|  |
| --- |
| Association Mining |
| MCDA 5580 Data Mining Assignment 3 Report |

|  |
| --- |
|  |

****

|  |  |
| --- | --- |
| ­­ | |
| Contents  [Executive Summary 2](#_Toc11958937)  [Data Summary 2](#_Toc11958938)  [Data Prepare 3](#_Toc11958939)  [Association Mining 6](#_Toc11958940)  [Conclusion 9](#_Toc11958941)  [Reference 11](#_Toc11958942)  [Appendix 12](#_Toc11958943)  [Captions 15](#_Toc11958944) | |

# Executive Summary

In this report, we will use apriori algorithm to mine the association rule between each product in the online retail dataset. We use OpenRefine to clean the transaction table, first, we remove non-merchandise product such as discount, carriage and Manual. Some transaction has negative quantity we treat it as returning products, we remove it and also using SQL query to remove correspond order in case those order has been refunded we will not use the data in our analysis. We also use ddply to basket all the items into one InvoiceNo to meet the requirement for apriori algorithm. In order to reduce the association rule amount, we set support to 0.02 and confidence to 0.5 to obtain 30 association rules. We have filtered the rule not in maximal and have our final association rules.

# Data Summary

The online retail dataset is a table in a database which included the online retail store’s invoice transaction. The table has nine columns to store InvoiceNo (An unique number to identify order invoice), StockCode (An unique text to identify a product), Description (Detail description about the product), Quantity (The number of product in this transaction), InvoiceDate (The date and time when the transaction occurs), UnitPrice (Unit price of the product), CustomerID (The unique ID to identify a customer), Country (The nation which the customer residence), InvoiceDateTime (The date and time when the transaction occurs).

There is a total of 541909 transaction records in this table.



Figure Data source

# Data Prepare

Step 1:

We exported the following columns from MySQL server using phpMyAdmin: InvoiceNo/StockCode/Quantity/CustomerID.

Step 2:

By observing the data, almost all StockCodes are with 5 digits and 1 optional letter. Those products with unnormal StockCodes are for service like carriage or postage. So we first use OpenRefine to split the column StockCode by length 5 and remove all records with unnormal StockCodes.

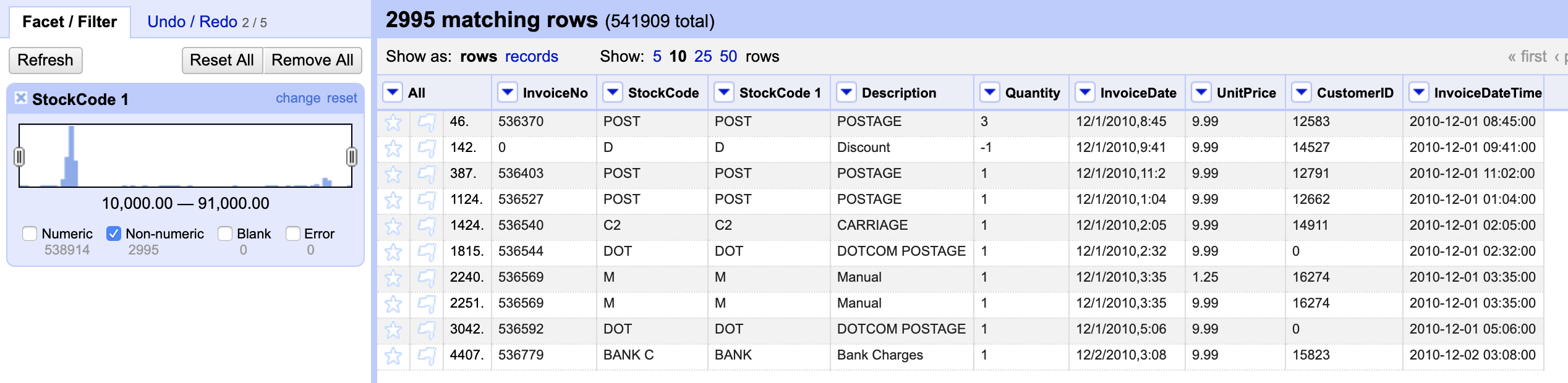


Figure 2 Data Cleaning

Step 3:

We find that there are negative values in the Quantity column, and we assume these are the records for turning back the product. As a result, we find it necessary to remove those records as well as the related positive value. First, we exported all records with negative values in the Quantity column and stored them in RStudio. Then we imported all records into MySQL database. By assuming that the turning back can’t happen outside 30 days, we use RStudio to connect MySQL database and run scripts to identify and remove all related records. Some records with negative values in the Quantity column don’t have matched records, but we still find 2227 related records and remove them. Finally, we got 535586 records. The scripts are shown below.

