Data Mining

Mid-term Project

Data Generation/Creating Association Rules via

Implementing Apriori Algorithm

From:

Name- Shashank Dubey

UCID- sd667

NJIT ID- 31435086

Project Requirements

1. Create 5 databases with 20 transactions each.
2. Use Apriori Algorithm to generate and print association rules and input transactions.
3. The Support and Confidence value needs to be User Specified.

Implementation

**Databases:**

1. Amazon Database
2. Kmart Database
3. HSM Database
4. Bestbuy Database
5. Licor Store Database

**Programming Language:** JAVA

**Operating System:** MacOS

**IDE:** Eclipse- Oxygen

Code:

package com.Apriori;

import java.io.File;

import java.util.ArrayList;

import java.util.Arrays;

import java.util.HashSet;

import java.util.Scanner;

public class Apriori

{

private static ArrayList<String>itemset = new ArrayList<String>();

private static ArrayList<String> transactions = new ArrayList<String>();

private static ArrayList<String> associations = new ArrayList<String>();

private static ArrayList<String>frequentitemsets = new ArrayList<String>();

private static double Support = 0;

private static double Conf = 0;

@SuppressWarnings("resource")

public static void main(String[] args)

{

try

{

Scanner input = new Scanner(System.in);

System.out.println("Enter Support value: ");

Support = input.nextDouble();

System.out.println("Enter Confidence value: ");

Conf = input.nextDouble();

System.out.println("Enter 'FileName.extension': ");

String fileName = input.next();

File Inputfile = new File(fileName);

input = new Scanner(Inputfile);

System.out.println("\n------------- TRANSACTIONS ------------\n");

while (input.hasNextLine())

{

String line = input.nextLine();

transactions.add(line);

System.out.println(line);

}

input.close();

}

catch(Exception e)

{

System.out.println("The file does not exists or is a bad format! Please enter correct file name.");

//e.printStackTrace();

}

candidateSet(transactions);

}

public static void candidateSet(ArrayList<String> list)

{

HashSet<String> set = new HashSet<String>();

for(inti=0; i<list.size(); i++)

{

String s = (String)list.get(i);

String temp[] = s.split(",");

for(int elements=0;elements<temp.length;elements++)

{

set.add(temp[elements]);

}

}

Object[] uniqueArr=set.toArray(); //array of unique elements

for(int i=0;i<uniqueArr.length;i++)

{

itemset.add(uniqueArr[i].toString());

}

generate(itemset);

}

public static void generate(ArrayList<String> items)

{

ArrayList<String> combinations = new ArrayList<String>();

//initializing combinations list by adding individual items

for(int p=0; p<items.size(); p++)

{

combinations.add(items.get(p));

}

ArrayList<String> temp=new ArrayList<String>();

//initializing by adding individual items to temp items list

for(int p=0; p<items.size(); p++)

{

temp.add(items.get(p));

}

ArrayList<String> temp1=new ArrayList<String>();

ArrayList<String> assign=new ArrayList<String>();

// generating all possible combinations (for Association rules)

for(int i=1;i<items.size();i++)

{

temp1.clear();

for(int elements=0;elements<items.size();elements++)

{

for(int k=0;k<temp.size();k++)

{

String str = items.get(elements).toString()+","+temp.get(k).toString();

if(checkDuplicate(str))

temp1.add(str);

}

}

temp.clear();

assign=checkElements(temp1);

//modified list, removed duplicate item-sets from temp list

for(int item=0;item<assign.size();item++)

{

temp.add(assign.get(item));

}

//updating combinations, concatenated result of temp list

for(intitemset=0;itemset<temp.size();itemset++)

{

combinations.add(temp.get(itemset));

}

}

//calculating support values

for(int comb=0;comb<combinations.size();comb++)

{

//calculating the support of each combination

double support = (calcSupport(combinations.get(comb).toString()) \*100)/ (double)transactions.size();

