**CS 634**

****

**MIDTERM PROJECT**

**Topic:**

**APRIORI ALGORITHM IMPLEMENTATION**

**Instructor:**

**Dr. Jason Wang**

**Submitted By:**

**IKJYOT SINGH GUJRAL**

**(isg6)**

Definition

The Apriori Algorithm is an influential algorithm for mining frequent itemsets for Boolean association rules. Apriori uses a "bottom up" approach, where frequent subsets are extended one item at a time (a step known as candidate generation, and groups of candidates are tested against the data. Apriori is designed to operate on database containing transactions (for example, collections of items bought by customers, or details of a website frequentation).

The Algorithm

**STEP 1:** Scan the transaction data base to get the support of S each 1-itemset, compare S with min\_sup, and get a support of 1-itemsets, L1

**STEP 2:** Use 𝐿 𝑘−1 join 𝐿 𝑘−1 to generate a set of candidate k-itemsets. And use Apriori property to prune the unfrequented k-itemsets from this set

**STEP 3:** Scan the transaction database to get the support S of each candidate k-itemset in the find set, compare S with min\_sup, and get a set of frequent k-itemsets 𝐿 𝑘

**STEP 4:** The candidate set = Null

**STEP 5:** For each frequent itemset 1, generate all nonempty subsets of 1

**STEP 6:** For every nonempty subset s of 1, output the rule “s=>(1-s)” if confidence C of the rule “s=>(1-s)” (=support s of 1/support S of s)’ min\_conf NO YES

Limitations

Apriori algorithm can be very slow and the bottleneck is candidate generation.

For example, if the transaction DB has 104 frequent 1-itemsets, they will generate 107 candidate 2-itemsets even after employing the downward closure.

To compute those with sup more than min sup, the database need to be scanned at every level. It needs (n +1 ) scans, where n is the length of the longest pattern.

Methods to Improve Apriori’s Efficiency

• Hash-based itemset counting: A k-itemset whose corresponding hashing bucket count is below the threshold cannot be frequent

• Transaction reduction: A transaction that does not contain any frequent k-itemset is useless in subsequent scans

• Partitioning: Any itemset that is potentially frequent in DB must be frequent in at least one of the partitions of DB.

• Sampling: mining on a subset of given data, lower support threshold + a method to determine the completeness

• Dynamic itemset counting: add new candidate itemsets only when all of their subsets are estimated to be frequent

Apriori Advantages/Disadvantages

**Advantages**

• Uses large itemset property • Easily parallelized • Easy to implement

**Disadvantages**

• Assumes transaction database is memory resident. • Requires many database scans

Implementation Details

In this project, The Apriori Algorithms has been implemented using Python Programming Language. The reason to choose Python is its simplicity, readability and flexibility. Python allows the user to implement complex logics using simple codes.

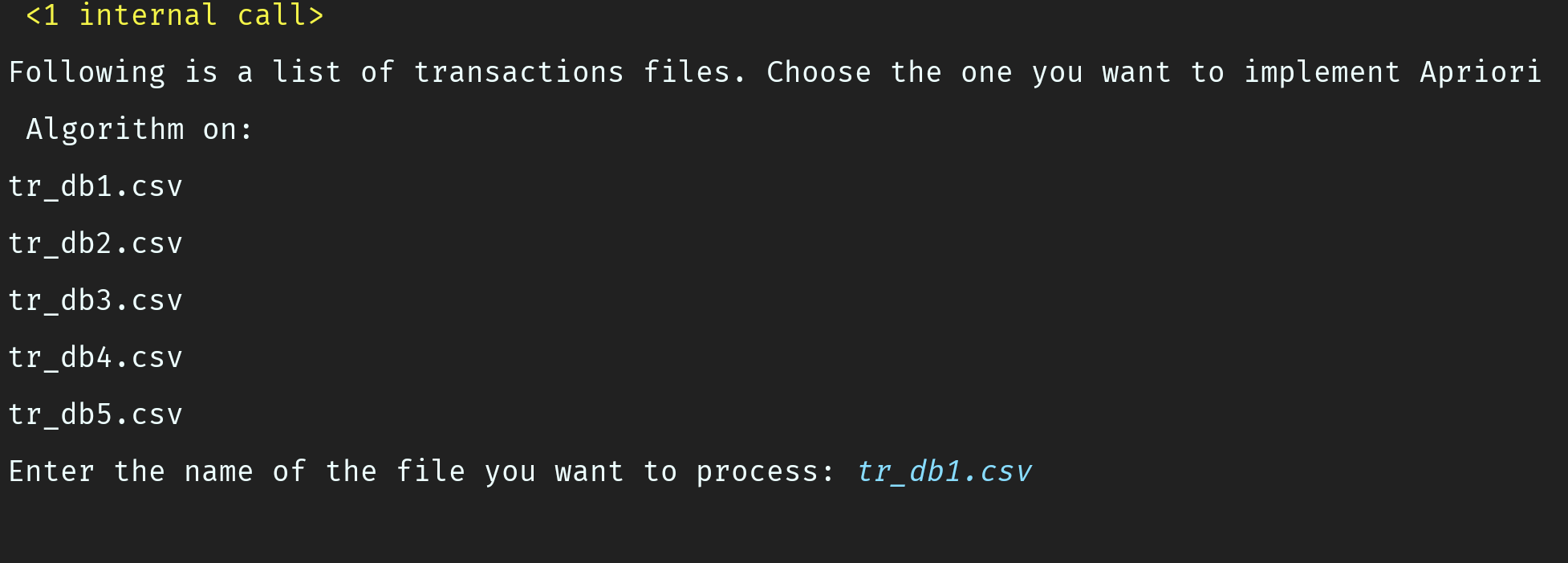
The entire project has been divided into two files:

1. The first file is responsible to generate transaction .csv files out of .txt files containing list of 10 items. There are 5 .csv files that are generated, each containing 20 transactions.
2. The second file itself is divided in two code blocks:
   1. Generate frequent itemsets for a database of 20 transactions based on the user defined support value.
   2. Generate Association rules out of the frequent itemsets based on the user defined confidence value.

