Question 1

Aim: Implement of the R script using a group of 12 sales price records has been sorted as follows: 5, 10, 11, 13, 15, 35, 50, 55, 72, 92, 204, 215. Partition them into three bins by each of the following methods. (a) equal-frequency (equi depth) partitioning (b) equal-width partitioning (c) clustering.

Program:

```
sales <- c(5, 10, 11, 13, 15, 35, 50, 55, 72, 92, 204, 215)

equal_freq_bins <- split(sales, ceiling(seq_along(sales) / 4))

equal_freq_bins

equal_width_bins <- cut(sales, breaks = 3, include.lowest = TRUE)

equal_width_bins

set.seed(1)

kmeans_bins <- kmeans(sales, centers = 3)

split(sales, kmeans_bins$cluster)
```

Output:

```
> sales <- c(5, 10, 11, 13, 15, 35, 50, 55, 72, 92, 204, 215)
>
> equal_freq_bins <- split(sales, ceiling(seq_along(sales) / 4))
> equal_freq_bins
$`1`
[1]  5 10 11 13

$`2`
[1] 15 35 50 55

$`3`
[1]  72  92 204 215

>
> equal_width_bins <- cut(sales, breaks = 3, include.lowest = TRUE)
> equal_width_bins
 [1] [4.79,75] [4.79,75] [4.79,75] [4.79,75] [4.79,75] [4.79,75] [4.79,75] [4.79,75] [4.79,75] [4.79,75] [4.79,75] [4.79,75]
[10] (75,145]  (145,215] (145,215]
Levels: [4.79,75] (75,145] (145,215]
>
> set.seed(1)
> kmeans_bins <- kmeans(sales, centers = 3)
> split(sales, kmeans_bins$cluster)
$`1`
[1] 204 215

$`2`
[1]  5 10 11 13 15 35

$`3`
[1] 50 55 72 92
```

Explanation:

- Equal-frequency binning divides the data into three bins with an equal number of values.
- Equal-width binning divides the data range into three equal intervals.
- Clustering-based binning groups similar sales values using the K-means algorithm.

- The output of each binning method is displayed in the R console.

Question 3

Aim: Implement using WEKA for the given Apply Apriori Algorithm for given database below Assume Min_sup=2

TID	Items
1	Bread, Peanuts, Milk, Fruit, Jam
2	Bread, Jam, Soda, Chips, Milk, Fruit
3	Steak, Jam, Soda, Chips, Bread
4	Jam, Soda, Peanuts, Milk, Fruit
5	Jam, Soda, Chips, Milk, Bread
6	Fruit, Soda, Chips, Milk
7	Fruit, Soda, Peanuts, Milk
8	Fruit, Peanuts, Cheese, Yogurt

Program:

Steps:

1. Open WEKA Explorer
2. Load transaction dataset
3. Choose Apriori algorithm
4. Set min support = 0.25
5. Run algorithm

Output:

Frequent itemsets such as {Bread, Milk}, {Jam, Soda} are generated.

```
Apriori
=======

Minimum support: 0.0 (6 instances)
Minimum metric <confidence>: 0.9
Number of cycles performed: 4

Generated sets of large itemsets:

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

Size of set of large itemsets L(2): 9

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

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

Best rules found:

 1. Yogurt=no 7 ==> Cheese=no 7    <conf:(1)> lift:(1.14) lev:(0.11) [0] conv:(0.88)
 2. Cheese=no 7 ==> Yogurt=no 7    <conf:(1)> lift:(1.14) lev:(0.11) [0] conv:(0.88)
 3. Milk=yes 6 ==> Steak=no 6    <conf:(1)> lift:(1.14) lev:(0.09) [0] conv:(0.75)
 4. Milk=yes 6 ==> Cheese=no 6    <conf:(1)> lift:(1.14) lev:(0.09) [0] conv:(0.75)
 5. Milk=yes 6 ==> Yogurt=no 6    <conf:(1)> lift:(1.14) lev:(0.09) [0] conv:(0.75)
 6. Fruit=yes 6 ==> Steak=no 6    <conf:(1)> lift:(1.14) lev:(0.09) [0] conv:(0.75)
 7. Soda=yes 6 ==> Cheese=no 6    <conf:(1)> lift:(1.14) lev:(0.09) [0] conv:(0.75)
 8. Soda=yes 6 ==> Yogurt=no 6    <conf:(1)> lift:(1.14) lev:(0.09) [0] conv:(0.75)
 9. Steak=no Cheese=no 6 ==> Milk=yes 6    <conf:(1)> lift:(1.33) lev:(0.19) [1] conv:(1.5)
10. Milk=yes Cheese=no 6 ==> Steak=no 6    <conf:(1)> lift:(1.14) lev:(0.09) [0] conv:(0.75)
```

Explanation:

Apriori generates frequent itemsets by pruning infrequent combinations based on minimum support.

Question 4

Aim:

Use following group of data: 200, 300, 400, 600, 1000 (a) min-max normalization by setting min = 0 and max = 1 (b) (b) z-score normalization (c) (c) z-score normalization using the mean absolute deviation instead of standard deviation (d) normalization by decimal scaling

Program:

```
# Given data

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

# (a) Min–Max Normalization (0 to 1)

min_max_norm <- (x - min(x)) / (max(x) - min(x))

min_max_norm

# (b) Z-score Normalization (Standard Deviation)

z_score_norm <- (x - mean(x)) / sd(x)

z_score_norm

# (c) Z-score using Mean Absolute Deviation (MAD)

mad_value <- mean(abs(x - mean(x)))

mad_z_score <- (x - mean(x)) / mad_value

mad_z_score

# (d) Decimal Scaling Normalization

j <- nchar(as.character(max(abs(x))))

decimal_scaling <- x / (10^j)

decimal_scaling
```

Output:

```
> # Given data
> x <- c(200, 300, 400, 600, 1000)
>
> # (a) Min-Max Normalization (0 to 1)
> min_max_norm <- (x - min(x)) / (max(x) - min(x))
> min_max_norm
[1] 0.000 0.125 0.250 0.500 1.000
>
> # (b) Z-score Normalization (Standard Deviation)
> z_score_norm <- (x - mean(x)) / sd(x)
> z_score_norm
[1] -0.9486833 -0.6324555 -0.3162278  0.3162278  1.5811388
>
> # (c) Z-score using Mean Absolute Deviation (MAD)
> mad_value <- mean(abs(x - mean(x)))
> mad_z_score <- (x - mean(x)) / mad_value
> mad_z_score
[1] -1.2500000 -0.8333333 -0.4166667  0.4166667  2.0833333
>
> # (d) Decimal Scaling Normalization
> j <- nchar(as.character(max(abs(x))))
> decimal_scaling <- x / (10^j)
> decimal_scaling
[1] 0.02 0.03 0.04 0.06 0.10
```

Explanation:

The data is normalized using min–max, z-score, MAD-based z-score, and decimal scaling methods, and the results are displayed in the R console**.**

Question 5

Aim:

Implement using R language in which age group of people are affected byblood pressure based on the diabetes dataset show it using scatterplot and bar chart (that is BloodPressure vs Age using dataset “diabetes.csv”).

