VIT
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

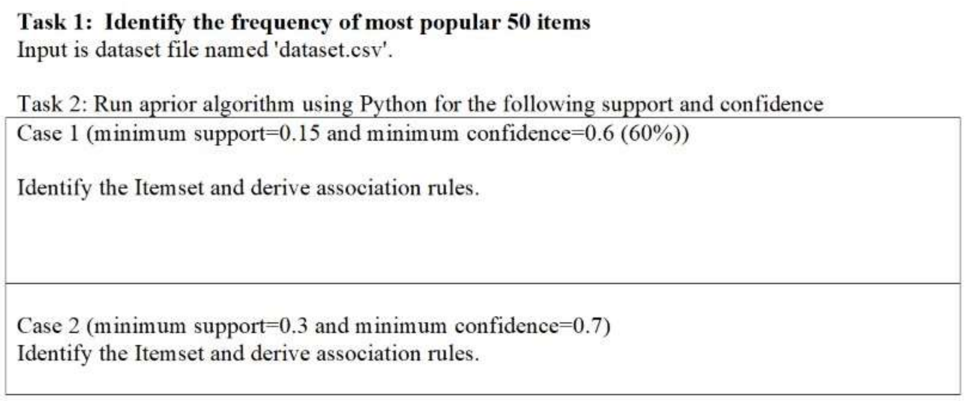
Programme	:	**B.Tech**	Semester	:	**Win 21-22**
Course	:	**Web Mining Lab**	Code	:	**CSE3024**
Faculty	:	**Dr.Bhuvaneswari A**	Slot	:	**L7+L8**
Date	:	**12-04-2022**	Marks	:	**10 Points**
Student name	:	**Tanmay Mahajan**	Registration No	:	**19BCE1735**

Market-Basket-Analysis of Grocery Dataset collected from an online grocery billing data. Market Basket Analysis is the analysis of past buying behaviour of online customers to find out which are the products that are bought together by the customers. That means to find out the association between various products. If the retail's management can find this association, while placing the products in the shop, these associated products can be put together. Or, when seeing that a customer is buying a product, the salesman can offer the associated product to the customer.

We find this association by Association Rule learning which a machinelearning rule is based approach that generates relationship between variables in a dataset. It has major application in retail industry including e-commerce.

Problem Statement

To determine the association between various products in the basket by analysing the customer purchase pattern of multiple items.



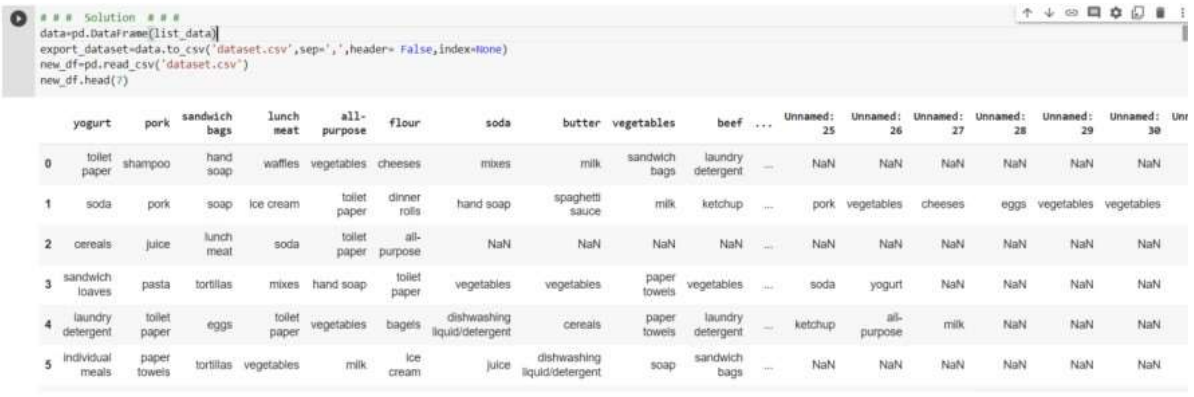
Task 1: Identify the frequency of most popular 50 items
Input is dataset file named 'dataset.csv'.

Task 2: Run aprior algorithm using Python for the following support and confidence

Case 1 (minimum support=0.15 and minimum confidence=0.6 (60%)) Identify the Itemset and derive association rules.
Case 2 (minimum support=0.3 and minimum confidence=0.7) Identify the Itemset and derive association rules.