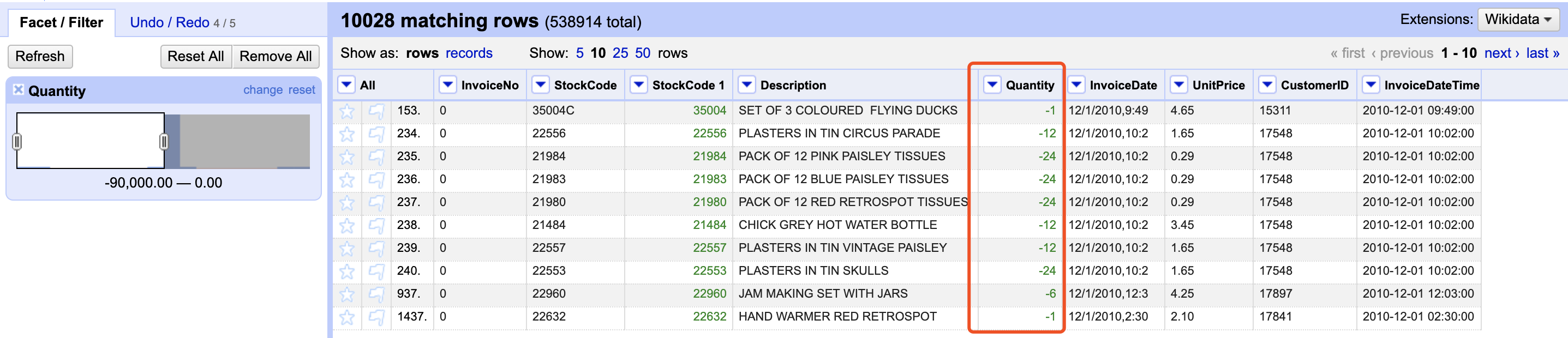
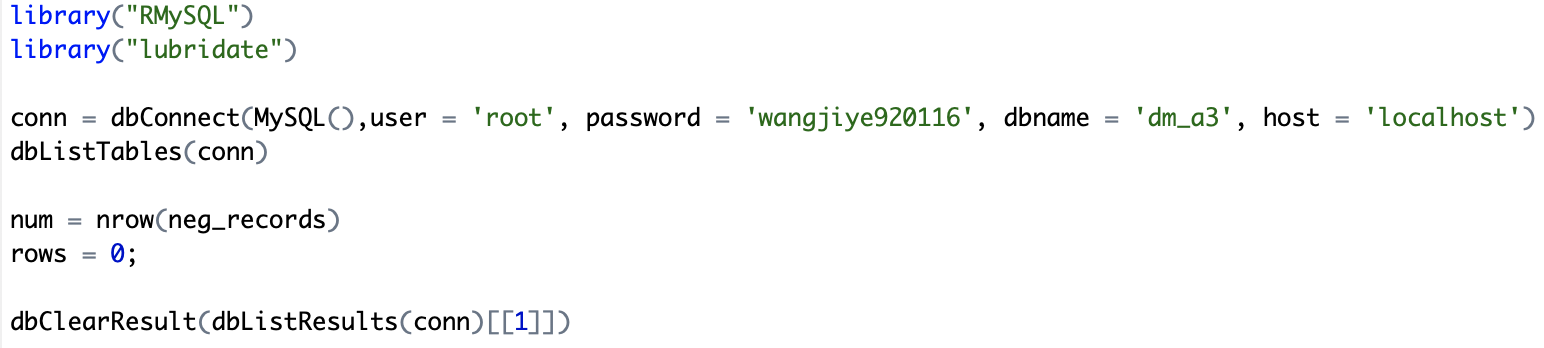
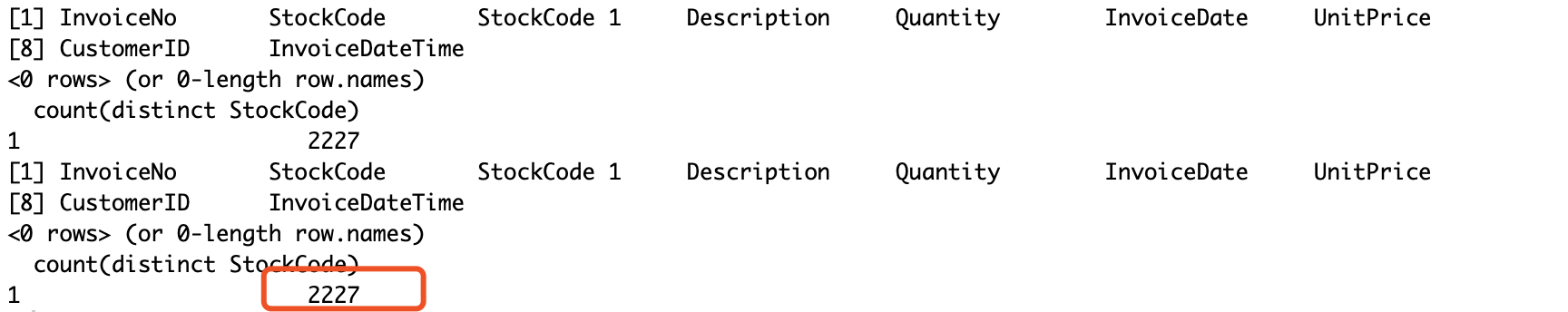
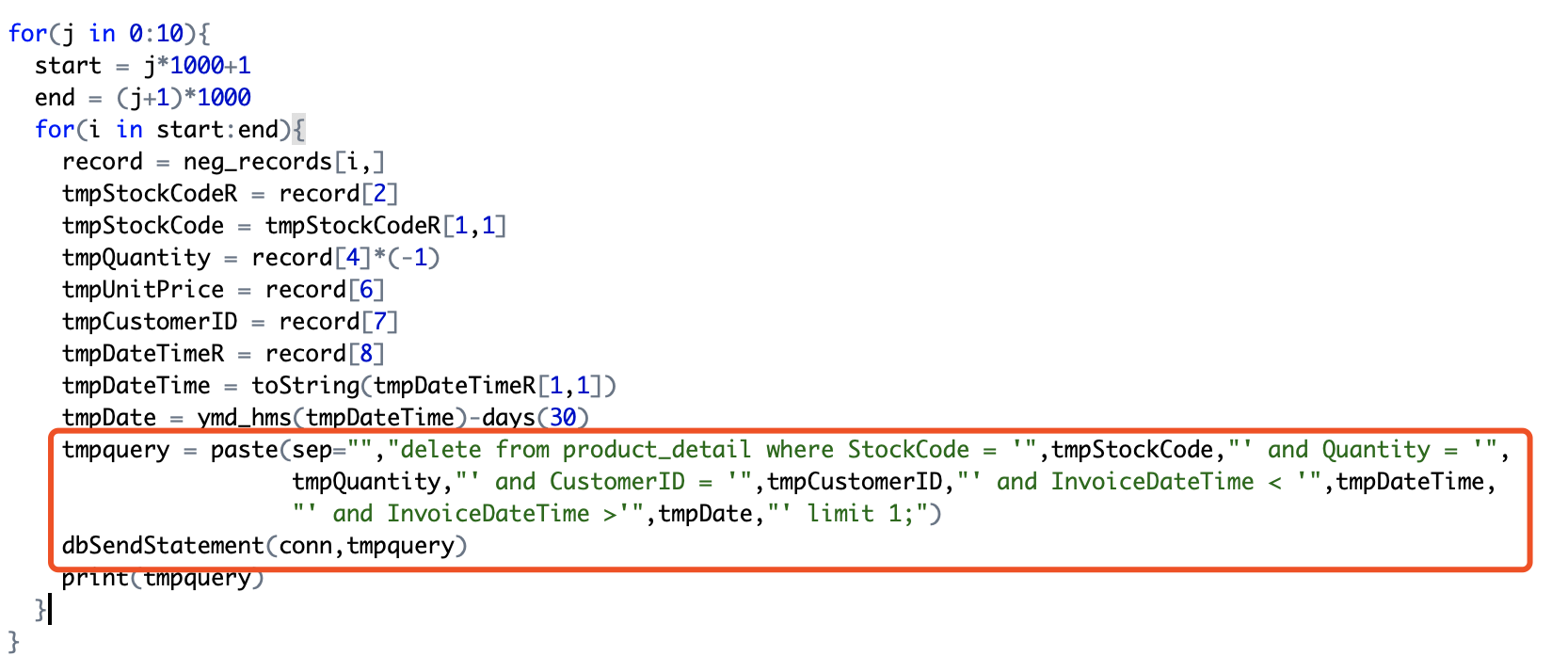


