1 :

```
#Reg No : 192224215
data <- data.frame(
  Age = c("5-6 years", "7-8 years", "9-10 years"),
  A = c(18, 2, 20),
  B = c(22, 28, 10),
  C = c(20, 40, 40)
)

cov_B_C <- cov(data$B, data$C)
print("Covariance between B and C:")
print(cov_B_C)

cov_matrix <- cov(data[, 2:4])
print("Covariance matrix:")
print(cov_matrix)

cor_B_C <- cor(data$B, data$C)
print("Correlation between B and C:")
print(cor_B_C)

cor_matrix <- cor(data[, 2:4])
print("Correlation matrix:")
print(cor_matrix)
```

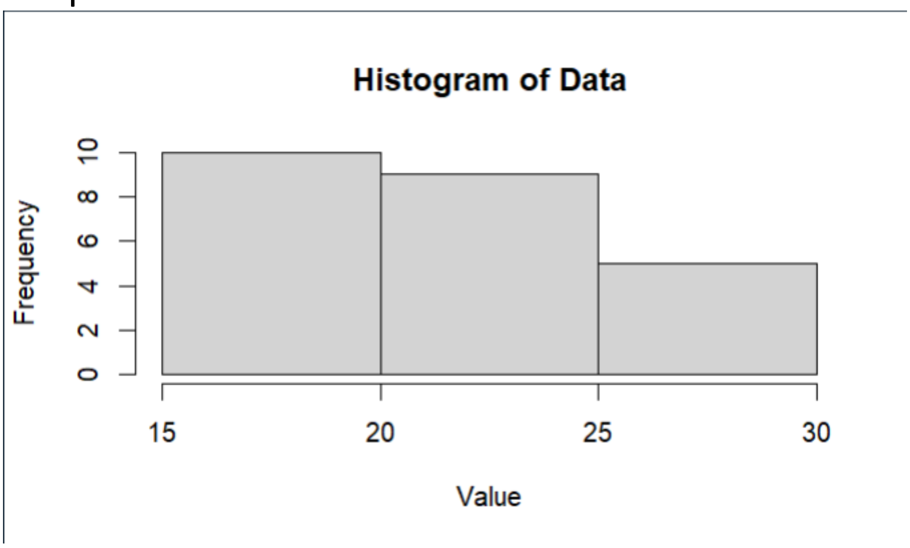
Output:

```
[1] "Covariance between B and C:"
> print(cov_B_C)
[1] -20
>
> cov_matrix <- cov(data[, 2:4])
> print("Covariance matrix:")
[1] "Covariance matrix:"
> print(cov_matrix)
          A   B         C
A  97.33333 -74 -46.66667
B -74.00000  84 -20.00000
C -46.66667 -20 133.33333
>
> cor_B_C <- cor(data$B, data$C)
> print("Correlation between B and C:")
[1] "Correlation between B and C:"
> print(cor_B_C)
[1] -0.1889822
>
> cor_matrix <- cor(data[, 2:4])
> print("Correlation matrix:")
[1] "Correlation matrix:"
> print(cor_matrix)
           A          B          C
A  1.0000000 -0.8183918 -0.4096440
B -0.8183918  1.0000000 -0.1889822
C -0.4096440 -0.1889822  1.0000000
>
```

Question 2:

```
#Reg No : 192224215
data <- c(18, 18, 18, 20, 20, 20, 20, 20, 20, 20, 21, 21, 21, 21, 25, 25, 25, 25, 25, 28, 28, 30, 30, 30)

bins <- cut(data, breaks = 3, labels = FALSE)

bin_means <- tapply(data, bins, mean)
smoothed_data <- bin_means[bins]

hist(data, breaks = 3, main = "Histogram of Data", xlab = "Value", ylab = "Frequency")
```

Output:

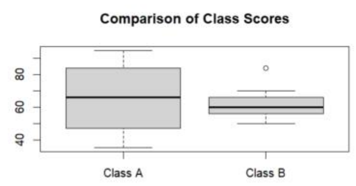


Question 3:

```r
#Reg No : 192224215A
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)
mean_B <- mean(class_B)
median_A <- median(class_A)
median_B <- median(class_B)
range_A <- max(class_A) - min(class_A)
range_B <- max(class_B) - min(class_B)

print(paste("Class A - Mean:", mean_A, "Median:", median_A, "Range:", range_A))
print(paste("Class B - Mean:", mean_B, "Median:", median_B, "Range:", range_B))

boxplot(class_A, class_B, names = c("Class A", "Class B"), main = "Comparison of Class Scores")
```

Output:

```
> #Reg No : 192224215A
> 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)
> mean_B <- mean(class_B)
> median_A <- median(class_A)
> median_B <- median(class_B)
> range_A <- max(class_A) - min(class_A)
> range_B <- max(class_B) - min(class_B)
>
> print(paste("Class A - Mean:", mean_A, "Median:", median_A, "Range:", range_A))
[1] "Class A - Mean: 65.7777777777778 Median: 66 Range: 60"
> print(paste("Class B - Mean:", mean_B, "Median:", median_B, "Range:", range_B))
[1] "Class B - Mean: 62.1111111111111 Median: 60 Range: 34"
>
> boxplot(class_A, class_B, names = c("Class A", "Class B"), main = "Comparison of Class Scores")
> |
```



Question 4:

```r
#Reg No : 192224215
min_max_normalize <- function(x) {
  (x - min(x)) / (max(x) - min(x))
}

z_score_normalize <- function(x) {
  (x - mean(x)) / sd(x)
}

data <- c(200, 300, 400, 600, 1000)

min_max_normalized <- min_max_normalize(data)
z_score_normalized <- z_score_normalize(data)

print("Min-Max Normalized:")
print(min_max_normalized)
print("Z-Score Normalized:")
print(z_score_normalized)
```

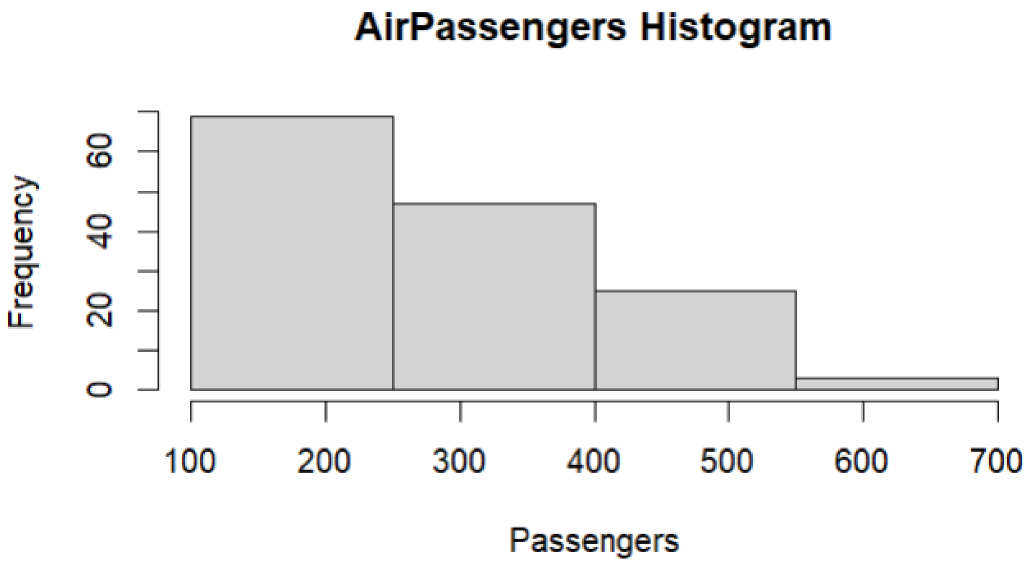
Output :

```
> print("Min-Max Normalized:")
[1] "Min-Max Normalized:"
> print(min_max_normalized)
[1] 0.000 0.125 0.250 0.500 1.000
> print("Z-Score Normalized:")
[1] "Z-Score Normalized:"
> print(z_score_normalized)
[1] -0.9486833 -0.6324555 -0.3162278  0.3162278  1.5811388
>
```

Question 5:

```
#Reg No : 192224215
data(AirPassengers)
hist(AirPassengers, breaks = seq(100, 700, by = 150), xlim = c(100, 700), main = "AirPassengers Histogram", xlab =
```