//System.out.println(combinations.get(comb).toString()+" - "+support);

if(support>Support)

frequentitemsets.add(combinations.get(comb));

}

System.out.println("\n------------- FREQUENT ITEMS ------------\n");

System.out.println(""+frequentitemsets);

getAssociations(frequentitemsets);

}

public static intcalcSupport(String S) // calculate support of an itemset

{

intctr = 0;

int count=0;

String arr[] = S.split(",");

for(int i=0;i<transactions.size();i++)

{

for(int j=0;j<arr.length;j++){

if(transactions.get(i).toString().contains(arr[j]))

ctr++;

}

if(ctr==arr.length)

count++;

ctr = 0;

}

return count;

}

public static void calcConfidence(ArrayList<String> associations) //checks confidence and prints association

{

double support=0, confidence=0;

System.out.println("\n----------- ASSOCIATION RULES -----------\n");

for(int i=0;i<associations.size();i++)

{

String arr[] = associations.get(i).toString().split(" -> ");

String str = arr[0].concat(",").concat(arr[1]);

support = calcSupport(str)/(double) transactions.size();

//System.out.println("Support - "+support);

confidence = support/(calcSupport(arr[0])/(double) transactions.size());

//System.out.println("Confidence - "+support);

support = Math.round(support\*10000.0)/100.0; // rounding upto 2 decimal

confidence = Math.round(confidence\*10000.0)/100.0; // rounding upto 2 decimal

if(support>Support && confidence>Conf)

System.out.println(associations.get(i)+" ("+support+","+confidence+")");

}

}

//create associations from frequent item-set

public static void getAssociations(ArrayList<String>itemset)

{

ArrayList<String> itemset1 = itemset; //new reference to item-set created

intctr=0;

for(int i=0;i<itemset.size();i++)

{

for(int j=0;j<itemset1.size();j++)

{

if(itemset.get(i).toString().length() >= itemset1.get(j).toString().length())//length check

{

if(!itemset.get(i).toString().equals(itemset1.get(j).toString()))// duplicate check

{

String []arr = itemset1.get(j).toString().split(",");

for(int k=0; k<arr.length; k++)

{

if(itemset.get(i).toString().contains(arr[k]))// contains check

{

ctr++;

}

}

if(ctr==0)

{

associations.add(itemset.get(i).toString()+" -> "+itemset1.get(j).toString());

//prints all possible transaction

//System.out.println(itemset.get(i).toString()+"->"+itemset1.get(j).toString());

}

}

ctr=0;

}

}

}

calcConfidence(associations);

}

public static booleancheckDuplicate(String S) // removes duplicate combinations (e.g- aa, bb)

{

String arr[] = S.split(",");

for(int i=0; i<arr.length; i++)

{

for(int j=i+1; j<arr.length; j++)

{

if(arr[i].equals(arr[j]))

return false;

}

}

return true;

}

public static ArrayList<String>checkElements(ArrayList<String> temp1) // checks for jumbled duplicates (ab = ba; abc = cab)

{

HashSet<String> set1,set2;

for(int i=0;i<temp1.size();i++)

{

String arr1[] = temp1.get(i).toString().split(",");

set1=new HashSet<String>(Arrays.asList(arr1));

for(int j=i+1;j<temp1.size();j++)

{

String arr2[] = temp1.get(j).toString().split(",");

set2=new HashSet<String>(Arrays.asList(arr2));

if(set1.equals(set2))

{

temp1.remove(j);

continue;

}

}

}

return temp1;