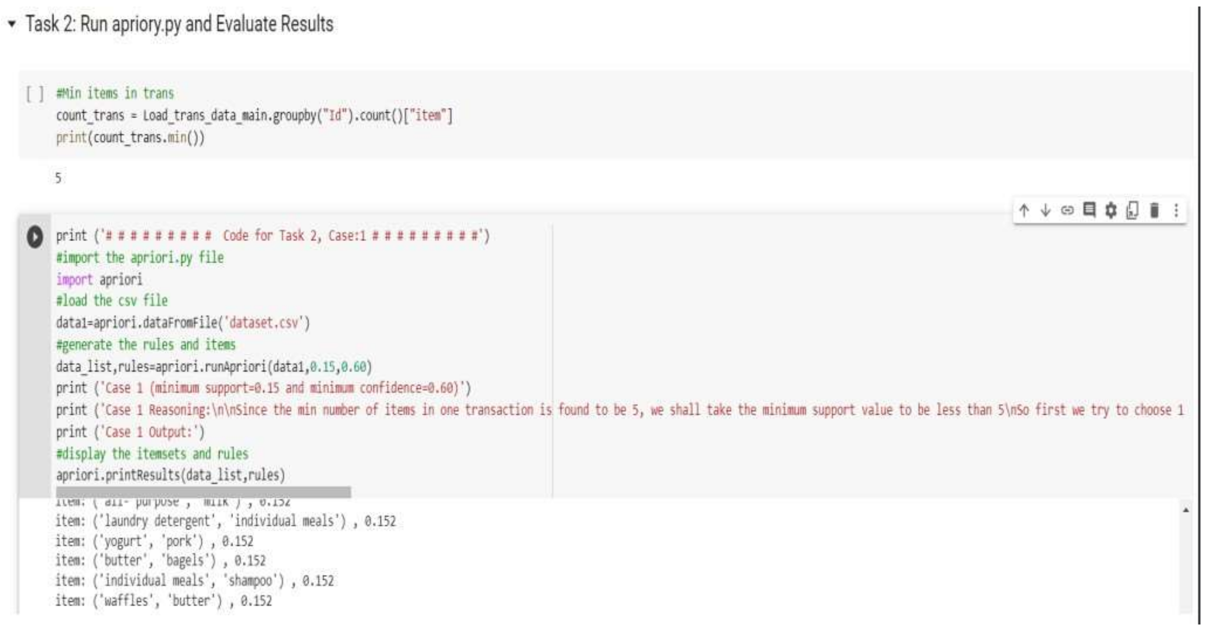
Expected Output: As per our rules with Min. Confidence of 70%,
Case 3 (minimum support=0.4 and minimum confidence=0.85) Identify the Itemset and derive association rules.

CODE:



```
### Solution ###
data=pd.DataFrame(list_data)
export_dataset=data.to_csv('dataset.csv',sep=',',header= False,index=None)
new_df=pd.read_csv('dataset.csv')
new_df.head(7)
```

	yogurt	pork	sandwich bags	lunch meat	all-purpose	flour	soda	butter	vegetables	beef	...	Unnamed: 25	Unnamed: 26	Unnamed: 27	Unnamed: 28	Unnamed: 29	Unnamed: 30	Unr
0	toilet paper	shampoo	hand soap	waffles	vegetables	cheeses	mixes	milk	sandwich bags	laundry detergent	...	NaN	NaN	NaN	NaN	NaN	NaN	
1	soda	pork	soap	ice cream	toilet paper	dinner rolls	hand soap	spaghetti sauce	milk	ketchup	...	pork	vegetables	cheeses	eggs	vegetables	vegetables	
2	cereals	juice	lunch meat	soda	toilet paper	all-purpose	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	
3	sandwich loaves	pasta	tortillas	mixes	hand soap	toilet paper	vegetables	vegetables	paper towels	vegetables	...	soda	yogurt	NaN	NaN	NaN	NaN	
4	laundry detergent	toilet paper	eggs	toilet paper	vegetables	bagels	dishwashing liquid/detergent	cereals	paper towels	laundry detergent	...	ketchup	all-purpose	milk	NaN	NaN	NaN	
5	individual meals	paper towels	tortillas	vegetables	milk	ice cream	juice	dishwashing liquid/detergent	soap	sandwich bags	...	NaN	NaN	NaN	NaN	NaN	NaN	



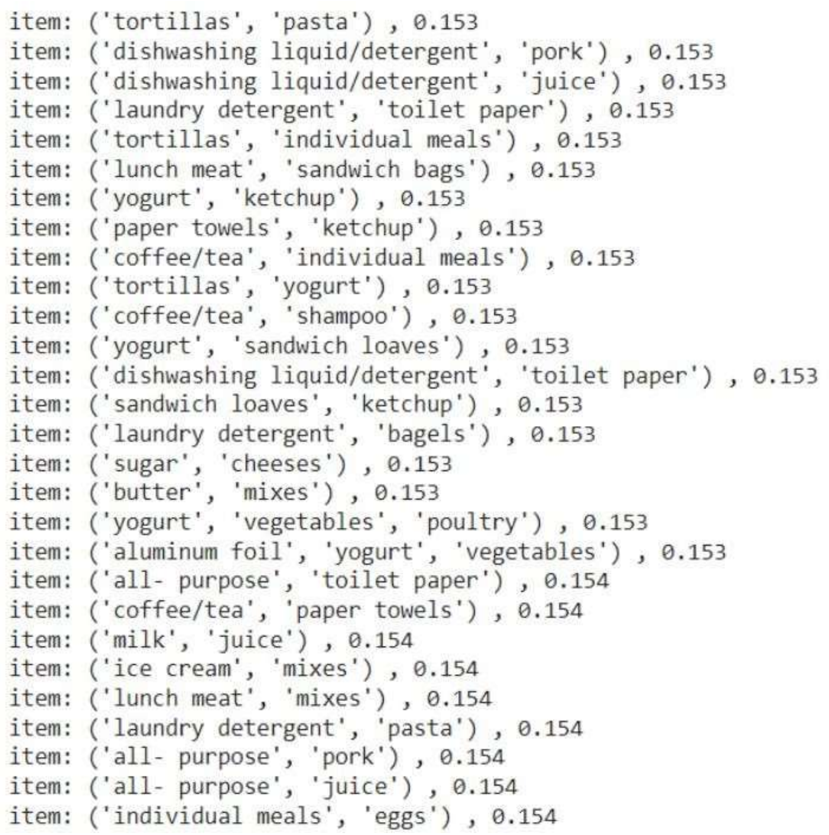
▾ Task 2: Run apriory.py and Evaluate Results

```
#Min items in trans
count_trans = Load_trans_data_main.groupby("Id").count()["item"]
print(count_trans.min())
```

```
5
```

```
print ('# # # # # # # # #  Code for Task 2, Case:1 # # # # # # # # #')
#import the apriori.py file
import apriori
#load the csv file
data1=apriori.dataFromFile('dataset.csv')
#generate the rules and items
data_list,rules=apriori.runApriori(data1,0.15,0.60)
print ('Case 1 (minimum support=0.15 and minimum confidence=0.60)')
print ('Case 1 Reasoning:\n\nSince the min number of items in one transaction is found to be 5, we shall take the minimum support value to be less than 5\nSo first we try to choose 1
print ('Case 1 Output:')
#display the itemsets and rules
apriori.printResults(data_list,rules)
```