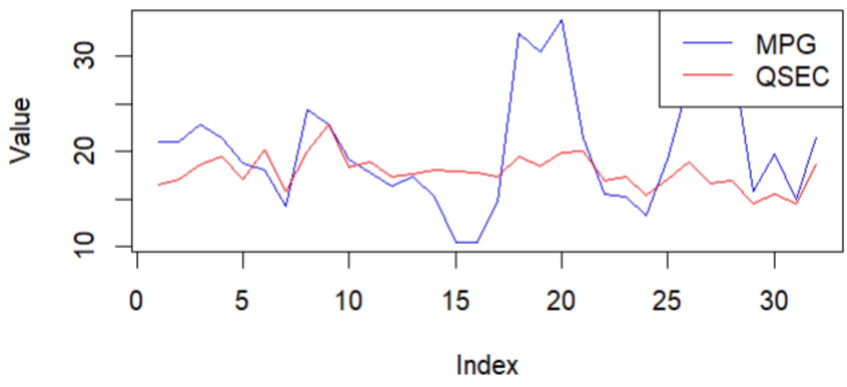
Output:



Question 6:

```
#Reg No : 192224215
data(mtcars)
plot(mtcars$mpg, type = "l", col = "blue", xlab = "Index", ylab = "Value")
lines(mtcars$qsec, col = "red")
legend("topright", legend = c("MPG", "QSEC"), col = c("blue", "red"), lty = 1)
```

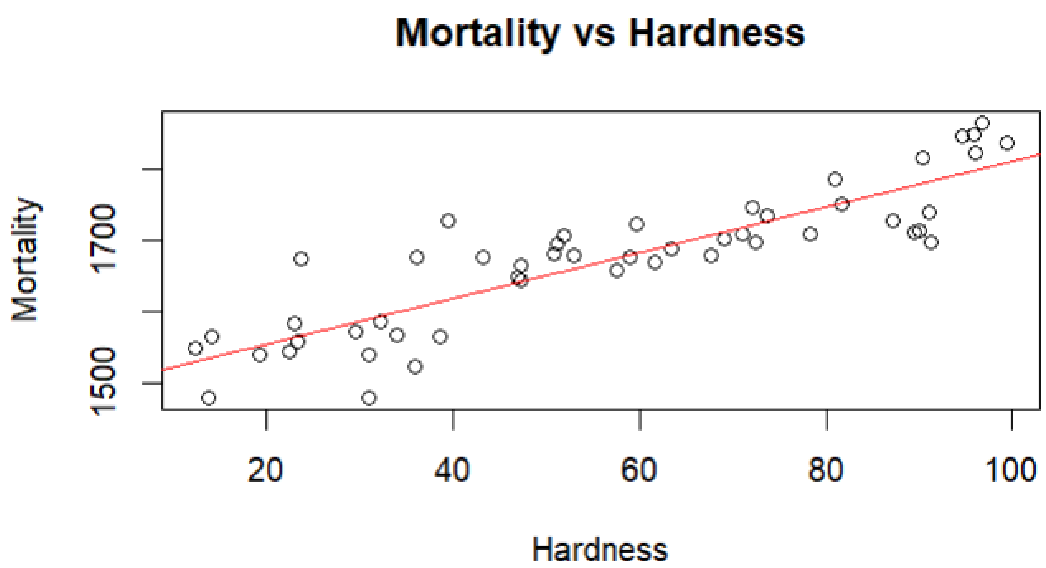
Output:



Question 7:

```
#Reg No : 192224215
install.packages("MASS")
library(MASS)
data(water)

plot(water$hardness, water$mortality, main = "Mortality vs Hardness", xlab = "Hardness", ylab = "Mortality")
abline(lm(mortality ~ hardness, data = water), col = "red")

model <- lm(mortality ~ hardness, data = water)
new_data <- data.frame(hardness = 88)
predicted_mortality <- predict(model, new_data)
print(paste("Predicted mortality for hardness 88:", predicted_mortality))
```

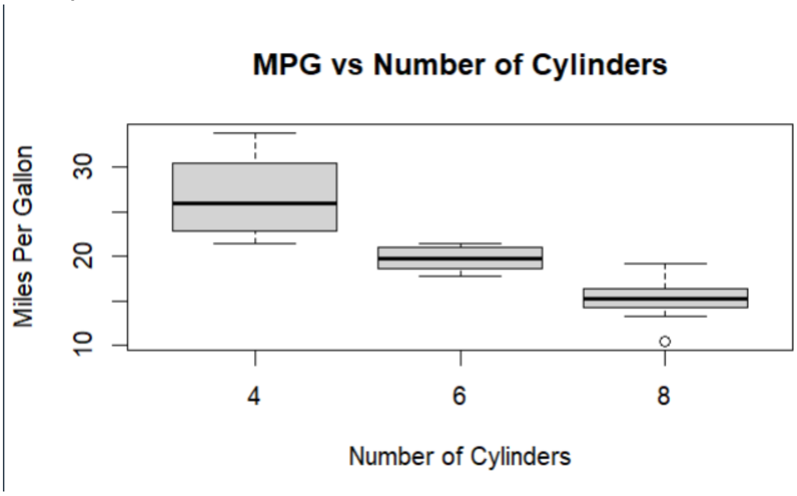
Output:



Question 8:

```
#Reg No : 192224215
data(mtcars)
boxplot(mpg ~ cyl, data = mtcars, main = "MPG vs Number of Cylinders", xlab = "Number of Cylinders", ylab = "Miles
```

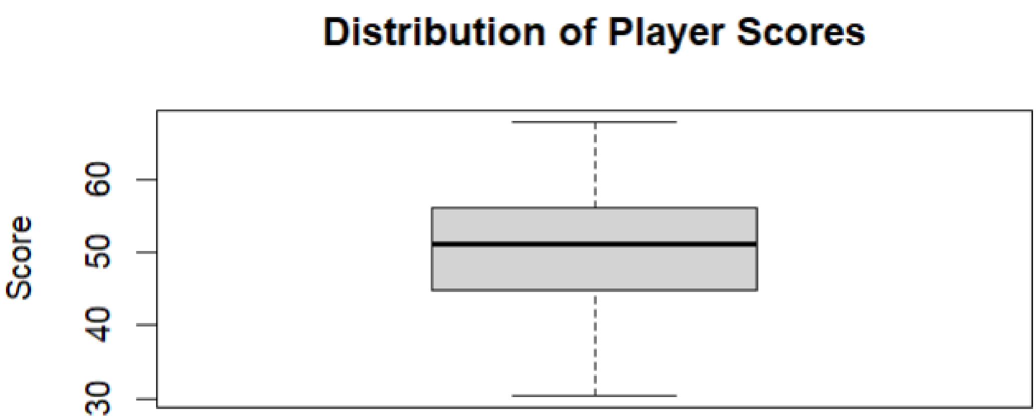
Output:



Question 9:

```
#Reg No : 192224215
set.seed(123)
player_scores <- rnorm(20, mean = 50, sd = 10)
boxplot(player_scores, main = "Distribution of Player Scores", ylab = "Score")
outliers <- boxplot.stats(player_scores)$out
points(rep(1, length(outliers)), outliers, col = "red", pch = 19)
```