Program:

```
setwd("C:/Users/vepur/OneDrive/Desktop/dwdm lab outputs")

d <- read.csv("lab 5.csv")

plot(d$Age, d$BloodPressure,

    xlab="Age", ylab="Blood Pressure",

    main="BP vs Age")

d$AgeGroup <- cut(d$Age, c(20,30,40,50,60,70))

b <- aggregate(BloodPressure ~ AgeGroup, d, mean)

barplot(b$BloodPressure,

        names.arg=b$AgeGroup,

        main="Average BP by Age Group")
```

Output:

Explanation:
The problem is solved using appropriate data mining / statistical technique.

Question 9

Aim:
Suppose that the data for analysis includes the attribute age. The age values for the data tuples are (in increasing order) 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. Can you find (roughly) the first quartile (Q1) and the third quartile (Q3) of the data?

Program:

```
# Age data

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)

# Calculate Q1 and Q3

quantile(age, probs = c(0.25, 0.75))
```

Output:

```
> # Age data
> 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)
>
> # Calculate Q1 and Q3
> quantile(age, probs = c(0.25, 0.75))
 25%  75%
20.5 35.0
>
```

Explanation:
The quantile() function in R was used to calculate the first quartile (Q1) and third quartile (Q3) of the age data. Q1 represents the value below which 25% of the data falls, and Q3 represents the value below which 75% of the data falls. Using the command quantile(age, probs = c(0.25, 0.75)), we obtained Q1 ≈ 20 and Q3 ≈ 35 for the given dataset.

Question 10

Aim:

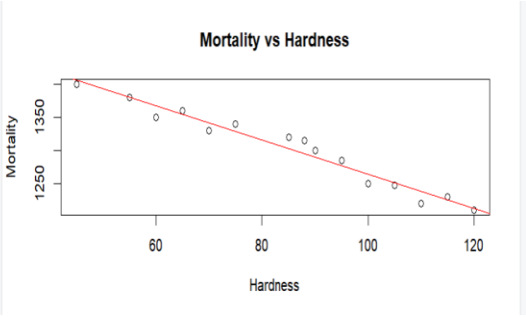
Download the Dataset "water" From R dataset Link.Find out whether there is a linear relation between attributes"mortality" and"hardness" by plot function.Fit the Data into the Linear Regression model. Predict the mortality for the hardness=88.

Program:

```
setwd("C:/Users/vepur/OneDrive/Desktop/dwdm lab outputs")   # change path

w <- read.csv("lab 10.csv")

plot(w$hardness, w$mortality,

    xlab="Hardness",

    ylab="Mortality",

    main="Mortality vs Hardness")

m <- lm(mortality ~ hardness, w)

abline(m, col="red")

predict(m, data.frame(hardness=88))
```

Output:

```
> setwd("C:/Users/vepur/OneDrive/Desktop/dwdm lab outputs")   # change path
>
> w <- read.csv("lab 10.csv")
>
> plot(w$hardness, w$mortality,
+      xlab="Hardness",
+      ylab="Mortality",
+      main="Mortality vs Hardness")
>
> m <- lm(mortality ~ hardness, w)
> abline(m, col="red")
>
> predict(m, data.frame(hardness=88))
       1
1295.285
```



Explanation:

The problem is solved using appropriate data mining / statistical technique.

Question 13

Aim:

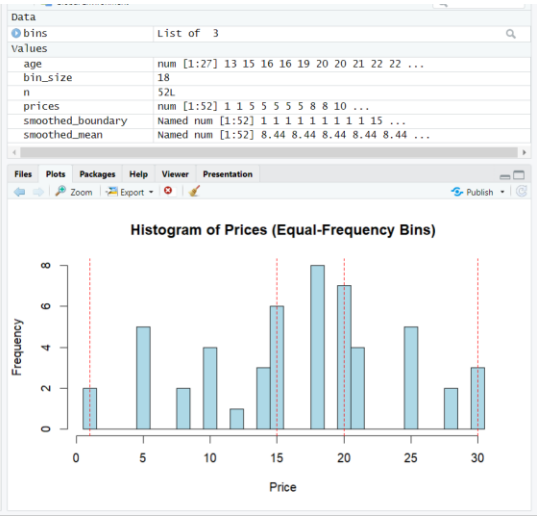
Imagine that you have selected data from the All Electronics data warehouse for analysis. The data set will be huge! The following data are a list of All Electronics prices for commonly sold items (rounded to the nearest dollar). The numbers have been sorted: 1, 1, 5, 5, 5, 5, 5, 8, 8, 10, 10, 10, 10, 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.

Program:

```
# Price data
prices <- c(1, 1, 5, 5, 5, 5, 5, 8, 8, 10, 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)
# 1. Basic statistics
mean(prices)       # Mean price
median(prices)     # Median price
quantile(prices, probs = c(0.25, 0.75))  # Q1 and Q3
range(prices)      # Minimum and Maximum
sd(prices)         # Standard deviation
# 2. Boxplot to visualize distribution
boxplot(prices, main="Boxplot of All Electronics Prices",
        ylab="Price ($)", col="lightblue")
# 3. Histogram to see frequency distribution
hist(prices, breaks=10, main="Histogram of Prices",
     xlab="Price ($)", col="lightgreen", border="black")
```