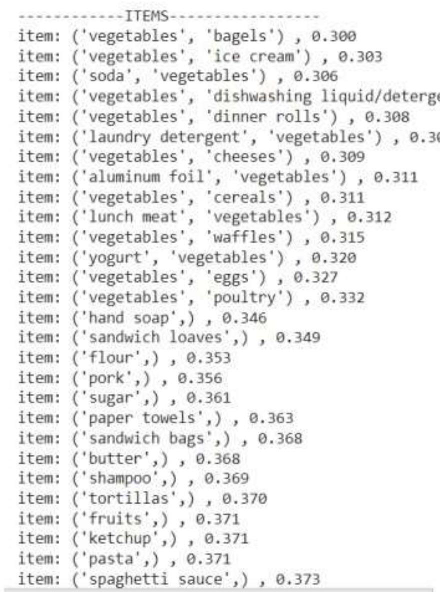
```
item: ('all- purpose', 'milk') , 0.152
item: ('laundry detergent', 'individual meals') , 0.152
item: ('yogurt', 'pork') , 0.152
item: ('butter', 'bagels') , 0.152
item: ('individual meals', 'shampoo') , 0.152
item: ('waffles', 'butter') , 0.152
```



```
item: ('tortillas', 'pasta') , 0.153
item: ('dishwashing liquid/detergent', 'pork') , 0.153
item: ('dishwashing liquid/detergent', 'juice') , 0.153
item: ('laundry detergent', 'toilet paper') , 0.153
item: ('tortillas', 'individual meals') , 0.153
item: ('lunch meat', 'sandwich bags') , 0.153
item: ('yogurt', 'ketchup') , 0.153
item: ('paper towels', 'ketchup') , 0.153
item: ('coffee/tea', 'individual meals') , 0.153
item: ('tortillas', 'yogurt') , 0.153
item: ('coffee/tea', 'shampoo') , 0.153
item: ('yogurt', 'sandwich loaves') , 0.153
item: ('dishwashing liquid/detergent', 'toilet paper') , 0.153
item: ('sandwich loaves', 'ketchup') , 0.153
item: ('laundry detergent', 'bagels') , 0.153
item: ('sugar', 'cheeses') , 0.153
item: ('butter', 'mixes') , 0.153
item: ('yogurt', 'vegetables', 'poultry') , 0.153
item: ('aluminum foil', 'yogurt', 'vegetables') , 0.153
item: ('all- purpose', 'toilet paper') , 0.154
item: ('coffee/tea', 'paper towels') , 0.154
item: ('milk', 'juice') , 0.154
item: ('ice cream', 'mixes') , 0.154
item: ('lunch meat', 'mixes') , 0.154
item: ('laundry detergent', 'pasta') , 0.154
item: ('all- purpose', 'pork') , 0.154
item: ('all- purpose', 'juice') , 0.154
item: ('individual meals', 'eggs') , 0.154
```

