

```
#External package need to install
!pip install apyori

Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting apyori
  Downloading apyori-1.1.2.tar.gz (8.6 kB)
  Preparing metadata (setup.py) ... done
Building wheels for collected packages: apyori
  Building wheel for apyori (setup.py) ... done
  Created wheel for apyori: filename=apyori-1.1.2-py3-none-any.whl size=5976 sha256=ba346218c428423239e5dba480160eef8ee3d64e634360c
  Stored in directory: /root/.cache/pip/wheels/32/2a/54/10c595515f385f3726642b10c60bf788029e8f3a1323e3913a
Successfully built apyori
Installing collected packages: apyori
Successfully installed apyori-1.1.2
```

```
#import all required packages..
import pandas as pd
import numpy as np
from apyori import apriori

#loading market basket dataset..

df = pd.read_csv('Market_Basket_Optimisation.csv',header=None)

df.head()
```

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	shrimp	almonds	avocado	vegetables mix	green grapes	whole weat flour	yams	cottage cheese	energy drink	tomato juice	low fat yogurt	green tea	honey	salad	mineral water	salmon
1	burgers	meatballs	eggs	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	chutney	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	turkey	avocado	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	mineral water	milk	energy bar	whole wheat rice	green tea	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN



```
#replacing empty value with 0.
df.fillna(0,inplace=True)
```

```
df.head()
```