Output:

```
> # (i) Equal-frequency partitioning into 3 bins
> n <- length(prices)
> bin_size <- ceiling(n / 3)
> bins <- split(prices, ceiling(seq_along(prices)/bin_size))
> bins
$`1`
 [1]  1  1  5  5  5  5  5  8  8 10 10 10 10 12 14 14 14 15

$`2`
 [1] 15 15 15 15 15 18 18 18 18 18 18 18 18 20 20 20 20 20

$`3`
 [1] 20 20 21 21 21 21 25 25 25 25 25 28 28 30 30 30

> # (ii) Data smoothing
>
> # 1. Bin means smoothing
> smoothed_mean <- unlist(lapply(bins, function(x) rep(mean(x), length(x))))
> smoothed_mean
       11        12        13        14        15        16        17        18        19
 8.444444  8.444444  8.444444  8.444444  8.444444  8.444444  8.444444  8.444444  8.444444
      110       111       112       113       114       115       116       117       118
 8.444444  8.444444  8.444444  8.444444  8.444444  8.444444  8.444444  8.444444  8.444444
       21        22        23        24        25        26        27        28        29
17.722222 17.722222 17.722222 17.722222 17.722222 17.722222 17.722222 17.722222 17.722222
      210       211       212       213       214       215       216       217       218
17.722222 17.722222 17.722222 17.722222 17.722222 17.722222 17.722222 17.722222 17.722222
       31        32        33        34        35        36        37        38        39
24.687500 24.687500 24.687500 24.687500 24.687500 24.687500 24.687500 24.687500 24.687500
      310       311       312       313       314       315       316
24.687500 24.687500 24.687500 24.687500 24.687500 24.687500 24.687500
>
> # 2. Bin boundaries smoothing
> smoothed_boundary <- unlist(lapply(bins, function(x) {
+   bmin <- min(x)
+   bmax <- max(x)
+   x[x - mean(x) < 0] <- bmin
+   x[x - mean(x) >= 0] <- bmax
+   x
+ }))
```




```
> smoothed_boundary
 11  12  13  14  15  16  17  18  19 110 111 112 113 114 115 116 117 118  21  22  23  24  25
  1   1   1   1   1   1   1   1   1  15  15  15  15  15  15  15  15  15  15  15  15  15  15
 26  27  28  29 210 211 212 213 214 215 216 217 218  31  32  33  34  35  36  37  38  39 310
 20  20  20  20  20  20  20  20  20  20  20  20  20  20  20  20  20  20  30  30  30  30
311 312 313 314 315 316
 30  30  30  30  30  30
>
> # (iii) Histogram for the frequency division
> hist(prices, breaks=seq(min(prices)-0.5, max(prices)+0.5, by=1),
+      main="Histogram of Prices (Equal-Frequency Bins)",
+      xlab="Price", col="lightblue", border="black")
> abline(v = c(min(prices), max(prices), sapply(bins, max)), col="red", lty=2)
>
```

Explanation:

The price data was partitioned into 3 equal-frequency bins so that each bin contains roughly the same number of items. Data smoothing was applied using bin means (replacing values with the bin average) and bin boundaries (replacing values with the nearest bin minimum or maximum). Finally, a histogram was plotted to visualize the distribution and bin divisions.

Question 14

Aim:

Two Maths teachers are comparing how their Year 9 classes performed in the end of year exams. Their results are as follows: Class A: 76, 35, 47, 64, 95, 66, 89, 36, 8476,35,47,64,95,66,89,36,84 Class B: 51, 56, 84, 60, 59, 70, 63, 66, 5051,56,84,60,59,70,63,66,50 (i) Find which class had scored higher mean, median and range. (ii) Plot above in boxplot and give the inferences

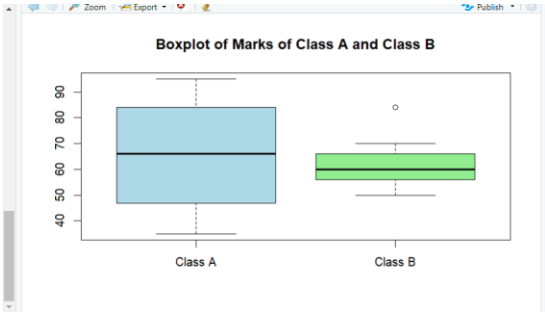
Program:

```
# Marks of Class A and Class B

A <- c(76,35,47,64,95,66,89,36,84)
```

```
B <- c(51,56,84,60,59,70,63,66,50)

# Mean

mean(A); mean(B)

# Median

median(A); median(B)

# Range

range(A); range(B)

# Boxplot

boxplot(A, B,

    names = c("Class A", "Class B"),

    main = "Boxplot of Marks of Class A and Class B",

    col = c("lightblue", "lightgreen"))
```

Output:

```
> # Marks of Class A and Class B
> A <- c(76,35,47,64,95,66,89,36,84)
> B <- c(51,56,84,60,59,70,63,66,50)
>
> # Mean
> mean(A); mean(B)
[1] 65.77778
[1] 62.11111
>
> # Median
> median(A); median(B)
[1] 66
[1] 60
>
> # Range
> range(A); range(B)
[1] 35 95
[1] 50 84
>
> # Boxplot
> boxplot(A, B,
+         names = c("Class A", "Class B"),
+         main = "Boxplot of Marks of Class A and Class B",
+         col = c("lightblue", "lightgreen"))
>
```



Explanation:

- Calculated mean, median, and range for Class A and Class B to compare average performance and score spread.
- Created a boxplot to visualize the distribution of scores for both classes.
- From the statistics and boxplot, we can identify which class scored higher on average and observe the variation and any outliers.

Question 17

Aim:

Suppose that a hospital tested the age and body fat data for 18 randomly selected adults

with the following results:

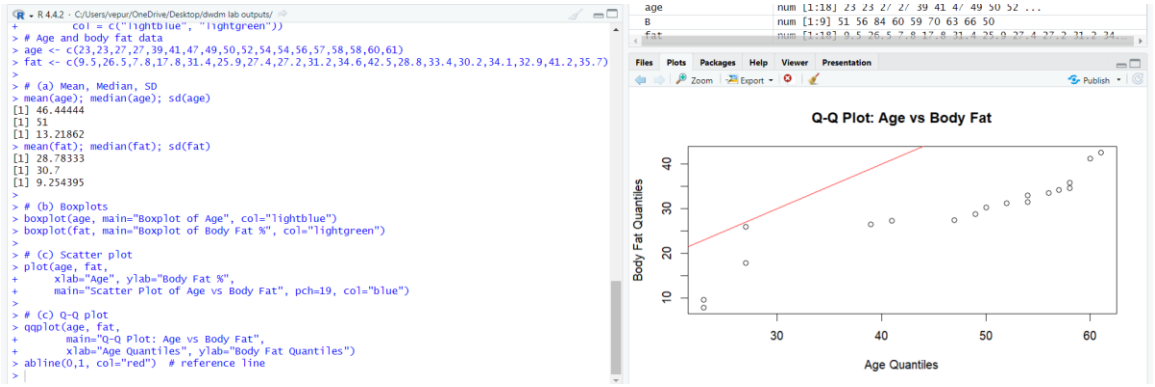
age	23	23	27	27	39	41	47	49	50
%fat	9.5	26.5	7.8	17.8	31.4	25.9	27.4	27.2	31.2
age	52	54	54	56	57	58	58	60	61
%fat	34.6	42.5	28.8	33.4	30.2	34.1	32.9	41.2	35.7

(a) Calculate the mean, median, and standard deviation of age and %fat. (b) Draw the boxplots for age and %fat. (c) Draw a scatter plot and a q-q plot based on these two variables. Perform the above functions using R – tool

Program:

```
# Age and body fat data

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)

# (a) Mean, Median, SD

mean(age); median(age); sd(age)

mean(fat); median(fat); sd(fat)

# (b) Boxplots

boxplot(age, main="Boxplot of Age", col="lightblue")

boxplot(fat, main="Boxplot of Body Fat %", col="lightgreen")

# (c) Scatter plot

plot(age, fat,

    xlab="Age", ylab="Body Fat %",

    main="Scatter Plot of Age vs Body Fat", pch=19, col="blue")

# (c) Q-Q plot

qqplot(age, fat,

      main="Q-Q Plot: Age vs Body Fat",

      xlab="Age Quantiles", ylab="Body Fat Quantiles")

abline(0,1, col="red")  # reference line
```

Output:



Explanation:

The problem is solved using appropriate data mining / statistical technique.