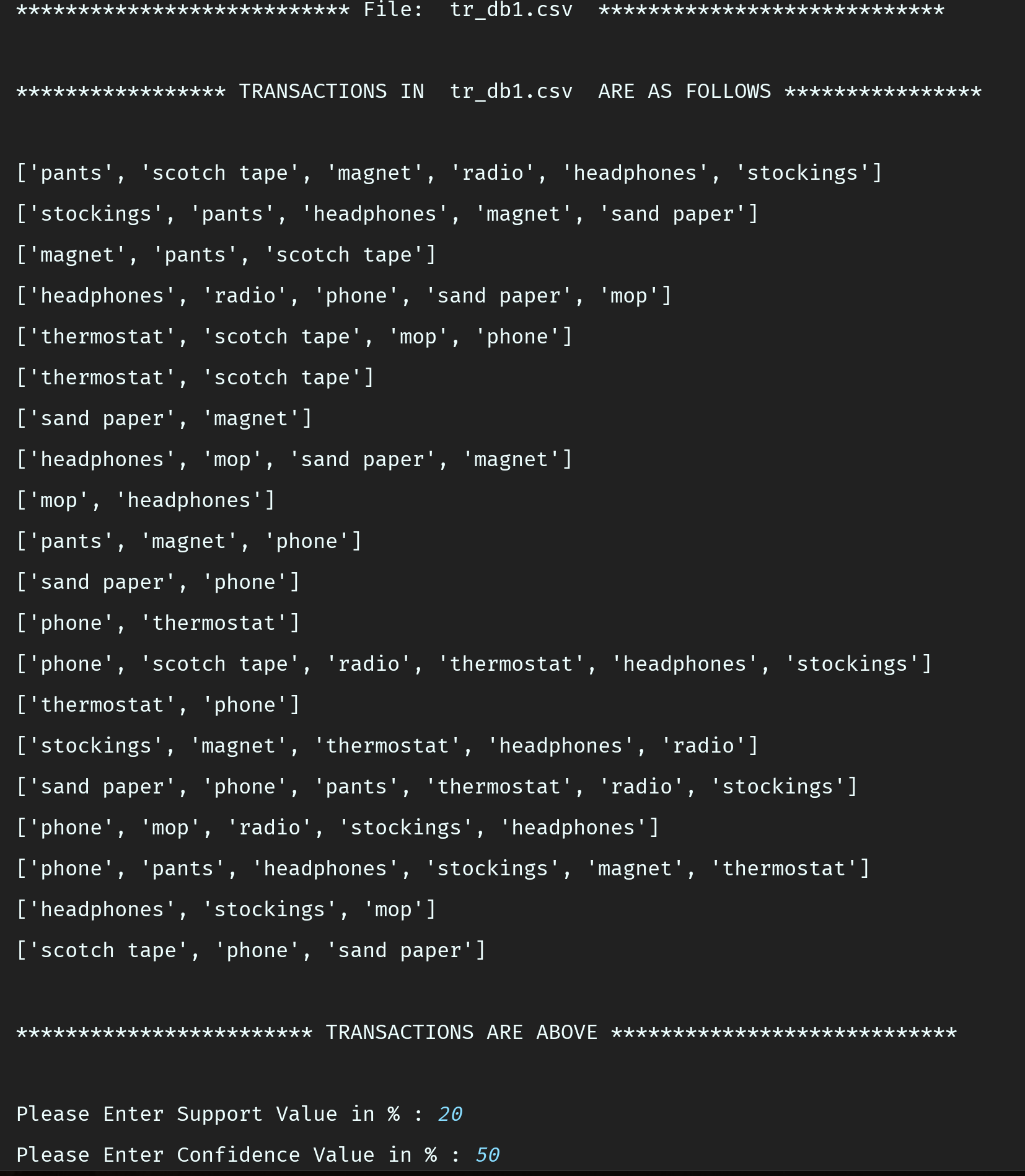
**Source Code: generate.py**

Procedure

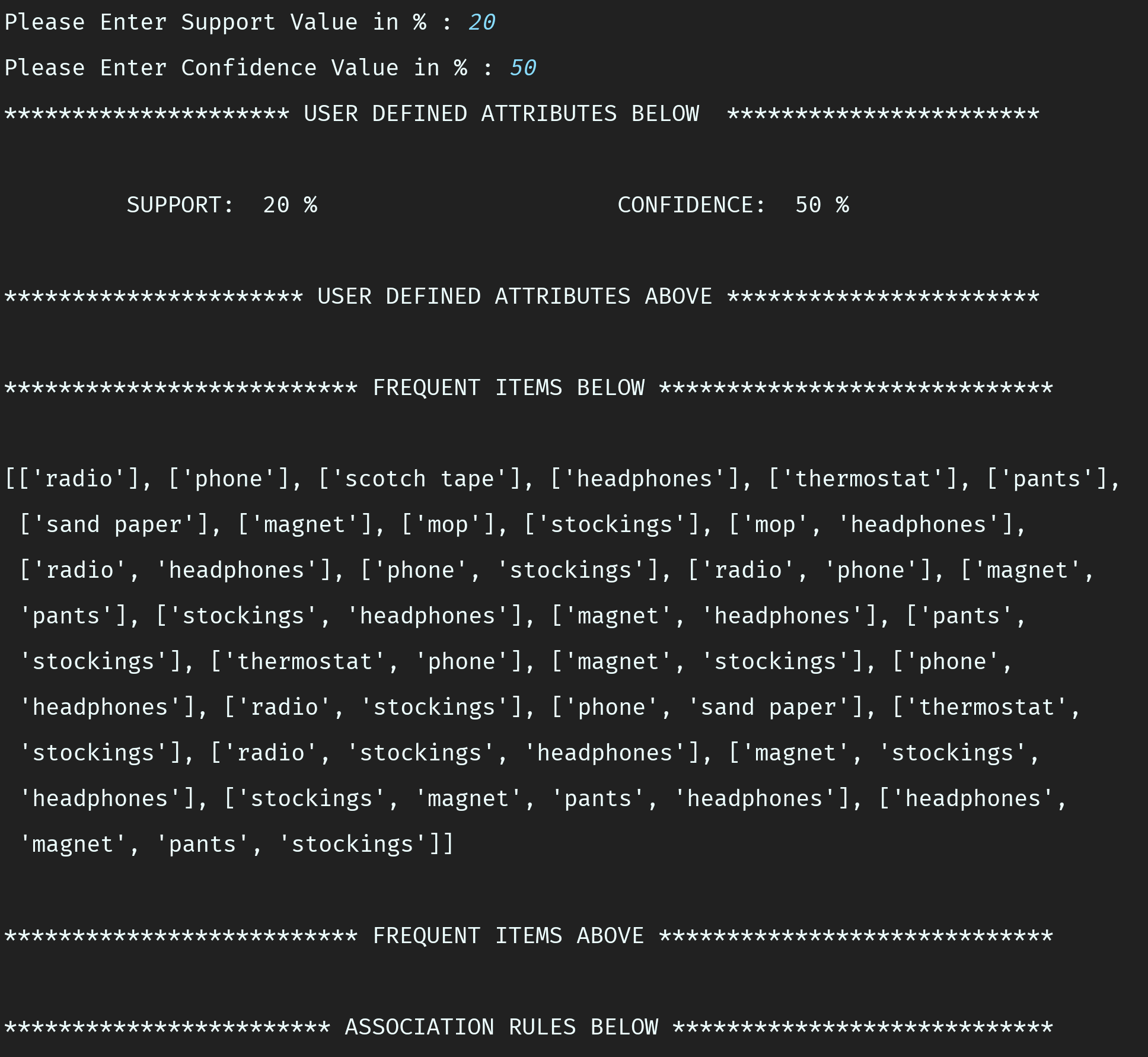
1. The user enters the name of the transactions database, he/she wants to perform the Apriori algorithm on. First, the user chooses tr\_db1.csv.



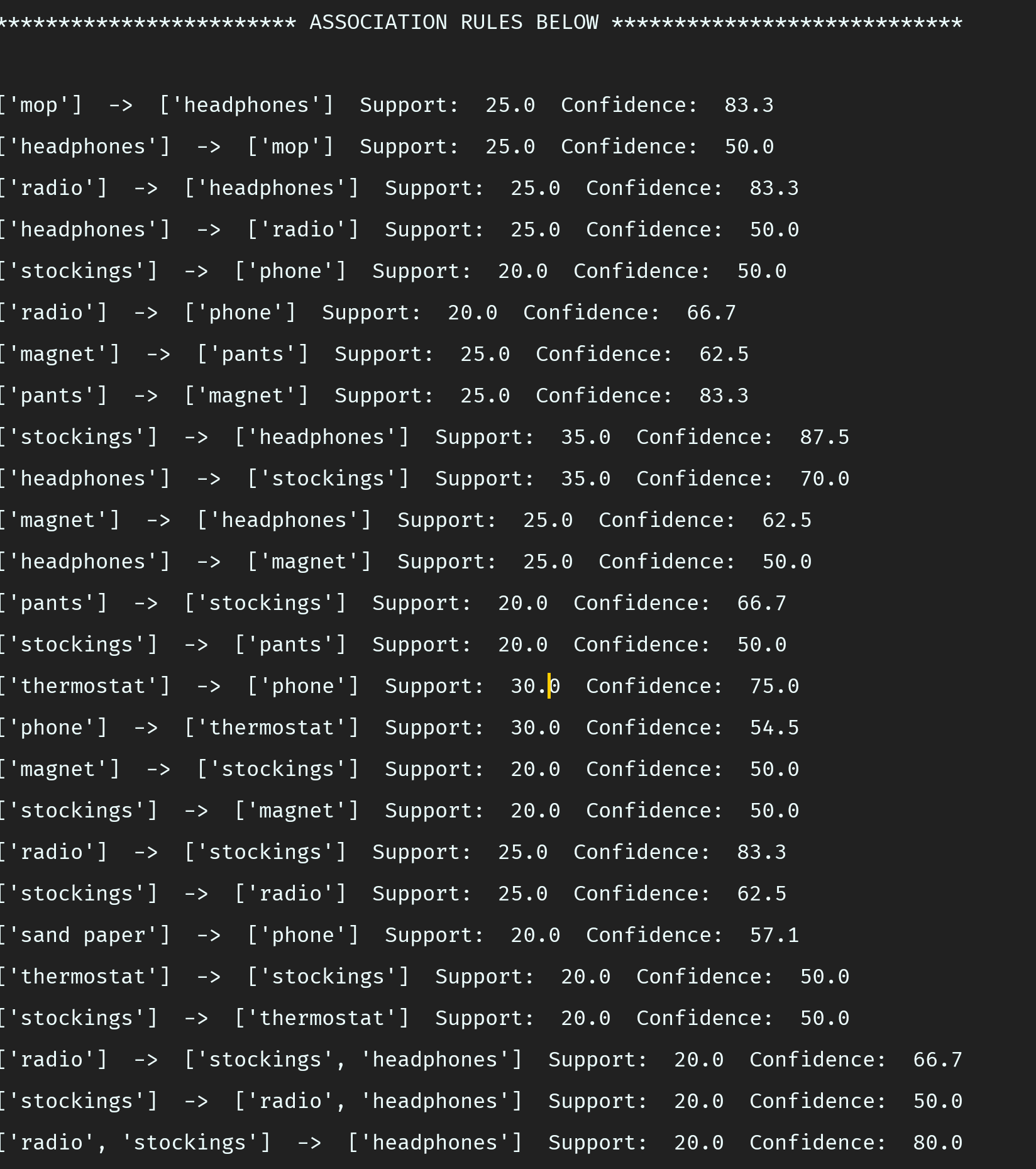
1. Hitting Enter displays all the transactions in tr\_db1.csv database. Next, the user is asked to enter a support and a confidence value.



