CS634-104 Data Mining MidTerm Project

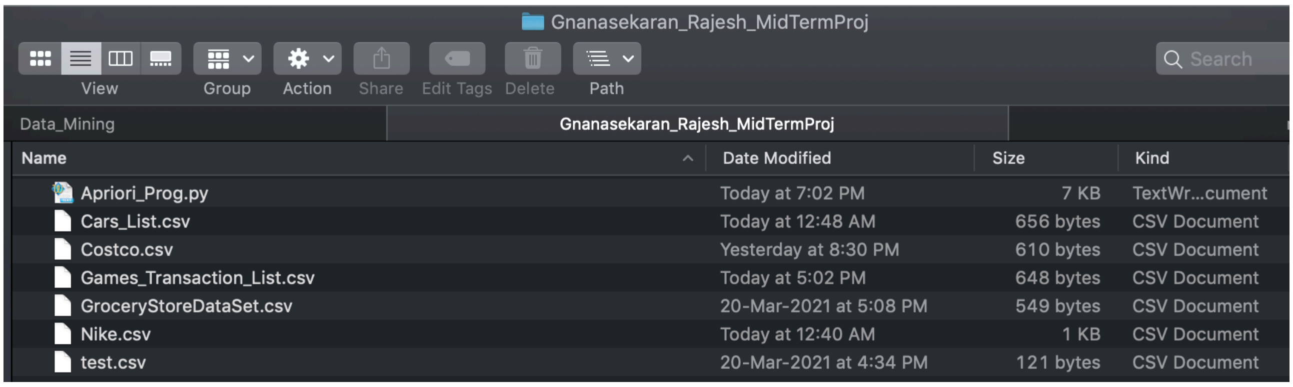
Name :- Rajesh Gnanasekaran (rg48@njit.edu)
Date :- 03/28/2021

Apriori Algorithm :

Using Apriori algorithm to generate all the association rules and by taking the input transaction for each of the 5 transactional data set (GroceryStoreDataSet.csv , Cars_List.csv , Games_Transaction_List.csv , Costco.csv , test.csv).

Configuration :-

- First Download the .zip file and extract all the files including .py file and the datasets which are included in it.
- The Extracted file must consist the following data.

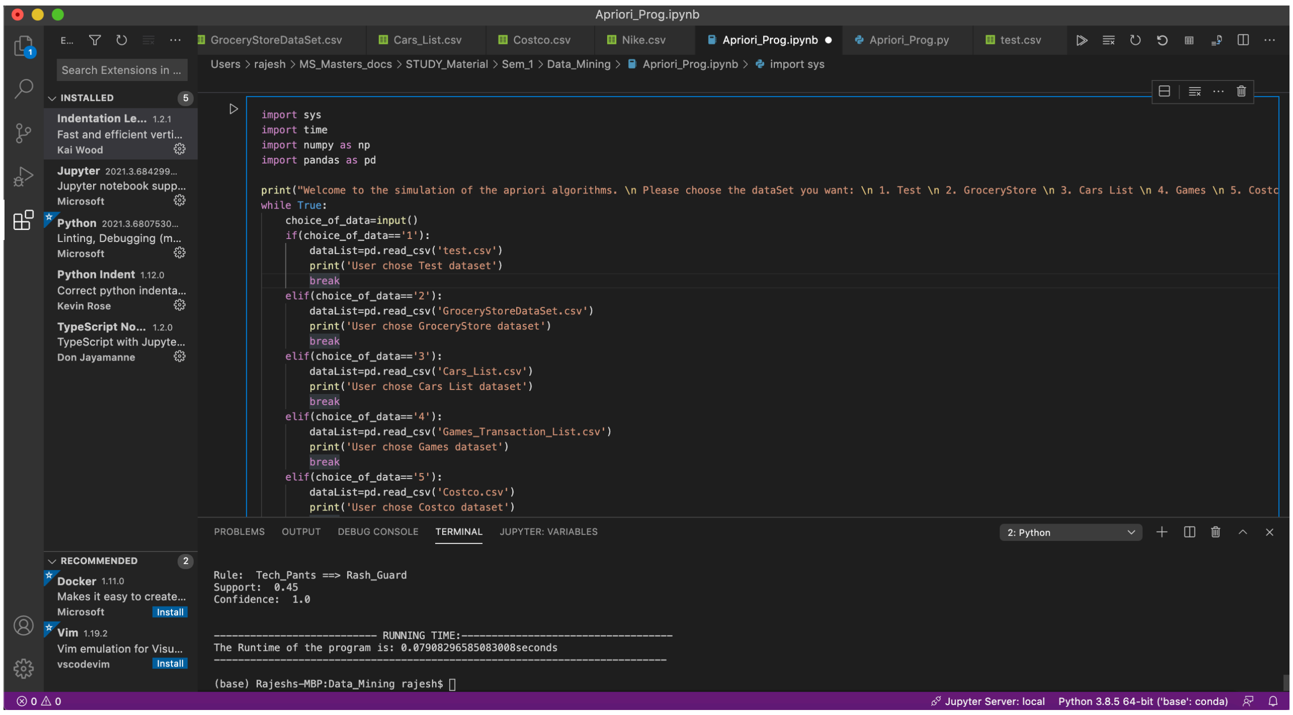

Gnanasekaran_Rajesh_MidTermProj
View
Group
Action
Share
Edit Tags
Delete
Path
Search
Data_Mining
Gnanasekaran_Rajesh_MidTermProj

Name	Date Modified	Size	Kind
Apriori_Prog.py	Today at 7:02 PM	7 KB	TextWr...cument
Cars_List.csv	Today at 12:48 AM	656 bytes	CSV Document
Costco.csv	Yesterday at 8:30 PM	610 bytes	CSV Document
Games_Transaction_List.csv	Today at 5:02 PM	648 bytes	CSV Document
GroceryStoreDataSet.csv	20-Mar-2021 at 5:08 PM	549 bytes	CSV Document
Nike.csv	Today at 12:40 AM	1 KB	CSV Document
test.csv	20-Mar-2021 at 4:34 PM	121 bytes	CSV Document

- From the above screenshot , we can see that I have made 6 different datasets each of different type of transactions. Also consist of one main Apriori_Prog.py file which will execute and read any one of the datasets and will show the working result of the file.

Most of the coding is done in VS (Visual Studio) and integrated it with python x Jupyter , so that its easy to read and display efficiently.

- In the below snapshot is an example of how the VS code is done and executed with all the datasets present accordingly.

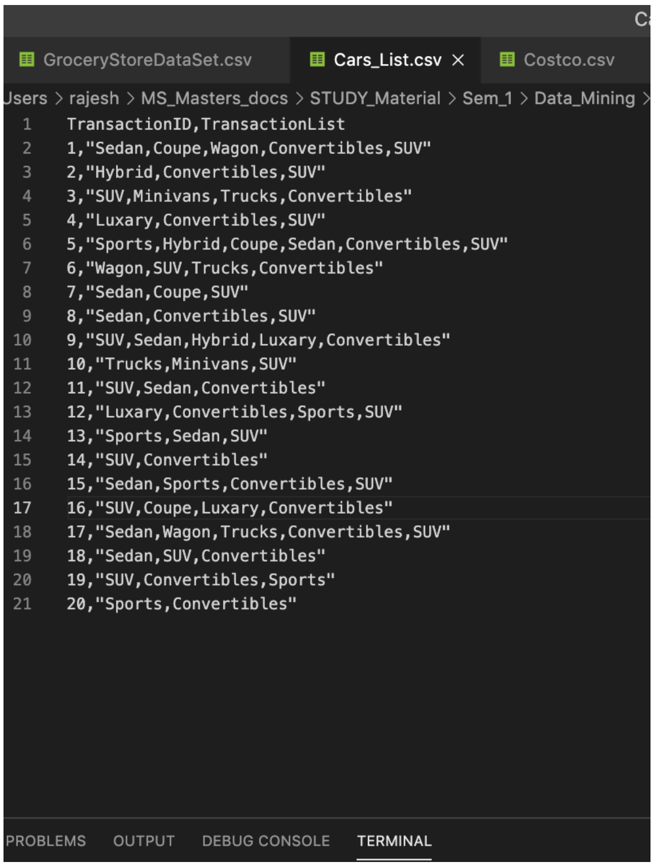


```python
import sys
import time
import numpy as np
import pandas as pd

print("Welcome to the simulation of the apriori algorithms. \n Please choose the dataSet you want: \n 1. Test \n 2. GroceryStore \n 3. Cars List \n 4. Games \n 5. Costc
while True:
    choice_of_data=input()
    if(choice_of_data=='1'):
        dataList=pd.read_csv('test.csv')
        print('User chose Test dataset')
        break
    elif(choice_of_data=='2'):
        dataList=pd.read_csv('GroceryStoreDataSet.csv')
        print('User chose GroceryStore dataset')
        break
    elif(choice_of_data=='3'):
        dataList=pd.read_csv('Cars_List.csv')
        print('User chose Cars List dataset')
        break
    elif(choice_of_data=='4'):
        dataList=pd.read_csv('Games_Transaction_List.csv')
        print('User chose Games dataset')
        break
    elif(choice_of_data=='5'):
        dataList=pd.read_csv('Costco.csv')
        print('User chose Costco dataset')
```

```
Rule:  Tech_Pants ==> Rash_Guard
Support:  0.45
Confidence:  1.0


--------------------------- RUNNING TIME:------------------------------------
The Runtime of the program is: 0.07908296585083008seconds
---------------------------------------------------------------------------

(base) Rajeshs-MBP:Data_Mining rajesh$ 
```

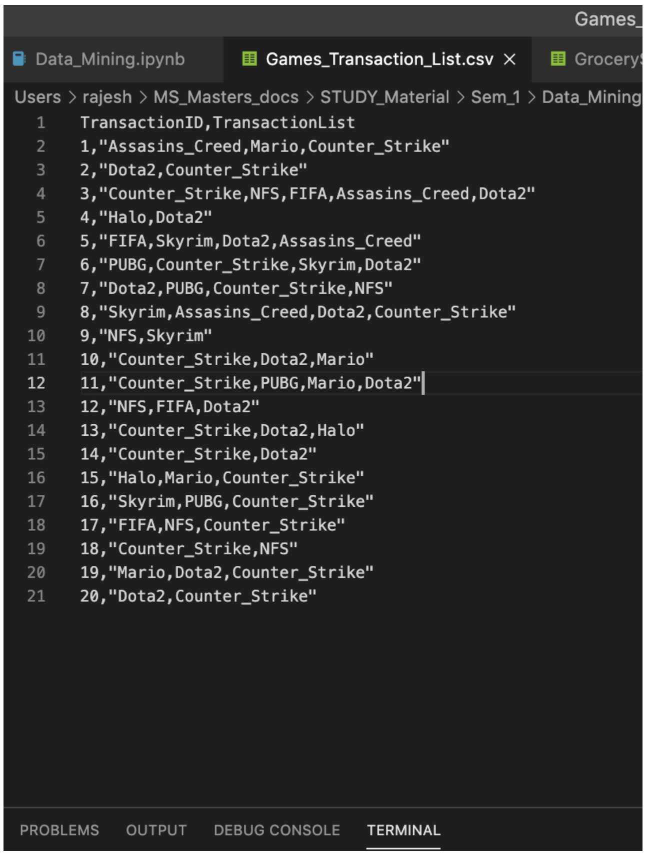
- I have 6 datasets included in which had made one dataset for testing purpose , below are few examples of a dataset which I created as a .csv file.