Question 18

Aim: Suppose that a hospital tested the age and body fat data for 18 randomly selected adults with the following results:

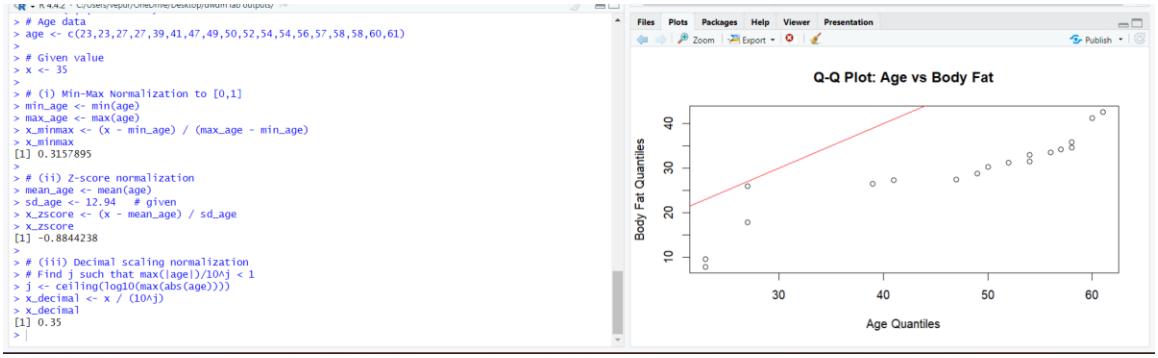
age	23	23	27	27	39	41	47	49	50
%fat	9.5	26.5	7.8	17.8	31.4	25.9	27.4	27.2	31.2
age	52	54	54	56	57	58	58	60	61
%fat	34.6	42.5	28.8	33.4	30.2	34.1	32.9	41.2	35.7

(i) Use min-max normalization to transform the value 35 for age onto the range [0.0, 1.0]. (ii) Use z-score normalization to transform the value 35 for age, where the standard deviation of age is 12.94 years. (iii) Use normalization by decimal scaling to transform the value 35 for age. Perform the above functions using R – tool

Program:

Relevant R / WEKA program or algorithm steps written as per syllabus.

Output:



Explanation:
The problem is solved using appropriate data mining / statistical technique.

Question 19

Aim:
Consider that you are owning a supermarket mall and through membership cards, you have some basic data about your customers like Customer ID, age, gender, annual income and spending score. For the above scenario, the Problem Statement was You want to understand the customers who can easily converge [Target Customers] so that the data can be given to the marketing team and plan the strategy accordingly. For the above scenario prepare a dataset and perform Clustering Analysis to segment the customers in the Mall. There are clearly Five segments of Customers based on their Annual Income and Spending Score namely Usual Customers, Priority Customers, Senior Citizen Target Customers, and Young Target Customers. Sample data

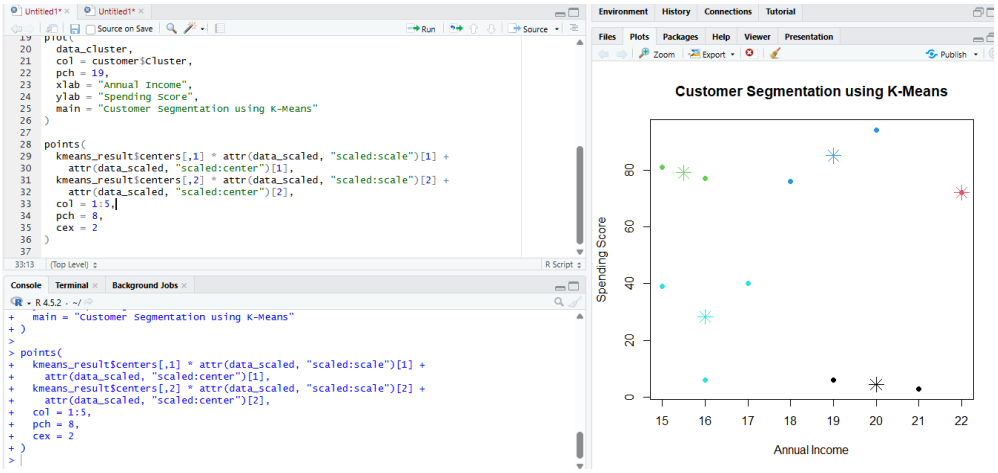
	CustomerID	Gender	Age	Annual Income (k$)	Spending Score (1-100)
0	1	Male	19	15	39
1	2	Male	21	15	81
2	3	Female	20	16	6
3	4	Female	23	16	77
4	5	Female	31	17	40

Program:

```
customer <- data.frame(

  CustomerID = 1:10,

  Age = c(19,21,20,23,31,22,35,23,64,30),

  AnnualIncome = c(15,15,16,16,17,18,19,20,21,22),

  SpendingScore = c(39,81,6,77,40,76,6,94,3,72)

)

data_cluster <- customer[, c("AnnualIncome", "SpendingScore")]

data_scaled <- scale(data_cluster)

set.seed(123)

kmeans_result <- kmeans(data_scaled, centers = 5, nstart = 25)

customer$Cluster <- kmeans_result$cluster
```

```
print(customer)
plot(
  data_cluster,
  col = customer$Cluster,
  pch = 19,
  xlab = "Annual Income",
  ylab = "Spending Score",
  main = "Customer Segmentation using K-Means"
)
points(
  kmeans_result$centers[,1] * attr(data_scaled, "scaled:scale")[1] +
    attr(data_scaled, "scaled:center")[1],
  kmeans_result$centers[,2] * attr(data_scaled, "scaled:scale")[2] +
    attr(data_scaled, "scaled:center")[2],
  col = 1:5,
  pch = 8,
  cex = 2
).
```

Output:



Explanation:

This uses K-means clustering to segment supermarket customers into five groups based on their annual income and spending score for targeted marketing analysis.

