Apriori Algorithm

Mid Term Project

Course: CS-634

Name: Neha Singh

Email: nn34@njit.edu

Code:

While running the code in cmd prompt, the name and path of datasets should be given according to your system's location where project is placed.

Libraries used:

Permutations and combinations libraries are used to create combinations for in pairs of 2,2 or 3,3 for

K=2 and K=3 after elimination of items based on minimum support provided by users.

```
from itertools import permutations
from itertools import combinations
```

File (dataset) Input:

Here, we need to provide input as dataset name with 'txt' extension

i.e. dataset should be saved in notepad format. The fileobject will read your file as indicated by 'r' and input_data will read line by line.

```
## Main program to run Algorithm using the functions defined

print("\n","Welcome to Anusha's Apriori Algorithm Implementation. ", '\n',
      "This program will take input data of transactions from you and ", '\n',
      "give you the most frequent association rules with your product lines in return. ", '\n',
      "This can give you an idea about what products sell in a combination following them can maximize your profit. ", "\n")


filename = input("Enter File name with '.txt' extension:")
fileobject = open(filename, "r", encoding="UTF-8",errors="surrogateescape")
input_data = fileobject.readlines()
```

Cleaning of data

Now, final data is the list in which transformed data (in form of list) will be stored. All the unnecessary

spaces in dataset will be replaced by none and have to differentiate between each transaction by

comma that is given in split. Here comma will act as a delimiter.

```
## Transforming the dataset
final_data = []
for lines in range(1,len(input_data)):
    input_data[lines] = input_data[lines].replace("\n","")
    final_data.append(input_data[lines].split(','))
```

Printing Transformed data:

We have indicated final_data to pick data after first element so that it doesn't pick Trans1, Trans2 and so on because it was picking each row after comma, so as to clean that part I wrote 1:

```
## Transforming final_data
for i in range(len(final_data)):
  final_data[i] = final_data[i][1:]

print("\n","This is the dataset you entered:", "\n", "\n", final_data, '\n')
```

Taking Input for confidence and support:

Now, we must consider the minimum support and confidence to evaluate the results.

```
## Declaring all the required variables
min_sup = int(input("Enter Minimum Support:"))
print("\n")
min_conf = int(input("Enter Minimum Confidence:"))
print("\n")
```

Count of Items in each Transaction:

I have defined all the functions on top and then called them below for my main. First Function here is to count all unique items.

1. (set) is used to remove redundant items.
2. len here will count the items in i.
3. Sorted will sort in ascending order.
4. uniquelist will have count of each item present in dataset.

```
## Find the count of unique items and make a dictionary
def uniquelist(final_data):
    mylist = []
    for i in final_data:
        for j in range(len(i)):
            mylist.append(i[j])
    uniquelist = set(mylist)
    uniquelist = sorted(uniquelist)
    return uniquelist, mylist
```

Creating a Dictionary:

After getting count of each item in each transaction, we represent that count in key:value form and for

that representation we have used dictionary data structure.

- firstdict{} will have key:value for item sets created for k=1

```
## Performing Iteration 1
def iteration1(mylist, uniquelist):
    firstdict = {}
    for i in uniquelist:
        firstdict[i] = mylist.count(i)
    return firstdict
```

Elimination of Items based on Minimum Support:

Support_eliminate is the function created to store the dictionary named here as mydict2 that will have the items only that met the minimum support requirement.

- firstdict is called as parameter that will be having the count of each item along with item itself.
- keylist will the item name and valuelist will have the item count.
- Formula for support is count of transactions for each item/ total number of transactions count so we took min_sup from user and after getting the percentage we compared it from which limituser gave then eliminated those items that did not meet the minimum support requirement
- mydict2 will have left out item sets that meets the min_sup.

```
## Eliminate those items not satisfying minimum support
def support_eliminate(firstdict):
    keylist = list(firstdict.keys())
    valuelist = list(firstdict.values())
    mydict2 = {}
    # support_dict = {}
    count = 0

    for i in valuelist:
        support = round(i * 100 / total_trxns, 2)
        count += 1

        # print(support, min_sup)
        if support >= min_sup:
            # firstdict.pop(keylist[count])
            mydict2[keylist[count - 1]] = i
            # support_dict[keylist[count-1]] = support
    return mydict2
```

Moving forward to K=1 where we create combination of elements left out after minimum support given by user:

I have used built-in function combination to create combinations for k=1.

- create_combinations will have mydict2 that has all itemsets that met the min_sup.
- I stored combinations created from mydict2 in comb variable.
- I then stored dictionary based itemset in up_comb which is a list
- Finally storing key:value that is dictionary type data in list called as up_comb

```
## Performing Iteration 2
def iter2(final_data, up_comb):
    updated_dict = {}
    for i in range(len(final_data)):
        c2=0
        for j in range(len(up_comb)):
            if (up_comb[j][0] and up_comb[j][1]) in final_data[i]:
                c2+=1
                updated_dict[up_comb[j]] = c2
    return updated_dict
```

Implementation in Main for K=1:

- Sup_elim1 will have the processed data to display after elimination.
- Iteration will have data stored in form of key: value that is dictionary
- Support_eliminate function will have all implementation for elimination based on user input for minimum support.

```
total_trxns = len(final_data)

uniques, full_list = uniquelist(final_data)

iteration1 = iteration1(full_list, uniques)

sup_elim1 = support_eliminate(iteration1)

print("\n","This is the result after first iteration:", "\n", "\n", sup_elim1, '\n')
```

```
new_combs = create_combinations1(sup_elim1, 2)
```

Moving forward to L=2 where we create count elements in each transaction that come together in possible combination in each Transaction:

- Visualize this step where we are counting A,C or A,C coming together in each transaction.
- Up_comb is the list where we have eliminated items from k=1 and final data is our transformed data set then we count those items coming together in each row and store in updated_dict.

Implementation in Main for K=2:

```
iteration2 = iter2(final_data, new_combs)

sup_elim2 = support_eliminate(iteration2)

print("\n","This is the result after second iteration:", "\n", "\n", sup_elim2, '\n')

mylist2iter = list(sup_elim2.keys())

##used this to put unique pairs in tuple to make it in a single list
listafter2 = []
for i in range(len(mylist2iter)):
    listafter2.append(mylist2iter[i][0])
    listafter2.append(mylist2iter[i][1])

listafter2 = set(listafter2)
```

Moving forward to K=2 where we create combinations of two:

- Create_combinations2 will have combination of two item sets appearing together in eachtransaction satisfying min_sup.

```
## Creating combinations of items in Iteration 2
def create_combinations2(targlist, n):
    comb = combinations(targlist, n)
    up_comb = []
    for i in comb:
        up_comb.append(i)

    return up_comb
```

Moving forward to K=3 where we create combinations of three:

We list all elements that comes in pair of three in each transaction.

```
## Performing Iteration 3
def iter3(final_data, combs_3):
    updated_dict3 = {}
    for i in range(len(final_data)):
        c2=0
        for j in range(len(combs_3)):
            if (combs_3[j][0] and combs_3[j][1] and combs_3[j][2]) in final_data[i]:
              c2= c2+1
              updated_dict3[combs_3[j]] = c2

    return updated_dict3
```

Implementation in Main for K=3:

```
combs_3 = create_combinations2(listafter2, 3)

iteration3 = iter3(final_data, combs_3)

sup_elim3 = support_eliminate(iteration3)

print("\n","This is the result after third iteration:", "\n", "\n", sup_elim3, '\n')
```

****

Implementation of Association Rules:

- Highkey is tuple created in main that is the most frequent
- Highval has the value which is count of how many times items appeared
- Since, rules are derivate from most frequent set so we take the possible combinations from most frequent set and divide by the set itself
- We make combinations of 2 from most frequent set that is why comb2s variable is storing highkey that is value of 2 pairs and combination built-in function is making all possible combinations.
- Left item is the combination created whereas right one is the element left out from created combination that is why set is used
- If condition eliminates rules/item sets based on minimum confidence.

```
## List of Frequent itemsets and Association rules
def assoc_rules(highkey, highval):
    support_dict = {}
    count = 0
    item_set = set(highkey)

    comb2s = combinations(highkey, 2)
    for i in comb2s:
        print("Association Rule for Pair - ", i)
        print("__Rule__", "__Confidence__")
        i = tuple(sorted(i))
        left_item = i
        right_item = tuple(item_set - set(i))

        # print(left_item, "=>" ,right_item)

        denom2 = sup_elim2[i]
        conf = (highval // denom2) * 100

        if conf >= min_conf:
            print(left_item, "=>", right_item, conf, "This Rule is Acceptable")
        else:
            print(left_item, "=>", right_item, conf, "This Rule is Rejected")
```

Again, association rules are taken for rules created as a->{b,c} where left item is the remaining item from item set pair created.

```
comb1s = highkey
for i in comb1s:
    print("Association Rule for Pair - ", i)
    print("__Rule__", "__Confidence__")

    left_item = i
    right_item = tuple(item_set - set(i))
    denom1 = sup_elim1[i]
    conf1 = highval * 100 // denom1

    # print(left_item, right_item)
    if conf1 >= min_conf:
        print(left_item, "=>", right_item, conf1, "This Rule is Acceptable")
    else:
        print(left_item, "=>", right_item, conf1, "This Rule is Rejected")
print()
```