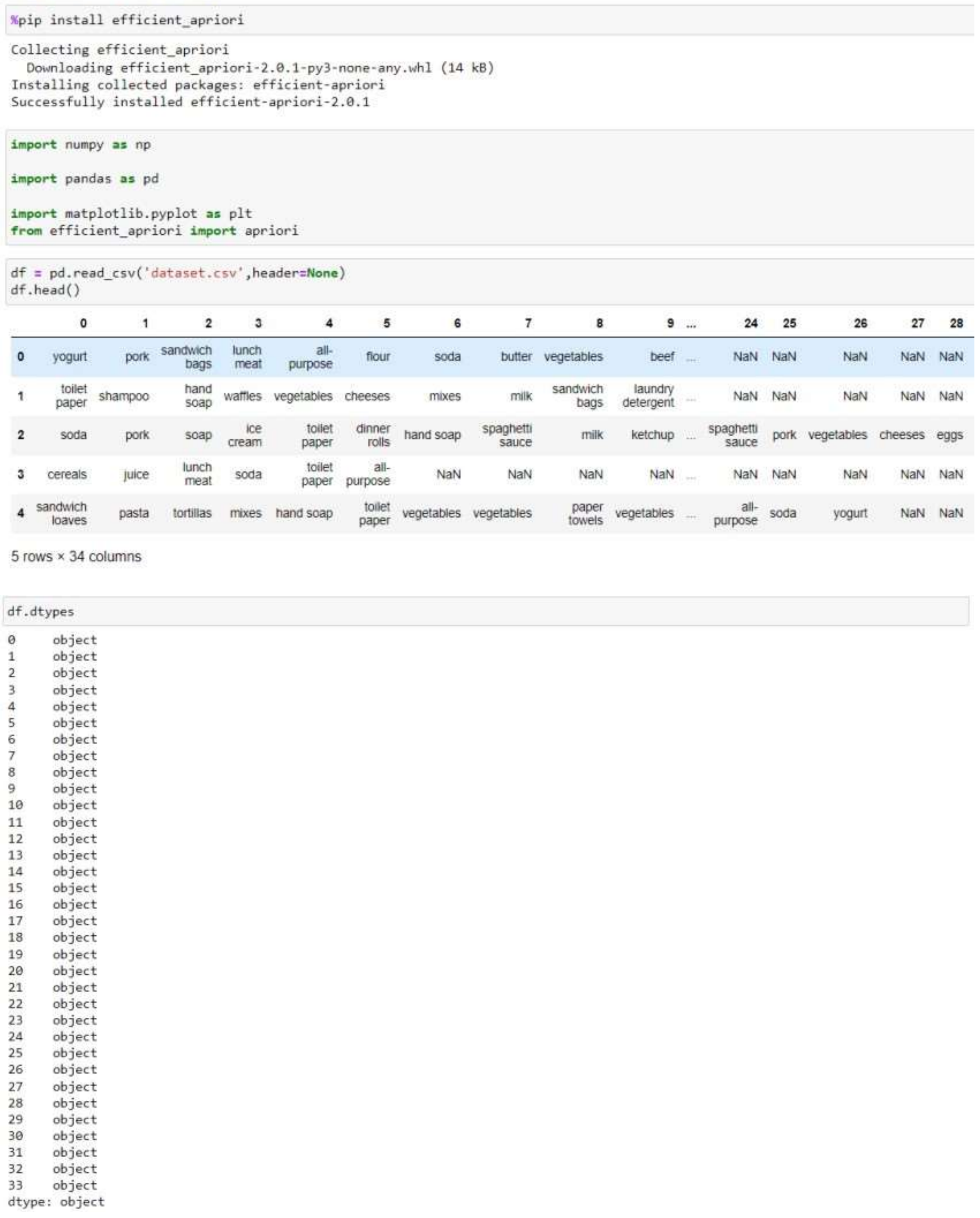
As we are choosing average or meadian values for both minimum support and confidence we would find most of the transactions to be having

Case 2 Output:



```
------------ITEMS-----------------
item: ('vegetables', 'bagels') , 0.300
item: ('vegetables', 'ice cream') , 0.303
item: ('soda', 'vegetables') , 0.306
item: ('vegetables', 'dishwashing liquid/detergent') , 0.306
item: ('vegetables', 'dinner rolls') , 0.308
item: ('laundry detergent', 'vegetables') , 0.309
item: ('vegetables', 'cheeses') , 0.309
item: ('aluminum foil', 'vegetables') , 0.311
item: ('vegetables', 'cereals') , 0.311
item: ('lunch meat', 'vegetables') , 0.312
item: ('vegetables', 'waffles') , 0.315
item: ('yogurt', 'vegetables') , 0.320
item: ('vegetables', 'eggs') , 0.327
item: ('vegetables', 'poultry') , 0.332
item: ('hand soap',) , 0.346
item: ('sandwich loaves',) , 0.349
item: ('flour',) , 0.353
item: ('pork',) , 0.356
item: ('sugar',) , 0.361
item: ('paper towels',) , 0.363
item: ('sandwich bags',) , 0.368
item: ('butter',) , 0.368
item: ('shampoo',) , 0.369
item: ('tortillas',) , 0.370
item: ('fruits',) , 0.371
item: ('ketchup',) , 0.371
item: ('pasta',) , 0.371
item: ('spaghetti sauce',) , 0.373
```

CODE and OUTPUT:



```
%pip install efficient_apriori
```

```
Collecting efficient_apriori
  Downloading efficient_apriori-2.0.1-py3-none-any.whl (14 kB)
Installing collected packages: efficient-apriori
Successfully installed efficient-apriori-2.0.1
```

```
import numpy as np

import pandas as pd

import matplotlib.pyplot as plt
from efficient_apriori import apriori
```