Cars_List.csv ->



```
TransactionID,TransactionList
1,"Sedan,Coupe,Wagon,Convertibles,SUV"
2,"Hybrid,Convertibles,SUV"
3,"SUV,Minivans,Trucks,Convertibles"
4,"Luxary,Convertibles,SUV"
5,"Sports,Hybrid,Coupe,Sedan,Convertibles,SUV"
6,"Wagon,SUV,Trucks,Convertibles"
7,"Sedan,Coupe,SUV"
8,"Sedan,Convertibles,SUV"
9,"SUV,Sedan,Hybrid,Luxary,Convertibles"
10,"Trucks,Minivans,SUV"
11,"SUV,Sedan,Convertibles"
12,"Luxary,Convertibles,Sports,SUV"
13,"Sports,Sedan,SUV"
14,"SUV,Convertibles"
15,"Sedan,Sports,Convertibles,SUV"
16,"SUV,Coupe,Luxary,Convertibles"
17,"Sedan,Wagon,Trucks,Convertibles,SUV"
18,"Sedan,SUV,Convertibles"
19,"SUV,Convertibles,Sports"
20,"Sports,Convertibles"
```

Games_Transaction_List.csv ->

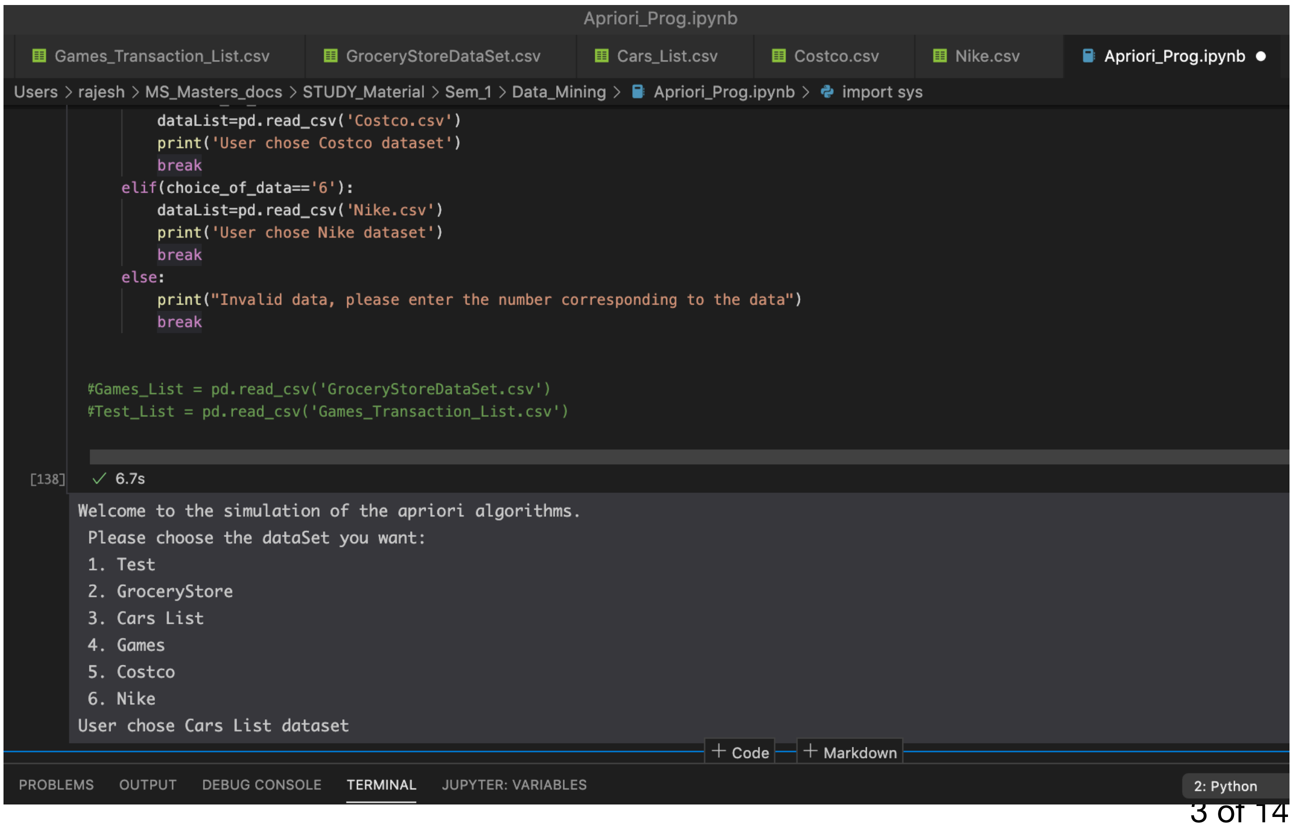

Games_
Data_Mining.ipynb
Games_Transaction_List.csv
Grocery
Users > rajesh > MS_Masters_docs > STUDY_Material > Sem_1 > Data_Mining
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL

```
TransactionID,TransactionList
1,"Assasins_Creed,Mario,Counter_Strike"
2,"Dota2,Counter_Strike"
3,"Counter_Strike,NFS,FIFA,Assasins_Creed,Dota2"
4,"Halo,Dota2"
5,"FIFA,Skyrim,Dota2,Assasins_Creed"
6,"PUBG,Counter_Strike,Skyrim,Dota2"
7,"Dota2,PUBG,Counter_Strike,NFS"
8,"Skyrim,Assasins_Creed,Dota2,Counter_Strike"
9,"NFS,Skyrim"
10,"Counter_Strike,Dota2,Mario"
11,"Counter_Strike,PUBG,Mario,Dota2"
12,"NFS,FIFA,Dota2"
13,"Counter_Strike,Dota2,Halo"
14,"Counter_Strike,Dota2"
15,"Halo,Mario,Counter_Strike"
16,"Skyrim,PUBG,Counter_Strike"
17,"FIFA,NFS,Counter_Strike"
18,"Counter_Strike,NFS"
19,"Mario,Dota2,Counter_Strike"
20,"Dota2,Counter_Strike"
```

Execution :-

Lets Roll back to our main program :-

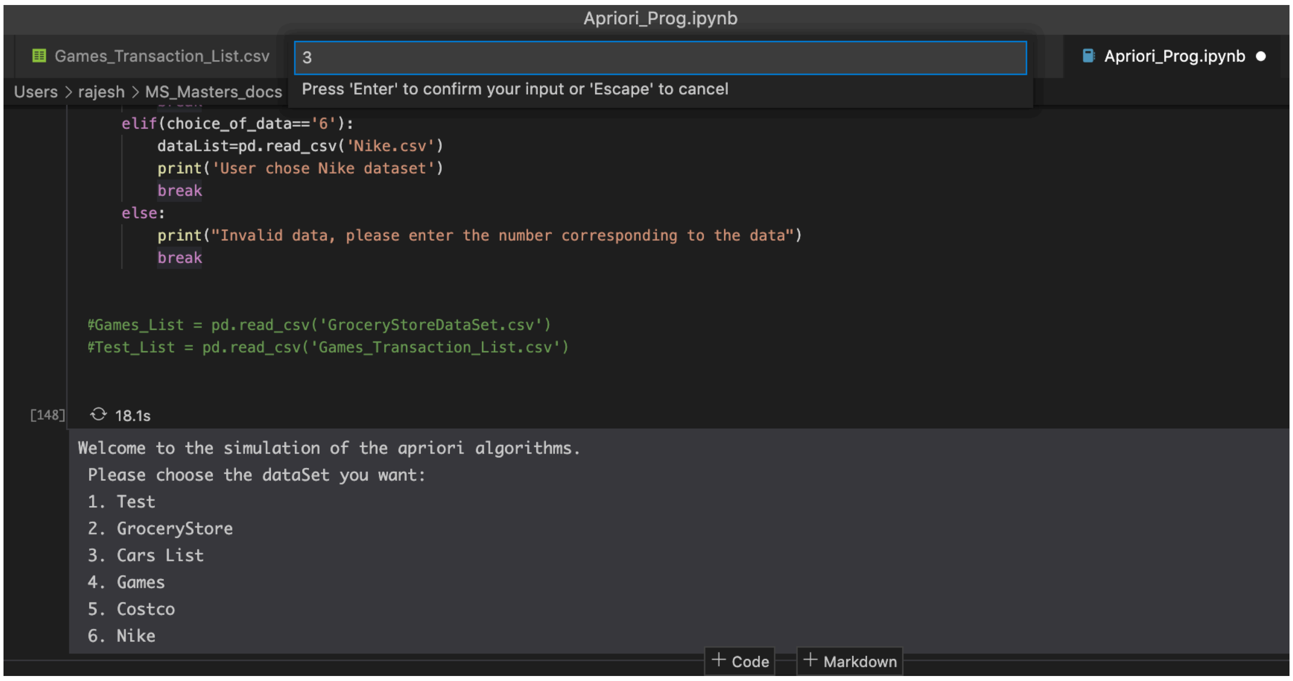
- At first I have created a switch case consisting of all 6 datasets which can be used by the user , and by taking an input from the user.



```python
            dataList=pd.read_csv('Costco.csv')
            print('User chose Costco dataset')
            break
        elif(choice_of_data=='6'):
            dataList=pd.read_csv('Nike.csv')
            print('User chose Nike dataset')
            break
        else:
            print("Invalid data, please enter the number corresponding to the data")
            break


    #Games_List = pd.read_csv('GroceryStoreDataSet.csv')
    #Test_List = pd.read_csv('Games_Transaction_List.csv')
```

```
[138]  ✓ 6.7s
Welcome to the simulation of the apriori algorithms.
 Please choose the dataSet you want:
 1. Test
 2. GroceryStore
 3. Cars List
 4. Games
 5. Costco
 6. Nike
User chose Cars List dataset
```