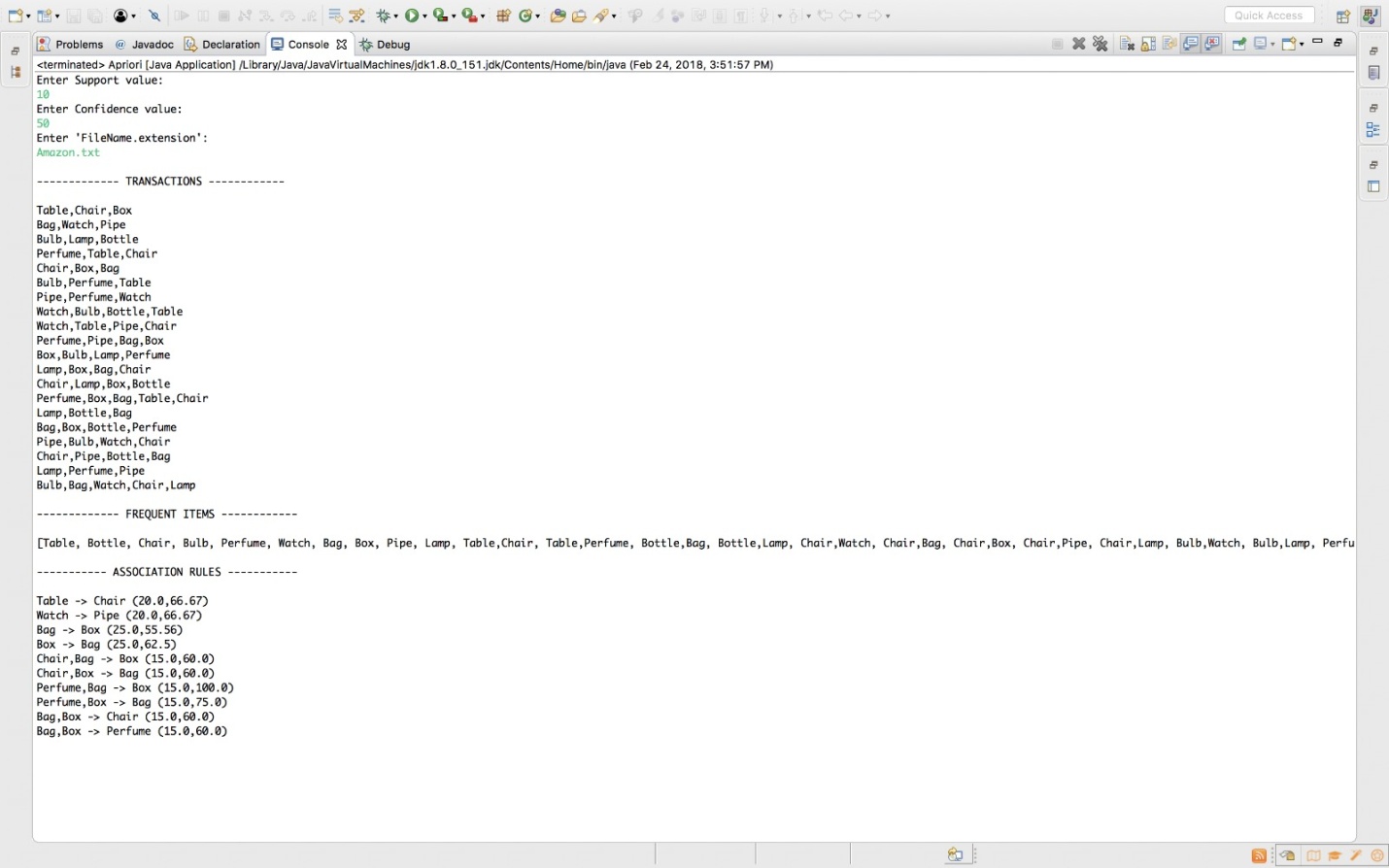
}

}

**Amazon Database:**

Support Value = 10

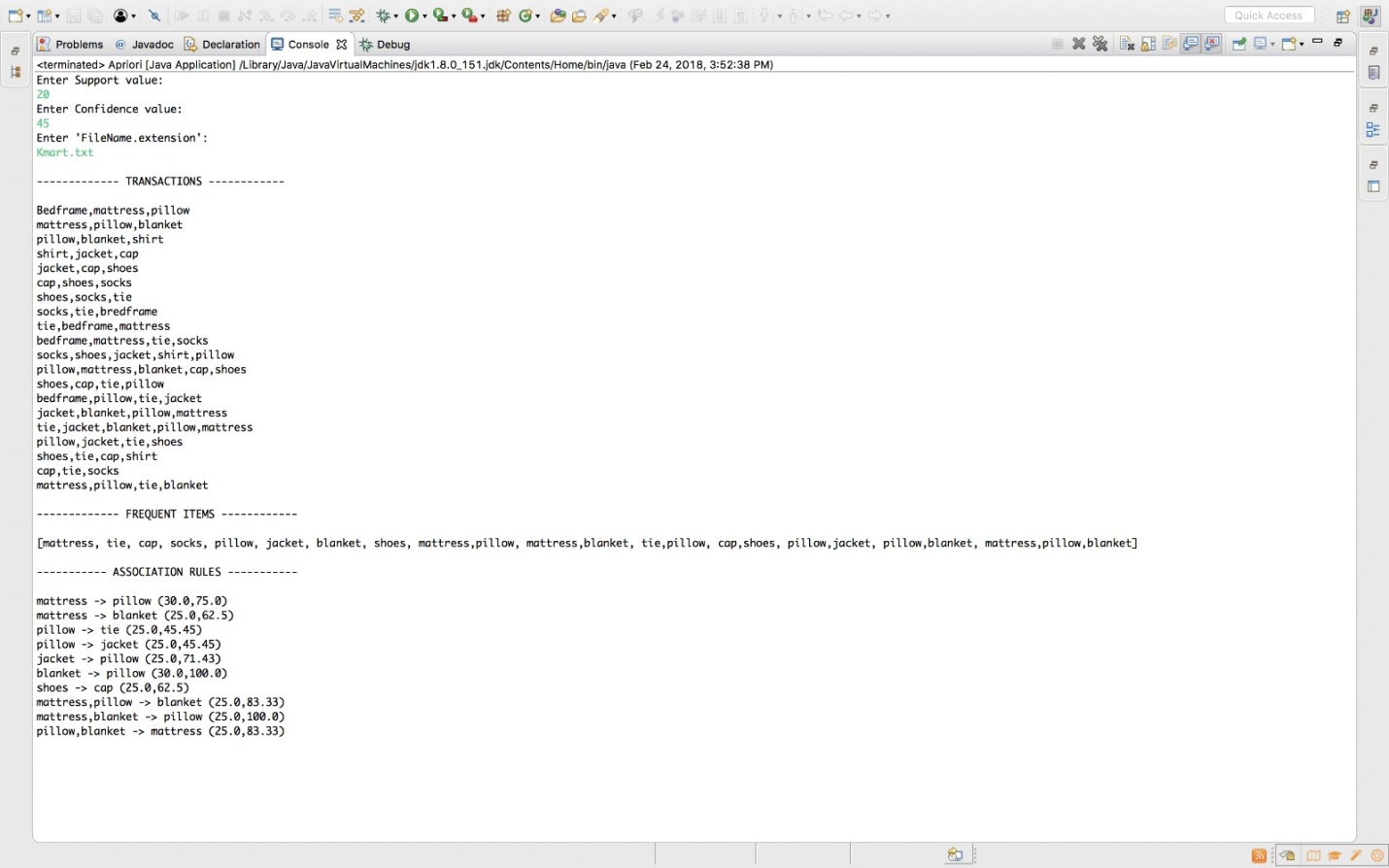