1. Hitting enter then displays the user defined attributes and finds out all the frequent itemsets. The frequent itemsets here range from a length of 1 to 4 items.

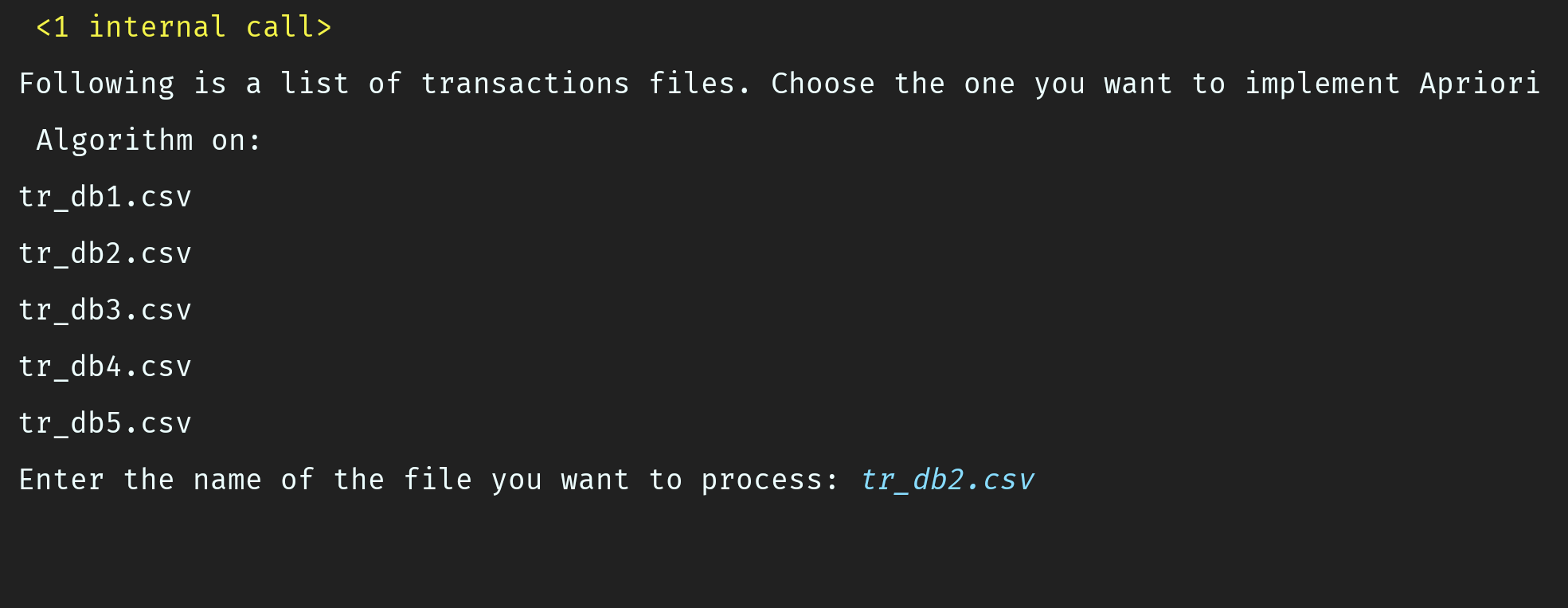


1. Below the frequent item sets, a list of all the association rules that can be created out of the frequent itemsets, are listed. Only those rules are displayed which satisfy the user defined support and confidence values.