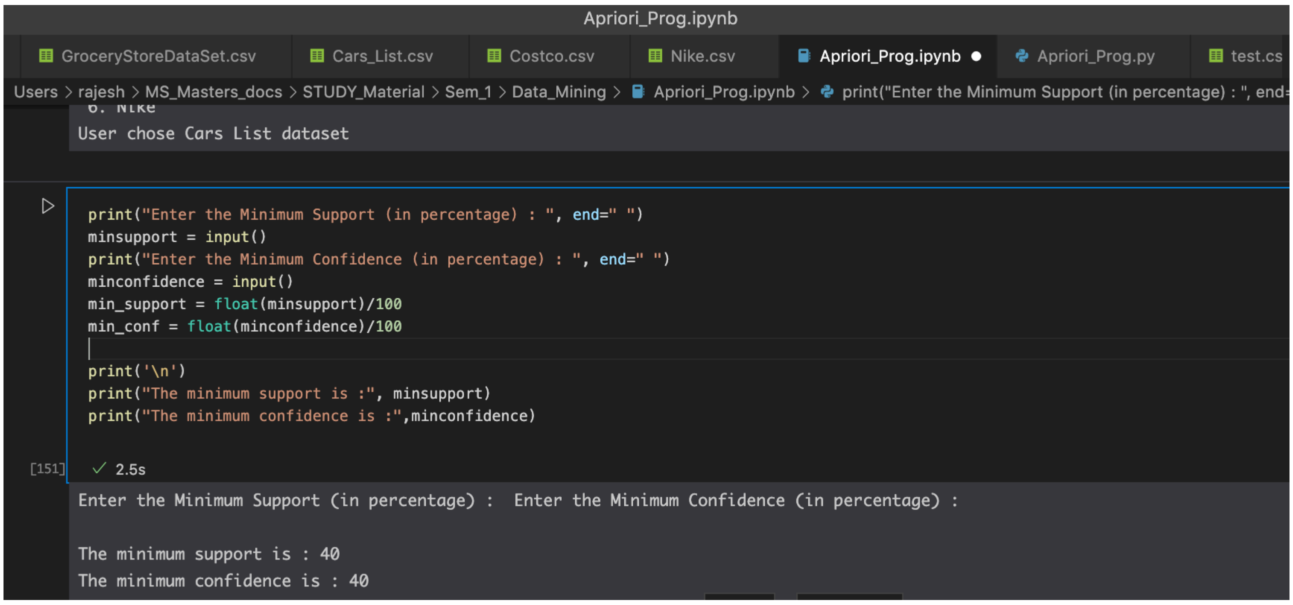
- From the above we can see that using the switch case , user can select any one of the dataset , if the users enters incorrect dataset it will prompt error saying “Enter the valid dataset”


Apriori_Prog.ipynb
Games_Transaction_List.csv
3
Press 'Enter' to confirm your input or 'Escape' to cancel
Apriori_Prog.ipynb
Users > rajesh > MS_Masters_docs
+ Code
+ Markdown

```
    elif(choice_of_data=='6'):
        dataList=pd.read_csv('Nike.csv')
        print('User chose Nike dataset')
        break
    else:
        print("Invalid data, please enter the number corresponding to the data")
        break


#Games_List = pd.read_csv('GroceryStoreDataSet.csv')
#Test_List = pd.read_csv('Games_Transaction_List.csv')
```

```
[148]  18.1s
Welcome to the simulation of the apriori algorithms.
 Please choose the dataSet you want:
 1. Test
 2. GroceryStore
 3. Cars List
 4. Games
 5. Costco
 6. Nike
```

- In above you can see after running the script , input is being asked , after I enter 3 in my Apriori_Prog.ipynb file the program will display the dataset which was selected.


Apriori_Prog.ipynb
GroceryStoreDataSet.csv
Cars_List.csv
Costco.csv
Nike.csv
Apriori_Prog.ipynb
Apriori_Prog.py
test.cs
Users > rajesh > MS_Masters_docs > STUDY_Material > Sem_1 > Data_Mining > Apriori_Prog.ipynb > print("Enter the Minimum Support (in percentage) : ", end=

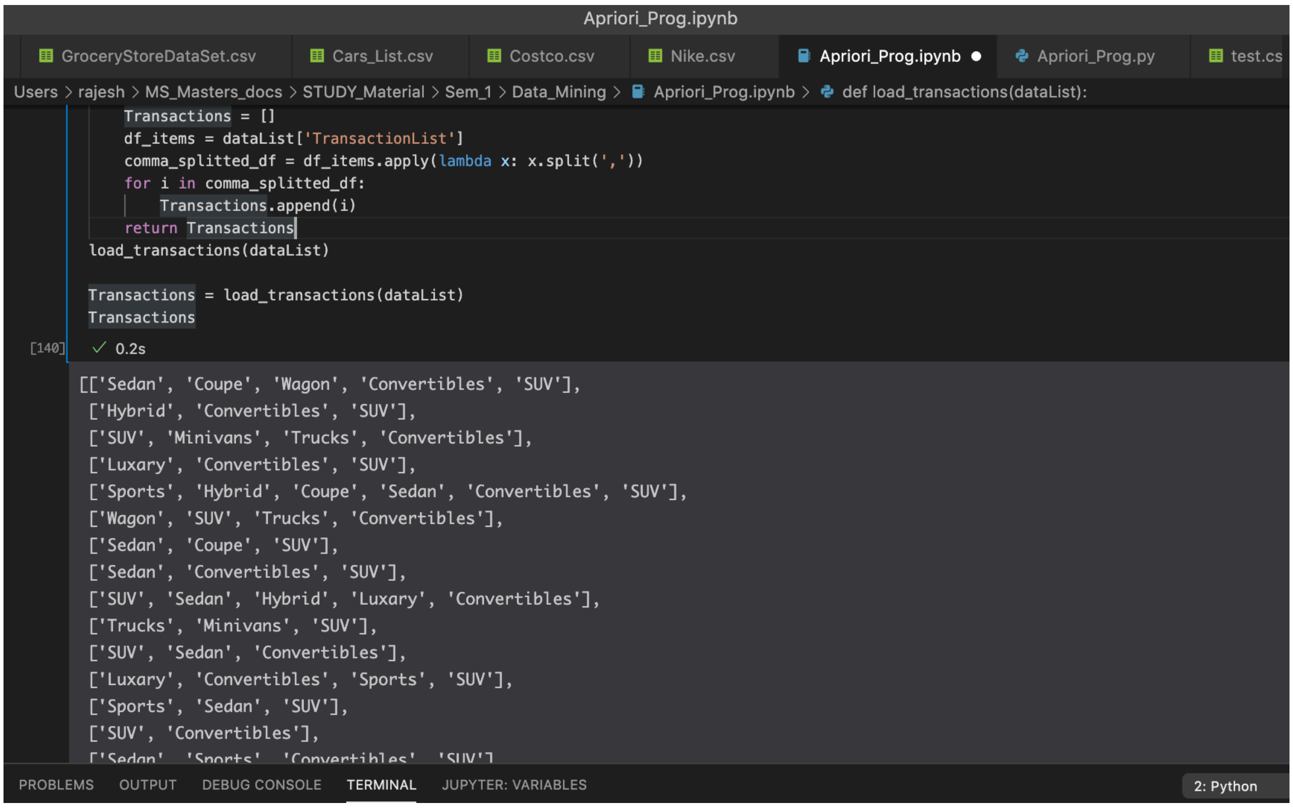
```
 6. Nike
User chose Cars List dataset
```

```
print("Enter the Minimum Support (in percentage) : ", end=" ")
minsupport = input()
print("Enter the Minimum Confidence (in percentage) : ", end=" ")
minconfidence = input()
min_support = float(minsupport)/100
min_conf = float(minconfidence)/100

print('\n')
print("The minimum support is :", minsupport)
print("The minimum confidence is :",minconfidence)
```

```
[151]  ✓ 2.5s
Enter the Minimum Support (in percentage) :  Enter the Minimum Confidence (in percentage) :

The minimum support is : 40
The minimum confidence is : 40
```

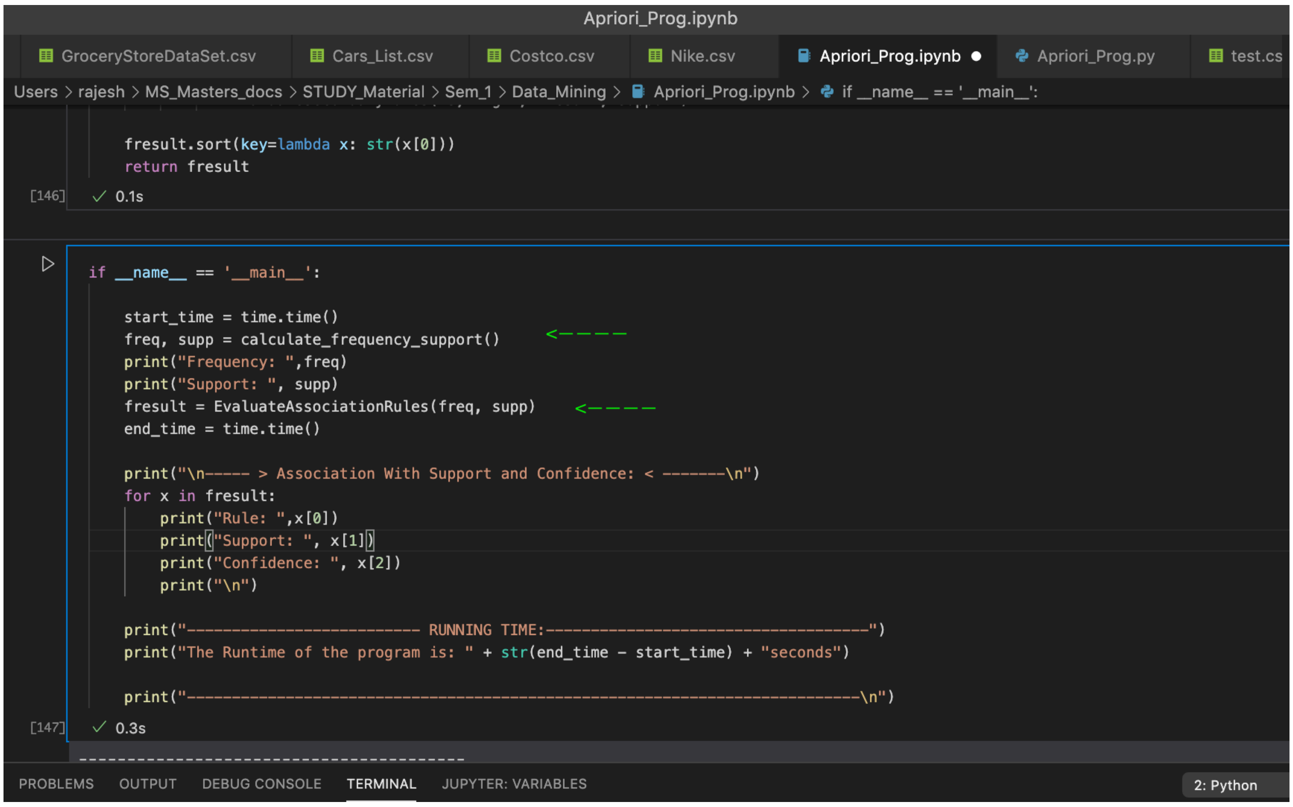
- In the above program , we are taking 2 input from the user i.e the minimum support and minimum confidence.

- After this we load the transaction list of the dataset as shown in below.