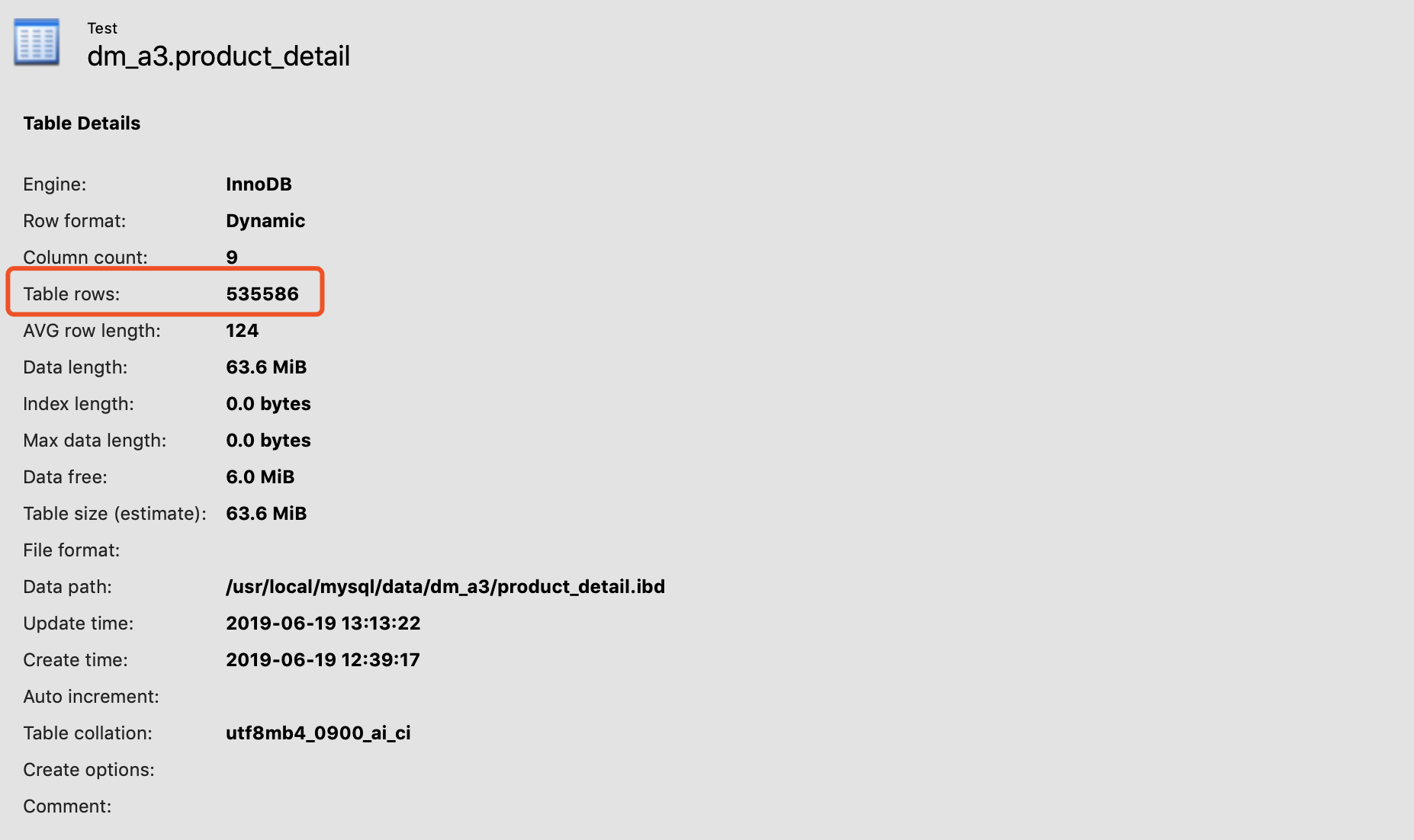
Figure 3 Remove Return Order









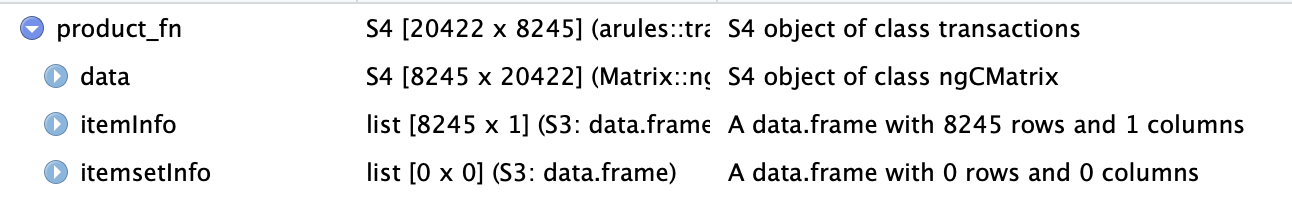


Step 4:

Apriori Algorithm required a basked data frame to be able to perform the analysis, for example in our case the data frame should have InvoiceNo as column one and follow with column two contain all the StockCode and separated by a comma. And we also want to see the product name instead of StockCode, so we created a table to