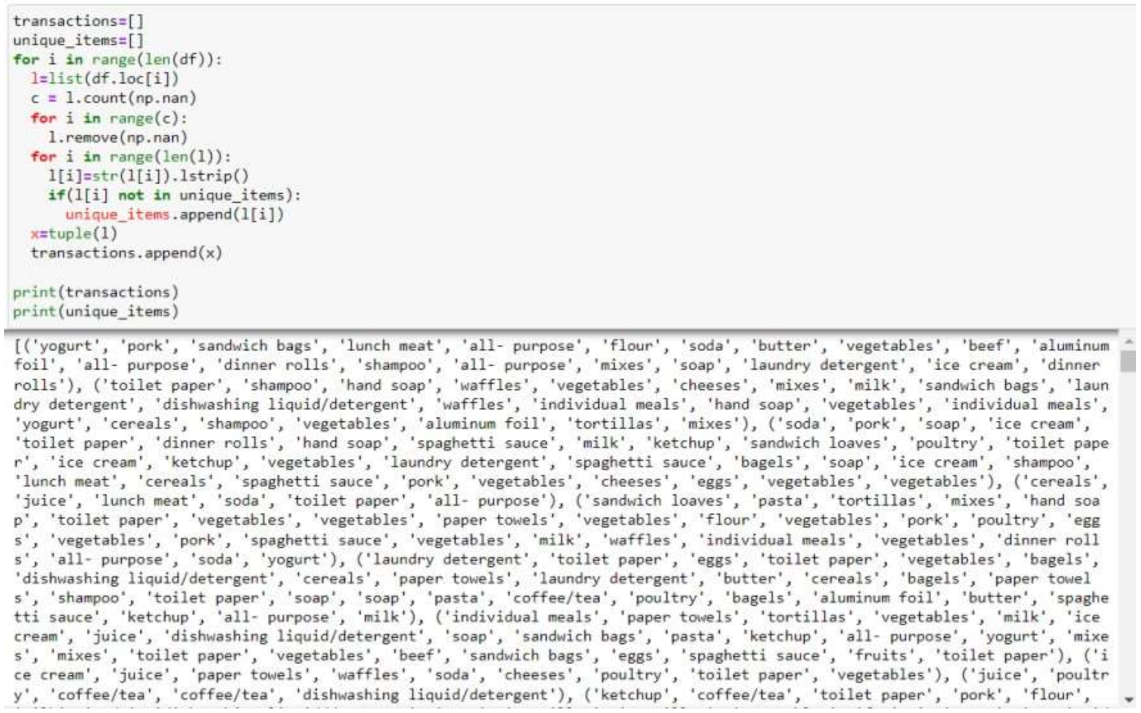
```
df = pd.read_csv('dataset.csv',header=None)
df.head()
```

	0	1	2	3	4	5	6	7	8	9	...	24	25	26	27	28
0	yogurt	pork	sandwich bags	lunch meat	all-purpose	flour	soda	butter	vegetables	beef	...	NaN	NaN	NaN	NaN	NaN
1	toilet paper	shampoo	hand soap	waffles	vegetables	cheeses	mixes	milk	sandwich bags	laundry detergent	...	NaN	NaN	NaN	NaN	NaN
2	soda	pork	soap	ice cream	toilet paper	dinner rolls	hand soap	spaghetti sauce	milk	ketchup	...	spaghetti sauce	pork	vegetables	cheeses	eggs
3	cereals	juice	lunch meat	soda	toilet paper	all-purpose	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN
4	sandwich loaves	pasta	tortillas	mixes	hand soap	toilet paper	vegetables	vegetables	paper towels	vegetables	...	all-purpose	soda	yogurt	NaN	NaN

5 rows × 34 columns

```
df.dtypes
```

```
0     object
1     object
2     object
3     object
4     object
5     object
6     object
7     object
8     object
9     object
10    object
11    object
12    object
13    object
14    object
15    object
16    object
17    object
18    object
19    object
20    object
21    object
22    object
23    object
24    object
25    object
26    object
27    object
28    object
29    object
30    object
31    object
32    object
33    object
dtype: object
```

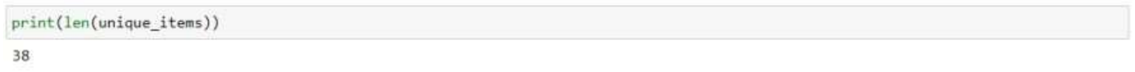


```
transactions=[]
unique_items=[]
for i in range(len(df)):
  l=list(df.loc[i])
  c = l.count(np.nan)
  for i in range(c):
    l.remove(np.nan)
  for i in range(len(l)):
    l[i]=str(l[i]).lstrip()
    if(l[i] not in unique_items):
      unique_items.append(l[i])
  x=tuple(l)
  transactions.append(x)

print(transactions)
print(unique_items)
```