Apriori_Prog.ipynb
GroceryStoreDataSet.csv
Cars_List.csv
Costco.csv
Nike.csv
Apriori_Prog.ipynb
Apriori_Prog.py
test.cs
Users > rajesh > MS_Masters_docs > STUDY_Material > Sem_1 > Data_Mining > Apriori_Prog.ipynb > def load_transactions(dataList):
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER: VARIABLES
2: Python

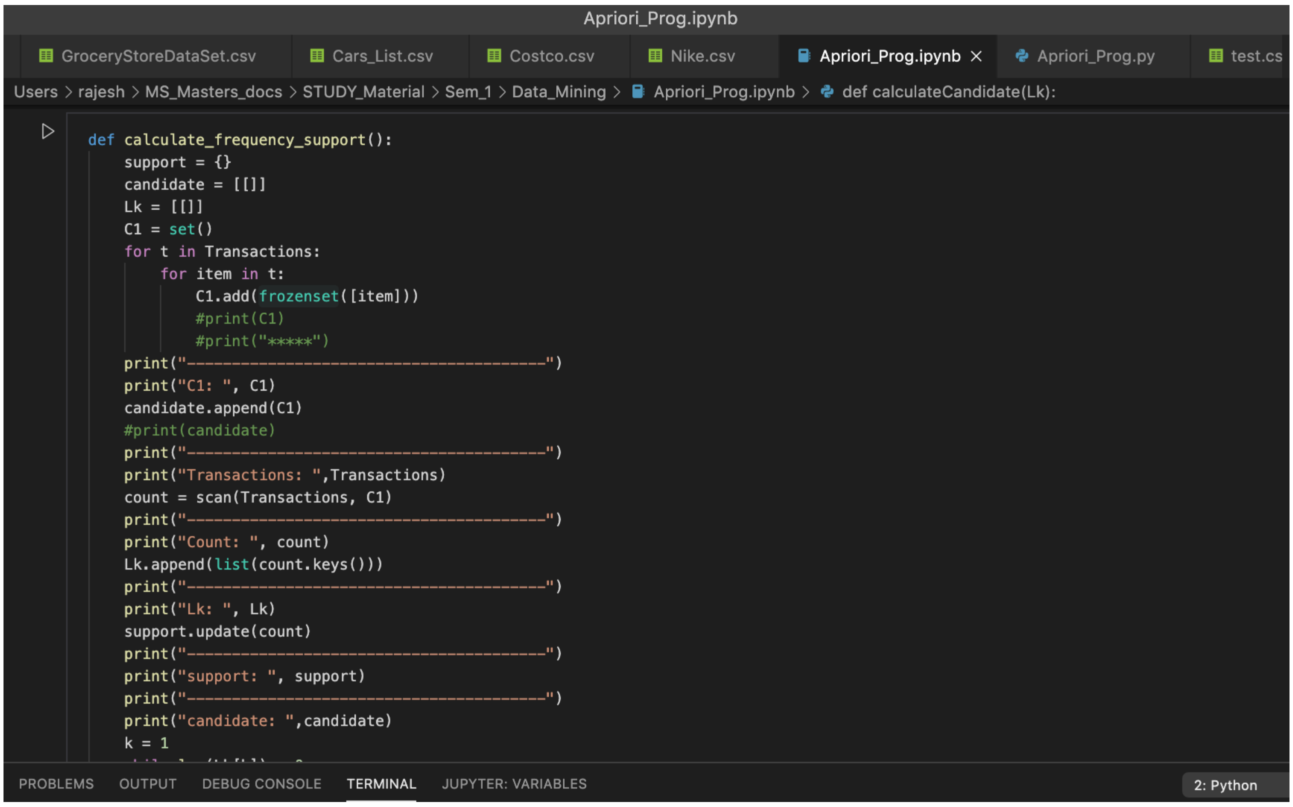
```
        Transactions = []
        df_items = dataList['TransactionList']
        comma_splitted_df = df_items.apply(lambda x: x.split(','))
        for i in comma_splitted_df:
            Transactions.append(i)
        return Transactions
    load_transactions(dataList)

    Transactions = load_transactions(dataList)
    Transactions
[140]  ✓ 0.2s

[['Sedan', 'Coupe', 'Wagon', 'Convertibles', 'SUV'],
 ['Hybrid', 'Convertibles', 'SUV'],
 ['SUV', 'Minivans', 'Trucks', 'Convertibles'],
 ['Luxary', 'Convertibles', 'SUV'],
 ['Sports', 'Hybrid', 'Coupe', 'Sedan', 'Convertibles', 'SUV'],
 ['Wagon', 'SUV', 'Trucks', 'Convertibles'],
 ['Sedan', 'Coupe', 'SUV'],
 ['Sedan', 'Convertibles', 'SUV'],
 ['SUV', 'Sedan', 'Hybrid', 'Luxary', 'Convertibles'],
 ['Trucks', 'Minivans', 'SUV'],
 ['SUV', 'Sedan', 'Convertibles'],
 ['Luxary', 'Convertibles', 'Sports', 'SUV'],
 ['Sports', 'Sedan', 'SUV'],
 ['SUV', 'Convertibles'],
 ['Sedan', 'Sports', 'Convertibles', 'SUV']
```

- We have used two main function for this apriori algorithm as highlighted in green arrows


Apriori_Prog.ipynb
GroceryStoreDataSet.csv
Cars_List.csv
Costco.csv
Nike.csv
Apriori_Prog.ipynb
Apriori_Prog.py
test.cs
Users > rajesh > MS_Masters_docs > STUDY_Material > Sem_1 > Data_Mining > Apriori_Prog.ipynb > if __name__ == '__main__':
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER: VARIABLES
2: Python

```
        fresult.sort(key=lambda x: str(x[0]))
        return fresult
[146]  ✓ 0.1s

    if __name__ == '__main__':

        start_time = time.time()
        freq, supp = calculate_frequency_support()     <-----
        print("Frequency: ",freq)
        print("Support: ", supp)
        fresult = EvaluateAssociationRules(freq, supp)     <-----
        end_time = time.time()

        print("\n----- > Association With Support and Confidence: < -------\n")
        for x in fresult:
            print("Rule: ",x[0])
            print("Support: ", x[1])
            print("Confidence: ", x[2])
            print("\n")

        print("------------------------- RUNNING TIME:-----------------------------------")
        print("The Runtime of the program is: " + str(end_time - start_time) + "seconds")

        print("------------------------------------------------------------------------\n")
[147]  ✓ 0.3s
   -------------------------------------------
```

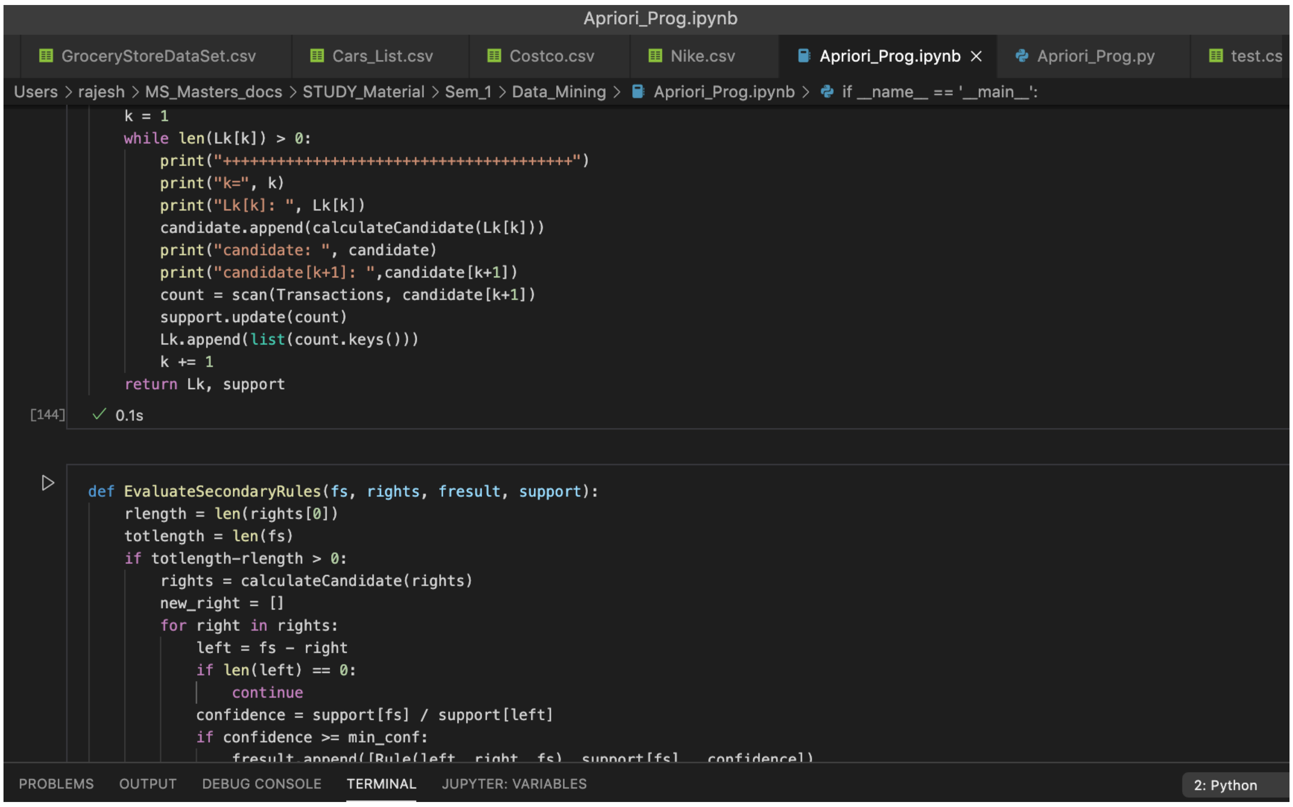
- One of function is used to calculate the support of the dataset and keeping it as a frozen set i.e to find the unique transaction items.



```python
def calculate_frequency_support():
    support = {}
    candidate = [[]]
    Lk = [[]]
    C1 = set()
    for t in Transactions:
        for item in t:
            C1.add(frozenset([item]))
            #print(C1)
            #print("*****")
    print("----------------------------------------")
    print("C1: ", C1)
    candidate.append(C1)
    #print(candidate)
    print("----------------------------------------")
    print("Transactions: ",Transactions)
    count = scan(Transactions, C1)
    print("----------------------------------------")
    print("Count: ", count)
    Lk.append(list(count.keys()))
    print("----------------------------------------")
    print("Lk: ", Lk)
    support.update(count)
    print("----------------------------------------")
    print("support: ", support)
    print("----------------------------------------")
    print("candidate: ",candidate)
    k = 1
```

- In above I have just used various print so that I can identify what the output is showing

- Another main function is to calculate the confidence of the dataset and after calculating the support of list of items and comparing it to the minimum support which was given by the user and then storing it in a list.