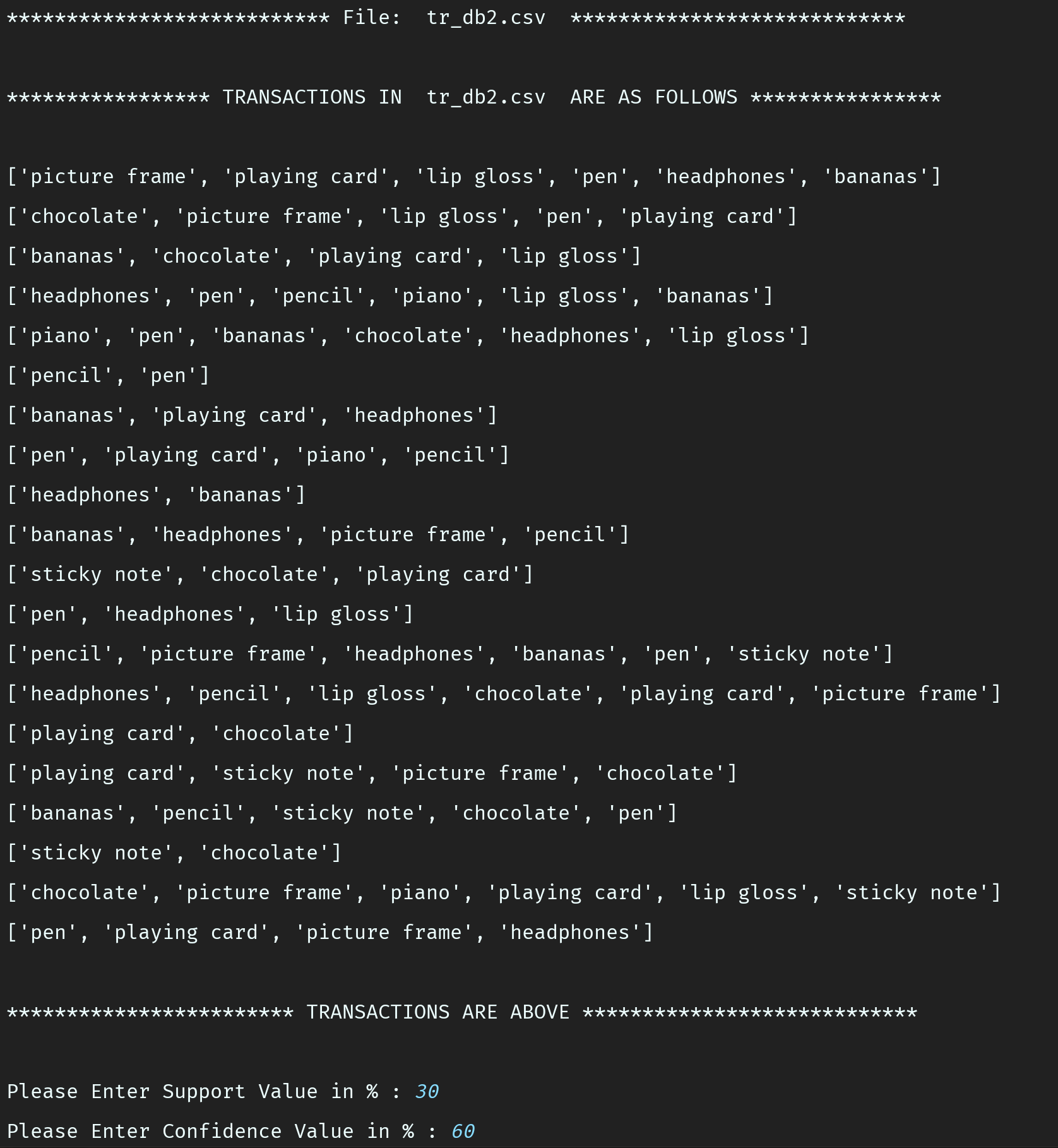




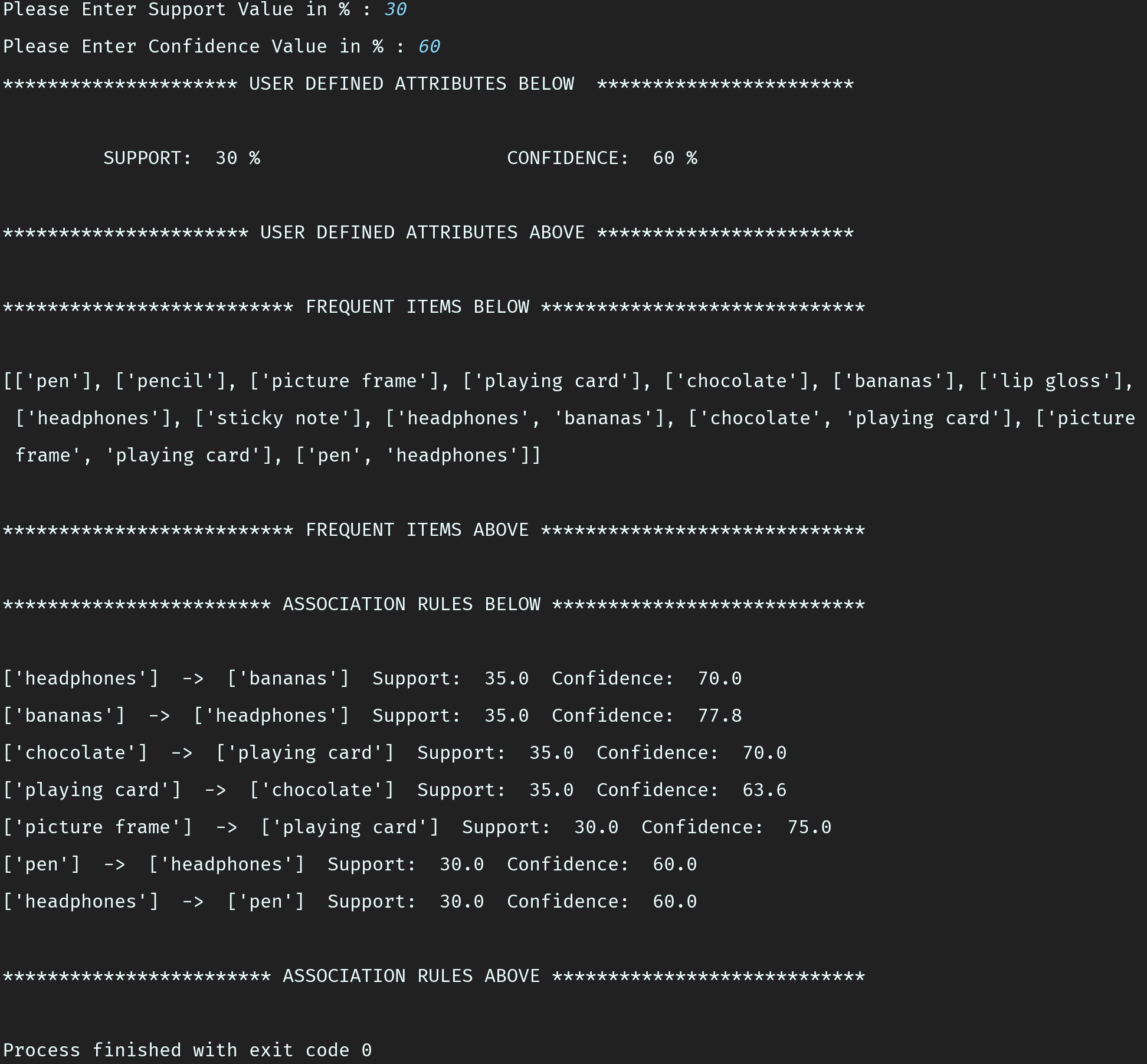
1. Once we are done with the first database, we move on to the next database i.e. tr\_db2.csv.



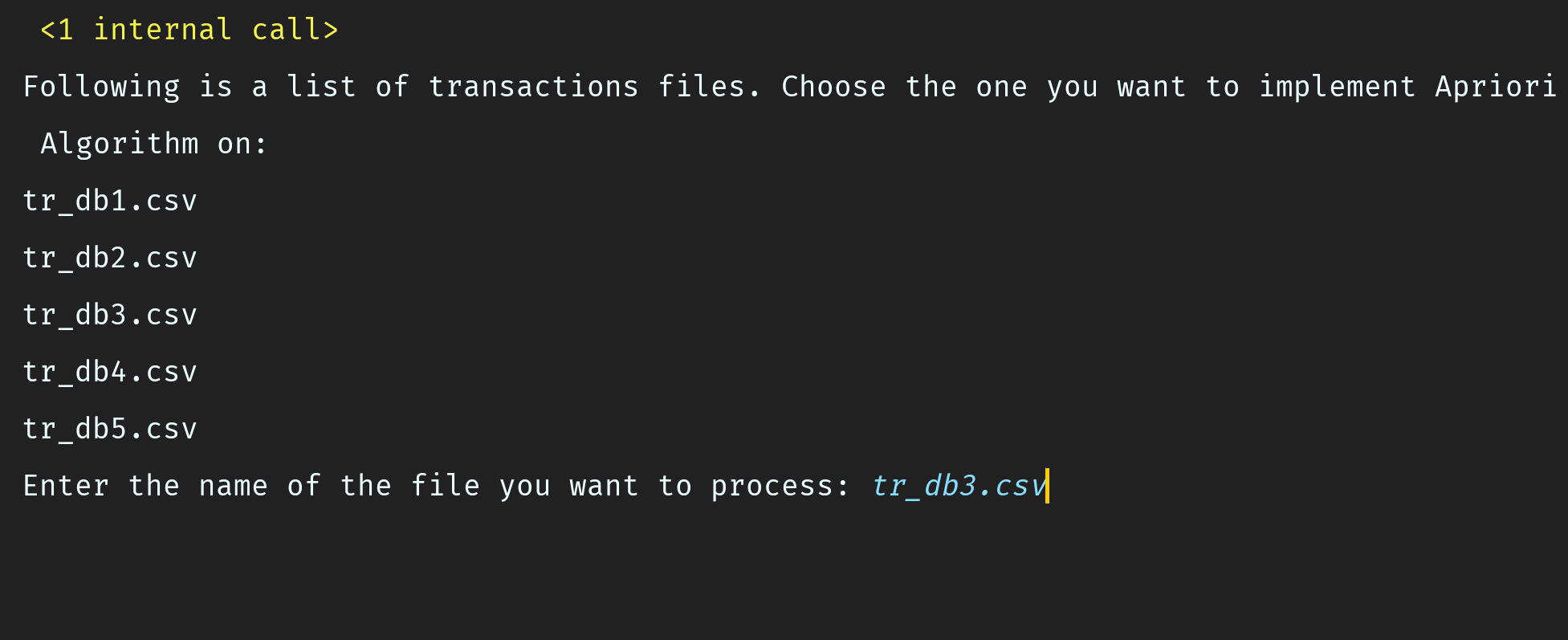
1. Like before, all the transactions in tr\_db2.csv database are displayed and the user is asked to give a support and a confidence value.



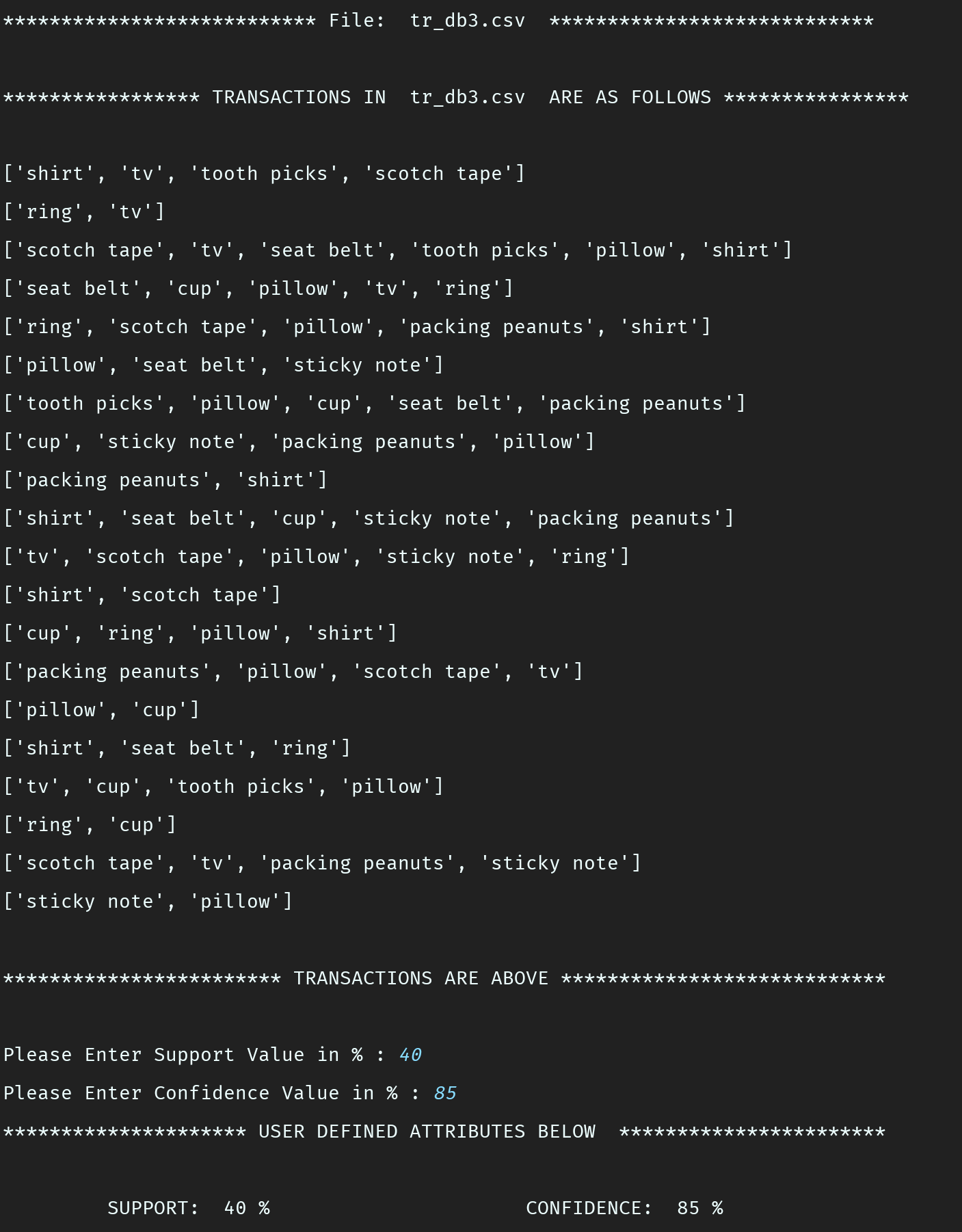
1. Once the user input is over, the program calculates all the frequent itemsets that satisfy the user defined support. Also, all the association rules that can be generated out of these frequent itemsets and satisfy the user defined confidence value, are displayed. As compared to the previous database, the number of association rules have reduced as both support and confidence have been increased.