Implementation in Main:

```
highkey = tuple(sorted(list(iteration3.keys())[list(iteration3.values()).index(max(iteration3.values()))]))
highval = max(iteration3.values())

assoc_rules(highkey, highval)
```

****

How to Run:

➔ Example: Python my_midtermproj.py

It will ask for Dataset:

Give any dataset then you want to give in txt:

I have used 5 datasets here:

- Nike.txt
- BestBuy.txt
- Amazon.txt
- Kmart.txt
- Generic.txt (for clarity)

BEST BUY

```
C:\Windows\System32\cmd.exe - Python  my_midtermproj.py

C:\Users\Administrator\AppData\Local\Programs\Python\Python39\Scripts\Project>Python my_midtermproj.py

 Welcome to Anusha's Apriori Algorithm Implementation.
 This program will take input data of transactions from you and
 give you the most frequent association rules with your product lines in return.
 This can give you an idea about what products sell in a combination following them can maximize your profit.

Enter File name with '.txt' extension:BestBuy.txt

 This is the dataset you entered:

 [[' Desk Top', ' Printer', ' Flash Drive', ' Microsoft Office', ' Speakers', ' Anti-Virus'], [' Lab Top', ' Flash Drive', ' Microsoft Office', ' Lab Top Case', ' Anti-
Virus'], [' Lab Top', ' Printer', ' Flash Drive', ' Microsoft Office', ' Anti-Virus', ' Lab Top Case', ' External Hard-Drive'], [' Lab Top', ' Printer', ' Flash Drive',
 ' Anti-Virus', ' External Hard-Drive', ' Lab Top Case'], [' Lab Top', ' Flash Drive', ' Lab Top Case', ' Anti-Virus'], [' Lab Top', ' Printer', ' Flash Drive', ' Micro
soft Office'], [' Desk Top', ' Printer', ' Flash Drive', ' Microsoft Office'], [' Lab Top', ' External Hard-Drive', ' Anti-Virus'], [' Desk Top', ' Printer', ' Flash Dr
ive', ' Microsoft Office', ' Lab Top Case', ' Anti-Virus', ' Speakers', ' External Hard-Drive'], [' Digital Camera', ' Lab Top', ' Desk Top', ' Printer', ' Flash Drive'
, ' Microsoft Office', ' Lab Top Case', ' Anti-Virus', ' External Hard-Drive', ' Speakers'], [' Lab Top', ' Desk Top', ' Lab Top Case', ' External Hard-Drive', ' Speake
rs', ' Anti-Virus'], [' Digital Camera', ' Lab Top', ' Lab Top Case', ' External Hard-Drive', ' Anti-Virus', ' Speakers'], [' Digital Camera', ' Speakers'], [' Digital
Camera', ' Desk Top', ' Printer', ' Flash Drive', ' Microsoft Office'], [' Printer', ' Flash Drive', ' Microsoft Office', ' Anti-Virus', ' Lab Top Case', ' Speakers', '
 External Hard-Drive'], [' Digital Camera', ' Flash Drive', ' Microsoft Office', ' Anti-Virus', ' Lab Top Case', ' External Hard-Drive', ' Speakers'], [' Digital Camera
', ' Lab Top', ' Lab Top Case'], [' Digital Camera', ' Lab Top Case', ' Speakers'], [' Digital Camera', ' Lab Top', ' Printer', ' Flash Drive', ' Microsoft Office', ' S
peakers', ' Lab Top Case', ' Anti-Virus'], [' Digital Camera', ' Lab Top', ' Speakers', ' Anti-Virus', ' Lab Top Case']]

Enter Minimum Support:
```