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	shrimp	almonds	avocado	vegetables mix	green grapes	whole weat flour	yams	cottage cheese	energy drink	tomato juice	low fat yogurt	green tea	honey	salad	mineral water	salmon
1	burgers	meatballs	eggs	0	0	0	0	0	0	0	0	0	0	0	0	0
2	chutney	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	turkey	avocado	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	mineral water	milk	energy bar	whole wheat rice	green tea	0	0	0	0	0	0	0	0	0	0	0



```
#for using aprori need to convert data in list format..
# transaction = [['apple','almonds'],['apple'],['banana','apple']]...
transactions = []
for i in range(0,len(df)):
    transactions.append([str(df.values[i,j]) for j in range(0,20) if str(df.values[i,j])!='0'])

transactions[0]

['shrimp',
 'almonds',
 'avocado',
```

```
'vegetables mix',  
'green grapes',  
'whole weat flour',  
'yams',  
'cottage cheese',  
'energy drink',  
'tomato juice',  
'low fat yogurt',  
'green tea',  
'honey',  
'salad',  
'mineral water',  
'salmon',  
'antioxydant juice',  
'frozen smoothie',  
'spinach',  
'olive oil']
```

```
#Call apriori function which requires minimum support, confidance and lift, min length is combination of item default is 2".  
rules = apriori(transactions,min_support=0.003,min_confidance=0.2,min_lift=3,min_length=2)
```

```
#it generates a set of rules in a generator file...  
rules
```

```
<generator object apriori at 0x7f214dea9350>
```

```
# all rules need to be converted in a list..  
Results = list(rules)  
Results
```

```
RelationRecord(items=frozenset({'spaghetti', 'chocolate', 'olive oil', 'mineral water'}), support=0.0038661511798426876,
ordered_statistics=[OrderedStatistic(items_base=frozenset({'olive oil'}), items_add=frozenset({'chocolate', 'spaghetti',
'mineral water'}), confidence=0.058704453441295545, lift=3.700353825740822),
OrderedStatistic(items_base=frozenset({'chocolate', 'mineral water'}), items_add=frozenset({'olive oil', 'spaghetti'}),
confidence=0.07341772151898734, lift=3.201780983220489), OrderedStatistic(items_base=frozenset({'chocolate', 'olive oil'}),
items_add=frozenset({'spaghetti', 'mineral water'}), confidence=0.23577235772357724, lift=3.947608159117306),
OrderedStatistic(items_base=frozenset({'chocolate', 'spaghetti'}), items_add=frozenset({'olive oil', 'mineral water'}),
confidence=0.09863945578231292, lift=3.5743698445561796), OrderedStatistic(items_base=frozenset({'olive oil', 'mineral
water'}), items_add=frozenset({'chocolate', 'spaghetti'}), confidence=0.1400966183574879, lift=3.5743698445561796),
OrderedStatistic(items_base=frozenset({'spaghetti', 'mineral water'}), items_add=frozenset({'chocolate', 'olive oil'}),
confidence=0.06473214285714286, lift=3.947608159117306), OrderedStatistic(items_base=frozenset({'olive oil', 'spaghetti'}),
items_add=frozenset({'chocolate', 'mineral water'}), confidence=0.1686046511627907, lift=3.2017809832204884),
OrderedStatistic(items_base=frozenset({'spaghetti', 'chocolate', 'mineral water'}), items_add=frozenset({'olive oil'}),
confidence=0.2436974789915966, lift=3.700353825740822)],
RelationRecord(items=frozenset({'spaghetti', 'chocolate', 'mineral water', 'pancakes'}), support=0.0037328356219170776,
ordered_statistics=[OrderedStatistic(items_base=frozenset({'chocolate', 'pancakes'}), items_add=frozenset({'spaghetti',
'mineral water'}), confidence=0.1879194630872483, lift=3.1463926174496644), OrderedStatistic(items_base=frozenset({'spaghetti',
'mineral water'}), items_add=frozenset({'chocolate', 'pancakes'}), confidence=0.0625, lift=3.1463926174496644)],
RelationRecord(items=frozenset({'shrimp', 'chocolate', 'mineral water', 'spaghetti'}), support=0.0034662045060658577,
ordered_statistics=[OrderedStatistic(items_base=frozenset({'shrimp'}), items_add=frozenset({'spaghetti', 'chocolate', 'mineral
water'}), confidence=0.04850746268656716, lift=3.0576006522011783), OrderedStatistic(items_base=frozenset({'chocolate',
'mineral water'}), items_add=frozenset({'shrimp', 'spaghetti'}), confidence=0.06582278481012657, lift=3.10526231987899),
OrderedStatistic(items_base=frozenset({'shrimp', 'chocolate'}), items_add=frozenset({'spaghetti', 'mineral water'}),
confidence=0.1925925925925926, lift=3.2246362433862434), OrderedStatistic(items_base=frozenset({'chocolate', 'spaghetti'}),
items_add=frozenset({'shrimp', 'mineral water'}), confidence=0.08843537414965985, lift=3.747761251393212),
OrderedStatistic(items_base=frozenset({'shrimp', 'mineral water'}), items_add=frozenset({'chocolate', 'spaghetti'}),
confidence=0.14689265536723162, lift=3.747761251393212), OrderedStatistic(items_base=frozenset({'spaghetti', 'mineral water'}),
items_add=frozenset({'shrimp', 'chocolate'}), confidence=0.05803571428571428, lift=3.2246362433862434),
OrderedStatistic(items_base=frozenset({'shrimp', 'spaghetti'}), items_add=frozenset({'chocolate', 'mineral water'}),
confidence=0.16352201257861634, lift=3.10526231987899), OrderedStatistic(items_base=frozenset({'spaghetti', 'chocolate',
'mineral water'}), items_add=frozenset({'shrimp'}), confidence=0.21848739495798317, lift=3.0576006522011783)],
RelationRecord(items=frozenset({'milk', 'eggs', 'mineral water', 'frozen vegetables'}), support=0.0037328356219170776,
ordered_statistics=[OrderedStatistic(items_base=frozenset({'milk'}), items_add=frozenset({'eggs', 'mineral water', 'frozen
vegetables'}), confidence=0.02880658436213992, lift=3.177620430888405), OrderedStatistic(items_base=frozenset({'eggs', 'frozen
vegetables'}), items_add=frozenset({'milk', 'mineral water'}), confidence=0.017177914110429446, lift=3.5792092706203134),
OrderedStatistic(items_base=frozenset({'milk', 'eggs'}), items_add=frozenset({'mineral water', 'frozen vegetables'}),
confidence=0.12121212121212122, lift=3.392582541836273), OrderedStatistic(items_base=frozenset({'eggs', 'mineral water'}),
items_add=frozenset({'milk', 'frozen vegetables'}), confidence=0.07329842931937172, lift=3.106279764545804),
OrderedStatistic(items_base=frozenset({'milk', 'frozen vegetables'}), items_add=frozenset({'eggs', 'mineral water'}),
confidence=0.1581920903954802, lift=3.106279764545804), OrderedStatistic(items_base=frozenset({'mineral water', 'frozen
vegetables'}), items_add=frozenset({'milk', 'eggs'}), confidence=0.10447761194029849, lift=3.3925825418362727),
OrderedStatistic(items_base=frozenset({'milk', 'mineral water'}), items_add=frozenset({'eggs', 'frozen vegetables'}),
confidence=0.07777777777777778, lift=3.5792092706203134), OrderedStatistic(items_base=frozenset({'eggs', 'mineral water',
'frozen vegetables'}), items_add=frozenset({'milk'}), confidence=0.411764705882353, lift=3.177620430888405)],
RelationRecord(items=frozenset({'milk', 'eggs', 'mineral water', 'ground beef'}), support=0.003332888948140248,
ordered_statistics=[OrderedStatistic(items_base=frozenset({'eggs', 'ground beef'}), items_add=frozenset({'milk', 'mineral
water'}), confidence=0.16666666666666669, lift=3.4726851851851857), OrderedStatistic(items_base=frozenset({'milk', 'mineral
water'}), items_add=frozenset({'eggs', 'ground beef'}), confidence=0.06944444444444445, lift=3.4726851851851857)],
RelationRecord(items=frozenset({'spaghetti', 'eggs', 'mineral water', 'ground beef'}), support=0.0038661511798426876,
ordered_statistics=[OrderedStatistic(items_base=frozenset({'eggs', 'ground beef'}), items_add=frozenset({'spaghetti', 'mineral
water'}), confidence=0.19333333333333336, lift=3.237038690476191), OrderedStatistic(items_base=frozenset({'spaghetti', 'mineral
water'}), items_add=frozenset({'eggs', 'ground beef'}), confidence=0.06473214285714286, lift=3.237038690476191)],
RelationRecord(items=frozenset({'milk', 'spaghetti', 'frozen smoothie', 'mineral water'}), support=0.003199573390214638,
ordered_statistics=[OrderedStatistic(items_base=frozenset({'frozen smoothie'}), items_add=frozenset({'milk', 'spaghetti',
'mineral water'}), confidence=0.05052631578947369, lift=3.2118465655664585), OrderedStatistic(items_base=frozenset({'milk'}),
items_add=frozenset({'frozen smoothie', 'spaghetti', 'mineral water'}), confidence=0.02469135802469136,
lift=3.6315662067296057), OrderedStatistic(items_base=frozenset({'milk', 'frozen smoothie'}), items_add=frozenset({'spaghetti',
'mineral water'}), confidence=0.22429906542056074, lift=3.7555073431241657), OrderedStatistic(items_base=frozenset({'frozen
smoothie', 'mineral water'}), items_add=frozenset({'milk', 'spaghetti'}), confidence=0.15789473684210528).
```

```
#convert result in a dataframe for further operation...
df_results = pd.DataFrame(Results)

# as we see order statistics itself a list so need to be converted in proper format..
df_results.head()
```