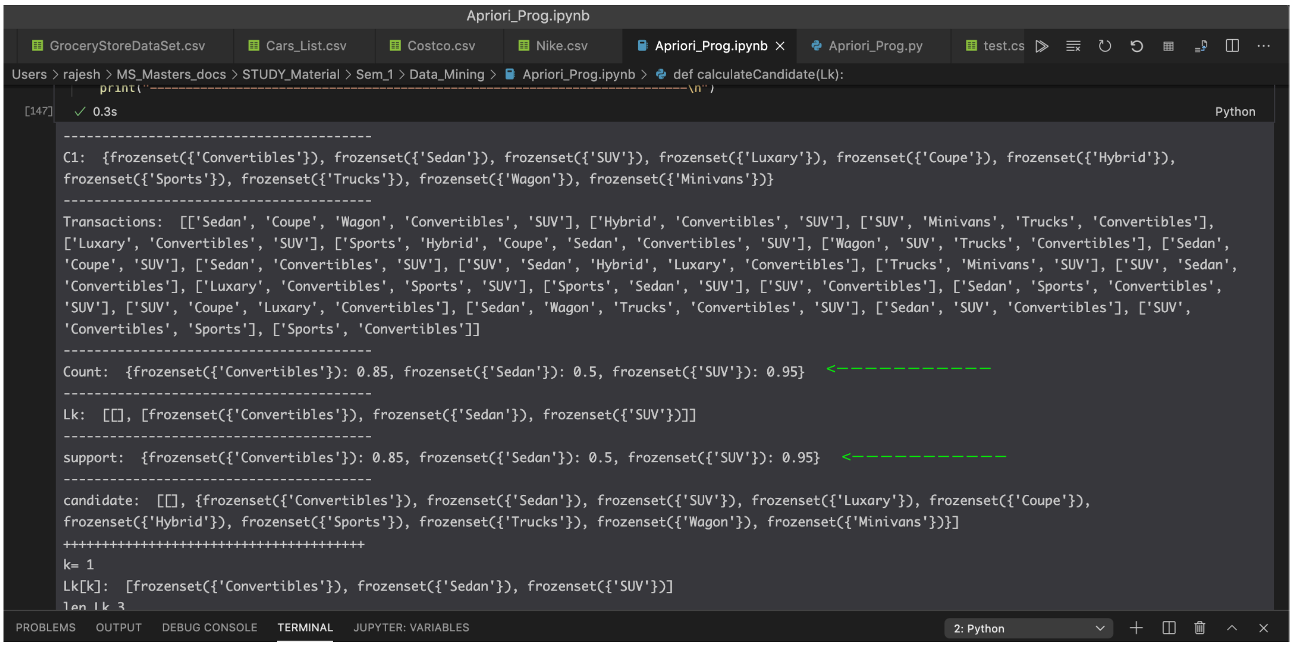


```
    k = 1
    while len(Lk[k]) > 0:
        print("++++++++++++++++++++++++++++++++++++++++")
        print("k=", k)
        print("Lk[k]: ", Lk[k])
        candidate.append(calculateCandidate(Lk[k]))
        print("candidate: ", candidate)
        print("candidate[k+1]: ",candidate[k+1])
        count = scan(Transactions, candidate[k+1])
        support.update(count)
        Lk.append(list(count.keys()))
        k += 1
    return Lk, support
[144]  ✓ 0.1s

def EvaluateSecondaryRules(fs, rights, fresult, support):
    rlength = len(rights[0])
    totlength = len(fs)
    if totlength-rlength > 0:
        rights = calculateCandidate(rights)
        new_right = []
        for right in rights:
            left = fs - right
            if len(left) == 0:
                continue
            confidence = support[fs] / support[left]
            if confidence >= min_conf:
                fresult.append([Rule(left, right, fs), support[fs], confidence])
```

- In the above we have used the formula to calculate the total confidence I.e

$$\text{Confidence}(X \rightarrow Z) = \text{Support}(X,Z) / \text{Support}(X)$$

- The output is shown below , as I have added few print lines , so to identify and easy to understand the algorithm

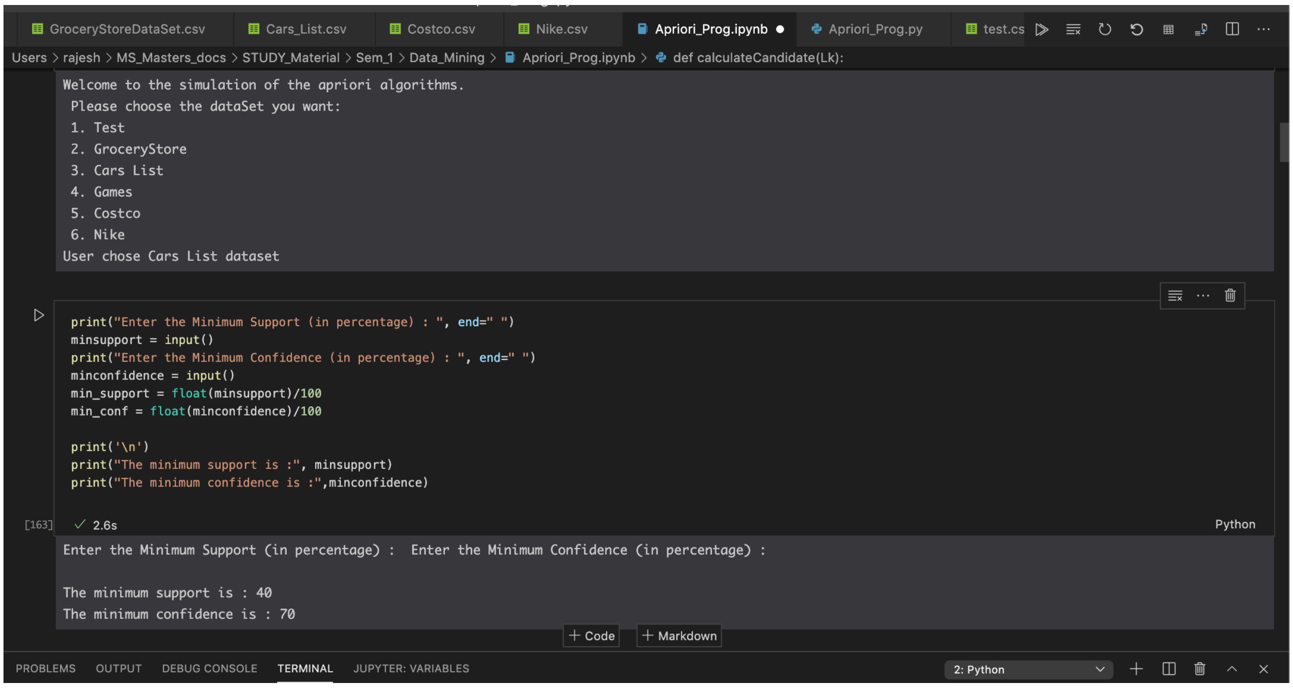


```
----------------------------------------
C1:  {frozenset({'Convertibles'}), frozenset({'Sedan'}), frozenset({'SUV'}), frozenset({'Luxary'}), frozenset({'Coupe'}), frozenset({'Hybrid'}),
frozenset({'Sports'}), frozenset({'Trucks'}), frozenset({'Wagon'}), frozenset({'Minivans'})}
----------------------------------------
Transactions:  [['Sedan', 'Coupe', 'Wagon', 'Convertibles', 'SUV'], ['Hybrid', 'Convertibles', 'SUV'], ['SUV', 'Minivans', 'Trucks', 'Convertibles'],
['Luxary', 'Convertibles', 'SUV'], ['Sports', 'Hybrid', 'Coupe', 'Sedan', 'Convertibles', 'SUV'], ['Wagon', 'SUV', 'Trucks', 'Convertibles'], ['Sedan',
'Coupe', 'SUV'], ['Sedan', 'Convertibles', 'SUV'], ['SUV', 'Sedan', 'Hybrid', 'Luxary', 'Convertibles'], ['Trucks', 'Minivans', 'SUV'], ['SUV', 'Sedan',
'Convertibles'], ['Luxary', 'Convertibles', 'Sports', 'SUV'], ['Sports', 'Sedan', 'SUV'], ['SUV', 'Convertibles'], ['Sedan', 'Sports', 'Convertibles',
'SUV'], ['SUV', 'Coupe', 'Luxary', 'Convertibles'], ['Sedan', 'Wagon', 'Trucks', 'Convertibles', 'SUV'], ['Sedan', 'SUV', 'Convertibles'], ['SUV',
'Convertibles', 'Sports'], ['Sports', 'Convertibles']]
----------------------------------------
Count:  {frozenset({'Convertibles'}): 0.85, frozenset({'Sedan'}): 0.5, frozenset({'SUV'}): 0.95}  <------------
----------------------------------------
Lk:  [[], [frozenset({'Convertibles'}), frozenset({'Sedan'}), frozenset({'SUV'})]]
----------------------------------------
support:  {frozenset({'Convertibles'}): 0.85, frozenset({'Sedan'}): 0.5, frozenset({'SUV'}): 0.95}  <------------
----------------------------------------
candidate:  [[], {frozenset({'Convertibles'}), frozenset({'Sedan'}), frozenset({'SUV'}), frozenset({'Luxary'}), frozenset({'Coupe'}),
frozenset({'Hybrid'}), frozenset({'Sports'}), frozenset({'Trucks'}), frozenset({'Wagon'}), frozenset({'Minivans'})}]
++++++++++++++++++++++++++++++++++++++++
k= 1
Lk[k]:  [frozenset({'Convertibles'}), frozenset({'Sedan'}), frozenset({'SUV'})]
len Lk 3
```

- The result shown above have calculated the support of each frequent dataset by comparing its minimum support and displayed only the items which are satisfied under the condition.

- Below is the final output resulting in the Association rules with Support and Confidence of each transaction items.

Result_1 :- Dataset = Cars_List.csv with min support = 40 and min confidence = 70

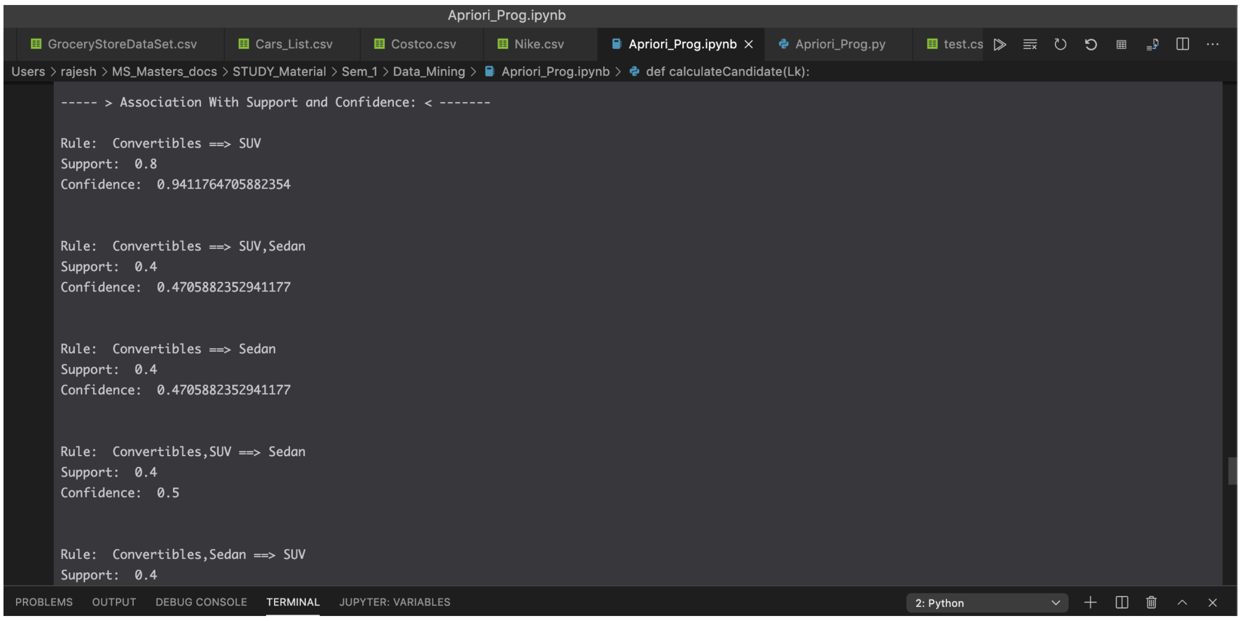


```
Welcome to the simulation of the apriori algorithms.
 Please choose the dataSet you want:
 1. Test
 2. GroceryStore
 3. Cars List
 4. Games
 5. Costco
 6. Nike
User chose Cars List dataset
```