➔Now give minimum support

➔Then, minimum confidence

This is how it will display support:

```
Enter File name with '.txt' extension:BestBuy.txt

 This is the dataset you entered:

 [[' Desk Top', ' Printer', ' Flash Drive', ' Microsoft Office', ' Speakers', ' Anti-Virus'], [' Lab Top', ' Flash Drive', ' Microsoft Office', ' Lab Top Case', ' Anti-
Virus'], [' Lab Top', ' Printer', ' Flash Drive', ' Microsoft Office', ' Anti-Virus', ' Lab Top Case', ' External Hard-Drive'], [' Lab Top', ' Printer', ' Flash Drive',
 ' Anti-Virus', ' External Hard-Drive', ' Lab Top Case'], [' Lab Top', ' Flash Drive', ' Lab Top Case', ' Anti-Virus'], [' Lab Top', ' Printer', ' Flash Drive', ' Micro
soft Office'], [' Desk Top', ' Printer', ' Flash Drive', ' Microsoft Office'], [' Lab Top', ' External Hard-Drive', ' Anti-Virus'], [' Desk Top', ' Printer', ' Flash Dr
ive', ' Microsoft Office', ' Lab Top Case', ' Anti-Virus', ' Speakers', ' External Hard-Drive'], [' Digital Camera', ' Lab Top', ' Desk Top', ' Printer', ' Flash Drive'
, ' Microsoft Office', ' Lab Top Case', ' Anti-Virus', ' External Hard-Drive', ' Speakers'], [' Lab Top', ' Desk Top', ' Lab Top Case', ' External Hard-Drive', ' Speake
rs', ' Anti-Virus'], [' Digital Camera', ' Lab Top', ' Lab Top Case', ' External Hard-Drive', ' Anti-Virus', ' Speakers'], [' Digital Camera', ' Speakers'], [' Digital
Camera', ' Desk Top', ' Printer', ' Flash Drive', ' Microsoft Office'], [' Printer', ' Flash Drive', ' Microsoft Office', ' Anti-Virus', ' Lab Top Case', ' Speakers', '
 External Hard-Drive'], [' Digital Camera', ' Flash Drive', ' Microsoft Office', ' Anti-Virus', ' Lab Top Case', ' External Hard-Drive', ' Speakers'], [' Digital Camera
', ' Lab Top', ' Lab Top Case'], [' Digital Camera', ' Lab Top Case', ' Speakers'], [' Digital Camera', ' Lab Top', ' Printer', ' Flash Drive', ' Microsoft Office', ' S
peakers', ' Lab Top Case', ' Anti-Virus'], [' Digital Camera', ' Lab Top', ' Speakers', ' Anti-Virus', ' Lab Top Case']]

Enter Minimum Support:26


Enter Minimum Confidence:11



 This is the result after first iteration:

{' Anti-Virus': 14, ' Desk Top': 6, ' Digital Camera': 9, ' External Hard-Drive': 9, ' Flash Drive': 13, ' Lab Top': 12, ' Lab Top Case': 14, ' Microsoft Office': 11,
 ' Printer': 10, ' Speakers': 11}


 This is the result after second iteration:

 {(' Anti-Virus', ' Printer'): 6, (' Desk Top', ' Flash Drive'): 9, (' Desk Top', ' Microsoft Office'): 12, (' Desk Top', ' Printer'): 13, (' Desk Top', ' Speakers'): 8
, (' Digital Camera', ' Flash Drive'): 15, (' Digital Camera', ' Microsoft Office'): 18, (' Digital Camera', ' Printer'): 19, (' Digital Camera', ' Speakers'): 11, (' E
xternal Hard-Drive', ' Flash Drive'): 21, (' External Hard-Drive', ' Microsoft Office'): 24, (' External Hard-Drive', ' Printer'): 25, (' External Hard-Drive', ' Speake
rs'): 14, (' Flash Drive', ' Microsoft Office'): 29, (' Flash Drive', ' Printer'): 30, (' Flash Drive', ' Speakers'): 17, (' Lab Top', ' Microsoft Office'): 33, (' Lab
Top', ' Printer'): 34, (' Lab Top', ' Speakers'): 19, (' Lab Top Case', ' Microsoft Office'): 36, (' Lab Top Case', ' Printer'): 37, (' Lab Top Case', ' Speakers'): 20,
 (' Microsoft Office', ' Printer'): 39, (' Microsoft Office', ' Speakers'): 21, (' Printer', ' Speakers'): 22, (' Desk Top', ' Lab Top'): 6, (' Desk Top', ' Lab Top Cas
e'): 7, (' Digital Camera', ' Lab Top'): 9, (' Digital Camera', ' Lab Top Case'): 10, (' External Hard-Drive', ' Lab Top'): 12, (' External Hard-Drive', ' Lab Top Case'
): 13, (' Flash Drive', ' Lab Top'): 15, (' Flash Drive', ' Lab Top Case'): 16, (' Lab Top', ' Lab Top Case'): 18, (' Desk Top', ' External Hard-Drive'): 8, (' Digital
Camera', ' External Hard-Drive'): 13}
```

```
This is the result after third iteration:

 {(' Lab Top', ' Lab Top Case', ' Microsoft Office'): 6, (' Lab Top', ' Anti-Virus', ' Flash Drive'): 8, (' Lab Top', ' Anti-Virus', ' Printer'): 9, (' Lab Top', ' Anti
-Virus', ' Microsoft Office'): 11, (' Lab Top', ' Anti-Virus', ' Desk Top'): 10, (' Lab Top', ' Digital Camera', ' Flash Drive'): 12, (' Lab Top', ' Digital Camera', '
Printer'): 13, (' Lab Top', ' Digital Camera', ' Speakers'): 6, (' Lab Top', ' Digital Camera', ' Microsoft Office'): 15, (' Lab Top', ' Digital Camera', ' Desk Top'):
14, (' Lab Top', ' Flash Drive', ' Printer'): 16, (' Lab Top', ' Flash Drive', ' Speakers'): 7, (' Lab Top', ' Flash Drive', ' Microsoft Office'): 18, (' Lab Top', ' Fl
ash Drive', ' Desk Top'): 17, (' Lab Top', ' Printer', ' Speakers'): 8, (' Lab Top', ' Printer', ' Microsoft Office'): 20, (' Lab Top', ' Printer', ' Desk Top'): 19, ('
 Lab Top', ' Speakers', ' Microsoft Office'): 21, (' Lab Top', ' Speakers', ' Desk Top'): 21, (' Lab Top', ' Microsoft Office', ' Desk Top'): 22, (' Lab Top', ' Externa
l Hard-Drive', ' Desk Top'): 23, (' Lab Top Case', ' Anti-Virus', ' Flash Drive'): 23, (' Lab Top Case', ' Anti-Virus', ' Printer'): 24, (' Lab Top Case', ' Anti-Virus'
, ' Speakers'): 10, (' Lab Top Case', ' Anti-Virus', ' Microsoft Office'): 26, (' Lab Top Case', ' Anti-Virus', ' Desk Top'): 28, (' Lab Top Case', ' Digital Camera', '
 Flash Drive'): 27, (' Lab Top Case', ' Digital Camera', ' Printer'): 28, (' Lab Top Case', ' Digital Camera', ' Speakers'): 11, (' Lab Top Case', ' Digital Camera', '
Microsoft Office'): 30, (' Lab Top Case', ' Digital Camera', ' Desk Top'): 32, (' Lab Top Case', ' Flash Drive', ' Printer'): 31, (' Lab Top Case', ' Flash Drive', ' Sp
eakers'): 12, (' Lab Top Case', ' Flash Drive', ' Microsoft Office'): 33, (' Lab Top Case', ' Flash Drive', ' Desk Top'): 35, (' Lab Top Case', ' Printer', ' Speakers')
: 13, (' Lab Top Case', ' Printer', ' Microsoft Office'): 35, (' Lab Top Case', ' Printer', ' Desk Top'): 37, (' Lab Top Case', ' Speakers', ' Microsoft Office'): 36, (
' Lab Top Case', ' Speakers', ' Desk Top'): 39, (' Lab Top Case', ' Microsoft Office', ' Desk Top'): 40, (' Lab Top Case', ' External Hard-Drive', ' Desk Top'): 41, ('
Anti-Virus', ' Digital Camera', ' Flash Drive'): 37, (' Anti-Virus', ' Digital Camera', ' Printer'): 38, (' Anti-Virus', ' Digital Camera', ' Speakers'): 14, (' Anti-Vi
rus', ' Digital Camera', ' Microsoft Office'): 40, (' Anti-Virus', ' Digital Camera', ' Desk Top'): 45, (' Anti-Virus', ' Flash Drive', ' Printer'): 41, (' Anti-Virus',
 ' Flash Drive', ' Speakers'): 15, (' Anti-Virus', ' Flash Drive', ' Microsoft Office'): 43, (' Anti-Virus', ' Flash Drive', ' Desk Top'): 48, (' Anti-Virus', ' Printer
', ' Speakers'): 16, (' Anti-Virus', ' Printer', ' Microsoft Office'): 45, (' Anti-Virus', ' Printer', ' Desk Top'): 50, (' Anti-Virus', ' Speakers', ' Microsoft Office
'): 46, (' Anti-Virus', ' Speakers', ' Desk Top'): 52, (' Anti-Virus', ' Microsoft Office', ' Desk Top'): 53, (' Anti-Virus', ' External Hard-Drive', ' Desk Top'): 54,
(' Digital Camera', ' Flash Drive', ' Printer'): 47, (' Digital Camera', ' Flash Drive', ' Speakers'): 17, (' Digital Camera', ' Flash Drive', ' Microsoft Office'): 49,
 (' Digital Camera', ' Flash Drive', ' Desk Top'): 57, (' Digital Camera', ' Printer', ' Speakers'): 18, (' Digital Camera', ' Printer', ' Microsoft Office'): 51, (' Di
gital Camera', ' Printer', ' Desk Top'): 59, (' Digital Camera', ' Speakers', ' Microsoft Office'): 52, (' Digital Camera', ' Speakers', ' Desk Top'): 61, (' Digital Ca
mera', ' Microsoft Office', ' Desk Top'): 62, (' Digital Camera', ' External Hard-Drive', ' Desk Top'): 63, (' Flash Drive', ' Printer', ' Speakers'): 19, (' Flash Driv
e', ' Printer', ' Microsoft Office'): 54, (' Flash Drive', ' Printer', ' Desk Top'): 65, (' Flash Drive', ' Speakers', ' Microsoft Office'): 55, (' Flash Drive', ' Spea
kers', ' Desk Top'): 67, (' Flash Drive', ' Microsoft Office', ' Desk Top'): 68, (' Flash Drive', ' External Hard-Drive', ' Desk Top'): 69, (' Printer', ' Speakers', '
Microsoft Office'): 56, (' Printer', ' Speakers', ' Desk Top'): 71, (' Printer', ' Microsoft Office', ' Desk Top'): 72, (' Printer', ' External Hard-Drive', ' Desk Top'
): 73, (' Speakers', ' Microsoft Office', ' Desk Top'): 74, (' Speakers', ' External Hard-Drive', ' Desk Top'): 75, (' Microsoft Office', ' External Hard-Drive', ' Desk
 Top'): 76, (' Lab Top', ' Lab Top Case', ' External Hard-Drive'): 6, (' Lab Top', ' Anti-Virus', ' External Hard-Drive'): 11, (' Lab Top', ' Digital Camera', ' Externa
l Hard-Drive'): 15, (' Lab Top', ' Flash Drive', ' External Hard-Drive'): 18, (' Lab Top', ' Printer', ' External Hard-Drive'): 21, (' Lab Top', ' Speakers', ' External
 Hard-Drive'): 23, (' Lab Top', ' Microsoft Office', ' External Hard-Drive'): 24, (' Lab Top Case', ' Anti-Virus', ' External Hard-Drive'): 29, (' Lab Top Case', ' Digi
tal Camera', ' External Hard-Drive'): 33, (' Lab Top Case', ' Flash Drive', ' External Hard-Drive'): 36, (' Lab Top Case', ' Printer', ' External Hard-Drive'): 39, (' L
ab Top Case', ' Speakers', ' External Hard-Drive'): 41, (' Lab Top Case', ' Microsoft Office', ' External Hard-Drive'): 42, (' Anti-Virus', ' Digital Camera', ' Externa
l Hard-Drive'): 46, (' Anti-Virus', ' Flash Drive', ' External Hard-Drive'): 49, (' Anti-Virus', ' Printer', ' External Hard-Drive'): 52, (' Anti-Virus', ' Speakers', '
 External Hard-Drive'): 54, (' Anti-Virus', ' Microsoft Office', ' External Hard-Drive'): 55, (' Digital Camera', ' Flash Drive', ' External Hard-Drive'): 58, (' Digita
l Camera', ' Printer', ' External Hard-Drive'): 61, (' Digital Camera', ' Speakers', ' External Hard-Drive'): 63, (' Digital Camera', ' Microsoft Office', ' External Ha
rd-Drive'): 64, (' Flash Drive', ' Printer', ' External Hard-Drive'): 67, (' Flash Drive', ' Speakers', ' External Hard-Drive'): 69, (' Flash Drive', ' Microsoft Office
', ' External Hard-Drive'): 70, (' Printer', ' Speakers', ' External Hard-Drive'): 72, (' Printer', ' Microsoft Office', ' External Hard-Drive'): 73, (' Speakers', ' Mi
crosoft Office', ' External Hard-Drive'): 74, (' Lab Top Case', ' Anti-Virus', ' Digital Camera'): 9}
```


Association rules for BestBuy display:

```
Association Rule for Pair -  (' Desk Top', ' External Hard-Drive')
__Rule__ __Confidence__
(' Desk Top', ' External Hard-Drive') => (' Microsoft Office',) 900 This Rule is Acceptable
Association Rule for Pair -  (' Desk Top', ' Microsoft Office')
__Rule__ __Confidence__
(' Desk Top', ' Microsoft Office') => (' External Hard-Drive',) 600 This Rule is Acceptable
Association Rule for Pair -  (' External Hard-Drive', ' Microsoft Office')
__Rule__ __Confidence__
(' External Hard-Drive', ' Microsoft Office') => (' Desk Top',) 300 This Rule is Acceptable
Association Rule for Pair -   Desk Top
__Rule__ __Confidence__
 Desk Top => (' Desk Top', ' External Hard-Drive', ' Microsoft Office') 1266 This Rule is Acceptable
Association Rule for Pair -   External Hard-Drive
__Rule__ __Confidence__
 External Hard-Drive => (' Desk Top', ' External Hard-Drive', ' Microsoft Office') 844 This Rule is Acceptable
Association Rule for Pair -   Microsoft Office
__Rule__ __Confidence__
 Microsoft Office => (' Desk Top', ' External Hard-Drive', ' Microsoft Office') 690 This Rule is Acceptable
```