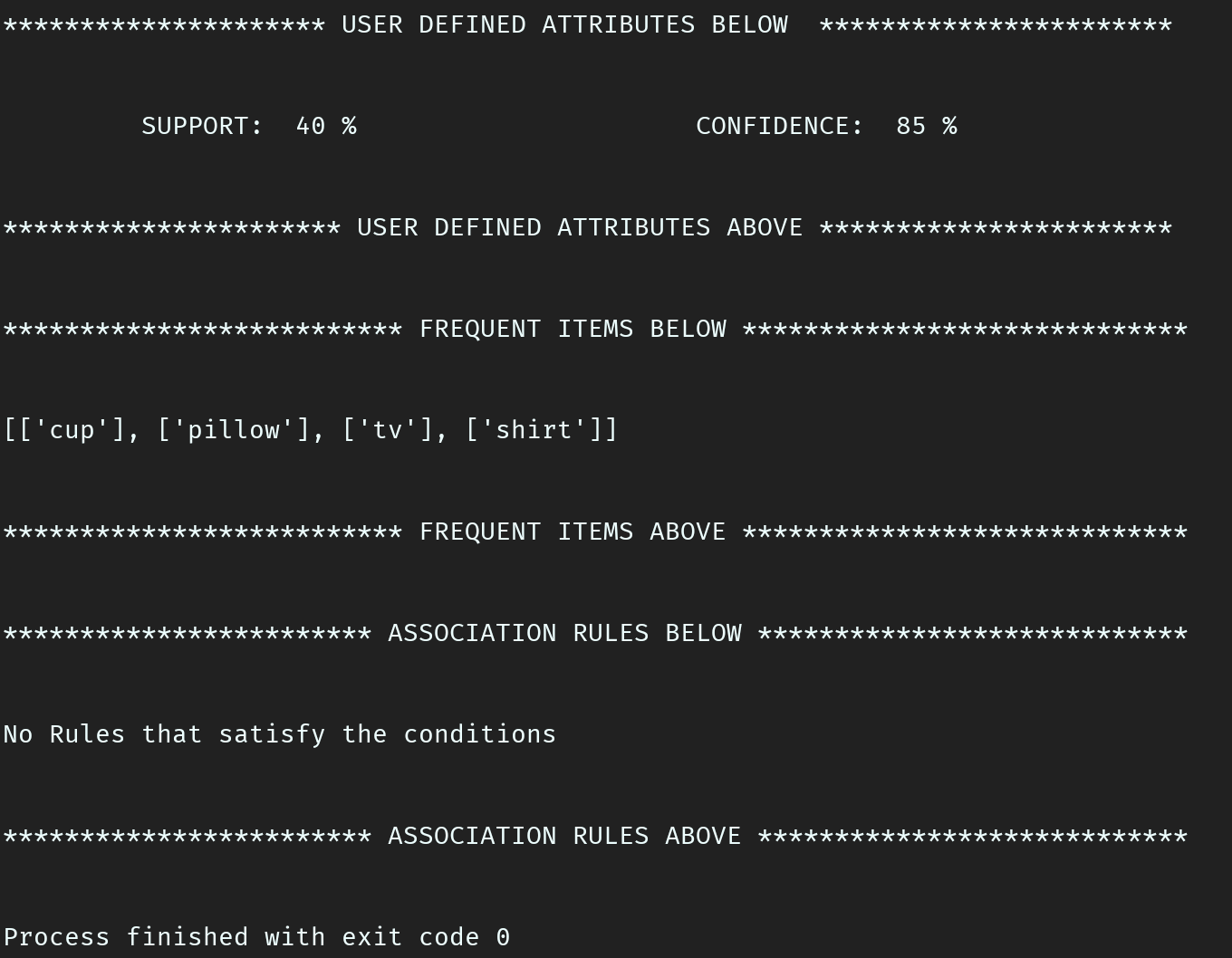
1. Now we take the 3rd database i.e. tr\_db3.csv.



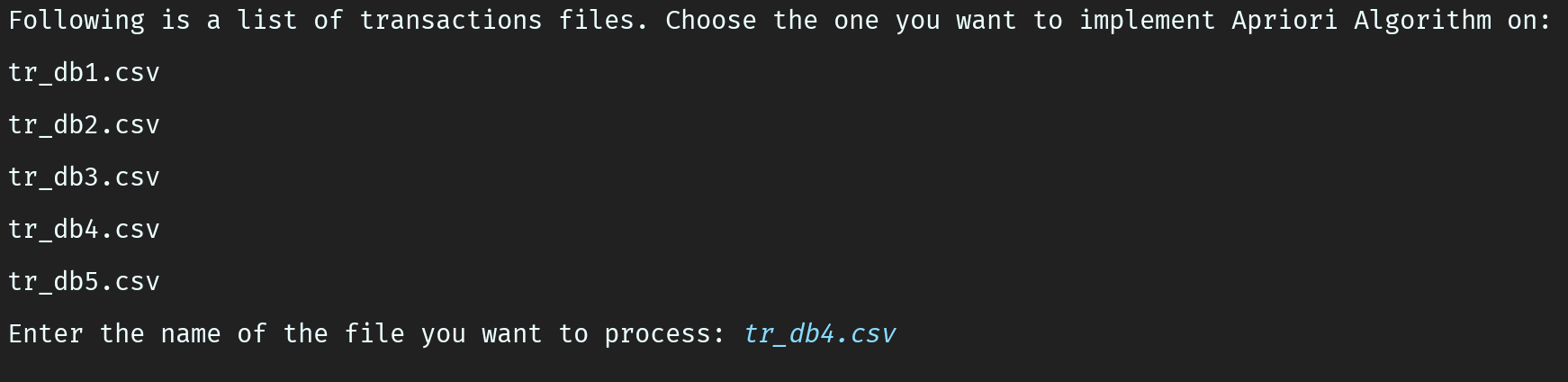
1. All the transaction in this database are displayed and the support and confidence values are entered.