Confidence Value = 50



**Kmart Database:**

Support Value = 20

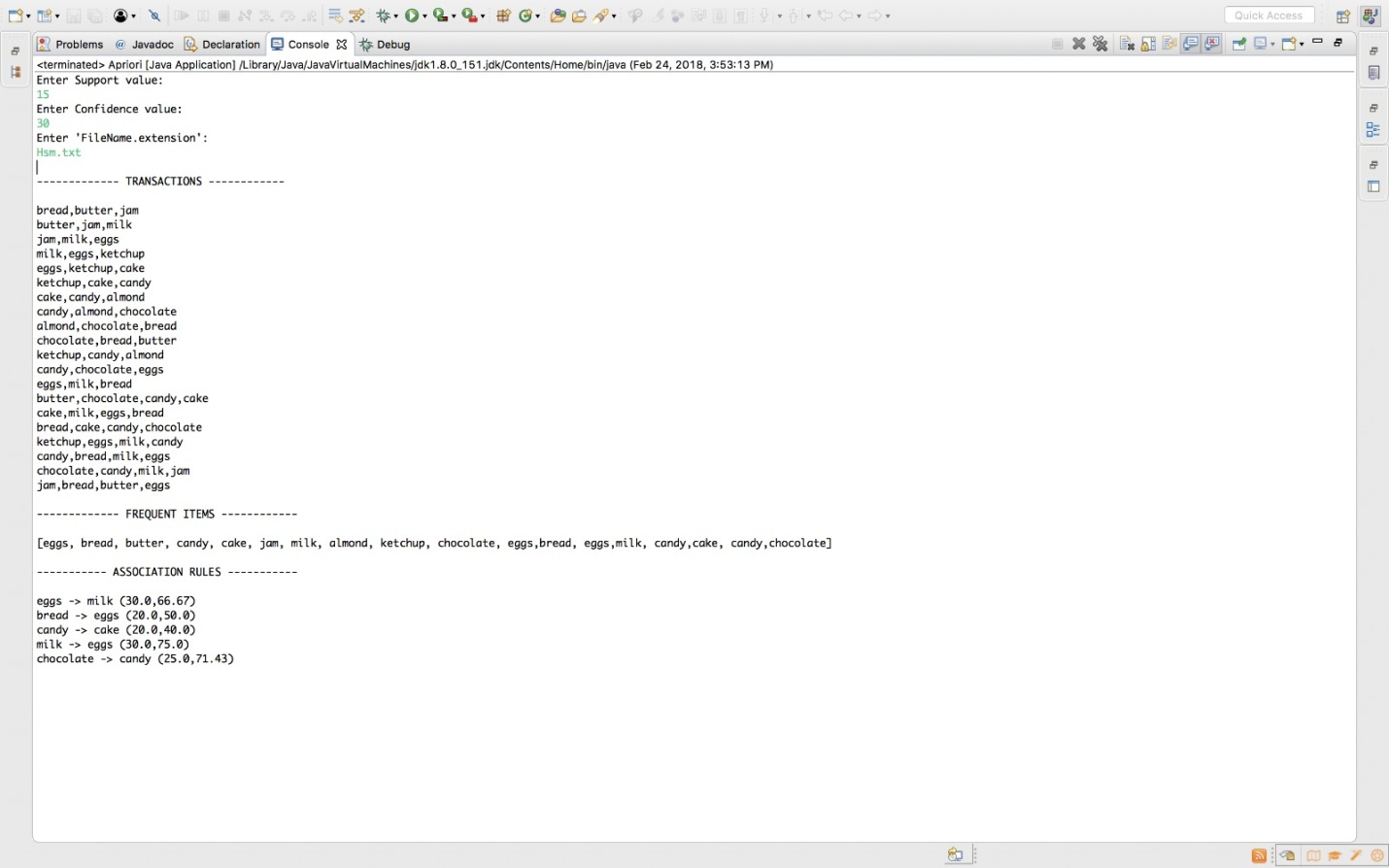
Confidence Value = 45