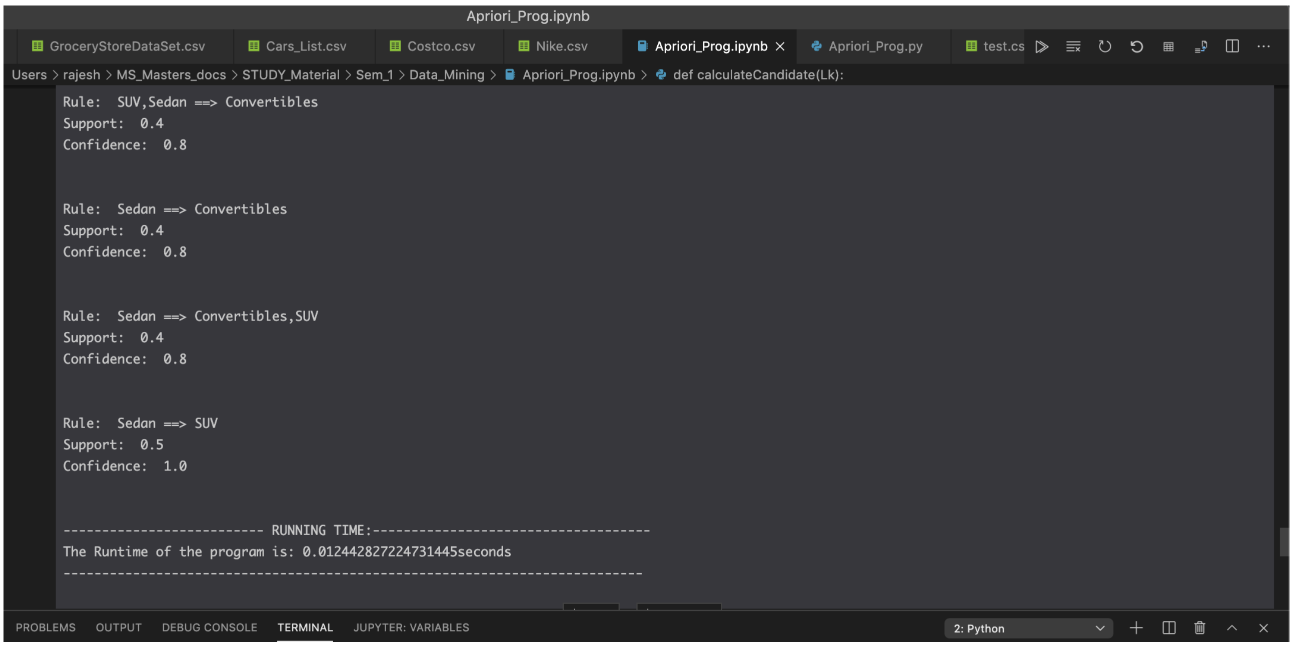
```python
print("Enter the Minimum Support (in percentage) : ", end=" ")
minsupport = input()
print("Enter the Minimum Confidence (in percentage) : ", end=" ")
minconfidence = input()
min_support = float(minsupport)/100
min_conf = float(minconfidence)/100

print('\n')
print("The minimum support is :", minsupport)
print("The minimum confidence is :",minconfidence)
```

[163] ✓ 2.6s Python

```
Enter the Minimum Support (in percentage) :  Enter the Minimum Confidence (in percentage) :

The minimum support is : 40
The minimum confidence is : 70
```



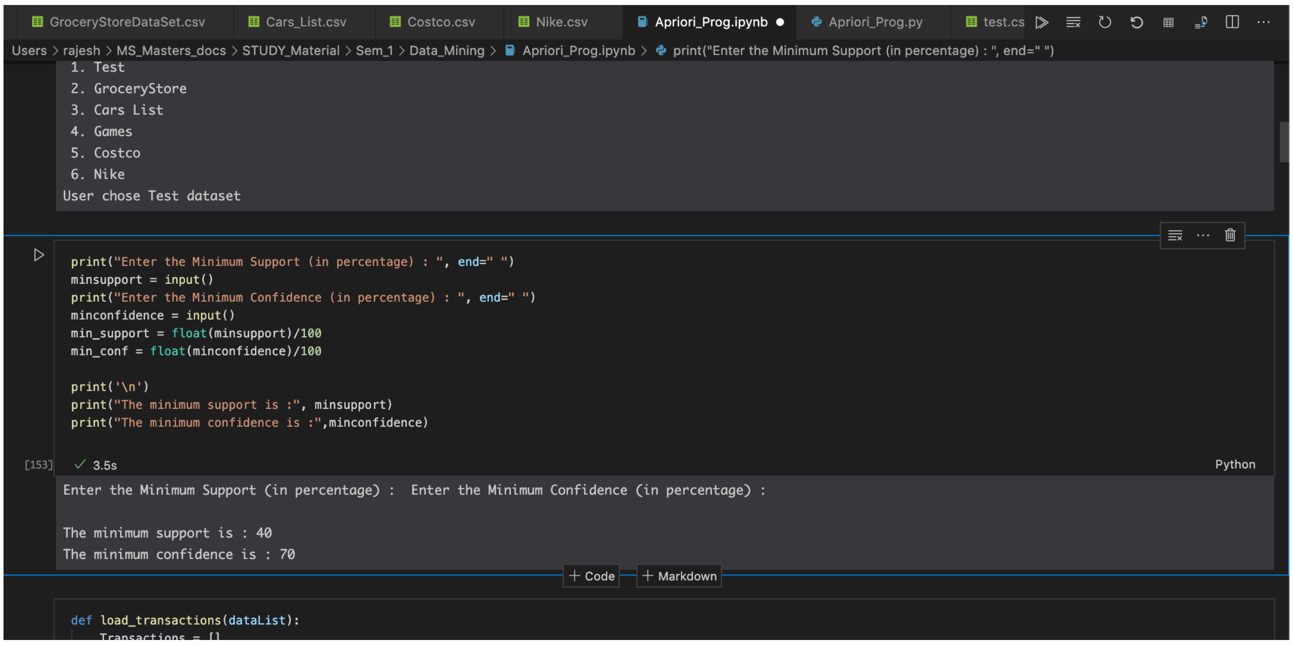
```
----- > Association With Support and Confidence: < -------

Rule:  Convertibles ==> SUV
Support:  0.8
Confidence:  0.9411764705882354


Rule:  Convertibles ==> SUV,Sedan
Support:  0.4
Confidence:  0.4705882352941177


Rule:  Convertibles ==> Sedan
Support:  0.4
Confidence:  0.4705882352941177


Rule:  Convertibles,SUV ==> Sedan
Support:  0.4
Confidence:  0.5


Rule:  Convertibles,Sedan ==> SUV
Support:  0.4
```



```
Rule:  SUV,Sedan ==> Convertibles
Support:  0.4
Confidence:  0.8


Rule:  Sedan ==> Convertibles
Support:  0.4
Confidence:  0.8


Rule:  Sedan ==> Convertibles,SUV
Support:  0.4
Confidence:  0.8


Rule:  Sedan ==> SUV
Support:  0.5
Confidence:  1.0


------------------------- RUNNING TIME:-----------------------------------
The Runtime of the program is: 0.012442827224731445seconds
----------------------------------------------------------------------------
```

- Also the program run-time is less by adding the start.time and end time , deducting the end time from the start time we can calculate the time taken by the apriori algorithm to execute the result

Result_2 :- Dataset = Test with min support = 40 and min confidence = 70

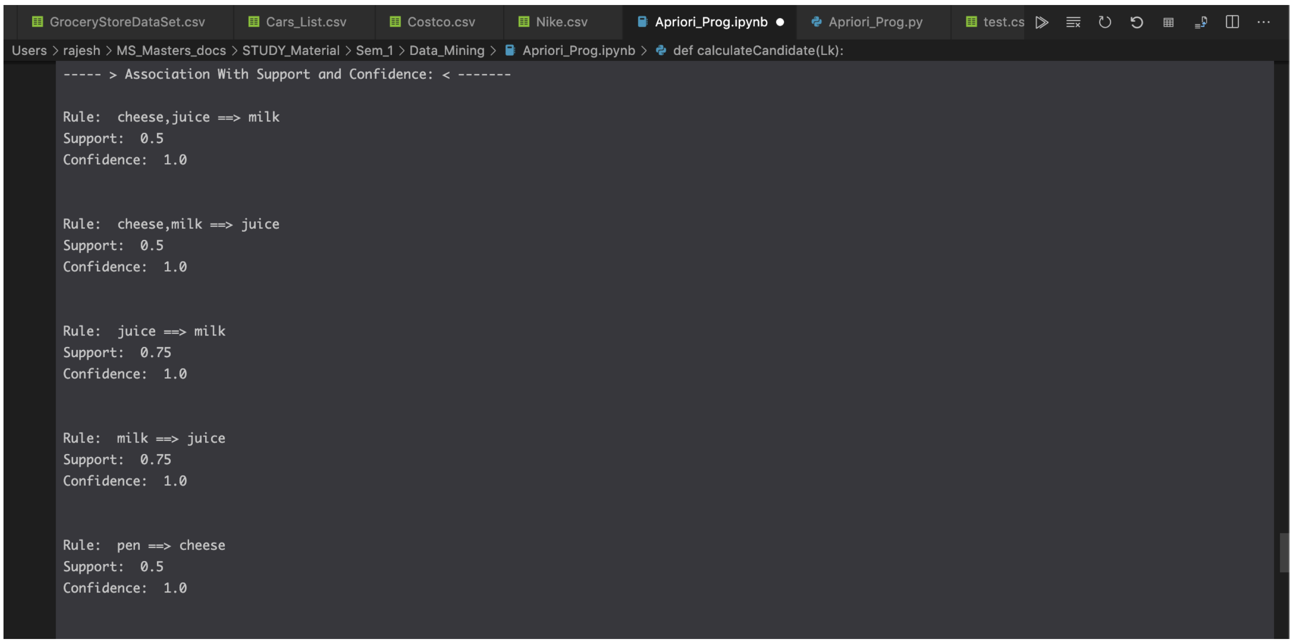

GroceryStoreDataSet.csv
Cars_List.csv
Costco.csv
Nike.csv
Apriori_Prog.ipynb
Apriori_Prog.py
test.cs
Users > rajesh > MS_Masters_docs > STUDY_Material > Sem_1 > Data_Mining > Apriori_Prog.ipynb > print("Enter the Minimum Support (in percentage) : ", end=" ")
[153]
✓ 3.5s
Python
+ Code
+ Markdown

```
 1. Test
 2. GroceryStore
 3. Cars List
 4. Games
 5. Costco
 6. Nike
User chose Test dataset
```

```python
print("Enter the Minimum Support (in percentage) : ", end=" ")
minsupport = input()
print("Enter the Minimum Confidence (in percentage) : ", end=" ")
minconfidence = input()
min_support = float(minsupport)/100
min_conf = float(minconfidence)/100

print('\n')
print("The minimum support is :", minsupport)
print("The minimum confidence is :",minconfidence)
```