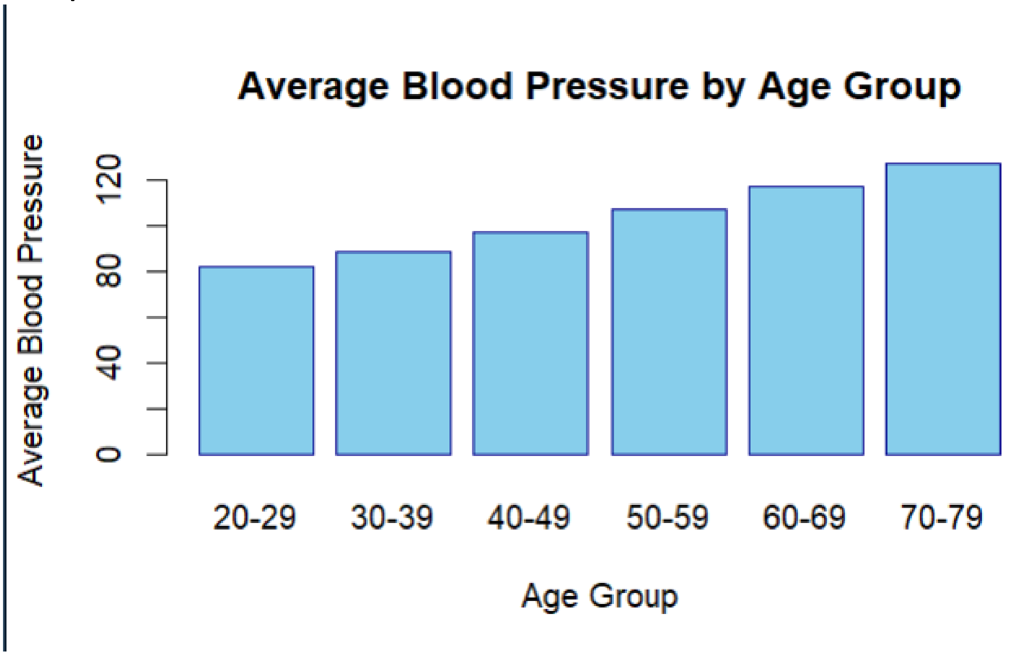
Output :

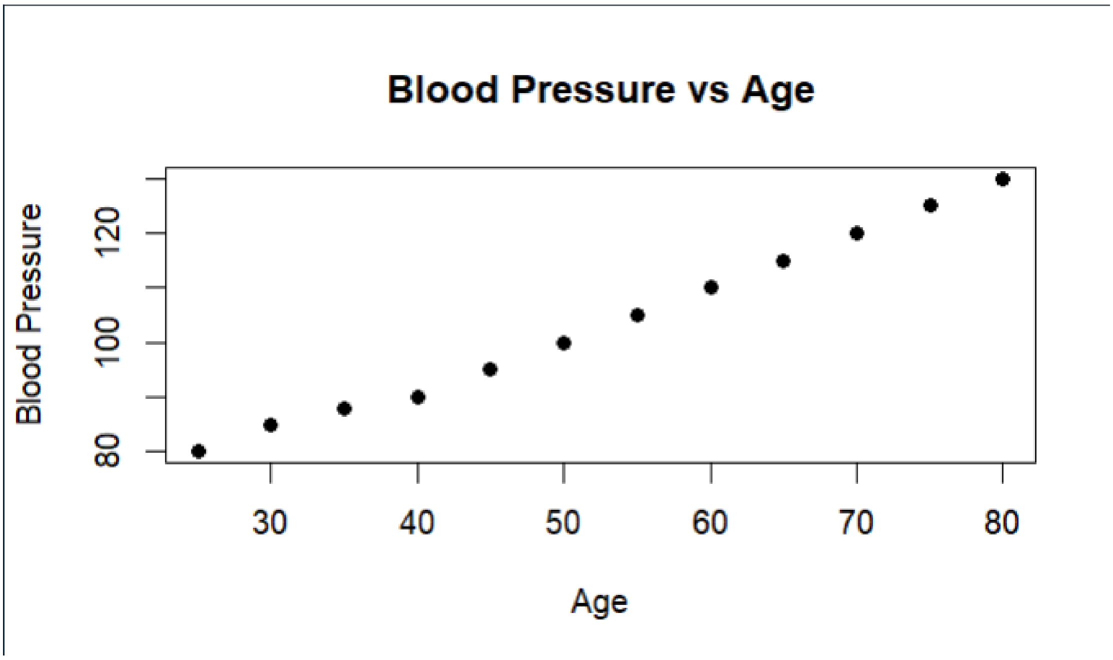


Question 10:

```r
#Reg No : 192224215
diabetes <- data.frame(
  Age = c(25, 30, 35, 40, 45, 50, 55, 60, 65, 70, 75, 80),
  BloodPressure = c(80, 85, 88, 90, 95, 100, 105, 110, 115, 120, 125, 130)
)
plot(diabetes$Age, diabetes$BloodPressure,
     main = "Blood Pressure vs Age",
     xlab = "Age",
     ylab = "Blood Pressure",
     pch = 19)
age_groups <- cut(diabetes$Age,
                  breaks = seq(20, 80, by = 10),
                  include.lowest = TRUE,
                  labels = c("20-29", "30-39", "40-49", "50-59", "60-69", "70-79"))
bp_means <- tapply(diabetes$BloodPressure, age_groups, mean)
barplot(bp_means,
        main = "Average Blood Pressure by Age Group",
        xlab = "Age Group",
        ylab = "Average Blood Pressure",
        col = "skyblue",
        border = "darkblue")
```

Output:





DAY 3

Question 1:

Output :

```
Apriori
=======

Minimum support: 0.85 (4 instances)
Minimum metric <confidence>: 0.9
Number of cycles performed: 3

Generated sets of large itemsets:

Size of set of large itemsets L(1): 4

Size of set of large itemsets L(2): 3

Size of set of large itemsets L(3): 1

Best rules found:

 1. D=f 4 ==> B=t 4    <conf:(1)> lift:(1.25) lev:(0.16) [0] conv:(0.8)
 2. B=t 4 ==> D=f 4    <conf:(1)> lift:(1.25) lev:(0.16) [0] conv:(0.8)
 3. E=t 4 ==> B=t 4    <conf:(1)> lift:(1.25) lev:(0.16) [0] conv:(0.8)
 4. B=t 4 ==> E=t 4    <conf:(1)> lift:(1.25) lev:(0.16) [0] conv:(0.8)
 5. E=t 4 ==> D=f 4    <conf:(1)> lift:(1.25) lev:(0.16) [0] conv:(0.8)
 6. D=f 4 ==> E=t 4    <conf:(1)> lift:(1.25) lev:(0.16) [0] conv:(0.8)
 7. D=f E=t 4 ==> B=t 4    <conf:(1)> lift:(1.25) lev:(0.16) [0] conv:(0.8)
 8. B=t E=t 4 ==> D=f 4    <conf:(1)> lift:(1.25) lev:(0.16) [0] conv:(0.8)
 9. B=t D=f 4 ==> E=t 4    <conf:(1)> lift:(1.25) lev:(0.16) [0] conv:(0.8)
10. E=t 4 ==> B=t D=f 4    <conf:(1)> lift:(1.25) lev:(0.16) [0] conv:(0.8)
```

Apriori

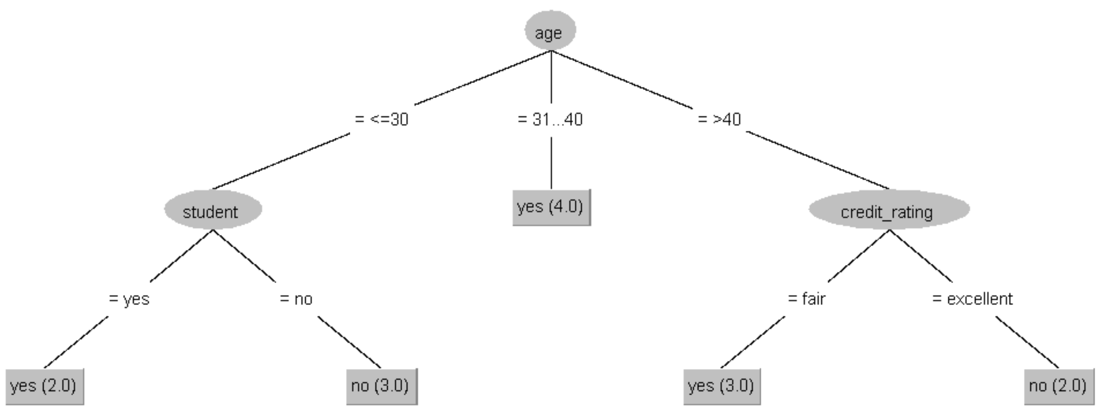
```
Scheme:       weka.associations.FPGrowth -P 2 -I -1 -N 10 -T 0 -C 0.9 -D 0.05 -U 1.0 -M 0.1
Relation:     market_basket
Instances:    5
Attributes:   5
              A
              B
              C
              D
              E
=== Associator model (full training set) ===

FPGrowth found 8 rules (displaying top 8)

1. [A=f]: 2 ==> [D=f]: 2   <conf:(1)> lift:(1.25) lev:(0.08) conv:(0.4)
2. [C=f]: 1 ==> [D=f]: 1   <conf:(1)> lift:(1.25) lev:(0.04) conv:(0.2)
3. [C=f]: 1 ==> [A=f]: 1   <conf:(1)> lift:(2.5) lev:(0.12) conv:(0.6)
4. [E=f]: 1 ==> [B=f]: 1   <conf:(1)> lift:(5) lev:(0.16) conv:(0.8)
5. [B=f]: 1 ==> [E=f]: 1   <conf:(1)> lift:(5) lev:(0.16) conv:(0.8)
6. [C=f]: 1 ==> [D=f, A=f]: 1   <conf:(1)> lift:(2.5) lev:(0.12) conv:(0.6)
7. [D=f, C=f]: 1 ==> [A=f]: 1   <conf:(1)> lift:(2.5) lev:(0.12) conv:(0.6)
8. [A=f, C=f]: 1 ==> [D=f]: 1   <conf:(1)> lift:(1.25) lev:(0.04) conv:(0.2)
```