**HSM (Harrison Super Market) Database:**

Support Value = 15

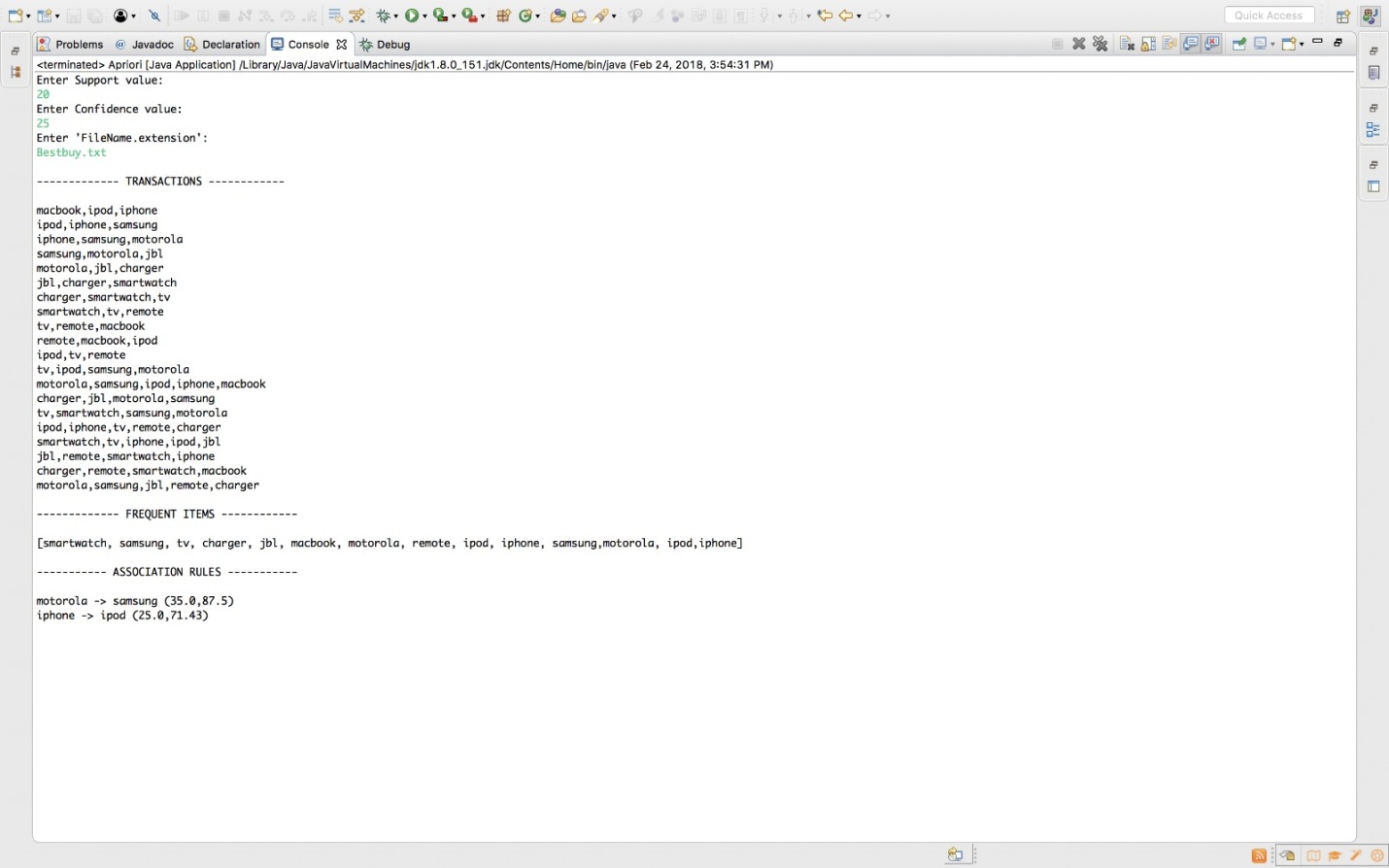
Confidence Value = 30