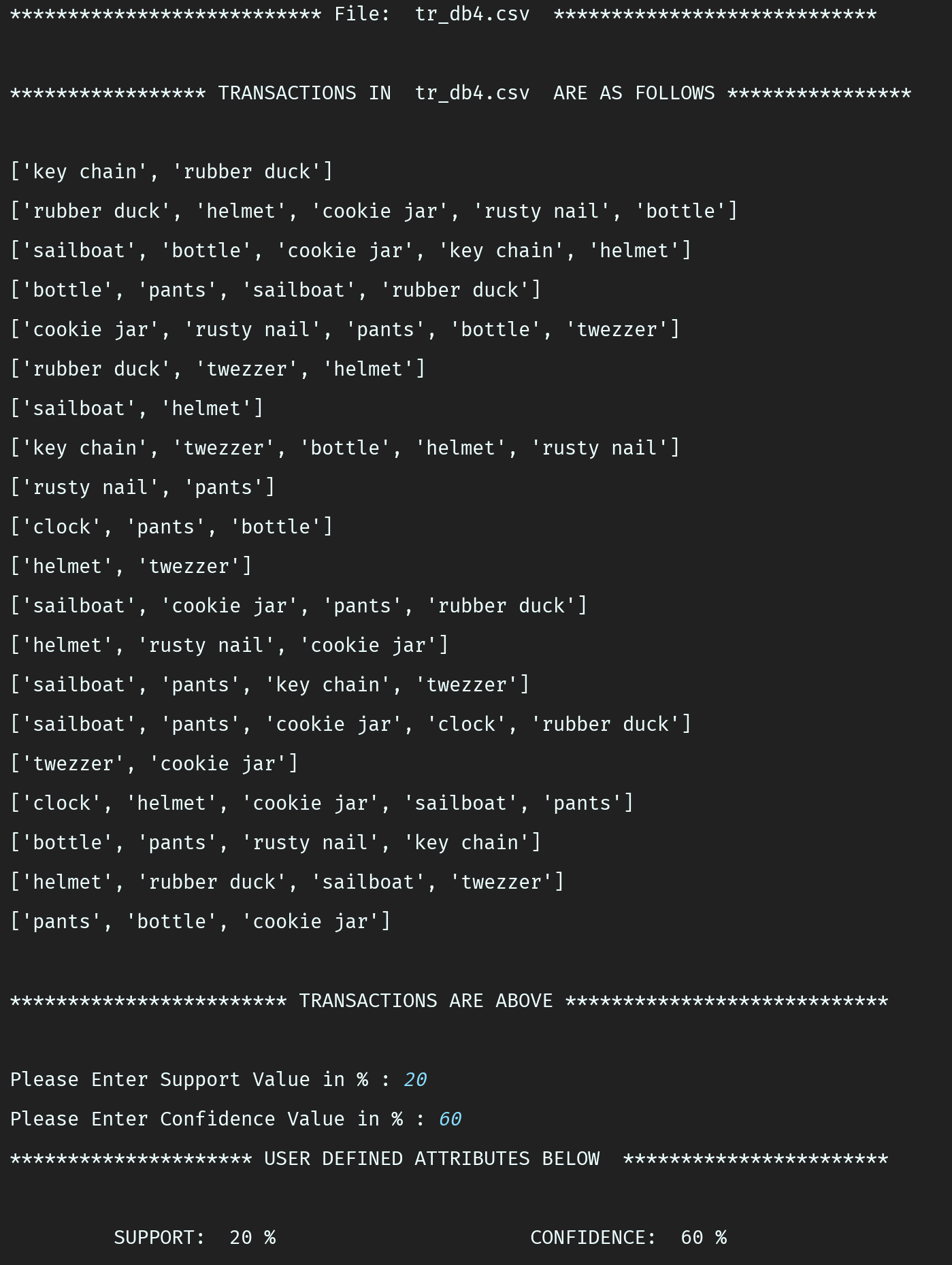
1. Surprisingly, there are only single item frequent itemsets and also there are no association rules that satisfy the user defined support of 40% and confidence of 85%.



1. We now move to the 4th database i.e. tr\_db4.csv.



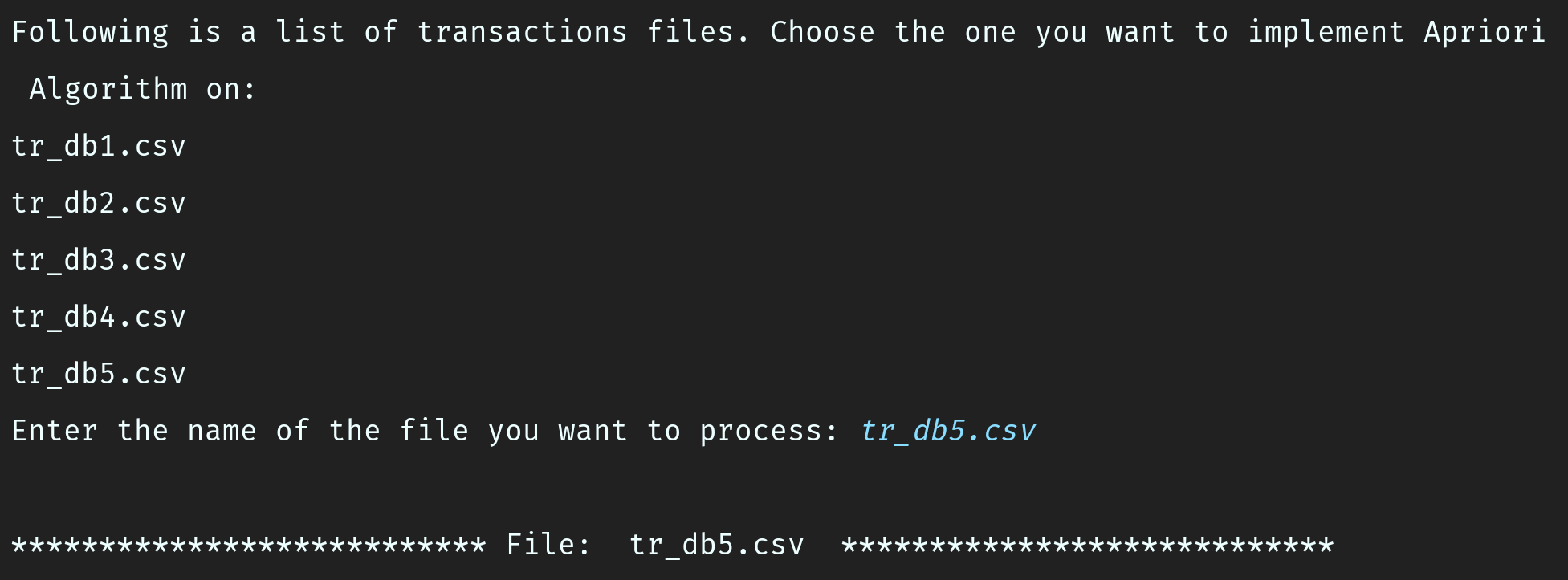
1. All transactions in tr\_db4.csv database are displayed and user provides support and confidence values.



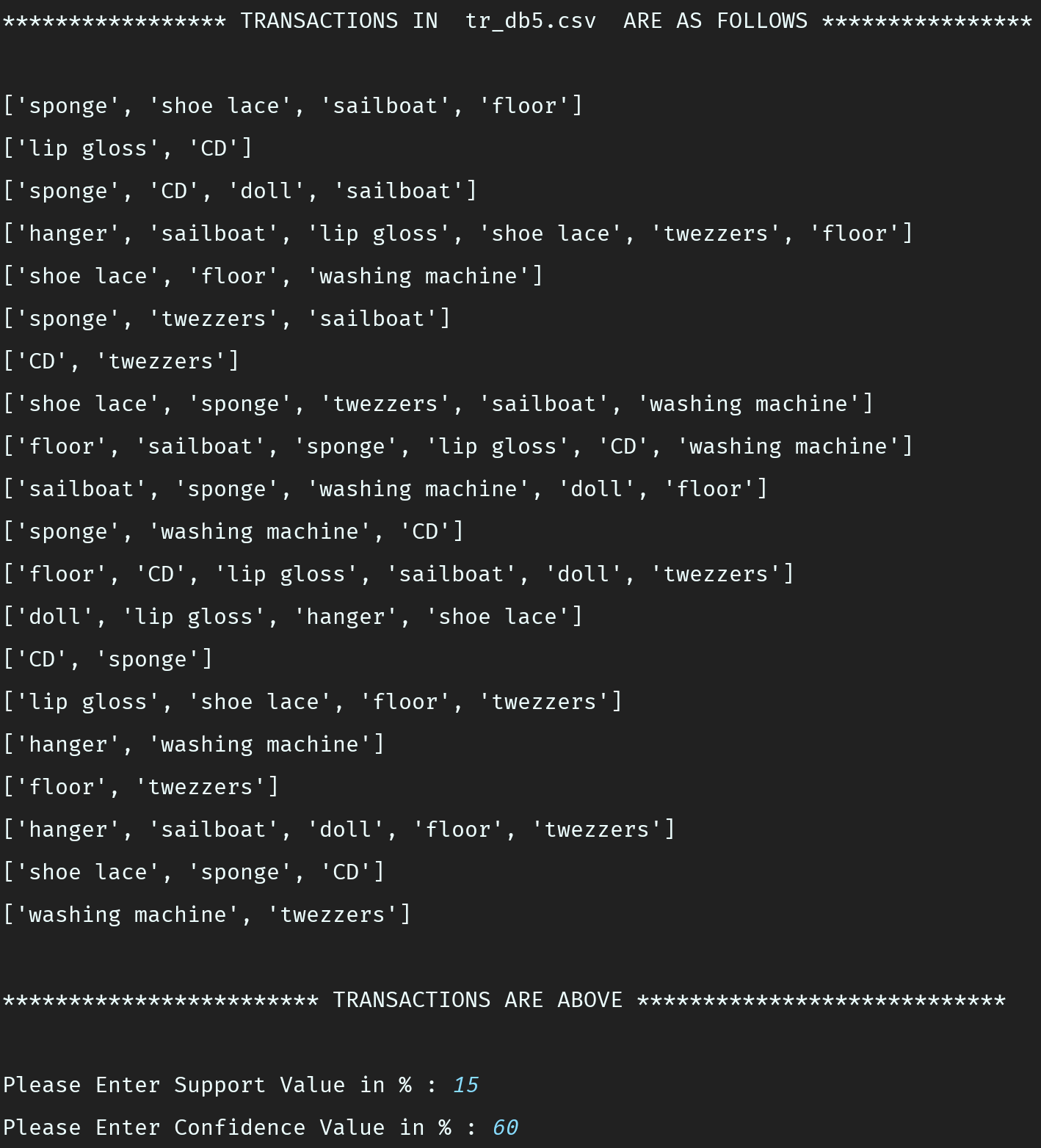
1. Now we get frequent item sets whose length ranges from 1 to 3 items per set. We see that the confidence value of 60 % does cut off a few association rules that might have been generated from the frequent itemsets.