remove StockCode with more than one description and then join with the original table. Then we sorted with InvoiceNo and grouped by the newly generated Description column and put the values in Description columns in one basket for each InvoiceNo and remove the Invoice column.



Here is our output data frame:



Step 5:

We also observed the 10 products with highest support.

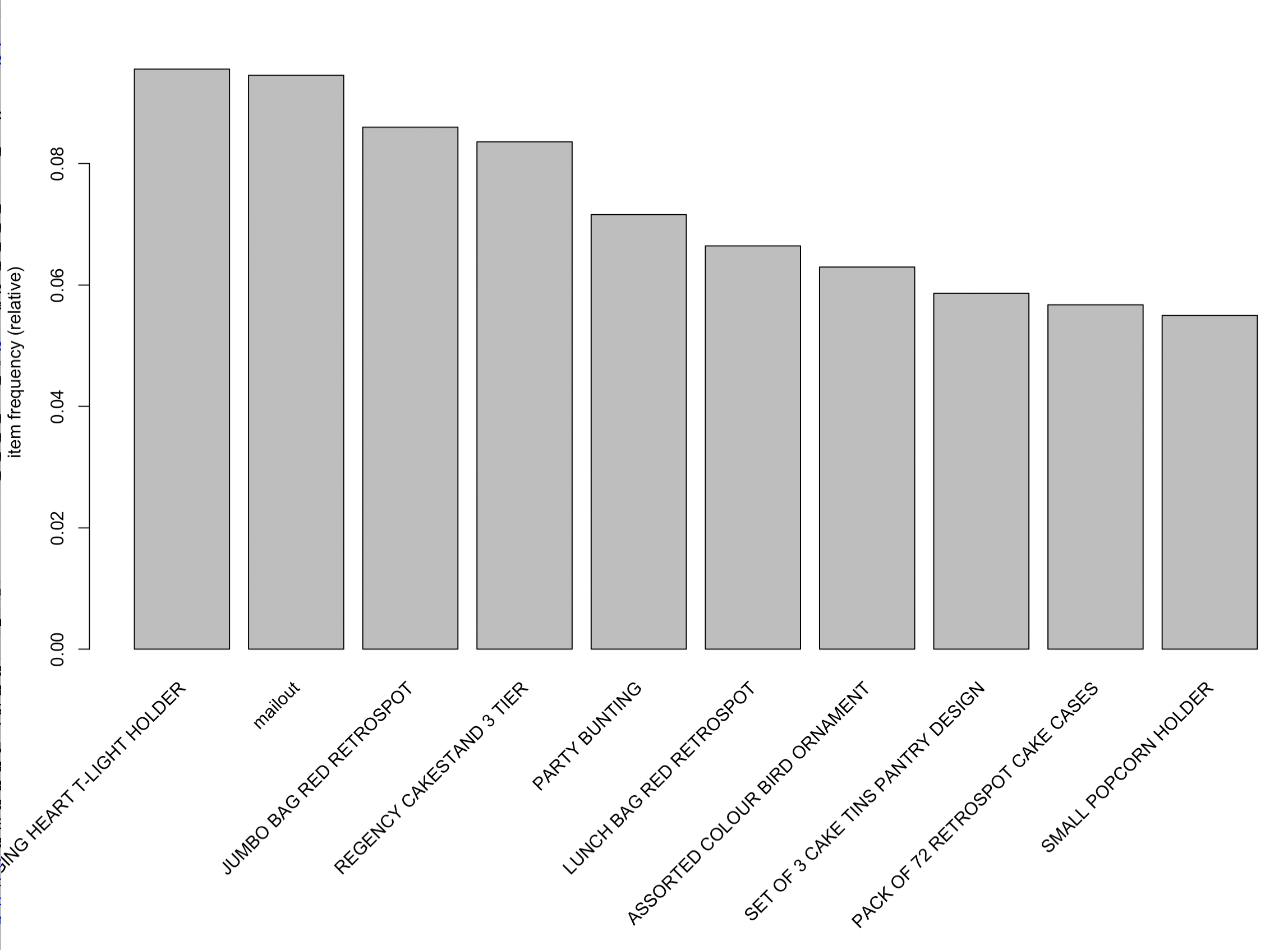
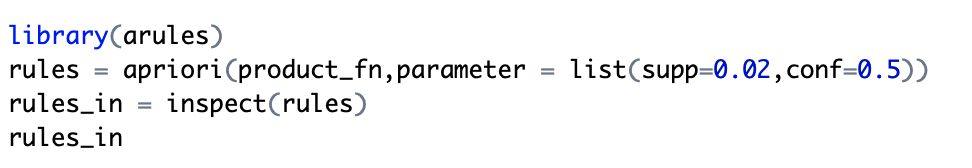