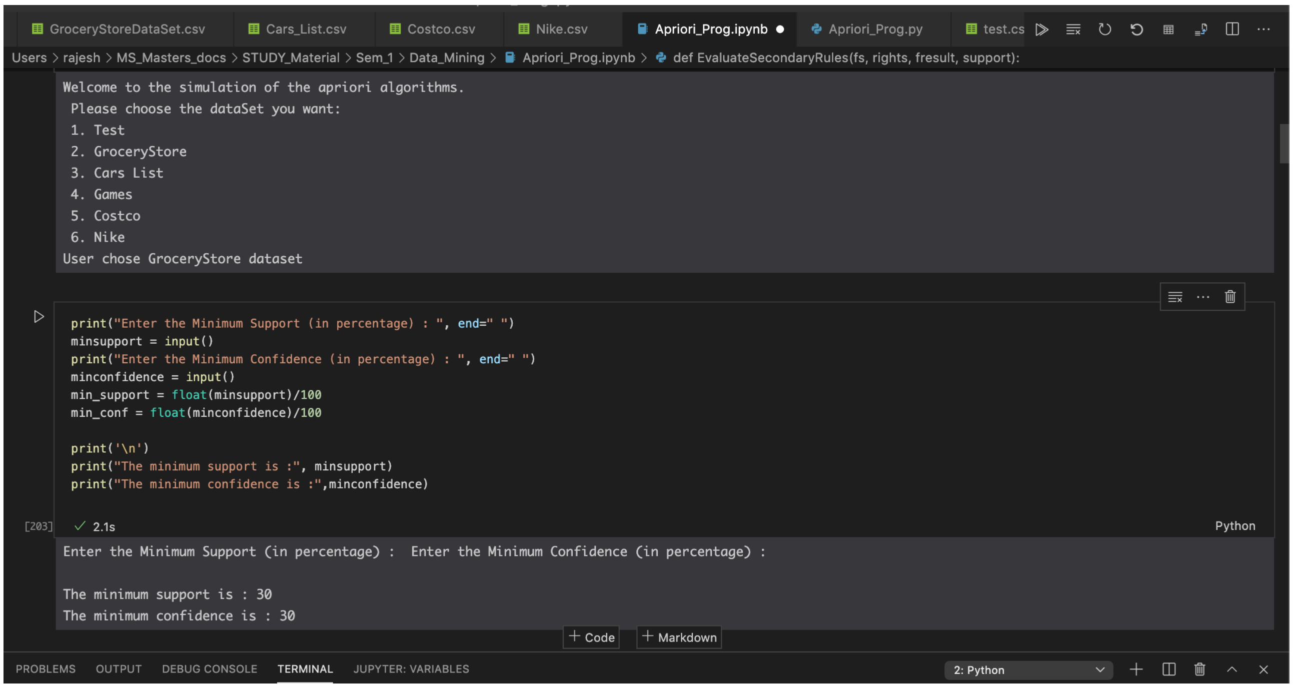
```
Enter the Minimum Support (in percentage) :  Enter the Minimum Confidence (in percentage) :

The minimum support is : 40
The minimum confidence is : 70
```

```python
def load_transactions(dataList):
    Transactions = []
```


GroceryStoreDataSet.csv
Cars_List.csv
Costco.csv
Nike.csv
Apriori_Prog.ipynb
Apriori_Prog.py
test.cs
Users > rajesh > MS_Masters_docs > STUDY_Material > Sem_1 > Data_Mining > Apriori_Prog.ipynb > def calculateCandidate(Lk):

```
----- > Association With Support and Confidence: < -------

Rule:  cheese,juice ==> milk
Support:  0.5
Confidence:  1.0


Rule:  cheese,milk ==> juice
Support:  0.5
Confidence:  1.0


Rule:  juice ==> milk
Support:  0.75
Confidence:  1.0


Rule:  milk ==> juice
Support:  0.75
Confidence:  1.0


Rule:  pen ==> cheese
Support:  0.5
Confidence:  1.0
```

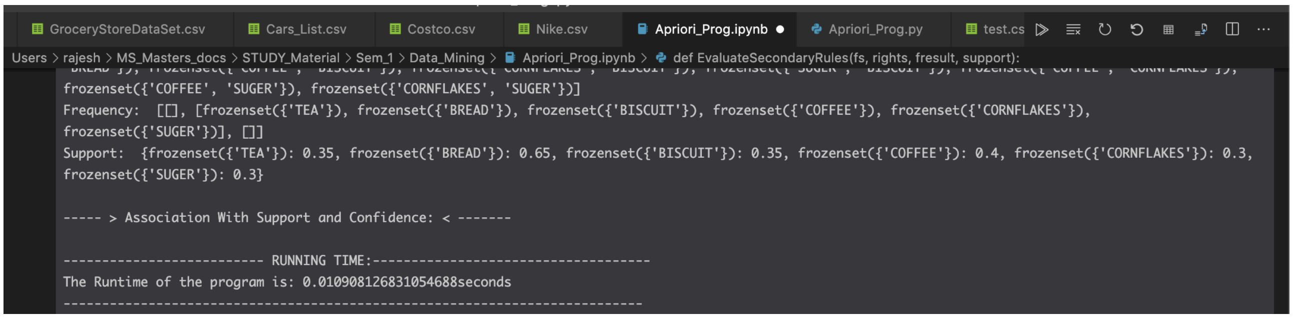
Result_3 :- Dataset = GroceryStore , where min support = 30 and min confidence = 30



```
Welcome to the simulation of the apriori algorithms.
 Please choose the dataSet you want:
 1. Test
 2. GroceryStore
 3. Cars List
 4. Games
 5. Costco
 6. Nike
User chose GroceryStore dataset
```

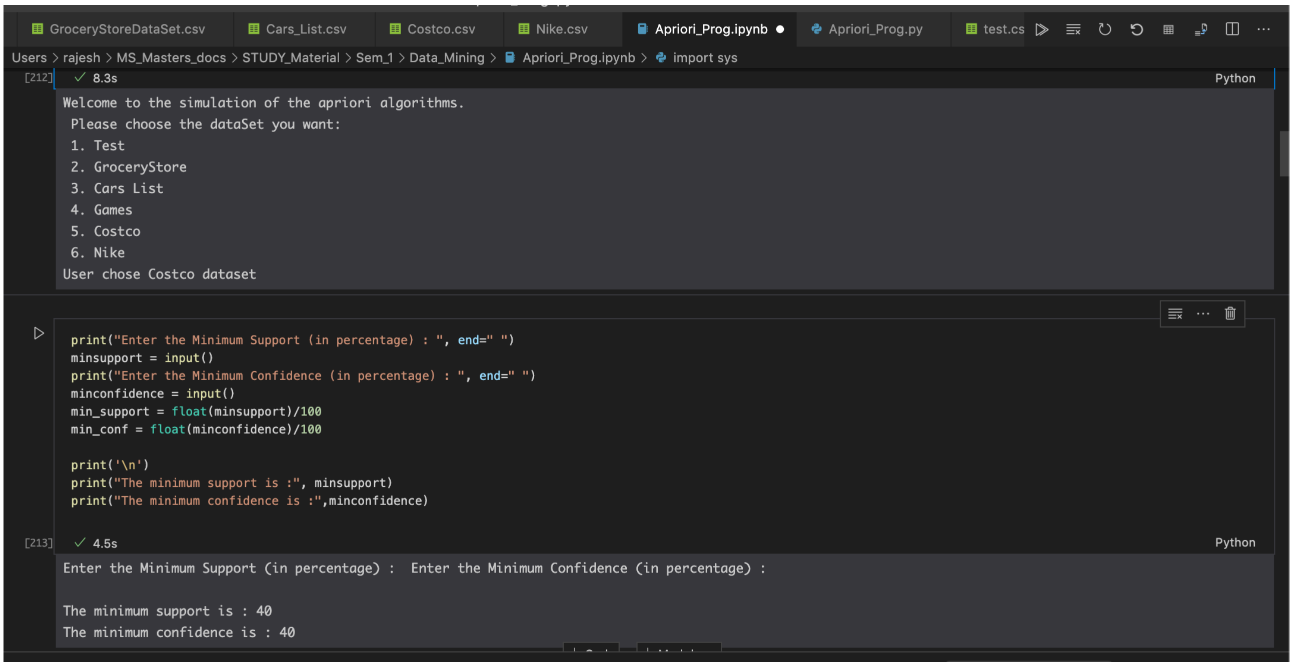
```python
print("Enter the Minimum Support (in percentage) : ", end=" ")
minsupport = input()
print("Enter the Minimum Confidence (in percentage) : ", end=" ")
minconfidence = input()
min_support = float(minsupport)/100
min_conf = float(minconfidence)/100

print('\n')
print("The minimum support is :", minsupport)
print("The minimum confidence is :",minconfidence)
```

```
Enter the Minimum Support (in percentage) :  Enter the Minimum Confidence (in percentage) :

The minimum support is : 30
The minimum confidence is : 30
```


GroceryStoreDataSet.csv
Cars_List.csv
Costco.csv
Nike.csv
Apriori_Prog.ipynb
Apriori_Prog.py
test.cs
Users > rajesh > MS_Masters_docs > STUDY_Material > Sem_1 > Data_Mining > Apriori_Prog.ipynb > def EvaluateSecondaryRules(fs, rights, fresult, support):

```
frozenset({'COFFEE', 'SUGER'}), frozenset({'CORNFLAKES', 'SUGER'})]
Frequency:  [[], [frozenset({'TEA'}), frozenset({'BREAD'}), frozenset({'BISCUIT'}), frozenset({'COFFEE'}), frozenset({'CORNFLAKES'}),
frozenset({'SUGER'})], []]
Support:  {frozenset({'TEA'}): 0.35, frozenset({'BREAD'}): 0.65, frozenset({'BISCUIT'}): 0.35, frozenset({'COFFEE'}): 0.4, frozenset({'CORNFLAKES'}): 0.3,
frozenset({'SUGER'}): 0.3}

----- > Association With Support and Confidence: < -------

------------------------ RUNNING TIME:-----------------------------------
The Runtime of the program is: 0.010908126831054688seconds
-------------------------------------------------------------------------
```