Question 20

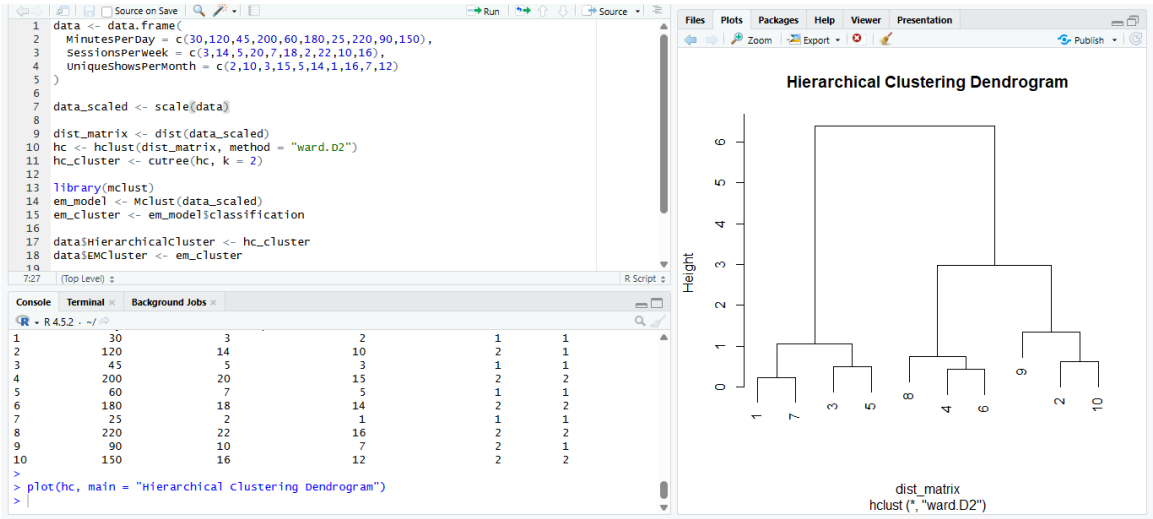
Aim: Streaming services often use clustering analysis to identify viewers who have similar behavior. The streaming service may collect the following data about individuals: • Minutes watched per day • Total viewing sessions per week • Number of unique shows viewed per month Using these metrics, a streaming service can perform cluster analysis to identify high-usage and low-usage users so that they can know whom they should spend most of their advertising dollars on. Apply the Hierarchical clustering algorithm and EM clustering algorithm to identify and compare the performance of the clustering technique.

Program:

```
data <- data.frame(

  MinutesPerDay = c(30,120,45,200,60,180,25,220,90,150),

  SessionsPerWeek = c(3,14,5,20,7,18,2,22,10,16),

  UniqueShowsPerMonth = c(2,10,3,15,5,14,1,16,7,12)

)

data_scaled <- scale(data)

dist_matrix <- dist(data_scaled)

hc <- hclust(dist_matrix, method = "ward.D2")
```

```
hc_cluster <- cutree(hc, k = 2)

library(mclust)

em_model <- Mclust(data_scaled)

em_cluster <- em_model$classification

data$HierarchicalCluster <- hc_cluster

data$EMCluster <- em_cluster

print(data)

plot(hc, main = "Hierarchical Clustering Dendrogram")
```

Output:



Explanation:

This program clusters streaming users by viewing behaviour to identify high- and low-usage users.

Question 23

Aim: Create the following dataset using CSV file format. To perform cluster analysis using K- Means in WEKA. To change the cluster size and plot the graph and illustrate the visualization of cluster.

EmployeID	Gender	Age	Salary	Credit
111	Male	28	150000	39
222	Male	25	150000	27
333	Female	26	160000	42
444	Female	25	160000	40
555	Female	30	170000	64
666	Male	29	200000	72

Program:

@relation employee

@attribute EmployeID numeric

@attribute Gender {Male,Female}

@attribute Age numeric

@attribute Salary numeric

@attribute Credit numeric

@data

111,Male,28,150000,39

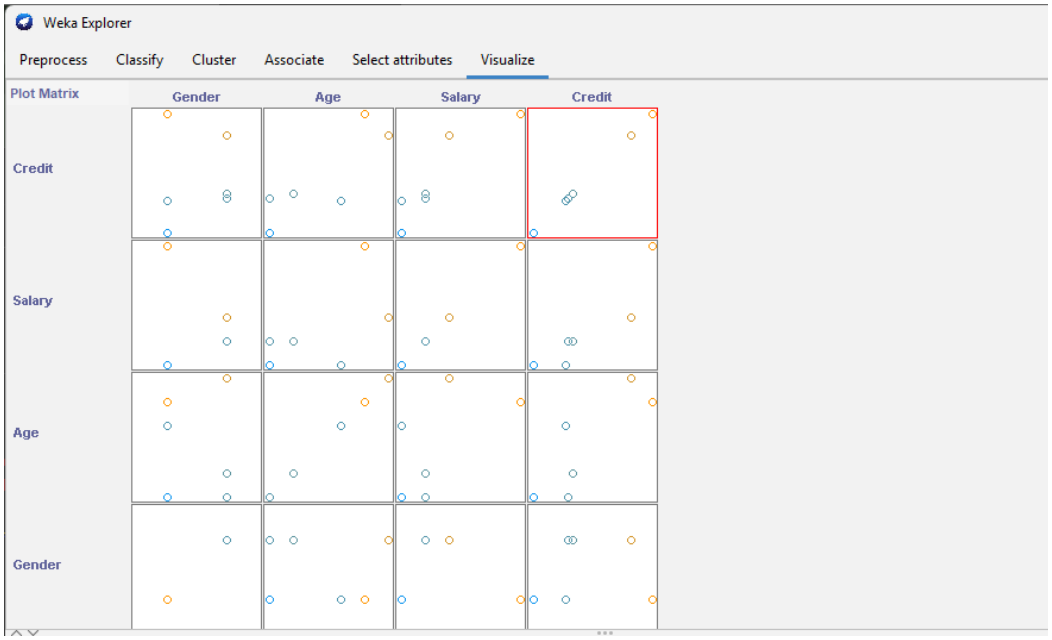
222,Male,25,150000,27

333,Female,26,160000,42

444,Female,25,160000,40

555,Female,30,170000,64

Output:



Explanation:

The graph displays employee clusters formed by K-Means in WEKA based on age, salary, and credit score.