**Bestbuy Database:**

Support Value = 20

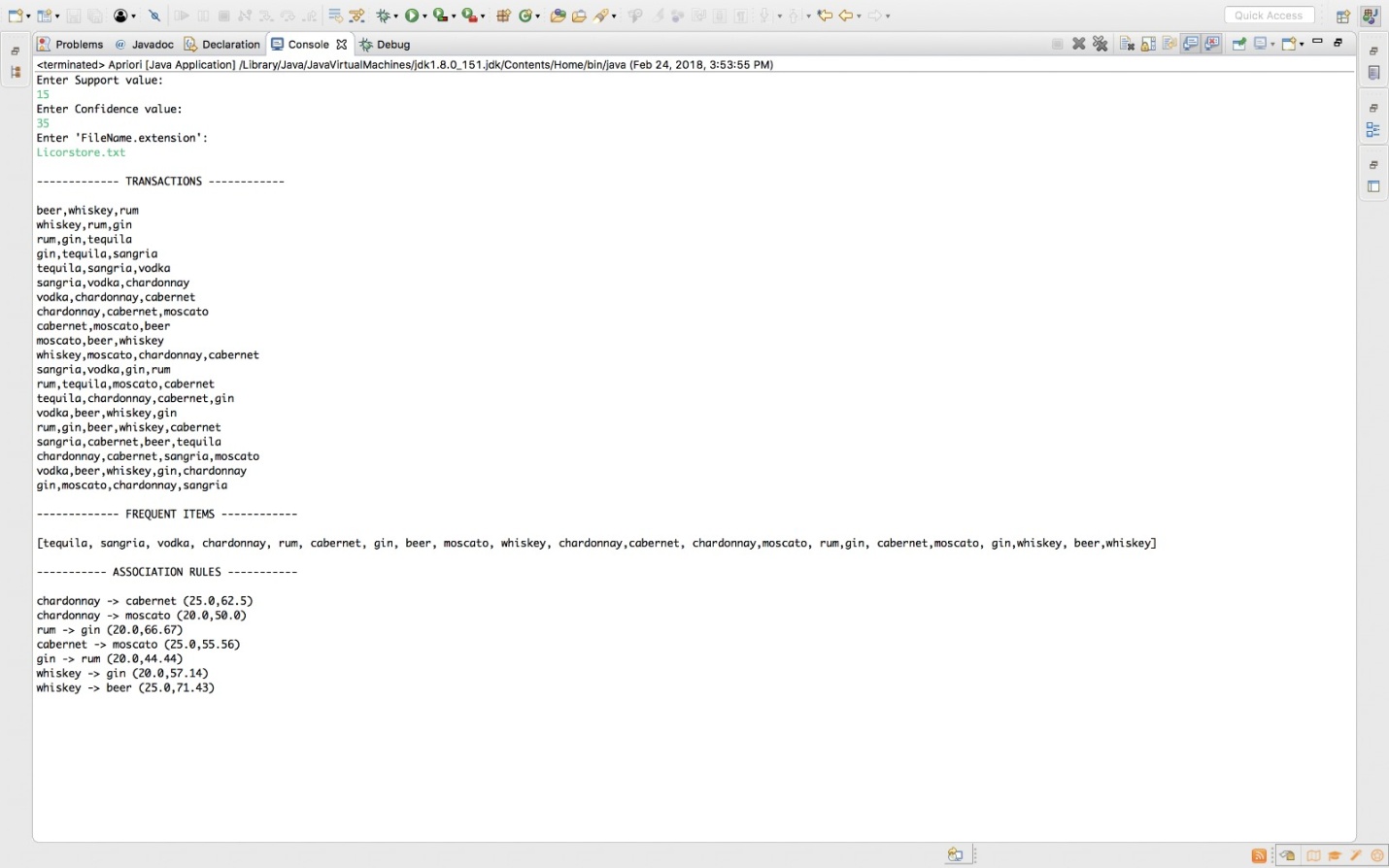
Confidence Value = 25



**Licor Store Database:**