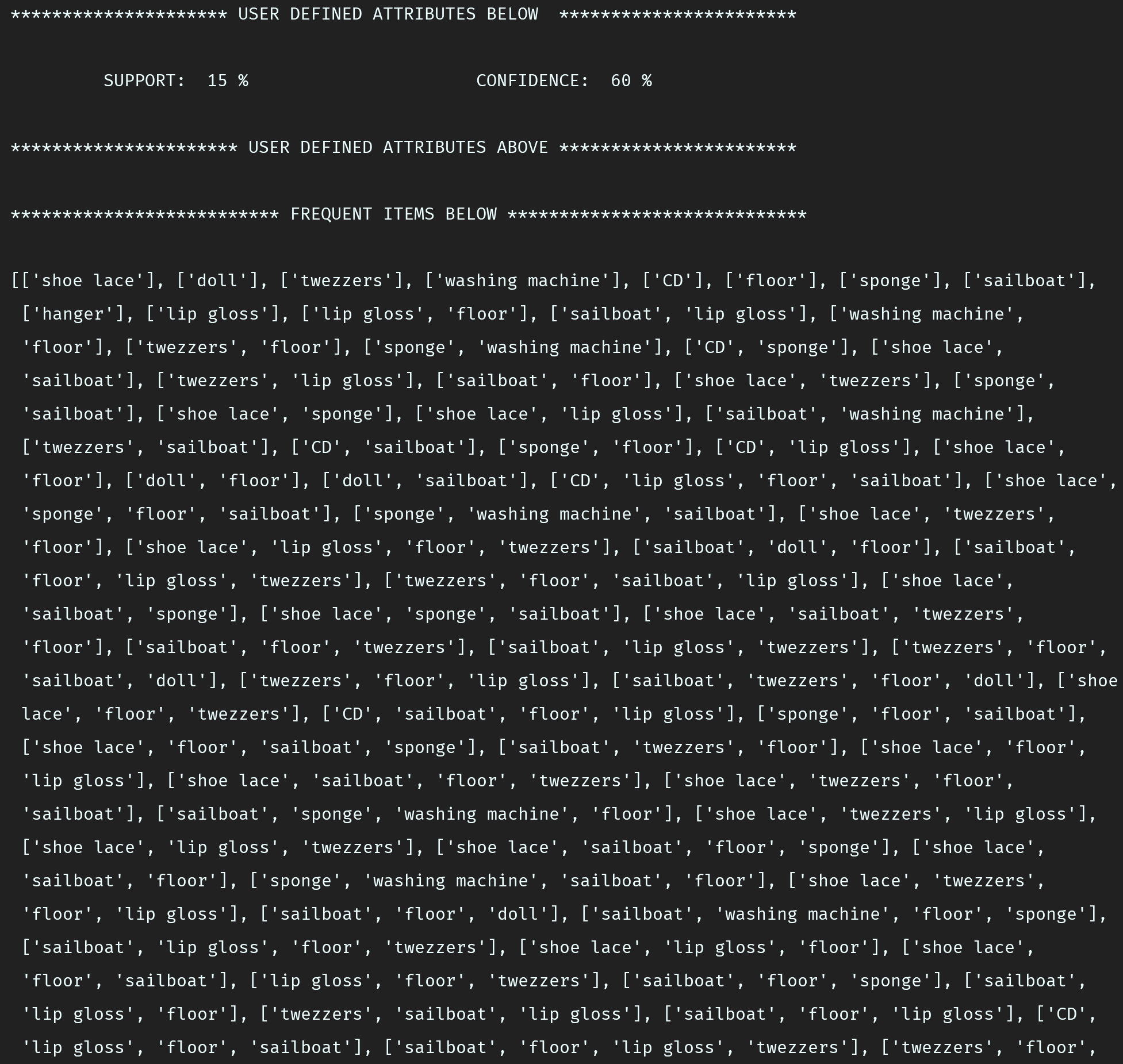
1. The last database is tr\_db5.csv.

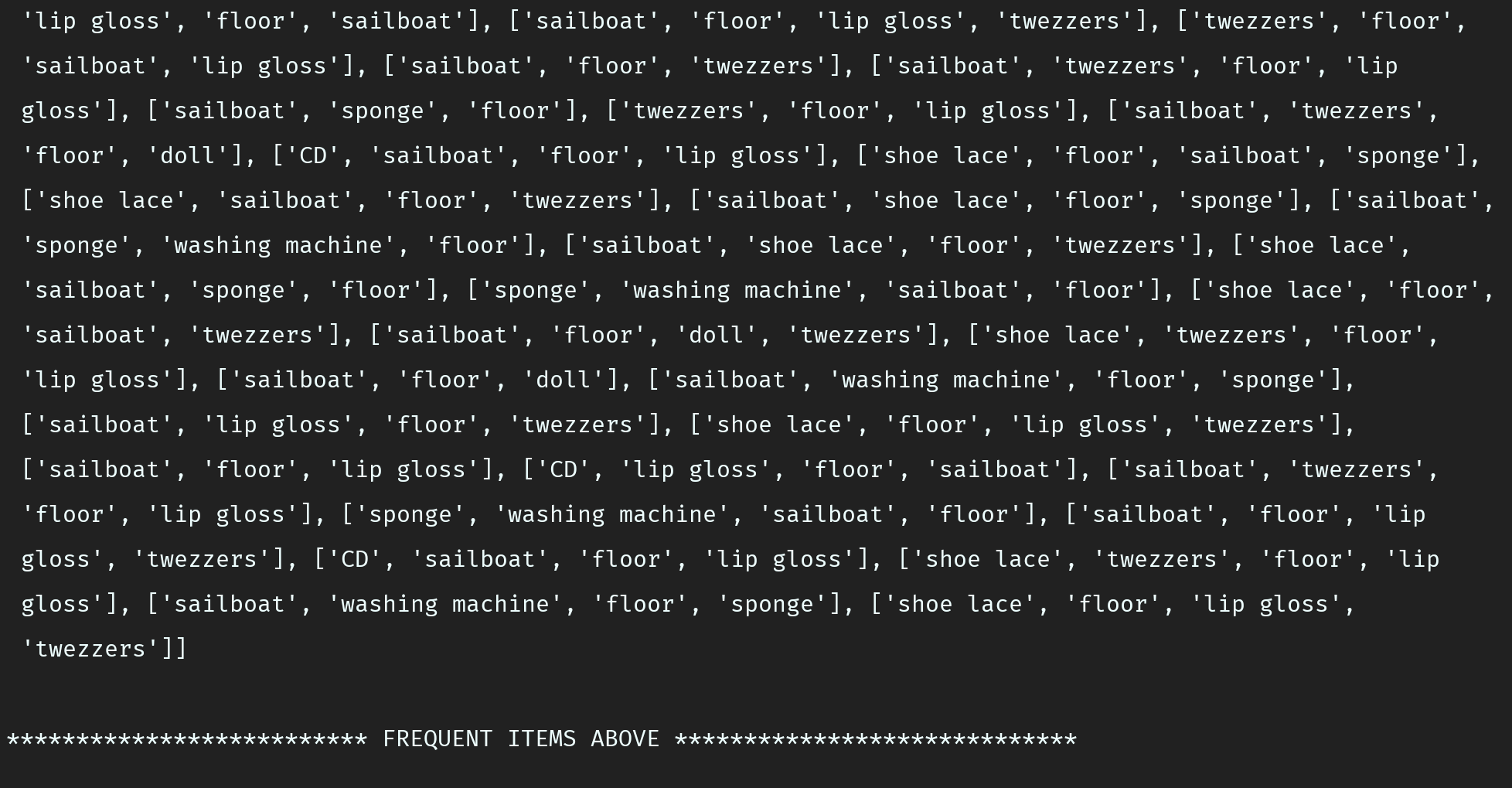


1. All transactions in this database are listed and the user is asked to enter a support and confidence value. Note that the support value is the minimum of all the support values entered. We can expect to see a lot of frequent itemsets.