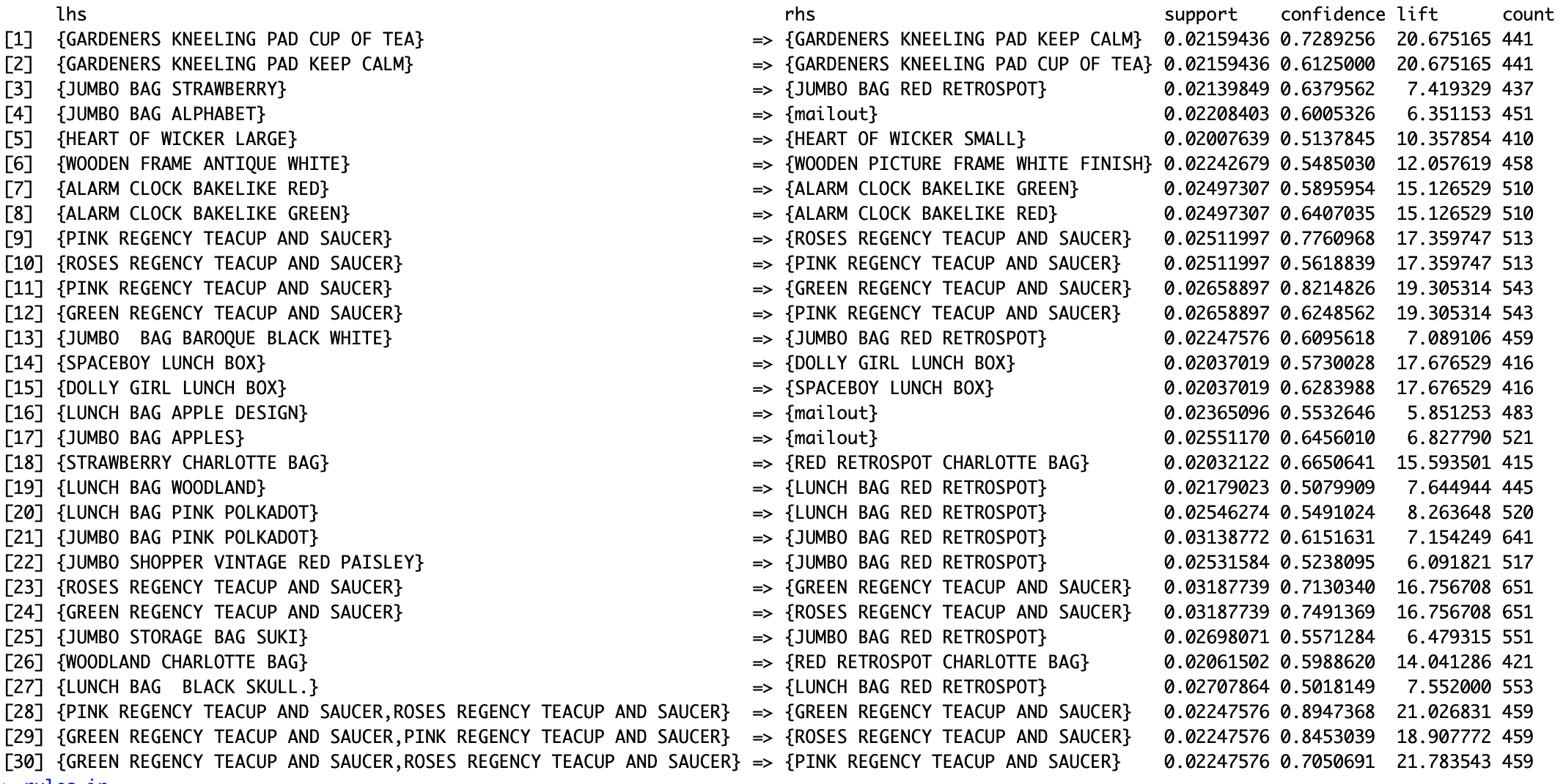
Figure 4 Products with Highest Support

# Association Mining

Step 1:

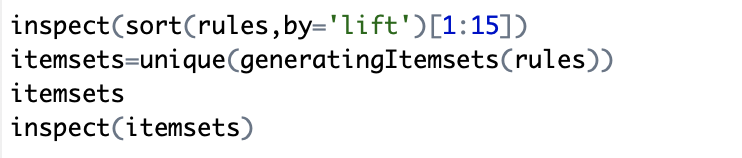
We tried different support and confidence values. Now that we understand that the support value is the ratio of records with the products or the sets of products in all records, and we also know that the dataset is large containing 20422 records and 3916 types of items, so the support value should be small so we can find any rules based on that. If the support value is more than 0.04, there will be no rules generated. Therefore, we choose 0.02 as the support values. Also, we want the confidence, which is the value of the support of the rhs and lhs divided by the support of rhs, the percentage that the records including rhs and lhs in the records including lhs, more than 0.5 to discover convincing rules. As a result, we find 30 rules in this dataset. The related products are mostly baking and cooking items.