FP Growth

Question 3:

```
=== Summary ===

Correctly Classified Instances            8               57.1429 %
Incorrectly Classified Instances          6               42.8571 %
Kappa statistic                           0.0667
Mean absolute error                       0.4381
Root mean squared error                   0.5418
Relative absolute error                  91.9971 %
Root relative squared error             109.8188 %
Total Number of Instances                14

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.667    0.600    0.667      0.667    0.667      0.067    0.511     0.690     yes
                 0.400    0.333    0.400      0.400    0.400      0.067    0.511     0.529     no
Weighted Avg.    0.571    0.505    0.571      0.571    0.571      0.067    0.511     0.632

=== Confusion Matrix ===

 a b   <-- classified as
 6 3 | a = yes
 3 2 | b = no
```

NaiveBayes



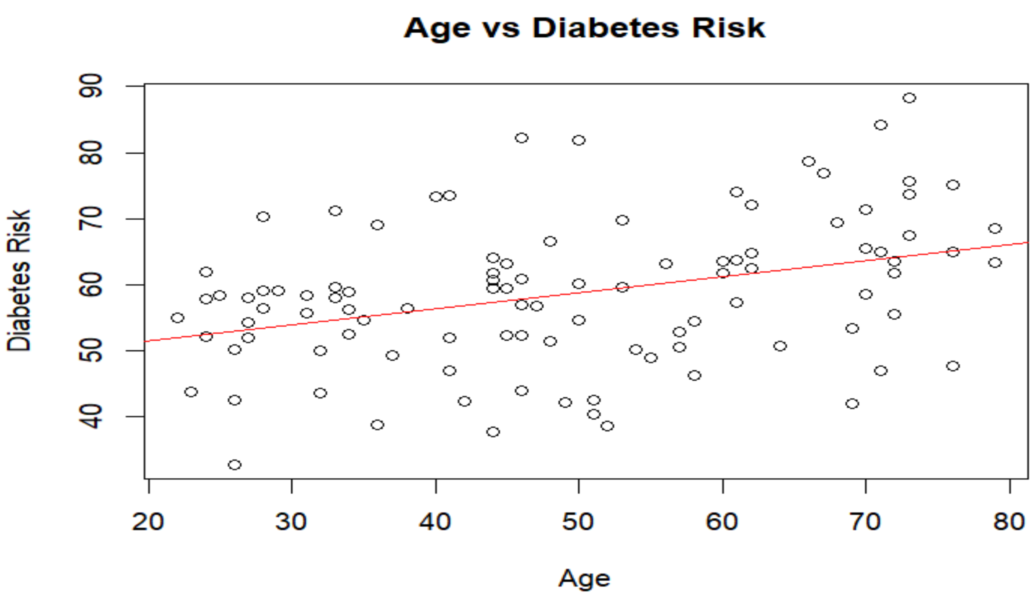
Decision Tree

Question 4:

Code :

```
set.seed(123)
n <- 100
age <- sample(20:80, n, replace = TRUE)
bmi <- rnorm(n, mean = 25, sd = 5)
blood_pressure <- rnorm(n, mean = 120, sd = 10)
glucose <- rnorm(n, mean = 100, sd = 20)
diabetes_risk <- 0.3 * age + 0.4 * bmi + 0.2 * blood_pressure + 0.1 * glucose + rnorm(n, mean = 0, sd = 10)
diabetes <- ifelse(diabetes_risk > mean(diabetes_risk), 1, 0)

diabetes_data <- data.frame(
  Age = age,
  BMI = bmi,
  BloodPressure = blood_pressure,
  Glucose = glucose,
  DiabetesRisk = diabetes_risk,
  Diabetes = diabetes
)

head(diabetes_data)

write.csv(diabetes_data, "diabetes_sample.csv", row.names = FALSE)

linear_model <- lm(DiabetesRisk ~ Age, data = diabetes_data)
summary(linear_model)

multiple_model <- lm(DiabetesRisk ~ Age + BMI + BloodPressure + Glucose, data = diabetes_data)
summary(multiple_model)

plot(diabetes_data$Age, diabetes_data$DiabetesRisk, main = "Age vs Diabetes Risk",
     xlab = "Age", ylab = "Diabetes Risk")
abline(linear_model, col = "red")
```

Output:

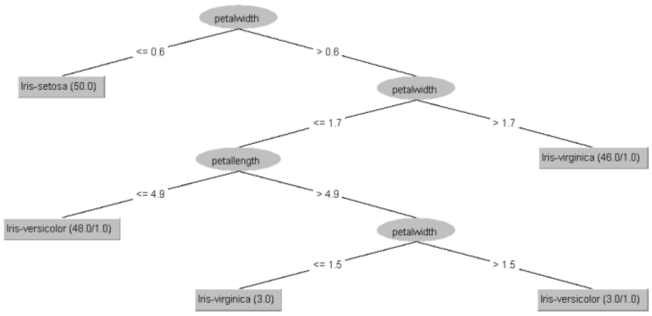


Question 5:
Output:

```
Apriori
=======

Minimum support: 0.85 (4 instances)
Minimum metric <confidence>: 0.8
Number of cycles performed: 3

Generated sets of large itemsets:

Size of set of large itemsets L(1): 6

Size of set of large itemsets L(2): 6

Size of set of large itemsets L(3): 1

Best rules found:

 1. E=t 4 ==> K=t 4    <conf:(1)> lift:(1) lev:(0) [0] conv:(0)
 2. D=f 4 ==> K=t 4    <conf:(1)> lift:(1) lev:(0) [0] conv:(0)
 3. A=f 4 ==> K=t 4    <conf:(1)> lift:(1) lev:(0) [0] conv:(0)
 4. U=f 4 ==> K=t 4    <conf:(1)> lift:(1) lev:(0) [0] conv:(0)
 5. I=f 4 ==> K=t 4    <conf:(1)> lift:(1) lev:(0) [0] conv:(0)
 6. U=f 4 ==> E=t 4    <conf:(1)> lift:(1.25) lev:(0.16) [0] conv:(0.8)
 7. E=t 4 ==> U=f 4    <conf:(1)> lift:(1.25) lev:(0.16) [0] conv:(0.8)
 8. E=t U=f 4 ==> K=t 4    <conf:(1)> lift:(1) lev:(0) [0] conv:(0)
 9. K=t U=f 4 ==> E=t 4    <conf:(1)> lift:(1.25) lev:(0.16) [0] conv:(0.8)
10. K=t E=t 4 ==> U=f 4    <conf:(1)> lift:(1.25) lev:(0.16) [0] conv:(0.8)
```

```
Scheme:       weka.associations.FPGrowth -P 2 -I -1 -N 10 -T 0 -C 0.8 -D 0.05 -U 1.0 -M 0.1
Relation:     market_basket_transactions
Instances:    5
Attributes:   11
              M
              O
              N
              K
              E
              Y
              D
              A
              U
              C
              I
=== Associator model (full training set) ===

FPGrowth found 71 rules (displaying top 10)

 1. [C=f]: 3 ==> [U=f]: 3   <conf:(1)> lift:(1.25) lev:(0.12) conv:(0.6)
 2. [Y=f]: 2 ==> [U=f]: 2   <conf:(1)> lift:(1.25) lev:(0.08) conv:(0.4)
 3. [M=f]: 2 ==> [U=f]: 2   <conf:(1)> lift:(1.25) lev:(0.08) conv:(0.4)
 4. [C=f]: 3 ==> [I=f]: 3   <conf:(1)> lift:(1.25) lev:(0.12) conv:(0.6)
 5. [O=f]: 2 ==> [I=f]: 2   <conf:(1)> lift:(1.25) lev:(0.08) conv:(0.4)
 6. [N=f]: 3 ==> [D=f]: 3   <conf:(1)> lift:(1.25) lev:(0.12) conv:(0.6)
 7. [Y=f]: 2 ==> [D=f]: 2   <conf:(1)> lift:(1.25) lev:(0.08) conv:(0.4)
 8. [O=f]: 2 ==> [D=f]: 2   <conf:(1)> lift:(1.25) lev:(0.08) conv:(0.4)
 9. [M=f]: 2 ==> [A=f]: 2   <conf:(1)> lift:(1.25) lev:(0.08) conv:(0.4)
10. [Y=f]: 2 ==> [N=f]: 2   <conf:(1)> lift:(1.67) lev:(0.16) conv:(0.8)
```