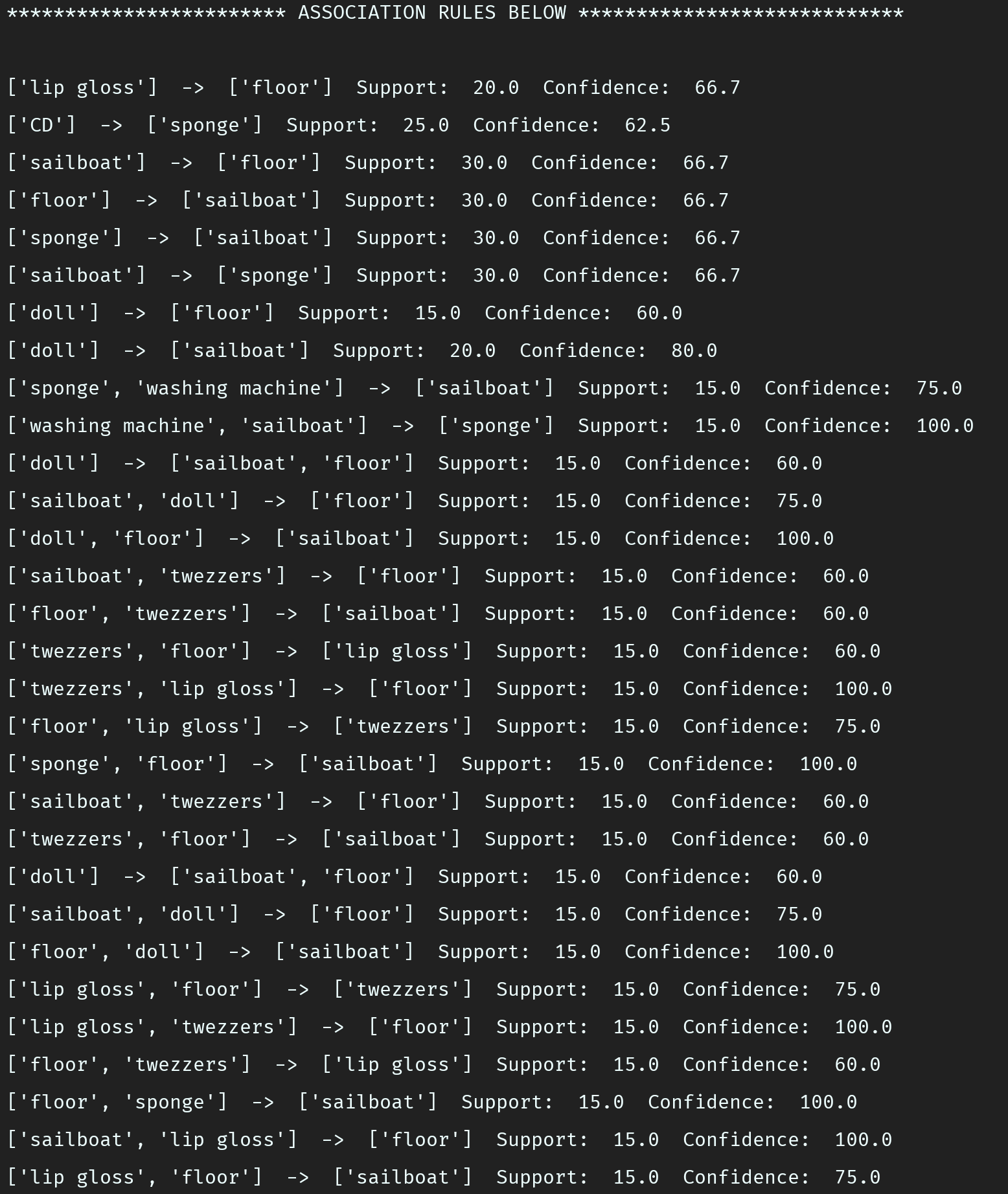


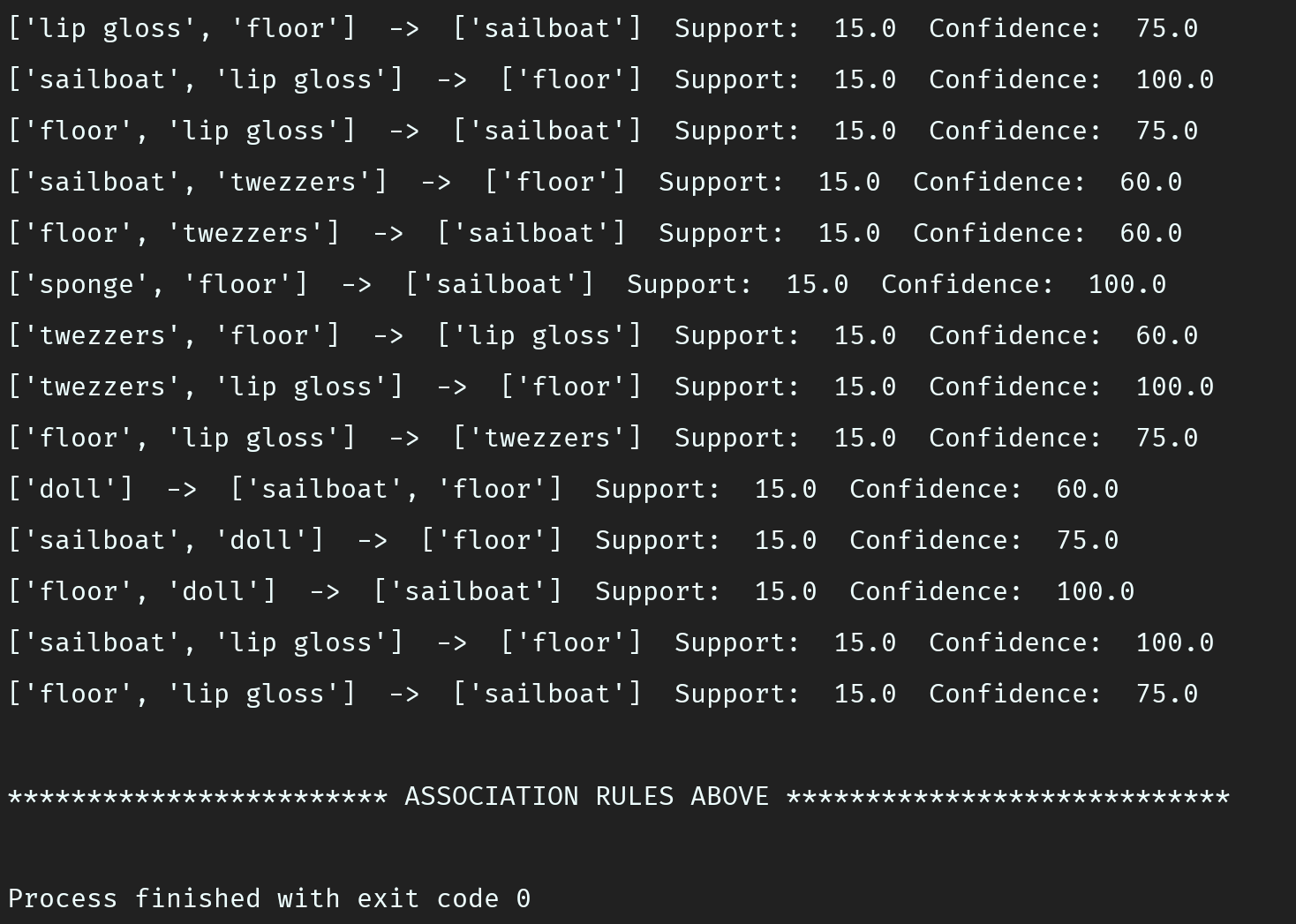
1. As expected, we get many frequent item sets whose length ranges from 1 to 4 items per set.





1. Also, we see that many association rules are generated, that satisfy the user defined support and confidence values.





**Conclusion**

This concludes are testing of databases. We see that for different transaction and different values of support and confidence, the number of association rules can vary a lot. For the same database, reducing the support and confidence values will always increase the number of association rules that satisfy those values.

**All Test Results**