In [1]:

```
# This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python docker image: https://github.com/kaggle/docker-python
# For example, here's several helpful packages to load in

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the "../input/" directory.
# For example, running this (by clicking run or pressing Shift+Enter) will list all files under the input directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# Any results you write to the current directory are saved as output.
```

In [2]:

```
!pip install apyori
```

```
Collecting apyori
  Downloading https://files.pythonhosted.org/packages/5e/62/5ffde5c473ea4b033490617ec5caa80d59804875ad3c3c57c0976533a21a/apyori-1.1.2.tar.gz
Building wheels for collected packages: apyori
  Building wheel for apyori (setup.py) ... done
  Stored in directory: /home/jovyan/.cache/pip/wheels/5d/92/bb/474bbadbc8c0062b9eb168f69982a0443263f8ab1711a8cad0
Successfully built apyori
Installing collected packages: apyori
Successfully installed apyori-1.1.2
```

In [4]:

```
from apyori import apriori
```

In [5]:

```
def load_dataset():
    return [[1,3,4],[2,3,5],[1,2,3,5],[2,5]]
```

In [6]:

```
x = load_dataset()
```

In [7]:

```
x
```

Out[7]:

```
[[1, 3, 4], [2, 3, 5], [1, 2, 3, 5], [2, 5]]
```

In [8]:

```
association_rules = list(apriori(x))
```

In [9]:

```
len(association_rules)
```

Out[9]:

19

In [11]:

```
for rule in association_rules:
    name_pair = list(rule[0])
    print("Rule:", name_pair)
    print("support",rule[1])
```

```
Rule: [1]
support 0.5
Rule: [2]
support 0.75
Rule: [3]
support 0.75
Rule: [4]
support 0.25
Rule: [5]
support 0.75
Rule: [1, 2]
support 0.25
Rule: [1, 3]
support 0.5
Rule: [1, 4]
support 0.25
Rule: [1, 5]
support 0.25
Rule: [2, 3]
support 0.5
Rule: [2, 5]
support 0.75
Rule: [3, 4]
support 0.25
Rule: [3, 5]
support 0.5
Rule: [1, 2, 3]
support 0.25
Rule: [1, 2, 5]
support 0.25
Rule: [1, 3, 4]
support 0.25
Rule: [1, 3, 5]
support 0.25
Rule: [2, 3, 5]
support 0.5
Rule: [1, 2, 3, 5]
support 0.25
```

In [12]:

```
data = load_dataset()
data
```

Out[12]:

```
[[1, 3, 4], [2, 3, 5], [1, 2, 3, 5], [2, 5]]
```

In [13]:

```
def create_C1(dataset):
    C1=[]
    for transaction in dataset:
        for item in transaction:
            if not [item] in C1:
                C1.append([item])
    C1.sort()
    return list(map(frozenset, C1))
```

In [14]:

```
min_support = 2
```

In [15]:

```
def generate_L(D, CK, min_support):
    modified_dict={}
    for transaction in D:
        for grp in CK:
            if grp.issubset(transaction):
                if not grp in modified_dict:
                    modified_dict[grp] = 1
                else:
                    modified_dict[grp] += 1
    #print("CK with support:",modified_dict,"\n")
    r_list = []
    dict_with_freq = {}
    for key in modified_dict:
        supp = modified_dict[key]
        if supp>= min_support:
            r_list.insert(0,key)
            dict_with_freq[key] = supp

    return r_list,dict_with_freq
```

In [16]:

```
D = list(map(set,data))
```

In [17]:

```
def create_CK(Lk, k):

    retList = []
    lenLk = len(Lk)
    for i in range(lenLk):
        for j in range(i+1, lenLk):
            L1 = list(Lk[i])[:k-2]; L2 = list(Lk[j])[:k-2]
            L1.sort(); L2.sort()
            if L1==L2:
                retList.append(Lk[i] | Lk[j])
    return retList
```

In [18]:

```
C1 = create_C1(data)
C1
```

Out[18]:

```
[frozenset({1}),
 frozenset({2}),
 frozenset({3}),
 frozenset({4}),
 frozenset({5})]
```

In [19]:

```
L1, supp1 = generate_L(D,C1,min_support)
print(supp1)
```

```
{frozenset({1}): 2, frozenset({3}): 3, frozenset({2}): 3, frozenset({5}):
3}
```

In [20]:

```
C2 = create_CK(L1,2)
C2
```

Out[20]:

```
[frozenset({2, 5}),
 frozenset({3, 5}),
 frozenset({1, 5}),
 frozenset({2, 3}),
 frozenset({1, 2}),
 frozenset({1, 3})]
```

In [21]:

```
L2, supp2 = generate_L(D,C2,min_support)
print(supp2)
```

```
{frozenset({1, 3}): 2, frozenset({2, 5}): 3, frozenset({3, 5}): 2, frozens
et({2, 3}): 2}
```

In [22]:

```
C3 = create_CK(L2,3)
C3
```

Out[22]:

```
[frozenset({2, 3, 5})]
```

In [23]:

```
L3, supp3 = generate_L(D,C3,min_support)
print(supp3)
```

```
{frozenset({2, 3, 5}): 2}
```

In [24]:

```
C4 = create_CK(L3,4)
C4
```

Out[24]:

```
[]
```

In [25]:

```
C1 = create_C1(data)
CK = C1
i = 2
while CK !=[]:
    LK,supp = generate_L(D,CK,min_support)
    CK = create_CK(LK,i)
    i+=1
print(LK,"\n",supp)
```

```
[frozenset({2, 3, 5})]
 {frozenset({2, 3, 5}): 2}
```

In [ ]:

```

```