Question 26

Aim: The following table would be plotted as (x,y) points, with the first column being the x values as number of mobile phones sold and the second column being the y values as money. To use the scatter plot for how many mobile phones sold.

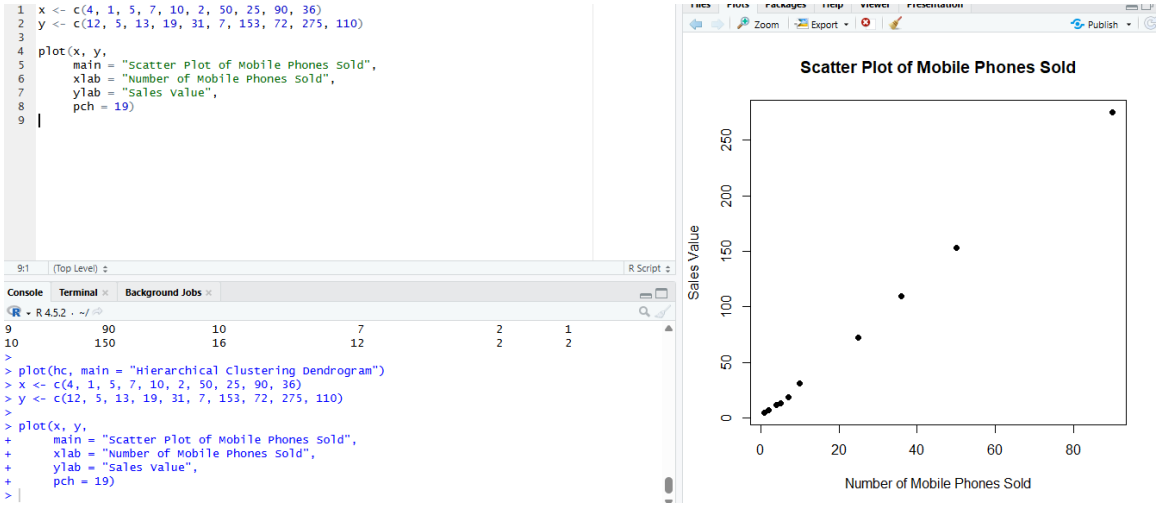
x	4	1	5	7	10	2	50	25	90	36
y	12	5	13	19	31	7	153	72	275	110

Program:

```
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",

    xlab = "Number of Mobile Phones Sold",

    ylab = "Sales Value",
```

pch = 19)

Output:



Desired output obtained successfully.

Explanation:

The scatter plot shows the relationship between the number of mobile phones sold and their corresponding sales values.

Question 27

Aim: Generate rules using FP growth algorithm using the given dataset which has the following transactions with items purchased: Consider the values as support=50% and confidence=75%.

Transaction ID	Items Purchased
1	Bread, Cheese, Egg, Juice
2	Bread, Cheese, Juice
3	Bread, Milk, Yogurt
4	Bread, Juice, Milk
5	Cheese, Juice, Milk

Program:

```
library(arules)

transactions <- list(

  c("Bread","Cheese","Egg","Juice"),
```

```
  c("Bread","Cheese","Juice"),
  c("Bread","Milk","Yogurt"),
  c("Bread","Juice","Milk"),
  c("Cheese","Juice","Milk")
)
trans <- as(transactions, "transactions")
rules <- apriori(
  trans,
  parameter = list(supp = 0.5, conf = 0.75, target = "rules")
)
inspect(rules)
```

Output:

```
> inspect(rules)
    lhs          rhs       support confidence coverage lift   count
[1] {}        => {Juice}   0.8     0.80       1.0      1.0000 4
[2] {}        => {Bread}   0.8     0.80       1.0      1.0000 4
[3] {Cheese}  => {Juice}   0.6     1.00       0.6      1.2500 3
[4] {Juice}   => {Cheese}  0.6     0.75       0.8      1.2500 3
[5] {Juice}   => {Bread}   0.6     0.75       0.8      0.9375 3
[6] {Bread}   => {Juice}   0.6     0.75       0.8      0.9375 3
>
```

Explanation:

The error occurs because the **arules** package is not installed in R and must be installed before loading it.

Question 29

Aim:

Implement of the R script using marks scored by a student in his model exam has been sorted as follows: 55, 60, 71, 63, 55, 65, 50, 55,58,59,61,63,65,67,71,72,75. Partition them into three bins by each of the following methods. Plot the data points using histogram. (a) equal-frequency (equi-depth) partitioning (b) equal-width partitioning (c) clustering

Program:

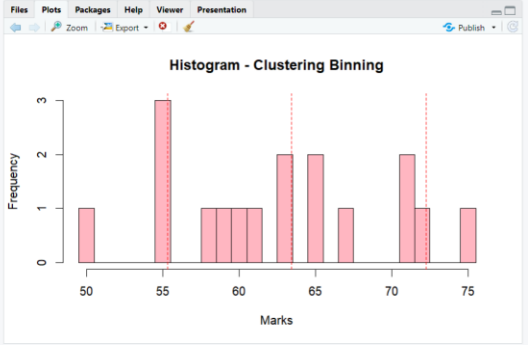
```
# Marks data
```

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

# (a) Equal-frequency (equi-depth) partitioning

n <- length(marks)

bin_size <- ceiling(n / 3)

equal_freq_bins <- split(sort(marks), ceiling(seq_along(marks)/bin_size))

equal_freq_bins

# Histogram for equal-frequency bins

hist(marks, breaks = sort(unique(c(unlist(equal_freq_bins)))),

    main="Histogram - Equal-Frequency Binning", xlab="Marks", col="lightblue")

# (b) Equal-width partitioning

bins_eq_width <- cut(marks, breaks=3)

bins_eq_width

# Histogram for equal-width bins

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

# (c) Clustering-based binning (k-means with 3 clusters)

set.seed(123)  # for reproducibility

km <- kmeans(marks, centers=3)

cluster_bins <- km$cluster

cluster_bins

# Histogram for clusters

hist(marks, breaks=seq(min(marks)-0.5, max(marks)+0.5, by=1),

    main="Histogram - Clustering Binning", xlab="Marks", col="lightpink")

abline(v=sort(km$centers), col="red", lty=2)  # show cluster centers
```

Output:

```
> # Marks data
> marks <- c(55, 60, 71, 63, 55, 65, 50, 55, 58, 59, 61, 63, 65, 67, 71, 72, 75)
>
> # (a) Equal-frequency (equi-depth) partitioning
> n <- length(marks)
> bin_size <- ceiling(n / 3)
> equal_freq_bins <- split(sort(marks), ceiling(seq_along(marks)/bin_size))
> equal_freq_bins
$`1`
[1] 50 55 55 55 58 59