Question 6:
Output:



Decision Tree

```
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         144               96      %
Incorrectly Classified Instances         6                4      %
Kappa statistic                          0.94
Mean absolute error                      0.0287
Root mean squared error                  0.1424
Relative absolute error                  6.456  %
Root relative squared error             30.2139 %
Total Number of Instances              150

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     Iris-set
                 0.920    0.020    0.958      0.920    0.939      0.910    0.972     0.934     Iris-ver
                 0.960    0.040    0.923      0.960    0.941      0.911    0.972     0.934     Iris-vir
Weighted Avg.    0.960    0.020    0.960      0.960    0.960      0.940    0.981     0.956

=== Confusion Matrix ===

  a  b  c   <-- classified as
 50  0  0 |  a = Iris-setosa
  0 46  4 |  b = Iris-versicolor
  0  2 48 |  c = Iris-virginica
```

Logistic

Question 7:
Output :

```
Apriori
=======

Minimum support: 0.55 (3 instances)
Minimum metric <confidence>: 0.6
Number of cycles performed: 9

Generated sets of large itemsets:

Size of set of large itemsets L(1): 6

Size of set of large itemsets L(2): 7

Size of set of large itemsets L(3): 4

Size of set of large itemsets L(4): 1

Best rules found:

 1. Chips=t 4 ==> Buns=f 4    <conf:(1)> lift:(1.5) lev:(0.22) [1] conv:(1.33)
 2. Buns=f 4 ==> Chips=t 4    <conf:(1)> lift:(1.5) lev:(0.22) [1] conv:(1.33)
 3. Coke=t 3 ==> Buns=f 3    <conf:(1)> lift:(1.5) lev:(0.17) [1] conv:(1)
 4. Coke=t 3 ==> Ketchup=f 3    <conf:(1)> lift:(1.5) lev:(0.17) [1] conv:(1)
 5. Coke=t 3 ==> Chips=t 3    <conf:(1)> lift:(1.5) lev:(0.17) [1] conv:(1)
 6. Ketchup=f Coke=t 3 ==> Buns=f 3    <conf:(1)> lift:(1.5) lev:(0.17) [1] conv:(1)
 7. Buns=f Coke=t 3 ==> Ketchup=f 3    <conf:(1)> lift:(1.5) lev:(0.17) [1] conv:(1)
 8. Buns=f Ketchup=f 3 ==> Coke=t 3    <conf:(1)> lift:(2) lev:(0.25) [1] conv:(1.5)
 9. Coke=t 3 ==> Buns=f Ketchup=f 3    <conf:(1)> lift:(2) lev:(0.25) [1] conv:(1.5)
10. Ketchup=f Chips=t 3 ==> Buns=f 3    <conf:(1)> lift:(1.5) lev:(0.17) [1] conv:(1)
```

```
FPGrowth found 5 rules (displaying top 5)

1. [Hot_Dogs=f]: 2 ==> [Buns=f]: 2   <conf:(1)> lift:(1.5) lev:(0.11) conv:(0.67)
2. [Chips=f]: 2 ==> [Coke=f]: 2   <conf:(1)> lift:(2) lev:(0.17) conv:(1)
3. [Ketchup=f]: 4 ==> [Buns=f]: 3   <conf:(0.75)> lift:(1.13) lev:(0.06) conv:(0.67)
4. [Buns=f]: 4 ==> [Ketchup=f]: 3   <conf:(0.75)> lift:(1.13) lev:(0.06) conv:(0.67)
5. [Coke=f]: 3 ==> [Chips=f]: 2   <conf:(0.67)> lift:(2) lev:(0.17) conv:(1)
```

Question 8:

Output :

```
Correctly Classified Instances         139               92.6667 %
Incorrectly Classified Instances        11                7.3333 %
Kappa statistic                          0.89
Mean absolute error                      0.092
Root mean squared error                  0.2087
Relative absolute error                 20.6978 %
Root relative squared error             44.2707 %
Total Number of Instances              150

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     Iris-set
                 0.880    0.050    0.898      0.880    0.889      0.834    0.946     0.861     Iris-ver
                 0.900    0.060    0.882      0.900    0.891      0.836    0.947     0.869     Iris-vir
Weighted Avg.    0.927    0.037    0.927      0.927    0.927      0.890    0.964     0.910

=== Confusion Matrix ===

  a  b  c   <-- classified as
 50  0  0 |  a = Iris-setosa
  0 44  6 |  b = Iris-versicolor
  0  5 45 |  c = Iris-virginica
```

Rule Based Accuracy - 93%

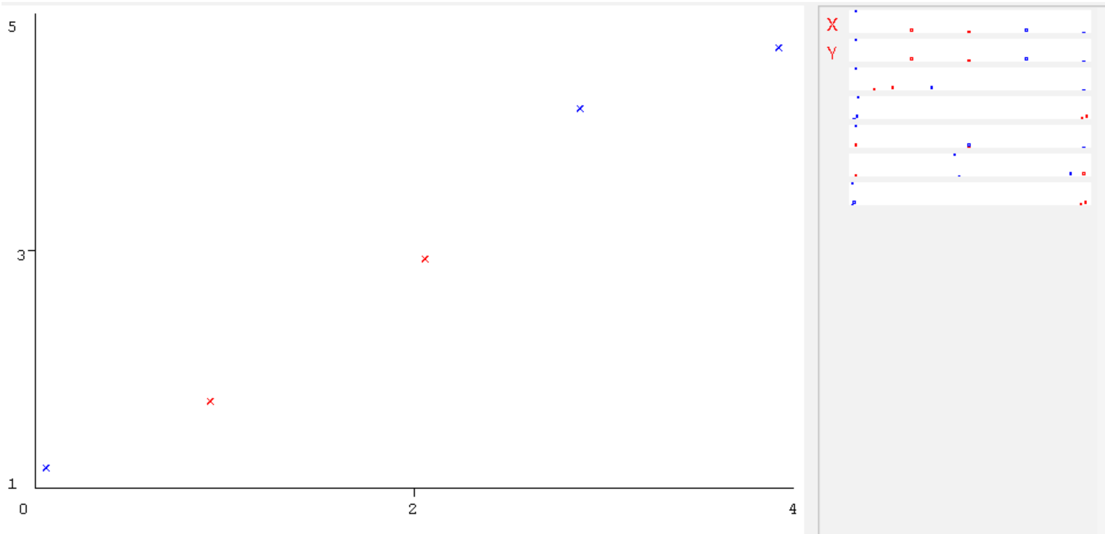
```
Correctly Classified Instances         144               96      %
Incorrectly Classified Instances         6                4      %
Kappa statistic                          0.94
Mean absolute error                      0.035
Root mean squared error                  0.1586
Relative absolute error                  7.8705 %
Root relative squared error             33.6353 %
Total Number of Instances              150

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area
                 0.980    0.000    1.000      0.980    0.990      0.985    0.990     0.987
                 0.940    0.030    0.940      0.940    0.940      0.910    0.952     0.880
                 0.960    0.030    0.941      0.960    0.950      0.925    0.961     0.905
Weighted Avg.    0.960    0.020    0.960      0.960    0.960      0.940    0.968     0.924

=== Confusion Matrix ===

  a  b  c   <-- classified as
 49  1  0 |  a = Iris-setosa
  0 47  3 |  b = Iris-versicolor
  0  2 48 |  c = Iris-virginica
```