Amazon:

Support:

```
C:\Users\Administrator\AppData\Local\Programs\Python\Python39\Scripts\Project>Python my_midtermproj.py

 Welcome to Anusha's Apriori Algorithm Implementation.
 This program will take input data of transactions from you and
 give you the most frequent association rules with your product lines in return.
 This can give you an idea about what products sell in a combination following them can maximize your profit.

Enter File name with '.txt' extension:Amazon.txt

 This is the dataset you entered:

 [[' A Beginner's Guide', ' Java: The Complete Reference', ' Java For Dummies', ' Android Programming: The Big Nerd Ranch'], [' A Beginner's Guide', ' Java: The Complet
e Reference', ' Java For Dummies'], [' A Beginner's Guide', ' Java: The Complete Reference', ' Java For Dummies', ' Android Programming: The Big Nerd Ranch', ' Head Fir
st Java 2nd Edition'], [' Android Programming: The Big Nerd Ranch', ' Head First Java 2nd Edition', ' Beginning Programming with Java'], [' Android Programming: The Big
 Nerd Ranch', ' Beginning Programming with Java', ' Java 8 Pocket Guide'], [' A Beginner's Guide', ' Android Programming: The Big Nerd Ranch', ' Head First Java 2nd Edi
tion'], [' A Beginner's Guide', ' Head First Java 2nd Edition', ' Beginning Programming with Java'], [' Java: The Complete Reference', ' Java For Dummies', ' Android Pr
ogramming: The Big Nerd Ranch'], [' Java For Dummies', ' Android Programming: The Big Nerd Ranch', ' Head First Java 2nd Edition', ' Beginning Programming with Java'],
 [' Beginning Programming with Java', ' Java 8 Pocket Guide', ' C++ Programming in Easy Steps'], [' A Beginner's Guide', ' Java: The Complete Reference', ' Java For Dumm
ies', ' Android Programming: The Big Nerd Ranch'], [' A Beginner's Guide', ' Java: The Complete Reference', ' Java For Dummies', ' HTML and CSS: Design and Build Websit
es'], [' A Beginner's Guide', ' Java: The Complete Reference', ' Java For Dummies', ' Java 8 Pocket Guide', ' HTML and CSS: Design and Build Websites'], [' Java For Dum
mies', ' Android Programming: The Big Nerd Ranch', ' Head First Java 2nd Edition'], [' Java For Dummies', ' Android Programming: The Big Nerd Ranch'], [' A Beginner's G
uide', ' Java: The Complete Reference', ' Java For Dummies', ' Android Programming: The Big Nerd Ranch'], [' A Beginner's Guide', ' Java: The Complete Reference', ' Jav
a For Dummies', ' Android Programming: The Big Nerd Ranch'], [' Head First Java 2nd Edition', ' Beginning Programming with Java', ' Java 8 Pocket Guide'], [' Android Pr
ogramming: The Big Nerd Ranch', ' Head First Java 2nd Edition'], [' A Beginner's Guide', ' Java: The Complete Reference', ' Java For Dummies']]
```



```
Enter Minimum Support:20


Enter Minimum Confidence:10



 This is the result after first iteration:

 {' A Beginner's Guide': 11, ' Android Programming: The Big Nerd Ranch': 13, ' Beginning Programming with Java': 6, ' Head First Java 2nd Edition': 8, ' Java 8 Pocket G
uide': 4, ' Java For Dummies': 13, ' Java: The Complete Reference': 10}


 This is the result after second iteration:

 {(' Android Programming: The Big Nerd Ranch', ' Java: The Complete Reference'): 4, (' Beginning Programming with Java', ' Java For Dummies'): 5, (' Beginning Programmi
ng with Java', ' Java: The Complete Reference'): 6, (' Head First Java 2nd Edition', ' Java For Dummies'): 7, (' Head First Java 2nd Edition', ' Java: The Complete Refe
rence'): 8, (' Java 8 Pocket Guide', ' Java For Dummies'): 9, (' Java 8 Pocket Guide', ' Java: The Complete Reference'): 10, (' Java For Dummies', ' Java: The Complete
 Reference'): 11, (' Beginning Programming with Java', ' Head First Java 2nd Edition'): 4, (' Android Programming: The Big Nerd Ranch', ' Beginning Programming with Java
'): 4, (' Android Programming: The Big Nerd Ranch', ' Java 8 Pocket Guide'): 6, (' Beginning Programming with Java', ' Java 8 Pocket Guide'): 8, (' Head First Java 2nd
 Edition', ' Java 8 Pocket Guide'): 9}


 This is the result after third iteration:

 {(' Java For Dummies', ' Java 8 Pocket Guide', ' Java: The Complete Reference'): 4, (' Java For Dummies', ' Android Programming: The Big Nerd Ranch', ' Java: The Compl
ete Reference'): 5, (' Java 8 Pocket Guide', ' Android Programming: The Big Nerd Ranch', ' Java: The Complete Reference'): 6, (' Head First Java 2nd Edition', ' Android
 Programming: The Big Nerd Ranch', ' Beginning Programming with Java'): 4, (' Head First Java 2nd Edition', ' Java: The Complete Reference', ' Beginning Programming wit
h Java'): 5, (' Java For Dummies', ' Java 8 Pocket Guide', ' Beginning Programming with Java'): 6, (' Java For Dummies', ' Android Programming: The Big Nerd Ranch', ' B
eginning Programming with Java'): 7, (' Java For Dummies', ' Java: The Complete Reference', ' Beginning Programming with Java'): 8, (' Java 8 Pocket Guide', ' Android P
rogramming: The Big Nerd Ranch', ' Beginning Programming with Java'): 9, (' Java 8 Pocket Guide', ' Java: The Complete Reference', ' Beginning Programming with Java'):
10, (' Android Programming: The Big Nerd Ranch', ' Java: The Complete Reference', ' Beginning Programming with Java'): 11}
```

Association rules for Amazon:

```
Association Rule for Pair -  (' Android Programming: The Big Nerd Ranch', ' Beginning Programming with Java')
__Rule__ __Confidence__
(' Android Programming: The Big Nerd Ranch', ' Beginning Programming with Java') => (' Java: The Complete Reference',) 200 This Rule is Acceptable
Association Rule for Pair -  (' Android Programming: The Big Nerd Ranch', ' Java: The Complete Reference')
__Rule__ __Confidence__
(' Android Programming: The Big Nerd Ranch', ' Java: The Complete Reference') => (' Beginning Programming with Java',) 200 This Rule is Acceptable
Association Rule for Pair -  (' Beginning Programming with Java', ' Java: The Complete Reference')
__Rule__ __Confidence__
(' Beginning Programming with Java', ' Java: The Complete Reference') => (' Android Programming: The Big Nerd Ranch',) 100 This Rule is Acceptable
Association Rule for Pair -   Android Programming: The Big Nerd Ranch
__Rule__ __Confidence__
 Android Programming: The Big Nerd Ranch => (' Beginning Programming with Java', ' Java: The Complete Reference', ' Android Programming: The Big Nerd Ranch') 84 This Ru
le is Acceptable
Association Rule for Pair -   Beginning Programming with Java
__Rule__ __Confidence__
 Beginning Programming with Java => (' Beginning Programming with Java', ' Java: The Complete Reference', ' Android Programming: The Big Nerd Ranch') 183 This Rule is A
cceptable
Association Rule for Pair -   Java: The Complete Reference
__Rule__ __Confidence__
 Java: The Complete Reference => (' Beginning Programming with Java', ' Java: The Complete Reference', ' Android Programming: The Big Nerd Ranch') 110 This Rule is Acce
ptable
```

NIKE:

Support:

```
C:\Users\Administrator\AppData\Local\Programs\Python\Python39\Scripts\Project>Python my_midtermproj.py

 Welcome to Anusha's Apriori Algorithm Implementation.
 This program will take input data of transactions from you and
 give you the most frequent association rules with your product lines in return.
 This can give you an idea about what products sell in a combination following them can maximize your profit.

Enter File name with '.txt' extension:Nike.txt

 This is the dataset you entered:

 [[' Running Shoe', ' Socks', ' Sweatshirts', ' Modern Pants'], [' Running Shoe', ' Socks', ' Sweatshirts'], [' Running Shoe', ' Socks', ' Sweatshirts', ' Modern Pants'
], [' Running Shoe', ' Sweatshirts', ' Modern Pants'], [' Running Shoe', ' Socks', ' Sweatshirts', ' Modern Pants', ' Soccer Shoe'], [' Running Shoe', ' Socks', ' Sweat
shirts'], [' Running Shoe', ' Socks', ' Sweatshirts', ' Modern Pants', ' Tech Pants', ' Rash Guard', ' Hoodies'], [' Swimming Shirt', ' Socks', ' Sweatshirts'], [' Swim
ming Shirt', ' Rash Guard', ' Dry Fit V-Nick', ' Hoodies', ' Tech Pants'], [' Swimming Shirt', ' Rash Guard', ' Dry'], [' Swimming Shirt', ' Rash Guard', ' Dry Fit V-Ni
ck'], [' Running Shoe', ' Swimming Shirt', ' Socks', ' Sweatshirts', ' Modern Pants', ' Soccer Shoe', ' Rash Guard', ' Hoodies', ' Tech Pants', ' Dry Fit V-Nick'], [' R
unning Shoe', ' Swimming Shirt', ' Socks', ' Sweatshirts', ' Modern Pants', ' Soccer Shoe', ' Rash Guard', ' Tech Pants', ' Dry Fit V-Nick', ' Hoodies'], [' Running Sho
e', ' Swimming Shirt', ' Rash Guard', ' Tech Pants', ' Hoodies', ' Dry Fit V-Nick'], [' Running Shoe', ' Swimming Shirt', ' Socks', ' Sweatshirts', ' Modern Pants', ' D
ry Fit V-Nick', ' Rash Guard', ' Tech Pants'], [' Swimming Shirt', ' Soccer Shoe', ' Hoodies', ' Dry Fit V-Nick', ' Tech Pants', ' Rash Guard'], [' Running Shoe', ' Soc
ks'], [' Socks', ' Sweatshirts', ' Modern Pants', ' Soccer Shoe', ' Hoodies', ' Rash Guard', ' Tech Pants', ' Dry Fit V-Nick'], [' Running Shoe', ' Swimming Shirt', ' R
ash Guard'], [' Running Shoe', ' Swimming Shirt', ' Socks', ' Sweatshirts', ' Modern Pants', ' Soccer Shoe', ' Hoodies', ' Tech Pants', ' Rash Guard', ' Dry Fit V-Nick'
]]

Enter Minimum Support:
```



```
Enter Minimum Support:20


Enter Minimum Confidence:50



 This is the result after first iteration:

 {' Dry Fit V-Nick': 9, ' Hoodies': 8, ' Modern Pants': 10, ' Rash Guard': 12, ' Running Shoe': 14, ' Soccer Shoe': 6, ' Socks': 13, ' Sweatshirts': 13, ' Swimming Shir
t': 11, ' Tech Pants': 9}


 This is the result after second iteration:

 {(' Dry Fit V-Nick', ' Running Shoe'): 4, (' Dry Fit V-Nick', ' Socks'): 6, (' Dry Fit V-Nick', ' Sweatshirts'): 7, (' Hoodies', ' Modern Pants'): 10, (' Hoodies', ' R
unning Shoe'): 12, (' Hoodies', ' Socks'): 14, (' Hoodies', ' Sweatshirts'): 15, (' Modern Pants', ' Running Shoe'): 19, (' Modern Pants', ' Socks'): 21, (' Modern Pant
s', ' Sweatshirts'): 22, (' Rash Guard', ' Running Shoe'): 25, (' Rash Guard', ' Socks'): 27, (' Rash Guard', ' Sweatshirts'): 28, (' Running Shoe', ' Socks'): 32, (' R
unning Shoe', ' Sweatshirts'): 33, (' Soccer Shoe', ' Socks'): 36, (' Soccer Shoe', ' Sweatshirts'): 37, (' Socks', ' Sweatshirts'): 40, (' Dry Fit V-Nick', ' Soccer Sh
oe'): 5, (' Hoodies', ' Soccer Shoe'): 13, (' Modern Pants', ' Soccer Shoe'): 20, (' Rash Guard', ' Soccer Shoe'): 26, (' Running Shoe', ' Soccer Shoe'): 31, (' Dry Fit
 V-Nick', ' Tech Pants'): 9, (' Hoodies', ' Rash Guard'): 11, (' Hoodies', ' Tech Pants'): 17, (' Modern Pants', ' Rash Guard'): 18, (' Modern Pants', ' Tech Pants'): 2
4, (' Rash Guard', ' Tech Pants'): 30, (' Running Shoe', ' Tech Pants'): 35, (' Soccer Shoe', ' Tech Pants'): 39, (' Socks', ' Tech Pants'): 42, (' Sweatshirts', ' Tech
 Pants'): 44, (' Swimming Shirt', ' Tech Pants'): 45, (' Dry Fit V-Nick', ' Swimming Shirt'): 8, (' Hoodies', ' Swimming Shirt'): 16, (' Modern Pants', ' Swimming Shirt
'): 23, (' Rash Guard', ' Swimming Shirt'): 29, (' Running Shoe', ' Swimming Shirt'): 34, (' Soccer Shoe', ' Swimming Shirt'): 38, (' Socks', ' Swimming Shirt'): 41, ('
 Sweatshirts', ' Swimming Shirt'): 43}
```

```
This is the result after third iteration:

 {(' Hoodies', ' Tech Pants', ' Modern Pants'): 5, (' Hoodies', ' Tech Pants', ' Running Shoe'): 7, (' Hoodies', ' Swimming Shirt', ' Socks'): 9, (' Hoodies', ' Swimmi
g Shirt', ' Sweatshirts'): 10, (' Hoodies', ' Swimming Shirt', ' Modern Pants'): 12, (' Hoodies', ' Swimming Shirt', ' Running Shoe'): 14, (' Hoodies', ' Socks', ' Swe
tshirts'): 16, (' Hoodies', ' Socks', ' Modern Pants'): 18, (' Hoodies', ' Socks', ' Running Shoe'): 20, (' Hoodies', ' Sweatshirts', ' Modern Pants'): 23, (' Hoodies'
 ' Sweatshirts', ' Running Shoe'): 25, (' Hoodies', ' Dry Fit V-Nick', ' Modern Pants'): 27, (' Hoodies', ' Dry Fit V-Nick', ' Running Shoe'): 29, (' Hoodies', ' Moder
 Pants', ' Running Shoe'): 32, (' Hoodies', ' Soccer Shoe', ' Running Shoe'): 34, (' Tech Pants', ' Swimming Shirt', ' Socks'): 37, (' Tech Pants', ' Swimming Shirt',
 Sweatshirts'): 38, (' Tech Pants', ' Swimming Shirt', ' Modern Pants'): 40, (' Tech Pants', ' Swimming Shirt', ' Running Shoe'): 42, (' Tech Pants', ' Socks', ' Sweat
hirts'): 44, (' Tech Pants', ' Socks', ' Modern Pants'): 46, (' Tech Pants', ' Socks', ' Running Shoe'): 48, (' Tech Pants', ' Sweatshirts', ' Modern Pants'): 51, (' T
ch Pants', ' Sweatshirts', ' Running Shoe'): 53, (' Tech Pants', ' Dry Fit V-Nick', ' Modern Pants'): 55, (' Tech Pants', ' Dry Fit V-Nick', ' Running Shoe'): 57, (' T
ch Pants', ' Modern Pants', ' Running Shoe'): 60, (' Tech Pants', ' Soccer Shoe', ' Running Shoe'): 62, (' Swimming Shirt', ' Socks', ' Sweatshirts'): 65, (' Swimming
hirt', ' Socks', ' Modern Pants'): 67, (' Swimming Shirt', ' Socks', ' Running Shoe'): 69, (' Swimming Shirt', ' Sweatshirts', ' Modern Pants'): 72, (' Swimming Shirt'
 ' Sweatshirts', ' Running Shoe'): 74, (' Swimming Shirt', ' Dry Fit V-Nick', ' Modern Pants'): 76, (' Swimming Shirt', ' Dry Fit V-Nick', ' Running Shoe'): 78, (' Swi
ming Shirt', ' Modern Pants', ' Running Shoe'): 81, (' Swimming Shirt', ' Soccer Shoe', ' Running Shoe'): 83, (' Socks', ' Sweatshirts', ' Modern Pants'): 87, (' Socks
, ' Sweatshirts', ' Running Shoe'): 89, (' Socks', ' Dry Fit V-Nick', ' Modern Pants'): 91, (' Socks', ' Dry Fit V-Nick', ' Running Shoe'): 93, (' Socks', ' Modern Pan
s', ' Running Shoe'): 96, (' Socks', ' Soccer Shoe', ' Running Shoe'): 98, (' Sweatshirts', ' Dry Fit V-Nick', ' Modern Pants'): 101, (' Sweatshirts', ' Dry Fit V-Nick
, ' Running Shoe'): 103, (' Sweatshirts', ' Modern Pants', ' Running Shoe'): 106, (' Sweatshirts', ' Soccer Shoe', ' Running Shoe'): 108, (' Dry Fit V-Nick', ' Modern
ants', ' Running Shoe'): 112, (' Dry Fit V-Nick', ' Soccer Shoe', ' Running Shoe'): 114, (' Modern Pants', ' Soccer Shoe', ' Running Shoe'): 117, (' Hoodies', ' Tech P
nts', ' Soccer Shoe'): 6, (' Hoodies', ' Swimming Shirt', ' Soccer Shoe'): 13, (' Hoodies', ' Socks', ' Soccer Shoe'): 19, (' Hoodies', ' Sweatshirts', ' Soccer Shoe')
 24, (' Hoodies', ' Dry Fit V-Nick', ' Soccer Shoe'): 28, (' Hoodies', ' Modern Pants', ' Soccer Shoe'): 31, (' Tech Pants', ' Swimming Shirt', ' Soccer Shoe'): 41, ('
Tech Pants', ' Socks', ' Soccer Shoe'): 47, (' Tech Pants', ' Sweatshirts', ' Soccer Shoe'): 52, (' Tech Pants', ' Dry Fit V-Nick', ' Soccer Shoe'): 56, (' Tech Pants'
 ' Modern Pants', ' Soccer Shoe'): 59, (' Swimming Shirt', ' Socks', ' Soccer Shoe'): 68, (' Swimming Shirt', ' Sweatshirts', ' Soccer Shoe'): 73, (' Swimming Shirt',
 Dry Fit V-Nick', ' Soccer Shoe'): 77, (' Swimming Shirt', ' Modern Pants', ' Soccer Shoe'): 80, (' Socks', ' Sweatshirts', ' Soccer Shoe'): 88, (' Socks', ' Dry Fit V
Nick', ' Soccer Shoe'): 92, (' Socks', ' Modern Pants', ' Soccer Shoe'): 95, (' Sweatshirts', ' Dry Fit V-Nick', ' Soccer Shoe'): 102, (' Sweatshirts', ' Modern Pants'
 ' Soccer Shoe'): 105, (' Dry Fit V-Nick', ' Modern Pants', ' Soccer Shoe'): 111, (' Hoodies', ' Tech Pants', ' Rash Guard'): 8, (' Hoodies', ' Swimming Shirt', ' Rash
Guard'): 15, (' Hoodies', ' Socks', ' Rash Guard'): 21, (' Hoodies', ' Sweatshirts', ' Rash Guard'): 26, (' Hoodies', ' Dry Fit V-Nick', ' Rash Guard'): 30, (' Hoodies
, ' Modern Pants', ' Rash Guard'): 33, (' Hoodies', ' Soccer Shoe', ' Rash Guard'): 35, (' Hoodies', ' Running Shoe', ' Rash Guard'): 36, (' Tech Pants', ' Swimming Sh
rt', ' Rash Guard'): 43, (' Tech Pants', ' Socks', ' Rash Guard'): 49, (' Tech Pants', ' Sweatshirts', ' Rash Guard'): 54, (' Tech Pants', ' Dry Fit V-Nick', ' Rash Gu
rd'): 58, (' Tech Pants', ' Modern Pants', ' Rash Guard'): 61, (' Tech Pants', ' Soccer Shoe', ' Rash Guard'): 63, (' Tech Pants', ' Running Shoe', ' Rash Guard'): 64,
(' Swimming Shirt', ' Socks', ' Rash Guard'): 70, (' Swimming Shirt', ' Sweatshirts', ' Rash Guard'): 75, (' Swimming Shirt', ' Dry Fit V-Nick', ' Rash Guard'): 79, ('
Swimming Shirt', ' Modern Pants', ' Rash Guard'): 82, (' Swimming Shirt', ' Soccer Shoe', ' Rash Guard'): 84, (' Swimming Shirt', ' Running Shoe', ' Rash Guard'): 85,
' Socks', ' Sweatshirts', ' Rash Guard'): 90, (' Socks', ' Dry Fit V-Nick', ' Rash Guard'): 94, (' Socks', ' Modern Pants', ' Rash Guard'): 97, (' Socks', ' Soccer Sho
', ' Rash Guard'): 99, (' Socks', ' Running Shoe', ' Rash Guard'): 100, (' Sweatshirts', ' Dry Fit V-Nick', ' Rash Guard'): 104, (' Sweatshirts', ' Modern Pants', ' Ra
h Guard'): 107, (' Sweatshirts', ' Soccer Shoe', ' Rash Guard'): 109, (' Sweatshirts', ' Running Shoe', ' Rash Guard'): 110, (' Dry Fit V-Nick', ' Modern Pants', ' Ras
 Guard'): 113, (' Dry Fit V-Nick', ' Soccer Shoe', ' Rash Guard'): 115, (' Dry Fit V-Nick', ' Running Shoe', ' Rash Guard'): 116, (' Modern Pants', ' Soccer Shoe', ' R
sh Guard'): 118, (' Modern Pants', ' Running Shoe', ' Rash Guard'): 119, (' Soccer Shoe', ' Running Shoe', ' Rash Guard'): 120, (' Hoodies', ' Tech Pants', ' Dry Fit V
Nick'): 4, (' Hoodies', ' Swimming Shirt', ' Dry Fit V-Nick'): 11, (' Hoodies', ' Socks', ' Dry Fit V-Nick'): 17, (' Hoodies', ' Sweatshirts', ' Dry Fit V-Nick'): 22,
' Tech Pants', ' Swimming Shirt', ' Dry Fit V-Nick'): 39, (' Tech Pants', ' Socks', ' Dry Fit V-Nick'): 45, (' Tech Pants', ' Sweatshirts', ' Dry Fit V-Nick'): 50, ('
wimming Shirt', ' Socks', ' Dry Fit V-Nick'): 66, (' Swimming Shirt', ' Sweatshirts', ' Dry Fit V-Nick'): 71, (' Socks', ' Sweatshirts', ' Dry Fit V-Nick'): 86}
```