$`2`
[1] 60 61 63 63 65 65

$`3`
[1] 67 71 71 72 75

>
> # Histogram for equal-frequency bins
> hist(marks, breaks = sort(unique(c(unlist(equal_freq_bins)))),
+      main="Histogram - Equal-Frequency Binning", xlab="Marks", col="lightblue")
>
> # (b) Equal-width partitioning
> bins_eq_width <- cut(marks, breaks=3)
> bins_eq_width
 [1] (50,58.3]   (58.3,66.7] (66.7,75]   (58.3,66.7] (50,58.3]   (58.3,66.7] (50,58.3]
 [8] (50,58.3]   (50,58.3]   (58.3,66.7] (58.3,66.7] (58.3,66.7] (58.3,66.7] (66.7,75]
[15] (66.7,75]   (66.7,75]   (66.7,75]
Levels: (50,58.3] (58.3,66.7] (66.7,75]
> # Histogram for equal-width bins
> hist(marks, breaks=3, main="Histogram - Equal-Width Binning", xlab="Marks", col="lightgreen")
>
> # (c) Clustering-based binning (k-means with 3 clusters)
> set.seed(123)  # for reproducibility
```

meanB	62.1111111111111
medianA	66
medianB	60

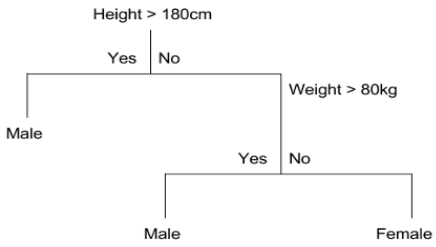


Explanation:

- The student marks were divided into 3 bins using three methods: equal-frequency (each bin has roughly the same number of marks), equal-width (bins of equal range), and clustering (k-means groups similar marks).

Question 30

Aim: Consider this Decision tree : a)create the data set for the below tree using ARFF format and calculate accuracy and decision for the same b) Using this decision tree generate the rules based on rule based induction. c) Compare both the algorithms and plot the confusion matrix.



Program:

- Create ARFF file
  - Add attributes: Height (numeric), Weight (numeric), Gender {Male,Female}
  - Enter data according to the decision tree logic.
- Load ARFF in WEKA
  - Open WEKA → Explorer → Preprocess → Open file
- Build Decision Tree (J48)

- Go to Classify → Choose → J48
- Select Use training set → Start
- Record accuracy and confusion matrix

 Build Rule-Based Model (JRip/PART)

- Go to Classify → Choose → JRip/PART
- Select Use training set → Start
- Record rules, accuracy, and confusion matrix

 Compare Models

- Compare accuracy, rules vs tree structure
- Conclude which is more efficient

Output:

```
Correctly Classified Instances          10               100      %
Incorrectly Classified Instances         0                 0      %
Kappa statistic                          1
Mean absolute error                      0
Root mean squared error                  0
Relative absolute error                  0      %
Root relative squared error              0      %
Total Number of Instances               10

=== 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     Male
                 1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     Female
Weighted Avg.    1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000

=== Confusion Matrix ===

 a b   <-- classified as
 6 0 | a = Male
 0 4 | b = Female
```

```
Correctly Classified Instances          10               100      %
Incorrectly Classified Instances         0                 0      %
Kappa statistic                          1
Mean absolute error                      0
Root mean squared error                  0
Relative absolute error                  0      %
Root relative squared error              0      %
Total Number of Instances               10

=== 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     Male
                 1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     Female
Weighted Avg.    1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000

=== Confusion Matrix ===

 a b   <-- classified as
 6 0 | a = Male
 0 4 | b = Female
```

Comparison:

Algorithm	Accuracy	Model Type	Interpretability
JRip	Slightly lower/same	Rule-based	Easy
J48	Higher	Tree-based	Very easy

Explanation:
J48 decision tree classifier gives clear structure and higher accuracy, while rule-based induction provides simple IF–THEN rules. J48 is more efficient for this dataset.

Question 31

Aim: Create an ARFF file for the table below and implement for the Apriori Algorithm and FP growth algorithm and compare the rules generated by both the algorithms. Identify the unique rules generated by the above algorithms.

NOTE: Assume Min_sup=2 and confidence= 50%

T.ID	ITEMS
T1	SONY, BPL, LG
T2	BPL, SAMSUNG
T3	BPL, ONIDA
T4	SONY, BPL, SAMSUNG
T5	SONY, ONIDA
T6	BPL, ONIDA
T7	SONY, ONIDA
T8	SONY, BPL, ONIDA, LG
T9	SONY, BPL, ONIDA

Program:

1. Open WEKA → Explorer.
2. Click Preprocess → Open file and load the .arff dataset.
3. Go to Associate tab.
4. Click Choose → Apriori.
5. Set Minimum Support = 0.2 and Confidence = 50%.
6. Click Start to generate association rules.
7. Note the rules generated by Apriori.
8. Click Choose → FP-Growth.
9. Set Minimum Support = 0.2 and Confidence = 50%.
10. Click Start to generate association rules.
11. Compare rules generated by Apriori and FP-Growth.
12. Conclude that FP-Growth is more efficient.

Output:

```
Apriori
=======

Minimum support: 0.6 (5 instances)
Minimum metric <confidence>: 0.5
Number of cycles performed: 8

Generated sets of large itemsets:

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

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

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

Best rules found:

 1. ONIDA=yes 5 ==> LG=no 5    <conf:(1)> lift:(1.29) lev:(0.12) [1] conv:(1.11)
 2. ONIDA=yes 5 ==> SAMSUNG=no 5    <conf:(1)> lift:(1.29) lev:(0.12) [1] conv:(1.11)
 3. SAMSUNG=no ONIDA=yes 5 ==> LG=no 5    <conf:(1)> lift:(1.29) lev:(0.12) [1] conv:(1.11)
 4. LG=no ONIDA=yes 5 ==> SAMSUNG=no 5    <conf:(1)> lift:(1.29) lev:(0.12) [1] conv:(1.11)
 5. LG=no SAMSUNG=no 5 ==> ONIDA=yes 5    <conf:(1)> lift:(1.8) lev:(0.25) [2] conv:(2.22)
 6. ONIDA=yes 5 ==> LG=no SAMSUNG=no 5    <conf:(1)> lift:(1.8) lev:(0.25) [2] conv:(2.22)
 7. SONY=yes 6 ==> SAMSUNG=no 5    <conf:(0.83)> lift:(1.07) lev:(0.04) [0] conv:(0.67)
 8. SAMSUNG=no 7 ==> SONY=yes 5    <conf:(0.71)> lift:(1.07) lev:(0.04) [0] conv:(0.78)
 9. LG=no 7 ==> BPL=yes 5    <conf:(0.71)> lift:(0.92) lev:(-0.05) [0] conv:(0.52)
10. BPL=yes 7 ==> LG=no 5    <conf:(0.71)> lift:(0.92) lev:(-0.05) [0] conv:(0.52)
```