Decision Tree - 96%

DAY 4

Question 1 :

Output :



Question 2 :

Output :

```
Within cluster sum of squared errors: 2.9514403292181073

Initial starting points (random):

Cluster 0: 444,Female,25,160000,40
Cluster 1: 111,Male,28,150000,39

Missing values globally replaced with mean/mode

Final cluster centroids:
                         Cluster#
Attribute      Full Data          0          1
                   (6.0)      (3.0)      (3.0)
================================================
EmployeeID         388.5        444        333
Gender              Male     Female       Male
Age              27.1667         27    27.3333
Salary            165000 163333.3333 166666.6667
Credit           47.3333    48.6667         46



Time taken to build model (full training data) : 0 seconds

=== Model and evaluation on training set ===

Clustered Instances

0      3 ( 50%)
1      3 ( 50%)
```

Question 3:

Output :

```
Correctly Classified Instances         144               96      %
Incorrectly Classified Instances         6                4      %
Kappa statistic                          0.94
Mean absolute error                      0.0342
Root mean squared error                  0.155
Relative absolute error                  7.6997 %
Root relative squared error             32.8794 %
Total Number of Instances              150
```

Naive Bayes = 96%

```
Correctly Classified Instances         144               96      %
Incorrectly Classified Instances         6                4      %
Kappa statistic                          0.94
Mean absolute error                      0.2311
Root mean squared error                  0.288
Relative absolute error                 52      %
Root relative squared error             61.101  %
Total Number of Instances              150
```

SVM = 96%

Question 4:

Code :

```r
people <- c("Gopu", "Babu", "Baby", "Gopal", "Krishna", "Jai", "Dev", "Malini", "Hema", "Anu")
vegetarian <- c(TRUE, TRUE, TRUE, FALSE, TRUE, FALSE, FALSE, TRUE, TRUE, TRUE)

veg_count <- sum(vegetarian)
non_veg_count <- sum(!vegetarian)

print(paste("Vegetarians:", veg_count))
print(paste("Non-vegetarians:", non_veg_count))
print(paste("Greater count:", ifelse(veg_count > non_veg_count, "Vegetarians", "Non-vegetarians")))
```

Output :

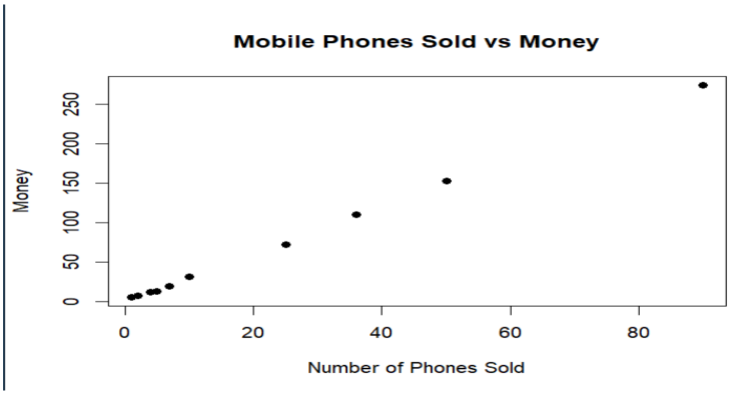
```
> print(paste("Vegetarians:", veg_count))
[1] "Vegetarians: 7"
> print(paste("Non-vegetarians:", non_veg_count))
[1] "Non-vegetarians: 3"
> print(paste("Greater count:", ifelse(veg_count > non_veg_count, "Vegetarians", "Non-vegetarians")))
[1] "Greater count: Vegetarians"
>
```

Question 5:

Code :

```r
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="Mobile Phones Sold vs Money", xlab="Number of Phones Sold", ylab="Money", pch=19)
```

Output :



Question 6:

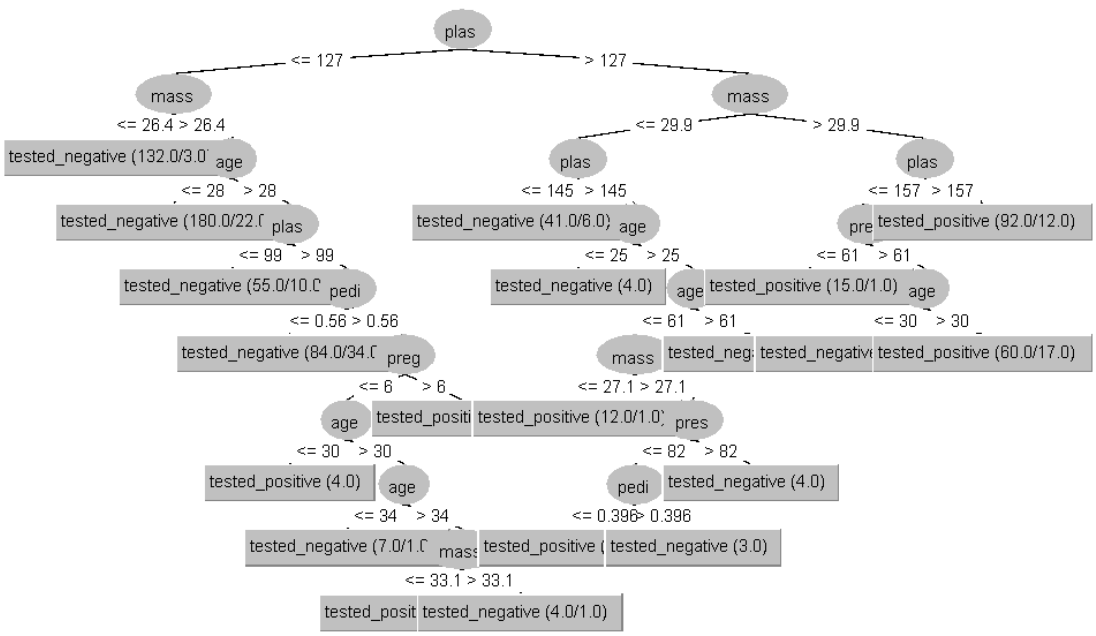
Output :

```
Scheme:       weka.associations.FPGrowth -P 2 -I -1 -N 10 -T 0 -C 0.9 -D 0.05 -U 1.0 -M 0.1
Relation:     transactions
Instances:    9
Attributes:   5
              SONY
              BPL
              LG
              SAMSUNG
              ONIDA
=== Associator model (full training set) ===

FPGrowth found 12 rules (displaying top 10)

 1. [SAMSUNG=t]: 2 ==> [BPL=t]: 2   <conf:(1)> lift:(1.29) lev:(0.05) conv:(0.44)
 2. [LG=t]: 2 ==> [BPL=t]: 2   <conf:(1)> lift:(1.29) lev:(0.05) conv:(0.44)
 3. [LG=t]: 2 ==> [SONY=t]: 2   <conf:(1)> lift:(1.5) lev:(0.07) conv:(0.67)
 4. [SONY=t, SAMSUNG=t]: 1 ==> [BPL=t]: 1   <conf:(1)> lift:(1.29) lev:(0.02) conv:(0.22)
 5. [LG=t]: 2 ==> [BPL=t, SONY=t]: 2   <conf:(1)> lift:(2.25) lev:(0.12) conv:(1.11)
 6. [BPL=t, LG=t]: 2 ==> [SONY=t]: 2   <conf:(1)> lift:(1.5) lev:(0.07) conv:(0.67)
 7. [SONY=t, LG=t]: 2 ==> [BPL=t]: 2   <conf:(1)> lift:(1.29) lev:(0.05) conv:(0.44)
 8. [ONIDA=t, LG=t]: 1 ==> [BPL=t]: 1   <conf:(1)> lift:(1.29) lev:(0.02) conv:(0.22)
 9. [ONIDA=t, LG=t]: 1 ==> [SONY=t]: 1   <conf:(1)> lift:(1.5) lev:(0.04) conv:(0.33)
10. [ONIDA=t, LG=t]: 1 ==> [BPL=t, SONY=t]: 1   <conf:(1)> lift:(2.25) lev:(0.06) conv:(0.56)
```