	items	support	ordered_statistics
0	(cottage cheese, brownies)	0.003466	[[(brownies), (cottage cheese), 0.102766798418...
1	(chicken, light cream)	0.004533	[[(chicken), (light cream), 0.075555555555555...
2	(escalope, mushroom cream sauce)	0.005733	[[(escalope), (mushroom cream sauce), 0.072268...
3	(escalope, pasta)	0.005866	[[(escalope), (pasta), 0.07394957983193277, 4....
4	(tomato juice, fresh bread)	0.004266	[[(fresh bread), (tomato juice), 0.09907120743...

```
#keep support in a separate data frame so we can use later..
support = df_results.support

'''
convert orderstatistic in a proper format.
order statistic has lhs => rhs as well rhs => lhs we can choose any one for convience i choose first one which is 'df_results['ordered_st
'''

#all four empty list which will contain lhs, rhs, confidance and lift respectively.

first_values = []
second_values = []
third_values = []
fourth_value = []

# loop number of rows time and append 1 by 1 value in a separate list.. first and second element was frozenset which need to be convertet
for i in range(df_results.shape[0]):
    single_list = df_results['ordered_statistics'][i][0]
    first_values.append(list(single_list[0]))
    second_values.append(list(single_list[1]))
    third_values.append(single_list[2])
    fourth_value.append(single_list[3])

#convert all four list into dataframe for further operation..
lhs = pd.DataFrame(first_values)
rhs= pd.DataFrame(second_values)
confidance=pd.DataFrame(third_values,columns=['Confidance'])
lift=pd.DataFrame(fourth_value,columns=['lift'])

#concat all list together in a single dataframe
df_final = pd.concat([lhs,rhs,support,confidance,lift], axis=1)
df_final
```

	0	1	0	1	2	support	Confidance	lift
0	brownies	None	cottage cheese	None	None	0.003466	0.102767	3.225330
1	chicken	None	light cream	None	None	0.004533	0.075556	4.843951
2	escalope	None	mushroom cream sauce	None	None	0.005733	0.072269	3.790833
3	escalope	None	pasta	None	None	0.005866	0.073950	4.700812
4	fresh bread	None	tomato juice	None	None	0.004266	0.099071	3.259356
...
89	ground beef	pancakes	spaghetti	mineral water	None	0.003066	0.211009	3.532991
90	ground beef	None	tomatoes	spaghetti	mineral water	0.003066	0.031208	3.344117
91	olive oil	None	milk	spaghetti	mineral water	0.003333	0.050607	3.216994
92	milk	mineral water	shrimp	spaghetti	None	0.003066	0.063889	3.014029
93	tomatoes	None	milk	spaghetti	mineral water	0.003333	0.048733	3.097846

94 rows x 8 columns


```
'''
we have some of place only 1 item in lhs and some place 3 or more so we need to a proper represenation for User to understand.
removing none with ' ' extra so when we combine three column in 1 then only 1 item will be there with spaces which is proper rather thar
example : coffee,none,none which converted to coffee, ,
'''
df_final.fillna(value=' ', inplace=True)

#set column name
df_final.columns = ['lhs',0,1,2,'rhs','support','confidance','lift']

#add all three column because those where the lhs itemset only
df_final['lhs'] = df_final['lhs']+str(", ")+df_final[1]+str(", ")+df_final[2]

#drop those 1,2 column because now we already appended to lhs column..
df_final.drop(columns=[1,2],inplace=True)

#this is final output.. you can sort based on the support lift and confidance..
df_final.head()
```

	lhs	0	rhs	support	confidance	lift	
0	brownies, cottage cheese,			0.003466	0.102767	3.225330	
1	chicken, light cream,			0.004533	0.075556	4.843951	
2	escalope, mushroom cream sauce,			0.005733	0.072269	3.790833	
3	escalope, pasta,			0.005866	0.073950	4.700812	
4	fresh bread, tomato juice,			0.004266	0.099071	3.259356	

✓ 0s completed at 17:19

×