Experiment 1.3

1. Aim:
Demonstration of association rule mining using Apriori algorithm

2. Objective:
To finds interesting associations and relationships among large sets of data items.

3. Script and Output:

Apriori algorithm is a popular algorithm used in association rule mining that scans datasets to identify frequent itemsets and generate association rules. It employs a bottom-up approach by generating candidate sets and iteratively pruning infrequent itemsets. It is widely used in market basket analysis and recommendation systems.

Association rule is a pattern or relationship between variables in a dataset that are often co-occurring. It is a type of unsupervised learning that discovers hidden associations and correlations among the data. Association rules are used in various applications such as market basket analysis, customer segmentation, and recommendation systems.

CODE-

```
#Saumyamani Bhardwaz
#20BCS1682

# Apriori algorithm

library(arules)              #association rules
library(arulesViz)           #association rules visualization
library(RColorBrewer)        #just for the sake of colors

# import data set
data("Income")

# using apriori() function
# list contaning support and confidence values
rules <- apriori(Income, parameter = list(supp = 0.01, conf = 0.2))

# using inspect() function
inspect(rules[1:10])

# using itemFrequencyPlot() function
arules::itemFrequencyPlot(Income, topN = 5,
                          col = brewer.pal(8, 'Set2'),
                          main = 'Relative Item Frequency Plot',
                          type = "relative",
                          ylab = "Item Frequency (Relative)")
```

OUTPUT-

```
R  R 4.2.2 · D:/Sem-6/Data Mining & R/Exp3/
> source("D:/Sem-6/Data Mining & R/Exp3/ScriptClasswork.R", echo=TRUE)

> #Saumyamani Bhardwaz
> #20BCS1682
>
> # Apriori algorithm
>
> library(arules)              #association rules

> library(arulesViz)           #association rules visualization

> library(RColorBrewer)        #just for the sake of colors

> # import data set
> data("Income")

> # using apriori() function
> # list contaning support and confidence values
> rules <- apriori(Income, parameter = list(supp = 0.01, conf = 0.2))
Apriori

Parameter specification:
 confidence minval smax arem  aval originalSupport maxtime support minlen maxlen target   ext
        0.2    0.1     1 none FALSE             TRUE       5    0.01      1     10  rules TRUE

Algorithmic control:
 filter tree heap memopt load sort verbose
    0.1 TRUE TRUE  FALSE TRUE    2    TRUE

Absolute minimum support count: 68
```

```
set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[50 item(s), 6876 transaction(s)] done [0.00s].
sorting and recoding items ... [49 item(s)] done [0.00s].
creating transaction tree ... done [0.00s].
checking subsets of size 1 2 3 4 5 6 7 8 9 10 done [1.00s].
writing ... [1724743 rule(s)] done [1.17s].
creating S4 object  ... done [0.83s].

> # using inspect() function
> inspect(rules[1:10])
     lhs      rhs                                       support   confidence coverage lift
[1]  {}  => {householder status=live with parents/family} 0.2050611 0.2050611 1        1
[2]  {}  => {dual incomes=yes}                            0.2533450 0.2533450 1        1
[3]  {}  => {type of home=apartment}                      0.2755963 0.2755963 1        1
[4]  {}  => {education=college graduate}                  0.2947935 0.2947935 1        1
[5]  {}  => {number in household=2+}                      0.3081734 0.3081734 1        1
[6]  {}  => {occupation=professional/managerial}          0.3392961 0.3392961 1        1
[7]  {}  => {years in bay area=1-9}                       0.3534031 0.3534031 1        1
[8]  {}  => {householder status=own}                      0.3757999 0.3757999 1        1
[9]  {}  => {income=$40,000+}                             0.3775451 0.3775451 1        1
[10] {}  => {number of children=1+}                       0.3781268 0.3781268 1        1
     count
[1]  1410
[2]  1742
[3]  1895
[4]  2027
[5]  2119
[6]  2333
[7]  2430
[8]  2584
[9]  2596
[10] 2600

> # using itemFrequencyPlot() function
> arules::itemFrequencyPlot(Income, topN = 5,
+                          col = brewer.pal(8, 'Set2'),
+       .... [TRUNCATED]
```

Relative Item Frequency Plot