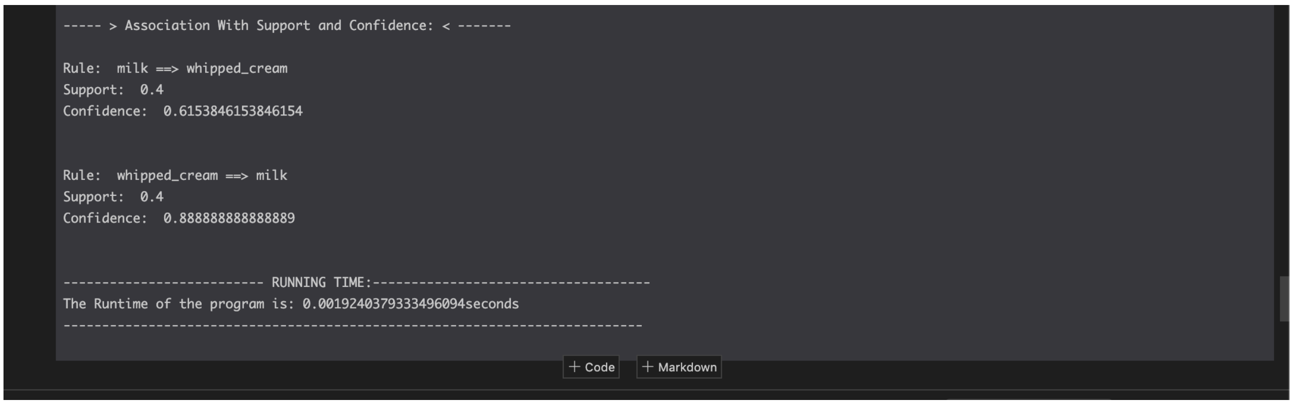
Result_4 :- Dataset = Costco , where min support = 40 and min confidence = 40



```
Welcome to the simulation of the apriori algorithms.
 Please choose the dataSet you want:
 1. Test
 2. GroceryStore
 3. Cars List
 4. Games
 5. Costco
 6. Nike
User chose Costco dataset
```

```python
print("Enter the Minimum Support (in percentage) : ", end=" ")
minsupport = input()
print("Enter the Minimum Confidence (in percentage) : ", end=" ")
minconfidence = input()
min_support = float(minsupport)/100
min_conf = float(minconfidence)/100

print('\n')
print("The minimum support is :", minsupport)
print("The minimum confidence is :",minconfidence)
```

```
Enter the Minimum Support (in percentage) :  Enter the Minimum Confidence (in percentage) :

The minimum support is : 40
The minimum confidence is : 40
```



```
----- > Association With Support and Confidence: < -------

Rule:  milk ==> whipped_cream
Support:  0.4
Confidence:  0.6153846153846154


Rule:  whipped_cream ==> milk
Support:  0.4
Confidence:  0.888888888888889


------------------------ RUNNING TIME:-----------------------------------
The Runtime of the program is: 0.0019240379333496094seconds
-------------------------------------------------------------------------
```

==

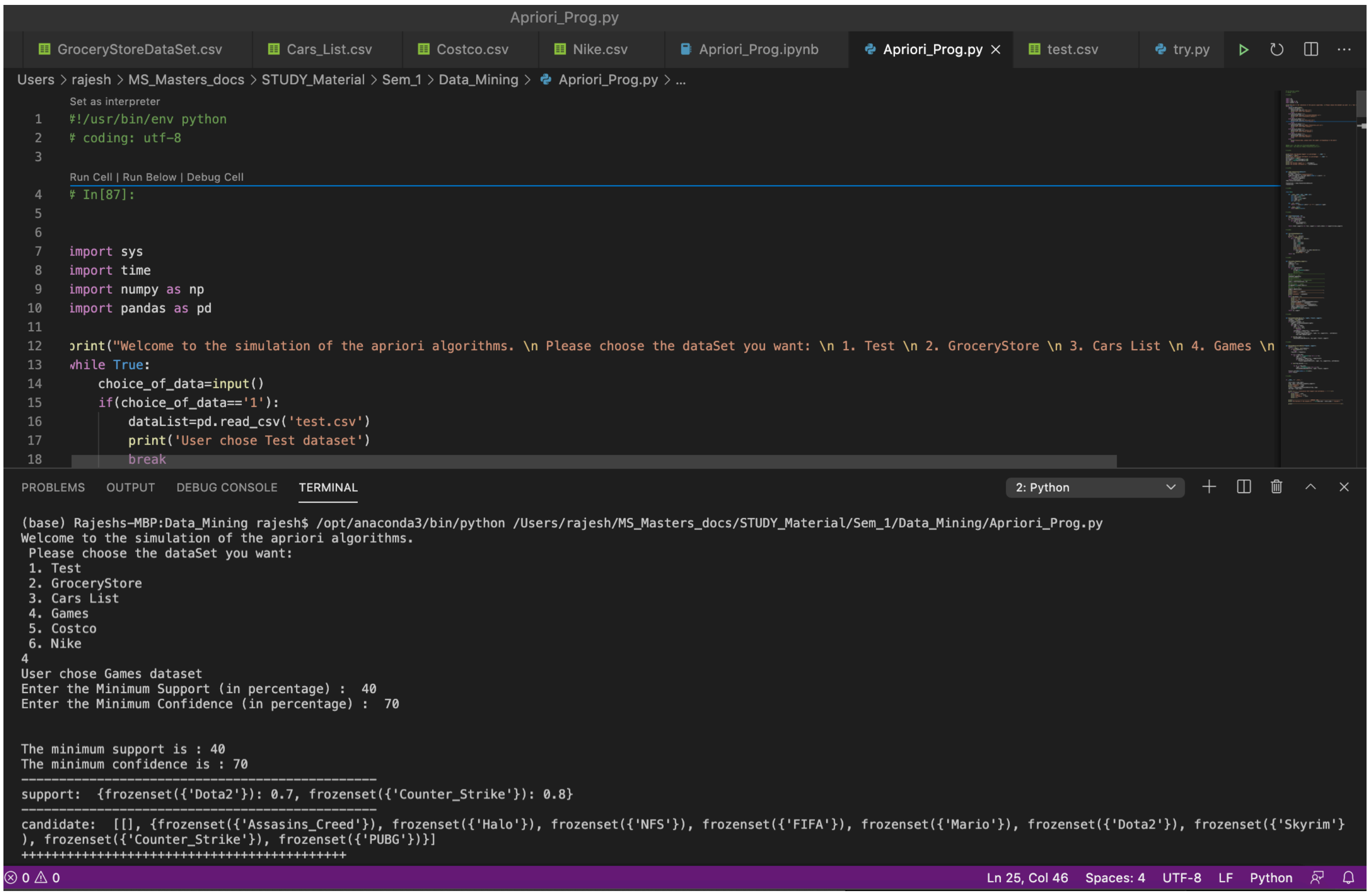
Executing the same program in Python :-

After converting the Jupyter file to python by executing the below command in terminal of Visual Studio

```
jupyter nbconvert --to script FileName.ipynb
```

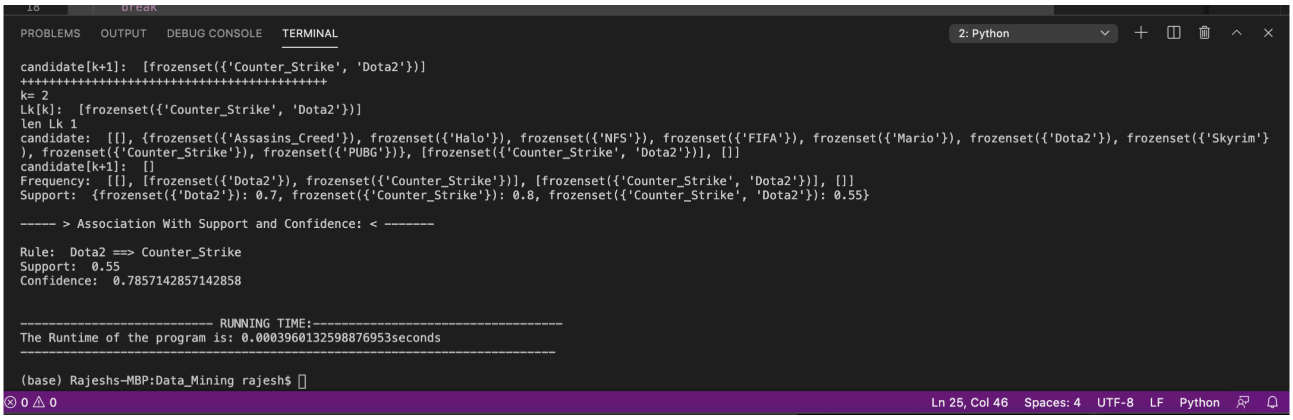
Executed the program using python , now selecting a different dataset Games_Transaction_List.csv

In the above snapshot as you can see we have taken the input from the user and presented the output by calculating the minimum support and confidence.



```python
#!/usr/bin/env python
# coding: utf-8

# In[87]:



import sys
import time
import numpy as np
import pandas as pd

print("Welcome to the simulation of the apriori algorithms. \n Please choose the dataSet you want: \n 1. Test \n 2. GroceryStore \n 3. Cars List \n 4. Games \n
while True:
    choice_of_data=input()
    if(choice_of_data=='1'):
        dataList=pd.read_csv('test.csv')
        print('User chose Test dataset')
        break
```

```
(base) Rajeshs-MBP:Data_Mining rajesh$ /opt/anaconda3/bin/python /Users/rajesh/MS_Masters_docs/STUDY_Material/Sem_1/Data_Mining/Apriori_Prog.py
Welcome to the simulation of the apriori algorithms.
 Please choose the dataSet you want:
 1. Test
 2. GroceryStore
 3. Cars List
 4. Games
 5. Costco
 6. Nike
4
User chose Games dataset
Enter the Minimum Support (in percentage) :  40
Enter the Minimum Confidence (in percentage) :  70


The minimum support is : 40
The minimum confidence is : 70
-----------------------------------------------
support:  {frozenset({'Dota2'}): 0.7, frozenset({'Counter_Strike'}): 0.8}
-----------------------------------------------
candidate:  [[], {frozenset({'Assasins_Creed'}), frozenset({'Halo'}), frozenset({'NFS'}), frozenset({'FIFA'}), frozenset({'Mario'}), frozenset({'Dota2'}), frozenset({'Skyrim'}
), frozenset({'Counter_Strike'}), frozenset({'PUBG'})}]
+++++++++++++++++++++++++++++++++++++++++++
```



```
candidate[k+1]:  [frozenset({'Counter_Strike', 'Dota2'})]
+++++++++++++++++++++++++++++++++++++++++++
k= 2
Lk[k]:  [frozenset({'Counter_Strike', 'Dota2'})]
len Lk 1
candidate:  [[], {frozenset({'Assasins_Creed'}), frozenset({'Halo'}), frozenset({'NFS'}), frozenset({'FIFA'}), frozenset({'Mario'}), frozenset({'Dota2'}), frozenset({'Skyrim'}
), frozenset({'Counter_Strike'}), frozenset({'PUBG'})}, [frozenset({'Counter_Strike', 'Dota2'})], []]
candidate[k+1]:  []
Frequency:  [[], [frozenset({'Dota2'}), frozenset({'Counter_Strike'})], [frozenset({'Counter_Strike', 'Dota2'})], []]
Support:  {frozenset({'Dota2'}): 0.7, frozenset({'Counter_Strike'}): 0.8, frozenset({'Counter_Strike', 'Dota2'}): 0.55}

----- > Association With Support and Confidence: < -------

Rule:  Dota2 ==> Counter_Strike
Support:  0.55
Confidence:  0.7857142857142858


-------------------------- RUNNING TIME:-----------------------------------
The Runtime of the program is: 0.0003960132598876953seconds
--------------------------------------------------------------------------

(base) Rajeshs-MBP:Data_Mining rajesh$ 
```

GitHub Link :- https://github.com/Rajesh007x/Data_Mining_MidTerm

Referral Links :- https://code.visualstudio.com/docs/python/data-science-tutorial
https://adataanalyst.com/machine-learning/apriori-algorithm-python-3-0/
https://code.visualstudio.com/docs/python/jupyter-support
https://jupyter-notebook.readthedocs.io/en/stable/