Support Value = 15

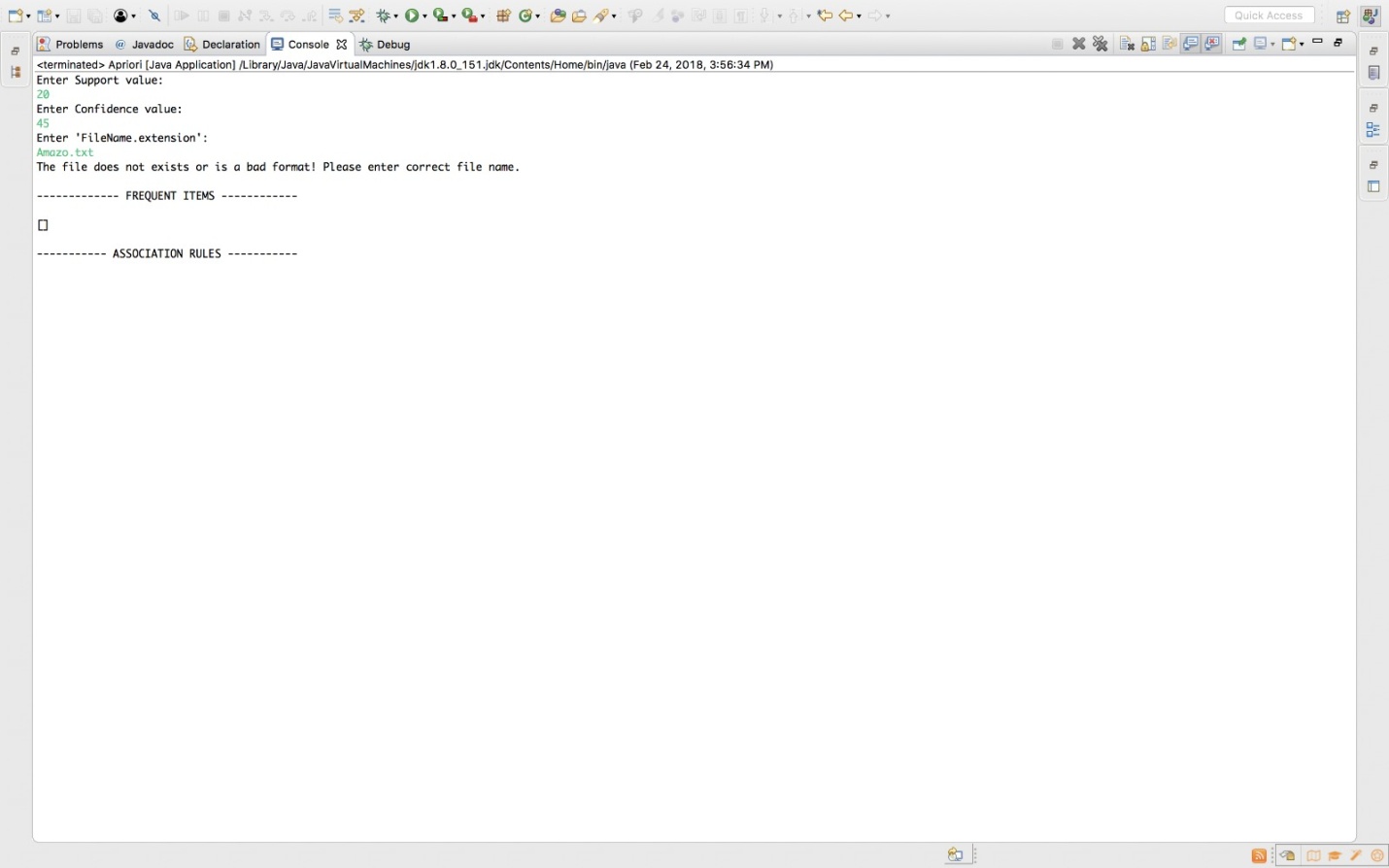
Confidence Value = 35



**Error Handling:**

**Incorrect File Name/Bad Format**

Here, I tried to enter incorrect file name of database file to check the Error handling of the code.



**Thank you**