Question 7:

Output:

Decision Tree:



SVM:

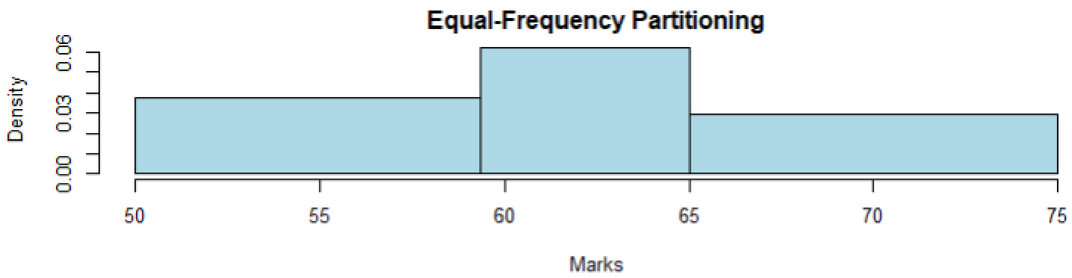
```
Time taken to build model: 0.02 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         594               77.3438 %
Incorrectly Classified Instances       174               22.6563 %
Kappa statistic                          0.4682
Mean absolute error                      0.2266
Root mean squared error                  0.476
Relative absolute error                 49.848  %
Root relative squared error             99.862  %
Total Number of Instances              768

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.898    0.459    0.785      0.898    0.838      0.480    0.720     0.771     tested_
                 0.541    0.102    0.740      0.541    0.625      0.480    0.720     0.560     tested_
Weighted Avg.    0.773    0.334    0.769      0.773    0.763      0.480    0.720     0.698

=== Confusion Matrix ===

   a   b   <-- classified as
 449  51 |   a = tested_negative
 123 145 |   b = tested_positive
```

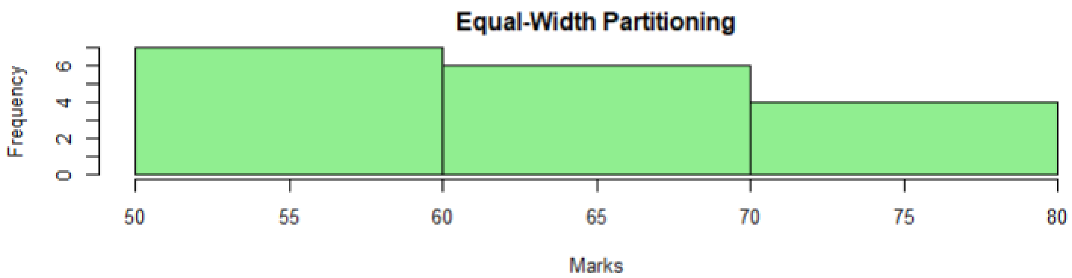
Question 8:

Code :

```r
marks <- c(55, 60, 71, 63, 55, 65, 50, 55, 58, 59, 61, 63, 65, 67, 71, 72, 75)

eq_freq <- cut(marks, breaks = quantile(marks, probs = seq(0, 1, length.out = 4)),
               labels = c("Low", "Medium", "High"), include.lowest = TRUE)

eq_width <- cut(marks, breaks = 3, labels = c("Low", "Medium", "High"))

set.seed(123)
km <- kmeans(marks, centers = 3)
cluster_bins <- km$cluster

par(mfrow = c(3, 1), mar = c(4, 4, 2, 1))

hist(marks, breaks = quantile(marks, probs = seq(0, 1, length.out = 4)),
     main = "Equal-Frequency Partitioning", xlab = "Marks", col = "lightblue")

hist(marks, breaks = 3, main = "Equal-Width Partitioning", xlab = "Marks", col = "lightgreen")

hist(marks, breaks = km$centers[order(km$centers)],
     main = "Clustering-based Partitioning", xlab = "Marks", col = "lightpink")

cat("Equal-Frequency Binning:\n")
print(table(eq_freq))

cat("\nEqual-Width Binning:\n")
print(table(eq_width))

cat("\nClustering-based Binning:\n")
print(table(factor(cluster_bins, levels = 1:3, labels = c("Low", "Medium", "High"))))
```

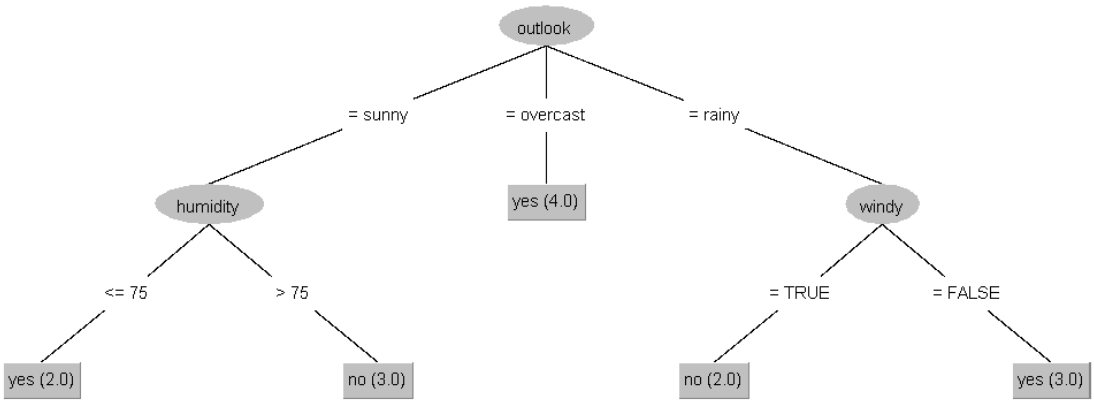
Output :





Question 9:

Output:



Question 10:

Output:

```
=== Run information ===

Scheme:       weka.associations.FPGrowth -P 2 -I -1 -N 10 -T 0 -C 0.9 -D 0.05 -U 1.0 -M 0.1
Relation:     transactions
Instances:    9
Attributes:   5
              SONY
              BPL
              LG
              SAMSUNG
              ONIDA
=== Associator model (full training set) ===

FPGrowth found 12 rules (displaying top 10)

 1. [SAMSUNG=t]: 2 ==> [BPL=t]: 2   <conf:(1)> lift:(1.29) lev:(0.05) conv:(0.44)
 2. [LG=t]: 2 ==> [BPL=t]: 2   <conf:(1)> lift:(1.29) lev:(0.05) conv:(0.44)
 3. [LG=t]: 2 ==> [SONY=t]: 2   <conf:(1)> lift:(1.5) lev:(0.07) conv:(0.67)
 4. [SONY=t, SAMSUNG=t]: 1 ==> [BPL=t]: 1   <conf:(1)> lift:(1.29) lev:(0.02) conv:(0.22)
 5. [LG=t]: 2 ==> [BPL=t, SONY=t]: 2   <conf:(1)> lift:(2.25) lev:(0.12) conv:(1.11)
 6. [BPL=t, LG=t]: 2 ==> [SONY=t]: 2   <conf:(1)> lift:(1.5) lev:(0.07) conv:(0.67)
 7. [SONY=t, LG=t]: 2 ==> [BPL=t]: 2   <conf:(1)> lift:(1.29) lev:(0.05) conv:(0.44)
 8. [ONIDA=t, LG=t]: 1 ==> [BPL=t]: 1   <conf:(1)> lift:(1.29) lev:(0.02) conv:(0.22)
 9. [ONIDA=t, LG=t]: 1 ==> [SONY=t]: 1   <conf:(1)> lift:(1.5) lev:(0.04) conv:(0.33)
10. [ONIDA=t, LG=t]: 1 ==> [BPL=t, SONY=t]: 1   <conf:(1)> lift:(2.25) lev:(0.06) conv:(0.56)
```