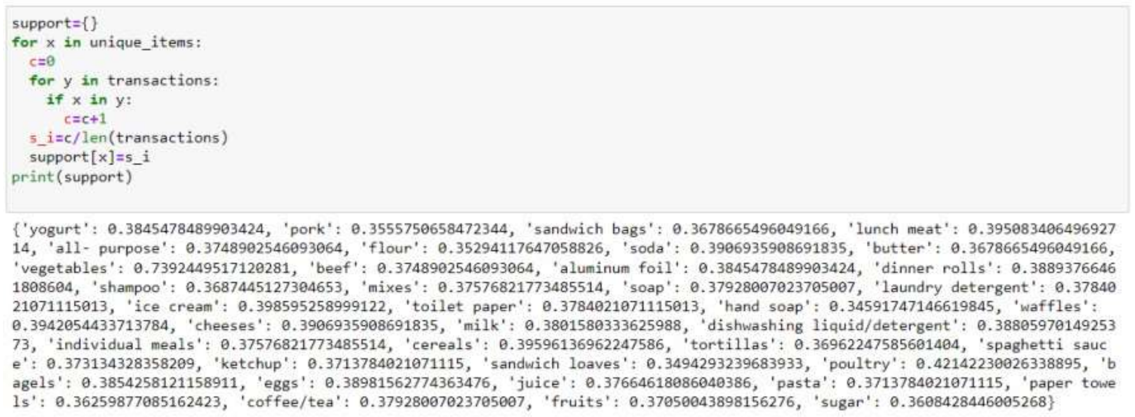
```
[('yogurt', 'pork', 'sandwich bags', 'lunch meat', 'all- purpose', 'flour', 'soda', 'butter', 'vegetables', 'beef', 'aluminum
foil', 'all- purpose', 'dinner rolls', 'shampoo', 'all- purpose', 'mixes', 'soap', 'laundry detergent', 'ice cream', 'dinner
rolls'), ('toilet paper', 'shampoo', 'hand soap', 'waffles', 'vegetables', 'cheeses', 'mixes', 'milk', 'sandwich bags', 'laun
dry detergent', 'dishwashing liquid/detergent', 'waffles', 'individual meals', 'hand soap', 'vegetables', 'individual meals',
'yogurt', 'cereals', 'shampoo', 'vegetables', 'aluminum foil', 'tortillas', 'mixes'), ('soda', 'pork', 'soap', 'ice cream',
'toilet paper', 'dinner rolls', 'hand soap', 'spaghetti sauce', 'milk', 'ketchup', 'sandwich loaves', 'poultry', 'toilet pape
r', 'ice cream', 'ketchup', 'vegetables', 'laundry detergent', 'spaghetti sauce', 'bagels', 'soap', 'ice cream', 'shampoo',
'lunch meat', 'cereals', 'spaghetti sauce', 'pork', 'vegetables', 'cheeses', 'eggs', 'vegetables', 'vegetables'), ('cereals',
'juice', 'lunch meat', 'soda', 'toilet paper', 'all- purpose'), ('sandwich loaves', 'pasta', 'tortillas', 'mixes', 'hand soa
p', 'toilet paper', 'vegetables', 'vegetables', 'paper towels', 'vegetables', 'flour', 'vegetables', 'pork', 'poultry', 'egg
s', 'vegetables', 'pork', 'spaghetti sauce', 'vegetables', 'milk', 'waffles', 'individual meals', 'vegetables', 'dinner roll
s', 'all- purpose', 'soda', 'yogurt'), ('laundry detergent', 'toilet paper', 'eggs', 'toilet paper', 'vegetables', 'bagels',
'dishwashing liquid/detergent', 'cereals', 'paper towels', 'laundry detergent', 'butter', 'cereals', 'bagels', 'paper towel
s', 'shampoo', 'toilet paper', 'soap', 'soap', 'pasta', 'coffee/tea', 'poultry', 'bagels', 'aluminum foil', 'butter', 'spaghe
tti sauce', 'ketchup', 'all- purpose', 'milk'), ('individual meals', 'paper towels', 'tortillas', 'vegetables', 'milk', 'ice
cream', 'juice', 'dishwashing liquid/detergent', 'soap', 'sandwich bags', 'pasta', 'ketchup', 'all- purpose', 'yogurt', 'mixe
s', 'mixes', 'toilet paper', 'vegetables', 'beef', 'sandwich bags', 'eggs', 'spaghetti sauce', 'fruits', 'toilet paper'), ('i
ce cream', 'juice', 'paper towels', 'waffles', 'soda', 'cheeses', 'poultry', 'toilet paper', 'vegetables'), ('juice', 'poultr
y', 'coffee/tea', 'coffee/tea', 'dishwashing liquid/detergent'), ('ketchup', 'coffee/tea', 'toilet paper', 'pork', 'flour',
```

Task 1 Identify the frequency of most popular 50 items



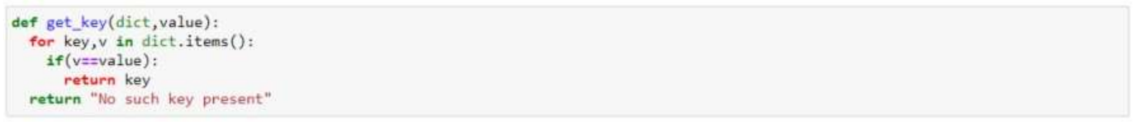
```
print(len(unique_items))
```

```
38
```



```
support={}
for x in unique_items:
  c=0
  for y in transactions:
    if x in y:
      c=c+1
  s_i=c/len(transactions)
  support[x]=s_i
print(support)
```

```
{'yogurt': 0.3845478489903424, 'pork': 0.3555750658472344, 'sandwich bags': 0.3678665496049166, 'lunch meat': 0.395083406496927
14, 'all- purpose': 0.3748902546093064, 'flour': 0.35294117647058826, 'soda': 0.3906935908691835, 'butter': 0.3678665496049166,
'vegetables': 0.7392449517120281, 'beef': 0.3748902546093064, 'aluminum foil': 0.3845478489903424, 'dinner rolls': 0.3889376646
1808604, 'shampoo': 0.3687445127304653, 'mixes': 0.37576821773485514, 'soap': 0.37928007023705007, 'laundry detergent': 0.37840
21071115013, 'ice cream': 0.398595258999122, 'toilet paper': 0.3784021071115013, 'hand soap': 0.34591747146619845, 'waffles':
0.3942054433713784, 'cheeses': 0.3906935908691835, 'milk': 0.3801580333625988, 'dishwashing liquid/detergent': 0.38805970149253
73, 'individual meals': 0.37576821773485514, 'cereals': 0.39596136962247586, 'tortillas': 0.36962247585601404, 'spaghetti sauc
e': 0.373134328358209, 'ketchup': 0.3713784021071115, 'sandwich loaves': 0.3494293239683933, 'poultry': 0.42142230026338895, 'b
agels': 0.3854258121158911, 'eggs': 0.38981562774363476, 'juice': 0.37664618086040386, 'pasta': 0.3713784021071115, 'paper towe
ls': 0.36259877085162423, 'coffee/tea': 0.37928007023705007, 'fruits': 0.37050043898156276, 'sugar': 0.3608428446005268}
```



```
def get_key(dict,value):
  for key,v in dict.items():
    if(v==value):
      return key
  return "No such key present"
```

```python
l=list(support.values())
l.sort(reverse=True)

x=[]
for h in l:
  x.append(get_key(support,h))

print(x)
```