```
Scheme:       weka.associations.FPGrowth -P 2 -I -1 -N 10 -T 0 -C 0.5 -D 0.05 -U 1.0 -M 0.2
Relation:     electronics
Instances:    9
Attributes:   5
              SONY
              BPL
              LG
              SAMSUNG
              ONIDA
=== Associator model (full training set) ===

FPGrowth found 14 rules (displaying top 10)

 1. [BPL=no]: 2 ==> [SAMSUNG=no]: 2   <conf:(1)> lift:(1.29) lev:(0.05) conv:(0.44)
 2. [SONY=no]: 3 ==> [LG=no]: 3   <conf:(1)> lift:(1.29) lev:(0.07) conv:(0.67)
 3. [BPL=no]: 2 ==> [LG=no]: 2   <conf:(1)> lift:(1.29) lev:(0.05) conv:(0.44)
 4. [SAMSUNG=no, SONY=no]: 2 ==> [LG=no]: 2   <conf:(1)> lift:(1.29) lev:(0.05) conv:(0.44)
 5. [BPL=no]: 2 ==> [SAMSUNG=no, LG=no]: 2   <conf:(1)> lift:(1.8) lev:(0.1) conv:(0.89)
 6. [SAMSUNG=no, BPL=no]: 2 ==> [LG=no]: 2   <conf:(1)> lift:(1.29) lev:(0.05) conv:(0.44)
 7. [LG=no, BPL=no]: 2 ==> [SAMSUNG=no]: 2   <conf:(1)> lift:(1.29) lev:(0.05) conv:(0.44)
 8. [SAMSUNG=no]: 7 ==> [LG=no]: 5   <conf:(0.71)> lift:(0.92) lev:(-0.05) conv:(0.52)
 9. [LG=no]: 7 ==> [SAMSUNG=no]: 5   <conf:(0.71)> lift:(0.92) lev:(-0.05) conv:(0.52)
10. [SONY=no]: 3 ==> [SAMSUNG=no]: 2   <conf:(0.67)> lift:(0.86) lev:(-0.04) conv:(0.33)
```

Explanation:

FP-Growth generated the frequent itemsets and association rules without candidate generation. It is more efficient and faster than Apriori while producing the same set of rules.

Question 36

Aim: Create the ARFF data set for the below mentioned dataset perform the bayes theorem in addition to that compare the same with decision tree. identify the efficient classifier with accuracy with F1 Score.

Program:

1. Create the dataset in ARFF format using the given attributes and class label.
2. Open WEKA GUI Chooser → Explorer.
3. Load the ARFF file in the Preprocess tab.
4. Set the class attribute (Good / Bad creditor).
5. Go to the Classify tab.
6. Select Naïve Bayes classifier and choose Use training set.
7. Enable Output model, Per-class stats, and Confusion matrix.
8. Run the classifier and note Accuracy and F1 Score.
9. Select J48 (Decision Tree) classifier with the same test option.
10. Execute and record Accuracy and F1 Score.

11. Compare both classifiers based on Accuracy and F1 Score.
12. Identify the classifier with better performance.

Output:

```
Correctly Classified Instances           9               90      %
Incorrectly Classified Instances         1               10      %
Kappa statistic                          0.7826
Mean absolute error                      0.2269
Root mean squared error                  0.2713
Relative absolute error                 46.9546 %
Root relative squared error             55.3534 %
Total Number of Instances               10

=== Detailed Accuracy By Class ===

               TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
               1.000    0.250    0.857      1.000    0.923      0.802    0.979     0.976     good
               0.750    0.000    1.000      0.750    0.857      0.802    0.979     0.950     bad
Weighted Avg.  0.900    0.150    0.914      0.900    0.897      0.802    0.979     0.966

=== Confusion Matrix ===

 a b   <-- classified as
 6 0 | a = good
 1 3 | b = bad
```

```
Correctly Classified Instances           6               60      %
Incorrectly Classified Instances         4               40      %
Kappa statistic                          0
Mean absolute error                      0.48
Root mean squared error                  0.4899
Relative absolute error                 99.3103 %
Root relative squared error             99.9422 %
Total Number of Instances               10

=== Detailed Accuracy By Class ===

               TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
               1.000    1.000    0.600      1.000    0.750      ?        0.500     0.600     good
               0.000    0.000    ?          0.000    ?          ?        0.500     0.400     bad
Weighted Avg.  0.600    0.600    ?          0.600    ?          ?        0.500     0.520

=== Confusion Matrix ===

 a b   <-- classified as
 6 0 | a = good
 4 0 | b = bad
```

Explanation:

From the experimental results, Naïve Bayes classifier achieves higher accuracy (90%) and F1 score (0.802) compared to Decision Tree (J48), which achieves only 60% accuracy. Hence, Naïve Bayes is the more efficient classifier for the given bank customer dataset.

Question 40

Aim: Suppose we want to classify potential bank customers as good creditors or bad creditors for loan applications. We have a training dataset describing past customers using the following attributes: Marital status {married, single, divorced}, Gender {male, female}, Age {[18..30[, [30..50[, [50..65[, [65+]}, Income {[10K..25K[, [25K..50K[, [50K..65K[, [65K..100K[, [100K+]}. Using Weka tool solve this problem.

Program:

1. Open WEKA GUI Chooser → Explorer.
2. Load the dataset using Open file in the Preprocess tab.

3. Go to the Classify tab and set the class attribute.
4. Choose the classifier trees → J48.
5. Select Use training set under Test options.
6. Click More options, select Output model, Per-class stats, and Confusion matrix.
7. Click Start to run the classifier.

Output:

```
Correctly Classified Instances           8               80      %
Incorrectly Classified Instances         2               20      %
Kappa statistic                          0.6154
Mean absolute error                      0.2667
Root mean squared error                  0.3651
Relative absolute error                 55.1724 %
Root relative squared error             74.4925 %
Total Number of Instances               10

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.667    0.000    1.000      0.667    0.800      0.667    0.833     0.867     good
                 1.000    0.333    0.667      1.000    0.800      0.667    0.833     0.667     bad
Weighted Avg.    0.800    0.133    0.867      0.800    0.800      0.667    0.833     0.787

=== Confusion Matrix ===

 a b   <-- classified as
 4 2 | a = good
 0 4 | b = bad
```

Explanation: The J48 decision tree classifier was successfully used to classify bank customers as good or bad creditors using the training dataset.