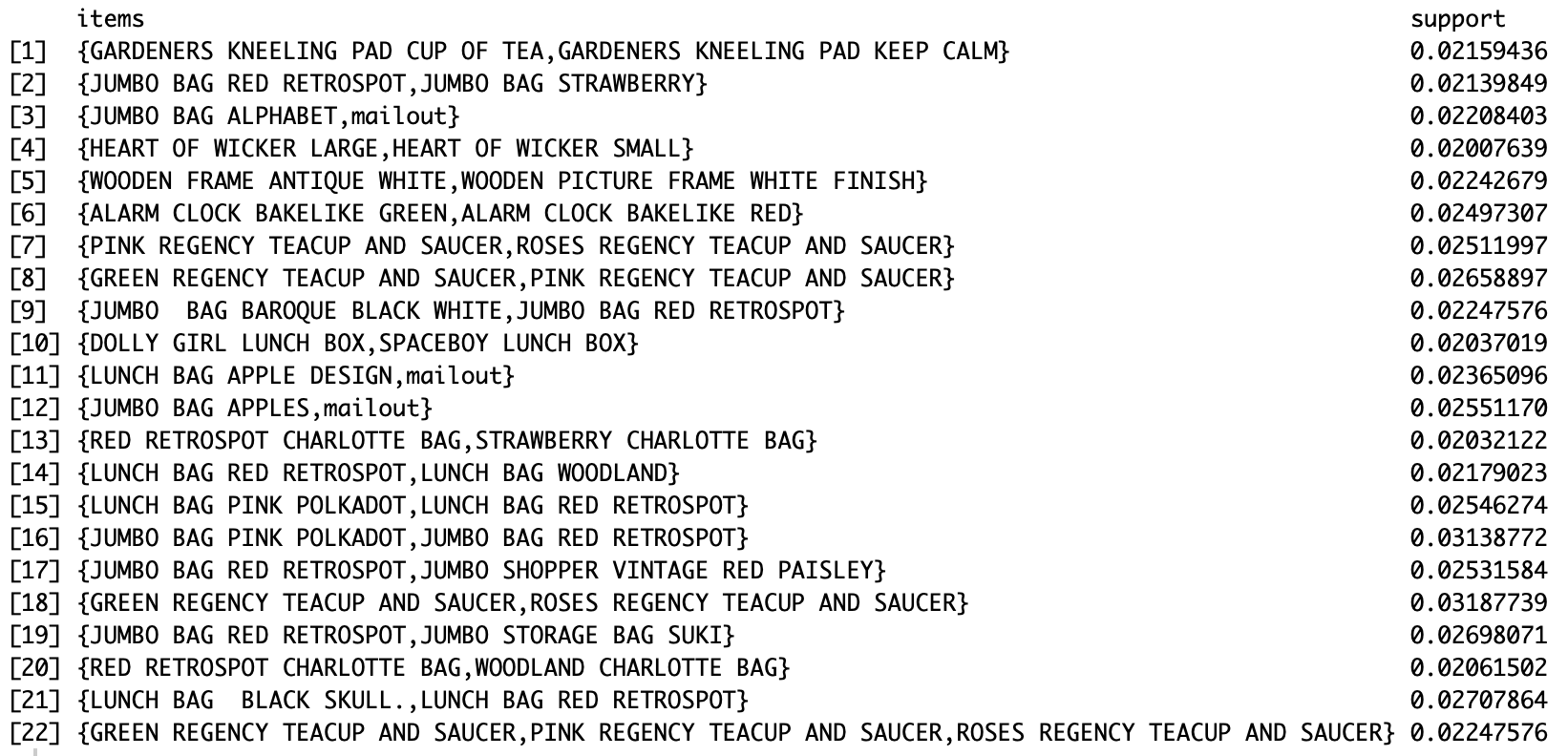




Step 2:

We ranked these sets and checked on the support values.





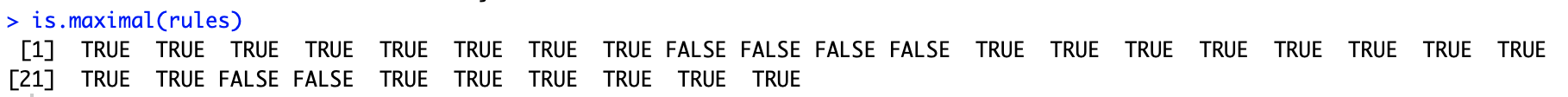
Step 3:

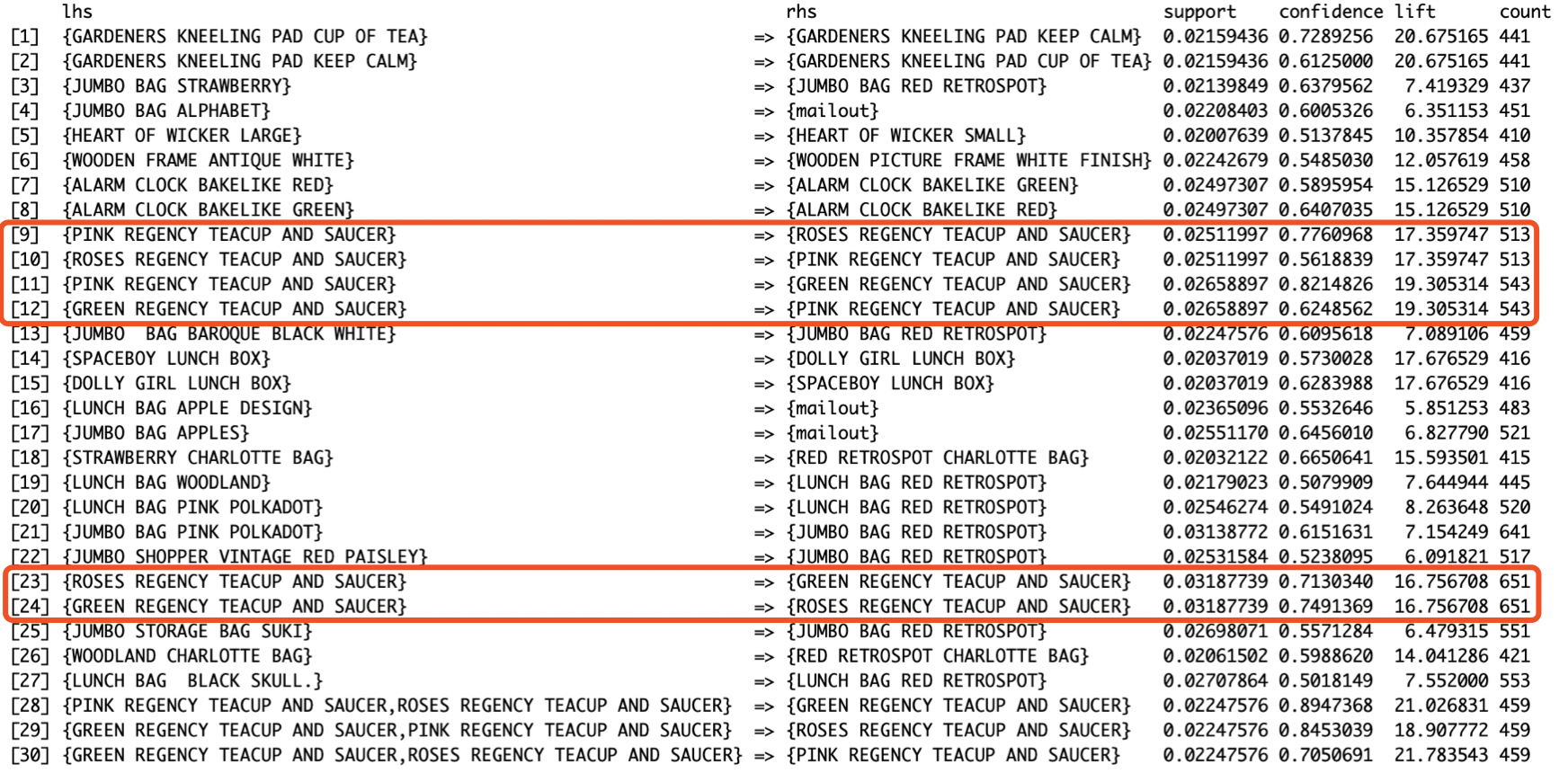
We can also generate the maximal sets and check the if the previous sets of products is maximal sets. Some of them are in maximal sets.