FP Growth

```
=== Associator model (full training set) ===


Apriori
=======

Minimum support: 0.16 (1 instances)
Minimum metric <confidence>: 0.9
Number of cycles performed: 17

Generated sets of large itemsets:

Size of set of large itemsets L(1): 5

Size of set of large itemsets L(2): 8

Size of set of large itemsets L(3): 5

Size of set of large itemsets L(4): 1

Best rules found:

 1. LG=t 2 ==> SONY=t 2    <conf:(1)> lift:(1.5) lev:(0.07) [0] conv:(0.67)
 2. LG=t 2 ==> BPL=t 2    <conf:(1)> lift:(1.29) lev:(0.05) [0] conv:(0.44)
 3. SAMSUNG=t 2 ==> BPL=t 2    <conf:(1)> lift:(1.29) lev:(0.05) [0] conv:(0.44)
 4. BPL=t LG=t 2 ==> SONY=t 2    <conf:(1)> lift:(1.5) lev:(0.07) [0] conv:(0.67)
 5. SONY=t LG=t 2 ==> BPL=t 2    <conf:(1)> lift:(1.29) lev:(0.05) [0] conv:(0.44)
 6. LG=t 2 ==> SONY=t BPL=t 2    <conf:(1)> lift:(2.25) lev:(0.12) [1] conv:(1.11)
 7. SONY=t SAMSUNG=t 1 ==> BPL=t 1    <conf:(1)> lift:(1.29) lev:(0.02) [0] conv:(0.22)
 8. LG=t ONIDA=t 1 ==> SONY=t 1    <conf:(1)> lift:(1.5) lev:(0.04) [0] conv:(0.33)
 9. LG=t ONIDA=t 1 ==> BPL=t 1    <conf:(1)> lift:(1.29) lev:(0.02) [0] conv:(0.22)
10. BPL=t LG=t ONIDA=t 1 ==> SONY=t 1    <conf:(1)> lift:(1.5) lev:(0.04) [0] conv:(0.33)
```

Apriori

Question 11:

Code :

```r
data <- c(100, 70, 60, 90, 90)

min_max_norm <- function(x) {
  (x - min(x)) / (max(x) - min(x))
}
result_min_max <- min_max_norm(data)

z_score_norm <- function(x) {
  (x - mean(x)) / sd(x)
}
result_z_score <- z_score_norm(data)

mad_norm <- function(x) {
  (x - mean(x)) / mean(abs(x - mean(x)))
}
result_mad <- mad_norm(data)

decimal_norm <- function(x) {
  j <- ceiling(log10(max(abs(x))))
  x / (10^j)
}
result_decimal <- decimal_norm(data)

print("Original data:")
print(data)
print("(a) Min-Max Normalization:")
print(result_min_max)
print("(b) Z-score Normalization:")
print(result_z_score)
print("(c) Z-score with Mean Absolute Deviation:")
print(result_mad)
print("(d) Decimal Scaling Normalization:")
print(result_decimal)
```

Output:

```
[1] "Original data:"
> print(data)
[1] 100  70  60  90  90
> print("(a) Min-Max Normalization:")
[1] "(a) Min-Max Normalization:"
> print(result_min_max)
[1] 1.00 0.25 0.00 0.75 0.75
> print("(b) Z-score Normalization:")
[1] "(b) Z-score Normalization:"
> print(result_z_score)
[1]  1.0954451 -0.7302967 -1.3388774  0.4868645  0.4868645
> print("(c) Z-score with Mean Absolute Deviation:")
[1] "(c) Z-score with Mean Absolute Deviation:"
> print(result_mad)
[1]  1.3235294 -0.8823529 -1.6176471  0.5882353  0.5882353
> print("(d) Decimal Scaling Normalization:")
[1] "(d) Decimal Scaling Normalization:"
> print(result_decimal)
[1] 1.0 0.7 0.6 0.9 0.9
```

Question 12:

Code:

```r
avg_speed <- c(78, 81, 82, 74, 83, 82, 77, 80, 70)
total_time <- c(39, 37, 36, 42, 35, 36, 40, 38, 46)

sd_avg_speed <- sd(avg_speed)
sd_total_time <- sd(total_time)

var_avg_speed <- var(avg_speed)
var_total_time <- var(total_time)

cat("a) Standard Deviation:\n")
cat("   AvgSpeed:", sd_avg_speed, "\n")
cat("   TotalTime:", sd_total_time, "\n\n")

cat("b) Variance:\n")
cat("   AvgSpeed:", var_avg_speed, "\n")
cat("   TotalTime:", var_total_time, "\n")
```

Output:

```
a) Standard Deviation:
> cat("   AvgSpeed:", sd_avg_speed, "\n")
   AvgSpeed: 4.304391
> cat("   TotalTime:", sd_total_time, "\n\n")
   TotalTime: 3.492054


>
> cat("b) Variance:\n")
b) Variance:
> cat("   AvgSpeed:", var_avg_speed, "\n")
   AvgSpeed: 18.52778
> cat("   TotalTime:", var_total_time, "\n")
   TotalTime: 12.19444
```

Question 13:

Output:

Apriori:

```
Apriori
=======

Minimum support: 0.5 (2 instances)
Minimum metric <confidence>: 0.9
Number of cycles performed: 10

Generated sets of large itemsets:

Size of set of large itemsets L(1): 7

Size of set of large itemsets L(2): 14

Size of set of large itemsets L(3): 12

Size of set of large itemsets L(4): 5

Size of set of large itemsets L(5): 1

Best rules found:

 1. E=t 4 ==> K=t 4    <conf:(1)> lift:(1) lev:(0) [0] conv:(0)
 2. M=t 3 ==> K=t 3    <conf:(1)> lift:(1) lev:(0) [0] conv:(0)
 3. O=t 3 ==> K=t 3    <conf:(1)> lift:(1) lev:(0) [0] conv:(0)
 4. O=t 3 ==> E=t 3    <conf:(1)> lift:(1.25) lev:(0.12) [0] conv:(0.6)
 5. Y=t 3 ==> K=t 3    <conf:(1)> lift:(1) lev:(0) [0] conv:(0)
 6. O=t E=t 3 ==> K=t 3    <conf:(1)> lift:(1) lev:(0) [0] conv:(0)
 7. O=t K=t 3 ==> E=t 3    <conf:(1)> lift:(1.25) lev:(0.12) [0] conv:(0.6)
 8. O=t 3 ==> K=t E=t 3    <conf:(1)> lift:(1.25) lev:(0.12) [0] conv:(0.6)
 9. N=t 2 ==> O=t 2    <conf:(1)> lift:(1.67) lev:(0.16) [0] conv:(0.8)
10. N=t 2 ==> K=t 2    <conf:(1)> lift:(1) lev:(0) [0] conv:(0)
```

FP Growth :

```
=== Run information ===

Scheme:       weka.associations.FPGrowth -P 2 -I -1 -N 10 -T 0 -C 0.9 -D 0.05 -U 1.0 -M 0.1
Relation:     transaction_data
Instances:    5
Attributes:   11
              M
              O
              N
              K
              E
              Y
              D
              A
              U
              C
              I
=== Associator model (full training set) ===

FPGrowth found 100 rules (displaying top 10)

 1. [E=t]: 4 ==> [K=t]: 4   <conf:(1)> lift:(1) lev:(0) conv:(0)
 2. [Y=t]: 3 ==> [K=t]: 3   <conf:(1)> lift:(1) lev:(0) conv:(0)
 3. [O=t]: 3 ==> [K=t]: 3   <conf:(1)> lift:(1) lev:(0) conv:(0)
 4. [M=t]: 3 ==> [K=t]: 3   <conf:(1)> lift:(1) lev:(0) conv:(0)
 5. [N=t]: 2 ==> [K=t]: 2   <conf:(1)> lift:(1) lev:(0) conv:(0)
 6. [C=t]: 2 ==> [K=t]: 2   <conf:(1)> lift:(1) lev:(0) conv:(0)
 7. [O=t]: 3 ==> [E=t]: 3   <conf:(1)> lift:(1.25) lev:(0.12) conv:(0.6)
 8. [N=t]: 2 ==> [E=t]: 2   <conf:(1)> lift:(1.25) lev:(0.08) conv:(0.4)
 9. [N=t]: 2 ==> [Y=t]: 2   <conf:(1)> lift:(1.67) lev:(0.16) conv:(0.8)
10. [N=t]: 2 ==> [O=t]: 2   <conf:(1)> lift:(1.67) lev:(0.16) conv:(0.8)
```