Association Rules for Nike:

```
Association Rule for Pair -  (' Rash Guard', ' Running Shoe')
__Rule__ __Confidence__
(' Rash Guard', ' Running Shoe') => (' Soccer Shoe',) 400 This Rule is Acceptable
Association Rule for Pair -  (' Rash Guard', ' Soccer Shoe')
__Rule__ __Confidence__
(' Rash Guard', ' Soccer Shoe') => (' Running Shoe',) 400 This Rule is Acceptable
Association Rule for Pair -  (' Running Shoe', ' Soccer Shoe')
__Rule__ __Confidence__
(' Running Shoe', ' Soccer Shoe') => (' Rash Guard',) 300 This Rule is Acceptable
Association Rule for Pair -   Rash Guard
__Rule__ __Confidence__
 Rash Guard => (' Rash Guard', ' Running Shoe', ' Soccer Shoe') 1000 This Rule is Acceptable
Association Rule for Pair -   Running Shoe
__Rule__ __Confidence__
 Running Shoe => (' Rash Guard', ' Running Shoe', ' Soccer Shoe') 857 This Rule is Acceptable
Association Rule for Pair -   Soccer Shoe
__Rule__ __Confidence__
 Soccer Shoe => (' Rash Guard', ' Running Shoe', ' Soccer Shoe') 2000 This Rule is Acceptable
```

K Mart:

Support:

```
C:\Users\Administrator\AppData\Local\Programs\Python\Python39\Scripts\Project>Python my_midtermproj.py

 Welcome to Anusha's Apriori Algorithm Implementation.
 This program will take input data of transactions from you and
 give you the most frequent association rules with your product lines in return.
 This can give you an idea about what products sell in a combination following them can maximize your profit.

Enter File name with '.txt' extension:KMart.txt

 This is the dataset you entered:

 [[' Decorative Pillows', ' Quilts', ' Embroidered Bedspread'], [' Embroidered Bedspread', ' Shams', ' Kids Bedding', ' Bedding Collections', ' Bed Skirts', ' Bedspread
s', ' Sheets'], [' Decorative Pillows', ' Quilts', ' Embroidered Bedspread', ' Shams', ' Kids Bedding', ' Bedding Collections'], [' Kids Bedding', ' Bedding Collections
', ' Sheets', ' Bedspreads', ' Bed Skirts'], [' Decorative Pillows', ' Kids Bedding', ' Bedding Collections', ' Sheets', ' Bed Skirts', ' Bedspreads'], [' Bedding Colle
ctions', ' Bedspreads', ' Bed Skirts', ' Sheets', ' Shams', ' Kids Bedding'], [' Decorative Pillows', ' Quilts'], [' Decorative Pillows', ' Quilts', ' Embroidered Bedsp
read'], [' Bedspreads', ' Bed Skirts', ' Shams', ' Kids Bedding', ' Sheets'], [' Quilts', ' Embroidered Bedspread', ' Bedding Collections'], [' Bedding Collections', '
Bedspreads', ' Bed Skirts', ' Kids Bedding', ' Shams', ' Sheets'], [' Decorative Pillows', ' Quilts'], [' Embroidered Bedspread', ' Shams'], [' Sheets', ' Shams', ' Bed
 Skirts', ' Kids Bedding'], [' Decorative Pillows', ' Quilts'], [' Decorative Pillows', ' Kids Bedding', ' Bed Skirts', ' Shams'], [' Decorative Pillows', ' Shams', ' B
ed Skirts'], [' Quilts', ' Sheets', ' Kids Bedding'], [' Shams', ' Bed Skirts', ' Kids Bedding', ' Sheets'], [' Decorative Pillows', ' Bedspreads', ' Shams', ' Sheets',
 ' Bed Skirts', ' Kids Bedding']]
```

```
Enter Minimum Support:30


Enter Minimum Confidence:40



 This is the result after first iteration:

 {' Bed Skirts': 11, ' Bedding Collections': 7, ' Bedspreads': 7, ' Decorative Pillows': 10, ' Embroidered Bedspread': 6, ' Kids Bedding': 12, ' Quilts': 8, ' Shams':
1, ' Sheets': 10}


 This is the result after second iteration:

 {(' Bedding Collections', ' Decorative Pillows'): 7, (' Bedspreads', ' Decorative Pillows'): 11, (' Bedspreads', ' Quilts'): 8, (' Decorative Pillows', ' Embroidered
edspread'): 7, (' Decorative Pillows', ' Quilts'): 11, (' Embroidered Bedspread', ' Quilts'): 14, (' Kids Bedding', ' Quilts'): 16, (' Bedding Collections', ' Bedsprea
s'): 6, (' Bedding Collections', ' Kids Bedding'): 8, (' Bedding Collections', ' Shams'): 9, (' Bedding Collections', ' Sheets'): 10, (' Bedspreads', ' Kids Bedding'):
12, (' Bedspreads', ' Shams'): 13, (' Bedspreads', ' Sheets'): 14, (' Decorative Pillows', ' Kids Bedding'): 15, (' Decorative Pillows', ' Shams'): 16, (' Decorative P
llows', ' Sheets'): 17, (' Embroidered Bedspread', ' Kids Bedding'): 18, (' Embroidered Bedspread', ' Shams'): 19, (' Embroidered Bedspread', ' Sheets'): 20, (' Kids B
dding', ' Shams'): 21, (' Kids Bedding', ' Sheets'): 22, (' Quilts', ' Shams'): 23, (' Quilts', ' Sheets'): 24, (' Shams', ' Sheets'): 25}
```

```
 This is the result after third iteration:

 {(' Quilts', ' Sheets', ' Decorative Pillows'): 6, (' Quilts', ' Decorative Pillows', ' Embroidered Bedspread'): 6, (' Quilts', ' Bedspreads', ' Embroidered Bedspread'
): 8, (' Quilts', ' Shams', ' Embroidered Bedspread'): 9, (' Bedding Collections', ' Sheets', ' Decorative Pillows'): 17, (' Bedding Collections', ' Sheets', ' Embroide
red Bedspread'): 11, (' Bedding Collections', ' Decorative Pillows', ' Embroidered Bedspread'): 13, (' Bedding Collections', ' Bedspreads', ' Embroidered Bedspread'): 1
5, (' Bedding Collections', ' Shams', ' Embroidered Bedspread'): 16, (' Sheets', ' Decorative Pillows', ' Embroidered Bedspread'): 18, (' Sheets', ' Bedspreads', ' Embr
oidered Bedspread'): 20, (' Sheets', ' Shams', ' Embroidered Bedspread'): 21, (' Decorative Pillows', ' Bedspreads', ' Embroidered Bedspread'): 23, (' Decorative Pillow
s', ' Shams', ' Embroidered Bedspread'): 24, (' Bedspreads', ' Shams', ' Embroidered Bedspread'): 25, (' Quilts', ' Sheets', ' Bedspreads'): 7, (' Quilts', ' Sheets', '
 Shams'): 8, (' Quilts', ' Sheets', ' Kids Bedding'): 9, (' Quilts', ' Decorative Pillows', ' Bedspreads'): 10, (' Quilts', ' Decorative Pillows', ' Shams'): 11, (' Qui
lts', ' Decorative Pillows', ' Kids Bedding'): 12, (' Quilts', ' Bedspreads', ' Shams'): 13, (' Quilts', ' Bedspreads', ' Kids Bedding'): 14, (' Quilts', ' Shams', ' Ki
ds Bedding'): 15, (' Quilts', ' Embroidered Bedspread', ' Kids Bedding'): 16, (' Bedding Collections', ' Sheets', ' Bedspreads'): 18, (' Bedding Collections', ' Sheets'
, ' Shams'): 19, (' Bedding Collections', ' Sheets', ' Kids Bedding'): 20, (' Bedding Collections', ' Decorative Pillows', ' Bedspreads'): 21, (' Bedding Collections',
' Decorative Pillows', ' Shams'): 22, (' Bedding Collections', ' Decorative Pillows', ' Kids Bedding'): 23, (' Bedding Collections', ' Bedspreads', ' Shams'): 24, (' Be
dding Collections', ' Bedspreads', ' Kids Bedding'): 25, (' Bedding Collections', ' Shams', ' Kids Bedding'): 26, (' Bedding Collections', ' Embroidered Bedspread', ' K
ids Bedding'): 27, (' Sheets', ' Decorative Pillows', ' Bedspreads'): 28, (' Sheets', ' Decorative Pillows', ' Shams'): 29, (' Sheets', ' Decorative Pillows', ' Kids Be
dding'): 30, (' Sheets', ' Bedspreads', ' Shams'): 31, (' Sheets', ' Bedspreads', ' Kids Bedding'): 32, (' Sheets', ' Shams', ' Kids Bedding'): 33, (' Sheets', ' Embroi
dered Bedspread', ' Kids Bedding'): 34, (' Decorative Pillows', ' Bedspreads', ' Shams'): 35, (' Decorative Pillows', ' Bedspreads', ' Kids Bedding'): 36, (' Decorative
 Pillows', ' Shams', ' Kids Bedding'): 37, (' Decorative Pillows', ' Embroidered Bedspread', ' Kids Bedding'): 38, (' Bedspreads', ' Shams', ' Kids Bedding'): 39, (' Be
dspreads', ' Embroidered Bedspread', ' Kids Bedding'): 40, (' Shams', ' Embroidered Bedspread', ' Kids Bedding'): 41}
```

Association rules for K Mart:

```
Association Rule for Pair -  (' Embroidered Bedspread', ' Kids Bedding')
__Rule__ __Confidence__
(' Embroidered Bedspread', ' Kids Bedding') => (' Shams',) 200 This Rule is Acceptable
Association Rule for Pair -  (' Embroidered Bedspread', ' Shams')
__Rule__ __Confidence__
(' Embroidered Bedspread', ' Shams') => (' Kids Bedding',) 200 This Rule is Acceptable
Association Rule for Pair -  (' Kids Bedding', ' Shams')
__Rule__ __Confidence__
(' Kids Bedding', ' Shams') => (' Embroidered Bedspread',) 100 This Rule is Acceptable
Association Rule for Pair -   Embroidered Bedspread
__Rule__ __Confidence__
 Embroidered Bedspread => (' Shams', ' Embroidered Bedspread', ' Kids Bedding') 683 This Rule is Acceptable
Association Rule for Pair -   Kids Bedding
__Rule__ __Confidence__
 Kids Bedding => (' Shams', ' Embroidered Bedspread', ' Kids Bedding') 341 This Rule is Acceptable
Association Rule for Pair -   Shams
__Rule__ __Confidence__
 Shams => (' Shams', ' Embroidered Bedspread', ' Kids Bedding') 372 This Rule is Acceptable
```

Generic Dataset:

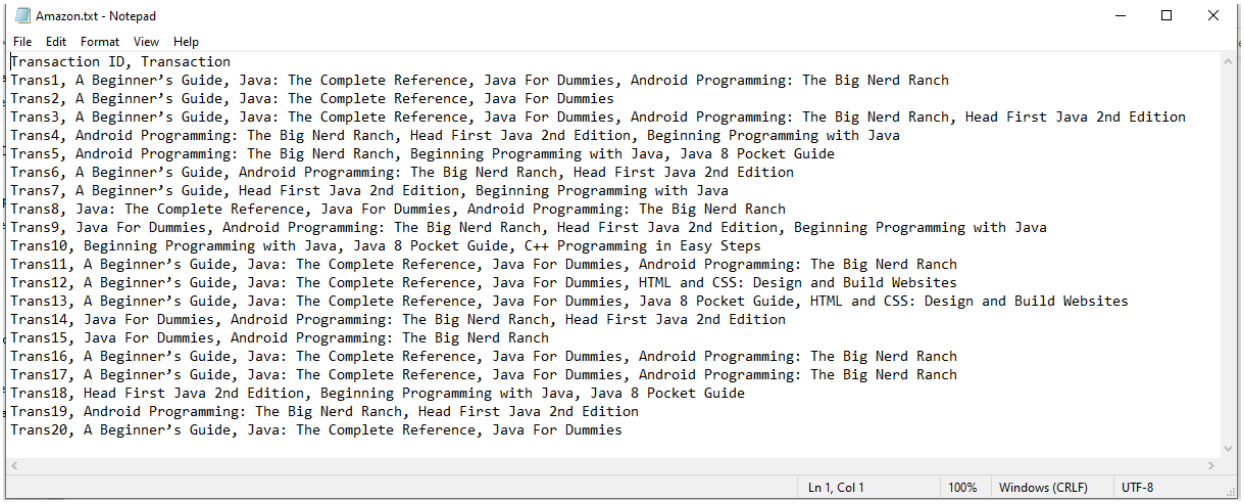
testing purpose

Support and Association rules:

```
Enter File name with '.txt' extension:Generic.txt

 This is the dataset you entered:

[[' A', ' B', ' C'], [' A', ' B', ' C'], [' A', ' B', ' C', ' D'], [' A', ' B', ' C', ' D', ' E'], [' A', ' B', ' D', ' E'], [' A', ' D', ' E'], [' A', ' E'], [' A',
E'], [' A', ' C', ' E'], [' A', ' C', ' E'], [' A', ' C', ' E']]
```

```
 This is the result after first iteration:

 {' A': 11, ' B': 5, ' C': 7, ' D': 4, ' E': 8}


 This is the result after second iteration:

 {(' B', ' C'): 3, (' B', ' D'): 3, (' C', ' D'): 5, (' B', ' E'): 4, (' C', ' E'): 5, (' D', ' E'): 6}


 This is the result after third iteration:

 {(' B', ' C', ' E'): 3, (' D', ' C', ' E'): 4}

Association Rule for Pair -  (' C', ' D')
__Rule__ __Confidence__
(' C', ' D') => (' E',) 0 This Rule is Rejected
Association Rule for Pair -  (' C', ' E')
__Rule__ __Confidence__
(' C', ' E') => (' D',) 0 This Rule is Rejected
Association Rule for Pair -  (' D', ' E')
__Rule__ __Confidence__
(' D', ' E') => (' C',) 0 This Rule is Rejected
Association Rule for Pair -   C
__Rule__ __Confidence__
 C => (' D', ' C', ' E') 57 This Rule is Acceptable
Association Rule for Pair -   D
__Rule__ __Confidence__
 D => (' D', ' C', ' E') 100 This Rule is Acceptable
Association Rule for Pair -   E
__Rule__ __Confidence__
 E => (' D', ' C', ' E') 50 This Rule is Acceptable
```

Dataset Amazon:



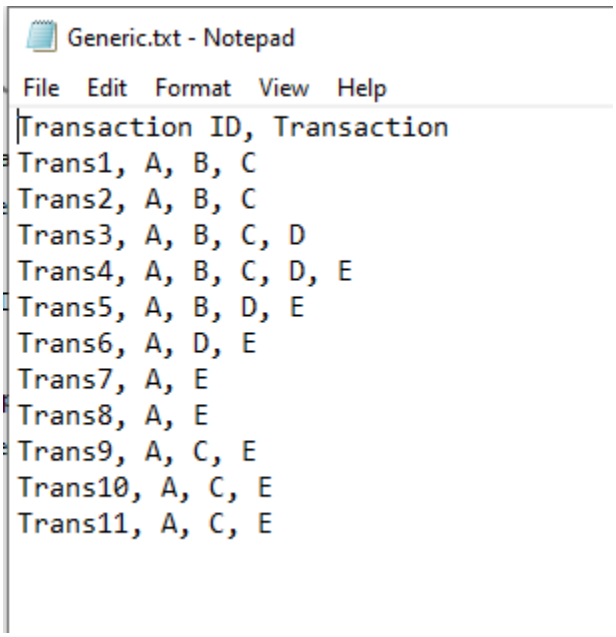
Dataset Nike:

Dataset BestBuy:

Dataset K Mart:

Dataset Generic:

I kept this to verify my work



Final compiled code (py files):

```
"""

Author: Neha Singh
Project Introduction:

## Set the directory as under before running the code
## Keep the program file and the dataset in the same directory
"""

from itertools import permutations
from itertools import combinations

## Defining all the Functions

## Find the count of unique items and make a dictionary
def uniquelist(final_data):
    mylist = []
    for i in final_data:
        for j in range(len(i)):
            mylist.append(i[j])
```

```
    uniquelist = set(mylist)
    uniquelist = sorted(uniquelist)
    return uniquelist, mylist


## Performing Iteration 1
def iteration1(mylist, uniquelist):
    firstdict = {}
    for i in uniquelist:
        firstdict[i] = mylist.count(i)
    return firstdict


## Eliminate those items not satisfying minimum support
def support_eliminate(firstdict):
    keylist = list(firstdict.keys())
    valuelist = list(firstdict.values())
    mydict2 = {}
    # support_dict = {}
    count = 0

    for i in valuelist:
        support = round(i * 100 / total_trxns, 2)
        count += 1

        # print(support, min_sup)
        if support >= min_sup:
            # firstdict.pop(keylist[count])
            mydict2[keylist[count - 1]] = i
            # support_dict[keylist[count-1]] = support
    return mydict2

## Creating combinations of items in Iteration 1
def create_combinations1(mydict2, n):
    comb = combinations(list(mydict2.keys()), n)
    up_comb = []
    for i in comb:
        up_comb.append(i)

    return up_comb

## Performing Iteration 2
def iter2(final_data, up_comb):
    updated_dict = {}
    for i in range(len(final_data)):
```

```
        c2=0
        for j in range(len(up_comb)):
            if (up_comb[j][0] and up_comb[j][1]) in final_data[i]:
                c2+=1
                updated_dict[up_comb[j]] = c2
    return updated_dict

## Creating combinations of items in Iteration 2
def create_combinations2(targlist, n):
    comb = combinations(targlist, n)
    up_comb = []
    for i in comb:
        up_comb.append(i)

    return up_comb


## Performing Iteration 3
def iter3(final_data, combs_3):
    updated_dict3 = {}
    for i in range(len(final_data)):
        c2=0
        for j in range(len(combs_3)):
            if (combs_3[j][0] and combs_3[j][1] and combs_3[j][2]) in fina
l_data[i]:

              c2= c2+1
              updated_dict3[combs_3[j]] = c2

    return updated_dict3


## List of Frequent itemsets and Association rules
def assoc_rules(highkey, highval):
    support_dict = {}
    count = 0
    item_set = set(highkey)

    comb2s = combinations(highkey, 2)
    for i in comb2s:
        print("Association Rule for Pair - ", i)
        print("__Rule__", "__Confidence__")
        i = tuple(sorted(i))
        left_item = i
        right_item = tuple(item_set - set(i))
```

```
        # print(left_item, "=>" ,right_item)

        denom2 = sup_elim2[i]
        conf = (highval // denom2) * 100

        if conf >= min_conf:
            print(left_item, "=>", right_item, conf, "This Rule is Accepta
ble")
        else:
            print(left_item, "=>", right_item, conf, "This Rule is Rejecte
d")


    comb1s = highkey
    for i in comb1s:
        print("Association Rule for Pair - ", i)
        print("__Rule__", "__Confidence__")

        left_item = i
        right_item = tuple(item_set - set(i))
        denom1 = sup_elim1[i]
        conf1 = highval * 100 // denom1

        # print(left_item, right_item)
        if conf1 >= min_conf:
            print(left_item, "=>", right_item, conf1, "This Rule is Accept
able")
        else:
            print(left_item, "=>", right_item, conf1, "This Rule is Reject
ed")

    print()



## Main program to run Algorithm using the functions defined

print("\n","Welcome to Anusha's Apriori Algorithm Implementation. ", '\n',
      "This program will take input data of transactions from you and ", '
\n',
      "give you the most frequent association rules with your product line
s in return. ", '\n',
      "This can give you an idea about what products sell in a combination
 following them can maximize your profit. ", "\n")


filename = input("Enter File name with '.txt' extension:")
```

```
fileobject = open(filename, "r", encoding="UTF-
8",errors="surrogateescape")
input_data = fileobject.readlines()

## Transforming the dataset
final_data = []
for lines in range(1,len(input_data)):
    input_data[lines] = input_data[lines].replace("\n","")
    final_data.append(input_data[lines].split(','))

## Transforming final_data
for i in range(len(final_data)):
  final_data[i] = final_data[i][1:]

print("\n","This is the dataset you entered:", "\n", "\n", final_data, '\n
')

## Declaring all the required variables
min_sup = int(input("Enter Minimum Support:"))
print("\n")
min_conf = int(input("Enter Minimum Confidence:"))
print("\n")

total_trxns = len(final_data)

uniques, full_list = uniquelist(final_data)

iteration1 = iteration1(full_list, uniques)

sup_elim1 = support_eliminate(iteration1)

print("\n","This is the result after first iteration:", "\n", "\n", sup_el
im1, '\n')

new_combs = create_combinations1(sup_elim1, 2)

iteration2 = iter2(final_data, new_combs)

sup_elim2 = support_eliminate(iteration2)

print("\n","This is the result after second iteration:", "\n", "\n", sup_e
lim2, '\n')

mylist2iter = list(sup_elim2.keys())
```

```
##used this to put unique pairs in tuple to make it in a
single listlistafter2 = []
for i in range(len(mylist2iter)):
    listafter2.append(mylist2iter[i][0])
    listafter2.append(mylist2iter[i][1])

listafter2 = set(listafter2)

combs_3 =

create_combinations2(listafter2, 3)

iteration3 = iter3(final_data,

combs_3) sup_elim3 =

support_eliminate(iteration3)

print("\n","This is the result after third iteration:", "\n",
"\n", sup_elim3, '\n')


highkey =
tuple(sorted(list(iteration3.keys())[list(iteration3.values()).i
ndex(max(iteration3.values()))]))
highval =

max(iteration3.values())

assoc_rules(highkey,

highval)
```