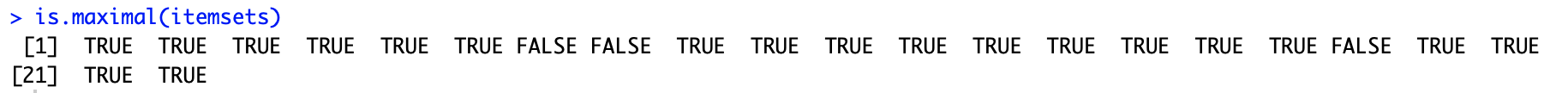


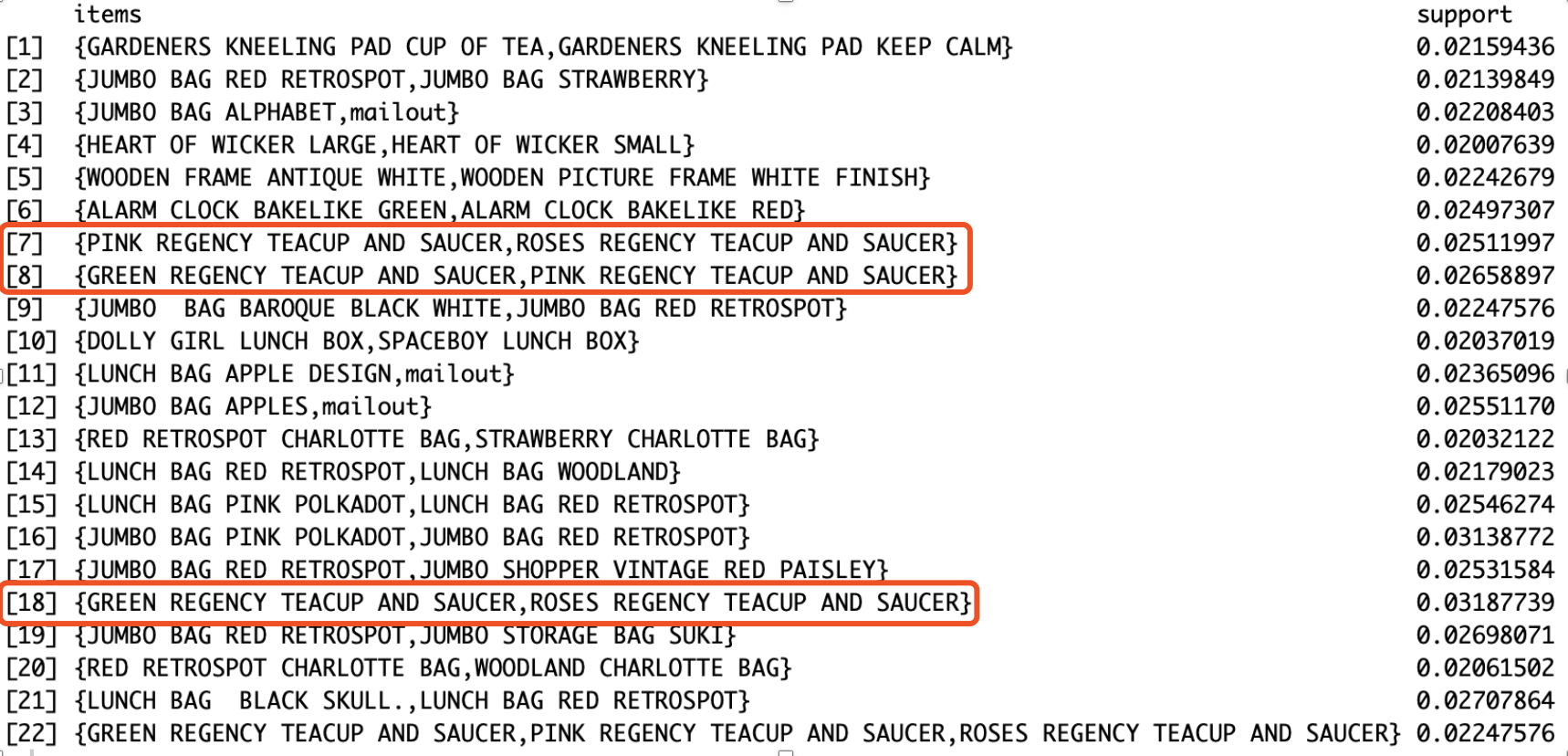












# Conclusion

This online retail dataset is definitely not a good dataset for association rule mining, since the association rule mostly is between same product but different color or pattern or size, for example gardeners kneeling pad cup of tea and gardeners kneeling pad keep calm is customer try to buy two different patterns together. Another example: alarm clock bakelike red and alarm clock bakelike green is customer try to buy two different colors of the clock together. One more example: heart of wicker large and heart of wicker small is customer try to buy two different of wicker together to have a set. Because most association relation is between same product, our association rule does not have too much meaningful insight, but we can learn their customers’ purchase habit is purchased same product with different color, pattern or size, we can suggest this online store to promote their products by applying buy three get one free, buy two get one on half price, etc to boost its sale.

# Reference

[1] *Package ‘RMySQL’* (March 4, 2019) available at

<https://cran.r-project.org/web/packages/RMySQL/RMySQL.pdf>

# Appendix

**Data Preparation**

library("RMySQL")

library("lubridate")

conn = dbConnect(MySQL(),user = 'root', password = 'wangjiye920116', dbname = 'dm\_a3', host = 'localhost')

dbListTables(conn)

num = nrow(neg\_records)

rows = 0;

dbClearResult(dbListResults(conn)[[1]])

for(j in 0:10){

start = j\*1000+1

end = (j+1)\*1000

for(i in start:end){

record = neg\_records[i,]

tmpStockCodeR = record[2]

tmpStockCode = tmpStockCodeR[1,1]

tmpQuantity = record[4]\*(-1)

tmpUnitPrice = record[6]

tmpCustomerID = record[7]

tmpDateTimeR = record[8]

tmpDateTime = toString(tmpDateTimeR[1,1])

tmpDate = ymd\_hms(tmpDateTime)-days(30)

tmpquery = paste(sep="","delete from product\_detail where StockCode = '",tmpStockCode,"' and Quantity = '",

tmpQuantity,"' and CustomerID = '",tmpCustomerID,"' and InvoiceDateTime < '",tmpDateTime,

"' and InvoiceDateTime >'",tmpDate,"' limit 1;")

dbSendStatement(conn,tmpquery)

print(tmpquery)

}

}

**Analysis**

install.packages("plyr", dependencies= TRUE)

install.packages("arules")

library(plyr)

library(data.table)

library(dplyr)

library(arules)

item = product\_clean[,2:3]

item\_name = item[!duplicated(item$StockCode),]

product\_clean\_new = left\_join(product\_clean,item\_name,c("StockCode" = "StockCode"))

sorted <- product\_clean\_new[order(product\_clean$InvoiceNo),]

product\_am <- ddply(sorted,'InvoiceNo',function(groupby)c(ItemName=paste(groupby$Description.y,collapse=',')))

product\_am$InvoiceNo = NULL

write.table(product\_am,"~/Desktop/Courses/Data\_and\_Text\_Mining/A3/product\_am.csv", quote=FALSE, row.names = FALSE, col.names = FALSE)

product\_fn = read.transactions("~/Desktop/Courses/Data\_and\_Text\_Mining/A3/product\_am.csv",format="basket",sep=",")

summary(product\_fn)

itemFrequencyPlot(product\_fn, topN=10)

rules = apriori(product\_fn,parameter = list(supp=0.02,conf=0.5))

rules\_in = inspect(rules)

rules\_in

maximal.sets = apriori(product\_fn, parameter=list(supp=0.02, conf=0.5, target="maximally frequent itemset"))

inspect(maximal.sets)

is.maximal(itemsets)

is.maximal(rules)

inspect(sort(rules,by='lift')[1:15])

itemsets=unique(generatingItemsets(rules))

itemsets

inspect(itemsets)

# Captions

[Figure 1 Data source 2](#_Toc11958925)

[Figure 2 Data Cleaning 3](#_Toc11958926)

[Figure 3 Remove Return Order 4](#_Toc11958927)

[Figure 4 Products with Highest Support 6](#_Toc11958928)