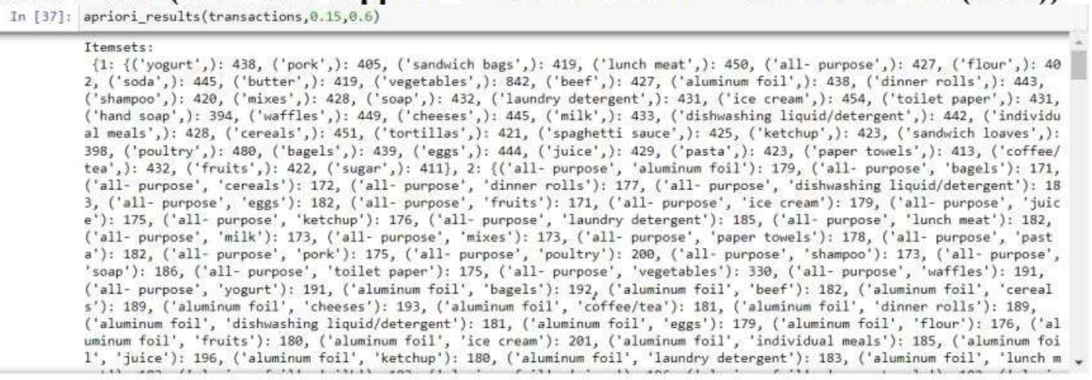
```
['vegetables', 'poultry', 'ice cream', 'cereals', 'lunch meat', 'waffles', 'soda', 'soda', 'eggs', 'dinner rolls', 'dishwashing
liquid/detergent', 'bagels', 'yogurt', 'yogurt', 'milk', 'soap', 'soap', 'laundry detergent', 'laundry detergent', 'juice', 'mi
xes', 'mixes', 'all- purpose', 'all- purpose', 'spaghetti sauce', 'ketchup', 'ketchup', 'fruits', 'tortillas', 'shampoo', 'sand
wich bags', 'sandwich bags', 'paper towels', 'sugar', 'pork', 'flour', 'sandwich loaves', 'hand soap']
```

```python
print("Frequency of most popular items in transactions")
for j in x:
  print(j,"=",support[j]*len(transactions))
```

```
Frequency of most popular items in transactions
vegetables = 842.0
poultry = 480.0
ice cream = 454.0
cereals = 451.0
lunch meat = 450.0
waffles = 449.0
soda = 445.0

soda = 445.0
eggs = 444.0
dinner rolls = 443.0
dishwashing liquid/detergent = 442.0
bagels = 439.0
yogurt = 438.0
yogurt = 438.0
milk = 433.0
soap = 432.00000000000006
soap = 432.00000000000006
laundry detergent = 431.0
laundry detergent = 431.0
juice = 429.0
mixes = 428.0
mixes = 428.0
all- purpose = 427.0
all- purpose = 427.0
spaghetti sauce = 425.0
ketchup = 423.0
ketchup = 423.0
fruits = 422.0
tortillas = 421.0
shampoo = 420.0
sandwich bags = 419.0
sandwich bags = 419.0
paper towels = 413.0
sugar = 411.0
pork = 405.0
flour = 402.0
sandwich loaves = 398.0
hand soap = 394.00000000000006
```

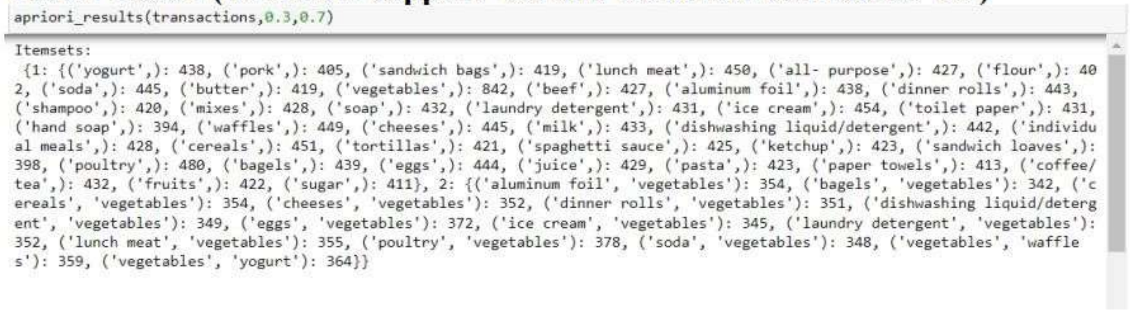
Task 2 Case1 (minimum support=0.15 and minimum confidence=0.6 (60%))



```
In [37]: apriori_results(transactions,0.15,0.6)
```

```
Itemsets:
 {1: {('yogurt',): 438, ('pork',): 405, ('sandwich bags',): 419, ('lunch meat',): 450, ('all- purpose',): 427, ('flour',): 40
2, ('soda',): 445, ('butter',): 419, ('vegetables',): 842, ('beef',): 427, ('aluminum foil',): 438, ('dinner rolls',): 443,
('shampoo',): 420, ('mixes',): 428, ('soap',): 432, ('laundry detergent',): 431, ('ice cream',): 454, ('toilet paper',): 431,
('hand soap',): 394, ('waffles',): 449, ('cheeses',): 445, ('milk',): 433, ('dishwashing liquid/detergent',): 442, ('individu
al meals',): 428, ('cereals',): 451, ('tortillas',): 421, ('spaghetti sauce',): 425, ('ketchup',): 423, ('sandwich loaves',):
398, ('poultry',): 480, ('bagels',): 439, ('eggs',): 444, ('juice',): 429, ('pasta',): 423, ('paper towels',): 413, ('coffee/
tea',): 432, ('fruits',): 422, ('sugar',): 411}, 2: {('all- purpose', 'aluminum foil'): 179, ('all- purpose', 'bagels'): 171,
('all- purpose', 'cereals'): 172, ('all- purpose', 'dinner rolls'): 177, ('all- purpose', 'dishwashing liquid/detergent'): 18
3, ('all- purpose', 'eggs'): 182, ('all- purpose', 'fruits'): 171, ('all- purpose', 'ice cream'): 179, ('all- purpose', 'juic
e'): 175, ('all- purpose', 'ketchup'): 176, ('all- purpose', 'laundry detergent'): 185, ('all- purpose', 'lunch meat'): 182,
('all- purpose', 'milk'): 173, ('all- purpose', 'mixes'): 173, ('all- purpose', 'paper towels'): 178, ('all- purpose', 'past
a'): 182, ('all- purpose', 'pork'): 175, ('all- purpose', 'poultry'): 200, ('all- purpose', 'shampoo'): 173, ('all- purpose',
'soap'): 186, ('all- purpose', 'toilet paper'): 175, ('all- purpose', 'vegetables'): 330, ('all- purpose', 'waffles'): 191,
('all- purpose', 'yogurt'): 191, ('aluminum foil', 'bagels'): 192, ('aluminum foil', 'beef'): 182, ('aluminum foil', 'cereal
s'): 189, ('aluminum foil', 'cheeses'): 193, ('aluminum foil', 'coffee/tea'): 181, ('aluminum foil', 'dinner rolls'): 189,
('aluminum foil', 'dishwashing liquid/detergent'): 181, ('aluminum foil', 'eggs'): 179, ('aluminum foil', 'flour'): 176, ('al
uminum foil', 'fruits'): 180, ('aluminum foil', 'ice cream'): 201, ('aluminum foil', 'individual meals'): 185, ('aluminum foi
l', 'juice'): 196, ('aluminum foil', 'ketchup'): 180, ('aluminum foil', 'laundry detergent'): 183, ('aluminum foil', 'lunch m
```

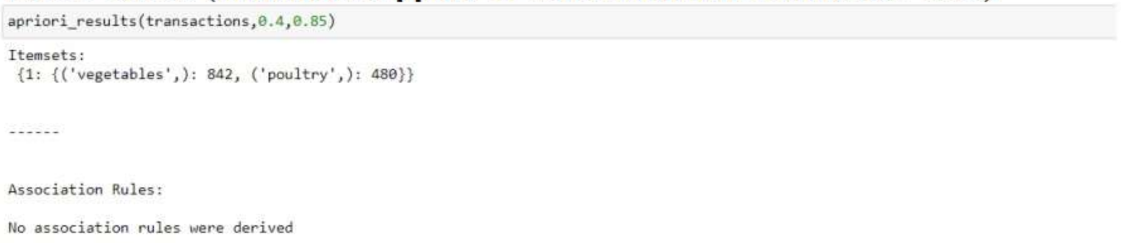
Task 2 Case 2 (minimum support=0.3 and minimum confidence=0.7)



```
apriori_results(transactions,0.3,0.7)
```

```
Itemsets:
 {1: {('yogurt',): 438, ('pork',): 405, ('sandwich bags',): 419, ('lunch meat',): 450, ('all- purpose',): 427, ('flour',): 40
2, ('soda',): 445, ('butter',): 419, ('vegetables',): 842, ('beef',): 427, ('aluminum foil',): 438, ('dinner rolls',): 443,
('shampoo',): 420, ('mixes',): 428, ('soap',): 432, ('laundry detergent',): 431, ('ice cream',): 454, ('toilet paper',): 431,
('hand soap',): 394, ('waffles',): 449, ('cheeses',): 445, ('milk',): 433, ('dishwashing liquid/detergent',): 442, ('individu
al meals',): 428, ('cereals',): 451, ('tortillas',): 421, ('spaghetti sauce',): 425, ('ketchup',): 423, ('sandwich loaves',):
398, ('poultry',): 480, ('bagels',): 439, ('eggs',): 444, ('juice',): 429, ('pasta',): 423, ('paper towels',): 413, ('coffee/
tea',): 432, ('fruits',): 422, ('sugar',): 411}, 2: {('aluminum foil', 'vegetables'): 354, ('bagels', 'vegetables'): 342, ('c
ereals', 'vegetables'): 354, ('cheeses', 'vegetables'): 352, ('dinner rolls', 'vegetables'): 351, ('dishwashing liquid/deterg
ent', 'vegetables'): 349, ('eggs', 'vegetables'): 372, ('ice cream', 'vegetables'): 345, ('laundry detergent', 'vegetables'):
352, ('lunch meat', 'vegetables'): 355, ('poultry', 'vegetables'): 378, ('soda', 'vegetables'): 348, ('vegetables', 'waffle
s'): 359, ('vegetables', 'yogurt'): 364}}
```

Task 3 Case 3 (minimum support=0.4 and minimum confidence=0.85)



```
apriori_results(transactions,0.4,0.85)
```

```
Itemsets:
 {1: {('vegetables',): 842, ('poultry',): 480}}

------

Association Rules:

No association rules were derived
```